Model Representation Considerations for Artificial Intelligence Opponent Development in Combat Games

Sarah Kitchen*, Chris McGroarty, and Timothy Aris****

*Michigan Tech Research Institute (MTRI), 3600 Green Ct., Ann Arbor, MI 48103,

**U.S. Army Combat Capabilities Development Command – Soldier Center (DEVCOM SC) Simulation and Training Technology Center (STTC), 12423 Research Parkway, Orlando, FL 32826,

*snkitche@mtu.edu, **christopher.mcgroarty.civ@army.mil, timothy.aris.civ@army.mil

Abstract

The performance and behavior of an Artificial Intelligence (AI) opponent in games requires coordination of multiple agents and complex tasks depends on many design choices made during implementation. Currently, gaming agents developed with Reinforcement Learning (RL) methods are constructed to play the game, leading to natural design choices for observations, actions, and rewards that are congruent with a human player's actions and objectives. However, in simulation and serious games, the objective of the implemented opponent should be developed in a way that supports the learning objectives for the user, such as by including additional ground truth environment data in the observation space or action structure. Therefore, the reward structure for the AI needs to incorporate more sophisticated considerations than just whether the game was won or lost by the AI. In this way the design space for opponent AI in these settings is considerably broader than what is traditionally used for RL gaming AI. This paper considers the implications of observation representation and reward design for the AI agent and associated actions in the context of 2-player battlefield-type games that are not strictly zero-sum. Semi-cooperative and fully competitive models are considered.

The environment in these games is a spatially extended battlefield in which agents must maneuver their forces to bring them into combat range of each other. The objective of the game is control over a pre-specified location in the game, and combat is executed via Lanchester attrition. We demonstrate the impact of aggregation on stochasticity of the model, where aggregation of the state model is controlled by various entropy-based metrics, as well as on the policy learned by an RL agent. Generalizations to alternative scenarios and objectives are discussed, as well as applications to the development of an AI combat opponent that can cohesively manage its forces over multiple scales.

Introduction

In games such as Chess and Go, each player has identical rules and objectives, and therefore an effective AI opponent for such games will behave nearly equivalently to a highly skilled human opponent (if not exceed human skill). Much of the application of Deep Reinforcement Learning (DRL) methods in recent years has focused on video games, including Real-Time Strategy (RTS) games such as StarCraft. The general approach is to leverage neural networks as policy and/or value function approximators, which are then trained in a reinforcement learning training loop that interfaces directly with the game. One of the significant features demonstrated by this approach is the ability of the AI to use images of the game as input observations, at the cost of complexity and computational time. Novel training strategies, such as league play for AlphaStar (Vinyals, et al. 2019), were developed to induce the DRL architectures to converge to policies that were broadly competitive across a wide variety of opponents. By competitively training agents, policy weaknesses were exploited at earlier stages of training.

The next stage of development for game AI-based in DRL is to expand to multi-agent settings. A recent survey (Wong, et al. 2022) identifies five state-of-the-art research areas for multi-agent RL, including opponent modeling and reward shaping, as well as the difficulty in overcoming complexity, high-dimensionality, and nonstationarity produced by agents changing their policies during training. For each agent training, the RL training loop for that agent treats the other agents as part of the environment. RL methods developed for single agents are based on Markov-Decision Processes, which assume stationarity of state transitions and rewards. Training methods, such as self-play, league play, and

Copyright © 2023, by the authors. All rights reserved.

curriculum learning, deliberately implement nonstationarity by changing opponent policies across training episodes to coax the policy of the agent to develop behaviors that are generally functional.

This paper considers the implications of state representation and reward design for the AI agent and associated actions in the context of 2-player battlefield-type games that are not strictly zero-sum. Specifically, while conflict exists between the objectives of the players, non-competitive objectives may also be included, such as metrics characterizing performance of each player beyond simply "win-loss" statistics. The environment in these games is a spatially extended battlefield in which agents must maneuver their forces to bring them into combat range. Combat of the units is automated according to Lanchester attrition, with attrition parameters that depend on the state and actions of the players. The objective of the game is control over a pre-specified location in the game, where each player controls its maneuver within the battlefield. We demonstrate the impact of aggregation of the battlefield representation on stochasticity of the model, where aggregation is controlled by various entropy-based metrics, as well as on the policy learned by an RL agent. Semi-cooperative and fully competitive models are considered. Generalizations to alternative scenarios and objectives are discussed, as well as applications to the development of an AI combat opponent that can cohesively manage its forces over multiple scales.

AI Players of Games vs. AI Opponents in Games

In the context of AI agents for serious games and training simulations, the function of the AI is not necessarily to "beat" the other player(s) (nominally, human users of the game), but to provide an environment to the human that meets specific training or learning objectives. For example, if the human player is expected to learn how to exploit preferences of a particular opponent (e.g. a more aggressive vs. more conservative AI), multiple AI opponents demonstrating behaviors consistent with "preference" characterizations must be included in the game. This presents novel design challenges to the general open problems above for Multi-Agent Reinforcement Learning (MARL). Three classes of such challenges include:

1. Design of rewards that potentially combine cooperative objectives (e.g. learning goals for the user) and competitive objectives (e.g. demonstrating the behavior of an opposing player).
2. Training of the agent, including modeling and/or implementing a stand-in for a human opponent.
3. Generalizability in that a single AI can support multiple uses (e.g. training objectives for the user) without retraining.

Item 3 could be as straightforward as enumerating training objectives for the user and training a separate AI for each. However, within a complex scenario, the human user may decompose their overarching goal/objective into intermediate tasks in ways that may not be anticipated during the design of the approach to Item 2. The co-design of the approach to address all three challenges is warranted.

In contrast to the development of an AI "player" of a game that may otherwise be black-boxed, the AI "opponent" described above is integrated directly into the game itself, and as such, can take advantage of more complete knowledge of the model underlying the environment of the game. AlphaStar, for example, converts game state features extracted by an API to StarCraft to build pixel-based frames of the state of the game. An AI with access to the navigation or terrain mesh used to build 3D environments in the game could instead map those features to an abstracted model that is designed to conform to that environment (Figure 1). The implications of this are expanded in the next section.

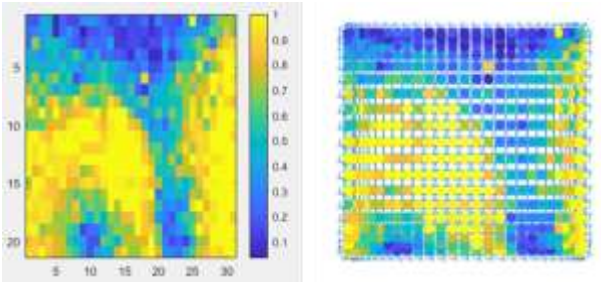

Figure 1: "Grid World" with slope preference overlay (left) and associated transition graph (right). Lower slope preference values correspond to steeper slopes in terrain.

Beyond the environment itself, an AI integrated into the game can operate on structurally different observations than players, since the AI knows internal game metrics, features, and "truth" data about the state of the game. Metrics collected about the player's behavior can be exposed to that AI, while remaining hidden to the player. This could be used, for example, to define an AI that adapts in difficulty to the player's performance in real time. Applications depend on the learning objectives for the user. In this way, the design space for opponent AI in the setting of serious games and training simulations is considerably broader than what is traditionally used for RL gaming AI.

State Representations and Abstractions

Reinforcement learning AI agents applied to games are frequently implicitly built on an underlying Markov Decision Process (MDP) that approximates the true environment. One of the design decisions required for implementation is that of the state representation used as input to the learner. In DRL, this is the format of the input to a deep neural network used to approximate the policy generating an action and/or additional networks used for feedback (e.g. state valuations

or state-action pair valuations). Effectively, the construction of a state representation input to the learner creates an analogue of a partially observable environment. However, without the additional implementation of a belief state function or approximator, the learner will approximate an optimal policy for an MDP that approximates the true environment, rather than solving the underlying Partially Observable Markov Decision Process (POMDP).

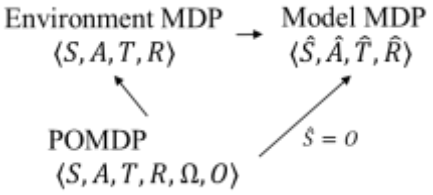

Figure 2: Model MDPs are implicitly used in reinforcement learning, as the true environment may be unknown or intractable. Even model-free reinforcement learning approaches usually use state representations $\hat{S}$ in a model MDP rather than a belief state as in a POMDP model.

Figure 2 illustrates the relationships between the MDP of the environment, in this case the game, a model MDP using a state representation $\hat{S}$ and a POMDP model with observations $O = \hat{S}$. Arrows in the figure indicate that information about the model is "forgotten." Specifically, the POMDP model includes all of the components of an MDP, as well as an observation set O and observation distribution Ω. A state representation $\hat{S}$ is defined as a set with a surjective map $\phi: S \to \hat{S}$. If this map is not injective, then the representation $\hat{S}$ also aggregates states in S. An example aggregation is shown in Figure 3, and the procedure used for this example is described below.

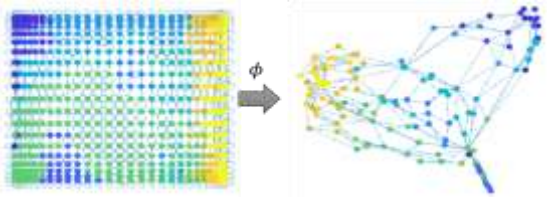

Figure 3: Example state aggregation map for Grid World example of Figure 1. The number of nodes is reduced by >4x. Colors indicate preimage nodes (left) of the node of the same color in the aggregated state space (right).

It may also be convenient to aggregate actions A of the original MDP to define a reduced action set $\hat{A}$ in the model MDP, and we will abuse notation below by letting ϕ denote either the map on states or the associated map $\phi: S \times A \to \hat{S} \times \hat{A}$. Options and meta-action frameworks follow from this approach as well, are frequently identified with hierarchical reinforcement learning methods, and are expected to be included in future work of the authors. This approach allows the AI to learn solutions to subproblems in the environment and combine them to achieve an optimal policy.

In the case that ϕ is an aggregation of the true state space, the transition and reward functions $\hat{T}$ and $\hat{R}$ are induced by ϕ by appropriately averaging the values of the preimage of any point. This process is called taking the direct image along ϕ. Namely, we define the direct image of the transition distribution to be:

$$\hat{T}(s'|s'_0, a'_0) = \sum_{\phi(s_0,a_0)=(s'_0,a'_0)} \sum_{\phi(s)=s'} \frac{T(s|s_0, a_0)}{|\phi^{-1}(s'_0, a'_0)|}$$

And the direct image of the reward function to be:

$$\hat{R}(s'_0, a'_0) = \sum_{\phi(s_0,a_0)=(s'_0,a'_0)} \frac{R(s_0, a_0)}{|\phi^{-1}(s'_0, a'_0)|}$$

A generalization of this definition allowing for a non-uniform prior over $S \times A$ can be found in (Jiang et al. 2015). MDP homomorphisms (Ravindran & Barto, 2004) are a special case of the above definition, but are defined so that the aggregated model is equivalent to the original MDP in the sense that an optimal policy π' for the aggregated model can be lifted directly to an optimal policy through composition $\pi = \pi' \circ \phi$ with the homomorphism ϕ, which is traditionally assumed to be bijective on actions. Homomorphisms are known to be strong forms of model abstractions. Model abstractions that do not require complete knowledge of the environment (i.e. a model for the "true" MDP) are themselves amenable to model-free approaches, and these are necessary to tackle computationally intractable problems. There is a broad literature on automatically learning abstractions that approximate MDP homomorphisms under varying desiderata including action equivariance (Van der Pol, et al. 2020) and effect equivalence (Mavor-Parker et al. 2022). A simulation environment that enables us to collect appropriate data for automatic model aggregation benefits the application to a gaming AI opponent.

One of the challenges that emerges in DRL is the associated design problem of reward specification that elicits desired behaviors. Sparse rewards, such as achieving win conditions, allow the AI to explore portions of policy space that may not be explored under the influence of designed incremental rewards. In optimal control, incremental rewards (rather, costs) often naturally arise, as the problems are often framed in terms of time, fuel, or other resource minimization. In reinforcement learning, intrinsic rewards schemes such as curiosity (Pathak et al. 2017) have been introduced to encourage exploration of policy space during training. Other reward-shaping parameters include progress estimator functions, subgoals/tasks, potential functions, and reward and intrinsic option discovery. Options are critical for Hierarchical Reinforcement Learning (HRL), as they temporally abstract the RL model by combining sequences of

actions into behaviors, often defined to achieve some subtask or goal. Recent work in this field (Zhang et al. 2021) is moving towards HRL methods that are capable of learning intrinsic options simultaneously with an option-conditional policy that applies them to the task required in the environment, without separately addressing the problem of subgoal discovery or task decomposition.

Aggregation Method

By defining the model MDP of interest as the direct image of an aggregation on state-action space, we trade the optimality of a policy trained on the aggregate model for greater flexibility in the reduction of the size of the state-action space, and consequently have the opportunity to greatly reduce the number of actions needed before achieving a sparse reward regime.

In game settings where scores are tabulated only at the end of the game, non-terminating state-action pairs have zero instantaneous reward. In complex scenarios such as wargaming, even instantaneous metrics of progress can reach thresholds that do not guarantee successful outcomes overall. The direct image approach allows us to, in effect, diffuse a sparse reward over states that are contracted in the state aggregation. What is desired from the state representation is:

1. The complexity of the model is reduced in terms of the number of states and actions.
2. The complexity of the model is reduced in terms of unidirectionality of the reward gradient.
3. The optimal state-action value in the abstracted model is *consistent* with the optimal state-action value in the true model in the following sense: If $\phi(s, a_i) = (s', a_i')$, for $i = 1,2$, and if $Q'(s', a_1') > Q'(s', a_2')$, then $Q(s, a_1) > Q(s, a_2)$, where Q and Q' are optimal Q-functions.

Item 1 is straightforward by design. Item 3 is expected to be necessary for the true optimal policy to be a refinement of the coarse optimal policy. Item 2 is directed at the observation that while the use of sparse reward leads to novel behaviors, training deep reinforcement learners on sparse rewards alone has poor sample complexity. Our approach to defining such an abstraction is to define a weighting function $w(s, a, s_0)$ on $S \times A \times S$ together with a threshold τ, removing triples from $S \times A \times S$ with weighting below the threshold, then aggregating along a partition defined by the strongly connected components (SCC) of the remaining graph. For demonstration purposes, heuristic weighting functions were used. In Figure 3, $w(s, a, s_0)$ was defined to be the difference in slopes between s and s_0, so edges corresponding to increasing slope were removed to define the aggregation via SCC on the remaining graph, where states are nodes and actions define edges whenever the transition probability $T(s|a, s_0) > 0$.

For the gaming application of interest, a selected weighting function was defined by diffusing sparse terminal rewards across $S \times A \times S$ using a given (either deterministic or known stochastic model) transition distribution, i.e. by applying Bellman's recursion k times for some k to R_{term}, a model for terminal rewards, using a model transition distribution T. The power k should be chosen just large enough for most triples to have nonzero weight attached to them. This is essentially executing value iteration on just the terminal rewards. This approach takes into account the location of the sparse rewards in state space as well as the topology of the environment. However, the dependence on the model to execute this calculation, as well as the subsequent constraints on model size, make this choice for a heuristic weighting infeasible in general. In the intended application, characters move in a fixed navigation mesh that can be preprocessed manually or with weightings such as the slope gradient prior to the introduction of reward diffusion. Future work will identify conditions on w that give rise to Lax bisimulation metrics (Taylor et al. 2008), as well as recent work on model learning (Grimm et al. 2021). Alternative heuristics that would make sense for battle games could leverage distance metrics in the environment, together with learning objectives, or tasks associated to gameplay.

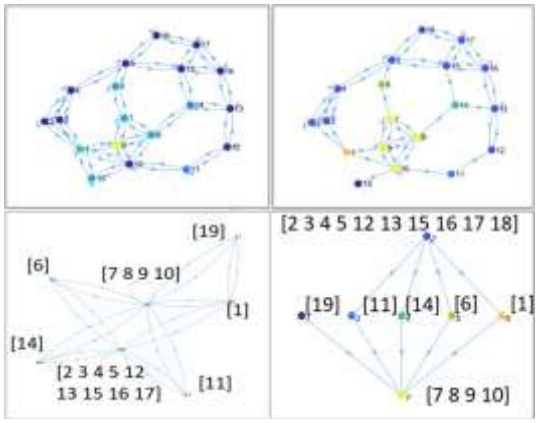

Figure 4: Demonstration of the weighted SCC method in 19-node battlefield. (Top Left) Battlefield nodes colored by mean diffused terminal reward; (Top Right) Edges removed by thresholding, with nodes colored by component membership; (Bottom Left) Contracted graph with node partition labels; (Bottom Right) Contracted thresholded graph with partition labels.

Strongly connected components were chosen as the partition method to contract the graph, since the condensation is directed acyclic. When we use diffused terminal rewards as the weighting function, the interpretation of the resulting condensation (Figure 4, Bottom Right) is that the directed paths down the graph to node 7 provide options for coarse policies, and additionally in this example identify structurally critical nodes in the environment (nodes 1, 6, 11, 14, 19, which appear as singletons in the SCC partition).

Game Environment and Experiments

Consider a two-player game in which each player has N units in a battlefield that is represented as a graph, as in (Figure 4, Top Left). Each player can move all units from one node to an adjacent node (for simplicity in the present paper, disaggregation of units to distribute across multiple nodes is not allowed). When the units of both players are located at the same node v, they lose forces according to the discrete update rule:

$$x[t+1] = x[t] - \alpha_y(v)y[t]dt,$$

where $\alpha_y(v)$ depends on the node in the environment, and x denotes the remaining number of units for one player, and y the remaining units for the other player. This model serves as an abstraction of strategy wargames, which is the motivating category of games for this effort.

The states of this game consist of the configuration in the graph of the two players' units, which can be presented to the learner as a pair of nodes. Actions are given by each player selecting an adjacent node to their current position or selecting to remain at the given node. Transitions are deterministic. The competitive objective for both players is to capture a designated goal node v_g in the environment. Player x ('blue') is the defensive player, initializing in a neighborhood of v_g with a competitive advantage at a subset of 'fortified' nodes in that neighborhood, with $\alpha_x(v) = \alpha_f > \alpha$ for v 'fortified', where α is the default value for all other nodes and for both players. Player y ('red') is the offensive player, initially at a disadvantage, but can change their force multiplier by approaching the goal node through 'surprise' nodes v_s, which the AI will have to identify through training.

Two reward structures are considered: A semi-cooperative structure, in which Player x ceding the goal node v_g to Player y after losing units in combat receives a nonzero (joint) terminal reward, and a non-cooperative structure, in which that terminal condition receives no such reward. No terminal reward was received if both units were completely attritted. Additional incremental and terminal rewards were included to bias a reinforcement learning agent towards desired behaviors. Since a semi-cooperative structure was considered with a global joint reward, a centralized policy was developed for both players.

The game graph environment (e.g. Figure 4, Top Left) is the simulation environment in which the game unfolds. We can study the impact of state representations via the SCC approach by defining a contracted graph model (e.g. Figure 4, Bottom Left), and mapping the true state of the game environment (i.e. the location of units for both players) to the associated aggregated node in the contracted model. In this way, the AI for the contracted model is only aware that Player x or y is present in a group of nodes. For example, if Player x initializes at node 9 in the game environment, the AI will receive only the information that Player x is at one of the nodes in the group [7, 8, 9, 10]. The nodes that are singletons in the aggregation map (nodes 1, 6, 11, 14, 19) take on an additional importance, then, as they are not only critical nodes defining distinct options for approach to v_g, they also are faithfully represented in the aggregate model. When an action is selected in the contracted state model, a random action in the preimage of that action is selected. For each player, an action is defined by source-target node pairs, therefore, candidate actions in the preimage are given by source-target pairs that map to the defined edge in the contracted graph. A lifted action is selected from the preimage uniformly randomly, and if that action is not feasible from that player's current location, no movement is taken. These two contributing factors of the definition are the origin of increased stochasticity of the model in the contracted graph. A consequence of stochasticity is demonstrated in Figure 5.

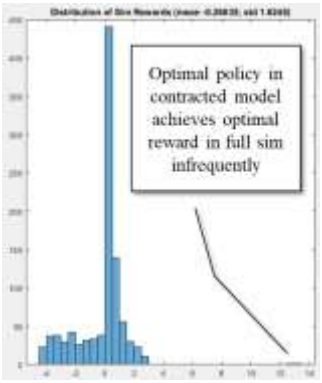

Figure 5: Reward distribution over 1000 games for AI trained in SCC aggregated model.

An AI agent was trained using Q-learning for the full graph model, a contraction using the SCC approach, and for an alternative hand-generated contraction of the same size, illustrated in Figure 6. To compare the models and policies, we use information entropy applied to distributions and normalized arrays of interest. Specifically, if $\vec{u}$ is a d-dimensional vector, then $\vec{p} = \vec{u}/\left|\left|\vec{u}\right|\right|_1$ is a finite distribution. The entropy of $\vec{p}$ is:

$$H(\vec{p}) = -\sum_{i=1..d} p_i \log(p_i)$$

To compare entropies of different dimensions, we normalize $H(\vec{p})$ by $H_1 = H\left(\vec{1}/d\right) = \log(d)/d$, to define

$$H_n(\vec{p}) = H(\vec{p})/H_1.$$

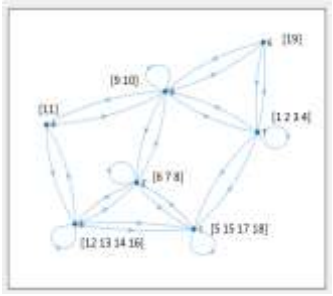

Figure 6: Manually generated aggregation.

For the transition distribution, we compute H_n for the conditional state distribution for each state-action pair. For the weighting function used to define the SCC, we compute the mean weighting for each state-action pair and normalize the resulting vector over actions for each state. Similarly, we normalize the Q-value over actions for each state. In both cases, we apply H_n to the action arrays for each state. In the "Results" section below, the mean normalized entropy is reported for the full model, the SCC aggregation, and the manually-defined aggregation. The Q-value table entropies are reported for the semi-cooperative and adversarial games.

Results

In Table 1 below, we see that apart from the Rewards, the full model has lower entropy than the aggregate models. Notable in particular is the transition distribution, which is nearly deterministic in the full model. The nonzero value appearing in the table is attributable to the fact that Player y is able to change their attrition multiplier $\alpha_y(v_g)$ by transiting through special nodes v_s. Also notable is the increase in entropy in all models from the semi-cooperative game to the adversarial game. Note that the high mean values for Q-tables reflects a strongly bimodal distribution of H_n, with one peak near 0 (deterministic policy) and one peak near 1 (uniform random policy). The mean value in the table reflects the proportion of states from which all actions are valued approximately equally. The higher values for the aggregated models indicate the valuation is less able to discriminate between actions for more states. The SCC aggregation improves the Semi-Cooperative Q and Transition distribution mean entropies, relative to the manually-generated policy.

Table 1: Mean Normalized Entropies

	Full	***SCC***	***Man.***
Semi-Coop. Q	0.78	0.87	0.88
Adversarial Q	0.83	0.94	0.94
Transition	0.09	0.23	0.33
Weighting	0.97	0.91	0.89

Qualitatively different policies were learned for the Semi-cooperative and Adversarial models in both the full (Figure 7) and aggregated (not shown) state representations.

Conclusions

State representation is a critical component of AI gaming agent and opponent design. By differentiating between AI game-playing agents and integrated AI opponents, we open the design space for representations used by AI opponents that will not only have access to internal game state and structural data not shown to a player, but development of the AI can include model development and training not directly

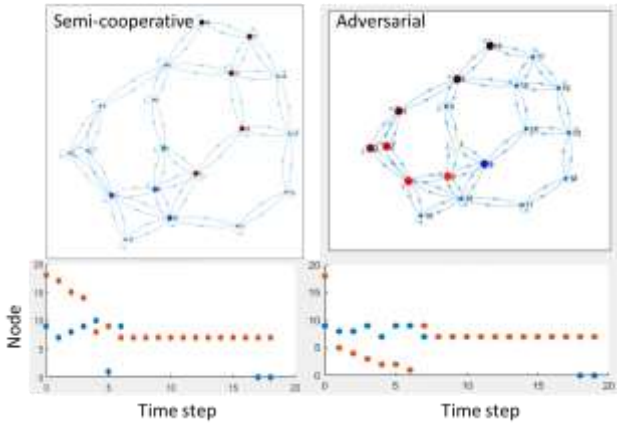

Figure 7: Policy demonstration for agent trained in full model for semi-cooperative and adversarial reward structures. The semi-cooperative model discovered a policy that routes the offensive player through a strategic node that increases their force multiplier $\alpha_y(v_g)$.

needed for game-playing. One way this manifests in strategy wargames is that the environment can be decomposed and represented for the AI as a graph, allowing the AI or player to reason over the impacts of various courses of action of aggregate forces. However, this process generically increases the stochasticity of the underlying MDP. The SCC approach can be used to select an aggregate model that reduces the impact of aggregation relative to alternative aggregation schemes. Further work needs to be done to characterize weighting schemes that give rise to metrics used in related state aggregation work.

Acknowledgements

This material is based upon work supported by the US Army Combat Capabilities Development Command Soldier Center (DEVCOM SC) Simulation and Training Technology Center (STTC) under contract No. W912CG-21-C-0016. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the DEVCOM SC STTC.

References

Grimm, C., Barreto, A., Farquhar, G., Silver, D. and Singh, S., 2021. Proper value equivalence. Advances in Neural Information Processing Systems, 34, pp.7773-7786.

Jiang, N., Kulesza, A. and Singh, S., 2015, June. Abstraction selection in model-based reinforcement learning. In International Conference on Machine Learning (pp. 179-188). PMLR.

Mavor-Parker, A.N., Banino, A., Griffin, L.D. and Barry, C., 2022. A simple approach for state-action abstraction using a learned mdp homomorphism. arXiv preprint arXiv:2209.06356.

Pathak, D., Agrawal, P., Efros, A.A. and Darrell, T., 2017, July. Curiosity-driven exploration by self-supervised prediction. In International conference on machine learning (pp. 2778-2787). PMLR.

Ravindran, B. and Barto, A.G., 2004. Approximate homomorphisms: A framework for non-exact minimization in Markov decision processes.

Taylor, J., Precup, D. and Panagaden, P., 2008. Bounding performance loss in approximate MDP homomorphisms. Advances in Neural Information Processing Systems, 21.

Van der Pol, E., Kipf, T., Oliehoek, F.A. and Welling, M., 2020. Plannable approximations to mdp homomorphisms: Equivariance under actions. arXiv preprint arXiv:2002.11963.

Vinyals, O., Babuschkin, I., Czarnecki, W.M., Mathieu, M., Dudzik, A., Chung, J., Choi, D.H., Powell, R., Ewalds, T., Georgiev, P. and Oh, J., 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. Nature, 575(7782), pp.350-354.

Wong, A., Bäck, T., Kononova, A.V. and Plaat, A., 2022. Deep multiagent reinforcement learning: Challenges and directions. Artificial Intelligence Review, pp.1-34.

Zhang, J., Yu, H. and Xu, W., 2021. Hierarchical reinforcement learning by discovering intrinsic options. arXiv preprint arXiv:2101.06521.