Dynamic Parameter Policies for LeadingOnes on Complex State Spaces

Gianluca Covini, Denis Antipov, Carola Doerr
LIP6 Sorbonne Université/CNRS

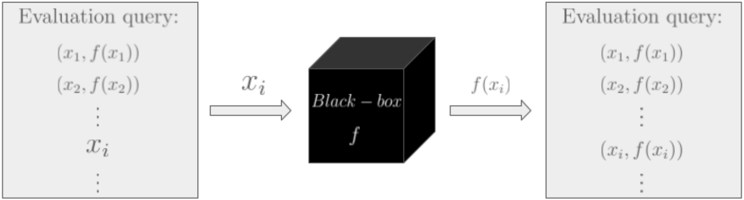
February 28, 2025

Outline

1. Problem Setting
2. Contributions
3. Computational Results
4. Conclusion

Black-box Algorithms

- We consider **black-box algorithms**, which can use only evaluations of the objective function and not on its analytical form.



Parameter Dependence

- Performance of black-box algorithms heavily depend on the choice of **parameters**;

Parameter Dependence

- Performance of black-box algorithms heavily depend on the choice of **parameters**;
- Parameter can be **static** or **dynamic**;

Parameter Dependence

- Performance of black-box algorithms heavily depend on the choice of **parameters**;
- Parameter can be **static** or **dynamic**;

Key insights

- **Adapting parameters dynamically** can significantly improve performance across varying problem landscapes;
- **Benchmark** are crucial for training of learning methods for parameter control.

Randomized Local Search (RLS)

RLS Algorithm

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

Problem Definition

Definition

For any bit string $x \in \{0,1\}^n$ we have

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

Optimization Problem

$$\text{find } x^* \in \operatorname*{argmax}_{x \in \{0,1\}^n} \mathrm{LO}(x)$$

Example

$$\textsc{LeadingOnes}(\color{red}{11}011001) = \mathrm{LO}(\color{red}{11}011001) = 2$$

Problem Definition

Definition

For any bit string $x \in \{0,1\}^n$ we have

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

Optimization Problem

$$\text{find } x^* \in \operatorname*{argmax}_{x \in \{0,1\}^n} \mathrm{LO}(x)$$

Definition

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

Algorithm Configuration Policy

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.

Algorithm Configuration Policy

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 policy π on problem f.

Algorithm Configuration Policy

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 policy π on problem f.

Cost is the **runtime**: the number of evaluation of the objective before evaluating the optimum (we assume it is reachable in our case).

Static Radius Policy for LeadingOnes

Static Policy [Rudolph 1997]

The static policy for the RLS radius is

$$\pi(l) := 1, \quad \forall \text{ LeadingOnes fitness } l$$

This results in an expected runtime of $0.5n^2$.

Dynamic Radius Policy for LeadingOnes

Doerr (2019) defined the first dynamic parameter policy for RLS radius on LeadingOnes.

States are defined as values of LeadingOnes fitness:

$$\pi : \mathcal{S}^{(1)} := [0..n] \to [1..n]$$

Dynamic Radius Policy for LeadingOnes

Doerr (2019) defined the first dynamic parameter policy for RLS radius on LeadingOnes.

States are defined as values of LeadingOnes fitness:

$$\pi : \mathcal{S}^{(1)} := [0..n] \rightarrow [1..n]$$

Idea

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

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

Dynamic Radius Policy for LeadingOnes

Doerr (2019) defined the first dynamic parameter policy for RLS radius on LeadingOnes.

States are defined as values of LeadingOnes fitness:

$$\pi : \mathcal{S}^{(1)} := [0..n] \to [1..n]$$

Idea

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

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

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

Dynamic Policy [Doerr 2019]

The optimal dynamic policy for the RLS radius defined on $\mathcal{S}^{(1)}$ is

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

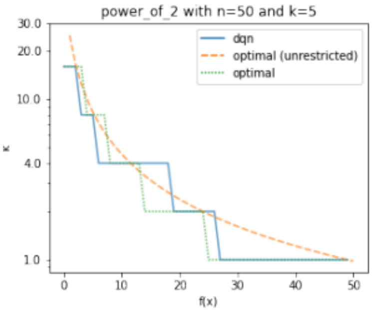
This results in an expected runtime of $0.39n^2$, which corresponds to a 22% improvement of the choice of fixed parameters.

DAC on LeadingOnes

In Biedenkapp et al. 2022, the use of a **DDQN agent** to learn the optimal radius policy for RLS on LeadingOnes with the optimal dynamic policy as ground truth.

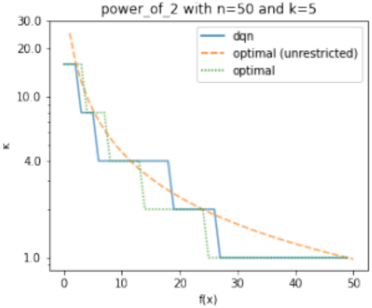
DAC on LeadingOnes

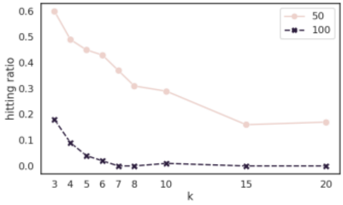
In Biedenkapp et al. 2022, the use of a **DDQN agent** to learn the optimal radius policy for RLS on LeadingOnes with the optimal dynamic policy as ground truth.



DAC on LEADINGONES

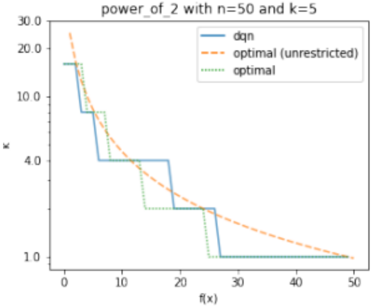
In Biedenkapp et al. 2022, the use of a **DDQN agent** to learn the optimal radius policy for RLS on LEADINGONES with the optimal dynamic policy as ground truth.

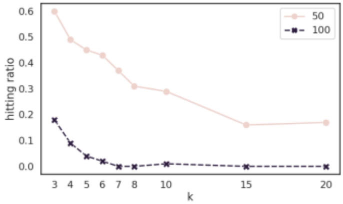




DAC on LeadingOnes

In Biedenkapp et al. 2022, the use of a **DDQN agent** to learn the optimal radius policy for RLS on LeadingOnes with the optimal dynamic policy as ground truth.





Problem

Generalization difficulties for growing n.

Our Setting

Our goal

Extend the radius policies for RLS using more information.

Our Setting

Our goal

Extend the radius policies for RLS using more information.

State spaces

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

Policies on $\mathcal{S}^{(2)}$: Lexicographic Selection

Lexicographic selection

Candidate y is accepted from x iff

- $\mathrm{LO}(y) > \mathrm{LO}(x)$;
- $\mathrm{LO}(y) = \mathrm{LO}(x)$ and $\mathrm{OM}(y) > \mathrm{OM}(x)$.

Policies on $\mathcal{S}^{(2)}$: Lexicographic Selection

Key Steps in Computing the Optimal Policy

- The expected runtime of each state depends only on
  1. expected runtime of lexicographically larger states,
  2. transition probabilities defined by radius k.
- We can compute optimal expected runtime for all states going in descending lexicographic order.
- For each state we can find optimal k in brute force manner.

Policies on $\mathcal{S}^{(2)}$: Standard Selection

Standard Selection

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

Policies on $\mathcal{S}^{(2)}$: Standard Selection

Standard Selection

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

In this setting, **loops** between states with the same LEADINGONES value but different ONEMAX values are possible. For example, the algorithm can transition from $(1,3)$ to $(1,2)$ and then back to $(1,3)$.

For each LEADINGONES fitness level l, we obtain a **non-singular system** of $n-l-1$ equations in $n-l-1$ unknowns.

Policies on $\mathcal{S}^{(2)}$: Standard Selection

For each LeadingOnes fitness level l, we obtain a **non-singular system** of $n - l - 1$ equations in $n - l - 1$ unknowns.

Approximation

To simplify computations, we used the following approximation:

- In the computation of the expected runtime of states with LeadingOnes fitness of l, the same k is applied across all states (l, m) with fixed l.
- Then, for each state, we select the optimal radius in brute force manner as before.

Policies on $\mathcal{S}^{(2)}$: Strict Standard Selection

Strict Standard Selection

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

Policies on $\mathcal{S}^{(2)}$: Strict Standard Selection

Strict Standard Selection

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

There is no possibility of loops between states with the same LO value but different OM values.

This reduces the complexity of the system and places us in a situation analogous to the one in the **lexicographic selection** setting.

Policy on $\mathcal{S}^{(n)}$: Lexicographic Selection

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

We extended the policy with **lexicographic** selection on the space $\mathcal{S}^{(n)} = \{0, 1\}^n$, where each **state** corresponds to a complete bit-string x.

The steps to compute the optimal policy are the same as the policy on $\mathcal{S}^{(2)}$ for lexicographic selection.

Policy on $\mathcal{S}^{(n)}$: Lexicographic Selection

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

We extended the policy with **lexicographic** selection on the space $\mathcal{S}^{(n)} = \{0,1\}^n$, where each **state** corresponds to a complete bit-string x.

The steps to compute the optimal policy are the same as the policy on $\mathcal{S}^{(2)}$ for lexicographic selection.

Results

We then evaluated the policies and settings by computing the **expected runtime** (in function evaluations) from a starting bit-string chosen uniformly at random.

Exact Lexicographic Selection

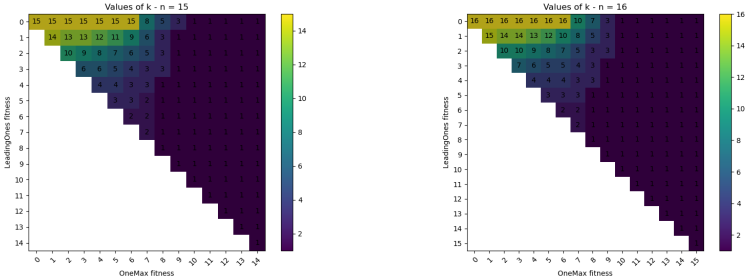
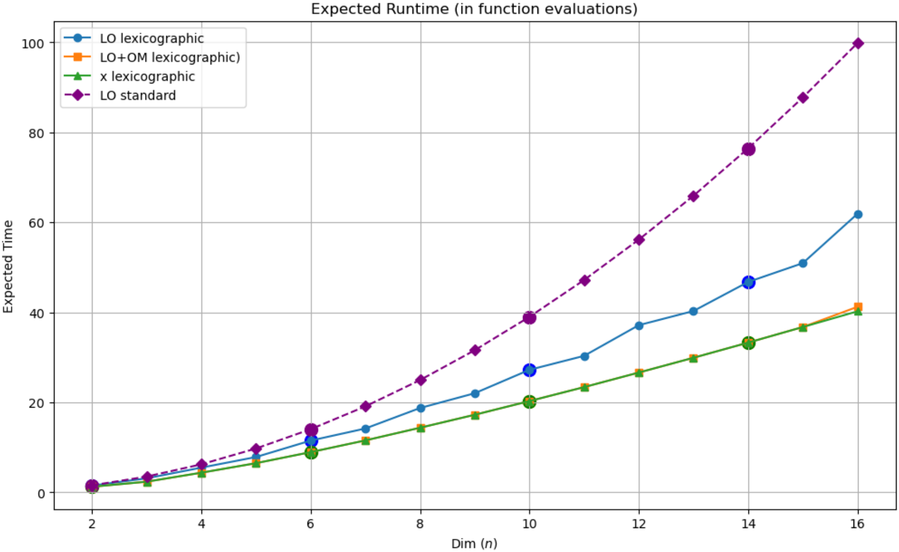


Figure: Heatmaps of optimal policies for lexicographic selection

Exact Results for Lexicographic Selection



Exact Standard Selection Policies

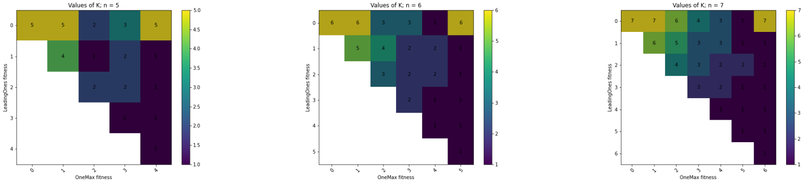
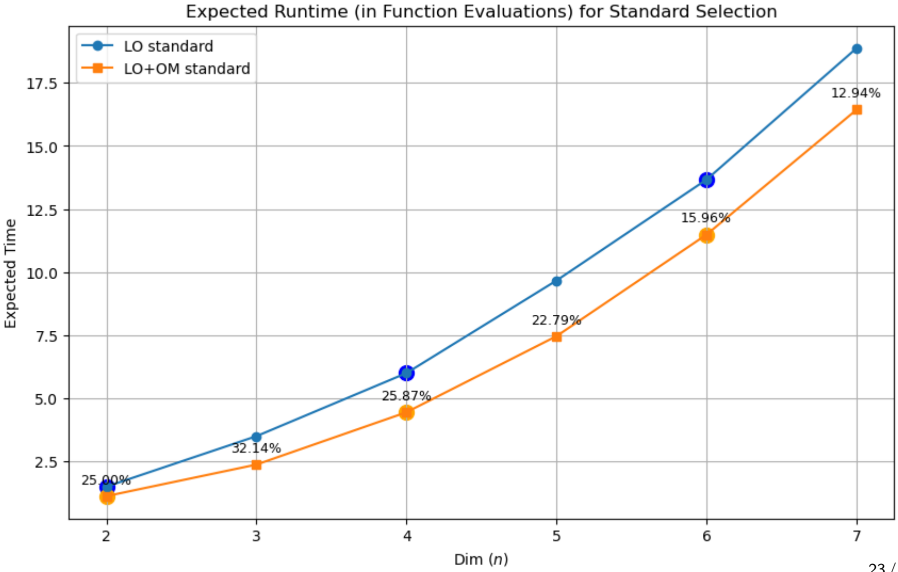


Figure: Heatmaps of approximated optimal policies for standard selection

Exact Standard Selection



Exact Strict Standard Selection

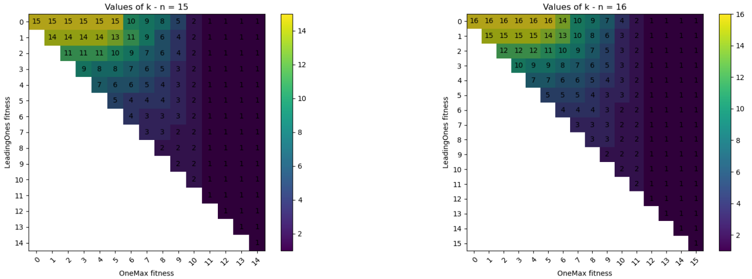
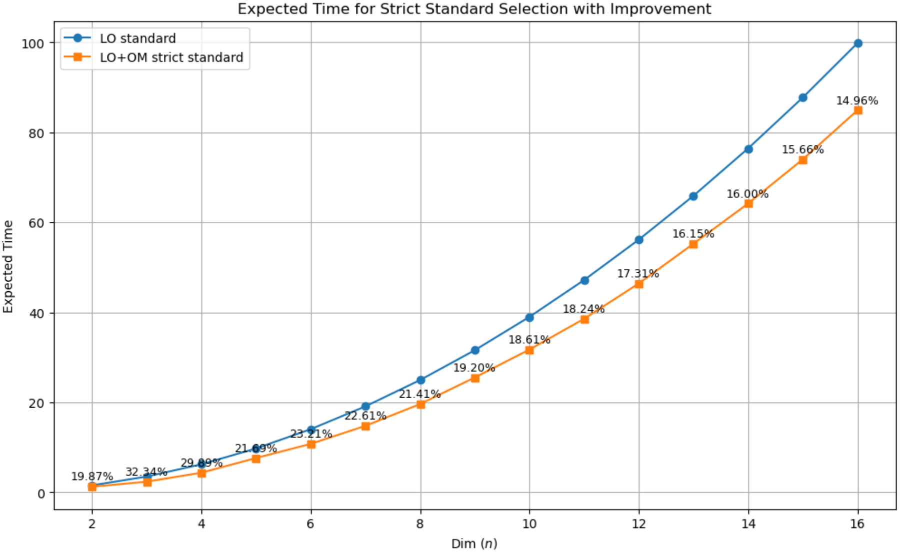


Figure: Heatmaps of policies for strict standard selection

Exact Strict Standard Selection



Heuristic Policy

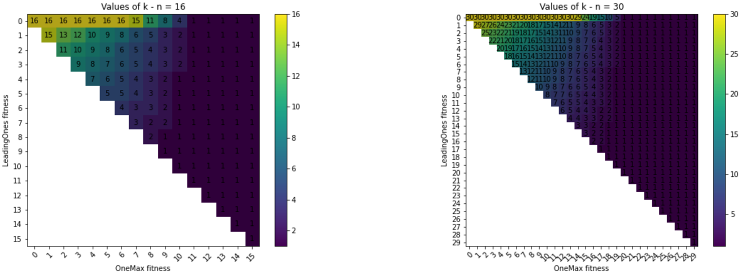
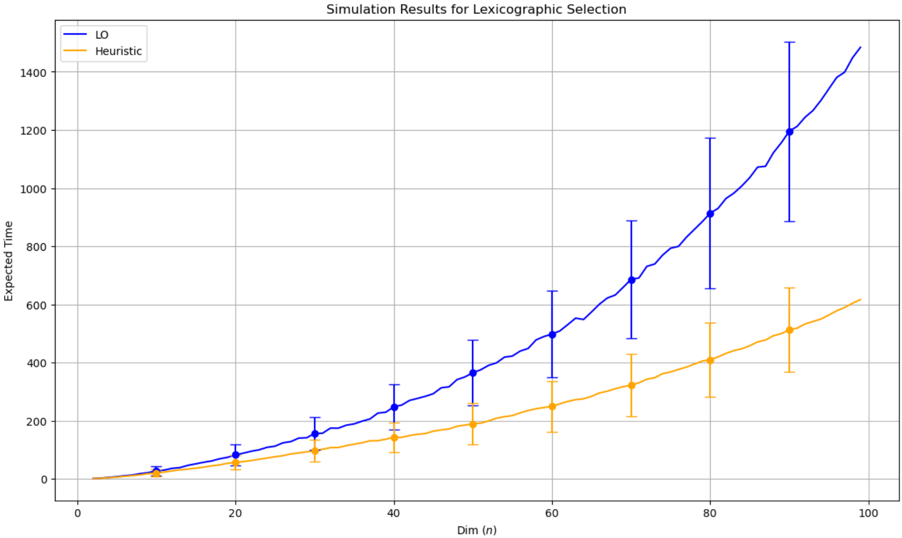
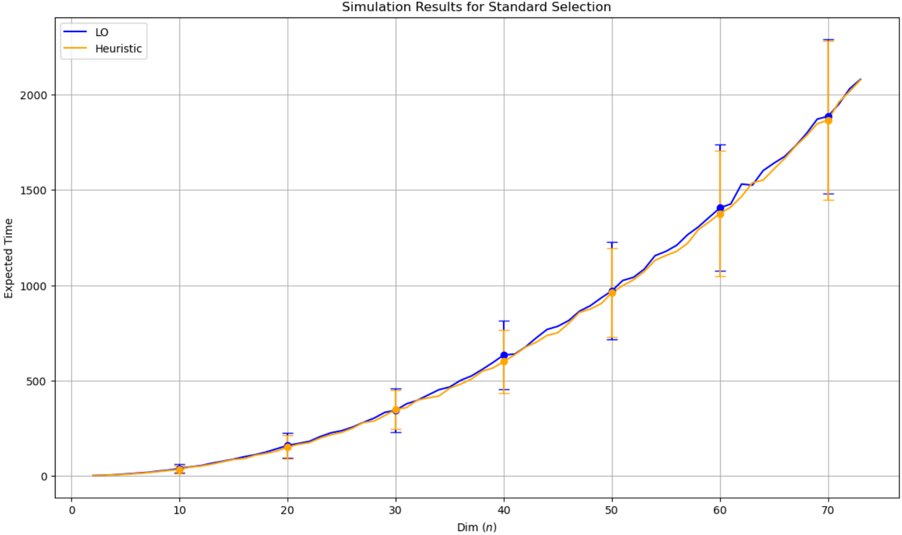


Figure: Heatmaps of heuristic policies

Approximated Lexicographic Selection Results



Approximated Standard Selection Results



Results

Results

- We developed new policies for radius control of RLS on LEADINGONES problem in a lexicographic selection setting and in the standard setting;

Results

Results

- We developed new policies for radius control of RLS on LeadingOnes problem in a lexicographic selection setting and in the standard setting;
- The result of the combination of auxiliary information of both in policy and selection show an improvement in performance;

Results

Results

- We developed new policies for radius control of RLS on LeadingOnes problem in a lexicographic selection setting and in the standard setting;
- The result of the combination of auxiliary information of both in policy and selection show an improvement in performance;
- Including more information than OneMax fitness seems not to lead a further notable improvements;

Results

Results

- We developed new policies for radius control of RLS on LeadingOnes problem in a lexicographic selection setting and in the standard setting;
- The result of the combination of auxiliary information of both in policy and selection show an improvement in performance;
- Including more information than OneMax fitness seems not to lead a further notable improvements;
- In the standard setting improvements in higher dimension are not clear.

Future Directions

Future Directions

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

Future Directions

Future Directions

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

Future Directions

Future Directions

- Study more rigorously the standard setting, trying to validate the results in high dimension.
- Use the tested settings and policies to train an RL agent in the proposed settings, using our policies as new baseline.
- Extend the idea of including more information in policy definition for other algorithms of practical use.

Future Directions

Future Directions

- Study more rigorously the standard setting, trying to validate the results in high dimension.
- Use the tested settings and policies to train an RL agent in the proposed settings, using our policies as new baseline.
- Extend the idea of including more information in policy definition for other algorithms of practical use.

Thank you for your attention!

References I

Biedenkapp, André et al. 2022. “Theory-inspired parameter control benchmarks for dynamic algorithm configuration” In: *Proceedings of the Genetic and Evolutionary Computation Conference*. Pp. 766–775.

Doerr, Benjamin. 2019. “Analyzing randomized search heuristics via stochastic domination”. 773. Pp. 115–137.

Rudolph, Günter. 1997. *Convergence properties of evolutionary algorithms*.

Runtime Heatmaps: Lexicographic Selection

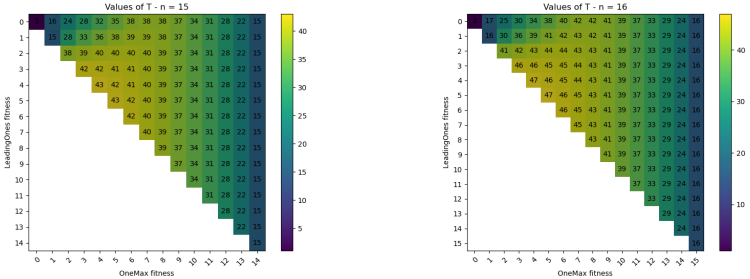


Figure: Heatmaps of runtime (lexicographic selection)

Runtime Heatmaps: Standard Selection

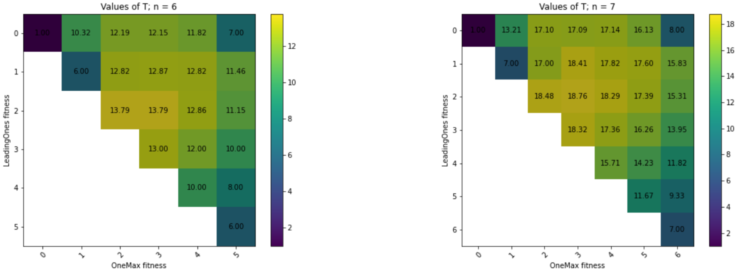


Figure: Heatmaps of runtime (standard selection)

Runtime Heatmaps: Strict Standard Selection

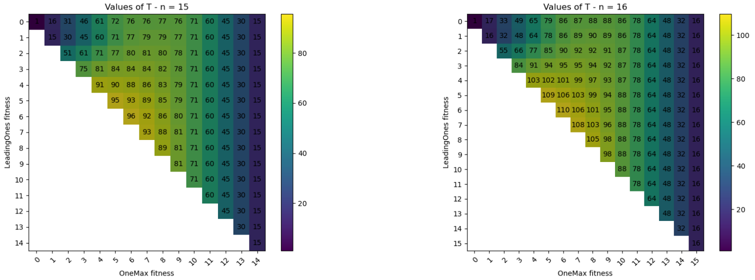


Figure: Heatmaps of runtime (strict standard selection)

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.

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

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

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

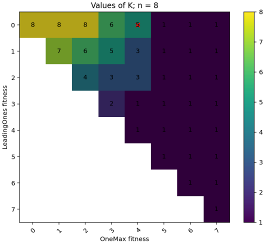
- We take $k_l = k_{l+1} = \cdots = k_{n-1} = k$ to compute $\mathbb{E}[T_{opt}^{(k)}(l,m)]$;
- We then take $k_{opt}(l,m) = \text{argmin}_{k\in[n-l]} \mathbb{E}[T_{opt}^{(k)}(l,m)]$.

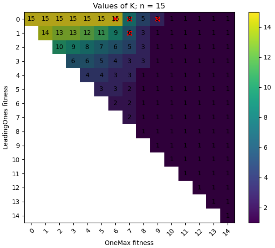
Limited Portfolio

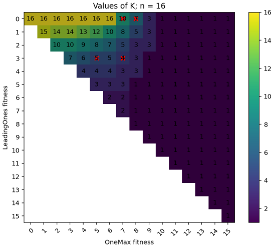
- powers_of_two: $\{2^i \mid 2^i \leq n\}$;
- initial_segment with 3 elements: $[1..3]$;
- evenly_spread with 3 elements: $\{i \cdot \lfloor n/3 \rfloor + 1 \mid i \in [0..2]\}$.

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

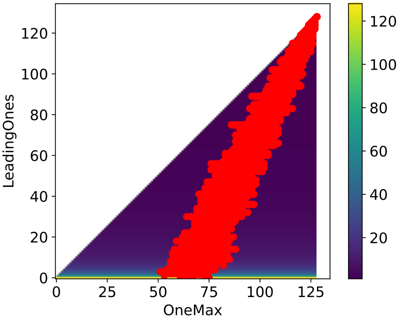
Results for $\mathcal{S}^{(n)}$



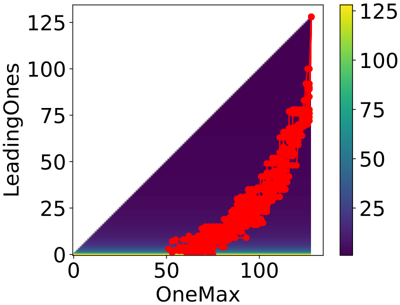




Runs



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



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