Forgetting Classification and Measurement for Decomposition-based Reinforcement Learning

Sean Mondesire[1] **and R. Paul Wiegand**[2]
[1]College of Engineering and Computer Science, University of Central Florida,Orlando, FL, USA
[2]Institute for Simulation & Training, University of Central Florida,Orlando, FL, USA

Abstract— *Forgetting is a memory phenomenon that affects knowledge used to perform behaviors and tasks. In this work, two metrics are presented to aid the diagnosis of forgetting in decomposition-based reinforcement learning systems. With these metrics, developers now have the ability to classify types of forgetting, identify when important knowledge is lost, located wasted computation effort, and verified if a task decomposition is best suited for learning a given task. Through monolithic and decomposition-based learning experiments, the validity of these metrics is examined and recommendations are made about when these metrics are most effective.*

Keywords: Forgetting metrics, reinforcement learning.

1. Introduction

This paper presents two new metrics that classify and measure certain types of *forgetting* challenges that can occur in decomposition-based reinforcement learning. The proposed metrics contribute to this learning domain by providing a method of measuring the effects of learning a new subtask on knowledge used to perform an older, previously learned subtask when forgetting occurs. The proposed metrics supplement traditional measurements since they relying solely on task and subtask performance data. Furthermore, the metrics weigh and aggregate the performances of all subtasks to make it convenient to identify and measure the type of forgetting that has taken place.

Once it occurs, it is important to classify and measure forgetting to identify inefficiencies in the learning system and the learning process it employs. Most importantly, because forgotten knowledge can affect the ability to perform a task, the proposed metrics determine if forgotten knowledge has a positive, negative, or neutral effect on performance. Next, the metrics can be used to identify subtask performance dips or stagnation in the context of all of the subtasks. Affected subtask performance can then be targeted for subtask retraining with the goal of recovering lost knowledge and proficiency. This simple analysis is made possible through the easily produced quantifiable values the metrics generate.

The two presented metrics are purely performance-based and can be applied to a wide variety of problems and underlying metrics; however, unlike traditional performance functions, these metrics use performance histories. This use of subtask performance history provides additional details of changes over time that occur at low-decomposition levels that can easily be missed by more immediate performance functions. Secondly, these proposed metrics aggregate all of the subtasks of a task decomposition to determine the significance of a change in knowledge prior to and post-forgetting. These aggregations may differ from a performance function used in a reinforcement learner's evaluation phase but the proposed metrics are used to primarily measure the difference in performance of altered, forgotten knowledge.

For example, in the decomposition-based paradigm of layered learning [1], the learning system learns a complex task by decomposing the task into multiple subtasks and sequentially learns to perform each subtask one-by-one. If the complex task is to play the game of soccer, a simple task decomposition is to first learn to pass the ball, receive a pass, and then to play a full game of soccer. The problem arises when the learner transitions from one subtask to another. As the soccer-playing system learns the third subtask of playing the full game of soccer, the ability to perform the subtask of passing the ball may be affected. In particular, the learner may forget some of the knowledge required to pass the ball as effectively as it did before it started to learn to play the full game.

In general, the two new metrics are intended to be deployed when it is suspected that performance of a sequential decomposition-based reinforcement learning system is affected by forgetting as the system iterates through a task decomposition. The goal is that these metrics will be able to identify periods in the learning process where forgetting causes performance to stagnate or suffer and to quantify the effect forgetting has on the system. With these tools, system developers will be able to analyze task decompositions, performance functions, and subtask transitions to identify inefficiencies and optimize their learning system designs.

To validate the metrics, we construct artificial problems that induce such forgetting challenges. These problems are not reinforcement learning problems but they abstract key aspects common to learning problems that are affected by forgetting. From our experiments, we compare the results of using the two forgetting metrics and make recommendations about when they might be most effectively used.

2. Background

In natural science, knowledge is information used to make a decision, perform a behavior, or have familiarity on a subject. Similarly, in machine learning, knowledge is the basis of all decisions. In this science, knowledge is represented by stimulus-response pairs that determine the conditions certain actions will be performed by the system.

Because acquired knowledge determines how well a behavior is performed, knowledge can be classified based on value. *Beneficial knowledge* leads a system towards optimal performance of a behavior. *Disadvantageous knowledge* is stimulus-response mapping that hinders behavior performance. *Unhelpful knowledge* is information that can be removed from the system with no positive or negative effect on performance.

According to Markovitch and Scott [2], the value of knowledge has several factors, including relevance, correctness, memory requirements, and influence on search time. Also, the importance of knowledge is dependent on other knowledge in the system. These five factors influence to which category an item of knowledge belongs.

Forgetting is the loss or modification of knowledge from a system's memory that may affect the performance of a behavior. Because memory is limited in machine learning, forgetting is an important mechanism that reconciles the storage of new information with old, acquired knowledge.

The act of forgetting can be caused by an explicitly invoked mechanism or an implicit side-effect of learning. Explicit forgetting mechanisms purposely drive out targeted acquired knowledge. A deletion strategy is an example of an explicit forgetting mechanism that removes knowledge from a system to increase behavioral performance or to free memory for new knowledge [3]. Markovitch and Scott's [2] randomly deleting knowledge items and Koychev's [4] and Nakayama and Yoshii's [5] time-based forgetting are examples of deletion strategies that have increased behavioral performance. Both strategies explicitly select and remove knowledge from a system's *policy* (the collection of a system's knowledge) with the goal of increasing behavioral performance.

Implicit forgetting, or *concept drift*, occurs when knowledge is lost as a consequence of learning [6]. Concept drift can happen when knowledge becomes outdated, inadequate, or loses performance due to a changing behavioral requirement. For example, in recommender systems, drift occurs when a user's preferences change, affecting the system's ability to identify accurate recommendations with outdated knowledge. A second implicit forgetting example is when a system's old knowledge or access to old knowledge is modified with the acquisition of new knowledge. The same as explicit forgetting, the unintended change to old knowledge can affect behavioral performance observed prior to new knowledge acquisition.

Similar to the classification of knowledge, forgetting can be classified as positive, negative, or neutral when based on performance. *Positive forgetting* is the loss or modification of knowledge that occurs when performance is increased compared to performance prior to the forgetting. *Negative forgetting* occurs when performance decreases when something is forgotten. *Neutral forgetting* is when knowledge is lost or modified and no change in performance takes place. This forgetting classification is directly related to knowledge being beneficial, unhelpful, or disadvantageous. By these definitions, if unhelpful knowledge is forgotten, then neutral forgetting occurs; if beneficial knowledge is forgotten then negative forgetting occurs; if negative knowledge is lost, positive forgetting occurs.

Performance change is not always caused by the loss of knowledge and performance is not always an accurate classifier for forgetting. Randomness, dynamic environment, and other external factors can affect the performance of a task. Additionally, depending on the system, the task, and its environment, beneficial and disadvantageous knowledge may be removed from a policy and not be reflected in a performance. With these points, classifying forgetting solely based on performance is best suited for instances of when factors, such as randomness and the environment, are controllable, task conditions are repeatable, and policy change has the greatest impact on performance.

The ability to classify and measure the magnitude of forgetting based on performance is important to a learning system because, under the right conditions, it can identify when beneficial knowledge has been lost, locate wasted computation effort, and verify if a learning approach is best suited for a task. From the developer's point-of-view, these unfruitful periods of learning can be examined to understand why learning did not improve and possibly be removed in future learning events to save on learning computation effort. Finally, with a quantifiable way to determine the magnitude of forgetting, the amount of forgetting can be compared between two approaches to determine which approach suffered or benefited the most from knowledge lost or modification. With these reasons to classify and measure forgetting, there are motivations for a process that quantifies forgetting.

2.1 Reinforcement Learning Systems

Although there are many different types of *reinforcement learning* (*RL*) approaches, this paper is concerned with *direct policy search* [8] in *evolutionary algorithms* (*EAs*), *genetic algorithms* (*GAs*), and primarily, *decomposition-based reinforcement learning*. EAs and GAs are RL algorithms that are based on ideas of evolution, genetics, and survival of the fittest. Here, the learner undergoes an evolution process, typically through the genetic operators of mutation and crossover, to modify the learner's representation. Through the cycle of selection, reproduction, modification, and evaluation, the system reinforces positive performance changes as the learner learns a task. More detailed information on EAs and GAs can be found in Holland's [9] and DeJong's [10] work on evolutionary learning systems.

Several studies show that decomposition-based learning can outperform monolithic learning approaches that learn to perform complex tasks all at once, including Jackson and Gibbon's [11] and Hsu and Gustafson's [12], [13] work. Stone and Veloso's [1] layered learning is an example of a decomposition-based reinforcement learning approach, as well as hierarchical reinforcement learning approaches *feudal reinforcement learning* [14] and *hierarchical abstract machines* (*HAMS*) [15]. In these approaches, a hierarchy of decomposition is used to guide the learning system to different abstractions of the overall task.

These decomposition-based approaches are exposed to forgetting valuable knowledge as the learner switches its subtask focus. The subtask transition can cause knowledge used for the new subtask to overwrite knowledge used for an older, previously learned subtask. A metric is needed to identify if this modification of knowledge is positive, negative, or neutral to performing the overall, complex task. In addition, the metric should be able to determine the magnitude of what is forgotten to indicate the severity of what has been lost.

2.2 Related Performance Metrics

The simplest policy comparison measures are performance ratio and difference, which directly compare a policy's (*P*) performance with the policy after some knowledge has been forgotten (*P'*). *Performance ratio* (*PR*) is the quotient of two performances. *Performance difference* (*PD*), $f(x)$, is the difference between the post-forgetting policy's performance $g(P')$ and the pre-forgetting policy performance $g(P)$. If $f(x)$ is positive, positive forgetting has taken place; if $f(x)$ is negative, negative forgetting has occurred; if $f(x)$ is zero, neutral forgetting has happened because there was no change in performance, although the policy has been modified. PD is defined in Equation 1.

$$f(x) = g(P') - g(P) \tag{1}$$

Similar to PD, Markovitch's and Scott's [2] economics of learning measures the value of knowledge. In their approach, the payoff of learning is measured, where payoff can be positive, negative, or neutral and indicates the affect missing knowledge has on a policy. Payoff is calculated by taking the difference between two benefits, which are two separate policies, solving the same task. A benefit for one policy is the difference between the quality of the solution and the cost of solving the problem.

In Gorski and Laird's [16] work on transfer learning metrics, *transfer ratio*, *transfer regret*, *calibrated transfer ratio* (*CTR*), and *average relative reduction* (*ARR*) are examined for validity in comparing learning performances. These metrics, based on overall task performance, determine if an experimental policy, one that has learned a new behavior from an old behavior, outperforms a controlled policy.

The simplest of the four performance-change metrics in Gorski and Laird's work and the most similar to PR is transfer ratio. *Transfer ratio*, used by Morrison *et al.* [17] for performance comparison, is the ratio of the area under the experimental policy's performance from time 0 to time t over the area of the control's learning curve in the same time range. Unlike simple PR, transfer ratio considers the entire learning curve of the two compared policies.

Although the metrics above do not form a comprehensive list of all performance measures in transfer learning, they represent common methods of comparing policy performance change. An overlooked issue with such performance measurements is that they do not capture an entire policy change that occurs in decomposition-based approaches; instead, they capture the difference of only a single performance criterion: the overall task the policy tries to solve.

In decomposition-based approaches, a task is decomposed into subtasks the policy must learn to perform to solve the overall task. Although these single criterion measurements are adequate for monolithic learning, the learning of a task without decomposition, they lack the ability to analyze the changes in subtask performance caused by a policy modification in decomposition-based approaches. Tracking forgotten subtask-specific knowledge is significant because it can identify instances of when beneficial knowledge is lost and when learning at the subtask level stagnates or declines. The significance is heightened when subtask performance is a necessary component to the overall task but the evaluation of the task does not explicitly measure proficiency of subtasks. In this case, a single criterion-based forgetting metric only considers overall task performance changes and can overlook the loss of beneficial knowledge or the acquisition of negative knowledge used by important subtasks. Additionally, measuring performance changes only at the task level makes it difficult to determine the magnitude the lost knowledge has on task performance. Again, if subtask proficiency is an integral part of task performance, then measuring the impact of loss can be inaccurate if the task evaluation does not explicitly consider the importance of each subtask.

In this work, two metrics are introduced that classify which type of forgetting occurs at any point of the learning process and quantifies the magnitude of that diagnosed classification. Furthermore, the proposed forgetting measurements can be used by a range of existing monolithic- and decomposition-based approaches to identify performance changes due to forgetting. The main distinction of the metrics is their increased level of fidelity because of their consideration of subtask performance changes. Through experiments, we determine the effectiveness of each method and make recommendation under which conditions these proposed metrics are best suited for measuring forgetting.

3. Forgetting Metrics for Decomposition-based Learning

Both of the proposed forgetting metrics are based on the performance changes a policy experiences while acquiring new and forgetting old knowledge. The metrics compare performance of each subtask with its corresponding best performance to determine what effect the policy change has on each component of the task being learned. Although these metrics are not intended to be used to identify occurrences of forgetting, they are to be used when forgetting is the cause of policy and performance changes and are designed to classify and measure the type of forgetting that has occurred.

The difference between these proposed metrics and the others mentioned is that the proposed metrics explicitly factor in changes to all subtask performances instead of only the performance of the one, overall task. By considering subtask performance changes, each metric serves as an indicator of

when a policy loses or gains performance for individual subtasks. With that said, the proposed metrics are influenced by Markovitchs and Scott's measure of the value of knowledge and the metrics examined by Gorski and Laird.

Before describing the proposed metrics, a few definitions must be made that are common for each method: a *policy*, a collection of knowledge from the knowledge set, is modified at time-step *t*. At time-step *t*, the policy's performance of each subtask is retrieved from the function $p(s_i, t)$, where s_i is a subtask in the set of all subtasks *S* used to perform task *T*. In addition, function *p* returns the performance measure of task *T* at time-step *t* with the parameters of $p(T, t)$. Function *p* returns the real value ratio of performance to the optimum and is bounded to inclusively range from 0 to 1, where 1 represents optimal task or subtask performance and 0 represents the converse.

For each subtask s_i, there is a corresponding weight w_i. Also, there exists a weight for the task, w_t. Each weight is a real value inclusively ranging from 0 to 1, where the sum of all subtask and task weights is equal to 1. The weights determine the importance each subtask has on task performance and is defined by the developer.

3.1 Direct Forgetting Metric

The first forgetting measurement, *direct forgetting metric* (*DFM*), calculates a direct difference between two policies using the weighted sum of subtask performances and is defined in Equation 2. For simplicity, we will use policies at time-steps t and *t*-1 as the immediate policies that will be directly compared. By calculating subtask PD from these two time-steps, $f(t)$ makes a direct comparison between a policy and its immediate change in the next time-steps to determine the significance of the knowledge that was lost or modified.

$$f(t) = \sum_{i=1}^{|S|} w_i \left(p\left(s_i, t\right) - p(s_i, t-1) \right) \tag{2}$$

3.2 Maximized Forgetting Metric

The second proposed forgetting measure, *maximized forgetting metric* (*MFM*), is represented as $g(t)$ and is the weighted sum of the difference of $p(s_i, t)$ and the best performance of s_i from time 0 to *t*-1. MFM is defined in Equation 3. This measure utilizes the *max* function that retrieves the best performance of policy P on subtask s_i up to the time *t*. By calculating this difference, the entire performance history of each subtask is factored into the forgetting measure.

$$g(t) = \sum_{i=1}^{|S|} w_i \left(p(s_i, t) \quad - \max_{t' \in \{0,\ldots,t-1\}} \{p(s_i, t')\} \right) \tag{3}$$

If $f(t)$ or $g(t)$ return a positive value, then positive forgetting has occurred in at time *t*. If the returned value is 0, then neutral forgetting has occurred. If $f(t)$ or $g(t)$ is less than 0, then negative forgetting has taken place. The magnitude of the occurred forgetting is represented by the returned value. For instance, if the returned value is negative, the smaller the value, the higher the negative forgetting magnitude is and denotes how much performance has suffered because of the lost or modified knowledge. Consequently, if the returned value is positive, the larger the number, the stronger performance has improved with the forgotten knowledge.

4. Experiment Setup

The proposed DFM and MFM metrics are compared to the PD metric (defined earlier) to evaluate the effectiveness of the proposed measures. Though our ultimate interest is in constructing improved methods to solve multi-agent decomposition-based reinforcement learning problems, such problems make for difficult initial study. Because of this, we construct simple Boolean-logic problems that have the properties we need to investigate our metrics. Our true problem involves a non-linear combination of two well-known Boolean problems: *LeadingOnes* (LO: the sum of the sequence of continuous ones in the prefix of the string) and *TrailingZeros* (TZ: the sum of the sequence of continuous zeros in the suffix of the string). For fixed-length binary strings $x \in \{0,1\}^n$, LOTZ(x) = LO(x)*TZ(x). The optimum of LO is the all one string; the optimum of TZ is the all zero string, and the optimum of the LOTZ function is a string in which the first 16 bits are 1 and the last 16 are 0. In these studies, we focus on bit strings of length 32. For comparison purposes, the result of any evaluation is always divided by the largest possible optimal value-for LOTZ, this is 256 (16 times 16).

In these experiments, each bit abstractly represents knowledge that is stored in a policy, which is represented as the bit string. Forgetting is simulated through the modification of any bit in the string and may affect performance of the task or subtask the policy is learning to solve. The bit string learns by toggling bits and is evaluated based on the task or subtask it is currently learning.

The Boolean-logic problem was chosen because it is a simple problem that allows for easier study of the effects of forgetting. Modifications to the policy is instantly recognizable and measurable because the policy is represented by a bit string. Secondly, LOTZ can be conveniently learned with a monolithic or decomposition-based approach. Examining the metrics on the two separate approaches directly allows a conclusion to be made if the new metrics satisfy the goal of measuring forgetting at the subtask and task levels. Lastly, it is acknowledged that these Boolean-logic problems reside in the optimization problem domain. These optimization problems have been chosen because the performance measures used to solve these problems are used in the exact same manner as how they would be employed in an RL technique. In both cases, performance information is available after evaluation with no additional effort or data collection is needed when translating the use of these metrics from optimization to RL. The only difference is that these optimization problems make it less complicated to examine the effectiveness of the proposed metrics than traditional RL problems. Though the problems are very basic, they make it convenient to induce

forgetting on the policy and provide a clear, straight-forward way of validating the metrics.

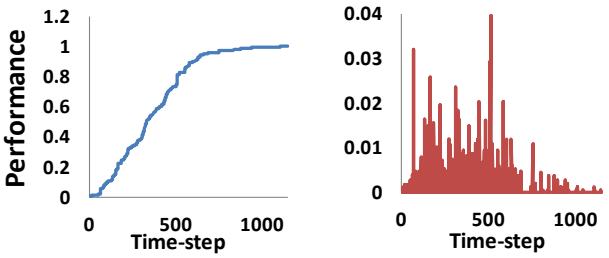
The PD metric is used as the control because it is widely used to compare performance differences of two policies, it is simple to compute and only requires performance data in its calculation, and its outputted value is easy to decipher (positive, negative, and zero values correlate to increased, decreased, and no change in performance). Although the other metrics also measure performance change, they do not easily translate to the forgetting classifications of positive, negative, and neutral and forgetting magnitude as the PD metric does.

Each metric will be evaluated on its ability to recognize the different types of forgetting as they evaluate two different learning methods employed to solve LOTZ. Method 1 learns LOTZ monolithically and method 2 learns the task with a three-subtask decomposition. Both methods use a 1+1 EA to optimize the bit string in solving the task. The *1+1 EA* is a evolutionary algorithm technique that has a single learner representation (the parent) produce a single modified version of itself (the child) through operators during the evolution process. The child replaces the parent if it outperforms or performs at least as well as the parent, depending on the system design. Borisovsky and Eremeev [18] and Wegener and Witt [19] provide performance studies on 1+1 EAs. In the context of this work, the bit string is the parent and a modified copy of the parent is the child. The mutation operator modifies the child by flipping each of the bits with independent probability $1/n$. The child replaces the parent if its performance is at least as well as the parent's. Bit flipping simulates forgetting when previously solved positions are lost or modified.

Method 1 learns the LOTZ task monolithically by repeating the child reproduction process until the optimal bit string is generated. The method is designed to evaluate the metrics on detecting non-negative forgetting. Method 1 guarantees only neutral and positive forgetting will occur because of the use of a 1+1 EA and its policy of only keeping a policy change if a mutated string produces a no-worse solution than the existing policy. The strict practice of only accepting equal or better performing policies also assures non-negative forgetting because only one performance criterion is used for policy evaluation and no decomposition is used.

Method 2 will learn LOTZ through a sequential decomposition-based approach, similar to Stone and Veloso's *layered learning* [1]. In method 2, the bit string will first learn the subtask of *leading ones* (*LO*) across the entire bit string, then *learn trailing zeros* (*TZ*) across the entire string, and finally learn the overall task of LOTZ. For the LO subtask, performance is calculated by counting the length of the all ones prefix and dividing it by the length of the string, 32. Similarly, the TZ subtask will calculate performance by counting the length of the all zero suffix and dividing it by the length of the string, 32. Both quotients indicate how close the bit string is at solving the subtask.

All forms of forgetting occur in method 2. Method 2's policy will experience positive forgetting as it learns LO from the initial randomly generated string. Method 2 will then see the policy suffer from forced negative forgetting as the subtask transitions from LO to TZ because the TZ subtask is purposely designed to pressure the EA to unlearn everything from the LO subtask to maximize TZ performance. Consequently, as the performance of TZ increases, the performance of LO will decrease, causing negative forgetting. In a similar manner, the bit string can again negatively forget knowledge learned from the TZ subtask when it starts to learn LOTZ, but the negative forgetting will be at a lesser magnitude than the transition from LO to TZ because not all of the zero bits acquired for TZ will be converted into ones. Finally, whenever the policy changes but performance remains the same, neutral forgetting occurs.



(a) LOTZ (b) PD/DFM/MFM

Fig. 1
METHOD 1: AVERAGED LOTZ AND FORGETTING VALUES OVER TIME.

The two experiment methods differ in weight assignment. Because method 1 is monolithic without any subtasks for LOTZ, the only weight value (w_t) will be set to 1. Method 2 has 3 subtasks and applies a .75 weight to the aggregate subtask of LOTZ because it is clearly the most important component to learning the overall task. The LO and TZ subtasks evenly split the remaining .25 weight, where each have .125 weight for forgetting equations DFM and MFM.

5. Results

Experiment method 1 demonstrates that PD and the two proposed forgetting metrics are equivalently well suited for classifying and measuring forgetting for the monolithic learning method. Because there is no decomposition in the monolithic method, the three forgetting metrics return the same values for each measured time-step, displayed in Figure 1 (b). Also, because the performance of the sole LOTZ task is never decreasing, the max function always returns the t -1's performance value (max returns t's performance value when $t = 0$), making all three metrics always return the same value.

The monolithic approach's use of a never decreasing performance value means only neutral and positive forgetting can be tested with the experiment setup. This never decreasing trend is displayed in Figure 1 (a), which plots the average LOTZ performance of the 10 trials over time. To test for neutral and positive forgetting, 10 independent trials of method 1

are performed and their average performance at each time-step is collected. For neutral forgetting verification, time-steps with unchanged performances for each trial and their average are compared with the metrics' returned values. With an accuracy of 100% for each trial, the metrics correctly return zero when performance does not improve. For positive forgetting, time-steps with an increase in performance are expected to result in positive forgetting values. Again, with 100% accuracy for each trial, the forgetting metrics correctly return forgetting values that correlate to positive forgetting when performance increased. For instance, at time-step 1,112, performance stagnates at .9996 for 35 time-steps. On the 36th time-step after reaching .9996, the bit string is modified to the optimal string for a performance value of 1. From this minuscule performance increase, the metrics return a small but correct .00039 positive forgetting value. From these results, it is concluded that given this experiment configuration, the proposed metrics are accurate in classifying and measuring performance-based forgetting in monolithic learning.

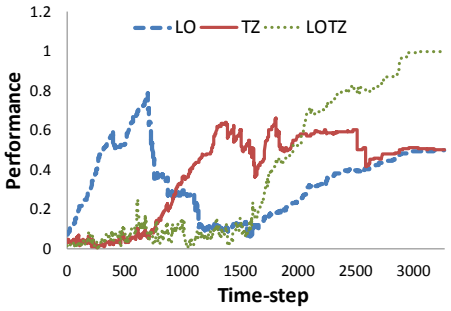
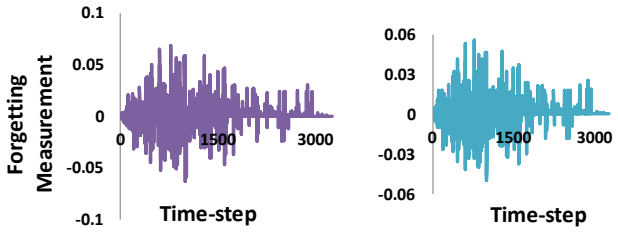


Fig. 2

METHOD 2: AVERAGED SUBTASK PERFORMANCE OVER TIME.

To validate the proposed metrics in a decomposition-based approach, method 2 collects and averages each subtask's performance per elapsed time-step for 10 independent trials. Figure 2 displays each subtask's average performance over all observed time-steps of the 10 trials. Further, subtask performance changes are noted and compared to their corresponding forgetting metrics' values. PD's averages are graphed in Figure 3 (a), DFM in Figure 3 (b), and MFM in Figure 4.



(a) PD (b) DFM

Fig. 3

METHOD 2: AVERAGED FORGETTING VALUES OVER TIME.

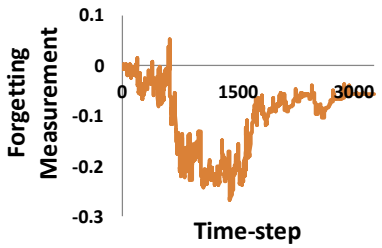


Fig. 4

METHOD 2: AVERAGED MFM VALUES OVER TIME.

By evaluating the 10 independent trials, it is expectedly observed that PD, DFM, and MFM can generate different classifications and magnitudes at the same time-step. Because PD only calculates the difference between two task performances, DFM's calculation is based on the weighted sum of subtask performances between two performances, and MFM uses the difference between each subtask performance at t and their respective maximum observed subtask performance through t-1, the forgetting calculations will differ at times.

For neutral forgetting validation, it is observed that for each of the 10 runs, PD and DFM remained unchanged when no change took place in performance. This neutral forgetting always results in PD and DFM returning 0 when subtask performance was not altered. Because MFM uses the entire history of each of the subtasks, the metric returns 0, the indicator of neutral forgetting, when the weighted aggregated value of all of the subtask is even with the best observed performance. In terms of the average of the 10 trials, MFM only detects neutral forgetting for a few instances early in the learning of LO and right before performance evaluation is switched to TZ. This neutral instance is caused by the bit string stagnating with the current best observed string while waiting to flip the correct bits for higher performance.

The ability to measure positive and negative forgetting is also validated through method 2. From the trials, PD always awards a positive value when task performance increases between two immediate time-steps and generates a negative value when there is performance decrease between t and t-1. Similarly, DFM always generates a positive value, which denotes positive forgetting, when the weighted differences in subtask performance is positive between t and t-1; negative forgetting takes place when the differences are below 0. DFM's trend is followed by MFM with the difference of comparing the best observed subtask performances with the performances at time-step t. Because the aggregate subtask of LOTZ is heavily weighted in these experiments, changes in LOTZ dominate the other subtasks in DFM and MFM calculations.

Finally, there are times where classification from the three metrics all differ. Again, these differences are caused by changes at the subtask level between PD and the proposed metrics and whether or not subtask performance at t-1 in-

cludes a best observed performance. For example, in the average of the 10 decomposition-based trials, at time-step 173, all three metrics returned different measurements. PD indicated neutral forgetting occurred with a value of 0 because LOTZ performance did not change between t and t-1. DFM indicated positive forgetting occurred with a value of 0.0016 due to a positive increase in LO performance while the other 2 subtasks maintained their values. MFM indicated negative forgetting because none of the subtask performances at t met or exceeded the observed best subtask performances. Because PD does not use the performance of each subtask, it misses subtask changes which can be responsible for performance degradation.

From methods 1's monolithic approach and 2's decomposition-based approach, we can conclude that all three metrics correctly classify forgetting at their respective levels; their differences lie in their subtask fidelity. PD is a simple metric that only considers direct changes in task performance. Although the easiest to calculate and requires the least amount of input, PD ignores changes at the subtask level. Even though it is more complex than PD and requires pre-defined subtask weights, DFM allows for direct comparison between two learning time-steps and factors in subtask changes. Finally, MFM compares each subtask performance at one time to the best observed.

6. Conclusion

Two proposed performance metrics were examined for accuracy and validity of forgetting classification and magnitude measuring. The first metric, direct forgetting metric, uses subtask performance and compares performance difference between two policies to determine the type of forgetting that has occurred. With the use of subtask performance weights and comparing performance changes between two policies, this metric is best used when a direct policy comparison is desired that compares two decomposition-based policies. The second metric, maximize forgetting metric, considers the entire history of subtask performance to determine which type of forgetting a policy has experienced. This metric is optimized to measure forgetting based on the best observed subtask performances.

As a control, both proposed metrics were compared to a performance difference measure. For monolithic learning, all three metrics are equivalent and prove accurate at classifying the type of forgetting that has occurred. Under decomposition-based learning, the performance difference measure does not factor all of the knowledge that is lost through the learning process. Instead, it ignores performance changes at the subtask level and solely relies on task performance for its calculation. This oversimplification can result in inaccuracies if subtasks are heavily weighted. On the contrary, the proposed metrics are shown to be robust enough to inform both a monolithic and decomposition-based approaches, capture subtask performance changes, and support direct and history-based calculations.

Although the metrics were not designed to detect forgetting, they are intended to be used when it is known that forgetting has occurred and is the cause of performance change in decomposition-based reinforcement learning. The metrics become valuable assets when the learning system struggles to retain important information used for solving older subtasks when new knowledge is obtained. With these metrics, developers can pinpoint times of the learning process where important knowledge is forgotten and determine its impact on performance at the task and subtask level.

References

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

[2] S. Markovitch and P. D. Scott, "The role of forgetting in learning," in *In Proceedings of the Fifth International Conference on Machine Learning*. Morgan Kaufmann, 1988, pp. 459–465.

[3] B. Smyth and M. T. Keane, "Remembering to forget: A competence-preserving case deletion policy for case-based reasoning systems." Morgan Kaufmann, 1995, pp. 377–382.

[4] I. Koychev, "Gradual forgetting for adaptation to concept drift," in *In Proceedings of ECAI 2000 Workshop Current Issues in Spatio-Temporal Reasoning*, 2000, pp. 101–106.

[5] H. Nakayama and K. Yoshii, "Effectiveness of active forgetting in machine learning applied to financial problems," pp. 24–29, March 2002.

[6] G. Widmer and M. Kubat, "Learning in the presence of concept drift and hidden contexts," *Mach. Learn.*, vol. 23, no. 1, pp. 69–101, Apr. 1996. [Online]. Available: http://dx.doi.org/10.1023/A:1018046501280

[7] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: a survey," *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, 1996.

[8] V. Heidrich-Meisner and C. Igel, "Evolution strategies for direct policy search," in *Proceedings of the 10th international conference on Parallel Problem Solving from Nature: PPSN X*. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 428–437.

[9] J. H. Holland, *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, 1975.

[10] K. A. D. Jong, *Evolutionary computation - a unified approach*. MIT Press, 2006.

[11] D. Jackson and A. Gibbons, "Layered learning in boolean gp problems," pp. 148–159, 2007.

[12] 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.

[13] ——, "Genetic programming and multi-agent layered learning by reinforcements," in *In Genetic and Evolutionary Computation Conference*. Morgan Kaufmann, 2002, pp. 764–771.

[14] P. Dayan and G. E. Hinton, "Feudal reinforcement learning," in *Advances in Neural Information Processing Systems 5*. Morgan Kaufmann, 1993, pp. 271–278.

[15] R. Parr and S. Russell, "Reinforcement learning with hierarchies of machines," in *Advances in Neural Information Processing Systems 10*. MIT Press, 1998, pp. 1043–1049.

[16] N. A. Gorski and J. E. Laird, "Evaluating evaluations: A comparative study of metrics for comparing learning performances," Computer Science Department, Rutgers University, Center for Cognitive Architecture, University of Michigan, 2260 Hayward Ave, Ann Arbor, Michigan 48109-2121, Tech. Rep., 2009.

[17] C. T. Morrison, Y. han Chang, P. R. Cohen, and J. Moody, "Experimental state splitting for transfer learning," in *Proceedings of the ICML-06 Workshop on Structural Knowledge Transfer for Machine Learning*, 2006.

[18] P. A. Borisovsky and A. V. Eremeev, "A study on performance of the (1+1)-evolutionary algorithm," in *FOUNDATIONS OF GENETIC ALGORITHMS, 7*. Morgan Kaufmann, 2003, pp. 271–287.

[19] I. Wegener and C. Witt, "On the behavior of the (1+1) evolutionary algorithm on quadratic pseudo-boolean functions," 2000.