Dynamic Parameter Policies for LeadingOnes on Enhanced State Spaces

Gianluca Covini

Supervisor:
Prof. Stefano Gualandi,
University of Pavia
Co-supervisor:
Prof. Carola Doerr,
Sorbonne Université/CNRS

UNIVERSITÀ
DI PAVIA

February 20, 2025

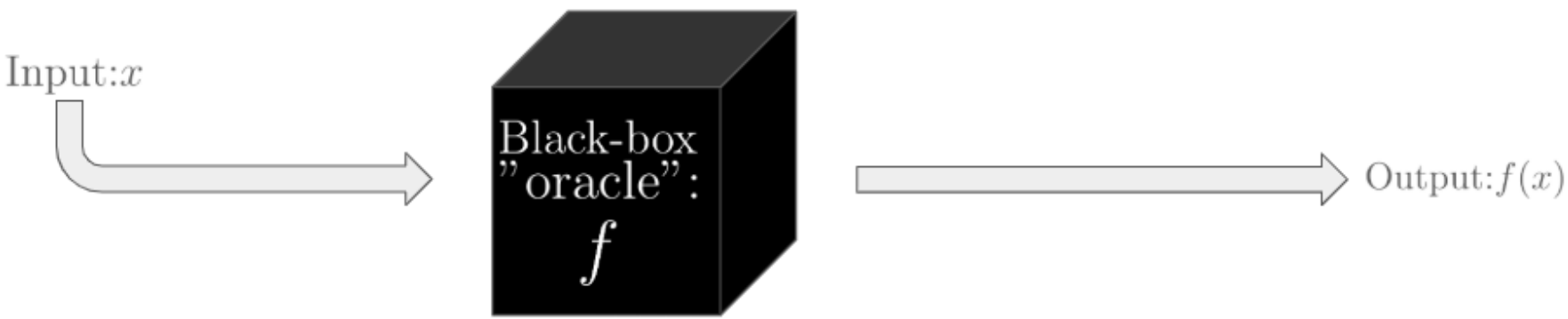
Outline

Introduction

Optimization problem

Given $f : S \to \mathbb{R}$,

$$\text{find } x^* \in \underset{x \in S}{\text{argmax}}\, f(x)$$



Genetic Algorithms

We consider a **pseudo-boolean objective** $f : \{0,1\}^n \to \mathbb{R}$.

Genetic Algorithms

We consider a **pseudo-boolean objective** $f : \{0,1\}^n \to \mathbb{R}$.

Genetic Algorithms

1. **Mutation**: candidate solutions $y_1, \ldots, y_N \in \{0,1\}^n$ are sampled from a family of distributions $(D(\cdot|x^{(1)}, \ldots, x^{(k)}))_{x^{(1)},\ldots,x^{(k)}}$;

Genetic Algorithms

We consider a **pseudo-boolean objective** $f : \{0,1\}^n \to \mathbb{R}$.

Genetic Algorithms

1. **Mutation**: candidate solutions $y_1, \ldots, y_N \in \{0,1\}^n$ are sampled from a family of distributions $(D(\cdot|x^{(1)}, \ldots, x^{(k)}))_{x^{(1)},\ldots,x^{(k)}}$;
2. recombination of candidate solutions is eventually conducted;

Genetic Algorithms

We consider a **pseudo-boolean objective** $f : \{0,1\}^n \to \mathbb{R}$.

Genetic Algorithms

1. **Mutation**: candidate solutions $y_1, \ldots, y_N \in \{0,1\}^n$ are sampled from a family of distributions $(D(\cdot|x^{(1)}, \ldots, x^{(k)}))_{x^{(1)},\ldots,x^{(k)}}$;
2. recombination of candidate solutions is eventually conducted;
3. **Selection**: the candidate solutions $y_1, \ldots, y_N$ are selected based on their fitness $f(y_1), \ldots, f(y_N)$;

Genetic Algorithms

We consider a **pseudo-boolean objective** $f : \{0,1\}^n \to \mathbb{R}$.

Genetic Algorithms

1. **Mutation**: candidate solutions $y_1, \ldots, y_N \in \{0,1\}^n$ are sampled from a family of distributions $(D(\cdot|x^{(1)}, \ldots, x^{(k)}))_{x^{(1)},\ldots,x^{(k)}}$;
2. recombination of candidate solutions is eventually conducted;
3. **Selection**: the candidate solutions $y_1, \ldots, y_N$ are selected based on their fitness $f(y_1), \ldots, f(y_N)$;
4. back to step 1 until termination criteria (reach of the optimum).

Randomized Local Search (RLS)

RLS Algorithm

1: **Input:** Fitness function f, bit-string length n
2: **Initialize:** Generate a random solution $x \in \{0,1\}^n$
3: **while** termination criteria are not met **do**
4: Choose the radius k
5: Create $y \leftarrow x$ by flipping k randomly chosen bits in x
6: **if** $f(y) \geq f(x)$ **then**
7: $x \leftarrow y$
8: **end if**
9: **end while**
10: **Output:** Best solution x found

Dynamic Algorithm Configuration

Definition

We call **policy** a function

$$\pi : S \to [1..n], s \mapsto k$$

where s describes the **state** of the algorithm at a certain iteration.

Dynamic Algorithm Configuration

Definition

We call **policy** a function

$$\pi : S \to [1..n], s \mapsto k$$

where s describes the **state** of the algorithm at a certain iteration.

Problem

$$\text{Find a } \pi^* \in \arg\min_{\pi} \mathbb{E}\left[c(\pi; f)\right]$$

where c is a (random) cost metric assessing the cost of using $\pi \in \Pi$ on the problem f.

Dynamic Algorithm Configuration

Definition

We call **policy** a function

$$\pi : S \to [1..n], s \mapsto k$$

where s describes the **state** of the algorithm at a certain iteration.

Problem

$$\text{Find a } \pi^* \in \arg\min_{\pi} \mathbb{E}\left[c(\pi; f)\right]$$

where c is a (random) cost metric assessing the cost of using $\pi \in \Pi$ on the problem f.

We take as cost the **runtime**, defined by the number of evaluations of the objective before evaluating the optimum, which we assume is reachable in our case.

LeadingOnes and OneMax

Definition

OneMax (OM for brevity) function is a function that returns the number of one-bits in its argument, that is,

$$\textsc{OneMax}(x) = \mathrm{OM}(x) = \sum_{i=1}^{n} x_i.$$

LeadingOnes and OneMax

Definition

OneMax (OM for brevity) function is a function that returns the number of one-bits in its argument, that is,

$$\textsc{OneMax}(x) = \mathrm{OM}(x) = \sum_{i=1}^{n} x_i.$$

Definition

LeadingOnes (LO for brevity) returns the size of the longest prefix consisting only of one-bits in its arguments. More formally, for any bit string $x \in \{0,1\}^n$ we have

$$\textsc{LeadingOnes}(x) = \mathrm{LO}(x) = \sum_{i=1}^{n} \prod_{j=1}^{i} x_i.$$

Unary Unbiased Complexity

Definition

A **unary unbiased distribution** $(D(\cdot|x^{(1)}))_{x^{(1)}}$ is a family of probability distributions over $\{0,1\}^n$ such that for all inputs $x^{(1)} \in \{0,1\}^n$ the following two conditions hold.

(i) $\forall y, z \in \{0,1\}^n : D(y|x^{(1)}) = D(y \oplus z|x^{(1)} \oplus z)$,

(ii) $\forall x \in \{0,1\}^n \; \forall \sigma \in S_n : D(y|x^{(1)}) = D(\sigma(y)|\sigma(x^{(1)}))$

A genetic algorithm that creates an offspring by sampling from a unary unbiased distribution is called a **unary unbiased algorithm**.

Unary Unbiased Complexity

Definition

A **unary unbiased distribution** $(D(\cdot|x^{(1)}))_{x^{(1)}}$ is a family of probability distributions over $\{0,1\}^n$ such that for all inputs $x^{(1)} \in \{0,1\}^n$ the following two conditions hold.

(i) $\forall y, z \in \{0,1\}^n : D(y|x^{(1)}) = D(y \oplus z|x^{(1)} \oplus z)$,

(ii) $\forall x \in \{0,1\}^n \; \forall \sigma \in S_n : D(y|x^{(1)}) = D(\sigma(y)|\sigma(x^{(1)}))$

A genetic algorithm that creates an offspring by sampling from a unary unbiased distribution is called a **unary unbiased algorithm**.

Definition

Defined $\mathcal{A}$ the set of unary unbiased algorithms, we define the **unary unbiased black-box complexity** of the problem of optimizing $f : S \to \mathbb{R}$ as

$$\mathbb{E}[T(\mathcal{A}, f)] := \inf_{A \in \mathcal{A}} \mathbb{E}[T(A, f)]$$

Complexity Results

Theorem

The unary unbiased black-box complexity of OneMax *is* $\Theta(n \log n)$.

Complexity Results

Theorem

The unary unbiased black-box complexity of OneMax *is* $\Theta(n \log n)$.

Theorem

The unary unbiased black-box complexity of LeadingOnes *is* $\Theta(n^2)$.

Dynamic Radius Policy for LeadingOnes

States are defined as values of LeadingOnes fitness.
The state-of-the-art dynamic parameter policy for RLS radius on LeadingOnes is defined as a function $\pi : \mathcal{S}^{(1)} := [0..n] \to [0..n]$.

Idea

The probability of obtaining a strictly better solution by flipping k random bits in a search point of fitness i is

$$q(k; l, n) = \frac{k(n-l-1)\cdot\dots\cdot(n-l-k+1)}{n(n-1)\cdot\dots\cdot(n-k+1)}$$

Dynamic Radius Policy for LeadingOnes

States are defined as values of LeadingOnes fitness.
The state-of-the-art dynamic parameter policy for RLS radius on LeadingOnes is defined as a function $\pi : \mathcal{S}^{(1)} := [0..n] \to [0..n]$.

Idea

The probability of obtaining a strictly better solution by flipping k random bits in a search point of fitness i is

$$q(k; l, n) = \frac{k(n-l-1)\cdot\dots\cdot(n-l-k+1)}{n(n-1)\cdot\dots\cdot(n-k+1)}$$

Remark

$q(k; l, n) \leq q(k+1; l, n)$ if and only if $l \leq (n-k)/(k+1)$

Dynamic Radius Policy for LeadingOnes

Proposition

The optimal policy for the RLS radius is

$$k(l;n) := \left\lfloor \frac{n}{l+1} \right\rfloor$$

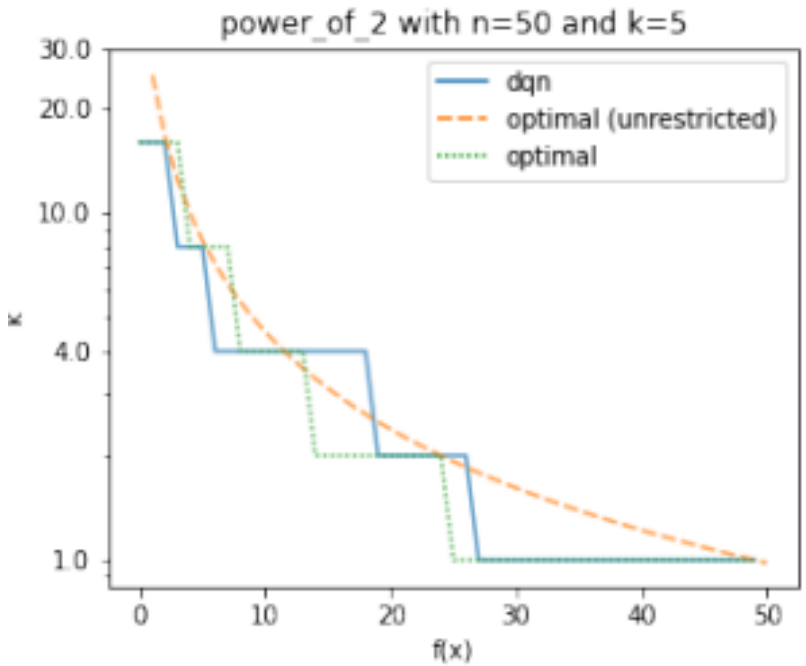
bits when the current fitness is i. This results in an expected runtime of $0.39n^2$, which corresponds to a 22% improvement of the choice of fixed parameters.

DAC on LeadingOnes

A recent approach consists in the use of a DDQN agent to learn the optimal radius policy for RLS on LeadingOnes.

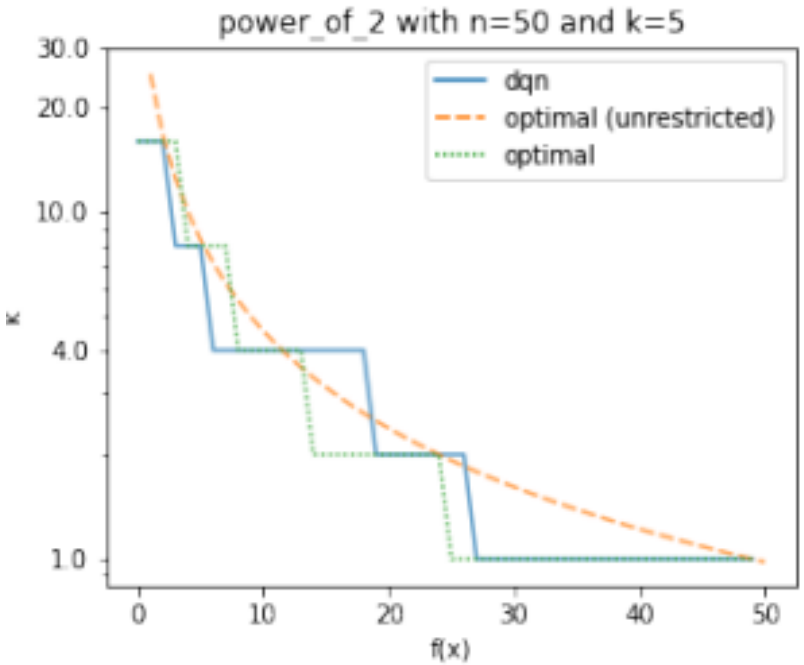
DAC on LeadingOnes

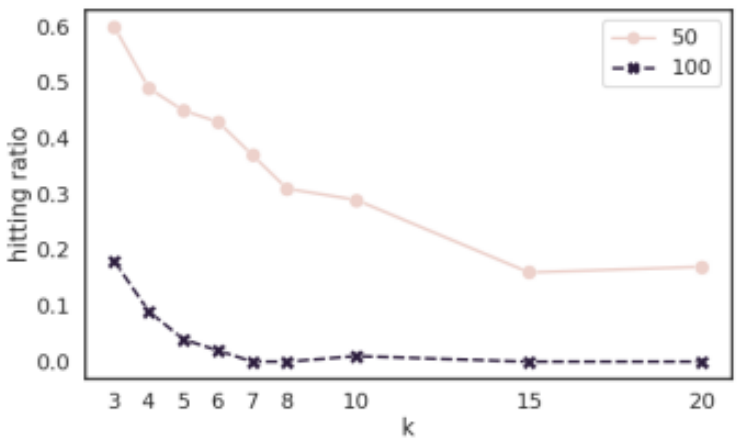
A recent approach consists in the use of a DDQN agent to learn the optimal radius policy for RLS on LeadingOnes.



DAC on LeadingOnes

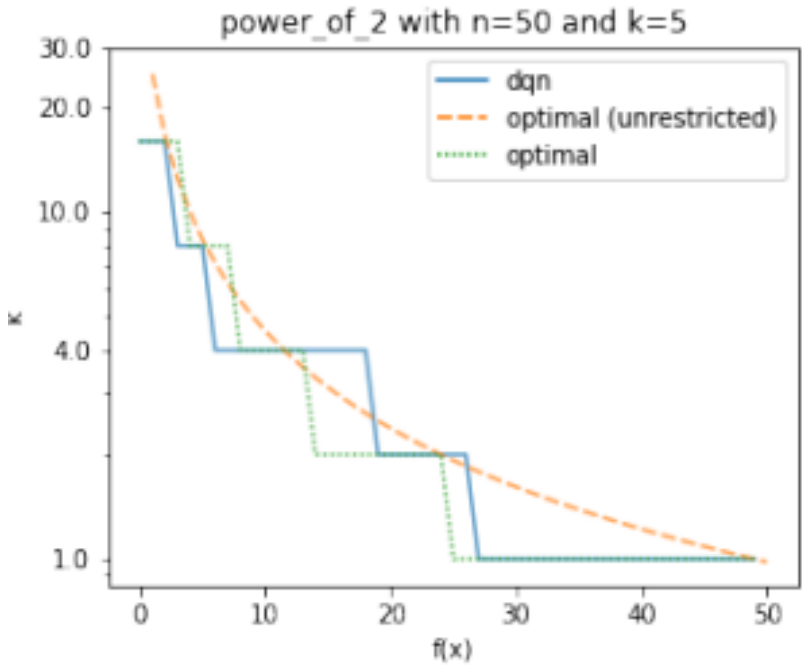
A recent approach consists in the use of a DDQN agent to learn the optimal radius policy for RLS on LeadingOnes.

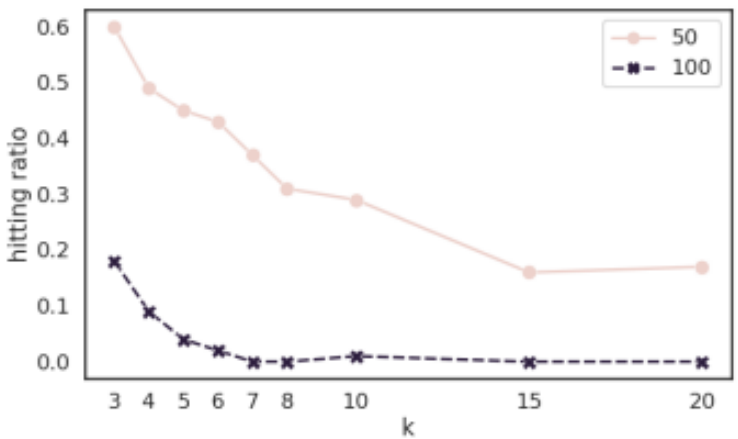




DAC on LeadingOnes

A recent approach consists in the use of a DDQN agent to learn the optimal radius policy for RLS on LeadingOnes.





Problem

Generalization difficulties for growing n.

Setting

Goal

Extend the radius policies for RLS including more information.

Setting

Goal

Extend the radius policies for RLS including more information.

State Spaces

- $\mathcal{S}^{(1)} = [0..n]$, values of LeadingOnes fitness;
- $\mathcal{S}^{(2)} := \{(l, m) : l \in [0..n], m \in [l..n]\}$, tuples of (LeadingOnes, OneMax) fitness;
- $\mathcal{S}^{(n)} := \{0, 1\}^n$, all possible bit-strings.

Setting

Goal

Extend the radius policies for RLS including more information.

State Spaces

- $\mathcal{S}^{(1)} = [0..n]$, values of LeadingOnes fitness;
- $\mathcal{S}^{(2)} := \{(l, m) : l \in [0..n], m \in [l..n]\}$, tuples of (LeadingOnes, OneMax) fitness;
- $\mathcal{S}^{(n)} := \{0,1\}^n$, all possible bit-strings.

Lexicographic selection

Candidate y is accepted from x according to the relation $(\mathrm{LO}(y), \mathrm{OM}(y)) > (\mathrm{LO}(x), \mathrm{OM}(x))$ if and only if $(\mathrm{LO}(y) > \mathrm{LO}(x))$ or $(\mathrm{LO}(y) = \mathrm{LO}(x)$ and $\mathrm{OM}(y) > \mathrm{OM}(x))$.

Policy on $\mathcal{S}^{(2)}$

Expected Runtime from a Starting State

$$\mathbb{E}[T_{opt}^{(k)}(l,m)] = 1 + \mathbb{P}((l,m)|(l,m)) \cdot \mathbb{E}[T_{opt}^{(k)}(l,m)] +$$
$$+ \sum_{\substack{\lambda=l \\ (\lambda,\mu)\neq(l,m)}}^{n-1} \sum_{\mu=\lambda}^{n-1} \mathbb{P}((\lambda,\mu)|(l,m)) \cdot \mathbb{E}[T_{opt}(\lambda,\mu)], \quad \forall (l,m) \in \mathcal{S}^{(2)}$$

$$k_{opt}(n-1,n-1) = 1$$
$$\mathbb{E}[T_{opt}(n-1,n-1)] = n+1$$

Policy on $\mathcal{S}^{(2)}$

Expected Runtime from a Starting State

$$\mathbb{E}[T_{opt}^{(k)}(l,m)] = 1 + \mathbb{P}((l,m)|(l,m)) \cdot \mathbb{E}[T_{opt}^{(k)}(l,m)] +$$
$$+ \sum_{\substack{\lambda=l \\ (\lambda,\mu)\neq(l,m)}}^{n-1} \sum_{\mu=\lambda}^{n-1} \mathbb{P}((\lambda,\mu)|(l,m)) \cdot \mathbb{E}[T_{opt}(\lambda,\mu)], \quad \forall (l,m) \in \mathcal{S}^{(2)}$$

$$k_{opt}(n-1,n-1) = 1$$
$$\mathbb{E}[T_{opt}(n-1,n-1)] = n+1$$

Optimal Policy

$$k_{opt}(l,m) = \operatorname*{argmin}_{k\in[n-l]} \mathbb{E}[T_{opt}^{(k)}(l,m)], \quad \forall (l,m) \in \mathcal{S}^{(2)}$$

Strict Standard Setting

Approximation: Strict Standard Selection

Candidate y is accepted from x according to the relation $\mathrm{LO}(y) > \mathrm{LO}(x)$.

Expected Runtime from a Starting State

$$\begin{aligned}\mathbb{E}[T^{(k)}_{opt}(l,m)] =& 1 + \mathbb{P}((l,m)|(l,m)) \cdot \mathbb{E}[T^{(k)}_{opt}(l,m)] + \\ &+ \sum_{\substack{\lambda=l \\ (\lambda,\mu)\neq(l,m)}}^{n-1} \sum_{\mu=\lambda}^{n-1} \mathbb{P}((\lambda,\mu)|(l,m)) \cdot \mathbb{E}[T_{opt}(\lambda,\mu)]\end{aligned}$$

Strict Standard Setting

Approximation: Strict Standard Selection

Candidate y is accepted from x according to the relation $\mathrm{LO}(y) > \mathrm{LO}(x)$.

Expected Runtime from a Starting State

$$\mathbb{E}[T_{opt}^{(k)}(l,m)] = 1 + \mathbb{P}((l,m)|(l,m)) \cdot \mathbb{E}[T_{opt}^{(k)}(l,m)] + \\ + \sum_{\substack{\lambda=l \\ (\lambda,\mu)\neq(l,m)}}^{n-1} \sum_{\mu=\lambda}^{n-1} \mathbb{P}((\lambda,\mu)|(l,m)) \cdot \mathbb{E}[T_{opt}(\lambda,\mu)]$$

Expected Runtime of the Algorithm

$$\mathbb{E}[T_{opt}] = \sum_{l=0}^{n-1} \sum_{m=l}^{n-1} \mathbb{P}(\mathrm{LO}(x^{(0)}) = l, \mathrm{OM}(x^{(0)}) = m) \cdot \mathbb{E}[T_{opt}(l,m)]$$

Lexicographic Low-dimensional

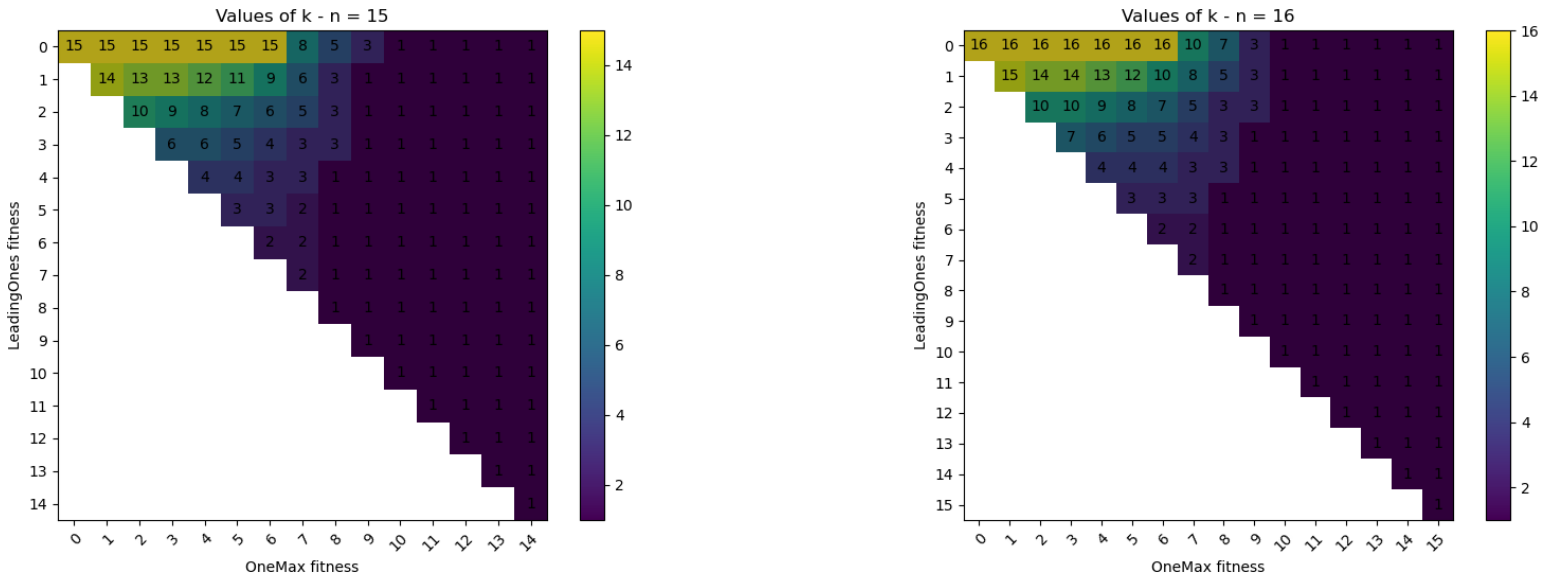


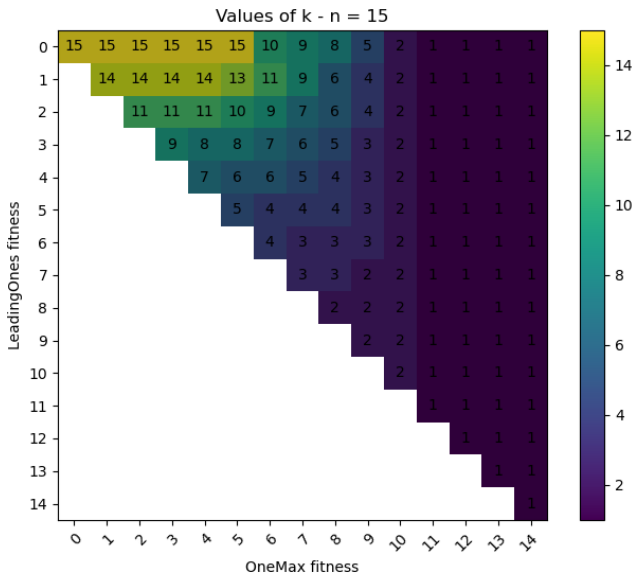
Figure: Heatmaps of optimal policies for lexicographic selection

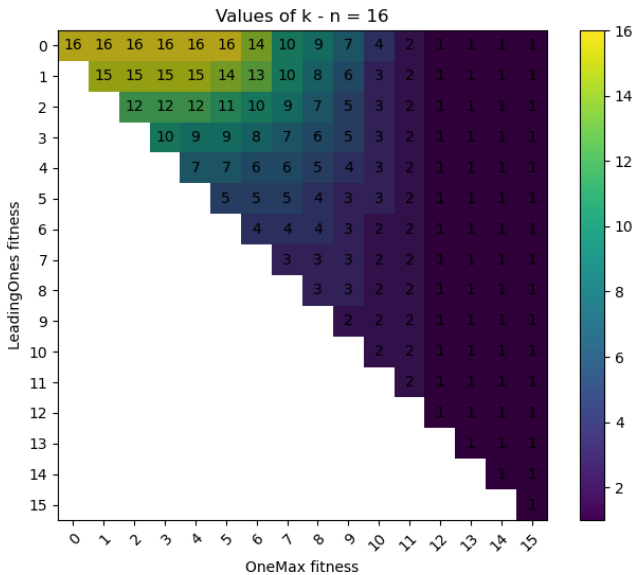
Lexicographic Low-dimensional

n	$\mathcal{S}^{(1)}$ std.	$\mathcal{S}^{(1)}$ lex.	$\mathcal{S}^{(2)}$ lex.
7	19.11	14.20	11.55
8	24.96	18.75	14.37
9	31.59	22.03	17.25
10	39.00	27.23	20.29
11	47.19	30.34	23.39
12	56.16	37.16	26.63
13	65.91	40.31	29.91
14	76.44	46.76	33.28
15	87.75	50.94	36.75
16	99.84	61.91	40.24

Table: Expected time for lexicographic selection

Strict Standard Low-dimensional



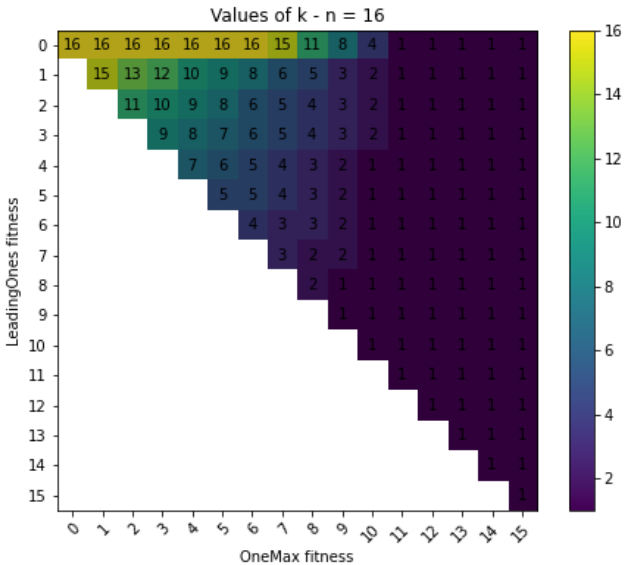


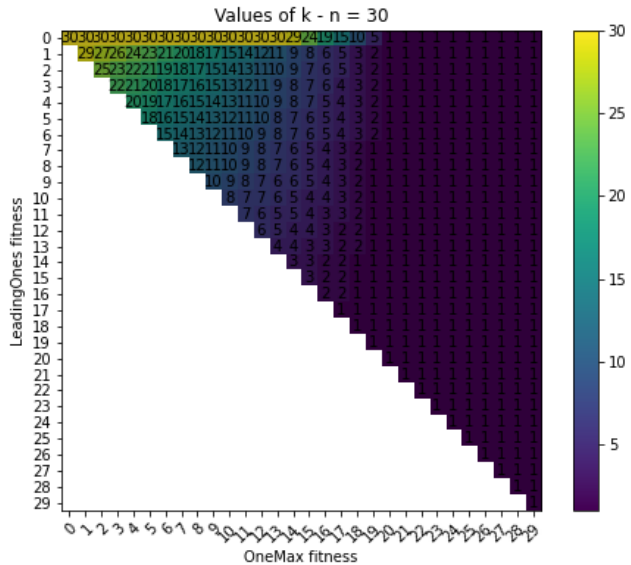
Strict Standard Low-dimensional

n	$\mathcal{S}^{(1)}$ std.	$\mathcal{S}^{(2)}$ s. std.	Improvement
7	19.110	14.783	22.61%
8	24.960	19.629	21.41%
9	31.590	25.530	19.20%
10	39.000	31.737	18.61%
11	47.190	38.575	18.24%
12	56.160	46.423	17.31%
13	65.910	55.265	16.15%
14	76.440	64.194	16.00%
15	87.750	74.006	15.66%
16	99.840	84.919	14.96%

Table: Expected time for strict standard selection

Heuristic Policy





Lexicographic High-dimensional

n	$\mathcal{S}^{(1)}$	Heuristic on $\mathcal{S}^{(2)}$
20	83.052 ± 35.401	56.817 ± 24.935
30	156.440 ± 57.204	97.100 ± 38.187
50	365.279 ± 113.261	189.796 ± 70.485
70	686.021 ± 201.404	322.379 ± 107.782
99	1484.031 ± 357.983	616.595 ± 161.838

Table: Simulated results for lexicographic selection

Standard High-dimensional

n	$\mathcal{S}^{(1)}$	Heuristic on $\mathcal{S}^{(2)}$
20	159.806 ± 63.532	151.438 ± 61.158
30	344.132 ± 114.888	347.866 ± 102.573
50	970.620 ± 254.967	960.984 ± 234.036
70	1886.518 ± 405.619	1866.500 ± 416.876

Table: Simulated results for standard selection

Future Directions

Future Directions

- Study more in depth the standard setting, trying to validate the results in high dimension.

Future Directions

Future Directions

- Study more in depth the standard setting, trying to validate the results in high dimension.
- Use the tested settings and policies to train a RL agent in the proposed settings, using our policies as new ground truths.

Future Directions

Future Directions

- Study more in depth the standard setting, trying to validate the results in high dimension.
- Use the tested settings and policies to train a RL agent in the proposed settings, using our policies as new ground truths.
- Our research contributes to the growing field of *AutoML*, where optimization heuristics for parameter control are explored as a means to automate the fine-tuning of machine learning algorithms.

Thank you for your attention!

Standard Setting

Standard Selection

Candidate y is accepted from x according to the relation $\mathrm{LO}(y) \geq \mathrm{LO}(x)$.

We obtain a linear system in matrix form $Ax = b$ as follows.

$$x = \begin{bmatrix} \mathbb{E}[T_{opt}^{(k_l)}(l,l)] \\ \mathbb{E}[T_{opt}^{(k_{l+1})}(l,l+1)] \\ \vdots \\ \mathbb{E}[T_{opt}^{(k_{n-1})}(l,n-1)] \end{bmatrix} \quad b = \begin{bmatrix} 1 + \sum_{\lambda=l+1}^{n-1} \sum_{\mu=\lambda}^{n-1} \mathbb{P}((\lambda,\mu)|(l,l)) \mathbb{E}[T_{opt}(\lambda,\mu)] \\ 1 + \sum_{\lambda=l+1}^{n-1} \sum_{\mu=\lambda}^{n-1} \mathbb{P}((\lambda,\mu)|(l,l+1)) \mathbb{E}[T_{opt}(\lambda,\mu)] \\ \vdots \\ 1 + \sum_{\lambda=l+1}^{n-1} \sum_{\mu=\lambda}^{n-1} \mathbb{P}((\lambda,\mu)|(l,n-1)) \mathbb{E}[T_{opt}(\lambda,\mu)] \end{bmatrix}$$

$$A = \begin{bmatrix} (1 - \mathbb{P}^{(k_l)}((l,l) \mid (l,l))) & \mathbb{P}^{(k_{l+1})}((l,l+1) \mid (l,l)) & \cdots & \mathbb{P}^{(k_{n-1})}((l,n-1) \mid (l,l)) \\ \mathbb{P}^{(k_l)}((l,l) \mid (l,l+1)) & (1 - \mathbb{P}^{(k_{l+1})}((l,l+1) \mid (l,l+1))) & \cdots & \mathbb{P}^{(k_{n-1})}((l,n-1) \mid (l,l+1)) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbb{P}^{(k_l)}((l,l) \mid (l,n-1)) & \mathbb{P}^{(k_{l+1})}((l,l+1) \mid (l,n-1)) & \cdots & (1 - \mathbb{P}^{(k_{n-1})}((l,n-1) \mid (l,n-1))) \end{bmatrix}$$

Standard Setting

We obtain linear system in matrix form $Ax = b$ as follows.

$$x = \begin{bmatrix} \mathbb{E}[T_{opt}^{(k_l)}(l,l)] \\ \mathbb{E}[T_{opt}^{(k_{l+1})}(l,l+1)] \\ \vdots \\ \mathbb{E}[T_{opt}^{(k_{n-1})}(l,n-1)] \end{bmatrix} \quad b = \begin{bmatrix} 1 + \sum_{\lambda=l+1}^{n-1} \sum_{\mu=\lambda}^{n-1} \mathbb{P}((\lambda,\mu)|(l,l)) \mathbb{E}[T_{opt}(\lambda,\mu)] \\ 1 + \sum_{\lambda=l+1}^{n-1} \sum_{\mu=\lambda}^{n-1} \mathbb{P}((\lambda,\mu)|(l,l+1)) \mathbb{E}[T_{opt}(\lambda,\mu)] \\ \vdots \\ 1 + \sum_{\lambda=l+1}^{n-1} \sum_{\mu=\lambda}^{n-1} \mathbb{P}((\lambda,\mu)|(l,n-1)) \mathbb{E}[T_{opt}(\lambda,\mu)] \end{bmatrix}$$

$$A = \begin{bmatrix} (1 - \mathbb{P}^{(k_l)}((l,l) \mid (l,l))) & \mathbb{P}^{(k_{l+1})}((l,l+1) \mid (l,l)) & \cdots & \mathbb{P}^{(k_{n-1})}((l,n-1) \mid (l,l)) \\ \mathbb{P}^{(k_l)}((l,l) \mid (l,l+1)) & (1 - \mathbb{P}^{(k_{l+1})}((l,l+1) \mid (l,l+1))) & \cdots & \mathbb{P}^{(k_{n-1})}((l,n-1) \mid (l,l+1)) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbb{P}^{(k_l)}((l,l) \mid (l,n-1)) & \mathbb{P}^{(k_{l+1})}((l,l+1) \mid (l,n-1)) & \cdots & (1 - \mathbb{P}^{(k_{n-1})}((l,n-1) \mid (l,n-1))) \end{bmatrix}$$

Approximation

- We take $k_l = k_{l+1} = \cdots = k_{n-1} = k$ to compute $\mathbb{E}[T_{opt}^{(k)}(l,m)]$;
- We then take $k_{opt}(l,m) = \text{argmin}_{k \in [n-l]} \mathbb{E}[T_{opt}^{(k)}(l,m)]$.

Standard Low-dimensional

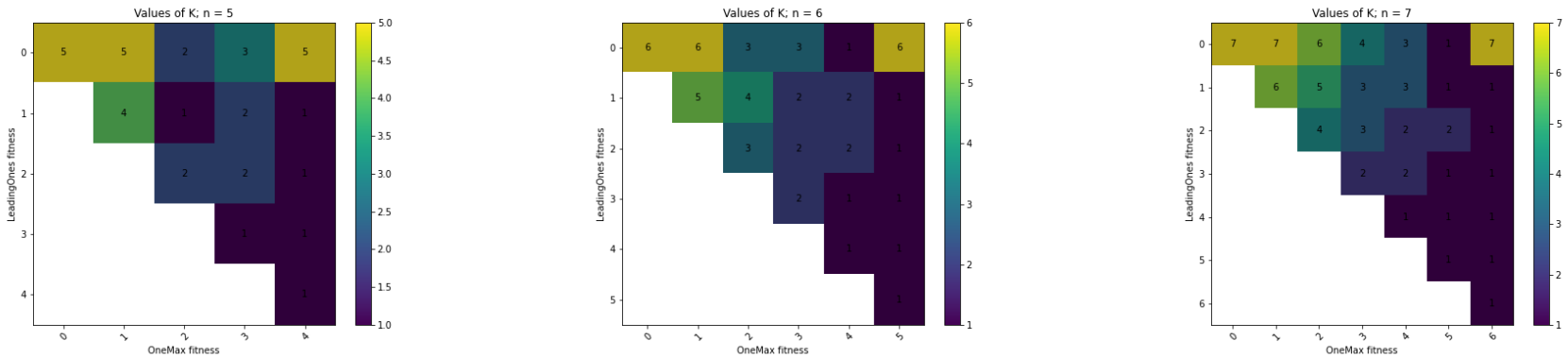


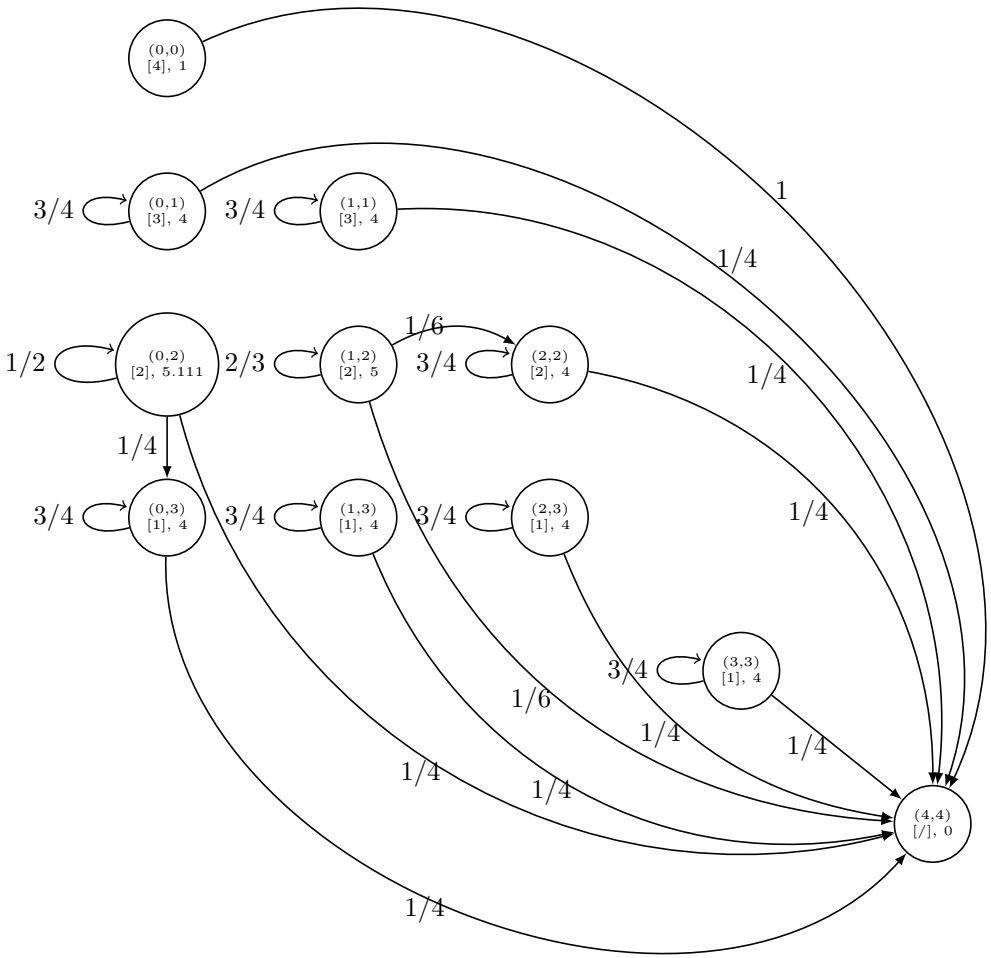
Figure: Heatmaps of approximated optimal policies for standard selection

Standard Low-dimensional

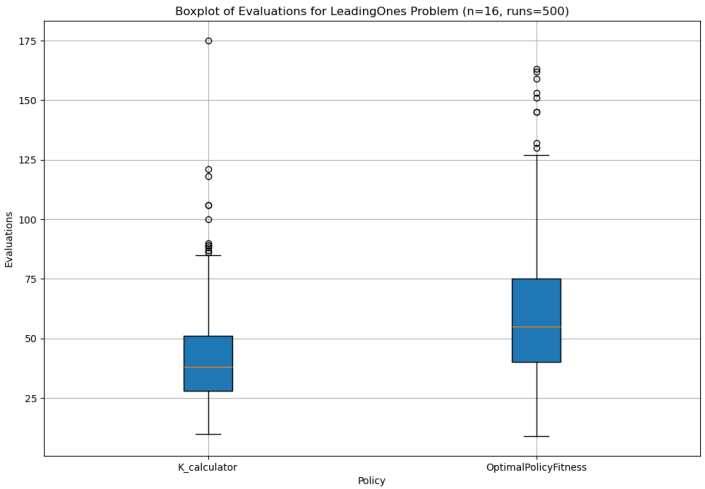
Dim (n)	$\mathcal{S}^{(1)}$ std	$\mathcal{S}^{(2)}$ std
2	1.5	1.125
3	3.5	2.375
4	6.0	4.448
5	9.667	7.463
6	13.667	11.488
7	18.875	16.435

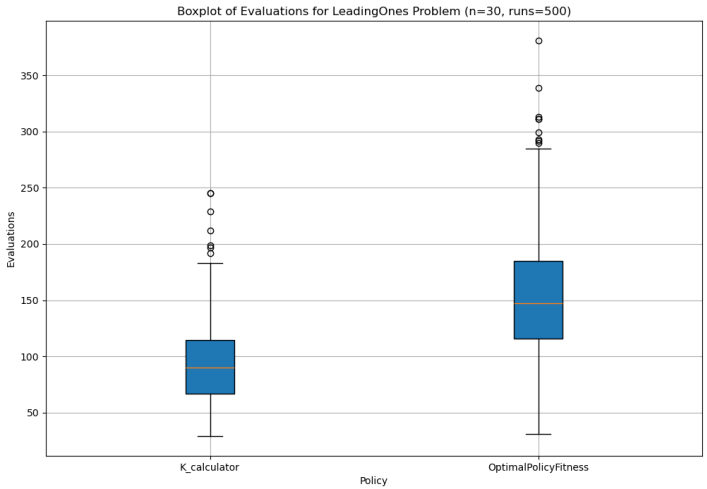
Table: Expected time for standard selection

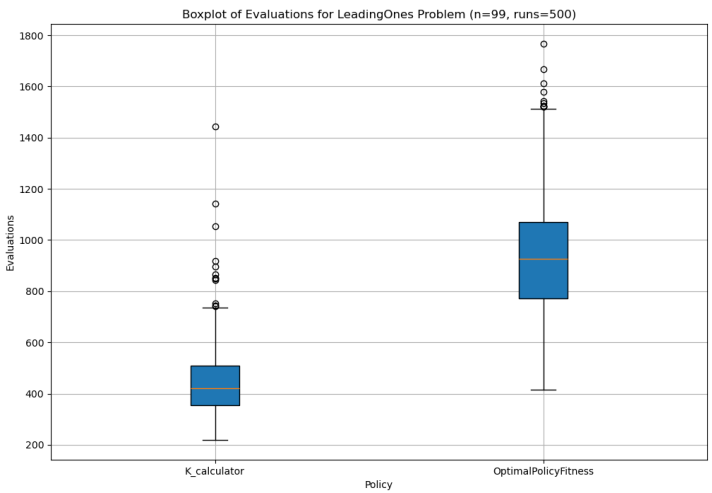
Graph $n = 4$


(0,0)
[4], 1
(0,1)
[3], 4
(1,1)
[3], 4
(0,2)
[2], 5.111
(1,2)
[2], 5
(2,2)
[2], 4
(0,3)
[1], 4
(1,3)
[1], 4
(2,3)
[1], 4
(3,3)
[1], 4
(4,4)
[/], 0
3/4
3/4
1/2
2/3
3/4
3/4
3/4
3/4
1
1/4
1/6
1/4
1/4
1/4
1/4
1/6
1/4
1/4
1/4
1/4

Boxplot lexicographic







Limited Portfolio

- `powers_of_two`: $\{2^i \mid 2^i \leq n\}$;
- `initial_segment` with 3 elements: $[1..3]$;
- `evenly_spread` with 3 elements: $\{i \cdot \lfloor n/3 \rfloor + 1 \mid i \in [0..2]\}$.

n	$\mathcal{S}^{(1)}$	$\mathcal{S}^{(2)}$	`powers_of_two`	`initial_segment`	`evenly_spread`
2	1.5	1.25	1.25	1.25	1.75
3	3.125	2.375	2.75	2.375	2.375
4	5.5	4.375	4.625	4.687	4.687
5	7.857	6.491	6.87	7.087	7.087
6	11.511	8.946	9.537	9.684	9.261
7	14.197	11.549	12.205	12.471	12.037
8	18.748	14.368	14.574	15.306	14.906
9	22.031	17.248	17.589	18.318	17.693
10	27.234	20.289	20.683	21.413	20.81
11	30.337	23.393	23.908	24.6	24.028
12	37.156	26.63	27.203	27.903	27.1
13	40.306	29.914	30.58	31.247	30.482
14	46.758	33.282	34.024	34.694	33.938
15	50.941	36.747	37.53	38.214	37.329
16	58.558	40.237	40.469	41.772	40.9

Limited portfolio

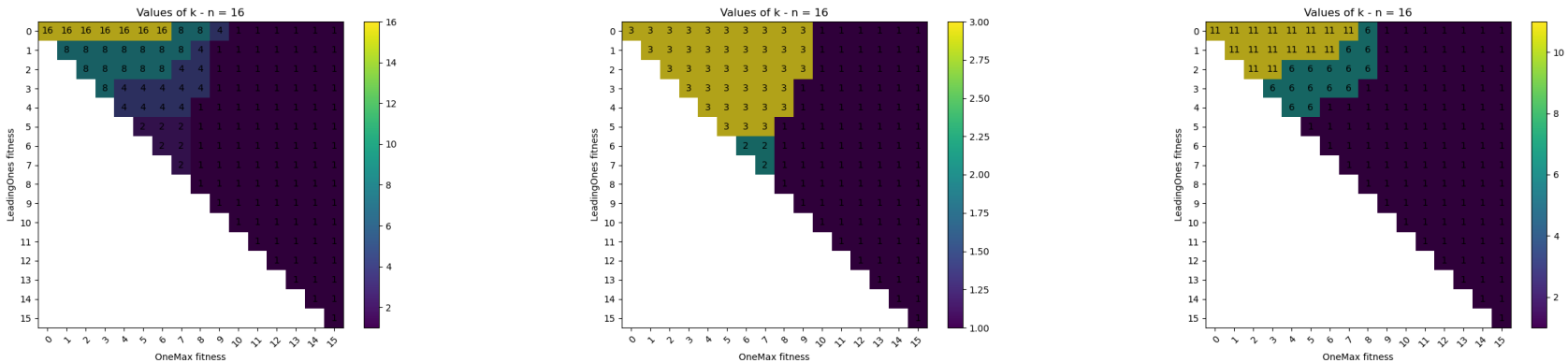


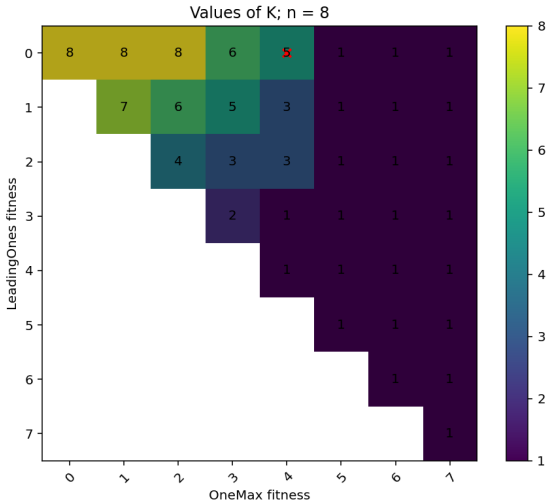
Figure: Heatmaps of optimal policy for lexicographic selection and limited portfolio

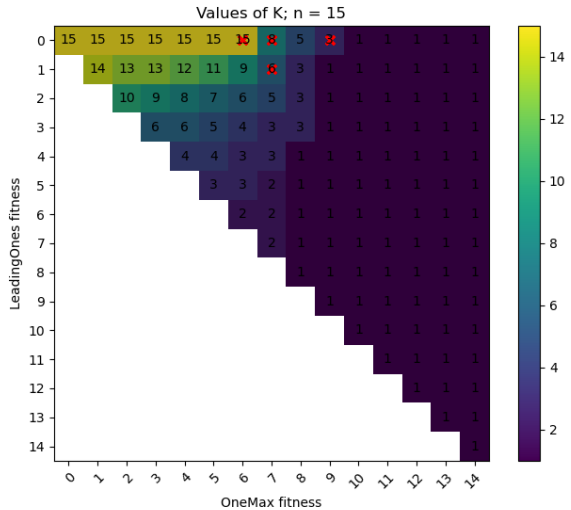
Results for $\mathcal{S}^{(n)}$

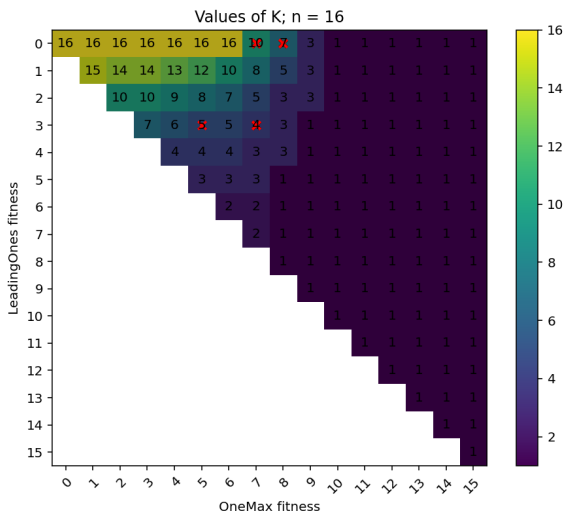
n	$\mathcal{S}^{(2)}$	$\mathcal{S}^{(n)}$
7	11.549	11.549
8	14.368	14.365
9	17.248	17.248
10	20.289	20.287
11	23.393	23.393
12	26.63	26.629
13	29.914	29.914
14	33.282	33.279
15	36.747	36.743
16	40.237	40.236

Table: Expected time for lexicographic selection

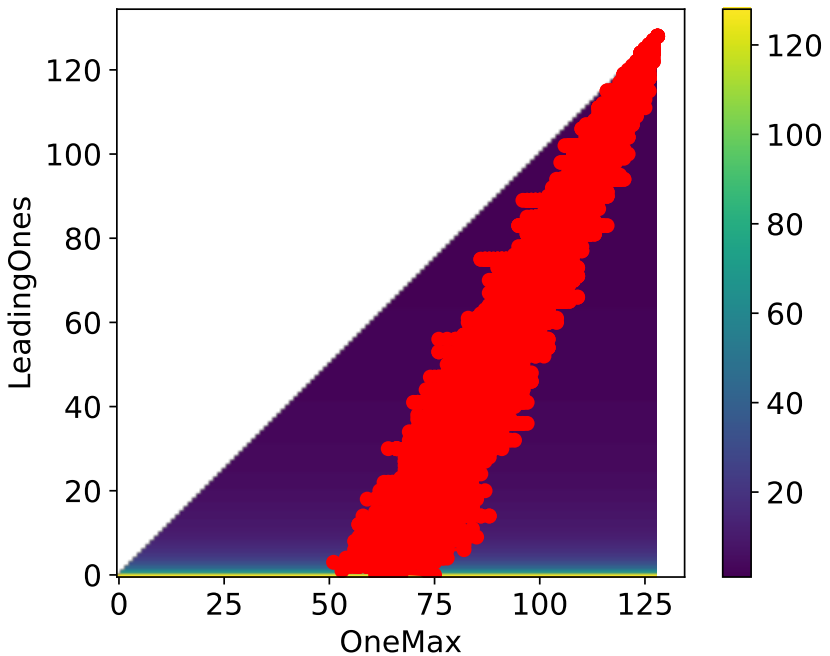
Results for $\mathcal{S}^{(n)}$

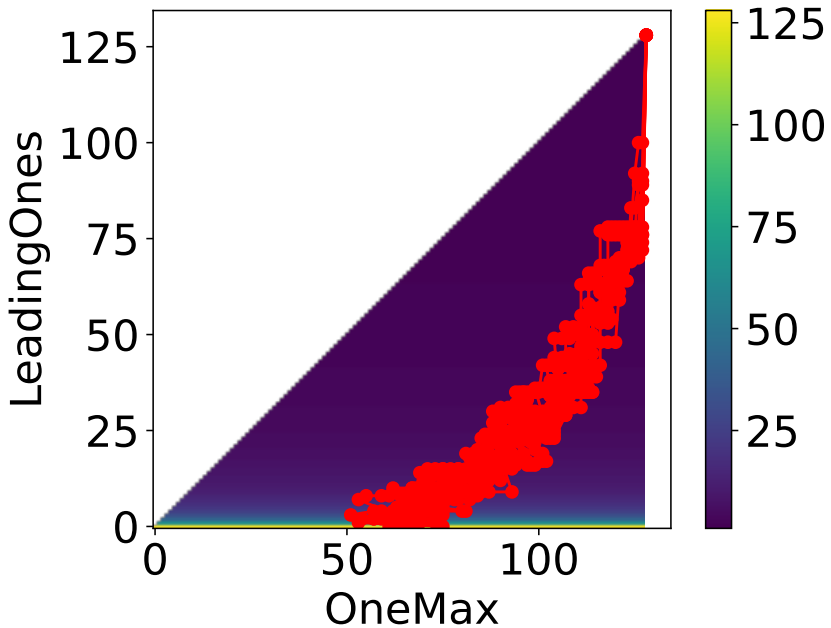






Runs


LeadingOnes
OneMax
0
20
40
60
80
100
120
0
25
50
75
100
125


LeadingOnes
OneMax
0
25
50
75
100
125
0
50
100
25
75

Dynamic Algorithm Configuration

Definition (DAC Problem)

Given $\langle A, \Theta, D, \Pi, c \rangle$:

- A step-wise reconfigurable target algorithm A with configuration space Θ.
- A distribution D over target problem instances with domain I.
- A space of dynamic configuration policies $\pi \in \Pi$ with $\pi : S \times I \to \Theta$ that choose a configuration $\theta \in \Theta$ for each instance $i \in I$ and state $s \in S$ of A.
- A cost metric $c : \Pi \times I \to \mathbb{R}$ assessing the cost of using $\pi \in \Pi$ on $i \in I$.

Find a $\pi^* \in \arg\min_{\pi \in \Pi} \mathbb{E}_{i \sim D} [c(\pi, i)]$.

Dynamic Algorithm Configuration

Definition (DAC Problem)

Given $\langle A, \Theta, D, \Pi, c \rangle$:

- A step-wise reconfigurable target algorithm A with configuration space Θ.
- A distribution D over target problem instances with domain I.
- A space of dynamic configuration policies $\pi \in \Pi$ with $\pi : S \times I \to \Theta$ that choose a configuration $\theta \in \Theta$ for each instance $i \in I$ and state $s \in S$ of A.
- A cost metric $c : \Pi \times I \to \mathbb{R}$ assessing the cost of using $\pi \in \Pi$ on $i \in I$.

Find a $\pi^* \in \arg\min_{\pi \in \Pi} \mathbb{E}_{i \sim D} [c(\pi, i)]$.

We take as cost the **runtime**, defined by the number of evaluations of the objective before evaluating the optimum, which we assume is reachable in our case.

DAC on LeadingOnes

Definition

$\mathcal{Q}$-**learning** consists in learning the $\mathcal{Q}$-function $\mathcal{Q} : S \times \mathcal{A} \longrightarrow \mathcal{R}$, which maps a state-action pair to the cumulative function reward that is received after playing an action a in state s.

DAC on LeadingOnes

Definition

$\mathcal{Q}$-**learning** consists in learning the $\mathcal{Q}$-function $\mathcal{Q} : S \times \mathcal{A} \longrightarrow \mathcal{R}$, which maps a state-action pair to the cumulative function reward that is received after playing an action a in state s.

Definition

To model the $\mathcal{Q}$-function it is possible to use two copies of a neural network, one used to select maximizing actions and the other to predict the value, in order to improve stability. The result is the so-called **double deep $\mathcal{Q}$ network**.

DAC on LeadingOnes

Remark

Given a state s_t and an action a_t, the $\mathcal{Q}$-value $\mathcal{Q}(s_t, a_t)$ can be updated using temporal differences (TD) as

$$\mathcal{Q}(s_t, a_t) \leftarrow \mathcal{Q}(s_t, a_t) + \alpha((r_t + \gamma \max \mathcal{Q}(s_{t+1}, \cdot)) - \mathcal{Q}(s_t, a_t))$$

where α is the learning rate and γ is the discount factor.
The reward is $r_t = \mathrm{LO}(x_t) - \mathrm{LO}(x_{t-1}) - 1$

DAC on LeadingOnes

Remark

Given a state s_t and an action a_t, the $\mathcal{Q}$-value $\mathcal{Q}(s_t, a_t)$ can be updated using temporal differences (TD) as

$$\mathcal{Q}(s_t, a_t) \leftarrow \mathcal{Q}(s_t, a_t) + \alpha((r_t + \gamma \max \mathcal{Q}(s_{t+1}, \cdot)) - \mathcal{Q}(s_t, a_t))$$

where α is the learning rate and γ is the discount factor.
The reward is $r_t = \mathrm{LO}(x_t) - \mathrm{LO}(x_{t-1}) - 1$

The reward-maximizing policy can then be defined as

$$\pi(s) = \operatorname*{argmax}_{a \in \mathcal{A}} \mathcal{Q}(s, \cdot)$$

Typically, for better exploration, ε-greedy approach is used, where ε gives the probability that an action a_t is replaced with a randomly sampled one.

DDQN DAC agent on LeadingOnes

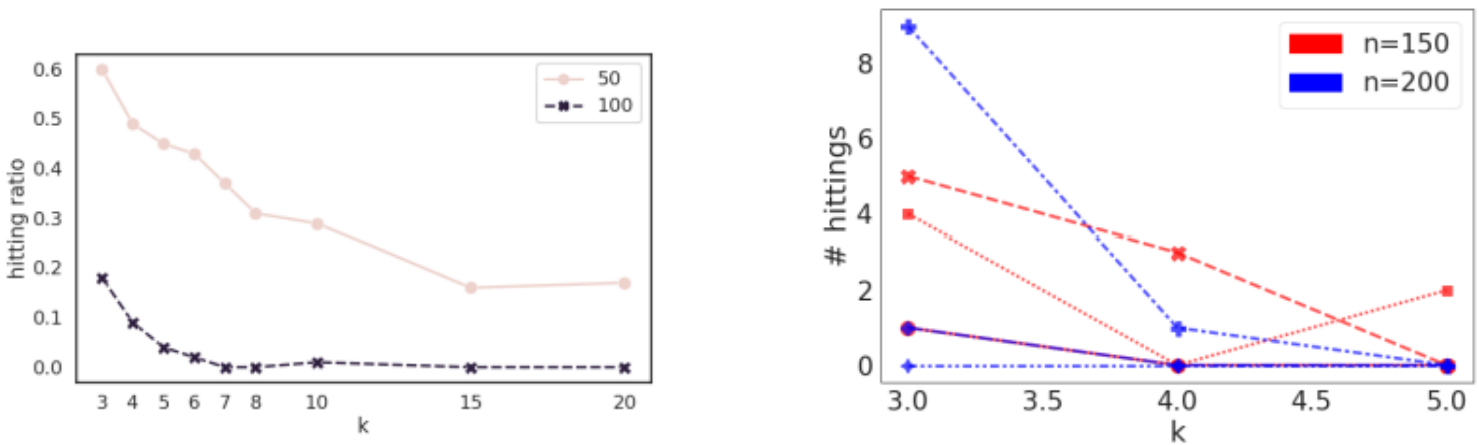


Figure: Hitting ratio and number of hitting points for the DDQN agent in various dimensions

Parameter Control: Exact Runtime for (1+1) Algorithms

General Analysis: Applicable to all (1+1) algorithms on LeadingOnes, RLS in particular.

Theorem (Exact Runtime for LeadingOnes)

Let T be the time until the optimum is found. Then:

$$T \sim \sum_{i=0}^{n-1} X_i \cdot Geom(q_i)$$

where:

- $X_0, \ldots, X_{n-1}$ *are i.i.d. binary random variables.*
- q_i*: Probability that mutation improves fitness* i*.*

Expected Time: $\mathbb{E}[T] = \frac{1}{2} \sum_{i=0}^{n-1} \frac{1}{q_i}$, *with* $\frac{1}{q_i} = \infty$ *if* $q_i = 0$.

Parameter Control: Proof Sketch (Part 1)

Setup:

- Assume uniform distribution of ones in the tail:

$$x_j \sim \text{Uniform}\{0,1\}, \quad \forall j \in [i+2..n]$$

- Define runtimes:
  - T_i^0: Runtime starting with fitness exactly i.
  - T_i^{rand}: Runtime starting with fitness at least i (with x_{i+1} random).
- Note: $T_n^0 = T_n^{\text{rand}}$.

Key Observation:

$$T_i^0 = \text{Geom}(q_i) + T_{i+1}^{\text{rand}}, \quad \forall i < n$$

- Waiting time to flip bit $i+1$ follows $\text{Geom}(q_i)$.

Parameter Control: Proof Sketch (Part 2)

Recursive Relation:

$$\begin{aligned} T_i^{\text{rand}} &= X_i T_i^0 + (1 - X_i) T_{i+1}^{\text{rand}} \\ &= X_i(\text{Geom}(q_i) + T_{i+1}^{\text{rand}}) + (1 - X_i) T_{i+1}^{\text{rand}} \\ &= X_i \text{Geom}(q_i) + T_{i+1}^{\text{rand}} \end{aligned}$$

where:

- X_i is a uniform binary random variable, independent from other randomness.

By Induction:

- Starting from T_n^{rand} and iterating backwards.
- Since $T = T_0^{\text{rand}}$, the theorem holds.

Conclusion:

- The runtime is a sum of independent geometric distributions, scaled by random binary variables.
- Provides a clean, exact expression for $\mathbb{E}[T]$.

Standard Setting

Standard Selection

Candidate y is accepted from x according to the relation $\mathrm{LO}(y) \geq \mathrm{LO}(x)$.

We obtain a linear system in matrix form $Ax = b$ as follows.

$$x = \begin{bmatrix} \mathbb{E}[T_{opt}^{(k_l)}(l,l)] \\ \mathbb{E}[T_{opt}^{(k_{l+1})}(l,l+1)] \\ \vdots \\ \mathbb{E}[T_{opt}^{(k_{n-1})}(l,n-1)] \end{bmatrix} \quad b = \begin{bmatrix} 1 + \sum_{\lambda=l+1}^{n-1} \sum_{\mu=\lambda}^{n-1} \mathbb{P}((\lambda,\mu)|(l,l)) \mathbb{E}[T_{opt}(\lambda,\mu)] \\ 1 + \sum_{\lambda=l+1}^{n-1} \sum_{\mu=\lambda}^{n-1} \mathbb{P}((\lambda,\mu)|(l,l+1)) \mathbb{E}[T_{opt}(\lambda,\mu)] \\ \vdots \\ 1 + \sum_{\lambda=l+1}^{n-1} \sum_{\mu=\lambda}^{n-1} \mathbb{P}((\lambda,\mu)|(l,n-1)) \mathbb{E}[T_{opt}(\lambda,\mu)] \end{bmatrix}$$

$$A = \begin{bmatrix} (1 - \mathbb{P}^{(k_l)}((l,l) \mid (l,l))) & \mathbb{P}^{(k_{l+1})}((l,l+1) \mid (l,l)) & \cdots & \mathbb{P}^{(k_{n-1})}((l,n-1) \mid (l,l)) \\ \mathbb{P}^{(k_l)}((l,l) \mid (l,l+1)) & (1 - \mathbb{P}^{(k_{l+1})}((l,l+1) \mid (l,l+1))) & \cdots & \mathbb{P}^{(k_{n-1})}((l,n-1) \mid (l,l+1)) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbb{P}^{(k_l)}((l,l) \mid (l,n-1)) & \mathbb{P}^{(k_{l+1})}((l,l+1) \mid (l,n-1)) & \cdots & (1 - \mathbb{P}^{(k_{n-1})}((l,n-1) \mid (l,n-1))) \end{bmatrix}$$

Standard Setting

We obtain linear system in matrix form $Ax = b$ as follows.

$$x = \begin{bmatrix} \mathbb{E}[T_{opt}^{(k_l)}(l,l)] \\ \mathbb{E}[T_{opt}^{(k_{l+1})}(l,l+1)] \\ \vdots \\ \mathbb{E}[T_{opt}^{(k_{n-1})}(l,n-1)] \end{bmatrix} \quad b = \begin{bmatrix} 1 + \sum_{\lambda=l+1}^{n-1} \sum_{\mu=\lambda}^{n-1} \mathbb{P}((\lambda,\mu)|(l,l)) \mathbb{E}[T_{opt}(\lambda,\mu)] \\ 1 + \sum_{\lambda=l+1}^{n-1} \sum_{\mu=\lambda}^{n-1} \mathbb{P}((\lambda,\mu)|(l,l+1)) \mathbb{E}[T_{opt}(\lambda,\mu)] \\ \vdots \\ 1 + \sum_{\lambda=l+1}^{n-1} \sum_{\mu=\lambda}^{n-1} \mathbb{P}((\lambda,\mu)|(l,n-1)) \mathbb{E}[T_{opt}(\lambda,\mu)] \end{bmatrix}$$

$$A = \begin{bmatrix} (1 - \mathbb{P}^{(k_l)}((l,l) \mid (l,l))) & \mathbb{P}^{(k_{l+1})}((l,l+1) \mid (l,l)) & \cdots & \mathbb{P}^{(k_{n-1})}((l,n-1) \mid (l,l)) \\ \mathbb{P}^{(k_l)}((l,l) \mid (l,l+1)) & (1 - \mathbb{P}^{(k_{l+1})}((l,l+1) \mid (l,l+1))) & \cdots & \mathbb{P}^{(k_{n-1})}((l,n-1) \mid (l,l+1)) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbb{P}^{(k_l)}((l,l) \mid (l,n-1)) & \mathbb{P}^{(k_{l+1})}((l,l+1) \mid (l,n-1)) & \cdots & (1 - \mathbb{P}^{(k_{n-1})}((l,n-1) \mid (l,n-1))) \end{bmatrix}$$

Approximation

- We take $k_l = k_{l+1} = \cdots = k_{n-1} = k$ to compute $\mathbb{E}[T_{opt}^{(k)}(l,m)]$;
- We then take $k_{opt}(l,m) = \text{argmin}_{k \in [n-l]} \cdot \mathbb{E}[T_{opt}^{(k)}(l,m)]$.

Probabilistic Method: Algorithm

Algorithm Steps:

1. Choose k from distribution $(p_0, p_1, \ldots, p_n)$, where $\Pr[k = i] = p_i$.
2. Flip k bits chosen uniformly at random.

Fitness Level $n-1$:

- Only one valid string: $1^{n-1}0$.
- To improve, flip exactly one bit $\Rightarrow p_1 = 1$.
- Success probability: $\frac{1}{n} \Rightarrow \mathbb{E}[T] = n$.

Fitness Level $n-2$:

- Two states: $S_1 = 1^{n-2}01$ and $S_2 = 1^{n-2}00$.
- Use $p_1 + p_2 = 1$ in both states.

Probabilistic Method: Expected Time Analysis

Define:

- $E[T_1]$ and $E[T_2]$: Expected times from states S_1 and S_2.

System of Equations:

$$\begin{cases} E[T_1] = 1 + \frac{p_1^{(1)}}{n} E[T_2] + \frac{2p_2^{(1)}}{n-1} + \left(1 - \frac{2p_1^{(1)}}{n}\right) E[T_1] \\ E[T_2] = 1 + p_1^{(2)} + \frac{p_1^{(2)}}{n} E[T_1] + \left(1 - \frac{2p_1^{(2)}}{n}\right) E[T_2] \end{cases}$$

Simplified:

$$\begin{cases} (2 - o(1))E[T_1] - E[T_2] = (1 + o(1))\frac{n}{p_1^{(1)}} \\ -E[T_1] + (2 - o(1))E[T_2] = (1 + p_1^{(2)})\frac{n}{p_1^{(2)}} \end{cases}$$

Probabilistic Method: Optimization and Conclusion

Optimization:

- Multiply and add the equations for $E[T_1]$ and $E[T_2]$ to solve.
- Result:

$$\begin{cases} (3-o(1))E[T_1] = (2 \pm o(1))\frac{n}{p_1^{(1)}} + \frac{n}{p_1^{(2)}} + n \\ (3-o(1))E[T_2] = (1+o(1))\frac{n}{p_1^{(1)}} + (2-o(1))\left(\frac{n}{p_1^{(2)}} + n\right) \end{cases}$$

Conclusion:

- Asymptotic minimization occurs when $p_1^{(1)} = p_1^{(2)} = 1$.
- **Optimal Strategy:** Always flip one bit in both states.

Black-box complexity

OneMax unary unbiased complexity

The lower bound $\Omega(n \log n)$ is consequence of the Theorem

Theorem (Theorem 6 in *Lehre, Witt, 2010*)

Let $f : \{0,1\}^n \to \mathbb{R}$ be a function that has a single global optimum (i.e., in the case of maximization, the size of the set $\arg\max f$ is one). The unary unbiased black-box complexity of f is $\Omega(n \log n)$.

The theorem is proved by multiplicative drift analysis, with potential defined as the smallest Hamming distance among the previously queried of any of the previously queried search points to the unique global optimum or its bit-wise complement.

Multiplicative Drift Analysis

Theorem (Multiplicative Drift Theorem)

Let $(X_t)_{t\geq 0}$ be a sequence of non-negative random variables with a finite state space $S \subseteq \mathbb{R}_{\geq 0}$ such that $0 \in S$. Let $s_{\min} := \min(S \setminus \{0\})$, let $T := \inf\{t \geq 0 \mid X_t = 0\}$, and for $t \geq 0$ and $s \in S$, let $\Delta_t(s) := \mathbb{E}[X_t - X_{t+1} \mid X_t = s]$. Suppose that there exists $\delta > 0$ such that for all $s \in S \setminus \{0\}$ and all $t \geq 0$, the drift is

$$\Delta_t(s) \geq \delta s.$$

Then

$$\mathbb{E}[T] \leq 1 + \frac{\mathbb{E}[\log(X_0/s_{\min})]}{\delta}.$$

Black-box complexity

LeadingOnes unary unbiased complexity

The result is proved via additive drift analysis with the potential function defined as a function that maps the state of the search process at time t (i.e., the sequence $\{x(1), f(x(1)), \ldots, x(t), f(x(t))\}$ of the pairs of search points evaluated so far and their respective function values) to the largest number of initial ones and initial zeros in any of the $t + 1$ strings $x(1), \ldots, x(n)$.

Additive drift

Theorem (Additive Drift Theorem)

Let $(X_t)_{t=0,1,2,\ldots}$ be a sequence of non-negative random variables with a finite state space $S \subseteq \mathbb{R}_{\geq 0}$ such that $0 \in S$. Let $T := \inf\{t \geq 0 | X_t = 0\}$.

- *If there exists $\delta > 0$ such that for all $s \in S \setminus \{0\}$ and for all $t \geq 0$,*

$$\Delta_t(s) := \mathbb{E}[X_t - X_{t+1} | X_t = s] \geq \delta,$$

then

$$\mathbb{E}[T] \leq \frac{\mathbb{E}[X_0]}{\delta}.$$

- *If there exists $\delta > 0$ such that for all $s \in S \setminus \{0\}$ and for all $t \geq 0$,*

$$\Delta_t(s) := \mathbb{E}[X_t - X_{t+1} | X_t = s] \leq \delta,$$

then

$$\mathbb{E}[T] \geq \frac{\mathbb{E}[X_0]}{\delta}.$$

Black-box complexity

Definition

For all $n \in \mathbb{N}$ and all $z \in \{0,1\}^n$, let

$$\mathrm{OM}_z : \{0,1\}^n \to [n], \quad x \mapsto \mathrm{OM}_z(x) = |\{i \in [n] \mid x_i = z_i\}|,$$

be the function that assigns to each length-n bit string x the number of bits in which x and z agree. Being the unique optimum of OM_z, the string z is called its *target string*.
We refer to $\textsc{OneMax}_n$, or simply $\textsc{OneMax} := \{\mathrm{OM}_z \mid z \in \{0,1\}^n\}$ as the set of all (generalized) $\textsc{OneMax}$ functions.

The unrestricted black-box complexity for $\textsc{OneMax}_n$ is $\Theta(n/\log n)$.

Black-box complexity

Definition

For all $n \in \mathbb{N}$ and $z \in \{0,1\}^n$, let

$$\mathrm{LO}_z : \{0,1\}^n \to [n], \quad x \mapsto \max\{i \in [0..n] \mid \forall j \in [i] : x_j = z_j\},$$

be the length of the maximal joint prefix of x and z. Let $\textsc{LeadingOnes}^*_n := \{\mathrm{LO}_z \mid z \in \{0,1\}^n\}$.
For $z \in \{0,1\}^n$ and $\sigma \in S_n$ permutation of n elements, let

$$\mathrm{LO}_{z,\sigma} : \{0,1\}^n \to [n], \quad x \mapsto \max\{i \in [0..n] \mid \forall j \in [i] : x_{\sigma(j)} = z_{\sigma(j)}\},$$

be the maximal joint prefix of x and z with respect to σ. The set $\textsc{LeadingOnes}_n$ is the collection of all such functions; i.e.,

$$\textsc{LeadingOnes}_n := \{\mathrm{LO}_{z,\sigma} \mid z \in \{0,1\}^n, \sigma \in S_n\}.$$

The unrestricted black-box complexity for $\textsc{LeadingOnes}_n$ is

CMA-ES (Part 1)

General Structure of CMA-ES

- Set parameters λ, w_i for $i = 1, \ldots, \mu$, c_σ, d_σ, c_c, c_1, and c_μ.
- Initialize:
  - Evolution paths: $p_\sigma = 0$, $p_c = 0$
  - Covariance matrix: $C = I$
  - Generation counter: $g = 0$
- Choose initial distribution mean $m \in \mathbb{R}^n$ and step-size $\sigma \in \mathbb{R}_{>0}$.

While termination criterion not met:

1. $g \leftarrow g + 1$ *(Increment generation counter)*
2. **Sample new population:**
   - For $k = 1, \ldots, \lambda$:
     - $z_k \sim \mathcal{N}(0, I)$
     - $y_k = BDz_k \sim \mathcal{N}(0, C)$
     - $x_k = m + \sigma y_k \sim \mathcal{N}(m, \sigma^2 C)$

CMA-ES (Part 2)

3. **Selection and Recombination:**
   - $\hat{y}_w = \sum_{i=1}^{\mu} w_i y_i$, where $\sum w_i = 1$ and $w_i > 0$
   - $m \leftarrow m + c_m \sigma \hat{y}_w$ *(Update mean)*
4. **Step-size Control:**
   - $p_\sigma \leftarrow (1 - c_\sigma) p_\sigma + \sqrt{c_\sigma (2 - c_\sigma) \mu_{\text{eff}}}\, C^{-1/2} \hat{y}_w$
   - $\sigma \leftarrow \sigma \cdot \exp\left(\frac{c_\sigma}{d_\sigma} \left(\frac{\|p_\sigma\|}{\mathbb{E}\|\mathcal{N}(0,I)\|} - 1 \right) \right)$
5. **Covariance Matrix Adaptation:**
   - $p_c \leftarrow (1 - c_c) p_c + \sqrt{c_c (2 - c_c) \mu_{\text{eff}}}\, \hat{y}_w$
   - $w_i^{\text{rank}} \leftarrow w_i \cdot \left(1 \text{ if } w_i \geq 0 \text{ else } \frac{n}{\|C^{-1/2} y_i\|^2} \right)$
   - $C \leftarrow (1 + c_1 h_\sigma - c_1 - c_\mu \sum w_i^{\text{rank}}) C + c_1 p_c p_c^T + c_\mu \sum_{i=1}^{\lambda} w_i^{\text{rank}} y_i y_i^T$

Return: m and σ as final solution.