LfD Training of Heterogeneous Formation Behaviors

William Squires, Sean Luke
Department of Computer Science
George Mason University
4400 University Dr
Fairfax, Virginia 22030

Abstract

Problem domains such as disaster relief, search and rescue, and games can benefit from having a human quickly train coordinated behaviors for a diverse set of agents. Hierarchical Training of Agent Behaviors (HiTAB) is a Learning from Demonstration (LfD) approach that addresses some inherent complexities in multiagent learning, making it possible to train complex heterogeneous behaviors from a small set of training samples. In this paper, we successfully demonstrate LfD training of formation behaviors using a small set of agents that, without retraining, continue to operate correctly when additional agents are available. We selected training of formations for the experiments because formations: require a great deal of coordination between agents, are heterogenous due to the differing roles of participating agents, and can scale as the number of agents grows. We also introduce some extensions to HiTAB that facilitate this type of training.

Introduction

Multiagent Learning from Demonstration (LfD) promises to allow a human to quickly train coordinated behaviors for a diverse set of agents in an online manner with the goal of producing combined behaviors that are more beneficial than agents acting concurrently but without coordination. To do this, Multiagent LfD typically draws on knowledge of each agent's sensors, behaviors, and of the problem domain. This research applies to problem domains in which agents or robots must be rapidly put to use, such as disaster relief, search and rescue, and games where players control a large and diverse set of agents. However, multiagent LfD is very sparsely researched, in large part due to the inherent complexities of multiagent learning due to the Curse of Dimensionality and what we refer to as the *Multiagent Inverse Problem*.

The Curse of Dimensionality states that the number of training samples required for effective machine learning increases exponentially with the dimensionality of the feature vector, which is likely exacerbated by heterogeneity due to increased variety in sensors. The Multiagent Inverse Problem is encountered when trying to learn the appropriate combination of individual agent behaviors to achieve the desired macro-level behavior: while we might have a function available that maps the individual behaviors to the macro-level behavior (namely, a simulator), we do not have the needed function that maps in the other direction. Such inverse problems are normally overcome using offline optimization methods, such as multiagent reinforcement learning or stochastic optimization, but the high cost of generating training samples by a human trainer makes these challenges much more daunting problem for LfD.

The Hierarchical Training of Agent Behaviors (HiTAB) LfD approach (Luke and Ziparo 2010) has addresses these learning challenges. The Curse of Dimensionality is dealt with through iterative behavior decomposition and manual feature selection. The trainer first decomposes the problem into a hierarchy of subproblems, such as breaking "play kiddie soccer" into "play offense" and "play defense", with (for example) "play offense" further broken down to "acquire ball", "manipulate ball", and "kick to goal", and so on. Training is done on the lowest-level behaviors, then the next level, and so on. This allows HiTAB to project the full joint space of features (sensor information) and actions of the top-level behavior into many smaller behaviors, each with its own much smaller subset of sensor features and actions, and consequently fewer training samples.

HiTAB addresses the inverse problem similarly with the introduction of a virtual controller agent hierarchy that allows the trainer to manually decompose the coordination of behaviors among subordinate agent groups. That is, we manually break the swarm into a hierarchy sub-swarms and sub-sub-swarms etc., each headed by a virtual controller agent (a boss). Then we can train small groups to do simple collective behaviors, then assign each a virtual controller (a boss), then train small groups of bosses to do collective behaviors involving directing their subordinates, and so on. Because we are only training small groups of agents at a time, the gulf between individual micro-level behaviors and the desired emergent macro-level phenomenon is mitigated.

Our research goal is to extend HiTAB to train swarm-like heterogeneous behaviors where the resulting behavior can scale to very large numbers of agents. Further, the training should be accomplished without knowing the precise number of agents available for each heterogenous agent types in operation. Training complexity is reduced by including a minimal number of agents to train the coordinated behaviors, with the end result being a minimal controller hierarchy. However, in

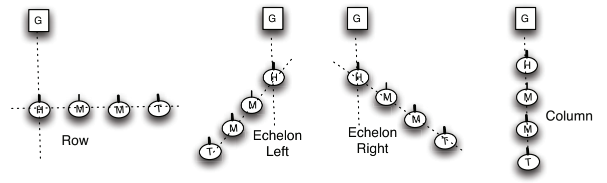

Figure 1: Line Formations

operation the controller hierarchy may need to be grown to effectively utilize large numbers of agents. For example, a grid formation might have a grid controller with some number of subordinate line controllers that are determined by number of available agents.

This paper presents work in progress toward this research goal by training line formations of agents. Line formations are heterogeneous due to differing roles of agents and require considerable coordination to achieve the desired behavior. Formations are also a well studied problem domain in multiagent learning where the effectiveness of the learned behaviors can be visually confirmed. The formations we will learn are shown in Figure 1, each one has a Head agent facing the goal with all other agents forming the line of at some angle to the goal.

Related Work

Heterogeneous Swarms and Hierarchies

Swarm research primarily focuses on the creation of behaviors in which the individual agents only interact with neighboring agents or the environment. Because the interaction is limited in this way, adding agents to the swarm scales with regard to communication. However, this scaling comes at the expense of global information which limits the level of cooperation that can be achieved. Introducing heterogeneity to swarms complicates the problem of having agents perform cooperative behaviors using only local interaction and often introduces a need for structured team organization not present in homogeneous swarms.

In (Elston and Frew 2008) and (Pinciroli et al. 2010), heterogeneous swarms use a hierarchical structure to create coordinated behaviors between an aerial agent and homogeneous sub-swarms. The hierarchy extends the capability of the sub-swarm by leveraging global information relayed to the sub-swarm by the aerial controller. In this case, the sub-swarms still scale because the only additional communication is between the sub-swarm agents and the controller. In (Soule and Heckendorn 2010), an evolutionary approach was presented to learn a controller hierarchy that scales to the available swarm agents with the introduction of force functions that balance agents in the hierarchy or create a new sub-swarms when needed.

Learning from Demonstration

Learning from Demonstration (LfD) is a supervised learning method where training samples are generated through human demonstration (Atkeson and Schaal 1997). LfD literature may be broken into two the categories. In the first category, learning plans or behaviors (Argall et al. 2009), the number of training samples generated by the human demonstrator is generally small since they are only generated when changing behavior or operation. The second category, learning motions or paths (Pastor et al. 2009), often have a large number of samples available as they are generated each time the trajectory changes. HiTAB falls into the first category and so it is focused on effectively learning behaviors from a small number of training samples.

Multiagent LfD produces additional very difficult challenges as previously discussed, and as a result is only lightly researched. In (Chernova and Veloso 2010), robots were trained to cooperatively sort colored balls into the appropriate bin where the robots requested additional demonstration when uncertain of correct action. Additional LfD multi-agent learning involves learning from the joint demonstration of multiple trainers. For example, in (Martins and Demiris 2010) an approach was developed where the individual sequence of actions for each robot are captured and then the sequence of group behaviors is determined through analysis of the individual action sequences over space and time. In (Blokzijl-Zanker and Demiris 2012), robots learn to collaboratively open a door by extracting a template for the behavior and adapting it to doors in other settings. These methods work well for small teams, but become dramatically more complex as more robots are added.

Learning Formation Behaviors

Formations are a well studied problem in learning coordinated multi-agent behaviors, many of them using motor schema (Balch and Arkin 1998) or other potential field based approaches. In (Das et al. 2002), formation control leverages multiple controllers based on vision sensors on all agents. More recent literature has focused on new potential field methods of formation control for swarms (Barnes, Fields, and Valavanis 2009). The formations is this paper, trained as leader-referenced formations, are not intended as a better method of formation control, but as an interesting test problem for heterogeneous multiagent LfD that can be extend to swarm-like behaviors.

Background on HiTAB

HiTAB was introduced in (Luke and Ziparo 2010) originally as a single-agent LfD method for training individual agent behaviors that addresses domain space complexity. HiTAB learns behaviors in the form of hierarchical finite-state automata (HFA), where the states are either atomic agent behaviors or lower level HFA learned earlier. The HFA are defined through manual decomposition of the desired top-level behavior, and the lowest level in the HFA only have atomic agent behaviors as states. For each HFA, the trainer manually selects the required states (subbehaviors) and features, or agent sensors, needed to determine the transition between states. Because the states and features are manually selected by the trainer, HiTAB is only learning the transition function for the HFA. By default the learning is in the form of a C4.5 decision tree.

Behaviors and features may be parameterized and be bound to a *target*, which is some object in the environment. An example for a feature is *DistanceTo(A)*, which is bound the target *ClosestAgent* by the trainer to get the *distance to the closest agent*. An example for a behavior is a trained behavior *Goto(A)*, which is bound to the target *Home Base* resulting in a *go to home base* behavior. Parameters may also be bound to a variable of the HFA so that the learned HFA is parameterized, such as *SpreadBetween(A,B)* defined later in this paper.

HiTAB also has a special atomic behavior called *Done*. Done sets a Done flag, which is accessible through the *Done* feature, and immediately transitions to Start and the flag remains set unless it is specifically clear or the top-level agent behavior is changed. This is useful when some behavior has to be completed before another begins. Done is also special in that it is also an atomic behavior for training controller agents.

Individual Agent Training

When decomposition of the HFA is complete, features have been selected, and parameters bound, the demonstrator can then begin training. While in *training mode* the trainer teleoperates the agent, changing behaviors at the appropriate time. Whenever a behavior is changed, a training sample is created containing the current behavior, the current feature values when the behavior was changed, and the new behavior. Behaviors that are meant to continue until the next behavior change by the trainer will add an additional *continuation* sample with the new behavior, the feature values, and the new behavior again.

The trainer switches to *testing mode* when all training samples have been created, causing the transition function of the HFA to be learned. The trainer then observer the trained behavior in operation and saves it to the behavior library if it is working correctly. Otherwise, the trainer switches back to training mode and provides additional samples and repeats the test mode step.

Homogeneous Multiagent Training

In (Sullivan and Luke 2012), HiTAB was extended to homogeneous multiagent training with the introduction of *virtual controller* agents responsible for coordinating a subordinate group of agents. HiTAB again models the controller's behavior as an HFA, which is decomposed so that at the lowest level the HFA only contain only the atomic behaviors of the controller.

These atomic behaviors correspond to the top-level behaviors of each of the agents in the controller's subordinate group. As such, atomic controller behaviors manipulate the subordinate agents rather than the controller itself, with a transition in the controller HFA directing all subordinates to change their behavior. Controller features are programmed by the trainer to provide statistical information from the features or states of the subordinate agents. As with individual agents, behaviors and features may be parameterized.

To increase cooperation among the homogeneous individual agents, a hierarchy of controllers may be trained. For a higher level controller the training method is the same, with its the atomic behaviors being the trained behaviors of the virtual controller agents in its subordinate group.

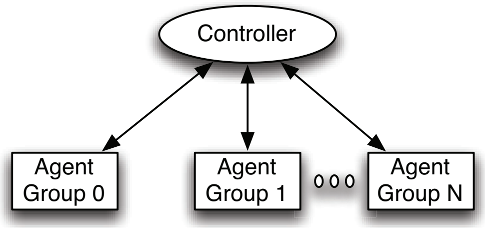

Figure 2: Heterogeneous Controller

Heterogeneous Multiagent Training

In (Sullivan et al. 2015), HiTAB was further extended to heterogeneous multiagent training, where virtual controllers may have more than one subordinate agent group as shown in Figure 2. The subordinate groups of a heterogeneous controller each contain a different agent type. Again the coordinating behavior is represented as an HFA, which is decomposed so that the lowest level HFA has only the atomic behaviors of the controller.

Each atomic behavior for heterogeneous controllers is now a *joint behavior*, which is some permutation of trained behaviors of the subordinate agent groups. A *continuation* behavior may be defined for one or more of the agent groups in the joint behavior, meaning that all agents in that group should continue the previously directed behavior. As in the homogeneous case, all agents within a single subordinate group are running the same behavior. Depending on the number of agent groups and the number of trained behaviors within subordinate agents, the number of joint behavior permutations can be quite lengthy. Additionally, a given permutation of behaviors may not be meaningful to the trainer in the context of the training problem. For this reason it is left to the trainer to define the joint behaviors for the controller, and consequently presented with a meaningful and minimal set of atomic behaviors during training.

Joint behaviors take the form *Name(Behavior 0, Behavior 1, ..., Behavior N)*, where *Behavior i* is a trained behavior of subordinate agent group *i* . For example, Init(Face(X), Surround(X), MoveBetween(X, Y)) is a joint behavior named Init that tells agents in group 0 to Face some target X, agents in group 1 to Surround some target X, and agents in group 2 to MoveBetween two targets X and Y. If a continuation is specified for an agent group, the subordinate behavior is left blank (□), for example SpreadAgents(□, AdjustSeparation(X, Y), SpreadBetween(X, Y)) has no behavior defined for agents in group 0.

While the work in (Sullivan et al. 2015) provided a successful demonstration of training heterogeneous controller agents it falls short of the research under way in this work. The controller hierarchy, shown in Figure 3, is two levels deep where each agent group has a single agent. While this is heterogeneous and mutliagent by definition, it did not demon-

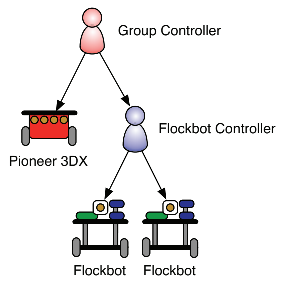

Figure 3: Box-pushing Controller

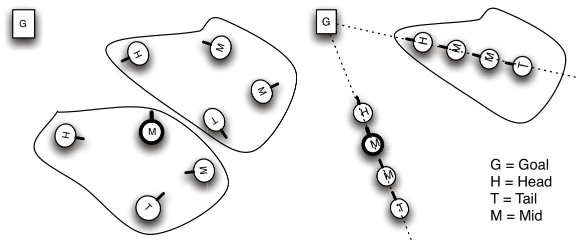

Figure 4: Column Formation

strate training of heterogenous behaviors that scale to some unknown number of agents in operation. Also, because each group contained a single agent there was no need for feature aggregation, which is necessary in many heterogeneous LfD training problems.

Our Extensions to HiTAB

Group Features

Introduced in this work, a *group feature* applies an *aggregator function* to a feature value for all agents within an agent group, or from agents in all subordinate groups if a group isn't specified. An aggregator function is a simple statistical function such as Max, Min, Average, and Range. Group features have the form *Name(group, aggregator, feature)*. For example, MostDistant(0, Max, DistanceTo(X)) defines a group feature named MostDistant whose value is the maximum value of the DistanceTo(X) feature for agents in group 0. For basic agent groups, a group feature can be defined for any feature of the basic agent or a feature common to all subordinate groups when a group is not specified. For controller agent groups, a group feature can be defined for any group feature of the subordinate controller. Because feature values are passed up the hierarchy, this means that group features may need to be defined in a lower level of the hierarchy even though they aren't needed for training coordinated behaviors at that level.

Targets with Context

Targets referencing other agents in the heterogeneous setting may require additional context that wasn't necessary in homogeneous training. For example, the SpreadBetween behavior, which moves an agent between one or more targets X and Y, may require targets X and Y to actually be bound to agents in groups 0 and 1 respectively. For the purposes of this work, the following targets have been defined:

- **Closest Agent - My Group** is the closest agent within the agent's own group.
- **Closest Agent - Group *n*** is the closest agent in subgroup *n* of the agent's controller.
- **Closest Agent - My Parent** is the closest agent in all subgroups of the agent's controller.
- **Second Closest Agent - My Parent** is the second closest agent in all subgroups of the agent's controller.

Joint Behaviors with Parameter Conflicts

Joint behaviors specify one behavior per subordinate group; these behaviors may also be parameterized. When multiple behaviors in a joint behavior are parameterized it is possible that parameters with the same name are not to be bound to the same target. In such cases the trainer can select one of the conflicting behaviors and trivially train a new behavior binding the parameters in the heterogeneous setting. Such conflicts are described in the *InitHT*, *MoveMids*, and *SpreadMids* joint behaviors in the ***Heterogeneous Controller Behavior Training*** section.

For example, in MoveMids the *MoveAwaySome(X,Y)* behavior has parameters that conflict with *MoveBetweenAvoid(X,Y)*, so it is trained for a Mid agent in the heterogenous setting by binding parameter A to *Closest Agent - My Group* and parameter B to *Closest Agent - Group 0*. The trainer then: starts training, selects the MoveAwaySome behavior, ends training, saves the behavior as *CreateSpace*, and replaces *MoveAwaySome(X,Y)* in the joint behavior with *CreateSpace*.

Training Column Formation Behaviors

In this work, training has focused on creating four line formations: column, row, echelon right, and echelon left with respect to some *Goal* in the environment. The controller agent for these problems has three subordinate agent groups to train according to the different agent roles in the formation: Head, Tail, and Mid agents whose agent groups are indexed 0–2 respectively. Individual agents are initially distributed randomly in the environment as shown on the left in Figure 4, and each line formation behavior coordinates the agents through a series of steps to position and orient them with respect to the Goal. The general steps to creating a line formation are shown below.

1. **Initialize** by orienting the Head agent at the Goal and positioning the Tail agent such that the angle between the Goal and Tail from the perspective of the Head agent is some value, depending on the formation.
2. **Move Mid Agents** between the Tail and Head agents
3. **Spread Mid Agents** between Tail and Head and adjust the distance from the Tail to the Head to achieve the desired agent spacing.
4. **Orient Mid and Tail Agents** like the Head agent.

Using this approach, the only difference in training between the four line formations is the **Initialize** step. The desired result of the column behavior is show on the right of Figure 4.

Individual Behavior Training

While the training of the individual behaviors is not the focus of this paper, there are some important points to make about training individual behaviors in light of the overall training problem. First, training complexity in individual behaviors is much preferred over training complexity in controller behaviors because we wish to maximize agent autonomy and consequently minimize communication with the controller. Second, training individual behaviors should should be parameterized to promote reuse. For example, the *MoveBetween* behavior for Mid agents is trained to move the agent between points A and B. The parameters A and B are bound later to *targets* in the heterogeneous setting as described in the next section. In total, we trained 26 individual behaviors to support the line formation behaviors. The behaviors listed below are the individual behaviors used in the definition of joint behaviors for the controller.

- **AlignFront**(X): Orient the agent so that it is facing the target X.
- **GotoAvoid**(X): Move the agent toward the target X while avoiding objects or agents in its path.
- **CircleAvoid**(X): Move the agent in a circle around the target X, with avoidance, in a clockwise direction. The radius of the circle is set as the distance to X at the time the behavior starts.
- **CircleNegAvoid**(X): Move the agent in a circle around the target X, with avoidance, in a counter-clockwise direction. The radius of the circle is set as the distance to X at the time the behavior starts.
- **MoveAwaySome**(X, Y): When the distance to the target X is below a threshold, move the agent a short distance from the opposite direction of target Y, with avoidance.
- **MoveCloserTo**(X, Y): Move the agent in the direction of target Y when the distance to target X is greater than some threshold.
- **MoveBetweenAvoid**(X, Y): Move the agent onto the line between two targets X and Y, with avoidance.
- **SpreadBetween**(X, Y): Move the agent equidistant between any agent in its group or one of the endpoints X and Y if the agent has no other agent between itself and X or Y.
- **AlignLike**(X): Orient the agent in the same direction as the target X.

The state machines used in training two of the more complex individual behaviors are shown in Figure 5 with the associated features defined below the corresponding state HFA. The LineAssoc behavior is intended to run after an agent has moved onto the line between two points (X, Y) and it is assumed that there is at least one other agent on the line and in the same agent group as the training agent. This behavior ensures that the agent is associated with an endpoint if it is directly adjacent, meaning there is no other agent between the agent and the endpoint, and otherwise associated with the closest agent in its agent group. Once the association is made the agent will remain in the Done state until the behavior is changed. The DirectionSimilarity(A,B) feature returns the cosine similarity between the vectors from the agent to A and the agent to B. With all agents being on the line, DS returns a value close to 1 when the closest agent and X are in the same direction and close 0 otherwise. Training LineAssoc was completed using 29 training samples.

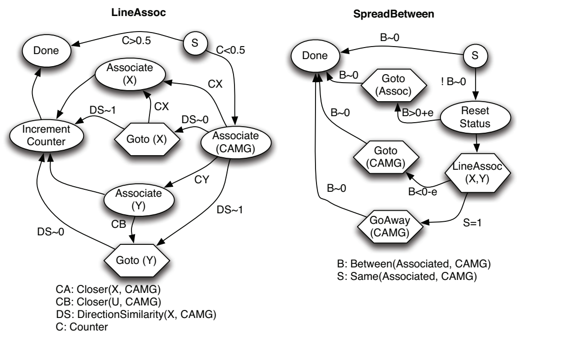

Figure 5: The *LineAssociate* and *SpreadBetween* HFA

The SpreadBetween behavior repositions an agent so that it is equidistant between the associated object and the closest agent in the same agent group. Because it is utilizing the LineAssoc behavior, the associated object is either one of the endpoints or another agent on the line between them. The Between(X, Y) feature returns a value between -1 and 1 with the zero value occurring at the point equidistant from X and Y. If X and Y are the same object, then zero is returned. Training SpreadBetween was completed using 15 training samples.

It is important to note that the Done behavior sets the Done flag and immediately transitions to Start, which is why additional transitions from Start to Done were trained. Also, since a behavior can only be bound to one target a behavior sometimes has to be trivially trained (with no transitions) and saved under a different name. Thus the Goto/Goto2 and Associate/Assoc2/Assoc3 behaviors.

Heterogeneous Controller Behavior Training

Heterogeneous controller agent behaviors use the same behavior decomposition, feature selection, and training method as individual behaviors and homogeneous controllers. However, there are a few additional steps required by the trainer for heterogeneous controllers.

1. Define joint behaviors from the trained individual behaviors of subordinate agents.
2. Perform trivial training of individual behaviors to eliminate parameter conflicts and update joint behaviors to reference the new behaviors.
3. Define group features
4. Bind joint behaviors to targets

5. Bind group features to targets
6. Train controller behavior FSA based on joint behaviors and group features.

Before describing the joint behaviors and group features, it is helpful to define a shorter and more descriptive notation for the common targets for group features and joint behaviors.

- Goal (A): The goal is bound to the parameter A.
- Head (H): The Head agent is *Closest Agent - Group 0*.
- Tail (T): The Tail agent is *Closest Agent - Group 1*.
- Closest Mid (CM): The closest Mid agent is *Closest Agent - Group 2*.
- Closest in Formation (CF): The closest agent in the formation is *Closest Agent - My Parent*.
- Second Closest in Formation (SCF): The second closest agent in the formation is *Second Closest Agent - My Parent*.

Joint Behaviors Seven joint behaviors are defined as atomic behaviors for the training of line formations.

- **InitHT**(AlignFront(X), GotoAvoid(X), □): To eliminate a parameter conflict, Goto(X) is trained as Goto(H) for the Tail agent and saved as GotoHead. The updated joint behavior is InitHT(AlignFront(X), GotoHead, □).
- **AlignTail**(□, CircleAvoid(X), □).
- **AlignTailNeg**(□, CircleNegAvoid(X), □).
- **TailDone**(□, Done, □): Note that Done is a special non-trained behavior that can referenced in a joint behavior.
- **MoveMids**(□, MoveAwaySome(X, Y), MoveBetweenAvoid(X, Y)): To eliminate a parameter conflict, MoveAwaySome(X, Y) is trained as MoveAwaySome(CM, H) for the Tail agent and saved as CreateSpace. The updated joint behavior is MoveMids(□, CreateSpace, MoveBetweenAvoid(X, Y)).
- **SpreadMids**(□, MoveCloserTo(X, Y), SpreadBetween(X, Y)): To eliminate a parameter conflict, MoveCloserTo(X, Y) is trained as MoveCloserTo(CM,H) for the Tail agent and saved as AdjustSeparation. The updated joint behavior is SpreadMids(□, AdjustSeparation, SpreadBetween(X, Y))
- **AlignLikeHead**(□, AlignLike(X), AlignLike(X)): There is no parameter conflict in this case.

Group Features Five group features are defined for the training and operation of the heterogeneous controller FSAs.

- **TailAligned**(0, Average, RelativeDirection(X, Y,Z)): Returns the angle between vectors XY and XZ as measured by the Head agent.
- **MidsBetween**(2, Max, DistanceBetween(X, Y)): Returns the maximum distance from Mid agents to nearest point on the line between X and Y.
- **MidsDistributed**(2, Max, DistanceTo(X)): DistanceTo returns the distance from the Mid agents to X.
- **TailProximal**(1, Max, DistanceTo(X): Returns the distance from the Tail agent to X.
- **MidsDone**(□, Min, Done): Returns 1 if the Done flag is set for all agents in the formation.

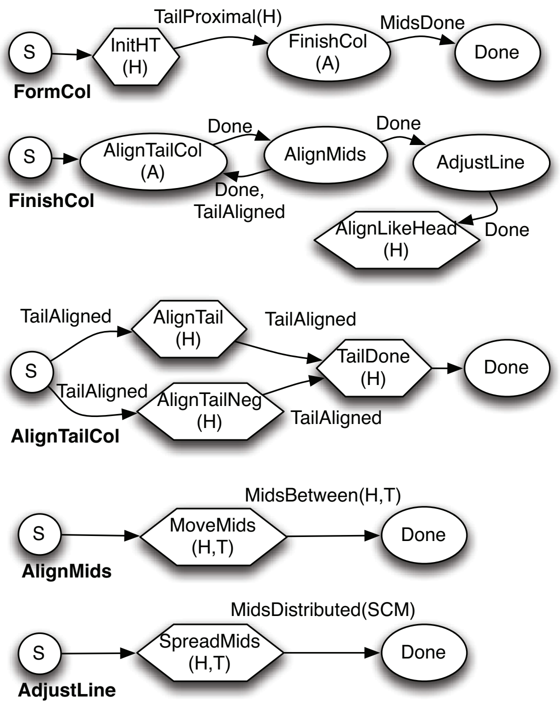

Figure 6: Column Formation HFA

Training and Results

Training was successfully performed for each of the four line formation behaviors. Training of the column formation behavior, FormCol, was completed by training the HFA shown in Figure 6 from the bottom up as pictured. Joint behaviors are represented by a hexagonal shape while trained behaviors (and Done) are elliptical shapes. The targets of the behaviors and features are noted in parentheses. The results of a run of the behavior in the HiTAB simulator with 3 Mid agents are shown in Figure 7 where the Home Base marker is selected as the goal. Training of the entire controller HFA was accomplished with a total of 36 training examples, the bulk of which were used to train AlignTailCol (18) and FinishCol (12).

Training the other formations is very similar by following these steps steps below. The column formation only differs in that the last two steps are not required.

1. Position Head near target and orient so that angle with respect to the goal is that of the desired line formation.
2. Form the other agents in a line behind the head.
3. Orient the Head to face the goal.
4. Orient the other agents like the head.

Because there was some reuse of trained behaviors in column training the additional line formations only required 33 training samples each even with the two additional steps. The goal of this work wasn't to minimize the number of training samples, but the totals indicate the efficacy of the manual

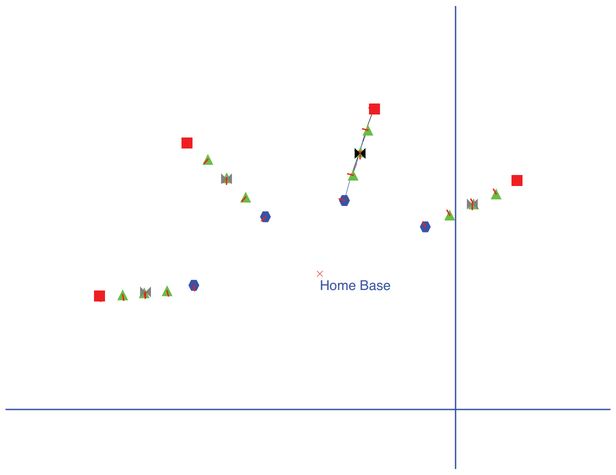

Figure 7: Column Formation in HiTAB

behavior decomposition to address the inverse problem for heterogeneous controller training.

After we trained the behaviors, we ran the learned behavior with different numbers of middle agents. As expected, the formation behavior with additional agents formed a longer line. The Line formation behaviors have the limitation that the controller hierarchy is fixed in size. So while this work accomplished the goal of training heterogeneous line formations that scale as the number of Mid agents grows, it does not (and cannot) grow the number of virtual controllers to effectively use additional Head or Tail agents.

Training the behaviors is complicated a problem related to behavior decomposition and feature selection. When we decompose the behaviors, features are selected based on the state transitions in the HFA and the way those features are used in the learned transition function sometimes differ from the expectation of the trainer because of the underlying machine learning method. As decision trees are the default method to learn the transition function, the small number of training samples often introduces some randomness in choosing which feature is determined to have the most information gain. This becomes more problematic when floating point features are used since they will often have a different value for all training samples. This can be overcome by decomposing behaviors so that they have at most one floating point feature. Using the *Done* behavior allows a higher level HFA to use the *Done* feature in training rather than a floating point feature. The decomposed HFA in Figure 6 and Figure 5 reflect this approach.

Conclusions and Future Work

This work demonstrates the training of complex heterogeneous multiagent behaviors using HiTAB. Specifically, we trained heterogeneous virtual controllers which coordinated subordinate agents to produce four different line formations. Without retraining, the behaviors continue to operate correctly when the number of agents is increased. This training required substantial effort on the part of the trainer to manually decompose the controller behavior, define joint behaviors and group features, and finally to train the controller behavior. However, the training required a very small number of samples and no special purpose code.

For future work, we will focus on training heterogeneous behaviors where controller agent groups may grow in size based on the number of agents available in operation. This is a complex problem since the hierarchy may be deep and unbalanced with a decision to be made at each level to expand the controller agent group. The basic agents then have to be effectively distributed in the expanded controller hierarchy.

Two types of test problem have been identified to further this research, N-deep formations and heterogeneous game scenarios. First, *N-deep formations* require a greater degree of coordination and will most likely present new challenges in terms of contextual agent targets and group features for controllers at level 2 and above in a controller hierarchy. As previously described, a grid formation my require growth of a controller agent group to expand the grid to effectively include the available agents. This problem can be extended to a line of grids, a wedge of grids, and so on.

Second, we will concentrate on *heterogeneous game scenarios*. While formations are an easily understood set of challenge problems for heterogeneous behavior training and have behaviors that can scale to the agents available, it is difficult to measure the effectiveness of growing the controller hierarchy. We will create a game scenario where resulting heterogeneous controller hierarchy can be grown to utilize additional agents and there is also some goal that can be measured. This will allow us to compare the effectiveness of our hierarchy growing algorithm to other methods.

Acknowledgments

This research in this paper was conducted with the support of research infrastructure developed under NSF grant 1727303.

References

Argall, B. D.; Chernova, S.; Veloso, M.; and Browning, B. 2009. A survey of robot learning from demonstration. *Robotics and Autonomous Systems* 57(5):469–483.

Atkeson, C. G., and Schaal, S. 1997. Robot learning from demonstration. In *ICML*, volume 97, 12–20.

Balch, T., and Arkin, R. C. 1998. Behavior-based formation control for multirobot teams. *IEEE Transactions on Robotics and Automation* 14(6):926–939.

Barnes, L. E.; Fields, M. A.; and Valavanis, K. P. 2009. Swarm formation control utilizing elliptical surfaces and limiting functions. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 39(6):1434–1445.

Blokzijl-Zanker, M., and Demiris, Y. 2012. Multi robot learning by demonstration. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 3*, 1207–1208. International Foundation for Autonomous Agents and Multiagent Systems.

Chernova, S., and Veloso, M. 2010. Confidence-based multi-robot learning from demonstration. *International Journal of Social Robotics* 2(2):195–215.

Das, A. K.; Fierro, R.; Kumar, V.; Ostrowski, J. P.; Spletzer, J.; and Taylor, C. J. 2002. A vision-based formation control framework. *IEEE Transactions on Robotics and Automation* 18(5):813–825.

Elston, J., and Frew, E. W. 2008. Hierarchical distributed control for search and tracking by heterogeneous aerial robot networks. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, 170–175. IEEE.

Luke, S., and Ziparo, V. A. 2010. Learn to behave! rapid training of behavior automata. In *Proceedings of Adaptive and Learning Agents Workshop at AAMAS 2010*.

Martins, M. F., and Demiris, Y. 2010. Learning multi-robot joint action plans from simultaneous task execution demonstrations. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, 931–938. International Foundation for Autonomous Agents and Multiagent Systems.

Pastor, P.; Hoffmann, H.; Asfour, T.; and Schaal, S. 2009. Learning and generalization of motor skills by learning from demonstration. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, 763–768. IEEE.

Pinciroli, C.; O'Grady, R.; Christensen, A. L.; and Dorigo, M. 2010. Coordinating heterogeneous swarms through minimal communication among homogeneous sub-swarms. In *International Conference on Swarm Intelligence*, 558–559. Springer Berlin Heidelberg.

Soule, T., and Heckendorn, R. B. 2010. A developmental approach to evolving scalable hierarchies for multi-agent swarms. In *Proceedings of the 12th Annual Conference Companion on Genetic and Evolutionary Computation*, 1769–1776. ACM.

Sullivan, K., and Luke, S. 2012. Learning from demonstration with swarm hierarchies. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, 197–204. International Foundation for Autonomous Agents and Multiagent Systems.

Sullivan, K.; Wei, E.; Squires, B.; Wicke, D.; and Luke, S. 2015. Training heterogeneous teams of robots. In *Autonomous Robots and Multirobot Systems (ARMS)*.