Factored Particle Swarm Optimization for Policy Co-training in Reinforcement Learning

Kordel K. France
Johns Hopkins University
Baltimore, MD, USA
kfrance8@jhu.edu

John W. Sheppard
Montana State University
Bozeman, MT, USA
john.sheppard@montana.edu

ABSTRACT

Uncertainty of the environment limits the circumstances with which any optimization problem can provide meaningful information. Multiple optimizers can combat this problem by communicating different information through cooperative coevolution. In reinforcement learning (RL), uncertainty can be reduced by applying learned policies collaboratively with another agent. Here, we propose policy Co-training with Factored Evolutionary Algorithms (CoFEA) to evolve an optimal policy for such scenarios. We hypothesize that self-paced co-training can allow factored particle swarms with imperfect knowledge to consolidate knowledge from each of their imperfect policies in order to approximate a single optimal policy. Additionally, we show how the performance of co-training swarms of RL agents can be maximized through the specific use of Expected SARSA as the policy learner. We evaluate CoFEA against comparable RL algorithms and attempt to establish limits for which our procedure does and does not provide benefit. Our results indicate that Particle Swarm Optimization (PSO) is effective in training multiple agents under uncertainty and that FEA reduces swarm and policy updates. This paper contributes to the field of cooperative coevolutionary algorithms by proposing a method by which factored evolutionary techniques can significantly improve how multiple RL agents collaborate under extreme uncertainty to solve complex tasks faster than a single agent can under identical conditions.

CCS CONCEPTS

• **Computing methodologies** → **Learning from implicit feedback**; **Semi-supervised learning settings**; **Cooperation and coordination**; **Multi-agent reinforcement learning**.

KEYWORDS

factored evolutionary algorithms, co-training, reinforcement learning

ACM Reference Format:
Kordel K. France and John W. Sheppard. 2023. Factored Particle Swarm Optimization for Policy Co-training in Reinforcement Learning. In *Genetic and Evolutionary Computation Conference (GECCO '23), July 15–19, 2023, Lisbon, Portugal.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3583131.3590376

GECCO '23, July 15–19, 2023, Lisbon, Portugal

ACM ISBN 979-8-4007-0119-1/23/07.
https://doi.org/10.1145/3583131.3590376

1 INTRODUCTION

The phrase "soldiers win skirmishes; armies win wars" suggests that individuals operating with a unifying objective can surmount much more complex challenges than if each individual acted alone. In computer science, this element of collaboration is illustrated well with multi-agent systems and multi-population evolutionary algorithms in being able to collaboratively solve an optimization problem. Collaborative problem solving implies having multiple agents observe the task in order to offer different angles of insight. The exchange of different knowledge is what partially manifests a solution to the problem faster than if each individual attempted to solve it alone. Especially under extreme uncertainty or when each individual possesses incomplete knowledge of the problem, collaboration is not only a convenience but a necessity. This is the primary motivation of this paper in which we propose a method for collaborative problem solving under extreme uncertainty.

We illustrate our method by considering a scenario of robotic navigation where a swarm of reinforcement learning agents assesses how to navigate from one end of a maze to another. They do not know where the goal state is, what obstacles are forthcoming, and cannot observe past the next single state. However, they can communicate with one another and must navigate through collaboration. One result of our study also reveals that Expected SARSA is particularly effective in a collaborative setting, most likely because of the smoothing that occurs across the policies. We hypothesize that Factored Evolutionary Algorithms (FEA) can provide increased speed and efficiency at which policies are learned for a Markov decision process (MDP) within swarms when evaluated against a comparable single-particle agent in conditions of uncertainty. Additionally, we hypothesize that factored evolutionary algorithms [23] combined with self-paced multi-view co-training (SPaMCo) [13, 14] can improve performance further by facilitating the use of pseudo-rewards and more efficient inter- and intra-swarm communication. Finally, we hypothesize that these methods can be especially advantageous in predicting actions for several states ahead given only the knowledge of the current occupied state. In what follows, we denote our method as CoFEA (Co-training with Factored Evolutionary Algorithms) and demonstrate that CoFEA can result in a higher cumulative reward and fewer policy updates when evaluated against conventional single-agent policy learners.

The contributions of this work can be summarized as follows:

(1) We develop an algorithm for using FEA with PSO as the optimizer on a given task, where each particle within the swarm is a separate reinforcement learning agent.
(2) We show how to improve performance by incorporating factored evolution and/or self-paced co-training.

(3) We demonstrate that our algorithm results in a larger cumulative reward under high uncertainty by leveraging Expected SARSA and comparing its performance to greedy Q-learning.
(4) We evaluate the potential of our algorithm to predict actions for several next states by demonstrating why co-training and pseudo-rewards can allow our algorithm to infer rewards for longer state-action trajectories.

We unify these contributions into a single method denoted as (Policy) Co-training for Factored Evolutionary Algorithms (CoFEA).

2 BACKGROUND

Our research is motivated primarily by the works of [13] and [19]. First, we extend the concept presented in [13] in which they utilize self-paced co-training to train classifiers on two or more different views and exchange pseudo labels of unlabeled instances. Effectively, they present a method called self-paced co-training (SPaMCo) that allows classifies unlabeled data based on minimal training data. SPaCo focuses exclusively on classification problems, and we consider extending their ideas to accommodate Markov decision processes and reinforcement learning. Specifically, we analyze state-action-reward triplets, where each state-action pair is sampled from a map whose rewards are mostly unobservable since the rewards for most of the state-action pairs are unknown or masked. We then learn the rewards and apply "pseudo-rewards" to unobservable states instead of "pseudo-labels" to unlabeled instances. In the context of multi-view co-training, a "view" in classification may be interpreted like an "observation" in an MDP.

In [19], Pillai and Sheppard define a method for allowing swarms to "overlap", which was shown to improve performance over backpropagation in training deep neural networks. They demonstrate that factoring the search space into subproblems promotes better swarm communication in what they define as overlapping swarm intelligence (OSI). Our work attempts to demonstrate how factoring the search space in this manner can be advantageous with co-training a single policy in reinforcement learning. Here, we leverage differential grouping for factoring, which is a decomposition method that can reveal interactions between decision variables, and use the resulting groups to define a structure for subcomponents such that dependence between them is minimized [18].

We use nearly identical notation to [19] when discussing factored PSO. Much of the same notation from [13] and SPaMCo can also be directly leveraged, but there are some adjustments needed to both the process and methodology in order to map from classification to reinforcement learning. We maintain the idea of self-paced regularization and attributing weights to each sample, except in our case, a "sample" is synonymous with a state-action pair, which we refer to as a "trajectory." The self-paced aspect of co-training is also maintained; however, the greediness ϵ of the reinforcement learning algorithm shall be a function of the self-paced hyperparameter λ_{sp}. Expected SARSA and Q-learning are the specific RL algorithms trained by co-training. Sutton and Barto demonstrate in [25] that, by allowing the greediness to be tuned throughout policy learning, Expected SARSA converges to Q-learning by the end of the training process; a formal proof of this can be found in [26]. We leverage this principle by increasing greediness ϵ in direct proportion to λ_{sp}. Additionally, we focus on a tabular-based approach here versus one based on neural networks as a means to demonstrate that the performance observed is correlated with CoFEA and not due to some radical network architecture.

3 RELATED WORK

With CoFEA, our intention is to investigate a derivative of Ma's 2017 [14] and 2020 [13] work with self-paced multi-view co-training (SPaMCo) but within the context of FEA-optimized Markov decision processes. Our analysis of related work contains references to algorithms that ultimately act as the building blocks each of FEA and SPaMCo, but to our knowledge, no prior work exists surrounding the exact methodology proposed in this paper.

3.1 Reinforcement Learning

Under the reinforcement learning umbrella exist imitation learning and inverse reinforcement learning. In their paper, Abbeel and Ng define inverse reinforcement learning (IRL) as the problem of deriving a reward function from observed behavior of a supposed "expert" [1] Conversely, Huang et al. define imitation learning (IL) as deriving a policy from demonstrations of expert behavior [11]. While parallels of each can be applied to the method proposed here, both IL and IRL suggest that the model to be imitated demonstrates perfect expert behavior, whereas the behavior of our co-trained policy is unobservable and, therefore, cannot qualify as an expert to imitate. IRL requires traditional model-based reinforcement learning in order to approximate the reward function, and imitation learning can be reduced to the same traditional methods for deterministic agents as demonstrated in [8]. Later we show how CoFEA can be retuned to accommodate both IL and IRL as subroutines.

3.2 Co-training and Co-regularization

The methodology behind co-training was shown to be tractable in 1998 by Blum and Mitchell under the initial assumption that instances from different views are independent contingent on the co-trainer's classifier making useful predictions on unlabeled data [3]. Brefeld and Scheffer implemented a co-training algorithm combining a naïve Bayes classifier with a support vector machine, improving upon prior work accomplished by Nigam and Ghani [4, 17].

Later attempts to provide a more resilient co-regularization function were published in [21] and [28], where both attempt to encode prediction dependencies among views into a single co-regularization term. The resulting objective function was difficult to optimize and orthogonal to the fundamental approach of co-training. Many attempts to harmonize technical nuances behind co-training up to this point leveraged only two-shot cases and gave unclear performance measures. Dai et al. published work on using pseudo labels and abductive learning to improve classification of unlabeled data that prefaced this strategy with neighboring graphs [9]. SPaCo was proposed by Ma et al. that established a more generalizable objective function and self-paced learning technique over a pseudo-supervised co-training algorithm [14]. However, this model only complied with two-view cases. Building on top of this work, Ma et al. incorporated additional methods into SPaCo that allowed for resiliency against false negative samples, accommodated multi-view cases, and improved on co-regularization that

manifested into what eventually became SPaMCo [13]. SPaMCo introduced co-training such that it could be used beyond the two-view scenario. Similar works to SPaMCo followed with [12] showing multi-view co-training in clustering that built upon the works of [32]. Zheng et al. demonstrated similar effects with image segmentation, among others [33]. Soon after, Wang et al. introduced the concept of self-paced and self-consistent co-training over image segmentation [27].

3.3 Co-training Theory

From our research, works attempting to apply co-training with multi-view or self-paced aspects to reinforcement learning have been scarce. Wang et al. show how two "dueling" models can achieve strong performance collaboratively, but they do not use co-training and suggest splitting the state-value and action functions, which is not the intent here [29]. Akella and Lin demonstrate how co-training can be used to train a reinforcement learning agent on when to select an action by learning a temporal policy [2]. Wu et al. propose training a Q-learning agent to learn a policy for data selection and then exploit that policy to train co-training classifiers automatically [31]. In this context, RL was used as a means to train a supervisor over a classification problem, but an RL-based agent was not trained with co-training itself. The work of Song et al. demonstrates how co-training can be used to enable an RL agent to learn a policy in settings with multiple state-action representations [22]. This work seems to align closely with the research proposed here in regard to RL being combined with co-training, but they do not incorporate the self-paced element that we do. Additionally, they demonstrate learning a co-trained policy to learn a policy more efficiently, not as a technique for learning policies whose domains have little known data. While they did use a Q-learning algorithm, they did not consider Expected SARSA, PSO, or FEA.

3.4 Factored Evolutionary Algorithms

Factored evolutionary algorithms (FEA) are cooperative co-evolutionary algorithms that create overlapping subpopulations (i.e., the "factors") that optimize over a subset of variables for a common objective function. These subpopulations can be considered as subproblems of the primary optimization function. FEA was defined formally by Strasser et al., where they introduce the importance in selecting a factor architecture [23]; however, they do so by extending the original definition proposed in [19] and [10] on overlapping swarm intelligence (OSI). In their 2011 work, Pillai and Sheppard demonstrate the effectiveness of OSI in training artificial neural networks by allowing each swarm to represent a unique path starting at an input node and ending at an output node. A global view of the neural network evaluated using a common vector of weights that is maintained across all swarms. This vector of weights is created by combining the weights of the particles from each swarm that attain the highest fitness. Our work most closely resonates with the OSI model. Pillai and Sheppard also showed that OSI outperformed several alternative cooperative co-evolutionary PSO-based methods in addition to backpropagation. Distributed OSI (DOSI) was Defined by Fortier et al. in which swarms were allowed to communicate values to facilitate competition with one another. Butcher et al. then extended this work by illustrating a concept of information sharing and conflict resolution through an actor model that could be accomplished through this communication via Pareto improvements [6, 7]. Both [15] and [27] apply PSO to reinforcement learning, but they do not leverage the factored approach.

4 COFEA

CoFEA is defined as a set of swarms of policy learners collaboratively minimizing regret toward a single unifying policy, where Muthukumar defines regret as the loss between the reward received under the current policy versus the reward received by the action taken under the optimal policy [16]. The set of swarms (each of which homogeneously contains either Expected SARSA or Q-learner agents) are trained iteratively through the exchange and refinement of predicted pseudo rewards assigned to state-action pairs (i.e., trajectories). Note that, although there are multiple swarms and multiple agents per swarm, all are constructing and updating the same policy.

4.1 Self-Paced Learning

Let the ground truth reward for the i^{th} trajectory be denoted as y_i where $i \in [0, N]$ and N is the number of trajectories available in the observable domain. Similarly, let the input state-action sequence—or trajectory—be denoted as $\phi_i \in \Phi$, the vector of swarm model parameters be denoted as $\theta \in \Theta = \{\theta^{(1)}, \theta^{(2)}, ..., \theta^{(n)}\}$, and g denote the function that estimates the pseudo reward. Let the loss function that calculates the cost between the ground truth reward and the estimated reward be denoted as $L(y_i, g(\phi_i, \theta, \epsilon))$, where $g(\phi_i, \theta, \epsilon)$ denotes the estimated reward as a function of the input sequence, model parameters, and greediness. Let $f(v, \lambda, \epsilon)$ denote the self-paced regularization function where v relates to the latent weights of unrewarded instances, λ is the age parameter that moderates learning within the self-paced model, and ϵ is the current greediness of the policy.

The goal, then, is to optimize the following:

$$\min_{\theta, v \in [0,1]^n} E(\theta, v; \lambda; \epsilon) = \sum_{i=0}^{N} (v_i L(y_i, g(\phi_i, \theta)) + f(v_i, \lambda, \epsilon)) \quad (1)$$

Note that the self-paced regularization term for the j^{th} observation of the i^{th} trajectory sample can be written as $\lambda^{(j)} v_i^{(j)}$ as in [14]. The adjustment of the age parameter λ coupled with the joint learning of the model parameters and the latent weights allow for the model to moderate the number of training samples during each iteration of the algorithm as a function of sample complexity. This self-paced model allows the loss between pseudo-rewards and true rewards to be minimized through the tuning of samples that are observed for training.

4.2 Soft Co-Regularization

CoFEA weights each unlabeled training instance to coordinate learning order; weights are added to rewarded sequences after enough sequences have been assigned rewards. These weights help determine the pseudo rewards to apply to sequences during training. The policy realized via co-training learns by analyzing two observations of the same state-action sequence and exchanging the pseudo rewards associated with that sequence. A soft co-regularization

term harmonizes this to aid the self-paced learning capability of the model. Different observations maintain common knowledge of the confidence in the pseudo rewards assigned to samples. Due to this, the inner product of the weights for both observations enforces the weight of the reward penalizing the loss of one trajectory observation that is similar to another. This co-regularization ensures that the learned reward estimates from multiple observations of unrewarded trajectories are consistent throughout training, while simultaneously working to moderate the complexity of observations trained.

We define a co-regularization procedure that normalizes rewards between two different observations. The co-regularization term is denoted as R for the p^{th} and q^{th} observation, where $p, q \in [1, 2, .., N]$, $q = N_u - p$, $p < q$, N defines the total number of state-action sequences, and N_u denotes the total number of unrewarded state-action sequences. Then R is defined as:

$$R(V) = \gamma_{sp} \sum_{p=1}^{N_u} \left(\mathbf{v}^{(p)} - \mathbf{v}^{(q)}\right)^\top \left(\mathbf{v}^{(p)} - \mathbf{v}^{(q)}\right) \tag{2}$$

where $v^{(p)}, v^{(q)} \in V$, $\mathbf{v}^{(p)}$ contains all the weights of unrewarded sequences in the p^{th} observation, and γ_{sp} denotes the degree to which one observation's pseudo-rewards influence the other observation's pseudo-rewards. The self-paced learning element of SPaMCo allows the model to grow its confidence in reward predictions incrementally with each successive iteration, allowing it to learn with increasingly complex sequences. The selection for proper values of the age parameter λ shall be defined in Section 5 to demonstrate the ability of self-paced learning to handle sequences with noisy rewards.

4.3 Expected SARSA

Q-learning is an off-policy temporal difference (TD) algorithm that learns to approximate the optimal action-value function directly, independent of the policy being followed [30]. It learns a policy that selects the next state-action pairs that produces the maximum reward. It is one of the most commonly used TD algorithms in reinforcement learning due to its computational efficiency and good approximation of the total cumulative reward.

Recall that CoFEA is training a set of swarms of policy learners iteratively through the back-and-forth exchange of pseudo-rewarded trajectories. The rate at which pseudo-rewards are applied to trajectories is initially highly erratic; one can effectively consider rewards as being assigned to state-action pairs at random. Intuitively, an agent seeking the average reward will be able to approximate the hidden true reward of an environment better than an agent that is always seeking the maximum reward, such as a Q-learner. The maximum reward sought is erroneous when training begins because there is low confidence in the assignment of rewards, so a Q-learner is maximizing a lossy policy while an Expected SARSA learner [25] is approximating it—it is effectively minimizing its losses. The regularization terms defined above do provide some protection against this, but they are not entirely effective.

In most state-of-the art reinforcement learning research where the environment can be observed directly, Q-learning and its variants strike a balance between performance and computational cost against other TD algorithms. However, based on our experiments, we have evidence to suggest that a Q-learner's performance is contingent on the degree to which it is able to observe the *true* rewards for its actions confidently. If there is a high degree of confidence that the rewards being observed match those of ground truth, a greedy Q-learner is expected to have superior performance. For these reasons, we hypothesize Expected SARSA to be the better performing RL algorithm for CoFEA and co-training in general, where the update rule for Expected SARSA is as follows.

$$Q^*(s,a) = Q(s,a) + \alpha \left[R(s,a,s') + \frac{\gamma}{n} \sum_{i=1}^{n} Q(s_i', a_i') - Q(s,a) \right]$$

As Ma et al. demonstrated in [13, 14] with image classification, we show that, as confidence in reward assignment grows through successive iterations, CoFEA becomes more accurate in correctly attributing rewards to state-action trajectories. Therefore, the Expected SARSA policy learner may be allowed to become more greedy. As ϵ approaches 1, Sutton and Barto [25] show it begins to select the action that results in the highest cumulative reward and evolves into exactly Q-learning. Expected SARSA will allow the algorithm to take advantage of both worlds by moderating the learning of its policy as a function of correct reward assignment and the self-paced learning parameter λ. In this fashion, early iterations of CoFEA will begin with a policy that approximates SARSA,

$$Q^*(s,a) = Q(s,a) + \alpha[R(s,a,s') + \gamma Q(s',a') - Q(s,a)]$$

moderating the high loss, and then evolve into Q-learning

$$Q^*(s,a) = Q(s,a) + \alpha \left[R(s,a,s') + \gamma \max_{a'} Q(s',a') - Q(s,a) \right]$$

as the policy becomes more refined. This also affords the algorithm to be used as both an on-policy and off-policy learner.

After careful examination, one can see that the only difference that Expected SARSA possesses over Q-learning is the averaging of the rewards in place of their maximization. Due to this, Expected SARSA is more computationally expensive because the expected reward is being calculated at each update state. However, as explained above, it is also more accurate and robust in co-training since it eliminates the early volatility in reward estimation. Especially in the beginning of co-training when the pseudo-rewards of unrewarded trajectories is highly erroneous, Expected SARSA can provide smoother learning. Since the objective is to infer rewards for unobservable states accurately, it may be that the increase in computational complexity is justified. Expected SARSA affords us the freedom to fluctuate between on-policy and off-policy methods while providing a natural means of controlling reward loss. For our experiments, we allow ϵ in Expected SARSA to increase as a function of the self-paced learning parameter.

4.4 CoFEA Algorithm

The CoFEA algorithm uses a loss function between the ground truth reward and the estimated reward and adds it to the self-paced learning model presented in Equation 1. Two regularization terms are also added: the co-regularization term from Equation 2, and a regularization term for all swarms' model parameters $\theta^{(1)}, \ldots, \theta^{(n_s)} \in \Theta$. In its entirety, the objective function for CoFEA can be

defined as:

$$\begin{aligned}\min_{\theta,\mathbf{v}} E(\theta,\mathbf{v};\lambda;\epsilon) &= \sum_{j=1}^{n}\sum_{i=1}^{N_r} L(y_i, g^{(j)}(\phi_i^{(j)};\theta^{(j)},\epsilon)) \\ &+ \sum_{j=1}^{n}\sum_{k=N_r+1}^{N_r+N_u} (\mathbf{v}_k^{(j)} L(y_k, g^{(j)}(\phi_i^{(j)};\theta^{(j)},\epsilon)) \\ &\quad -\lambda^{(j)}\mathbf{v}_k^{(j)}) \\ &- \gamma_{sp}\sum_{p=1}^{N_u}(\mathbf{v}^{(p)}-\mathbf{v}^{(q)})^\top(\mathbf{v}^{(p)}-\mathbf{v}^{(q)}) \\ &+ \frac{1}{n}\sum_{j=1}^{n}\|\theta^{(j)}\|_n \end{aligned} \tag{3}$$

where $g(\phi_i;\theta,\epsilon)$ denotes the reward received by the policy of the corresponding swarm as a function of the trajectory ϕ_i, model parameters θ, and greediness ϵ for each agent.

5 OPTIMIZATION STRATEGY

The routine for optimizing the loss of CoFEA is defined in Algorithm 1. While the algorithm generates two observations—one trajectory observation per swarm—we describe the steps of the algorithm in terms of one observation for the ease of interpretation since both agents are optimized in an identical manner. In other words, Algorithm 1 describes the procedure each of the two RL agents follows in order to exchange policy information with the other.

The algorithm first initializes the input instances $\phi^{(1)}$, ..., $\phi^{(n_s)}$ $\in \Phi$ as unrewarded trajectories; it assumes the RL agents (the particles) within the set of swarms of policy learners, $\alpha^{(1)}, \ldots, \alpha^{(n_s)}$, are also appropriately initialized. We need a small amount of unrewarded data to begin training each swarm so we initialize the age parameters $\lambda^{(1)}, \ldots, \lambda^{(n_s)} \in \Lambda$ to very small values to allow only a few unrewarded trajectories into the training pool. For the age parameters, we found values of 0.3–0.5 provide a robust training pace that generalizes well. Unless otherwise stated, we elect 0.3 as the age parameter. We set each of the weight vectors $\mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(n_s)}$ as zero vectors and then proceed with training the two swarms over the selected instances to acquire the initial losses over both unrewarded and rewarded trajectories.

From here, we begin co-training by selecting trajectories that we are confident are rewarded in accordance to the optimal policy. Rewarded instances with loss values less than $\lambda^{(n_s)}$ from the observations of $\alpha^{(n_s)}$ are viewed as confidently rewarded trajectories that should be selected for the training pool for the next co-training iteration. In a similar manner, we then select which unrewarded trajectories will be added into the training pool for each $\alpha^{(n_s)}$ by sampling from values whose loss is greater than that of the largest $\lambda^{(i)}$. If the age parameters are tuned appropriately and the self-paced nature of the training is correctly controlled, trajectories selected to the training pool of, for example, $\alpha^{(1)}$, will maintain a higher probability of being selected for the training pool of agent $\alpha^{(2)}$ and vice-versa.

The model parameter vectors $\theta^{(1)}, \ldots, \theta^{(n_s)}$ are updated next to reflect new knowledge from the previous iteration. These weights are updated by optimizing the RL policy via either standard PSO or

Algorithm 1 CoFEA Optimization Algorithm

Input: samples $\left[\phi_1^{(1)}, \ldots, \phi_{N_r+N_u}^{(1)}\right], \left[\phi_1^{(n_s)}, \ldots, \phi_{N_r+N_u}^{(n_s)}\right]$,
rewards $y_1, \ldots, y_{N_r}$, parameters $\lambda^{(1)}, \ldots, \lambda^{(n_s)}$, and γ_{sp}
Output: $\mathbf{v}^{(1)}, ..., \mathbf{v}^{(n_s)}$
Initialize $\left[\phi^{(1)}, ..., \phi^{(n_s)}\right]$, $\left[\lambda^{(1)}, \ldots, \lambda^{(n_s)}\right]$, and γ_{sp}
Update $v^{(1)}, ..., v^{(n)}$
training_iter = 1
while not converge || training_iter < max_iter **do**
  for $j \leftarrow 1 : n_s$ **do**
    Initialize $\theta^{(j)}, \mathbf{u}^{(j)}$: agent weights, params $\in$ jth obs
    $\theta^{(j)}, \mathbf{u}^{(j)} \leftarrow$ **PSO**
    if do factorization **then**
      $\theta^{(j)}, u^{(j)} \leftarrow$ **Factored PSO**: weights of RL agent
    end if
    Update $\mathbf{v}_k^{(n_s+1-j)}$: Prepare confident trajectories
    Update $\mathbf{v}_k^{(j)}$: Add unrewarded trajectories to pool
    Update $\theta^{(j)}, \mathbf{u}^{(j)}$: Train on pool of j^{th} observation
    Update y_k : Find unrewarded pseudo rewards
    Augment $\lambda^{(1)}, ..., \lambda^{(n_s)}$
  end for
end while
Return $\mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(n_s)}$

Factored PSO (shown in Algorithm 2). If Factored PSO is selected, we determine the factors through overlapping differential grouping as defined in [20] and [24]. We select this factorization method due to its ability to produce consistent results within our experiments and leave the assessment of other factorization techniques as opportunity for future work. Factoring the state space provides an additional heuristic upon which the swarm can be organized to learn the policy. The updated weights are used to evaluate the pseudo labels for the j^{th} observation. One can interpret Algorithm 1 as optimizing the assignment of pseudo-rewards and PSO or Factored PSO as optimizing the policy being learned by each swarm given new knowledge obtained from those pseudo rewards.

By updating the model parameter vectors $\theta^{(1)}, \ldots, \theta^{(n_s)}$, we are effectively training a model from truly rewarded and pseudo-rewarded samples. Note that this model is separate from the set of swarms of policy learners $\alpha^{(1)}, \ldots, \alpha^{(n_s)}$. The model here is a separate learner that is being trained to perform binary classification over rewarded and pseudo-rewarded trajectories. In fact, Ma et al. show that the SPaCo model degenerates to simple support vector machine (SVM) optimization by setting the loss function to hinge loss. We do not provide the formal proof of this here, but by direct extension of their work, we can derive the following SVM optimization model from Equation 3:

$$\min_{\theta^{(i)}\in[0,1]} \frac{1}{n}\|\theta^{(i)}\|_n + \sum_{j=1}^{N_r} L_j^{(i)} + \sum_{k=N_r+1}^{N_r+N_u} \mathbf{v}_k^{(i)} L_k^{(i)}$$

where $L_t^{(i)} = L(y_t, g^{(i)}(\phi_t^{(i)};\theta^{(i)},\epsilon)),\ t = 1, \ \ldots\ , \ N_u + N_r$. Following the update of model parameters, we update the pseudo-rewards of each trajectory. We can do so by computing each reward through

Algorithm 2 Factored Particle Swarm Optimization

Input: $\theta^{(j)}, \mathbf{u}^{(j)}$ network weights, parameters from j^{th} obs.
Output: $\theta^{(j)}, \mathbf{u}^{(j)}$

```
Initialize population as the network weights
gvn ← u^(j)
Create and initialize particles of each swarm s_i in population S
for all s_i ∈ S do
    for all x_{i,j} ∈ s_i do
        Construct sub-policy p_{i,j}
        Evaluate particle fitness f(p_{i,j})
        Assign particle x_{i,j} fitness, f(x_{i,j}) = f(p_{i,j})
        if f(x_{i,j}) < f(pbest_{i,j}) then
            pbest_{i,j} = x_{i,j}
        end if
        Evaluate global fitness f(gvn)
        if f(x_{i,j}) < f(gvn) then
            Update sharedgvn_{i,j}
        end if
        Update velocity for each particle x_{i,j}
        r ← Random(0, 1)
        if r < σ_acc then
            v_{i,j} = 0
        end if
        if r > σ_acc then
            v_{i,j} = ωv_{i,j} + c1 r1 (p_{i,j} − x_{i,j}) + c2 r2 (pbest_{i,j} − x_{i,j})
        end if
        x_{i,j} = x_{i,j} + v_{i,j}
    end for
end for
Update θ^(j)
u^(j) ← P_j
Return updated θ^(j), u^(j)
```

solving the following minimization problem:

$$y_k = \underset{y_k}{\operatorname{argmin}} \sum_{i=1}^{n_s} \mathbf{v}_k^{(i)} L(y_k, g^{(i)}(\phi_k^{(i)}; \theta^{(i)}))$$

Finally, we marginally increment each of the age parameters, $\lambda^{(1)}$, ..., $\lambda^{(n_s)}$, in order to allow for more unrewarded trajectories to participate in policy training. The process above is performed for each model at each iteration until there are no more unrewarded trajectories of which to assign pseudo-rewards, the maximum amount of training iterations has been reached, or (ideally) the optimal policy to the goal has been found.

6 EXPERIMENTS

To evaluate the performance of CoFEA, we conducted several experiments in a simulated environment. Each experiment corresponds to a two-dimensional gridworld-style navigation task. There are five tasks: the Cliff Walking problem from [25]; the Frozen Lake problem from [5][1]; and three custom *racetrack* tasks that represent additional navigational routes in the shapes of letters *L*, *R*, and *O* of which we denote as *L-track*, *R-track* and *O-track*, respectively. The former two problems represent discrete state spaces while the latter three represent continuous state spaces. Graphical representations of these tasks are given in Appendix C of the Supplementary Material. For each task, particles are placed at the starting position *S* and then cooperate to find a direct route to the finish position *F*. For the Cliff Walking and Frozen Lake tasks, we follow their default specifications regarding rewards. For the racetrack problems, each state transition decrements the reward by 1 while a collision with boundaries of the track result in an immediate reset to the start position; boundary collisions are determined by the Bresenham algorithm and reaching the goal receives a reward of 100.

To allow some practicality to the real world, each particle has a velocity and an acceleration that are used to define its position within the map. For adjusting position, speed, and velocity, we incorporate fundamental kinematic principles to each particle and swarm update. In order to address a further level of stochasticity, we establish a parameter σ_{acc} that denotes a probability for which acceleration of the particle fails such that the particle does not perform a velocity update as expected. We evaluate each single-agent control over both its deterministic (accelerator works every time) and stochastic (accelerator fails $100 * \sigma_{acc}$% of the time) behaviors; deterministic versions are marked with a (D) suffix in each table. Each algorithm is run 20 times per task and their results averaged.

We use a non-decaying learning rate of 0.1 and a discount factor of 0.8 for each RL agent. In evaluating the influence of swarm size on the performance of CoFEA, we assess swarm sizes of 4, 8, and 16. Each time a particle is updated, it performs a single episode within the environment with 10 steps. We allow only a single episode to occur because this allows each particle to update the policy in a controlled manner and thus maintain a global consensus of sorts. In general, we see the best performance with 2–5 co-training iterations and, when applicable, the number of FEA iterations is equivalent to the number of co-training iterations. As a means of control, we perform all experimental runs involving co-training with 3 co-training and FEA iterations. Furthermore, we limit the number of generations for evolution to 1000 across all variants of CoFEA. For the control, the maximum number of episodes is limited to 500 with a step limit of 100. In other words, there is a maximum of 500,000 policy updates allowed. For all other algorithms, each particle is allowed a single episode with a step limit of 10 on the *update* subroutine of either PSO or Factored PSO.

We capture three primary metrics to assess the validity of our experiments in evaluating our hypotheses: (1) the cumulative reward achieved by the policy, (2) the total number of swarm updates, and (3) the average number of policy updates per particle needed to reach the goal. We evaluate each task with Expected SARSA and Q-learning and use their single-agent variants as the experimental control. We compare the performance of this control to the performance of swarms of Expected SARSA and Q-learning agents each evaluated over the same tasks but as part of (a) a PSO routine, (b) a factored PSO routine, (c) a PSO routine with co-training, and (d) a factored PSO routine with co-training. Finally, we evaluate the effects of learning extended trajectories in each of these four scenarios, totaling eight experiments per task. In each scenario, we denote a "particle" as a single RL agent of either Expected SARSA

[1] As they are defined in [25] and [5], the Cliff Walking and Frozen Lake problems contain state spaces of 48 and 64 respectively. These state space sizes are too small to be of relative interest here, so we scale each of these problems up to 16x their original size, keeping all other dynamics of the problem unchanged.

or Q-learning type. [2] In this context, the control can be interpreted as a single-particle "swarm" since there is only one agent learning the policy with which to solve each task. We consider there to be sufficient evidence to support our hypotheses if any of these four variations result in a higher cumulative reward than the control, less swarm updates, less policy updates per particle than the control, or any combination of the three.

7 RESULTS

Results for Expected SARSA are shown in Table 1 while analogous results for Q-learning are shown in Table 2. For each task, we observe that each of the four target algorithms consistently construct a policy that solves the task in fewer total policy updates in comparison to the Control, regardless of whether Expected SARSA or Q-learning are used. For each algorithm, Wilcoxon Signed-Rank tests were conducted against the control, and it was observed that p-values less than 0.05 were achieved when comparing both the rewards and cumulative number of policy updates. For all but one scenario on the L-track, we see that results from Table 1 show Expected SARSA to complete tasks with fewer total policy and swarm updates than equivalent Q-learning tasks in Table 2. We observe that Expected SARSA achieves an equivalent or greater cumulative reward over every task in comparison to those of Q-learning, with one exception in each the L-track and O-track racetrack tasks. We observe that, for both Q-learning and Expected SARSA, PSO achieves fewer policy updates than the Control. In all but only a few cases with the R-track racetrack task, FEA completes the task with even fewer iterations than unfactored PSO, and with fewer swarm updates. The O-track is the most complex task with respect to the average number of policy updates needed to find the optimal policy, and we note that the single-agent Control for Q-learning did not converge within 500,000 iterations for this task; we mark this task with an asterisk in Table 2 as a result.

Tables 3 and 4 show results over each task for two additional FEA iterations. In every task, this resulted in fewer policy and swarm updates than the 3-iteration variant. In addition, p-values less than 0.05 were observed when comparing the rewards, policy updates, and swarm updates of all corresponding Expected SARSA and Q-learning algorithms indicating statistically significant evidence to support Expected SARSA as having strategic benefit when coupled with FEA. This applied to both the 3- and 5-iteration FEA variants. However, in only 60% of scenarios, the additional FEA iterations provided a higher or equivalent reward for Expected SARSA, while this was true for 70% of scenarios for Q-learning. P-values less than 0.05 were not achieved when comparing the differences between 3-iteration and 5-iteration Expected SARSA. This also held for Q-learning indicating that, while the introduction of FEA into training suggests improved performance, the additional iterations may not produce statistically significant results.

Tables 5–14 in Appendix A of the supplementary material present the effects of different swarm sizes, and we see that the performance

[2] For these experiments, one may consider the swarm as "homogeneous" due to identical reinforcement learning agent types among particles. In other words, each swarm will either have all particles as Expected SARSA agents or all particles as Q-learning agents, but will not include both simultaneously. While preliminary experimentation suggests that a heterogeneous swarm has potential to facilitate encouraging results, we leave this as an opportunity for future work.

Table 1: CoFEA Results – Expected SARSA

Algorithm	Task	# Swarm	# Policy	Reward
Control (D)	Cliff	N/A	156,107	-327
Control	Cliff	N/A	156,007	-331
PSO	Cliff	2877	2793	-9
PSO+FEA	Cliff	542	281	-12
PSO+Co	Cliff	3822	8911	-5
PSO+FEA+Co	Cliff	2700	3062	-6
Control (D)	Lake	N/A	469,292	-1004
Control	Lake	N/A	463,722	-992
PSO	Lake	7994	19,974	-43
PSO+FEA	Lake	3993	19,264	-54
PSO+Co	Lake	10,650	26,639	-25
PSO+FEA+Co	Lake	10,691	13,420	-5
Control (D)	L-track	N/A	53,695	-164
Control	L-track	N/A	51,437	-143
PSO	L-track	6947	12,655	-36
PSO+FEA	L-track	3606	9717	-18
PSO+Co	L-track	14,343	24,494	-17
PSO+FEA+Co	L-track	14,237	9612	-11
Control (D)	R-track	N/A	158,390	-698
Control	R-track	N/A	162,063	-728
PSO	R-track	23,771	59,996	-21
PSO+FEA	R-track	12,021	60,112	-5
PSO+Co	R-track	31,992	79,818	-10
PSO+FEA+Co	R-track	31,974	39,624	-3
Control (D)	O-track	N/A	227,580	-1001
Control	O-track	N/A	224,022	-971
PSO	O-track	37,721	99,974	-62
PSO+FEA	O-track	13,328	50,055	-9
PSO+Co	O-track	53,333	133,329	-21
PSO+FEA+Co	O-track	53,312	66,740	-6

comparisons observed in Tables 1 and 2 above are consistent but with slightly lower rewards and policy updates and fewer swarm updates in general. We expect the effects of swarm size to be more prominent for larger problems and leave this as a primary interest of future work. Tables 15 and 16 in Appendix B of the supplementary material present the effects of varying trajectory length. For tasks such as Cliff Walking and L-track, where minimal direction changes are needed to evaluate a complete policy, we see that performance increase is most noticeable. For Frozen Lake, R-track, and O-track, we observe that co-training shows the highest gain in performance for both Expected SARSA *and* Q-learning with the exception of the PSO+Co-training variant of the O-track and Frozen Lake tasks. This suggests that the use of pseudo-rewards in co-training may aid in estimating the reward horizon under ambiguity.

Tables 17 and 19 in Appendix A of the supplementary material show that co-training 3 swarms produced higher reward for 19 of 20 tasks in comparison to co-training 2 swarms, with 75% of these metrics resulting in fewer average policy updates per particle. Similarly, in tables 18 and 19, we demonstrate how increasing the number of co-training swarms to 4 further improved the total cumulative reward in 17 of 20 tasks, albeit with only 45% of these tasks resulting in fewer policy updates. Those tasks using Expected SARSA that did not improve with additional swarms *did* improve

Table 2: CoFEA Results – Q-Learning

Algorithm	Task	# Swarm	# Policy	Reward
Control (D)	Cliff	N/A	166,801	-398
Control	Cliff	N/A	166,246	-406
PSO	Cliff	1401	6639	-15
PSO+FEA	Cliff	159	1839	-36
PSO+Co	Cliff	5463	13,107	-4
PSO+FEA+Co	Cliff	3153	3638	-5
Control (D)	Lake	N/A	110,289	-457
Control	Lake	N/A	111,981	-485
PSO	Lake	7994	19,603	-42
PSO+FEA	Lake	3961	17,858	-46
PSO+Co	Lake	10,656	26,621	-21
PSO+FEA+Co	Lake	10,648	13,350	-25
Control (D)	L-track	N/A	52,021	-240
Control	L-track	N/A	54,646	-240
PSO	L-track	6615	14,342	-36
PSO+FEA	L-track	3518	11,298	-20
PSO+Co	L-track	15,627	20,393	-18
PSO+FEA+Co	L-track	15,423	11,832	-24
Control (D)	R-track	N/A	493,825	-1054
Control	R-track	N/A	498,672	-1058
PSO	R-track	23,997	59,997	-22
PSO+FEA	R-track	12,109	99,773	-9
PSO+Co	R-track	32,048	79,997	-18
PSO+FEA+Co	R-track	31,991	40,097	-18
Control (D)	O-track	N/A	500,000*	-1064
Control	O-track	N/A	500,000*	-1063
PSO	O-track	40,006	100,042	-67
PSO+FEA	O-track	19,992	99,773	-9
PSO+Co	O-track	53,337	144,381	-33
PSO+FEA+Co	O-track	53,401	77,820	-19

using Q-learning, and vice-versa, which may suggest that a heterogeneous mixture of both Expected SARSA and Q-learning agents within each swarm could further refine performance.

We acknowledge the fact that, although there is statistical evidence that FEA and co-training may result in fewer total policy updates with higher rewards, there is indeed added complexity. Factored evolution acts as an excellent method for decomposing the underlying search space organically, and co-training proves to be an excellent heuristic for learning sparse rewards. Together, they allow for distributed and parallel RL, but their cooperation entails several more operations per iteration versus a simple single-agent model. We recognize the room this provides for optimizing our work and leave this as an area of future research.

8 CONCLUSION

In consideration of our results, we find evidence to support our hypothesis regarding the advantage of Expected SARSA in comparison to Q-learning under conditions of high uncertainty. Additionally, we see strong evidence supporting the use of PSO for optimizing multiple agents in constructing a single policy with fewer cumulative updates than a single agent. Furthermore, we find that evolutionary factorization of the search space can generally result in even fewer swarm and cumulative policy updates than one that is unfactored. With additional swarms added to the task, we see even fewer cumulative updates and higher total rewards for both factored and unfactored PSO when co-training 2, 3, and 4 swarms. Finally, we show evidence to suggest that benefits compound when coupling co-training with Expected SARSA and Factored PSO for trajectories of varying length.

Table 3: CoFEA Results – Expected SARSA, 5 FEA Iterations

Algorithm	Task	# Swarm	# Policy	Reward
PSO+FEA	Cliff	391	1317	-11
PSO+FEA+Co	Cliff	2794	1943	-5
PSO+FEA	Lake	2399	12,112	-8
PSO+FEA+Co	Lake	11,333	9096	-5
PSO+FEA	L-track	2198	7018	-18
PSO+FEA+Co	L-track	10,054	6088	-11
PSO+FEA	R-track	2400	12,111	-6
PSO+FEA+Co	R-track	11,330	9090	-4
PSO+FEA	O-track	12,000	60,112	-10
PSO+FEA+Co	O-track	5664	45,097	-6

Table 4: CoFEA Results – Q-learning, 5 FEA Iterations

Algorithm	Task	# Swarm	# Policy	Reward
PSO+FEA	Cliff	69	86	-49
PSO+FEA+Co	Cliff	3444	2446	-5
PSO+FEA	Lake	1601	6066	-4
PSO+FEA+Co	Lake	11,324	8961	-25
PSO+FEA	L-track	2122	5867	-20
PSO+FEA+Co	L-track	11,021	7836	-24
PSO+FEA	R-track	1598	6154	-6
PSO+FEA+Co	R-track	11,440	9074	-20
PSO+FEA	O-track	8000	3070	-10
PSO+FEA+Co	O-track	56,667	44,704	-18

Our results with CoFEA establish a foundation from which parallel and distributed reinforcement learning is enabled through cooperative coevolution, but it is by no means comprehensive. We acknowledge that the tasks we investigate regard only the domain of navigation, but emphasize that the success we demonstrate here lays the necessary groundwork for extrapolation to other domains such as computer vision and natural language processing. In future work, we intend to improve the performance of CoFEA in such domains and evaluate more complex tasks to further demonstrate reasoning about uncertainty. Opportunities to replicate CoFEA's performance over network-based learning, as well as assessing the performance of different factor architectures as defined in [23], also exist. Our work is motivated by the interest in training autonomous agents to act collaboratively under uncertainty and we hope that CoFEA inspires other coevolutionarymethods within this context. Source code for CoFEA is available from the authors upon request.

ACKNOWLEDGMENTS

We wish to thank members of the Numerical Intelligent Systems Laboratory at Montana State University for their assistance as this work unfolded. We especially thank Dr. Amy Peerlinck for providing access to her FEA code. Finally, we thank the administration in the Johns Hopkins University Engineering for Professionals program for their willingness to support research among students in their professional-based graduate program.

REFERENCES

[1] Pieter Abbeel and Andrew Y. Ng. 2004. Apprenticeship Learning via Inverse Reinforcement Learning. In *Proceedings of the Twenty-First International Conference on Machine Learning* (Banff, Alberta, Canada) *(ICML '04)*. Association for Computing Machinery, New York, NY, USA, 1.

[2] Ashlesha Akella and Chin-Teng Lin. 2021. Time and Action Co-Training in Reinforcement Learning Agents. *Frontiers in Control Engineering* 2 (2021).

[3] Avrim Blum and Tom Mitchell. 1998. Combining Labeled and Unlabeled Data with Co-Training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory* (Madison, Wisconsin, USA) *(COLT' 98)*. Association for Computing Machinery, New York, NY, USA, 92–100.

[4] Ulf Brefeld and Tobias Scheffer. 2004. Co-EM Support Vector Learning. In *Proceedings of the Twenty-First International Conference on Machine Learning* (Banff, Alberta, Canada) *(ICML '04)*. Association for Computing Machinery, New York, NY, USA.

[5] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. OpenAI Gym. *CoRR* abs/1606.01540 (2016). arXiv:1606.01540 http://arxiv.org/abs/1606.01540

[6] Stephyn Butcher and John Sheppard. 2018. An Actor Model Implementation of Distributed Factored Evolutionary Algorithms. In *Proceedings of the GECCO Workshop on Parallel and Distributed Evolutionary Inspired Methods* (Kyoto, Japan). 1276–1283.

[7] Stephyn G. W. Butcher, John W. Sheppard, and Shane Strasser. 2018. Information Sharing and Conflict Resolution in Distributed Factored Evolutionary Algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference* (Kyoto, Japan) *(GECCO '18)*. Association for Computing Machinery, New York, NY, USA, 5–12.

[8] Kamil Ciosek. 2022. Imitation Learning by Reinforcement Learning. In *International Conference on Learning Representations*.

[9] Wang-Zhou Dai, Qiuling Xu, Yang Yu, and Zhi-Hua Zhou. 2019. Bridging Machine Learning and Logical Reasoning by Abductive Learning. In *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, Inc.

[10] Brian K. Haberman and John W. Sheppard. 2012. Overlapping Particle Swarms for Energy-Efficient Routing in Sensor Networks. *Wirel. Netw.* 18, 4 (may 2012), 351–363.

[11] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, Gabriel Dulac-Arnold, John Agapiou, Joel Z. Leibo, and Audrunas Gruslys. 2018. Deep Q-Learning from Demonstrations. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence* (New Orleans, Louisiana, USA) *(AAAI'18/IAAI'18/EAAI'18)*. AAAI Press.

[12] Zongmo Huang, Yazhou Ren, Xiaorong Pu, Lili Pan, Dezhong Yao, and Guoxian Yu. 2021. Dual self-paced multi-view clustering. *Neural Networks* 140 (2021), 184–192.

[13] Fan Ma, Deyu Meng, Xuanyi Dong, and Yi Yang. 2020. Self-paced Multi-view Co-training. *Journal of Machine Learning Research* 21, 57 (2020), 1–38.

[14] Fan Ma, Deyu Meng, Qi Xie, Zina Li, and Xuanyi Dong. 2017. Self-Paced Co-training. In *Proceedings of the 34th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 70)*, Doina Precup and Yee Whye Teh (Eds.). PMLR, 2275–2284.

[15] Syed Irfan Ali Meerza, Moinul Islam, and Md. Mohiuddin Uzzal. 2019. Q-Learning Based Particle Swarm Optimization Algorithm for Optimal Path Planning of Swarm of Mobile Robots. In *2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT)*. 1–5.

[16] Vidya Muthukumar. 2020. *Learning from an unknown environment.* Ph. D. Dissertation. EECS Department, University of California, Berkeley.

[17] Kamal Nigam and Rayid Ghani. 2000. Analyzing the Effectiveness and Applicability of Co-Training. In *Proceedings of the Ninth International Conference on Information and Knowledge Management* (McLean, Virginia, USA) *(CIKM '00)*. Association for Computing Machinery, New York, NY, USA, 86–93.

[18] Mohammad Nabi Omidvar, Xiaodong Li, Yi Mei, and Xin Yao. 2014. Cooperative Co-Evolution With Differential Grouping for Large Scale Optimization. *IEEE Transactions on Evolutionary Computation* 18, 3 (2014), 378–393.

[19] Karthik Ganesan Pillai and John W. Sheppard. 2011. Overlapping swarm intelligence for training artificial neural networks. In *2011 IEEE Symposium on Swarm Intelligence*. 1–8.

[20] Elliott Pryor, Amy Peerlinck, and John Sheppard. 2021. A Study in Overlapping Factor Decomposition for Cooperative Co-Evolution. In *Proceedings of the IEEE Swarm Intelligence Symposium (SIS)* (Orlando, FL).

[21] Vikas Sindhwani, Partha Niyogi, and Mikhail Belkin. 2005. A Co-Regularization Approach to Semi-supervised Learning with Multiple Views.

[22] Jialin Song, Ravi Lanka, Yisong Yue, and Masahiro Ono. 2019. Co-training for Policy Learning. *CoRR* abs/1907.04484 (2019). arXiv:1907.04484 http://arxiv.org/abs/1907.04484

[23] Shane Strasser and John W. Sheppard. 2017. Convergence of Factored Evolutionary Algorithms. In *Proceedings of the 14th ACM/SIGEVO Conference on Foundations of Genetic Algorithms* (Copenhagen, Denmark) *(FOGA '17)*. Association for Computing Machinery, New York, NY, USA, 81–94.

[24] Yuan Sun, Xiaodong Li, Andreas Ernst, and Mohammad Nabi Omidvar. 2019. Decomposition for Large-scale Optimization Problems with Overlapping Components. In *2019 IEEE Congress on Evolutionary Computation (CEC)*. 326–333.

[25] Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction* (second ed.). The MIT Press.

[26] Harm van Seijen, Hado van Hasselt, Shimon Whiteson, and Marco Wiering. 2009. A theoretical and empirical analysis of Expected Sarsa. In *2009 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*. 177–184.

[27] Feng Wang, Xujie Wang, and Shilei Sun. 2022. A Reinforcement Learning Level-Based Particle Swarm Optimization Algorithm for Large-Scale Optimization. *Inf. Sci.* 602, C (jul 2022), 298–312.

[28] Jiao Wang, Siwei Luo, and Yan Li. 2010. A Multi-view Regularization Method for Semi-supervised Learning. In *Advances in Neural Networks - ISNN 2010*, Liqing Zhang, Bao-Liang Lu, and James Kwok (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 444–449.

[29] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot, and Nando De Freitas. 2016. Dueling Network Architectures for Deep Reinforcement Learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48* (New York, NY, USA) *(ICML'16)*. JMLR.org, 1995–2003.

[30] Christopher Watkins. 1989. *Learning from Delayed Rewards.* Ph. D. Dissertation. Department of Computer Science, Kings College, Cambridge University.

[31] Jiawei Wu, Lei Li, and William Yang Wang. 2018. Reinforced Co-Training. *CoRR* abs/1804.06035 (2018). arXiv:1804.06035 http://arxiv.org/abs/1804.06035

[32] Chang Xu, Dacheng Tao, and Chao Xu. 2015. Multi-View Self-Paced Learning for Clustering. In *Proceedings of the 24th International Conference on Artificial Intelligence* (Buenos Aires, Argentina) *(IJCAI'15)*. AAAI Press, 3974–3980.

[33] Xu Zheng, Chong Fu, Haoyu Xie, Jialei Chen, Xingwei Wang, and Chiu-Wing Sham. 2022. Uncertainty-Aware Deep Co-Training for Semi-Supervised Medical Image Segmentation. *Comput. Biol. Med.* 149, C (Oct 2022).

SUPPLEMENTARY MATERIAL

A EXPERIMENTAL RESULTS

A.1 Varying Swarm Size

Results for varying swarm sizes for each algorithm are shown below. We evaluate swarm sizes of 2, 4, 8, 16, and 32 particles. Each result was run with 3 FEA iterations (if applicable), 3 co-training iterations, and 1000 generations, over the specified swarm size.

Table 5: CoFEA Results - Expected SARSA, Particles = 2

Algorithm	Task	# Swarm	# Policy	Reward
PSO	Cliff	3196	14,769	-4
PSO+FEA	Cliff	569	3841	-6
PSO	Lake	7987	39,968	26
PSO+FEA	Lake	3996	40,203	-4
PSO	L-track	6992	29,651	-17
PSO+FEA	L-track	3648	22,993	-9
PSO	R-track	8000*	40,000*	-11
PSO+FEA	R-track	2663	20,130	-5
PSO	O-track	40,000*	200,000*	-32
PSO+FEA	O-track	19,992	199,162	-5

Table 6: CoFEA Results - Q-Learning, Particles = 2

Algorithm	Task	# Swarm	# Policy	Reward
PSO	Cliff	340	440	-25
PSO+FEA	Cliff	165	604	-18
PSO	Lake	7989	39,366	-29
PSO+FEA	Lake	3982	40,207	-2
PSO	L-track	6724	25,971	-19
PSO+FEA	L-track	3587	19,639	-10
PSO	R-track	40,000*	8000*	-10
PSO+FEA	R-track	3898	39,892	-3
PSO	O-track	40,000*	200,000*	-32
PSO+FEA	O-track	13,328	100,110	-5

Table 7: CoFEA Results - Expected SARSA, Particles = 4

Algorithm	Task	# Swarm	# Policy	Reward
PSO	Cliff	577	769	-577
PSO+FEA	Cliff	116	134	-49
PSO	Lake	8000*	19,378	-58
PSO+FEA	Lake	3996	20,102	-4
PSO	L-track	6652	12,482	-40
PSO+FEA	L-track	3590	9306	-18
PSO	R-track	8000*	19,991	-21
PSO+FEA	R-track	3996	20,102	-5
PSO	O-track	40,000*	100,000*	-65
PSO+FEA	O-track	13,328	50,055	-10

Table 8: CoFEA Results - Q-Learning, Particles = 4

Algorithm	Task	# Swarm	# Policy	Reward
PSO	Cliff	525	621	-33
PSO+FEA	Cliff	141	239	-40
PSO	Lake	8000*	19,398	-56
PSO+FEA	Lake	2664	10,065	-4
PSO	L-track	6371	10,800	-37
PSO+FEA	L-track	3571	9083	-20
PSO	R-track	8000*	20,000*	-19
PSO+FEA	R-track	2664	10,064	-6
PSO	O-track	40,000*	98,126*	-64
PSO+FEA	O-track	13,338	50,053	-9

Table 9: CoFEA Results - Expected SARSA, Particles = 8

Algorithm	Task	# Swarm	# Policy	Reward
PSO	Cliff	2740	3166	-20
PSO+FEA	Cliff	620	1066	-24
PSO	Lake	7989	9896	-95
PSO+FEA	Lake	3986	10,050	-17
PSO	L-track	7013	7364	-68
PSO+FEA	L-track	3697	5933	-34
PSO	R-track	8000*	10,003	-48
PSO+FEA	R-track	3976	10,051	-14
PSO	O-track	40,000*	50,019	-128
PSO+FEA	O-track	19,993	50,041	-21

Table 10: CoFEA Results - Q-Learning, Particles = 8

Algorithm	Task	# Swarm	# Policy	Reward
PSO	Cliff	519	260	-68
PSO+FEA	Cliff	130	111	-89
PSO	Lake	8000*	9828	-115
PSO+FEA	Lake	2664	5033	-8
PSO	L-track	6408	5610	-75
PSO+FEA	L-track	3541	4927	-37
PSO	R-track	8000*	9997	-34
PSO+FEA	R-track	2664	5033	10
PSO	O-track	40,000*	50,029	-117
PSO+FEA	O-track	13,328	25,028	-20

Table 11: CoFEA Results - Expected SARSA, Particles = 16

Algorithm	Task	# Swarm	# Policy	Reward
PSO	Cliff	541	142	-139
PSO+FEA	Cliff	139	117	-122
PSO	Lake	7998	4874	-235
PSO+FEA	Lake	3996	5026	-17
PSO	L-track	6456	2853	-151
PSO+FEA	L-track	2188	1329	-90
PSO	R-track	8000*	5000*	-72
PSO+FEA	R-track	3996	5025	-22
PSO	O-track	40,000*	25,000*	-48
PSO+FEA	O-track	13,328	12,514	-44

Table 12: CoFEA Results - Q-Learning, Particles = 16

Algorithm	Task	# Swarm	# Policy	Reward
PSO	Cliff	412	81	-165
PSO+FEA	Cliff	114	25	-197
PSO	Lake	8000	4854	-241
PSO+FEA	Lake	2664	2516	-16
PSO	L-track	6262	2758	-149
PSO+FEA	L-track	3553	2419	-83
PSO	R-track	8000*	4999	-76
PSO+FEA	R-track	2664	2515	-23
PSO	O-track	40,000*	24,806	-250
PSO+FEA	O-track	13,328	12,513	-40

Table 13: CoFEA Results - Expected SARSA, Particles = 32

Algorithm	Task	# Swarm	# Policy	Reward
PSO	Cliff	3203	936	-76
PSO+FEA	Cliff	743	328	-87
PSO	Lake	7989	2497	-409
PSO+FEA	Lake	3997	2513	-69
PSO	L-track	7092	1853	-303
PSO+FEA	L-track	2364	861	-176
PSO	R-track	8007	2500	-165
PSO+FEA	R-track	3996	2512	-56
PSO	O-track	40,003	12,501	-513
PSO+FEA	O-track	19,992	12,510	-74

Table 14: CoFEA Results - Q-Learning, Particles = 32

Algorithm	Task	# Swarm	# Policy	Reward
PSO	Cliff	391	25	-348
PSO+FEA	Cliff	126	25	-359
PSO	Lake	7999	2464	-476
PSO+FEA	Lake	2664	1258	-36
PSO	L-track	6793	1686	-337
PSO+FEA	L-track	3570	1230	-141
PSO	R-track	8007	2503	-158
PSO+FEA	R-track	3987	2517	-38
PSO	O-track	40,048	12,503	-488
PSO+FEA	O-track	13,328	6257	-87

A.2 Extended Trajectories

We evaluate the performance of each CoFEA variant to construct a policy with trajectories of extended length. We denote the trajectory length as τ where a $\tau = 1$ indicates the prediction of a reward for a single state given the next action, a value of $\tau = 2$ indicates the prediction of the next two rewards for a single state given the next two actions, and so on. Each result was run with 3 FEA iterations (if applicable), 3 co-training iterations, 1000 generations, and swarm sizes of 4 particles.

Table 15: CoFEA Results - Expected SARSA, $\tau = 2$

Algorithm	Task	# Swarm	# Policy	Reward
PSO	Cliff	7814	19,329	-6
PSO+FEA	Cliff	2664	10,065	-4
PSO+Co	Cliff	31,795	26,428	-2
PSO+FEA+Co	Cliff	10,573	13,323	-2
PSO	Lake	7986	19,991	-43
PSO+FEA	Lake	2664	10,065	-5
PSO+Co	Lake	31,999	26,665	-24
PSO+FEA+Co	Lake	13,320	20,112	-4
PSO	L-track	7937	19,744	-35
PSO+FEA	L-track	3994	20,038	-8
PSO+Co	L-track	31,997	26,647	-8
PSO+FEA+Co	L-track	13,319	20,107	-3
PSO	R-track	8004	19,983	-38
PSO+FEA	R-track	3996	20,103	-8
PSO+Co	R-track	31,999	26,687	-9
PSO+FEA+Co	R-track	13,991	21,783	-4
PSO	O-track	39,994	99,801	-49
PSO+FEA	O-track	19,992	100,094	-11
PSO+Co	O-track	159,937	133,334	-16
PSO+FEA+Co	O-track	63,308	91,754	-6

Table 16: CoFEA Results - Q-Learning, $\tau = 2$

Algorithm	Task	# Swarm	# Policy	Reward
PSO	Cliff	7831	19,327	-6
PSO+FEA	Cliff	2620	9862	-4
PSO+Co	Cliff	31,489	26,183	-2
PSO+FEA+Co	Cliff	10,569	13,304	-2
PSO	Lake	7995	19,978	-44
PSO+FEA	Lake	3996	20,103	-7
PSO+Co	Lake	31,998	26,664	-24
PSO+FEA+Co	Lake	11,988	16,765	-4
PSO	L-track	7963	19,788	-37
PSO+FEA	L-track	3990	19,976	-9
PSO+Co	L-track	32,004	26,647	-7
PSO+FEA+Co	L-track	14,650	23,454	-3
PSO	R-track	8007	19,984	-37
PSO+FEA	R-track	3998	20,141	-7
PSO+Co	R-track	32,003	26,668	-9
PSO+FEA+Co	R-track	13,986	21,785	-4
PSO	O-track	39,995	99,793	-50
PSO+FEA	O-track	19,993	100,083	-11
PSO+Co	O-track	159,966	133,194	-25
PSO+FEA+Co	O-track	56,621	75,031	-18

A.3 Multi-Swarm Co-training

For co-training, we evaluate the effects of leveraging greater than 2 swarms for the exchange and evolution of pseudo-rewards. Each result was run with 3 FEA iterations (if applicable), 3 co-training iterations, 1000 generations, and swarm sizes of 4 particles.

Table 17: CoFEA Results - Expected SARSA, 3 swarms

Algorithm	Task	# Swarm	# Policy	Reward
PSO+Co	Cliff	3782	8826	-3
PSO+FEA+Co	Cliff	2697	3085	-4
PSO+Co	Lake	10,652	26,641	-17
PSO+FEA+Co	Lake	10,656	13,719	-4
PSO+Co	L-track	9616	20,669	-12
PSO+FEA+Co	L-track	9484	9548	-7
PSO+Co	R-track	10,771	26,668	-8
PSO+FEA+Co	R-track	10,657	13,419	-3
PSO+Co	O-track	10,667	26,677	-20
PSO+FEA+Co	O-track	10,656	13,420	-7

Table 18: CoFEA Results - Expected SARSA, 4 swarms

Algorithm	Task	# Swarm	# Policy	Reward
PSO+Co	Cliff	4032	9456	-2
PSO+FEA+Co	Cliff	2556	2885	-3
PSO+Co	Lake	10,643	26,637	-13
PSO+FEA+Co	Lake	10,564	13,319	-3
PSO+Co	L-track	9693	20,951	-9
PSO+FEA+Co	L-track	9454	9524	-5
PSO+Co	R-track	10,666	26,689	-6
PSO+FEA+Co	R-track	10,659	13,424	-2
PSO+Co	O-track	10,670	26,701	-15
PSO+FEA+Co	O-track	10,662	13,442	-5

Table 19: CoFEA Results - Q-learning, 3 swarms

Algorithm	Task	# Swarm	# Policy	Reward
PSO+Co	Cliff	4893	11,697	-3
PSO+FEA+Co	Cliff	3633	4264	-3
PSO+Co	Lake	10,677	26,635	-14
PSO+FEA+Co	Lake	10,648	13,340	-17
PSO+Co	L-track	10,412	24,326	-12
PSO+FEA+Co	L-track	10,349	11,899	-16
PSO+Co	R-track	10,667	26,668	-7
PSO+FEA+Co	R-track	10,654	13,423	-2
PSO+Co	O-track	10,775	26,758	-14
PSO+FEA+Co	O-track	10,650	13,400	-13

Table 20: CoFEA Results - Q-learning, 4 swarms

Algorithm	Task	# Swarm	# Policy	Reward
PSO+Co	Cliff	3799	8842	-2
PSO+FEA+Co	Cliff	2663	3028	-3
PSO+Co	Lake	10,653	26,638	-13
PSO+FEA+Co	Lake	10,655	13,382	-3
PSO+Co	L-track	9681	20,877	-9
PSO+FEA+Co	L-track	9479	9664	-5
PSO+Co	R-track	10,666	26,724	-6
PSO+FEA+Co	R-track	10,687	13,419	-2
PSO+Co	O-track	10,674	26,773	-15
PSO+FEA+Co	O-track	10,672	13,534	-5

B IMAGES OF RACETRACK ENVIRONMENTS

We provide further clarification for the three *Racetrack* environments evaluated by CoFEA since they are non-standard RL environments. Simplified graphical representations for each of the three environments is illustrated below. For each environment, "." indicates a vacant position, "#" denotes a penalized boundary, "S" denotes a starting position, and "F" denotes a finish/goal position.

B.1 L-track

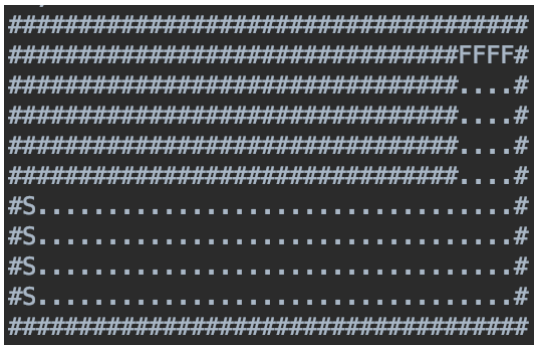

Figure 1: A simplified representation of the *L-track* environment

B.2 R-track

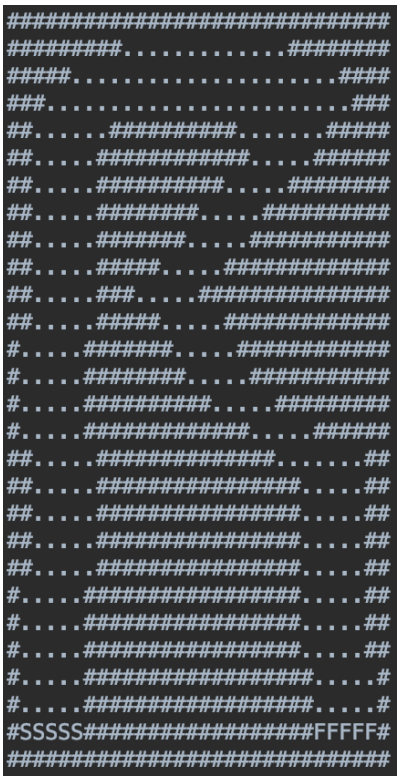

Figure 2: A simplified representation of the *R-track* environment

B.3 O-track

Figure 3: A simplified representation of the *O-track* environment