Aggregation of Hierarchically-Organized Agents in a Multi-Agent System

Sarah Kitchen* and Keith Brawner**

*Michigan Tech Research Institute (MTRI), 3600 Green Ct., Ann Arbor, MI 48103,
**U.S. Army Combat Capabilities Development Command – Soldier Center (CCDC SC), SFC Paul Ray Smith Simulation & Training Technology Center (STTC), 12423 Research Parkway, Orlando, FL 32826,
*snkitche@mtu.edu, **keith.w.brawner.civ@army.mil

Abstract

Artificially intelligent agents are enjoying increased adoption in both the video game and simulation industry - being used for both training and education as well as entertainment purposes with largescale real-time strategy games. Both type of systems have a need for the simulation of large numbers of units. While traditional scalability solutions (e.g. segmenting the terrain and using separate hardware to process various sections) can be used, this is wasteful of processing power which could have been better used to approximate the results of far-flung conflict while disaggregating close-in conflict for the benefit of the user/trainee. Approaches to do so haven't been investigated due to the smaller scales of simulations until recent developments. This paper provides an introduction to a modeling approach that can be used to bridge optimal control techniques with wargaming AI agents in order to provide cohesive aggregated and disaggregated AI forces within simulations.

Introduction: Use of Large Scale Training Simulations in the Military

The United State Army has invested in 6 primary Modernization efforts, with "Lethality" being among them. The Lethality Cross-Functional Team (CFT) has among it the majority of the technologies of interest to this conference audience – virtual reality, augmented reality, rapid AI processing of pictures and images, and of specific relevance to this paper - a Synthetic Training Environment (STE). The purpose of the STE CFT, as the name suggests, is to develop and acquire a synthetic training environment in which to acquire the skills to perform militarily-relevant tasks. Simulation offers the ability to do many of the tasks that one might expect to do in Soldiering, while the augmented reality headset allows for the simultaneous physical conditioning required for such tasks.

The use of simulation enables the training of large-scale maneuvers not previously possible at the echelons of relevance to modern combat. The modern US Army is segmented into 5 corps with 20,000 to 45,000 Soldiers, in turn divided into 2 or more divisions (with 10-15,000 Soldiers each), each divided into 3 brigades (with 2-5,000 Soldiers), each divided, divided again, and divided until you obtain the smallest unit (fire team – 2 Soldiers) and eventually the individual Soldier. Simulation of individual Soldiers and their actions is the subject of a variety of research projects – but the simulation of the group-level behaviors is, and must be, different. Two people are simply not the sum of the parts of one person – not in decision-making, ability, speed, maneuverability, or anything else. A battalion of 100-1,000 Soldiers does not respond as though it is a simple mass of manpower.

That said, an aggregated unit does not (should not) behave distinctly differently from the strategy and tactics which it is ordered. A video game unit order to attack should attack – an Army unit ordered to attack should do so. Famously, Eisenhower summarized the status of the end of D-Day as "The mission of this Allied Force was fulfilled at 0241, local time, May 7th, 1945." The order of the mission had somewhat more detail at each lower echelon of combat organization.

Training the individual units is a separate problem not discussed within this work, but once trained, these simulated units can be modified and dropped into a training scenario as the appropriate echelon of units. The aggregated units can both aggregate and disaggregate in simulation – being represented as large block approximations when needed by the underlying simulation engine and as small block individual units when being observed by a player for close-up combat.

Due to recent advances in neural networks, reinforcement learning (RL) has provided a powerful suite of tools for developing AI agents that are able to win complex games such as StarCraft and Go against expert human players, but the existing approaches do not yet meet the needs described

above, which in a standard implementation would provide scalability and training sample complexity challenges intrinsic to many-agent problems. In order to expand the capability of RL agents, our work leverages the inherent structure provided by the military hierarchy, and the corresponding multi-timescale nature of decision-making, through the use of aggregation/disaggregation across varying degrees of model abstraction in graph-based models. Primary challenges in designing such a hierarchy of abstraction and training a suitable AI across it are those intrinsic to multi-scale modeling (Qu, Garfinkel, Weiss, & Nival, 2011). Our approach is analogous to a type of coarse-graining in which the aggregated models effectively carry the averaged structure and behavior of the finer-scale models. Additionally, we leverage models in deterministic systems to guide design of the RL models.

The remainder of this paper is the first of a series in establishing the technologies that will be used to facilitate aggregate and disaggregate unit planning, the methods used ensure reasonable approximation levels, a discussion of cohesion and extensibility, and the initial evaluations of the early-stage work accomplished thus far.

Underlying Research Areas

Graph Prerequisites

Though the majority of this section is standard, it is included to establish notation and terminology used later. A directed graph $G = (V, E, t, h)$ consists of the information of a vertex (aka node) set V, an set of directed edges (aka arrows) E, and maps $t, h: E \to V$, sending an edge e configured as in Figure 1 to its source vertex $v = t(e)$ and target vertex $w = h(e)$. The notation t, h indicates the "tail" and "head" of the arrow. Graphs considered will allow multiple edges, but will disallow self-loops (i.e. $t(e) = h(e)$).

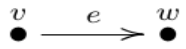


Figure 1: A directed edge with its source and target vertices

A neighborhood N_v of a vertex v is the set of all of the vertices in V connected to v directly by an edge that is not a self-loop. We will use the notation $\bar{N}_v$ for $N_v \cup \{v\}$; i.e., to indicate inclusion of v in the neighborhood. Another useful construction of N_v is the following: Let E_v^+ denote the edges in E for which $h(e) = v$, and let E_v^- denote the edges in E for which $t(e) = v$. Let N_v^+ denote the nodes $t(e)$ such that $e \in E_v^+$ (i.e. sources of edges pointing in to v), and let N_v^- denote the nodes $h(e)$ such that $e \in E_v^-$ (i.e. targets of edges pointing out of v).

For aggregation/disaggregation mechanics, we will require maps between graphs and associated systems. A map of graphs $\phi: G \to G'$ consists of the data of set maps $\phi_V: V \to V'$ and $\phi_E: E^{\phi,c} \to E'$, where E^{ϕ} is the set of edges such that $\phi_V(t(e)) = \phi_V(h(e))$ and $E^{\phi,c} = E \backslash E^{\phi}$, satisfying the compatibility conditions $\phi_V(t(e)) = t(\phi_E(e))$ and $\phi_V(h(e)) = h(\phi_E(e))$. Alternatively, we may define ϕ to take $e \in E^{\phi}$ to $v = \phi_V(t(e)) = \phi_V(h(e))$. Note that either definition deviates somewhat from the usual definition of a graph morphism, which requires a set map $E \to E'$.

We need the following definitions for properties of graph maps:

- A graph map $\phi: G \to G'$ is *injective on nodes* if $\phi_V(v) = \phi_V(v')$ implies that $v = v'$.
- A graph map $\phi: G \to G'$ is *injective on edges* if $\phi_E(e) = \phi_E(e')$ implies that $e = e'$.
- We will say $\phi: G \to G'$ is *injective* if it is injective on both nodes and edges.
- A graph map $\phi: G \to G'$ is *surjective on nodes* if for all $v' \in V'$, there is a $v \in V$ such that $\phi_V(v) = v'$.
- A graph map $\phi: G \to G'$ is *surjective on edges* if for all $e' \in E'$, there is an $e \in E$ such that $\phi_E(e) = e'$.
- A graph map $\phi: G \to G'$ is *full* if whenever $e \in E'$ is an edge such that $t(e), h(e) \in \phi_V(V)$, then $e \in \phi_E(E)$.

We define the following classes of maps that will be of interest:

1. *Embeddings:* A graph map $\phi: G \to G'$ that is injective and full.
2. *Projections on Nodes:* A graph map $\phi: G \to G'$ that is surjective on nodes, and injective on edges in $E^{\phi,c}$. This is equivalent to the operation of edge contraction on E^{ϕ}.
3. *Projections on Edges:* A graph map $\phi: G \to G'$ that is surjective on edges, and injective on nodes. This is a quotient operation on equivalence classes of edges defined by ϕ.
4. *Contractions:* A graph map $\phi: G \to G'$ that is a composition $\phi = \phi_2 \circ \phi_1$ of a projection on nodes $\phi_1: G \to G''$ followed by a projection on edges $\phi_2: G'' \to G'$. This is equivalent to the construction of a quotient graph.
5. *Fibrations:* A graph map $\phi: G \to G'$ with the property that for any edge e' in G' and node v in G such that $h(e') = \phi_V(v)$, there is a unique e in G with $h(e) = v$ and $\phi_E(e) = e'$.

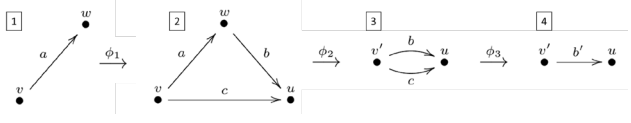


Figure 2: An example of an embedding ϕ_1, followed by a projection on nodes ϕ_2, then a projection on edges ϕ_3. The composition $\phi_3 \circ \phi_2$ is an example of a contraction.

Note that by definition, only graphs with multi-edges between node pairs allow nontrivial projections on edges. A fundamental example to keep in mind is presented in Figure 2. Graph embeddings are both graph morphisms and fibrations.

Graph Dynamical Systems

A *state* on G is an assignment of a dimension n_v and a vector $x_v \in \boldsymbol{R}^{n_v}$ to each vertex v of G. For a fixed enumeration on V, the state x may also be formatted as a vector in $\boldsymbol{R}^{n_1+\cdots n_{|V|}}$ by vertically stacking the vectors x_i for v_i in order:

$$x = \left(x_1^\top, x_2^\top, \dots, x_{|V|}^\top\right)^\top$$

An *interaction* on G is a matrix-valued function $a: V \to \mathrm{Mat}(n_v)$; i.e., an assignment of an $n_v \times n_v$ matrix a_v to each vertex v of G, and a *form* on G is a matrix-valued function $\omega: E \to \mathrm{Mat}(n_{h(e)}, n_{t(e)})$; i.e., an assignment of an $n_{h(e)} \times n_{t(e)}$ matrix ω_e to each edge e of G. The data of the pair (a, ω) consisting of an interaction and a form will define a first order dynamical system on a graph. Such a system on a graph is a linear system over the combined vector spaces over all nodes of G, where for each vertex v, we have the system

$$\dot{x}_v = a_v x_v - \sum_{e \in E_v^-} \omega_e x_v + \sum_{e \in E_v^+} \omega_e x_{t(e)} \quad (1)$$

A matrix form of the system $\dot{x} = Lx$ can be formed blockwise, with the blocks of the form $a_v - \sum_{e \in E_v^-} \omega_e$ on the diagonal, and ω_e in blocks corresponding to adjacencies $t(e) \to h(e)$ in the directed graph. Note that in our preliminaries, we indicated that we are disallowing self-loops in the graph model. Here, the presence of a non-trivial interaction a on the graph does in fact correspond to allowing self-loops in our graph. However, this is materially of importance in the relationships between the dynamical systems, but not in the definitions of the maps on the underlying graphs.

A time-discretization of the above formulation can also be defined, where the finite difference form of equation (1) defines a vertex update function at v mapping the values of $x(t)$ on $\overline{N}_v$ to a new state value $x_v(t+1)$ at time $t+1$. The discrete graph dynamical system updating scheme in this case is simultaneous across all v, thereby defining an update rule $x(t+1) = (I + L)x(t) = Fx(t)$. Graph cellular automata can be defined in this setting, with $x(t)$ in a finite state set, often binary.

New Applications in AI

Control Model on a GDS

The role of these models in our approach to AI development are that they provide a deterministic approach to understanding the extremal policies in associated MDP models (defined later in this section) via optimal control problems. Specifically, we consider bilinear control problems on a graph G, in which the state vector x_v consists of an abstracted potential vector for our simulation model, and the control variables are forms ω on directed graphs. Let $\dot{x} = L(\omega)x$ denote the associated GDS, with terminal loss function $\Psi(x(T), \omega(T))$, then the optimal control problem is to find a control trajectory $\omega^*(t)$, given an initial state $x(0) = x_0$ such that the corresponding state trajectory $x^*(t)$ has terminal state $x^*(T)$ such that $\Psi(x^*(T), \omega^*(T))$ is minimized. In this setting, the optimal control is extremal in the sense that ω^* assigns only values of 0 and 1 to the edges of the graph ("bang-bang") control. More generally, and in the hierarchical reinforcement learning model of the next section, we will replace the terminal loss with an incremental and terminal reward function. Correspondingly, we will exchange minimization of the loss with maximization of the reward. We will also introduce stochasticity into the model.

The pair $(x, L(\omega)x)$ is a vector field on the graph, as defined in (DeVille & Lerman, 2015). In the models of interest, the control variables ω are matrices indexed by edges. The main result of loc. cit. is that the pull-back of a vector field along a graph fibration is a vector field. In the case of an embedding $\phi: G' \to G$, this pullback corresponds to a projection of dynamical systems onto the subspace determined by the vertices of the full subgraph. Flows for the projection $L^\phi(\omega) = \phi^* L(\omega) \phi^{*\dagger}$ of a fixed control matrix can be extended by zero outside of the closed subsystem defined by the full subgraph supporting the embedding. Since the control variables ω are indexed by edges, this corresponds to studying the control problem on edges in G'.

Upcoming work will extend the results of loc. cit. to node projections. In this way, we can decompose a system $\dot{x} = L(\omega)x$ across a subgraph and the complementary node projection. Since edge projections, and therefore general contractions (aka graph quotients), are not fibrations, further work will need to be done to complete the extension of the formal theory to the primary case of interest. The approach will be to generalize the formalism of vector fields on graphs to the formalism of sheaves on graphs (Hansen & Ghrist, 2019), which both provides a more general toolbox, and which directly includes data associated to edges in their construction, unlike the networks on graphs defined in (DeVille & Lerman, 2015), though at the expense of restricting our attention from general manifolds considered in networks to the special case of vector spaces.

We consider next a stochastic version of the deterministic time-discretized model, where we only have control over part of the state update matrix $F(\omega)$. That is, suppose the state vector decomposes $x = (y^\top, z^\top)^\top$ such that $x(t+1) = F(\omega)x(t)$ can be written as:

$$y(t+1) = B_y(\omega)y(t) + A_y z(t) \quad (2)$$

$$z(t+1) = A_z y(t) + B_z z(t) \quad (3)$$

where A_y, A_z are constant and B_z is assumed to be a random matrix. Note we use subscripts x_t in place of function notation $x(t)$ below to save space.

In the case of interest, the vectors y and z are competitive populations, where we control only the behavior of population y through the matrix $B_y(\omega)$, though we may assume a comparable control model exists for z. We may assume $B_y(\omega)$ is locally constant and B_z is drawn from a matrix distribution that is constrained by the underlying graph structure of the problem.

This stochastic difference equation on a graph is the basis for the Markov Decision Process (MDP) model of interest. MDPs consist of the data $M = \langle S, A, T, r, \gamma \rangle$, where the set S is the set of states for the environment, the set A consists of actions the agent can take, the transition function $T: S \times A \to \Delta(S)$ defines for each state-action pair (s, a) a conditional distribution $T(\cdot\,|s, a)$ on S for the possible next state in the evolution of the system, the function $r: S \times A \to [0, r_{max}]$ defining an instantaneous reward for a state-action pair, and the discount factor $\gamma \in [0,1)$ to be used in discounting the cumulative reward.

The objective of such a model is to design a policy $\pi: S \to \Delta(A)$, where $\Delta(A)$ denotes distributions over A, which maximizes the expected cumulative discounted reward $R_H = \sum_{t \le H} \gamma^t r(s_t, a_t)$ to a selected horizon H (which may be infinite), where the sequence $(s_0, a_0, s_1, \dots, a_{t-1}, s_t, a_t)$ to time t evolves according to T and π. That is, we have

$$a_t \sim \pi(\cdot\,|s_t), \text{ and}$$

$$s_{t+1} \sim T(\cdot\,|s_t, a_t)$$

An MDP with states given by $x = (y^\top, z^\top)^\top$, actions given by matrices $a = B_y(\omega)$, the constraint equations (2)-(3) govern the transition distribution

$$T(x_{t+1}|x_t, a_t) = T(z_{t+1}|z_t, y_t)\delta(y_{t+1}, a_t y_t + A_y z_t)$$

where $T(z_{t+1}|z_t, y_t)$ is governed by the distribution from which B_z is drawn. Assumed nonstationarity of B_z induces us to explore model-free approaches to policy optimization, such as reinforcement learning.

Hierarchal Reinforcement Learning

Reinforcement Learning AI agents have demonstrated impressive capabilities in competitive environments in recent years (Sutton & Barto, 2018). All reinforcement learning is based on an underlying MDP model $M = \langle S, A, T, r, \gamma \rangle$ and aims to maximize the value function, defined to be the desired expected value as a function on state space for a given policy π:

$$V^\pi(s) = E[R_H|\pi, s_0 = s].$$

In model-free reinforcement learning methods, the strategy for approximating an optimal policy is to approximate the value function for the unknown optimal policy $V^*(s)$ or the function $Q^*(s, a) = r(s, a) + \gamma \sum_{s' \in S} T(s'|s, a)V^*(s')$, and leverages these approximations to define the optimal policy $\pi^*(a|s)$ as the arg max over, or more generally a distribution over, actions according to the value estimates $Q^*(s, a)$.

One of the significant benefits afforded by approximate policy optimization through function approximation methods for $Q^*(s, a)$, $V^*(s)$, and/or $\pi^*(a|s)$, is the trade-off in time and memory complexity for sample complexity. The size of the state-action space $|S \times A| = |S||A|$ fundamentally affects the complexity bounds for algorithms that exactly find or approximately estimate the optimal policy.

A hierarchical model may be understood in one of two ways: (1) We may define a homomorphism of models $\Phi: M \to M'$ such that an optimal policy of M' can be lifted to M, with ideally a significant reduction in the complexity of M', (2) A model M' is a "model of models", in which the action set A' of M' is a set of optimal policies from a set of models $\{M_1, \dots, M_n\}$. The first type of hierarchy is based on model abstraction (Ravindran & Barto, 2004), and the second on the options approach to temporal abstraction (Vezhnevets, Wu, Leblond, & Leibo, 2019). This paper will focus on formalism for (1), though our long-term objective is to present a unified approach to both in the context of the graph-based models presented below.

In the case of interacting multi-agent systems, the basic state space of interest is a configuration space for the set of agents I_S in a finite set S_0, and a state may be defined to be a map $s: I_S \to S_0$. The actions of interest may be formulated similarly, as maps $a: I_A \to A_0$, where I_A indexes a set of choices that must be made to define an action, and A_0 defines options for each choice, which we will call an action value set. In such a state and action space, if all maps are allowed without constraints, we have

$$|S||A| = |S_0|^{|I_S|}|A_0|^{|I_A|}. \quad (2)$$

We define a graph environment multi-agent MDP model to consist of a graph $G = (V, E, t, h)$, an agent set $I = I_S$, an action value set A_0, and an MDP $\langle S, A, T, r, \gamma \rangle$, with S consisting of a subset of maps $s: I_S \to V$, and A consisting of a subset of maps $a: E \to A_0$. Write $\langle G, S, A, T, r, \gamma \rangle$ for such a model. We may also write $S(I_S, V)$ and $A(E, A_0)$ when we wish to emphasize the dependence on I_S and A_0 in the model definition.

There are effectively four independent ways that complexity in (2) can be reduced by changing the model:

- Contract $S_0 = V$ through projections on nodes,
- Contract $I_A = E$ through contractions on edges,
- Aggregate agents I_S (defined below), or
- Reduce the number of values in A_0.

Contraction of I_A is enabled by contraction of S_0 in our graph structure, since only multi-edges can be contracted through

projections on edges. We will see that projections on nodes enable agent aggregation and contractions on edges. In this paper, we will not focus on the reduction of A_0, so it may be assumed for simplicity that A_0 is constant across all models.

Let $M = \langle G, S, A, T, r, \gamma \rangle$ and $M' = \langle G', S', A', T', r', \gamma' \rangle$ are be two models with a graph contraction $\phi: G \to G'$ and a surjective map $\phi_I: I_S \to I_{S'}$ we will call an agent aggregation. For $s: I_S \to V$ a state in M, and $s': I_{S'} \to V'$, write $s \in s'$ if $s'\big(\phi_I(x_i)\big) = \phi_V(s(x_i))$ for all x_i in I. Let $S_\phi \subset S$ be the subset of states $s: I_S \to V$ for which a state $s': I_{S'} \to V'$ exists such that $s \in s'$.

As an example, let G be graph 2 of Figure 2, and let G' be graph 4, with $\phi = \phi_3 \circ \phi_2$ the illustrated contraction. Suppose I_S consists of two agents: x_1, x_2, and $I_{S'}$ consists of a single agent y, and define the aggregation map

$$\phi_I(x_1) = \phi_I(x_2) = y.$$

There are two states in M': $s'_1(y) = u$ and $s'_2(y) = v'$. For $s \in s'_1$, we must have

$$\phi_V\big(s(x_i)\big) = s'_1(y) = u,$$

and therefore $s(x_i) = u$ for each i. For $s \in s'_2$, we must have

$$\phi_V\big(s(x_i)\big) = s'_2(y) = v',$$

and therefore $s(x_i)$ must be either v or w for each i. In summary, a state s in S_ϕ must have $s(x_1) = s(x_2) = u$ or $s(x_1)$ and $s(x_2)$ are both in the embedded graph 1 of Figure 2 because we have

$$s'(y) = \phi_V(s(x_i)).$$

Note in particular that $S_\phi \neq S$, so the complexity of the model M itself can be reduced by restricting the states to those which are compatible with the pair of maps $\Phi = (\phi, \phi_I)$.

Similar to states, for a an action in A and a' in A', we write $a \in a'$ if $a'(e') = \max a(\phi_E^{-1}(e'))$ for all e' in E'. Write $a' = \phi(a)$ for such a and a'. In the future work, we expect to replace the max in this definition with a more general function on the inverse image sets. We define A_ϕ to be the actions in A such that $a \in a'$ for some a' in A', and have a well-defined map $\phi: A_\phi \to A$. Note that by construction, if two actions a_1, a_2 in A_ϕ agree on $E \backslash E^\phi$, then $\phi(a_1) = \phi(a_2)$.

We will say Φ defines a model hierarchy if there is a distribution μ on $S_\phi \times A_\phi$ such that

$$T'(s'_1 | s'_0, a') = \sum_{s_0 \in s'_0, a \in a', s_1 \in s'_1} T(s_1 | s_0, a) \frac{\mu(s_0, a)}{\mu'(s'_0, a')}$$

where

$$\mu'(s', a') \coloneqq \textstyle\sum_{s \in s', a \in a'} \mu(s, a).$$

In (Jiang, Kulesz, & Singh, 2015) the authors define an approximate homomorphism between models to satisfy both a condition of the above type on the transition distributions and the corresponding equality for the reward function:

$$r'(s', a') = \textstyle\sum_{s \in s', a \in a'} r(s, a) \frac{\mu(s,a)}{\mu'(s', a')}.$$

The results of that paper and the references provided therein provide tools for selecting node projections in the present setting, under the assumption that we have access to a dataset of sequences $(s_0, a, r(s_0, a), s_1)$ from the model M. Some additional problem constraints must be satisfied to apply the results directly.

The pull-back of a policy $\pi': S' \to \Delta(A')$ along Φ is the map defined by

$$\phi^* \pi'(a|s) = \pi'\big(\phi(a) \big| \phi(s)\big) \frac{\mu(a)}{\mu'\big(\phi(a)\big)}$$

for all $(s, a) \in S_\phi \times A_\phi$. Observe that this policy is constant along the sets $\phi(a)$ and $\phi(s)$, and the optimal policy on M' may not pull back to an optimal policy on M. In order to construct more general policies on M, we define the notion of a lift of a policy $\pi': S' \to \Delta(A')$ along Φ to be a policy $\pi: S_\phi \to \Delta(A_\phi)$ such that

$$\pi'(a'|s') = \textstyle\sum_{s \in s', a \in a'} \pi(a|s) \frac{\mu(s)}{\mu'(s')}.$$

For a fixed model hierarchy defined by Φ and μ (the latter of which we suppress from our notation), let the pair (M_Φ, M') where

$$M_\Phi = \langle G, S_\phi, A_\phi, T, r, \gamma \rangle$$

and

$$M' = \langle G', S', A', T', r', \gamma' \rangle$$

denote the corresponding compatible pair of models. We have shown that the aggregation of agents in M' strongly constrains the lifts of policies π' to M_Φ. An optimal policy for the pair (M_Φ, M') is an optimal policy π' for M' together with an optimal policy for M_Φ that is a lift of π'.

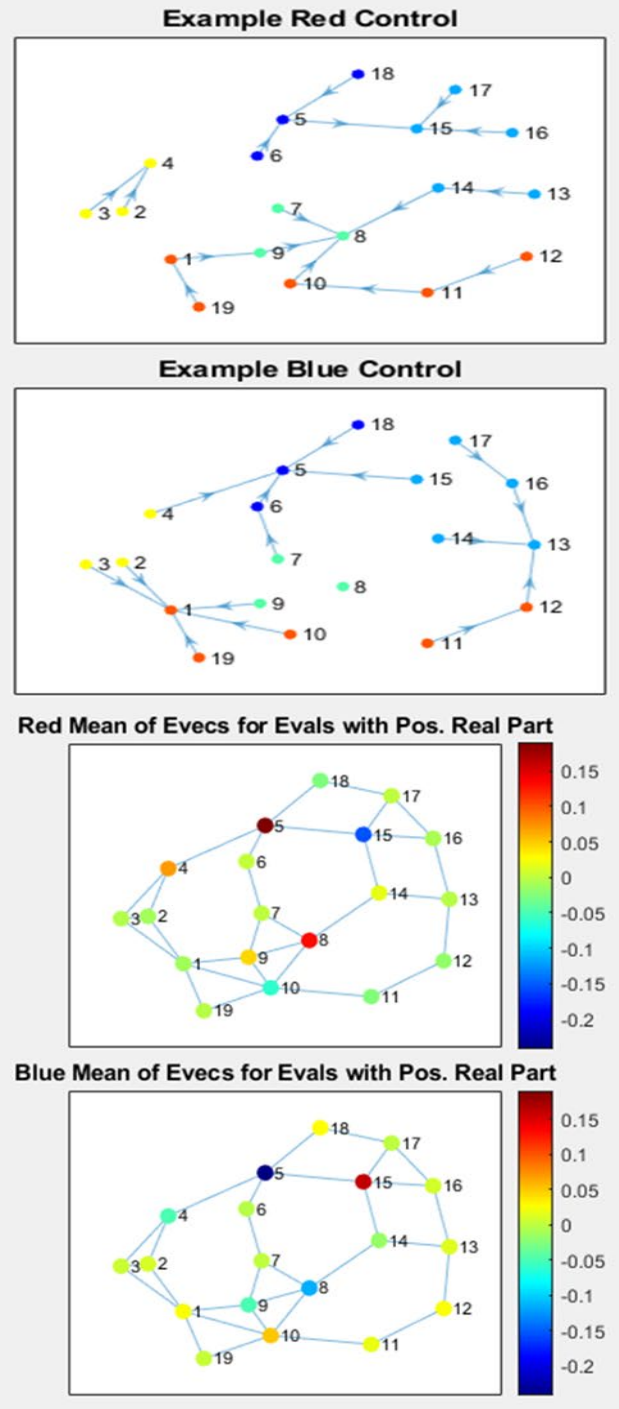
Dynamic Model Aggregation/Disaggregation

In the above RL models, we began with a fixed model homomorphism or hierarchy. However, in dynamic engagements across the graph models implemented in simulation, we naturally will need to simulate both coarse and fine models, depending on the complexity of the engagement. Our hypothesis is that subgraphs of the fully disaggregated graph model with nontrivial solutions to the pullback of a locally constant optimal control define the appropriate the appropriate time-local aggregation model, through node projection of these models. Control change points in the disaggregated GDS provide change conditions for the hierarchical RL model. This can be extended across multiple scales.

To begin to demonstrate the feasibility of the approach, we begin with a spectral analysis on an example GDS $\dot{x} = L(\omega)x$, where $x_v = (r_v, b_v)$ at node v consists of a red and blue population at that node, and the interactions are governed by Lanchester equations for attrition. Assume the control ω imposes a pair of directed acyclic graph (DAG) structures on the model, one for population r (red), and the other

for population b (blue). The eigenvalues of $L(\omega)$ with positive real parts, and their corresponding eigenvectors, govern the asymptotic behavior of the system. Figure 3 demonstrates that the environment is expected to have a non-uniform distribution of heavily contested areas for a uniform initial force distribution and independently drawn random control for the red and blue population. Contested means one of the two sides will obtain control at the end of the engagement, as well as areas where one side or the other is not expected to dominate. Though these nodes are sinks for the control DAGS, not all sinks demonstrate this asymmetric outcome. A nominal node partition defining a graph contraction is also given in the control plots.

Figure 3: The most contested nodes (5,15,8) are associated with sinks for red or blue. The coloring of the control plots show a nominal partition of nodes for contraction based on asymptotic behavior of dynamical system.



Discussion

This paper presents the formal language required to model aggregation and disaggregation both in deterministic optimal control models, through the formalism of vector fields on graph models, and in hierarchical MDP models in which the state and action spaces are also based on aggregation/disaggregation of forces in GDS. The purpose of the deterministic model is primarily to abstract the units to continuous "force potentials", with which we investigate optimal control solutions to the models cohesively across aggregation/disaggregation of models based on graph fibrations. Generalizing past graph fibrations will be performed in future work. In the analogous MDP models for RL, model homomorphisms via graph contractions are naturally defined, as well as the associated notion of policy lifting. Cohesively defining aggregation/disaggregation across both the GDS and RL models will allow us to use the GDS control solutions to design dynamic aggregation/disaggregation in the RL model.

References

DeVille, L., & Lerman, E. (2015). Modular Dynamical Systems on Networks. *Journal of the European Mathematical Society*, 2977-3013.

Hansen, J., & Ghrist, R. (2019). Toward a spectral theory of cellular sheaves. *Journal of Applied and Computational Topology 3, no. 4*, 315-358.

Jiang, N., Kulesz, A., & Singh, S. (2015). Abstraction selection in model-based reinforcement learning. *International Conference on Machine Learning* (pp. 179-188). PMLR.

Qu, Z., Garfinkel, A., Weiss, J., & Nival, M. (2011). Multi-scale modeling in biology: how to bridge the gap between scales?". *Progress in biophysics and molecular biology 107 no. 1*, 21-31.

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

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction.* Cambridge: MIT Press.

Vezhnevets, A. S., Wu, Y., Leblond, R., & Leibo, J. Z. (2019). *Options as responses: Grounding behavioural hierarchies in multi-agent RL.* Retrieved from arXiv preprint: arXiv:1906.01470