Self-adaptive Parameter Control Mechanism in Evolutionary Computation

Xiaoyu Qin*

Theory of Evolutionary Computation Group
School of Computer Science
University of Birmingham
United Kingdom

Nov 2023

*xxq896@cs.bham.ac.uk

About Me

Hello!

I'm Xiaoyu QIN 秦霄羽.

I enrolled in Feb. 2020 as a PhD student
in University of Birmingham, supervised by
Per Kristian Lehre and Ata Kaban.

Research Interests:

- *Evolutionary Computation* $\sim$ *Optimisation & Search* $\sim$ *AI*
- *AI & ML and Applications*

Education:

- **PhD** in Computer Sci (*2020 - 2023*) *University of Birmingham*
- **MSc** in Advanced Computer Sci (*2019*) *University of Birmingham*
- **Visiting Student** in Computer Sci (*2017 - 2018*) *Loughborough University*
- **BEng** in Computer Sci & Tech (*2018*) *Shanxi University*

Teaching:

- **Teaching Assistance** (*2020 - 2022*) *University of Birmingham*

  Modules: *Neural Computation*, *AI & ML*, etc.

About Me

Outputs:

[1] *Self-adaptation Can Improve the Noise-tolerance of Evolutionary Algorithms.* (with P.K. Lehre)* In Proc. of FOGA '23.

[2] *Self-adaptation Can Help Evolutionary Algorithms Track Dynamic Optima.* (with P.K. Lehre)* In Proc. of GECCO '23.

[3] *Optimizing Chance-Constrained Submodular Problems with Variable Uncertainties.* (X. Yan, A.V. Do, F. Shi, X. Qin and F. Neumann) In Proc. of ECAI '23.

[4] *Self-adaptation via Multi-objectivisation: An Empirical Study.* (X. Qin and P.K. Lehre) In Proc. of PPSN '22.

[5] *More Precise Runtime Analyses of Non-elitist Evolutionary Algorithms in Uncertain Environments.* (with P.K. Lehre)* In *Algorithmica* (2022).

[6] *Self-adaptation via Multi-objectivisation: A Theoretical Study.* (with P.K. Lehre)* In Proc. of GECCO '22.

[7] *Fast Non-elitist Evolutionary Algorithms with Power-law Ranking Selection.* (with D.C. Dang, A. Eremeev and P.K. Lehre)* In Proc. of GECCO '22.

[8] *More Precise Runtime Analyses of Non-elitist EAs in Uncertain Environments.* (with P.K. Lehre)* In Proc. of GECCO '21.

[9] *Fast Non-elitist Evolutionary Algorithms with Power-law Ranking Selection.* (with D.C. Dang, A. Eremeev and P.K. Lehre)* Submitted to *Algorithmica* (Under review).

---

*Following the convention of Theoretical Computer Science that the co-authors are listed in *Alphabetical Order*.

About Me

Professional Activities:

- **PC Member** of conferences GECCO[†], PPSN[‡], ECAI[§], etc.
- **Reviewer** of journals *IEEE Transactions on Evolutionary Computation* (TEVC), *Theoretical Computer Science* (TCS), *China Information Science*, etc.
- **Invited Talks** & **Presentations** in seminars and universities.

[†] GECCO: The Genetic and Evolutionary Computation Conference.

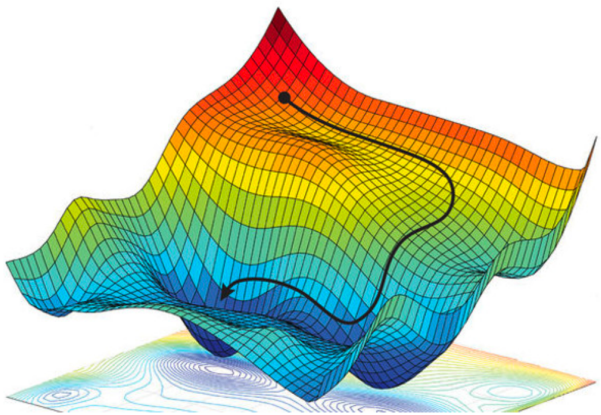
[‡] PPSN: International Conference on Parallel Problem Solving from Nature.

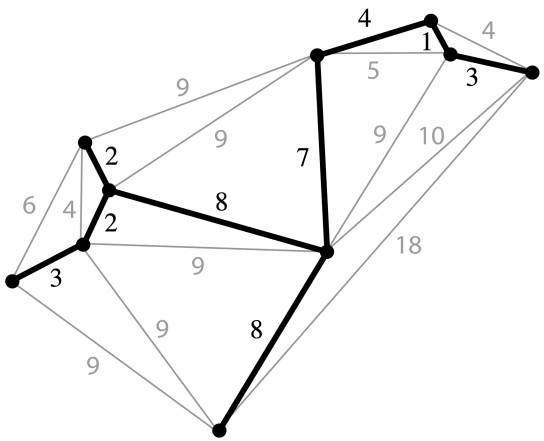
[§] ECAI: European Conference on Artificial Intelligence.

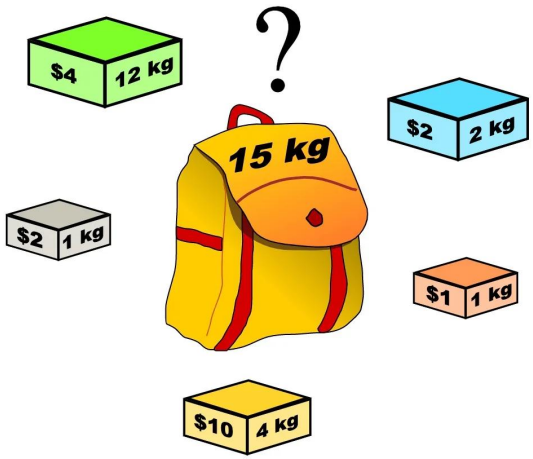
Self-adaptive Parameter Control Mechanism in Evolutionary Computation

$$\underline{Theory} \sim EvolutionaryComputation \sim Optimisation\&Search \sim AI$$
$$\underline{ParameterConfiguration} \sim EvolutionaryComputation \sim Optimisation\&Search \sim AI$$

Optimisation & Search ∼ AI







* Image sources: https://arxiv.org/pdf/1805.04829.pdf, https://www.51cto.com/article/641654.html, https://en.wikipedia.org/wiki/Combinatorial_optimization

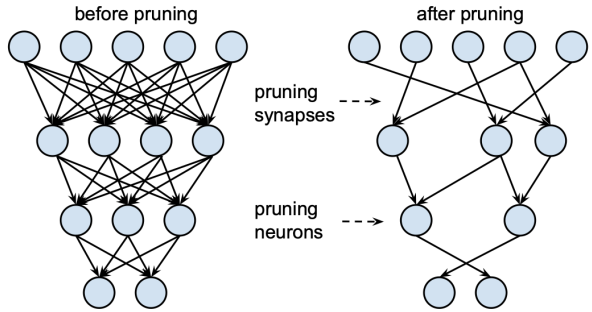
Optimisation & Search in Real World

- **Machine Learning**
  - ML Models training & parameters tuning
  - Neural Architecture Search (NAS)
  - Neural network pruning
- **Software Engineering**
  - Project management
  - Testing
  - The next release problem
- **Intelligent Vehicles**
  - Route planning
  - Decision making
- **Industrial Field**
  - Robot design
  - Antenna design
  - Warehouse Management

‡ Image sources: https://arxiv.org/pdf/1506.02626.pdf, https://en.wikipedia.org/wiki/Evolved_antenna, https://www.nissan-global.com/EN/INNOVATION/TECHNOLOGY/ARCHIVE/IRP/, https://robot.ofweek.com/2022-10/ART-8321200-8100-30576584.html

Optimisation & Search in Real World

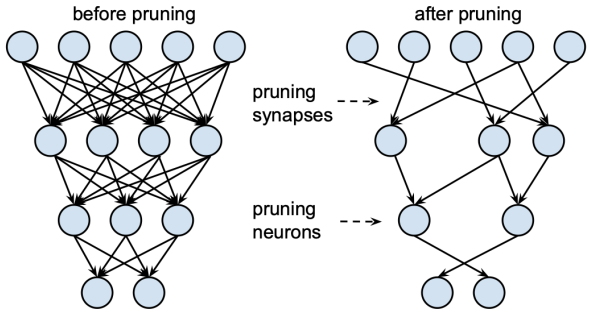
- **Machine Learning**
  - ML Models training & parameters tuning
  - Neural Architecture Search (NAS)
  - Neural network pruning
- **Software Engineering**
  - Project management
  - Testing
  - The next release problem
- **Intelligent Vehicles**
  - Route planning
  - Decision making
- **Industrial Field**
  - Robot design
  - Antenna design
  - Warehouse Management



‡ Image sources: https://arxiv.org/pdf/1506.02626.pdf, https://en.wikipedia.org/wiki/Evolved_antenna, https://www.nissan-global.com/EN/INNOVATION/TECHNOLOGY/ARCHIVE/IRP/, https://robot.ofweek.com/2022-10/ART-8321200-8100-30576584.html

Optimisation & Search in Real World

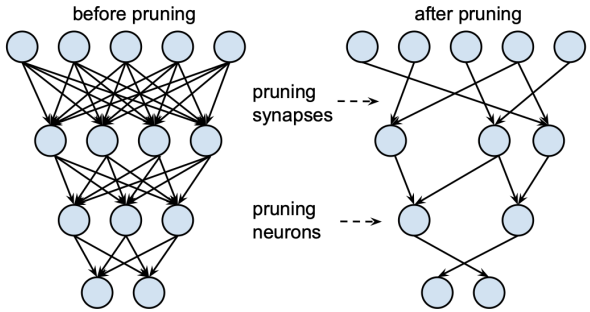
- **Machine Learning**
  - ML Models training & parameters tuning
  - Neural Architecture Search (NAS)
  - Neural network pruning
- **Software Engineering**
  - Project management
  - Testing
  - The next release problem
- **Intelligent Vehicles**
  - Route planning
  - Decision making
- **Industrial Field**
  - Robot design
  - Antenna design
  - Warehouse Management

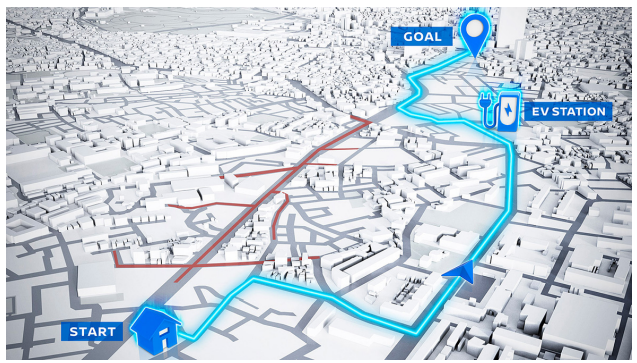


‡ Image sources: https://arxiv.org/pdf/1506.02626.pdf, https://en.wikipedia.org/wiki/Evolved_antenna, https://www.nissan-global.com/EN/INNOVATION/TECHNOLOGY/ARCHIVE/IRP/, https://robot.ofweek.com/2022-10/ART-8321200-8100-30576584.html

Optimisation & Search in Real World

- **Machine Learning**
  - ML Models training & parameters tuning
  - Neural Architecture Search (NAS)
  - Neural network pruning
- **Software Engineering**
  - Project management
  - Testing
  - The next release problem
- **Intelligent Vehicles**
  - Route planning
  - Decision making
- **Industrial Field**
  - Robot design
  - Antenna design
  - Warehouse Management

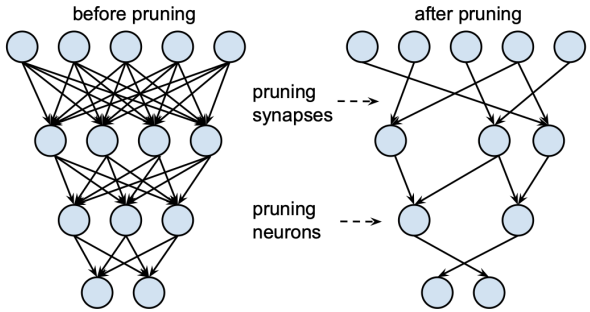


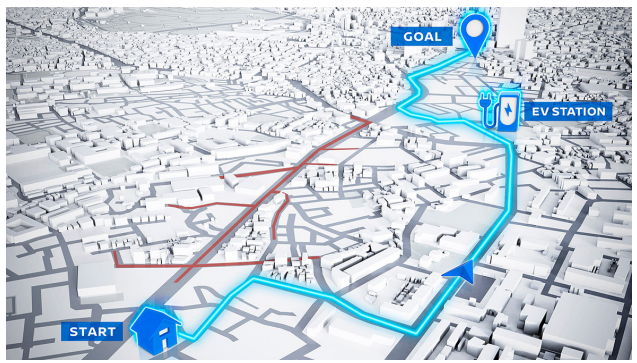


‡ Image sources: https://arxiv.org/pdf/1506.02626.pdf, https://en.wikipedia.org/wiki/Evolved_antenna, https://www.nissan-global.com/EN/INNOVATION/TECHNOLOGY/ARCHIVE/IRP/, https://robot.ofweek.com/2022-10/ART-8321200-8100-30576584.html

Optimisation & Search in Real World

- **Machine Learning**
  - ML Models training & parameters tuning
  - Neural Architecture Search (NAS)
  - Neural network pruning
- **Software Engineering**
  - Project management
  - Testing
  - The next release problem
- **Intelligent Vehicles**
  - Route planning
  - Decision making
- **Industrial Field**
  - Robot design
  - Antenna design
  - Warehouse Management





‡ Image sources: https://arxiv.org/pdf/1506.02626.pdf, https://en.wikipedia.org/wiki/Evolved_antenna, https://www.nissan-global.com/EN/INNOVATION/TECHNOLOGY/ARCHIVE/IRP/, https://robot.ofweek.com/2022-10/ART-8321200-8100-30576584.html

Optimisation Problem

- **Aim:** To find the best solution from all feasible solutions.
- **Mathematical description:**
  - **Given:** a function $f : \mathcal{S} \to \mathbb{R}$
  - **Sought:** find an element $\hat{x} \in \mathcal{S}$ such that $f(\hat{x}) \geq f(x)$ for any $x \in \mathcal{S}$[¶]
- The *search space* $\mathcal{S}$ can be continuous, discrete, or mixed.
- The *objective (fitness) function* f is usually unknown for algorithms.
- The "property" of f is sometime unknown for algorithms.

$\Downarrow$

Black-box Optimisation Problem (BBOP)

[¶] Assume to maximise the function.

Optimisation Problem

- **Aim:** To find the best solution from all feasible solutions.
- **Mathematical description:**
  - **Given:** a function $f : \mathcal{S} \to \mathbb{R}$
  - **Sought:** find an element $\hat{x} \in \mathcal{S}$ such that $f(\hat{x}) \geq f(x)$ for any $x \in \mathcal{S}$[¶]
- The *search space* $\mathcal{S}$ can be continuous, discrete, or mixed.
- The *objective (fitness) function* f is usually unknown for algorithms.
- The "property" of f is sometime unknown for algorithms.

$\Downarrow$

Black-box Optimisation Problem (BBOP)

[¶] Assume to maximise the function.

Optimisation Problem

- **Aim:** To find the best solution from all feasible solutions.
- **Mathematical description:**
  - **Given:** a function $f : \mathcal{S} \to \mathbb{R}$
  - **Sought:** find an element $\hat{x} \in \mathcal{S}$ such that $f(\hat{x}) \geq f(x)$ for any $x \in \mathcal{S}$[¶]
- The *search space* $\mathcal{S}$ can be continuous, discrete, or mixed.
- The *objective (fitness) function* f is usually unknown for algorithms.
- The "property" of f is sometime unknown for algorithms.

$\Downarrow$

Black-box Optimisation Problem (BBOP)

[¶] Assume to maximise the function.

Optimisation Problem

- **Aim:** To find the best solution from all feasible solutions.
- **Mathematical description:**
  - **Given:** a function $f : \mathcal{S} \to \mathbb{R}$
  - **Sought:** find an element $\hat{x} \in \mathcal{S}$ such that $f(\hat{x}) \geq f(x)$ for any $x \in \mathcal{S}$[¶]
- The *search space* $\mathcal{S}$ can be continuous, discrete, or mixed.
- The *objective (fitness) function* f is usually unknown for algorithms.
- The "property" of f is sometime unknown for algorithms.

$\Downarrow$

Black-box Optimisation Problem (BBOP)

[¶] Assume to maximise the function.

Optimisation Problem

- **Aim:** To find the best solution from all feasible solutions.
- **Mathematical description:**
  - **Given:** a function $f : \mathcal{S} \to \mathbb{R}$
  - **Sought:** find an element $\hat{x} \in \mathcal{S}$ such that $f(\hat{x}) \geq f(x)$ for any $x \in \mathcal{S}$[¶]
- The *search space* $\mathcal{S}$ can be continuous, discrete, or mixed.
- The *objective (fitness) function* f is usually unknown for algorithms.
- The "property" of f is sometime unknown for algorithms.

$\Downarrow$

Black-box Optimisation Problem (BBOP)

[¶] Assume to maximise the function.

Optimisation Problem

- **Aim:** To find the best solution from all feasible solutions.
- **Mathematical description:**
  - **Given:** a function $f : \mathcal{S} \to \mathbb{R}$
  - **Sought:** find an element $\hat{x} \in \mathcal{S}$ such that $f(\hat{x}) \geq f(x)$ for any $x \in \mathcal{S}$[¶]
- The *search space* $\mathcal{S}$ can be continuous, discrete, or mixed.
- The *objective (fitness) function* f is usually unknown for algorithms.
- The "property" of f is sometime unknown for algorithms.

$\Downarrow$

Black-box Optimisation Problem (BBOP)

[¶] Assume to maximise the function.

Optimisation Problem

- **Aim:** To find the best solution from all feasible solutions.
- **Mathematical description:**
  - **Given:** a function $f : \mathcal{S} \to \mathbb{R}$
  - **Sought:** find an element $\hat{x} \in \mathcal{S}$ such that $f(\hat{x}) \geq f(x)$ for any $x \in \mathcal{S}$[¶]
- The *search space* $\mathcal{S}$ can be continuous, discrete, or mixed.
- The *objective (fitness) function* f is usually unknown for algorithms.
- The "property" of f is sometime unknown for algorithms.

$\Downarrow$

Black-box Optimisation Problem (BBOP)

[¶] Assume to maximise the function.

Black-box Optimisation Problem

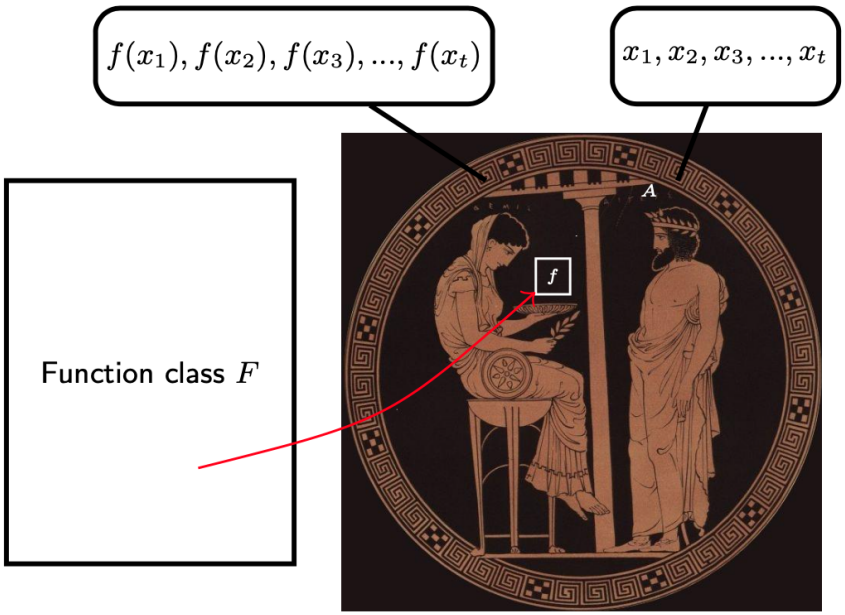


Photo: E. Gerhard (1846).

- Optimise function f by ONLY evaluating the fitness value of search points.
- **Formal description:**
  - An unknown optimisation problem f is chosen from a problem class $\mathcal{F} \subseteq \{f : S \to \mathbb{R}\}$ known to the algorithm A.
  - For every $t \in \mathbb{N}$, using the obtained information $(x_1, f(x_1)), \cdots, (x_t, f(x_t))$, the algorithm queries a new search point x_{t+1} to obtain $f(x_{t+1})$ from the *oracle*.

Runtime*: $T_{A,f} := \min\{t \mid \forall y \in \{0,1\}^n,\ f(x_t) \geq f(y)\}$

$\Downarrow$

The number of evalutions until the optimal (desired) solution found.

*Reference: (Droste et al., 2006). (Assume that the objective is maximising f)

Black-box Optimisation Problem

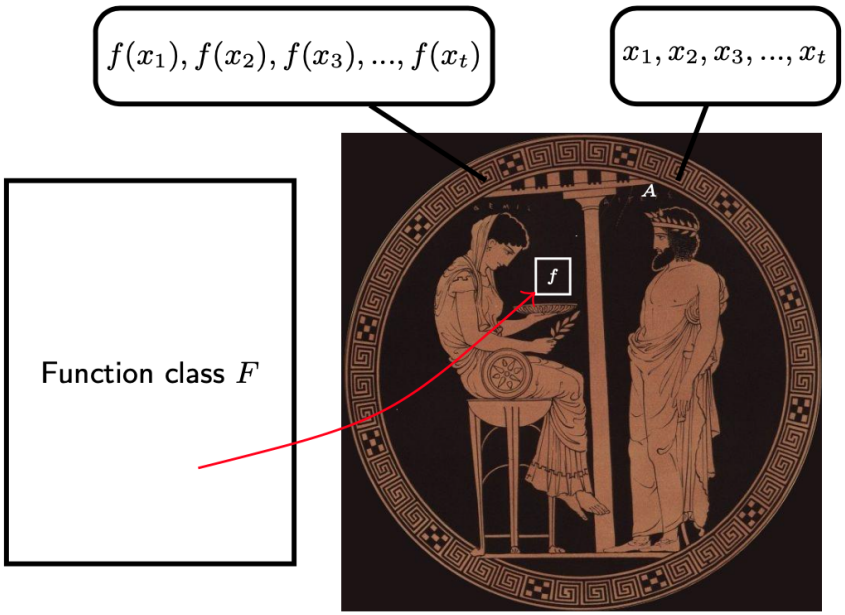


Photo: E. Gerhard (1846).

- Optimise function f by ONLY evaluating the fitness value of search points.
- **Formal description:**
  - An unknown optimisation problem f is chosen from a problem class $\mathcal{F} \subseteq \{f : S \to \mathbb{R}\}$ known to the algorithm A.
  - For every $t \in \mathbb{N}$, using the obtained information $(x_1, f(x_1)), \cdots, (x_t, f(x_t))$, the algorithm queries a new search point x_{t+1} to obtain $f(x_{t+1})$ from the *oracle*.

Runtime*: $T_{A,f} := \min\{t \mid \forall y \in \{0,1\}^n, f(x_t) \geq f(y)\}$

⇓

The number of evalutions until the optimal (desired) solution found.

*Reference: (Droste et al., 2006). (Assume that the objective is maximising f)

Black-box Optimisation Problem

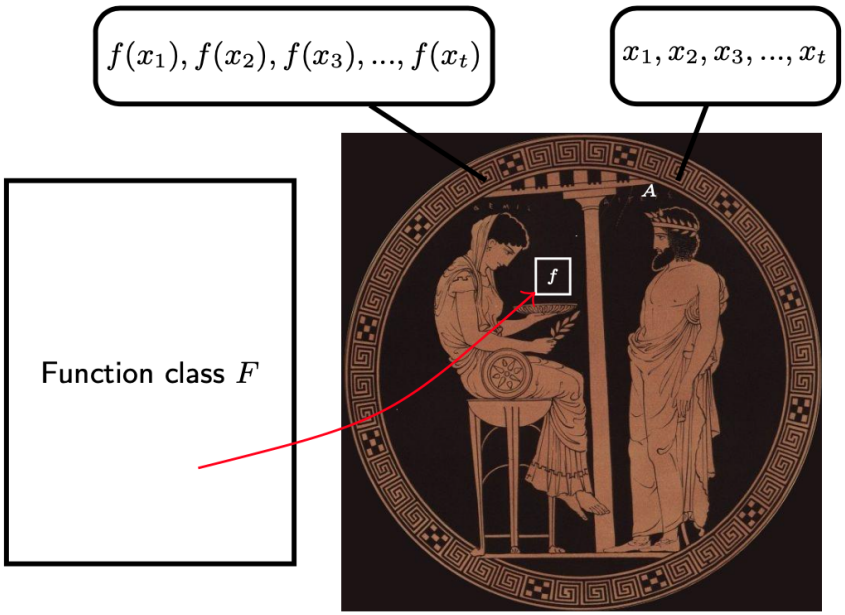


Photo: E. Gerhard (1846).

- Optimise function f by ONLY evaluating the fitness value of search points.
- **Formal description:**
  - An unknown optimisation problem f is chosen from a problem class $\mathcal{F} \subseteq \{f : \mathcal{S} \to \mathbb{R}\}$ known to the algorithm A.
  - For every $t \in \mathbb{N}$, using the obtained information $(x_1, f(x_1)), \cdots, (x_t, f(x_t))$, the algorithm queries a new search point x_{t+1} to obtain $f(x_{t+1})$ from the *oracle*.

Runtime*: $T_{A,f} := \min\{t \mid \forall y \in \{0,1\}^n,\ f(x_t) \geq f(y)\}$

$\Downarrow$

The number of evalutions until the optimal (desired) solution found.

*Reference: (Droste et al., 2006). (Assume that the objective is maximising f)

Black-box Optimisation Problem

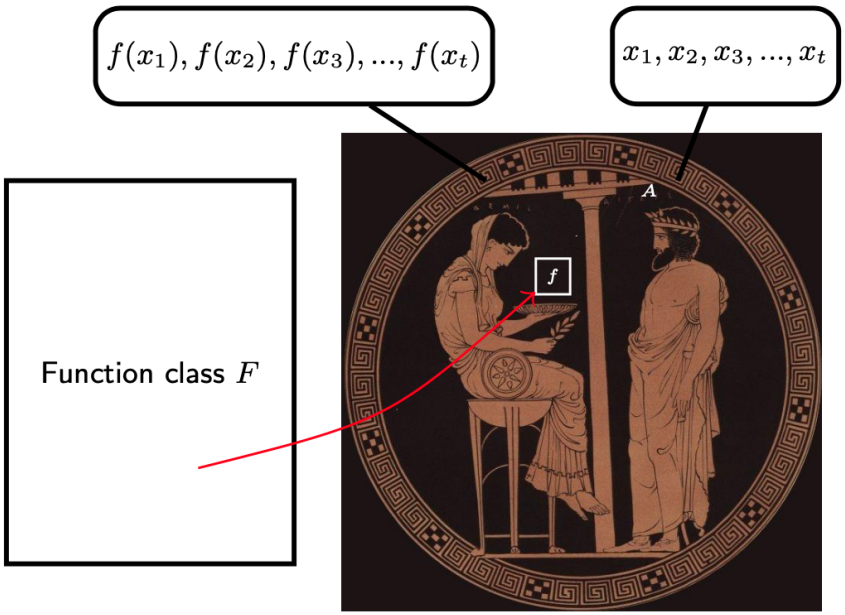


Photo: E. Gerhard (1846).

- Optimise function f by ONLY evaluating the fitness value of search points.
- **Formal description:**
  - An unknown optimisation problem f is chosen from a problem class $\mathcal{F} \subseteq \{f : \mathcal{S} \to \mathbb{R}\}$ known to the algorithm A.
  - For every $t \in \mathbb{N}$, using the obtained information $(x_1, f(x_1)), \cdots, (x_t, f(x_t))$, the algorithm queries a new search point x_{t+1} to obtain $f(x_{t+1})$ from the *oracle*.

Runtime*: $T_{A,f} := \min\{t \mid \forall y \in \{0,1\}^n, f(x_t) \geq f(y)\}$

⇓

The number of evalutions until the optimal (desired) solution found.

*Reference: (Droste et al., 2006). (Assume that the objective is maximising f)

Black-box Optimisation Problem

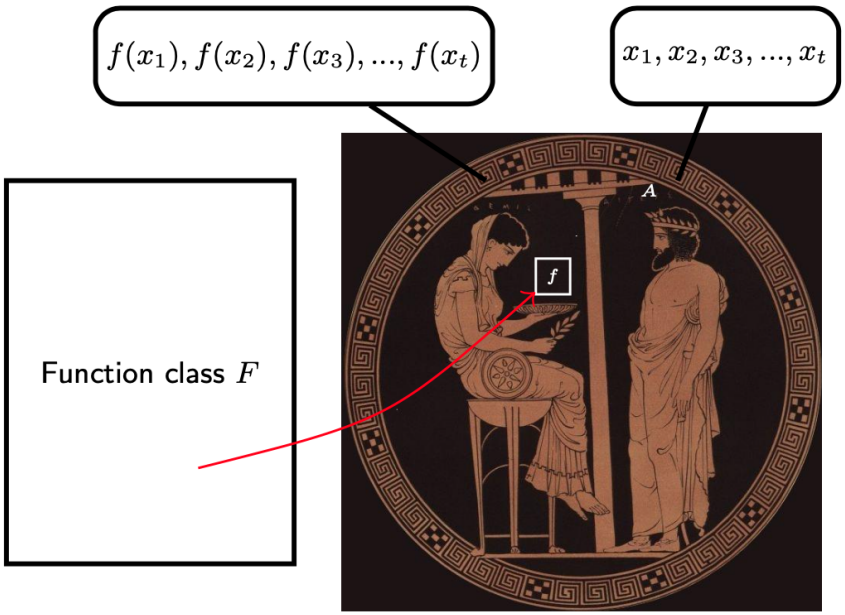


Photo: E. Gerhard (1846).

- Optimise function f by ONLY evaluating the fitness value of search points.
- **Formal description:**
  - An unknown optimisation problem f is chosen from a problem class $\mathcal{F} \subseteq \{f : \mathcal{S} \to \mathbb{R}\}$ known to the algorithm A.
  - For every $t \in \mathbb{N}$, using the obtained information $(x_1, f(x_1)), \cdots, (x_t, f(x_t))$, the algorithm queries a new search point x_{t+1} to obtain $f(x_{t+1})$ from the *oracle*.

$$\textbf{Runtime}^*\textbf{:}\ T_{A,f} := \min\{t \mid \forall y \in \{0,1\}^n,\ f(x_t) \geq f(y)\}$$

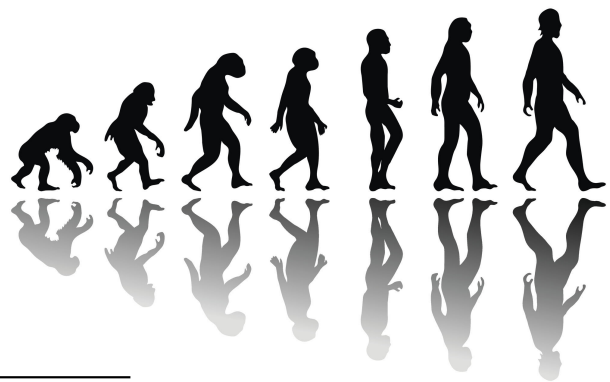
The number of evalutions until the optimal (desired) solution found.

*Reference: (Droste et al., 2006). (Assume that the objective is maximising f)

Evolutionary Computation (EC)

Evolutionary Computation is a family of *randomised heuristic algorithms* for optimisation inspired by biological evolution. *

- **Efficiency** $\Rightarrow$ Heuristic $\Rightarrow$ Achieve the (nearly) optimum in short time
  $\Rightarrow$ Generality $\Rightarrow$ Independent of problem 'property' (BBOP)
  $\Rightarrow$ Parallelism $\Rightarrow$ Population-based
- **Practicability** $\Rightarrow$ Multi-objectivisation
  $\Rightarrow$ Global optimisation $\Rightarrow$ Can escaping local optima
- **Robustness** $\Rightarrow$ Noisy/Dynamic Optimisation



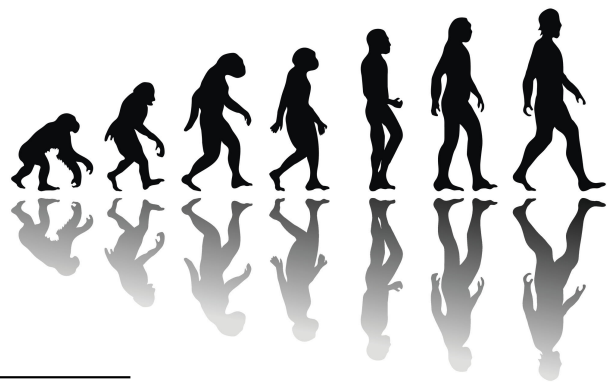
*The generalised EC also includes other natural-inspired algorithms, e.g., Particle Swarm Optimisation (PSO), Ant Colony Optimisation (ACO), Differential Evolution (DE), Estimation of Distribution Algorithms (EDAs), Simulated Annealing (SA), etc.

† Image source: https://www.baptistpress.com/resource-library/news/fossil-find-complicates-theory-of-human-evolution/

Evolutionary Computation (EC)

Evolutionary Computation is a family of *randomised heuristic algorithms* for optimisation inspired by biological evolution. *

- **Efficiency** ⇒ Heuristic ⇒ Achieve the (nearly) optimum in short time
  ⇒ Generality ⇒ Independent of problem 'property' (BBOP)
  ⇒ Parallelism ⇒ Population-based
- **Practicability** ⇒ Multi-objectivisation
  ⇒ Global optimisation ⇒ Can escaping local optima
- **Robustness** ⇒ Noisy/Dynamic Optimisation



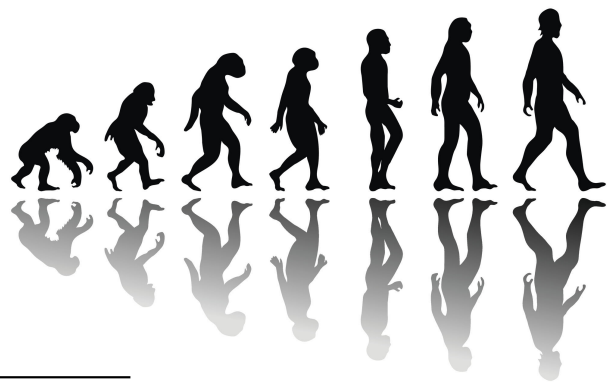
*The generalised EC also includes other natural-inspired algorithms, e.g., Particle Swarm Optimisation (PSO), Ant Colony Optimisation (ACO), Differential Evolution (DE), Estimation of Distribution Algorithms (EDAs), Simulated Annealing (SA), etc.

† Image source: https://www.baptistpress.com/resource-library/news/fossil-find-complicates-theory-of-human-evolution/

Evolutionary Computation (EC)

Evolutionary Computation is a family of *randomised heuristic algorithms* for optimisation inspired by biological evolution. *

- **Efficiency** ⇒ Heuristic ⇒ Achieve the (nearly) optimum in short time
  ⇒ Generality ⇒ Independent of problem 'property' (BBOP)
  ⇒ Parallelism ⇒ Population-based
- **Practicability** ⇒ Multi-objectivisation
  ⇒ Global optimisation ⇒ Can escaping local optima
- **Robustness** ⇒ Noisy/Dynamic Optimisation



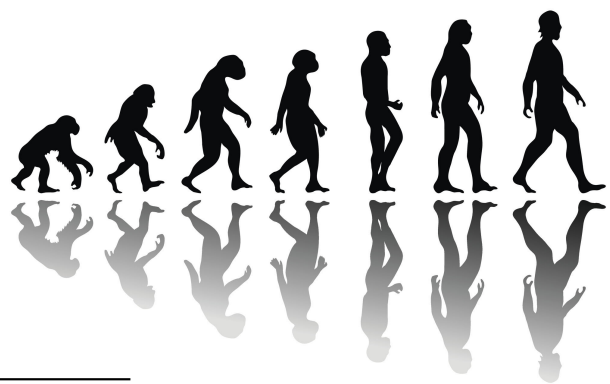
*The generalised EC also includes other natural-inspired algorithms, e.g., Particle Swarm Optimisation (PSO), Ant Colony Optimisation (ACO), Differential Evolution (DE), Estimation of Distribution Algorithms (EDAs), Simulated Annealing (SA), etc.

† Image source: https://www.baptistpress.com/resource-library/news/fossil-find-complicates-theory-of-human-evolution/

Evolutionary Computation (EC)

Evolutionary Computation is a family of *randomised heuristic algorithms* for optimisation inspired by biological evolution. *

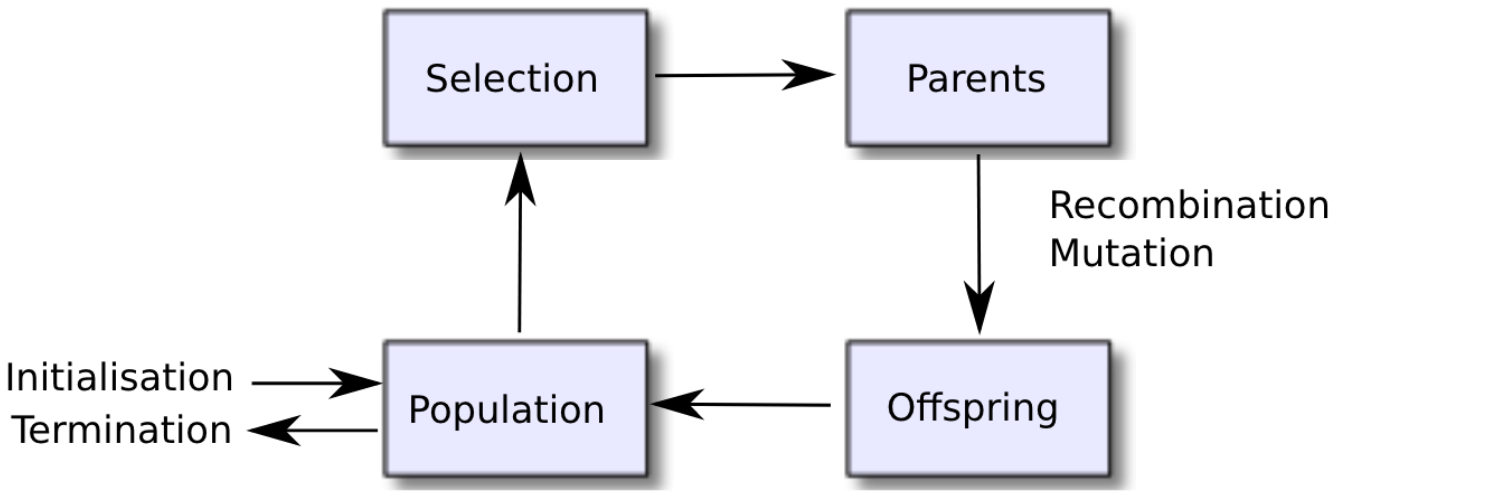
- **Efficiency** $\Rightarrow$ Heuristic $\Rightarrow$ Achieve the (nearly) optimum in short time
  $\Rightarrow$ Generality $\Rightarrow$ Independent of problem 'property' (BBOP)
  $\Rightarrow$ Parallelism $\Rightarrow$ Population-based
- **Practicability** $\Rightarrow$ Multi-objectivisation
  $\Rightarrow$ Global optimisation $\Rightarrow$ Can escaping local optima
- **Robustness** $\Rightarrow$ Noisy/Dynamic Optimisation



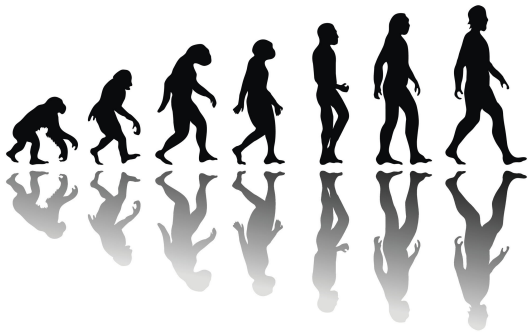
*The generalised EC also includes other natural-inspired algorithms, e.g., Particle Swarm Optimisation (PSO), Ant Colony Optimisation (ACO), Differential Evolution (DE), Estimation of Distribution Algorithms (EDAs), Simulated Annealing (SA), etc.

† Image source: https://www.baptistpress.com/resource-library/news/fossil-find-complicates-theory-of-human-evolution/

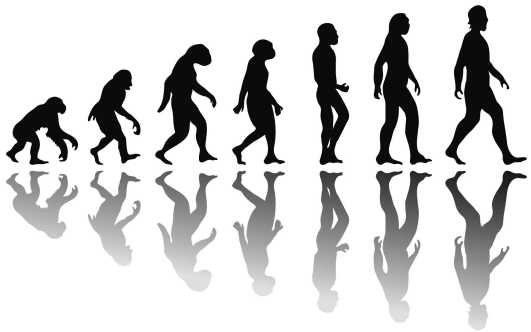
From Natural Evolution to Evolutionary Algorithms (EAs)

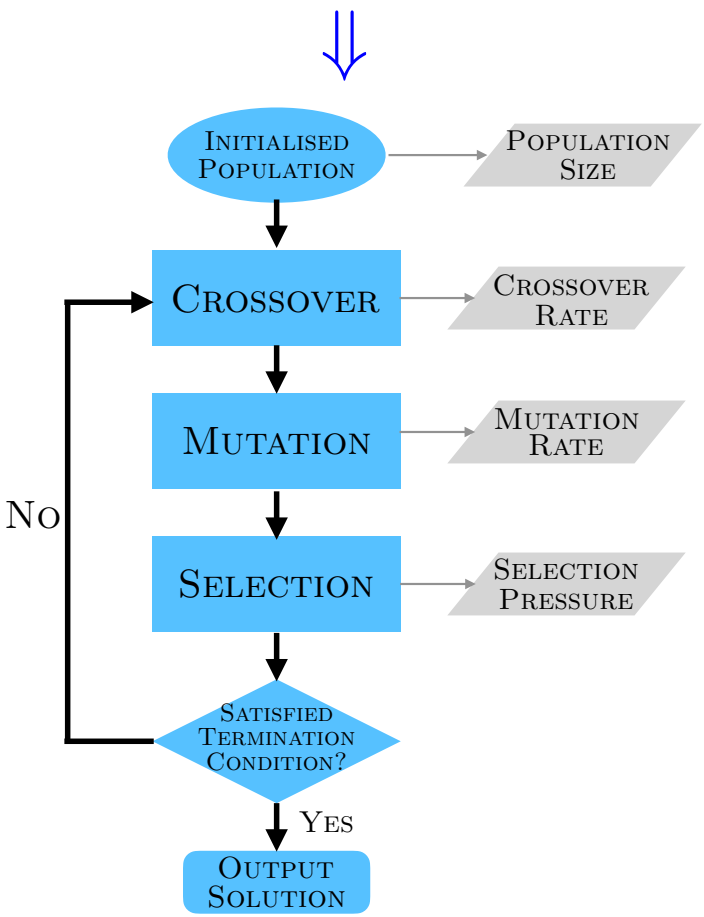


From Natural Evolution to Evolutionary Algorithms (EAs)

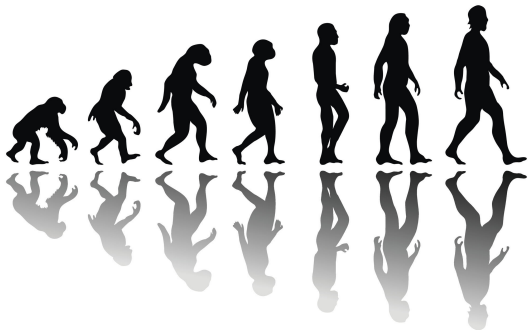


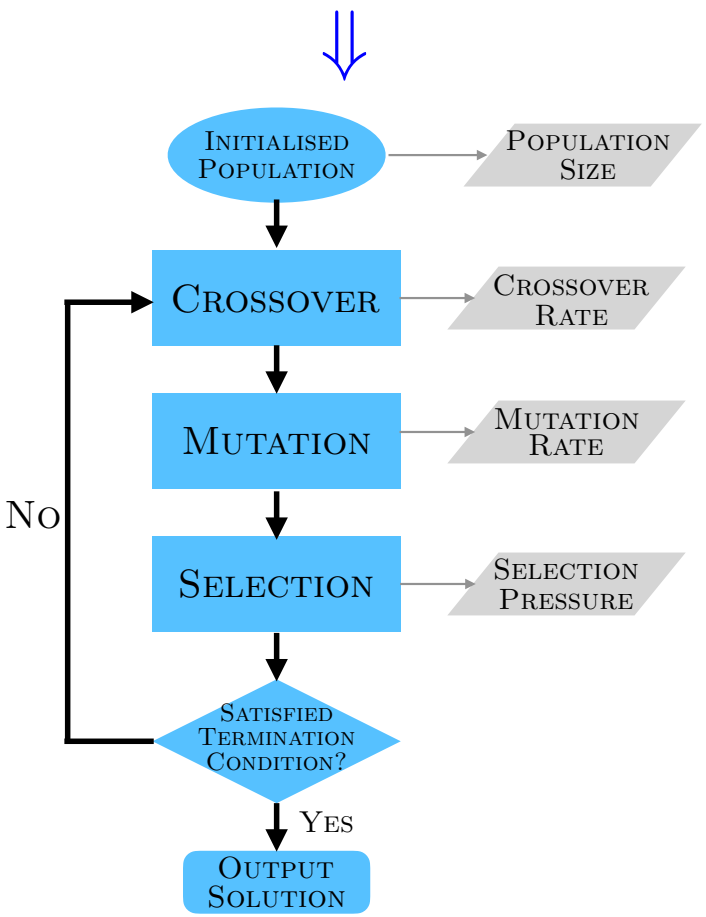
From Natural Evolution to Evolutionary Algorithms (EAs)





From Natural Evolution to Evolutionary Algorithms (EAs)



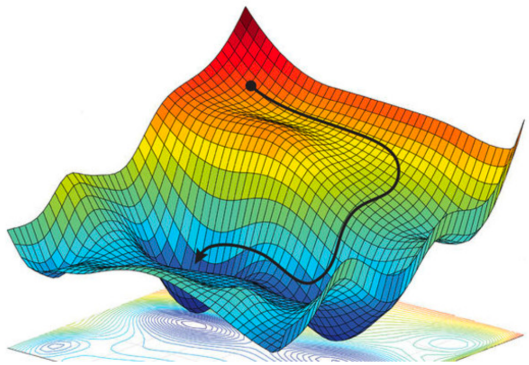


Classification of EAs:

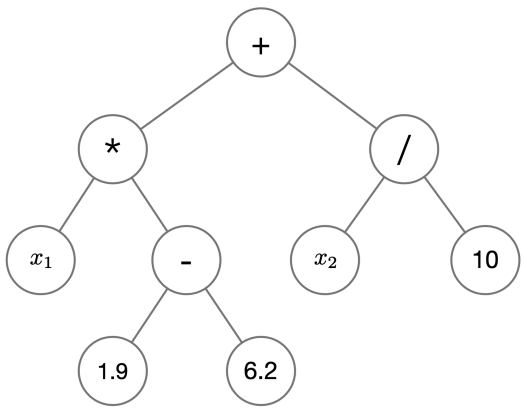
- **Genetic Algorithms (GAs)** $\Rightarrow$ **Discrete/Binary**

$$11000010 \cdots 010001$$

- **Evolutionary Strategy (ESs)** $\Rightarrow$ Continuous



- **Genetic Programming (GPs)** $\Rightarrow$ Tree



A Simple EA for Binary Search Space

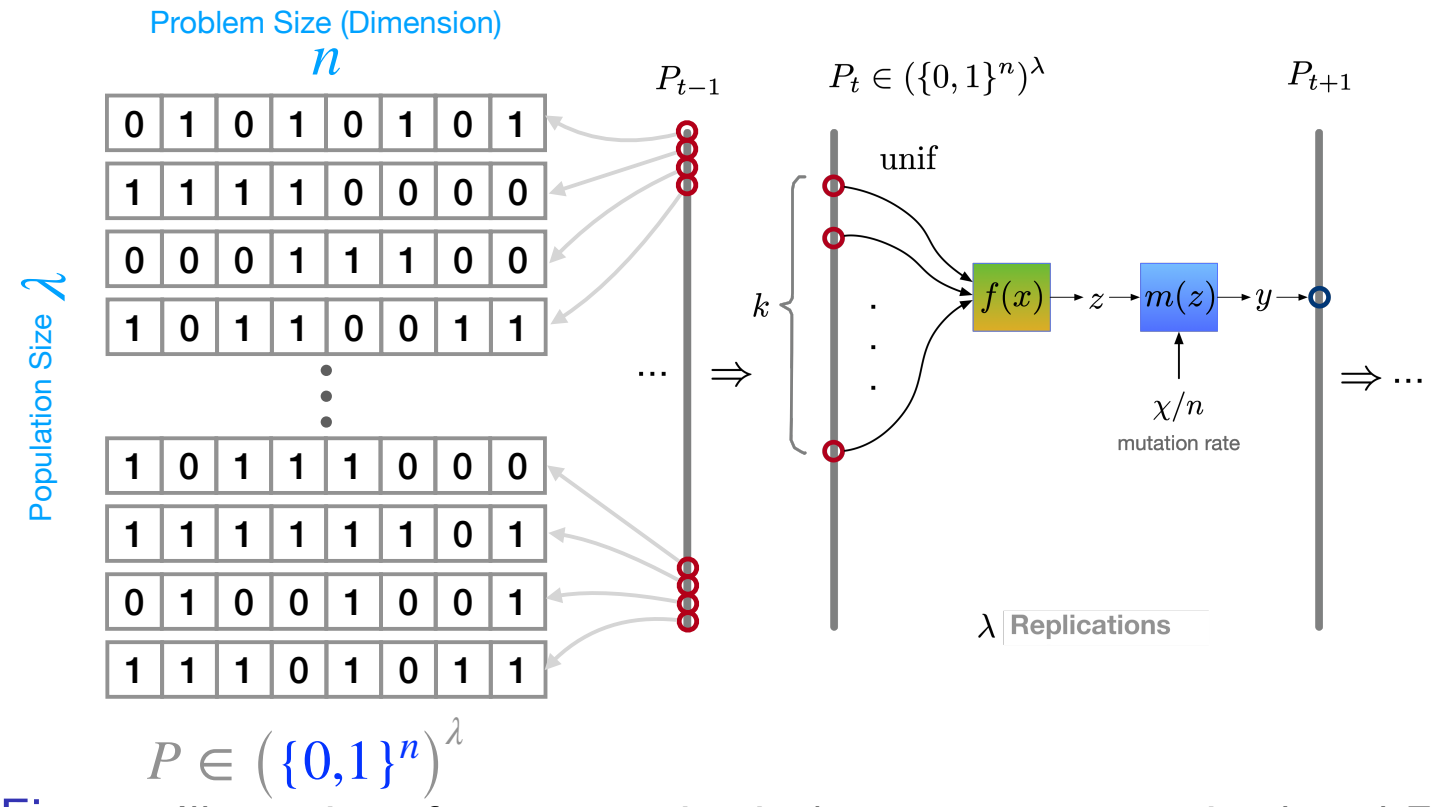


Figure: Illustration of one generation in k-tournament mutation-based EA

- **Selection Mechanism:** k-tournament (*here*), (μ,λ), Power-law, etc.
- **Mutation Operator:** $m : \{0,1\}^n \to \{0,1\}^n$
  Bit-wisely flip a solution z with probability χ/n, so-called **Mutation Rate**.

A Simple EA for Binary Search Space

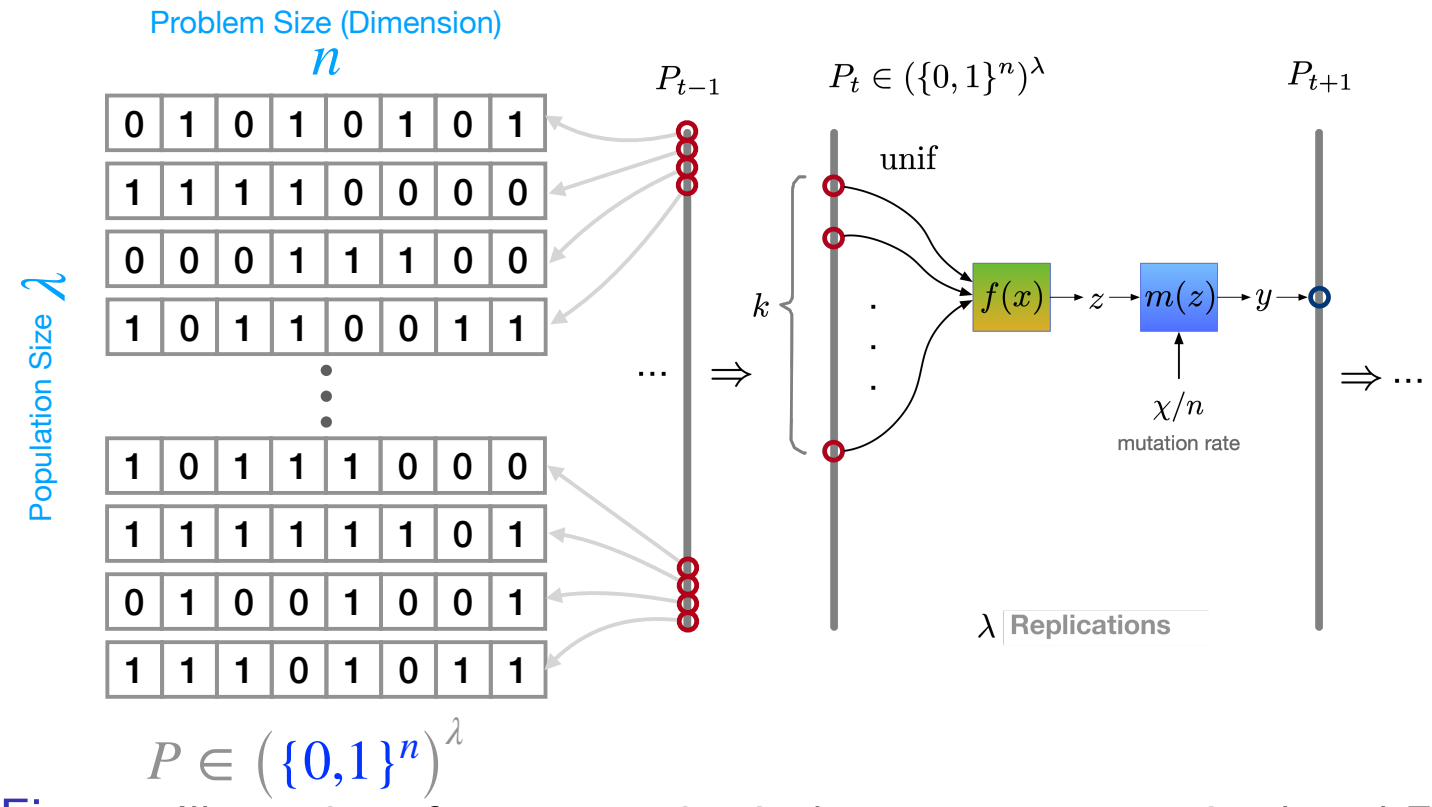


Figure: Illustration of one generation in k-tournament mutation-based EA

- **Selection Mechanism:** k-tournament (*here*), (μ,λ), Power-law, etc.
- **Mutation Operator:** $m : \{0,1\}^n \to \{0,1\}^n$
  Bit-wisely flip a solution z with probability χ/n, so-called **Mutation Rate**.

A Simple EA for Binary Search Space

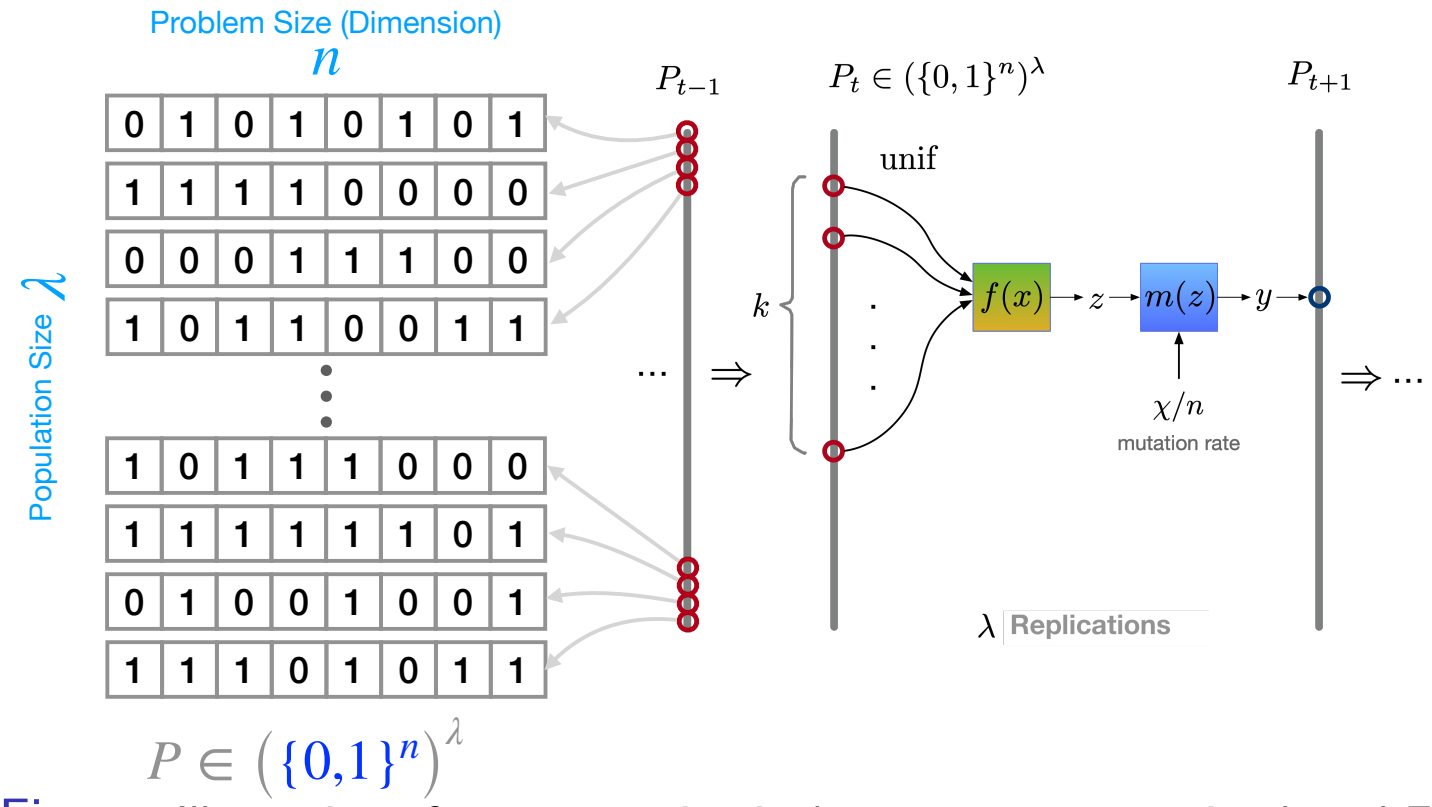


Figure: Illustration of one generation in k-tournament mutation-based EA

- **Selection Mechanism:** k-tournament (*here*), (μ,λ), Power-law, etc.
- **Mutation Operator:** $m : \{0,1\}^n \to \{0,1\}^n$
  Bit-wisely flip a solution z with probability χ/n, so-called **Mutation Rate**.

Methodology in EC

Methodology in EC

Theoretical Study:

Corollary 3.1. *The expected runtime of the 2-tournament EA using mutation rate $\chi = a/n$ for any constant $a > 0$ on LeadingOnes is $\Omega(n^2 \log(n))$.*

Theorem 5.2. *For any constant $\chi_{\text{high}} > 0$, $\chi_{\text{low}} = a/n$ where $a > 0$ is any constant, any constant $q \in [0, 1/2)$ and $p_c \in o(1) \cap \Omega(1/n)$, the expected runtime of the 2-tournament EA using SA-2mr from $\{\chi_{\text{high}}/n, \chi_{\text{low}}/n\}$ with adaptation parameter p_c and population size $\lambda > c \log(n)$ for a sufficiently large constant c on optimising LeadingOnes in the symmetric noise model (C, q) is $O\left(n\lambda \log(n) + n^3\right)$.*

Theorem 2. *The expected runtime of the $(\mu + \lambda)$ EA with $\lambda, \mu \in \text{poly}(n)$, $\lambda/\mu \geq 1$, initial population $P_0 = \{0^k *^{n-k}\}^\mu$, and mutation parameter $\chi \in O(1)$ on PeakedLO$_{m,k}$ with any $k \leq n$ and $k, m \in \Omega(n)$ satisfies $\Pr(T \leq e^{cn^d}) \leq e^{-\Omega(n)}$ for some constants $c, d > 0$.*

Theorem 9. *For any constant $\zeta \in (0, 1)$ and any $\chi \in (0, \ln(1 + 2\theta\zeta))$, where $\theta := 1/(2n)$, Algorithm 1 with mutation rate χ/n and population size $\lambda > cn^2 \log(n)$ for a large enough constant c achieves the optimum on DynBV in expected time $O\left(n^3\lambda \log(1/\chi) + n^3 \log(n)/\chi\right)$.*

Theorem 10. *For any constants $\alpha, \varepsilon, \sigma, c$ in $(0, 1)$ and $\chi \geq 1$ such that the following equation holds: $\varepsilon = \sigma\left(c(1 + \alpha\chi) - 1\right) e^{-\chi}$ then Algorithm 1 using the power-law ranking selection mechanism with parameter c, bitwise mutation with parameter χ and population size $\text{poly}(n) \ni \lambda \geq d \ln n$ for some sufficiently large constant $d > 0$ optimises SparseLocalOpt$_{\alpha, \frac{\varepsilon}{1+\varepsilon}}$ in expected polynomial time.*

Methodology in EC

Theoretical Study:

Corollary 3.1. *The expected runtime of the 2-tournament EA using mutation rate $\chi = a/n$ for any constant $a > 0$ on LeadingOnes is $\Omega(n^2 \log(n))$.*

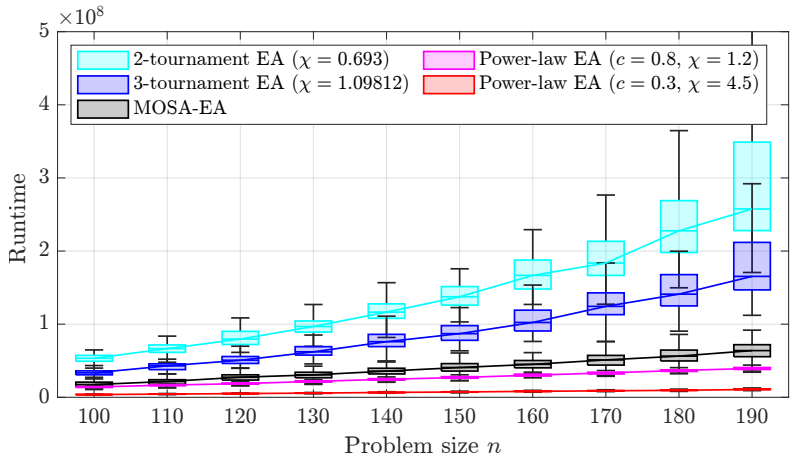
Theorem 5.2. *For any constant $\chi_{\text{high}} > 0$, $\chi_{\text{low}} = a/n$ where $a > 0$ is any constant, any constant $q \in [0, 1/2)$ and $p_c \in o(1) \cap \Omega(1/n)$, the expected runtime of the 2-tournament EA using SA-2mr from $\{\chi_{\text{high}}/n, \chi_{\text{low}}/n\}$ with adaptation parameter p_c and population size $\lambda > c \log(n)$ for a sufficiently large constant c on optimising LeadingOnes in the symmetric noise model (C, q) is $O\left(n\lambda \log(n) + n^3\right)$.*

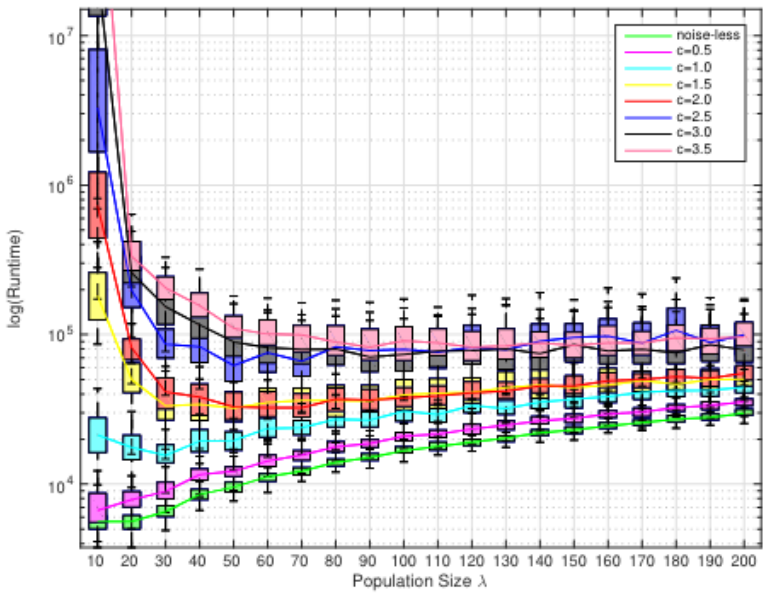
Theorem 2. *The expected runtime of the $(\mu + \lambda)$ EA with $\lambda, \mu \in \text{poly}(n)$, $\lambda/\mu \geq 1$, initial population $P_0 = \{0^k *^{n-k}\}^\mu$, and mutation parameter $\chi \in O(1)$ on PeakedLO$_{m,k}$ with any $k \leq n$ and $k, m \in \Omega(n)$ satisfies $\Pr(T \leq e^{cn^d}) \leq e^{-\Omega(n)}$ for some constants $c, d > 0$.*

Theorem 9. *For any constant $\zeta \in (0, 1)$ and any $\chi \in (0, \ln(1 + 2\theta\zeta))$, where $\theta := 1/(2n)$, Algorithm 1 with mutation rate χ/n and population size $\lambda > cn^2 \log(n)$ for a large enough constant c achieves the optimum on DynBV in expected time $O\left(n^3\lambda \log(1/\chi) + n^3 \log(n)/\chi\right)$.*

Theorem 10. *For any constants $\alpha, \varepsilon, \sigma, c$ in $(0, 1)$ and $\chi \geq 1$ such that the following equation holds: $\varepsilon = \sigma\left(c(1 + \alpha\chi) - 1\right) e^{-\chi}$ then Algorithm 1 using the power-law ranking selection mechanism with parameter c, bitwise mutation with parameter χ and population size $\text{poly}(n) \ni \lambda \geq d \ln n$ for some sufficiently large constant $d > 0$ optimises SparseLocalOpt$_{\alpha, \frac{\varepsilon}{1+\varepsilon}}$ in expected polynomial time.*

Empirical Study:





Methodology in EC

Theoretical Study:

Corollary 3.1. *The expected runtime of the 2-tournament EA using mutation rate $\chi = a/n$ for any constant $a > 0$ on LeadingOnes is $\Omega(n^2 \log(n))$.*

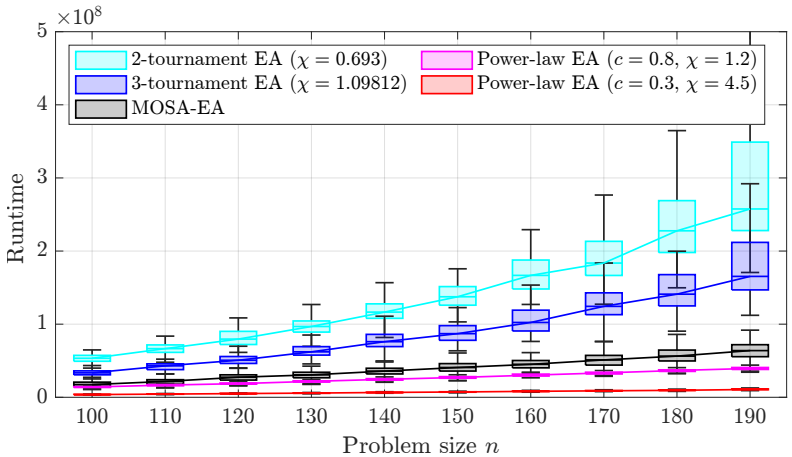
Theorem 5.2. *For any constant $\chi_{\text{high}} > 0$, $\chi_{\text{low}} = a/n$ where $a > 0$ is any constant, any constant $q \in [0, 1/2)$ and $p_c \in o(1) \cap \Omega(1/n)$, the expected runtime of the 2-tournament EA using SA-2mr from $\{\chi_{\text{high}}/n, \chi_{\text{low}}/n\}$ with adaptation parameter p_c and population size $\lambda > c \log(n)$ for a sufficiently large constant c on optimising LeadingOnes in the symmetric noise model (C, q) is $O\left(n\lambda \log(n) + n^3\right)$.*

Theorem 2. *The expected runtime of the $(\mu + \lambda)$ EA with $\lambda, \mu \in \text{poly}(n)$, $\lambda/\mu \geq 1$, initial population $P_0 = \{0^k *^{n-k}\}^\mu$, and mutation parameter $\chi \in O(1)$ on PeakedLO$_{m,k}$ with any $k \leq n$ and $k, m \in \Omega(n)$ satisfies $\Pr(T \leq e^{cn^d}) \leq e^{-\Omega(n)}$ for some constants $c, d > 0$.*

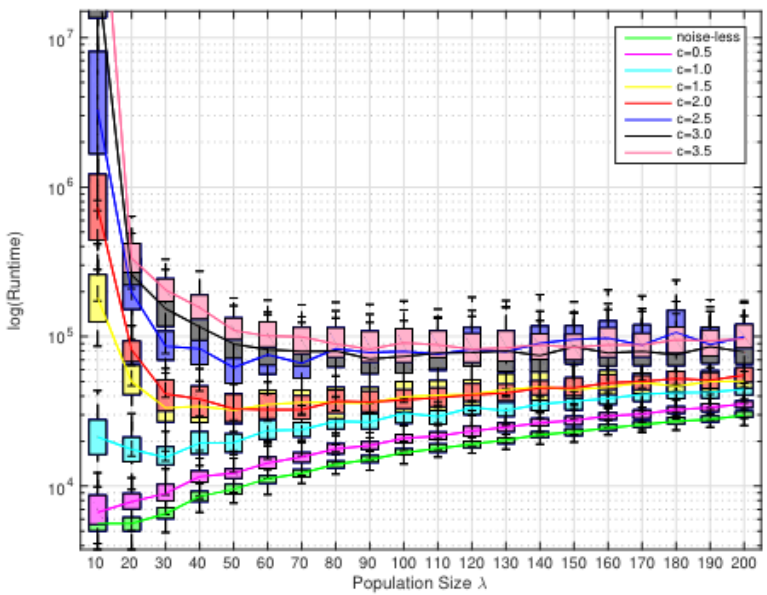
Theorem 9. *For any constant $\zeta \in (0, 1)$ and any $\chi \in (0, \ln(1 + 2\theta\zeta))$, where $\theta := 1/(2n)$, Algorithm 1 with mutation rate χ/n and population size $\lambda > cn^2 \log(n)$ for a large enough constant c achieves the optimum on DynBV in expected time $O\left(n^3\lambda \log(1/\chi) + n^3 \log(n)/\chi\right)$.*

Theorem 10. *For any constants $\alpha, \varepsilon, \sigma, c$ in $(0, 1)$ and $\chi \geq 1$ such that the following equation holds: $\varepsilon = \sigma\left(c(1 + \alpha\chi) - 1\right) e^{-\chi}$ then Algorithm 1 using the power-law ranking selection mechanism with parameter c, bitwise mutation with parameter χ and population size $\text{poly}(n) \ni \lambda \geq d \ln n$ for some sufficiently large constant $d > 0$ optimises SparseLocalOpt$_{\alpha, \frac{\varepsilon}{1+\varepsilon}}$ in expected polynomial time.*

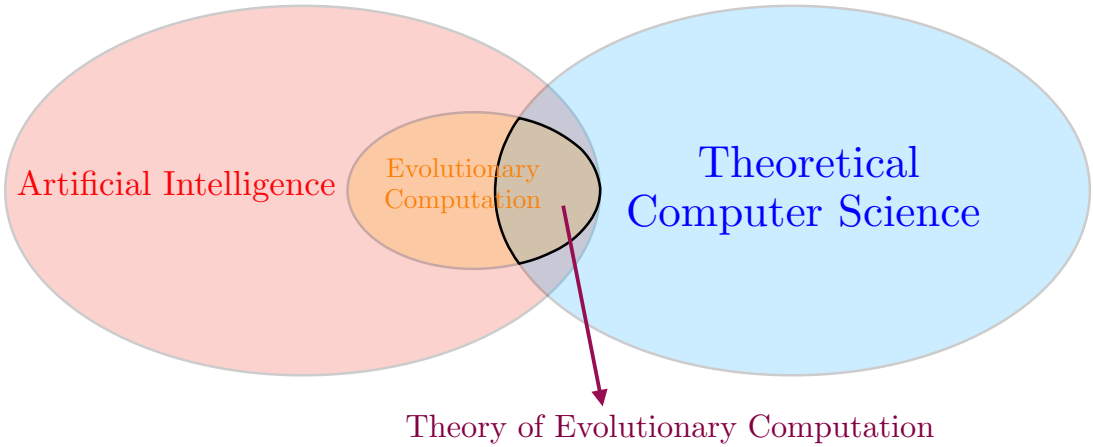
Empirical Study:



VS



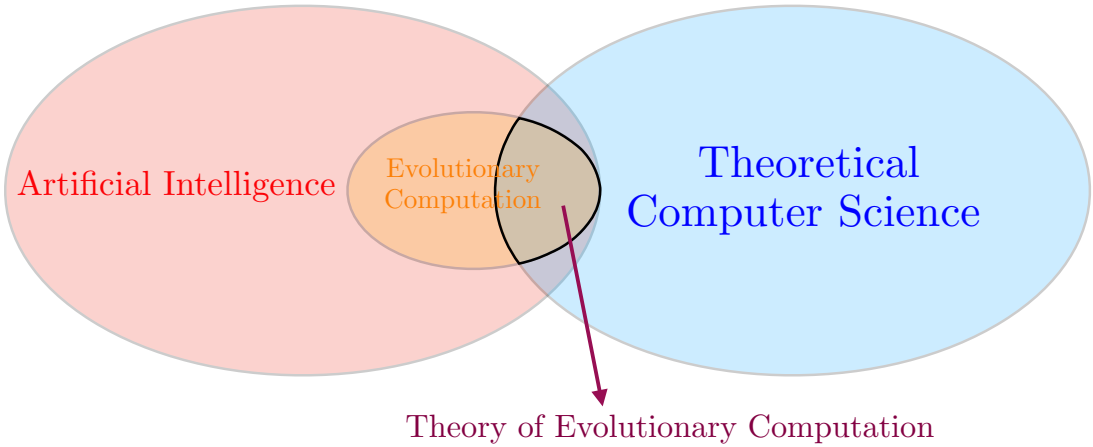
Theory of EC



- **Definition:** Results proven with **mathematical rigour** (Doerr, 2020)
- **Aim:** Provide a solid theoretical **foundation** and **guide** for EC:
  Explain and predict the performance $\Rightarrow$ **Explainable AI** $\Rightarrow$ **Trustworthy AI**
- **Tasks:** Runtime Analysis, Fitness Landscape Analysis, etc.
- **Conferences:** Theory Track in GECCO (TH-CPL B; CCF C; Core A)
  PPSN (CCF B; Core A)
  FOGA (Core A*)
- **Journals:** Algorithmica (TH-CPL B; CCF B; SCI 4)
  Theoretical Computer Science (TCS) (TH-CPL B; CCF B; SCI 4)
  IEEE Trans' on Evol' Comp' (TEVC) (TH-CPL B; CCF B; SCI Top1)
  Evolutionary Computation (ECJ) (TH-CPL B; CCF B; SCI 3)

After about 30 years of development,
the theory of EC has evolved into an **independent** and **important** sub-field in EC.

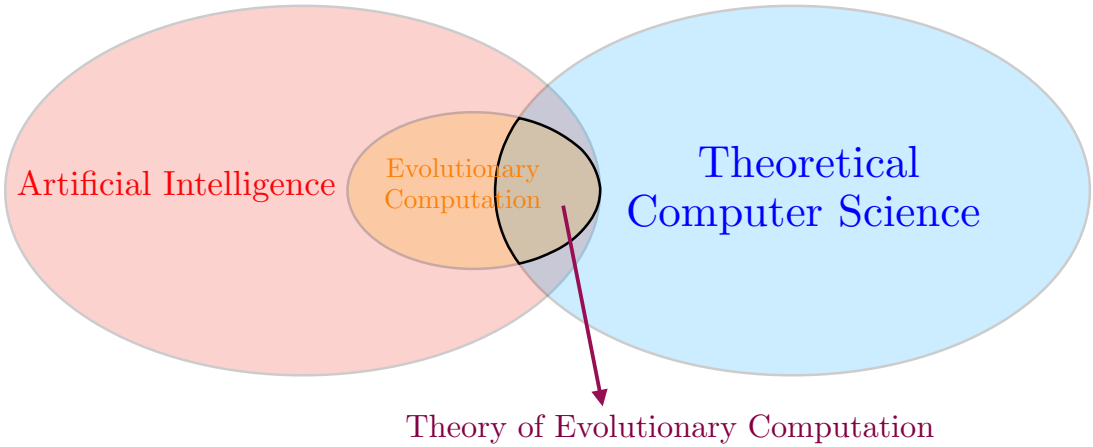
Theory of EC



- **Definition:** Results proven with **mathematical rigour** (Doerr, 2020)
- **Aim:** Provide a solid theoretical **foundation** and **guide** for EC:
  Explain and predict the performance $\Rightarrow$ **Explainable AI** $\Rightarrow$ **Trustworthy AI**
- **Tasks:** Runtime Analysis, Fitness Landscape Analysis, etc.
- **Conferences:** Theory Track in GECCO (TH-CPL B; CCF C; Core A)
  PPSN (CCF B; Core A)
  FOGA (Core A*)
- **Journals:** Algorithmica (TH-CPL B; CCF B; SCI 4)
  Theoretical Computer Science (TCS) (TH-CPL B; CCF B; SCI 4)
  IEEE Trans' on Evol' Comp' (TEVC) (TH-CPL B; CCF B; SCI Top1)
  Evolutionary Computation (ECJ) (TH-CPL B; CCF B; SCI 3)

After about 30 years of development,
the theory of EC has evolved into an **independent** and **important** sub-field in EC.

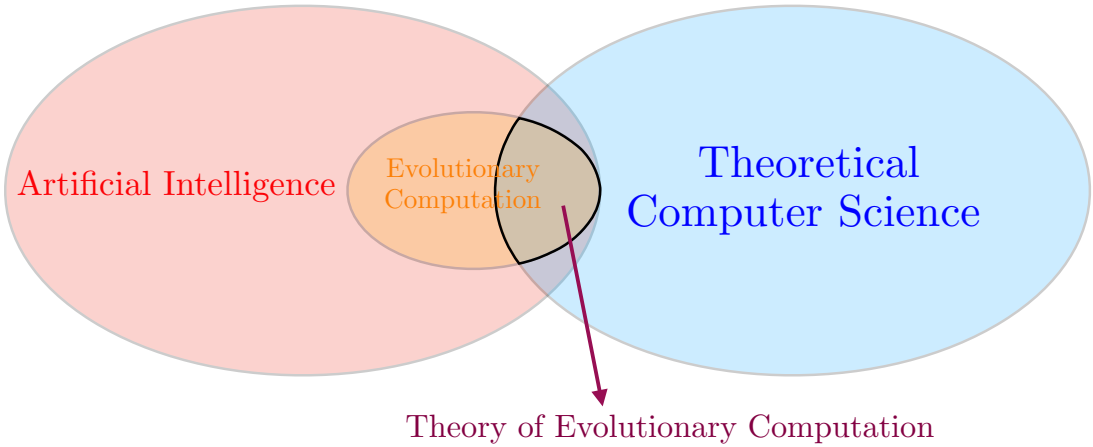
Theory of EC



- **Definition:** Results proven with **mathematical rigour** (Doerr, 2020)
- **Aim:** Provide a solid theoretical **foundation** and **guide** for EC:
  Explain and predict the performance $\Rightarrow$ **Explainable AI** $\Rightarrow$ **Trustworthy AI**
- **Tasks:** Runtime Analysis, Fitness Landscape Analysis, etc.
- **Conferences:** Theory Track in GECCO (TH-CPL B; CCF C; Core A)
  PPSN (CCF B; Core A)
  FOGA (Core A*)
- **Journals:** Algorithmica (TH-CPL B; CCF B; SCI 4)
  Theoretical Computer Science (TCS) (TH-CPL B; CCF B; SCI 4)
  IEEE Trans' on Evol' Comp' (TEVC) (TH-CPL B; CCF B; SCI Top1)
  Evolutionary Computation (ECJ) (TH-CPL B; CCF B; SCI 3)

After about 30 years of development,
the theory of EC has evolved into an **independent** and **important** sub-field in EC.

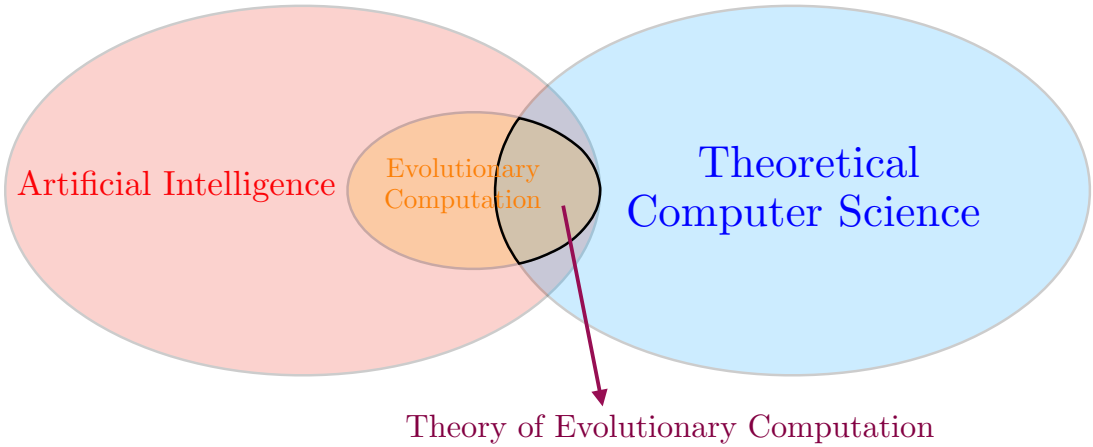
Theory of EC



- **Definition:** Results proven with **mathematical rigour** (Doerr, 2020)
- **Aim:** Provide a solid theoretical **foundation** and **guide** for EC:
  Explain and predict the performance $\Rightarrow$ **Explainable AI** $\Rightarrow$ **Trustworthy AI**
- **Tasks:** Runtime Analysis, Fitness Landscape Analysis, etc.
- **Conferences:** Theory Track in GECCO (TH-CPL B; CCF C; Core A)
  PPSN (CCF B; Core A)
  FOGA (Core A*)
- **Journals:** Algorithmica (TH-CPL B; CCF B; SCI 4)
  Theoretical Computer Science (TCS) (TH-CPL B; CCF B; SCI 4)
  IEEE Trans' on Evol' Comp' (TEVC) (TH-CPL B; CCF B; SCI Top1)
  Evolutionary Computation (ECJ) (TH-CPL B; CCF B; SCI 3)

After about 30 years of development,
the theory of EC has evolved into an **independent** and **important** sub-field in EC.

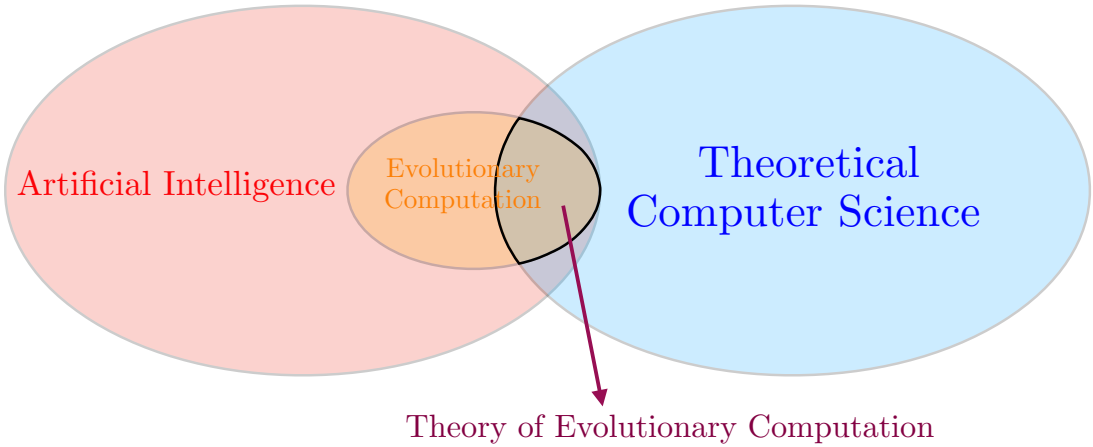
Theory of EC



- **Definition:** Results proven with **mathematical rigour** (Doerr, 2020)
- **Aim:** Provide a solid theoretical **foundation** and **guide** for EC:
  Explain and predict the performance $\Rightarrow$ **Explainable AI** $\Rightarrow$ **Trustworthy AI**
- **Tasks:** Runtime Analysis, Fitness Landscape Analysis, etc.
- **Conferences:** Theory Track in GECCO (TH-CPL B; CCF C; Core A)
  PPSN (CCF B; Core A)
  FOGA (Core A*)
- **Journals:** Algorithmica (TH-CPL B; CCF B; SCI 4)
  Theoretical Computer Science (TCS) (TH-CPL B; CCF B; SCI 4)
  IEEE Trans' on Evol' Comp' (TEVC) (TH-CPL B; CCF B; SCI Top1)
  Evolutionary Computation (ECJ) (TH-CPL B; CCF B; SCI 3)

After about 30 years of development,
the theory of EC has evolved into an **independent** and **important** sub-field in EC.

Theory of EC



- **Definition:** Results proven with **mathematical rigour** (Doerr, 2020)
- **Aim:** Provide a solid theoretical **foundation** and **guide** for EC:
  Explain and predict the performance $\Rightarrow$ **Explainable AI** $\Rightarrow$ **Trustworthy AI**
- **Tasks:** Runtime Analysis, Fitness Landscape Analysis, etc.
- **Conferences:** Theory Track in GECCO (TH-CPL B; CCF C; Core A)
  PPSN (CCF B; Core A)
  FOGA (Core A*)
- **Journals:** Algorithmica (TH-CPL B; CCF B; SCI 4)
  Theoretical Computer Science (TCS) (TH-CPL B; CCF B; SCI 4)
  IEEE Trans' on Evol' Comp' (TEVC) (TH-CPL B; CCF B; SCI Top1)
  Evolutionary Computation (ECJ) (TH-CPL B; CCF B; SCI 3)

After about 30 years of development,
the theory of EC has evolved into an **independent** and **important** sub-field in EC.

Why Do Theory for EC? (Doerr, 2020)

- **Why Do Theory?**
  - **Absolute guarantee** that the result is correct:
    - Correctness (It is proven)
    - Reproducibility
  - **Many results can only be obtained by theory:**
    - Make a statement on a very large or infinite set, e.g.,
      Problem size n
      TSP instances on n vertices
  - **A proof (automatically) give insight in**
    - Working principles of EC
    - Why the result is as it is
  - **Guide to design new algorithms**
    - Can see what exactly is the bottleneck for a result
    - How improve algorithms

Why Do Theory for EC? (Doerr, 2020)

- **Why Do Theory?**
  - **Absolute guarantee** that the result is correct:
    - Correctness (It is proven)
    - Reproducibility
  - **Many results can only be obtained by theory:**
    - Make a statement on a very large or infinite set, e.g.,
      Problem size n
      TSP instances on n vertices
  - **A proof (automatically) give insight in**
    - Working principles of EC
    - Why the result is as it is
  - **Guide to design new algorithms**
    - Can see what exactly is the bottleneck for a result
    - How improve algorithms

Why Do Theory for EC? (Doerr, 2020)

- **Why Do Theory?**
  - **Absolute guarantee** that the result is correct:
    - Correctness (It is proven)
    - Reproducibility
  - **Many results can only be obtained by theory:**
    - Make a statement on a very large or infinite set, e.g.,
      Problem size n
      TSP instances on n vertices
  - **A proof (automatically) give insight in**
    - Working principles of EC
    - Why the result is as it is
  - **Guide to design new algorithms**
    - Can see what exactly is the bottleneck for a result
    - How improve algorithms

Why Do Theory for EC? (Doerr, 2020)

- **Why Do Theory?**
  - **Absolute guarantee** that the result is correct:
    - Correctness (It is proven)
    - Reproducibility
  - **Many results can only be obtained by theory:**
    - Make a statement on a very large or infinite set, e.g.,
      Problem size n
      TSP instances on n vertices
  - **A proof (automatically) give insight in**
    - Working principles of EC
    - Why the result is as it is
  - **Guide to design new algorithms**
    - Can see what exactly is the bottleneck for a result
    - How improve algorithms

Why Do Theory for EC? (Doerr, 2020)

- **Why Do Theory?**
  - **Absolute guarantee** that the result is correct:
    - Correctness (It is proven)
    - Reproducibility
  - **Many results can only be obtained by theory:**
    - Make a statement on a very large or infinite set, e.g.,
      Problem size n
      TSP instances on n vertices
  - **A proof (automatically) give insight in**
    - Working principles of EC
    - Why the result is as it is
  - **Guide to design new algorithms**
    - Can see what exactly is the bottleneck for a result
    - How improve algorithms

Limitations of Theoretical Research (Doerr, 2020)

- **Limitations:**
  - **Restricted scope:**
    - So far, only known-structure optimisation problems, e.g., OneMax*, LeadingOnes†, Linear function‡, SparseLocalOpt Minimum Spanning Tree, Shortest Path, Knapsack, etc.
  - **Less precise results:** Constants are not tight, e.g.,
    - "$O(n^2)$" = "less than cn^2 for some unspecified constant c"
  - **Theory results can be very difficult to obtain:**
    - The proof might be short, but finding it took long hours

Usually supplemented with experimental results.

*OneMax$(x) :=: \mathrm{OM}(x) := \sum_{i=1}^{n}(x_i)$, where $x \in \{0,1\}^n$

†LeadingOnes$(x) :=: \mathrm{LO}(x) := \prod_{i=1}^{n}\sum_{i=1}^{j}(x_i)$, where $x \in \{0,1\}^n$

‡LinearFunciton$(x) :=: \mathrm{LF}(x) := \sum_{i=1}^{j}(w_i x_i)$ where $w_i \in \mathbb{R}$ for all $i \in [n]$ and $x \in \{0,1\}^n$

Limitations of Theoretical Research (Doerr, 2020)

- **Limitations:**
  - **Restricted scope:**
    - So far, only known-structure optimisation problems, e.g., OneMax*, LeadingOnes†, Linear function‡, SparseLocalOpt Minimum Spanning Tree, Shortest Path, Knapsack, etc.
  - **Less precise results:** Constants are not tight, e.g.,
    - "$O(n^2)$" = "less than cn^2 for some unspecified constant c"
  - **Theory results can be very difficult to obtain:**
    - The proof might be short, but finding it took long hours

Usually supplemented with experimental results.

*OneMax(x) =: OM(x) := $\sum_{i=1}^{n}(x_i)$, where $x \in \{0,1\}^n$

†LeadingOnes(x) =: LO(x) := $\prod_{i=1}^{n}\sum_{i=1}^{j}(x_i)$, where $x \in \{0,1\}^n$

‡LinearFunciton(x) =: LF(x) := $\sum_{i=1}^{j}(w_i x_i)$ where $w_i \in \mathbb{R}$ for all $i \in [n]$ and $x \in \{0,1\}^n$

Limitations of Theoretical Research (Doerr, 2020)

- **Limitations:**
  - **Restricted scope:**
    - So far, only known-structure optimisation problems, e.g., OneMax*, LeadingOnes†, Linear function‡, SparseLocalOpt Minimum Spanning Tree, Shortest Path, Knapsack, etc.
  - **Less precise results:** Constants are not tight, e.g.,
    - "$O(n^2)$" = "less than cn^2 for some unspecified constant c"
  - **Theory results can be very difficult to obtain:**
    - The proof might be short, but finding it took long hours

Usually supplemented with experimental results.

*OneMax(x) =: OM(x) := $\sum_{i=1}^{n}(x_i)$, where $x \in \{0,1\}^n$

†LeadingOnes(x) =: LO(x) := $\prod_{i=1}^{n}\sum_{i=1}^{j}(x_i)$, where $x \in \{0,1\}^n$

‡LinearFunciton(x) =: LF(x) := $\sum_{i=1}^{j}(w_i x_i)$ where $w_i \in \mathbb{R}$ for all $i \in [n]$ and $x \in \{0,1\}^n$

Limitations of Theoretical Research (Doerr, 2020)

- **Limitations:**
  - **Restricted scope:**
    - So far, only known-structure optimisation problems, e.g., OneMax*, LeadingOnes†, Linear function‡, SparseLocalOpt Minimum Spanning Tree, Shortest Path, Knapsack, etc.
  - **Less precise results:** Constants are not tight, e.g.,
    - "$O(n^2)$" = "less than cn^2 for some unspecified constant c"
  - **Theory results can be very difficult to obtain:**
    - The proof might be short, but finding it took long hours

Usually supplemented with experimental results.

---

*OneMax(x) =: OM(x) := $\sum_{i=1}^{n}(x_i)$, where $x \in \{0,1\}^n$

†LeadingOnes(x) =: LO(x) := $\prod_{i=1}^{n}\sum_{i=1}^{j}(x_i)$, where $x \in \{0,1\}^n$

‡LinearFunciton(x) =: LF(x) := $\sum_{i=1}^{j}(w_i x_i)$ where $w_i \in \mathbb{R}$ for all $i \in [n]$ and $x \in \{0,1\}^n$

Limitations of Theoretical Research (Doerr, 2020)

- **Limitations:**
  - **Restricted scope:**
    - So far, only known-structure optimisation problems, e.g., OneMax*, LeadingOnes†, Linear function‡, SparseLocalOpt Minimum Spanning Tree, Shortest Path, Knapsack, etc.
  - **Less precise results:** Constants are not tight, e.g.,
    - "$O(n^2)$" = "less than cn^2 for some unspecified constant c"
  - **Theory results can be very difficult to obtain:**
    - The proof might be short, but finding it took long hours

Usually supplemented with experimental results.

*OneMax(x) =: OM(x) := $\sum_{i=1}^{n}(x_i)$, where $x \in \{0,1\}^n$

†LeadingOnes(x) =: LO(x) := $\prod_{i=1}^{n}\sum_{i=1}^{j}(x_i)$, where $x \in \{0,1\}^n$

‡LinearFunciton(x) =: LF(x) := $\sum_{i=1}^{j}(w_i x_i)$ where $w_i \in \mathbb{R}$ for all $i \in [n]$ and $x \in \{0,1\}^n$

Recall: A Simple EA for Binary Search Space

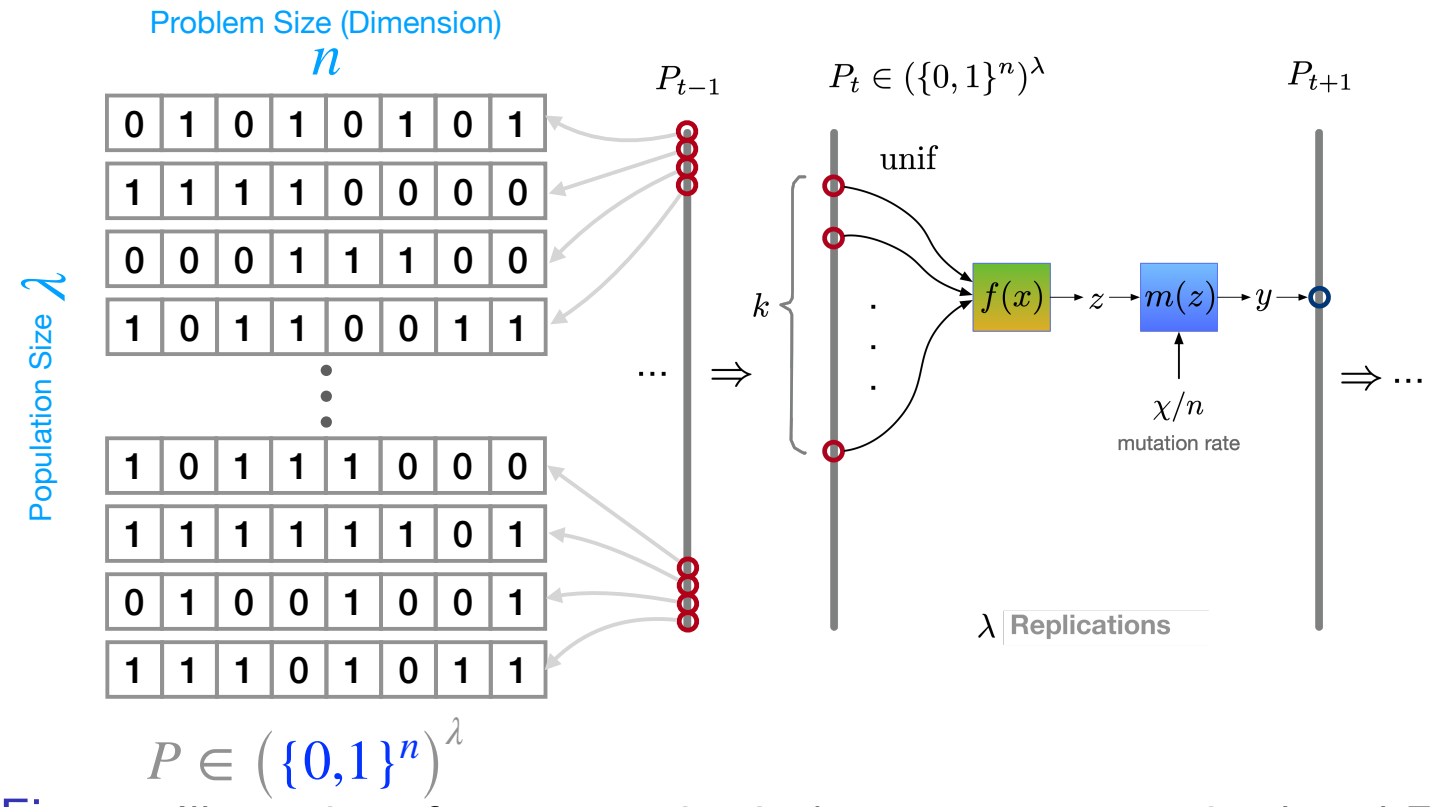


Figure: Illustration of one generation in k-tournament mutation-based EA

- **Selection Mechanism:** k-tournament (*This*), (μ,λ), Power-law, etc.
- **Mutation Operator:** $m : \{0,1\}^n \to \{0,1\}^n$
  Bit-wisely flip a solution z with probability χ/n, so-called **Mutation Rate**.

Parameter Settings in EAs

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

Parameter Settings in EAs

- 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**

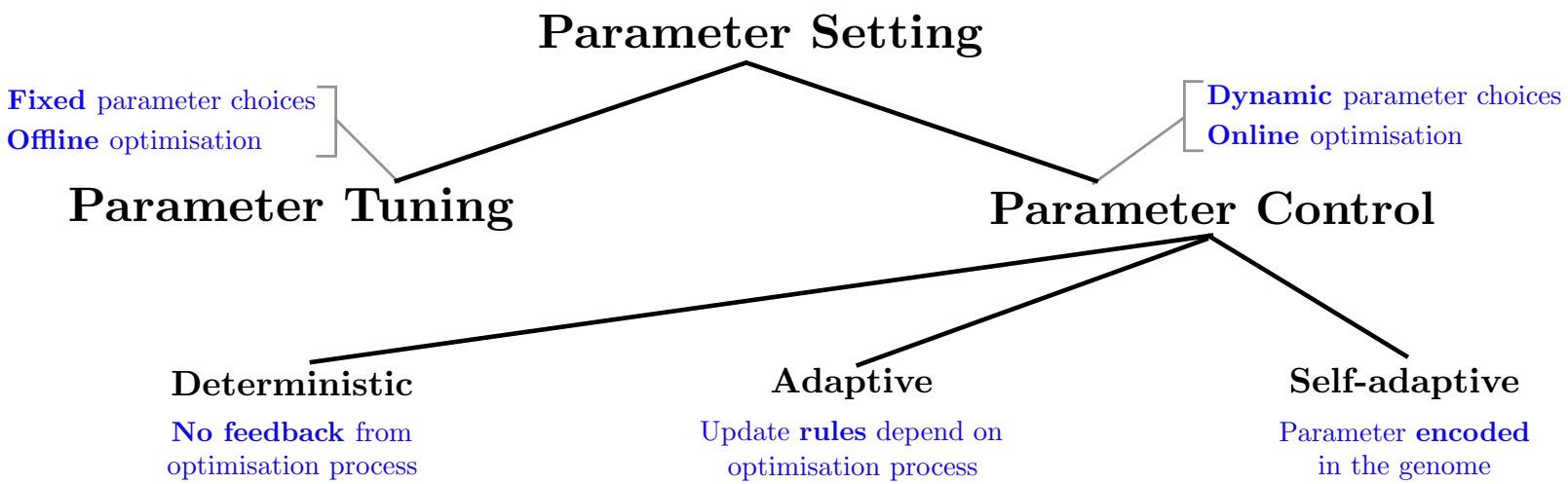
- Classification scheme of parameter setting (Eiben et al., 1999):

Parameter Settings in EAs

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

Parameter Settings in EAs

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



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

$\Rightarrow$ **Encoded** parameters into genome and

$\Rightarrow$ **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

$\Rightarrow$ **Encoded** parameters into genome and

$\Rightarrow$ **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

$\Rightarrow$ **Encoded** parameters into genome and

$\Rightarrow$ **Evolve together** parameters with solutions

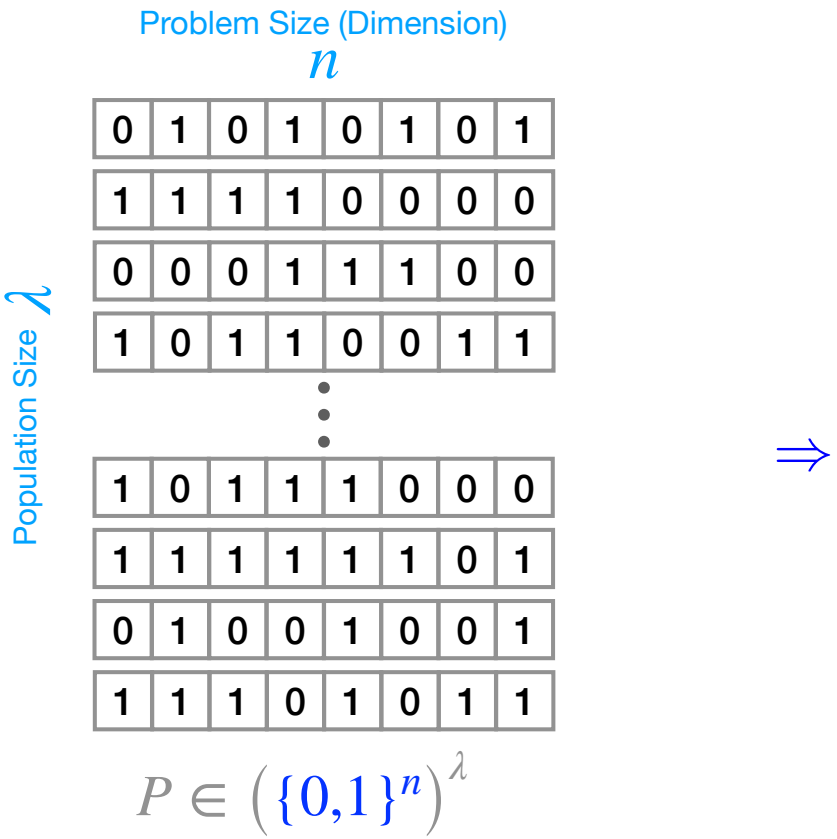


Figure: Static Population

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

$\Rightarrow$ **Encoded** parameters into genome and

$\Rightarrow$ **Evolve together** parameters with solutions

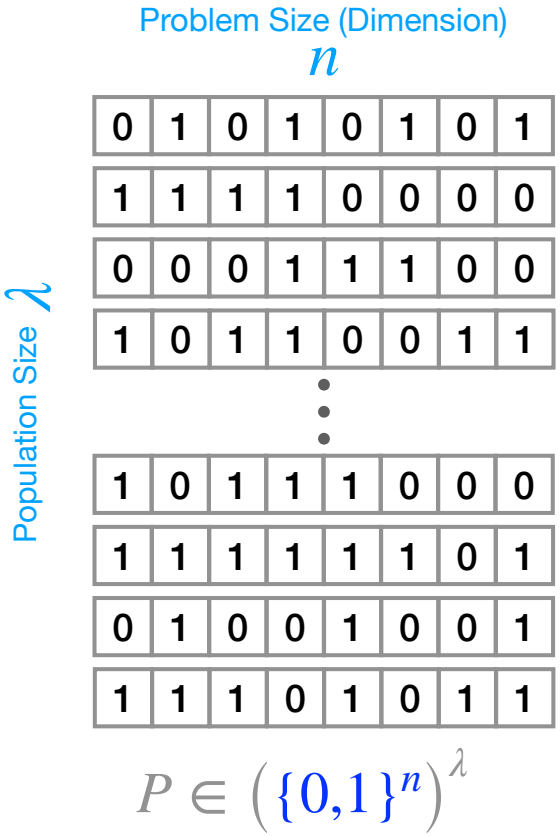


Figure: Static Population

$\Rightarrow$

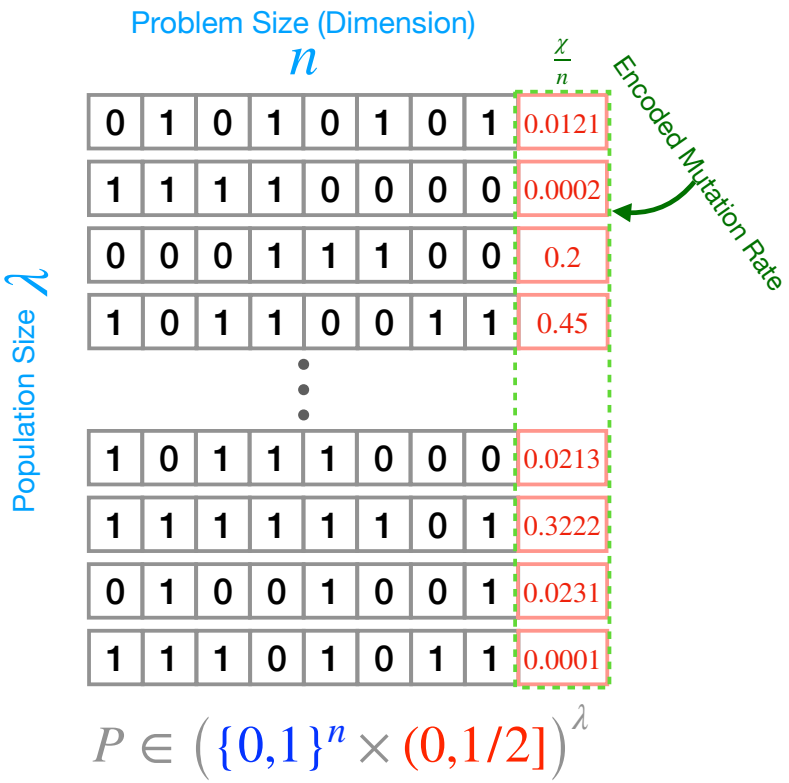


Figure: Self-adaptive Population

An example EA with Self-adapting Mutation Rate

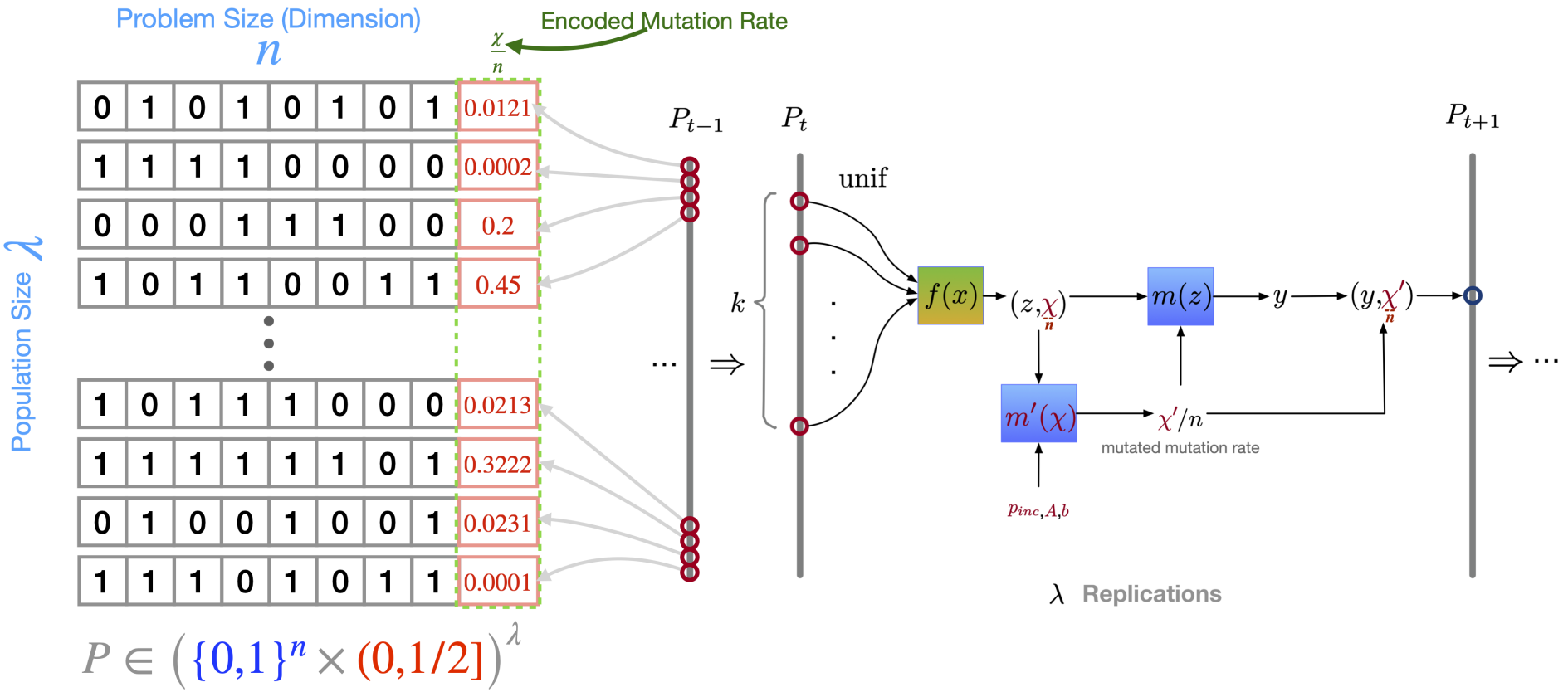


Figure: Illustration of one generation in k-tournament EA with self-adapting mutation rate

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

An example EA with Self-adapting Mutation Rate

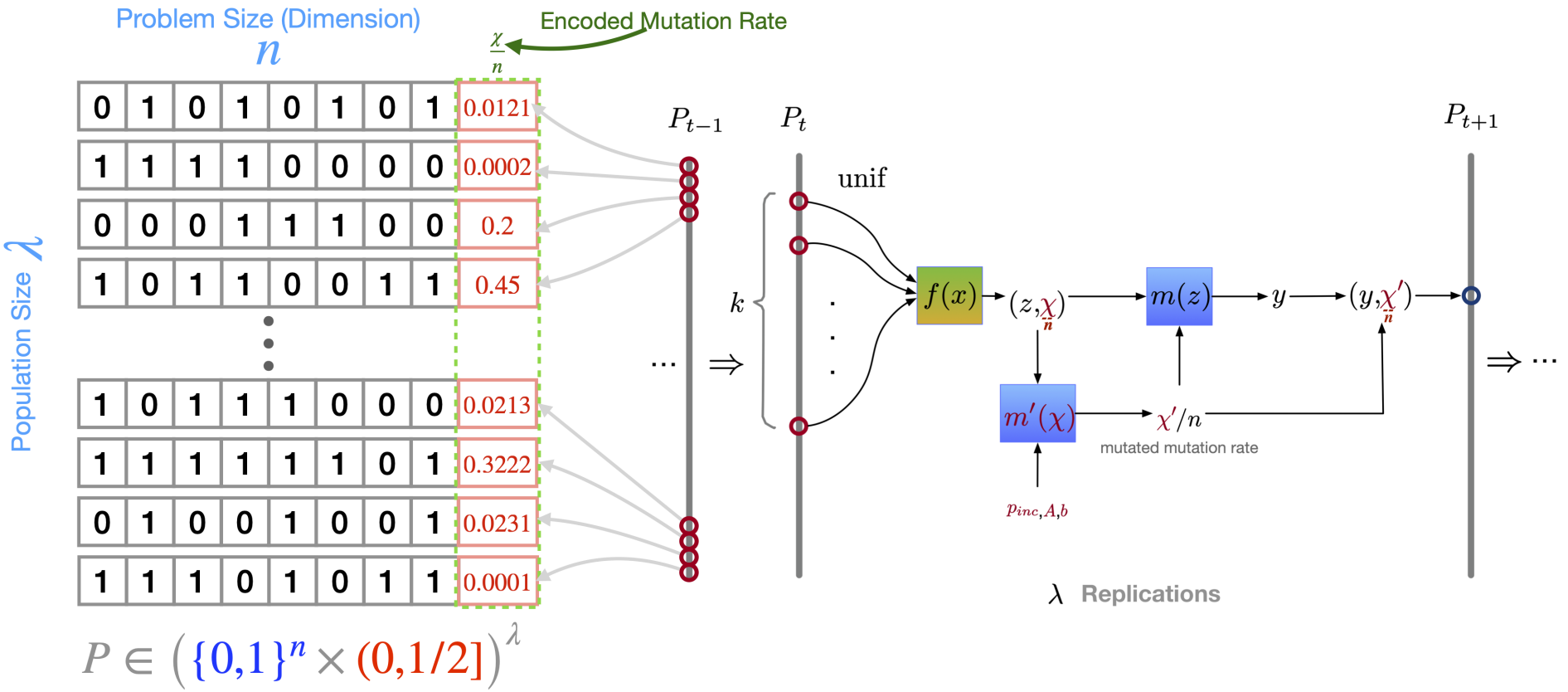


Figure: Illustration of one generation in k-tournament EA with self-adapting mutation rate

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

An example EA with Self-adapting Mutation Rate

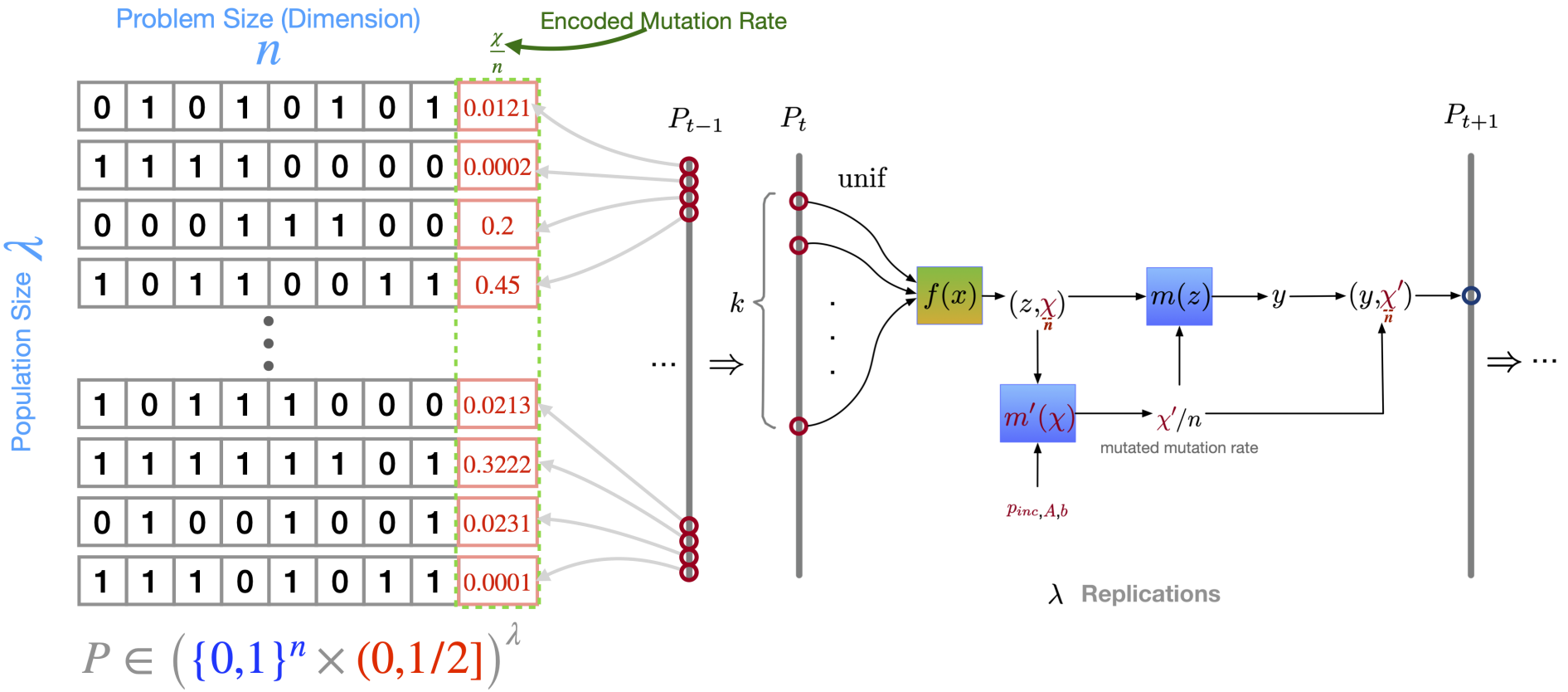


Figure: Illustration of one generation in k-tournament EA with self-adapting mutation rate

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

An example EA with Self-adapting Mutation Rate

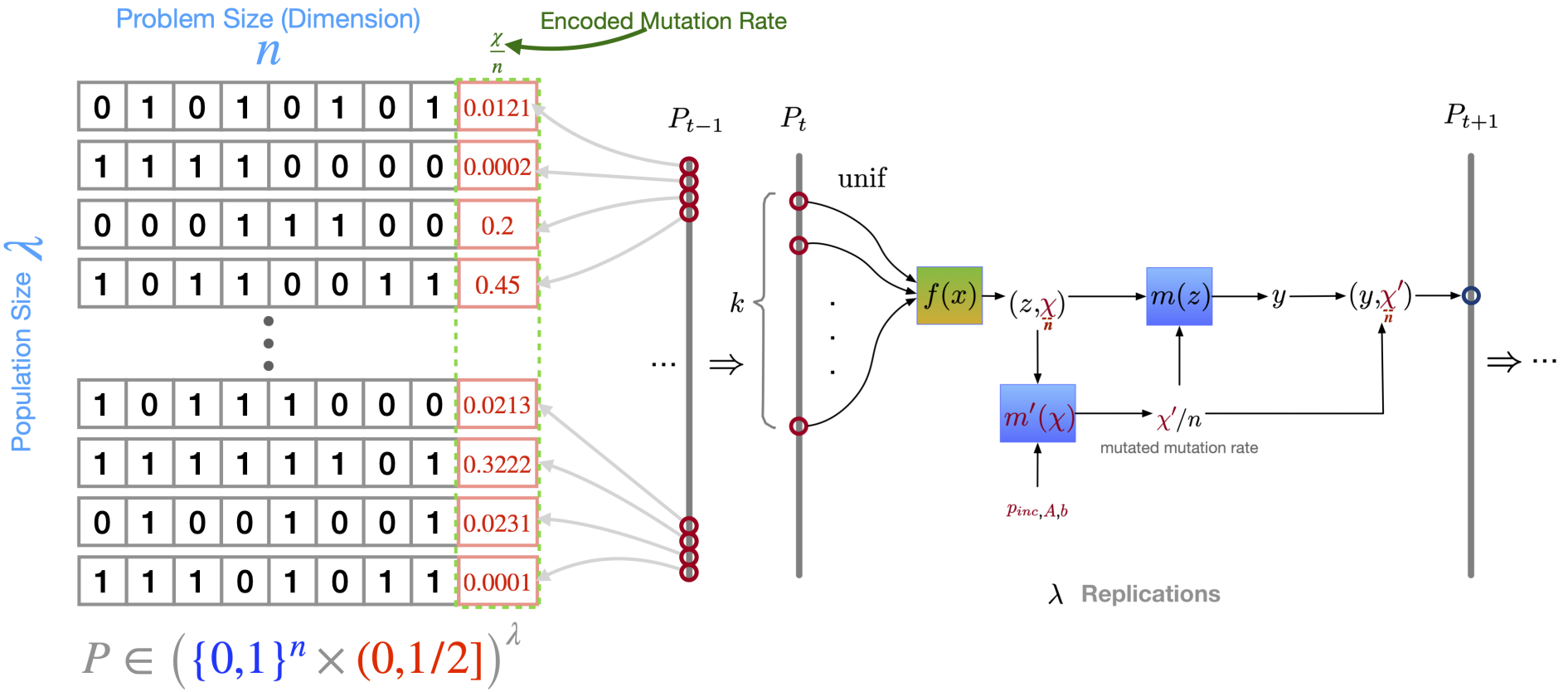


Figure: Illustration of one generation in k-tournament EA with self-adapting mutation rate

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

An example EA with Self-adapting Mutation Rate

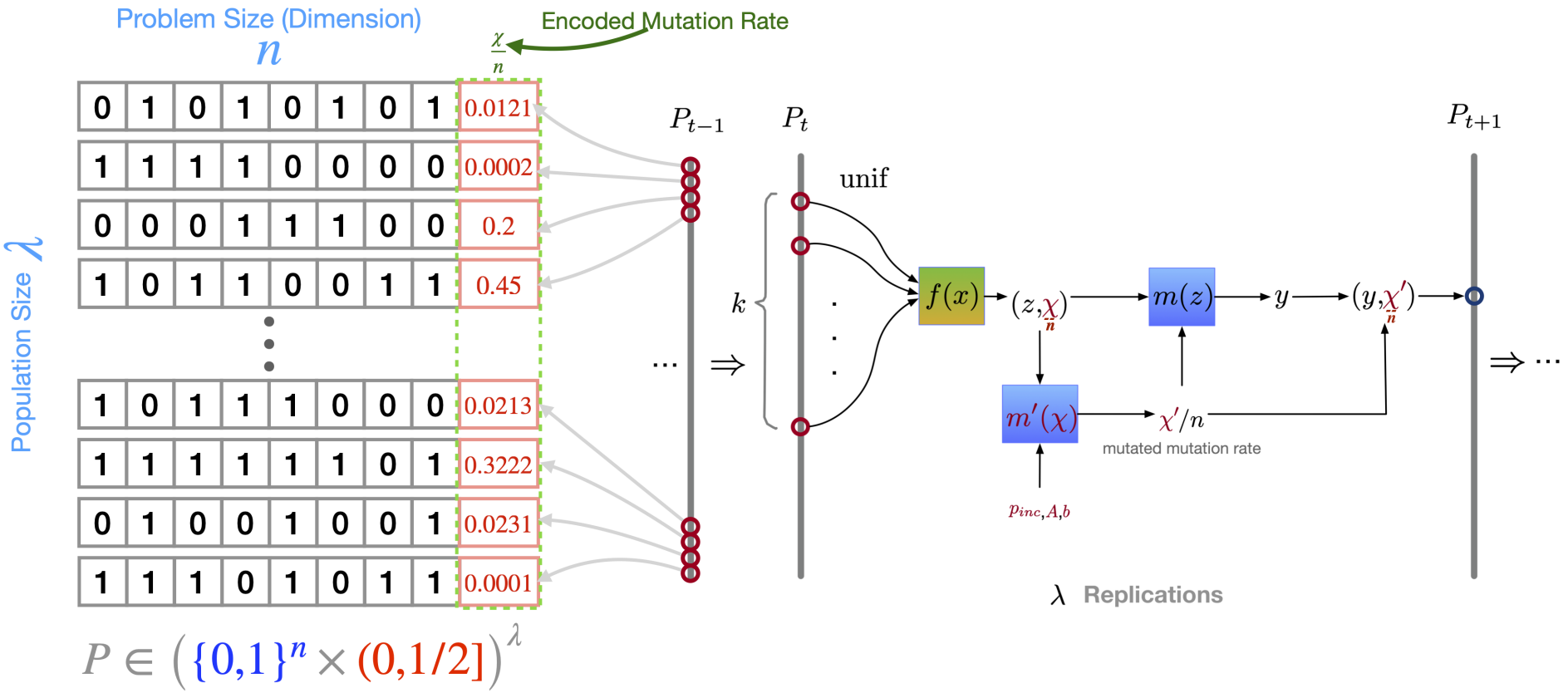


Figure: Illustration of one generation in k-tournament EA with self-adapting mutation rate

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

Existing Results on Self-adaptation in EC

Existing Studies on Self-adaptive EAs (SAEAs)

Several theoretical and empirical studies have been conducted on SAEAs, demonstrating their advantages in some scenarios:

- **Toy Problems:**
  - (Doerr et al., 2021)
- **Uncertain Optimisation:**
  - **Unknown-length Functions**
    - (Case and Lehre, 2020)
  - **Noisy Optimisation**
    - (Lehre and Qin, 2021, 2022a, 2023a)
  - **Dynamic Optimisation (Tracking Dynamic Optima)**
    - (Lehre and Qin, 2023b)
- **Multi-modal Landscape (Escaping Local Optima)**
  - (Dang and Lehre, 2016; Lehre and Qin, 2022b)
- **Complex Combinatorial Optimisation Problems**
  - (Qin and Lehre, 2022)

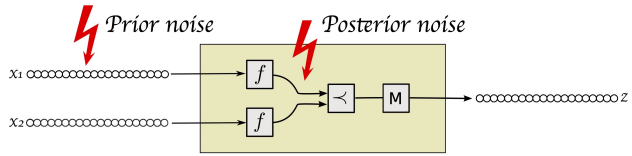
Existing Studies on Self-adaptive EAs (SAEAs)

Several theoretical and empirical studies have been conducted on SAEAs, demonstrating their advantages in some scenarios:

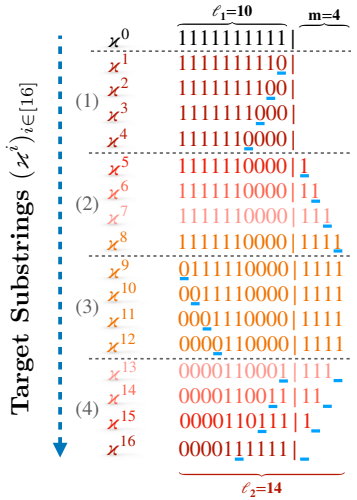
- **Toy Problems:**
  - (Doerr et al., 2021)
- **Uncertain Optimisation:**
  - **Unknown-length Functions**
    - (Case and Lehre, 2020)
  - **Noisy Optimisation**
    - (Lehre and Qin, 2021, 2022a, 2023a)
  - **Dynamic Optimisation (Tracking Dynamic Optima)**
    - (Lehre and Qin, 2023b)
- **Multi-modal Landscape (Escaping Local Optima)**
  - (Dang and Lehre, 2016; Lehre and Qin, 2022b)
- **Complex Combinatorial Optimisation Problems**
  - (Qin and Lehre, 2022)

Existing Studies on Self-adaptive EAs (SAEAs)

Several theoretical and empirical studies have been conducted on SAEAs, demonstrating their advantages in some scenarios:

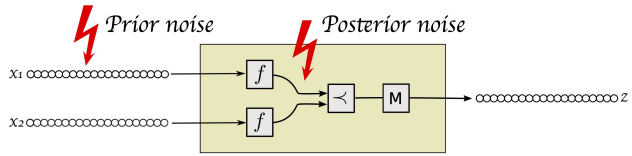


- **Toy Problems:**
  - (Doerr et al., 2021)
- **Uncertain Optimisation:**
  - **Unknown-length Functions**
    - (Case and Lehre, 2020)
  - **Noisy Optimisation**
    - (Lehre and Qin, 2021, 2022a, 2023a)
  - **Dynamic Optimisation (Tracking Dynamic Optima)**
    - (Lehre and Qin, 2023b)
- **Multi-modal Landscape (Escaping Local Optima)**
  - (Dang and Lehre, 2016; Lehre and Qin, 2022b)
- **Complex Combinatorial Optimisation Problems**
  - (Qin and Lehre, 2022)

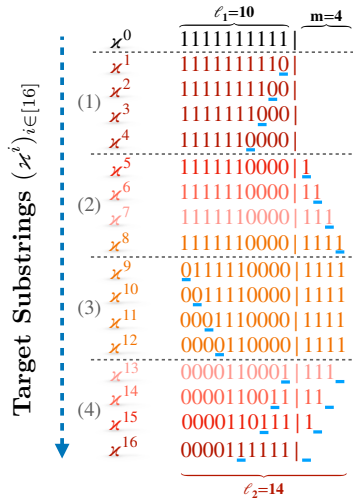


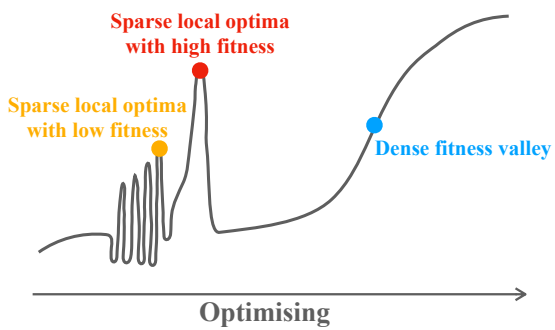
Existing Studies on Self-adaptive EAs (SAEAs)

Several theoretical and empirical studies have been conducted on SAEAs, demonstrating their advantages in some scenarios:



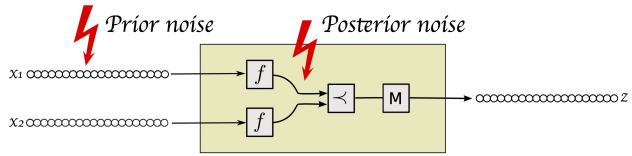
- **Toy Problems:**
  - (Doerr et al., 2021)
- **Uncertain Optimisation:**
  - **Unknown-length Functions**
    - (Case and Lehre, 2020)
  - **Noisy Optimisation**
    - (Lehre and Qin, 2021, 2022a, 2023a)
  - **Dynamic Optimisation (Tracking Dynamic Optima)**
    - (Lehre and Qin, 2023b)
- **Multi-modal Landscape (Escaping Local Optima)**
  - (Dang and Lehre, 2016; Lehre and Qin, 2022b)
- **Complex Combinatorial Optimisation Problems**
  - (Qin and Lehre, 2022)



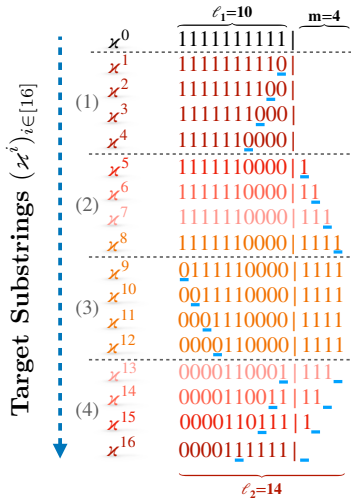


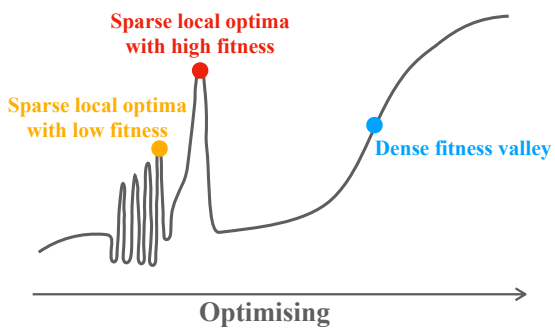
Existing Studies on Self-adaptive EAs (SAEAs)

Several theoretical and empirical studies have been conducted on SAEAs, demonstrating their advantages in some scenarios:



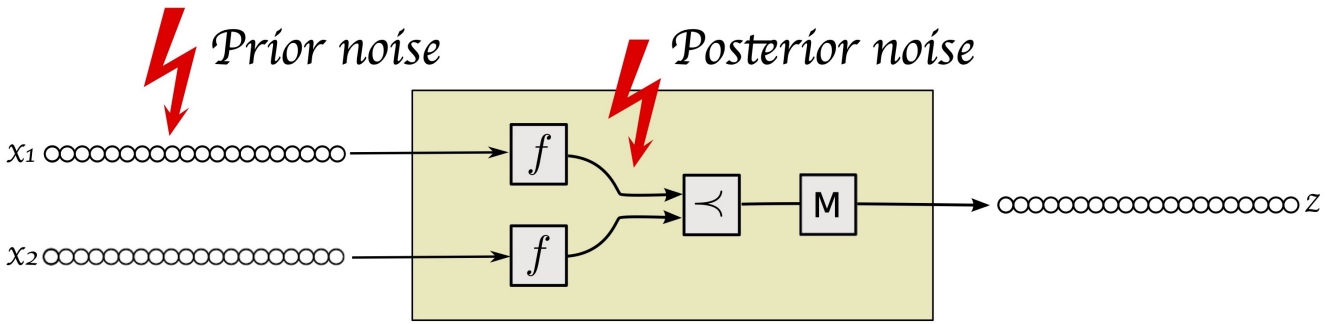
- **Toy Problems:**
  - (Doerr et al., 2021)
- **Uncertain Optimisation:**
  - **Unknown-length Functions**
    - (Case and Lehre, 2020)
  - **Noisy Optimisation**
    - (Lehre and Qin, 2021, 2022a, 2023a)
  - **Dynamic Optimisation (Tracking Dynamic Optima)**
    - (Lehre and Qin, 2023b)
- **Multi-modal Landscape (Escaping Local Optima)**
  - (Dang and Lehre, 2016; Lehre and Qin, 2022b)
- **Complex Combinatorial Optimisation Problems**
  - (Qin and Lehre, 2022)





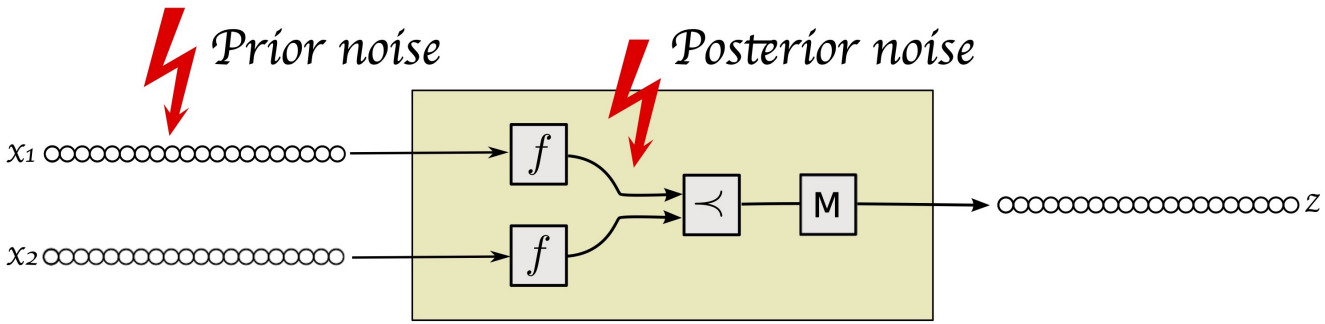
Self-adaptation in Noisy Optimisation

SAEAs on Noisy Optimisation



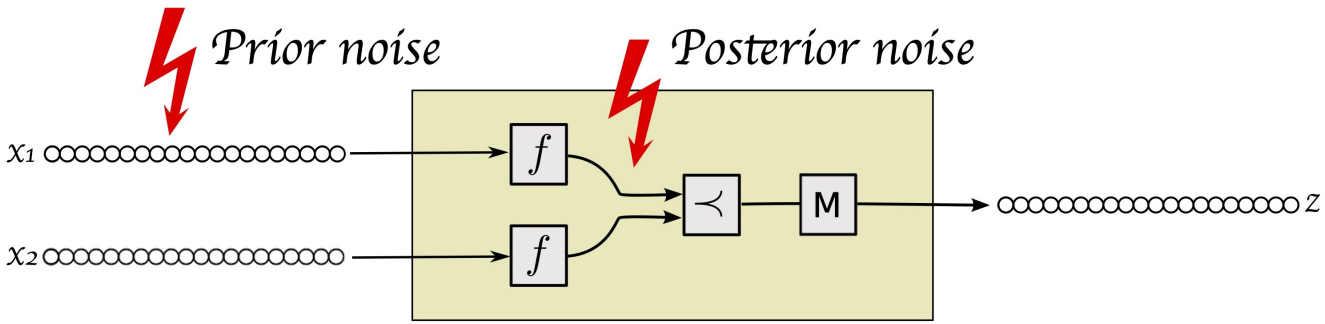
- From the previous theoretical study (Lehre and Qin, 2021, 2022a):
  - Non-elitist EAs can cope with the higher levels of noise by reducing the mutation rate.
  - Too small mutation rate slows down optimisation process.
  - However, we need to know the **exact** noise level to set a **proper** mutation rate.

SAEAs on Noisy Optimisation



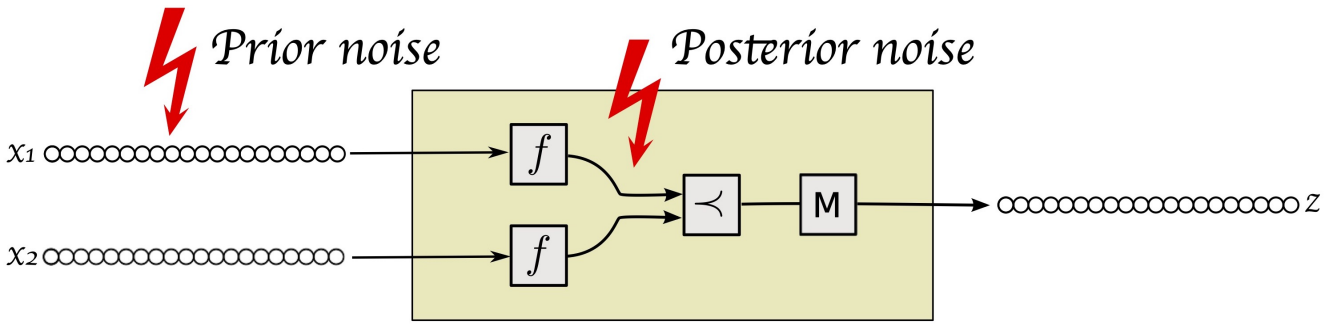
- From the previous theoretical study (Lehre and Qin, 2021, 2022a):
  - Non-elitist EAs can cope with the higher levels of noise by reducing the mutation rate.
  - Too small mutation rate slows down optimisation process.
  - However, we need to know the **exact** noise level to set a **proper** mutation rate.

SAEAs on Noisy Optimisation



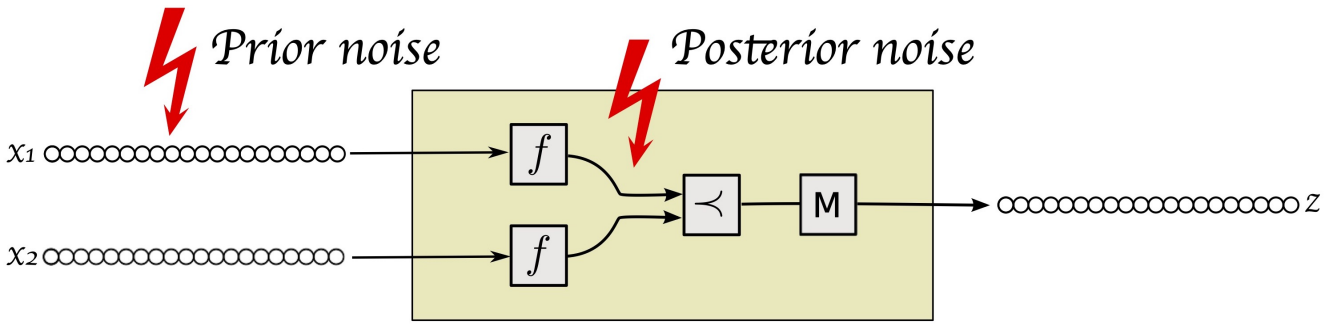
- From the previous theoretical study (Lehre and Qin, 2021, 2022a):
  - Non-elitist EAs can cope with the higher levels of noise by reducing the mutation rate.
  - Too small mutation rate slows down optimisation process.
  - However, we need to know the **exact** noise level to set a **proper** mutation rate.

SAEAs on Noisy Optimisation



- From the previous theoretical study (Lehre and Qin, 2021, 2022a):
  - Non-elitist EAs can cope with the higher levels of noise by reducing the mutation rate.
  - Too small mutation rate slows down optimisation process.
  - However, we need to know the **exact** noise level to set a **proper** mutation rate.

SAEAs on Noisy Optimisation



- From the previous theoretical study (Lehre and Qin, 2021, 2022a):
  - Non-elitist EAs can cope with the higher levels of noise by reducing the mutation rate.
  - Too small mutation rate slows down optimisation process.
  - However, we need to know the **exact** noise level to set a **proper** mutation rate.

Runtime Analysis for SAEAs under Noise

Lehre and Qin (2023) rigorously analysed runtimes of a EA with self-adapting from high/low mutation rates under symmetric noise:

Algorithm	Noise-free	Under Noise
2-tour' EA with χ_{high}	$O(n^2)$ (Well-known)	$e^{\Omega(n)}$ * (Lehre and Qin, 2022a)
2-tour' EA with χ_{low}	$\Omega(n^2 \log(n))$ (Sudholt, 2013)	$O(n^3)$ † (Lehre and Qin, 2022a)
Random 2-tour' EA	$O(n^2)$ (Lehre and Qin, 2023a)	$e^{\Omega(n)}$ * (Lehre and Qin, 2023a)
Self-adaptive 2-tour' EA	$O(n^2)$ (Lehre and Qin, 2023a)	$O(n^3)$ (Lehre and Qin, 2023a) †

- **High mutation rate** fails EA under noise.
- **Low mutation rate** slows down EA without noise.
- **Randomly choosing mutation rate** also fails EA under noise.
- **Self-adapting** mutation rates guarantees lowest runtime regardless of noise.

*For some constant noise level $q \in (0, 1/2)$.

†For all constant noise level $q \in (0, 1/2)$.

Runtime Analysis for SAEAs under Noise

Lehre and Qin (2023) rigorously analysed runtimes of a EA with self-adapting from high/low mutation rates under symmetric noise:

Algorithm	**Noise-free**	**Under Noise**
2-tour' EA with χ_{high}	$O(n^2)$ (Well-known)	$e^{\Omega(n)}$ * (Lehre and Qin, 2022a)
2-tour' EA with χ_{low}	$\Omega(n^2 \log(n))$ (Sudholt, 2013)	$O(n^3)$ † (Lehre and Qin, 2022a)
Random 2-tour' EA	$O(n^2)$ (Lehre and Qin, 2023a)	$e^{\Omega(n)}$ * (Lehre and Qin, 2023a)
Self-adaptive 2-tour' EA	$O(n^2)$ (Lehre and Qin, 2023a)	$O(n^3)$ (Lehre and Qin, 2023a) †

- **High mutation rate** fails EA under noise.
- **Low mutation rate** slows down EA without noise.
- **Randomly choosing mutation rate** also fails EA under noise.
- **Self-adapting** mutation rates guarantees lowest runtime regardless of noise.

*For some constant noise level $q \in (0, 1/2)$.

†For all constant noise level $q \in (0, 1/2)$.

Runtime Analysis for SAEAs under Noise

Lehre and Qin (2023) rigorously analysed runtimes of a EA with self-adapting from high/low mutation rates under symmetric noise:

Algorithm	**Noise-free**	**Under Noise**
2-tour' EA with χ_{high}	$O(n^2)$ (Well-known)	$e^{\Omega(n)}$ * (Lehre and Qin, 2022a)
2-tour' EA with χ_{low}	$\Omega(n^2 \log(n))$ (Sudholt, 2013)	$O(n^3)$ † (Lehre and Qin, 2022a)
Random 2-tour' EA	$O(n^2)$ (Lehre and Qin, 2023a)	$e^{\Omega(n)}$ * (Lehre and Qin, 2023a)
Self-adaptive 2-tour' EA	$O(n^2)$ (Lehre and Qin, 2023a)	$O(n^3)$ (Lehre and Qin, 2023a) †

- **High mutation rate** fails EA under noise.
- **Low mutation rate** slows down EA without noise.
- **Randomly choosing mutation rate** also fails EA under noise.
- **Self-adapting** mutation rates guarantees lowest runtime regardless of noise.

*For some constant noise level $q \in (0, 1/2)$.

†For all constant noise level $q \in (0, 1/2)$.

Runtime Analysis for SAEAs under Noise

Lehre and Qin (2023) rigorously analysed runtimes of a EA with self-adapting from high/low mutation rates under symmetric noise:

Algorithm	**Noise-free**	**Under Noise**
2-tour' EA with χ_{high}	$O(n^2)$ (Well-known)	$e^{\Omega(n)}$ * (Lehre and Qin, 2022a)
2-tour' EA with χ_{low}	$\Omega(n^2 \log(n))$ (Sudholt, 2013)	$O(n^3)$ † (Lehre and Qin, 2022a)
Random 2-tour' EA	$O(n^2)$ (Lehre and Qin, 2023a)	$e^{\Omega(n)}$ * (Lehre and Qin, 2023a)
Self-adaptive 2-tour' EA	$O(n^2)$ (Lehre and Qin, 2023a)	$O(n^3)$ (Lehre and Qin, 2023a) †

- **High mutation rate** fails EA under noise.
- **Low mutation rate** slows down EA without noise.
- **Randomly choosing mutation rate** also fails EA under noise.
- **Self-adapting** mutation rates guarantees lowest runtime regardless of noise.

*For some constant noise level $q \in (0, 1/2)$.

†For all constant noise level $q \in (0, 1/2)$.

Runtime Analysis for SAEAs under Noise

Lehre and Qin (2023) rigorously analysed runtimes of a EA with self-adapting from high/low mutation rates under symmetric noise:

Algorithm	**Noise-free**	**Under Noise**
2-tour' EA with χ_{high}	$O(n^2)$ (Well-known)	$e^{\Omega(n)}$ * (Lehre and Qin, 2022a)
2-tour' EA with χ_{low}	$\Omega(n^2 \log(n))$ (Sudholt, 2013)	$O(n^3)$ † (Lehre and Qin, 2022a)
Random 2-tour' EA	$O(n^2)$ (Lehre and Qin, 2023a)	$e^{\Omega(n)}$ * (Lehre and Qin, 2023a)
Self-adaptive 2-tour' EA	$O(n^2)$ (Lehre and Qin, 2023a)	$O(n^3)$ (Lehre and Qin, 2023a) †

- **High mutation rate** fails EA under noise.
- **Low mutation rate** slows down EA without noise.
- **Randomly choosing mutation rate** also fails EA under noise.
- **Self-adapting** mutation rates guarantees lowest runtime regardless of noise.

*For some constant noise level $q \in (0, 1/2)$.

†For all constant noise level $q \in (0, 1/2)$.

Runtime Analysis for SAEAs under Noise

Lehre and Qin (2023) rigorously analysed runtimes of a EA with self-adapting from high/low mutation rates under symmetric noise:

Algorithm	**Noise-free**	**Under Noise**
2-tour' EA with χ_{high}	$O(n^2)$ (Well-known)	$e^{\Omega(n)}$ * (Lehre and Qin, 2022a)
2-tour' EA with χ_{low}	$\Omega(n^2 \log(n))$ (Sudholt, 2013)	$O(n^3)$ † (Lehre and Qin, 2022a)
Random 2-tour' EA	$O(n^2)$ (Lehre and Qin, 2023a)	$e^{\Omega(n)}$ * (Lehre and Qin, 2023a)
Self-adaptive 2-tour' EA	$O(n^2)$ (Lehre and Qin, 2023a)	$O(n^3)$ (Lehre and Qin, 2023a) †

- **High mutation rate** fails EA under noise.
- **Low mutation rate** slows down EA without noise.
- **Randomly choosing mutation rate** also fails EA under noise.
- **Self-adapting** mutation rates guarantees lowest runtime regardless of noise.

*For some constant noise level $q \in (0, 1/2)$.

†For all constant noise level $q \in (0, 1/2)$.

Empirical study on Noise

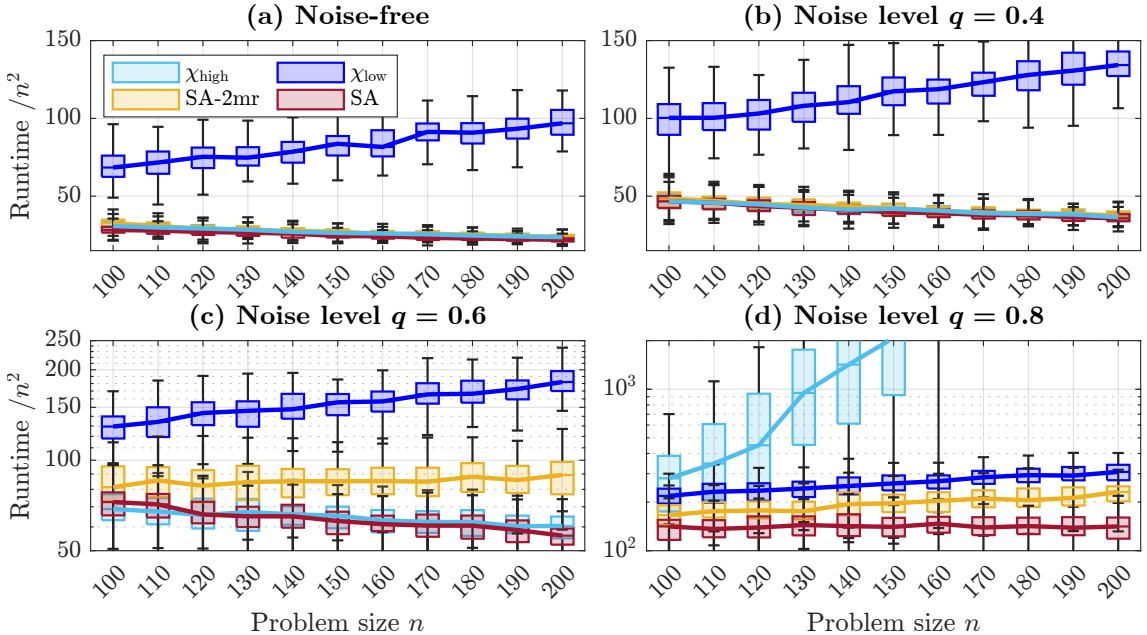


Figure: Runtimes on LeadingOnes under one-bit noise with noise levels q.

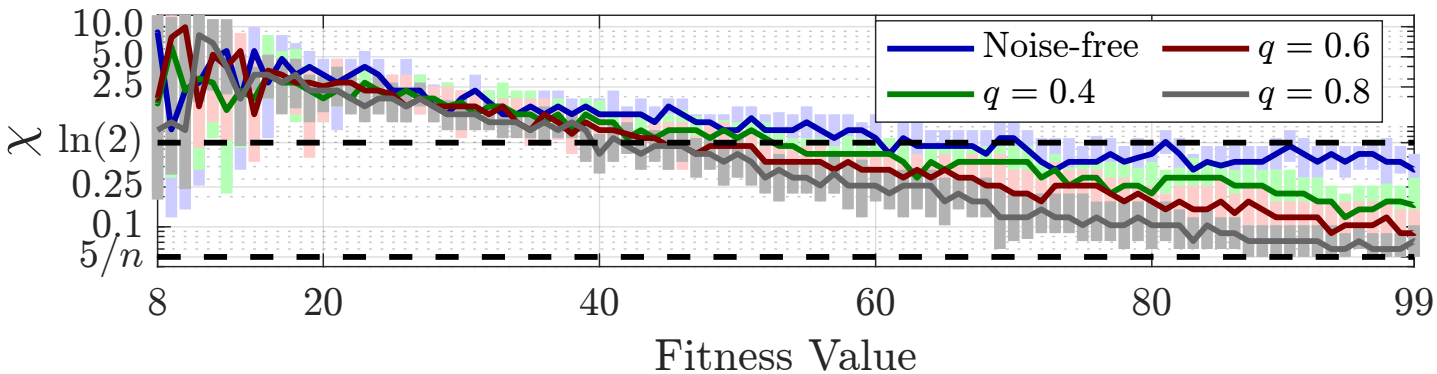
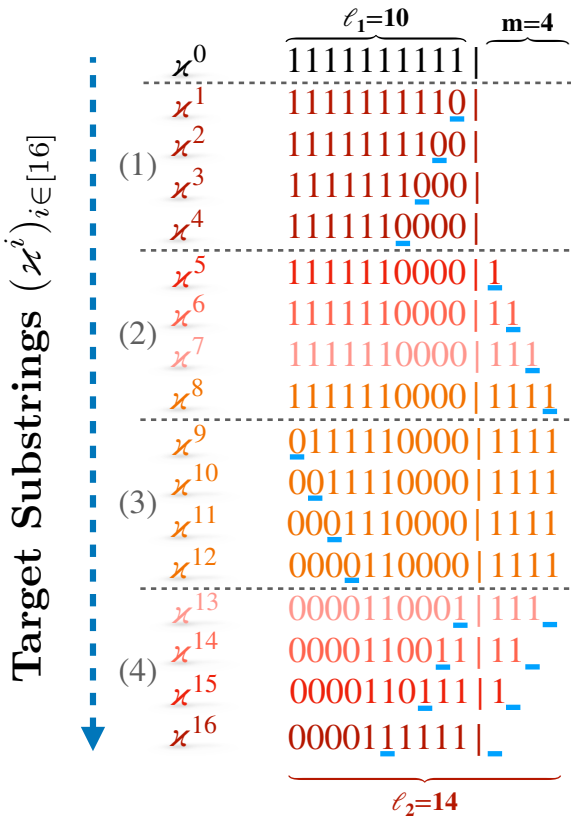


Figure: Mutation rate of SAEAs on LeadingOnes under one-bit noise with noise levels q.

Self-adaptation in Dynamic Optimisation

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



Four phases:

$\Leftarrow$ 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

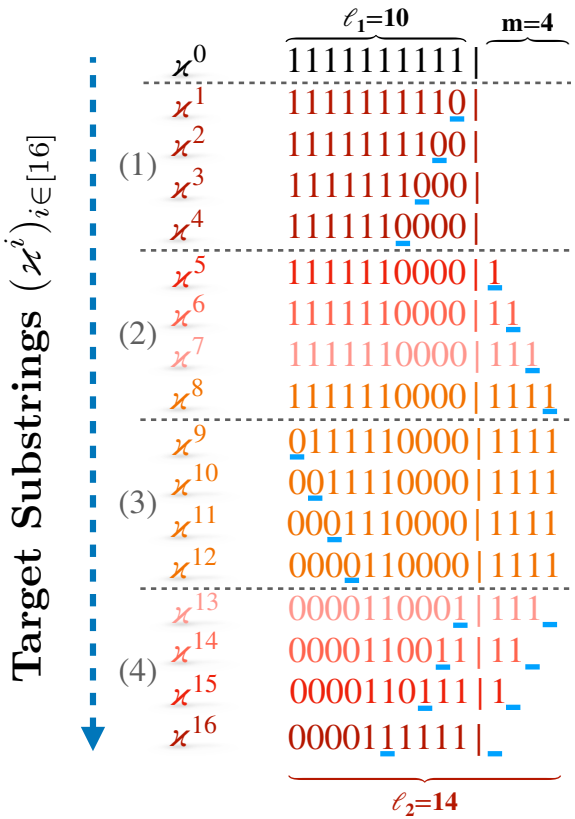
$\Leftarrow$ (2) $i \in [m+1..2m]$, the target substrings are becoming longer

$\Leftarrow$ (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 $\boxed{kn^{\varepsilon}|\varkappa^i|}$.

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



Four phases:

$\Leftarrow$ 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

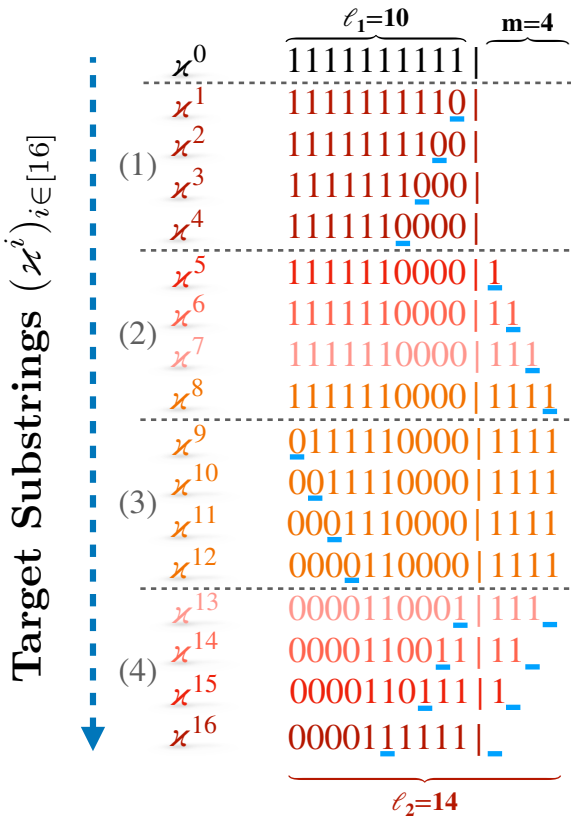
$\Leftarrow$ (2) $i \in [m+1..2m]$, the target substrings are becoming longer

$\Leftarrow$ (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 $\boxed{kn^{\varepsilon}|\varkappa^i|}$.

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



Four phases:

$\Leftarrow$ 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

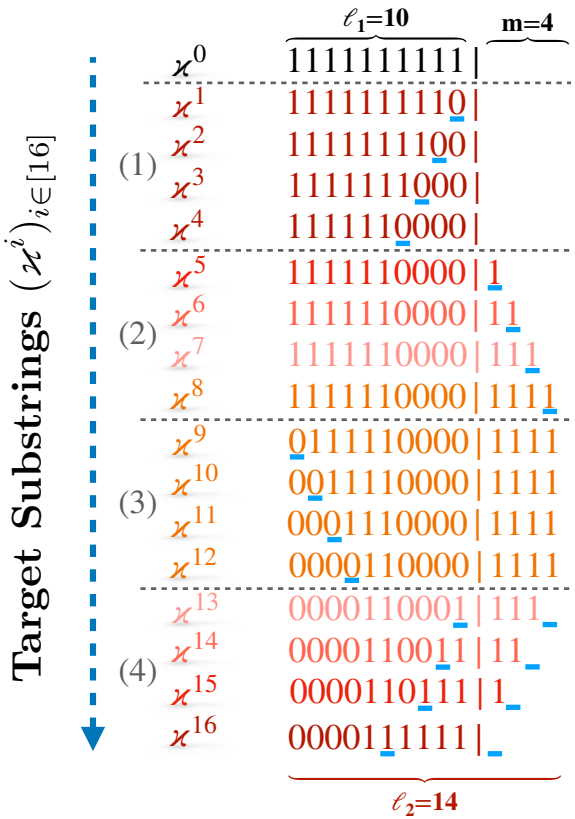
$\Leftarrow$ (2) $i \in [m+1..2m]$, the target substrings are becoming longer

$\Leftarrow$ (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 phases:

$\Leftarrow$ 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

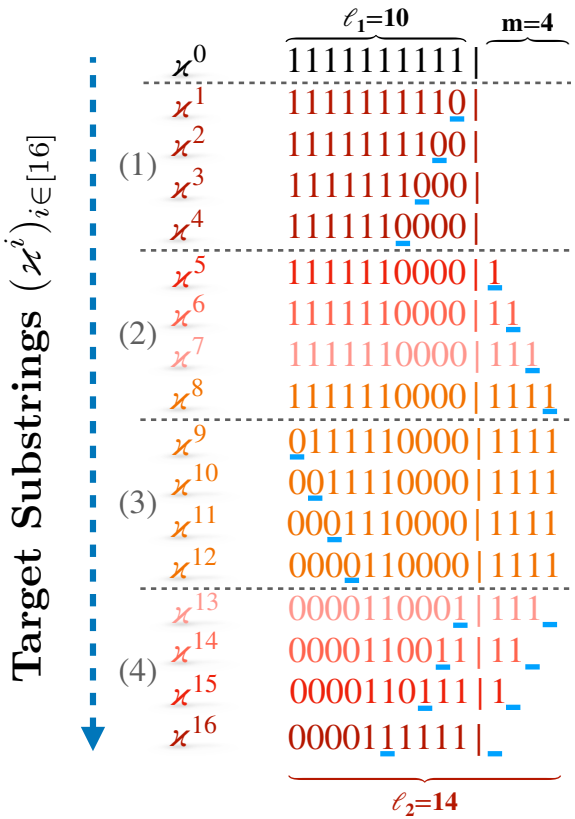
$\Leftarrow$ (2) $i \in [m+1..2m]$, the target substrings are becoming longer

$\Leftarrow$ (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 phases:

$\Leftarrow$ 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

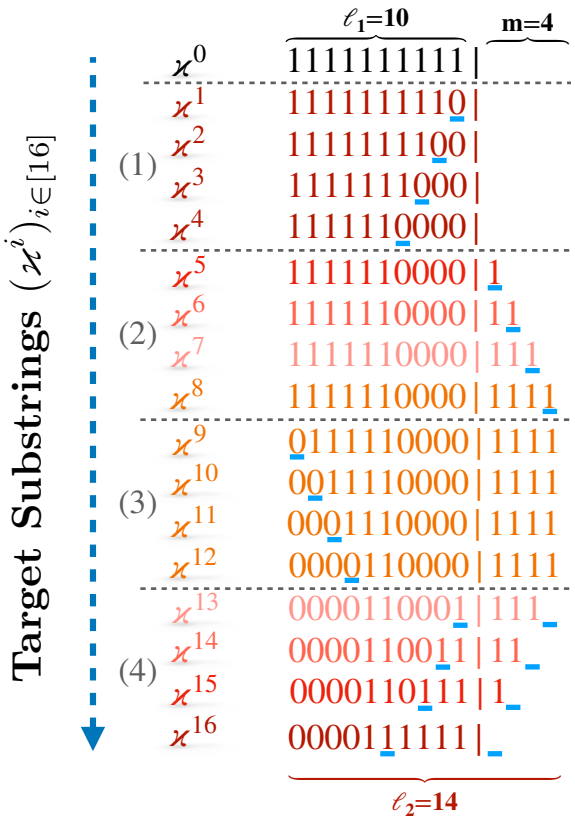
$\Leftarrow$ (2) $i \in [m+1..2m]$, the target substrings are becoming longer

$\Leftarrow$ (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 phases:

$\Leftarrow$ 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 $\boxed{kn^{\varepsilon}|\varkappa^i|}$.

Theoretical Analysis

Lehre and Qin (2023b) performed runtime analysis of EAs on the DSM problem:

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

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

Design new Self-adaptive EAs (SAEAs)

Moreover, the exploration of **designing self-adaptive algorithm** is currently ongoing in the community.

- **Sorting and selection mechanism**
  How to address the "order" of fitness and parameters.
- **Parameter adaptation mechanism**
  How to "mutate" parameters.
- **Parameter encoding method**
  How to encode parameters into the genome.
- **Self-adaptation of other parameters**
  How to self-adapt other parameters,
  e.g., crossover rate, selecture pressure, population size, etc.
- ...

Design new Self-adaptive EAs (SAEAs)

Moreover, the exploration of **designing self-adaptive algorithm** is currently ongoing in the community.

- **Sorting and selection mechanism**
  How to address the "order" of fitness and parameters.
- **Parameter adaptation mechanism**
  How to "mutate" parameters.
- **Parameter encoding method**
  How to encode parameters into the genome.
- **Self-adaptation of other parameters**
  How to self-adapt other parameters,
  e.g., crossover rate, selecture pressure, population size, etc.
- ...

Design new Self-adaptive EAs (SAEAs)

Moreover, the exploration of **designing self-adaptive algorithm** is currently ongoing in the community.

- **Sorting and selection mechanism**
  How to address the "order" of fitness and parameters.
- **Parameter adaptation mechanism**
  How to "mutate" parameters.
- **Parameter encoding method**
  How to encode parameters into the genome.
- **Self-adaptation of other parameters**
  How to self-adapt other parameters,
  e.g., crossover rate, selecture pressure, population size, etc.
- ...

Design new Self-adaptive EAs (SAEAs)

Moreover, the exploration of **designing self-adaptive algorithm** is currently ongoing in the community.

- **Sorting and selection mechanism**
  How to address the "order" of fitness and parameters.
- **Parameter adaptation mechanism**
  How to "mutate" parameters.
- **Parameter encoding method**
  How to encode parameters into the genome.
- **Self-adaptation of other parameters**
  How to self-adapt other parameters,
  e.g., crossover rate, selecture pressure, population size, etc.
- ...

Design new Self-adaptive EAs (SAEAs)

Moreover, the exploration of **designing self-adaptive algorithm** is currently ongoing in the community.

- **Sorting and selection mechanism**
  How to address the "order" of fitness and parameters.
- **Parameter adaptation mechanism**
  How to "mutate" parameters.
- **Parameter encoding method**
  How to encode parameters into the genome.
- **Self-adaptation of other parameters**
  How to self-adapt other parameters,
  e.g., crossover rate, selecture pressure, population size, etc.
- ...

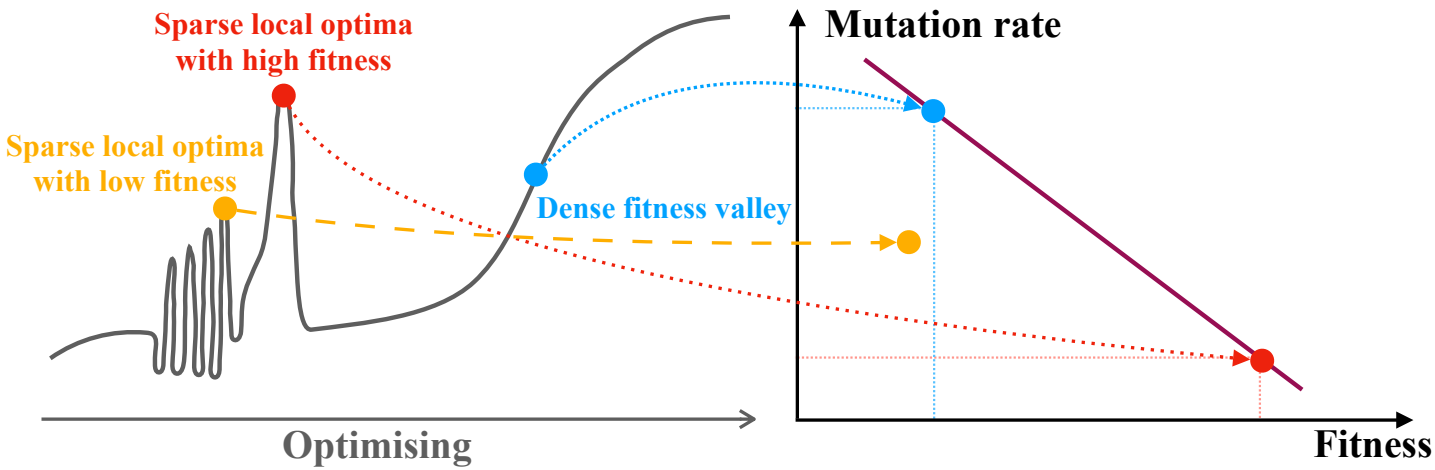
A new EA for single-objective: MOSA-EA

A new EA for single-objective: MOSA-EA

- The Multi-Objective Self-Adaptive Evolutionary Algorithms

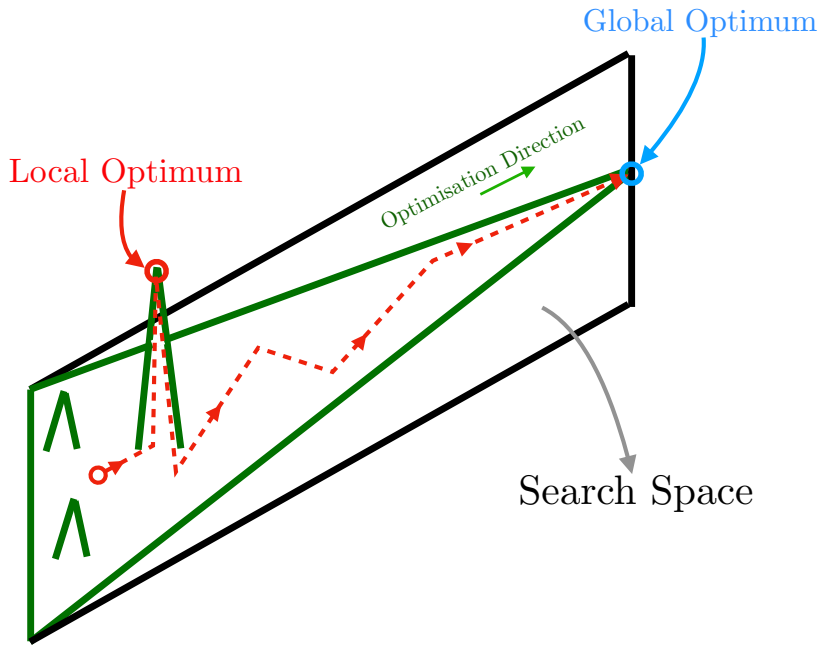
  In short, MOSA-EA[‖] (Lehre and Qin, 2022b).

  - **Non-elitism**
  - **Self-adaptation**
  - **Multi-objectivisation**



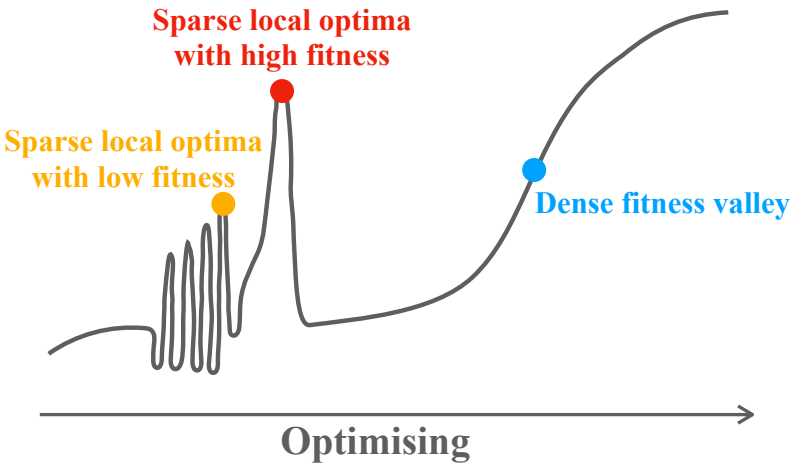
[‖] Implementation can be found in https://github.com/ChengCheng-Qin/mosa-ea.

Motivation: Sparse Local Optima



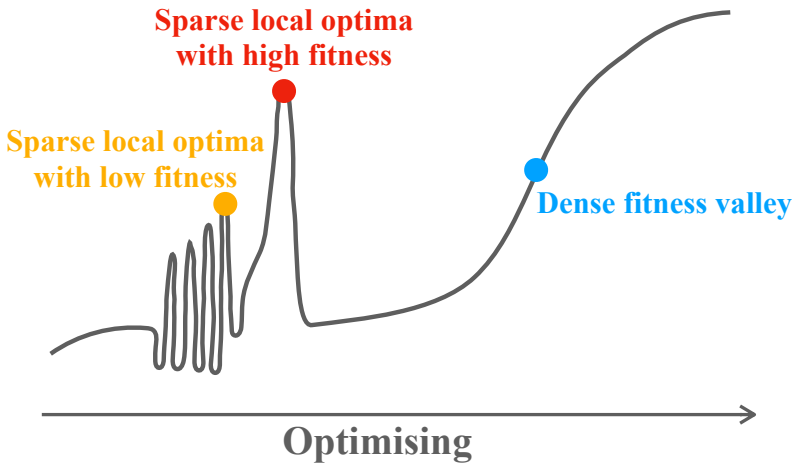
- Elitist EAs can get stuck on local optima (Jagerskupper and Storch, 2007; Dang et al., 2021a; Doerr, 2022; Dang et al., 2021b).
- *Sparse deceptive regions* (sparse local optima)

Motivation: Sparse Local Optima - Previous Works



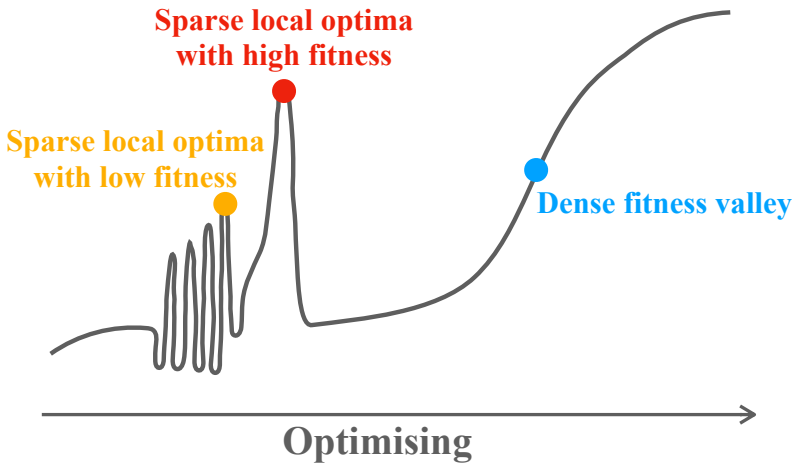
- SparseLocalOpt $\Rightarrow$ a kind of fitness landscapes with *sparse deceptive regions* (local optima) and *dense fitness valleys* (Dang et al., 2021b).
- *Non-linear non-elitist selection* and *sufficiently high mutation rate* (Dang et al., 2021b) $\Rightarrow$
  - Sparse local optimal individuals $\Rightarrow$ higher chance to be selected but only survive a small percentage of such individuals after mutation;
  - Dense fitness valley individuals $\Rightarrow$ less chance of being selected but can have higher chance of surviving mutation.

Motivation: Sparse Local Optima - Previous Works



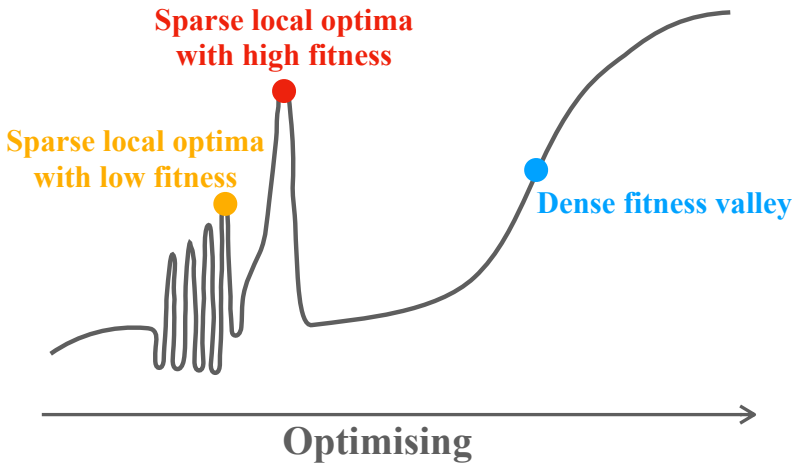
- SPARSELOCALOPT $\Rightarrow$ a kind of fitness landscapes with *sparse deceptive regions* (local optima) and *dense fitness valleys* (Dang et al., 2021b).
- *Non-linear non-elitist selection* and *sufficiently high mutation rate* (Dang et al., 2021b) $\Rightarrow$
  - Sparse local optimal individuals $\Rightarrow$ higher chance to be selected but only survive a small percentage of such individuals after mutation;
  - Dense fitness valley individuals $\Rightarrow$ less chance of being selected but can have higher chance of surviving mutation.

Motivation: Sparse Local Optima - Previous Works



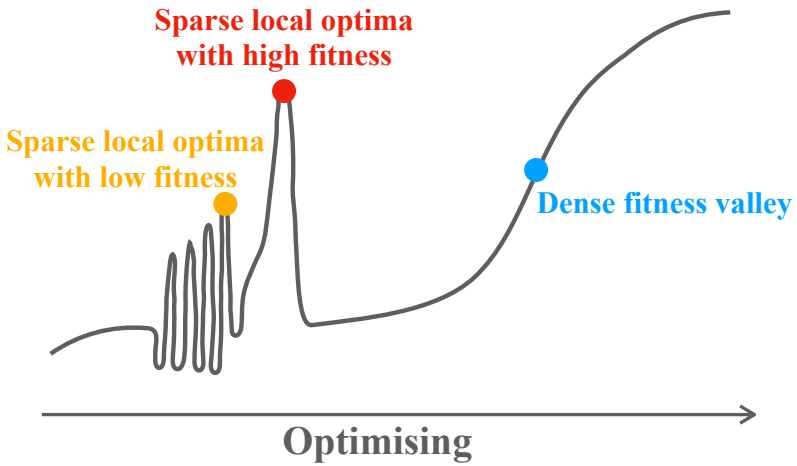
- SPARSELOCALOPT $\Rightarrow$ a kind of fitness landscapes with *sparse deceptive regions* (local optima) and *dense fitness valleys* (Dang et al., 2021b).
- *Non-linear non-elitist selection* and *sufficiently high mutation rate* (Dang et al., 2021b) $\Rightarrow$
  - Sparse local optimal individuals $\Rightarrow$ higher chance to be selected but only survive a small percentage of such individuals after mutation;
  - Dense fitness valley individuals $\Rightarrow$ less chance of being selected but can have higher chance of surviving mutation.

Motivation: Sparse Local Optima - Previous Works



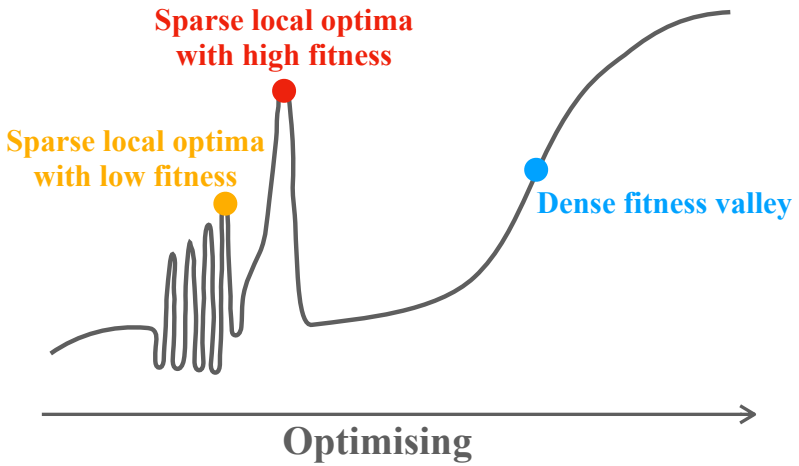
- SPARSELOCALOPT $\Rightarrow$ a kind of fitness landscapes with *sparse deceptive regions* (local optima) and *dense fitness valleys* (Dang et al., 2021b).
- *Non-linear non-elitist selection* and *sufficiently high mutation rate* (Dang et al., 2021b) $\Rightarrow$
  - Sparse local optimal individuals $\Rightarrow$ higher chance to be selected but only survive a small percentage of such individuals after mutation;
  - Dense fitness valley individuals $\Rightarrow$ less chance of being selected but can have higher chance of surviving mutation.

Motivation: Sparse Local Optima - Previous Works



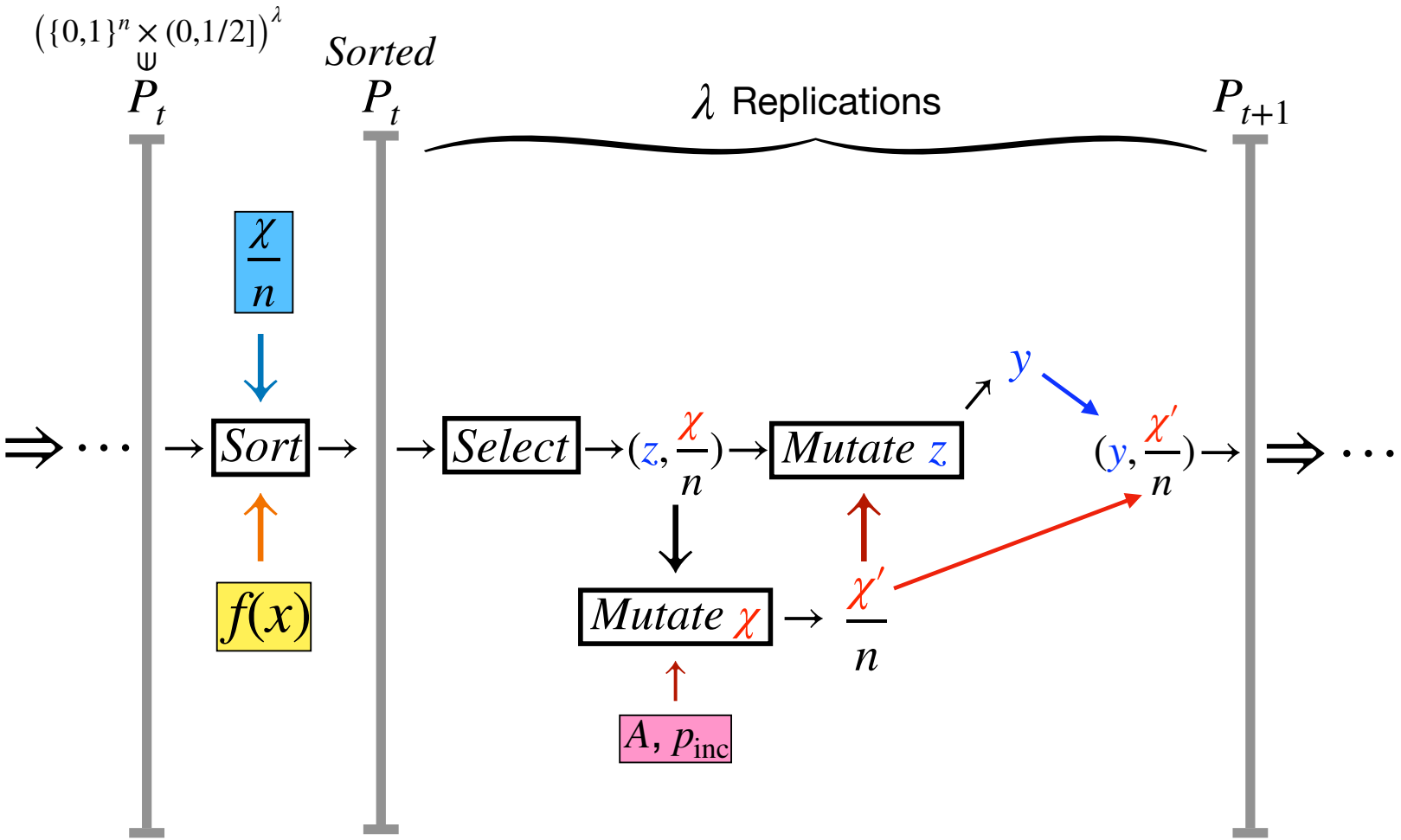
- SPARSELOCALOPT $\Rightarrow$ a kind of fitness landscapes with *sparse deceptive regions* (local optima) and *dense fitness valleys* (Dang et al., 2021b).
- *Non-linear non-elitist selection* and *sufficiently high mutation rate* (Dang et al., 2021b) $\Rightarrow$
  - Sparse local optimal individuals $\Rightarrow$ higher chance to be selected but only survive a small percentage of such individuals after mutation;
  - Dense fitness valley individuals $\Rightarrow$ less chance of being selected but can have higher chance of surviving mutation.

Motivation: Sparse Local Optima - Problems



- However,
  - We need to know the sparsity of local optima to set the mutation rate;
  - Fitness functions could contain several local optimums with different sparsities.

A Framework of Self-adaptive EAs



Components of MOSA-EA

- MOSA-EA:
  - Multi-objective sorting mechanism $\Rightarrow$ Strict non-dominated Pareto fronts
  - (μ, λ) selection $\Rightarrow$ from sorted population.
  - Self-adapting mutation rate strategy $\Rightarrow$ New mutation rate χ' is
    - $A\chi$ with probability p_{inc};
    - χ/A otherwise.
  - Bit-wise mutation operator $\Rightarrow$ New population.

Components of MOSA-EA

- MOSA-EA:
  - Multi-objective sorting mechanism $\Rightarrow$ Strict non-dominated Pareto fronts
  - (μ, λ) selection $\Rightarrow$ from sorted population.
  - Self-adapting mutation rate strategy $\Rightarrow$ New mutation rate χ' is
    - $A\chi$ with probability p_{inc};
    - χ/A otherwise.
  - Bit-wise mutation operator $\Rightarrow$ New population.

Components of MOSA-EA

- MOSA-EA:
  - Multi-objective sorting mechanism $\Rightarrow$ Strict non-dominated Pareto fronts
  - (μ, λ) selection $\Rightarrow$ from sorted population.
  - Self-adapting mutation rate strategy $\Rightarrow$ New mutation rate χ' is
    - $A\chi$ with probability p_{inc};
    - χ/A otherwise.
  - Bit-wise mutation operator $\Rightarrow$ New population.

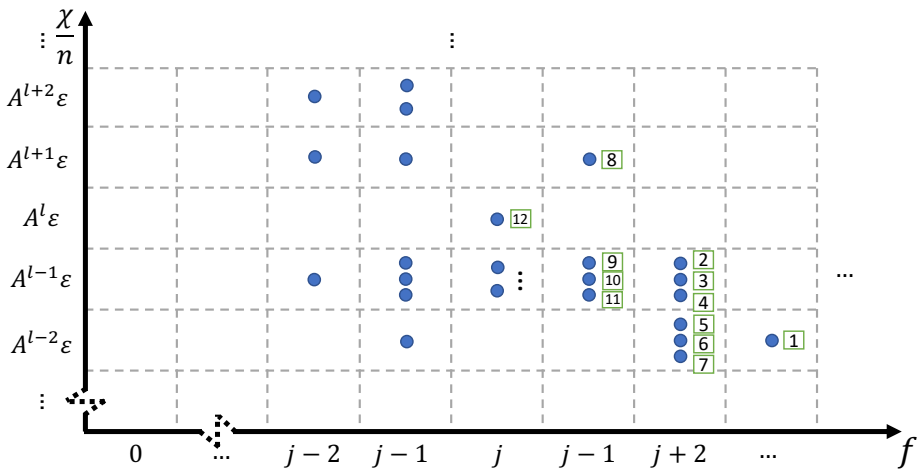


Figure: Fitness-first sorting (Case and Lehre, 2020)

Components of MOSA-EA

- MOSA-EA:
  - Multi-objective sorting mechanism $\Rightarrow$ Strict non-dominated Pareto fronts
  - (μ, λ) selection $\Rightarrow$ from sorted population.
  - Self-adapting mutation rate strategy $\Rightarrow$ New mutation rate χ' is
    - $A\chi$ with probability p_{inc};
    - χ/A otherwise.
  - Bit-wise mutation operator $\Rightarrow$ New population.

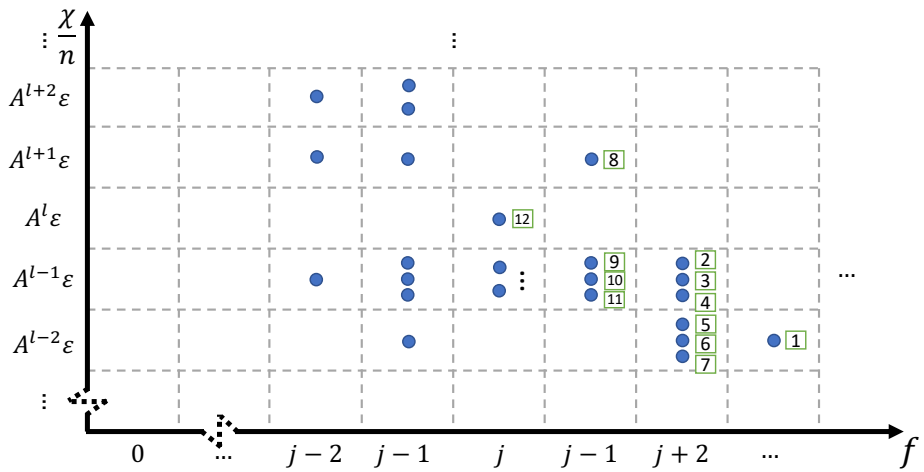


Figure: Fitness-first sorting (Case and Lehre, 2020)

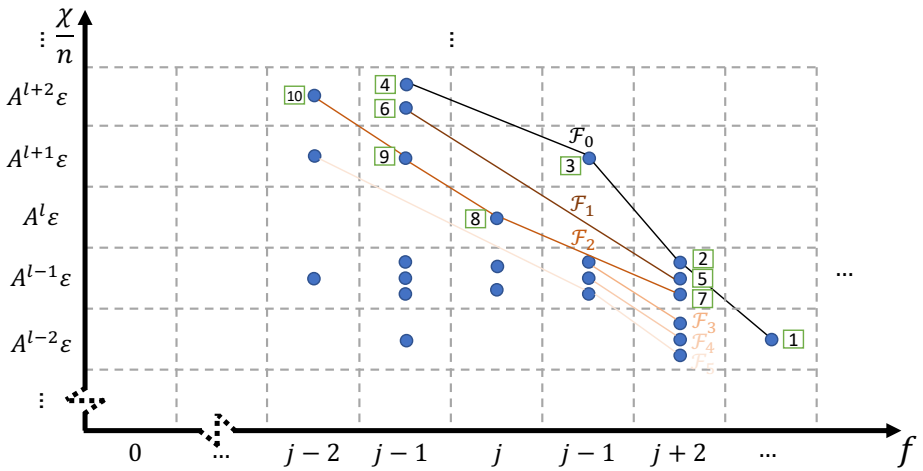


Figure: Multi-objective sorting

Components of MOSA-EA

- MOSA-EA:
  - Multi-objective sorting mechanism $\Rightarrow$ Strict non-dominated Pareto fronts
  - (μ, λ) selection $\Rightarrow$ from sorted population.
  - Self-adapting mutation rate strategy $\Rightarrow$ New mutation rate χ' is
    - $A\chi$ with probability p_{inc};
    - χ/A otherwise.
  - Bit-wise mutation operator $\Rightarrow$ New population.

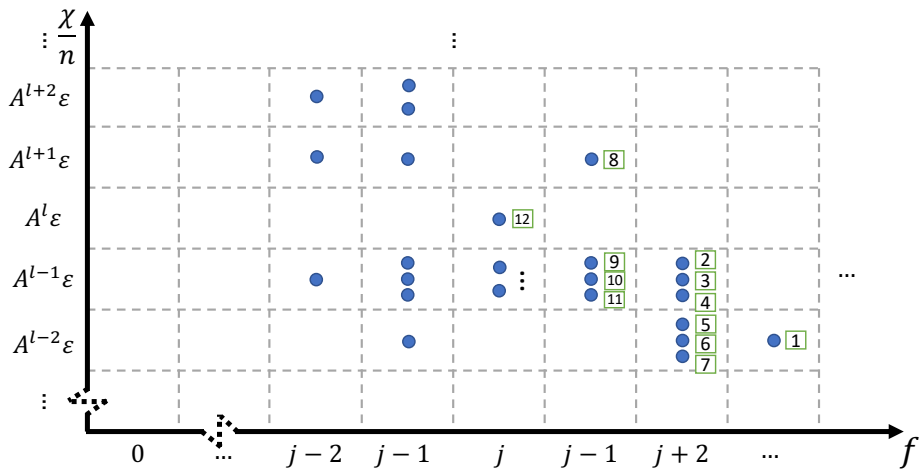


Figure: Fitness-first sorting (Case and Lehre, 2020)

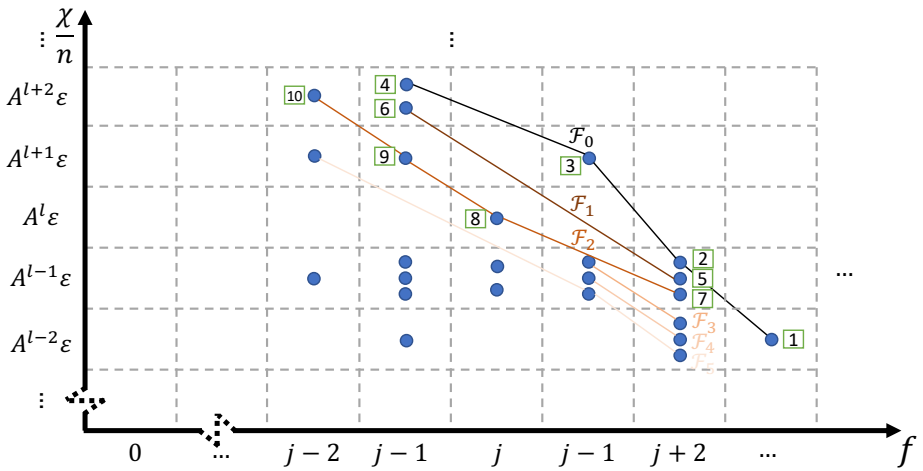


Figure: Multi-objective sorting

Components of MOSA-EA

- MOSA-EA:
  - Multi-objective sorting mechanism $\Rightarrow$ Strict non-dominated Pareto fronts
  - (μ, λ) selection $\Rightarrow$ from sorted population.
  - Self-adapting mutation rate strategy $\Rightarrow$ New mutation rate χ' is
    - $A\chi$ with probability p_{inc};
    - χ/A otherwise.
  - Bit-wise mutation operator $\Rightarrow$ New population.

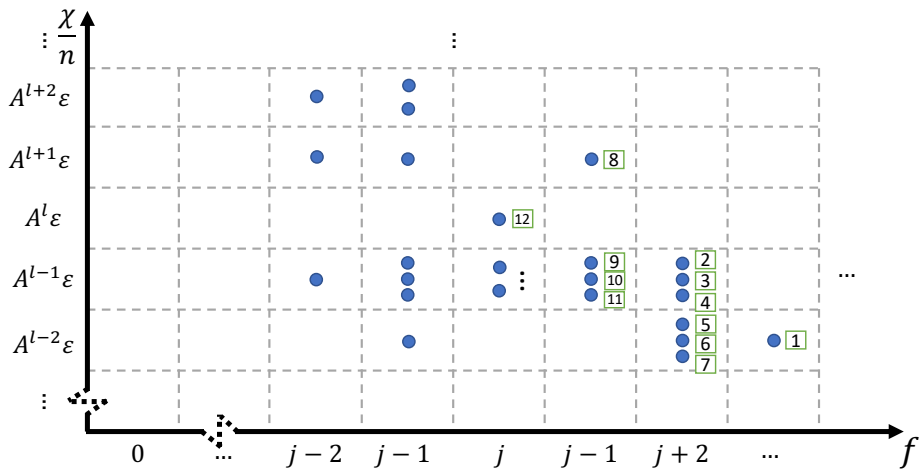


Figure: Fitness-first sorting (Case and Lehre, 2020)

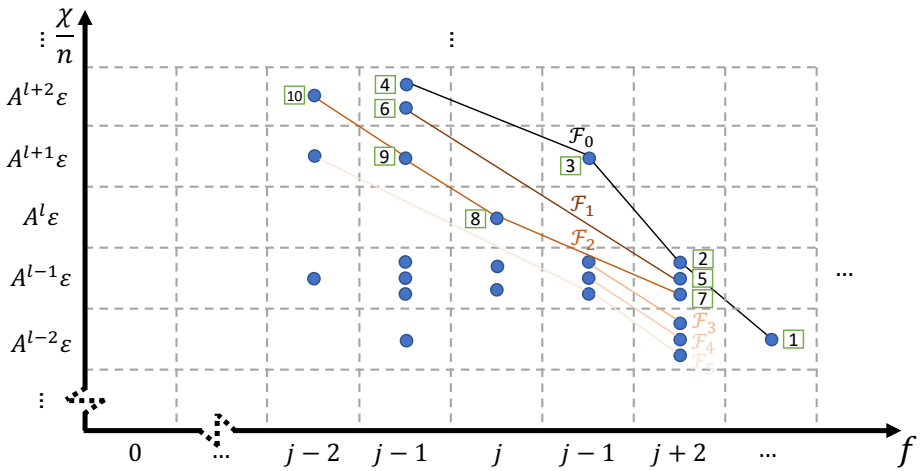


Figure: Multi-objective sorting

Components of MOSA-EA

- MOSA-EA:
  - Multi-objective sorting mechanism $\Rightarrow$ Strict non-dominated Pareto fronts
  - (μ, λ) selection $\Rightarrow$ from sorted population.
  - Self-adapting mutation rate strategy $\Rightarrow$ New mutation rate χ' is
    - $A\chi$ with probability p_{inc};
    - χ/A otherwise.
  - Bit-wise mutation operator $\Rightarrow$ New population.

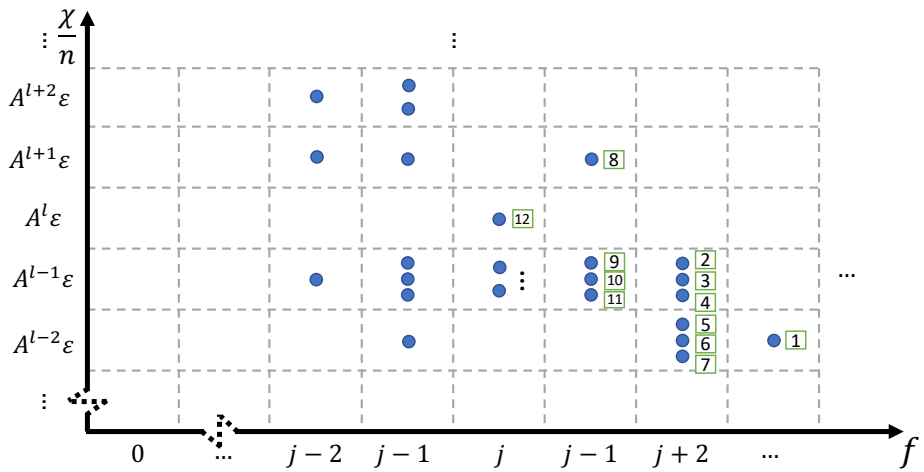


Figure: Fitness-first sorting (Case and Lehre, 2020)

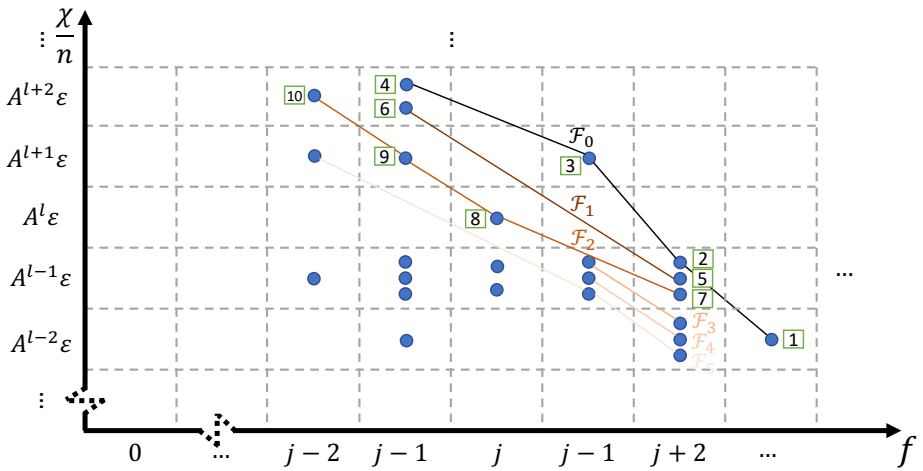


Figure: Multi-objective sorting

Theoretical Study on MOSA-EA

Lehre and Qin (2022b) mathematical analysed MOSA-EA on an artificial two-peak function $\textsc{PeakedLO}_{m,k}$ with tunable sparsity:

- MOSA-EA can efficiently escape an artificial local optimum with unknown sparsity.
- Other fixed mutation rate EAs fail.

Algorithm	$\textsc{PeakedLO}_{m,k}$	Runtime T
$(\mu+\lambda)$ EA	Any $k \leq n$ and $k, m \in \Omega(n)$	$\Pr(T \leq e^{cn}) \leq e^{-\Omega(n)}$
(μ,λ) EA	Any $k \leq n$ and $k, m \in \Omega(n)$	$\Pr(T \leq e^{cn}) \leq e^{-\Omega(n)}$ †
2-tour. EA	Any $k < (\ln(3/2)-\delta)\,n$ and $k, m \in \Omega(n)$	$\Pr(T \leq e^{cn}) \leq e^{-\Omega(\lambda)}$
(μ,λ) MOSA-EA	Any $n \geq k \in \Omega(n)$, $\lceil m \rceil < 2A\,(1+\ln(p_{\text{inc}})/\ln(\alpha_0) - o(1))\,k$ ‡	$E[T] = O\left(n^2 \log(n)\right)$

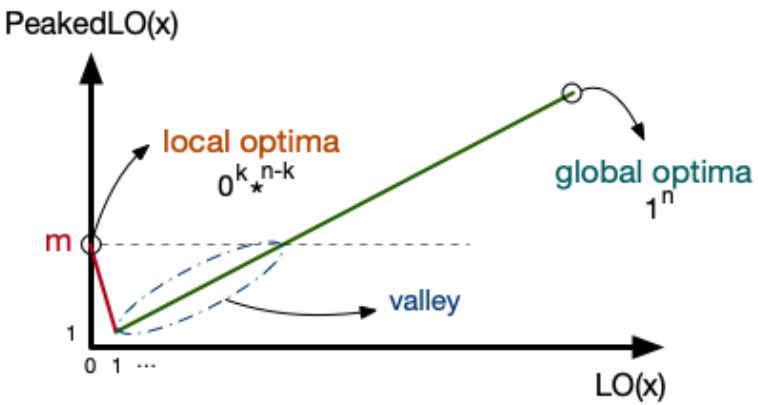
*With the initial population $P_0 = \{0^k *^{n-k}\}^\lambda$

†$\lambda, \mu \in \text{poly}(n)$

‡For some constants $p_{\text{inc}} < 2/5$, $\alpha \geq 4$, $A > 1$ based on restrictions in Theorem 5 of (Lehre and Qin, 2022b)

Theoretical Study on MOSA-EA

Lehre and Qin (2022b) mathematical analysed MOSA-EA on an artificial two-peak function $\textsc{PeakedLO}_{m,k}$ with tunable sparsity:



$$\textsc{PeakedLO}_{m,k}(x) = \begin{cases} m & \text{if } x = \{0\}^k *^{n-k} \\ \mathrm{LO}(x) & \text{otherwise.} \end{cases}$$

- MOSA-EA can efficiently escape an artificial local optimum with unknown sparsity.
- Other fixed mutation rate EAs fail.

Algorithm	$\textsc{PeakedLO}_{m,k}$	Runtime T
$(\mu+\lambda)$ EA	Any $k \leq n$ and $k, m \in \Omega(n)$	$\Pr(T \leq e^{cn}) \leq e^{-\Omega(n)}$
(μ,λ) EA	Any $k \leq n$ and $k, m \in \Omega(n)$	$\Pr(T \leq e^{cn}) \leq e^{-\Omega(n)}$ †
2-tour. EA	Any $k < (\ln(3/2) - \delta)\, n$ and $k, m \in \Omega(n)$	$\Pr(T \leq e^{cn}) \leq e^{-\Omega(\lambda)}$
(μ,λ) MOSA-EA	Any $n \geq k \in \Omega(n)$, $\lceil m \rceil < 2A\,(1 + \ln(p_{\text{inc}})/\ln(\alpha_0) - o(1))\, k$ ‡	$E[T] = O\left(n^2 \log(n)\right)$

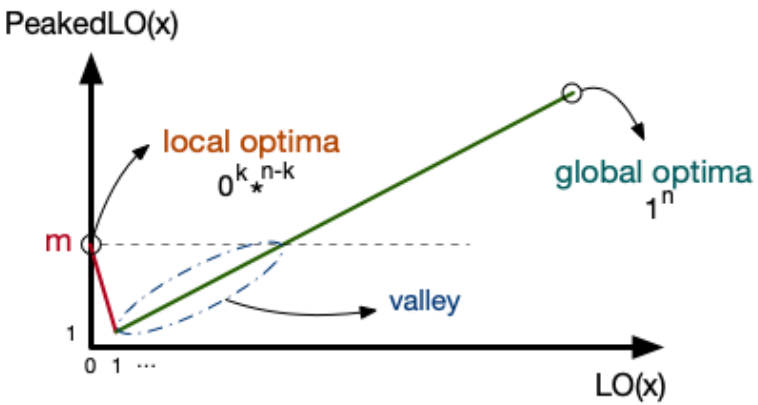
*With the initial population $P_0 = \{0^k *^{n-k}\}^\lambda$

†$\lambda, \mu \in \text{poly}(n)$

‡For some constants $p_{\text{inc}} < 2/5$, $\alpha \geq 4$, $A > 1$ based on restrictions in Theorem 5 of (Lehre and Qin, 2022b)

Theoretical Study on MOSA-EA

Lehre and Qin (2022b) mathematical analysed MOSA-EA on an artificial two-peak function $\textsc{PeakedLO}_{m,k}$ with tunable sparsity:



$$\textsc{PeakedLO}_{m,k}(x) = \begin{cases} m & \text{if } x = \{0\}^k *^{n-k} \\ \text{LO}(x) & \text{otherwise.} \end{cases}$$

- MOSA-EA can efficiently escape an artificial local optimum with unknown sparsity.
- Other fixed mutation rate EAs fail.

Algorithm	$\text{PeakedLO}_{m,k}$	Runtime T
$(\mu+\lambda)$ EA	Any $k \leq n$ and $k, m \in \Omega(n)$	$\Pr(T \leq e^{cn}) \leq e^{-\Omega(n)}$
(μ,λ) EA	Any $k \leq n$ and $k, m \in \Omega(n)$	$\Pr(T \leq e^{cn}) \leq e^{-\Omega(n)}$ †
2-tour. EA	Any $k < (\ln(3/2) - \delta)\, n$ and $k, m \in \Omega(n)$	$\Pr(T \leq e^{cn}) \leq e^{-\Omega(\lambda)}$
(μ,λ) MOSA-EA	Any $n \geq k \in \Omega(n)$, $\lceil m \rceil < 2A\,(1 + \ln(p_{\text{inc}})/\ln(\alpha_0) - o(1))\, k$ ‡	$E[T] = O\left(n^2 \log(n)\right)$

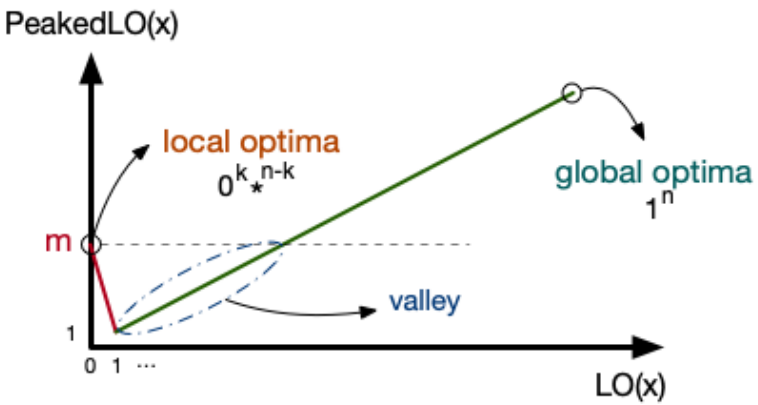
*With the initial population $P_0 = \{0^k *^{n-k}\}^\lambda$

†$\lambda, \mu \in \text{poly}(n)$

‡For some constants $p_{\text{inc}} < 2/5$, $\alpha \geq 4$, $A > 1$ based on restrictions in Theorem 5 of (Lehre and Qin, 2022b)

Theoretical Study on MOSA-EA

Lehre and Qin (2022b) mathematical analysed MOSA-EA on an artificial two-peak function $\textsc{PeakedLO}_{m,k}$ with tunable sparsity:



$$\textsc{PeakedLO}_{m,k}(x) = \begin{cases} m & \text{if } x = \{0\}^k *^{n-k} \\ \text{LO}(x) & \text{otherwise.} \end{cases}$$

- MOSA-EA can efficiently escape an artificial local optimum with unknown sparsity.
- Other fixed mutation rate EAs fail.

Algorithm	$\text{PeakedLO}_{m,k}$	Runtime T
$(\mu+\lambda)$ EA	Any $k \leq n$ and $k, m \in \Omega(n)$	$\Pr(T \leq e^{cn}) \leq e^{-\Omega(n)}$
(μ,λ) EA	Any $k \leq n$ and $k, m \in \Omega(n)$	$\Pr(T \leq e^{cn}) \leq e^{-\Omega(n)}$ †
2-tour. EA	Any $k < (\ln(3/2)-\delta)\,n$ and $k, m \in \Omega(n)$	$\Pr(T \leq e^{cn}) \leq e^{-\Omega(\lambda)}$
(μ,λ) MOSA-EA	Any $n \geq k \in \Omega(n)$, $\lceil m \rceil < 2A\,(1+\ln(p_{\text{inc}})/\ln(\alpha_0) - o(1))\,k$ ‡	$E[T] = O\left(n^2 \log(n)\right)$

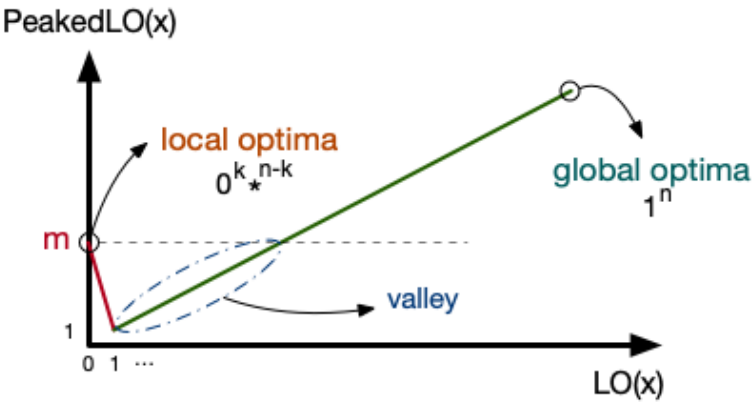
*With the initial population $P_0 = \{0^k *^{n-k}\}^\lambda$

†$\lambda, \mu \in \text{poly}(n)$

‡For some constants $p_{\text{inc}} < 2/5$, $\alpha \geq 4$, $A > 1$ based on restrictions in Theorem 5 of (Lehre and Qin, 2022b)

Theoretical Study on MOSA-EA

Lehre and Qin (2022b) mathematical analysed MOSA-EA on an artificial two-peak function $\textsc{PeakedLO}_{m,k}$ with tunable sparsity:



$$\textsc{PeakedLO}_{m,k}(x) = \begin{cases} m & \text{if } x = \{0\}^k *^{n-k} \\ \text{LO}(x) & \text{otherwise.} \end{cases}$$

- MOSA-EA can efficiently escape an artificial local optimum with unknown sparsity.
- Other fixed mutation rate EAs fail.

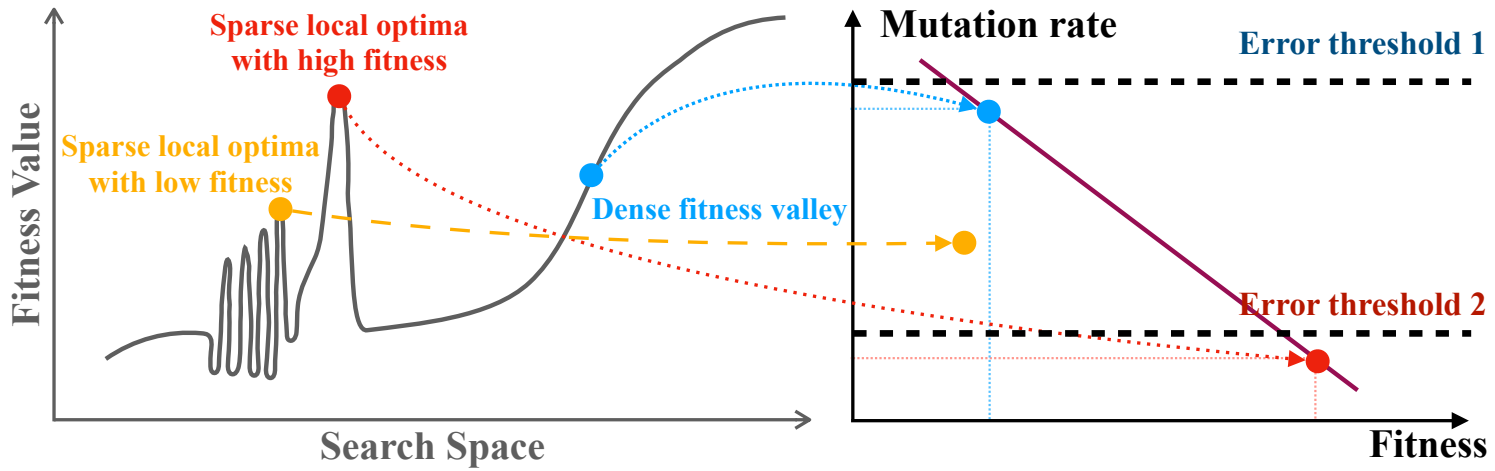
Algorithm	**PeakedLO**$_{m,k}$	**Runtime** T
$(\mu+\lambda)$ EA	Any $k \leq n$ and $k, m \in \Omega(n)$	$\Pr(T \leq e^{cn}) \leq e^{-\Omega(n)}$
(μ,λ) EA	Any $k \leq n$ and $k, m \in \Omega(n)$	$\Pr(T \leq e^{cn}) \leq e^{-\Omega(n)}$ †
2-tour. EA	Any $k < (\ln(3/2) - \delta)\, n$ and $k, m \in \Omega(n)$	$\Pr(T \leq e^{cn}) \leq e^{-\Omega(\lambda)}$
(μ,λ) MOSA-EA	Any $n \geq k \in \Omega(n)$, $\lceil m \rceil < 2A\,(1 + \ln(p_{\text{inc}})/\ln(\alpha_0) - o(1))\, k$ ‡	$E[T] = O\left(n^2 \log(n)\right)$

*With the initial population $P_0 = \{0^k *^{n-k}\}^\lambda$

†$\lambda, \mu \in \text{poly}(n)$

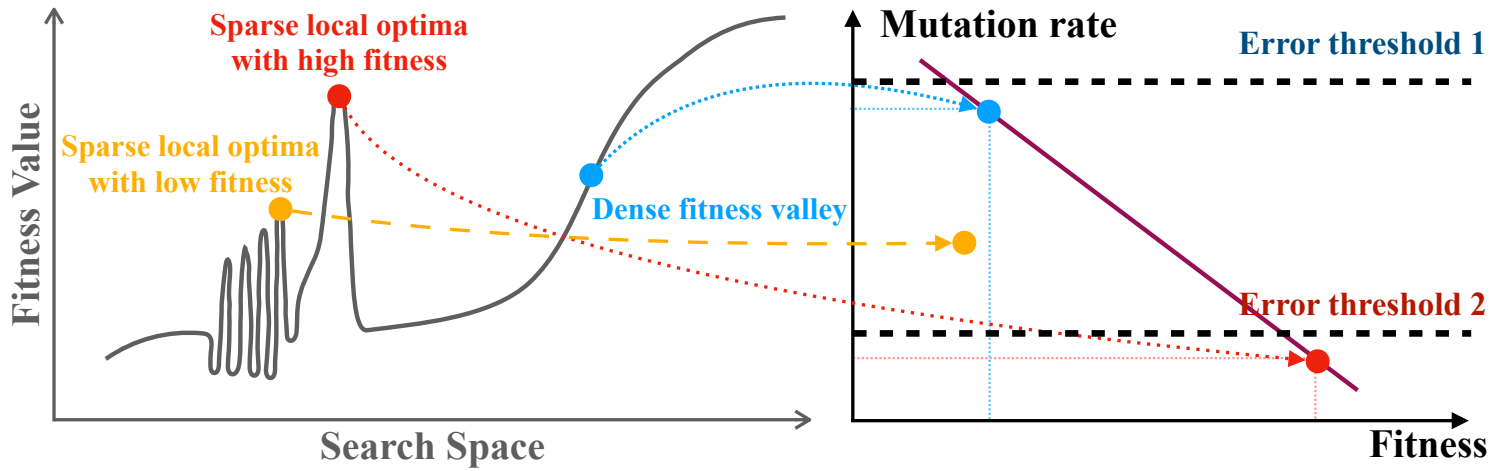
‡For some constants $p_{\text{inc}} < 2/5$, $\alpha \geq 4$, $A > 1$ based on restrictions in Theorem 5 of (Lehre and Qin, 2022b)

Proof Idea (Insight)



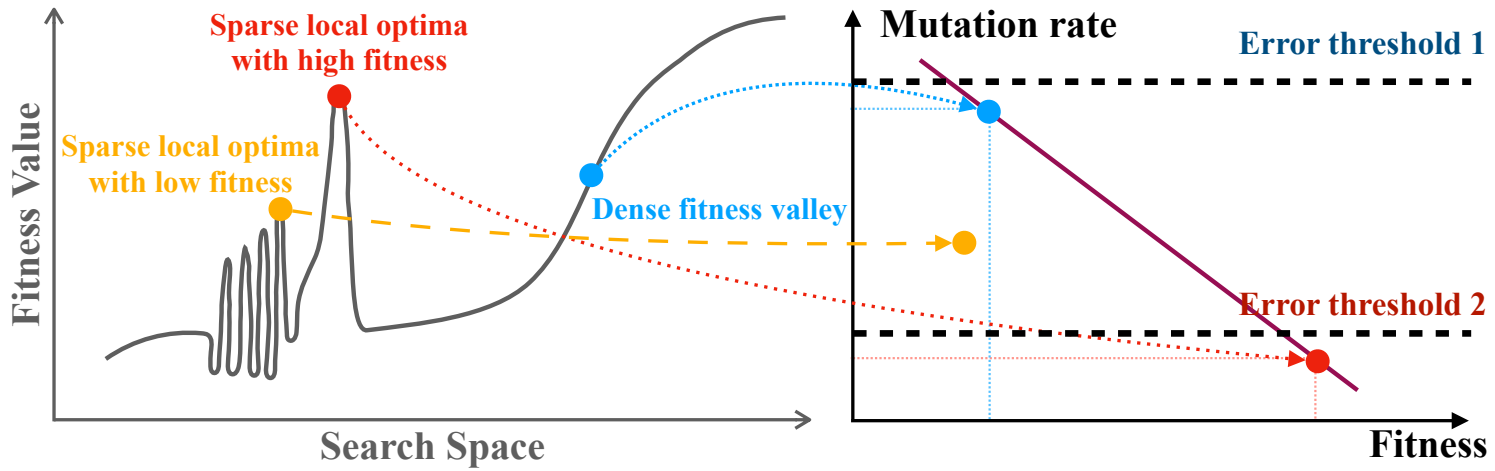
- The error thresholds for sparse and dense regions are different.
- MOSA-EA maximises fitness and mutation rate on Pareto fronts.
- For each individual, the mutation rate will be closed to its error threshold.
- Individuals with mutation rates larger than error thresholds will "vanish" in the next generation.

Proof Idea (Insight)



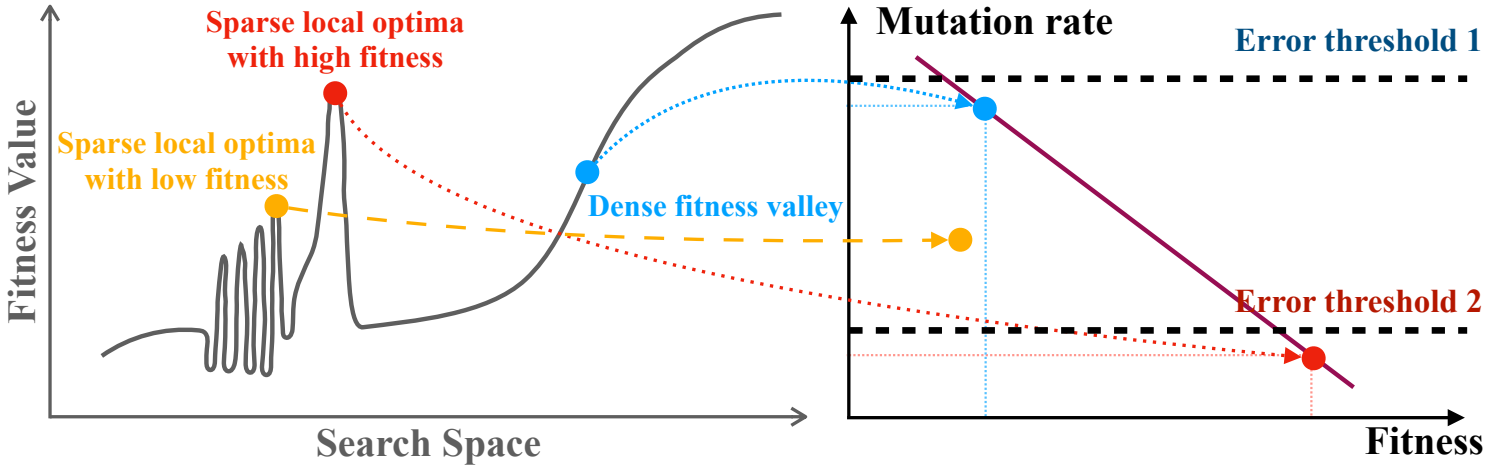
- The error thresholds for sparse and dense regions are different.
- MOSA-EA maximises fitness and mutation rate on Pareto fronts.
- For each individual, the mutation rate will be closed to its error threshold.
- Individuals with mutation rates larger than error thresholds will "vanish" in the next generation.

Proof Idea (Insight)



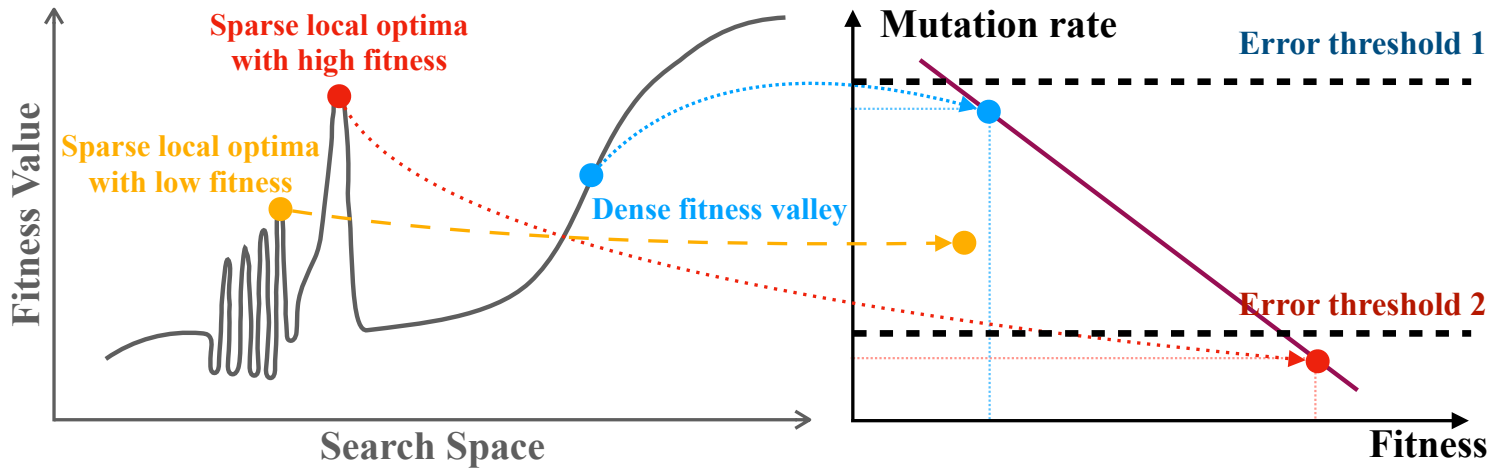
- The error thresholds for sparse and dense regions are different.
- MOSA-EA maximises fitness and mutation rate on Pareto fronts.
- For each individual, the mutation rate will be closed to its error threshold.
- Individuals with mutation rates larger than error thresholds will "vanish" in the next generation.

Proof Idea (Insight)



- The error thresholds for sparse and dense regions are different.
- MOSA-EA maximises fitness and mutation rate on Pareto fronts.
- For each individual, the mutation rate will be closed to its error threshold.
- Individuals with mutation rates larger than error thresholds will "vanish" in the next generation.

Proof Idea (Insight)



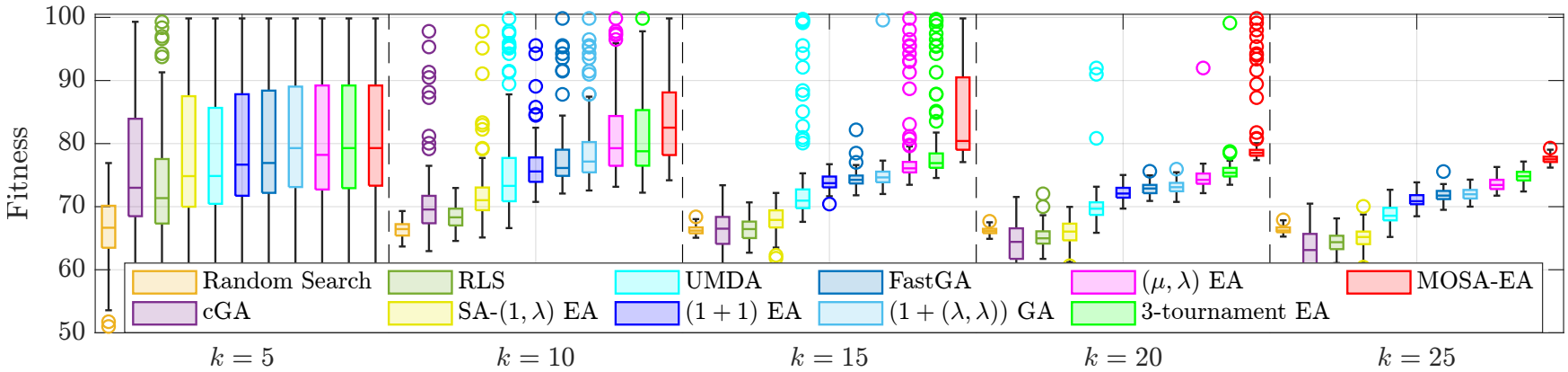
- The error thresholds for sparse and dense regions are different.
- MOSA-EA maximises fitness and mutation rate on Pareto fronts.
- For each individual, the mutation rate will be closed to its error threshold.
- Individuals with mutation rates larger than error thresholds will "vanish" in the next generation.

Empirical Study on Combinatorial Optimisation Problems

Contain many local optimums

⇑

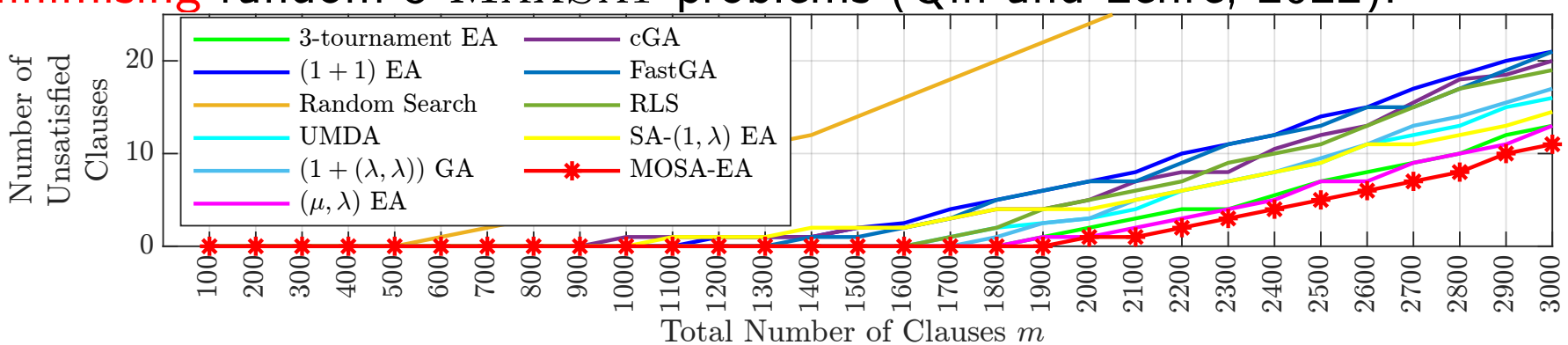
Maximising random NK-LANDSCAPE problems (Qin and Lehre, 2022):



In the complexity class NP

⇑

Minimising random 5-MAXSAT problems (Qin and Lehre, 2022):



Summary of MOSA-EA

- **Novelty:**
  - MOSA-EA encodes parameters into individuals.
  - MOSA-EA treats parameter control as another objective.
- **Significance:**
  - MOSA-EA can escape local optima with unknown sparsity
  - MOSA-EA can self-adapt mutation rate to the noise level in noisy optimisation.
  - MOSA-EA can outperform other EAs on complex optimisation problems.
  - MOSA-EA is free to set mutation rate.

Summary of MOSA-EA

- **Novelty:**
  - MOSA-EA encodes parameters into individuals.
  - MOSA-EA treats parameter control as another objective.
- **Significance:**
  - MOSA-EA can escape local optima with unknown sparsity
  - MOSA-EA can self-adapt mutation rate to the noise level in noisy optimisation.
  - MOSA-EA can outperform other EAs on complex optimisation problems.
  - MOSA-EA is free to set mutation rate.

Summary of MOSA-EA

- **Novelty:**
  - MOSA-EA encodes parameters into individuals.
  - MOSA-EA treats parameter control as another objective.
- **Significance:**
  - MOSA-EA can escape local optima with unknown sparsity
  - MOSA-EA can self-adapt mutation rate to the noise level in noisy optimisation.
  - MOSA-EA can outperform other EAs on complex optimisation problems.
  - MOSA-EA is free to set mutation rate.

Summary of MOSA-EA

- **Novelty:**
  - MOSA-EA encodes parameters into individuals.
  - MOSA-EA treats parameter control as another objective.
- **Significance:**
  - MOSA-EA can escape local optima with unknown sparsity
  - MOSA-EA can self-adapt mutation rate to the noise level in noisy optimisation.
  - MOSA-EA can outperform other EAs on complex optimisation problems.
  - MOSA-EA is free to set mutation rate.

Summary of MOSA-EA

- **Novelty:**
  - MOSA-EA encodes parameters into individuals.
  - MOSA-EA treats parameter control as another objective.
- **Significance:**
  - MOSA-EA can escape local optima with unknown sparsity
  - MOSA-EA can self-adapt mutation rate to the noise level in noisy optimisation.
  - MOSA-EA can outperform other EAs on complex optimisation problems.
  - MOSA-EA is free to set mutation rate.

Summary of MOSA-EA

- **Novelty:**
  - MOSA-EA encodes parameters into individuals.
  - MOSA-EA treats parameter control as another objective.
- **Significance:**
  - MOSA-EA can escape local optima with unknown sparsity
  - MOSA-EA can self-adapt mutation rate to the noise level in noisy optimisation.
  - MOSA-EA can outperform other EAs on complex optimisation problems.
  - MOSA-EA is free to set mutation rate.

Summary of MOSA-EA

- **Novelty:**
  - MOSA-EA encodes parameters into individuals.
  - MOSA-EA treats parameter control as another objective.
- **Significance:**
  - MOSA-EA can escape local optima with unknown sparsity
  - MOSA-EA can self-adapt mutation rate to the noise level in noisy optimisation.
  - MOSA-EA can outperform other EAs on complex optimisation problems.
  - MOSA-EA is free to set mutation rate.

Summary of MOSA-EA

- **Novelty:**
  - MOSA-EA encodes parameters into individuals.
  - MOSA-EA treats parameter control as another objective.
- **Significance:**
  - MOSA-EA can escape local optima with unknown sparsity
  - MOSA-EA can self-adapt mutation rate to the noise level in noisy optimisation.
  - MOSA-EA can outperform other EAs on complex optimisation problems.
  - MOSA-EA is free to set mutation rate.

Thank You

Xiaoyu QIN 秦霄羽

xxq896@cs.bham.ac.uk

www.cs.bham.ac.uk/~xxq896/

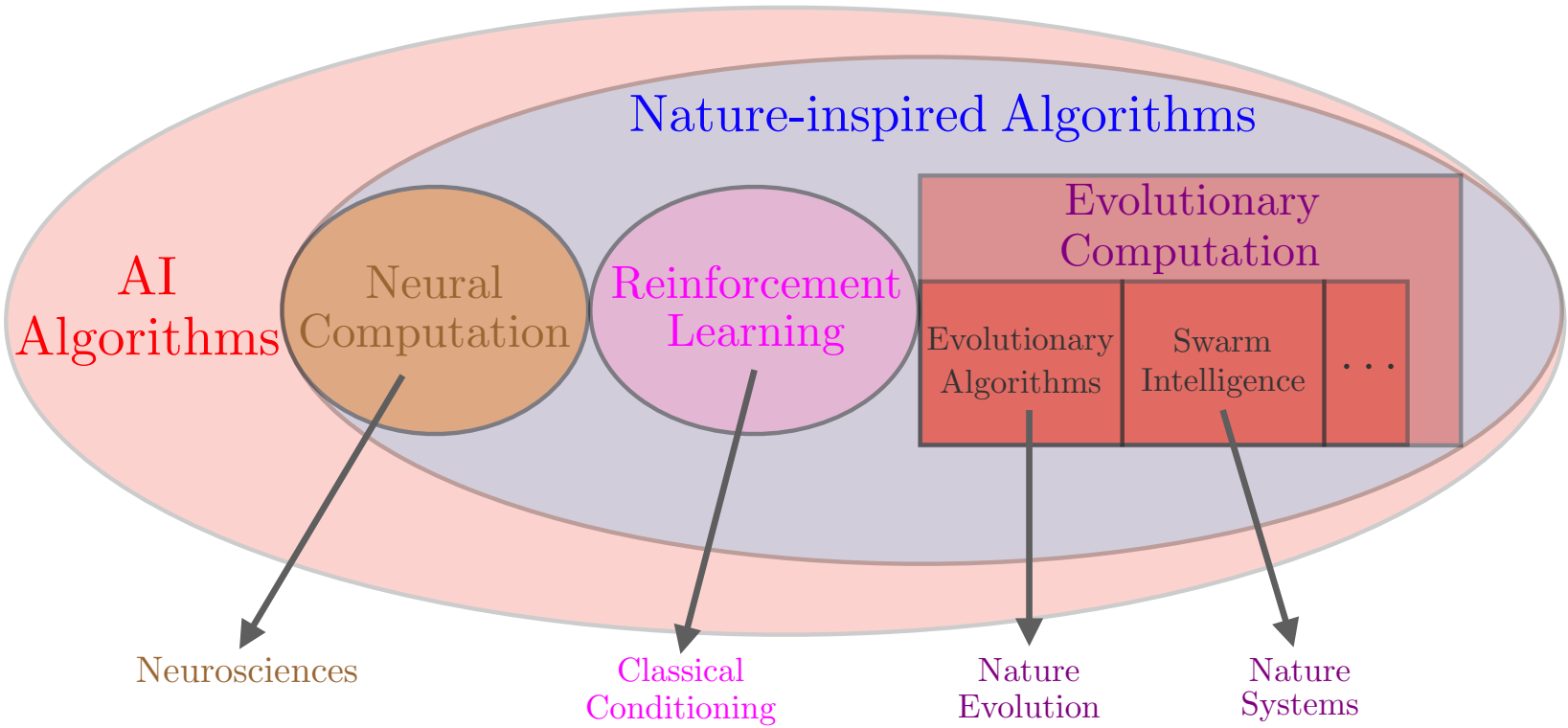
Theory of Evolutionary Computation Group

School of Computer Science

University of Birmingham

United Kingdom

Appendix: Broader Evolutionary Computation



References

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., Eremeev, A., and Lehre, P. K. (2021a). Escaping Local Optima with Non-Elitist Evolutionary Algorithms. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 12275–12283.

Dang, D.-C., Eremeev, A., and Lehre, P. K. (2021b). Non-Elitist Evolutionary Algorithms Excel in Fitness Landscapes with Sparse Deceptive Regions and Dense Valleys. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '21, pages 1133–1141. Association for Computing Machinery.

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. (2020). A gentle introduction to theory (for non-theoreticians). In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*, pages 373–403. ACM.

Doerr, B. (2022). Does Comma Selection Help to Cope with Local Optima? *Algorithmica*, 84(6):1659–1693.

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.

Droste, S., Jansen, T., and Wegener, I. (2006). Upper and Lower Bounds for Randomized Search Heuristics in Black-Box Optimization. *Theory of Computing Systems*, 39(4):525–544.

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

Jagerskupper, J. and Storch, T. (2007). When the Plus Strategy Outperforms the Comma Strategyand When Not. In *2007 IEEE Symposium on Foundations of Computational Intelligence*, pages 25–32. IEEE.

Lehre, P. K. and Qin, X. (2021). More Precise Runtime Analyses of Non-elitist EAs in Uncertain Environments. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1160–1168. ACM.

Lehre, P. K. and Qin, X. (2022a). More Precise Runtime Analyses of Non-elitist Evolutionary Algorithms in Uncertain Environments. *Algorithmica*.

Lehre, P. K. and Qin, X. (2022b). 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.

Lehre, P. K. and Qin, X. (2023a). Self-Adaptation Can Help Evolutionary Algorithms Track Dynamic Optima. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '23, pages 1619–1627. Association for Computing Machinery.

Lehre, P. K. and Qin, X. (2023b). Self-Adaptation Can Improve the Noise-Tolerance of Evolutionary Algorithms. In *Proceedings of the 17th ACM/SIGEVO Conference on Foundations of Genetic Algorithms*, FOGA '23, pages 105–116. 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.

Sudholt, D. (2013). A New Method for Lower Bounds on the Running Time of Evolutionary Algorithms. *IEEE Transactions on Evolutionary Computation*, 17(3):418–435.