Towards Multi-objective Co-evolutionary Problem Solving

Anirudh Suresh, Kalyanmoy Deb[0000−0001−7402−9939], and Vishnu Naresh Boddeti[0000−0002−8918−9385]

Michigan State University, East Lansing, MI, USA
{suresha2,kdeb,vishnu}@msu.edu

Abstract. Co-evolutionary algorithms deal with two co-evolving populations, each having its own objectives. The linking of the two populations comes from the fact that the evaluation of the objective of a member of the first population requires a companion member from the second population, and vice versa. These algorithms are of great interest in cooperative and competing games and search tasks in which multiple agents having different interests are in play. While there have been significant studies devoted to single-objective co-evolutionary optimization algorithms and applications, they have not been paid much attention when each population must be evolved for more than one conflicting objectives. In this paper, we propose a multi-objective co-evolutionary (MOCoEv) algorithm and present its working on two interesting problems. This proof-of-principle study suggests that the presence of multiple Pareto solutions for each population and the ensuing multi-criterion decision-making complexities make the MOCoEv research and application be challenging. This paper should spur immediate further attention to multi-objective co-evolutionary problem solving studies.

Keywords: Co-evolutionary algorithm · multi-payoff game · multi-objective optimization · multi-criterion decision making.

1 Introduction

Many problems in practice, particularly in secure and trustworthy system design, must be considered from the point of view of two independent agents who have either altruistic or adversarial interests [10]. Recent studies in generative adversarial networks (GANs) [9] in computer vision literature in arriving at deep neural networks (DNNs) for generating realistic fake images, or the dynamics of adversarial attacks [13] and defenses of DNNs are some examples. Designing power systems, for example, from possible adversarial attacks require a model of generating plausible attacks so that a better secured system can be developed. Often, such a consideration leads to an 'arms race' in evolving defender-and-attacker systems simultaneously. Another wide application of such co-evolutionary system design is in multi-agent games [11].

These problems give rise to two inter-connected sub-problems, each having its own decision variables (or strategies) which must be searched for a better combination to maximize a goal. A difficulty arises due to the fact that the evaluation

of the goal (or objective) requires a combination of both sub-problem's decision variables. Unfortunately, the second sub-problem's decision variables cannot be controlled by the first sub-problem and vice versa. Thus, these problems are usually posed as co-evolutionary optimization problems and two populations (one dealing with each sub-problem) are evolved with a close interaction with each other in an evolutionary framework [14]. Competitive co-evolutionary methods have been used to solve single objective min-max problems [1]. Some recent works have focused on finding the objective trade-offs for multi-objective games (MOGs) and have proposed a new solution concept based on rational strategies from game theory [6–8, 11]. Eisenstadt et al. [6] used co-evolution to compute multiple rational strategies in an MOG. No efficient aggregate performance is computed to reduce the cardinality of the final solution set, but an elaborate multi-objective decision making (MCDM) approach [5] is used to choose the desired solutions. Żychowski et al. [15] used memetic co-evolution to solve MOGs using the nadir point (worst case) as the representative vector of rational fronts.

Despite all these studies, co-evolutionary frameworks have not been studied enough for optimizing multiple conflicting objectives efficiently at each sub-problem. Considering multiple conflicting objectives in a co-evolutionary framework introduces a number of challenges. First, each sub-problem must find a set of Pareto-optimal (PO) solutions optimizing its own objectives well. This requires a widely diverse population to be maintained at each sub-problem throughout the optimization process. This calls for developing efficient methodologies for computing aggregate fitness vectors that would emphasize survival of a diverse population. Second, the final PO solutions from each sub-population must be analyzed using a multi-criterion decision-making approach and decision-makers' preferences to pick a single preferred solution. We address all the above concerns in this paper.

In the remainder, we define a multi-objective co-evolutionary (MOCoEv) problem in Section 2 by highlighting the challenges they introduce to any algorithm. Then, in Section 3, we present our proposed MOCoEv algorithm in detail. Two multi-criterion decision-making (MCDM) approaches are discussed in this section. Results on two problems are presented in Section 4. Finally, conclusions are discussed in Section 5.

2 Multi-objective Co-evolutionary Problems

Without loss of generality, we consider two co-evolving problems in this study. Problems P_1 and P_2 deal with variable vectors $\mathbf{x}$ and $\mathbf{y}$ of sizes n_1 and n_2 , respectively. The objective and constraint functions for P_i is given as $\mathbf{f}^{(i)}(\mathbf{x}, \mathbf{y})$ of size M_i and $\mathbf{g}^{(i)}(\mathbf{x}, \mathbf{y}) \leq \mathbf{0}$ of size J_i. The optimization problem formulation is given below (for simplicity, we do not consider equality constraints here):

Problem P_1	Problem P_2
$\min_{\mathbf{x}} \left(f_1^{(1)}(\mathbf{x}, \mathbf{y}), \ldots, f_{M_1}^{(1)}(\mathbf{x}, \mathbf{y})\right),$	$\min_{\mathbf{y}} \left(f_1^{(2)}(\mathbf{x}, \mathbf{y}), \ldots, f_{M_2}^{(2)}(\mathbf{x}, \mathbf{y})\right),$
$s.t.\ g_j^{(1)}(\mathbf{x}, \mathbf{y}) \leq 0,\ j = 1, \ldots, J_1,$ (1)	$s.t.\ g_j^{(2)}(\mathbf{x}, \mathbf{y}) \leq 0,\ j = 1, \ldots, J_2,$ (2)
$x_i^{(L)} \leq x_i \leq x_i^{(U)},\ i = 1, \ldots, n_1.$	$y_j^{(L)} \leq y_j \leq y_j^{(U)},\ j = 1, \ldots, n_2.$

It is expected that the objectives of each problem are in conflict with each other, thereby resulting in its own Pareto-optimal (PO) solutions. However, since each objective function of a Problem depends on a variable vector of the other Problem, each multi-objective optimization problem cannot be optimized independently. In other words, if we ignore P_2, and optimize P_1 alone by including $\mathbf{y}$ in its variable set, the resulting PO front is expected to be better than the PO front of the above MOCoEv problem P_1. The same is true for the independently optimized P_2 as well. The solution consists of PO front for each problem, with choices and objective preferences of the other problem unknown. This necessitates the use of a co-evolutionary search and optimization algorithm to solve such inter-linked problems and find a compromised PO front for each problem simultaneously, which is the focus of this paper.

2.1 Cooperative and Competing Problems

Let us first consider that there is a single objective function for each problem. Thus, Problem 1 will correspond to a single optimal solution $\mathbf{z}^{(1),*} = (\mathbf{x}^{(1),*}, \mathbf{y}^{(1),*})$ and Problem 2 will correspond to another optimal solution $\mathbf{z}^{(2),*} = (\mathbf{x}^{(2),*}, \mathbf{y}^{(2),*})$. If these two solutions are exactly the same (or close to each other in their respective variable spaces), a co-evolutionary framework will work in a cooperative manner, thereby making the problem relatively easier to solve.

On the other hand, if these two optimal solutions are distant in their respective variable spaces, a standard co-evolutionary framework (which emphasizes its own variables only based on its performance on its own objective and constraint functions) must preserve population members close to each of these optima. Population 1 has no motivation to preserve points close to $\mathbf{z}^{(2),*}$, since its goal is to minimize $\mathbf{f}_1^{(1)}$ with its constraints, and Population 2 has no motivation for saving solutions close to $\mathbf{z}^{(1),*}$. It is clear that to solve competing problems, the survival of solutions in each population must also consider the importance of its own variables in the other population.

For multi-objective co-evolutionary problems, Population 1 will correspond to a set of PO solutions ($\mathbf{Z}^{(1)} = (\mathbf{z}^{(1),i},\ i = 1, \ldots, N_1)$). Similarly, Population 2 will correspond to another set of PO solutions ($\mathbf{Z}^{(2)} = (\mathbf{z}^{(2),i},\ i = 1, \ldots, N_2)$). In a cooperative MOCoEv problem, these two sets are expected to be close to each other in their respective variable spaces and in a completely competing MOCoEv problem, the two sets are expected to be far away from each other. However, in a generic case, it is expected that the two sets will have some common solutions and some other solutions that are different from each other. To find and maintain two sets of solutions, each population must, not only emphasize its own best solutions, but also must emphasize its own solutions corresponding to good solutions of the other population. This co-evolutionary problem is different from a traditional many-objective problem with objectives from both populations. In the many objective problem, the solution would involve explicit trade-off between objectives of each population. However, in the co-evolutionary problem, the objectives of the two populations implicitly reach a compromise that helps in achieving a selfish trade-off amongst each population's own objectives. Next, we

present a new multi-objective co-evolutionary algorithm based on a well-known EMO algorithm – NSGA-II [4].

3 Proposed Multi-Objective Co-evolutionary (MOCoEv) Algorithm

In an 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. 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 $Pop_1^{(t)}$ can be obtained. Here, we follow the NSGA-II's operations for this purpose. A total τ_1 iterations can be continued as above before the next population is updated for a consecutive τ_2 iterations. Then, again Population 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-algorithm of the MOCoEv procedure is presented in Algorithm 1.

Algorithm 1: Proposed multi-objective co-evolutionary algorithm.

Input: Population sizes N_1 and N_2 for Pop_1 and Pop_2, number of generations T, number of generations of each population τ_1 and τ_2
Output: Final populations $Pop_1^{(T)}$ and $Pop_2^{(T)}$
Initialize populations $Pop_1^{(0)}$ and $Pop_2^{(0)}$
for *gen = 0 to T or another termination condition not satisfied* **do**
  $Pop_1^{(0)} = Pop_1^{(gen)}$, $Pop_2^{(0)} = Pop_2^{(gen)}$
  for t *= 1 to* τ_1 **do**
    Generate offspring $Q_1^{(t)}$ for $Pop_1^{(t-1)}$ using selection and variation operators
    Merge and redefine $Pop_1^{(t)} = Pop_1^{(t-1)} \cup Q_1^{(t)}$
    Evaluate $Pop_1^{(t)}$ with all members of $Pop_2^{(gen)}$
    Survive elite population of $Pop_1^{(t)}$ of size N_1
  end
  $Pop_1^{(gen)} = Pop_1^{(t)}$
  for t *= 1 to* τ_2 **do**
    Generate offsprings $Q_2^{(t)}$ for $Pop_2^{(t-1)}$ using selection and variation operators
    Merge and redefine $Pop_2^{(t)} = Pop_2^{(t-1)} \cup Q_2^{(t)}$
    Evaluate $Pop_2^{(t)}$ with all members of $Pop_1^{(gen)}$
    Survive elite population of $Pop_2^{(t)}$ of size N_2
  end
  $Pop_2^{(gen)} = Pop_2^{(t)}$
end

However, there is a complication in the evaluation process which we discuss next. For evaluating a single population member $\mathbf{x}^{(1),i}$ of $P_1^{(t)}$, we need a specific variable vector $\mathbf{y}$ from the second population. This is where a number of strategies can be adopted in an MOCoEv algorithm. We consider three ways to evaluate a solution:

1. **Mean Aggregate Fitness:** Population member $\mathbf{x}^{(1),k}$ is paired with every member $\mathbf{y}^{(2),l}$ of the second population one at a time and objectives are evaluated. Then, a mean fitness value of i-th objective function of Population 1 is computed as follows:

$$F_i(\mathbf{x}^{(1),k}) = \frac{1}{N_2}\sum_{l=1}^{N_2} f_i(\mathbf{x}^{(1),k}, \mathbf{y}^{(2),l}). \tag{3}$$

   Figure 1 illustrates the procedure for a specific solution in the first population for the entire population members of the second population. Similarly, the fitness of each member of the second population can also be computed by averaging the respective objective values over all Population 1 members. Note that instead of considering all N_2 population members as above, the equation can be used for non-dominated solutions of the second population only. This would be particularly useful for cooperative MOCoEv problems.
2. **Best Aggregate Fitness:** Instead of the mean value, the best objective value can be assigned as a fitness: $F_i(\mathbf{x}^{(1),k}) = \min_{l=1}^{N_2} f_i(\mathbf{x}^{(1),k}, \mathbf{y}^{(2),l})$. This is useful for a greedy evaluation of a solution, if desired in a problem. Figure 1 shows the best aggregate fitness value.
3. **Worst Aggregate Fitness:** The worst objective value can be assigned as a fitness: $F_i(\mathbf{x}^{(1),k}) = \max_{l=1}^{N_2} f_i(\mathbf{x}^{(1),k}, \mathbf{y}^{(2),l})$. Figure 1 shows the worst aggregate fitness value. This will be particularly suitable for a pessimistic (conservative) evaluation of a solution, if suitable for a problem. For example, in min-max problems, this can be an effective aggregation function.

After the aggregate fitness function for each objective is computed by using one of the above strategies, they can be used for domination check and other niche-preserving operator of the EMO algorithm. Consider Figure 2, in which the non-domination evaluation procedure using the mean aggregate fitness function is illustrated for Population 1. Every member is evaluated for both F_1 and F_2 using member of the second population. Then, the mean fitness vector is computed (shown with a star) for each member and is used for domination check. For the shown example, yellow and blue colored points of Population 1 belong to the first non-dominated (ND) front, followed by the brown point in the second ND front. The crowding distance values for each member is also computed using the aggregate fitness values. Thus, the final trade-off set of solutions of each population ($\mathbf{Z}^{(1)}$ and $\mathbf{Z}^{(2)}$) will correspond to non-domination principle of the chosen aggregate fitness functions.

3.1 Multi-Criterion Decision Making in MOCoEv Problems

The next step in a multi-objective optimization problem solving task is to choose a single preferred solution based on decision-maker's preferences. There exist a

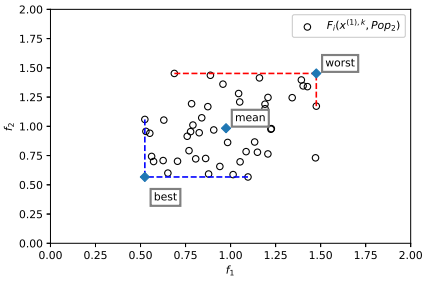


Fig. 1: Three aggregation procedures.

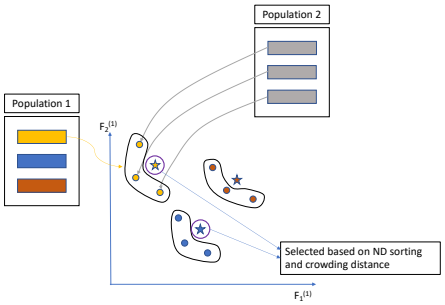


Fig. 2: Non-dominated sorting using mean aggregate fitness values.

number of methods in the multi-criterion decision-making (MCDM) literature [12] for this purpose. Here, we discuss a few MCDM methods, specifically adapted for solving MOCoEv problems. Note that a decision-making process for choosing a preferred solution for Population 1 must now involve likely decision-making in the second population and vice versa.

Pseudo-weight Approach: From the obtained ND set of i-th population, a pseudo-weight vector [2] can be computed for k-th member ($\mathbf{x}^{(i),k}$) using the aggregate fitness function considered during the optimization process:

$$w_m^{(i),k} = \frac{\left(F_m^{(i),\max} - F_m(\mathbf{x}^{(i),k})\right) / \left(F_m^{(i),\max} - F_m^{(i),\min}\right)}{\sum_{m=1}^{M_i}\left(F_m^{(i),\max} - F_m(\mathbf{x}^{(i),k})\right) / \left(F_m^{(i),\max} - F_m^{(i),\min}\right)}, \; \forall m = 1, \ldots, M_i, \tag{4}$$

where $F_m^{(i),\min}$ and $F_m^{(i),\max}$ are the minimum and maximum fitness values of the final ND members of the i-th population, respectively. The pseudo-weight vector $(w_1^{(i),k}, \ldots, w_{M_i}^{(i),k})$ for k-th member of i-th population indicates the relative importance of each objective in the span of the entire ND set of the population. Thereafter, the member $(\mathbf{x}^{(i),T})$ which has the pseudo-weight vector closest to a desired preference vector $\mathbf{d} = (f_1^{(i),T}, \ldots, f_{M_i}^{(i),T})$ is chosen for the i-th population. For two-population coevolution, the interaction between two populations in arriving at the final preferred solutions ($\mathbf{x}^{(1),T}$ and $\mathbf{y}^{(2),T}$) come from the aggregation used during the optimization process. The same can also be applied to Population 2 members.

Minimum Sensitivity Approach: Another pragmatic decision-making (DM) approach is proposed here. When a specific ND solution is chosen for Population 1 for implementation, there is a distribution of the fitness vector caused by the presence of different ND solutions at the final generation of Population 2. Thus, one DM idea would be to pick that solution from Population 1 which makes the tighter distribution in the resulting fitness vectors. One way to compute the extent of distribution would be to compute the normalized average distance of every objective vector from the mean objective vector:

$$\mathbf{x}^{(1),*} = \text{argmin}_{k=1}^{N_1} \frac{1}{N_2} \sum_{l=1}^{N_2} \|\mathbf{F}_{\text{mean}}^{(1)}(\mathbf{x}^{(1),k}), \mathbf{F}(\mathbf{x}^{(1),k}, \mathbf{y}^{(2),l})\|_2, \tag{5}$$

where $\|\cdot\|_2$ is the normalized Euclidean distance measure and $\mathbf{F}^{(1)}_{\text{mean}}(\mathbf{x}^{(1),k})$ is the mean fitness vector of ND solution $\mathbf{x}^{(1),k}$. There are other well-known DM approaches [12], such as, the trade-off method, compromise programming approach, which can also be used.

The above approaches can be made more pragmatic by assuming scenarios in which the DM information for the second population (say, the hacker's problem) is guessed from their past practices and are available. We can simulate this scenario by choosing a specific set of neighboring ND solutions ($\mathbf{Z}^{(2),S}$) from Population 2's final ND set. The defender's (Population 1) DM task will then be to choose a single preferred solution that is most appropriate to the likely chosen solutions of Population 2. One of the above procedures can be repeated using the neighboring ND solutions of Population 2, instead of the entire population. An advantage of first finding multiple Pareto strategies is that an iterative decision-making procedure can choose the most appropriate strategy from the obtained Pareto sets, thereby allowing an online and computationally fast procedure for choosing multi-objective co-evolutionary results.

4 Results

We consider two example problems to demonstrate the effectiveness of the proposed algorithm. We first consider an existing problem – Tug of War – as a proof-of-concept result and thereafter apply our MOCoEv algorithm and decision-making approaches to a numerical problem.

4.1 Tug-of-War Problem

Tug-of-war is a differential game proposed in [6]. It is a two player, non-cooperative, imperfect information game with one decision variable for each player. The game consists of a mass m placed on a friction-less horizontal 2-dimensional plane, shown in Figure 3. Player 1 applies a force F_1 at an angle $x = \theta_1$ and Player 2 applied F_2 at an angle $y = \theta_2$ aiming to move the mass. The players aim to maximize or minimize the resulting Cartesian coordinates of the mass, m_x and m_y.

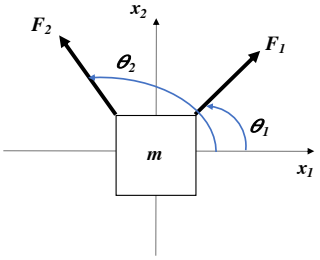


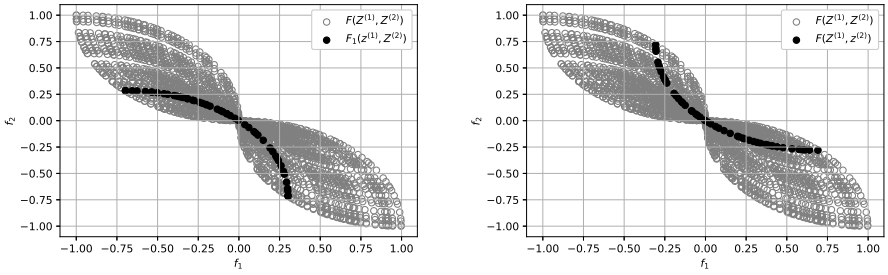
Fig. 3: Tug of war problem.

Forces and acceleration are considered constants. Acceleration along axes are: $\ddot{x_1} = F_1\cos(\theta_1) + F_2\cos(\theta_2)$ and $\ddot{x_2} = F_1\sin(\theta_1) + F_2\sin(\theta_2)$. Final position of the mass are: $x_1(t) = x_1(0) + \dot{x}_1(0)t + \frac{1}{2}\ddot{x}_1t^2$ and $x_2(t) = x_2(0) + \dot{x}_2(0)t + \frac{1}{2}\ddot{x}_2t^2$. To simplify the problem, initial conditions are assumed as $x_1(0) = x_2(0) = 0$ and $\dot{x}_1(0) = \dot{x}_2(0) = 0$. $F_1 = F_2 = 1N$ and $m = 1kg$ are assumed. The position after $t_f = \sqrt{2}$ sec is used to construct the objective functions with $x = \theta_1$ and $y = \theta_2$, both in the range $[0, 2\pi]$: $m_x(x, y) = \cos(x) + \cos(y)$ and $m_y(x, y) = \sin(x) + \sin(y)$. Thus, according to our MOCoEv problem formulation, the minimization objectives are given as follows:

$$f_1^{(1)}(x, y) = m_x(x, y), \qquad f_2^{(1)}(x, y) = m_y(x, y),$$
$$f_1^{(2)}(x, y) = -m_x(x, y), \qquad f_2^{(2)}(x, y) = -m_y(x, y).$$

It is clear that $f_i^{(1)}$ and $f_i^{(2)}$ are in conflict with each other. To achieve minimum coordinate values in Population 1, variable θ_1 is expected to lie within $[\pi, 1.5\pi]$, ideally with θ_2 values also in the same range. However, variable θ_2 lying in $[0, \pi/2]$ is expected to be optimal for Population 2, ideally with θ_1 values also in the same range. In these independent cases, $\theta_1 = \theta_2$ for all PO solutions of each population and the respective PO front will lie on the third quadrant of a circle of radius 2.0 for Population 1 and on the first quadrant of the same circle for Population 2. However, since neither of these ranges is optimal for the other population, the PO front for each population of the MOCoEv problem will be compromised from these idealized fronts, as we shall demonstrate with our results.

In our simulations, we use a population size of $N_1 = N_2 = 50$, SBX crossover operator [3] with $\eta_c = 20$ and $p_c = 0.5$, and polynomial mutation with $\eta_m = 50$, $p_m = 0.5$ and $T = 50$ generations for each population. Frequency of each iteration $\tau_1 = \tau_2 = 1$ is used here. We use the mean aggregation fitness function for both populations. The gray points in Figure 4a show all final x-y positions of mass for each solution of $\mathbf{Z}^{(1)}$ evaluated with every solution from $\mathbf{Z}^{(2)}$. The distribution of the mass position for a specific solution ($\theta_1 = 225.71$ deg), lying in the middle of the PO front of Population 1, occurs due to a spread in the second population θ_2 solutions, as shown by the dark colored circles. For this specific θ_1 in the third quadrant, several θ_2 values in the first quadrant creates a range of PO solutions. Similarly, in Figure 4b, we show distributions of each θ_2 from Population 2 for every θ_1 from Population 1. For a specific $\theta_2 = 46.05$ deg, the respective PO front is shown in dark color. These two figures make it clear that by keeping a good distribution of θ_1 values in the third quadrant and θ_2 values in the first quadrant, both populations can achieve a compromise PO front for each player.

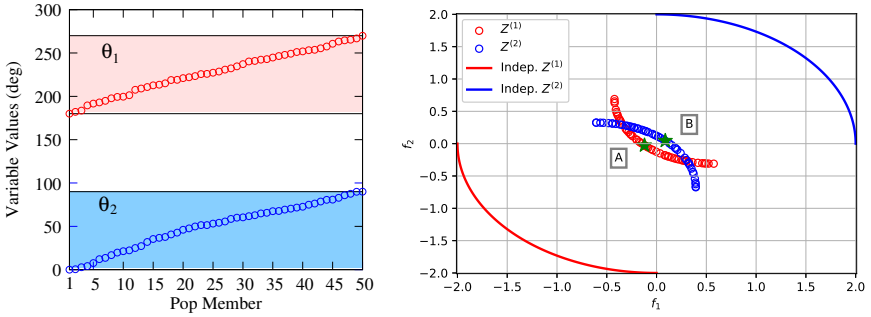


(a) Population 1. (b) Population 2.

Fig. 4: Final positions of mass for PO solutions of two populations.

Figure 5a shows the θ_1 and θ_2 of the final populations. It can be seen that they lie in third and first quadrant, respectively, with a good uniform spread. Figure 5b shows the PO front with the mean fitness values for each population. The compromise of these two fronts from their idealized independent fronts (shown in red and blue colors) to achieve each population's effect on each other is very

clear from this figure. Population 1 mean fitness values are worse than the independent P_1 PO front, as a co-evolutionary P_2 optimization does not allow independent P_1 front to survive. These results are similar to that in [6].



(a) PO solutions in x and y space. (b) PO fronts of both populations with mean aggregate fitness.

Fig. 5: Final solutions for the tug-of-war problem.

4.2 Decision-making for Tug-of-war Problem

Figure 5b represents the PO front with respect to the mean performance. These points can be used to choose a preferred point when the opponent's choice is unknown. If we desire to choose a point from the PO front that has pseudo-weight (according to Equation 4) close to $(0.5, 0.5)$ (having equal importance to both objectives) for both the populations, points marked with a star are points in Populations 1 and 2, respectively: $\theta_1 = 225.71$ deg, $\theta_2 = 46.05$ deg. They seem to lie in the middle of each PO front, ensuring an equal importance to each objective. The dark colored points used in Figure 4 used these points as well.

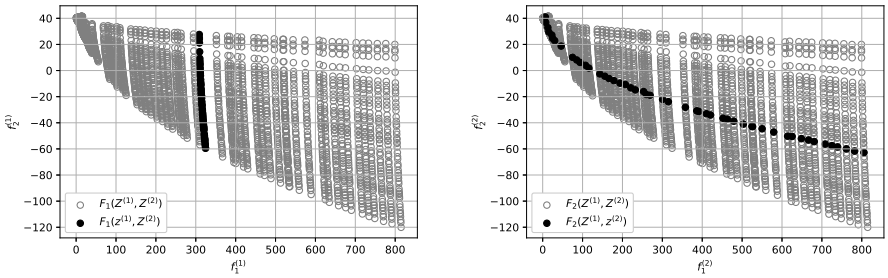
4.3 Numerical Problem NP

Next, we design a two-objective numerical problem for each of two populations which must be solved in a co-evolutionary manner. Problem 1 involves three variables and Problem 2 involves two variables: For P_1:

$$f_1^{(1)}(\mathbf{x},\mathbf{y}) = x_1^2 + x_2^2 + x_3^2 + y_1^2, \quad f_2^{(1)}(\mathbf{x},\mathbf{y}) = -(x_1+x_2)y_1 + 10y_2 + x_3^2,$$
$$f_1^{(2)}(\mathbf{x},\mathbf{y}) = x_1^2 + x_2^2 - x_3^2 + y_1^2, \quad f_2^{(2)}(\mathbf{x},\mathbf{y}) = -(x_1+x_2)y_1 + 10y_2 - x_3^2,$$

where, $-20 \le (x_1, x_2, x_3) \le 20$; $0.5 \le (y_1, y_2) \le 4$. The objectives of P_1 needs to be minimized and P_2 needs to be maximized. Identical parameter setting to that in the tug-of-war problem is used here. One aspect is clear: since x_3 comes as a positive quadratic form in P_1 and in negative quadratic form in P_2, it is intuitive that $x_3 = 0$ will for all PO solutions. With $x_3 = 0$, $f_i^{(1)} = f_i^{(2)}$ (for $i = 1, 2$), but since P_1 and P_2 objectives are to minimized and maximized, respectively, they will provide a competing scenario within the MOCoEv optimization. Moreover, smaller absolute values of x_1 and x_2 will be better for $f_1^{(1)}$, but worse for $f_2^{(1)}$, thereby indicating that each problem will constitute a PO front.

First, the mean fitness aggregation is used for both populations. Figure 6a displays mean fitness values of each population computed with all members of the other population. The distribution of mean fitness values of a specific $\mathbf{x}^{(1)} = (12.6201, 12.2150, 0.0230)$ due to variation in $\mathbf{Z}^{(2)}$ is marked with dark colored circles. A similar plot is shown in Figure 6b with $\mathbf{y}^{(2)} = (2.5714, 3.9999)$. In Figure 6a, every near-vertical line describes evaluation of a specific member



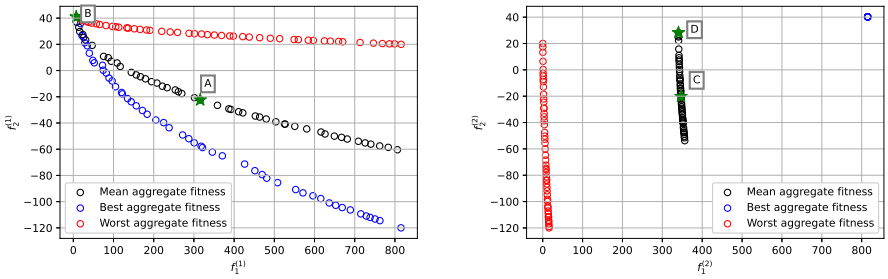
(a) Population 1. (b) Population 2.

Fig. 6: Evaluation score of middle point of Pareto front and all evaluations of Generation 50. Optimized using mean performance.

of Population 1 and all members of Population 2. It is observed that all PO solutions of both populations, $y_2 = 4$ (upper bound). This variable is directly controlled by P_2. Since a higher value of y_2 makes $f_2^{(2)}$ better and y_2 does not appear in $f_1^{(2)}$, our MOCoEv decides to fix y_2 at its maximum allowable value. We also observe that y_1 values are distributed among population members within its entire range $[0.5, 4.0]$. Since y_1 makes a trade-off between $f_1^{(2)}$ and $f_2^{(2)}$ as shown in their expressions, it is also expected, provided the term $(x_1 + x_2)$ is non-negative. Since P_1 controls the $\mathbf{x}$-vector, it is clear that a positive value of $(x_1 + x_2)$ will dominate a negative value. we observe that both x_1 and x_2 is well distributed within $[0, 20]$. With the above details, it is clear that the limits of $f_1^{(1)}$ on the PO front will be $[0.5^2, 20^2 + 20^2 + 4^2]$ or [0.25, 816], which matches with the figure. On a similar effort, $f_2^{(1)}$ lies in $[-120, 40]$.

The PO front with the mean aggregation fitness functions is shown in Figure 7a. The respective PO front for Population 2 is shown in Figure 7b with black circles. Despite a near-vertical nature of the fitness values, there exists a trade-off in these points.

To compare the PO fronts with the best and worst fitness aggregation functions, we rerun MOCoEv algorithm with identical parameter setting and present the final points in Figure 7. It is clear that the mean aggregation fitness vectors are sandwiched in between the best and worst aggregation fitness vectors. It is interesting that when best case aggregation fitness is used, Population 2 converges to a single solution. It can be seen from Figure 6b that the maximum $f_2^{(2)}$ is always 40 for all $\mathbf{y}$-vectors. The value comes from $10y_2$ term only in $f_2^{(2)}$, thereby choosing the $x_1 = x_2 = 0$ solution from P_1 and by forcing $y_1 = 4$ to maximize $f_1^{(2)}$.



(a) Population 1 PO fronts. (b) Population 2 PO fronts.

Fig. 7: PO fronts for the numerical problem NP.

4.4 Decision-making for Problem NP

First, we apply the pseudo-weight approach to a single preferred solution from both populations. Equation 4 is used to compute pseudo-weight vector and compared with a desired weight vector of (0.5,0.5). The process finds the following ND solutions from two populations: $\mathbf{x}^{(1)} = (12.6201, 12.2150, 0.0230)$, $\mathbf{y}^{(2)} = (2.5714, 3.9999)$. This point is marked as A and C on the mean aggregation fitness plots in Figure 7.

Next, we follow the minimum sensitivity approach and finds solutions B and D as the most preferred ones for two problems. The solutions are given as: $\mathbf{x}^{(1)} = (-0.2155, -0.1699, 0.0231)$, $\mathbf{y}^{(2)} = (0.5003, 3.9999)$. and also marked in mean aggregate fitness plot in Figure 6a. It is clear that this happens for the minimum $f_1^{(1)}$ solution.

The above shows how a systematic approach of starting with finding a number of PO solutions using the proposed MOCoEv algorithm and then using a decision-making strategy to choose a single preferred solution can be obtained in a multi-objective co-evolutionary problem.

5 Conclusions

This paper has dealt with developing a multi-objective co-evolutionary (MOCoEv) optimization algorithm involving two interacting populations co-evolving for more than one conflicting objectives of their own. The motivation for pursuing this study has stemmed from recent surge in interests in solving adversarial network design problems and 'arms-race' problems which involve a defender and an attacker who clearly have more than one goals of protecting and harming a system. The proposed MOCoEv algorithm is applied on two proof-of-principle problems. The first problem is an existing two-player game with a clear idea of compromised strategies. The second problem is a numerical problem, developed in this study, to also test proposed MOCoEv algorithm's performance on explainable outcomes. In addition, we have proposed two multi-criterion decision-making principles which can be applied to a MOCoEv problem to eventually choose a single preferred solution for each population. Extended methods have also been suggested to demonstrate how a preferred solution can be chosen when some knowledge of decision-making of the second population is available, to make the approach more pragmatic. Future research could investigate scalable test problems and potential performance metrics for this class of problems.

To the best of our knowledge, this study stays as one of the few studies on simultaneous multi-objective solution of co-evolutionary problems. Our results in this paper have shown that MOCoEv problem solving is quite a challenging task, but extensions of evolutionary multi-objective optimization methods may make MOCoEv problem solving tractable and attractive. The proposed algorithm can now be applied to GAN, MOGs, and other co-evolutionary problems.

References

1. 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)
2. Deb, K.: Multi-objective optimization using evolutionary algorithms. Wiley, Chichester, UK (2001)
3. Deb, K., Agrawal, R.B.: Simulated binary crossover for continuous search space. Complex Systems **9**(2), 115–148 (1995)
4. 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)
5. Eisenstadt, E., Moshaiov, A.: Decision-making in non-cooperative games with conflicting self-objectives. Journal of Multi-Criteria Decision Analysis **25**(5-6), 130–141 (2018)
6. Eisenstadt, E., Moshaiov, A., Avigad, G.: Co-evolution of strategies for multi-objective games under postponed objective preferences. In: 2015 IEEE conference on computational intelligence and games (CIG). pp. 461–468. IEEE (2015)
7. Eisenstadt-Matalon, E., Avigad, G., Moshaiov, A., Branke, J.: Rationalizable strategies in multi-objective games under undecided objective preferences (2016)
8. Eisenstadt-Matalon, E., Moshaiov, A.: Mutual rationalizability in vector-payoff games. In: International Conference on Evolutionary Multi-Criterion Optimization. pp. 593–604. Springer (2019)
9. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 27, pp. 2672–2680. Curran Asso. (2014)
10. Li, X., Yao, X.: Cooperatively coevolving particle swarms for large scale optimization. IEEE Transactions on Evolutionary Computation **16**(2), 210–224 (2012)
11. Matalon-Eisenstadt, E., Moshaiov, A., Avigad, G.: The competing travelling salespersons problem under multi-criteria. In: International Conference on Parallel Problem Solving from Nature. pp. 463–472. Springer (2016)
12. Miettinen, K.: Nonlinear Multiobjective Optimization. Kluwer, Boston (1999)
13. Nguyen, A., Yosinski, J., Clune, J.: Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In: IEEE Conference on Computer Vision and Pattern Recognition (2015)
14. Paredis, J.: Coevolutionary constraint satisfaction. In: Parallel Problem Solving from Nature III (PPSN-III). pp. 46–55 (1994)
15. Żychowski, A., Gupta, A., Mańdziuk, J., Ong, Y.S.: Addressing expensive multi-objective games with postponed preference articulation via memetic co-evolution. Knowledge-Based Systems **154**, 17–31 (2018)