How Fitness Aggregation Methods Affect the Performance of Competitive CoEAs on Bilinear Problems

Mario Alejandro Hevia Fajardo
School of Computer Science
University of Birmingham, Birmingham, UK

Per Kristian Lehre
School of Computer Science
University of Birmingham, Birmingham, UK

ABSTRACT

Competitive co-evolutionary algorithms (CoEAs) do not rely solely on an external function to assign fitness values to sampled solutions. Instead, they use the aggregation of outcomes from interactions between competing solutions allowing to rank solutions and make selection decisions. This makes CoEAs a useful tool for optimisation problems that have intrinsically interactive domains.

Over the past decades, many ways to aggregate the outcomes of interactions have been considered. At the moment, it is unclear which of these is the best choice. Previous research is fragmented and most of the fitness aggregation methods (fitness measures) proposed have only been studied empirically.

We argue that a proper understanding of the dynamics of CoEAs and their fitness measures can only be achieved through rigorous analysis of their behaviour. In this work we make a step towards this goal by using runtime analysis to study two commonly used fitness measures. We show a dichotomy in the behaviour of a $(1, \lambda)$ CoEA when optimising a Bilinear problem. The algorithm finds a Nash equilibrium efficiently if the worst interaction is used as a fitness measure but it takes exponential time w. o. p. if the average of all interactions is used instead.

CCS CONCEPTS

• **Theory of computation → Theory of randomized search heuristics**.

KEYWORDS

Runtime analysis, Competitive coevolution, Maximin Optimisation

ACM Reference Format:
Mario Alejandro Hevia Fajardo and Per Kristian Lehre. 2023. How Fitness Aggregation Methods Affect the Performance of Competitive CoEAs on Bilinear Problems. In *Genetic and Evolutionary Computation Conference (GECCO '23), July 15–19, 2023, Lisbon, Portugal.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3583131.3590506

1 INTRODUCTION

In biology the term co-evolution is used to describe a process where two or more species interact with each other reciprocally affecting their success and survival, and consequently their evolution. Co-evolution may happen in a mutualistic relationship, where both species profit from the interaction or a competitive relationship where the species are adversaries (e. g. predator/prey, parasite/host).

Co-evolutionary algorithms (CoEAs) try to mimic biological co-evolution to solve optimisation problems [15]. As in their biological counterparts, they are divided in Cooperative CoEAs [12] and Competitive CoEAs depending on the kind of interactions the solutions take part in. We focus on Competitive CoEAs (we omit the *competitive* label and simply call them CoEAs).

CoEAs have found great success in several applications, ranging from test problems [1, 7] to real-world applications [3, 5, 17]. Despite their successes it has been well documented that CoEAs present pathological algorithm dynamics during the optimisation process [11, 15]. These pathological behaviours include, *evolutionary forgetting*: individuals with good characteristics are lost due to lack of selection pressure from the current opponent population, *cyclic dynamics*: traveling through some part of the search space more than once with apparent improvement, *disengagement*: one population overwhelmingly outperforms the other and the fitness gradient disappears, among others. Although these pathologies are well-known, due to the complex population dynamics of CoEAs, they have eluded our understanding. Albeit some remedies have been proposed [4, 5, 16, 17] we argue that without a rigorous understanding of what causes these harmful algorithm dynamics any attempt to solve them has a high likelihood to result futile.

CoEAs use populations of solutions that in each generation interact with each other and later assign a fitness to each solution based on the results (payoffs) of their interactions. A common practice is to aggregate the payoffs into a single numerical fitness value [15], e. g. by averaging the performance outcomes, use the maximum, minimum, median [18][1] or more complicated statistics such as fitness sharing [16]. Unfortunately each fitness aggregation method may come with (sometimes harmful) biases depending on the optimisation problem being solved. For example, in cooperative CoEAs averaging interactions was observed to be harmful [19]. Bucci [2] even attributed some pathological behaviours encountered in CoEAs to the use of **any** aggregation method. At the moment, it is unclear when or if certain fitness aggregation methods (also called fitness measures) help the optimisation process or when they result in poor performance. Previous research is fragmented and all the fitness measures proposed have only been studied empirically[2].

Due to their complexity, there is little rigorous understanding of the algorithm dynamics in (cooperative and competitive) CoEAs. Jansen and Wiegand [6] 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 [8] analysed a population based CoEA on some instances of

[1]studied in Cooperative CoEAs
[2]Other than small tailored examples showing fitness measures can be misleading [2]

GECCO '23, July 15–19, 2023, Lisbon, Portugal
© 2023 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0119-1/23/07.
https://doi.org/10.1145/3583131.3590506

Bilinear, showing that given the correct parameters the algorithm finds an ε-approximation efficiently, but an incorrect parameter setting leads to exponential runtime. This work is a step forward towards a rigorous understanding of how different fitness measures can alleviate or aggravate the pathological behaviours of CoEAs.

We consider the $(1, \lambda)$ CoEA (Algorithm 1) on a class of Bilinear problems with an infinite discrete search space (c. f. Section 2.2). Bilinear is a challenging class of Maximin-problems because it has intransitive properties and the continuous versions are known to result in cyclic behaviour for some gradient-based algorithms [10, 13, 20]. Here, we ask whether the gradient-free $(1, \lambda)$ CoEA is able to find a suitable solution despite the intransitivity of the problem and what is the role of the fitness measures in its performance.

In Sections 3 and 4 we characterise how the fitness measures assign fitness values based on the current population and how the mutation operator creates solutions as a stepping stone of the following theoretical analysis. Afterwards, in Section 5 we show that the $(1, \lambda)$ CoEA using the average of interactions as fitness measure not only results in a cyclic behaviour, but every cycle the algorithm moves away from the optimum in expectation. This result in an exponential time (with respect to the initial distance to the optimum) to find an optimal solution w. o. p..

In sharp contrast, in Section 6 we show that the $(1, \lambda)$ CoEA (creating offspring in a deterministic way) using the worst interaction as fitness measure is efficient, finding the optimum in polynomial time. We note that the algorithm still presents a cycling behaviour but the fitness measure helps alleviate the problem.

Finally, we present an experimental analysis where we test whether our theoretical analysis translates to other search domains. We observe that our theoretical analysis translates to other search domains. That is, for all search domains studied, using the average as a fitness measure results in exponential runtimes and using the minimum as a fitness measure results in efficient runtimes.

2 PRELIMINARIES

For any natural number $n \in \mathbb{N}$ we define $[n] := \{1, 2, \dots, n\}$.

We study how different fitness aggregation methods (fitness measures) affect the performance of CoEAs on Bilinear problems. In particular, we study the expected number of fitness evaluations (interactions) of the $(1, \lambda)$ CoEA (Algorithm 1) with two different fitness measures on the class of Maximin-optimisation problems called Bilinear. Given a Maximin-objective function $g : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ (where $\mathcal{X}, \mathcal{Y}$ are any *domain*, *strategy space* or *search space*), a Maximin-optimisation problem is the problem of finding a solution $x \in \mathcal{X}$ that maximises g, assuming that the adversary $y \in \mathcal{Y}$ minimises g for that solution.

The $(1, \lambda)$ CoEA as defined in Algorithm 1 optimises a Maximin-optimisation problem. The $(1, \lambda)$ CoEA uses any mutation operators $\text{mut}_{\mathcal{X}}\{\cdot\} : \mathcal{X} \to \mathcal{X}$ and $\text{mut}_{\mathcal{Y}}\{\cdot\} : \mathcal{Y} \to \mathcal{Y}$. In concordance to the algorithm, the class of Bilinear problems can be defined in different search spaces too. We explore further this problem class in Section 2.2. For our theoretical analysis we consider $\mathcal{X} = \mathcal{Y} = \mathbb{Z}$ and the mutation operator used is described in detail in Section 4.

We choose the search spaces $\mathcal{X} = \mathcal{Y} = \mathbb{Z}$ for several reasons. The domain is discrete, therefore, gradient-based algorithms do not work denoting the importance of gradient-free algorithms such as CoEAs. Despite being a discrete domain, $\mathbb{Z}$ is infinite, which allows the creation of *real* unbiased mutation operators, in the sense that movement towards one direction of the search space is not limited by the search space itself making it less likely to happen. This reason is particularly important in this study, because we want to focus our analysis on how the fitness measures affect the performance of CoEAs and this allows us to decouple the behaviour of the selection mechanism (based on the fitness measures) from the inherent tendency away from the boundaries of the search space that a mutation operator in a finite search space have.

Due to space constraints, we removed some proofs from the paper; the detailed proofs can be found in the supplementary material.

2.1 The $(1, \lambda)$ CoEA

The $(1, \lambda)$ CoEA uses a parent population of size one for each search space $\mathcal{X}, \mathcal{Y}$. Each generation it creates 2λ offspring by mutating each parent λ times. Later it uses a fitness measure to assign a fitness value to every offspring in one population depending on their interactions with the other offspring population.

Algorithm 1: $(1, \lambda)$ CoEA

```
1 Require: Maximin-objective function $g : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$.
2 Require: Population size $\lambda \in \mathbb{N}$, fitness measures
          $f_{\mathcal{X}} : \mathcal{X} \times \mathcal{Y}^{\lambda}$, $f_{\mathcal{Y}} : \mathcal{Y} \times \mathcal{X}^{\lambda}$, mutation operators
          $\text{mut}_{\mathcal{X}}\{\cdot\} : \mathcal{X} \to \mathcal{X}$, $\text{mut}_{\mathcal{Y}}\{\cdot\} : \mathcal{Y} \to \mathcal{Y}$ and
          initialisation methods $\text{init}_{\mathcal{X}}\{\mathcal{X}\}$, $\text{init}_{\mathcal{Y}}\{\mathcal{Y}\}$
3 Initialization: $x_1 \leftarrow \text{init}_{\mathcal{X}}\{\mathcal{X}\}$ and $y_1 \leftarrow \text{init}_{\mathcal{Y}}\{\mathcal{Y}\}$.
4 for $t \in \mathbb{N}$ do
5     Mutation: for $i \in \{1, \dots, \lambda\}$ do
6         $P_t(i) \leftarrow \text{mut}\{x_t\}$ and $Q_t(i) \leftarrow \text{mut}\{y_t\}$;
7     Selection:
8     Choose $x_{t+1} \leftarrow \arg\max_{P_t(i)}\{f_{\mathcal{X}}(P_t(i), Q_t)\}$;
9     Choose $y_{t+1} \leftarrow \arg\max_{Q_t(i)}\{f_{\mathcal{Y}}(Q_t(i), P_t)\}$;
```

We consider two fitness measures f:

- Individual vs all (average) f^{avg}: Each individual in a population is evaluated against every other individual in its competing population, and the average of the evaluations is used to determine the individuals' fitness. That is,

$$f^{\text{avg}}_{\mathcal{X}}(P_t(i), Q_t) = \frac{1}{\lambda}\sum_{j=1}^{\lambda} g(P_t(i), Q_t(j))$$

$$f^{\text{avg}}_{\mathcal{Y}}(Q_t(i), P_t) = -\frac{1}{\lambda}\sum_{j=1}^{\lambda} g(P_t(j), Q_t(i)).$$

- Individual vs all (worst) f^{wrs}: Each individual in a population is evaluated against every other individual in its competing population, and the worst of the evaluations is used to determine the individuals' fitness. That is,

$$f^{\text{wrs}}_{\mathcal{X}}(P_t(i), Q_t) = \min_{j \in [\lambda]}\{g(P_t(i), Q_t(j))\}$$

$$f^{\text{wrs}}_{\mathcal{Y}}(Q_t(i), P_t) = -\max_{j \in [\lambda]}\{g(P_t(j), Q_t(i))\}.$$

2.2 Bilinear

The class of Bilinear problems is a simple and well-defined class of Maximin-optimisation problems. The Bilinear problems have been extensively used to understand the behaviour of Maximin-optimisation algorithms (e.g. [10, 13, 20]) and it was recently used by Lehre [8] to analyse a population-based CoEA.

The general form of Bilinear considers an n-dimensional (continuous or discrete) domain for the solutions x, y. Since we consider the 1-dimensional search spaces $\mathcal{X} = \mathcal{Y} = \mathbb{Z}$, we give a simpler form.

$$\textsc{Bilinear}_{\alpha,\beta}(x,y) := xy - \alpha x - \beta y$$

The parameters α and β denote where the Maximin-solutions (also called Nash equilibria or optimal solutions) are found. We denote these solutions as OPT, that is, $\text{OPT} := \{(x,y) \mid x = \beta \wedge y = \alpha\}$.

During our analysis, we divide the search space into four *quadrants*. We say that a pair of search points (x, y) is in:

- the first quadrant if $x < \beta \wedge y \geq \alpha$,
- the second quadrant if $x \geq \beta \wedge y > \alpha$,
- the third quadrant if $x > \beta \wedge y \leq \alpha$, and
- the fourth quadrant if $x < \beta \wedge y < \alpha$.

2.3 Notation

By $\mathcal{G}(p)$ we denote the geometric distribution with parameter $p \in [0, 1]$. The geometric distribution is often defined in two different ways, we define it as the probability distribution of the number of failures before the first success of Bernoulli trials with success probability p. Then, for $X \sim \mathcal{G}(p)$ we have

$$\Pr[X = k] = (1-p)^k p, \quad \mathrm{E}[X] = \frac{1-p}{p}.$$

Definition 2.1. For populations $P_t \in \mathcal{X}$ and $Q_t \in \mathcal{Y}$ we define:

$$\begin{aligned}
X_t &:= \sum_{x \in P_t} (x - \beta) & \overline{X}_t &:= \sum_{x \in P_t} |x - \beta| \\
Y_t &:= \sum_{y \in Q_t} (y - \alpha) & \overline{Y}_t &:= \sum_{y \in Q_t} |y - \alpha| \\
x_{\min} &:= \arg\min_{x \in P_t}\{x\} & x_{\max} &:= \arg\max_{x \in P_t}\{x\} \\
y_{\min} &:= \arg\min_{y \in Q_t}\{y\} & y_{\max} &:= \arg\max_{y \in Q_t}\{y\} \\
P_t^- &:= \{x \mid x \in P_t \wedge x \leq \beta\} & P_t^+ &:= \{x \mid x \in P_t \wedge x \geq \beta\} \\
Q_t^- &:= \{y \mid y \in Q_t \wedge y \leq \alpha\} & Q_t^+ &:= \{y \mid y \in Q_t \wedge y \geq \alpha\} \\
x_{\min}^+ &:= \arg\min_{x \in P_t^+}\{x\} & x_{\max}^- &:= \arg\max_{x \in P_t^+}\{x\} \\
y_{\min}^+ &:= \arg\min_{y \in Q_t^+}\{y\} & y_{\max}^- &:= \arg\max_{y \in Q_t^+}\{y\} \\
x_{\min}^* &:= \arg\min_{x \in P_t}\{|x - \beta|\} & y_{\min}^* &:= \arg\min_{y \in Q_t}\{|y - \alpha|\}
\end{aligned}$$

3 ANALYSIS OF THE FITNESS MEASURES ON BILINEAR

In this section we focus our attention on how the fitness measures assign fitness to individual solutions. In particular, we explore what solutions have the *best* fitness with respect to each fitness measure on Bilinear. We start with the fitness measure f^{avg}.

Lemma 3.1 (Fitness Based on Averaging Interactions). *Consider two populations of solutions P_t and Q_t and the fitness measure f^{avg} on Bilinear. Then f^{avg} assigns the highest fitness to all solutions in the sets $\xi \subseteq P_t, \Upsilon \subseteq Q_t$ defined as,*

$$\xi = \begin{cases} \{x \mid x \in P_t \wedge x = x_{\min}\} & \text{if } Y_t < 0, \\ \{x \mid x \in P_t \wedge x = x_{\max}\} & \text{if } Y_t > 0, \\ P_t & \text{otherwise,} \end{cases}$$

and

$$\Upsilon = \begin{cases} \{y \mid y \in Q_t \wedge y = y_{\max}\} & \text{if } X_t < 0, \\ \{y \mid y \in Q_t \wedge y = y_{\min}\} & \text{if } X_t > 0, \\ Q_t & \text{otherwise.} \end{cases}$$

The most important thing to note from Lemma 3.1 is that the decisions are made independent of the distance of the solutions to their respective optimum. Additionally, the highest fitness is assigned to solutions that at first glance seem to be detrimental to be selected. In fact, our analyses focus on the decisions made by the $(1, \lambda)$ CoEA based on these *ill-informed* fitness values.

Lemma 3.2 describes the fitness measure f^{wrs}. The behaviour of this fitness measure is noticeably more complex than f^{avg}. Due to the number of different cases we only display the cases for x in Lemma 3.2. The cases for y are the same, exchanging x, P_t and α for y, Q_t and β and vice versa. The full lemma and its proof can be found in the supplementary material.

Lemma 3.2 (Fitness Based on Worst Interactions). *Consider two populations of solutions P_t and Q_t and the fitness measure f^{wrs} on Bilinear. Then f^{wrs} assigns the highest fitness to all solutions in the set $\xi \subseteq P_t$ defined as,*

$$\xi = \begin{cases}
\{x \mid x \in P_t \wedge x = x_{\max}^-\} & \text{if } |Y_t| \neq \overline{Y}_t \wedge |X_t| \neq \overline{X}_t \wedge \\ & (\beta - x_{\max}^-)(\alpha - y_{\max}) \geq \\ & \quad (\beta - x_{\min}^+)(\alpha - y_{\min}), \quad (1) \\
\{x \mid x \in P_t \wedge x = x_{\min}^+\} & \text{if } |Y_t| \neq \overline{Y}_t \wedge |X_t| \neq \overline{X}_t \wedge \\ & (\beta - x_{\max}^-)(\alpha - y_{\max}) \leq \\ & \quad (\beta - x_{\min}^+)(\alpha - y_{\min}), \quad (2) \\
\{x \mid x \in P_t \wedge x = x_{\min}^*\} & \text{if } |Y_t| \neq \overline{Y}_t \wedge |X_t| = \overline{X}_t, \quad (3) \\
P_t^+ & \text{if } |Y_t| = \overline{Y}_t \wedge Y_t > 0 \wedge \\ & \exists y \in Q_t, y = \alpha \wedge P_t^+ \neq \emptyset, \quad (4) \\
\{x \mid x \in P_t \wedge x = x_{\min}^*\} & \text{if } |Y_t| = \overline{Y}_t \wedge Y_t > 0 \wedge \\ & \exists y \in Q_t, y = \alpha \wedge P_t^+ = \emptyset, \quad (5) \\
P_t^- & \text{if } |Y_t| = \overline{Y}_t \wedge Y_t < 0 \wedge \\ & \exists y \in Q_t, y = \alpha \wedge P_t^- \neq \emptyset, \quad (6) \\
\{x \mid x \in P_t \wedge x = x_{\min}^*\} & \text{if } |Y_t| = \overline{Y}_t \wedge Y_t < 0 \wedge \\ & \exists y \in Q_t, y = \alpha \wedge P_t^- = \emptyset, \quad (7) \\
\{x \mid x \in P_t \wedge x = x_{\max}\} & \text{if } |Y_t| = \overline{Y}_t \wedge Y_t > 0 \wedge \\ & \forall y \in Q_t, y \neq \alpha, \quad (8) \\
\{x \mid x \in P_t \wedge x = x_{\min}\} & \text{if } |Y_t| = \overline{Y}_t \wedge Y_t < 0 \wedge \\ & \forall y \in Q_t, y \neq \alpha, \quad (9) \\
P_t & \text{if } y = \alpha \ \forall\, y \in Q_t, \quad (10)
\end{cases}$$

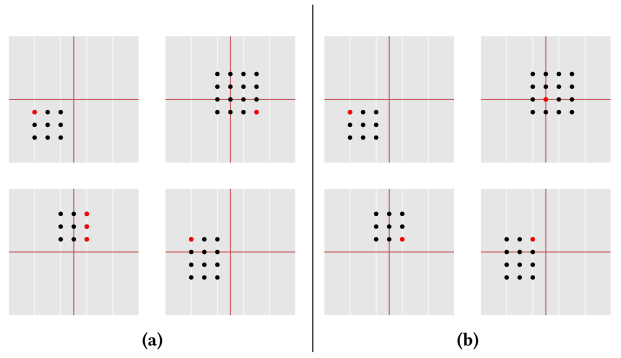

Figure 1: Example populations $P \times Q$ with the pair(s) of individuals with highest fitness value highlighted (red point) for fitness measures f^{avg} (a) and f^{wrs} (b). α β are denoted by the red lines and the optimum is at the crossing of these lines.

Figure 1 shows some example populations and their highest individuals for fitness measures f^{avg} (a) and f^{wrs} (b). The most interesting example is when $P \times Q$ comprises the four quadrants and the optimal solution is included (top right in 1 (a) and 1 (b)). In this case f^{wrs} correctly assigns the highest fitness to the optimum but f^{avg} does not. When $P \times Q$ comprises two quadrants (bottom left and right in 1 (a) and 1 (b)) f^{avg} tend to assign the highest fitness to individuals farther away from the optimum than the ones chosen by f^{wrs}. Finally if $P\times Q$ comprises one quadrant (top left in 1 (a) and 1 (b)) both fitness measures behave identically: the individual with highest fitness is always the corner *ahead* in a clockwise direction.

4 ANALYSIS OF THE MUTATION OPERATOR

The $(1, \lambda)$ CoEA analysed here uses the following simple mutation operator, which is called geometric mutation. For a parent x it creates the i-th offspring x' by sampling $X_i \sim \mathcal{G}(p)$ and adding or subtracting that value to the parent. To simplify the analysis the mutation operator creates exactly $\lambda/2$ offspring as $x' = x + X_i$ and the other $\lambda/2$ offspring as $x' = x - X_i$. This could be randomised, maintaining the same expected number of offspring increasing and decreasing value from the parent as follows: with probability $1/2$ the operator adds X_i and subtracts X_i otherwise. Although we believe our analysis holds for the randomised case, we decided not to add it to avoid over-complicating the already complex analysis.

In Section 3 we have seen that the fitness measures assign fitness to an individual based on the sum of distances to α or β. To this effect the following lemma is useful to understand how much this sum can change.

Lemma 4.1. *Let $h \in \mathbb{N}$ and Y_i be i.i.d. random variables sampled from $\mathcal{G}(p)$. Then,*

$$\Pr\left[\sum_{i=1}^{\psi}(Y_i + h) - \sum_{i=\psi+1}^{2\psi}(Y_i - h) \le 0\right] = e^{-\Omega(h\psi)}.$$

Another important thing that we learnt in Section 3 is that the offspring with highest fitness is often the farthest away from the parent (with respect to one of the directions). Therefore, we need to characterise the behaviour of the maximum value of several geometric random variables. In the following lemma we study its expected value.

Lemma 4.2. *Let $0 < p < 1$, $\psi \in \mathbb{N}$, $X_i \sim \mathcal{G}(p)$ and $X^{(\psi)} := \max_{i\in[\psi]}\{X_i\}$ then*

$$\left(1 - e^{-\frac{1}{1-p}}\right)\log_{\frac{1}{1-p}}\psi \le \mathrm{E}\left[X^{(\psi)}\right] \le \log_{\frac{1}{1-p}}\psi + \frac{1-p}{p} \tag{11}$$

The expected value is not sufficient to fully understand the behaviour of the algorithm. In the following we show tail bounds on the maximum value of several geometric random variables.

Lemma 4.3. *Let $0 < p < 1$ and $c > 0$. Let $\delta > 0$, $\psi \in \mathbb{N}$, $X_i \sim \mathcal{G}(p)$ and $X^{(\psi)} := \max_{i\in[\psi]}\{X_i\}$ then*

$$\Pr\left[X^{(\psi)} \ge (1+\delta)\frac{\ln\psi}{p}\right] \le \min\{\psi^{-\delta}, 1\} \tag{12}$$

$$\Pr\left[X^{(\psi)} \ge \log_{\frac{1}{1-p}} c\psi\right] \le 1/c \tag{13}$$

$$\Pr\left[X^{(\psi)} < (1-\delta)\frac{p\ln\psi}{1-p}\right] \le \exp\left(-\psi^{\delta}\right) \tag{14}$$

Algorithm 1 uses two competing populations and while one population moves towards the optimum the other can move away from it. In these cases we need to understand what is the expected movement overall and what is the probability of this deviating from its expectation. The following lemmas analyse these cases.

Lemma 4.4. *Let $0 < p < 1$. Let $\psi \in \mathbb{N}$, $\tau \in \mathbb{N}$, X_i, Y_i be independent and identically distributed (i.i.d.) random variables sampled from $\mathcal{G}(p)$, $X_j^{(\psi)} := \max_{i\in[\psi]}\{X_i\}$ and $Y_j^{(\psi)} := \max_{i\in[\psi]}\{Y_i\}$. Let $Z := \sum_{j=1}^{\tau}\left(X_j^{(\psi)} - Y_j^{(\psi)}\right)$. Then $\mathrm{E}[Z] = 0$.*

Lemma 4.5. *Let $0 < \varepsilon < p < 1$. Let $|\eta| < p - \varepsilon$, $\psi \in \mathbb{N}$, $\tau \in \mathbb{N}$, $X_i \sim \mathcal{G}(p)$ and $X^{(\psi)} := \max_{i\in[\psi]}\{X_i\}$. Then $\mathrm{E}\left[e^{\eta X^{(\psi)}}\right] \le \frac{\psi}{\varepsilon}$ and $\mathrm{E}\left[e^{-\eta X^{(\psi)}}\right] \le \psi\left(1 + \frac{1}{\varepsilon}\right)$.*

Lemma 4.6. *Let $0 < p < 1$. Let $\psi^6 \ge \left(1 + \frac{2}{p}\right)^2$ with $\psi \in \mathbb{N}$. Let $r' := \frac{8r\sqrt{2\ln\psi}}{p}$, $r \ge 2$, $j \in \mathbb{N}_0$, $\tau \in \mathbb{N}$, $X_i, Y_i \sim \mathcal{G}(p)$ be i.i.d. random variables, $X^{(\psi)} := \max_{i\in[\psi]}\{X_i\}$ and $Y^{(\psi)} := \max_{i\in[\psi]}\{Y_i\}$. Let $Z := \sum_{i=1}^{\tau}(X^{(\psi)} - Y^{(\psi)})$. Then,*

$$\Pr\left[|Z| \ge jr'\right] \le e^{-jr}.$$

5 AVERAGING INTERACTIONS IS INEFFICIENT

In this section we study the fitness measure f^{avg}. We show that the $(1, \lambda)$ CoEA using f^{avg} takes exponential time to reach the optimum.

Theorem 5.1. *Consider Algorithm 1 with fitness measure f^{avg} and geometric mutation operator on* Bilinear. *Define $T = \min\{t \mid \mathrm{OPT} \cap (P_t \times Q_t) \neq \emptyset\}$. Let $d := |x_1 - \beta| + |y_1 - \alpha|$. If the following conditions hold:*

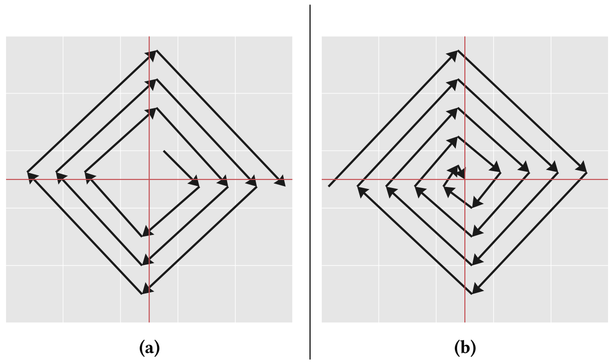

Figure 2: Expected behaviour of $(1,\lambda)$ CoEA with fitness measure f^{avg} (a) and fitness measure f^{wrs} (b) on Bilinear.

- $\lambda \le d^{a-2}$ *for some constant* $a > 2$.
- $\left(1 - e^{-\frac{1}{1-p}}\right)\log_{\frac{1}{1-p}} \lambda/2 \ge 2$

Then, $T \ge e^{\Omega(d)}$ *with probability* $1 - e^{-\Omega(d)}$.

The main idea of the proof of Theorem 5.1 is that the algorithm moves through each quadrant in a clockwise manner. While inside a quadrant the algorithm maintains its distance to the optimum in expectation and each time it moves from one quadrant to the other the last generation continues in the wrong direction moving away from the optimum. Figure 2 (a) visualises this behaviour.

To facilitate the analysis we divide a run of the algorithm into different intervals that denote when the algorithm moves from one quadrant to the other (not necessarily in a clockwise manner).

Definition 5.2 (Blocks). Let $\tau_i = \min\{t > \tau_{k-1} \mid (x_t - \beta)(x_{\tau_{k-1}} - \beta) < 0 \lor (y_t - \alpha)(y_{\tau_{k-1}} - \alpha) < 0\}$ and $\tau_0 := 0$. We divide a run into generation intervals $(\tau_k, \tau_{k+1}]$ that we call blocks.

The first step is to show that the algorithm does not spend too much time in one block. This is shown in the following lemma.

Lemma 5.3. *Let* $c > 0$, $a > 2$ *be constants such that for some* $d > 0$, $d^a - cd = \Omega(d^{a-1})$. *Consider the* $(1,\lambda)$ CoEA *with* $\lambda \le d^{a-2}$ *starting a block with a Manhattan distance* $cd > 0$. *The probability that the algorithm spends more than* d^a *generations during a block is* $e^{-\Omega(d)}$.

Proof. By the definition of a block once the algorithm moves to another quadrant a block finishes. Here we omit all possibilities of moving to another quadrant except by moving to the quadrant that is clockwise to the current one. Since the initial Manhattan distance is cd the algorithm needs to move in one direction (clockwise) by at least cd.

To explain the computations we assume that we are in the second quadrant ($x_t - \beta > 0$ and $y_t - \alpha > 0$) and the algorithm moves *downwards* to the next quadrant in expectation, that is y is decreasing until it reaches $y < \alpha$. Thanks to the symmetry of the behaviour of the algorithm in the quadrants, the computations hold for all quadrants.

To fit the perspective of the additive drift tail bounds [9, Theorem 2.4.7] we use a potential function $h(y_t) := \min\{0, y_t - \alpha\}$. Let $\Delta := h(y_t) - h(y_{t+1})$ and note that $\Delta \ge 0$ as long as $X_t > 0$ because the algorithm creates at least $\lambda/2$ offspring with the same value of y_t or less and by Lemma 3.1 the algorithm selects the offspring with the smallest value. If $X_t \le 0$, the opposite is true, and then $\Delta \le 0$.

Recall that $X_t := \sum_{x \in P_t}(x - \beta)$. Let $h := x_t - \beta > 0$. Then X_t is equivalent to

$$\sum_{i=1}^{\lambda/2}(X_i + h) - \sum_{i=1}^{\lambda/2}(X_i - h),$$

where X_i are i.i.d. random variables sampled from $\mathcal{G}(p)$. Then, by Lemma 4.1, $\Pr[X_t \le 0 \mid x_t - \beta > 0] = e^{-\Omega(\lambda)}$.

By Lemma 4.2 using $\psi = \lambda/2$ we obtain

$$\begin{aligned}
\mathrm{E}[\Delta] &\ge \Pr[X_t \le 0 \mid x_t - \beta > 0]\,\mathrm{E}[\Delta \mid X_t \le 0] \\
&\quad + \Pr[X_t > 0 \mid x_t - \beta > 0]\,\mathrm{E}[\Delta \mid X_t > 0] \\
&\le e^{-\Omega(\lambda)}\left(\log_{\frac{1}{1-p}}(\lambda/2) + \frac{1-p}{p}\right) \\
&\quad + \left(1 - e^{-\Omega(\lambda)}\right)\left(1 - e^{-\frac{1}{1-p}}\right)\log_{\frac{1}{1-p}} \lambda/2 \\
&= \Omega(\log \lambda)
\end{aligned}$$

This shows condition (2.24) from [9, Theorem 2.4.7].

Independent on if the algorithm moves upward or downward the $\Pr[|\Delta| > j] \le \Pr[Z \ge j]$ with $Y_i \sim \mathcal{G}(p)$ and $Z := \max_{i \in [\lambda/2]}\{Y_i\}$. Therefore, by Lemma 4.3 (13) with $c = 2e^j/\lambda$ we obtain,

$$\Pr[|\Delta| > j] \le \frac{\lambda}{2e^j},$$

and the last condition (2.21) from [9, Theorem 2.4.7] is met with $\eta = 1 - e$ and $r = \lambda/2$.

Let, τ be the time the algorithm spends in a block, then by the additive drift tail bounds.

$$\begin{aligned}
&\Pr\left[\tau \ge d^a\right] \\
&\le \exp\left(\left(-\frac{(1-e)(d^a - cd))}{8}\right)\left(\frac{(1-e)^2(d^a - cd)}{16\lambda cd}\right)\right) \\
&= \exp\left(-\Omega(d^{a-1})\right) = e^{-\Omega(d)}.
\end{aligned}$$

□

Now, we show that the algorithm does not *jump* far away from its current parents during one generation.

Lemma 5.4. *Let* $d > 0$. *The probability that a generation from Algorithm 1 using the geometric mutation operator starts with parents* x, y *and creates a pair of points* x', y' *with a Manhattan distance from their parents* $|x - x'| + |y - y'| \ge d$ *is at most* $\lambda^3 e^{-dp}$. *If* $\lambda^3 = e^{o(d)}$ *this is* $e^{-\Omega(d)}$.

Proof. By Lemma 4.3 with $\psi = \lambda$, the probability of creating offspring x' and y' with distance $|x - x'|$ and $|y - y'|$ from x and y respectively is

$$\min\left\{\lambda^{1-|x-x'|\frac{p}{\ln\lambda}}, 1\right\} \cdot \min\left\{\lambda^{1-|y-y'|\frac{p}{\ln\lambda}}, 1\right\} \tag{15}$$

If none of the two expressions are minimised by 1, then this becomes

$$\lambda^{2-(|x-x'|+|y-y'|)\frac{p}{\ln\lambda}} < \lambda^{2-cd\frac{p}{\ln\lambda}} = \lambda^2 e^{-dp}.$$

On the other hand if one of the expressions is minimised by 1, then Equation (15) implies that either $|x - x'| \le \ln\lambda/p$ and $|y - y'| \ge$

$d - \ln \lambda/p$ or $|y - y'| \le \ln \lambda/p$ and $|x - x'| \ge d - \ln \lambda/p$. Therefore, Equation (15) yields

$$\lambda^{2-(d-\frac{\ln\lambda}{p})\frac{p}{\ln\lambda}} = \lambda^{3-cd\frac{p}{\ln\lambda}} = \lambda^3 e^{-dp}.$$

Both cases are at most $\lambda^3 e^{-dp}$. If $\lambda^3 = e^{o(d)}$ then there is a constant $c > 0$ for which $\lambda^3 e^{-dp} \le e^{-dc}$. □

The main part of the analysis is to show that there is a large region (of size $\Theta(d)$) in the search space where the algorithm moves away from the optimum in expectation after each block.

Lemma 5.5. *Let $c > 0$ be a constant. Let $\left(1 - e^{-\frac{1}{1-p}}\right)\log_{\frac{1}{1-p}} \lambda/2 \ge 2$. Let $M_k := \left|x_{\tau_k} - \beta\right| + \left|y_{\tau_k} - \alpha\right| > cd$ be the Manhattan distance at the beginning of block k. Then, there exists a constant $\delta > 0$ such that $\mathrm{E}[M_{k+1} - M_k \mid M_k > cd] \ge \delta$.*

Proof. By Lemma 5.3 there are at most d^a generations (for some constant $a > 2$) in one block with probability $1 - e^{-\Omega(d)}$. We call E_1 to the event that a block takes longer than d^a generations. If event E_1 happens we pessimistically assume that the optimum is found during the block, that is $M_{k+1} = 0$.

By the dynamics of the algorithm on the Bilinear problem every block is spent in a quadrant of the search space located clockwise from the previous block, unless there was a generation within the block with at least one pair of search points from $P_t \times Q_t$ in all of the four quadrants of the search space. We call E_2 the event that during the current block there is a generation t where the algorithm creates at least one pair of search points from $P_t \times Q_t$ in all of the four quadrants and pessimistically assume that if this happens the optimum is found during the block. Therefore $\overline{E_2}$ guarantees that the next block is located clockwise from the previous block.

If event E_1 does not happen, by Lemma 5.4 the probability that event E_2 happens during one generation is $e^{-\Omega(d)}$. By a union bound over d^a generations, the probability of E_2 is still $e^{-\Omega(d)}$. Then

$$\Pr[E_1 \vee E_2]\,\mathrm{E}[M_{k+1} - M_k \mid M_k > cd \wedge (E_1 \vee E_2)] \ge -cde^{-\Omega(d)} \tag{16}$$

From now on we consider only cases where events E_1 and E_2 does not happen. For simplicity, in the following we also assume that the block k is in the second quadrant of the search space and note that thanks to the symmetry of the behaviour of the algorithm the computation holds for all quadrants.

Recall the definitions $X_t := \sum_{x \in P_t}(x - \beta)$ and $Y_t := \sum_{y \in Q_t}(y - \alpha)$. We assume that if $x_t - \beta > 0$ then $X_t > 0$ because of the following: if this is not the case during one generation, (with block k in the second quadrant) then by Lemma 3.1 the algorithm would choose either the pair of offspring farthest away from the optimum (if $X_t < 0$) or a pair of offspring that is at least as far as the one that would be chosen otherwise (if $X_t = 0$). Given this assumption, by Lemma 4.4 all generations but the last one have an expected change in distance to the optimum of 0.

There are two different cases for the expected change in distance depending on where the last generation $t = \tau_{k+1} - 1$ of a block starts. Specifically, the two cases are $y_t - \alpha = 0$ and $y_t - \alpha > 0$. We denote these cases as E_3 and E_4.

If the last generation starts with $y_t - \alpha = 0$ (event E_3) then there is a probability ξ that $Y_t = 0$ and x_{t+1} is chosen uniformly at random from P_t. Since the offspring in P_t have the same probability of having a value $x_t + i$ and $x_t - i$ for all $i \in \mathbb{N}$ then in expectation $x_t = x_{t+1}$ and the pair x_{t+1}, y_{t+1} are farther from the optimum by $y_t - y_{t+1}$ which is at least $\left(1 - e^{-\frac{1}{1-p}}\right)\log_{\frac{1}{1-p}} \lambda/2$ in expectation (Lemma 4.2). There is also a probability $(1-\xi)/2$ of $Y_t > 0$ and $(1-\xi)/2$ of $Y_t < 0$. In the former case by Lemma 3.1 the algorithm selects $x_{t+1} = \arg\max_{x \in P_t}\{x\}$, that is, the offspring farthest away from the optimum. Pessimistically assuming that $y_t = y_{t+1}$, the pair x_{t+1}, y_{t+1} are farther from the optimum by at least $\left(1 - e^{-\frac{1}{1-p}}\right)\log_{\frac{1}{1-p}} \lambda/2$ (Lemma 4.2). Finally, if $Y_t < 0$ in expectation the distance to the optimum is at least the same, therefore,

$$\begin{aligned}
&\mathrm{E}[M_{k+1} - M_k \mid M_k > cd \wedge E_3] \\
&\ge \left(\frac{1-\xi}{2} + \xi\right)\left(1 - e^{-\frac{1}{1-p}}\right)\log_{\frac{1}{1-p}} \lambda/2 \\
&\ge \frac{1}{2}\left(1 - e^{-\frac{1}{1-p}}\right)\log_{\frac{1}{1-p}} \lambda/2 \ge 1
\end{aligned}$$

For event E_4, we denote $h := y_t - \alpha > 0$. As event E_3 there are three outcomes depending on the value of Y_t. Therefore, we use Lemma 4.1 to compute $\Pr[Y_t \le 0 \mid y_t - \alpha > 0]$. Recall that $Y_t := \sum_{y \in Q_t}(y - \alpha)$. Then Y_t is equivalent to

$$\sum_{i=1}^{\lambda/2}(Y_i + h) - \sum_{i=\lambda/2+1}^{\lambda}(Y_i - h),$$

where Y_i are i.i.d. random variables sampled from $\mathcal{G}(p)$. Then, by Lemma 4.1

$$\Pr[Y_t \le 0 \mid y_t - \alpha > 0] = e^{-\Omega(\lambda)}.$$

If $Y_t < 0$, by Lemmas 3.1 and 4.2 x_{t+1} moves towards the optimum by at most $\left(\log_{\frac{1}{1-p}} \lambda/2 - \frac{1-p}{p}\right)$ in expectation. If $Y_t = 0$, $x_t = x_{t+1}$ in expectation. For the y direction, we are interested in the change in distance away from the optimum solution given that $y_t > \alpha$ and $y_{t+1} < \alpha$, that is, $(\alpha - y_{t+1}) - (y_t - \alpha)$. The event $Y_t \le 0$ is positively correlated with $\alpha - y_{t+1}$ because it implies that there are more offspring with values less than α than the expectation and they are farther from the optimum. For the same reason it is also negatively correlated with $y_t - \alpha$. Therefore we can bound the change in distance to the optimum in the y direction by the unconditional expected step size given by Lemma 4.2. Adding both directions results in a decrease in distance to the optimum by at most $2\left(\log_{\frac{1}{1-p}} \lambda/2 - \frac{1-p}{p}\right)$ in expectation for both $Y_t < 0$ and $Y_t = 0$.

Finally, for $Y_t > 0$ with $y_t - \alpha > 0$ we denote $\Delta_y := y_{t+1} - y_t$ and $\Delta_x := x_{t+1} - x_t$. We aim to show that $\mathrm{E}[\Delta_y] = \mathrm{E}[\Delta_x] - o(1)$ which in conjunction with the fact that $y_{t+1} > \alpha$ imply that in the worst case (best case for the algorithm) the Manhattan distance to the optimum increases by at least $2 - o(1)$.

Given the condition $Y_t > 0$ we have $\mathrm{E}[\Delta_y] = \mathrm{E}\left[Y^{(\lambda/2)} \mid Y_t > 0\right]$, where $Y^{(\lambda/2)}$ is the sum of $\lambda/2$ geometrically distributed random variables. By the law of total expectation,

$$\mathrm{E}\left[Y^{(\lambda/2)} \mid Y_t > 0\right] = \frac{\mathrm{E}\left[Y^{(\lambda/2)}\right] - \mathrm{E}\left[Y^{(\lambda/2)} \mid Y_t \le 0\right] \cdot \Pr[Y_t \le 0]}{\Pr[Y_t > 0]}$$

We note that the condition $Y_t \le 0$ is equivalent to

$$\sum_{i=1}^{\lambda/2}(Y_i + h) - \sum_{i=\lambda/2+1}^{\lambda}(Y_i - h) \le 0.$$

If all Y_i with $i \in (\lambda/2, \lambda]$ are greater or equal to $2h + \max_{i\in[1,\lambda/2]}\{Y_i\}$ then $Y_t \le 0$ is guaranteed. Thanks to the *forgetfulness* of the geometric distribution, for $Y_i \sim \mathcal{G}(p)$, $\mathrm{E}[Y_i \mid Y_i \ge j] = j + \mathrm{E}[Y_i]$. Therefore,

$$\mathrm{E}\left[Y^{(\lambda/2)} \mid Y_t \le 0\right] \le 2h + 2\mathrm{E}\left[Y^{(\lambda/2)}\right] \le 4h\mathrm{E}\left[Y^{(\lambda/2)}\right],$$

where the last inequality holds because $\mathrm{E}\left[Y^{(\lambda/2)}\right] > 1$ by the assumptions and Lemma 4.2 and $h \ge 1$. As mentioned before, by Lemma 4.1 $\Pr[Y_t \le 0] = e^{-\Omega(h\lambda)}$ and

$$\mathrm{E}\left[Y^{(\lambda/2)} \mid Y_t \le 0\right] \cdot \Pr[Y_t \le 0] \le 4h\mathrm{E}\left[Y^{(\lambda/2)}\right]e^{-\Omega(h\lambda)} = o(1),$$

Hence,

$$\mathrm{E}\left[Y^{(\lambda/2)} \mid Y_t > 0\right] \ge \frac{\mathrm{E}\left[Y^{(\lambda/2)}\right]}{1 - e^{-\Omega(\lambda)}} - o(1) = \mathrm{E}\left[Y^{(\lambda/2)}\right] - o(1),$$

and the Manhattan distance to the optimum increases by at least $2 - o(1)$. Joining the three cases for event E_4, we obtain

$$\mathrm{E}[M_{k+1} - M_k \mid M_k > cd \wedge E_4]$$
$$\ge -2\left(\log_{\frac{1}{1-p}} \lambda/2 - \frac{1-p}{p}\right)e^{-\Omega(\lambda)} + (2 - o(1))(1 - e^{-\Omega(\lambda)}) \ge \delta'.$$

Where $0 < \delta' < 2 - o(1)$ is a constant independent of d and λ.

Joining all events together yields,

$$\begin{aligned}
&\mathrm{E}[M_{k+1} - M_k \mid M_k > cd] = \\
&\Pr[E_1 \vee E_2]\,\mathrm{E}[M_{k+1} - M_k \mid M_k > cd \wedge (E_1 \vee E_2)] + \\
&\Pr[E_3]\,\mathrm{E}[M_{k+1} - M_k \mid M_k > cd \wedge E_3] + \\
&\Pr[E_4]\,\mathrm{E}[M_{k+1} - M_k \mid M_k > cd \wedge E_4] \\
&\quad \ge -cde^{-\Omega(d)} + \Pr[E_3] + \delta'\Pr[E_4] \\
&\quad \ge -cde^{-\Omega(d)} + \delta'\Pr[E_3 \vee E_4]/2 \\
&\quad \ge -cde^{-\Omega(d)} + \delta'(1 - e^{\Omega(d)})/2 \ge \delta,
\end{aligned}$$

where $\delta > 0$ is a constant independent of d and λ. □

With the previous helper lemmas we are now in position to prove the main theorem of this section.

Proof of Theorem 5.1. We aim to use the negative drift theorem with scaling [14, Theorem 2]. By Lemma 5.5 every block starting with a Manhattan distance cd from the optimum has at least constant negative drift $\delta > 0$. This meets condition (1) of [14, Theorem 2].

For Condition (2), we use Lemma 4.6 with $r' = \frac{16\sqrt{2\ln\lambda/2}}{p}$. Then, for all blocks excluding the last generation (from $t = \tau_k + 1$ to $t = \tau_{k+1} - 1$), the following holds,

$$\Pr\left[\left|M_{\tau_k+1} - M_{\tau_{k+1}-1}\right| \ge jr'\right] \le e^{-2j}$$

Then, we need to only consider the last generation. In the worst case both $x_{\tau_{k+1}}$ and $y_{\tau_{k+1}}$ move towards/away from the optimum. By Lemma 5.4 the probability of creating an offspring at a distance at least ℓ is at most $\lambda^3 e^{-\ell p} = e^{3\ln\lambda - \ell p}$. Using $\ell = \frac{4j\ln\lambda}{p}$ yields,

$$e^{3\ln\lambda - \ell p} = e^{3\ln\lambda - 4j\ln\lambda} \le e^{-j\ln\lambda}.$$

Then choosing $r = r' + \frac{4\ln\lambda}{p}$,

$$\Pr\left[\left|M_{\tau_k+1} - M_{\tau_{k+1}}\right| \ge jr\right] \le e^{-2j} + e^{-j\ln\lambda} \le e^{-j}.$$

We note that $r^2 = O(\log^2\lambda)$. Then Condition (3) of [14, Theorem 2] holds thanks to the assumption $\lambda = e^{o(d)}$. □

6 FITNESS BASED ON WORST INTERACTIONS IS EFFICIENT

Owing to the large number of different behaviours the fitness measure f^{wrs} can take, its analysis is much more involved. In Section 4 we have seen that the geometric mutation operator generates offspring around the parents in a *square* pattern (as the example populations shown in Figure 1 (b)), and deviations from the expected behaviour have exponential decay. Although we believe the main result of this section (Theorem 6.1) holds for the geometric mutation operator, we simplify the computations by assuming that the offspring in each generation are always created around the parents in an exact *square* pattern. For the readers' convenience we present the pseudo-code of this instantiation of the $(1,\lambda)$ CoEA in Algorithm 2.

Algorithm 2: Deterministic $(1,\lambda)$ CoEA

```
1  Require: Max-min-objective function $g : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$.
2  Require: Population size $\lambda = 2r + 1$ with radius $r \in \mathbb{N}$, and
           fitness measures $f_x : \mathcal{X} \times \mathcal{Y}^\lambda$, $f_y : \mathcal{Y} \times \mathcal{X}^\lambda$
3  Initialization: Sample $x_1 = 0$ and $y_1 = 0$.
4  for $t \in \mathbb{N}$ do
5      Mutation: for $i \in \{0, \ldots, \lambda - 1\}$ do
6          $P_t(i) \leftarrow x_t - r + i$;
7          $Q_t(i) \leftarrow y_t - r + i$;
8      Selection:
9      Choose $x_{t+1} \leftarrow \arg\max_{P_t(i)}\{f_x(P_t(i), Q_t)\}$;
10     Choose $y_{t+1} \leftarrow \arg\min_{Q_t(i)}\{f_y(Q_t(i), P_t)\}$;
```

Theorem 6.1. *Consider Algorithm 2 with fitness measure f^{wrs} and radius $r \ge 1$ on* Bilinear. *Define $T = \min\{t \mid \mathrm{OPT} \cap (P_t \times Q_t) \ne \emptyset\}$. Let $M_1 := |x_1 - \beta| + |y_1 - \alpha| > r$ be the initial Manhattan distance to the optimum. Then, $\mathrm{E}[T] \le \frac{M_1^2}{r} - r$. If $M_t \le r$ then $T = 1$.*

7 EXPERIMENTS

Although our theoretical results have given us important insights of how the fitness measures affect the performance of CoEAs, they are limited to only one search space ($\mathcal{X} = \mathcal{Y} = \mathbb{Z}$). It is unclear if our results translate to other search spaces. In this section, we conduct an experimental analysis that aims to complement our

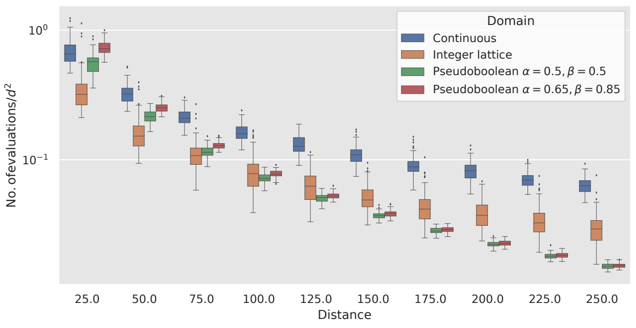

Figure 3: Runtime of the $(1,\lambda)$ CoEA **(Algorithm 1) with fitness measure** f^{wrs} **on** **Bilinear.**

theoretical results with precise runtime results for the $(1,\lambda)$ CoEA (Algorithm 1) with different fitness measures on Bilinear problems defined over three different search spaces ($\mathcal{X} = \mathcal{Y} = \mathbb{Z}, \mathcal{X} = \mathcal{Y} = \mathbb{R}$ and $\mathcal{X} = \mathcal{Y} = \{0,1\}^n$). For the search space $\mathcal{X} = \mathcal{Y} = \{0,1\}^n$ we use the definition of Bilinear used by [8] with $n = 500$ in all experiments and two parameter values ($\alpha = \beta = 0.5$ and $\alpha = 0.8, \beta = 0.75$). We note that the mutation operator in the case $\alpha = \beta = 0.5$ has an inherent *genetic drift* towards the optimum ($n/2$). Therefore the case $\alpha = \beta = 0.5$ should be easy to optimise even for a $(1,\lambda)$ CoEA that selects a random offspring every time.

For the three different search spaces the $(1,\lambda)$ CoEA uses the following mutation operators:

- $\mathcal{X} = \mathcal{Y} = \mathbb{Z}$ - **Geometric mutation** (random direction and $p = 1/2$): For a parent x, the offspring $x' = x + \mathcal{G}(1/2)$ with probability $1/2$ and $x' = x - \mathcal{G}(1/2)$.
- $\mathcal{X} = \mathcal{Y} = \mathbb{R}$ - **Gaussian mutation** (step size of 1): For a parent x, the offspring $x' = x + \mathcal{N}(0,1)$.
- $\mathcal{X} = \mathcal{Y} = \{0,1\}^n$ - **Standard bit mutation** ($p = 1/n$): An offspring is created by copying the parent and flipping each bit with probability $1/n$.

All experiments comprise of 100 runs for each algorithm-problem pair, recording the number of evaluations (interactions) to reach the optimum. Each run is limited to 10^9 evaluations, if the run does not find the optimum within this time limit, 10^9 is reported as the runtime. For the search space $\mathcal{X} = \mathcal{Y} = \mathbb{R}$ we assume the optimum is found if both $x = [\beta - 0.1, \beta + 0.1]$ and $y = [\alpha - 0.1, \alpha + 0.1]$.

We first explore the fitness measure f^{wrs}. In Figure 3 we show in the x-axis the initial Manhattan distance to the optimum[3] and in the y-axis the runtime divided by d^2 and log-scaled. Given that the number of evaluations are normalised by d^2 we can appreciate that for the distances studied and all search spaces, the runtime seems to grow slower than d^2. This indicates that our bounds might not be tight or that they are tight but only for large initial distances d. Nonetheless, these experiments are evidence that our theoretical results on f^{wrs} do translate to other search spaces.

Similarly, the experimental results for the fitness measure f^{avg} (Figure 4) show that our theoretical results translate to other search spaces. For these experiments it is particularly interesting to see

[3]For the search space $\mathcal{X} = \mathcal{Y} = \{0,1\}^n$ the Manhattan distance is defined as the sum of the Hamming distances of both x and y to an optimal solution.

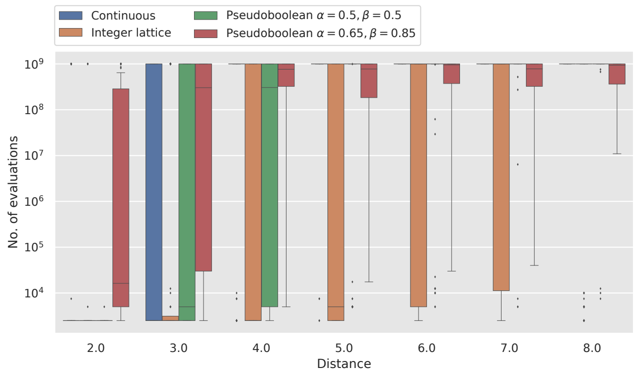

Figure 4: Runtime of the $(1,\lambda)$ CoEA **(Algorithm 1) with fitness measure** f^{avg} **on** **Bilinear.**

that the runtime *explodes* even for small initial distances to the optimum.

8 CONCLUSIONS

We have shown how the selection of fitness measures dramatically affects the performance of the $(1,\lambda)$ CoEA on Bilinear. Both fitness measures studied here present a cyclic behaviour on this problem. Using the average payoff of interactions worsens this behaviour resulting in exponential time to find an optimal solution. On the other hand using the worst payoff alleviates the problem resulting in an efficient optimisation.

An important insight from our theoretical analysis is that both fitness measures tend to maintain the distance to the optimum in most generations and what differentiates them is the behaviour during the generation where the current solutions are part of more than one quadrant of the search space. While using the worst interaction as fitness decreases the distance to the optimum averaging interactions increases it. We hope that this insight inspires the design of better CoEAs that exploit this behaviour.

It remains an open problem, whether the results shown here apply to other intransitive problems and how other fitness measures affect the performance of CoEAs on intransitive problems. Furthermore, an interesting venue for future work is to theoretically analyse other classes of problems to examine whether CoEAs are affected by the fitness measure on these problems to the same extent as seen in this work.

ACKNOWLEDGMENTS

This research was supported by a Turing AI Fellowship (EPSRC grant ref EP/V025562/1). The computations described in this paper were performed using the University of Birmingham's BlueBEAR HPC service. See http://www.birmingham.ac.uk/bear for more details.

REFERENCES

[1] Jürgen Branke and Johanna Rosenbusch. New approaches to coevolutionary worst-case optimization. In *Parallel Problem Solving from Nature – PPSN X*, pages 144–153, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.
[2] Anthony Bucci. *Emergent Geometric Organization and Informative Dimensions in Coevolutionary Algorithms*. Brandeis University, Massachusetts, USA, 2007.

[3] Diana Flores, Erik Hemberg, Jamal Toutouh, and Una-May O'Reily. Coevolutionary generative adversarial networks for medical image augumentation at scale. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '22, page 367–376, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450392372.

[4] 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. ISSN 2688-299X.

[5] W.Daniel Hillis. Co-evolving parasites improve simulated evolution as an optimization procedure. *Physica D: Nonlinear Phenomena*, 42(1-3):228–234, June 1990.

[6] Thomas Jansen and R. Paul Wiegand. The Cooperative Coevolutionary (1+1) EA. *Evolutionary Computation*, 12(4):405–434, 12 2004.

[7] Mikkel T Jensen. A New Look at Solving Minimax Problems with Coevolution. *Metaheuristics: Computer Decision-Making*, 86:369, 2003.

[8] 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.

[9] Johannes Lengler. *Drift Analysis*, pages 89–131. Springer, 2020.

[10] Tengyuan Liang and James Stokes. Interaction Matters: A Note on Non-asymptotic Local Convergence of Generative Adversarial Networks. *ArXiv e-prints*, 2019.

[11] Sean Luke and R Paul Wiegand. When coevolutionary algorithms exhibit evolutionary dynamics. In *Genetic and Evolutionary Computation Conference Workshop Program*, pages 236–241, 2002.

[12] Xiaoliang Ma, Xiaodong Li, Qingfu Zhang, Ke Tang, Zhengping Liang, Weixin Xie, and Zexuan Zhu. A survey on cooperative co-evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 23(3):421–441, 2019. doi: 10.1109/TEVC.2018.2868770.

[13] Aryan Mokhtari, Asuman Ozdaglar, and Sarath Pattathil. A Unified Analysis of Extra-gradient and Optimistic Gradient Methods for Saddle Point Problems: Proximal Point Approach. *ArXiv e-prints*, 2019.

[14] Pietro S. Oliveto and Carsten Witt. Improved time complexity analysis of the simple genetic algorithm. *Theoretical Computer Science*, 605:21 – 41, 2015. ISSN 0304-3975.

[15] 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 Berlin Heidelberg, Berlin, Heidelberg, 2012.

[16] Christopher D. Rosin and Richard K. Belew. New Methods for Competitive Coevolution. *Evolutionary Computation*, 5(1):1–29, 03 1997. ISSN 1063-6560.

[17] Karl Sims. Evolving 3d morphology and behavior by competition. *Artificial Life*, 1(4):353–372, 1994.

[18] R. Paul Wiegand, William C. Liles, and Kenneth A. De Jong. An empirical analysis of collaboration methods in cooperative coevolutionary algorithms. In *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*, GECCO'01, page 1235–1242, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 1558607749.

[19] Rudolf Paul Wiegand. *An analysis of cooperative coevolutionary algorithms*. George Mason University, Virginia, USA, 2004.

[20] Guojun Zhang and Yaoliang Yu. Convergence of Gradient Methods on Bilinear Zero-Sum Games. *ArXiv e-prints*, 2020.