Attacker-Defender Strategy Optimization Using Multi-objective Competitive Co-evolution

Ritam Guha[1][0000−0002−1375−777X], Ryan Mckendrick[2], Bradley Feest[2], and Kalyanmoy Deb[1][0000−0001−7402−9939]

[1] Michigan State University, East Lansing, MI-48824, U.S.A.
{guharita, kdeb}@msu.com
[2] Northrop Grumman, Falls Church, VA 22042, U.S.A.
{ryan.mckendrick, bradley.feest}@ngc.com

COIN Report 2024001

Abstract. Attacker-defender strategy optimization deals with optimizing and deciding on different tactics used by two independent entities working in tandem. Unlike in standard optimization problems, a complete solution of the entire two-agent problem consists of strategies of both agents and evaluation of a solution requires precise information of both strategies. For this reason, a co-evolutionary optimization framework is proposed in this paper to keep two co-evolving populations interacting with each other in tandem to reach their optimal strategies. While co-evolutionary algorithms have been proposed in the past, multi-objective co-evolutionary problems make the optimization task more complex, resulting in a set of Pareto-optimal strategies for each entity. In this paper, we apply a multi-objective competitive co-evolutionary optimization algorithm to a real-world wargame strategy optimization problem. The proposed co-evolutionary algorithm is used to find trade-off sets of competitive wargame strategies for both entities and a novel post-optimization decision-making procedure is also proposed to choose preferred strategies for each entity in tandem, leading to a stable or a cycle of sequential strategies. To the best of our knowledge, this paper marks one of the first-ever applications of multi-objective, competitive, co-evolutionary optimization approaches to a real-world wargame scenario, revealing their impact and importance in practice.

Keywords: Attacker-defender system · Competitive co-evolution · Decision-making · Multi-objective games · Multi-agent systems.

1 Introduction

Attacker-defender strategy optimization is a critical aspect of many security systems. In wargame scenarios, a win is determined by how each side utilizes its resources and develops effective strategies to counter the opponent's movements. While our discussion sheds some light on the study of these strategies, it should be noted that we do not support or glorify the act of war here. We simply

illustrate the wargame situation as a possible application scenario leading to a challenging multi-agent co-evolutionary optimization problem. War simulations or games [6,12,20] have a large following in the gaming community and it is a very challenging problem to find optimal strategies for each of the participating entities.

In an attacker-defender system, each entity has its own set of parameters related to different resources, such as logistic information, number of assets, etc., which can be tuned to find a response strategy against the opponent. However, the outcome of one entity's parameter setting can only be evaluated properly by knowing the opponent's parameter settings. Thus, any effort to optimize one entity's strategy cannot be developed in isolation, as it is achieved in most single entity-based optimization problems. Since both entities are trying to optimize their own strategies and since they are responsive to one another, there is a need for co-evolving both entities together. While one entity's strategy gets better with optimization iterations, the other entity is also learning to produce better and better strategies of its own. This kind of two intertwined optimization problems result in the co-evolutionary optimization problem. However, if both entities have multiple conflicting objectives to optimize, the resulting problem becomes more challenging as each entity will now have a Pareto set of alternate solutions to pick a strategy from. In recent years, co-evolutionary algorithms have been gaining interest in various applications of multi-agent systems [15,14,23]. There are two types of co-evolutionary process: cooperative [13,3,9,10] and competitive [11,17,16,24]. When two or more species evolve by cooperating with each other in achieving their own goals, the process is called *cooperative* co-evolution. On the other hand, when multiple species evolve by competing against each other to fulfill their individual goals, it is called *competitive* co-evolution. In the case of wargame strategy optimization, generally, two agents will contradict each other's goals, thereby making the task a competitive co-evolution.

Co-evolutionary algorithms are mainly proposed in the literature for a single objective for each agent [21,5]. However, in a practical multi-agent system, such as in a wargame strategy optimization problem, each agent may have more than one conflicting objective to consider. The objectives of an agent can be completely different from those of the other agent or they can have opposing purpose of minimizing or maximizing the objectives. For example, the defense may minimize the loss of its assets, while the offense would, most likely, try to maximize the loss of the defense's assets. The above discussion amply indicates that solution of the attacker-defender system for multiple conflicting objectives is challenging, requiring efficient optimization algorithms and decision-making procedures.

The rest of the paper is organized as follows: Section 2 describes the wargame strategy optimization problem in more detail by outlining the structure of the problem, the objectives, and the complexity of optimization. The proposed multi-objective competitive co-evolutionary framework, including the optimization algorithm and post-optimization decision-making, is described in Section 3. The results obtained by applying the proposed framework to the wargame strategy

optimization problem are analyzed and discussed in Section 4. Finally, the paper is concluded in Section 5. From here onwards, we have used the terms attacker for offense, and defender for defense interchangeably to mean the same thing.

2 wargame Strategy Optimization Problem

Wargame Strategy Optimization Problems (WSOPs) are highly idiosyncratic, and there is not one computing model that can fully represent all types of wargames. Here, we focus on a wargame where there is a clear attacker and a defender. The attacker attempts to inflict as much damage on an Air Base (Air to ground missile hits) while incurring the least cost (monetary expenses and loss of attacker lives) to themselves, but also enforcing the greatest cost on the defender (monetary expenses and loss of defender lives). The defender's goals are in direct competition with the attackers. Both agents' strategies can be broken down into three subsets: research and engineering (RE), force composition (FC), and mission plans (MP). RE and FC decisions have upfront monetary costs and are considered fixed once the mission begins. RE decisions define the capability of assets (e.g., their stealth, speed, sensor ranges, and payload capacities). FC decisions define the number of assets brought to the conflict across roles (e.g., Strike assets and Electronic Warfare assets). MP decisions define how assets coordinate and behave during the conflict (e.g., flight formations, rules of engagement, and routes). Overall, the attacker has 50 variables they can manipulate, and the defender has 24 variables. The attacker variables mainly include striker, sweeper and jammer information, while the defense variables include information about integrated air defense (IAD) systems and interceptors. In this paper, we have attempted to solve the WSOP using a multi-objective competitive co-evolutionary optimization approach.

3 Proposed Multi-objective Co-evolutionary (MoCoEv) Optimization and Decision-Making Methods

In this section, we present the MoCoEv algorithm in detail. Since each entity uses multiple conflicting objectives, MoCoEv is expected to find two distinct sets of Pareto-optimal solutions, each involving all 74 decision variables. Thereafter, a decision-making procedure is needed to choose a sequence of preferred solutions from each Pareto front in a systematic manner.

3.1 MoCoEv Optimization Algorithm

In a MoCoEv algorithm, there are two interacting populations, each containing its own decision variables. Like in other evolutionary algorithms, initial populations $Pop_1^{(0)}$ and $Pop_2^{(0)}$ of two problems can contain random solutions $\mathbf{x}$ and $\mathbf{y}$ of sizes N_1 and N_2, respectively; however, for practical problems with previously known good solutions, initial populations can be seeded with known solutions.

Due to the computational expense in evaluating a strategy ($\mathbf{x}$,$\mathbf{y}$), we use surrogate functions during the optimization process created from a large number of pairs of strategies ($\mathbf{x}$,$\mathbf{y}$) evaluated using hi-fidelity wargame simulation software offline. While the accuracy of the developed surrogate function is a matter of their practical use, we do not address this issue here and focus on the multi-objective co-evolutionary optimization and decision-making aspects of the problem, which is already quite complex and challenging to discuss and comprehend. Here, we want to state our assumption that both entities do not know the decisions made by the other entity but the surrogate models are representative of each entity's assumption about how the other entity might behave. The proposed method can be used to have a better insight into the outcome and dynamics of the wargame based on chosen decision-making aspects, rather than actually using the method in real time.

At iteration t, genetic operations can be performed on $Pop_1^{(t-1)}$ as usual, using its own objectives and constraints and a new population $Pop_1^{(t)}$ of size N_1 can be created. Here, we follow the NSGA-II's operations for this purpose. A total of τ_1 iterations can continue as above before the next population is updated for consecutive τ_2 iterations. Then again Pop_1 can be updated for another τ_1 iterations. This process can continue until a termination condition is satisfied. For simplicity, we ignore constraints in presenting our proposed algorithm. A pseudo-code of the MoCoEv procedure is presented in the supplementary document[3].

Evaluation of a single population member $\mathbf{x}$ of $Pop_1^{(t)}$ requires a specific variable vector $\mathbf{y}$ from the second population Pop_2. This is where a number of strategies can be adopted in an MoCoEv algorithm like summing, averaging, considering minimum, or maximum. But, we consider an averaging strategy here as it is the most used one. The k-th population member of Pop_1, $\mathbf{x}^{(1),k}$, is paired with every member $\mathbf{y}^{(2),l}$ ($l = 1, \ldots, N_2$) of the Population Pop_2 one at a time and N_2 objective vectors are evaluated. Then, a mean fitness value of i-th objective function of the k-th Pop_1 member is computed as follows:

$$F_i(\mathbf{x}^{(1),k}) = \frac{1}{N_2} \sum_{l=1}^{N_2} f_i(\mathbf{x}^{(1),k}, \mathbf{y}^{(2),l}). \tag{1}$$

Similarly, the fitness of each member of the second population can also be computed by averaging the respective objective values over all Pop_1 members.

After the average function values for each objective is computed, they can be used for domination check and other niche-preserving operators of the chosen evolutionary multi-objective (EMO) algorithm. Every member is evaluated for both F_1 and F_2 using all members of the second population. Then, the mean fitness vector is computed for each member and is used for the domination check. The *crowding distance* values for each member, needed for diversity preservation of non-dominated solutions, are also computed using the aggregate fitness values. Thus, the final trade-off set of solutions of each population ($\mathbf{Z}^{(1)}$ and $\mathbf{Z}^{(2)}$)

[3] Google Drive Supplementary Link

will correspond to the non-domination principle of the chosen aggregate fitness functions.

The variation operators in the MoCoEv algorithm are recombination and mutation operators which are applied to the respective population, independently. For example, for every generation of Pop_1, recombination and mutation operators are applied to $\mathbf{x}$-vectors of Pop_1 only, while keeping the current $\mathbf{y}$-vectors of Pop_2 unchanged. Every newly created $\mathbf{x}$-vector is then evaluated using the fixed $\mathbf{y}$-vectors of Pop_2 and average fitness is used for the survival operator of Pop_1. After τ_1 such generations are executed with Pop_1, then τ_2 generations are performed by keeping $\mathbf{x}$-vectors fixed.

The above successive cycles of two population updates are continued until a termination condition is met. In this study, we run until a pre-defined number of combined generations (T) is elapsed. After the MoCoEv run, a separate non-dominated (ND) set will be found for each agent.

3.2 MoCoEv Decision-Making Procedure

Selecting a single Pareto solution for agent from their respective ND set is a challenging decision-making task. We propose the following procedure. Since the wargame is to be played alternatively between the two players, an offline computation of ND strategies becomes a pragmatic approach. In the case of attacker-defender WSOP, the defender (or, attacker) may start by choosing a strategy from its own ND set based on certain initial preference information among its objectives. With the defender's strategy revealed, the attacker can then choose the most preferred strategy from its own ND set, so that maximum effect can be imposed on the defender's chosen strategy. Next, it will be the defender's turn to choose the next appropriate strategy from its own ND set to negotiate the attacker's chosen strategy. These iterative moves can be continued until either an equilibrium state (converged attacker and defender strategies) or an equilibrium cycle (converged cycle of attacker and defender strategies) is arrived.

One of the remaining tasks is to discuss the specific multi-criterion decision-making (MCDM) method where each player can adopt to pick a single appropriate strategy from its own ND set. Let us say that there are N_1 attacking strategies and N_2 defending strategies after the MoCoEv run.

The first step of the decision-making process is to select one of the agents for initiation. Let us say we choose the defender at first. So, the defender needs to choose one of the N_2 final ND strategies. Each of these strategies will have a distribution of objective values as one defending strategy is evaluated against N_1 different attacking strategies from the attacker's ND set, leading to N_1 objective scores. So, we can calculate the normalized standard deviation of all objective values for each defending strategy. The initial defending strategy is the one having the lowest average normalized standard deviation for the objectives. Intuitively, having a lower normalized standard deviation reflects some type of robustness of a strategy against all the opposing strategies.

After the first strategy is selected from the defender, the attacker needs to select its own strategy. Every attacking strategy is evaluated against the selected defending strategy and the corresponding ND set is identified from them. Finally, for selecting a single attacking strategy from the ND set, we use the highest trade-off-based selection method described in [19]. For any ND set, the trade-off value for an ND point x_i with a neighboring ND point x_j can be calculated as:

$$R(x_i, x_j) = \frac{w_{loss} \times Loss_f(x_i \to x_j)}{w_{gain} \times Gain_f(x_i \to x_j)}. \tag{2}$$

Here, w_{loss} and w_{gain} represent the weights provided to loss and gain acquired through moving from one solution to another, respectively. This trade-off value gives an approximation of the amount of loss that should be accepted, compared to the gain for moving from x_i to x_j. The final trade-off value for each point gets computed by taking an average of its trade-off with its neighbors. A higher trade-off value for a point indicates that moving away from the point results in a high loss. Thus, the decision-makers usually prefer the highest trade-off point from the ND set. In this way, we keep selecting the highest trade-off point from the ND set of each side from hereon.

After selecting the attacking strategy, the defender selects a strategy using the same procedure as the attacker. This process continues until we can find an equilibrium point where the same defending strategy and attacking strategy keep on getting selected in each iteration or an equilibrium cycle where the same sequence of defending and attacking strategies get selected in a cycle.

4 Results and Discussion

After carefully formulating the proposed MoCoEv algorithm and decision-making procedure, we apply the proposed framework to the WSOP. In this section, we first show the final formulation of the problem, followed by some preliminary analysis of the optimization of the problem without any co-evolution, and finally, we evaluate the performance of the proposed co-evolutionary process.

4.1 Solution Evaluation and Surrogate Model

In our WSOP case study, we execute each scenario using the 'Command Modern Air and Naval Operations' (CMANO) [2] system. Each scenario takes, on average, about six minutes under a graphics-less accelerated execution. It is still too slow to use this high-fidelity model within an optimization code. Therefore, a surrogate model is learned for each of the five competing attacker/defender objectives.

4.2 WSOP Objectives

The objectives and the goals of the WSOP entities are presented in Table 1. The analysis of the dataset used for training the surrogate models reveals that

Table 1: The original objectives and attacker-defender goals for the WSOP.

Objective	Att.	Def.
OffenseHits	↑	↓
OffenseLifeCost	↓	↑
DefenseLifeCost	↑	↓
OffenseExpenditures	↓	↑
DefenseExpenditures	↑	↓

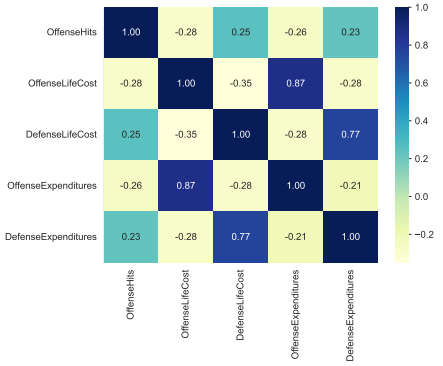


Fig. 1: Correlation analysis of WSOP objectives.

some of the objectives are correlated with each other. In Figure 1, we can see that *OffenseLifeCost* is highly correlated with *OffenseExpenditures* (corr. coeff.: 0.87) and similarly *DefenseLifeCost* is highly correlated with *DefenseExpenditures* (corr. coeff.: 0.77). These observations make sense because intuitively more *OffenseLifeCost* means more resources of the attacker getting destroyed leading to more *OffenseExpenditures*. The same logic applies to the defending entity, as well. For this reason and to make the problem simpler to solve, we eliminate two objectives – *OffenseLifeCost* and *DefenseLifeCost* – from further consideration. This reduces the number of objectives of the problem to three. We further reduce one more objective by constructing two conflicting objectives, as mentioned in Table 2.

Table 2: Reduced objectives for WSOP.

i	Objective (f_i)	Offense Goal	Defense Goal
1	*OffenseExpenditures* - *DefenseExpenditures*	↓	↑
2	*OffenseHits*	↑	↓

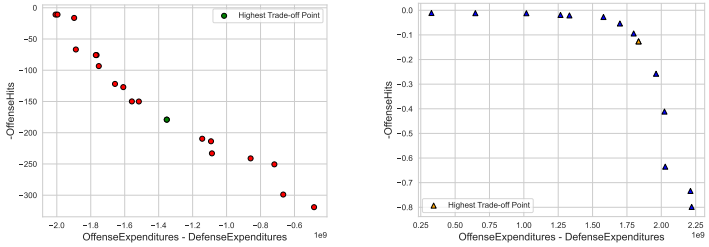
4.3 Multi-objective Optimization without Co-evolution

To have an understanding of the optimal solutions for each entity separately and without having control from the other entity, we optimize the WSOP objectives for each entity without using co-evolution as a multi-objective optimization problem. For the offense entity, the second objective (Table 2) is to be maximized. We convert this objective to formulate a two-objective minimization problem, as follows:

$$\min \{(\mathit{OffenseExpenditures} \text{ - } \mathit{DefenseExpenditures}), -\mathit{OffenseHits}\}\,.$$

Intuitively, they are in conflict, as attempting to cause a large damage to the defense (small value of negative offense hits) will incur a large expenditure for the offense, thereby causing a large first objective value.

We employ the standard NSGA-II procedure [7] without co-evolution and run with 50 population members for 200 generations. The ND front is presented in Figure 2. From the Figure, we can observe that to increase the number of offense hits from 10 to 300, the offense's expenditure needs to be close to the defense's expenditure. In all cases, an independent optimization of the offense's objectives causes the defense to make larger expenditures than the offense and still not protect the defense's assets to a large number.

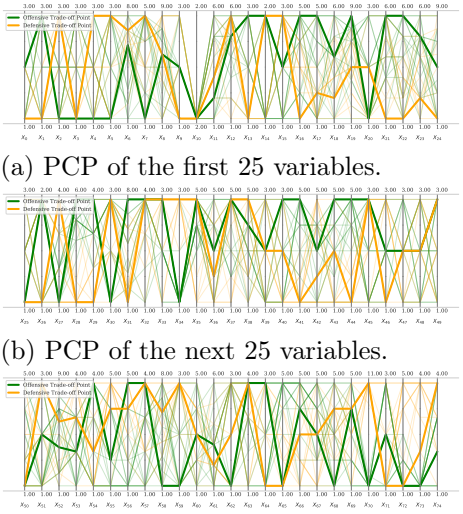


(a) Offense's objectives. (b) Defense's objectives.

Fig. 2: Non-dominated fronts for independently optimizing each entity's objectives without co-evolution.

We repeat the NSGA-II application for the defense entity next. Since the first objective is to be maximized (Table 2) for defense, we use the following formulation: $\min\{-(\mathit{OffenseExpenditures}\text{ - }\mathit{DefenseExpenditures}), \mathit{OffenseHits}\}$.

Figure 2b presents the trade-off ND front. Interestingly, the number of offense hits in the ND front is between zero and 0.8 (average over multiple scenarios), much smaller than that obtained by the offense's independent optimization runs. This is because the objective *OffenseHits* is now being minimized, instead of being maximized. A complete control by defense via an independent optimization of its own objectives cause offense to make more expenditure and still not generate too much damage to the defense assets. This demonstrates the power of an optimization run in providing the best possible strategies for the chosen objectives. To demonstrate the difference between the obtained solutions, we choose the highest trade-off objective vectors for each entity from the respective plots: Figure 2a for offense, and Figure 2b for



(a) PCP of the first 25 variables.

(b) PCP of the next 25 variables.

(c) PCP of the final 25 variables.

Fig. 3: PCP for all ND solutions obtained using independent optimizations of offense and defense entities. The highest trade-off solutions are shown in bold.

defense. In the respective ND front,
the specific trade-off solution causes a maximum loss in one objective for a unit gain in the other objective [19], making it a preferred choice among other ND solutions. It is clear that the two trade-off solutions are completely different from each other. This reveals the competitive nature of optimal strategies for the two entities. To understand the difference of ND strategies for two entities, we present the decision variables corresponding to trade-off as well as other ND points using the parallel coordinate plots (PCPs) in Figure 3. Clearly, the green and orange lines are aligned differently. Focusing on the two highest trade-off solutions (shown in bold), we observe that out of a total of 74 offense and defense variables, 53 variables (almost 70%) have completely different values. This experiment confirms that when we optimize the contradicting objectives, the algorithm searches in different portions of the search space resulting in solutions which are widely different in objective as well as variable space.

4.4 Multi-objective Competitive Co-evol. (MoCoEv) Optimization

Figures 2a and 2b have clearly shown that when strategies from two entities are intricately involved in defining objectives, individual optimization of one entity alone does not produce satisfactory results for both entities. The solutions are biased towards the entity for which the optimization is performed. This motivates us to consider the WSOP as a co-evolutionary optimization (CoEv) problem. Since each entity has two conflicting objectives on its own, the problem becomes a multi-objective co-evolutionary optimization (MoCoEV) problem. Moreover, since there is conflict in the individual optimal solutions between the two entities, as found by widely different values of the ND front solutions in Figure 2, the problem becomes more challenging and is known as a multi-objective competitive co-evolutionary optimization (MoCCoEv) problem. In this subsection, we apply the MoCoEV algorithm presented in Section 3 to find an ND set of trade-off solutions, each of which takes into account both entities during the optimization process.

Competitive Trade-off Solutions: We use a co-evolutionary version of NSGA-II having a population of size 50 for each co-evolving population and run the proposed MoCoEv algorithm for 200 generations with $\tau_1 = \tau_2 = 5$. The final outcome of the MoCoEv algorithm are two ND sets of solutions (having 10 solutions each), one for each entity. We compute average objective values for each solution for one entity with every member of the other entity and then identify the

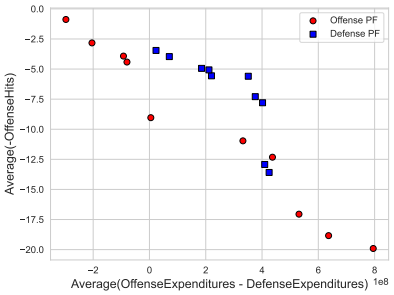


Fig. 4: Offense and Defense ND sets by MoCoEv.

ND solutions for both entities based on these average values.

The respective ND fronts based on these average objective vectors of each entity are plotted in Figure 4. Clearly, for the offense population, objectives (*OffenseExpenditures* - *DefenseExpenditures*) and $-$*OffenseHits* are minimized, while for the defense population, the negative of the objectives are minimized.

It is interesting to note that both ND fronts are now closer to each other, meaning that the MoCoEV algorithm is able to emphasize both entities' interests well in arriving at competitive solutions. The number of offense hits is now limited to a maximum of 20, instead of around 300 obtained using the offense's independent optimization. In all defense ND solutions, offense expenditure is more than that of defense. To achieve up to 10 offense hits, the offense needs to make little more expenditure than the defense, while the offense has to outspend the defense to destroy more than 10 assets of defense. For example, to achieve 20 offense hits, the offense has to spend about 8 million units more than the defense, however with about similar expenditure, the offense can achieve 9 offense hits. On the other hand, if the offense is happy with damage of about 2.5 units of assets, the offense entity can spend 2 million units of expenditure less than that of the defense. Moreover, ND solutions for the offense entity have a wider range of objectives than that of the defense entity. This can be due to the existence of more offense variables, thereby providing more ways to find a wider combination of variables. All these observations are interesting providing offense and defense users with a better insight into various alternate solutions before they prepare to launch any action.

Extracting Common Patterns in Trade-off Solutions: The process of extracting common patterns in a set of ND solutions was termed as the task of "innovization" [8]. Patterns can be extracted from the ND solutions manually using certain problem information [8] or using an automated machine learning process [4]. Here, we use a manual process in which all 74 variables of both entities are plotted in the order of their similarity among the obtained ND solutions. For this purpose, we compute the coefficient of variation (CV - σ/μ) of each solution and the order of the offense and defense variables in Figure 6.

A higher CV indicates higher dispersion around the mean. So, a lower CV reflects a better-converged value of variables towards the mean value. It is reported in [1] that a threshold of $\sigma/\mu = 0.3$ is an acceptable limit for assuming a good convergence. With this threshold, we observe that six of the 50 offense variables and six of the 24 defense variables can be considered well converged. The number of values that each variable can take in the original formulation is mentioned in parentheses in Figure 6.

Fig. 5: Heatmap of converged variables for offense and defense entities.

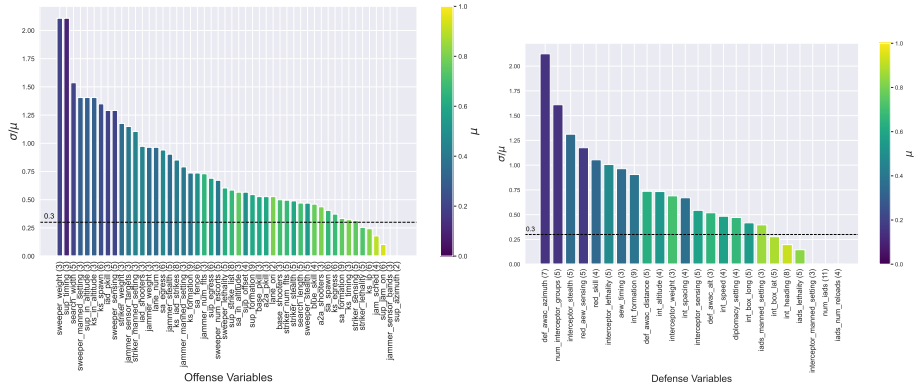


Fig. 6: Coefficient of variation for offense and defense variables indicate convergence of certain variables across ND solutions.

To analyze further, we normalize values for the converged variables over the PFs between zero and one and are plotted in a heat map in Figure 5. Each variable's values in 10 ND solutions are shown in columns for each entity. Almost identical colors for each row (variable) indicate the convergence level visually. For the offense variables, we note that three of the six variables are related to electronic warfare MP, suggesting that there is a set of plans for the offense entity that significantly inhibit the perception and communication of the defender. Other converged offense variables dictate where a strike is safe and effective (KS_IP), as well as the simple fact that carrying more air-to-ground munitions is consistently a good choice (striker_lethality). From the defense's perspective, we observe that more surface-to-air missile launchers (num_iads) that can fire more missiles (num_reloads) for a longer duration (iad lethality) is a cost-effective strategy. This also pairs well with autonomous interceptors (interceptor manned).

Optimization With and Without Coevolution: Figure 7 marks the original high-fidelity objective vectors (in yellow circles) used to construct the surrogate models of the objectives. The data were centered around equal expenditures for both entities causing on average around four offense hits. These solutions are marked with the label "Surrogate Fitted Training Points". If we do not perform any optimization run and try to locate the best strategies for the offense entity, we find the respective ND front marked using orange circles on the top-left part of the yellow circles. Similarly, when we locate the best strategies for defense, they are at the bottom-right corner of the yellow circles, marked using brown squares.

We embed the individually optimized ND solutions in the plot for both offense and defense and they are marked using brown circles and black squares, respectively. Notice that both these sets clearly dominate the respective sets obtained from initial high-fidelity data only and without resorting to any optimization. Clearly, these individual optimal solutions are better due to the efforts put in by the optimization algorithm. However, it was discussed before that these

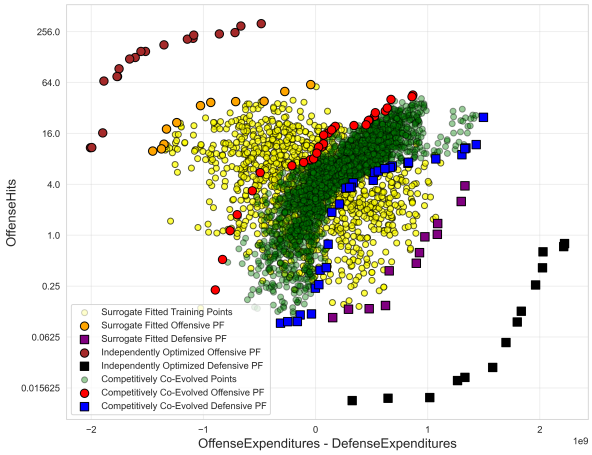


Fig. 7: Training data, independently-optimized and MoCoEv strategies.

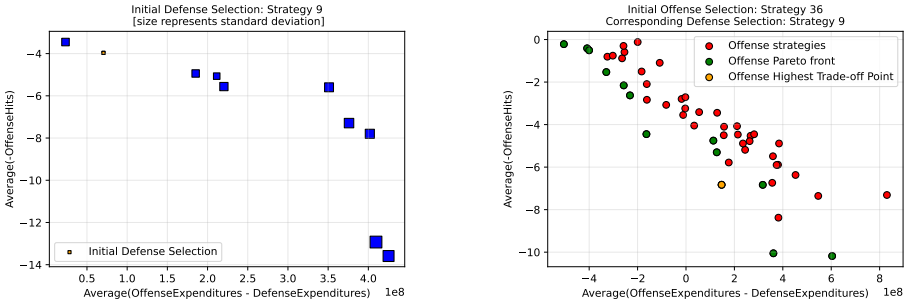
individual solutions are not practical, as the optimizations ignore the ability of the other entity to influence the solutions of its own entity.

Next, we plot the MoCoEV ND solutions found for both entities in red circles and blue squares. All final strategies found via the MoCoEV optimization are also shown in green-colored open circles. It is clear that the coevolution-based solution sets are closer to each other, thereby respecting each other's abilities to arrive at challenging solutions for each other. Due to the consideration of two conflicting objectives for each entity, both ND sets produce trade-off solutions between two objectives. We argue that the MoCoEv trade-off solutions stay as viable strategies from which each entity can choose a solution.

Multi-Criterion Decision Making on MoCoEV Solutions: Finding a set of ND solutions for each entity in a co-evolutionary process is a challenging computing task, but it only completes a part of the whole WSOP task. The next important task is to choose preferred solutions for implementation. Despite a plethora of multi-criterion decision-making (MCDM) studies in the literature [18,22], MCDM studies for two co-evolved Pareto sets is missing. In this paper, we make an effort to propose a viable MCDM procedure.

Since two entities – defense and offense – are independent, likely they will also make decisions independently. However, the linking of the two entities in defining the objectives suggests that a sequential decision-making task must be made alternating between them. This will lead to a defender-attacker simulation game which will start with declaring a defense's solution. The offense then has the opportunity to choose its solution to counteract the declared defense's solution. After the offense's solution is announced, the defense has the next move to choose its most appropriate action. This iterative process can continue until it reaches an equilibrium pair of strategies or an equilibrium cycle of strategies. With this iterative scheme of the MCDM procedure in mind, the question remains as to how to choose a preferred solution when the opponent has made its move.

Before we propose an MCDM scheme for this study, we make an important assertion about our decision-making approach. We assume that each entity will confine its decision-making to its own ND set corresponding to the final set of strategies for the other entity. Since the MoCoEv is expected to find the best possible trade-off solution set considering the opponent's best possible moves, each set is expected to have the best counter-moves in them. Hence, it makes sense to use the obtained ND set to choose a solution from.



(a) Selection of initial defense strategy. (b) Initial offense strategy selection.

Fig. 8: Initial strategy selection by offense and defense.

As outlined in Section 3.2, we first identify the specific defense strategy corresponding to the smallest standard deviation in its objective vectors arising from the various combinations of solutions in the offense population. Figure 8a marks the size of each defense solution in proportion to the corresponding standard deviation arising from all offense strategies. Strategy 9 (second from top-left point), when combined with each offense ND solution produces the least standard deviation in both defense objective values. Hence, it may be considered the most robust strategy against any strategy that the offense entity may choose next. Thus, *Defense_ 9* is selected as the initial defense strategy.

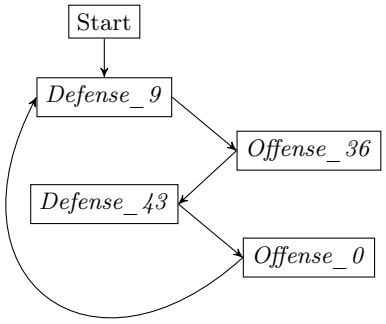


Fig. 9: Proposed method results in an equlibrium cycle of strategies.

After selecting the initial defense strategy, we now plot the objective vector of every offense strategy with this defense strategy (*Defense_ 9*) in Figure 8b. Clearly, not all objective vectors are non-dominated to each other. We now identify the non-dominated set from these vectors and choose the highest trade-off point. This strategy *Offense_ 36*, marked in a yellow circle, is the best response by the offense to the *Defense_ 9* strategy of the defense.

Due to this lop-sided loss-to-gain ratio, there is no motivation to move to its neighbor for a better outcome. Next, it is the turn of the defense to find the best possible defense strategy from its own MoCoEv-obtained set of strategies. With a

similar trade-off analysis of non-dominated solutions obtained from all objective vectors computed using *Offense_ 36* and each defense's ND solution, we observe that *Defense_ 43* is the best option. Next, we find *Offense_ 0* strategy of offense ND set is the best response. One more trade-off analysis of the defense ND set reveals that *Defense_ 9* is now the best defense strategy. Interestingly, *Defense_ 9* was one of the chosen strategies considered before and which led to *Offense_ 0* in a few iterations. Continuing the decision-making process further will lead to a cycle of offense and defense strategies which we have already observed before. Thus, the MCDM process ends up in an equilibrium cycle of strategies between the two entities. This ends the WSOP task. Figure 9 shows the cycle by clearly marking the sequence of defense and offense strategies.

5 Conclusions and Future Studies

In this paper, we have proposed a multi-objective, competitive, co-evolutionary optimization algorithm and a decision-making strategy to deal with two agents, whose evaluation functions require both agent's variables. In every sense, the problem has introduced challenges in arriving at practical solutions. The optimization problem is challenging due to the involvement of multiple conflicting objectives and inter-dependencies of both agent's variables, leading to two non-dominated fronts. The decision-making is challenging due to involvement of two independent decision-makers, each deciding from a different non-dominated set of strategies in tandem and in response to opponent's moves. Not only does each agent choose a preferred strategy from its own ND set by trading-off two conflicting goals, the chosen strategy must also be appropriate in response to opponent's recently chosen strategy. Standard evolutionary multi-objective optimization (EMO) and multi-criterion decision-making (MCDM) methods are not as involved as MoCoEV and ensuing decision-making tasks.

We have applied the proposed approaches to a wargame strategy optimization problem (WSOP) to illustrate its working and revealing the complex interactions involved in solving an attacker-defender system. From the five objectives of interest for defense and offense agents, a correlation analysis has allowed us to reduce the problem to have only two objectives for each agent. For the first time, we have proposed a sequential MCDM approach by involving one agent at a time. A trade-off-based MCDM approach has been proposed to find the best ND solution of one entity as a response to all ND solutions of the other strategy. In the specific case study, the proposed MCDM scheme has resulted in an equilibrium cycle of offense-defense strategies as an end result.

This study has clearly shown the advantage of using evolutionary computation in addressing multi-objective competitive multi-agent systems and investigates a number of future studies. Iterated MCDM schemes may involve the depletion of resources into consideration as the game progresses. It may also restrict future moves only to moves allowed by the initial moves by each entity. More than two objectives can be considered to have more flexible trade-off solutions. The MoCoEv and ensuing MCDM approach can be applied to other

similar multi-agent systems in achieving a better understanding of the effect of sequential decision-making strategies in achieving safe and secure systems.

Disclosure of Interests. I declare no competing interests as defined by Springer Nature, or other interests that might be perceived to influence results and/or discussion reported in this manuscript.

References

1. Coefficient of variation, https://www.isixsigma.com/dictionary/coefficient-of-variation/, accessed on September 27, 2023
2. Command Modern Air and Naval Operations. https://www.matrixgames.com/game/command-modern-air-naval-operations-wargame-of-the-year-edition, [Accessed 19-06-2024]
3. Atashpendar, A., Dorronsoro, B., Danoy, G., Bouvry, P.: A scalable parallel cooperative coevolutionary PSO algorithm for multi-objective optimization. Journal of Parallel and Distributed Computing **112**, 111–125 (2018)
4. Bandaru, S., Ng, A.H.C., Deb, K.: Data mining methods for knowledge discovery in multi-objective optimization: Part A – survey. Expert Systems With Applications **70**, 139–159 (2017)
5. Barbosa, H.J.C.: A genetic algorithm for min-max problems. In: Proceedings of the First International Conference on Evolutionary Computation and Its Application (EvCA'96). pp. 99–109 (1996)
6. De Lima Filho, G.M., Kuroswiski, A.R., Medeiros, F.L.L., Voskuijl, M., Monsuur, H., Passaro, A.: Optimization of unmanned air vehicle tactical formation in war games. IEEE Access **10**, 21727–21741 (2022)
7. Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A fast and elitist multi-objective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation **6**(2), 182–197 (2002)
8. Deb, K., Srinivasan, A.: Innovization: Innovating design principles through optimization. In: Proceedings of the 8th annual conference on Genetic and evolutionary computation. pp. 1629–1636 (2006)
9. Dorronsoro, B., Danoy, G., Nebro, A.J., Bouvry, P.: Achieving super-linear performance in parallel multi-objective evolutionary algorithms by means of cooperative coevolution. Computers & Operations Research **40**(6), 1552–1563 (2013)
10. Garcıa-Pedrajas, N., Hervás-Martınez, C., Munoz-Pérez, J.: Multi-objective cooperative coevolution of artificial neural networks (multi-objective cooperative networks). Neural Networks **15**(10), 1259–1278 (2002)
11. Goh, C.K., Tan, K.C., Liu, D., Chiam, S.C.: A competitive and cooperative coevolutionary approach to multi-objective particle swarm optimization algorithm design. European Journal of Operational Research **202**(1), 42–54 (2010)
12. Jia, Z.X., Kiang, J.F.: War game between two matched fleets with goal options and tactical optimization. AI **3**(4), 890–930 (2022)
13. Keerativuttitumrong, N., Chaiyaratana, N., Varavithya, V.: Multi-objective cooperative co-evolutionary genetic algorithm. In: International Conference on Parallel Problem Solving from Nature. pp. 288–297. Springer (2002)
14. Li, Y., Wang, J., Liu, Z.: A simple two-agent system for multi-objective flexible job-shop scheduling. Journal of Combinatorial Optimization **43**(1), 42–64 (2022)

15. Luo, J., Cooper, J., Cao, C., Pham, K.: Cooperative adaptive control of a two-agent system. In: 2012 American Control Conference (ACC). pp. 2413–2418. IEEE (2012)
16. McIntyre, A.R., Heywood, M.I.: Multi-objective competitive coevolution for efficient gp classifier problem decomposition. In: 2007 IEEE International Conference on Systems, Man and Cybernetics. pp. 1930–1937. IEEE (2007)
17. Meneghini, I.R., Guimaraes, F.G., Gaspar-Cunha, A.: Competitive coevolutionary algorithm for robust multi-objective optimization: The worst case minimization. In: 2016 IEEE congress on evolutionary computation (CEC). pp. 586–593. IEEE (2016)
18. Miettinen, K.: Nonlinear Multiobjective Optimization. Kluwer, Boston (1999)
19. Mittal, S., Kumar, D., Deb, S.K.: A unified automated innovization framework using threshold-based clustering. In: 2020 IEEE Congress on Evolutionary Computation (CEC). pp. 1–8. IEEE (2020)
20. Ozaki, A., Furuichi, M., Takahashi, K., Matsukawa, H.: Design and implementation of parallel and distributed wargame simulation system and its evaluation. IEICE TRANSACTIONS on Information and Systems **84**(10), 1376–1384 (2001)
21. Paredis, J.: Coevolutionary constraint satisfaction. In: Parallel Problem Solving from Nature III (PPSN-III). pp. 46–55 (1994)
22. Steuer, R.E.: Multiple Criteria Optimization: Theory, Computation and Application. New York: Wiley (1986)
23. Wang, K., Gou, C., Duan, Y., Lin, Y., Zheng, X., Wang, F.Y.: Generative adversarial networks: introduction and outlook. IEEE/CAA Journal of Automatica Sinica **4**(4), 588–598 (2017)
24. Zeng, F., Decraene, J., Low, M.Y.H., Cai, W., Hingston, P.: Studies on pareto-based multi-objective competitive coevolutionary dynamics. In: 2011 IEEE Congress of Evolutionary Computation (CEC). pp. 2383–2390. IEEE (2011)