2015 IEEE 14th International Conference on Machine Learning and Applications

A Demonstration of Stability-Plasticity Imbalance in Multi-Agent, Decomposition-Based Learning

Sean C. Mondesire
Electrical Engineering and Computer Science
University of Central Florida
Orlando, FL. USA
sean@cs.ucf.edu

R. Paul Wiegand
Institute for Simulation and Training
University of Central Florida
Orlando, FL. USA
wiegand@ist.ucf.edu

Abstract— **Layered learning is a machine learning paradigm used in conjunction with direct-policy search reinforcement learning methods to find high performance agent behaviors for complex tasks. At its core, layered learning is a decomposition-based paradigm that shares many characteristics with robot shaping, transfer learning, hierarchical decomposition, and incremental learning. Previous studies have provided evidence that layered learning has the ability to outperform standard monolithic methods of learning in many cases.**

The dilemma of balancing stability and plasticity is a common problem in machine learning that causes learning agents to compromise between retaining learned information to perform a task with new incoming information. Although existing work implies that there is a stability-plasticity imbalance that greatly limits layered learning agents' ability to learn optimally, no work explicitly verifies the existence of the imbalance or its causes. This work investigates the stability-plasticity imbalance and demonstrates that indeed, layered learning heavily favors plasticity, which can cause learned subtask proficiency to be lost when new tasks are learned. We conclude by identifying potential causes of the imbalance in layered learning and provide high level advice about how to mitigate the imbalance's negative effects.

Keywords— Stability-Plasticity Dilemma, Layered Learning,

Decomposition-based Reinforcement Learning

I. Introduction

This work provides a demonstration that *layered learning*, a sequential, decomposition-based learning paradigm, can be affected by a stability-plasticity imbalance. The *stability-plasticity dilemma* is a common problem in machine learning that causes learning agents to struggle with finding the right balance of retaining important information used to perform previously learned behaviors and tasks while learning to solve new problems [1]. On the one hand, imbalances often result in the learner quickly losing proficiency in learned, older tasks in favor of new, incoming knowledge. While on the other hand, the imbalance can cause the system to struggle to adapt and perform newer tasks fast enough to thrive in a reinforcement environment. This work serves as the first empirical study of the imbalance in the abstract paradigm of layered learning. Furthermore, we investigate the causes and effects for the imbalance in layered learning.

The demonstration is needed because the paradigm is being successfully used by a subset of the machine learning and robotics community to develop the behavioral policies of autonomous agents that solve complex tasks without regard to such potential issues [2]. Additionally, layered learning shares many similarities with other established decomposition-based *reinforcement learning* (*RL*) techniques; these similarities further motivate this work to gain an understanding of the stability-plasticity dilemma in an abstract paradigm to uncover potential imbalance issues with other, more commonly used and domain-specific, algorithms that share core decomposition-based fundamentals with layered learning. Also, because other stability-plasticity studies examine very specific and specialized decomposition-based learning algorithms (typically including artificial neural networks), this work's main contribution is its investigation of the dilemma in a broader scope of reinforcement learning.

By analyzing the dilemma with a paradigm that can employ a diverse range of machine learning algorithms, the results of this work can be of value to similar learning methods. At the very least, this demonstration provides evidence that an imbalance can drastically affect learning efficiencies and encourage investigations in similar learning approach, an area that is lacking in attention in the decomposition-based reinforcement learning community.

In the presented work, we construct a direct-policy search multi-agent problem with general properties common to many RL problems. We use task and subtask performance to clearly demonstrate that layered learning can be seriously impacted by a stability-plasticity imbalance. We conclude by identifying causes of the imbalance and outline future work that aims to mitigate the negative effects of the imbalance phenomenon.

II. Background

Stone and Veloso [2] introduced layered learning as a machine learning paradigm that relies on task decomposition to simplify a complex task into manageable components (*subtasks*). This paradigm should not be confused with other learning techniques, such as multi-layered neural networks, that rely on layers in their own processes.

At a high level, layered learning is given a decomposition of one complex task from an external source, the developer. The process then takes the decomposition's series of simpler subtasks and modifies a decision-making policy to perform each subtask sequentially. In machine learning, a *policy* is a mapping of states to actions that determines how an agent be-

978-1-5090-0287-0/15 $31.00 © 2015 IEEE
DOI 10.1109/ICMLA.2015.106

haves. A policy ultimately determines an agent's effectiveness at performing desired tasks and is the target of learning.

A single layered learning instance, where an agent is responsible for learning a single complex task, is comprised of a collection of layers, the set L. The agent learns to perform the complex task by sequentially iterating through each layer in L. Defined in (1), each subtask learning phase is a *layer* (L_i), where the agent is given a training environment, performance evaluation function, and set of training examples designed for learning the *subtask* (T_i). The layer also accepts as input the set of all state variables for the learner to analyze its environment (F_i), a set of all possible actions the learning agent can make (O_i), and a machine learning technique to learn the subtask (M_i). The final input h_{i-1} is the initial policy that will be optimized to perform subtask T_i. Once the layer is complete, the outputted policy (h_i) is used as the initial policy for the subsequent layer (L_{i+1}), creating a dependency of learning among the layers. Finally, in each layer, the agent optimizes its policy to perform subtask T_i until a halting condition is satisfied. The halting condition determines when the agent transitions to the next layer to learn a new subtask.

$$L_i = (F_i, O_i, T_i, M_i, h_{i-1}) \quad (1)$$

Two configurations in which manner a policy can be developed in layered learning are layer isolation and layer overlapping. In *layer isolation*, independent sub-policies are adapted to learn subtasks, and the learning in one layer does not alter the sub-policy used in another. In *layer overlapping*, two or more layers can modify shared policy information. Intuitively, layer overlapping mechanisms run the highest risk of being impacted by a stability-plasticity imbalance, so we focus our attention on such cases. Still, there are advantages to using overlapping, including the removal of isolation's need for sub-policies, less design time dedicated to guaranteeing layers are truly isolated, reduction in redundant learning because subtasks can share basic, common information that does not have to be relearned, and increased freedom in how the policy transfers learned knowledge from one layer to the next.

Layered learning was introduced by Stone and Veloso [3] and extended in [2] to train simulated, autonomous agents to play the complex game of soccer. Since then, the learning paradigm has been used to develop the decision-making policies for a range of diverse problems.

Early layered learning works examine the feasibility of the paradigm and focus mostly on comparing the paradigm's ability to develop policies against standard, monolithic learning. Monolithic learning is a fair comparison control because in a learning scheme that employs it, there is no decomposition and the agent must learn the complex task in one learning phase, contradicting the core principles of layered learning. These works include those of Hsu and Gustafson's [4, 5] and Gustafson and Hsu's [6] layered learning development of Keep-away (a soccer variant) playing agents and Jackson and Gibbons' [7] study of layered learning's ability to solve Boolean-logic problems and the impact of using decompositions based on lower-ordered sub-problems. These studies demonstrate that layered learning can often rival monolithic learning in producing better performing policies that solve complex tasks.

Other works examine layered learning in other problem areas: Whiteson and Stone [8] introduced *concurrent layered learning*, a coevolutionary adaptation to layered learning, and Cherubini, Giannone, and Iocchi [9] and Fidelman and Stone [10] study the feasibility of using layered learning on real, physical robots. Both studies demonstrate that layered learning can be used to teach physical robots how to perform complex tasks.

The majority of layered learning works use layer isolation to safeguard layers from modifying state-action pairs of the policy relied on by other layers. Two negatives of using layer isolation are that 1) it requires developers to dedicate design time to plan for independent sub-policies to be developed and assigned to each layer and 2) the agent must be able to recombine all of the sub-policies into one policy to perform the overall task.

Two works that use the alternative to layer isolation, layer overlapping, are the aforementioned Jackson and Gibbons' [7] study of layered learning to solve Boolean-logic problems and Mondesire and Wiegand's [11] work that uses layered learning to train predators in a predator-prey scenario. Without explicitly studying stability and plasticity, this latter work's findings suggest that layered learning can suffer from a stability-plasticity imbalance. The study's experiment results show that as the predator agent learned a new subtask with layer overlapping, the performance of a previously learned subtask plummets, ultimately negatively affecting overall task performance. This loss in the oldest subtask's proficiency showed that layers can be detrimental to each other and the time and effort required to regain lost important information is wasted opportunity to search for more optimal solutions instead of repairing what was lost.

Unfortunately, there is no study in the literature that clearly and explicitly demonstrates that proficiency loss takes place as layered learning agents learn to perform new subtasks. Moreover, as layered learning grows in use and popularity, its true potential is limited by our lack of understanding of such performance losses. Because layered learning shares many features with other learning methods, there is much to be learned about challenges that face layered learning by examining the difficulties faced by other methods. Alternatively, perhaps deeper understanding can be gained in other methods by examining these issues in layered learning.

III. Methodology

The goals of this work are to demonstrate that layered learning with layer overlapping is susceptible to a stability-plasticity imbalance and to identify the causes of the imbalance. Through these experiments, this work uncovers that layered learning can easily favor plasticity, causing newly acquired knowledge to drive out critical information needed to perform subtasks of earlier layers. The driven out knowledge is *negatively forgotten* [12], which causes older subtasks' performance to decrease drastically. The loss of older subtask proficiency is significant because it negatively affects the

performance of the complex task that relies heavily on the lost information and the ability to perform its forgotten subtasks.

At the heart of layered learning are the layer transitions, which change the learning focus from one subtask to another. Experiment results conclude that these performance evaluation changes can negatively affect the learner's ability to maintain proficiency in older subtasks as newer ones are learned in decomposition with layer overlapping.

To demonstrate the susceptibility of a stability-plasticity imbalance in layered learning with overlapping, experiments applied to a multi-agent problem are conducted. The experiments are explicitly design to demonstrate that as the performance evaluation functions transitions from one subtask to another, the policy experiences drastic changes to its state-action pairs, subtask and task performance degrades, and computational effort is wasted when a policy attempts to regain lost proficiency.

Specifically, in this studied problem, each agent represents a quadcopter *unmanned aerial vehicle* (*UAV*) that must learn when and how to take off and land, move to a coordinate in the environment, refuel, and transmit reached coordinates to its UAV teammates. Quadcopters are similar to remote controlled helicopters with four rotors instead only one. They are known for their mobility and ability to change direction and heading quickly. The problem is deliberately abstracted to keep the focus on the study of plasticity.

A. The UAV Surveying Problem

In this surveying task, four UAVs are initially grounded at four different corners of the 50x50 two-dimensional discrete grid environment, depicted in Fig. 1. In each run of the simulation with this task, performance is evaluated on the percentage of unique grid coordinates the UAV team has navigated to out of the 2500 total number of cells in 2000 time-steps. At every time-step, each UAV determines its state, selects and performs an action with that corresponding state from its policy. UAVs have the movement abilities to take off, land, and move towards any coordinate in the environment, one cell at a time.

UAVs remain active in the simulation as long as they have fuel. With each time-step a UAV is in flight, its fuel level depletes by one unit. If a UAV's fuel level reaches zero, the UAV becomes inactive and can no longer execute actions, such as movement to survey the environment or refuel. To replenish depleting fuel levels, a UAV must navigate back to a refueling depot (located on the ground of each of the four corners of the grid), land, and explicitly execute the "refuel" action. Refueling replenishes the UAV 10 units per time-step the action is executed. Initially, a UAV has 100 units of fuel and refueling never exceeds a level of 100 units. Multiple UAVs can refuel from the same depot at the same time.

The surveying problem requires collaboration among the UAV teammates to maximize the task's performance: no single UAV can navigate to every cell in the environment in the limited amount of time and duplication with UAVs covering the same cell wastes time. To facilitate collaboration, each

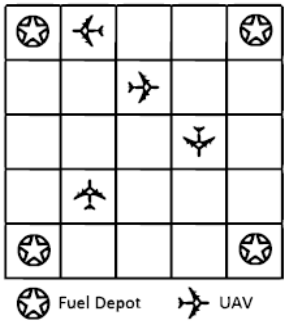

Figure 1. A screen capture of the UAV simulation (at 10th of the scale)

UAV maintains a list of coordinates the team has visited and may execute the action "transmit coordinates" to broadcast a complete history of all of the coordinates that UAV has reached during the simulation up until the current time-step. This broadcast is instant and is guaranteed to reach each of the other UAVs.

Because each UAV must decide when to transmit and when to perform other actions, a UAV also contains a buffer of coordinates it has surveyed since its last transmit action was executed. Sub-states, such as "is buffer size at least 5", exist so each UAV can determine their own thresholds of when to transmit unshared, reached coordinates with their teammates.

The problem is similar to those studied in other reinforcement learning and direct-policy search works, including UAV navigation and search-and-rescue problems. This UAV exploration problem was selected because of its task complexity, straight-forward decomposition into subtasks, requirement of UAV collaboration among teammates to solve the problem optimally, and direct mapping to problems used in the real world.

B. Learning Configuration

To learn to solve the problem, a decomposition of the UAV surveying problem and a layer configuration, including a subtask sequence, training scenarios, and halting conditions are used with layered learning to optimize the UAV policies. The employed decomposition is as follows: the agent team first learns to refuel in layer 1, take off and survey the environment in layer 2, and lastly, the overall task of surveying and refueling when necessary in the final, aggregate layer of layer 3.

The process of layered learning takes one set of policies, one policy for each UAV, and passes it into layer 1 for optimization on the refueling subtask. In that layer, a learning algorithm is used to optimize performance on the subtask by repeatedly running the refueling simulation and modifying the policies to map states to appropriate actions. This evaluation and modification process repeats for a set number of iterations, called epochs. We use the term *epoch* to denote a generation in the employed evolutionary algorithm. The refueling optimized policy set is then fed into the second layer, which does the same process as layer 1 but optimizes the policy set on the take off and survey subtask. Finally, the policy set outputted in layer 3 is optimized on the overall task. Through this decomposition, the aim is to gradually learn each important

aspect of the overall task in incremental phases of learning and have a policy set that solves the overall task at the end of the last epoch of layer 3.

C. Policy Representation:

Each UAV can perform exactly one action for each state that it is in at any given time-step. A UAV's state is the collective value of the UAV's internal conditions, called sub-states, similar to a Boolean array where each element corresponds to a specific sub-state value. A sub-state is a true or false value of a condition the UAV has at a time-step. For example, the sub-state "is grounded" is true if the UAV is on the ground, not in flight. At every time-step, each UAV evaluates all of its sub-states and represents that collection as a single state (similar to converting a Boolean array into a decimal value). Table I lists all of the UAV sub-states and table II lists all of the actions.

The policy is a direct state-action mapping that links each state to one of seven actions that UAV can perform. In this problem, there are 7 unique actions with 8 sub-states. The simulation ignores impossible or redundant actions, such as landing when already grounded or flying to a new coordinate when grounded. The actions are to move to the closes unvisited coordinate, take off, land, NOP (remain idle), move towards the closest fuel depot, refuel, and to transmit coordinates. For the states, each UAV checks its own conditions, which are the following: is my fuel level low, am I grounded, am I at a fuel depot, and is my fuel level above 90% full. Also, there are four states asking if the UAV's transmission buffer size was at least 1, 5, 10, or 20 coordinates full.

D. (1+1) Evolutionary Algorithm

A *(1+1) Evolutionary Algorithm* [*(1+1) EA*] is employed with layered learning to facility policy optimization over the UAV problem. A (1+1) EA is a simplified type of EA where one genetic operator is used to manipulate the only individual in the population. Borisovsky and Eremeev [13] and Wegener and Witt [14] studied (1+1) EA performance and provide additional details on the approach. The algorithm is chosen because it is a simple, yet effective, learning technique. Although the EA is well-studied, the stability-plasticity dilemma's effects and causes in this type of learning lacks investigation and understanding. Further, because we wish to focus on layered learning in general and not the underlying machine learning methods, we chose as simple of an example from direct-policy search technique as possible. These experiments demonstrate that although the employed learning algorithms can introduce their own biases toward stability or plasticity, it is layered learning's layer transitions that cause performance to negatively be affected by an imbalance with the decomposition-based approach.

The (1+1) EA uses a single "parent" candidate solution, generates a "child" candidate solution via a "mutation" genetic operator, and replaces the parent with the child if it performs at least as well as the parent. The sole parent is the set of policies of the team, where each policy represents the decision making capability of a UAV teammate.

For the (1+1) EA, a policy is represented as collection of state-action pairs, where no two pairs in a policy share a state. During an epoch, every reached pair corresponding to an activated state in the simulation has a 25% chance of being mutated by changing the pair's action to a random one. Through trial and error, this mutation rate produced a high enough exploration rate to find optimal subtask performing policies quickly. When an agent reaches a state that is not mapped in its policy, a new pair is created, linking that new state to a random action.

In the (1+1) EA, every layer is given 10,000 epochs to learn a layer's subtask, limiting the entire learning process to 30,000 epochs and simulations to learn the overall task. Each instance is evaluated over 100 runs to create a large enough sample size to reduce error.

The first layer trains the UAVs to refuel. The refueling subtask trains the agents to recognize when their individual fuel levels are low and how to navigate to the refueling depot, land, and refuel. This layer's simulation initializes the UAVs to be 20 cells away from a fuel depot with 20 units of fuel, to force the UAVs to be low on fuel. Each UAV receives .25 points for accomplishing each successive milestone of first navigating to a depot, then land, start to refuel, and continue to refuel until fuel level was above 90% full. Each simulation run lasts for 35 time-steps. To evaluate the entire policy set, performance for one simulation is the average score of each UAV, where 1.0 means the entire UAV team has mastered the subtask of refueling.

Under the second layer, the (1+1) EA optimizes the UAVs policies to learn to take off and survey. This second layer's subtask is the same as the overall UAV Survey task, except the simulation lasts 100 time-steps instead of 2000 and the maximum unique cells to reach is 396. Although the environment dimensions remains the same 50x50 grid, the reduction in the number of time-steps means the number of possible cells the team can reach is 396 (each UAV can reach a maximum of 99 cells after taking off in the time limit). Similarly, because the team is evaluated on the percentage of unique cells navigated to and because a refueling UAV would waste time refueling, the UAVs do not have to refuel to maximize performance of this subtask.

The limitations of this subtask make this layer critical and create layer overlapping. Because the simulation runs for 100 time-steps, the first and second layer shares states of when a UAV is low on fuel. The conflict occurs because the first layer's subtask relies on a UAV to refuel while the second layer's subtask relies on a UAV to survey the environment. Although it is acknowledged that the subtasks could be decomposed in a completely non-overlapping manner, the layer configuration is explicitly designed to uncover how layered learning responds to the forced conflict between desired actions with the shared states. Also, both of these subtasks are integral aspects of the overall aggregate task, optimized in the third layer. The decomposition allows the agents to master these critical and difficult subtasks individually instead of learning all of the required steps of each subtask all at once.

Table 1: AVERAGE POLICY SET COMPOSITION CHANGES PER LAYER

Comparison	Pair Changes		Unchanged Pairs		New Pairs	
	AVG	STD	AVG	STD	AVG	STD
Layers 1 & 2	14.58	4.65	42.06	7.25	65.27	9.75
Layers 1 & 3	21.77	5.49	34.87	6.84	83.20	10.52
Layers 2 & 3	39.55	10.52	82.36	12.33	17.93	7.81

Table 2: AVERAGE SUBTASK PERFORMANCE AFTER EACH LAYER

	Refuel	Take off & Survey	UAV Surveying Task
Layer 1	0.87063	0.02907	0.00408
Layer 2	0.36500	0.73896	0.12144
Layer 3	0.28875	0.43652	0.55824

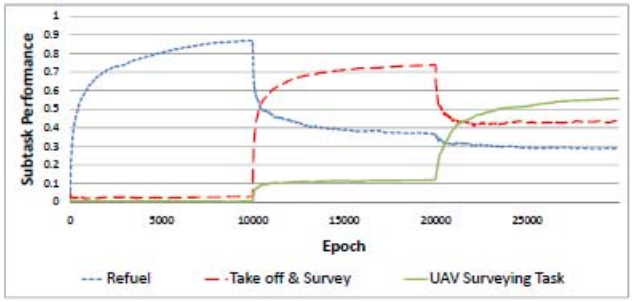

Figure 2: Average subtask performance.

IV. Results

The imbalance is evident in how the policies are modified to favor newer subtasks and in how their changes in composition altered proficiencies in older, learned subtasks. First, compositional changes are analyzed. Tables 1 displays the average state-action pair changes between a layer and one of its subsequent layers. From the experiment, an average of 14.58 pairs are changed between the last epoch of layer 1 and the last of layer 2. Therefore, of the approximately 56 shared pairs of layers 1 and 2, about 26% of them are modified in the optimization of layer 2's subtask. Similarly, 32% of the overlapped pairs of layers 2 and 3 are modified in the final layer.

These composition changes cause subtask performance of the first two layers to drastically decrease during the next, immediate layer. When learning how to refueling, the subtask's proficiency decreases from 87% of optimality earned at the end of layer 1 to 37% at the conclusion of layer 2. The second layer's subtask earns a proficiency of 74% optimality but drops to 44% in the overall task's layer. All of the average subtask proficiency changes can be found in table 2.

Additionally, subtask proficiency of the previous layer is observed to sharply decrease at the beginning of each new layer. Figure 2 plots subtask performance throughout the entire learning process, where each 10,000 epochs denotes a layer transition. This plot illustrates that after the first transition, re-fueling proficiency of 0.86 decreases rapidly before stabilizing around .38 for the remainder of layer 2. At the same time and consequently, performance of the take off and survey subtask increase quickly until halting at .74. A similar performance drop as the refueling case took place between layers 2 and 3, where the second layer's subtask decreases suddenly after the 20,000th epoch, when the performance evaluation criteria switches to the overall task.

Lastly, after observing that the oldest learned subtask from layer 1 was the most volatile in terms of subtask performance and composition changes, the number of pairs that are reverted to layer 1's state at the end of layer 3 is used as an indicator of the damage caused by its subsequent layers. For example, of the 14.58 pairs that are changed in layer 2, 13.33 are permanently damaged at the conclusion of learning, and remain different from the first layer. This high rate of permanent damage signifies that once a refueling pair is changed, the learning algorithm had difficult in reverting the change to restore lost proficiency.

V. Discussion

Although purposely designed to demonstrate an imbalance, these experiments show that not only do new layers modify existing pairs that are shared with other layers, but proficiency is modified as well. Because subtask proficiency is not retrained in this decomposition, subsequent layers have a high probability of modifying existing pairs. This occurrence is detrimental for subtasks that see their vital pairs modified in newer layers because, as the permanently damaged pair statistics have highlighted, the probability for repair is small, only 9%.

Furthermore, layered learning behaves greedily in these experiments by prioritizing policy changes that maximize the current layer's subtask over pair retention for older subtasks. This complete prioritization occurs because there is no incentive to keep pairs used in older layers, resulting in the learner purposely driving older, obtained knowledge out of a policy if it conflicts with a pair that improves the latest layer's subtask. This strong affinity for pair changes that lead the policy set closer to the latest layer's subtask optimality results in the learner to favor plasticity over stability; consequently, the affinity means imbalance can exist in the layered paradigm.

Additionally, the transitions to new layers with new sub-tasks have a traumatic effect on the policies trying to converge toward optimal performance. These transitions switch the learner's priority from optimizing one subtask to another, activating the greediness of the learning algorithm. The identification of these layer transitions being a significant factor of imbalance is noteworthy because these transitions are integral to layered learning; without the transitions, layered learning cannot guide policies toward solving complex tasks via the mastering of simpler problems and transfer learned knowledge from one layer to another.

VI. Conclusion

This work provides a demonstration of a stability-plasticity imbalance in the abstract, decomposition-based reinforcement learning paradigm of layered learning. Through explicitly designed experiments that provide the demonstration, we show that the paradigm can heavily favor plasticity, causing the learner to drive out important knowledge used to perform older subtasks in favor of knowledge needed to optimize more recently learned subtasks.

Three main causes for the imbalance are identified. First, although hypothesized and not surprising, layer overlapping's ability to share state-action pairs creates the high likelihood for pairs needed in an older layer to be modified by other, subsequent layers. This characteristic creates the possibility for layers to drive out knowledge that can drastically affect a previously learned subtask's performance. Second, without incentive to keep older pairs intact, the paradigm and the learning algorithm drive out knowledge that conflicts with newer subtask performance. Finally, combined with the learner's greedy characteristics and lack of incentive to retain older knowledge, layer transitions change the learner's priority and sets what the learner will optimize. With the lessons learned from these experiments, researchers and developers who rely on layered learning and similar decomposition-based reinforcement learners now know the pitfalls of layer overlapping and the causes of the imbalance.

Future work seeks to mitigate the imbalance's negative effects on layered learning. Therefore, upcoming research will evaluate existing techniques relied on in artificial neural networks for efficacy in subtask proficiency retention and overall task performance, such as subtask rehearsal and concurrent subtask learning. We target rehearsal and concurrent learning because it is clear from the work in this paper that mitigation techniques should provide a means of retaining and re-introducing particularly successful portions of policies; our future work relies on this idea directly.

VII. Acknowledgement

The authors would like to thank the Florida Education Fund and the McKnight Fellowship for their support. Additionally, the authors thank and appreciate the high performance computing capabilities received from the Advanced Research Computing Center (ARCC) in the Institute for Simulation & Training at the University of Central Florida. Access to STOKES has been instrumental to the research performed in this work.

References

[1] M. McCloskey and N. Cohen, “Catastrophic interference in connectionist networks: The sequential learning problem,” The psychology of learning and motivation, vol. 24, pp. 109–165, 1989.

[2] P. Stone and M. Veloso, “Layered learning,” in Proceedings of the Eleventh European Conference on Machine Learning. Springer Verlag, 1999, pp. 369–381.

[3] P. Stone and M. Veloso, “A layered approach to learning client behaviors in the robocup soccer server,” Applied Artificial Intelligence, vol. 12, pp. 165–188, 1998. [Online]. Available: http://www.cs.utexas.edu/users/ai-lab/pub-view.php?PubID=126580

[4] W. H. Hsu and S. M. Gustafson, “Genetic programming for layered learning of multi-agent tasks,” in 2001 Genetic and Evolutionary Computation Conference Late Breaking Papers, E. D. Goodman, Ed., San Francisco, California, USA, 9-11 July 2001, pp. 176–182. [Online]. Available: http://www.cs.nott.ac.uk/smg/research/publications/gecco-2001.pdf

[5] W. H. Hsu and S. M. Gustafson, “Genetic programming and multi-agent layered learning by reinforcements,” in Genetic and Evolutionary Computation Conference. Morgan Kaufmann, 2002, pp. 764–771.

[6] S. M. Gustafson and W. H. Hsu, “Layered learning in genetic programming for a cooperative robot soccer problem.” in EuroGP, ser. Lecture Notes in Computer Science, J. F. Miller, M. Tomassini, P. L. Lanzi, C. Ryan, A. Tettamanzi, and W. B. Langdon, Eds., vol. 2038. Springer, 2001, pp. 291–301.

[7] D. Jackson and A. Gibbons, “Layered learning in boolean gp problems,” 148–159, 2007.

[8] S. Whiteson and P. Stone, “Concurrent layered learning,” in Proceedings of the Eleventh European Conference on Machine Learning. Springer Verlag, 2003, pp. 369–381.

[9] A. Cherubini, F. Giannone, and L. Iocchi, “Robocup 2007: Robot soccer world cup xi,” U. Visser, F. Ribeiro, T. Ohashi, and F. Dellaert, Eds. Berlin, Heidelberg: Springer-Verlag, 2008, ch. Layered Learning for a Soccer Legged Robot Helped with a 3D Simulator, pp. 385–392.

[10] P. Fidelman and P. Stone, “Layered learning on a physical robot,” 2005.

[11] S. Mondesire and R. P. Wiegand, “Evolving a non-playable character team with layered learning,” in IEEE Symposium on Computational Intelligence in Multicriteria Decision-Making (MDCM), 2011, pp. 52–59.

[12] S. Mondesire and R. P. Wiegand, “Forgetting classification and measurement for decomposition-based reinforcement learning,” in Proceedings of The 15th International Conference on Artificial Intelligence (ICAI’13), 2013.