

Theory of Competitive Co-evolutionary Algorithms^{1 2 3}

Per Kristian Lehre

School of Computer Science, University of Birmingham
Birmingham, United Kingdom
p.k.lehre@bham.ac.uk

San Jose, Costa Rica
July 2026



¹For the latest version of these slides, please see <https://pklehre.github.io/EC-Theory-UoB/CoEvolutionTutorial/>.

²These slides are based on a PPSN2014 tutorial by Mario Alejandro Hevia Fajardo and Per Kristian Lehre.

³Research supported by a Turing AI Acceleration Fellowship (EPSRC grant ref EP/V025562/1).



Optimisation



Decision space $\mathcal{X}$

x_1 x_i x_N

$f(x_i)$

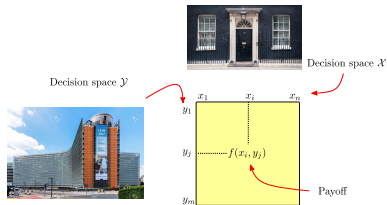
Objective value

Assumptions

- ▶ $\mathcal{X}$ large
- ▶ f has some “structure”

Goal: Find optimal solution

Game Theory

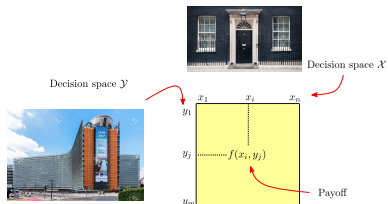


Assumptions

- $\mathcal{X}$ and $\mathcal{Y}$ small

Goal: Nash equilibrium

Game Theory + Optimisation = Adversarial Optimisation



Assumptions

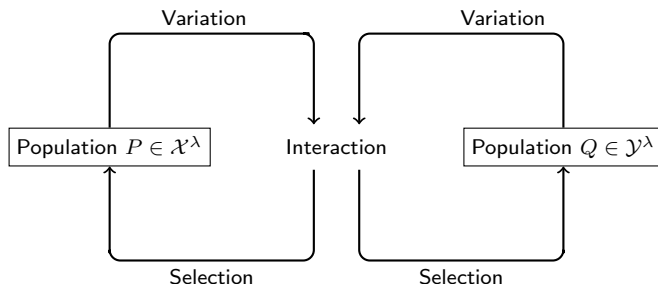
- ▶ $\mathcal{X}$ and $\mathcal{Y}$ large **and discrete**
- ▶ f has some "structure"
- ▶ **oracle access to f**

Goal: Nash equilibrium

Classical methods fail

- ▶ Gradient descent/ascent
- ▶ Best response dynamics
- ▶ Multiplicative weight update
- ▶ Mathematical programming
- ▶ Lemke-Howson

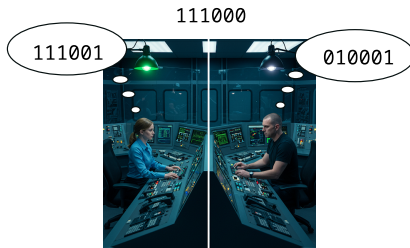
Competitive Coevolutionary Algorithms



Competitive coevolutionary algorithms evolve competing populations, where each population adapts in response to the other, often used to model adversarial scenarios.

- “Payoff function” $g : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$, determining outcome of interactions

Diagonal Game (two player, zero-sum, normal form game)



- ▶ Alice and Bob placed in separate control rooms with no communication.
- ▶ Each control room has n on/off-buttons. Each button has an unknown “correct” setting.
- ▶ In each game, Alice and Bob pushes buttons, and whoever has the most correct buttons wins (green lamp) with ties in Alice’s favour. NE is to push all buttons correctly.
- ▶ Exponentially large strategy space of $N = 2^n$ making exhaustive search impractical.
- ▶ From info theory, $\Omega(\log N)$ rounds needed. Does any natural learning dynamic find NE in $\text{polylog } N$ rounds?

Example: Sorting Networks

Goal: sort a sequence of objects (e.g., numerical or alphanumeric)

Sorting network consisting of:

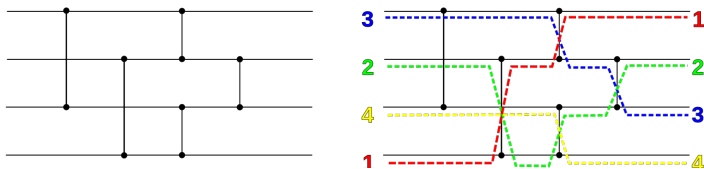
- ▶ Horizontal lines = elements in the list
- ▶ Comparison components, two input, two output
- ▶ If inputs are in incorrect order, swap them

Example: Sorting Networks

Goal: sort a sequence of objects (e.g., numerical or alphanumeric)

Sorting network consisting of:

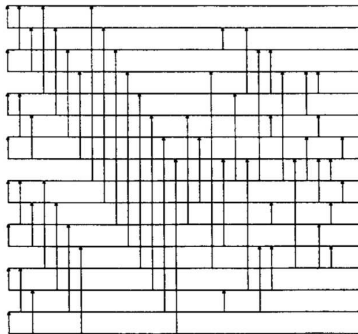
- ▶ Horizontal lines = elements in the list
- ▶ Comparison components, two input, two output
- ▶ If inputs are in incorrect order, swap them



Example of a sorting network sorting.

Co-Evolving Sorting Networks [Hillis, 1990]⁴

Hand designed

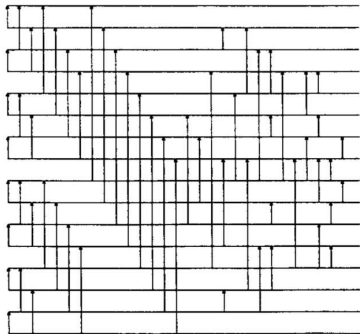


- ▶ Best network ($n = 16$) in 1969
 - ▶ Green – 60 comparisons

⁴First 32 comparisons initialised to common pattern shared by known sorting networks.

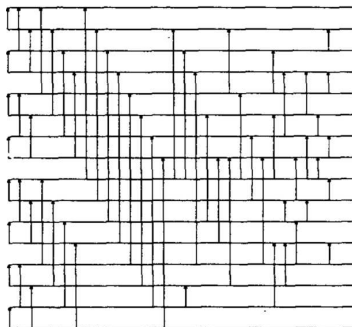
Co-Evolving Sorting Networks [Hillis, 1990]⁴

Hand designed



- ▶ Best network ($n = 16$) in 1969
 - ▶ Green – 60 comparisons

(Co-)evolved



- ▶ Networks vs Test Cases
- ▶ Best co-evolved network ($n = 16$): 61 comparisons

⁴First 32 comparisons initialised to common pattern shared by known sorting networks.

Outline

Introduction

Optimisation + Game Theory = Adversarial Optimisation

Games and Solution Concepts

Basic Concepts

Normal form games, Extensive form games

Solution concepts: Nash equilibrium, maxmin

Computational Models for Adversarial Optimisation

A Black-Box Scenario

Fundamental Limitations of Algorithms and a Niche for Co-Evolution

Design of Co-Evolutionary Algorithms

Population structure and interaction schemes

Co-Evolutionary Pathologies

Population Archives and Monotone Solution Concepts

Replacing Monotone Progress by Positive Drift

Runtime Analysis of Co-Evolutionary Algorithms

Drift Analysis

Level-based Analysis

Conclusion

Why Game Theory?

Game Theory

- ▶ Studies strategic decision-making.
- ▶ Helps predict outcomes in competitive and cooperative scenarios

Everyday Relevance

- ▶ **Negotiating Prices:** Buyers and sellers strategize.
- ▶ **Voting:** Analyzes voting strategies.
- ▶ **Business:** Companies predict competitors' moves.
- ▶ **Social:** Cooperation and competition in daily life.

Real-World examples

- ▶ Pricing strategies among competing businesses.
- ▶ Negotiations in international trade agreements.
- ▶ Strategies in sports and games like chess and poker.

Key Concepts of a Game

Players

Who is interacting?

Strategies

What are their options? Who acts first?

- ▶ Pure strategies: Deterministic
- ▶ Mixed strategies: Probabilistic

Payoffs

How do choices lead to outcomes? What do players prefer?

Information/Beliefs

What do players know or believe? What do they see before acting?

Rationality

How do players think?

Representations - Normal-Form Games

- ▶ A representation of a game $G := (\{1, 2, \dots, m\}, \mathcal{X}, g)$ using a matrix.
- ▶ m **players**: Typically two or more.
- ▶ **Strategies**: Each player has a set of strategies.
- ▶ **Payoffs**: Each cell of the matrix represents the payoffs for each player.

Example

	Strategy C	Strategy D
Strategy A	(3,2)	(1,4)
Strategy B	(2,3)	(4,1)

- ▶ The first number in each cell is the payoff for the **row player**.
- ▶ The second number in each cell is the payoff for the **column player**.

Zero-Sum Games

- ▶ One player's gain is exactly the other players' loss.
- ▶ Total sum of all players' payoffs is zero: $\sum_{i=1}^n g_i = 0$.
- ▶ Commonly used in competitive scenarios like poker, chess, and economic models.

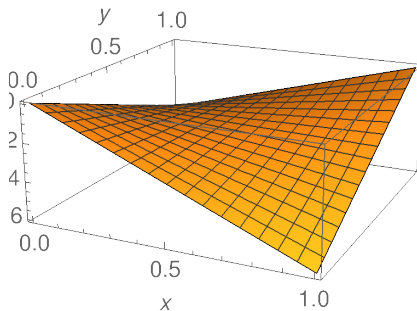
Example: Rock-Paper-Scissors

	Rock	Paper	Scissors
Rock	(0, 0)	(-1, 1)	(1, -1)
Paper	(1, -1)	(0, 0)	(-1, 1)
Scissors	(-1, 1)	(1, -1)	(0, 0)

Constant-Sum Games

- ▶ Total sum of all players' payoffs is a constant C : $\sum_{i=1}^n g_i = C$

Bilinear Maximin Objective Function



Definition

Given $\alpha, \beta \in (0, 1)$, for all $x, y \in \{0, 1\}^n$

$$\text{BILINEAR}_{\alpha, \beta}(x, y) := |y|_1(|x|_1 - \beta n) - \alpha n|x|_1$$

Test-Based Games

Special case of zero-sum normal form game

- ▶ Two players: “Designs” and “Tests”
- ▶ Payoff function g takes binary values only

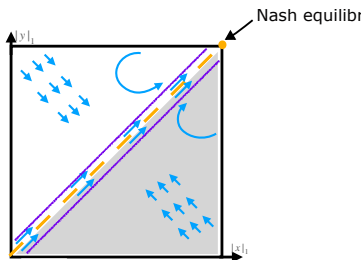
$$g(x, y) = 1 \implies \text{design } x \text{ passes test } y$$

$$g(x, y) = 0 \implies \text{design } x \text{ fails test } y$$

Important sub-class of games, e.g.:

- ▶ Co-evolution of sorting networks
- ▶ Board games with one winner and one loser

Example: Theoretical Benchmark Diagonal



$$\text{DIAGONAL}(x, y) = \begin{cases} 1 & \text{if } x \text{ contains more 1-bits than } y \\ 0 & \text{otherwise} \end{cases}$$

Theorem (Lehre and Lin [2024])

- ▶ *W.o.p., $(1, \lambda)$ EA with needs $e^{\Omega(n)}$ evaluations to find Nash of DIAGONAL*
- ▶ *$(1, \lambda)$ CoEA with $\lambda \in \Omega(n^{4/(1-\chi)})$ and restarts finds a δ -approximation of Nash in expected time $O(\lambda n)$.*

Symmetric vs Asymmetric Games

► Symmetric Games:

- Payoff depends only on the strategies, not on who is playing them.

Example: Prisoner's Dilemma

	Silent	Testify
Silent	(1, 1)	(5, 0)
Testify	(0, 5)	(3, 3)

► Asymmetric Games:

- Payoff differs depending on the player.
- It is possible, to have identical strategy spaces for all players.

Example: Battle of the Sexes

	Football	Opera
Football	(2, 1)	(0, 0)
Opera	(0, 0)	(1, 2)

Solution Concepts

Recall: We want to predict the outcome of the games. (what is the optimal way that individuals should play?).

Formally specifies which strategies are a *desired* solution.

► Nash Equilibrium:

- No player can improve their payoff by unilaterally changing their strategy.
- $(s_1^*, s_2^*, \dots, s_m^*)$ is a Nash equilibrium if for every player $i \in I$

$$g_i(s_1^*, \dots, s_i^*, \dots, s_m^*) \geq g_i(s_1^*, \dots, s_i, \dots, s_m^*) \quad \forall s_i \in \mathcal{X}_i$$

► Maxmin:

- Maximizing the minimum payoff.
- For player i , the maxmin strategy is:

$$\max_{s_i \in S_i} \min_{s_{-i} \in \mathcal{X}_{-i}} g_i(s_i, s_{-i})$$

- Ensures the best worst-case scenario.

► Maximum Expected Utility:

- Choosing the strategy with the highest expected utility.
- Formally, the expected utility of a strategy s_i for player i is defined as:

$$EU(s_i) = \sum_{s_{-i} \in S_{-i}} g_i(s_i, s_{-i}) \cdot \Pr(s_{-i})$$

- Balances potential payoffs with their probabilities.

Computational Models

- ▶ Design and analysis of algorithms assume a computational model
 - ▶ Design: What information is provided to the algorithm? What operations can the algorithm make?
 - ▶ Analysis: How is the cost of the running the algorithm defined?
- ▶ Full Information Model
 - ▶ Payoff matrix given as input to the algorithm
 - ▶ Cost determined by number of computational operations
 - ▶ Model consistent with the Lemke-Howson algorithm and other classical algorithms
- ▶ Query Complexity/Black Box Model
 - ▶ Algorithm learns payoff by interacting with the game
 - ▶ Cost determined by the number of payoff queries
 - ▶ Model consistent with operations of co-evolutionary algorithms and other heuristics

Black Box Complexity (or Query Complexity) for Pure Strategies

Game Class F

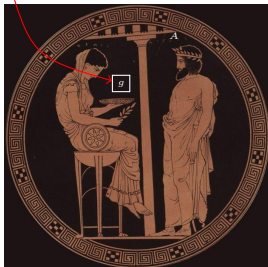


Photo: E. Gerhard (1846).

Black-box Algorithm A on Class of Games F

- ▶ Information available to algorithm A
 - ▶ $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_m$, where $\mathcal{X}_i$ is set of strategies for player $i \in [m]$
 - ▶ Class of games F where each game has
 - ▶ a payoff function $g_i : \mathcal{X} \rightarrow \mathbb{R}$ for each player $i \in [m]$
 - ▶ a solution concept $B_g \subseteq \mathcal{X}$
- ▶ An adversary selects a game $g \in F$.
At iteration $t = 0$, A does not know g or B_g .

Babichenko [2020]

Black Box Complexity (or Query Complexity) for Pure Strategies

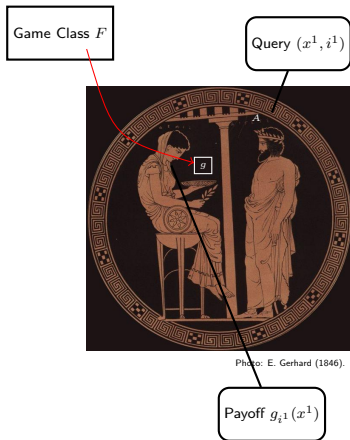


Photo: E. Gerhard (1846).

Black-box Algorithm A on Class of Games F

- ▶ Information available to algorithm A
 - ▶ $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_m$, where $\mathcal{X}_i$ is set of strategies for player $i \in [m]$
 - ▶ Class of games F where each game has
 - ▶ a payoff function $g_i : \mathcal{X} \rightarrow \mathbb{R}$ for each player $i \in [m]$
 - ▶ a solution concept $B_g \subseteq \mathcal{X}$
 - ▶ An adversary selects a game $g \in F$.
At iteration $t = 0$, A does not know g or B_g .
- ▶ In each iteration t , given what is known, A either
 - ▶ selects a strategy profile $x^t \in \mathcal{X}$ and a player $i_t \in [m]$, learns the payoff $g_{i_t}(x^t)$, and continues
 - ▶ outputs an element in $\mathcal{X}$ and halts

Babichenko [2020]

Black Box Complexity (or Query Complexity) for Pure Strategies

Game Class F

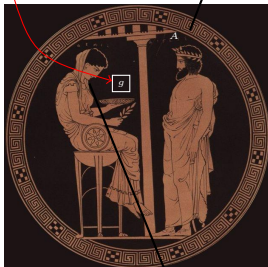


Photo: E. Gerhard (1846).

$(x^1, i^1), (x^2, i^2), (x^3, i^3), \dots, (x^t, i^t)$

$g_{i^1}(x^1), g_{i^1}(x^2), g_{i^1}(x^3), \dots, g_{i^t}(x^t)$

Black-box Algorithm A on Class of Games F

Information available to algorithm A

- ▶ $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2 \times \dots \times \mathcal{X}_m$, where $\mathcal{X}_i$ is set of strategies for player $i \in [m]$
- ▶ Class of games F where each game has
 - ▶ a payoff function $g_i : \mathcal{X} \rightarrow \mathbb{R}$ for each player $i \in [m]$
 - ▶ a solution concept $B_g \subseteq \mathcal{X}$
- ▶ An adversary selects a game $g \in F$.
At iteration $t = 0$, A does not know g or B_g .
- ▶ In each iteration t , given what is known, A either
 - ▶ selects a strategy profile $x^t \in \mathcal{X}$ and a player $i_t \in [m]$, learns the payoff $g_{i_t}(x^t)$, and continues
 - ▶ outputs an element in $\mathcal{X}$ and halts
- ▶ Goal: output a solution in B_g using the smallest number of queries possible.
- ▶ Algorithm A is correct on F if for all games $g \in F$, it outputs an element in B_g with probability at least $2/3$.

Babichenko [2020]

Black Box Complexity (or Query Complexity) for Mixed Strategies

Game Class F

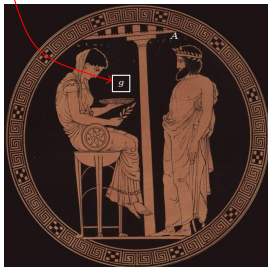


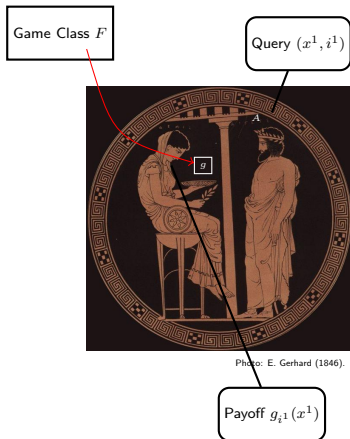
Photo: E. Gerhard (1846).

Black-box Algorithm A on Class of Games F

- ▶ Information available to algorithm A
 - ▶ $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_m$, where $\mathcal{X}_i$ is set of strategies for player $i \in [m]$
 - ▶ Class of games F where each game has
 - ▶ a payoff function $g_i : \mathcal{X} \rightarrow \mathbb{R}$ for each player $i \in [m]$
 - ▶ a solution concept $B_g \subseteq \Delta(\mathcal{X})$ where $\Delta(\mathcal{X}) := \Delta(\mathcal{X}_1) \times \Delta(\mathcal{X}_2) \times \cdots \times \Delta(\mathcal{X}_m)$
 - ▶ An adversary selects a game $g \in F$.
At iteration $t = 0$, A does not know g or B_g .
- ▶ In each iteration t , given what is known, A either
 - ▶ selects a strategy profile $x^t \in \mathcal{X}$ and a player $i_t \in [m]$, learns the payoff $g_{i_t}(x^t)$, and continues
 - ▶ outputs an element in $\Delta(\mathcal{X})$ and halts
- ▶ Goal: output a solution in B_g using the smallest number of queries possible.
- ▶ Algorithm A is correct on F if for all games $g \in F$, it outputs an element in B_g with probability at least $2/3$.

Babichenko [2020]

Black Box Complexity (or Query Complexity) for Mixed Strategies



Black-box Algorithm A on Class of Games F

- ▶ Information available to algorithm A
 - ▶ $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_m$, where $\mathcal{X}_i$ is set of strategies for player $i \in [m]$
 - ▶ Class of games F where each game has
 - ▶ a payoff function $g_i : \mathcal{X} \rightarrow \mathbb{R}$ for each player $i \in [m]$
 - ▶ a solution concept $B_g \subseteq \Delta(\mathcal{X})$ where $\Delta(\mathcal{X}) := \Delta(\mathcal{X}_1) \times \Delta(\mathcal{X}_2) \times \cdots \times \Delta(\mathcal{X}_m)$
 - ▶ An adversary selects a game $g \in F$.
At iteration $t = 0$, A does not know g or B_g .
- ▶ In each iteration t , given what is known, A either
 - ▶ selects a strategy profile $x^t \in \mathcal{X}$ and a player $i_t \in [m]$, learns the payoff $g_{i_t}(x^t)$, and continues
 - ▶ outputs an element in $\Delta(\mathcal{X})$ and halts
- ▶ Goal: output a solution in B_g using the smallest number of queries possible.
- ▶ Algorithm A is correct on F if for all games $g \in F$, it outputs an element in B_g with probability at least $2/3$.

Babichenko [2020]

Black Box Complexity (or Query Complexity) for Mixed Strategies

Game Class F

$(x^1, i^1), (x^2, i^2), (x^3, i^3), \dots, (x^t, i^t)$

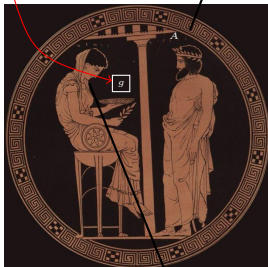


Photo: E. Gerhard (1846).

$g_{i^1}(x^1), g_{i^1}(x^2), g_{i^1}(x^3), \dots, g_{i^t}(x^t)$

Black-box Algorithm A on Class of Games F

Information available to algorithm A

- ▶ $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2 \times \dots \times \mathcal{X}_m$, where $\mathcal{X}_i$ is set of strategies for player $i \in [m]$
- ▶ Class of games F where each game has
 - ▶ a payoff function $g_i : \mathcal{X} \rightarrow \mathbb{R}$ for each player $i \in [m]$
 - ▶ a solution concept $B_g \subseteq \Delta(\mathcal{X})$ where $\Delta(\mathcal{X}) := \Delta(\mathcal{X}_1) \times \Delta(\mathcal{X}_2) \times \dots \times \Delta(\mathcal{X}_m)$
- ▶ An adversary selects a game $g \in F$.
At iteration $t = 0$, A does not know g or B_g .
- ▶ In each iteration t , given what is known, A either
 - ▶ selects a strategy profile $x^t \in \mathcal{X}$ and a player $i_t \in [m]$, learns the payoff $g_{i_t}(x^t)$, and continues
 - ▶ outputs an element in $\Delta(\mathcal{X})$ and halts
- ▶ Goal: output a solution in B_g using the smallest number of queries possible.
- ▶ Algorithm A is correct on F if for all games $g \in F$, it outputs an element in B_g with probability at least $2/3$.

Babichenko [2020]

Runtime of Black Box Algorithms

- ▶ Runtime $T_{A,g}$ of algorithm A on game g is the number of queries made

- ▶ Runtime $T_{A,F}$ of an algorithm A on class of games F

$$T_{A,F} = \max_{g \in F} \mathbf{E} [T_{A,g}]$$

- ▶ Black-box (or query) complexity of F is

$$T_F = \min\{T_{A,F} \mid A \text{ is correct on } F\}$$

Runtime of Black Box Algorithms

- ▶ Runtime $T_{A,g}$ of algorithm A on game g is the number of queries made
 - ▶ Co-evolutionary algorithms usually do not specify a termination criterion. For such algorithms (in the case of pure strategy profiles), define

$$T_{A,g} = \min\{t \mid x^t \in B_g\}$$

- ▶ Runtime $T_{A,F}$ of an algorithm A on class of games F

$$T_{A,F} = \max_{g \in F} \mathbf{E} [T_{A,g}]$$

- ▶ Black-box (or query) complexity of F is

$$T_F = \min\{T_{A,F} \mid A \text{ is correct on } F\}$$

BB Complexity of Nash Equilibrium (NE)

Sol. concept	Strats.	Players	Complexity	Reference
Pure NE ⁵	N	2	$\Omega(N^2 / \log N)$	Conitzer and Sandholm [2004]
Unique NE ⁶	N	2	$\geq \frac{Nk}{8}$	Maiti, Boczar, Jamieson, and Ratliff [2023]
	N	2	$O(Nk^4 \text{polylog}(\frac{N}{\delta}))$	
ε -NE	2	m	$\exp(m)$	Rubinstein [2016]
ε -NE, $\varepsilon = O(\frac{1}{\log N})$	N	2	$\Omega(N \log N)$	Fearnley, Gairing, Goldberg, and Savani [2018]
Potential games	N	2	$\Omega(N^c)$	Babichenko and Rubinstein [2020]
	2	m	$2^{\Omega(\sqrt{m})}$	

Observations

- ▶ Any algorithm needs to look at a constant fraction of the strategies.
No algorithm efficient when the number of strategies is very large.
- ▶ If $m = 2$, enough to query slightly more than a constant fraction of payoffs.
Co-evolution unlikely to be competitive when the number of strategies is small.
- ▶ Co-evolution should focus on the scenario with a large number of strategies, but where the class of games is restricted to have some “structure”.

⁵Proof uses communication complexity.

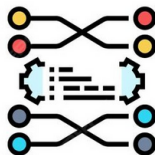
⁶ k is the size of support of the unique mixed NE

Adversarial No Free Lunch

Algorithm A



Algorithm B

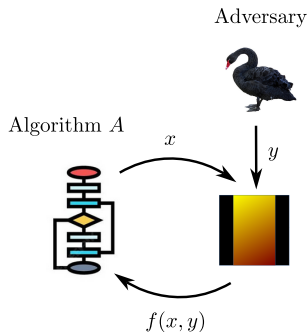


Theorem (L. & Lin, NeurIPS 2024)

If $\mathcal{F}$ is a class of games *closed under permutation*, then for any algorithms A and B ,

$$\mathbb{E}_{f \sim \text{Unif}(\mathcal{F})} [T_{A,f}] = \mathbb{E}_{f \sim \text{Unif}(\mathcal{F})} [T_{B,f}].$$

Adversarial Black Box Lower Bound



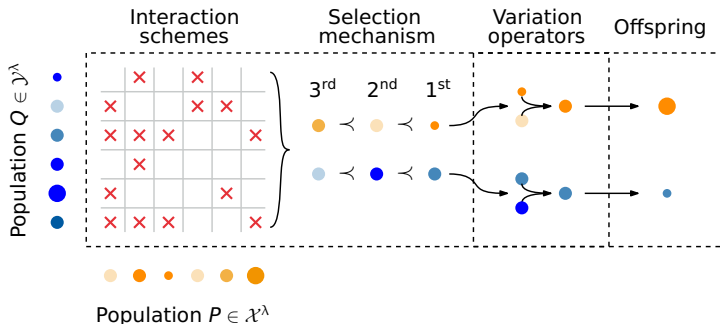
Theorem (L. & Lin, NeurIPS 2024)

Consider any game $f : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \mathbb{R}$ with a unique Nash equilibrium where

$$\left| \sum_{i=1}^n y_i - \frac{n}{2} \right| \leq \epsilon n \quad \text{and} \quad \left| \sum_{i=1}^n y'_i - \frac{n}{2} \right| \leq \epsilon n \quad \text{implies} \quad f(x, y) = f(x, y').$$

Let $F = \{f(x, y \oplus z) \mid z \in \{0, 1\}^n\}$. Then all algorithms A have runtime $T_{A,F} = e^{\Omega(n)}$.

Designing a Simple Coevolutionary Algorithm



Similarities with EAs:

- ▶ Initialisation
- ▶ Representation
- ▶ Variation operators
- ▶ Stopping criteria

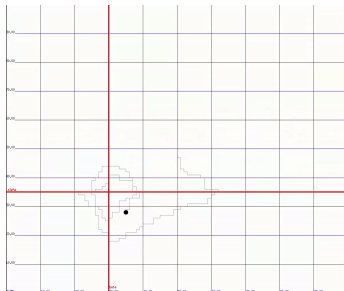
Unique to coevolutionary algorithms:

- ▶ How many populations?
- ▶ How the populations interact?
- ▶ How to assign fitness or select parents?

Populations vs. Single Individuals

Single individuals:

- ▶ Fast on extremely easy games
- ▶ Tends to be unstable and noisy:
 - ▶ (1+1) RLS forgets and stays away from the Nash Equilibrium on Bilinear Hevia Fajardo, Lehre, and Lin [2023]
- ▶ Lacks of diversity to solve complex games:
 - ▶ Any CoEA that retains only one search point as the current population is inefficient on a Class of Symmetric Zero-Sum Games Benford and Lehre [2024]



Populations - Single vs Multi-population

Single population:

- ▶ Applicable for symmetric games $(\mathcal{X}_1, \mathcal{X}_2, \dots, \mathcal{X}_m)$.
 - ▶ Population-level self-learning.
 - ▶ Analogue to self-play.
- ▶ Optimise tuples of strategies instead of individual strategies for asymmetric games $(\mathcal{X}_1 \neq \mathcal{X}_2)$.

Two or more populations:

- ▶ One population per player.
- ▶ Separate populations into teachers and learners.
 - ▶ Applicable for symmetric and asymmetric games.

Open question: In what situations is one option preferred over the other??

Interaction Schemes

Universal - Single/Multi Population and Symmetric/Asymmetric Games:

- ▶ All-vs-all
- ▶ All-vs-previous best
- ▶ k-random opponents in competing population
- ▶ Spatial populations:
 - ▶ Interaction happens only with the neighbouring opponents
 - ▶ Examples - cybersecurity [Hevia Fajardo, Lehre, Toutouh, Hemberg, and O'Reilly, 2024], GANs [Hemberg, Toutouh, Al-Dujaili, Schmiedlechner, and O'Reilly, 2021], sorting networks [Hillis, 1990]

Single population:

- ▶ Fitnessless evaluation (Symmetric Games):
 - ▶ Single-elimination tournament (ant wars [Jaśkowski, Krawiec, and Wieloch, 2008])
 - ▶ Round-robin
- ▶ Asymmetric Games:
 - ▶ Treating strategy profiles (tuples of strategies) as individuals
 - ▶ Assign a fitness to the tuple with game theory concepts:
 - ▶ Exploitability
 - ▶ Social welfare

Two or more populations:

- ▶ Deterministic tournament selection
 - ▶ Create subgames deterministically
 - ▶ Example - Shared sampling [Rosin and Belew, 1997]: Use individuals that complement each other
- ▶ Random tournament selection
 - ▶ Create subgames at random
 - ▶ Example - PDCoEA [Lehre, 2022]

Selection Mechanisms

The parents are selected based on their interactions with other solutions.

Objective Fitness (complete information):

- ▶ Fitness is calculated using a principled selection of elements.
- ▶ Examples:
 - ▶ Expected utility: "Average" game outcome against all possible opponents (computationally unfeasible).
 - ▶ Game outcome against a predefined opponent (risk of overspecialization).

Subjective Fitness (incomplete information):

- ▶ Fitness is calculated using a sample of the current population.
- ▶ Incomplete information of the strategy space (exponentially big).
- ▶ Example:
 - ▶ Average game outcome against opponents from the current population.

Q: How to select parents?

Selection Mechanisms

Single interaction:

- ▶ Assign the payoff as fitness
- ▶ Cheap to obtain the fitness
- ▶ **Caution:** May be unstable and noisy

Multiple interactions:

- ▶ Fitnessless selection (Symmetric Games)
- ▶ Average payoff
- ▶ Worst-case payoff
- ▶ Median payoff
- ▶ Fitness sharing [Rosin and Belew, 1997]
 - ▶ Weighted average, giving more importance to solutions that are difficult to beat
- ▶ Pairwise dominance [Lehre, 2022]
- ▶ Ranking diversity [Hevia Fajardo and Lehre, 2024]

The design of your coevolutionary algorithm depends on the game you want to solve.

Maximin-Optimisation Problems

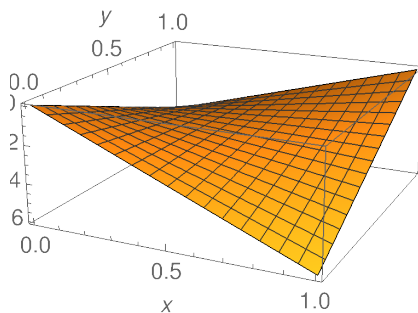
Given maximin-objective function

$$g : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$$

maximise the function

$$f(x) := \min_{y \in \mathcal{Y}} g(x, y).$$

Bilinear Maximin Objective Function

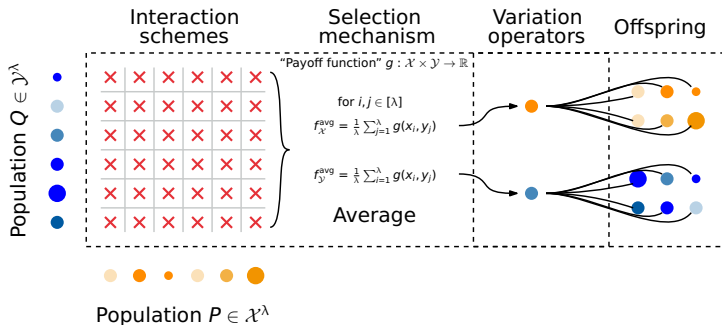


Definition

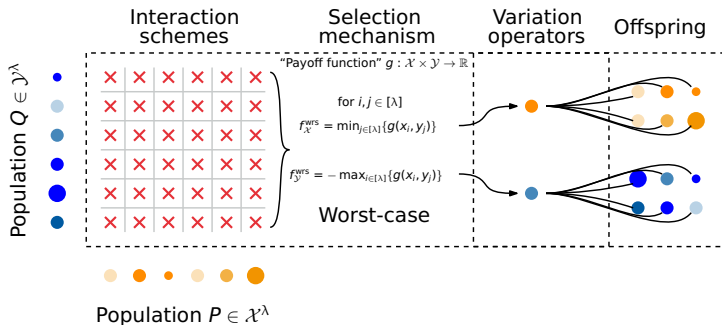
Given $\alpha, \beta \in (0, 1)$, for all $x, y \in \{0, 1\}^n$

$$\text{BILINEAR}_{\alpha, \beta}(x, y) := |y|_1(|x|_1 - \beta n) - \alpha n |x|_1$$

The $(1, \lambda)$ CoEA



The $(1, \lambda)$ CoEA



Example: Runtime for the $(1, \lambda)$ CoEA on BILINEAR Hevia Fajardo and Lehre [2023] using different fitness aggregation methods:



Average: Inefficient runtime.



Worst-case: Polynomial runtime with high probability.

Coevolutionary Pathologies

Coevolutionary algorithms may present pathologies.

Cycling:

- ▶ Revisiting a portion of search-space periodically with no global progress.
- ▶ Particularly likely in intransitive games.

Disengagement:

- ▶ Opponents are either trivial or too difficult to beat.
- ▶ Selection pressure diminishes and there is a loss of “gradient”.

Overspecialization:

- ▶ Mastering skills to beat certain opponent strategies while neglecting others.

Evolutionary forgetting:

- ▶ Forget previously good strategies because they are not needed for the current competitors.

To avoid these pathologies it is critical to choose a good evaluation environment (set of opponents) [Rosin and Belew, 1995, Miconi, 2009, Krawiec and Heywood, 2020]

Population Archives in Coevolutionary Algorithms

What is an Archive?

A set of solutions that is systematically updated during the coevolutionary process.

Motivation for Using Archives:

- ▶ **Prevent Pathologies:**
 - ▶ *Forgetting* - Ensuring valuable solutions/behaviours are not lost over time.
 - ▶ *Cycling* - Avoiding repetitive cycles that hinder progress.
- ▶ **Promote Diversity:**
 - ▶ Ensure a wide range of solutions to evaluate against.
- ▶ **Long-term memory:**
 - ▶ Collect “good” individuals.

Population Archives in Coevolutionary Algorithms

What is an Archive?

A set of solutions that is systematically updated during the coevolutionary process.

Motivation for Using Archives:

- ▶ **Prevent Pathologies:**
 - ▶ *Forgetting* - Ensuring valuable solutions/behaviours are not lost over time.
 - ▶ *Cycling* - Avoiding repetitive cycles that hinder progress.
- ▶ **Promote Diversity:**
 - ▶ Ensure a wide range of solutions to evaluate against.
- ▶ **Long-term memory:**
 - ▶ Collect “good” individuals.
- ▶ **Create monotonic progress:**
 - ▶ Once you discard an individual, you will never select it again.

Archive Methods in Coevolutionary Algorithms

Hall of Fame [Rosin and Belew, 1997]

- ▶ Best individuals from each generation.
- ▶ Populations are tested against each other and the archive.

Nash Memory [Ficici and Pollack, 2003]

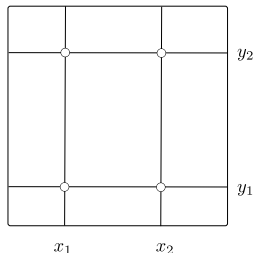
- ▶ Aims to represent a *mixed strategy* Nash equilibrium.
- ▶ Monotonic progress

Pareto Archives (IPCA) [De Jong, 2004]

- ▶ Test-based games: tests are treated as objectives.
- ▶ Maintains a set of non-dominated solutions.
- ▶ Monotonic progress

Dimension extraction / Coordinate systems

- ▶ Designed for test-based games.
- ▶ Aims to find sets of tests that form underlying objectives.



Definition (Dominance)

Given two predators $x_1, x_2 \in \mathcal{X}$, and a set of prey $Q \subseteq \mathcal{Y}$, predator x_1 *dominates* predator x_2 with respect to the prey Q , denoted

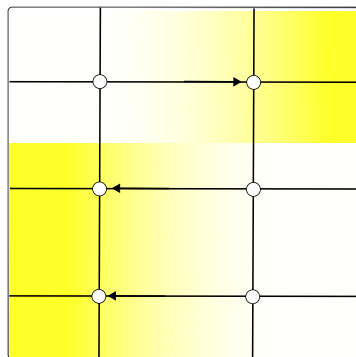
$$x_1 \succeq_Q x_2 \quad \text{if and only if} \quad \forall y \in Q, g(x_1, y) \geq g(x_2, y).$$

Definition (Dominance (set of predators))

Given a predator $x \in \mathcal{X}$ a set of predators $P \subseteq \mathcal{X}$, and a set of prey $Q \subseteq \mathcal{Y}$, then the predators P *dominate* predator x with respect to the prey Q , denoted

$$P \succeq_Q x \quad \text{if and only if} \quad \exists x' \in P, x' \succeq_Q x.$$

Pareto Archives [De Jong, 2004]



y_3



$x_1 \succeq_{Q_1} x_2$ for $Q_1 = \{y_1, y_2\}$

because

$$g(x_1, y_1) > g(x_2, y_1)$$

$$g(x_1, y_2) > g(x_2, y_2)$$

y_2



y_1



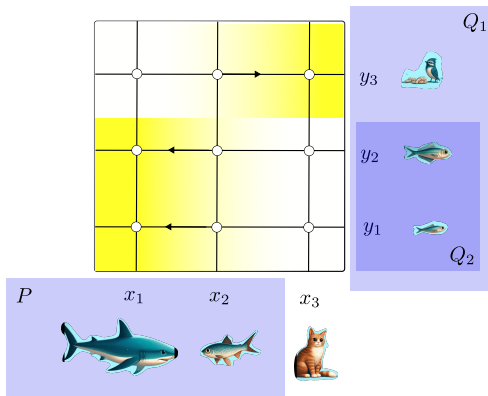
$x_1 \not\succeq_{Q_2} x_2$ for $Q_2 = \{y_1, y_2, y_3\}$

because

$$g(x_1, y_1) > g(x_2, y_1)$$

$$g(x_1, y_2) > g(x_2, y_2)$$

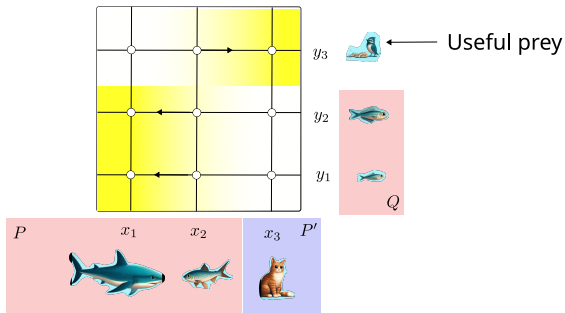
$$g(x_1, y_3) < g(x_2, y_3)$$



Definition

A predator $x \in \mathcal{X}$ is *useful* with respect to a set of predators $P \subseteq \mathcal{X}$ and prey $Q \subseteq \mathcal{Y}$ if and only if

$$P \not\preceq_Q x.$$



Definition

Given two sets of predators $P, P' \subseteq \mathcal{X}$, and set of prey $Q \subseteq \mathcal{Y}$, a prey $y \in \mathcal{Y}$ is called *useful* if there exists a predator $x \in P'$ such that

$$P \succeq_Q x \quad \text{and} \quad P \not\preceq_{Q \cup \{y\}} x.$$

Algorithm Incremental Pareto-Coevolution Archive (IPCA)

Require: Payoff function $g : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$.

- 1: Initialise $P_0 := \emptyset$, $Q_0 := \emptyset$ and $t = 0$.
 - 2: **while** termination condition not satisfied **do**
 - 3: $P_t := \{x \in P_t \mid P_t \not\preceq_{Q_t} x\}$ ▷ Only keep non-dominated predators in P_t .
 - 4: $P_{t+1} := P_t$
 - 5: $Q_{t+1} := Q_t$
 - 6: $P' := \text{generatePredators}(P_t)$ ▷ Reproduction via mutation/crossover.
 - 7: $Q' := \text{generatePrey}(Q_t)$ ▷ Reproduction via mutation/crossover.
 - 8: **for** $y \in Q'$ **do** ▷ Add useful prey from Q' to Q_{t+1} .
 - 9: **if** $\exists x \in P'$ such that $P_t \succeq_{Q_{t+1}} x$ and $P_t \not\preceq_{Q_{t+1} \cup \{y\}} x$ **then**
 - 10: $Q_{t+1} := Q_{t+1} \cup \{y\}$
 - 11: **for** $x \in P'$ **do** ▷ Add useful predators from P' to P_{t+1} .
 - 12: **if** $P_{t+1} \not\preceq_{Q_{t+1}} x$ **then**
 - 13: $P_{t+1} := P_{t+1} \cup \{x\}$
 - 14: **if** $P_t \neq P_{t+1}$ **then**
 - 15: $t := t + 1$
-

Monotonic Progress of IPCA

- Prey are never removed

$$Q_0 \subseteq Q_1 \subseteq \dots \subseteq Q_t \subseteq Q_{t+1} \quad (1)$$

- Predators are only removed if they are Pareto-dominated by a predator in a later generation

$$x \in P_t \text{ and } x \notin P_{t+1} \implies \exists x' \in P_{t+1} \text{ such that } x' \succeq_Q x \quad (2)$$

- Monotonic Progress (by induction from (3) and (4))
 - For all $t_1 < t_2$, for all $Q \subseteq Q_{t_1}$ and for all $x \in P_{t_1}$

$$P_{t_2} \succeq_Q x$$

Monotonic Progress of IPCA

- ▶ Prey are never removed

$$Q_0 \subseteq Q_1 \subseteq \dots \subseteq Q_t \subseteq Q_{t+1} \quad (1)$$

- ▶ Predators are only removed if they are Pareto-dominated by a predator in a later generation

$$x \in P_t \text{ and } x \notin P_{t+1} \implies \exists x' \in P_{t+1} \text{ such that } x' \succeq_Q x \quad (2)$$

- ▶ Monotonic Progress (by induction from (3) and (4))
 - ▶ For all $t_1 < t_2$, for all $Q \subseteq Q_{t_1}$ and for all $x \in P_{t_1}$

$$P_{t_2} \succeq_Q x$$

- ▶ Not guarantee to find solution using realistic computational budget
 - ▶ The set of prey Q can grow exceedingly large, i.e., unreasonable memory requirements
 - ▶ No guarantee on the time (number of generations) to find a solution.

Monotonic Progress of IPCA

- Prey are never removed

$$Q_0 \subseteq Q_1 \subseteq \cdots \subseteq Q_t \subseteq Q_{t+1} \quad (3)$$

- Predators are only removed if they are Pareto-dominated by a predator in a later generation

$$x \in P_t \text{ and } x \notin P_{t+1} \implies \exists x' \in P_{t+1} \text{ such that } x' \succeq_Q x \quad (4)$$

- Monotonic Progress (by induction from (3) and (4))
 - For all $t_1 < t_2$, for all $Q \subseteq Q_{t_1}$ and for all $x \in P_{t_1}$

$$P_{t_2} \succeq_Q x$$

Monotonic Progress of IPCA

- ▶ Prey are never removed

$$Q_0 \subseteq Q_1 \subseteq \cdots \subseteq Q_t \subseteq Q_{t+1} \quad (3)$$

- ▶ Predators are only removed if they are Pareto-dominated by a predator in a later generation

$$x \in P_t \text{ and } x \notin P_{t+1} \implies \exists x' \in P_{t+1} \text{ such that } x' \succeq_Q x \quad (4)$$

- ▶ Monotonic Progress (by induction from (3) and (4))
 - ▶ For all $t_1 < t_2$, for all $Q \subseteq Q_{t_1}$ and for all $x \in P_{t_1}$

$$P_{t_2} \succeq_Q x$$

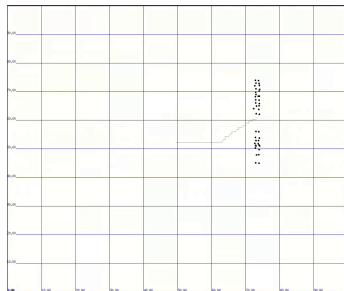
- ▶ Not guarantee to find solution using realistic computational budget
 - ▶ The set of prey Q can grow exceedingly large, i.e., unreasonable memory requirements
 - ▶ No guarantee on the time (number of generations) to find a solution.

Monotonic Progress of IPCA

- ▶ Not guarantee to find solution using realistic computational budget
 - ▶ The set of prey Q can grow exceedingly large, i.e., unreasonable memory requirements
 - ▶ No guarantee on the time (number of generations) to find a solution.



(1, λ) CoEA with average on
DIAGONAL.



IPCA on DIAGONAL.

Monotonic progress is costly, hard to achieve and not necessary.

A Different Approach to Designing CoEAs

Insights from Runtime Analysis of Classical EAs

- ▶ To reach the optimum, it is not necessary to decrease the distance monotonically. It suffices to have positive progress on average He and Yao [2001b].
- ▶ Weak selective bias suffices for non-elitist population-based EAs
 - ▶ E.g., with 2-tournament selection, selecting the “better” individual with probability $\geq 1/2 + \delta$ is sufficient assuming large population and small mutation rate Dang and Lehre [2016]

A Different Approach to Designing CoEAs

Insights from Runtime Analysis of Classical EAs

- ▶ To reach the optimum, it is not necessary to decrease the distance monotonically. It suffices to have positive progress on average. He and Yao [2001b].
- ▶ Weak selective bias suffices for non-elitist population-based EAs
 - ▶ E.g., with 2-tournament selection, selecting the “better” individual with probability $\geq 1/2 + \delta$ is sufficient assuming large population and small mutation rate. Dang and Lehre [2016]

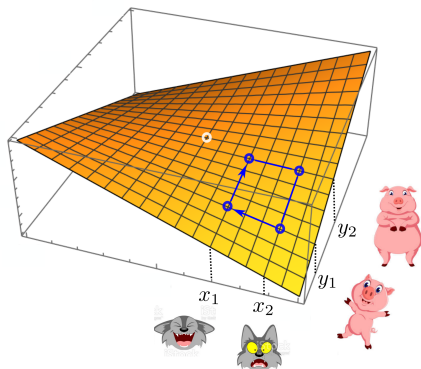
Alternative Approach to Designing CoEAs

- ▶ Replace costly “monotone progress” property with “positive drift”
- ▶ Ensure that selection chooses the “right” individuals sufficiently often
- ▶ If needed, prove performance guarantees using runtime analysis.

Pros and Cons

- ▶ Much simpler and more flexible algorithms
- ▶ Runtime analysis provides stronger and more useful performance guarantees: upper and lower bounds on time to reach solution
- ▶ Runtime analysis is challenging and may not be available/possible for large problem classes

Pairwise Dominance⁷



Let $g : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$ be a maximin-objective, then

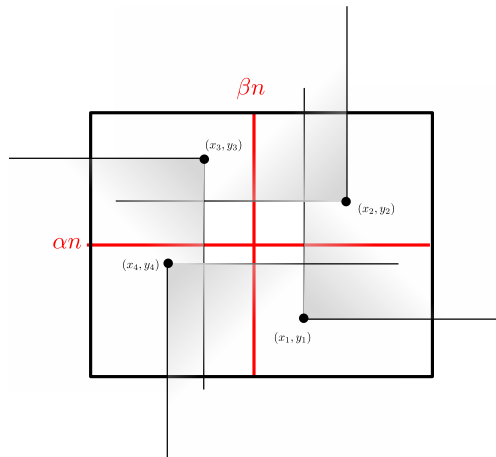
$$(x_1, y_1) \succeq (x_2, y_2) \iff g(x_1, y_2) \geq g(x_1, y_1) \geq g(x_2, y_1)$$

Lemma

If $g(x_1, y_1) \neq g(x_2, y_2)$, then $(x_1, y_1) \succeq_g (x_2, y_2) \implies (x_2, y_2) \not\succeq_g (x_1, y_1)$.

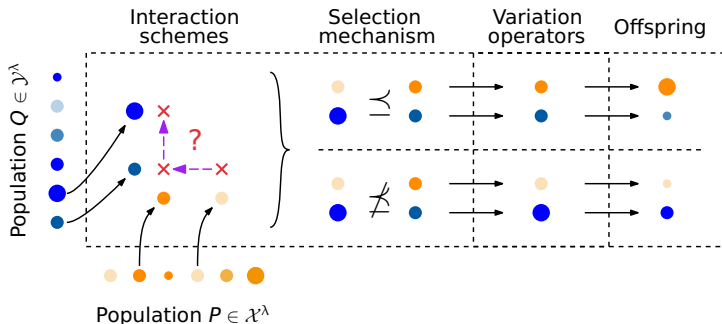
⁷ Not to be confused with Pareto dominance.

Pairwise Dominance on Bilinear



$$\text{BILINEAR}_{\alpha, \beta}(x, y) := |y|_1(|x|_1 - \beta n) - \alpha n|x|_1$$

Pairwise Dominance Co-Evolutionary Algorithm (PDCoEA)



Let $g : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$ be a maximin-objective, then

$$(x_1, y_1) \succeq (x_2, y_2) \iff g(x_1, y_2) \geq g(x_1, y_1) \geq g(x_2, y_1)$$

Lemma

If $g(x_1, y_1) \neq g(x_2, y_2)$, then $(x_1, y_1) \succeq_g (x_2, y_2) \implies (x_2, y_2) \not\succeq_g (x_1, y_1)$.

Pairwise Dominance Co-Evolutionary Algorithm (PDCoEA)

Input: Maxmin-objective function $g : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \mathbb{R}$.

Input: Population size $\lambda \in \mathbb{N}$ and mutation rate $\chi \in (0, n]$

```
1: for  $i \in [\lambda]$  do
2:   Sample  $P_0(i) \sim \text{Unif}(\{0, 1\}^n)$ 
3:   Sample  $Q_0(i) \sim \text{Unif}(\{0, 1\}^n)$ 
4: for  $t \in \mathbb{N}$  until termination criterion met do
5:   for  $i \in [\lambda]$  do
6:     Sample  $(x_1, y_1) \sim \text{Unif}(P_t \times Q_t)$ 
7:     Sample  $(x_2, y_2) \sim \text{Unif}(P_t \times Q_t)$ 
8:     if  $(x_1, y_1) \succeq_g (x_2, y_2)$  then
9:        $(x, y) := (x_1, y_1)$ 
10:    else
11:       $(x, y) := (x_2, y_2)$ 
12:    Obtain  $x'$  by flipping each bit in  $x$  with prob.  $\chi/n$ .
13:    Obtain  $y'$  by flipping each bit in  $y$  with prob.  $\chi/n$ .
14:    Set  $P_{t+1}(i) := x'$  and  $Q_{t+1}(i) := y'$ .
```

[Lehre, 2022]

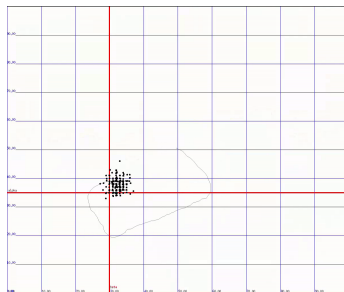
PDCoEA on Bilinear

Theorem

There is a constant c_0 , such that the runtime of PD-CoEA with

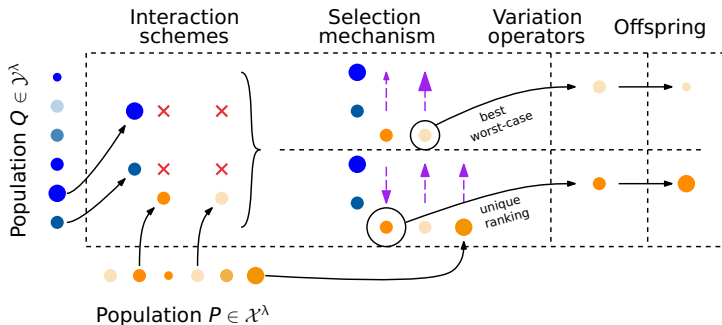
- ▶ population size $c \log n \leq \lambda \in \text{poly } n$ for some constant $c > 0$
- ▶ sufficiently small mutation rate $\chi \in \Theta(1)$

on BILINEAR^8 satisfies $\Pr(T > c_0 r \lambda^3 n) \leq (1/r)(1 + o(1))$ for all $r \in \text{poly}(n)$ [Lehre, 2022]

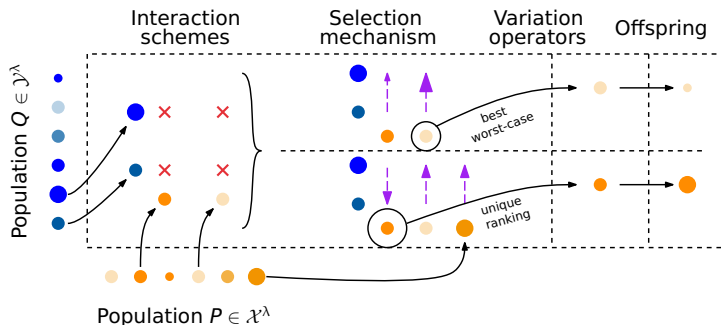


⁸with some restrictions

RankDiv-CoEA [Hevia Fajardo and Lehre, 2024]



RankDiv-CoEA [Hevia Fajardo and Lehre, 2024]



Best worst case:

$$\arg \max_{x' \in \{x_1, x_2\}} \min(g(x', y_1), g(x', y_2))$$

if $\text{sign}(g(x_1, y_1) - g(x_1, y_2)) = \text{sign}(g(x_2, y_1) - g(x_2, y_2))$

$$\text{sign}(x) := \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -1 & \text{if } x < 0 \end{cases}$$

Unique ranking:

$$(g(x_1, y_1) > g(x_1, y_2))$$

$$(g(x_2, y_1) < g(x_2, y_2))$$

$$(g(x_3, y_1) < g(x_3, y_2))$$

RankDiv-CoEA [Hevia Fajardo and Lehre, 2024]

Algorithm Ranking Diversity CoEA (RankDivCoEA)

Require: Strategy spaces $\mathcal{X}$ and $\mathcal{Y}$.

Require: Payoff function $g : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$.

Require: Population size $\lambda \in \mathbb{N}$.

Require: $\text{mut}_{\mathcal{X}}(\cdot) : \mathcal{X} \rightarrow \mathcal{X}$, $\text{mut}_{\mathcal{Y}}(\cdot) : \mathcal{Y} \rightarrow \mathcal{Y}$.

Require: Initial populations $P_0 \in \mathcal{X}^\lambda$ and $Q_0 \in \mathcal{Y}^\lambda$.

```
1: for  $t \in \mathbb{N}_0$  until termination condition satisfied do
2:   for  $i \in [\lambda]$  do
3:     /* Select predator parent  $x'$  and produce offspring  $x$ . */
4:     Sample  $x_1, x_2, x_3 \sim \text{Unif}(P_t)$  and  $y_1, y_2 \sim \text{Unif}(Q_t)$ 
5:     while  $g(x_1, y_1) = g(x_1, y_2) \wedge g(x_2, y_1) = g(x_2, y_2)$  do
6:       Flip one bit u. a. r. from either  $y_1$  or  $y_2$ 
7:        $s_1 \leftarrow \text{sign}(g(x_1, y_1) - g(x_1, y_2))$ 
8:        $s_2 \leftarrow \text{sign}(g(x_2, y_1) - g(x_2, y_2))$ 
9:        $s_3 \leftarrow \text{sign}(g(x_3, y_1) - g(x_3, y_2))$ 
10:       $x' \leftarrow \begin{cases} x_1 & \text{if } s_1 \neq s_2 \wedge s_1 \neq s_3 \\ x_1 & \text{if } s_1 = s_2 \wedge \min(g(x_1, y_1), g(x_1, y_2)) \geq \min(g(x_2, y_1), g(x_2, y_2)) \\ x_2 & \text{otherwise.} \end{cases}$ 
11:      Create  $P_{t+1}(i) := \text{mut}_{\mathcal{X}}(x')$ 
12:     /* Select prey parent  $y'$  and produce offspring  $y$ . */
13:     Sample  $y_1, y_2, y_3 \sim \text{Unif}(Q_t)$  and  $x_1, x_2 \sim \text{Unif}(P_t)$ 
14:     while  $g(x_1, y_1) = g(x_2, y_1) \wedge g(x_1, y_2) = g(x_2, y_2)$  do
15:       Flip one bit u. a. r. from either  $x_1$  or  $x_2$ 
16:        $s_1 \leftarrow \text{sign}(g(x_1, y_1) - g(x_2, y_1))$ 
17:        $s_2 \leftarrow \text{sign}(g(x_1, y_2) - g(x_2, y_2))$ 
18:        $s_3 \leftarrow \text{sign}(g(x_1, y_3) - g(x_2, y_3))$ 
19:        $y' \leftarrow \begin{cases} y_1 & \text{if } s_1 \neq s_2 \wedge s_1 \neq s_3 \\ y_1 & \text{if } s_1 = s_2 \wedge \max(g(x_1, y_1), g(x_2, y_1)) \leq \max(g(x_1, y_2), g(x_2, y_2)) \\ y_2 & \text{otherwise.} \end{cases}$ 
20:      Create  $Q_{t+1}(i) := \text{mut}_{\mathcal{Y}}(y')$ 
```

RankDiv CoEA on Bilinear

Theorem

The expected runtime of RankDiv CoEA with

- ▶ population size $\lambda \geq c \log n$ for some constant $c > 0$
- ▶ sufficiently small mutation rate $\chi \in \Theta(1)$

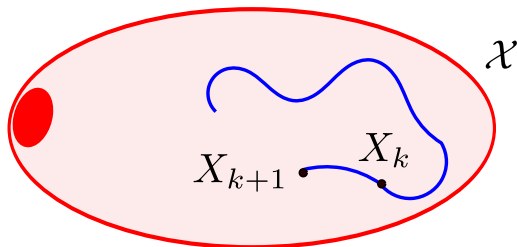
on BILINEAR is $O(n\lambda^3 - n\lambda \ln(\alpha\beta(1 - \beta)(1 - \alpha)))$ [Hevia Fajardo and Lehre, 2024]



Runtime Analysis of Competitive Co-Evolutionary Algorithms

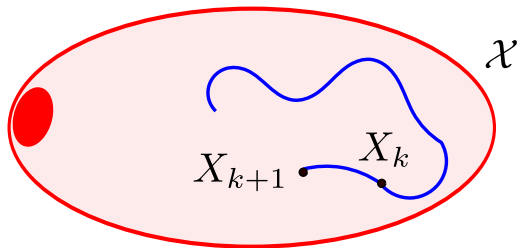
- ▶ Runtime Analysis gives bounds on runtime as a function of problem size
 - ▶ Standard approach in classical algorithms research (eg mergesort runs in $O(n \log n)$)
 - ▶ Appeared in Evolutionary Computation in the late 1990s
 - ▶ First results for simple settings, e.g. $(1+1)$ EA on ONEMAX, later advanced to more complex algorithms and problems
 - ▶ Now a major approach to theory of EAs (see [Doerr and Neumann, 2020] for a recent book)
- ▶ Runtime analysis of *cooperative* coevolutionary $(1+1)$ EA in 2004 [Jansen and Wiegand, 2004]
 - ▶ Cooperative coevolution not necessarily faster on separable problems.
- ▶ Runtime Analysis of CoEAs considered a major open problem since 2012 [Popovici, Bucci, Wiegand, and de Jong, 2012]
- ▶ First runtime result for competitive co-evolution in 2022 [Lehre, 2022]
 - ▶ PDCoEA has polynomial runtime on some BILINEAR instances, and exponential runtime on any game when the mutation rate is too high
- ▶ Recently, several analyses have become available
 - ▶ Ranking diversity: Hevia Fajardo & Lehre (PPSN 2024)
 - ▶ Binary games: Lin & Lehre (PPSN 2024)
 - ▶ Tail bounds: Lin & Lehre (IJCAI 2024)
 - ▶ Symmetric zero-sum games: Benford & Lehre (GECCO 2024)
 - ▶ Overcoming negative drift: Hevia Fajardo, Lehre, & Lin (FOGA 2023)
 - ▶ Fitness aggregation methods: Hevia Fajardo & Lehre (GECCO 2023)
 - ▶ Tight bounds for CC- $(1+1)$ EA: Lehre & Lehre (CEC 2023)

Runtime Analysis: From short-term to long-term properties



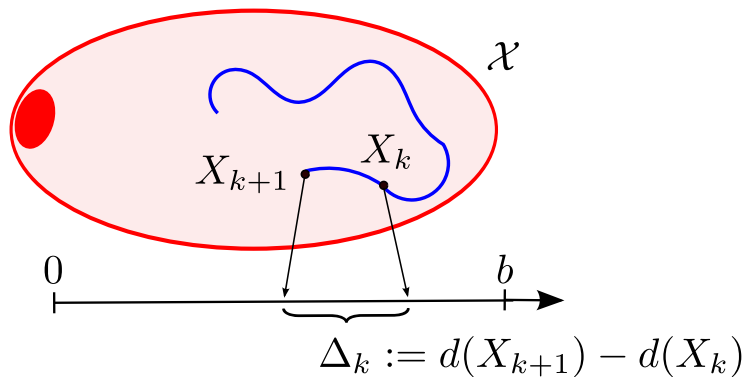
- ▶ An EA can be described by a stochastic process in a state space $\mathcal{X}$:
 - ▶ In iteration k the EA is in a state X_k . Given X_k and some randomness, the EA transitions to a next state X_{k+1} .
- ▶ Short term prediction is easy:
Given typical design of EAs (selection + variation), it is often “easy” to describe mathematically how X_{k+1} depends on X_k .
- ▶ Long term prediction is harder:
We are looking for long-term properties of X_k , e.g., the smallest k such that X_k belongs to a target set B (runtime).
- ▶ Theory research focused on developing tools to predict long-term from short-term.

Drift⁹ Analysis?



⁹NB! (Stochastic) drift is a different concept than *genetic drift* in population genetics.

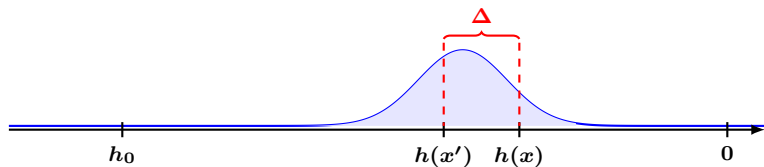
Drift⁹ Analysis?



- Prediction of the long term behaviour of a process X
 - hitting time, stability, occupancy time etc.
- from properties of Δ .

⁹NB! (Stochastic) drift is a different concept than *genetic drift* in population genetics.

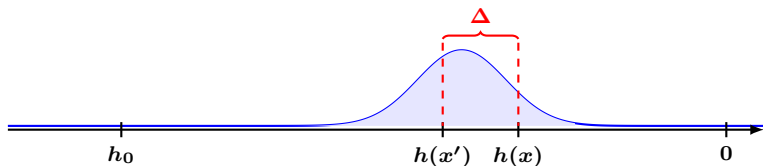
Additive Drift Theorem



$$(C1+) \quad \forall k \quad \mathbb{E} [Y_{k+1} - Y_k \mid Y_k > 0] \leq -\varepsilon$$

$$(C1-) \quad \forall k \quad \mathbb{E} [Y_{k+1} - Y_k \mid Y_k > 0] \geq -\varepsilon$$

Additive Drift Theorem



$$(C1+) \quad \forall k \quad \mathbf{E} [Y_{k+1} - Y_k \mid Y_k > 0] \leq -\varepsilon$$

$$(C1-) \quad \forall k \quad \mathbf{E} [Y_{k+1} - Y_k \mid Y_k > 0] \geq -\varepsilon$$

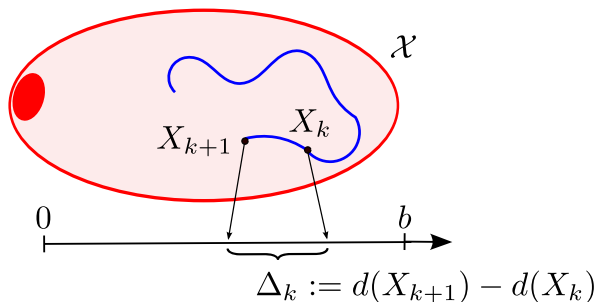
Theorem (He and Yao [2001a], Jägersküpper [2007], Jägersküpper [2008])

Given a stochastic process $Y_1, Y_2, \dots$ over an interval $[0, b] \subset \mathbb{R}$.

Define $T := \min\{k \geq 0 \mid Y_k = 0\}$, and assume $\mathbf{E}[T] < \infty$.

- ▶ If (C1+) holds for an $\varepsilon > 0$, then $\mathbf{E}[T \mid Y_0] \leq b/\varepsilon$.
- ▶ If (C1-) holds for an $\varepsilon > 0$, then $\mathbf{E}[T \mid Y_0] \geq Y_0/\varepsilon$.

State Aggregation Problem of Drift Analysis



How to define a mapping d from state space to “distance” (the art of drift analysis)

- ▶ Relatively easy for $(1 + \lambda)$ or $(1, \lambda)$ -type of algorithms: In each generation, the EA selects a single individual. Define d to be the “distance” of that individual to the optimum. Given a current distance, often easy to estimate distribution of offspring.
- ▶ **Hard for population-based EAs:** Many different individuals can contribute to offspring distribution, so d should not depend only on a single individual. When estimating drift at a given distance d (i.e., a real value), need to consider all populations that have distance d .
- ▶ The Level-Based Theorem is an alternative to drift analysis, and is specifically designed for (non-elitist) population-based EAs.

Algorithm assumed by Level-Based Theorem

Require: Population size λ

Require: Initial population $P_0 \in \mathcal{X}^\lambda$

for $t = 0$ to ∞ do

for $i = 1$ to λ do

$P_{t+1}(i) \sim \mathcal{D}(P_t)$

- ▶ In each generation, the algorithm produces λ offspring, where each offspring is produced by running a “program” $\mathcal{D}$ that takes the current population P_t as input.
- ▶ The program $\mathcal{D}$ can be anything, e.g., compute fitnesses, apply selection and variation operators etc.
- ▶ Many non-elitist EAs fit this framework.

Level-based Analysis

Problem

- ▶ Search space $\mathcal{X}$ and a “target region” $B \subseteq \mathcal{X}$ (e.g. optima)
- ▶ What is the expected time to reach the target set B ?
More precisely, how does $\mathbf{E}[T_B]$ depend on $\mathcal{D}$ and λ ?

Level-based Analysis

Problem

- ▶ Search space $\mathcal{X}$ and a “target region” $B \subseteq \mathcal{X}$ (e.g. optima)
- ▶ What is the expected time to reach the target set B ?
More precisely, how does $\mathbf{E}[T_B]$ depend on $\mathcal{D}$ and λ ?

Level-based theorem

- ▶ Gives upper bound on $\mathbf{E}[T_B]$ if $\mathcal{D}$ satisfies some conditions.
- ▶ The theorem is both highly general and precise.
The upper bounds are nearly tight (Corus et al., 2016).
- ▶ Used to *analyse* the runtime of EAs, often in complex settings
 - ▶ e.g., genetic algorithms, estimation of distribution algorithms, self-adaptation, noisy optimisation, rugged fitness landscapes...
- ▶ and to *design* efficient EAs for a given problem
 - ▶ e.g., choose appropriate parameterisation of the algorithm, design GA for shortest paths problem (Corus & Lehre, MIC'15)

Level Partitioning of Search Space $\mathcal{X}$

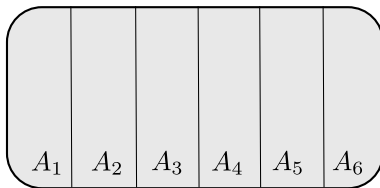
Definition

$(A_1, \dots, A_m)$ is a **level-partitioning** of search space $\mathcal{X}$ if

- ▶ $\bigcup_{j=1}^m A_j = \mathcal{X}$ (together, the levels cover the search space)
- ▶ $A_i \cap A_j = \emptyset$ whenever $i \neq j$ (the levels are nonoverlapping)
- ▶ the last level A_m covers the optima for the problem

We write $A_{\geq j}$ to denote everything in level j and higher, i.e.,

$$A_{\geq j} := \bigcup_{i=j}^m A_i.$$



Current level of a population P wrt $\gamma_0 \in (0, 1)$

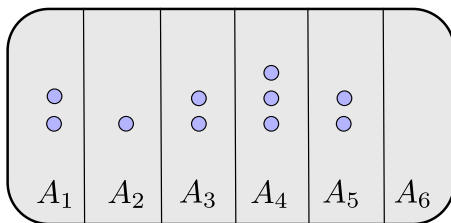
Definition

The unique integer $j \in [m - 1]$ such that

$$|P \cap A_{\geq j}| \geq \gamma_0 \lambda > |P \cap A_{\geq j+1}|$$

Example

Current level wrt $\gamma_0 = \frac{1}{2}$ is



Current level of a population P wrt $\gamma_0 \in (0, 1)$

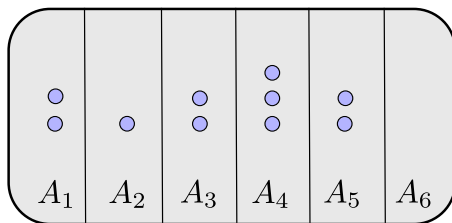
Definition

The unique integer $j \in [m - 1]$ such that

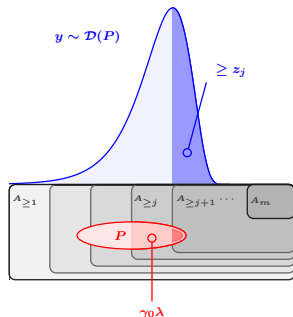
$$|P \cap A_{\geq j}| \geq \gamma_0 \lambda > |P \cap A_{\geq j+1}|$$

Example

Current level wrt $\gamma_0 = \frac{1}{2}$ is 4.



Level-based Theorem [Corus, Dang, Ereemeev, and Lehre, 2018]



If for all populations $P \in \mathcal{X}^\lambda$, an individual $y \sim \mathcal{D}(P)$ has

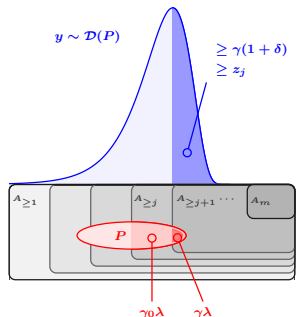
$$\Pr(y \in A_{\geq j+1}) \geq z_j, \quad (\text{G1})$$

(G2)

where $j \in [m-1]$ is the current level of population P , i.e.,

$$|P \cap A_{\geq j}| \geq \gamma_0 \lambda > |P \cap A_{\geq j+1}|$$

Level-based Theorem [Corus et al., 2018]



If for all populations $P \in \mathcal{X}^\lambda$, an individual $y \sim \mathcal{D}(P)$ has

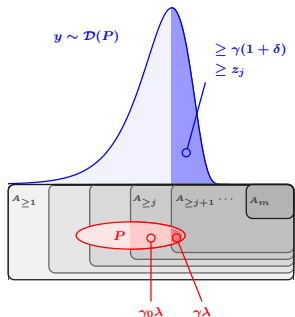
$$\Pr(y \in A_{\geq j+1}) \geq z_j, \quad (\text{G1})$$

$$\Pr(y \in A_{\geq j+1}) \geq \gamma(1 + \delta), \quad (\text{G2})$$

where $j \in [m - 1]$ is the current level of population P , i.e.,

$$|P \cap A_{\geq j}| \geq \gamma_0\lambda > |P \cap A_{\geq j+1}| = \gamma\lambda,$$

Level-based Theorem [Corus et al., 2018]



If for all populations $P \in \mathcal{X}^\lambda$, an individual $y \sim \mathcal{D}(P)$ has

$$\Pr(y \in A_{\geq j+1}) \geq z_j, \quad (\text{G1})$$

$$\Pr(y \in A_{\geq j+1}) \geq \gamma(1 + \delta), \quad (\text{G2})$$

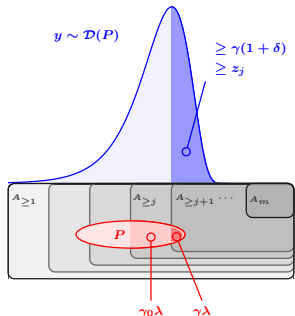
where $j \in [m - 1]$ is the current level of population P , i.e.,

$$|P \cap A_{\geq j}| \geq \gamma_0\lambda > |P \cap A_{\geq j+1}| = \gamma\lambda,$$

and the population size λ is bounded from below by

$$\lambda \geq \left(\frac{4}{\gamma_0\delta^2} \right) \ln \left(\frac{128m}{z_{\min}\delta^2} \right), \quad (\text{G3})$$

Level-based Theorem [Corus et al., 2018]



If for all populations $P \in \mathcal{X}^\lambda$, an individual $y \sim \mathcal{D}(P)$ has

$$\Pr(y \in A_{\geq j+1}) \geq z_j, \quad (\text{G1})$$

$$\Pr(y \in A_{\geq j+1}) \geq \gamma(1 + \delta), \quad (\text{G2})$$

where $j \in [m - 1]$ is the current level of population P , i.e.,

$$|P \cap A_{\geq j}| \geq \gamma_0\lambda > |P \cap A_{\geq j+1}| = \gamma\lambda,$$

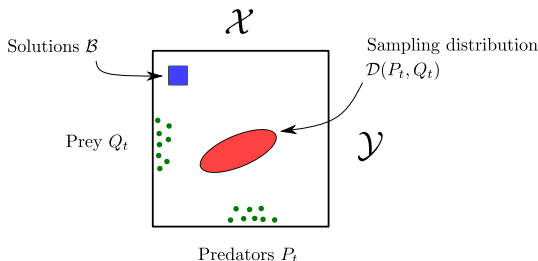
and the population size λ is bounded from below by

$$\lambda \geq \left(\frac{4}{\gamma_0\delta^2} \right) \ln \left(\frac{128m}{z_{\min}\delta^2} \right), \quad (\text{G3})$$

then the algorithm reaches the last level A_m in expected time

$$\mathbb{E}[T_{A_m}] \leq \left(\frac{8}{\delta^2} \right) \sum_{j=1}^{m-1} \left(\lambda \ln \left(\frac{6\delta\lambda}{4 + z_j\delta\lambda} \right) + \frac{1}{z_j} \right).$$

Co-Evolutionary Processes¹⁰



co-Algorithm A

Require: Strategy spaces $\mathcal{X}, \mathcal{Y}$.

Require: Populations $P_0 \in \mathcal{X}^\lambda$, $Q_0 \in \mathcal{Y}^\lambda$.

- 1: **for** each generation $t \in \mathbb{N}_0$ **do**
- 2: **for** each interaction $i \in [\lambda]$ **do**
- 3: Sample $(x, y) \sim \mathcal{D}(P_t, Q_t)$.
- 4: Set i -th “predator” $P_{t+1}(i) := x$
- 5: Set i -th “prey” $Q_{t+1}(i) := y$.

Definition (Runtime)

For a solution set $\mathcal{B} \subset \mathcal{X} \times \mathcal{Y}$

$$T_{A, \mathcal{B}} := \min\{t\lambda \mid (P_t \times Q_t) \cap \mathcal{B} \neq \emptyset\}$$

¹⁰This definition of runtime is slightly different from query complexity: 1. The algorithm only evaluates the payoff values of the pairs (x_i, y_i) for $i \in [\lambda]$. Hence, the event $(P_t \times Q_t) \cap \mathcal{B} \neq \emptyset$ does not guarantee that the algorithm has evaluated the payoff of a pair $(x, y) \in \mathcal{B}$. However, given an algorithm A with runtime $T_{A, \mathcal{B}}$, we can define an algorithm A' which behaves exactly as A , except that at the end of every generation, it evaluates the payoff for all pairs $(x, y) \in P_t \times Q_t$. This algorithm has query complexity τ of at most $(1 + \lambda)T_{A, \mathcal{B}}$. 2. An optimal pair may have been found in the first iteration of generation t of algorithm A' , but the definition counts all payoff-evaluations in the generation. Hence, in the worst case, the query complexity of A' is by at most an additive term $\lambda - 1$ larger than the true runtime. Hence, in overall, we have $\tau / (1 + \lambda) \leq T_{A, \mathcal{B}} \leq \tau + \lambda$.

Theorem (Level-based Theorem for Co-Evolution)

Given

- ▶ “predator levels” $A_1, \dots, A_m \subseteq \mathcal{X}$ with $A_1 = \mathcal{X}$
- ▶ “prey levels” $B_1, \dots, B_m \subseteq \mathcal{Y}$ with $B_1 = \mathcal{Y}$
- ▶ “upgrade” probabilities $z_1, \dots, z_{m-1} \in (0, 1]$
- ▶ parameters $\delta, \gamma_0 \in (0, 1)$

For any $P \in \mathcal{X}^\lambda$ and $Q \in \mathcal{Y}^\lambda$, define

- ▶ “Current level” $j := \max\{i : |(P \times Q) \cap (A_i \times B_i)| \geq \gamma_0 \lambda^2\}$
- ▶ $\gamma := \lambda^{-2} |(P \times Q) \cap (A_{j+1} \times B_{j+1})|$

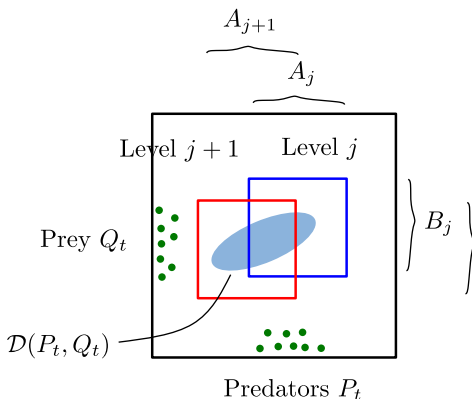
If for all $P \in \mathcal{X}^\lambda$ and $Q \in \mathcal{Y}^\lambda$, the sample $(x, y) \sim \mathcal{D}(P, Q)$ satisfies

- (G1) $\Pr(x \in A_{j+1}) \Pr(y \in B_{j+1}) \geq z_j$
- (G2a) $\Pr(x \in A_{j+1}) \Pr(y \in B_{j+1}) \geq (1 + \delta)\gamma$
- (G2b) $\Pr(x \in A_j) \Pr(y \in B_j) \geq (1 + \delta)\gamma_0$,
- (G3) pop. size $\lambda \geq \max_{j \in [m-1]} 2 \left(\frac{1}{\gamma_0 \delta^2} \right)^{1+\eta} \ln\left(\frac{m}{z_j}\right)$,

then $\mathbf{E}[T_{A_m \times B_m}] \leq \frac{c'' \lambda}{\delta} \lambda \left(\lambda^2 m + 16 \sum_{i=1}^{m-1} \frac{1}{z_i} \right)$, for any constant $c'' > 0$,

Lehre (Algorithmica, 2024)

(G2a)-condition



For $0 < \gamma < \gamma_0 < 1$, if

$$\gamma_0 \lambda^2 \leq |(\mathbf{P}_t \times \mathbf{Q}_t) \cap (\mathbf{A}_j \times \mathbf{B}_j)|,$$

$$\gamma \lambda^2 = |(\mathbf{P}_t \times \mathbf{Q}_t) \cap (\mathbf{A}_{j+1} \times \mathbf{B}_{j+1})|$$

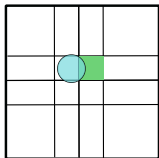
then (G2a) requires

$$B_{j+1} \Pr(x \in A_{j+1}) \Pr(y \in B_{j+1}) \geq (1 + \delta) \gamma$$

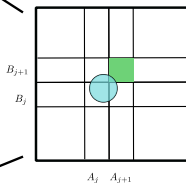
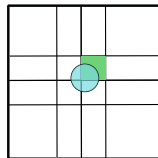
where $(x, y) \sim \mathcal{D}(P_t, Q_t)$.

Transitioning to the Next Level: Four Scenarios

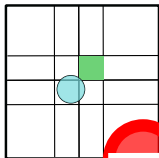
Pred pop declines
Prey pop grows



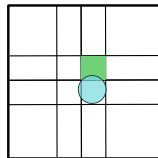
Pred pop grows
Prey pop grows



Pred decline
Prey decline



Pred pop grows
Prey pop declines



When is condition (G2a) satisfied?

Assume that

1. Populations at level j with

$$|P_t \cap A_{j+1}| = \gamma_x \lambda$$

$$|Q_t \cap B_{j+1}| = \gamma_y \lambda$$

2. Offspring distribution $(x, y) \sim \mathcal{D}(P_t, Q_t)$

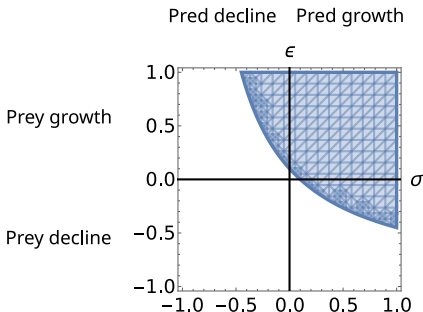
$$\Pr(x \in A_{j+1}) = \gamma_x(1 + \varepsilon)$$

$$\Pr(y \in B_{j+1}) = \gamma_y(1 + \sigma)$$

Then (G2a) is satisfied if for some $\delta > 0$

$$\gamma_x(1 + \varepsilon)\gamma_y(1 + \sigma) \geq \gamma_x\gamma_y(1 + \delta)$$

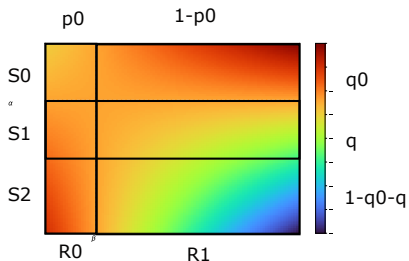
$$\implies \sigma > -\frac{\varepsilon}{1 + \varepsilon}$$



Conclusion

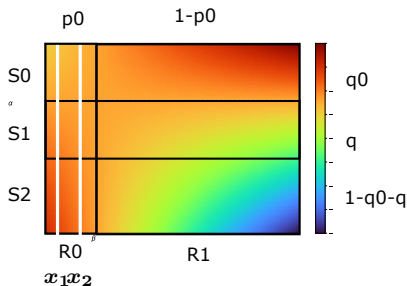
- ▶ (G2a) not satisfied when both populations have negative growth rates.
- ▶ (G2a) satisfied when both populations have positive growth rates.
- ▶ (G2a) can be satisfied with one growing and one declining population
 - ▶ Negative predator growth rate ε , positive prey growth rate $\sigma > -\frac{\varepsilon}{1+\varepsilon} > 0$
 - ▶ Positive predator growth rate ε , negative prey growth rate $0 > \sigma > -\frac{\varepsilon}{1+\varepsilon}$

Example calculation



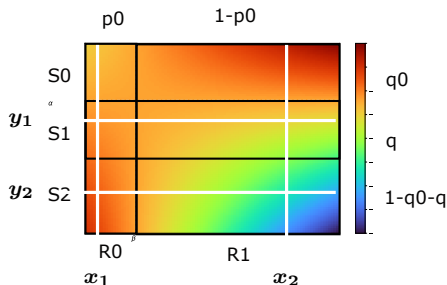
$$\begin{aligned}
 p_{\text{sel}}(R_0) = & \Pr(x_1 \in R_0 \wedge x_2 \in R_0) + \\
 & + \Pr(x_1 \in R_0 \wedge x_2 \notin R_0) \times \Pr((x_1, y_1) \succeq (x_2, y_2) \mid x_1 \in R_0 \wedge x_2 \notin R_0) \\
 & + \Pr(x_1 \notin R_0 \wedge x_2 \in R_0) \times (1 - \Pr((x_1, y_1) \succeq (x_2, y_2) \mid x_1 \notin R_0 \wedge x_2 \in R_0))
 \end{aligned}$$

Example calculation



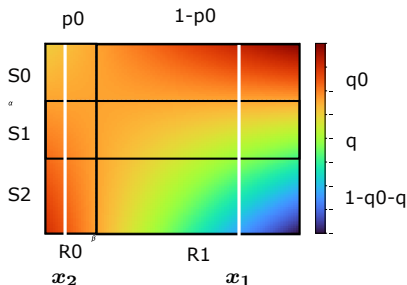
$$\begin{aligned}
 p_{\text{sel}}(R_0) &= \Pr(x_1 \in R_0 \wedge x_2 \in R_0) + \\
 &\quad + \Pr(x_1 \in R_0 \wedge x_2 \notin R_0) \times \Pr((x_1, y_1) \succeq (x_2, y_2) \mid x_1 \in R_0 \wedge x_2 \notin R_0) \\
 &\quad + \Pr(x_1 \notin R_0 \wedge x_2 \in R_0) \times (1 - \Pr((x_1, y_1) \succeq (x_2, y_2) \mid x_1 \notin R_0 \wedge x_2 \in R_0)) \\
 &\geq \Pr(x_1 \in R_0 \wedge x_2 \in R_0) +
 \end{aligned}$$

Example calculation



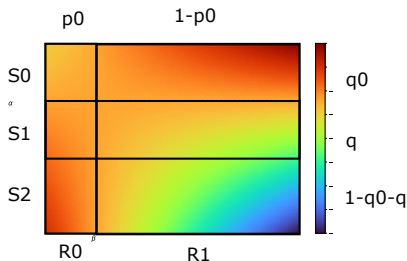
$$\begin{aligned}
 p_{\text{sel}}(R_0) &= \Pr(x_1 \in R_0 \wedge x_2 \in R_0) + \\
 &\quad + \Pr(x_1 \in R_0 \wedge x_2 \notin R_0) \times \Pr((x_1, y_1) \succeq (x_2, y_2) \mid x_1 \in R_0 \wedge x_2 \notin R_0) \\
 &\quad + \Pr(x_1 \notin R_0 \wedge x_2 \in R_0) \times (1 - \Pr((x_1, y_1) \succeq (x_2, y_2) \mid x_1 \notin R_0 \wedge x_2 \in R_0)) \\
 &\geq \Pr(x_1 \in R_0 \wedge x_2 \in R_0) + \\
 &\quad + \Pr((x_1, y_1) \in R_0 \times S_1 \cup S_2 \wedge (x_2, y_2) \in R_1 \times S_1 \cup S_2) / 2
 \end{aligned}$$

Example calculation



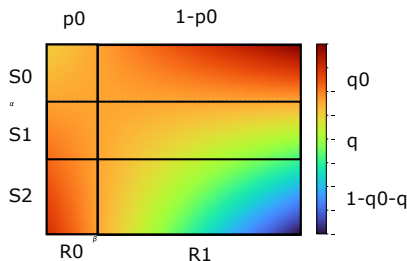
$$\begin{aligned}
 p_{\text{sel}}(R_0) &= \Pr(x_1 \in R_0 \wedge x_2 \in R_0) + \\
 &\quad + \Pr(x_1 \in R_0 \wedge x_2 \notin R_0) \times \Pr((x_1, y_1) \succeq (x_2, y_2) \mid x_1 \in R_0 \wedge x_2 \notin R_0) \\
 &\quad + \Pr(x_1 \notin R_0 \wedge x_2 \in R_0) \times (1 - \Pr((x_1, y_1) \succeq (x_2, y_2) \mid x_1 \notin R_0 \wedge x_2 \in R_0)) \\
 &\geq \Pr(x_1 \in R_0 \wedge x_2 \in R_0) + \\
 &\quad + \Pr((x_1, y_1) \in R_0 \times S_1 \cup S_2 \wedge (x_2, y_2) \in R_1 \times S_1 \cup S_2) / 2 \\
 &\quad + \Pr(x_1 \notin R_0 \wedge x_2 \in R_0) (1 - \Pr(y_1 \in S_0 \wedge y_2 \in S_0))
 \end{aligned}$$

Example calculation



$$\begin{aligned}
 p_{\text{sel}}(R_0) &= \Pr(x_1 \in R_0 \wedge x_2 \in R_0) + \\
 &\quad + \Pr(x_1 \in R_0 \wedge x_2 \notin R_0) \times \Pr((x_1, y_1) \succeq (x_2, y_2) \mid x_1 \in R_0 \wedge x_2 \notin R_0) \\
 &\quad + \Pr(x_1 \notin R_0 \wedge x_2 \in R_0) \times (1 - \Pr((x_1, y_1) \succeq (x_2, y_2) \mid x_1 \notin R_0 \wedge x_2 \in R_0)) \\
 &\geq \Pr(x_1 \in R_0 \wedge x_2 \in R_0) + \\
 &\quad + \Pr((x_1, y_1) \in R_0 \times S_1 \cup S_2 \wedge (x_2, y_2) \in R_1 \times S_1 \cup S_2) / 2 \\
 &\quad + \Pr(x_1 \notin R_0 \wedge x_2 \in R_0) (1 - \Pr(y_1 \in S_0 \wedge y_2 \in S_0)) \\
 &\geq p_0^2 + p_0(1-p_0)(1-q_0)^2/2 + p_0(1-p_0)(1-q_0^2) \\
 &= \frac{p_0}{2} ((3+q_0)(1-q_0) - p_0(1-q_0(2+q_0)))
 \end{aligned}$$

Checking (G2) of PDCoEA on Bilinear in level $R_0 \times S_1$



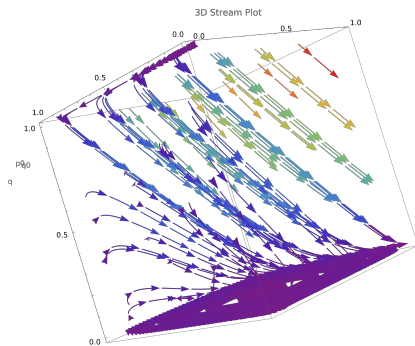
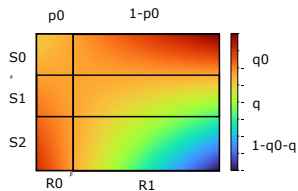
Lemma

$$\frac{p_{sel}(R_0)}{p_0} \frac{p_{sel}(S_1)}{q} \geq \frac{1}{4} (2q + 3(2 - p_0)p_0(1 - q) - 8q_0) \times (3 - p_0 - 4q_0)$$

Furthermore, if $1/3 < p_0 < 4/5$ then for sufficiently small q_0, χ, δ

$$\frac{p_{sel}(R_0)}{p_0} \frac{p_{sel}(S_1)}{q} \left(1 - \frac{\chi}{n}\right)^{2n} > \left(\frac{11}{12} - 12q_0\right) e^{-2\chi} > 1 + \delta$$

Dynamics



Conclusion

- ▶ Real world optimisation is often adversarial: playing against opponents
- ▶ CoEAs may have a niche in adversarial optimisation scenarios: games with exponentially many strategies and payoff functions with (hidden) structures
- ▶ Vast design space for CoEAs
 - ▶ population structure, interaction schemes, fitness aggregation, ...
- ▶ Design of CoEAs traditionally focused on overcoming pathological behaviours
 - ▶ Population archives to avoid forgetting
- ▶ Algorithms with Monotonic Progress wrt a Solution Concept (IPCA)
 - ▶ Guaranteed to find solution, but may take exceedingly long time to do so
- ▶ Algorithms with Positive Drift (PDCoEA, DivRank-CoEA, ...)
 - ▶ Motivated by insights from runtime analysis of classical EAs
 - ▶ Sufficient to select individuals correctly more than 50% of the time
- ▶ Runtime Analysis of CoEAs (from 2022)
 - ▶ Bounds on runtime (num payoff evals) as function of number of strategies (and other game characteristics) and algorithmic parameters
 - ▶ Obtained using tools: drift analysis, level-based analysis

Acknowledgements

Thanks to EC theory group members in Birmingham contributing to our project on co-evolution, including Alistair Benford, Mario Alejandro Hevia Fajardo, and Shishen Lin. The research was funded by a Turing AI Acceleration Fellowship (EPSRC grant ref EP/V025562/1).

References I

- Yakov Babichenko. Informational bounds on equilibria (a survey). *ACM SIGecom Exchanges*, 17(2):25–45, January 2020. doi: 10.1145/3381329.3381333. URL <https://dl.acm.org/doi/10.1145/3381329.3381333>.
- Yakov Babichenko and Aviad Rubinfeld. Communication complexity of nash equilibrium in potential games (extended abstract). In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1439–1445, 2020. doi: 10.1109/FOCS46700.2020.00137.
- Alistair Benford and Per Kristian Lehre. Runtime analysis of coevolutionary algorithms on a class of symmetric zero-sum games. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '24*, page 1542–1550, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400704949. doi: 10.1145/3638529.3654216. URL <https://doi.org/10.1145/3638529.3654216>.
- Vincent Conitzer and Tuomas Sandholm. Communication complexity as a lower bound for learning in games. In *Proceedings of the twenty-first international conference on Machine learning, ICML '04*, page 24, New York, NY, USA, July 2004. Association for Computing Machinery. ISBN 978-1-58113-838-2. doi: 10.1145/1015330.1015351. URL <https://dl.acm.org/doi/10.1145/1015330.1015351>.
- Dogan Corus, Duc-Cuong Dang, Anton V. Eremeev, and Per Kristian Lehre. Level-Based Analysis of Genetic Algorithms and Other Search Processes. *IEEE Transactions on Evolutionary Computation*, 22(5):707–719, October 2018. ISSN 1941-0026. doi: 10.1109/TEVC.2017.2753538.

References II

- Duc-Cuong Dang and Per Kristian Lehre. Runtime analysis of non-elitist populations: From classical optimisation to partial information. *Algorithmica*, 75(3):428–461, 2016. doi: 10.1007/S00453-015-0103-X. URL <https://doi.org/10.1007/s00453-015-0103-x>.
- Edwin D. De Jong. The incremental pareto-coevolution archive. In Kalyanmoy Deb, editor, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '04*, pages 525–536, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. ISBN 978-3-540-24854-5.
- Benjamin Doerr and Frank Neumann, editors. *Theory of Evolutionary Computation - Recent Developments in Discrete Optimization*. Natural Computing Series. Springer, 2020. ISBN 978-3-030-29413-7. doi: 10.1007/978-3-030-29414-4. URL <https://doi.org/10.1007/978-3-030-29414-4>.
- John Fearnley, Martin Gairing, Paul Goldberg, and Rahul Savani. Learning equilibria of games via payoff queries. In *Proceedings of the Fourteenth ACM Conference on Electronic Commerce, EC '13*, page 397–414, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450319621. doi: 10.1145/2482540.2482558. URL <https://doi.org/10.1145/2482540.2482558>.
- Sevan G. Ficici and Jordan B. Pollack. A game-theoretic memory mechanism for coevolution. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '03*, page 286–297, Berlin, Heidelberg, 2003. Springer-Verlag. ISBN 3540406026.
- Jun He and Xin Yao. Drift analysis and average time complexity of evolutionary algorithms. *Artificial Intelligence*, 127(1):57–85, March 2001a.

References III

- Jun He and Xin Yao. Drift analysis and average time complexity of evolutionary algorithms. *Artif. Intell.*, 127(1):57–85, 2001b. doi: 10.1016/S0004-3702(01)00058-3. URL [https://doi.org/10.1016/S0004-3702\(01\)00058-3](https://doi.org/10.1016/S0004-3702(01)00058-3).
- Erik Hemberg, Jamal Toutouh, Abdullah Al-Dujaili, Tom Schmiedlechner, and Una-May O'Reilly. Spatial coevolution for generative adversarial network training. *ACM Trans. Evol. Learn. Optim.*, 1(2), jul 2021. doi: 10.1145/3458845. URL <https://doi.org/10.1145/3458845>.
- Mario Hevia Fajardo, Per Kristian Lehre, Jamal Toutouh, Erik Hemberg, and Una-May O'Reilly. *Analysis of a Pairwise Dominance Coevolutionary Algorithm with Spatial Topology*, pages 19–44. Springer Nature Singapore, Singapore, 2024. ISBN 978-981-99-8413-8. doi: 10.1007/978-981-99-8413-8_2. URL https://doi.org/10.1007/978-981-99-8413-8_2.
- Mario Alejandro Hevia Fajardo and Per Kristian Lehre. How fitness aggregation methods affect the performance of competitive coeas on bilinear problems. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '23*, page 1593–1601, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400701191.
- Mario Alejandro Hevia Fajardo and Per Kristian Lehre. Ranking diversity benefits coevolutionary algorithms on an intransitive game. In *Parallel Problem Solving from Nature – PPSN XVIII*, Cham, 2024. Springer International Publishing. ISBN 978-3-031-14714-2.

References IV

- Mario Alejandro Hevia Fajardo, Per Kristian Lehre, and Shishen Lin. Runtime analysis of a co-evolutionary algorithm: Overcoming negative drift in maximin-optimisation. In *Proceedings of the 17th ACM/SIGEVO Conference on Foundations of Genetic Algorithms*, FOGA '23, page 73–83, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400702020. doi: 10.1145/3594805.3607132. URL <https://doi.org/10.1145/3594805.3607132>.
- W.Daniel Hillis. Co-evolving parasites improve simulated evolution as an optimization procedure. *Physica D: Nonlinear Phenomena*, 42(1):228–234, 1990. ISSN 0167-2789. doi: [https://doi.org/10.1016/0167-2789\(90\)90076-2](https://doi.org/10.1016/0167-2789(90)90076-2). URL <https://www.sciencedirect.com/science/article/pii/0167278990900762>.
- Jens Jägersküpper. Algorithmic analysis of a basic evolutionary algorithm for continuous optimization. *Theoretical Computer Science*, 379(3):329–347, 2007. ISSN 0304-3975. doi: <http://dx.doi.org/10.1016/j.tcs.2007.02.042>.
- Jens Jägersküpper. A blend of markov-chain and drift analysis. In *Proceedings of the 10th International Conference on Parallel Problem Solving from Nature (PPSN'2008)*, 2008. URL archive/2008/jagerskupper2008driftmc.pdf.
- Thomas Jansen and R. Paul Wiegand. The cooperative coevolutionary (1+1) EA. *Evol. Comput.*, 12(4):405–434, 2004. doi: 10.1162/1063656043138905. URL <https://doi.org/10.1162/1063656043138905>.
- Wojciech Jaśkowski, Krzysztof Krawiec, and Bartosz Wieloch. Winning ant wars: evolving a human-competitive game strategy using fitnessless selection. In *Proceedings of the 11th European Conference on Genetic Programming, EuroGP'08*, page 13–24, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 3540786708.

References V

- Krzysztof Krawiec and Malcolm Heywood. Solving complex problems with coevolutionary algorithms. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*, GECCO '20, page 832–858, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450371278. doi: 10.1145/3377929.3389874.
- Per Kristian Lehre. Runtime analysis of competitive co-evolutionary algorithms for maximin optimisation of a bilinear function. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '22, page 1408–1416, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450392372. doi: 10.1145/3512290.3528853. URL <https://doi.org/10.1145/3512290.3528853>.
- Per Kristian Lehre and Shishen Lin. Overcoming binary adversarial optimisation with competitive coevolution. In *Proceedings of Parallel Problem Solving from Nature*, 2024.
- Arnab Maiti, Ross Boczar, Kevin Jamieson, and Lillian J. Ratliff. Query-Efficient Algorithms to Find the Unique Nash Equilibrium in a Two-Player Zero-Sum Matrix Game, November 2023. URL <http://arxiv.org/abs/2310.16236>. arXiv:2310.16236 [cs].
- Thomas Miconi. Why coevolution doesn't "work": Superiority and progress in coevolution. In Leonardo Vanneschi, Steven Gustafson, Alberto Moraglio, Ivanoe De Falco, and Marc Ebner, editors, *Genetic Programming*, pages 49–60, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. ISBN 978-3-642-01181-8.

References VI

- Elena Popovici, Anthony Bucci, R. Paul Wiegand, and Edwin D. de Jong. Coevolutionary principles. In Grzegorz Rozenberg, Thomas Bäck, and Joost N. Kok, editors, *Handbook of Natural Computing*, pages 987–1033. Springer, 2012. doi: 10.1007/978-3-540-92910-9_31. URL https://doi.org/10.1007/978-3-540-92910-9_31.
- Christopher D. Rosin and Richard K. Belew. Methods for competitive co-evolution: Finding opponents worth beating. In *Proceedings of the International Conference on Genetic Algorithms*, pages 373–381, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc. ISBN 978-1-55860-370-7.
- Christopher D. Rosin and Richard K. Belew. New methods for competitive coevolution. *Evolutionary Computation*, 5(1):1–29, March 1997. ISSN 1063-6560, 1530-9304.
- Aviad Rubinstein. Settling the Complexity of Computing Approximate Two-Player Nash Equilibria. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 258–265, October 2016. doi: 10.1109/FOCS.2016.35. URL <https://ieeexplore.ieee.org/document/7782938>. ISSN: 0272-5428.