Self-adaptation Can Help Evolutionary Algorithms Track Dynamic Optima

Per Kristian Lehre * & Xiaoyu Qin †

School of Computer Science
University of Birmingham
United Kingdom

July 2023

* p.k.lehre@cs.bham.ac.uk
† xxq896@cs.bham.ac.uk

Outline

- **Background**
  - **Self-adaptive parameter control mechanism**
    - Previous results:
      ONEMAX (Doerr et al., 2021)
      Unknown-structure (Case and Lehre, 2020)
      Multi-modal (Lehre and Qin, 2022; Dang and Lehre, 2016)
      Noisy optimisaiton (Qin and Lehre, 2022)
    - (μ, λ) self-adaptive EA (Case and Lehre, 2020)
  - **Dynamic Optimisation Problems:**
    - The objective function changes over time (Jin and Branke, 2005).
    - Dynamic Substring Matching (DSM) problem
- **Our results**
  - Static mutation-based EAs **unlikely solve** DSM problems.
  - (μ, λ) self-adaptive EA **tracks** dynamic optima in DSM w.h.p.
- **Conclusion**

Motivation

- EAs are parameterised algorithms.
- Parameter setting can dramatically impact the performance of EAs (Doerr and Doerr, 2020). $\Longrightarrow$ **IMPORTANCE**
- Parameter setting is instance- and state-dependent (Doerr and Doerr, 2020). $\Longrightarrow$ **DIFFICULTY**

$\downarrow$

Potentially harder in dynamic environments

- Classification scheme of parameter setting (Eiben et al., 1999):

Motivation

- EAs are parameterised algorithms.
- Parameter setting can dramatically impact the performance of EAs (Doerr and Doerr, 2020). $\Longrightarrow$ **IMPORTANCE**
- Parameter setting is instance- and state-dependent (Doerr and Doerr, 2020). $\Longrightarrow$ **DIFFICULTY**

$\downarrow$

Potentially harder in dynamic environments

- Classification scheme of parameter setting (Eiben et al., 1999):

Motivation

- EAs are parameterised algorithms.
- Parameter setting can dramatically impact the performance of EAs (Doerr and Doerr, 2020).
- Parameter setting is instance- and state-dependent (Doerr and Doerr, 2020).

$\Longrightarrow$ **IMPORTANCE**

$\Longrightarrow$ **DIFFICULTY**

$\downarrow$

Potentially harder in dynamic environments

- Classification scheme of parameter setting (Eiben et al., 1999):

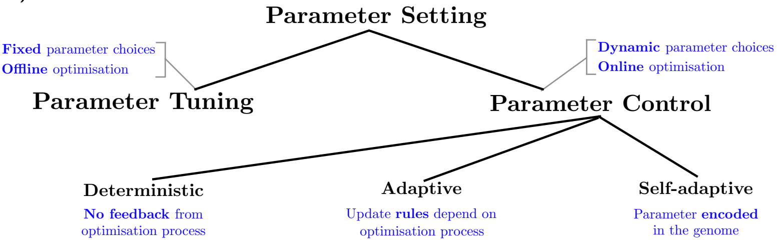
Motivation

- EAs are parameterised algorithms.
- Parameter setting can dramatically impact the performance of EAs (Doerr and Doerr, 2020). ⟹ **IMPORTANCE**
- Parameter setting is instance- and state-dependent (Doerr and Doerr, 2020). ⟹ **DIFFICULTY**

↓

Potentially harder in dynamic environments

- Classification scheme of parameter setting (Eiben et al., 1999):



Self-adaptive Parameter Control Mechanism

The individual with "right" parameter setting is promising to improve.

⇑

Self-adaptation is a more **natural** way to control parameters

⇒ **Encode** parameters into genome and

⇒ **Evolve together** parameters with solutions

Self-adaptive Parameter Control Mechanism

The individual with "right" parameter setting is promising to improve.

⇑

Self-adaptation is a more **natural** way to control parameters

- ⇒ **Encode** parameters into genome and
- ⇒ **Evolve together** parameters with solutions

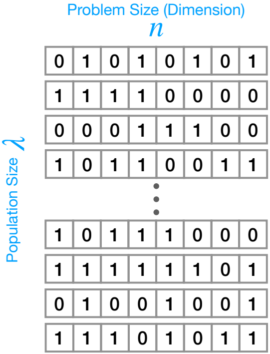
Self-adaptive Parameter Control Mechanism

The individual with "right" parameter setting is promising to improve.

$\Uparrow$

Self-adaptation is a more **natural** way to control parameters

- $\Rightarrow$ **Encode** parameters into genome and
- $\Rightarrow$ **Evolve together** parameters with solutions



$$P \in \left(\{0,1\}^n\right)^\lambda$$

Figure: Static Population

$\Rightarrow$

Self-adaptive Parameter Control Mechanism

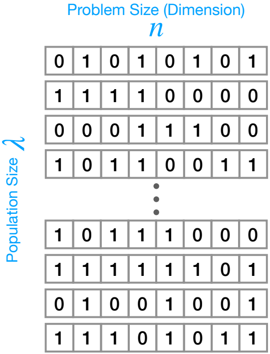
The individual with "right" parameter setting is promising to improve.

⇑

Self-adaptation is a more **natural** way to control parameters

⇒ **Encode** parameters into genome and

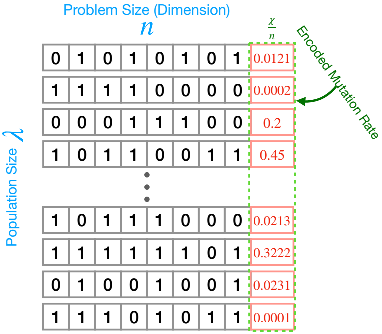
⇒ **Evolve together** parameters with solutions



$P \in \left(\{0,1\}^n\right)^\lambda$

Figure: Static Population

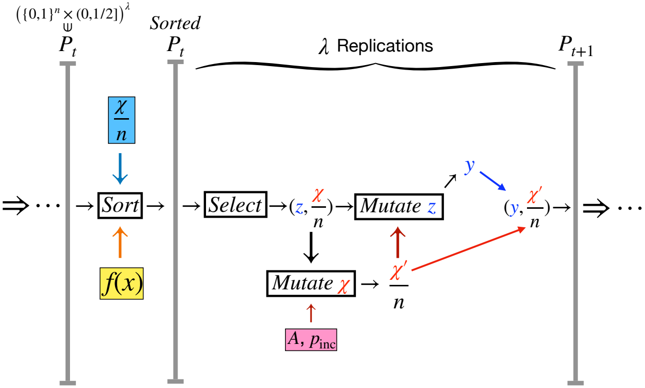
⇒



$P \in \left(\{0,1\}^n \times (0,1/2]\right)^\lambda$

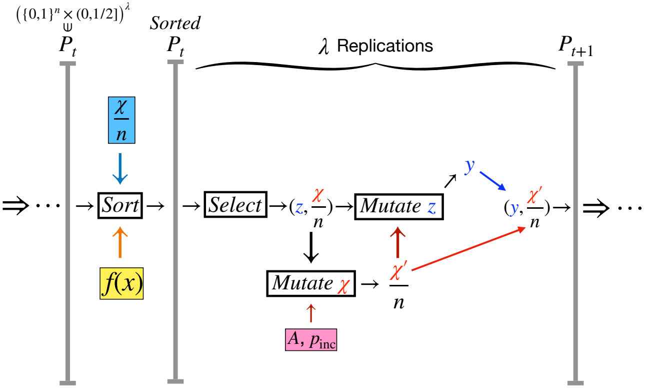
Figure: Self-adaptive Population

A Framework of Self-adaptive EAs



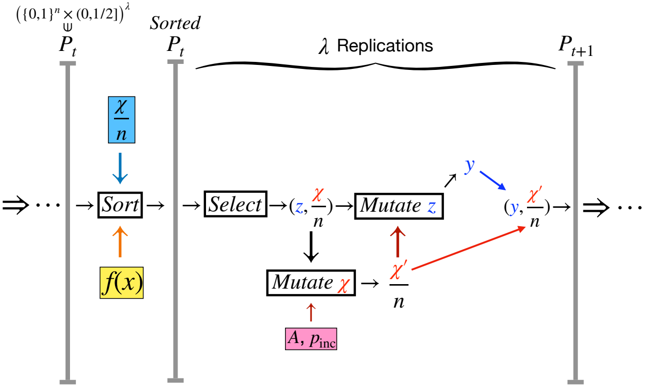
- **Comparison (sorting) rule:** Prefer to higher fitness than higher mutation rate.
- **Selection Mechanism:** k-tournament, (μ,λ), Power-law, etc.
- **Mutation Rate Adaptation Strategy:** $m'(\chi) : \mathbb{R} \to \mathbb{R}$
  $\Rightarrow$ Increase by $\times A > 1$ with prob. p_{inc}, otherwise decrease by $\times 0 < b < 1$.
- **Bitwise Mutation Operator:** Mutate the solution z with adapted (new) mutation rate.

A Framework of Self-adaptive EAs



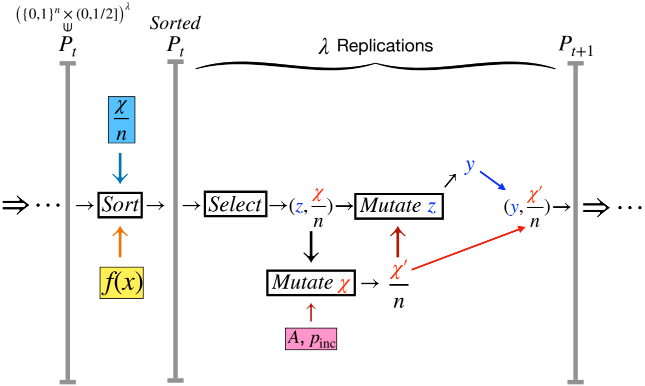
- **Comparison (sorting) rule:** Prefer to higher fitness than higher mutation rate.
- **Selection Mechanism:** k-tournament, (μ,λ), Power-law, etc.
- **Mutation Rate Adaptation Strategy:** $m'(\chi) : \mathbb{R} \to \mathbb{R}$
  $\Rightarrow$ Increase by $\times A > 1$ with prob. p_{inc}, otherwise decrease by $\times 0 < b < 1$.
- **Bitwise Mutation Operator:** Mutate the solution z with adapted (new) mutation rate.

A Framework of Self-adaptive EAs



- **Comparison (sorting) rule:** Prefer to higher fitness than higher mutation rate.
- **Selection Mechanism:** k-tournament, (μ,λ), Power-law, etc.
- **Mutation Rate Adaptation Strategy:** $m'(\chi) : \mathbb{R} \to \mathbb{R}$
  $\Rightarrow$ Increase by $\times A > 1$ with prob. p_{inc}, otherwise decrease by $\times 0 < b < 1$.
- **Bitwise Mutation Operator:** Mutate the solution z with adapted (new) mutation rate.

A Framework of Self-adaptive EAs



- **Comparison (sorting) rule:** Prefer to higher fitness than higher mutation rate.
- **Selection Mechanism:** k-tournament, (μ,λ), Power-law, etc.
- **Mutation Rate Adaptation Strategy:** $m'(\chi) : \mathbb{R} \to \mathbb{R}$
  $\Rightarrow$ Increase by $\times A > 1$ with prob. p_{inc}, otherwise decrease by $\times 0 < b < 1$.
- **Bitwise Mutation Operator:** Mutate the solution z with adapted (new) mutation rate.

Studied Self-adaptive EA

Algorithm 1 (μ, λ) self-adaptive EA (Case and Lehre, 2020)

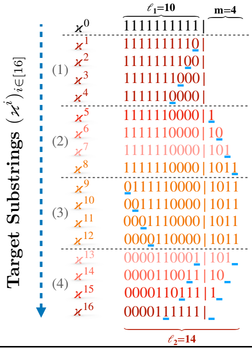
Require: Fitness function $f : \mathcal{X} \to \mathbb{R}$.
Require: Population sizes $\mu, \lambda \in \mathbb{N}$, where $1 \leq \mu \leq \lambda$.
Require: Adaptation parameters $A > 1$, and $b, p_{\text{inc}}, \epsilon \in (0, 1)$.
Require: Initial population $P_0 \in \mathcal{Y}^\lambda$.

```
for τ = 0, 1, 2, ··· until termination condition met do
    Sort P_τ based on P_τ(1) ⪰ ··· ⪰ P_τ(λ) †.
    for i = 1, ..., λ do
        Set (x, χ/n) := P_t(I_t(i)), I_t(i) ~ Unif([μ]).
        Set χ' := { min{Aχ, n/2} with probability p_inc
                  { max{bχ, εn} otherwise.
        Create x' by independently flipping each bit of x with probability χ'/n.
        Set P_{τ+1}(i) := (x', χ'/n).
```

† $(x, \chi) \succeq (x', \chi') \Leftrightarrow f(x) > f(x') \vee (f(x) = f(x') \wedge \chi \geq \chi')$,

Dynamic Substring Matching ($\mathrm{DSM}^{\varkappa,m,\varepsilon,k}$) problem ‡

- To match a sequence of **bit-flipping** and **length-varying** target substrings $(\varkappa^i)_{i\in[4m]}$ in a sequence of corresponding **evaluation budgets** $(\mathcal{T}_i)_{i\in[4m]}$.
- The **length** of target substrings **varies** between ℓ_1 and ℓ_2 where $\ell_2 = \ell_1 + m$.
- **Evaluation budgets** $(\mathcal{T}_i)_{i\in[4m]}$ **depends** on the **lengths** of the target substrings, i.e., $kn^{\varepsilon}|\varkappa^i|$.
- The target substrings are **changed** after evaluation budgets **run out**.

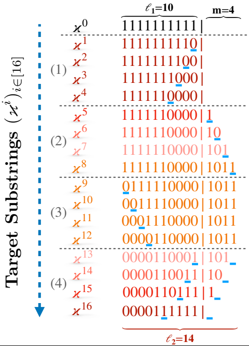


⇐ A sequence of target substrings in an **example** of $\mathrm{DSM}^{\varkappa,m,\varepsilon,k}$ ($\varkappa = 1^{10}$, $n = 20$, $m = 4$), s.t. $\ell_1 = 10$ and $\ell_2 = 14$.

‡ $\varepsilon \in (0,1)$, $k > 0$, $\varkappa \in \{0,1\}^{\ell_1}$ where $\ell_1 \in [n-1]$, and $m \in [n-\ell_1]$ are the parameters of the DSM problem

Dynamic Substring Matching ($\mathrm{DSM}^{\varkappa,m,\varepsilon,k}$) problem ‡

- To match a sequence of **bit-flipping** and **length-varying** target substrings $(\varkappa^i)_{i\in[4m]}$ in a sequence of corresponding **evaluation budgets** $(\mathcal{T}_i)_{i\in[4m]}$.
- The **length** of target substrings **varies** between ℓ_1 and ℓ_2 where $\ell_2 = \ell_1 + m$.
- **Evaluation budgets** $(\mathcal{T}_i)_{i\in[4m]}$ **depends** on the **lengths** of the target substrings, i.e., $kn^{\varepsilon}|\varkappa^i|$.
- The target substrings are **changed** after evaluation budgets **run out**.

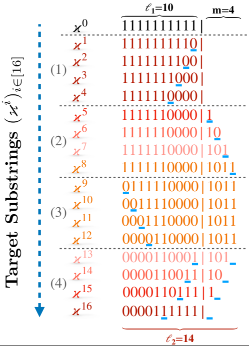


⇐ A sequence of target substrings in an **example** of $\mathrm{DSM}^{\varkappa,m,\varepsilon,k}$ ($\varkappa = 1^{10}$, $n = 20$, $m = 4$), s.t. $\ell_1 = 10$ and $\ell_2 = 14$.

‡ $\varepsilon \in (0,1)$, $k > 0$, $\varkappa \in \{0,1\}^{\ell_1}$ where $\ell_1 \in [n-1]$, and $m \in [n-\ell_1]$ are the parameters of the DSM problem

Dynamic Substring Matching ($\mathrm{DSM}^{\varkappa,m,\varepsilon,k}$) problem ‡

- To match a sequence of **bit-flipping** and **length-varying** target substrings $(\varkappa^i)_{i\in[4m]}$ in a sequence of corresponding **evaluation budgets** $(\mathcal{T}_i)_{i\in[4m]}$.
- The **length** of target substrings **varies** between ℓ_1 and ℓ_2 where $\ell_2 = \ell_1 + m$.
- **Evaluation budgets** $(\mathcal{T}_i)_{i\in[4m]}$ **depends** on the **lengths** of the target substrings, i.e., $kn^{\varepsilon}|\varkappa^i|$.
- The target substrings are **changed** after evaluation budgets **run out**.

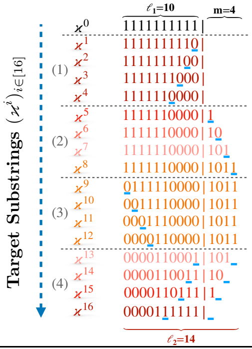


$\Leftarrow$ A sequence of target substrings in an **example** of $\mathrm{DSM}^{\varkappa,m,\varepsilon,k}$ ($\varkappa = 1^{10}$, $n = 20$, $m = 4$), s.t. $\ell_1 = 10$ and $\ell_2 = 14$.

‡ $\varepsilon \in (0,1)$, $k > 0$, $\varkappa \in \{0,1\}^{\ell_1}$ where $\ell_1 \in [n-1]$, and $m \in [n-\ell_1]$ are the parameters of the DSM problem

Dynamic Substring Matching ($\mathrm{DSM}^{\varkappa,m,\varepsilon,k}$) problem ‡

- To match a sequence of **bit-flipping** and **length-varying** target substrings $(\varkappa^i)_{i\in[4m]}$ in a sequence of corresponding **evaluation budgets** $(\mathcal{T}_i)_{i\in[4m]}$.
- The **length** of target substrings **varies** between ℓ_1 and ℓ_2 where $\ell_2 = \ell_1 + m$.
- **Evaluation budgets** $(\mathcal{T}_i)_{i\in[4m]}$ **depends** on the **lengths** of the target substrings, i.e., $kn^{\varepsilon}|\varkappa^i|$.
- The target substrings are **changed** after evaluation budgets **run out**.

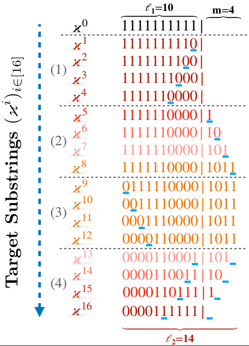


⇐ A sequence of target substrings in an **example** of $\mathrm{DSM}^{\varkappa,m,\varepsilon,k}$ ($\varkappa = 1^{10}$, $n = 20$, $m = 4$), s.t. $\ell_1 = 10$ and $\ell_2 = 14$.

‡ $\varepsilon \in (0,1)$, $k > 0$, $\varkappa \in \{0,1\}^{\ell_1}$ where $\ell_1 \in [n-1]$, and $m \in [n-\ell_1]$ are the parameters of the DSM problem

Dynamic Substring Matching ($\mathrm{DSM}^{\varkappa,m,\varepsilon,k}$) problem ‡

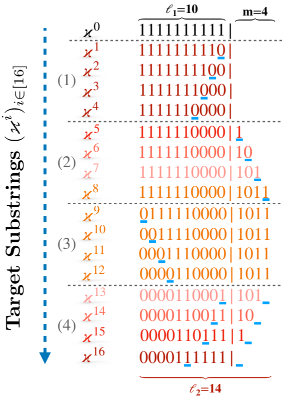
- To match a sequence of **bit-flipping** and **length-varying** target substrings $(\varkappa^i)_{i\in[4m]}$ in a sequence of corresponding **evaluation budgets** $(\mathcal{T}_i)_{i\in[4m]}$.
- The **length** of target substrings **varies** between ℓ_1 and ℓ_2 where $\ell_2 = \ell_1 + m$.
- **Evaluation budgets** $(\mathcal{T}_i)_{i\in[4m]}$ **depends** on the **lengths** of the target substrings, i.e., $kn^{\varepsilon}|\varkappa^i|$.
- The target substrings are **changed** after evaluation budgets **run out**.



$\Leftarrow$ A sequence of target substrings in an **example** of $\mathrm{DSM}^{\varkappa,m,\varepsilon,k}$ ($\varkappa = 1^{10}$, $n = 20$, $m = 4$), s.t. $\ell_1 = 10$ and $\ell_2 = 14$.

‡ $\varepsilon \in (0,1)$, $k > 0$, $\varkappa \in \{0,1\}^{\ell_1}$ where $\ell_1 \in [n-1]$, and $m \in [n-\ell_1]$ are the parameters of the DSM problem

Dynamic Substring Matching ($\mathrm{DSM}^{\varkappa,m,\varepsilon,k}$) problem



Four Stages:

$\Leftarrow$ Starting target substring $\varkappa \in \{0,1\}^{\ell_1}$

$\Leftarrow$ (1) $i \in [m]$, $\varkappa^{i-1}$ and $\varkappa^i$ are the same length but one bit different

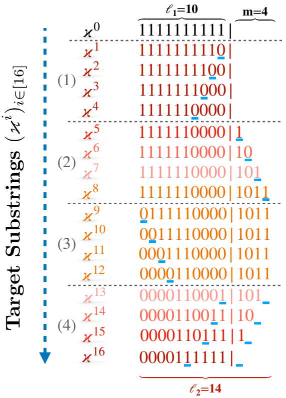
$\Leftarrow$ (2) $i \in [m+1..2m]$, the target substrings are becoming longer

$\Leftarrow$ (3) $i \in [2m+1..3m]$, similar to stage (1): $\varkappa^{i-1}$ and $\varkappa^i$ are the same length but one bit different

$\Leftarrow$ (4) $i \in [3m+1..4m]$, the target substrings are becoming shorter

Algorithms are required to find solutions that match the current target substring within the evaluation budget of $kn^{\varepsilon}|\varkappa^i|$.

Dynamic Substring Matching ($\mathrm{DSM}^{\varkappa,m,\varepsilon,k}$) problem



Four Stages:

$\Leftarrow$ Starting target substring $\varkappa \in \{0,1\}^{\ell_1}$

$\Leftarrow$ (1) $i \in [m]$, $\varkappa^{i-1}$ and $\varkappa^i$ are the same length but one bit different

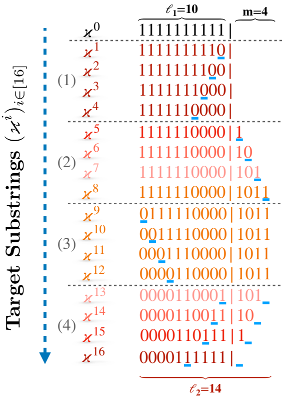
$\Leftarrow$ (2) $i \in [m+1..2m]$, the target substrings are becoming longer

$\Leftarrow$ (3) $i \in [2m+1..3m]$, similar to stage (1): $\varkappa^{i-1}$ and $\varkappa^i$ are the same length but one bit different

$\Leftarrow$ (4) $i \in [3m+1..4m]$, the target substrings are becoming shorter

Algorithms are required to find solutions that match the current target substring within the evaluation budget of $kn^{\varepsilon}|\varkappa^i|$.

Dynamic Substring Matching ($\mathrm{DSM}^{\varkappa,m,\varepsilon,k}$) problem



Four Stages:

$\Leftarrow$ Starting target substring $\varkappa \in \{0,1\}^{\ell_1}$

$\Leftarrow$ (1) $i \in [m]$, $\varkappa^{i-1}$ and $\varkappa^i$ are the same length but one bit different

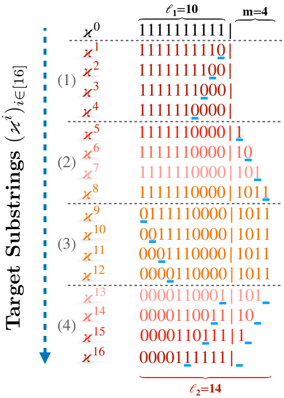
$\Leftarrow$ (2) $i \in [m+1..2m]$, the target substrings are becoming longer

$\Leftarrow$ (3) $i \in [2m+1..3m]$, similar to stage (1): $\varkappa^{i-1}$ and $\varkappa^i$ are the same length but one bit different

$\Leftarrow$ (4) $i \in [3m+1..4m]$, the target substrings are becoming shorter

Algorithms are required to find solutions that match the current target substring within the evaluation budget of $kn^{\varepsilon}|\varkappa^i|$.

Dynamic Substring Matching ($\mathrm{DSM}^{\varkappa,m,\varepsilon,k}$) problem



Four Stages:

$\Leftarrow$ Starting target substring $\varkappa \in \{0,1\}^{\ell_1}$

$\Leftarrow$ (1) $i \in [m]$, $\varkappa^{i-1}$ and $\varkappa^i$ are the same length but one bit different

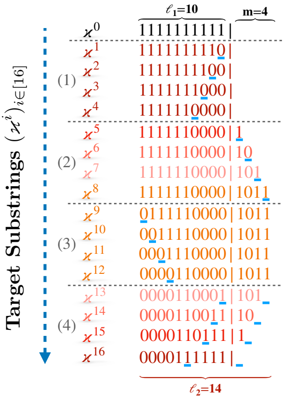
$\Leftarrow$ (2) $i \in [m+1..2m]$, the target substrings are becoming longer

$\Leftarrow$ (3) $i \in [2m+1..3m]$, similar to stage (1): $\varkappa^{i-1}$ and $\varkappa^i$ are the same length but one bit different

$\Leftarrow$ (4) $i \in [3m+1..4m]$, the target substrings are becoming shorter

Algorithms are required to find solutions that match the current target substring within the evaluation budget of $kn^{\varepsilon}|\varkappa^i|$.

Dynamic Substring Matching ($\mathrm{DSM}^{\varkappa,m,\varepsilon,k}$) problem



Four Stages:

$\Leftarrow$ Starting target substring $\varkappa \in \{0,1\}^{\ell_1}$

$\Leftarrow$ (1) $i \in [m]$, $\varkappa^{i-1}$ and $\varkappa^i$ are the same length but one bit different

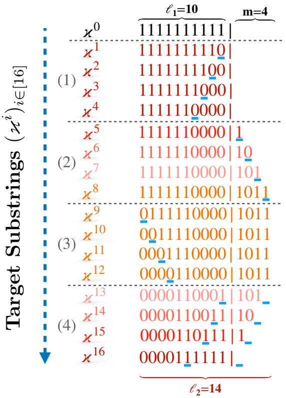
$\Leftarrow$ (2) $i \in [m+1..2m]$, the target substrings are becoming longer

$\Leftarrow$ (3) $i \in [2m+1..3m]$, similar to stage (1): $\varkappa^{i-1}$ and $\varkappa^i$ are the same length but one bit different

$\Leftarrow$ (4) $i \in [3m+1..4m]$, the target substrings are becoming shorter

Algorithms are required to find solutions that match the current target substring within the evaluation budget of $kn^{\varepsilon}|\varkappa^i|$.

Dynamic Substring Matching ($\mathrm{DSM}^{\varkappa,m,\varepsilon,k}$) problem



Four Stages:

$\Leftarrow$ Starting target substring $\varkappa \in \{0,1\}^{\ell_1}$

$\Leftarrow$ (1) $i \in [m]$, $\varkappa^{i-1}$ and $\varkappa^i$ are the same length but one bit different

$\Leftarrow$ (2) $i \in [m+1..2m]$, the target substrings are becoming longer

$\Leftarrow$ (3) $i \in [2m+1..3m]$, similar to stage (1): $\varkappa^{i-1}$ and $\varkappa^i$ are the same length but one bit different

$\Leftarrow$ (4) $i \in [3m+1..4m]$, the target substrings are becoming shorter

Algorithms are required to find solutions that match the current target substring within the evaluation budget of $kn^{\varepsilon}|\varkappa^i|$.

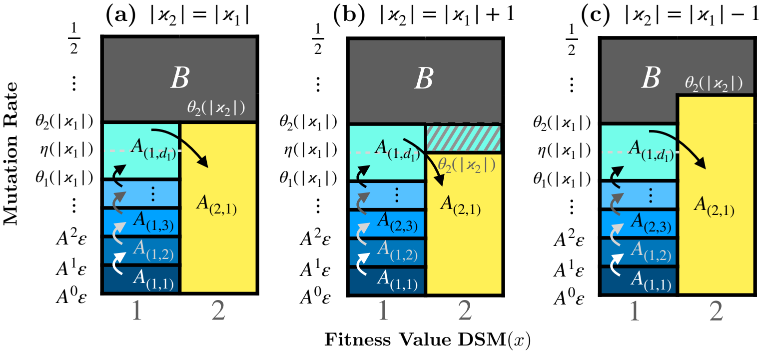
Theoretical Analysis on $\mathrm{DSM}^{\varkappa,m,\varepsilon,k}$

Static mutation-based EAs (Theorem 5.1)	(μ,λ) **self-adaptive EA*** (Theorem 4.1[†])
Constants $\varepsilon, a, \beta, \xi \in (0,1)$ and $k > 0$	
Starting target substring $\lvert\varkappa\rvert = \ell_1 \in \Theta(n^a)$ and $m \in \Theta(n^\beta)$	
$\Downarrow$	
$\ell_1 \in \Theta(n^a)$ and $\ell_2 \in \Theta(n^\beta)$	
$1/2 + \varepsilon < a + 2\varepsilon < \beta \leq 1 - \varepsilon$	$1/34 \leq a \leq \beta \leq 1$
Solves with prob. $e^{-\Omega(n^\xi)}$	Solves with prob. $1 - e^{-\Omega(n^\xi)}$

*Using parameters satisfying $\lambda, \mu = \Theta(n^\xi)$, $\lambda/\mu = \alpha_0 \geq 4$, $A > 1$, $0 < b < 1/(1 + \sqrt{1/\alpha_0(1 - p_{\text{inc}})})$, $(1+\delta)/\alpha_0 < p_{\text{inc}} < 2/5$, and $\epsilon := b'/n$ for any constant $b' > 0$, where A and b are constants.

[†] We provide a **level-based theorem with tail bounds**, which bounds the chance of the algorithm finding the current optima within a given evaluation budget.

Theoretical Analysis on $\mathrm{DSM}^{\varkappa,m,\varepsilon,k}$



Theoretical Analysis on $\mathrm{DSM}^{\varkappa,m,\varepsilon,k}$

Static mutation-based EAs (Theorem 5.1)	(μ, λ) **self-adaptive EA*** (Theorem 4.1†)
Constants $\varepsilon, a, \beta, \xi \in (0,1)$ and $k > 0$	
Starting target substring $\lvert\varkappa\rvert = \ell_1 \in \Theta(n^a)$ and $m \in \Theta(n^\beta)$	
$\Downarrow$	
$\ell_1 \in \Theta(n^a)$ and $\ell_2 \in \Theta(n^\beta)$	
$1/2 + \varepsilon < a + 2\varepsilon < \beta \leq 1 - \varepsilon$	$1/34 \leq a \leq \beta \leq 1$
Solves with prob. $e^{-\Omega(n^\xi)}$	Solves with prob. $1 - e^{-\Omega(n^\xi)}$

* Using parameters satisfying $\lambda, \mu = \Theta(n^\xi)$, $\lambda/\mu = \alpha_0 \geq 4$, $A > 1$, $0 < b < 1/(1 + \sqrt{1/\alpha_0(1 - p_{\text{inc}})})$, $(1 + \delta)/\alpha_0 < p_{\text{inc}} < 2/5$, and $\epsilon := b'/n$ for any constant $b' > 0$, where A and b are constants.

† We provide a **level-based theorem with tail bounds**, which bounds the chance of the algorithm finding the current optima within a given evaluation budget.

Conclusion

- Mutation-based EAs with a fixed mutation rate are **unlikely to track** dynamic optima in DSM problems.
- The self-adaptive EA **tracks** them with an overwhelmingly high probability.
- Furthermore, we provide a **level-based theorem with tail bounds**.
- **Future work:**
  - To analyse self-adaptive EAs on other existing dynamic problems.
  - To explore other parameter control mechanisms under dynamics.
  - . . .

Thank You

Per Kristian Lehre & Xiaoyu QIN

Theory of Evolutionary Computation Group

School of Computer Science

University of Birmingham

United Kingdom

Reference I

Case, B. and Lehre, P. K. (2020). Self-adaptation in non-Elitist Evolutionary Algorithms on Discrete Problems with Unknown Structure. *IEEE Transactions on Evolutionary Computation*, 24(4):650–663.

Dang, D.-C. and Lehre, P. K. (2016). Self-adaptation of Mutation Rates in Non-elitist Populations. In *Parallel Problem Solving from Nature PPSN XIV*, volume 9921, pages 803–813. Springer International Publishing.

Doerr, B. and Doerr, C. (2020). Theory of Parameter Control for Discrete Black-Box Optimization: Provable Performance Gains Through Dynamic Parameter Choices. In Doerr, B. and Neumann, F., editors, *Theory of Evolutionary Computation: Recent Developments in Discrete Optimization*, pages 271–321. Springer International Publishing.

Doerr, B., Witt, C., and Yang, J. (2021). Runtime Analysis for Self-adaptive Mutation Rates. *Algorithmica*, 83(4):1012–1053.

Eiben, A. E., Hinterding, R., and Michalewicz, Z. (1999). Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 3(2):124–141.

Jin, Y. and Branke, J. (2005). Evolutionary optimization in uncertain environments-a survey. *IEEE Transactions on Evolutionary Computation*, 9(3):303–317.

Lehre, P. K. and Qin, X. (2022). Self-Adaptation via Multi-Objectivisation: A Theoretical Study. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '22, pages 1417–1425. Association for Computing Machinery.

Qin, X. and Lehre, P. K. (2022). Self-adaptation via Multi-objectivisation: An Empirical Study. In *Parallel Problem Solving from Nature PPSN XVII*, pages 308–323. Springer International Publishing.

Definition of the DSM Problem

Definition

Let $\varkappa$ be some starting target substring where $|\varkappa| =: \ell_1 \in [n-1]$, and m be a positive integers where $\ell_1 + m =: \ell_2 \leq n$. Let $(\varkappa^i)_{i\geq 0}$ be a sequence of target substrings generated by

$$\varkappa^i := \begin{cases} \varkappa & \text{if } i = 0, \\ z, \text{ where } z \sim \mathrm{Unif}(\mathrm{N}(\varkappa^{i-1})) & \text{if } 1 \leq i \leq m, \\ \varkappa^{i-1} \diamond a, \text{ where } a \sim \mathrm{Unif}(\{0,1\}) & \text{if } m+1 \leq i \leq 2m, \\ z, \text{ where } z \sim \mathrm{Unif}(\mathrm{N}(\varkappa^{i-1})) & \text{if } 2m+1 \leq i \leq 3m, \\ z, \text{ where } z \sim \mathrm{Unif}\left(\mathrm{N}\left(\varkappa^{i-1}_{1:(|\varkappa^{i-1}|-1)}\right)\right) & \text{if } 3m+1 \leq i \leq 4m. \end{cases}$$

Let $(\mathcal{T}_i)_{i\in\mathbb{N}}$ be a sequence of the numbers of evaluation moving from $\varkappa^{i-1}$ to $\varkappa^i$ (evaluation budget for $\varkappa^i$) generated by $\mathcal{T}_i := kn^{\varepsilon}|\varkappa_{i+1}|$, where $\varepsilon \in (0,1)$ and $k > 0$ are some constants. For $t \in \mathbb{N}$, the dynamic substring matching (DSM) problem with the starting target substring $\varkappa$ is defined as:

$$\mathrm{DSM}_t^{\varkappa,m,\varepsilon,k}(x) := \begin{cases} 2 & \text{if } \mathrm{M}(\varkappa(t), x) = 1, \\ 1 & \text{else if } \mathrm{M}(\varkappa'(t), x) = 1, \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

where $\varkappa(t) := \varkappa^i$, and $\varkappa'(t) := \varkappa^{i-1}$,

for $i = \begin{cases} 1 & \text{if } t \leq \mathcal{T}_1, \\ 1 + \max\left\{j \mid \sum_{i=1}^{j} \mathcal{T}_i \leq t\right\} & \text{otherwise.} \end{cases}$

Level-based Theorem with Tail Bounds

Theorem

Let $(B, A_0, A_1, \cdots, A_m)$ *be a partition of* $\mathcal{X}$*. Suppose there exist* $z_1, \cdots, z_{m-1}$*,* $\delta \in (0,1)$*, and* $\gamma_0, \psi_0 \in (0,1)$*, such that the following conditions hold for any population* $P \in \mathcal{X}^\lambda$ *in Algorithm 2 of this paper,*

(C0) *for all* $\psi \in [\psi_0, 1]$*, if* $|P \cap B| \leq \psi\lambda$*, then* $\Pr_{y \sim D(P)}(y \in B) \leq (1-\delta)\psi$*,*

(C1) *for all* $j \in [m-1]$*, if* $|P \cap B| \leq \psi_0\lambda$ *and* $|P \cap A_{\geq j}| \geq \gamma_0\lambda$*, then* $\Pr_{y \sim D(P)}\left(y \in A_{\geq j+1}\right) \geq z_j$*,*

(C2) *for all* $j \in [0..m-1]$*, and* $\gamma \in [1/\lambda, \gamma_0]$ *if* $|P \cap B| \leq \psi_0\lambda$ *and* $|P \cap A_{\geq j}| \geq \gamma_0\lambda$ *and* $|P \cap A_{\geq j+1}| \geq \gamma\lambda$*, then* $\Pr_{y \sim D(P)}\left(y \in A_{\geq j+1}\right) \geq (1+\delta)\gamma$*.*

Let $T := \min\{t\lambda \mid |P_t \cap A_m| \geq \gamma_0\lambda \text{ and } |P_t \cap B| \leq \psi_0\lambda\}$*, and assume the algorithm with population size* $\lambda \in \mathbb{N}$ *and an initial population* P_0 *satisfying* $|P_0 \cap A_1| \geq \gamma_0\lambda$ *and* $|P_0 \cap B| \leq \psi_0\lambda$*, then*

$$\Pr(T \leq \eta\tau) > \left(1 - 2\eta\tau e^{-\delta^2 \min\{\psi_0,\gamma_0\}\lambda/4}\right)\left(1 - me^{-\eta\rho^{-\frac{\ln(\gamma_0)}{\ln(1+\delta/2)}-2}}\right)$$

for any $\eta \in \left(0, e^{\delta^2 \min\{\psi_0,\gamma_0\}\lambda/4}/\tau\right)$*, where* $\rho = \frac{e^{\delta^2/8}}{e^{\delta^2/8}-1}$ *and* $\tau := \lambda^{17/\delta^3}\left(\sum_{j=1}^{m-1}\frac{1}{z_j} + m\lambda\left(\frac{\ln(\gamma_0\lambda)}{\ln(1+\delta/2)}+1\right)\right)$.