CS748: Advances in Intelligent and Learning agents report

Sai Kalyan, Rahul Chunduru, Suraj Narra

June 2020

1 Introduction

HFO (Half field Offence) is a platform where one can build, train and test offence and defensive abilities of soccer agents in a half-court of soccer game [1]. It is widely considered as a useful benchmark for determining an agent's ability in a full-fledged soccer game. In this project, we experimented on improving the defence of soccer agents in this task, especially using the notions of Action Repetition.

2 Nomenclature

2.1 In built Agents

HFO already involved two inbuilt powerful agents Agent2D, Helios.

1 **Agent2D** - This agent is Robocup 2012 winner, which is released for public use later by Hidehisa Akiyama [2]

2 **Helios** - This agent is based on Helios' 2013 team Eindhoven(Robocup 2013 was held there.) release. Helios is a much powerful agent compared to Agent2D.

In this project, we worked to build defensive agents against Agent2D. The below table shows defense rates when the inbuilt agents are played against each other. Note, how powerful Helios agents are.

Type	2v2 Defense rate	3v3 Defense rate
Agent2D	0.29	0.62
Helios	0.73	0.8

2.2 Feature Set

We used following features for defense. These are high-level features built and modeled by HFO authors. Descriptions from HFO_Manual [3]

0 **X position** - The agent's x-position on the field.
1 **Y position** - The agent's y-position on the field.
2 **Ball X** - The ball's x-position on the field.
3 **Ball Y** - The ball's y-position on the field.
4 **Proximity to Opponent** - If an opponent is present, proximity to the closest opponent. Invalid if there are no opponents.

T **Proximity from Teammate i to Opponent** - For each teammate i: the proximity from the teammate to the closest opponent. This feature is invalid if there are no opponents or if teammates are present but not detected.
$2T$ **X, Y** - For each teammate: the x-position, y-position
$2O$ **X, Y** - For each opponent: the x-position, y-position

2.3 Actions

We used the following actions for defence. These are Actions built and modelled by HFO authors. Descriptions from HFO_Manual [3].

- **Intercept**(): Moves to intercept the ball, taking into account the ball velocity. More efficient than chasing the ball.
- **Move**(): Re-positions the agent according to the the strategy given by Agent2D.
- **Reduce_Angle_To_Goal**(): Moves the agent to a point on the field, such that the kicker has the least open angle to the goal.
- **Defend_Goal**(): Moves the agent to a point on the goal line on the field, such that the kicker has the least angle to the goalposts.
- **Go_To_Ball**(): Makes the agent go towards the ball.
- **Mark_Player**(uniform_number): Moves the agent to mark the player with the specified uniform number.

3 Experiments

In this project, we experimented with two variants of 3v3 gameplay, one controlling a single defence agent and another controlling two defence agents. For each, we employ two learning algorithms, namely, CMAES with action repetition(AR) and SARSA-λ with action repetition(AR). The offence team uses **Agent2D** policy.

3.1 Learning Algorithms

Here, we present the learning algorithm's pseudo-codes.

Algorithm 1: CMAES with AR

```
Input: population size, decision interval d, Episodes E
solver = EvolutionStrategy();
while True do
    solutions = solver.ask();
    fitness list = np.zeros(solver.population size);
    for iterator i 1 ... population size do
        fitness list[i] = RewardsimulateHFO(solutions[i],E,d);
    end
    solver.tell(fitness list);
    best solution,best fitness = solver.result();
end
```

Algorithm 2: Sarsa-λ with AR

Input: λ, decision interval d
$Q(s, a) \leftarrow 0$;
for *episodes 1 ... E* **do**
  step $\leftarrow 0$;
  s $\leftarrow 0$;
  action $\leftarrow \epsilon$-greedy(Q) ;
  while *episode not over* **do**
    $r, s' \leftarrow$ simulateHFO(action);
    step $\leftarrow$ step + 1 ;
    if *step % d == 0* **then**
      action $\leftarrow \epsilon$-greedy(Q);
      SARSA(λ) update of $Q(s, a)$;
      step $\leftarrow 0$;
    else
    end
    $s \leftarrow s'$
  end
  SARSA(λ) update of $Q(s, a)$;
end

3.2 Implementation details

For **SARSA**, we set $\lambda = 0.9$ and learning rate of policy, $\alpha = 0.0001$ for smooth learning.

For **CMA-ES**, we used a single hidden neural net of 32 neurons between *features* layer and *action* layer and optimised the parameters. We trained with a population size of 8, each trail consisting of 250 episodes with decision interval of 32. After every 25 trails we validated the best solution obtained till then with 2000 episodes.

3.3 3v3 with Single Agent (3v3_sa)

This is the case when there is one defence agent follows our policy and other two defence agents (which includes the goalie) use in-built Agent2D base policy. Upon experimenting CMAES with various decision intervals(d), The results were not varying and all the results were converging to 64-65%. So we continued further experiments only with d=32.

3.3.1 Training plots

The following are the plots for 3v3_sa HFO task. Di-sarsa plots predicts the running average of past 2000 episodes which is a good approximate to validation on 2000 episodes instead of validating at frequent intervals.

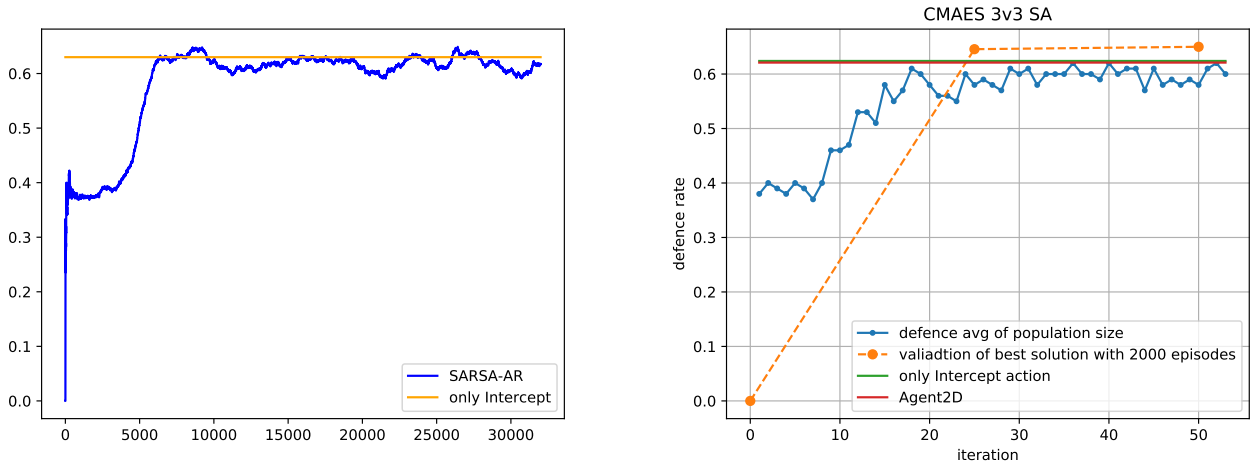

Figure 1: Learning Curve of 3v3_sa SARSA-λ, CMAES with d = 32 vs trails

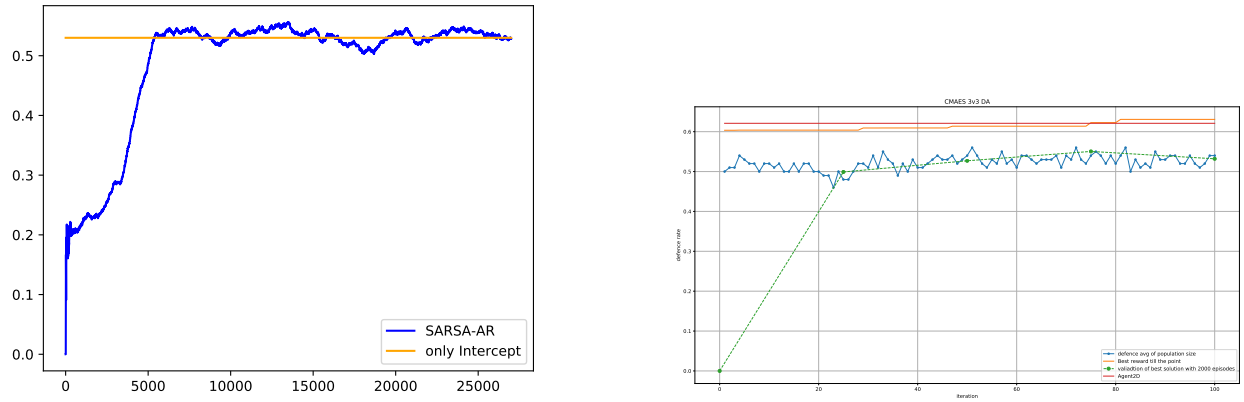

Figure 2: Learning Curve of 3v3_da SARSA-λ, CMAES with d = 32 vs trails

3.4 3v3 with Double Agent (3v3_da)

They are trained using pre-trained agents from 3v3_sa using CMAES.As pre-trained agents are used there is very little improvement in defence rate. Figure (2) depicts the result.

3.5 Observations

Upon simulating the test episodes and checking the trace of the trained solution, we observed that most of the time, **Intercept** action is taken by the trained agent. Also, **Intercept** is found to be more responsive than **GoToBall** action in predicting and attacking ball. When Agent takes only Intercept action in 3v3-sa, Agent outperforms Agent2D defence.

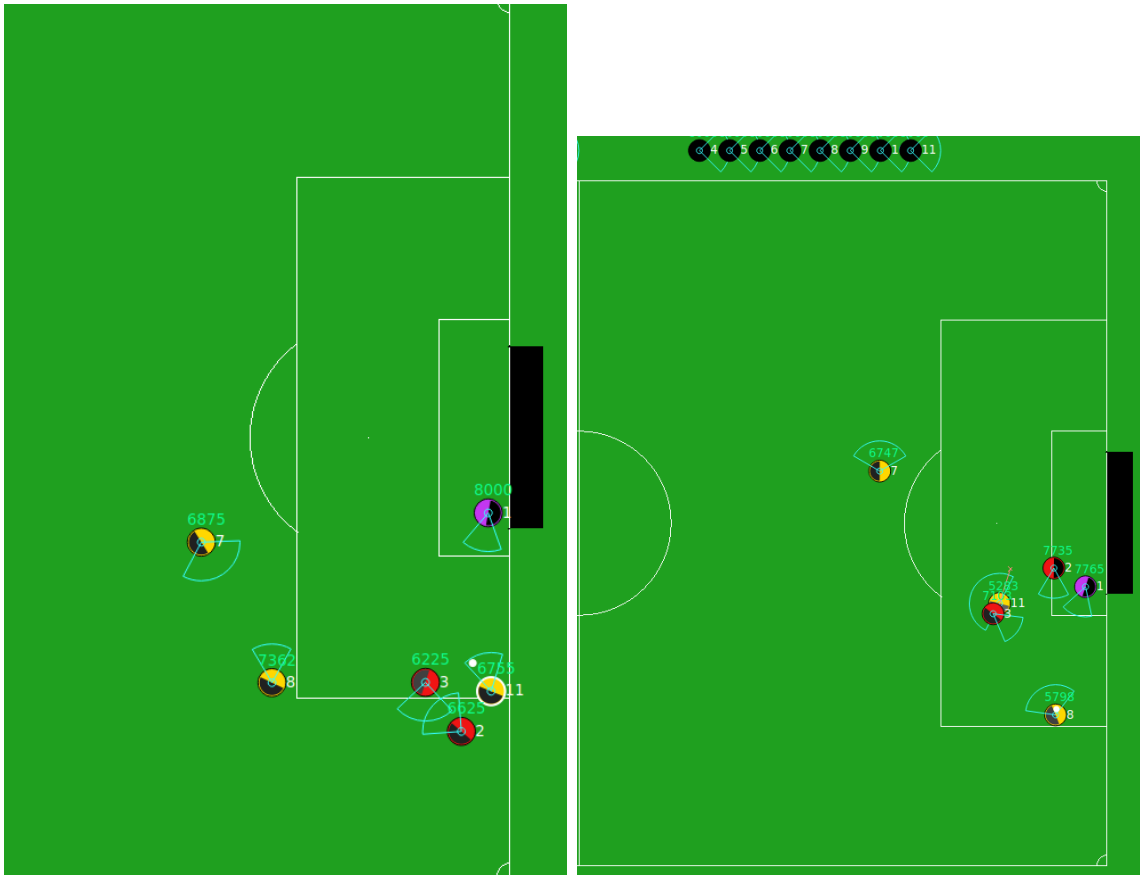

Figure 3: 1.Two agents chasing the ball in our Double agent, 2. Helios working where one agent is near goalie

4 Analysing the policies

4.1 Shortcomings in the solution

- The agents march towards the ball to intercept most of the time. In 3v3_da case, this leaves a lot of room for the offence team to attack. HELIOS or even in 3v3_sa (the in-built agent), tries to defend when the other agent tries to intercept the ball in a micro-region (The larger box near the goal). This is shown in figure (3)

- The book Soccer Defending [4] suggest when one defender is defending the ball, other should stay nearer to the goal compared to the first one; to avoid immediate pass to other offence agent and goal situation

- Upon examining how Helios agent is working. In 3v3 of Helios, we found that when one agent was exploring the entire region and the other agent comes back near goalie; minimise open-angle staying near goalie; attacks the ball when ball's kicked to the goal by an attacker. shown in figure (3), also linked a video (link) showing this. So we thought of implementing the same thing in our solution

- Naturally, we want our agent to intercept the ball when the attacker is very near to the goal post or kicks the ball; To take a defensive action (ReduceAngleToGoal) when the ball is in contact with

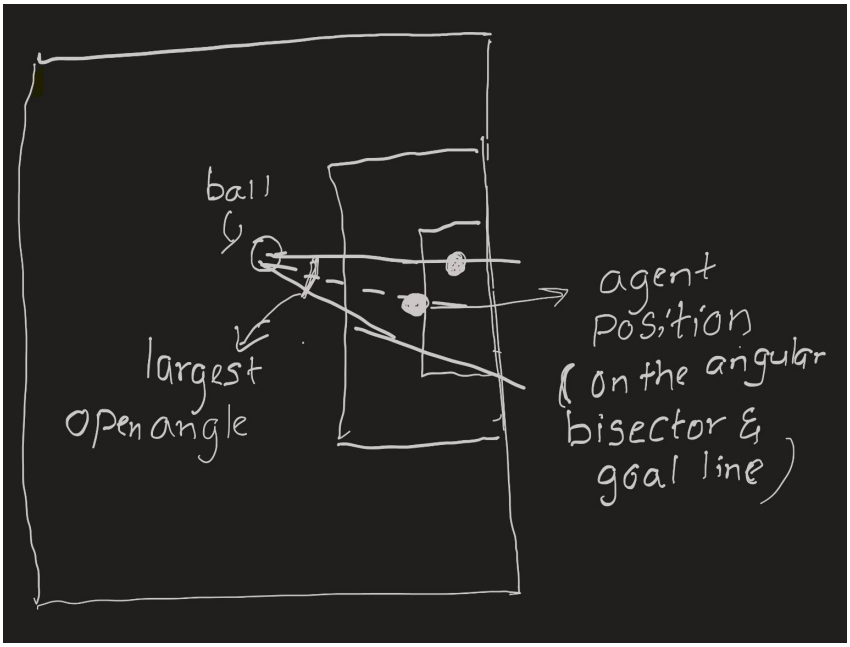

Figure 4: Reduce angle action

the kicker. Our agent is unable to learn this and is constantly intercepting and less defending (ReduceAngleToGoal).

- While **Intercept** predicts ball position accurately and marches towards it, however when the agent is near the ball, it slows down, and so is unable to get hold of the ball when the attacker is dribbling or kicks the ball to goal. This is a shortcoming of the **Intercept** action.
- The Agent2D goalie is not as responsive as Helios goalie (which employs communication with players) which also leads to their better performance.

4.2 Improving the solution

To replicate HELIOS behaviour, we need one agent to explore the entire region, another agent to defend very well when ball enters micro-region (bigger box near goalie). So we started exploring 3v3-sa to make our agent learn to defend well when the ball is in micro-region; This agent can be used in 3v3_da for good defence in micro-region when we train while the another agent explores entire field. We hoped that in our learning algorithms, initializing the agents in this manner will result in learning a co-operative strategy.

There is no action in action set which reduces open-angle to goal and stays near the goalie. So we modified ReduceAngleToGoal action to stay near the goalkeeper and fixing the x-position to be on goal-line gave better results. Figure (4) shows the modified action **Reduce angle**.

After adding the action, we restricted the agent to work only in micro-region (bigger box near goalie) and trained 3v3_sa with only 2-actions Intercept and ReduceAngleToGoal, hoping that it learns to defend (ReduceAngleToGoal) more when the ball in micro-region and intercept when required. We obtained the following result with a defence rate of 0.53, as shown in figure (5).

We added extra features, hoping that micro-region agent learns that it needs to intercept only when the ball is not in contact with an opponent.

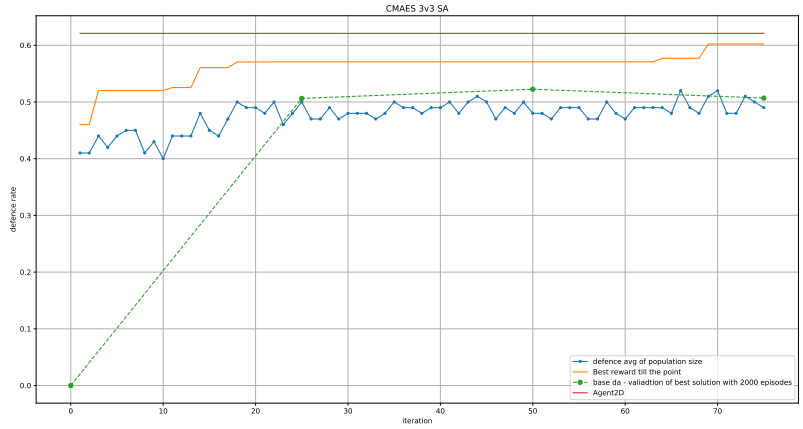

Figure 5: micro-region agent

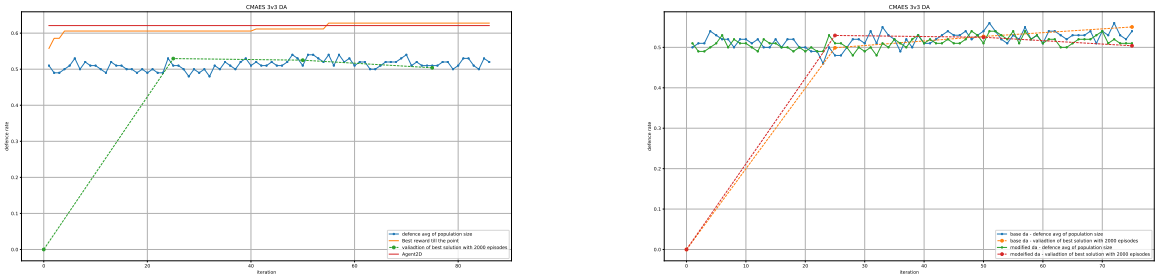

Figure 6: modified agent, comparison

1 **Ball proximity to agent** - The agent's distance from ball.
T **Ball proximity to Teammate** - For each Teammate: distance from ball.
O **Ball proximity to Opponent** - For each opponent: distance from ball.

But there was no change in defence rate, and graph is identical to figure(5)

We then continued to train 3v3_da with pretrained agents, one agent trained for micro-region and works only in it and another agent explores the entire space. Figure (6) shows comparison between normal agents vs modified agents. There was no improvement. As we used pre-trained agents and no new strategies are being learnt, the defence rate doesn't improve much.

4.3 Challenges

- To introduce more complicated macro actions, we believe there is a need to introduce communication between the agents. However, we found it difficult to achieve in an RL setting.

5 Conclusion

Overall, we found the HFO defence task exciting and at the same time, challenging to improve upon. The added macro actions and state variables didn't improve the defence rate as expected, and the learned policy from training seems to take a single action most of the time. Although, our experiments have verified the benefit of using action repetition (AR) in the defence tasks the learned policy still falls short of HELIOS and We were able to figure one way on what areas agent needs to be worked upon to improve the defence. Solving the issues in section (4.1) with better ideas would improve the agent. Additional training against powerful HELIOS agents would have improved defence against Agent2D we couldn't explore this idea as the **rcssserver** kept shutting down with HELIOS as offence agents in CMAES. Using communication to learn appropriate macro actions seems a promising way to improve the defence agent further and is left for future work.

References

[1] Matthew Hausknecht, Prannoy Mupparaju, Sandeep Subramanian, Shivaram Kalyanakrishnan, and Peter Stone. Half field offense: An environment for multiagent learning and ad hoc teamwork. In *AAMAS Adaptive Learning Agents (ALA) Workshop*, May 2016.

[2] Hidehisa Akiyama. Agent2d base code. `https://osdn.jp/projects/rctools/,2010.`

[3] Matthew Hausknecht. Hfo manual. `https://github.com/LARG/HFO/blob/master/doc/manual.pdf`.

[4] Dylan Joseph. Soccer defending: A step-by-step guide on how to stop the other team. `https://www.amazon.com/Soccer-Defending-Step-Step-Understand/dp/1949511111,2010`.