

Ranking Diversity Benefits Coevolutionary Algorithms on an Intransitive Game

Hevia Fajardo, Mario; Lehre, Per Kristian

DOI:

[10.1007/978-3-031-70071-2_14](https://doi.org/10.1007/978-3-031-70071-2_14)

License:

Other (please specify with Rights Statement)

Document Version

Peer reviewed version

Citation for published version (Harvard):

Hevia Fajardo, M & Lehre, PK 2024, Ranking Diversity Benefits Coevolutionary Algorithms on an Intransitive Game. in M Affenzeller, SM Winkler, AV Kononova, H Trautmann, T Tušar, P Machado & T Bäck (eds), Parallel Problem Solving from Nature – PPSN XVIII: 18th International Conference, PPSN 2024, Hagenberg, Austria, September 14–18, 2024, Proceedings, Part III. vol. 3, Lecture Notes in Computer Science, vol. 15150, Springer, 18th International Conference on Parallel Problem Solving From Nature PPSN 2024, Hagenberg, Austria, 14/09/24. https://doi.org/10.1007/978-3-031-70071-2_14

[Link to publication on Research at Birmingham portal](#)

Publisher Rights Statement:

This version of the article has been accepted for publication, after peer review (when applicable) and is subject to Springer Nature's AM terms of use, but is not the Version of Record and does not reflect post-acceptance improvements, or any corrections. The Version of Record is available online at: [10.1007/978-3-031-70071-2_14](https://doi.org/10.1007/978-3-031-70071-2_14)

General rights

Unless a licence is specified above, all rights (including copyright and moral rights) in this document are retained by the authors and/or the copyright holders. The express permission of the copyright holder must be obtained for any use of this material other than for purposes permitted by law.

- Users may freely distribute the URL that is used to identify this publication.
- Users may download and/or print one copy of the publication from the University of Birmingham research portal for the purpose of private study or non-commercial research.
- User may use extracts from the document in line with the concept of 'fair dealing' under the Copyright, Designs and Patents Act 1988 (?)
- Users may not further distribute the material nor use it for the purposes of commercial gain.

Where a licence is displayed above, please note the terms and conditions of the licence govern your use of this document.

When citing, please reference the published version.

Take down policy

While the University of Birmingham exercises care and attention in making items available there are rare occasions when an item has been uploaded in error or has been deemed to be commercially or otherwise sensitive.

If you believe that this is the case for this document, please contact UBIRA@lists.bham.ac.uk providing details and we will remove access to the work immediately and investigate.

Ranking Diversity Benefits Coevolutionary Algorithms on an Intransitive Game

Mario Alejandro Hevia Fajardo¹[0000–0003–3529–0434] and Per Kristian Lehre¹[0000–0002–9521–1251]

University of Birmingham, Birmingham B15 2TT, UK

Abstract. Competitive coevolutionary algorithms (CoEAs) often encounter so-called coevolutionary pathologies particularly cycling behavior, which becomes more pronounced for games where there is no clear hierarchy of superiority among the possible strategies (intransitive games). In order to avoid these pathologies and ensure an efficient optimisation, it has been suggested that it is critical to choose a *good* evaluation environment (set of solutions used for evaluation).

In this paper, we use runtime analysis to increase our understanding of the essential characteristics that the evaluation environments should possess to ensure efficient runtime on the intransitive problem class $\text{BILINEAR}_{\alpha,\beta}$. For this problem class, we observe that it is beneficial to maintain a high *diversity of rankings* in the evaluation environment, that is, a set of individuals used for evaluation which are diverse in how they rank opponents.

We propose and analyse two mechanisms that implement this idea. In the first approach, we ensure diversity of rankings through an archive. In the second approach, we introduce a CoEA without an archive, but with a ranking diversity mechanism. Both approaches optimise $\text{BILINEAR}_{\alpha,\beta}$ in expected polynomial time.

Keywords: Runtime analysis · Competitive coevolution · Archives · Maximin optimisation.

1 Introduction

Within the field of Evolutionary Computation, coevolutionary algorithms stand out for harnessing the dynamic interplay between solutions (individuals) to tackle optimisation problems where computing the fitness of isolated individuals is infeasible [17]. Coevolutionary algorithms are categorized into cooperative and competitive variants based on the nature of interactions among the evolved solutions. The focus of this work is on competitive coevolutionary algorithms.

Competitive coevolutionary algorithms (we omit the competitive label and simply call them CoEAs), simultaneously evolve one or more populations of solutions that compete and adapt to one another. CoEAs do not rely on a global fitness function to assign fitness values to sampled solutions. Instead, they use the aggregation of outcomes from interactions between competing solutions to

rank solutions and make selection decisions. Thus, this optimisation technique is well suited for problems that have intrinsically competitive domains or for problems where finding a suitable global fitness measure is hard. Accordingly, in this work, we use coevolutionary algorithms to find a maximin-optimum, that is, a pair of solutions (x, y) such that the performance of x against the least favourable y is the best, and the same for y .

Despite their success at solving non-trivial problems (e.g. zero-sum games [1,29], software engineering [14,2], and generative models [10,11]) CoEAs often encounter so-called *coevolutionary pathologies* [30,21]. This is particularly likely if the problem is intransitive, that is, if there is no clear hierarchy of superiority for all possible solutions [17]. Aggravating the situation, Czarnecki et al. [3] showed that real-world games tend to have intransitive dynamics.

In order to avoid these pathologies and ensure an efficient optimisation, it has been suggested that it is critical to choose a *good* evaluation environment (set of opponents) [27,22,17]. The rationale for this is that CoEAs use a varying set of opponents to assign fitness and select individuals. This may result in apparent progress (*local* progress) against a poorly selected set of opponents. However, this does not necessarily translate into real progress toward the optimal solution when comparing against the entire set of possible opponents (*global* progress) [22]. Therefore, a *good* evaluation environment that represents all possible opponents could enable global progress and as a consequence an efficient optimisation.

Previous studies have focused on guaranteeing monotonic global progress (see Section 2). But Popovici et al. [26] warned that monotonic progress does not guarantee an efficient optimisation and recommends instead to focus on studying the performance of algorithms and in particular the authors propose the use of runtime analysis to understand CoEAs better.

We aim to use runtime analysis to increase our understanding of the key characteristics required for evaluation environments to ensure efficient runtime on intransitive problems. In particular, we study the class of intransitive problems called $\text{BILINEAR}_{\alpha,\beta}$ [18]. $\text{BILINEAR}_{\alpha,\beta}$ is a challenging class of maximin-problems where CoEAs have been shown to lack a clear signal towards the optimal solutions during most of the optimisation steps and the algorithms tend to cycle around the optimum due to its intransitive properties [13,12]. We aim to find the properties of an evaluation environment that gives a clear signal towards the optimal solution, accelerating the optimisation and avoiding the cycling behaviour characteristic of these intransitive problems.

We start our analyses by isolating the evaluation environments, separating the search populations from the evaluation environments using archive populations (Archived Tournament Selection CoEA – Algorithm 1). Doing this, we observe that it is beneficial to maintain a high *diversity of rankings* in the evaluation environment, that is, given that the search populations are ranked (ordered) by the payoff when interacting with an opponent, the individuals in the evaluation environment rank these solutions in a diverse way.

Given this insight, we propose two mechanisms. In the first approach, we ensure diversity of rankings through an archive update scheme that maintains a

diverse evaluation environment. In the second approach, we introduce a CoEA without an archive that we call RankDivCoEA (Ranking Diversity CoEA). This algorithm implements an intuitive ranking diversity mechanism that maintains solutions in the current populations that rank their competitors differently without the overhead of maintaining an archive.

In Section 5 we then analyse ATS-CoEA with our proposed archive update scheme and show that ATS-CoEA solves $\text{BILINEAR}_{\alpha,\beta}$ for all α and β in $O(\lambda^4 n)$ expected function evaluations, with λ representing the population size and n the problem size. Finally, in Section 6 we show that RankDivCoEA solves $\text{BILINEAR}_{\alpha,\beta}$ with $\alpha n, \beta n \notin \mathbb{Z}$ in $O(\lambda^3 n)$ expected function evaluations.

2 Related Work

Archives A common strategy for maintaining a robust evaluation environment in CoEAs involves the use of archives. An archive is a set of solutions that is systematically updated throughout the coevolutionary process and used to compare evolved solutions. The aim is that the archives accumulate knowledge of the problem and maintain diversity and progress [17].

Rosin and Belew [28] proposed the *hall of fame* archive, one of the first and simplest archives for CoEAs. The hall of fame is an unbounded archive that adds individuals that are successful against the current population and a subset of the archive. This helps CoEAs select individuals that outperform their ancestors. A simple extension of the hall of fame where the archive is bounded was made by Nolfi and Floreano [24].

Ficici and Pollack [9] proposed an unbounded archive built upon the game-theoretic principle of Nash Equilibrium called Nash memory. This archive guarantees global monotonic progress in symmetric zero-sum games [9,8]. Oliehoek et al. [25] extended it to asymmetric games maintaining the progress guarantee.

De Jong [4] introduced an unbounded archived CoEA called IPCA (Incremental Pareto Coevolution Archive). In IPCA one population is considered as a set of objectives in the sense of Evolutionary Multi-Objective Optimisation and aims to find the Pareto-front. IPCA guarantees global monotonic progress. A bounded variant named LAPCA (LAYered Pareto Coevolution Archive) was later proposed by De Jong [5] without the monotonicity guarantees. Based on the same concept Yang et al. [31] proposed EPCA (Efficient Pareto Coevolution Archive) showing experimentally that it can outperform IPCA.

Another approach when using archives is to extract dimension information (underlying objectives implicitly defined) in order to create a suitable archive. De Jong and Bucci [6] proposed one of the first archives of this kind, named DECA (Dimension Extracting Coevolutionary Algorithm) and showed empirically that it is at least as good as IPCA and LAPCA for the problems tested. Based on this idea Yang et al. [32] and Jaśkowski and Krawiec [16] proposed SCA-BDE (Simple Coevolution Archive based on Bidirectional Dimension Extraction) and COSA (Coordinate System Archive) respectively, showing improved performances to all previous archives in the problems tested.

We remark that all mentioned archive mechanisms ensuring global monotonic progress require unlimited memory. Additionally, as shown in the later studies, archives without this guarantee can outperform these archives in practice.

Runtime Analysis of Coevolutionary Algorithms Due to their complexity, there is little rigorous understanding of the algorithm dynamics in (cooperative and competitive) CoEAs.

Cooperative CoEAs: Jansen and Wiegand [15] rigorously analysed the runtime of a cooperative CoEA on *separable* functions and showed that problem separability does not guarantee a speedup over traditional EAs. Lehre and Lin [19] showed that the cooperative CoEA CC-(1 + 1) EA solves all linear functions in the same asymptotic time as the (1 + 1) EA.

Competitive CoEAs: Lehre [18] analysed for the first time the runtime of a competitive CoEA, the PD-CoEA, on some instances of the pseudo-Boolean BILINEAR problem, showing that given the correct parameters the algorithm finds an ε -approximation efficiently, but a too high mutation rate leads to exponential runtime. Similarly, Hevia Fajardo and Lehre [12] analysed a $(1, \lambda)$ CoEA on a version of BILINEAR that uses an integer lattice as search space, showing that the algorithm finds a solution near the maximin-optima in polynomial time with high probability if the worst-case interaction is used to assign fitness (i.e. the fitness of a solution is its worst performance) but is inefficient if the average of all interactions is used instead.

Hevia Fajardo et al. [13] analysed the runtime and total regret of a CoEA named RLS-PD on some instances of a slight variation of the pseudo-Boolean BILINEAR problem. The authors showed that despite finding the optimum in $O(n^{1.5})$ the algorithm quickly forgets this solution and stays far away from it.

3 Preliminaries

We define $[n] := \{1, \dots, n\}$. We denote the 1-norm of a bit string $x \in \{0, 1\}^n$ by $\|x\| := \sum_{i=1}^n x_i$, i.e., the number of 1-bits in x . We use the function

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

We denote the power set of a set $\mathcal{S}$ as $\mathbb{P}(\mathcal{S})$.

Due to space constraints, we removed the proofs and provide proof sketches instead; the detailed proofs can be found in the supplementary material¹.

In this work we introduce two algorithms, ATS-CoEA (Algorithm 1) and RankDivCoEA (Algorithm 2). The algorithms are defined for payoff functions $g : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$ defined over arbitrary (finite) search spaces $\mathcal{X}$ and $\mathcal{Y}$.

Mutation Operator Both Algorithms 1 and 2 can use any mutation operators $\text{mut}_x(\cdot) : \mathcal{X} \rightarrow \mathcal{X}$, $\text{mut}_y(\cdot) : \mathcal{Y} \rightarrow \mathcal{Y}$. Given that for our analysis $\mathcal{X} = \mathcal{Y} = \{0, 1\}^n$,

¹ https://mariohevia.github.io/assets/pdf/PPSN_2024_sup_material.pdf

we consider $\text{mut}_x(\cdot) = \text{mut}_y(\cdot)$. Furthermore, we use the following unbiased mutation operator.

Definition 1 (Unbiased mutation operator $\text{mut}_{\mathcal{D}}$). *For a probability distribution $\mathcal{D}$ on $\{0\} \cup [n]$ with probabilities $r_{(0)}, \dots, r_{(n)} \geq 0$. Let $\text{mut}_{\mathcal{D}}$ be the mutation operator that samples $k \sim \mathcal{D}$ and then flips a uniform random set of exactly k positions.*

We note that any unary unbiased mutation operator (see [20]) can be expressed as $\text{mut}_{\mathcal{D}}$ (Lemma 1 in [7]).

Algorithm 1: Archived Tournament Selection CoEA (ATS-CoEA)

```

1 Require: Search spaces  $\mathcal{X}$  and  $\mathcal{Y}$ .
2 Require: Payoff function  $g : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$ .
3 Require: Population size  $\lambda \in \mathbb{N}$ .
4 Require: Tournament and competition size  $2 \leq k \in [\lambda] \ 2 \leq \ell \in [\lambda]$ .
5 Require:  $\text{mut}_x(\cdot) : \mathcal{X} \rightarrow \mathcal{X}$ ,  $\text{mut}_y(\cdot) : \mathcal{Y} \rightarrow \mathcal{Y}$ .
6 Require: Initial populations  $P_0 \in \mathcal{X}^\lambda$  and  $Q_0 \in \mathcal{Y}^\lambda$ .
7 Require: Initial archives  $V_{-1} \in \mathbb{P}(\mathcal{X})$  and  $W_{-1} \in \mathbb{P}(\mathcal{Y})$ .
8 Require: Archive update scheme  $\text{arch\_update}(\cdot, \cdot, \cdot, \cdot)$ .
9 for  $t \in \mathbb{N}_0$  until termination condition satisfied do
10    $V_t, W_t := \text{arch\_update}(V_{t-1}, W_{t-1}, P_t, Q_t)$ ;
11   for  $i \in [\lambda]$  do
12     Sample  $k$  predators  $x_1, \dots, x_k \sim \text{Unif}(P_t)$  i.i.d.;
13     Sample  $\ell$  prey  $y_1, \dots, y_\ell \sim \text{Unif}(W_t)$  i.i.d.;
14     Select parent index  $j := \arg \max_{r \in [k]} \min_{s \in [\ell]} g(x_r, y_s)$  where  $j \in [k]$ , breaking ties u.a.r.;
15     Create  $P_{t+1}(i) := \text{mut}_x(x_j)$ ;
16   for  $i \in [\lambda]$  do
17     Sample  $k$  prey  $y_1, \dots, y_k \sim \text{Unif}(Q_t)$  i.i.d.;
18     Sample  $\ell$  predators  $x_1, \dots, x_\ell \sim \text{Unif}(V_t)$  i.i.d.;
19     Select parent index  $j := \arg \min_{s \in [k]} \max_{r \in [\ell]} g(x_r, y_s)$  where  $j \in [\ell]$ , breaking ties u.a.r.;
20     Create  $Q_{t+1}(i) := \text{mut}_y(y_j)$ ;

```

Algorithmic Description ATS-CoEA uses two populations $P \in \mathcal{X}^\lambda$ and $Q \in \mathcal{Y}^\lambda$ which we sometimes will refer to as the “predators” and the “prey” and two archives $V \in \mathbb{P}(\mathcal{X})$ and $W \in \mathbb{P}(\mathcal{Y})$. Initially, we make minimal assumptions about the archives: the archives can only contain individuals seen by the algorithm and are updated at the start of each generation using the current archives and the current populations. In our analyses we assume that the initial archives are empty, but in practice they can be seeded with known solutions.

ATS-CoEA selects λ parents by using λ tournaments with k solutions sampled u.a.r. from the populations P or Q that compete against ℓ competitors sampled exclusively from the archives V or W . The best worst-case solution (a solution’s fitness is its worst performance) within the tournament is selected as the parent. Afterwards an offspring is created by mutating the parent and the offspring is added to the next population.

RankDivCoEA uses two populations $P \in \mathcal{X}^\lambda$ and $Q \in \mathcal{Y}^\lambda$ which we sometimes will refer to as the “predators” and the “prey”. A central component of RankDivCoEA is a diversity mechanism which is defined in the same way for the predators and the prey. We now describe how it works for the predators.

Each predator x imposes a total order $\leq_x$ on the set of prey $\mathcal{Y}$ given by $y_1 \leq_x y_2$ if and only if $g(x, y_1) \leq g(x, y_2)$. Clearly, different predators x lead to different orders/ranks. To achieve this, in each iteration the algorithm samples two predators x_1 and x_2 , and two prey y_1 and y_2 . First, the algorithm checks if $g(x_1, y_1) = g(x_1, y_2)$ and $g(x_2, y_1) = g(x_2, y_2)$. In such a case y_1 and y_2 appear to have the same phenotype, bypassing the diversity mechanism for x_1 and x_2 . Therefore, in this case a random bit is flipped of either y_1 or y_2 until $g(x_1, y_1) \neq g(x_1, y_2)$ or $g(x_2, y_1) \neq g(x_2, y_2)$ ².

Algorithm 2: Ranking Diversity CoEA (RankDivCoEA)

```

1 Require: Search spaces  $\mathcal{X}$  and  $\mathcal{Y}$ .
2 Require: Payoff function  $g : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$ .
3 Require: Population size  $\lambda \in \mathbb{N}$ .
4 Require:  $\text{mut}_x(\cdot) : \mathcal{X} \rightarrow \mathcal{X}$ ,  $\text{mut}_y(\cdot) : \mathcal{Y} \rightarrow \mathcal{Y}$ .
5 Require: Initial populations  $P_0 \in \mathcal{X}^\lambda$  and  $Q_0 \in \mathcal{Y}^\lambda$ .
6 for  $t \in \mathbb{N}_0$  until termination condition satisfied do
7   for  $i \in [\lambda]$  do
8     Sample  $x_1, x_2, x_3 \sim \text{Unif}(P_t)$  and  $y_1, y_2 \sim \text{Unif}(Q_t)$ ;
9     while  $g(x_1, y_1) = g(x_1, y_2) \wedge g(x_2, y_1) = g(x_2, y_2)$  do Flip one bit u. a. r. from either  $y_1$  or  $y_2$  ;
10     $s_1 \leftarrow \text{sign}(g(x_1, y_1) - g(x_1, y_2))$ ;
11     $s_2 \leftarrow \text{sign}(g(x_2, y_1) - g(x_2, y_2))$ ;
12     $s_3 \leftarrow \text{sign}(g(x_3, y_1) - g(x_3, y_2))$ ;
13     $x' \leftarrow \begin{cases} x_1 & \text{if } s_1 \neq s_2 \wedge s_1 \neq s_3 \\ x_1 & \text{if } s_1 = s_2 \wedge \min(g(x_1, y_1), g(x_1, y_2)) \geq \min(g(x_2, y_1), g(x_2, y_2)) \\ x_2 & \text{otherwise.} \end{cases}$ 
14    Create  $P_{t+1}(i) := \text{mut}_x(x')$ ;
15    Sample  $y_1, y_2, y_3 \sim \text{Unif}(Q_t)$  and  $x_1, x_2 \sim \text{Unif}(P_t)$ ;
16    while  $g(x_1, y_1) = g(x_2, y_1) \wedge g(x_1, y_2) = g(x_2, y_2)$  do Flip one bit u. a. r. from either  $x_1$  or  $x_2$  ;
17     $s_1 \leftarrow \text{sign}(g(x_1, y_1) - g(x_2, y_1))$ ;
18     $s_2 \leftarrow \text{sign}(g(x_1, y_2) - g(x_2, y_2))$ ;
19     $s_3 \leftarrow \text{sign}(g(x_1, y_3) - g(x_2, y_3))$ ;
20     $y' \leftarrow \begin{cases} y_1 & \text{if } s_1 \neq s_2 \wedge s_1 \neq s_3 \\ y_1 & \text{if } s_1 = s_2 \wedge \max(g(x_1, y_1), g(x_2, y_1)) \leq \max(g(x_1, y_2), g(x_2, y_2)) \\ y_2 & \text{otherwise.} \end{cases}$ 
21    Create  $Q_{t+1}(i) := \text{mut}_y(y')$ ;

```

Afterwards, if x_1 and x_2 order y_1 and y_2 differently (e.g. $y_1 \leq_{x_1} y_2$ and $y_1 \geq_{x_2} y_2$), then the algorithm samples a third predator x_3 uniformly at random. If x_2 and x_3 order y_1 and y_2 in the same way (e.g. $y_1 \geq_{x_2} y_2$ and $y_1 \geq_{x_3} y_2$), then (probabilistically) the predator x_1 orders the prey more uniquely than x_2 , and the algorithm selects x_1 . (See illustration in Figure 1). Otherwise, if x_1 and x_3 order y_1 and y_2 in the same way (e.g. $y_1 \leq_{x_1} y_2$ and $y_1 \leq_{x_3} y_2$), then the predator x_2 (probabilistically) orders the prey in a more unique way than x_1 and x_2 is selected. If both x_1 and x_2 order y_1 and y_2 identically (e.g. $y_1 \leq_{x_1} y_2$ and $y_1 \leq_{x_2} y_2$), then the algorithm selects the best worst-case solution, that is, the solution with the highest payoff value for its worst interaction.

Problem Definition Maximin optimisation is a decision-making approach that seeks to find a candidate solution that maximise the possible payoff, assuming that the adversary takes the least favourable action for that solution. Formally,

² This mechanism does not change the original solutions in the population, they are just changed locally for the diversity mechanism.

given a function $g : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$ representing the payoff, maximin optimisation involves finding a solution $x \in \mathcal{X}$ such that $\max_{x \in \mathcal{X}} \min_{y \in \mathcal{Y}} g(x, y)$ is achieved.

We consider the function $\text{BILINEAR}_{\alpha, \beta} : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \mathbb{R}$ [18] defined for arbitrary, not necessarily constant, $\alpha, \beta \in (0, 1)$ as $\text{BILINEAR}_{\alpha, \beta}(x, y) := \|y\|(\|x\| - \beta n) - \alpha n \|x\|$. The maximin-optima of $\text{BILINEAR}_{\alpha, \beta}$ are all pairs (x, y) with $\|x\| = \beta n$ and $\|y\| = \alpha n$.

We focus on $\text{BILINEAR}_{\alpha, \beta}$ because it has been shown to be a challenging class of problems to solve due to its intransitive behaviour (rock-paper-scissors-like interaction). Most studied algorithms do not have a clear signal towards/away the optimal solutions during most of the optimisation, leaving the algorithms susceptible to dynamics akin to random walks, cycling around the target solutions [12, 13].

During our analysis, we partition the search space into four *quadrants* (Figure 1). We say that a pair of search points (x, y) is in: the first quadrant if $\|x\| < \beta n \wedge \|y\| \geq \alpha n$, the second quadrant if $\|x\| \geq \beta n \wedge \|y\| > \alpha n$, the third quadrant if $\|x\| > \beta n \wedge \|y\| \leq \alpha n$, and the fourth quadrant if $\|x\| \leq \beta n \wedge \|y\| < \alpha n$.

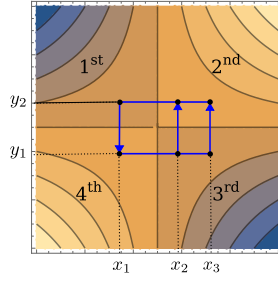


Fig. 1: Contour of $\text{BILINEAR}_{\alpha, \beta}$ ($\alpha = \beta = 1/2$) with the quadrants enumerated and illustration of diversity mechanism in RankDivCoEA, where the blue arrows indicate increasing g -values. Given a uniformly selected predator x_3 and uniformly selected prey y_1 and y_2 , predator x_1 is preferred over predator x_2 because it has a more unique ordering of y_1 and y_2 .

3.1 Level-Based Theorem

In this work we use the level-based theorem for coevolutionary processes from [18]. In addition we also use a slight modification shown in the appendix (Theorem 4). In this modified version, rather than assuming that the initial levels cover the whole search space, we allow for arbitrary initial levels, as long as it can be ensured that a sufficiently large proportion of the populations are initialised within these initial levels. Furthermore, the runtime is stated as a tail bound rather than an upper bound on the expected runtime.

We now divide the search in a suitable sequence $(A_j \times B_j)_{j \in m}$ of subsets of $\mathcal{X} \times \mathcal{Y}$ (levels) where $A_m \times B_m$ is the target set. Given our objective is to ensure

the general applicability of our results across a wide range of algorithms (i.e. Algorithm 1 with any appropriate archive and Algorithm 2), we give a general definition of a level that partitions the search space in three customisable regions for each population. Then, when analysing a particular algorithm, we can define a particular sequence of levels with this definition.

Following [18] and as illustrated in Figure 2 (a), for any $r_1 \leq r_2$ and any $s_2 \leq s_1$, we partition $\mathcal{X}$ and $\mathcal{Y}$ into three sets each,

$$\begin{aligned} R_0(r_1) &:= \{x \in \mathcal{X} \mid 0 \leq \|x\| < r_1\}, & S_0(s_1) &:= \{y \in \mathcal{Y} \mid s_1 < \|y\| \leq n\}, \\ R_1(r_1, r_2) &:= \{x \in \mathcal{X} \mid r_1 \leq \|x\| \leq r_2\}, & S_1(s_2, s_1) &:= \{y \in \mathcal{Y} \mid s_2 \leq \|y\| \leq s_1\}, \\ R_2(r_2) &:= \{x \in \mathcal{X} \mid r_2 < \|x\| \leq n\}, & S_2(s_2) &:= \{y \in \mathcal{Y} \mid 0 \leq \|y\| < s_2\}. \end{aligned}$$

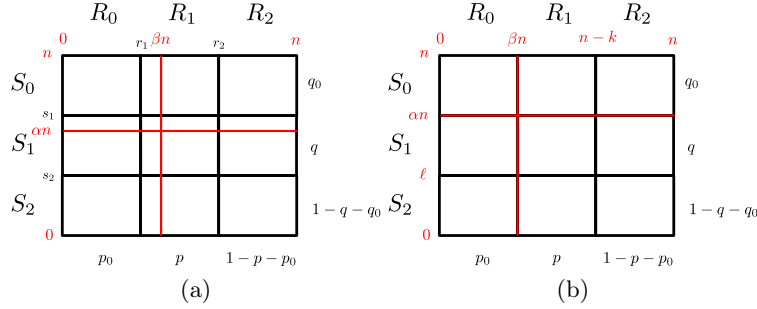


Fig. 2: Partitioning of search space $\mathcal{X} \times \mathcal{Y}$ of BILINEAR. (Adapted from [18])

For the sake of clarity, when the parameters r_1 , r_2 , s_1 and s_2 are clear from context, we will just denote these sets as R_0 , R_1 , R_2 , S_0 , S_1 and S_2 .

In Section 6, we redefine slightly the partition of the search space, as illustrated in Figure 2 (b). For any $k \in [0, (1 - \beta)n]$ and $\ell \in [0, \alpha n]$, we partition $\mathcal{X}$ and $\mathcal{Y}$ into three sets each,

$$\begin{aligned} R_0 &:= \{x \in \mathcal{X} \mid 0 \leq \|x\| < \beta n\}, & S_0 &:= \{y \in \mathcal{Y} \mid \alpha n \leq \|y\| \leq n\}, \\ R_1(k) &:= \{x \in \mathcal{X} \mid \beta n \leq \|x\| < n - k\}, & S_1(\ell) &:= \{y \in \mathcal{Y} \mid \ell \leq \|y\| < \alpha n\}, \\ R_2(k) &:= \{x \in \mathcal{X} \mid n - k \leq \|x\| \leq n\}, & S_2(\ell) &:= \{y \in \mathcal{Y} \mid 0 \leq \|y\| < \ell\}. \end{aligned}$$

For ease of notation, when the k and ℓ are clear from the context, we will simply refer to these sets as R_0 , R_1 , R_2 , S_0 , S_1 , and S_2 .

We now proceed to define the probability of sampling an individual inside each of these regions within the search space from the current populations.

$$\begin{aligned} p_0 &= \Pr_{x \sim \text{Unif}(P)} [x \in R_0], & q_0 &= \Pr_{y \sim \text{Unif}(Q)} [y \in S_0], \\ p &= \Pr_{x \sim \text{Unif}(P)} [x \in R_1], & q &= \Pr_{y \sim \text{Unif}(Q)} [y \in S_1]. \end{aligned}$$

Furthermore, for the random parents x' and y' selected in lines 14 and 19 of Algorithm 1 and lines 13 and 20 of Algorithm 2, for all $C \subseteq \mathcal{X} \times \mathcal{Y}$, let $p_{\text{sel}}(C) := \Pr((x', y') \in C)$. In addition, we define the probability of sampling an individual with more, less and exactly the same number of ones as the optimal solution from the current archives.

$$\begin{aligned} v_1 &= \Pr_{x \sim \text{Unif}(V)} [\|x\| < \beta n], & w_1 &= \Pr_{y \sim \text{Unif}(W)} [\|y\| < \alpha n], \\ v_2 &= \Pr_{x \sim \text{Unif}(V)} [\|x\| = \beta n], & w_2 &= \Pr_{y \sim \text{Unif}(W)} [\|y\| = \alpha n], \\ v_3 &= \Pr_{x \sim \text{Unif}(V)} [\|x\| > \beta n], & w_3 &= \Pr_{y \sim \text{Unif}(W)} [\|y\| > \alpha n]. \end{aligned}$$

4 What is a Good Evaluation Environment?

In this section we explore what are the characteristics that the evaluation environment (archive population) used by Algorithm 1 needs to guarantee conditions (G2a) and (G2b) in the level-based theorem for coevolution [18] when using a tournament of size $k = 2$ and a competition size $\ell = 2$ on $\text{BILINEAR}_{\alpha, \beta}$. In the context of $\text{BILINEAR}_{\alpha, \beta}$ (and likely many other problems), this is the most challenging scenario. This is because the algorithm must simultaneously select a *good* individual from the population to participate in the tournament and also a diverse set of competitors to ensure an accurate ranking, while sampling only two times from the population and two times from the archive. If the tournament size or the competition size is increased we conjecture that the restrictions on the archive would be lessened.

To begin this section we assume that a *good* archive would make the individuals in the current populations P and Q approach the maximin-optima. Using the partitions from Section 3.1 we can define the levels that we will use in this section. $A(r_1, r_2) := R_1(r_1, r_2)$, $B(s_2, s_1) := S_1(s_2, s_1)$. In this section, we always use $A_1 := A(0, n) = \mathcal{X}$ and $B_1 := B(0, n) = \mathcal{Y}$.

In Lemma 1 we show that for any level with $\beta n - r_1 = r_2 - \beta n$ and $\alpha n - s_2 = s_1 - \alpha n$ (Figure 5 (a)) the archive populations V and W need to have a certain ranking diversity. More specifically for archive V (W) there needs to be at least $1 - \frac{1}{\sqrt{2}} \approx 0.29$ fraction of individuals with more than βn (αn) 1-bits ranking $y \in Q$ ($x \in P$) with respect to the number of 0-bits, and $1 - \frac{1}{\sqrt{2}}$ fraction of individuals with less than βn (αn) 1-bits ranking $y \in Q$ ($x \in P$) with respect to the number of 1-bits.

We note that the values needed to meet the conditions of Lemma 1 for $r_{(0)}$ are relatively high. This is in part because the tournament size is small but also because the proof uses approximations and some pessimistic assumptions to ease the computations.

Lemma 1. *Let $r_1 \leq \beta n$, $\beta n \leq r_2$, $s_2 \leq \alpha n$, $\alpha n \leq s_1$ with $\beta n - r_1 = r_2 - \beta n$ and $\alpha n - s_2 = s_1 - \alpha n$. Let $\varepsilon > 0$ be a constant. Let x and y be the offspring created in Lines 15 and 20 of Algorithm 1 with $k = \ell = 2$, and $r_{(0)}$ be the*

probability of not flipping a bit during mutation. If $v_1, v_3, w_1, w_3 \geq 1 - \frac{1}{\sqrt{2}} + \varepsilon$ and there exist constants $\delta, \delta' > 0$ such that $\frac{1+\delta}{1+(1-\delta')(1+\sqrt{2})\varepsilon} \leq r_{(0)}^2 \leq 1$, then, for all $\gamma \in (0, \delta'/2]$ any population with $P \in \mathcal{X}^\lambda$ and $Q \in \mathcal{Y}^\lambda$ with $|(P \times Q) \cap (A(r_1, r_2) \times B(s_2, s_1))| \geq \gamma\lambda^2$ guarantees that

$$\Pr[x \in A(r_1, r_2)] \Pr[y \in B(s_2, s_1)] \geq (1 + \delta)\gamma.$$

It is clear that if the algorithm does not know beforehand that the optimisation problem is $\text{BILINEAR}_{\alpha, \beta}$, it is unlikely that its archive populations are initialised in such a way that the conditions of Lemma 1 are met. Nonetheless, with the insights of Lemma 1 we can design algorithms that create and maintain such diversity of rankings.

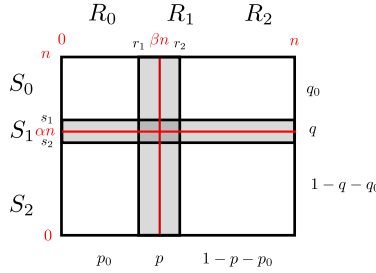


Fig. 3: Levels for Lemma 1 on $\text{BILINEAR}_{\alpha, \beta}$.

5 An Example Archive with Efficient Runtime on BILINEAR

In the previous section we learnt that if the competitors in the archives rank solutions in the current populations P and Q differently then the algorithm tends towards the maximin-optima.

Based on this notion we propose an archive update scheme (Algorithm 3) that we call Diverse Ranking Archives Update Scheme (DRAUS). This update scheme is used in line 10 of Algorithm 1. It uses the concept of archive domination (Definition 2) to add to the archive any solution in the current population that ranks solutions differently than other solutions already in the archive.

Definition 2 (Archive Domination). *Given a function $g: \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$, a population $Q \in \mathcal{Y}^\lambda$ and two archives $V_1, V_2 \subseteq \mathcal{X}$, we say that V_1 dominates V_2 with respect to Q and g denoted $V_1 \succeq_Q V_2$ if and only if:*

1. *for all $x_2 \in V_2$ and all $(y_1, y_2) \in Q \times Q$ where $g(x_2, y_1) > g(x_2, y_2)$ there exist $x_1 \in V_1$ st. $g(x_1, y_1) > g(x_1, y_2)$ and*

2. for all $x_2 \in V_2$ and all $(y_1, y_2) \in Q \times Q$ where $g(x_2, y_1) = g(x_2, y_2)$ there exist $x_1 \in V_1$ st. $g(x_1, y_1) = g(x_1, y_2)$.

Analogously, $W_1 \succeq_P W_2$ denotes that W_1 dominates W_2 with respect to P and g .

Algorithm 3: Diverse Ranking Archives Update Scheme (DRAUS)

```

1 Require: Search spaces  $\mathcal{X}$  and  $\mathcal{Y}$ .
2 Require: Payoff function  $g : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$ .
3 Require: Current archives  $V \subseteq \mathcal{X}$ ,  $W \subseteq \mathcal{Y}$ .
4 Require: Current populations  $P \in \mathcal{X}^\lambda$ ,  $Q \in \mathcal{Y}^\lambda$ 
5 for  $i \in [\lambda]$  do
6   if  $\neg(V \succeq_Q \{P(i)\})$  then  $V := V \cup \{P(i)\}$  ;
7   if  $\neg(W \succeq_P \{Q(i)\})$  then  $W := W \cup \{Q(i)\}$  ;
8 return  $V, W$ 
    
```

To check whether $V \succeq_Q \{x\}$ for a solution $x \in P$ takes at most $(|V|+1)|Q| = (|V|+1)\lambda$ evaluations. If the results of the evaluations between the archive V and the population Q are stored, then to check $V \succeq_Q \{x'\}$ for a new solution $x' \in P$ only requires λ extra evaluations. In total this results in at most $(|V|+\lambda)\lambda$ evaluations to update the archive V . The same argument can be applied to the archive W . This shows the bounds in Lemma 2.

Lemma 2. *Let $|V|$ and $|W|$ be the size of the archives. Then, Algorithm 3 uses at most $(|V|+\lambda)\lambda + (|W|+\lambda)\lambda$ evaluations to update the archive.*

On $\text{BILINEAR}_{\alpha,\beta}$ both the predator and prey individuals can only be ranked in three different ways each. Here we explain the three ranks that the predators induce in the prey. All predators x with $\|x\| > \beta n$ order all $y \in \mathcal{Y}$ by their number of 1-bits in descending order, that is, the higher the number of 1-bits in y the smaller their payoff when compared against x with $\|x\| > \beta n$. In turn all predators x with $\|x\| < \beta n$ order all $y \in \mathcal{Y}$ by their number of 1-bits in ascending order. Finally all predators x with $\|x\| = \beta n$ give the same payoff to all $y \in \mathcal{Y}$.

Therefore the maximum size of both the predator and prey archives using DRAUS (Algorithm 3) is three. Thanks to this, the maximum number of evaluations needed to update the archive on $\text{BILINEAR}_{\alpha,\beta}$ is in the worst case $O(\lambda^2)$. We remark that this might be different on other problems and it could grow exponentially on the problem size n . Therefore, we recommend to account for that if it is used on other problems.

Now, we show that ATS-CoEA (Algorithm 1) using DRAUS (Algorithm 3) builds a diverse archive and optimise $\text{BILINEAR}_{\alpha,\beta}$ efficiently for $\alpha, \beta \in (0, 1)$.

Theorem 1. *Let $\alpha, \beta \in (0, 1)$. Consider Algorithm 1 using Algorithm 3 as archive update scheme on $\text{BILINEAR}_{\alpha,\beta}$. Define $\text{OPT} := \{(x, y) \in (\mathcal{X} \times \mathcal{Y}) \mid \|x\| = \beta n \wedge \|y\| = \alpha n\}$ and $T := \min\{\lambda^2 t \mid P_t \times Q_t \cap \text{OPT} \neq \emptyset\}$. Then if there are constants $\delta, \delta' > 0$ such that the probability $r_{(0)}$ of the mutation operator is at least*

$$\max \left\{ \frac{8(1+\delta)}{14(1-\delta')}, \sqrt{\frac{6(1+\delta)}{6+(1-\delta')(2-\sqrt{2})}} \right\},$$

$r_{(1)} > 0$ is constant and for a sufficiently large constant c , $c \log n \leq \lambda \in \text{poly}(n)$ then, it holds that $\mathbb{E}[T] = O(\lambda^4 n)$.

The main idea of the proof is that if the algorithm does not have an individual for each possible ranking in both archives, the archives guide the search towards a region of the search space where that ranking exist. After finding all individuals that rank their opponents differently, then the archive meets the conditions for Lemma 1 and the algorithm has a clear signal towards the optimum.

6 Enforcing Diversity Without an Archive

In this section we analyse RankDivCoEA on $\text{BILINEAR}_{\alpha, \beta}$ with $\alpha n, \beta n \notin \mathbb{Z}$. This algorithm maintains populations that rank their opponents in distinct ways without the overhead of maintaining an archive.

6.1 Balancing of populations across βn and αn

We will show that the diversity mechanism will eventually ensure that the number of predators in R_0 converge to approximately $\lambda/2$ (i.e., half the predators). The reason is that predators in R_0 order prey differently than predators outside R_0 . If the predators in R_0 are in a minority, the diversity mechanism will prefer them over predators outside R_0 , and *vice versa*. With overwhelmingly high probability, not much more than half of the predators are R_0 (Lemma 3 with $A_0 = R_0$). Furthermore, if a constant, but less than half, fraction of the predators are in R_0 , then the number of such predators will increase with overwhelmingly high probability (Lemma 4). Due to symmetry of the problem and the algorithm, analogous statements can be proved about the prey population with respect to the region S_0 .

Lemma 3. *Consider Algorithm 2 with x_1, x_2, y_1, y_2 from line 8 and x' from line 13. Let $\varepsilon \in (0, 1/2)$ be any constant. Let $A \subset \mathcal{X}$ be any subset such that $\forall y_1, y_2 \in \mathcal{Y}, \forall x_1 \in A, \forall x_2 \in \mathcal{X} \setminus A, g(x_1, y_1) \geq g(x_1, y_2)$ if and only if $g(x_2, y_1) \leq g(x_2, y_2)$, and for all $x' \in \mathcal{X} \setminus A$, $\Pr_{x \sim \text{mut}_x(x')}(x \in A) \leq \varepsilon/6$. For all $t \geq 0$, if $|P_t \cap A| \leq \frac{\lambda}{2}(1 + \varepsilon)$, then $\Pr[|P_{t+1} \cap A| > \frac{\lambda}{2}(1 + \varepsilon)] = e^{-\Omega(\lambda)}$.*

Lemma 4. *Consider Algorithm 2 with x_1, x_2, y_1, y_2 from line 8 and x' from line 13. Let $\varepsilon \in (0, 1/2)$ be any constant. Let $A \subset \mathcal{X}$ be any subset such that $\forall y_1, y_2 \in \mathcal{Y}, \forall x_1 \in A, \forall x_2 \in \mathcal{X} \setminus A, g(x_1, y_1) \geq g(x_1, y_2)$ if and only if $g(x_2, y_1) \leq g(x_2, y_2)$, and for all $x' \in A$, $\Pr_{x \sim \text{mut}_x(x')}(x \in A) \geq 1 - \frac{\varepsilon}{4}$. For all $t \in \mathbb{N}$, define $X_t := |P_t \cap A|$. Then, for all $t \geq 0$, $\Pr[X_{t+1} \geq (1 + \frac{\varepsilon}{16}) \min\{X_t, \frac{\lambda}{2}(1 - \varepsilon)\} \mid X_t] \geq 1 - e^{-\Omega(X_t)}$.*

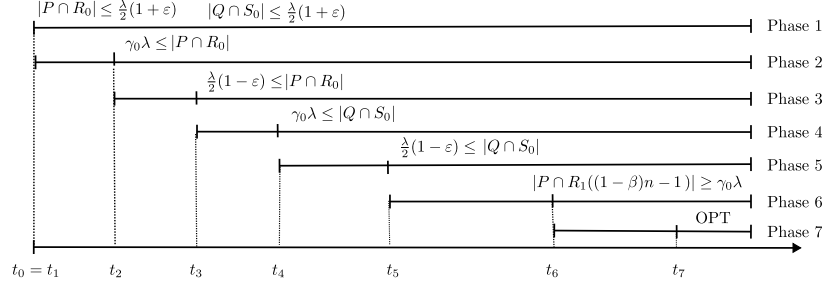


Fig. 4: Overview of an era.

6.2 Runtime Analysis Sketch

We will divide the run of the algorithm into *eras*, where each era consists of seven overlapping *phases* as illustrated in Figure 4. Phase $i \in [7]$ starts at deterministic generation number t_{i-1} and lasts until the last generation number t_7 . Each phase $i \in [7]$ has a “warm-up” sub-phase from generation t_{i-1} , lasting for $\tau_i := t_i - t_{i-1}$ generations. We let $\tau := \sum_{i=1}^7 \tau_i = t_7 - t_0$ denote the duration of one era. Each phase i is associated with a predicate $\mathcal{E}_i$ defined on the set of possible populations $\mathcal{X}^\lambda \times \mathcal{Y}^\lambda$. Phase i is called *successful* if for every generation $t \in [t_i, t_7]$, predicate $\mathcal{E}_i(P_t, Q_t)$ holds. The final event $\mathcal{E}_7(P, Q)$ corresponds to the populations P and Q containing optimal solutions. To compute the expected time until the algorithm finds the optimum, we compute the expected number of eras until all phases within an era are successful.

It will become apparent from the analysis that once the populations are approximately equally divided in the four quadrants of the search space, the populations will steadily evolve towards the optimum. The first phases capture the fact that the algorithm will eventually reach a configuration where the predators are almost evenly divided above and below βn , and the prey are almost evenly divided above and below αn . More precisely,

$$\frac{1-\varepsilon}{2} \leq \frac{|P \cap R_0|}{\lambda} \leq \frac{1+\varepsilon}{2} \quad \text{and} \quad \frac{1-\varepsilon}{2} \leq \frac{|Q \cap S_0|}{\lambda} \leq \frac{1+\varepsilon}{2}. \quad (1)$$

Unless the populations are already in this configuration in generation t_0 , there must exist one quadrant which contains the majority of the predators and prey. Without loss of generality, we will assume that the era starts in generation t_0 with a large fraction of the predator-prey pairs in the third quadrant. More precisely, we will make the following assumption:

Assumption 1. *The populations at the start of the era satisfy*

$$\frac{|P_{t_0} \cap R_0|}{\lambda} < \frac{1+\varepsilon}{2} \quad \text{and} \quad \frac{|Q_{t_0} \cap S_0|}{\lambda} < \frac{1+\varepsilon}{2}. \quad (2)$$

Clearly, other initial configurations are possible. However, due to symmetry of the problem and the algorithm, the analysis starting from other configurations

is the same. Given the assumption about the initial populations, the phases with corresponding predicates are given in Definition 3, and illustrated in Figure 4.

Definition 3. Let $\varepsilon, \gamma_0, \delta$ be some constants satisfying $\delta \in (0, 1/5)$, $0 < \gamma_0 \leq (1 + 3\varepsilon)/4$ and $\gamma_0 + \varepsilon \leq \frac{2}{3}(1 - \delta)$. Define the predicates

$$\mathcal{E}_1(P, Q) := |Q \cap S_0| \leq \frac{\lambda}{2}(1 + \varepsilon) \wedge |P \cap R_0| \leq \frac{\lambda}{2}(1 + \varepsilon) \quad (3)$$

$$\mathcal{E}_2(P, Q) := |P \cap R_0| \geq \gamma_0 \lambda \quad (4)$$

$$\mathcal{E}_3(P, Q) := |P \cap R_0| \geq \frac{\lambda}{2}(1 - \varepsilon) \quad (5)$$

$$\mathcal{E}_4(P, Q) := |Q \cap S_0| \geq \gamma_0 \lambda \quad (6)$$

$$\mathcal{E}_5(P, Q) := |Q \cap S_0| \geq \frac{\lambda}{2}(1 - \varepsilon) \quad (7)$$

$$\mathcal{E}_6(P, Q) := |P \cap R_1((1 - \beta)n - 1)| \geq \gamma_0 \lambda \quad (8)$$

$$\mathcal{E}_7(P, Q) := \exists x \in P, \exists y \in Q, \|x\| = \beta n \wedge \|y\| = \alpha n. \quad (9)$$

In addition, we assume the following for the mutation operator.

Assumption 2. The probabilities $r_{(1)}$ and $r_{(0)}$ of the mutation operator satisfy $r_{(1)} > 0$ is constant, $r_{(0)}^2 \geq 1 - \delta/2$ and $r_{(0)} \geq 1 - \varepsilon/6$, with δ and ε given in Definition 3, and the population size is $\lambda \geq c \log(n)$ for a sufficiently large constant c .

Following the proof sketch above we can show the main result of this section. We note that preliminary experiments show that the restriction $\alpha n, \beta n \notin \mathbb{Z}$ in Theorem 2 is not necessary, but it simplifies the already lengthy proofs.

Theorem 2. For all $\alpha n, \beta n \notin \mathbb{Z}$, Algorithm 2 with parameter satisfying Assumption 2 has expected runtime $O(n\lambda^3 - n\lambda \ln(\alpha\beta(1-\beta)(1-\alpha)))$ on $\text{BILINEAR}_{\alpha,\beta}$.

7 Conclusions

We have shown that for the intransitive problem class $\text{BILINEAR}_{\alpha,\beta}$ it is beneficial to maintain a high *diversity of rankings* in the evaluation environment, that is, a set of individuals used for evaluation which are diverse in how they rank opponents. We proposed two algorithms that ensure high diversity of rankings via archive populations (ATS-CoEA with DRAUS) and an intrinsic ranking diversity mechanism (RankDivCoEA). We rigorously showed that both approaches optimise $\text{BILINEAR}_{\alpha,\beta}$ in $O(\lambda^4 n)$ and $O(\lambda^3 n)$ expected evaluations respectively.

The $\text{BILINEAR}_{\alpha,\beta}$ problem class only has a small number of possible rankings for each population. We showed that the algorithms proposed here work well for this case, but it remains an open problem whether these mechanisms are effective on intransitive problems with a large number of possible rankings.

Acknowledgments. This research was supported by a Turing AI Fellowship (EPSRC grant ref EP/V025562/1).

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Angeline, P.J., Pollack, J.B.: Competitive environments evolve better solutions for complex tasks. In: *Proceedings of the International Conference on Genetic Algorithms*. p. 264–270. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1993)
2. Arcuri, A., Yao, X.: A novel co-evolutionary approach to automatic software bug fixing. In: *2008 IEEE Congress on Evolutionary Computation*. pp. 162–168 (2008)
3. Czarnecki, W.M., Gidel, G., Tracey, B., Tuyls, K., Omidshafiei, S., Balduzzi, D., Jaderberg, M.: Real world games look like spinning tops. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (eds.) *Advances in Neural Information Processing Systems*. vol. 33, pp. 17443–17454. Curran Associates, Inc. (2020)
4. De Jong, E.D.: The incremental pareto-coevolution archive. In: Deb, K. (ed.) *Proceedings of the Genetic and Evolutionary Computation Conference*. pp. 525–536. GECCO ’04, Springer Berlin Heidelberg, Berlin, Heidelberg (2004)
5. De Jong, E.D.: Towards a bounded pareto-coevolution archive. In: *Proceedings of the 2004 Congress on Evolutionary Computation*. vol. 2, pp. 2341–2348 (2004)
6. De Jong, E.D., Bucci, A.: DECA: dimension extracting coevolutionary algorithm. In: *Proceedings of the Genetic and Evolutionary Computation Conference*. p. 313–320. GECCO ’06, Association for Computing Machinery, New York, NY, USA (2006)
7. Doerr, B., Doerr, C., Yang, J.: Optimal parameter choices via precise black-box analysis. In: *Proceedings of the Genetic and Evolutionary Computation Conference*. p. 1123–1130. GECCO ’16, Association for Computing Machinery, New York, NY, USA (2016)
8. Ficici, S.G.: Solution concepts in coevolutionary algorithms. Ph.D. thesis, Department of Computer Science, Brandeis University, Waltham, MA (2004)
9. Ficici, S.G., Pollack, J.B.: A game-theoretic memory mechanism for coevolution. In: *Proceedings of the Genetic and Evolutionary Computation Conference*. p. 286–297. GECCO ’03, Springer-Verlag, Berlin, Heidelberg (2003)
10. Flores, D., Hemberg, E., Toutouh, J., O’Reilly, U.M.: Coevolutionary generative adversarial networks for medical image augmentation at scale. In: *Proceedings of the Genetic and Evolutionary Computation Conference*. p. 367–376. GECCO ’22, Association for Computing Machinery, New York, NY, USA (2022)
11. Hemberg, E., Toutouh, J., Al-Dujaili, A., Schmiedlechner, T., O’Reilly, U.M.: Spatial coevolution for generative adversarial network training. *ACM Trans. Evol. Learn. Optim.* **1**(2) (2021)
12. Hevia Fajardo, M.A., Lehre, P.K.: How fitness aggregation methods affect the performance of competitive CoEAs on bilinear problems. In: *Proceedings of the Genetic and Evolutionary Computation Conference*. p. 1593–1601. GECCO ’23, Association for Computing Machinery, New York, NY, USA (2023)
13. Hevia Fajardo, M.A., Lehre, P.K., Lin, S.: Runtime analysis of a co-evolutionary algorithm: Overcoming negative drift in maximin-optimisation. In: *Proceedings of the ACM/SIGEVO Conference on Foundations of Genetic Algorithms*. p. 73–83. FOGA ’23, Association for Computing Machinery, New York, NY, USA (2023)

14. Hillis, W.: Co-evolving parasites improve simulated evolution as an optimization procedure. *Physica D: Nonlinear Phenomena* **42**(1-3), 228–234 (1990)
15. Jansen, T., Wiegand, R.P.: The Cooperative Coevolutionary (1+1) EA. *Evolutionary Computation* **12**(4), 405–434 (2004)
16. Jaśkowski, W., Krawiec, K.: Coordinate system archive for coevolution. In: *IEEE Congress on Evolutionary Computation*. pp. 1–10 (2010)
17. Krawiec, K., Heywood, M.: Solving complex problems with coevolutionary algorithms. In: *Proceedings of the Genetic and Evolutionary Computation Conference*. p. 832–858. *GECCO '20*, Association for Computing Machinery, New York, NY, USA (2020)
18. Lehre, P.K.: Runtime analysis of competitive co-evolutionary algorithms for maximin optimisation of a bilinear function. In: *Proceedings of the Genetic and Evolutionary Computation Conference*. p. 1408–1416. *GECCO '22*, Association for Computing Machinery, New York, NY, USA (2022)
19. Lehre, P.K., Lin, S.: Is CC-(1+1) EA more efficient than (1+1) EA on separable and inseparable problems? In: *2023 IEEE Congress on Evolutionary Computation (CEC)*. pp. 1–9 (2023)
20. Lehre, P.K., Witt, C.: Black-box search by unbiased variation. *Algorithmica* **64**(4), 623–642 (2012)
21. Luke, S., Wiegand, R.P.: When coevolutionary algorithms exhibit evolutionary dynamics. In: *Genetic and Evolutionary Computation Conference Workshop Program*. pp. 236–241 (2002)
22. Miconi, T.: Why coevolution doesn't "work": Superiority and progress in coevolution. In: Vanneschi, L., Gustafson, S., Moraglio, A., De Falco, I., Ebner, M. (eds.) *Genetic Programming*. pp. 49–60. Springer Berlin Heidelberg, Berlin, Heidelberg (2009)
23. Motwani, R., Raghavan, P.: *Randomized Algorithms*. Cambridge University Press (1995)
24. Nolfi, S., Floreano, D.: Coevolving predator and prey robots: Do "Arms Races" arise in artificial evolution? *Artificial Life* **4**(4), 311–335 (1998)
25. Oliehhoek, F.A., De Jong, E.D., Vlassis, N.: The parallel Nash Memory for asymmetric games. In: *Proceedings of the Genetic and Evolutionary Computation Conference*. p. 337–344. *GECCO '06*, Association for Computing Machinery, New York, NY, USA (2006)
26. Popovici, E., Bucci, A., Wiegand, R.P., De Jong, E.D.: Coevolutionary principles. In: Rozenberg, G., Bäck, T., Kok, J.N. (eds.) *Handbook of Natural Computing*, pp. 987–1033. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
27. Rosin, C.D., Belew, R.K.: Methods for competitive co-evolution: Finding opponents worth beating. In: *Proceedings of the International Conference on Genetic Algorithms*. pp. 373–381. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1995)
28. Rosin, C.D., Belew, R.K.: New methods for competitive coevolution. *Evolutionary Computation* **5**(1), 1–29 (1997)
29. Sims, K.: Evolving 3D morphology and behavior by competition. *Artificial Life* **1**(4), 353–372 (1994)
30. Watson, R.A., Pollack, J.B.: Coevolutionary dynamics in a minimal substrate. In: *Proceedings of the Genetic and Evolutionary Computation Conference*. p. 702–709. *GECCO '01*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2001)
31. Yang, L., Huang, H., Yang, X.: An efficient pareto-coevolution archive. In: *Proceedings of the Third International Conference on Natural Computation - Volume 04*. p. 484–488. *ICNC '07*, IEEE Computer Society, USA (2007)

32. Yang, L., Huang, H., Yang, X.: A simple coevolution archive based on bidirectional dimension extraction. In: Proceedings of the 2009 International Conference on Artificial Intelligence and Computational Intelligence - Volume 01. p. 596–600. AICI '09, IEEE Computer Society, USA (2009)