The Thirty-First International Florida
Artificial Intelligence Research Society Conference (FLAIRS-31)

On-Line Agent Detection of Goal Changes

Nathan Ball, Jason Bindewald, Gilbert Peterson
Department of Electrical and Computer Engineering
Air Force Institute of Technology *
WPAFB, OH 45433

Abstract

An increasingly important job for the autonomous agents is determining what goal they should be accomplishing. In dynamic environments the goal of the autonomous agents does not always remain constant. This research examines how to detect and adapt to goal changes within a dynamic game environment. An adaptive learner capable of detecting concept drift is used to detect when a goal change has occurred within the game environment and exploration techniques are used to adapt to the change. Initial results show that the agent has an 84% detection rate.

Introduction

As autonomous agents continue to expand into new and unique environments an important capability for them will be intelligent goal detection. Whether implemented in a completely autonomous system or as part of a human-machine team, future autonomous agents will need to be able to operate in highly dynamic environments where goals can constantly change. To function efficiently, future autonomous agents will need to detect and adapt to changing goals without having to be explicitly told to do so by a human. For example, if an autonomous transport truck encounters an accident on the road it would be beneficial for it to briefly change its goal to ensure that the accident is reported, and medical authorities are en route. Additionally, the capability to recognize goal changes can benefit human machine teams. By keeping their goals in alignment with those of the human, agents can avoid hindering operational performance (Endsley 2015).

This paper presents an agent that can adapt to a variety of goals within a game environment. We present an adaptive learning agent that utilizes concept drift techniques to determine when the games goal has changed. Reinforcement learning is then utilized to learn the new goal. We aim to determine how well the agent can detect changes and whether certain categories of goals are more difficult to adapt to than others.

The rest of the paper is organized as follows. First, we discuss the related work: goal classification, concept drift, and the *Space Navigator* environment. Next, we review the methodology, explaining the agent and experimental design. Then we analyze the early results obtained from initial trials of the experiment. The paper closes with a conclusion of the research.

Background & Related Work

An important first step before developing a system that can adapt to changing goals, is defining exactly what goals are. When it comes to games, goals are not always easily defined. Often, they are generalized as, “acquire the most points”, “survive for a length of time”, or “kill the enemy units.” Within each of these “meta-goals” are a subset of gameplay goals that must be accomplished. Djaouti *et al* (2008) proposed a unique taxonomy for classifying games through their mechanics. The list of mechanics is divided into either gameplay mechanics or goal mechanics. In this division, gameplay mechanics define actions that rely on player input, while goal mechanics provide feedback to the player about their performance. The four goal mechanics (avoid, match, destroy, and create) have been adopted as the target goals between which our system must adapt.

Goal determination and adaption is not an inherently new concept to games. Many games have implemented an adaptive artificial intelligence that can reason over goals to improve realism and difficulty. The AI enemies in, *No One Lives Forever 2: A Spy in H.A.R.M.s Way* (Orkin 2003) and *F.E.A.R.* (Orkin 2006), dynamically choose between different goals and create an action plan according to the current goal. In both games, however, the agents are hard coded with all the possible goal states and need only determine which goal they should accomplish. They do not formulate new goals beyond the ones that have been hard coded. In this research the agent has no knowledge of how the goals can change and instead must observe the environment to determine the current goal state.

The problem of identifying goal changes is closely related to concept drift. In any supervised learning problem, the objective is to predict a target variable y given some set of input features X using known (X, y) training pairs (Hoens, Polikar, and Chawla 2012). Here, y can be viewed as the feedback from goals within the environment and X are the

---

[*] The views expressed in this paper are those of the authors, and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the U.S. Government.

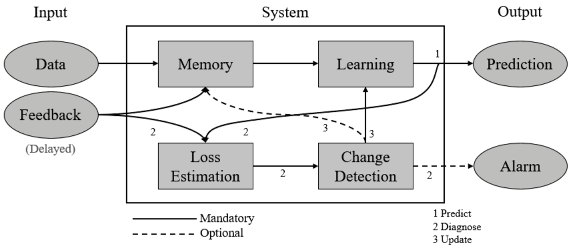

Figure 1: Generic framework for an adaptive learning system. Recreated from (Gama et al. 2014).

features of the game that are used to determine what goal should be accomplished. Concept drift occurs when the goal changes and the previously learned (X, y) pairs become obsolete, requiring new relations must be learned.

A system that can adapt to concept drift must be able to overcome two key problems (Hoens, Polikar, and Chawla 2012, Gama et al. 2014); it must detect legitimate concept drift while filtering out noise in the data and adapt to the new data. Gama *et al* (2014) present a generic framework, seen in Figure 1, to create a capable adaptive learner. The framework posits that a drift learning system only needs four key components: memory, learning, loss estimation, and change detection modules. These control how much data is stored, how to update the predictive model, and what change in the data triggers a drift alarm. For each of these modules, design decisions are dependent on the problem domain and the data being analyzed.

The contribution of this research is the application of concept drift adaption to a game environment with multiple dynamic goals that start unknown to the autonomous agent. The agent begins with no knowledge of the goals, only an understanding of the actions it can take. Through change detection and reinforcement learning the agent can formulate new goals.

Experiment Environment

The environment utilized for this research is the *Space Navigator* (Bindewald, Miller, and Peterson 2014) route creation game. Shown in Figure 2, *Space Navigator* is a tablet-based air traffic control style game. There are three primary ship colors (red, blue, and yellow) that spawn on the edges of the screen at regular intervals. The player draws a trajectory that the ship will follow by dragging the ships colored marker around the screen. The ship follows its given trajectory until it lands on its home planet, is destroyed, or reaches the end of the trajectory. If two primary ships of different colors collide, they will combine to form a new ship that is the fusion of the two colors. Red and yellow combine to form orange, blue and yellow form green, finally, red and blue form purple. Initially when a ship collides with another ship of the same color or a fusion ship, its “shield” will absorb the collision. A second collision will destroy the ship, removing it from the game. For this research, autonomous play is implemented using a pair of agents that, generate trajectories for the ships and avoid collisions with all objects other than the intended target.

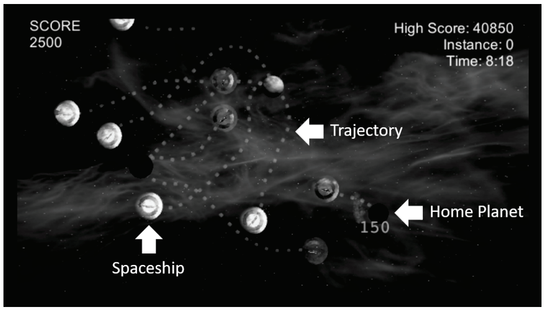

Figure 2: An example game of *Space Navigator* being played by the autonomous agent.

Table 1: Breakdown of points by interaction for each goal scenario.

Points by Goal	Planet Collision	Shield Depletion	Ship Destrustion	Ship Fusion (per ship)
Avoid	150	-50	-100	-50
Match	50	100	-150	-75
Destroy	-150	50	100	-150
Create	100	-50	-100	100

Methodology

The goals, (avoid, match, destroy, and create) are implemented in *Space Navigator* through changing the number of points earned for the interactions between ships and the other objects in the environment. Interactions are defined as a collision between a ship and another ship or planet. A description of the four goal scenarios are presented as follows:

- Avoid: Ships must avoid collisions with other ships. Points are earned from landing on planets and lost for all forms of collisions.
- Match: Ships of the same color must collide to deplete their shield. Points are earned when a ship depletes its shields and points are lost from ship destruction or fusion.
- Destroy: Ships must destroy themselves through multiple collisions. A small number of points are earned for shield depletion then more points are earned from destruction of the ship.
- Create: The three base colored planets have been replaced by the fusion ship color planets. Fusion ships must be formed to earn points for landing on planets.

Table 1 shows the point breakdown for each interaction across the four goals. The points were distributed across the goals such that, at most a single ship can earn 150 points and at worst lose 150 points.The goal in this research is carried by the environment not the specific agents.

The goals in the environment are tracked through an adaptive learner agent. The adaptive learner primarily follows the framework presented by Gama *et al* (Gama et al. 2014) modified with an additional reinforcement learning layer of exploration and exploitation. Algorithm 1 gives an overview of how the agent functions. Single example memory is used to populate the predictive model. Only the most recent example of an interaction is stored by the agent. When a new example of an interaction is received, it overwrites the previous example. The predictive model, which predicts the number of points a ship will earn for an interaction, is formed via a lookup table between the interactions and their associated point reward in memory.

When a ship completes its assigned interaction, the agent receives a noisy value for the number of points earned. The noise added to the score is drawn from a unit normal distribution then multiplied by 25 to increase its effect. The noise avoids giving perfect information to the agent that would allow it to instantly detect when a change occurs. This represents how a player may not notice the change due to their attention being taken up from drawing trajectories and avoiding collisions. The feedback is compared against the predicted number of points for the interaction to get the prediction error. The agent maintains a short (five sample) and long (10 sample) window of the error to compare the loss over time. The average loss in the short window will be affected more quickly after a change than the long window. When the average error in the short window is 25 points greater than the average loss in the long window a change alarm is triggered. The alarm signals to the agent that it should begin exploring the environment to learn the full extent of the changes.

Exploration occurs using a semirandom recency based approach. While exploring the environment the agent maintains memory of the interactions that it has explored and only explores new interactions. The exploration is partially random due to the random color of the ships that spawn, so there is no way to control the order in which interactions will occur. The agent explores for 45 seconds before switching to exploitation. At the end of an exploration phase the agent compiles the exploration knowledge of the reward for each interaction. The data is used to formulate the current goal, which is then distributed across all ship colors. This way, even if a ship color did not explore an interaction it can estimate the points based on what the other ship colors have experienced. The agent always begins in an exploration phase and only returns to exploration after a change is detected.

While not exploring the agent enters an exploitation phase. During exploitation the agent utilizes the predictive model generated by the learner to assign ship targets. When a ship spawns the agent compiles a list of objects that will yield the most points based on the ships state. When an object on the list, that also has the other ship on their targets list, spawns or becomes available the two objects assign each other as goals. During exploration if a ship is on screen for 20 seconds before a target is assigned then it will query the goal agent for a random target. This ensures that some exploration occurs after a change even if it is not explicitly detected.

Algorithm 1 Adaptive Learner Agent algorithm for detecting change and assigning targets.

```
Start:
$exploreTime = currentTime + 45$
$Exploring = true$
Update:
On Ship Spawn: $assignTarget$(Ship)
After Delay: $checkDrift(feedback, prediction)$
if $Exploring$ && $currentTime > exploreTime$ then
    $Exploring = false$
    Compile exploration knowledge
    Formulate goal
    Distribute goal across all ships
function ASSIGNTARGET($ship$)
    if $Exploring = true$ then
        Explore new interaction
    else
        Compile list of targets
        if Target exists then
            Assign target
        else if Target not assigned in 20 sec then
            Assign random target
function CHECKDRIFT($feedback, prediction$)
    Update memory
    Calculate loss
    Update loss windows
    if $avg(\text{5 sample Loss}) > avg(\text{10 sample Loss}) + 25$ then
        Clear loss windows.
        $exploreTime = currentTime + 45$.
        $Exploring = true$
```

Experiment

An initial experiment tests the adaptive agents performance and examines which goal is the hardest for the agent to learn. For each condition, the environment starts in one of the four goal states, then the goal switches to one of the other three during the game. A goal change window occurs at either a quarter of the way through the game or half way through the game. For both times, randomness is added such that the change can occur within a 30 second window on either side. Pairing each set of goal changes with the two switch windows results in 24 different conditions.

Several assumptions are made during this experiment. First, all ships share the same goal, at no point during the experiment do two ships have different goals. The goal distribution that occurs at the end of each exploration phase is dependent on this assumption. Second, the true points value is known everywhere except in the loss calculation. The last assumption is that the adaptive agent starts with no information about the environment. At the start of each new trial the agent is refreshed and does not carry over any information from the previous trial.

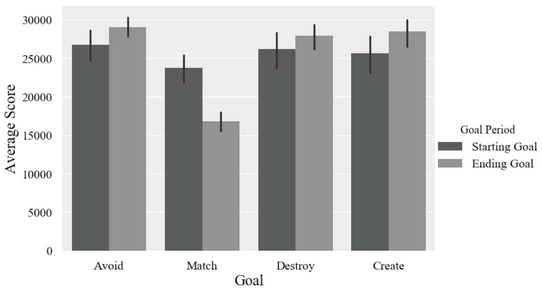

Figure 3: Average scores binned by goal and goal period. Plot displays the average score and 95% confidence interval.

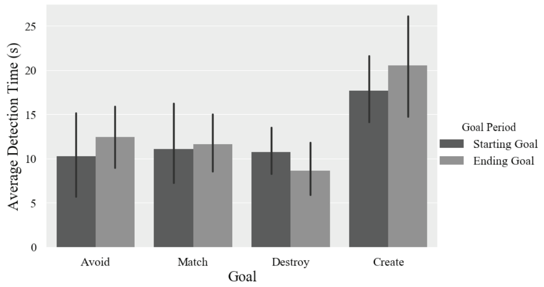

Figure 4: Average time taken to detect the goal change binned by the goal that was switched away from, or to. Plot displays the average detection time and 95% confidence interval.

Results

Over seven full batteries of trials, 168 total games, the average score across all games was 25,570. Figure 3 shows a box plot of the score distribution separated by goal and goal period, being either the starting goal or ending goal. While the average score for games starting with the match goal is lower than the other three, there is not enough of a difference to be considered statistically significant. However, when sorted by the second goal a large difference is seen. Games with match as the second goal had a significantly lower score than games with the other three goals. There is a 12,000-point difference in the average score between match and the other three goals. The extreme difference between the two goal periods is a result of second goal being active longer than the first goal. In half the games the first goal is only active for a quarter of the total game, thus when match is the second goal it is active longer leading to lower scores. The poor match goal performance was due to the goal agent triggering a high number of false positives. A false positive here meaning that the goal agent detects a change when none has occurred. Of the 183 false positives that were detected by the learner 173 of them occurred during the match goal. This resulted in the agent exploring far more than it should have instead of exploiting the learned knowledge.

The goal change was properly detected in 141 of the 168 games in an average of 13.3 seconds. Of the 27 games where a change was not properly detected; in 12 the change was detected over a minute after the switch occurred, in 10 games a false positive change was detected closely before the switch such that the exploration period overlapped with the change, and in five games no change was ever detected. For games with proper detection, Figure 4 shows the time taken to detect the goal change, separated by goal and goal period, being either the starting goal, changed away from, or ending goal, changed to. On average, it took significantly longer to both detect change to, and away from the create goal. The create goal takes longer to detect because the ships often rely on the time triggered exploration to detect the change. Reducing the 20 second timer could improve the change detection time but might also trigger more often even when the goal has not changed.

These results show that the adaptive learner agent is capable of quickly detecting goal changes in the environment. The final scores show that the agent can adapt to the new goal and achieve a high score. Additionally, the results indicate that the agent can detect and adapt to the change independent of the starting or ending goal. The exceptions to this statement are the match goal score performance and create goal detection time. This could apply to other games to create agents that can react to unexpected player events. In games players find methods to subvert the intended goal in ways the AI is unable to adapt to. This system could be used to detect when a player has changed their goal, then adapt to the new goal the player is attempting to accomplish.

Conclusion

The problem of detecting goals within an environment is increasingly important as autonomous agents spread into new dynamic environments. This paper presented a framework for adapting to dynamic goals within the *Space Navigator* environment. The results show that the agent can successfully detect and adapt to drift in the goal. Future work will aim to improve agent performance in the match and create goal scenarios. Additionally, we are interested in comparing these results to human play to compare which is better at detecting changes.

References

Bindewald, J. M.; Miller, M. E.; and Peterson, G. L. 2014. A function-to-task process model for adaptive automation system design. *International Journal of Human-Computer Studies* 72(12):822–834.

Endsley, M. 2015. Autonomous horizons: System autonomy in the air force–a path to the future. *US Department of the Air Force, Washington*.

Gama, J.; Žliobaitė, I.; Bifet, A.; Pechenizkiy, M.; and Bouchachia, A. 2014. A survey on concept drift adaptation. *ACM Computing Surveys (CSUR)* 46(4):44.

Hoens, T. R.; Polikar, R.; and Chawla, N. V. 2012. Learning from streaming data with concept drift and imbalance: an overview. *Progress in Artificial Intelligence* 1(1):89–101.

Orkin, J. 2003. Applying goal-oriented action planning to games. *AI Game Programming Wisdom* 2:217–228.

Orkin, J. 2006. Three states and a plan: the ai of fear. In *Game Developers Conference*, volume 2006, 4.