MARMARA
UNIVERSITY

CSE4097 - Engineering Project I
Analysis and Design Document

USING HIERARCHIES IN REINFORCEMENT LEARNING FRAMEWORK WITH NON-STATIONARY ENVIRONMENTS

Group Members

Ezdin ASLANCI 150112022

Ahmet Kutalmış COŞKUN 150113018

Project Advisor

Assoc. Professor M. Borahan TÜMER

1. Introduction

Reinforcement Learning (RL) is a behavioral learning approach to solve sequential decision making problems, in which the environment is generally assumed to be stationary. This assumption on stationarity is often considered to be optimistic for the reason that it holds for a limited amount of real world problems. In dynamic (non-stationary) environments, using RL is usually nontrivial since the agent is forced to re-learn the policy from scratch – and forgets the old policy – after changes. Making the agent able to detect and respond to the changes so as to learn and adapt to the current behavior of environment improves the learning performance [4]. In this work, we plan to improve the change detection mechanism of Hierarchical Reinforcement Learning with Context Detection (HRL-CD) by tracking the convergence tendency of First Order Markov (FOM) dependencies of action sequences.

1.1 Problem Description and Motivation

Reinforcement Learning (RL) is a learning method inspired from behaviorist psychology. An RL agent interacts with the environment that surrounds it, by taking actions which are defined in its current state . The environment responds to the action of agent by returning a reward signal and passing to a new state. Throughout this process, the aim of the agent is to reach a goal state by learning the optimal sequence of actions which maximizes the cumulative reward it obtains from the environment.

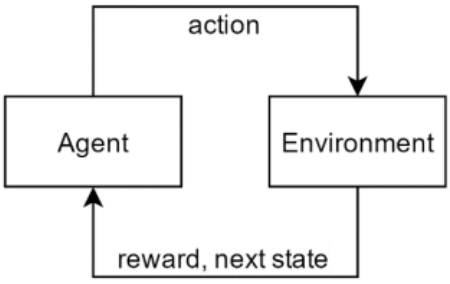

Figure 1: Interaction between agent and environment

A model-free, off-policy TD control algorithm, Q-Learning, estimates the Q values by taking actions and using related rewards for each step. In each step, Q value of a single state-action pair is updated by an update equation. In an improved model-based algorithm known as Prioritized Sweeping (PS), multiple eligible state-action pairs are updated with the same equation [3]. Both of these approaches become infeasible when the state space is large or continuous. In these kinds of problems, using hierarchies has advantages and makes RL feasible. Hierarchical Reinforcement Learning (HRL) approach divides the problem into solvable independent tasks instead of trying to solve the whole problem as a single task. An agent with HRL approach can combine the solutions of individual tasks to reach its goal [2,3].

1.2 Scope of the Project

Our research is focused on deterministic, discrete and dynamic environments with different dynamics that occur infrequently and independently from the agent. We plan to adopt grid world problem and run our simulations of Reinforcement Learning on such environments. In a grid world problem, an RL agent learns how to reach to a goal state from its start state by taking right, left, up and down actions on the grid. Dynamism will be provided by the changing the grid randomly at the training process of the agent. We expect the agent to realize and develop some strategies such as relearning or using its previous experiences, when some changes appear on the grid. The changes on the environment (grid) will be detected by observing the sequence of actions which is executed by the agent.

Our another study which is on the improvement process, is "Detection of Regime Switching Points in Non-Stationary Sequences using Stochastic Learning based Weak Estimation Method (SCD)". Our main objective is to combine these two studies which means; changes on environment will be detected by our SCD method. We expect to detect very slight changes by using SCD method.

Simulating learning process on continuous environments currently is not in the scope of our project. We plan to conduct experiments only on discrete environments that can be represented with a grid. For a future work, our method may be improved so that it works on both discrete and continuous problems.

1.3 Definitions, Acronyms, and Abbreviations

RL : Reinforcement Learning

RL-CD : Reinforcement Learning with Context-Detection

HRL : Hierarchical Reinforcement Learning

HRL-CD : Hierarchical Reinforcement Learning with Context-Detection

MDP : Markov Decision Process

SMDP : Semi-Markov Decision Process

SLWE : Stochastic Learning base Weak Estimation

SCD : SLWE Change Detection

BC : Betweenness Centrality

PQueue : Prioritized Queue

FOM : First Order Markov

SOM : Second Order Markov

HD : Hamming Distance

2. Related Work

We have performed a detailed literature survey for our research and in this section Analysis and Design Document, we explain each related paper with giving information about how the paper and our work is related.

Dealing with Non-Stationary Environments using Context Detection

In the work of da Silva, Basso, Bazzan and Engel [1], authors introduced "RL-CD" which is a method for solving reinforcement learning problems in non-stationary environments. Their method is based on creating, updating and selecting one among several partial models of the environment and it makes the learning system capable of partitioning knowledge into models. The non-stationary environments that are covered by the authors on this work are those whose behavior is given by one among several different stationary dynamics. Each of these dynamics is called a context. The authors assumed that each context can be detected by only tracking the transitions and rewards. The contribution they made is that they overcame an important restriction of previously proposed methods, which is requiring a fixed number of models. RL-CD dynamically creates a new model if the quality of the best model is still worse than some minimum quality. At any time, only the model with the highest quality is activated. They presented empirical results of RL-CD for three different validation scenarios. One of them is "Ball Catching" which we would like to create a similar experiment for testing our method.

Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning

In the work of R. S. Sutton, D. Precup, S. Singh [2], authors worked on an important challenge of Artificial Intelligence, which is representing knowledge at multiple levels of temporal abstraction. Their research is focused on addressing this challenge within the mathematical framework of Reinforcement Learning and Markov Decision Processes (MDPs). They extended the usual notion of action in Reinforcement Learning framework to include temporally extended actions. Options can be considered as policies for taking actions to reach a sub-goal over a period of time. A set of options defined on MDPs creates a semi-Markov Decision Process (SMDP). Extending their work to deal with non-stationary environments is the main motivation for us on this project.

Hierarchical Reinforcement Learning with Context Detection (HRL-CD)

In the work of Y. E. Yücesoy and M. B. Tümer [4], the authors proposed an autonomous agent which learns a dynamic environment by taking the advantage of Hierarchical Reinforcement Learning. They presented their empirical results the grid world problem with different dynamics. We noticed that the weak point of this method is that it fails to detect slight changes in the environment. We hope to contribute their work by proposing a more sensitive method using our Change Point Detection algorithm which uses Stochastic Learning based Weak Estimation (SLWE). Combining these two studies is the main goal of our project.

3. System Design

3.1 System Model

Environmental changes should be detected as the agent learns and the learning process must be intervened according to magnitude and structure of environmental changes. In this context, learning of the agent and the detection of changes in non-stationary environments can be expressed as a client-server connection. On the client side, while learning is being done by the agent, at certain time intervals, server (SCD) will respond to agent whether the environment has changed, and if so, how the new learning process should continue.

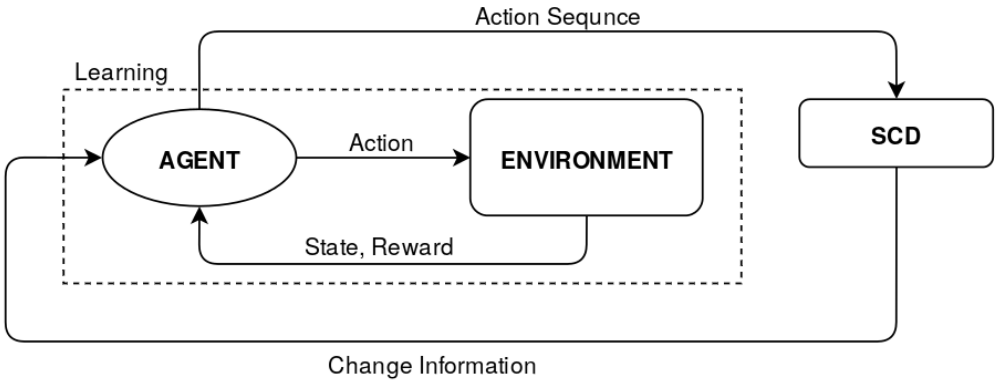

Figure 2: Communication between change detection and learning processes

As seen in the communication diagram above, the changes will be detected by observing the actions of the agent because if the environment changes, the reward values which are responses of the environment to actions of the agent will also change. After SCD observes the action sequence, it will inform to agent about the current state of the environment. This information may be:

- No change in the environment, continue learning.
- Change has been detected! An environment you have not encountered before, learn from scratch!
- Change has been detected! Model of this environment exists in your memory, continue with this model.
- Change has been detected! Only a certain part of the environment has changed, just relearn that part.

3.2 Flowchart and/or Pseudo Code of Proposed Algorithms

As mentioned above, our project has two main processes which are continuously in communication. In the first process, agent learns the environment with RL algorithms which are Q-Learning, PS and HRL and listens to response messages from second process. If any change is detected in environment either a new model will be created or a model from memory will be loaded. In both cases, an old model will be saved to the memory.

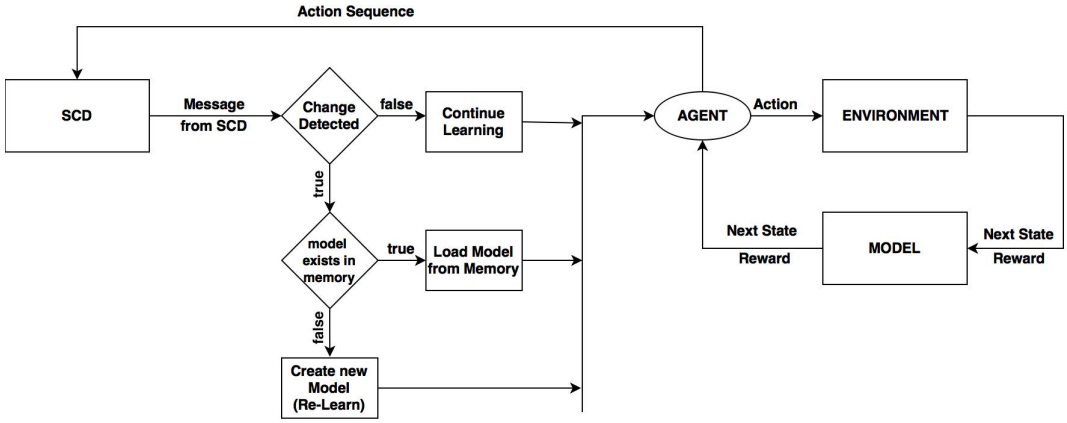

Figure 3: Flow diagram of the learning module.

In server side, detection process will wait until the learning process sends actions that the agent performed. When the server receives the actions, it starts tracking Markov Dependencies and sends back the information whether the environment is changed or not.

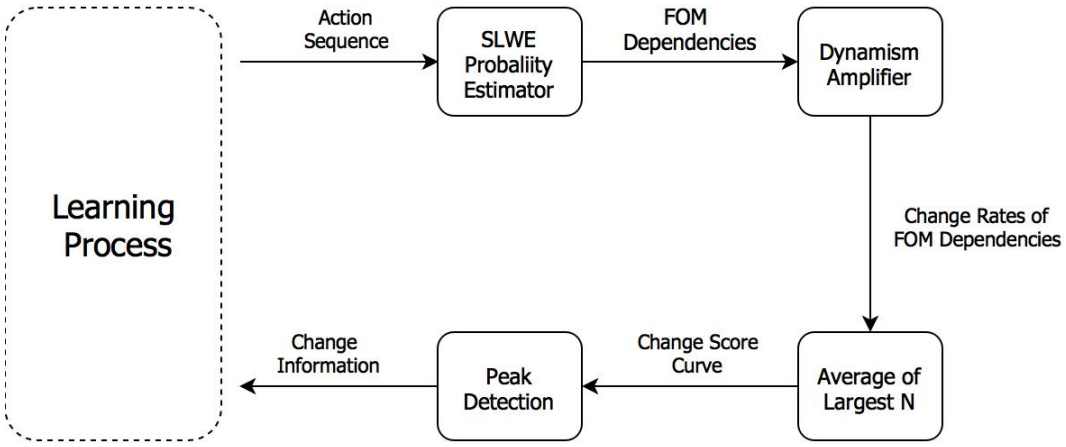

Figure 4: Flow diagram of change detection module.

Algorithm: Hierarchical Reinforcement Learning

Initialize $Q_0(s, a)$, $Model_0(s, a)$ and PQueue to empty, Current Context c = 0
Do forever
  s = current state, a = policy(s, Qc)
  Time-step count, k = 0
  (s', r, k) = execute(s, a)
  Update E values of every context considering <s, a, s', r>
  c = Context of E_{max}
  Update Tc and Rc for current context c
  $Model_c(s, a) = (s', a, r, k)$
  $p = r + \gamma^k max_a Q_c(s, a) - Q_c(s, a)$
  if $(p > \Theta)$
    Insert (s, a) into PQueue with priority p
  Repeat N times while PQueue is not empty
    (s, a) = pop from PQueue
    $(s', r) = Model_c(s, a)$
    $Q_c(s, a) = Q_c(s, a) + \varphi[r + \gamma^k max_{a'} Q_c(s', a') - Q_c(s, a)]$
    Repeat for all s'', a'' predicted to lead to s
      r'' = predicted reward from Model
      $p = r'' + \gamma^k max_{a'} Q_c(s, a) - Q_c(s'', a'')$
      if $p > \Theta$
        push (s'', a'') into PQueue with priority p
  If end of the episode
    Search sub-goals with Betweenness Centrality
    If Variance of Sub-goal counts < 1
      Create options
      Learn options

Algorithm 1: Hierarchical Reinforcement Learning algorithm [4].

3.3 Comparison Metrics

There are few metrics being used for measuring the learning performance of Reinforcement Learning algorithms. Specifically, on the grid world problem, the agent's objective is to reach the goal state with the minimum number of steps. This makes the number of steps a good concept for tracking convergence and divergence of learning process. We expect to get instant jumps on the step count curve after the current behavior of environment changes.

Moreover, we are planning to use the sequence of actions as an input to the change point detection algorithm we developed. The change points we get from the algorithm will be compared with the true change points we expect to get. We know these true change points because while conducting experiments, we will determine the points where the current behavior of the environment changes. We are planning to use Standard Accuracy Criteria (SAC) for measuring the success of behavior change detection.

	Change Present	Change Absent	Total
Change Detected	a	b	a+b
Change not Detected	c	d	c+d
Total	a+c	b+d	a+b+c+d

Table 1: Symbol Definitions for Change Point Detection Results

Measure Name	Definition	Ideal Value
Sensitivity	$\frac{a}{a+c}$	1
Specificity	$\frac{d}{b+d}$	1
False Negative Rate	$\frac{c}{a+c}$	0
False Positive Rate	$\frac{b}{b+d}$	0
Predictive Value Positive (PVP)	$\frac{a}{a+b}$	1
Predictive Value Negative (PVN)	$\frac{d}{c+d}$	1
False Alarm Rate	$\frac{b}{a+b}$	0
False Reassurance	$\frac{c}{c+d}$	0

Table 2: Definitions of Standard Accuracy Criteria

3.4 Data Sets or Benchmarks

The current version of project is an offline version of what we planned. So, we get data from RL training and we know time points where the environment is changed. We change the environment several times while agent learns and agent does not know that the environment has changed. When the agent completes training we get the action sequence of agent that the agent has performed during the traning time.

Our data, the action sequnce is one dimensional data which is the combination of four elements; left, right, down and up. According to grid size and RL parameters the size of our data changes. For example; when we increase grid size linearly, the size of our data increases exponentially. The other factors which affect data size are parameters and one of the most important parameters is epsilon because epsilon affects data size directly. Epsilon is the probability of non-greedy action selection. When we decrease epsilon, traing time gets longer.

As mentioned above, our data is one dimensional and combination of 4 elements. We know where we have change points (i.e. where the environment is changed). We analyse data with SCD process and we expect to find true change points.

4. System Architecture

We have two main modules which are learning module and change detection module. The general view of the communication between these modules can be represented with the figure below.

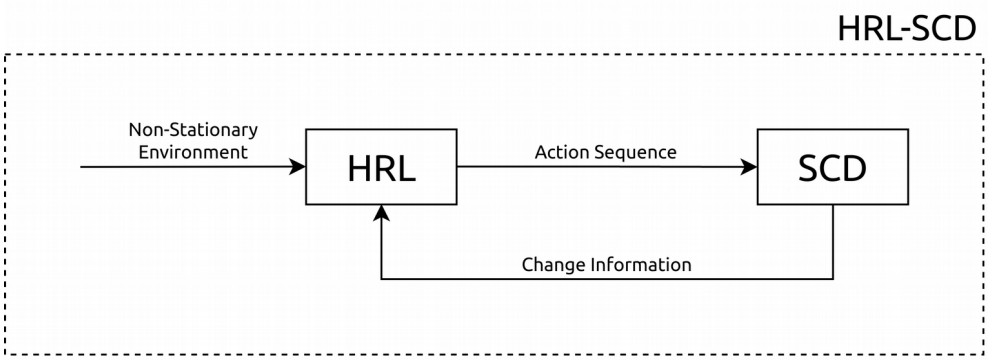

Figure 5: Communication between two main modules: HRL: Learning Module, SCD: Change Detection Module

The data/control flow of Learning Module can be expressed with the following figure.

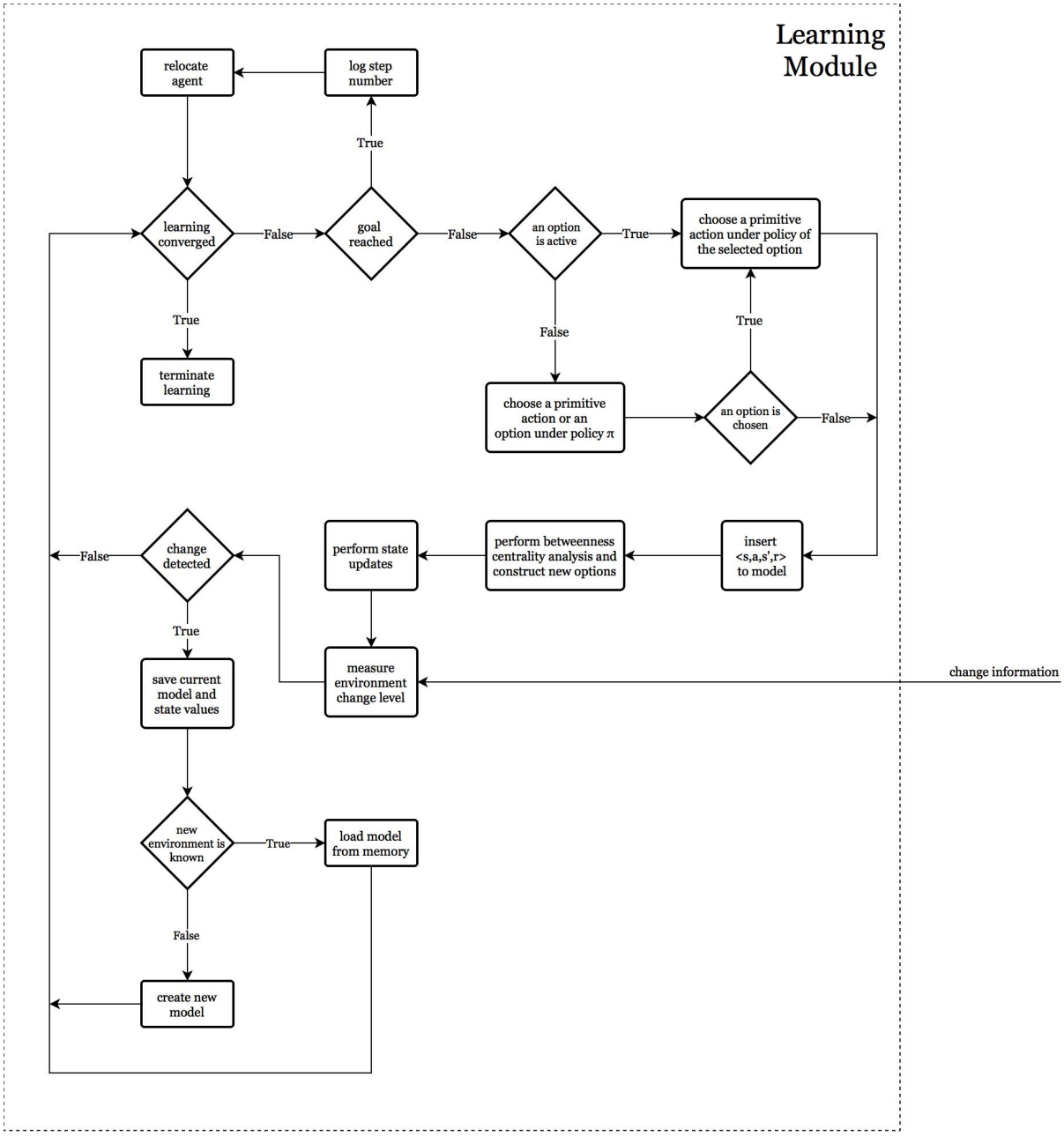

Figure 6: Detailed flow diagram of learning module.

5. Experimental Study

We plan to test our algorithm with the experiment sets we design in order to measure the SAC metrics we explained in the “Comparison Metrics” part of this document.

Since our research in on non-stationary environments, we plan to conduct experiments that include multiple environments with random switching points. The important point about length of regimes is, each regime should be given enough time to allow the convergence of Markov Dependencies.

Another important criteria about the experiment sets is, the difference of consecutive regimes. Since we detect changes by tracking the FOM dependencies, the peak we get on the change score curve gets higher if the changes on the FOM dependency curves are more drastic. This makes detecting a change between completely different regimes easier. This will also be a parameter we want to observe the effect on our controlled experiments. The difference between environments can be expressed with the Hamming Distance of the optimal action sequences of those environments.

Another important parameter we want to see the effect of is the probability of non-greedy action selection. This probability makes the agent explore the environment and discover new paths & states. In Reinforcement Learning framework, this probability is expressed with the parameter ε. We plan to run experiments with different ε values and different decay factors. We can consider the effect of ε as noise since it makes the agent select a non-greedy action.

Here is a preliminary experiment result we conducted by using the artificial data we generated by combining the optimal action sequences of three different regimes. This sequence is quite similar to what we expect to get from our online system we explained in the "System Design" part of this document.

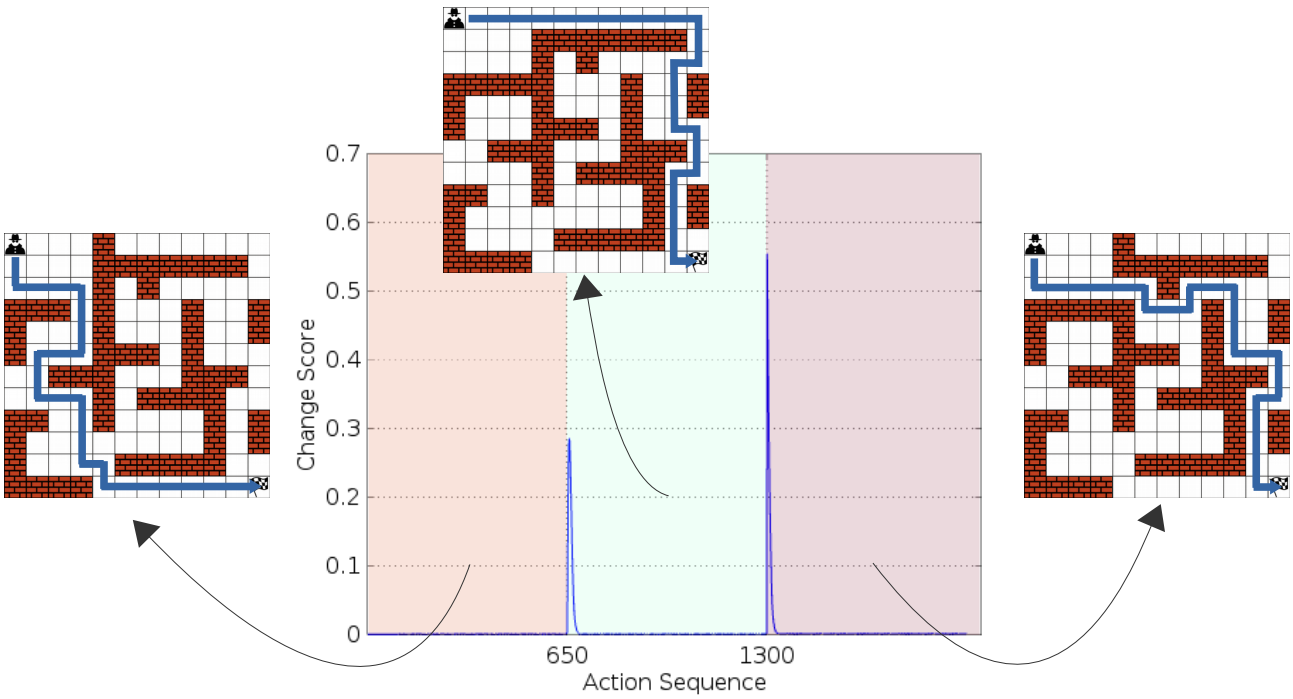

Figure 7: Peaks of change score curve indicate change points on the non-stationary action sequence.

6. Tasks Accomplished

6.1 Current State of the Project

To explain the current state of the project, we should take our project as two sub-projects; Reinforcement Learning and Change Detection.

In Reinforcement Learning side, we have implemented two basic learning algorithms; Prioritized Sweeping and Q-Learning. Our software that we designed on Python for the learning part of the project supports multi environments and also our software has betweenness centrality calculator tool and all the way we implemented those are compatible with Hirarchial Reinforcement Learning. We also have an alternative version of Q-Learning that is written in C, in case of a situation that we need to generate action sequences faster.

In Change Detection side, we have finished SCD implementation on MATLAB and it can run a set of experiments but the constraint we have is that SCD shoud be as fast as possible. Response time of SCD is not acceptable for an online system with MATLAB. So we re-implemented SCD on C++ in order to make it more responsive.

6.2 Task Log

During this semester, we had meeting with our advisor each week on Monday. In these meetings, we reported our weekly progress and we made brainstroms with our advisor about existing and possible problems we had on our research.

We compared and analyzed the results we obtained from our preliminary experiments. We have done risk analysis against possible problems, for example what to do if the data is too large or the SCD is weak against small changes.

Until now, we have observed promising experimental results with different parameters in different environments which are approved by our advisor.

6.3 Task Plan with Milestones

The most important task that should be completed immediately is finishing improvement and publishing SCD C++ implementation. We have some real world industrial data and we will publish our SCD algorithm after we get expected results from industial data before second semester.

HRL impementation is also a very important task for improving our project because Q-Learning and PS algorithms becomes infeasible on environments with large state spaces. Once we implemented HRL, we will be able to run experiments on more than 100x100 grids. The second task about HRL is HRL-CD implementation. We want to re-implement HRL-CD which is written by Yücesoy and Tumer and as a contribution, we will test HRL-CD on ball catching problem instead of grid world problem.

Currently, we can manually run our experiments with the C++ implementation of SCD. So our other task is to complete improving SCD and new version will be able to run a set of experiments automatically. Our other task about SCD was writing muti-thread support version but the response of SCD is fast enough on C++ for an online system, so we removed this task from our list.

The main goal of the project is making an online end to end system which detects changes while agent learns and SCD will send information about changes and agent will update learning strategies according to this information. After we built end to end sytem, we will test our results systematically.

7. References

[1] Silva, B.D. da, Basso, E.W., Bazzan, A.L.C. & Engel, P.M., Dealing with Non-Stationary Environments using Context Detection. *23rd International Conference on Machine Learning (ICML)*, 2006.

[2] Sutton, R.S., Precup, D. & Singh, S., Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, pages 181-211, 1999.

[3] Sutton, R.S. & Barto, A.G., Reinforcement learning. *Learning* **3**, 322, 2012.

[4] Yücesoy, Y.E. & Tümer, M.B., Hierarchical Reinforcement Learning with Context Detection (HRL-CD). *International Journal of Machine Learning and Computing*, 7763, 2015.

[5] Freeman, L., A set of measures of centrality based on betweenness. *Sociometry*, **40**: 35–41, 1977.

[6] Oommen, B.J. & Rueda, L. Stochastic learning-based weak estimation of multinomial random variables and its applications to pattern recognition in non-stationary environments. *Pattern Recognition* **39**, 328-341, 2006.