

# Explainable Deep Adaptative Programs

Magesh Kumar Murali

School of Electrical and  
Computer Science

Oregon State University  
Corvallis, Oregon 97331

Email: muralim@oregonstate.edu

Martin Erwig

School of Electrical and  
Computer Science

Oregon State University  
Corvallis, Oregon 97331

Email: erwig@oregonstate.edu

Alan Fern

School of Electrical and  
Computer Science

Oregon State University  
Corvallis, Oregon 97331

Email: Alan.Fern@oregonstate.edu

**Abstract**—TODO

**Index Terms**—Adaptive Programs, Programming Languages, Reinforcement Learning

## I. INTRODUCTION

Programming is often difficult and tedious process, it takes years of training and experience for a person to become good at creating software that can solve complex real world problems. One of the reason for this is that as the problem domain becomes more complex a lot of analysis and critical thinking is needed to capture all the scenarios that the program will encounter. The programmer has to not only think about the different scenarios that can occur but also has to make decisions based on the different scenarios. Complex problems will also contain multiple decision points and decision made at one location might have an adverse effect on the rest of the decision making process. This makes writing deterministic programs that captures all such scenarios and decisions difficult for the programmer. This is one of the reason why it is difficult to program an AI agent for games like Chess or Go that could beat human player.

There are also special type of problems where you would not know beforehand what kind of scenario that the program can encounter or the environment itself gradually changes overtime. Consider a simple problem of sorting a list of integers, the type of sorting algorithm to choose will depend on the size of the input array. Insertion sort will perform better when compared to Quicksort for smaller input size. If the initial assumption about the input size changes then the programmer has to completely rewrite the program to use a better sorting algorithm or he has to have the foresight that the input size will vary drastically.

This is one of the issues that is often faced in software development process, the assumptions being made about the environment during the initial development of the software will change over time while the overall objective that the program solves remains the same and this leads to extending the project to make it adapt to the new assumptions about the environment and making it more expensive to develop software. It has even led to the development of new software practices like Agile Process that takes into account the change in environment as part of the development process.

The issues mentioned above has led to the development of new programming paradigm called Adaptation Based Programming (ABP). ABP consists of simple level of abstraction called Adaptive Variable (AV). Adaptive Variables are same as regular variables in that the value it holds can vary at runtime, however in ABP it has three additional constraints:

- The values that it can hold, also called choices, are restricted by the programmer
- The adaptive variable is tightly dependent on the state provided by the programmer
- The objective of the adaptive variable is described using reward signals

Whenever a programmer encounters a point in the program that requires complex decision to be made, he/she creates an adaptive variable with a list of possible choices that the adaptive variable can hold. The choices are complemented by a set of reward signals, the reward signals serves as feedback to the adaptive variable to indicate whether the value it holds will achieve the objective that the programmer wants it to achieve. By defining the objective of the program using adaptive variables and reward signal, the programmer does not need make the complex decision instead the adaptive variable uses powerful reinforcement learning algorithms to make the decision.

The programmer also provides a state object as input to the adaptive variable, the choice of the value that the adaptive variable holds will also depend on this state object. The state object will make sure that the adaptive variable adapts to the current state of the program. Whenever the adaptive variable encounters a new state that it hasnt encountered before it adapts itself to make sure it reaches the objective even for the new state. This way the adaptive variable not only makes complex decisions, it also adapts to changes in the environment or even new environments that it hasnt encountered before.

One of the issues with using state of the art reinforcement learning algorithm to make the choices is that the reason for making the decision is often not clear to the user. This makes it hard to understand the output of the program and even debugging becomes tedious. We try to overcome this issue by providing explanations to the output generated using annotation at the decision and reward points.

This paper provides three improvements to adaptation based programming paradigm.

- Deep Adaption Based Programming: Extending the adaptive variable to use state of the art deep reinforcement learning.
- Explanations: using annotations
- Using General Value Functions (GVFs) to improve the generated explanations

## II. EXPLANATION IN ABP

### A. Motivating Example

To illustrate explanation in XDAP let us consider the following simple example problem. Figure 1 shows a simple  $5 \times 2$  gridworld. The agent in this example is a traveller whose goal is to reach the destination (Home) within 8 days and collect as much reward as possible. In order to reach home the traveller has to cross different types of terrains. It would take 4 days to cross a mountain, 2 days to cross a small hill and 2 days to cross a river. The gridworld contains two types of treasures: Diamond and Gold. Diamond being slightly more valuable than Gold. The traveller has four actions (North, South, East, West) to choose from at every step.

1. Should explain high negative reward for not reaching the house?
2. Type of reinforcement learning problem?

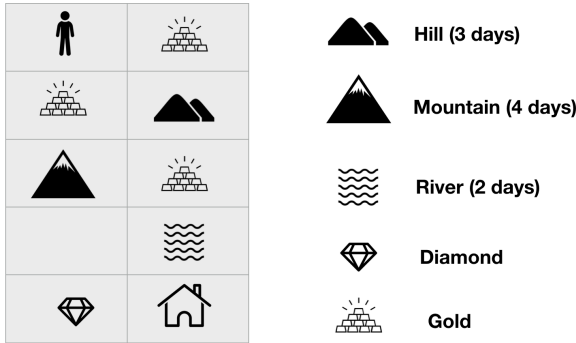


Figure 1. Traveller Problem

### B. Adaptive Program

Should the reward be from the environment? or add if statements showing how the reward is assigned?

```
traveller = Adaptive(choices = [NORTH, SOUTH, EAST, WEST])
state = env.reset()

for days in range(8):
    direction_to_move = traveller.predict(state)

    state, reward, done, info = env.step(direction_to_move)
    traveller.reward(reward)

    if done:
        break
```

Is it necessary to explain the program? Or is it self explanatory?

When the ABP program is run, the traveller decides to take the path as shown in Figure 2

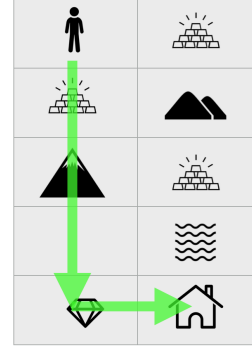


Figure 2. Traveller Decision

However it is not clear why the decision was made to take the current path. There could be several reasons for making the decision and many unanswered questions that we can ask: "Why did the traveller not move EAST and collect the gold on the first step?"

This can be considered as one of the disadvantages of using adaptive variables in the program. Even though we assume the value that the adaptive variable holds is optimal, it is not clear as to why it made the decision to hold the current value. Also making it difficult for the programmer to debug programs containing these adaptive variables.

We explain the decisions being made by the adaptive variable by extending the adaptive programming library to use reward decomposition and using predictors

Need a better way to say this! Should it be called predictors?

### C. Explaining using Q value

1. So far we havent talked about how the adaptive variable choose its value. Is it needed? Or should it be a reference to another paper?
2. Need a better title

The adaptive variable uses reinforcement learning techniques to choose the correct value [1]. After the initial training period, the adaptive variable generates a Q function [1]. The value that the adaptive variable holds is then based on the Q function. The Q function also called value function has an explicit semantics [2]. In the case of the traveller example it represents the knowledge of which action to take to maximize the reward or to be more explicit which action should the traveller take to reach the destination in 8 days and collect as much treasure as possible.

Need to rewrite this in a better way! Important concept for the paper



Show an example of traveller step and Q value histogram

### III. CONCLUSION

TODO

### REFERENCES

- [1] T. Bauer, M. Erwig, A. Fern, and J. Pinto, "Adaptation-based programming in java," in *Proceedings of the 20th ACM SIGPLAN workshop on Partial evaluation and program manipulation*. ACM, 2011, pp. 81–90.
- [2] R. S. Sutton, J. Modayil, M. Delp, T. Degris, P. M. Pilarski, A. White, and D. Precup, "Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction," in *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*. International Foundation for Autonomous Agents and Multiagent Systems, 2011, pp. 761–768.