Analysis of a Pairwise Dominance Coevolutionary Algorithm And DefendIt

Per Kristian Lehre[*]
Mario Hevia Fajardo
p.k.lehre@bham.ac.uk
m.heviafajardo@bham.ac.uk
University of Birmingham, UK

Jamal Toutouh
jamal@uma.es
University of Malaga, Spain

Erik Hemberg[*]
Una-May O'Reilly
erik.hemberg@gmail.com
unamay@csail.mit.edu
MIT, USA

ABSTRACT

While competitive coevolutionary algorithms are ideally suited to model adversarial dynamics, their complexity makes it difficult to understand what is happening when they execute. To achieve better clarity, we introduce a game named DefendIt and explore a previously developed pairwise dominance coevolutionary algorithm named PDCoEA. We devise a methodology for consistent algorithm comparison, then use it to empirically study the impact of population size, the impact of relative budget limits between the defender and attacker, and the impact of mutation rates on the dynamics and payoffs. Our methodology provides reliable comparisons and records of run and multi-run dynamics. Our supplementary material also offers enticing and detailed animations of a pair of players' game moves over the course of a game of millions of moves matched to the same run's populations' payoffs.

CCS CONCEPTS

• **Theory of computation** → **Evolutionary algorithms**; **Adversary models**; • **Security and privacy** → *Vulnerability management*.

KEYWORDS

co-evolution, cyber security, evolutionary algorithms

ACM Reference Format:
Per Kristian Lehre, Mario Hevia Fajardo, Jamal Toutouh, Erik Hemberg, and Una-May O'Reilly. 2023. Analysis of a Pairwise Dominance Coevolutionary Algorithm And DefendIt. In *Genetic and Evolutionary Computation Conference (GECCO '23), July 15–19, 2023, Lisbon, Portugal.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3583131.3590411

1 INTRODUCTION

Security scenarios frequently feature adversarial behavior where a defensive side must repel attacks. The evolving dynamics of such two-sided scenarios quickly become complicated when each side is able to adapt using the feedback from competitions and when a scenario is comprised of multiple defensive and offensive adversaries that mix through competitions among each other. Algorithms can effectively express these salient features of security scenarios, allowing modeling and simulation to help understand them better. For example, competitive coevolutionary algorithms (CCAs) have competing populations that map to the multiple adversaries of security scenarios, such as a multiple different organized crime groups attempting to compromise the computer network of a financial institution. They set up and evaluate pairwise competitions and they genotypically vary population members based on fitness - a means of modeling the collection of feedback and its influence on adaptation of the adversaries. They also have selection that mimics the proliferation of successful behavior over less successful behavior. However, the aspects of the CCA that mimic these security scenarios generate enough complexity to make the algorithms as intractable for sound empirical or theoretical analysis as the security scenarios themselves. In addition to stochasticity and population-based aspects of evolutionary algorithms, CCAs have unique aspects including relativistic fitness scoring and the interactions of two populations in the context of two-player competitions.

In this contribution, to gain more clarity, we combine a new game, DefendIt, that is simple to describe but complex in behavior with a previously theoretically analyzed CCA named PDCoEA [16] (see in Section 3.1, Algorithm 1). Consider the two player game, named FlipIt [26], between an attacker and defender with each player being able to move or not move (coded as one or zero respectively) at each step in a time sequence. The environment consists of a resource under ownership of either the attacker or defender with the defender initially being the owner. A move is an attempt to take or retain ownership of a resource. Players have no access to each other's moves at any time, nor any feedback during the game. Accordingly, at the beginning of a game, each player states a *strategy* which is a series of moves (and non-moves) to be taken in sequence. The sequence then is stepped forward and moves are played whenever at least one player has specified them to be taken. When there are no moves at a time step, resource ownership does not change. When only one player moves, that player assumes or retains ownership of the resource. When both players move, the ownership does not change. At the end of the sequence of steps, players are awarded a payoff proportional to the duration they were owners of the resource. The objective of each player is to maximize its ownership duration while minimizing its total move cost. This is illustrated in Figure 1.

[*]Corresponding authors.

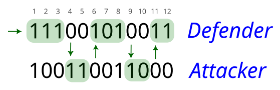


Figure 1: FlipIt or equivalently a round of DefendIt on a single resource. Initially, the resource is held by the defender. At time step 1, both the defender and the attacker attempt to acquire the resource, and ownership remains with the defender. At time step 4, the attacker acquires the resource. The defender re-acquires the resource at time step 6 and keeps it until time step 9 when the attacker regains ownership. The resource changes hand a final time at time step 11. At the end of the game, the defender has owned the resource for 8 time steps, while the attacker has owned it for 4 time steps.

The game we study in this contribution, DefendIt, follows from FlipIt, making FlipIt more complex in two respects, e.g. contesting multiple devices in a heterogeneous network with finite attack and defense resources. DefendIt has multiple resources with independent move costs. Total payoff is the summed ownership payoffs of each resource. Second, DefendIt also introduces budgets for both players. These budgets are taken into account after all moves have been played and ownership durations have been determined. If the total cost of a player's moves has exceeded their budget, their payoff is docked a penalty. This penalty is equivalent to the number of moves made. We describe DefendIt formally in Section 3.2 and analyze its hardness in Section 4.

A coevolutionary algorithm is ideal to model and study DefendIt and, *vice versa*, DefendIt's move costs, multiple resources, and budgets are ideal parameters with which to study coevolution.

2 RELATED WORK

Biological coevolution refers to the influences two or more species exert on each other's evolution [23]. A seminal paper on reciprocal relationships between insects and plants coined "coevolution" [7]. Coevolution can be cooperative, i.e., mutual benefit, or competitive, i.e., negative interactions arising from constrained and shared resources or from predator-prey relationships.

Well known to the reader, an Evolutionary Algorithm (EA) typically evolves individual solutions, e.g., fixed length bit strings as in Genetic Algorithms (GAs) [10] using an *a-priori* defined fitness function to evaluate an individual's quality. In contrast, coevolutionary algorithms calculate an individual's fitness based on its interactions with other individuals or a dynamic environment allowing them to mimic coupled natural species-to-species interactions.

This contribution attempts to extend the insightful body of prior work studying and exploiting coevolutionary algorithms [2, 12, 14, 19, 21–24]. It focuses on Competitive Coevolutionary Algorithms (CCAs). In a basic CCA at each generation an individual is assigned a fitness score derived from some function of its performance outcomes in its competitions. E.g., the function can sum the performance outcomes or use the average, maximum, minimum, or median outcome [3]. Variations of the CCA have been defined for specific problem domains, e.g., [1, 4, 9, 17, 18, 25].

The dynamics of CCAs are difficult to analyze because their fitness evaluation is derived *interactively*, [22]. Effectively an individual's score (or payoffs in the case of DefendIt) is a sample-based estimate of their performance where the samples are drawn from the opposing population which itself is evolving. We introduce a methodology in Section 5 where samples from multiple runs, collected at the same time points in evolution are used to estimate fitness. This decreases the bias of the estimate from the run where the individual is evolving.

Different competitive games or simplified problems have been studied [3, 8, 13, 14]. As stated in the introduction, DefendIt is inspired by FlipIt. FlipIt was introduced in 2012 in a cryptographic setting by [26]. It was originally described as a game of "Stealthy Takeover" because it models "situations in which an attacker periodically compromises a system or critical resource completely, learns all its secret information and is not immediately detected by the system owner or defender." DefendIt is motivated by a more recent application of CCAs, that of modeling security scenarios such as those seen in cyber-networks, information and disinformation proliferation, and regulatory environments, e.g. such as in taxation where non-compliance competes with auditing [11, 12, 21].

There is a nascent body of theoretical analysis of coevolutionary algorithms. Relevant to this contribution is the topic of *error thresholds*, a phenomenon first studied in molecular biology [15, 20]. An error threshold is an essential characteristic of a non-elitist evolutionary algorithm. Informally, the threshold describes how the performance of the algorithm suddenly degrades when the mutation rate is increased beyond a certain point which depends on the selective pressure. For traditional non-elitist evolutionary algorithms which use selection and bitwise mutation applied to optimization on the Boolean hypercube, this threshold occurs when each bit is flipped with probability $\chi/n \approx \ln(\alpha_0)/n$, where α_0 is the reproductive rate of the selection mechanism (expected number of offspring of the fittest individual), and n is the bitstring length. Mutation rates above this threshold lead to exponentially large runtime on problems with at most a polynomial number of global optima (Theorem 4 in [15]), while mutation rates below this threshold lead to polynomial expected runtime assuming some additional algorithmic and problem conditions are met (Theorem 1 in [5]).

Error thresholds were recently documented in coevolutionary algorithms. Informally, for any sufficiently small subset $A \times B$ of the search space, the PDCoEA (Algorithm 1 in this paper) with bitwise mutation probability above $\ln(2)/n$ needs exponential time to sample any point in $A \times B$ with overwhelmingly high probability. For the formal statement, see Theorem 14 in [16].

3 PRELIMINARIES

We use the following notational conventions. For any $n \in \mathbb{N}$, we write $[n] := \{1, \ldots, n\}$. Given a set X and a function $f : X \longrightarrow \mathbb{R}$, we let $\arg\max_{x \in X} f(x)$ refer to an arbitrary element in X that takes the maximal f-value. Algorithm and game parameters are introduced in the Algorithm descriptions.

We start with coevolutionary algorithms, following PDCoEA (Algorithm 1) with two algorithms we use to gauge the value of a population size greater than one: *Pairwise Dominance (1+1) EA* (Algorithm 2) and *(1,λ) CoEA* (all-vs-all worst) (Algorithm 3)

3.1 Coevolutionary Algorithms

The basis of Algorithm 1 and Algorithm 2 is selection at the level of pairs of opposing players. In each selection step, the algorithms compare two pairs of opponents and select the "dominating" pair.

Definition 3.1 ([16]). Given two functions $g_1, g_2 : \mathcal{X} \times \mathcal{Y} \longrightarrow \mathbb{R}$ and two pairs $(x_1, y_1), (x_2, y_2) \in \mathcal{X} \times \mathcal{Y}$, we say that (x_1, y_1) dominates

(x_2, y_2) w.r.t. g_1 and g_2, denoted $(x_1, y_1) \succeq_g (x_2, y_2)$, if and only if $g_1(x_1, y_1) \geq g_1(x_2, y_1)$ and $g_2(x_1, y_1) \geq g_2(x_1, y_2)$.

Definition 3.1 is an immediate generalization of the maxmin-dominance relation defined in [16] (Definition 2) for the special case where the second payoff function is $g_2(x, y) := -g_1(x, y)$.

The pseudo-code for the pairwise dominance (1+1) EA and the (1,λ) CoEA (all-vs-all worst) is in Appendix C.

Algorithm 1 Pairwise Dominance CoEA (PDCoEA) [16]

```
Require: Payoff functions $g_1, g_2 : \{0,1\}^n \times \{0,1\}^n \to \mathbb{R}$.
Require: Population size $\lambda \in \mathbb{N}$ and mutation rate $\chi \in (0, n]$
1: for $i \in [\lambda]$ do
2:     Sample $P_0(i) \sim \mathrm{Unif}(\{0,1\}^n)$
3:     Sample $Q_0(i) \sim \mathrm{Unif}(\{0,1\}^n)$
4: for $t \in \mathbb{N}$ until termination criterion met do
5:     for $i \in [\lambda]$ do
6:         Sample $(x_1, y_1) \sim \mathrm{Unif}(P_t \times Q_t)$
7:         Sample $(x_2, y_2) \sim \mathrm{Unif}(P_t \times Q_t)$
8:         if $(x_1, y_1) \succeq_g (x_2, y_2)$ then
9:             $(x, y) := (x_1, y_1)$
10:        else
11:            $(x, y) := (x_2, y_2)$
12:        Obtain $x'$ by flipping each bit in $x$ with prob. $\chi/n$.
13:        Obtain $y'$ by flipping each bit in $y$ with prob. $\chi/n$.
14:        Set $P_{t+1}(i) := x'$ and $Q_{t+1}(i) := y'$.
```

3.2 Multi-Resource DefendIt

An instance of the multi-resource DefendIt game is given by a tuple $(k, \ell, v, c, B^D, B^A)$ where $k \in \mathbb{N}$ is the number of resources, $\ell \in \mathbb{N}$ is the number of time-steps, $v = (v^{(1)}, \ldots, v^{(k)})$ where $v^{(j)} \in [0, \infty)$ is the value of resource $j \in [k]$, $c = (c^{(1)}, \ldots, c^{(k)})$ where $c^{(j)} \in [0, \infty)$ is the cost of resource $j \in [k]$, $B^D \in [0, \infty)$ is the defender's budget, and $B^A \in [0, \infty)$ is the attacker's budget.

Defender and attacker strategies are represented by bitstrings of length $n := k \cdot \ell$. We adopt the notation

$$x = (x_1^{(1)}, \ldots, x_\ell^{(1)}, \ldots, x_1^{(k)}, \ldots, x_\ell^{(k)}) \in \{0,1\}^n$$

for the defender's strategy, where $x_i^{(j)} = 1$ for $j \in [k]$ and $i \in [\ell]$ means that the defender attempts to acquire resource j at time i.

$$y = (y_1^{(1)}, \ldots, y_\ell^{(1)}, \ldots, y_1^{(k)}, \ldots, y_\ell^{(k)}) \in \{0,1\}^n$$

for the attacker's strategy, where $y_i^{(j)} = 1$ for $j \in [k]$ and $i \in [\ell]$ means that the attacker attempts to acquire resource j at time i.

We will define the payoff of strategies in terms of the ownership of the resources. In particular, $z_i^{(j)} \in \{0,1\}$ is the ownership of resource $j \in [k]$ at time $i \in \{0\} \cup [\ell]$, where $z_i^{(j)} = 1$ indicates that the defender owns resource j at time i, and $z_i^{(j)} = 0$ means that the attacker owns resource j at time i. For all $j \in [k]$, we define $z_0^{(j)}(x, y) := 1$, which corresponds to the assumption that the defender is in possession of all resources at the beginning of the game.

Table 1: Ownership outcomes and color codings for the 4 combinations of moves on a single resource in DefendIt (and FlipIt).

Defender/Attacker	0	1
0	Previous owner (Unchanged)	Attacker owns
1	Defender owns	Previous owner

The evolution of the ownership of a resource j is defined inductively for $i \in [\ell]$ as follows

$$z_i^{(j)}(x, y) := \begin{cases} z_{i-1}^{(j)}(x, y) & \text{if } x_i^{(j)} = y_i^{(j)}, \\ 1 & \text{if } x_i^{(j)} = 1 \text{ and } y_i^{(j)} = 0, \\ 0 & \text{if } x_i^{(j)} = 0 \text{ and } y_i^{(j)} = 1. \end{cases} \tag{1}$$

Table 1 describes ownership outcomes for the move combinations in DefendIt. Because these combinations will be color-coded in Section 6.2, the colorings are noted here for reference. Intuitively, this means that if a player attempts to acquire the resource while the opponent does not, the player obtains the resource. If neither the defender or attacker move, the ownership does not change. If both the defender and the attacker attempt to acquire the resource, the defender gets or retains ownership.

The overall cost of a defender or an attacker strategy x is

$$C(x) := \sum_{j=1}^{k} c^{(j)} \sum_{i=1}^{\ell} x_i^{(j)},$$

i.e., the number of attempts to acquire a resource weighted by the cost of that resource. A defender strategy x is called *over-budget* if $C(x) > B^D$. Similarly, an attacker strategy y is called *over-budget* if $C(y) > B^A$.

Finally, the payoff function g_1 for the defender (respectively g_2 for the attacker) are defined as

$$g_1(x, y) := \begin{cases} \sum_{j=1}^{k} v^{(j)} \sum_{i=1}^{\ell} z_i^{(j)}(x, y) & \text{if } C(x) \leq B^D \\ -\sum_{j=1}^{k} \sum_{i=1}^{\ell} x_i^{(j)} & \text{otherwise} \end{cases}$$

$$g_2(x, y) := \begin{cases} \sum_{j=1}^{k} v^{(j)} (\ell - \sum_{i=1}^{\ell} z_i^{(j)}(x, y)) & \text{if } C(y) \leq B^A \\ -\sum_{j=1}^{k} \sum_{i=1}^{\ell} y_i^{(j)} & \text{otherwise.} \end{cases}$$

Informally, if the overall cost of a strategy exceeds the player's budget (over-budget strategy), the payoff is minus the number of times the player attempts to acquire any resource. Note that the payoff function for an over-budget defender strategy is independent of the attacker strategy, and isomorphic to the OneMax problem [6], and similarly for over-budget attacker strategies. If the overall cost of a strategy is within the budget (within-budget strategy), then the payoff is the number of time steps the player is in possession of the resource multiplied by the value of the resource.

We now proceed to analyze the hardness of finding an optimal player strategy.

4 DEFENDIT HARDNESS

DefendIt has a non-trivial problem structure. The following theorem implies that, assuming P≠NP, it is computationally intractable to find optimal (with respect to a fixed opponent) strategies to DefendIt. The proof is in Appendix D.

Theorem 4.1. *The decision problem to determine whether a defender (or attacker) strategy of DefendIt can achieve a given payoff value for a fixed opponent is NP-complete.*

This paper only consider DefendIt instances where the cost of an item is identical to the value of the resource. In the context of Theorem 4.1, this corresponds to the NP-hard subset sum problem which is a special case of the knapsack problem. Hence, the decision variant of DefendIt is still NP-complete for our choice of costs.

Note that if the cost values satisfies $c^{(j)} \sim \text{Gamma}(\alpha, \theta)$, i.e., independent gamma-distributed random variables with shape parameter α and scale parameter θ, then the expected cost of a uniformly sampled search point x is

$$E\left[C(x)\right] = \sum_{j=1}^{k} \sum_{i=1}^{\ell} E\left[c^{(j)}\right] E\left[x_i^{(j)}\right] = n\alpha\theta/2. \tag{2}$$

In the special case of scale parameter $\alpha = 1$, the cost values are exponentially distributed with rate $1/\theta$. The cumulative distribution function in this special case is therefore for $x > 0$

$$\Pr\left(c^{(j)} \leq x\right) = 1 - e^{-x/\theta}. \tag{3}$$

Note furthermore that for all $c > 0$, if $X \sim \text{Gamma}(\alpha, \theta)$, then $cX \sim \text{Gamma}(\alpha, c\theta)$. Furthermore, for k independent random variables. $X_j \sim \text{Gamma}(\alpha_j, \theta)$, it holds $\sum_{j=1}^{k} X_j \sim \text{Gamma}(\sum_{j=1}^{k} \alpha_j, \theta)$ Hence, if the cost values are distributed $c^{(i)} \sim \text{Gamma}(\alpha, \theta)$, then the cost of any given a fixed strategy x with $|x^{(j)}| = r$ 1-bits for each $j \in [k]$ has distribution

$$C(x) \sim \text{Gamma}(k\alpha, \theta r). \tag{4}$$

The same holds for defenders.

5 EXPERIMENTAL METHODOLOGY

Due to the NP-hardness of the relative payoff functions (cf Theorem 4.1), it is intractable to compare how close PDCoEA strategies are to the optimum. Also, it may be meaningless to compare strategies with randomly chosen strategies. This dilemma also occurs in more complex versions of coevolutionary algorithms.

We therefore develop a methodology where the performance of an algorithm $\mathcal{A}$ is compared relative to one or more reference algorithms $\mathcal{B}$. We run $\mathcal{A}$ and all the algorithms of $\mathcal{B}$ independently for the same number of function evaluations and collect "champion" defenders and attackers at regular time intervals, specified by a period length τ. We then evaluate the individuals in the population of an algorithm at a time t against the champions collected from both algorithms up until time t.

We now describe this methodology formally. Assume that after generation t, algorithms $\mathcal{A}$ and $\mathcal{B}$ have predator-prey populations $(P_t^{\mathcal{A}}, Q_t^{\mathcal{A}}) \in \mathcal{X}^{\lambda^{\mathcal{A}}} \times \mathcal{Y}^{\lambda^{\mathcal{A}}}$, respectively $(P_t^{\mathcal{B}}, Q_t^{\mathcal{B}}) \in \mathcal{X}^{\lambda^{\mathcal{B}}} \times \mathcal{Y}^{\lambda^{\mathcal{B}}}$, where $\lambda^{\mathcal{A}}$ and $\lambda^{\mathcal{B}}$ refer to the population sizes of the algorithms.

Since the two algorithms may have different population sizes (i.e., $\lambda^{\mathcal{A}} \neq \lambda^{\mathcal{B}}$), and it may be impossible to obtain individuals halfway through a generation, we collect the i-th period champions from algorithm $\mathcal{A}$ in generation

$$T_i^{\mathcal{A}} := \min\{i \in \mathbb{N} \mid \mathcal{A} \text{ has made at least } i\tau \text{ fevals after gen. } i\}$$

and the i-th period champions from algorithm $\mathcal{B}$ in generation

$$T_i^{\mathcal{B}} := \min\{i \in \mathbb{N} \mid \mathcal{B} \text{ has made at least } i\tau \text{ fevals after gen. } i\}$$

Recall that τ refers to the period length, which is a parameter of the method. We use the abbreviation *fevals* for function evaluations.

The defender champions in the i-th period are

$$u_i^{\mathcal{A}} := \arg\max_{x \in P^{\mathcal{A}}_{T_i^{\mathcal{A}}}} \min_{y \in Q^{\mathcal{B}}_{T_i^{\mathcal{B}}}} g_1(x, y), \text{ and } u_i^{\mathcal{B}} := \arg\max_{x \in P^{\mathcal{B}}_{T_i^{\mathcal{B}}}} \min_{y \in Q^{\mathcal{A}}_{T_i^{\mathcal{A}}}} g_1(x, y).$$

Similarly, the attacker champions in the i-th period are

$$v_i^{\mathcal{A}} := \arg\max_{y \in Q^{\mathcal{A}}_{T_i^{\mathcal{A}}}} \min_{x \in P^{\mathcal{B}}_{T_i^{\mathcal{B}}}} g_2(x, y), \text{ and } v_i^{\mathcal{B}} := \arg\max_{y \in Q^{\mathcal{B}}_{T_i^{\mathcal{B}}}} \min_{x \in P^{\mathcal{A}}_{T_i^{\mathcal{A}}}} g_2(x, y).$$

The combined defender champions U_i and attacker champions V_i after i periods are

$$U_i := \cup_{t=0}^{i} \left\{u_t^{\mathcal{A}}, u_t^{\mathcal{B}}\right\} \text{ and } V_i := \cup_{t=0}^{i} \left\{v_t^{\mathcal{A}}, v_t^{\mathcal{B}}\right\}$$

Finally, the defender-performance $D_i^{\mathcal{A}}$ of algorithm $\mathcal{A}$ relative to algorithm $\mathcal{B}$ in the i-th period (respectively the defender-performance $D_i^{\mathcal{B}}$ of algorithm $\mathcal{B}$ relative to algorithm $\mathcal{A}$) is now defined as

$$D_i^{\mathcal{A}} := \max_{x \in P^{\mathcal{A}}_{T_i^{\mathcal{A}}}} \min_{y \in V_i} g_1(x, y) \text{ and } D_i^{\mathcal{B}} := \max_{x \in P^{\mathcal{B}}_{T_i^{\mathcal{B}}}} \min_{y \in V_i} g_1(x, y).$$

Similarly, the respective attacker performances of the two algorithms in period i are defined as

$$A_i^{\mathcal{A}} := \max_{y \in Q^{\mathcal{A}}_{T_i^{\mathcal{A}}}} \min_{x \in U_i} g_2(x, y) \text{ and } A_i^{\mathcal{B}} := \max_{y \in Q^{\mathcal{B}}_{T_i^{\mathcal{B}}}} \min_{x \in U_i} g_2(x, y).$$

6 EXPERIMENTS

There are two algorithmic parameters we vary to study their impact: population size λ and mutation rate χ. We also choose nominal values of $\lambda = 500$ and mutation rate $\chi = 0.3$ when we study other factors, unless noted otherwise. Taking into account condition (G2b) of Theorem 3 in [16], we deemed this population size to be sufficiently large, although we do not have a formal proof that condition (G2) is satisfied. This mutation rate is below the *error threshold* for PDCoEA of approximately $\ln(2)$ identified in Theorem 15 in [16].

We set the number of resources to $k = 10$ after some experimentation that considered the complexity of the resulting dynamics (other problem sizes show no significant change, see Figure 10 in the supplementary material). We set the time steps in a DefendIt game to $\ell = 40$. Hence, the strategies are represented by bitstrings of length $n = k\ell = 400$. The number of games (fevals) in a run is $2 \cdot 10^6$, unless noted otherwise. This was determined by experimental observation that payoffs were relatively steady for a long duration. We varied resource costs (with ownership payoff being equal to resource cost) and decided upon a way to hold them steady across all experiments presented in the main paper. The cost of resource $i \in [40]$ is X_i where X_i is an independently sampled gamma-distributed random variable with shape parameter $\alpha = 1$, and scale parameter $\theta = 200$. The resource values and cost are identical.

We use the champions methodology of Section 5 unless otherwise noted, setting the period length to $\tau = 10,000$. Our champions have the best-worst case (minmax) payoff, unless otherwise noted. We repeat each experiment at least 50 times aiming for 100 unless computationally they are too costly. We use Kruskal-Wallis when we test for statistically significant differences (p-value < 0.01).

We study one game parameter, the budgets for the attacker and defender. We also explore budgets' interaction with mutation rates. When budgets are not integral to the study question, we make them equal and specify their values from the set $\{B_{\text{low}} := 280, B_{\text{med}} := 2,800, B_{\text{high}} := 28,000\}$

6.1 Population Size

6.1.1 Is a population better than evolving a single Attacker - Defender pair? We start by comparing PDCoEA to the two single individual algorithms described in Section 3.1, *Pairwise Dominance (1+1) EA* and *(1,λ) CoEA (all-vs-all worst)* using the champions methodology of Section 5 with period $\tau = 10,000$.

$\mathcal{A}$ is either PD (1+1) EA or (1,λ) CoEA and the PDCoEA is the lone reference algorithm $\mathcal{B}$. For fair comparison, both the PDCoEA and the (1,λ) CoEA use mutation rate $\chi = 3/10$ and population size $\lambda = 500$. For the PD (1+1) EA, we use mutation rate $\chi = 1$ which is standard for the (1+1) EA.

The results are shown in Figures 2a and 2b. The figures show boxplots of the defender payoff $D_i^{\mathcal{A}}$ of algorithm $\mathcal{A}$ (red) over 100 independent runs, and the maxmin (best worst case) defender payoff $D_i^{\mathcal{B}}$ of the reference algorithm PDCoEA (in blue). Plots are truncated at 500,000 function evaluations since the payoffs do not change much after this. Attacker payoffs show qualitatively the same behavior so they are not plotted. In Figure 2b PD (1+1) EA starts with significantly better payoff than PDCoEA (0 - 20,000 fevals), then (30,000 - 50,000) there is no significant differences, then PDCoEA is significantly better (50,000 - 500,000 fevals). (1,λ) CoEA is slightly different in Figure 2b, first (0 - 20,000 fevals) there is no significant difference then PDCoEA is significantly better. In addition, the (1,λ) CoEA takes more fevals to reduce the variance of the best individual.

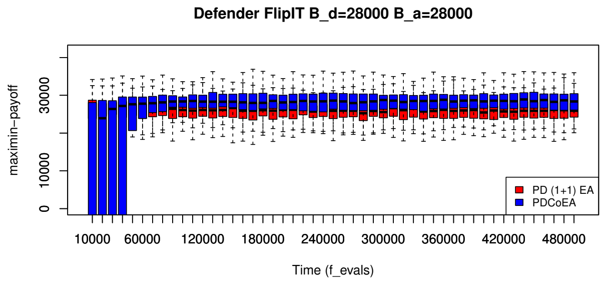
To summarize, after 50,000 fitness evaluations, a population-based CoEA generates statistically higher payoffs for a defender.

6.1.2 Does Incrementally Increasing Population Size Result in Higher Payoffs? We ask whether a slightly larger population is better. This question can be posed at each time interval τ.

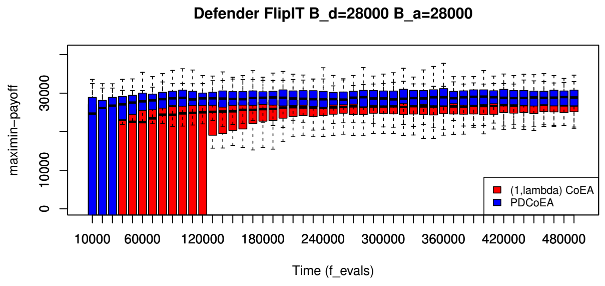
For Algorithm $\mathcal{A}$ we chose PDCoEA with population size $\lambda \in \{1, 2, \ldots, 100\}$. As reference algorithm $\mathcal{B}$, we chose the PDCoEA with population size $\lambda = 300$. Both algorithms used mutation rate $\chi = 0.3$. The plots in Figure 3 show $D_i^{\mathcal{A}}$ (left) and $A_i^{\mathcal{A}}$ as a function of population size (λ) and fitness evaluations (time).

We first visually observe that larger population sizes generate higher payoff values. We then statistically test the distribution of minmax payoff values when the population size (λ) is increased ($\Delta\lambda$). We find that 72% of defenders and 77% of attackers significantly improve their payoff when the increase is one ($\Delta\lambda = 1$). In addition, 95% of defenders and 87% of attackers significantly improve when the increase is two ($\Delta\lambda = 2$).

We can also ask whether a larger population leads to higher payoffs at the end of a run. Statistically, we find that the minmax payoff does not change significantly after the payoff has stabilized, i.e., (the payoffs exhibit no significant difference). The defenders take an average of $7.63 \cdot 10^6$ fevals to stabilize, and attackers take an average of $5.616 \cdot 10^6$ fevals to get payoffs to stabilize.



(a) PDCoEA and PD(1+1) EA.



(b) PDCoEA and PD (1,λ) EA

Figure 2: Comparing algorithms. X-axes show the number of fitness evaluations and y-axes show the best worst case (minmax) defender payoff at different times.

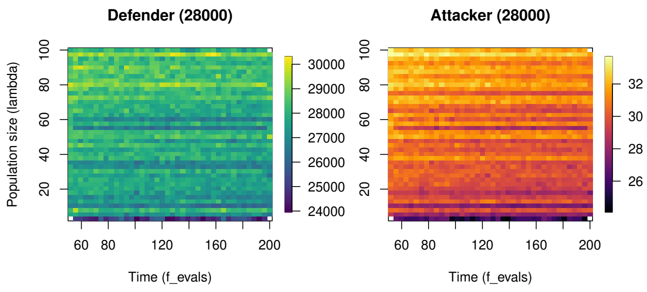
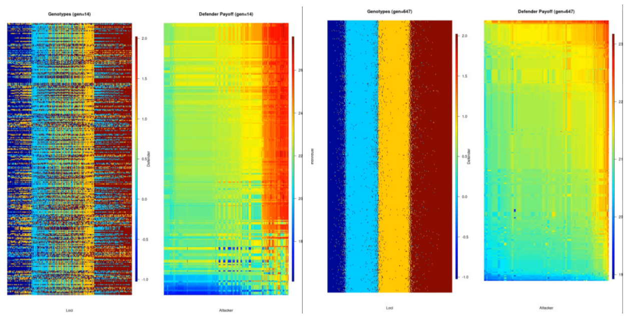


Figure 3: population size (Y-axis) and fevals (X-axis) , the color shows the payoff of the mean of the best worst case (maxmin) payoff against a PDCoEA ($\lambda = 300$). Note, the first 500,000 fevals are removed to improve the color range.

To summarize, at modest populations sizes, small increments result in higher payoffs at each time interval but they do not lead to higher payoffs at the end of a run.

6.2 PDCoEA Single Run Dynamics

We now investigate what happens to the PDCoEA population over a run in terms of strategies and payoffs. Drawing intuitions from payoffs of a CoEA run can be misleading because 1) payoffs are relative to a set of competitors (at the current time point/generation



(a) Gen 14 (4 200 fevals) (b) Gen 647 (194 100 fevals)

Figure 4: Left of each subfigure: each row is the move combinations arising from a pair of attacker's and defender's strategies. Combinations are color coded, blue: D=0, A=0;cyan: D=1, A=0;orange: D=0 A=1;red: D=1 A=1. All pairs of defenders versus attackers in a generation are shown. Right pane is the same as Figure 5. Figure 4a shows an early generation and Figure 4b a later generation when median payoffs have stabilized.

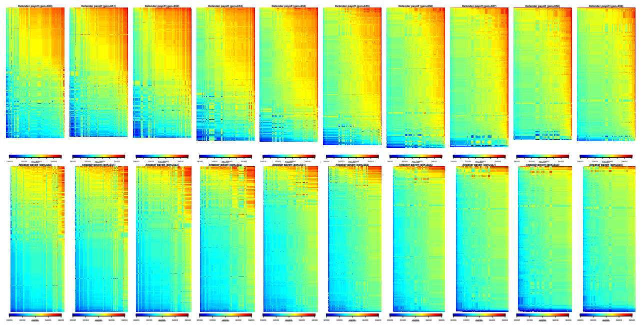


Figure 5: Population dynamics illustrated by payoff matrices for defender population vs attacker populations over 10 generations. Only within-budget defenders are shown. The top row shows heatmaps (payoff matrix) for the Defender payoff and bottom row shows the Attacker payoff. Each column is a generation, 650-659 (fevals: 195, 000 - 197, 700). Over-Budget payoffs are shown as white. A heatmap shows the payoff value (color) for each attacker against each defender in each cell. The individuals in the population are ranked according to average payoff. The defender population is on the y-axis and the attacker population is on the x-axis in each heatmap.

of the algorithm) and the same strategy's payoff can change from one time point to the next (generation to the next) as its competitors evolve, or 2) payoffs can look steady but the underlying strategies that garner the payoffs may have changed.

One way to eliminate misconceptions is to visualize the strategies and their evolution over time while simultaneously visualizing payoffs over time. We do this in Figure 4[1] which shows two snapshots of the population Figure 4a at Gen 14 after 4, 200 fevals and Figure 4b at Gen 647 after 194, 100 fevals. The budget is $B^D_{\text{high}} = B^A_{\text{high}}$ for this particular run. In each snapshot the visualization on the left, our "move plot" color codes the move combinations for every resource in a loci of the bit string, for a pair of defender and attacker, along the x-axis. See Table 1 for the color move coding. Along the y-axis it shows each move combination pair in the generation (from Line 5 in Algorithm 1). (For more informative visualization, and without loss of accuracy, we arbitrarily group engagements and then sort the groups by color.) In each snapshot the heatmap on the right shows the same run's payoffs for "within-budget" defenders (y-axis) in population vs the attacker (x-axis) population in that generation.

We can first examine the coupling of the move plot and heat map. At Gen 14 we observe a lot of variation in move combinations from high to low (i.e., across the two populations) and we discern 4 fuzzy vertical columns of color (blue and light blue on left, and orange and red on right) with a 5th column of mixed colors in the middle. The genotype's loci are ordered by resources with decreasing cost (and it sequences moves for all time steps, for each resource). These columns correspond to selection of resources of different payoffs (since cost and payoff are equal in our game settings). If we compare columns across the populations (i.e. moving up the y-axis), we observe that the move combinations of two competing strategies don't consistently protect or try to claim ownership of the same resource. Concurrently, we can factor in the heat map of payoffs for the defender. We note that there are more extreme payoffs with a lower horizontal section that is blue (lower payoffs) and a vertical right-side section that is red. The variation we see in move combinations is complemented by variation in payoffs.

[1]Videos of snapshots with two runs are shown in the supplementary material.

When we examine Gen 647's move plots we observe very cleanly separated columns of the 4 distinct move combinations and the 5th column of Gen 14 does not appear. It appears the move combinations have differentiated the different payoff values of the resources. They are not competing for the highest cost resources, they are iterating control of the mid-cost resources and competing head to head for the lowest cost ones. This organization can be juxtaposed with the defender's payoff heat map that shows a large number of middle valued game payoffs and relatively fewer extreme (lower or upper) payoffs. The range of payoffs seen in Gen 14 has shrunk.

The insight value of these comparisons, whether within a pair or across generations is limited because they are snapshots. Obviously, single generations snapshots can be assembled into animations. Because animations cannot be inserted into a document, we provide a number of these animations in our supplemental material where we vary budgets for the players. At the time of submission, a clear analysis of the animation defies the authors. We (wryly) observe that when we add complexity to an analysis, with coevolutionary algorithms, clear analysis can quickly decrease.

We can however pursue a facet of the more complex animations. Figure 5 shows the population dynamics of one experiment for generations 650 to 659 by plotting the payoff values of the within-budget individuals for attackers (top) and defenders (bottom). We chose these generations because at this point, the average population fitness was stabilized. All PDCoEA runs stabilize so these plots are representative (of the stable phase).

We can observe the payoff transition of each individual heatmap. The color gradient is quite consistent with blue in the bottom and left changing to green to yellow to red in the top and right. Viewing strictly by row (column), we also see monotonic payoff increase.

Next, we can look along the columns of the figure to compare pairs of defender and attacker heatmaps at the same generation. They look quite symmetrical in color distribution, even though the values are different (the visualizations use different scales).

Finally, we can examine the payoffs over time, i.e., dynamics. We see that for both rows, the first column has a more uniform distribution of colors (values) compared to the last column (10 generations

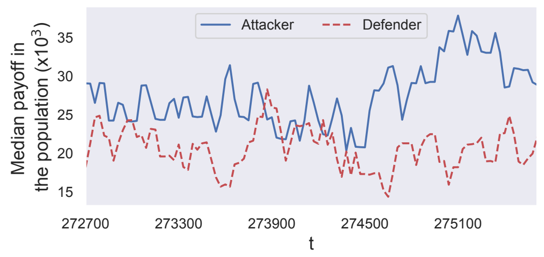


Figure 6: Fluctuations of payoff in the populations (attacker solid line, defender dashed line) shown for median attacker payoff (left y-axis) and defender (right y-axis) for fitness evaluations $272,700 - 276,600$ for one experiment.

later). We see that this changes gradually to be less uniform for each generation. This means that the diversity of payoffs in the population is decreasing. At Gen 650 there are fewer strong attackers and defenders in the population compared to at Gen 659. There are more weak defenders and weak attackers in Gen 650 compared to Gen 659. Note also that the change in height/width of the heatmap means that the number of over-budget individuals is fluctuating.

The population dynamics can also be observed at a larger time scale (more fitness evaluations). Figure 6 illustrates the coupling and fluctuations of the population payoffs by showing the median attacker and defender payoff over time (fitness evaluations $272,700 - 276,600$) for one experiment.

To this point, we have not varied any parameters of the game or the algorithm beyond our population size exploration. To explore the impact of problem scale, we explored the problem parameters $k = \ell = \sqrt{n}$ for $n \in \{40, 80, 120, \dots, 2,000\}$. Results are not shown as problem size has no significant impact on run dynamics.

6.3 Impact of Budget Levels

To experimentally study the impact of budget levels we first need to minimize confounding effects of the relationship between problem size and budget, making the budgets independent of problem size. To do this we set the cost of each resource to be distributed by $c^{(j)} \sim \text{Gamma}(\alpha(k), \theta(\ell))$ where $\alpha(k) := 10/k$ and $\theta(\ell) := 8,000/\ell$. Note that for a problem size $n = k \cdot \ell$, by (Eq. 4), the cost of any fixed strategy $x = (x^{(1)}, \dots, x^{(k)})$ with $|x^{(j)}| = \ell/2$ for each $j \in [k]$ (corresponding to the expected number of 1-bits in a uniformly sampled strategy) has distribution $C(x) \sim \text{Gamma}(k\alpha(k), \theta(\ell)\ell/2) = \text{Gamma}(10, 4000)$, i.e., the distribution is independent of the problem size n. In particular, for this setting, we get for the fixed bitstring x above $\Pr\left(C(x) \leq B_{\text{high}}\right) \approx 0.169504$ and $\Pr\left(C(x) \leq B_{\text{med}}\right) \approx 4.1267 \cdot 10^{-9}$.

Appropriate population sizes for PDCoEA are currently not well understood. We rely on advice from results for non-elitist evolutionary algorithm applied to pseudo-Boolean optimization, where the population size must be chosen such that $\lambda \geq c \ln(n)$ (see condition (G3) of Theorem 1 in [5]). Here, we set population size $\lambda := 50 \ln(n)$. All experiments were run for $2 \cdot 10^6$ function evaluations.

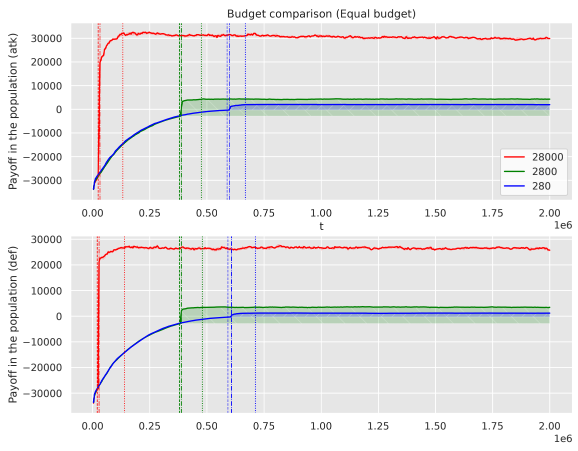


Figure 7: Median population payoff (y-axis) with shaded quartiles for different budgets (Lines) over time (x-axis) for attacker (top) and defender (bottom). There are three budget levels for equal budgets. The vertical lines indicate the population phase composition borders.

6.3.1 *Impact Findings.*

Our findings are shown in Figure 7. Each plot line shows a defender's median population member's payoff for a different pairing of equal budget levels. (Recall that all the figures show the average of 100 runs.) Across budgets, we observe a clear difference between the quantity of fitness evaluations it takes to reach different payoff levels. Zooming in, we can also ascertain payoff phases which the figure marks with vertical lines. These phases are: `all-over-budget`, $\{x \in X | \forall C(x) > B\}$, `one-within-budget` (there exists at least one individual within-budget), $\{x \in X | \exists C(x) < B\}$, `majority-within-budget` (the median individual is within-budget), $C(\text{med}\{X\}) < B$, and `payoff-stalemate` (the median payoff slope is negative).

The vertical lines in Figure 7 show these phase boundaries. This makes it easier to discern that a higher budget allows the median population member to use fewer fitness evaluations to reach the `one-within-budget` payoff phase, as well as reach the `majority-within-budget` and `payoff-stalemate` phases. In addition, the fitness evaluations usages are similar between attacker and defender. Note that the shaded portions of the plot lines for B^D_{low} and B^D_{med} indicate that there are over-budget solutions in the 3rd quartile of the population.

Figure 8 compares the mean population payoff for attacker and defender over time with B_{low} for attackers and 3 levels of budgets for defenders. With a higher defender budget it takes fewer fitness evaluations to reach a population which has at least one within-budget individual, as well as to reach a payoff stalemate for both attacker and defender. The shaded portions of the line plots for B^D_{med} indicate that there are over-budget solutions in the 3rd quartile of the population up to $\approx 50,000$ fitness evaluations.

Having explored budgetary impacts, we next consider the impact of mutation rates on payoffs while also varying budget levels.

6.4 Mutation Rates and the Error Threshold

Figure 9 shows $D^{\mathcal{A}}_i$ (defender maxmin payoff) and $A^{\mathcal{A}}_i$ (attacker maxmin payoff) as a function of time (fevals), mutation rate χ, and different combinations of defender and attacker budgets B^D

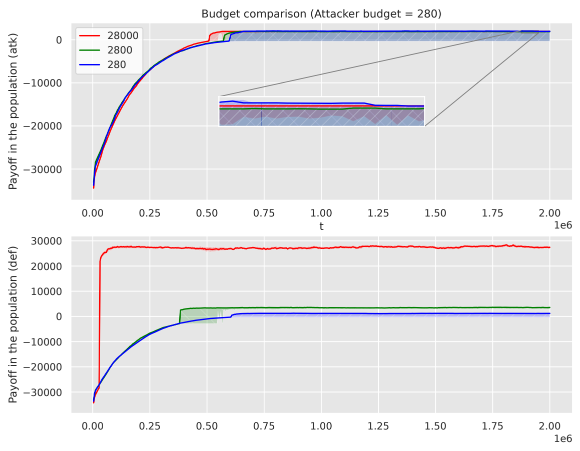
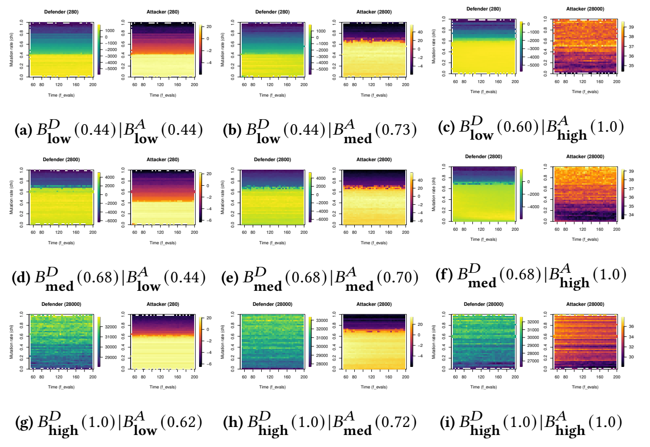


Figure 8: B^D_* vs B^A_{low}, Median population payoff (y-axis) with shaded quartiles for different budgets (Lines) over time (x-axis) for attacker (top) and defender (bottom). There are three budget levels B_{low} (280), B_{med} (2,800) and B_{high} (28,000), the budget combinations are the same as in Figure 9.



(a) $B^D_{\text{low}}(0.44)|B^A_{\text{low}}(0.44)$ (b) $B^D_{\text{low}}(0.44)|B^A_{\text{med}}(0.73)$ (c) $B^D_{\text{low}}(0.60)|B^A_{\text{high}}(1.0)$

(d) $B^D_{\text{med}}(0.68)|B^A_{\text{low}}(0.44)$ (e) $B^D_{\text{med}}(0.68)|B^A_{\text{med}}(0.70)$ (f) $B^D_{\text{med}}(0.68)|B^A_{\text{high}}(1.0)$

(g) $B^D_{\text{high}}(1.0)|B^A_{\text{low}}(0.62)$ (h) $B^D_{\text{high}}(1.0)|B^A_{\text{med}}(0.72)$ (i) $B^D_{\text{high}}(1.0)|B^A_{\text{high}}(1.0)$

Figure 9: Payoff for different error thresholds and budget sizes over time. A heatmap has error threshold (Mutation rate χ) on the y-axis and time (fitness evaluations) on the x-axis, the cell shows the average payoff for a population. There is a heatmap for the attacker and the defender each with the same or different budget. There are three budget levels B_{low}, B_{med} and B_{high} to illustrate an arms-race and different adversary capabilities. The defender budget increases from top to bottom and the attacker budget increases from left to right. The subcaption (value) χ^A_F indicates the investigated mutation rate to have a majority of within-budget attackers, and χ^D_F defenders. Larger figure in Appendix Figure 13.

and B^A. The subcaptions in Figure 9 show χ^A_F thresholds for the mutation rate to have a majority of within-budget attackers, and χ^D_F defenders.

The error threshold, as identified by the mutation rate (χ) is affected by the player budget. Note, the change in payoff for different χ is mostly significant. An increase in player budget changes the error threshold when the adversary has a fixed budget, e.g., $\chi^D_F \leq 0.44, B_{\text{low}}$ for defender (Figure 9a), to $\chi^D_F \leq 0.68, B_{\text{med}}$ for defender (Figure 9d). The error threshold is also impacted by a change in the budget of the adversary, e.g., $\chi^D_F \leq 0.44, B_{\text{low}}$ for defender and $\chi^A_F \leq 0.73, B_{\text{med}}$ for attacker (Figure 9b), to $\chi^A_F \leq 1.0, B_{\text{high}}$ for attacker (Figure 9f). In addition, we observe that error threshold changes from B_{low} to B_{med} the error threshold increases (low values of χ have high payoff), but from B_{med} to B_{high} higher values of χ have higher payoff than lower. Note, that with a higher mutation rate $\chi > 1.0$ for B^*_{high} the error threshold will be reached.

One can interpret the plots as indicating that change in budget is a proxy for an arms-race, i.e., equilibrium search with increasing budgets. The arms-race approximation provides information regarding the setting of mutation rates χ for the PDCoEA algorithm. By following column/row we see how the payoff changes based on adversary budget increase, and static player budget. For example, the top-left to bottom-right diagonal (Figures 9a, 9e, and 9i) show how the payoff changes in an arms-race with equal budget.

A possible explanation for the error threshold for different budgets is that it depends on the number of within-budget individuals. With B_{low} most bits will need to be zero in order for the individual to be within-budget. This bias towards zeros implies that within-budget individuals are confined to a narrow region of the search space. Individuals can only approach such narrow regions when the mutation rate is sufficiently low (cf. parameter $b(n)$ in Theorem 4 in [15]). When the budget is increased to B_{high} then the bias towards zeros is removed and individuals can enter a larger region of the search space, allowing for a higher mutation rate.

7 CONCLUSION

We introduced the NP-hard problem DefendIt and played with a coevolutionary algorithm named PDCoEA. DefendIt is more complex than Bilinear [16], i.e., a simple representation that has high individual solution diversity, non-trivial payoff and dynamics. Despite the simplicity of its genome (a *loci* per resource per time step), and just 4 moves and payoff rules, DefendIt is complicated, making it a good choice to study. We showed that a population-based coevolutionary algorithm, rather than single-individual EA, attains higher payoffs.

We developed a "moves plot" to see the move combinations of a game step, and paired it with payoff heat maps. This allowed us to notice more details about a single run at a snapshot in time. The supplementary material includes an animation. We also varied the adversaries' budgets, exploring equal levels and asymmetric levels. Finally, we explored the mutation rate's impact on error threshold. The population dynamics of PDCoEA on DefendIt show they do not converge on a solution, but the payoffs are coupled and fluctuate. We find that there are multiple distinct phases in the population payoff. In addition, the mutation rate should be altered for better payoff when the individual and adversaries' budget change.

Future work will investigate a self-adapting mutation rate to see if it evolves to the error threshold. We will study the solutions more quantitatively, e.g., measure coevolutionary pathologies. Finally, we will investigate more problem parameters, e.g., payoff function and different resource distributions.

ACKNOWLEDGMENTS

Lehre and Hevia Fajardo were supported by a Turing AI Fellowship (EPSRC grant ref EP/V025562/1). The computations were performed using the University of Birmingham's BlueBEAR HPC service. See http://www.birmingham.ac.uk/bear for more details. Toutouh was supported by the University of Malaga.

REFERENCES

[1] Peter J. Angeline and Jordan B. Pollack. 1993. Competitive environments evolve better solutions for complex tasks. In *Proceedings of the Fifth International Conference (GA93), Genetic Algorithms*. 264–270.

[2] L. M. Antonio and C. A. C. Coello. 2018. Coevolutionary Multi-objective Evolutionary Algorithms: A Survey of the State-of-the-Art. *IEEE Transactions on Evolutionary Computation* (2018), 1–16. https://doi.org/10.1109/TEVC.2017.2767023

[3] Robert Axelrod. 1984. *The Evolution of Cooperation.* Basic, NY, New York.

[4] A. B. Cardona, J. Togelius, and M. J. Nelson. 2013. Competitive coevolution in Ms. Pac-Man. In *2013 IEEE Congress on Evolutionary Computation.* 1403–1410.

[5] Dogan Corus, Duc-Cuong Dang, Anton V. Eremeev, and Per Kristian Lehre. 2018. Level-Based Analysis of Genetic Algorithms and Other Search Processes. *IEEE Transactions on Evolutionary Computation* 22, 5 (Oct. 2018), 707–719. https://doi.org/10.1109/TEVC.2017.2753538

[6] Stefan Droste, Thomas Jansen, and Ingo Wegener. 2002. On the analysis of the (1+1) evolutionary algorithm. *Theoretical Computer Science* 276, 1-2 (April 2002), 51–81.

[7] Paul R Ehrlich and Peter H Raven. 1964. Butterflies and plants: a study in coevolution. *Evolution* 18, 4 (1964), 586–608.

[8] Sevan Gregory Ficici. 2004. *Solution concepts in coevolutionary algorithms.* Ph. D. Dissertation. Brandeis University.

[9] D Fogel. 2001. Blondie24: Playing at the Edge of Artificial Intelligence.

[10] David E. Goldberg. 1989. *Genetic Algorithms in Search, Optimization and Machine Learning* (1st ed.). Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[11] Sean N Harris and Daniel R Tauritz. 2021. Competitive coevolution for defense and security: Elo-based similar-strength opponent sampling. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion.* 1898–1906.

[12] Erik Hemberg, Jacob Rosen, Geoff Warner, Sanith Wijesinghe, and Una-May O'Reilly. 2016. Detecting tax evasion: a co-evolutionary approach. *Artificial Intelligence and Law* 24 (2016), 149–182.

[13] Stephen T Jones, Alexander V Outkin, Jared Lee Gearhart, Jacob Aaron Hobbs, John Daniel Siirola, Cynthia A Phillips, Stephen Joseph Verzi, Daniel Tauritz, Samuel A Mulder, and Asmeret Bier Naugle. 2015. *Evaluating moving target defense with pladd.* Technical Report. Sandia National Lab.(SNL-NM), Albuquerque, NM (United States).

[14] Krzysztof Krawiec and Malcolm Heywood. 2016. Solving Complex Problems with Coevolutionary Algorithms. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion.* ACM, 687–713.

[15] Per Kristian Lehre. 2010. Negative Drift in Populations. In *Proceedings of the 11th International Conference on Parallel Problem Solving from Nature (PPSN 2010) (LNCS, Vol. 6238).* Springer Berlin / Heidelberg, 244–253. https://doi.org/10.1007/978-3-642-15844-5_25

[16] Per Kristian Lehre. 2022. Runtime Analysis of Competitive Co-Evolutionary Algorithms for Maximin Optimisation of a Bilinear Function. In *Proceedings of the Genetic and Evolutionary Computation Conference* (Boston, Massachusetts) *(GECCO '22).* Association for Computing Machinery, New York, NY, USA, 1408–1416. https://doi.org/10.1145/3512290.3528853

[17] Chong-U Lim, Robin Baumgarten, and Simon Colton. 2010. Evolving behaviour trees for the commercial game DEFCON. In *European Conference on the Applications of Evolutionary Computation.* Springer, 100–110.

[18] Sean Luke et al. 1998. Genetic programming produced competitive soccer softbot teams for robocup97. *Genetic Programming* 1998 (1998), 214–222.

[19] Melanie Mitchell. 2006. Coevolutionary learning with spatially distributed populations. *Computational intelligence: principles and practice* 400 (2006).

[20] Gabriela Ochoa. 2006. Error Thresholds in Genetic Algorithms. *Evolutionary Computation* 14, 2 (June 2006), 157–182. https://doi.org/10.1162/evco.2006.14.2.157

[21] Una-May O'Reilly, Jamal Toutouh, Marcos Pertierra, Daniel Prado Sanchez, Dennis Garcia, Anthony Erb Luogo, Jonathan Kelly, and Erik Hemberg. 2020. Adversarial genetic programming for cyber security: A rising application domain where GP matters. *Genetic Programming and Evolvable Machines* 21 (2020), 219–250.

[22] Elena Popovici, Anthony Bucci, R. Paul Wiegand, and Edwin D. De Jong. 2012. *Coevolutionary Principles.* Springer Berlin Heidelberg, Berlin, Heidelberg, 987–1033.

[23] Christopher D Rosin and Richard K Belew. 1997. New methods for competitive coevolution. *Evolutionary Computation* 5, 1 (1997), 1–29.

[24] Karl Sims. 1994. Evolving 3D morphology and behavior by competition. *Artificial life* 1, 4 (1994), 353–372.

[25] J. Togelius, P. Burrow, and S. M. Lucas. 2007. Multi-population competitive co-evolution of car racing controllers. In *2007 IEEE Congress on Evolutionary Computation.* 4043–4050.

[26] Marten van Dijk, Ari Juels, Alina Oprea, and Ronald L. Rivest. 2013. FlipIt: The Game of "Stealthy Takeover". *Journal of Cryptology* 26, 4 (Oct. 2013), 655–713. https://doi.org/10.1007/s00145-012-9134-5