# Surrogate-Based Scalarization for Behavioral Adjustment in Actor-Critic Algorithms

1st Jeremy Kedziora
*Department of Computer Science and Software Engineering*
*Milwaukee School of Engineering*
Milwaukee, WI USA
kedziora@msoe.edu

2nd Patrick Marshall
*Department of Computer Science and Software Engineering*
*Milwaukee School of Engineering*
Milwaukee, WI USA
marshallp@msoe.edu

***Abstract*—In this paper, we extend the concept of reward decomposition and masking for AI control to policy-based agents, specifically within the framework of reinforcement learning. By utilizing the Masked A2C ensemble method, we explore how multiple, independent policy networks can be trained on different components of reward decomposition, aggregating them through an ensemble approach. This paper discusses both the theoretical background and practical implementation, focusing on how policy aggregation with masked policies can enhance performance in environments with complex strategic interactions, like board games such as Stratego. The use of reward decomposition enables the agent to separately handle different components of the reward function, adjusting its behavior for a more controlled learning process. Additionally, we discuss the challenges of training an ensemble of policies and how off-policy corrections are applied to ensure stable learning.**

***Index Terms*—Policy Masking, Reward Decomposition, Ensemble Methods, Masked A2C, Policy-based Reinforcement Learning**

## I. Introduction

Reinforcement learning (RL) agents are typically trained to maximize a single, scalar reward signal that encodes multiple—and often competing—objectives. In complex environments, this scalarization can obscure the contribution of individual behavioral factors and limit a user's ability to shape or constrain an agent's conduct. As a result, an RL agent may learn highly effective but undesirable strategies, such as exploiting loopholes in the reward structure, acting in an unethical fashion, or prioritizing actions that yield short-term gains but are weakly related to long-term goals.

Multi-objective RL and, specifically, reward decomposition offers a way to restore interpretability and control by explicitly modeling the environment's reward as a collection of distinct components, each reflecting a separate aspect of performance or behavior. **However, most prior applications of decomposition have focused on value-based methods, where additive value functions are trained for each component and later combined. Extending this paradigm to policy-based learning—where policies are updated through gradient estimates rather than explicit value aggregation—remains an open challenge.**

In this work, we introduce and study an Actor–Critic algorithm that models specific agent behaviors using reward decomposition along behavioral lines, ensemble learning and scalarization-based aggregation. We term this model Behavioral Actor-Critic (BAC). In BAC, each decomposed component of the reward function is paired with its own actor–critic learner, producing an ensemble of specialized policies. These individual policies are combined through a policy aggregator, which determines the final action distribution as a function of user-defined scalarization weights. Adjusting these weights enables post-training modification of the agent's behavior—effectively a behavioral control knob for balancing objectives such as efficiency, safety, and aggressiveness.

Use of aggregation to determine the final action distribution makes BAC an off-policy algorithm; to maintain stable learning under this modular, ensemble-driven structure, we employ a correction mechanism based on importance sampling ratios derived from the aggregator distribution. Furthermore, because users may not know in advance which scalarization best balances interpretability and performance, we apply surrogate optimization to search for and identify training scalarizations that balance agent performance with post-training flexibility.

We demonstrate this approach in a simplified version of the board game Stratego, an adversarial domain that requires long-horizon planning and behavioral trade-offs between task-completion and aggressiveness. Our results illustrate that BAC can produce agents with distinct, interpretable behavioral tendencies and that modifying scalarization weights after training can predictably adjust agent strategy without retraining.

The main contributions of this paper are:

- BAC, a policy-based framework that explicitly models the balance between multiple objectives via ensemble learning and an aggregation mechanism based on linear scalarization;
- An off-policy correction scheme ensuring stable actor–critic updates under dynamic behavior policies; and
- A surrogate optimization procedure that identifies scalarizations maximizing post-training behavioral controllability and robustness.

## II. Related Work

The approach studied in this paper is an ensemble actor-critic method with scalarized rewards. As such, it leverages multiple strands of research in the RL literature including ensemble methods, multiple-objective RL, and off-policy learning.

Decomposition of reward streams is a key technique analyzed in multi-objective RL (MORL, e.g. [7]). In MORL the goal is typically to either: 1) train a single policy under linear (scalarized expected returns) or non-linear (expected scalarized returns) conditions if the decomposition is known *a priori* (e.g. [1],[6],[18]); or 2) learn a set of policies covering a specific solution set (e.g. the Pareto Front, [15]) if the decomposition is only known *a posteriori*. These methods have been extended to policy-based settings

This paper represents a hybrid approach in which we do not know the appropriate decomposition aggregation ahead of time but still seek to learn a single policy that is relatively robust to different possible decompositions adopted during inference.

Reward decomposition has also been used in hierarchical RL (e.g. [10], [4],[**shu2018**],[25],[22]), as a means for expediting learning (e.g. [21], [8]), for explainability (e.g. [9]), and for managing exploration/exploitation tradeoffs (e.g. [2]). Our work differs from this in that we seek to combine reward decomposition with masking as a means to shape agent behavior by altering the contribution of reward dimensions associated with specific action-level characteristics. While prior works have explored learning action masks on a per-state basis [27] to adjust agent behavior, we are not aware of any such approach employed on reward action-level characteristics.

We take the Actor-Critic family of policy-based RL algorithms (e.g. [12], [16]) as our starting point, which combines a parameterized policy adjusted by gradient estimates (e.g. [30], [29]) with a learned long-term value estimate to reduce variance.

Ensemble methods such as Bootstrapped DQN [17] and SUNRISE [13] have demonstrated the value of maintaining multiple policies for improved exploration and uncertainty estimation. These typically use random initialization or bootstrap resampling. In contrast, our ensemble agents are explicitly assigned different objectives via reward decomposition, allowing more interpretable modularity in behavior and enabling policy-level control through the ensemble aggregator. Our work utilizes this interpretable decomposition to produce an ensemble that supports behavioral constraints.

We extend A2C with multiple actor-critic learners, one per reward component, and develop an off-policy correction mechanism to enable training under a dynamic behavior policy, a scenario not directly handled by classic actor-critic methods [12]. Each learner operates independently on its assigned reward signal, enabling modular training and policy specialization.

### A. Off-Policy Actor-Critic Methods

Off-policy actor-critic methods such as ACER [28], DDPG [14], and IMPALA [5] tackle the instability of learning from experience not generated by the current policy. Our approach adds another layer of complexity: each agent is trained off-policy relative to an ensemble-driven behavior policy. We resolve this through a structured importance sampling ratio based on the policy aggregator. This correction makes up for the difference between individual learning policies and the ensemble action distribution, ensuring stable gradient updates.

Our approach connects to multiple areas in the reinforcement learning (RL) literature, including actor-critic optimization, reward decomposition, ensemble methods, and off-policy learning.

### B. Actor-Critic Algorithms

### C. Reward Decomposition

Reward decomposition has been explored in both theoretical and applied RL contexts. The Q-Decomposition framework [20] introduced additive reward modeling for factored state-action spaces. More recently, Hybrid Reward Architecture (HRA) [26] demonstrated that training separate value functions for each component of a reward can lead to improved learning efficiency. Unlike these value-based systems, we apply decomposition in a policy-gradient context, with separate actor-critic pairs each optimizing for a single dimension of the reward vector. Our ensemble construction allows us to align each learner's behavior with an interpretable component of the environment's landscape.

### D. Ensemble Reinforcement Learning

### E. Dynamic Behavior Policies

Dynamic or nonstationary behavior policies—common in distributed RL systems like IMPALA—complicate credit assignment due to shifting data distributions. V-trace [5] and similar techniques address this with truncation and correction. Our ensemble setup yields a behavior policy that changes with the ensemble composition and aggregator settings. We address this using importance sampling corrections across multiple agents and trajectories. Our contribution aims to extend these correction methods to a modular setting, allowing policy-based decomposition to stay stable under dynamic aggregation.

### F. DeepMind's Stratego Project

The DeepNash system [19] achieved incredible performance in Stratego by modeling imperfect information as a Bayesian game and solving it via regularized game-theoretic reinforcement learning. Our use of Stratego serves a different goal: instead of targeting equilibrium play, we seek controllability and modularity through reward decomposition. This allows strategic guardrails to be explicitly encoded in learning.

In summary, our work integrates insights from previous research to create an ensemble framework that extends A2C for policy-level reward decomposition. We introduce corrections for ensemble-driven behavior policies and demonstrate how strategic control can be achieved in complex environments through different dimensions of the reward.

## III. Background

In reinforcement learning, control problems are modeled as a Markov Decision Process (MDP) [3] which is defined by:

- A set of states $S$ that describe the current environmental conditions facing the agent;
- A set of actions $A(s)$ that an agent can take;
- Probabilities $p(s' \mid a, s)$ for transitioning from state $s$ to state $s'$ given action $a$;
- A function $r : S \times A \times S \to \mathbb{R}$ so that $r(s', a, s)$ supplies the immediate reward associated with going from $s$ to $s'$ after action $a$.

In environments where decision making takes place across a finite number of discrete time steps $T$, the sequence of periods the agent participates in is referred to as an episode. The goal of the agent is to learn a policy $\pi(s) \in \Delta A(s)$ where $\pi(a|s)$ describes the probability that a trained agent should take action $a$ in state $s$, to maximize the sequence of rewards across across an episode: $\sum_{t=0}^{T} \gamma^t r(s_{t+1}, a_t, s_t)$. Here $a_t$ and $s_t$ are the action and state at time $t$ and $\gamma \in [0, 1]$ is the discount factor on future rewards. Throughout we will slightly abuse notation to write $r(s_{t+1}, a_t, s_t)$ as $r_{t+1}$.

In policy-based reinforcement learning, the experiences of the agent are used to adjust the policy directly, using the policy gradient. To do this, we assume that the policy is a differentiable function of parameters $\mathbf{u}$ and write it as $\pi_{\mathbf{u}}(\cdot|s)$. In an Actor-Critic model long term values are also estimated and used within the policy gradient to reduce variance and expedite learning. The policy is referred to as the actor and the long-term value estimate is referred to as the critic.

For example, we may define the long-term state-action value function as the expected sum of discounted rewards:

$$Q_r^{(\pi)}(s, a) = \mathbb{E}\left( \sum_{j=0}^{T-t-1} \gamma^j r_{t+j+1} \middle| \begin{array}{c} s_t = s \\ a_t = a \\ \pi \end{array} \right). \quad (1)$$

This yields the following policy gradient estimate associated with a single episode's worth of data [24]:

$$\sum_{t=0}^{T-1} \nabla_{\mathbf{u}} \ln \pi_{\mathbf{u}}(a_t|s_t) Q_r^{(\pi)}(s_t, a_t).$$

This estimate of the policy gradient is then used to update the policy to put high probability on actions with large state-action values. When the state space $S$ is large the state-action value function is also modeled as a differentiable function of parameters. Both policy and long-term value functions can be learned using e.g. a vanilla Actor-Critic algorithm (see [23] or, for an A3C approach [16]).

### A. Behavioral Objectives

In general, there may be multiple factors that contribute to aggregate rewards, to the adjustments of the policy via the policy gradient theorem, and so to the decisions that agents make. When rewards are modeled as functions that yield scalars it obscures the contribution that each factor makes to agent policies and prevents users from adjusting those contributions to guide the agent to appropriate behavior.

To address this, we apply a behavioral version of reward decomposition [9]; specifically, we suppose the existence of a set of $K$ behaviors and associate with the $k$th behavior a function $b_k : S \times A \to \{0, 1\}$. This function takes in a state and an action and returns a value of 1 if $a$ constitutes behavior $k$ in state $s$, and a value of 0 if not. We require of these functions that:

1) For all $a \in A$ there exists $s, s' \in S$ such that $b_k(s, a) \neq b_k(s', a)$;
2) There exists $a, a' \in A$ and $s \in S$ such that $b_k(s, a) = b_k(s, a') = 1$.

These requirements imply that it is neither necessary (the first) nor sufficient (the second) to eliminate an action from $A$ to eliminate a behavior. In this sense the behaviors captured by $b_k(\cdot)$ are distinct from actions.

With these behaviors and associated functions in mind, we define a vector-valued function where each dimension represents a single factor:

$$\vec{r}(s', a, s) = \langle r(s', a, s), b_1(a, s), \ldots, b_K(s, a) \rangle.$$

It is natural to associate a long-term value estimate with each behavior dimension of the vectorized reward and to collect them in a vector, say $\vec{Q}^{(\pi)}(s, a)$. In analogue with equation 1 the dimension of $\vec{Q}^{(\pi)}(s, a)$ for behavior $k$ is:

$$Q_{b_k}^{(\pi)}(s, a) = E\left( \sum_{j=0}^{T-1} \gamma^j b_k(s_{t+j}, a_{t+j}) \middle| \begin{array}{c} s_t = s \\ a_t = a \\ \pi \end{array} \right).$$

### B. Policy Aggregation

In a value-based context we could use a linear scalarization to aggregate the rewards additively across dimensions by computing the dot product between $\vec{Q}^{(\pi)}(\cdot)$ and a vector of coefficients whose role is to select dimensions of the decomposed state-action value function for inclusion into the sum and weight them as appropriate:

$$\vec{m} = \langle m_r, m_{b_1}, \ldots, m_{b_K} \rangle.$$

We study the use of $\vec{m}$ to control value-based agents in [11].

In policy-based reinforcement learning long-term value estimates are only used to train policies and do not contribute to decision-making directly (e.g. using an $\varepsilon$-greedy policy). Thus, to use $\vec{m}$ to control agent behavior in a policy-based context we propose the use of an ensemble of agents, each trained on a single dimension of the reward decomposition, and an aggregative mechanism. Formally, consider weights:

$$\mathbf{u} = \{\mathbf{u}_r, \mathbf{u}_{b_1}, \ldots, \mathbf{u}_{b_K}\}$$

each of which fully parameterizes a policy and:

$$\mathbf{w} = \{\mathbf{w}_r, \mathbf{w}_{b_1}, \ldots, \mathbf{w}_{b_K}\}$$

each of which fully parameterizes a state-action value. This leads to a set of policies:

$$\pi_{\mathbf{u}}(\cdot|s) = \left\{ \pi_{\mathbf{u}_r}(\cdot|s), \pi_{\mathbf{u}_{b_1}}(\cdot|s), \ldots, \pi_{\mathbf{u}_{b_K}}(\cdot|s) \right\}$$

and state-action value functions:

$$\vec{Q}_{\mathbf{w}}(s,a) = \left\langle Q_{\mathbf{w}_r}(s,a), Q_{\mathbf{w}_{b_1}}(s,a), \ldots, Q_{\mathbf{w}_{b_K}}(s,a) \right\rangle.$$

Finally, with multiple policies involved we require some sort of aggregative process to take the outputs of individual policies and generate an action. To do this, we specify a policy aggregator function $C_{\mathbf{u}}^{(\vec{m})}(s) \in \Delta A(s)$.

The choice of aggregator is consequential, since it controls both the exploration/exploitation tradeoff and also is the main avenue to leverage the reward decomposition to adjust the behavior of the agents during and after training. In the experiments to follow we will use a linearly scalarized and softmaxed policy aggregator:

$$C_{\mathbf{u}}^{(\vec{m})}(a|s) = \frac{\exp\left\{m_r \pi_{\mathbf{u}_r}(a|s) + \sum_k m_{b_k} \pi_{\mathbf{u}_{b_k}}(a|s)\right\}}{\sum_{a' \in A} \exp\left\{m_r \pi_{\mathbf{u}_r}(a'|s) + \sum_k m_{b_k} \pi_{\mathbf{u}_{b_k}}(a'|s)\right\}}.$$

### C. Policy aggregation requires off-policy corrections

Aggregating multiple policies to determine the distribution that is actually used to generate actions that evolve the environment implies that the data-generating process is off-policy for any individual actor. We correct for this by using the importance sampling ratio:[1]

$$\rho_{b_k}(t,h) = \prod_{j=t}^{\min\{h,T-1\}} \frac{\pi_{\mathbf{u}_{b_k}}(a_j|s_j)}{C_{\mathbf{u}}^{(\vec{m})}(a_j|s_j)}.$$

[1]Note that this is only well-defined if $C_{\mathbf{u}}^{(\vec{m})}(\cdot)$ has "coverage" so that $\pi_{\mathbf{u}_{b_k}}(a|s) > 0$ implies that $C_{\mathbf{u}}^{(\vec{m})}(a|s) > 0$ for all $a$, $s$, and $k$. In general we meet this requirement by assuming that the aggregator is stochastic so that we have $C_{\mathbf{u}}^{(\vec{m})}(a|s) > 0$ for all $a$.

**Algorithm 1** Behavioral Actor-Critic with $n$-step estimates

**Input:** $\vec{m} \in \mathbb{R}^{K+1}$, $\gamma, \alpha_{\mathbf{w}}, \alpha_{\mathbf{u}} \in (0,1)$, and $n, N \in \mathbb{N}_+$

**Init:** $\mathbf{u}_r, \mathbf{u}_{b_1}, ..., \mathbf{u}_{b_K} \in \mathbb{R}^{d_{\mathbf{u}}}$ and $\mathbf{w}_r, \mathbf{w}_{b_1}, ..., \mathbf{w}_{b_K} \in \mathbb{R}^{d_{\mathbf{w}}}$

**for** Episode $1, 2, 3, \ldots, N$ **do**

  Sample an initial state $s_0 \in S$

  Set $I = 1$

  **for** $t = 1, 2, 3, \ldots$ **do**

    Sample action $a_t \sim C_{\mathbf{u}}^{(\vec{m})}(s)$

    Take action $a_t$, observe $s_{t+1}$ and $\vec{r}_{t+1}$

    **if** $t - n + 1 \geq 0$ **then**

      Set $\tau \leftarrow t - n + 1$

      **for** $r, b_1, \ldots, b_K$ **do**

        Set:

$$\rho \leftarrow \rho_{b_k}(\tau, \tau + n)$$
$$Q_\tau \leftarrow Q_{\mathbf{w}_{b_k}}(s_\tau, a_\tau)$$
$$Q_{\tau+n} \leftarrow Q_{\mathbf{w}_{b_k}}(s_{\tau+n}, a_{\tau+n})$$
$$G \leftarrow \sum_{j=1}^{\min\{n, T-\tau\}} \gamma^{j-1} b_k(s_{\tau+j}, a_{\tau+j}) + \gamma^n \times \begin{cases} Q_{\tau+n} & \text{if } \tau + n < T \\ 0 & \text{otherwise .} \end{cases}$$

        Update critic parameters w/ semi-gradients:

$$\mathbf{w}_{b_k} \leftarrow \mathbf{w}_{b_k} - \alpha_{\mathbf{w}} \rho [G - Q_\tau] \nabla Q_\tau$$

        Update actor parameters:

$$\mathbf{u}_{b_k} \leftarrow \mathbf{u}_{b_k} + \alpha_{\mathbf{u}} I \rho Q_\tau \nabla \ln \pi_{\mathbf{u}_{b_k}}(a_\tau|s_\tau)$$

      **end for**

      Set $I \leftarrow \gamma I$

    **end if**

  **end for**

**end for**

### D. Finding a Robust Scalarization

Incorporating a behavioral scalarization is an opportunity to adjust agent behavior after training by tuning the weights applied to the different reward and behavior dimensions used during inference.[2] Prior to actually observing a trained agent at work, a user may not have a good understanding of its learned-but-unmodified behaviors, may not know what adjustments will be needed during inference, and so may not know what scalarization to apply during training. To accommodate *a priori* uncertainty over post-training adjustments, we apply ideas from surrogate optimization to learn a training scalarization that maximizes post-training flexibility.

[2]Note that when the training mask differs from the mask used during inference, this approach is distinct from simply imposing a reward structure with fewer reward dimensions.

Consider the following problem:[3]

$$\max_{\vec{m}}\{h(\vec{m})\} \text{ subject to } \vec{m} \in M \tag{2}$$

where $h(\cdot)$ is a function quantifying agent performance and behavioral characteristics using data from inference episodes, and $M \subseteq \mathbb{R}^{K+1}$ is a set of possible linear scalarizations. The function $h(\cdot)$ may in general depend on other parameters and will be noisy since the inference episodes will change over time. The only information available to optimize $h(\cdot)$ comes from observing evaluations of it at specific input values. A sequence of $n$ such evaluations leads to a history:

$$D_n = \{(\vec{m}_1, h(\vec{m}_1|X_1)), \ldots, (\vec{m}_n, h(\vec{m}_n|X_n))\}$$

where $X_i$ is the batch of inference episodes generated by the agent trained using $\vec{m}_i$.

One approach to solve 2 is to use $D_n$ to build an inexpensive-to-evaluate model of the dependence of $h(\cdot)$ on $\vec{m}$ and then to leverage this model to make informed guesses as to which $\vec{m}$ to evaluate next. In surrogate optimization this model is referred to as the *surrogate* and the method of using it to make guesses about new parameters to try is called *acquisition*.

Finding a good surrogate is usually framed as an attempt to estimate $p_H(h|D_n, \vec{m})$ using the available data $D_n$. Common choices for the surrogate model are Gaussian Processes, Tree Parzen estimators, random forests, and Bayesian additive regression trees (NEED CITES). A common strategy for acquisition is to specify a utility function that measures the value associated with evaluating $h(\cdot)$ at a new point $\vec{m}$, say $u(\cdot)$. We can then choose $\vec{m}$ to maximize the expected utility by finding:

$$\vec{m}_{n+1} \in \underset{\vec{m}\in M}{\operatorname{argmax}} \left\{ \int_{h^*}^{\infty} u(z) p_H(z|D_n, \vec{m}) dz \right\}$$

where $h^*$ is the current largest known value for $h(\cdot)$. If $u(z) = z - h^*$ then this corresponds to choosing the next evaluation as the one that maximizes expected improvement, while if $u(z) = 1$ it is the one that maximizes probability of improvement.

[3]Surrogate optimization techniques are a family of methods aimed at optimizing so-called blackbox functions. SEE BLAR FOR FULL REVIEW OF SURROGATE OPTIMIZATION. The key characteristics of such functions are 1) that they are expensive to evaluate and 2) that they do not possess structure (e.g. concavity, differentiability) that can be leveraged to make optimization more efficient.

**Algorithm 2** (Surrogate Optimization to find the Scalarization)

**Input:** Initial evaluations $n_0$ and total iterations $N$

**Init:**
**for** $i = 1, \ldots, n_0$ **do**
  Sample $\vec{m}_i$ and train the agent using algorithm 1
  Generate a set $X_i$ of inference episodes sampling a new $\vec{m}$ from $M$ for each
  Evaluate $h(\vec{m}_i|X_i)$ using the data in $X_i$
**end for**
Set

$$D \leftarrow \{(\vec{m}_1, h(\vec{m}_1|X_1), \ldots, (\vec{m}_n, h(\vec{m}_n|X_n))\}$$
$$n \leftarrow n_0$$

**while** $n \leq N$ **do**
  Fit the surrogate model on $D$ to estimate $p(h|\vec{m})$
  Given $p(\cdot)$ choose a new $\vec{m}_{n+1}$
  Train the agent using $\vec{m}_{n+1}$ with algorithm 1
  Generate a set $X_{n+1}$ of inference episodes sampling a new $\vec{m}$ from $M$ for each
  Evaluate $h(\vec{m}_{n+1}|X_{n+1})$ using the data in $X_{n+1}$
  Update the history by setting:

$$D \leftarrow D \bigcup \{(\vec{m}_{n+1}, h(\vec{m}_{n+1}|X_{n+1}))\}$$

  Update $n \leftarrow n + 1$
**end while**

### *E. What should $h(\cdot)$ be?*

Our goal is to maximize the post-training flexibility of the user to adjust agent behavior without undermining agent performance. Given a batch of inference episodes generated by an agent trained under $\vec{m}$:

$$X = \{s_0^{(j)}, a_0^{(j)}, s_1^{(j)}, a_1^{(j)}, \ldots, s_{T-1}^{(j)}, a_{T-1}^{(j)}, s_T^{(j)}\}_{j=1,2,\ldots}$$

let:

$$W(x) = \begin{cases} 1 & \text{if the agent succeeds} \\ 0 & \text{otherwise} \end{cases}$$

indicate whether the agent succeeds against its primary task in episode $x \in X$. Then maximizing:

$$h(\vec{m}|X) = \sum_{x\in X} \frac{W(x)}{|X|} + \sum_{k=1}^{K} \text{Var}\left( \left\{ \sum_{t=0}^{T_j} \frac{b_k(s_t^{(j)}, a_t^{(j)})}{T_j} \right\}_{j=1}^{|X|} \right)$$

would involve finding a scalarization used during training that maximized the impact of changing scalarizations during inference on undesirable behavior but minimized impact on the performance of the agent against the primary task.

## IV. Asymptotic Behavior

N/A for now.

## V. Experiments

### A. The Stratego Environment

We conducted our experiments in a simplified Stratego environment with varying board sizes (3×3, 5×5, and 7×7) to examine the agent's performance under different levels of complexity. For the 3×3 board, each player begins with three pieces: a Flag (F) and ranks 1 and 2. For the 5×5 board, each player receives pieces `F, 1, 2, 2, 3`. For the 7×7 board, each player receives `F, 1, 1, 1, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3`. These pieces are randomly placed on each player's respective side of the board.

A legal move consists of selecting one of the current player's pieces and moving it one square up, down, left, or right (if not blocked), giving a maximum of $4B^2$ discrete actions on a $B \times B$ board; however, many actions are illegal in a given state due to piece positions. An episode terminates when one player captures the opponent's flag (win/loss) or when a maximum step count is reached (draw). For this work, the game is treated as fully observable, with both players' piece identities and positions made available to the agents, to focus the study on behavioral control rather than information asymmetry.

**Reward Decomposition:** The environment provides a vectorized reward signal with three components: a primary reward and two behavioral rewards. The primary reward $r^{(0)}$ is 1.0 for capturing the opponent's flag and 0 otherwise. The first behavioral reward $r^{(1)}$ is an indicator for capturing an enemy piece (excluding the flag). It yields 1 for any move that results in a successful capture and 0 otherwise. The second behavioral component $r^{(2)}$ is an efficiency penalty per time step, returning 1 for any non-terminal move and 0 for moves that end the episode. This encourages the agent to finish games in fewer moves.

At each time step, these components form a vector:

$$\mathbf{r} = (r^{(0)}, r^{(1)}, r^{(2)}),$$

which is fed to the learning algorithm. Reward decomposition allows explicit tracking of the agent's game-winning success, its tendency to capture enemy pieces, and the degree to which it wastes time with unnecessary moves.

**Fictitious Self-Play Opponents:** To evaluate and improve the agent in a competitive setting, we implement a fictitious self-play (FSP) scheme for opponent selection during training and inference. During training, the agent's opponent is drawn from a pool of past snapshots of itself, along with a random policy agent. After each Bayesian Optimization (BO) training chunk, we freeze the current agent's policy and add it to the opponent pool. For any new episode, there is a probability of sampling an opponent from this pool, weighted according to the current agent's win rate against each snapshot. This prevents overfitting to a single adversary and exposes the agent to a diverse set of tactics.

We tested two FSP update frequencies: (1) episodic FSP, switching opponents every 400 episodes, and (2) chunked FSP, using one fixed opponent for an entire training chunk before resampling. Chunked FSP produced stronger overall learning, likely because training against a single fixed opponent per chunk gives the agent enough data to adapt effectively. All experiments reported here use the chunked FSP approach, switching opponents every 2,000 episodes.

**Surrogate Optimization Pipeline:** A key novelty of our approach is the integration of a surrogate optimization loop that automatically adjusts the scalarization weights $\vec{m}$ over the course of training. Rather than fixing these weights, we treat them as tunable parameters.

Training is divided into "chunks" of episodes. At the start of each chunk, the agent is assigned a scalarization weight vector $\vec{m}$. It then trains against a fixed opponent for that chunk. After training, we evaluate the current policy across a range of scalarization values by playing evaluation episodes and measuring both win rates and behavioral statistics. These measurements feed into a heuristic objective function $h(\vec{m})$ that rewards (i) strong win rate and (ii) high behavioral variance, which indicates good downstream controllability.

A Bayesian Optimization routine (Algorithm 2) proposes the next scalarization vector for the following chunk. This train–evaluate–optimize loop continues for several iterations, producing a final policy trained under a weight vector estimated to optimize $h$. Thus, the environment is non-stationary across chunks: both the opponent and the scalarization weights evolve over training, guided by FSP and surrogate optimization, respectively.

### B. Training

Training follows the Behavioral Actor–Critic (BAC) algorithm (Algorithm 1). We maintain an ensemble of $K+1$ actor–critic learners, where $K = 2$ behavioral components plus one primary reward learner. Each learner $i$ receives its own reward component $r^{(i)}$ and trains its own policy $\pi_i$.

At each time step, the aggregated action distribution $C_{\mathbf{u}}^{(\vec{m})}$ is produced via a Boltzmann aggregation variant:

$$C_{\mathbf{u}}^{(\vec{m})}(a|s) = \frac{\exp\left\{m_r \pi_{\mathbf{u}_r}(a|s) + \sum_k m_{b_k} \pi_{\mathbf{u}_{b_k}}(a|s)\right\}}{\sum_{a' \in A} \exp\left\{m_r \pi_{\mathbf{u}_r}(a'|s) + \sum_k m_{b_k} \pi_{\mathbf{u}_{b_k}}(a'|s)\right\}}.$$

where the scalarization weights $\vec{m}$ do not have to fit in a probability simplex. The aggregation function drives the environment during data collection.

Because data is generated by $C$ rather than any individual $\pi_i$, each learner applies importance sampling with weight:

$$\rho_t^{(i)} = \frac{\pi_i(a_t \mid s_t)}{C(a_t \mid s_t)}.$$

We clip log-ratios to $[-10, 10]$ for stability.

Each actor–critic learner uses a two-layer MLP with 128 hidden units per layer and ReLU activations, producing action logits and a state-value estimate. We train using 3-step A2C with discount $\gamma = 0.99$. After every $n$ steps, we compute the $n$-step return $G^{(i)}$ and TD error:

$$\delta^{(i)} = G^{(i)} - V_i(s).$$

Critics minimize squared TD error, and actors ascend the gradient:

$$\rho_t^{(i)}\,\delta^{(i)}\nabla_{\mathbf{u}_i}\log\pi_i(a_t\mid s_t).$$

We optimize all networks with Adam at learning rate $3\times10^{-4}$. Each training chunk contains 2,000 episodes, and with 500 surrogate iterations, we obtain approximately 1 million total episodes. The opponent pool begins empty, defaulting to a random opponent when no historical snapshots exist. Opponents use the same architecture and frozen parameters. Scalarization weight search does not perform gradient updates: the agent trains within each chunk using a fixed $\vec{m}$.

### C. Results

We evaluate our trained agents on each board size (3×3, 5×5, 7×7) against a baseline random action-selection opponent. Each evaluation consists of 2,000 episodes.

**Random Policy Baseline (2,000 Episodes)**

| Board | Wins | Losses | Ties | Win Perc. |
|---|---|---|---|---|
| 3×3 | 657 | 660 | 683 | 32.85 |
| 5×5 | 846 | 850 | 304 | 42.30 |
| 7×7 | 917 | 953 | 130 | 45.85 |

**Trained Agent Results (2,000 Episodes)**

| Board | Wins | Losses | Ties | Win Perc. |
|---|---|---|---|---|
| 3×3 | 1339 | 372 | 289 | 66.95 |
| 5×5 | 1720 | 233 | 47 | 86.00 |
| 7×7 | 994 | 642 | 364 | 49.70 |

These results show substantial improvement over baseline performance. The 3×3 and 5×5 boards exhibit nearly double the win percentage compared to baseline play, while the 7×7 board—despite being significantly more complex—still exhibits a meaningful improvement. Overall, our trained agents achieve substantially higher win rates than a random policy, demonstrating robust learning across board sizes.

**Training Progress:** ***[INCLUDE INFERENCE WINRATE GRAPHS] [DISCUSS TRAINING RESULTS AND HOW WINRATE INCREASES AND STAYS STEADY AS WE REFINE OUR SCALARIZATIONS] [KDE GRAPHS TO SHOW SELECTED SCALARIZATION VALUES]***

**Behavioral Control Experiments:** A central claim of our work is that the learned agent's behavior can be adjusted by altering the scalarization weights while maintaining competitive performance. To test this, we performed behavior sweep experiments across all three board sizes, focusing on the piece-capturing reward component.

For each board size, we evaluated the final trained agent against two fixed opponents: (1) a random policy, and (2) a clone of the agent using its final scalarization vector.

We tested two scalarization configurations:

- **Zero Mask:** All scalarization components are set to zero except the targeted behavior (index 1), which is swept across a one-dimensional grid.
- **Current Mask:** All components except the targeted behavior are fixed at their final trained values; only the capture weight is swept.

For each configuration, board size, and opponent, we sweep the capture scalarization from $-3.0$ to $3.0$ in increments of $0.05$, yielding approximately 120 scalarization values. At each sweep point, we run 50 evaluation episodes and record: (1) average number of enemy pieces captured per episode, and (2) win rate.

This produces two families of curves per board size—zero mask vs. current mask—against both baseline opponents.

***[FICT BOLTZ BEHAVIOR GRID GRAPHS] [ANALYSIS AND RATIONALE THAT BEHAVIOR INCREASES AS SCALAR DOES, WINRATE DOESN'T DIP]***

## VI. Discussion and Future Work

### A. Discussion

**Strategic Diversity vs. Performance:** The experimental results confirm that our approach yields agents with interpretable and adjustable behaviors without severely hindering performance. In multi-objective settings, there is often an inherent tension between optimizing the main task and adhering to secondary objectives. Our work demonstrates a practical way to balance that tension by training an ensemble on decomposed rewards and using surrogate optimization for scalarization tuning. In effect, the agent learns a family of policies encoded within a single network. The ability to toggle behaviors on demand has potential real-world importance—for instance, a self-driving system might dynamically increase its weighting of defensive-driving behavior when a hazardous situation is detected, without requiring retraining. We showed that across a reasonable range of scalarization adjustments, the agent's win rate remains high as long as excessive weight is not assigned to behavioral components at the expense of the primary win objective. This indicates that training successfully identified a robust scalarization region: the agent's policy can move along behavioral dimensions while preserving strong baseline performance. However, extreme weight changes—such as setting $m_0 = 0$ and completely ignoring winning—will predictably degrade performance.

**Impact of Board Size and Environment Complexity:** We explicitly tested multiple board sizes to investigate how the benefits of behavior decomposition scale with environment complexity. On the small 3×3 board, changes in scalarization produced only subtle differences in behavior. The environment is so constrained that rushing the opponent's flag is almost always dominant; there is little room for evasion, positional play, or systematic piece capture. For example, an agent can often reach the enemy flag within two moves, and defending against a first-mover advantage is nearly impossible. Thus, regardless of the value of $m_1$, the agent frequently converges to flag-capturing behavior simply because alternatives are strategically inferior.

In contrast, the 7×7 board exhibited significantly richer behavioral diversity. With more space and more pieces, the agent faces meaningful decisions: whether to capture opponent pieces before advancing, whether to prioritize a direct flag approach, or whether to reposition defensively. Adjusting $m_1$ on

this board produced clearly distinguishable strategies—high-$m_1$ agents captured substantially more enemy pieces, while lower-$m_1$ agents tended toward aggressive flag-seeking behavior. The larger state space also demands stronger generalization. Our surrogate optimization procedure successfully identified weights that preserved robust performance across strategies, but baseline results show that environmental complexity naturally reduces the magnitude of the win-rate gains. Training variance was also noticeably higher on the $7\times7$ board, suggesting that managing multiple objectives becomes more difficult in larger, more complex environments. These findings collectively imply that reward decomposition is most impactful in richer domains where behavioral structure has room to manifest; simpler environments may not meaningfully reflect the advantages of such decomposition.

**Implications and Future Work:** The success of our experiments in this Stratego-like domain highlights the promise of Behavioral Actor–Critic (BAC) for tasks involving complex trade-offs. Many real-world settings—autonomous driving, resource allocation, multi-robot coordination, and others—feature multiple objectives beyond maximizing a single metric. In such domains, a system might need to dynamically adjust its behavioral priorities depending on context or user preference. Our framework makes this possible by enabling a single trained agent to behave differently under different scalarization vectors. Moreover, since the ensemble maintains separate policies for each reward dimension, the method provides a degree of interpretability: one can inspect the behavior of the "aggressive-only" or "efficiency-only" learner to understand how individual rewards influence decision-making.

***[FUTURE WORK INFORMATION BASED ON SOME OF THESE IMPLICATIONS/REAL EXAMPLES]***

There are, however, limitations to the current approach. The surrogate optimization loop, while effective, introduces substantial computational overhead. Each update effectively trains the agent under multiple scalarization vectors for relatively short durations to evaluate the objective $h(\vec{m})$. In more complex environments, a more sample-efficient search method—or a differentiable mechanism for updating $\vec{m}$—may be necessary. Additionally, our experiments remain confined to a relatively small-scale, fully observable game. Scaling to full Stratego or other partially observable, stochastic environments will introduce new challenges, including credit assignment across hidden information and a far larger action space. Longer training chunks or more surrogate iterations could also improve performance, though at the cost of increased computation.

In conclusion, our experiments provide strong evidence that the Behavioral Actor–Critic framework can train agents that maintain competitive performance while supporting controllable behavioral variation. By designing an environment with multiple reward channels, incorporating a self-play opponent pool, and integrating surrogate scalarization selection into training, we achieve a balance between adaptability and performance. Future work will extend this framework to larger domains and investigate whether the relationships we observed between board size, complexity, and behavioral controllability continue to hold in more challenging settings.

## VII. Conclusion

In this work, we introduced the Behavioral Actor–Critic (BAC) framework which is a policy-gradient method that combines reward decomposition, ensemble learning, and scalarization-based policy aggregation to enable post-training behavioral control. Rather than training a single policy on a monolithic scalar reward, BAC maintains an ensemble of actor–critic learners, each aligned with a distinct reward component, and aggregates their policies via a tunable scalarization vector. This structure allows users to adjust behavioral trade-offs after training by modifying scalarization vector values for specific components, while off-policy corrections based on the aggregator distribution maintain stable learning despite the mismatch between individual learners and the behavior policy.

We further integrated BAC with a surrogate optimization loop that searches over scalarization weights used during training. The surrogate objective is designed to favor both strong task performance and high behavioral variance under scalarization sweeps, yielding policies that are both effective and controllable. This loop, combined with a fictitious self-play opponent pool, creates a dynamic but structured training environment in which both the opponent distribution and the behavior aggregation weights evolve over time to challenge and refine the agent.

Our experiments in a simplified Stratego-like domain demonstrate that BAC can produce agents that achieve substantially higher win rates than a random baseline across multiple board sizes, while also supporting meaningful behavioral adjustment through scalarization. On larger boards, we observe clear and interpretable shifts in strategy, such as increased piece-capturing behavior under higher capture weights, without a loss in win rate or task performance, as long as the primary reward is not overly de-emphasized. These results support the central claim that reward-decomposed, ensemble-based policies can be shaped along specific behavioral dimensions without retraining, provided that training scalarizations are chosen to be robust and maximize post-training flexibility.

Beyond this environment, the BAC framework and surrogate-based scalarization search have broad implications for multi-objective control problems where safety, efficiency, fairness, or risk sensitivity must be traded off against primary performance metrics. Many such domains require both high performance and an ability to rapidly re-prioritize behaviors in response to context or user preference. Our approach offers a concrete mechanism for encoding these trade-offs into a single trained agent and then adjusting them at deployment time via scalarization. Future work will focus on scaling BAC and the surrogate optimization pipeline to larger, partially observable, and stochastic environments (including full Stratego and other complex games), exploring alternative aggregators and search strategies, and tightening theoretical connections between behavioral variability or robustness.

## Acknowledgment

The authors would like to thank the Milwaukee School of Engineering for supporting this research through computational resources and faculty guidance. This work was completed as part of the undergraduate research curriculum in the Department of Computer Science and Software Engineering.

## References

[1] N. Aissani, D. Trentesaux, and B. Beldjilali. "Multi-agent reinforcement learning for adaptive scheduling: application to multi-site company". In: *IFAC Proceedings Volumes* 42.4 (2009). 13th IFAC Symposium on Information Control Problems in Manufacturing, pp. 1102–1107. ISSN: 1474-6670. DOI: https://doi.org/10.3182/20090603-3-RU-2001.0280. URL: https://www.sciencedirect.com/science/article/pii/S1474667016339428.

[2] Adrià Puigdomènech Badia et al. "Never Give Up: Learning Directed Exploration Strategies". In: (2020). arXiv: 2002.06038. URL: http://arxiv.org/abs/2002.06038.

[3] Richard Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[4] Thomas Dietterich. "Hierarchical reinforcement learning with the maxq value function decomposition". In: *Journal of Artifcial Intelligence Research* 13 (2000).

[5] Lasse Espeholt et al. "IMPALA: Scalable distributed deep-RL with importance weighted actor-learner architectures". In: *International Conference on Machine Learning (ICML)*. 2018, pp. 1407–1416.

[6] Ying Guo, Astrid Zeman, and Rongxin Li. "A Reinforcement Learning Approach to Setting Multi-Objective Goals for Energy Demand Management". In: *International Journal of Agent Technologies and Systems (IJATS)* 1 (2 2009).

[7] Conor F. Hayes et al. "A practical guide to multi-objective reinforcement learning and planning". In: *Autonomous Agents and Multi-Agent Systems* 36 (26 2022).

[8] Rodrigo Toro Icarte et al. "Using Reward Machines for High-Level Task Specification and Decomposition in Reinforcement Learning". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, Oct. 2018, pp. 2107–2116. URL: https://proceedings.mlr.press/v80/icarte18a.html.

[9] Paulius Juozapaitis et al. "Explainable Reinforcement Learning via Reward Decomposition". In: *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*. 2019.

[10] Jonas Karlsson. "Task Decomposition in Reinforcement Learning". In: *1994 AAAI Spring Symposium*. 1994. URL: https://aaai.org/papers/0006-ss94-02-006-task-decomposition-in-reinforcement-learning/.

[11] Jonathan Keane, Sam Keyser, and Jeremy Kedziora. "Strategy Masking: A Method for Guardrails in Value-based Reinforcement Learning Agents". In: *Proceedings of the Milwaukee School of Engineering Undergraduate Research Symposium*. Available at https://arxiv.org/abs/2501.05501. 2025.

[12] Vijay R. Konda. "Actor-critic algorithms". PhD thesis. Massachusetts Institute of Technology, 2000.

[13] Kimin Lee, Seungyong Lee, and et al. "SUNRISE: A simple unified framework for ensemble learning in reinforcement learning". In: *International Conference on Machine Learning (ICML)*. 2021, pp. 6131–6141.

[14] Timothy Lillicrap et al. "Continuous control with deep reinforcement learning". In: *International Conference on Learning Representations (ICLR)*. 2016.

[15] Erlong Liu et al. *Pareto Set Learning for Multi-Objective Reinforcement Learning*. 2025. arXiv: 2501.06773 [cs.LG]. URL: https://arxiv.org/abs/2501.06773.

[16] Volodymyr Mnih et al. "Asynchronous methods for deep reinforcement learning". In: *International Conference on Machine Learning (ICML)*. 2016, pp. 1928–1937.

[17] Ian Osband et al. "Deep exploration via bootstrapped DQN". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2016, pp. 4026–4034.

[18] J. Perez et al. "Responsive elastic computing". In: *Proceedings of the 6th International Conference Industry Session on Grids Meets Autonomic*. 2009.

[19] Julien Pérolat et al. "Mastering the game of Stratego with model-free multiagent reinforcement learning". In: *Science* 378.6620 (2022), pp. 912–918.

[20] Stuart Russell and Andrew Zimdars. "Q-decomposition for reinforcement learning agents". In: *Proceedings of the 20th International Conference on Machine Learning (ICML)*. 2003.

[21] Stuart J. Russell and Andrew Zimdars. "Q-Decomposition for Reinforcement Learning Agents". In: *International Conference on Machine Learning*. 2003. URL: https://api.semanticscholar.org/CorpusID:5376984.

[22] Harm van Seijen et al. "Hybrid reward architecture for reinforcement learning". In: *CoRR* abs/1706.04208 (2017).

[23] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018. ISBN: 0262039249.

[24] Richard S. Sutton et al. "Policy gradient methods for reinforcement learning with function approximation". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2000.

[25] Chen K Tham and Richard W Prager. "A modular q-learning architecture for manipulator task decomposition". In: *In Machine Learning Proceedings*. 1994.

[26] Harm Van Seijen et al. "Hybrid reward architecture for reinforcement learning". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2017, pp. 5392–5402.

[27] Ziyi Wang et al. "Learning State-Specific Action Masks for Reinforcement Learning". In: *Algorithms* 17.2 (2024). ISSN: 1999-4893. DOI: 10.3390/a17020060. URL: https://www.mdpi.com/1999-4893/17/2/60.

[28] Ziyu Wang et al. "Sample efficient actor-critic with experience replay". In: *International Conference on Learning Representations (ICLR)*. 2017.

[29] R. J. Williams. "Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning". In: *Machine Learning* 8 (1992), pp. 229–256.

[30] R. J. Williams. *Toward a Theory of Reinforcement Learning Connectionist Systems*. Technical Report NU-CCS-88-3. Northeastern University, College of Computer Science, 1988.