Pekiştirmeli Öğrenme Algoritmalarının DeepRTS Oyunu Üzerinde Performans Karşılaştırması Reinforcement Learning Algorithms Performance Comparison on the Game of DeepRTS

Safa Onur Şahin
Elektrik-Elektronik Mühendisliği Bölümü,
İhsan Dogramacı Bilkent Üniversitesi
ASELSAN Araştırma Merkezi
Ankara, Türkiye
ssahin@bilkent.edu.tr

Veysel Yücesoy
ASELSAN Araştırma Merkezi
Ankara, Türkiye
vyucesoy@aselsan.com.tr

2021 29th Signal Processing and Communications Applications Conference (SIU) | 978-1-6654-3649-6/21/$31.00 ©2021 IEEE | DOI: 10.1109/SIU53274.2021.9477938

***Özetçe* —Bu bildiride,** i**) yapay zeka ile öğrenme amaçlı geliştirilen DeepRTS oyunu için makro aksiyonların kullanıldığı bir çevre oluşturulmuş ve** ii**) belirli pekiştirmeli öğrenme algoritmaları, üzerlerinde gerekli değişiklikler yapılarak bu çevre üzerinde eğitimleri sağlanmış ve ilgili performans analizleri yapılmıştır. Gerçek hayat planlamalarıyla paralellik çizme ve görece düşük donanımlarla bir gerçek zamanlı strateji oyununu oynama amacıyla DeepRTS oyunu üzerinde değişikliklere gidilmiştir. İlk olarak, bir makro aksiyon seti hazırlanmış ve ajanların yalnızca bu set içerisinden aksiyon alabilmesi sağlanmıştır. İkinci olarak, gerçek hayat planlamalarına paralel olarak sistem herhangi bir anda komut alabilecek bütün birimler için aksiyon alınabilecek duruma getirilmiştir. Bu durum makro aksiyonların farklı zaman adımları sürmesi ile birlikte ele alındığında, herhangi bir anda birden fazla aksiyonun başlayıp, birden fazla aksiyonun tamamlanmasına olanak sağladığı için klasik pekiştirmeli öğrenme probleminden bir miktar ayrılmıştır. Ayrıca, literatürde bilinen kredi atama problemine farklı bir boyut ekleyerek daha karmaşık hale getirmektedir. Ajanların eğitiminde kullanılma amacı ile ofansif, defansif ve rastgele kural tabanlı ajanlar oluşturulmuş ve pekiştirmeli öğrenme tabanlı ajanların eğitimleri sırasında dönüşümlü şekilde düşman ajan olarak kullanılmıştır. Eğitimi tamamlanan ajanların birbirlerine karşı oyuncu-1 ve oyuncu-2 olarak performansı raporlanmıştır.**

Anahtar Kelimeler—Derin Pekiştirmeli Öğrenme, Gerçek Zamanlı Strateji, Makro Aksiyon, DeepRTS, Eş Zamanlı Aksiyonlar

***Abstract*—In this paper,** i**) we build a framework using macro actions on top of DeepRTS, which is specifically developed for learning purposes, and** ii**) we train a number of deep reinforcement learning based agents and then we conduct a detailed performance analysis on the performance of these agents. We make changes on the publicly available version of DeepRTS to make it parallel to real life planning problems and make it trainable with a decent hardware. Firstly, we create a set of macro actions based on human heuristics and we train our agent using only these actions. Secondly, similar to the real life planning problem, we change the problem such that we simultaneously take actions for all available unit at a time step. When we consider this situation together with duration of macro actions being different, multiple actions may start and multiple actions ends at any time step. Therefore, the problem draw apart from classical reinforcement learning problem in a certain amount. In addition, the credit assignment problem becomes more sophisticated since each action lasts a duration and at any time multiple actions are conducted. We create three rule based agents to use them as enemy players in training of our agents. We report the performance of the agents as player-1 and player-2 against each others.**

Keywords—Deep Reinforcement Learning, Real Time Strategy, Macro Actions, DeepRTS, Simultaneous Actions

978-1-6654-3649-6/21/$31.00 ©2021 IEEE

I. GİRİŞ

Bu bildiride, derin pekiştirmeli öğrenme kullanılarak, öğrenme amaçlı geliştirilmiş bir gerçek zamanlı strateji oyunu olan DeepRTS için yapay zeka tabanlı ajanlar eğitilmiştir. Pekiştirmeli öğrenmede temel işleyiş, bir ajanın karşısında bulunan çevreyi gözlemlemesi ve bir aksiyon alması, bunun sonucunda bir ödül elde etmesi şeklindedir. Ajan, zaman içerisinde tecrübe kazanarak aksiyon alma algoritmasını ortalama azaltılmış ödülü arttıracak yönde geliştirmektedir. Pekiştirmeli öğrenme, satranç ve Go oyunlarında insana karşı üstünlük sağlamasıyla, üzerinde yoğunlukla çalışılan bir konu haline gelmiştir. Günümüzde, Starcraft [1] ve Dota2 gibi karmaşık oyunlarda dahi insana karşı üstün konuma gelmiştir.

II. GERÇEK ZAMANLI STRATEJİ OYUNU (DEEPRTS) VE YAPILAN GÜNCELLEMELER

Pekiştirmeli öğrenme çalışmalarını desteklemek ve hızlandırmak amacıyla geliştirilmiş olan oyunlar arasında yapılan değerlendirme sonucunda DeepRTS [2] oyunu, durum güncelleme hızı ve fazla basit olan MicroRTS ile tüm oyunu oynamak için fazla karmaşık olan Starcraft oyunları arasında kalan karmaşıklık düzeyi ile bir adım öne çıkmaktadır.

DeepRTS oyunu[1], gerçek zamanlı strateji oyunlarından beklenen pek çok özelliği barındırmaktadır. Oyun içerisinde 3 yapı tipi (üs, kışla ve çiftlik), 3 insan modeli (işçi, okçu ve asker) ve 3 farklı kaynak (altın, tahta ve benzin) tanımlanmıştır. Temel olarak kaynakların toplanması, yapıların inşaatı, insanların hareketi ve saldırması için alt seviye komutlar barındırmaktadır.

[1] https://github.com/cair/deep-rts

TABLO I: MAKRO AKSİYON LİSTESİ

No	Aksiyon Adı	No	Aksiyon Adı
1.	Koordinata Gönder	7.	İşçi Üretme Aksiyonu
2.	Üs Kurma Aksiyonu	8.	Saldırma Aksiyonu
3.	Kışla Kurma Aksiyonu	9.	Konumu Savunma Aksiyonu
4.	Çiftlik Kurma Aksiyonu	10.	Odun Toplama Aksiyonu
5.	Asker Üretme Aksiyonu	11.	Altın Toplama Aksiyonu
6.	Okçu Üretme Aksiyonu		

Pekiştirmeli öğrenme literatürü incelendiği zaman sınırlı işlem gücü ile daha iyi performans alınabilmesi için düşük seviyeli aksiyonlar yerine (mikro aksiyonlar) yüksek seviyeli aksiyonların (makro aksiyonlar) tercih edilmesinin önemli bir rolü olduğu anlaşılmaktadır [3]. Bu bakış açısı ile DeepRTS oyunu için tüm oyuncuların kullanabileceği bir makro aksiyon listesi hazırlanmıştır. On bir adet makro aksiyondan oluşan bu liste Tablo I'de verilmiştir.

Genel pekiştirmeli öğrenme problemine uygun olarak, oyunun açık kaynak kodlu versiyonunda herhangi bir anda yalnızca bir birime aksiyon verilebilmektedir. Gerçek hayat planlama problemlerine paralel olarak, oyunda aynı anda komuta almaya uygun bütün birimler için ayrı ayrı aksiyon alınabildiği bir değişiklik yapılmıştır. Bu değişiklik için oyun motorunun temel işlevleri değiştirilmeden sadece oyuncuların oyun motoru ile etkileşimini sağlayan arayüzde değişiklik yapılmıştır. Ajan karar verirken, oyuncunun durum vektörü ve birimin durum vektörü birleştirilerek, ajana, bulunduğu durumda aksiyon verdiği birimin kimliği bilgisi sağlanmıştır.

DeepRTS oyununda belirli aksiyonların gerçekleşmesi için gerek şartlar bulunmaktadır. Örneğin, kışla kurma aksiyonu için, işçinin o anda bulunduğu konum kışla için uygun olmalıdır. Bu bildiri kapsamında oluşturulan çerçevede bir özellik olarak yapışkan aksiyon yapısı sunulmaktadır. Bu yapıda, kışla kurma emri verildiğinde, işçinin bulunduğu konum bu aksiyon için uygun değil ise, işçi en yakın uygun konuma koordinata gönder aksiyonu ile gönderilerek ardından kışla kurması sağlanmaktadır.

Ajanların eğitimi amacıyla, ofansif, defansif ve rastgele doğrultuda karar alan kural tabanlı ajanlar oluşturulmuştur. Pekiştirmeli öğrenme tabanlı ajanlar bu kural tabanlı ajanlara karşı eğitilmiştir.

III. PROBLEM VE MODEL TANIMI

Pekiştirmeli öğrenme problemlerinde, temel olarak bir Markov Karar Süreci (MKS) ele alınmaktadır. Bu MKS'de, A ayrık aksiyon kümesi, S ayrık durum kümesi, $P : S \times A \times S \rightarrow R$ durumlar arası geçiş olasılık dağılımı olarak ifade edilmektedir. Ayrıca, $R : S \times A \rightarrow R$ ödül fonksiyonunu, $\gamma \in [0, 1]$ gelecekteki ödüllerin ağırlıklarının azaltılma oranını göstermektedir. Klasik bir pekiştirmeli öğrenme probleminde, herhangi bir t anında, çevrenin durumu $s_t \in S$, ajanın bu durumu gözlemleyerek aldığı aksiyon $a_t \in A$, alınan bu aksiyon sonucunda çevrenin geçeceği bir sonraki durum $s_{t+1} \in S$ ve bu geçiş dolayısıyla alınan ödül r_t olarak ifade edilir. Alınan a_t aksiyonunun gelecekteki durumları da etkilemesi sebebiyle anlık ödül fonksiyonu yerine azaltılmış toplam ödül fonksiyonuna, $R_t = \sum_{i=t}^{T} \gamma^{i-t} r_i$, odaklanılmaktadır. Bir pekiştirmeli öğrenme probleminde, eğitilen ajanın amacı, ilk andan itibaren toplanan ortalama azaltılmış ödül fonksiyonunu, $E[R_1]$, en iyileyen plana, $\pi : S \rightarrow A$, ulaşmaktır.

Durum-aksiyon değer fonksiyonu $Q_\pi(s, a)$, durum s_t'de a_t aksiyonu alındığı ve π planı takip edildiği taktirde beklenen ortalama ödül fonksiyonudur ve

$$Q_\pi(s, a) = E[r_t + \gamma r_{t+1} + \ldots | s = s_t, a_t, \pi] \quad (1)$$
$$= E[R | s = s_t, a = a_t, \pi] \quad (2)$$

olarak gösterilebilir. Durum-aksiyon değer fonksyionu aynı zamanda yinelemeli olarak Bellman denklemi şeklinde de yazılabilir: $Q_\pi(s, a) = r(S, a) + \gamma \sum_{S'} P(s'|d, a) Q_\pi(s', \pi(s'))$.

En iyi durum-aksiyon fonksiyonu Q-Öğrenme [4] algoritması ile tablo şeklinde, teorik garantileri olacak şekilde öğrenilebilmektedir. Ancak, mümkün olan durum sayısının çok yüksek olması sebebiyle bu yapay sinir ağları gibi yakınsayıcılara ihtiyaç duyulmaktadır. Durum-aksiyon değerinin kestirimini yapacak sinir ağının parametreleri θ ile gösterildiğinde, durum-aksiyon fonksiyonu $Q(s, a, \theta)$ şeklinde ifade edilmektedir.

DQN algoritması [5], Q-öğrenme algoritmasının yapay sinir ağı yakınsıyacıları ile birleştirilmesi sonucu oluşmuştur. Gerçek durum-aksiyon değeri bilinmediğinden, diğer zamansal fark yöntemleri gibi kendi tahmininden faydalanarak yineleyici bir şekilde öğrenim gerçekleştirir:

$$Y_t^{DQN} = r_{t+1} + \gamma \max_a Q(s_{t+1}, a, \theta_t^-). \quad (3)$$

Burada, θ_t^-, aynı sinir ağının belirli periyotlarda güncellenen bir kopyasını ifade etmektedir.

IV. KULLANILAN PEKİŞTİRMELİ ÖĞRENME ALGORİTMALARI

Bu çalışmada, ajanların eğitimi için pekiştirmeli öğrenme algoritması olarak Rainbow [6], Curl [7], Sunrise [8] ve PPO [9] adlı metotlar kullanılmıştır.

A. Rainbow

Rainbow [5] algoritması, klasik DQN [5] algoritmasının üzerine yapılan farklı çalışmalarda sunulan geliştirmelerin birleştirilmesi ile elde edilmiş bir algoritmadır. Atari oyunlarında insan üstü bir performans göstererek önemli bir kilometre taşı olmuştur. Ayrıca, tecrübe sayısı bakımından verimli bir algoritma olduğu gözlenmiş ve literatürde sıklıkla kullanılmıştır [10]. Birleştirilen geliştirmeler şu şekildedir.

1) Double-Q Öğrenme: Yapay sinir ağlarından gelen kestirim hataları, en iyileme operasyonu ile birleştiğinde, DQN [5] algoritmasının öğrenmesini zorlaştırmaktadır. Literatürde kestirim hatalarına karşı aşırı iyimserlik problemi olarak adlandırılan bu soruna, Double-DQN [11] algoritması, DQN'deki en iyi aksiyonun seçilmesi ve durum-aksiyon değerlerini üretme adımlarını birbirlerinden ayırarak bir çözüm getirmiştir. DDQN algoritmasında durum-aksiyon değerleri, $Y_t^{DDQN} = r_{t+1} + \gamma Q(s_{t+1}, arg\max_{a'} Q(s_{t+1}, a', \theta), \theta_t^-)$ şeklinde hesaplanmaktadır.

2) Düello mimari: Düello mimari [12] kullanan sinir ağlarında çıktı olarak durumun değerini ifade eden $V \in R$ ve avantajı ifade eden $A \in R^{|A|}$ şeklinde iki akış verilmektedir. Bu akışların ilk katmanlarında bulunan özellik çıkarma yapıları ortak olarak kullanılmaktadır. Düello mimaride, durum-aksiyon değerleri, mimarinin çıktıları türünden şu şekilde ifade

edilmektedir:

$$Q(s,a,\theta,\theta_v,\theta_a) = V(s,a,\theta_v) + \left(A(s,a,\theta,\theta_a) - \frac{1}{|A|}\sum_{a'} A(s,a',\theta,\theta_a)\right) \quad (4)$$

3) Çoklu adım ile öğrenme: Çoklu adım ile öğrenme [13] yönteminde, uzun süreli ilişkileri daha güçlendirerek öğrenimi kolaylaştırmak amacıyla Denklem (3)'teki gibi bir sonraki adımı kullanmak yerine $n-$adım sonraki durum-aksiyonunu aday cevap olarak kullanmaktadır: $Y_t^{DQN(n)} = r_t^{(n)} + \gamma_t^k \max_{a'} Q(s_{t+n}, a', \theta^-)$. Burada, $r_t^{(n)}$, n uzunluktaki bir pencere içerisindeki azaltılmış ödül fonksiyonunu ifade etmektedir: $r_t^{(n)} = \sum_{k=0}^{n-1} \gamma_t^k r_{t+k}$.

4) Dağılımsal öğrenme: Dağılımsal öğrenmede, ortalama ödül fonksiyonu, $E[R_1]$, yerine, ödül fonksiyonunun dağılımına odaklanmaktadır [14]. Bu durumda, skaler $Q(s_t, a_t)$ yerine bir dağılımı ifade eden $Z(s_t, a_t)$ kullanılmaktadır. Algoritmanın tahmini ve aday cevap bir dağılım olduğu için kareli hata yerine Kullbeck-Leibler ayrıklığı kullanılmaktadır.

5) Gürültülü Sinir Ağları: ϵ-greedy algoritması, uzun vadeli planlamalara ihtiyaç duyan ve bu sebeple ardışık aksiyonlar gerektiren ortamlarda yeterli keşif performansı gösterememektedir [15]. Bu probleme çözüm amacıyla, kendi ağırlık matrislerinin yanında gürültüyü de operasyona dahil eden gürültülü sinir ağları ortaya çıkmıştır. Gürültülü sinir ağları şu şekilde modellenmektedir: $y = (Wx + b) + \left((W_{noisy} \odot \epsilon^w)x + b_{noisy} \odot \epsilon^b\right)$.

6) Öncelikli Tecrübe Tekrarı: Öncelikli Tecrübe Tekrarı [16], her tecrübeden eşit derece öğrenim yapılamayacağı, bu sebeple tecrübeler arasında bir önceliklendirme yapılması gerektiği prensibine dayanır. Bu önceliklendirme son karşılaşılan zamansal fark hatasına bağlı olarak: $p_i \propto |Y_t^{DQN} - Q(s_t, a, \theta)|^\omega$ şeklinde gerçekleştirilir. Burada p_i, $i.$ indekste bulunan tecrübenin örneklenme olasılığıdır. ω ise dağılımın şeklini belirleyen bir parametredir.

B. CURL

CURL (Contrastive Unsupervised Representations for Reinforcement Learning) [7] algoritması devamlı aksiyon setleri için SAC [17], ayrık aksiyon setleri için Rainbow algoritmasının üzerine inşa edilmiş, durum s_t'nin daha iyi temsil edilmesi amacı taşıyan bir algoritmadır. Bu amaç doğrultusunda, eğitim sırasında öncelikle rastgele kırpma ve kaydırma gibi işlemler ile s_t üzerinde veri arttırımı operasyonu gerçekleştirilerek, s_t^a ve s_t^h üretilir. Hedef, aynı s_t üzerinden türetilmiş iki durumun örtülü uzaydaki temsillerinin birbirlerine yakın olmasını, farklı durumlardan türetilen temsillerin ise uzak olmasını sağlamaya çalışmaktadır. Bu amaçla, ajanın eğitimi sırasında karşıtsal bir hasar fonksiyonu kullanılmaktadır:

$$L_q = \log \frac{\exp q^T W k_+}{\exp q^T W k_+ + \sum_{i=0}^{K-1} \exp q^T W k_i}. \quad (5)$$

Burada, k_+, sorgu q ile aynı durumdan üretilmiş s_t^h'i ifade etmektedir. $q^T W k_+$ ifadesi iki durum arasındaki benzerliği bileener olarak modellemektedir.

C. Sunrise

Sunrise [8] algoritması da CURL'e paralel olarak devamlı aksiyon setleri için SAC [17], ayrık aksiyon setleri için Rainbow algoritmasının üzerine inşa edilmiştir. Sunrise algoritması, belirli bir sayıda ajanın birleşiminden oluşmaktadır. Temel olarak üç geliştirme öne sürmektedir. İlk olarak, ajanların kullandığı sinir ağlarının farklı ilklendirilmelerinden dolayı, bütün ajan daha gürbüz bir öğrenim gerçekleştirmektedir. İkinci olarak, ağırlıklandırılmış Bellman Yedeklemesi ile ajanların fikir birliğine vardığı tecrübelerin daha yüksek ağırlıkla, fikir ayrılığına düştüğü tecrübelerin ise daha düşük ağırlıkla güncellenmesini sağlayarak eğitimi daha gürbüz bir hale getirmektedir. Ağırlıklandırılmış Bellman Yedeklemesinde hasar fonksiyonu aşağıdaki gibidir:

$$L_{WQ} = (\omega(s_{t+1}))\left(Q(s_t, a_t, \theta) - r_t - \gamma \max_a Q(s_{t+1}, a, \theta^-)\right) \quad (6)$$

Burada, ağırlık $\omega(s) = \sigma(-\bar{Q}_{std}(s) \times T) + 0.5$ olarak tanımlanmaktadır. T bir sıcaklık parametresidir. $\bar{Q}_{std}(s)$ seçilen aksiyon için ajanların önerdiği Q değerlerinin deneysel standard sapmasını ifade etmektedir. Üçüncü olarak, ajan eğitim sırasında UCB (Upper Confidence Bound) keşfini kullanmaktadır. Eğitim sırasında seçilecek aksiyon $a_t = \max_a\{Q_{\text{mean}}(s_t, a) + \lambda Q_{\text{std}}(s_t, a)\}$ formülü ile belirlenmektedir. Burada, $Q_{\text{mean}}(\cdot)$ ve $Q_{\text{std}}(\cdot)$ ortalama ve standard sapma değerlerini ifade etmekte olup her bir aksiyon için ayrı olarak hesaplanır. λ bir dengeleme parametresidir.

D. PPO

PPO (Proximal Policy Optimization) [9] aktör-kritik yapısına sahip bir algoritmadır. Aktör aksiyon alarak çevre ile etkileşime girerken, kritik yapısı bulunulan durumun değerini tahmin eder. Bir aktör-kritik yapısında, kritik kareli hata ile eğitilirken, aktör, kritik yapısının değerini arttıracak yönde güncellemeler almaktadır. PPO algoritmasında ise aktör avantaj kestirimini, $\hat{A}_t = -V(s_t) + r_t + \sum_{k=t+1}^{T} \gamma^k r_{k-t} + \gamma^{T-t} V(s_T)$, arttıracak yönde güncellemeler almaktadır. PPO aktör için muhafazakar bir güncelleme önermektedir. Bu amaçla aktör için önerdiği hasar fonksiyonu aşağıdaki gibidir:

$$L_t^{Clip}(\theta) = E\left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon+1+\epsilon)\hat{A}_t)\right] \quad (7)$$

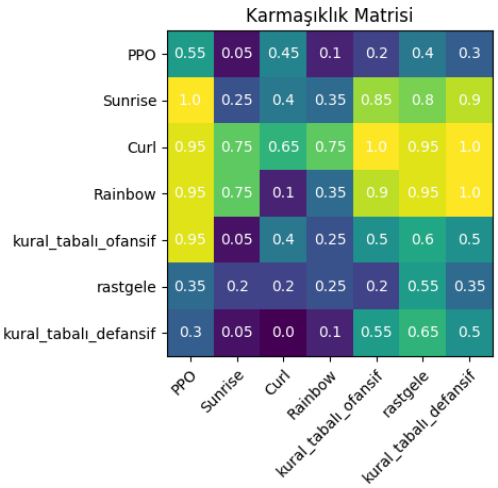
Burada, $r_t(\theta) = \frac{\pi_\theta(a_t, s_t)}{\pi_{\theta\,old}(a_t, s_t)}$ ifadesi belirli sınırlar arasına hapsedilerek, sinir ağının güncelleme öncesi ve sonrası arasındaki çıktılarının arasındaki uzaklık belirli bir seviyenin altında tutulmaktadır.

V. Deneysel Veriler

Bu bölümde, bu bildiri kapsamında eğitimi gerçekleştirilen derin pekiştirmeli öğrenme algoritmalarından Rainbow, Curl, Sunrise ve PPO algoritmalarının oyuncu-1 ve oyuncu-2 olarak birbirlerine ve kural tabanlı ajanlara karşı performansı sunulmaktadır. Tüm ajanlar için eğitimler üç adet kural tabanlı ajana karşı gerçekleştirilmiştir. Rainbow, Curl ve Sunrise için her 5000 adımda bir (oyun tamamlandıktan sonra), bir sonraki kural tabanlı ajan düşman olarak seçilmiştir. PPO algoritması içinse paralel çevreler kullanılabildiği için her çevrede düşman kural tabanlı ajan $k = \mod$ (Çevre Numarası$/3$) olacak

şekilde {Defansif, Rastgele, Ofansif} içerisinden verilen sıra ile seçilmiştir. Her 50.000 eğitim adımında, ajan üç kural tabanlı ajana karşı 5 oyunluk teste tabi tutulmuştur. Ajanın eğitim sonucundaki en iyi ortalama test skoruna sahip olan versiyonu o ajanın final versiyonu olarak seçilmiştir.

Rainbow, Curl ve Sunrise algoritmaları için eğitime başlanmadan önce 20×10^3 zaman adımı boyunca, tekrar hafızasında tecrübe biriktirilmiştir. PPO'nun "on-policy" bir algoritma olması sebebiyle eğitimine direkt olarak başlanmıştır. Rainbow, Curl ve Sunrise algoritmaları için tekrarında parti büyüklüğü 32, Adam en iyileyicisinin öğrenme adımı: 625×10^{-4}, $\epsilon_{Adam} = 1.5 \times 10^{-4}$, gürültülü sinir ağlarının standart sapması: $\sigma_0 = 0.5$, hedef plan ağının güncellenme periyodu: 8000 öğrenme adımıdır. Dağılımsal öğrenmedeki atom sayısı 51 seçilmiştir ve değerler $[-200, 200]$ aralığına yerleştirilmiştir. Gürültülü sinir ağları kullanılmasına rağmen daha fazla keşif ihtiyacı olduğu gözlemlenmiş ve $\epsilon-$greedy algoritmasının eşik değeri eğitim sırasında $\epsilon = 0.05$, test sırasında $\epsilon = 0.01$ olarak seçilmiştir. Önceliklendirilmiş tecrübe tekrarı için önceliklendirme üssü $\omega = 0.5$ olarak seçilmiş, önemlilik örneklemesi ise $\beta = 0.4 \rightarrow 1.0$ olacak şekilde değiştirilmiştir. Aksiyonların sürelerinin farklılık göstermesi sebebi ile, çoklu adım ile öğrenmede, her aksiyon için kendi süresi baz alınmıştır.



Şekil 1: Satırlar oyuncu-1, sütunlar ise oyuncu-2 olarak aksiyon alan pekiştirmeli öğrenme algoritmasını temsil etmektedir. Matriste bulunan her bir girdi 20 oyun üzerinden değerlendirilmekte olup oyuncu-1'in oyuncu-2'ye karşı kazanma oranını belirtmektedir.

Şekil 1'de gösterilen sonuçlara göre, bir ajanın her iki oyuncu için performansı $S_i = \sum_{j=1}^{7} C_{ij} + (1 - C_{ji})$ şeklinde hesaplandığında, en yüksek 13.00 olan skor üzerinden 11.85 ile Curl algoritması en iyi skoru elde etmektedir. C, karmaşıklık matrisini ifade etmektedir.

VI. Sonuçlar

Bu bildiride, DeepRTS adlı gerçek zamanlı strateji oyunu üzerine pekiştirmeli öğrenme algoritmaları kullanan ajanlar eğitilmiştir. DeepRTS, açık kaynak kodlarının üzerinde değişiklikler yapılarak, bir zaman adımında, emir almaya uygun bütün birimlere komut verilebilir hale getirilmiştir. Ayrıca, belirli aksiyonlar için yapışkan aksiyon yapısı kullanılarak uygun olan en yakın koordinata giderek orada bina kurma komutu gerçeklenmiştir. İnsan içgüdülerine uygun hareket eden üç adet kural tabanlı ajan oluşturularak, pekiştirmeli öğrenme tabanlı ajanların eğitimi bu ajanlara karşı gerçekleştirilmiştir. Ajanların performansı oyuncu-1 ve oyuncu-2 olarak ayrı ayrı birbirlerine karşı değerlendirilmiştir. En yüksek performansı Curl algoritması ile eğitilen ajan göstermiştir.

Kaynaklar

[1] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev *et al.*, "Grandmaster level in starcraft ii using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[2] P.-A. Andersen, M. Goodwin, and O.-C. Granmo, "Deep rts: a game environment for deep reinforcement learning in real-time strategy games," in *2018 IEEE conference on computational intelligence and games (CIG)*. IEEE, 2018, pp. 1–8.

[3] P. Sun, X. Sun, L. Han, J. Xiong, Q. Wang, B. Li, Y. Zheng, J. Liu, Y. Liu, H. Liu *et al.*, "Tstarbots: Defeating the cheating level builtin ai in starcraft ii in the full game," *arXiv preprint arXiv:1809.07193*, 2018.

[4] C. J. C. H. Watkins, "Learning from delayed rewards," 1989.

[5] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[6] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver, "Rainbow: Combining improvements in deep reinforcement learning," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[7] A. Srinivas, M. Laskin, and P. Abbeel, "Curl: Contrastive unsupervised representations for reinforcement learning," *arXiv preprint arXiv:2004.04136*, 2020.

[8] K. Lee, M. Laskin, A. Srinivas, and P. Abbeel, "Sunrise: A simple unified framework for ensemble learning in deep reinforcement learning," *arXiv preprint arXiv:2007.04938*, 2020.

[9] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[10] S. O. Şahin and V. Yücesoy, "A generalized circle agent based on the deep reinforcement learning for the game of geometry friends," in *2020 28th Signal Processing and Communications Applications Conference (SIU)*. IEEE, 2020, pp. 1–4.

[11] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Thirtieth AAAI conference on artificial intelligence*, 2016.

[12] Z. Wang, T. Schaul, M. Hessel, H. Van Hasselt, M. Lanctot, and N. De Freitas, "Dueling network architectures for deep reinforcement learning," *arXiv preprint arXiv:1511.06581*, 2015.

[13] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Machine learning*, vol. 3, no. 1, pp. 9–44, 1988.

[14] M. G. Bellemare, W. Dabney, and R. Munos, "A distributional perspective on reinforcement learning," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 449–458.

[15] M. Fortunato, M. G. Azar, B. Piot, J. Menick, I. Osband, A. Graves, V. Mnih, R. Munos, D. Hassabis, O. Pietquin *et al.*, "Noisy networks for exploration," *arXiv preprint arXiv:1706.10295*, 2017.

[16] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.

[17] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," *arXiv preprint arXiv:1801.01290*, 2018.