CSE 643 Artificial Intelligence
Paper Critique

Saksham Suri
IIIT-Delhi
2015082

I. Paper Information

1) Title: Real-Time Navigation in Classical Platform Games via Skill Reuse
2) Authors:
   - Michael Dann
   - Fabio Zambetta
   - John Thangarajah
3) Year Published: 2017
4) Conference Name: IJCAI
5) Link: https://www.ijcai.org/proceedings/2017/0219.pdf

II. Summary

This paper considers the problem of performing real-time navigation in domains where a gods eye view is provided. It tests the proposed algorithm in the Infinite Mario environment. The main theme of the paper is to teach the agent "skills",but also knowledge of what those skills can and cannot achieve. The agent, using these skills can derive complex navigation plans for new game configurations. Skills are modelled mathematically using the Option's framework. Formally, an option, is defined as a tuple $< I, \pi, \beta >$ where:

- I $\subseteq$ S is the set of states the option can be initiated from.
- π : S $\times$ A $\rightarrow$ [0,1] is a policy that returns the probability of selecting action a when in state s.
- β : S $\rightarrow$ [0,1] returns the probability that the option will terminate in a given state.

The proposed algorithm can be summarized as:

- Rather than attempting to identify problem-specific bottlenecks upfront, or trying to train distinct skills such as running and jumping, the agent is taught a general skill for performing local navigation.
- After this skill has been acquired, the agent is trained to estimate the likelihood that a given local movement can be executed successfully. Theoretically, the agent ought to learn high probabilities for straightforward movements such as running a few squares to the left unobstructed, and low probabilities for impossible movements such as running through walls.
- Lastly the game is modelled as a graph, with edge weights between nearby grid squares equal to the log-likelihood of the agent being able to perform the corresponding movement. Given a target destination on the current screen, path with the greatest likelihood of success is calculated through Dijkstras algorithm.

For the training algorithm, Q-Learning was used with a neural network as the functional approximator for action-values.

A. Local Training

For local training, firstly a neighbourhood is defined with the current position of the agent as the origin. The neighborhood is the set of tiles within Manhattan distance D of the current position of the agent. D is set to the maximum height the agent can jump. Now a goal tile is chosen randomly inside this local neighbourhood. To train the agent two different reward mechanisms were compared. One was binary reward where the agent received +1 for reaching the training goal and zero elsewhere and the other an incremental reward for reducing the distance to the training goal. For any two tiles t_a and t_b such that t_b lies in the neighbourhood of t_a, the trained policy induces a natural skill for navigating from t_a to t_b. Following the Options Framework, this skill is modelled as an option $o_a^b =< I_a, \pi_b, \beta_b >$ where:

- $I_a = \psi^{-1}(t_a)$ where ψ is mapping from state to tile.
- π_b is a policy for navigating to t_b, derived by setting the local movement policys goal equal to t_b.
- β_b is 1 if tile t_b is reached else 0.

B. Likelihood of Success

After the local movement policy has been trained, random training goals are set. A time limit, T_{max} is set and it is observed whether the local movement policy succeeds within this limit. This generates a series of samples from which the following function can be approximated:
$Pr(o_a^b$,$T_{max}|s)$=$Pr(o_a^b$ terminates within T_{max} time when started from state $I_a)$

C. Using Learnt Skills

To create a long sequence of moves for navigation, hypothetical initiation states are used to estimate the loglikelihood of a sequence of steps succeeding, by summing the log-likelihoods of the individual steps:

$$\begin{aligned} \textit{Log-Likelihood}_{seq} = -\big(&\log[\Pr(o_a^b, T_{max}|s)] \\ &+ \log[\Pr(o_b^c, T_{max}|\hat{s}_{t_b})] \\ &+ \log[\Pr(o_c^d, T_{max}|\hat{s}_{t_c})] + \cdots\big) \end{aligned}$$

Given an arbitrary goal tile on the current screen, the plan with the greatest likelihood of success by treating log-likelihoods as path lengths and applying Dijkstras algorithm is chosen as the path the agent takes. Figure 1 shows an example path created by the agent to reach the goal using its local knowledge.

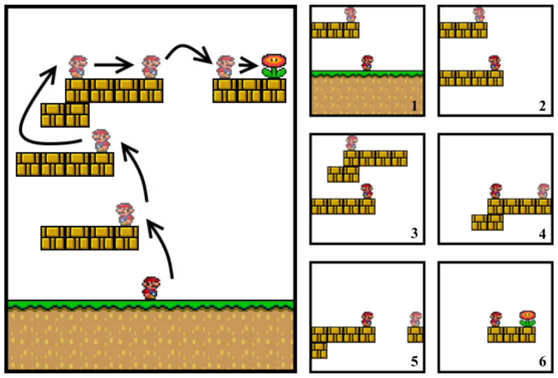

Fig. 1. Skill based lookahead path creation(Image taken from the original paper)

D. Replanning

As the policy learnt is probabilistic, the agent may sometimes fail during the execution thus leading to the requirement of replanning as the goal at that moment may become unreachable from the current state (in case of Mario this can be due to a fail from teh ledge). Replanning is done only if the current step leads to a failure. This happens if either the target is not in the agents movement range or if the agent gets stuck in local loop while moving making it revisit the same states(tiles) again and again.
On doing experimental analysis it was observed that the proposed algorithm outperform the A^* algorithm in terms of number of times the agent successfully reached the desired goal. Between binary and incremental reward schemes, the binary reward scheme gave better performance and on observing the path returned by both the schemes, it was found the binary scheme gave a comparatively shorter and simpler path while the incremental scheme gave a comparatively complex and longer path.

III. Rejection Basis

- In terms of comparison with other algorithms the authors have just compared the results with those of the A^* algorithm, I feel they should have atleast compared it with more recent and advanced Reinforcement Learning algorithms as they have shown great promise in the field of game playing from game screens.
- They have cited the work of Mnih et al. [Mnih et al., 2015] ie, the DQN algorithm and said that it would not perform well in the current scnario based on its performance on a similar game called Montezumas Revenge, but on checking out the game I found that the game not only involves navigation but also has enemies and adversaries which was not the case in the environment the paper under review is considering hence the assumption that DQN will fail might not be correct.
- The neighbourhood of the agent under consideration was defined as the maximum height the agent could jump but this does not account for possibilities such as running and jumping etc. due to which the agent can reach distances greater than the normal jump height.
- It uses Manhattan Distance to define the neighbourhood but by using Manhattan distance a square frame will not be obtained as claimed in the paper. Each local frame will be somewhat circular.
- There is no concrete experimental results supporting the choosing of the value of T_{max}, where T_{max} was the limit within which it was checked whether the local movement policy succeeds or not.
- A major flaw I feel in their approach is the removal of adversary or enemies from their environment. Without the presence of an adversary the problem of navigation gets highly simplified as otherwise the algorithm would have to account for adversaries that too of different kinds and keep track of their movement in order to evade them. I feel this is one of the major drawbacks as almost all games have adversaries and navigating while escaping them is a complex yet practical problem which needs to be addressed.
- The neural net being used as a functional approximator for Q-Learning has a lot of different kind of inputs being concatenated and being fed to the same network. These include the velocity, current position, goal position and overall layout of the current screen (binary encoding of position of bricks). I think this kind of representation is going to cause problems for the learning of the neural network as they represent completely different quantities with different units, scale and range of values. Also there is no mention as to how the velocity being fed as an input to the network is being estimated.

IV. Improvements

- A major improvement at the algorithmic level would be experimenting with Policy Gradient based methods instead of Q-Learning. Work done by Mnih et al.[Asynchronous Methods for Deep Reinforcement Learning] clearly highlights the superiority of Policy Gradient methods in game playing and these surpass DQN in almost all games tested with a substantial margin. One of the most popular algorithms in policy gradient is the Asynchronous Actor-Critic Agent.
- Using Convolutional neural networks as functional approximators will help in better learning as they work directly on pixels and can extract localized features too. This will also reduce the original input representation complexity which earlier required using different kinds of representation for different things such as velocity, goal etc. and concatenating them, while the CNN can automatically learn these kind of features from the input frames.
- To address the real problem of navigation the environment should be changed to include adversary/enemies as

well as this would make the problem being addressed complete.

- Neighbourhood should be defined as the maximum distance reachable in any direction rather than maximum height.
- Value of T_{max} should be found experimentally by trying out different values rather than using a fixed value.
- By using Convolutional Neural Networks and Policy Gradient the local training can be removed and a global approach can be used in turn, in which end to end training is done rather than locally finding the best path. We can teach the agent to navigate in the complete grid at once rather than by parts.