PEKİŞTİRMELİ ÖĞRENME YÖNTEMİYLE FARKLI GÖREVLERİN ROBOTLARA ÖĞRETİLMESİ

Rüstem ÖZAKAR

Yüksek Lisans Tezi
Bilgisayar Mühendisliği Anabilim Dalı
Yrd. Doç. Dr. Barış ÖZYER
2016

ATATÜRK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

PEKİŞTİRMELİ ÖĞRENME YÖNTEMİYLE FARKLI GÖREVLERİN ROBOTLARA ÖĞRETİLMESİ

Rüstem ÖZAKAR

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

ERZURUM
2016

T.C.
ATATÜRK ÜNİVERSİTESİ
Fen Bilimleri Enstitüsü Müdürlüğü

TEZ ONAY FORMU

PEKİŞTİRMELİ ÖĞRENME YÖNTEMİYLE FARKLI GÖREVLERİN ROBOTLARA ÖĞRETİLMESİ

Yrd. Doç. Dr. Barış ÖZYER danışmanlığında, Rüstem ÖZAKAR tarafından hazırlanan bu çalışma, 30/09/2016 tarihinde aşağıdaki jüri tarafından Bilgisayar Mühendisliği Anabilim Dalı Yüksek Lisans tezi olarak **oybirliği** / ~~**oy çokluğu**~~ **(.../...)** ile kabul edilmiştir.

Başkan: Yrd. Doç. Dr. Umut TİLKİ İmza :

Üye : Yrd. Doç. Dr. Barış ÖZYER İmza :

Üye : Yrd. Doç. Dr. Tolga AYDIN İmza :

Yukarıdaki sonuç;

Enstitü Yönetim Kurulu'nun 06./10./2016 tarih ve ..38..../..35..... nolu kararı ile onaylanmıştır.

Prof. Dr. Cavit KAZAZ
Enstitü Müdürü

Not: Bu tezde kullanılan özgün ve başka kaynaklardan yapılan bildiriş, çizelge, şekil ve fotoğrafların kaynak olarak kullanımı, 5846 sayılı Fikir ve Sanat Eserleri Kanunundaki hükümlere tabidir.

ÖZET

Yüksek Lisans Tezi

PEKİŞTİRMELİ ÖĞRENME YÖNTEMİYLE FARKLI GÖREVLERİN ROBOTLARA ÖĞRETİLMESİ

Rüstem ÖZAKAR

Atatürk Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Yrd. Doç. Dr. Barış ÖZYER

Pekiştirmeli Öğrenme, bilgisayar bilimleri dalı başta olmak üzere, birçok bilim dalında kullanılan, son yıllarda oldukça yoğun ilgi gösterilen bir makine öğrenmesi metodudur. Pekiştirmeli öğrenmede, ajan istenen bir davranışı deneme-yanılma yöntemiyle öğrenir. Önceden hazırlanmış bir veriseti ile eğitim görmez. Öğrenme ödül sistemiyle gerçekleşir. Bu tezde, üç farklı kontrol problemine, üç farklı pekiştirmeli öğrenme algoritması farklı parametreler kullanılarak uygulanmıştır. Bu problemlerden ilki, literatürde sıklıkla çalışılan ters sarkacın *(inverted pendulum)* dengede tutulması problemidir. Diğer iki problem, top düşürmeme ve top fırlatma, bu tezde oluştur-duğumuz yeni problemlerdir. top düşürmeme, iki robotik link üzerine bırakılan topun linklerin hareketiyle düşmeden tutulmasını öğretmeyi amaçlamaktadır. Top fırlatma, sabit açısal hızla dönen bir linkin ucuna bağlı bir topu en uzağa fırlatabilmek için topun fırlatılacağı anı öğretmeyi amaçlamaktadır. Bu problemlere, Q-öğrenmesi, SARSA ve Adaptif Sezgisel Kritik *(Adaptive Heuristic Critic)* algoritmaları, Box2d simülatöründe robotik manipülatörler oluşturularak gerçeklenmiştir. Bu manipülatörler için, bir pekiştirmeli öğrenme durum-uzay oluşturma biçimi olan, yeni Boxes algoritmaları geliştirilmiştir. Simülasyonlarda elde edilen sonuçlar detaylı bir şekilde analiz edilmiştir.

2016, 79 sayfa

Anahtar Kelimeler: Pekiştirmeli öğrenme, makine öğrenmesi, robotik, simülasyon

ABSTRACT

MS Thesis

TEACHING DIFFERENT TASKS TO ROBOTS WITH REINFORCEMENT LEARNING

Rüstem ÖZAKAR

Atatürk University
Graduate School of Natural and Applied Sciences
Department of Computer Engineering

Supervisor: Asst. Prof. Dr. Barış ÖZYER

Reinforcement learning is a commonly researched machine learning method which is researched in several disciplines especially in computer science in recent years. In reinforcement learning, an agent learns the desired behavior by trial and error. There isn't a pre-defined training dataset in reinforcement learning. Learning is actualized through a reward system. In this thesis, three different reinforcement learning algorithms are applied to three different control problems using different parameters. First one of these problems is balancing the inverted pendulum which is commonly researched in literature. Other two problems, ball-cradling and ball-throwing are newly created problems in this thesis. Ball-cradling aims for teaching the balancing of a ball which falls onto two robotic links by the movement of the links. Ball-throwing aims for teaching the moment to throw a ball to furthest distance, which is tied to a rotating link with a constant linear velocity. Q-learning, SARSA and Adaptive Heuristic Critic algorithms are applied to these problems using the robotic manipulators created in Box2d simulator. For each of these manipulators, new Boxes algorithms, a way of defining reinforcement learning state-spaces, were created. The results of simulations are analyzed particularly.

2016, 79 pages

Keywords: Reinforcement learning, machine learning, robotics, simulation

TEŞEKKÜR

Bu çalışma sürecinde, danışmanım Sayın Yrd. Doç. Dr. Barış ÖZYER'e, karşılaştığım sorunlarda anlayışlı ve yardımcı yaklaşımı dolayısıyla bünyesinde çalıştığım Erzurum Teknik Üniversitesi Mühendislik Fakültesi Dekanı Sayın Prof. Dr. İrfan KAYMAZ'a, Sayın Prof. Dr. Bülent ÇAKMAK'a, katkılarından dolayı Sayın Yrd. Doç. Dr. Bülent ÇAVUŞOĞLU'na teşekkürlerimi sunarım.

Rüstem ÖZAKAR
Ekim, 2016

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

1. GİRİŞ

1.1. Amaç

Son yıllarda, robot tasarımında, elektronik ve mekanik kısımların teknolojik ilerlemelerle gelişmesi sayesinde büyük ilerlemeler kaydedilmiştir. İşlemci ve hafıza kapasitesi artan robotlar, makine öğrenme algoritmalarının uygulanması sayesinde, zor ve karmaşık işleri yerine getirebilen, algılayabilen, kendi kendine öğrenebilen sistemler haline gelmiştir. Robotlar, endüstri alanında kullanılan makineler olmaktan çıkıp, insanların günlük hayatına girmeye başlamıştır. Bulaşık makinesinden bulaşıkları boşaltan, etrafı süpüren, futbol oynayan robotlar günümüzde artık insanların benimsediği bir olgu olmuştur. Gözetimli öğrenme *(supervised learning)*, gözetimsiz öğrenme *(unsupervised learning)* ve pekiştirmeli öğrenme, araştırmacıların robotlara bu tarz işleri yaptırmakta yaygın olarak kullandıkları metodlar olmuştur (Kober *et al*. 2013).

Robotik manipülatörler, önceden programlanmış tekrarlanabilen görevleri başarıyla gerçekleştirebilmektedir. Özellikle endüstriyel uygulamalarda kullanılan bu robotlar tutma, kaldırma, delme vb. gibi önceden tanımlanmış görevleri belirli bir ortam içerisinde uygulayabilmektedir. Peki, bu robotlar bilinmeyen bir ortamda daha önceden tanımlanmamış bir görevi yerine getirebilir mi? Bu soru son yıllarda robotik, makine ögrenmesi ve sinirbilimi gibi farklı alanlarda çalışan araştırmacıların üzerinde çalıştığı zor bir konudur (Najafi *et al.* 2013). Robotların önceden programlanmış hareketlere sahip olması, onların tanımadıkları bir ortamda başarılı olmalarını zorlaştırmaktadır (Khansari-Zadeh *et al.* 2012). Bu probleme çözüm getirmek amacı ile insan davranışları ve öğrenmesini örnek alan pekiştirmeli öğrenme metodları geliştirilmiştir. Pekiştirmeli öğrenme, önceden bir kontrol mekanizması tasarlamanın zor olduğu problemlerde kullanılmaya uygundur (Adam *et al.* 2012).

İnsan, doğduğu andan itibaren temel duyu organlarını kullanarak çevresi ile etkileşime

girip, öğrenen akıllı bir varlıktır. Geçmişte öğrendiği bilgileri beyninde saklayarak yeni bir durum ile karşılaştığında karar verme özelliğine sahiptir. Bu öğrenme mekanizması, deneme-yanılma yönteminden büyük oranda faydalanarak çalışmaktadır. Pekiştirmeli öğrenme, bu deneme-yanılma metodunu baz alarak geliştirilmiş bir makine öğrenmesi türüdür. Pekiştirmeli öğrenmenin diğer makine öğrenme metodlarından bariz farkı, önceden hazırlanmış bir veriseti ile alıştırma yapılmamasıdır. Pekiştirmeli öğrenme bugün birçok robotta, onlara yeni davranışlar öğretmede kullanılmaktadır (Schaal 1999). Bu robotlar futbol, tenis veya beyzbol oynayabilmekte, kendi kendilerine uçabilmektedir (Sammut 1994; Andrew *et al.* 2006; Riedmiller *et al.* 2009).

Biz bu çalışmamızda, pekiştirmeli öğrenme algoritmaları kullanarak, üç farklı probleme, ters sarkaç (*inverted pendulum*), top düşürmeme, top fırlatma sistemlerinin dengede kalmasını kontrol edebilen ve bu sistemin farklı başlangıç koşullarına göre kendi kendine dengede kalmasını öğrenen yöntemler önerdik ve denetim ortamında simülasyonunu gerçekleştirdik. Ters sarkaç, top düşürmeme, top fırlatma sistemi üzerinde, üç farklı pekiştimeli öğrenme algoritmasının performanslarını inceledik ve karşılaştırdık. Top düşürmeme problemi, birbirine eklemlenmiş iki link üzerine bırakılan bir topun, yere düşmeden linkler tarafında kontrol edilebilmesini amaçlamaktadır. Top fırlatma probleminde ise, sürekli aynı yönde dönen bir link üzerine bağlı bulunan bir topun, en uzağa fırlatılabilmesi için bırakılması gereken noktayı sisteme öğretme amaçlanmaktadır. Top düşürmeme ve top fırlatma problemleri, yeni birer pekiştirmeli öğrenme problemleri olup, böylesi bir fizik simülatöründe modellenmeleri, bu alanda yeni olup, bu tez ile getirdiğimiz yeniliklerden bir diğeridir. Bu problemler üzerinde çalışılıp tecrübe kazanıldıkça, elde edilen verilerle bir robot elin parmaklarıyla yine pekiştirmeli öğrenme kullanılarak bu problemler gerçeklenebilir.

Bu tezde çalışılan ters sarkaç, top düşürmeme ve top fırlatma problemleri, birer kontrol problemi olup, üçünde de amaç benzerdir. Ters sarkaç'ta amaç çubuğu araç üzerinde düşürmeden en uzun süre dengede tutmak iken, top düşürmemede amaç topu yere düşürmeden en uzun süre tutmaktır. Top fırlatmada ise topu en uzak mesafeye fırlatmaktır. Üç problem de, kullanılan simülatörde (Box2d), yerçekim ivmesi 9.8 olan

bir çevre modeli kullanılarak gerçeklenmiştir.

Ters sarkaç sisteminde, araç kontrolü sağa veya sola uygulanacak kuvvetlerin seçilip bunların tekerleklere dönme kuvveti olarak uygulanması ile sağlanmaktadır. Top düşürmemede ise, linklere belirlenen bir açısal hız yukarı ya da aşağı doğru uygulanmaktadır. Top fırlatma'de ise, dönen linke bağlı bulunan topun fırlatılabilmesi için, topun konumuna göre her iki yönde kuvvet uygulanmaktadır. Ters sarkaçta, sistemin kontrolü için, çubuğun açısı, çubuğun açısal hızı, aracın konumu ve aracın hızından faydalanılmaktadır. Top düşürmemede ise, topun konumu, topun lineer hızı, linkin açısı ve linkin açısal hızı ile kontrol sağlanmaktadır. Top fırlatmada, topun x konumu, topun y konumu, linkin açısı kullanılmaktadır.

Pekiştirmeli öğrenme algoritması, bir ajana (robot) çevreyle etkileşim tecrübesiyle, durum ve hareket tahminlerini bir ödül veya ceza fonksiyonu kullanarak öğretme yöntemidir. Bu öğrenme deneme ve yanılma ile gerçekleşir. Ajanın çevreyi keşfedip istenen davranışı kendi kendine öğrenmesi beklenir. Pekiştirmeli öğrenme algoritması özellikle insansı robotlarda masa tenisi veya beyzbol oynayan, dengede kalma veya yürüme gibi zor görevleri öğretmek amacı ile literatürde uygulanmıştır.

Çalışmamızda, Box2d adlı ücretsiz, açık kaynak kodlu simülasyon ortamı tercih edilmiştir. Box2d, gerçekçi bir fizik modelleme simülasyonu olup, iki boyutlu fiziksel etkileşimleri ekrana görsel olarak en doğru biçimde yansıtmaya yarayan bir kütüphanedir.

1.2. Tezin Sunduğu Katkılar

- Bir fizik simülasyonu kütüphanesi yardımıyla, robot manipülatörlerle üç farklı probleme pekiştirmeli öğrenme algoritmaları uygulanmış ve analiz sonuçları paylaşılmıştır.
- Ters sarkaç probleminin, daha önceki çalışmalarda yaygın olmayan bir biçimde, fizik simülatörü kullanılarak simülasyonu gerçekleştirimiştir.

• Kendi oluşturduğumuz top düşürmeme ve top fırlatma problemleri için, yeni Boxes algoritmaları oluşturulmuştur. Boxes algoritmasının başka problemlere adapte edilebilirliği gösterilmiştir.
• Box2d ortamında, yeni problemler oluşturmaya uygun bir iskelet geliştirilmiştir.
• Pekiştirmeli öğrenme detaylı bir konu anlatımıyla sunulmuştur.

1.3. Tezin Organizasyonu

Tezin organizasyonu şu şekildedir; ilk bölümde, giriş kısmında amaç, tezin sunduğu katkılar ve tezin organizasyonu sunulmuştur. İkinci bölümde, robotikte pekiştirmeli öğrenme kullanımı, pekiştirmeli öğrenmenin literatürdeki örnekleri ve ters sarkaç üzerine literatür örnekleri irdelenmiştir. Üçüncü bölümde, problemlerin tanımı yapılıp, metodoloji ve matematiksel modeller anlatılmıştır. Dördüncü bölümde, simülasyon ve elde edilen sonuçlar, kullanılan sistem ve araçlar anlatılmıştır. Beşinci bölümde tezin sonuçları tartışılmıştır. Ek olarak pekiştirmeli öğrenme geniş konu anlatımı sunulmuştur.

2. KURAMSAL TEMELLER

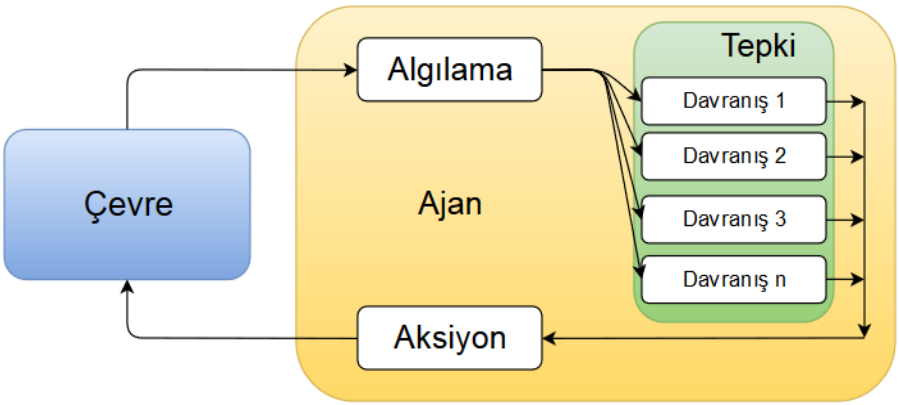
2.1. Pekiştirmeli Öğrenme

Makine öğrenmesi, kontrol mühendisliği, yöneylem araştırması ve robotik dalında kullanılan reinforcement learning, aktif bir araştırma konusu olmuştur. Bir supervision veya bütün çevre modeli olmadan hedef tabanlı öğrenmeyi bilgisayar yardımıyla otomatikleştirme yöntemine pekiştirmeli öğrenme *(reinforcement learning)* denir. Ajan, başlangıçta bilinmeyen bir ortama bırakılır ve zamanla çevreden geribesleme alır. Bu geribeslemeye ödül *(reward)* denir. Amaç, en yüksek ödülü elde edecek aksiyonları öğrenmektir (Sutton and Barto 1998; Xu *et al.* 2002).

Pekiştirmeli öğrenme, durumları aksiyonlara nasıl haritalayacağını, ne yapacağını öğrenmektir. En önemli iki elementi, deneme yanılmayla arama ve bekletilen ödüldür. Pekiştirmeli öğrenme ajanı önceden belirlenmiş bir çevre temsiline gerek duymaz. Ajan, geçmiş tecrübelerine dayanarak bir aksiyon seçer ve gözlemler, amacına ulaşana kadar bu böyle devam eder. Bu süreçte ajanın davranışını belirleyecek bir ilişki kurulur (Faustino 2011).

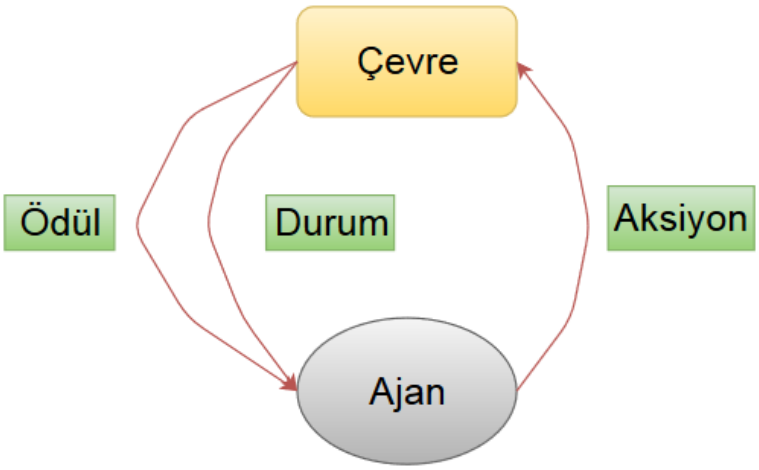
Pekiştirmeli öğrenme, denetlenen öğrenmeden *(supervised)* farklıdır. Denetlenen öğrenmede problemler tek adım iken pekiştirmeli öğrenmede çoklu adımdır. Çoklu adım problemlerinde, öğrenen sistem gelecek seçimlere dayalı sonuçları tahmin etmelidir.

Ödül, s durumundan a aksiyonuyla geçişin ne kadar etkili olduğunu gösteren pekiştirmeli öğrenme sinyalidir. Denetlenen öğrenmede öğrenme aşamasında ajan bir aksiyon seçtiğinde bunun doğru ya da yanlış olduğu ona söylenir. Böylece istenen sonuçla üretilen sonuç arasındaki hata payı minimize edilmeye çalışılır. Pekiştirmeli öğrenmede bu yoktur. Hareketin doğruluğu ya da yanlışlığı bildirilmez. Sadece anlık ödül vardır (Faustino 2011).



Şekil 2.1. Ajan mimarisi (Faustino 2011)

Pekiştirmeli öğrenmenin anahtar özelliklerinden biri bekletilmiş ödüllerdir. Önemli bir ödül katkısı son durumlara ulaşınca elde edilebilir, daha önemsiz ödüller ise bu sırada gerçekleştirilebilir. Bu da öğrenme sürecini zorlaştırabilir, son duruma ulaşmak için geçtiği bütün statelere önemli ödül değerini geri-yayılım etmelidir. Ödüle gecikmeli denmesinin sebebi budur. Ajan anlık ödülleri uzun dönemde toplam olarak maksimize edebilmelidir (Faustino 2011).



Şekil 2.2. Pekiştirmeli öğrenme (Faustino 2011)

Pekiştirmeli öğrenme metodları en uygun davranışı deneme yanılmayla öğrenir, en uygun davranış için bilgi gerektirmez. Bu özellik onu ilginç kılar, çünkü örneklere bağlı

değildir. Fonksiyon tahminleyiciler örneklerden daha iyi olmayı öğrenemezler. Ancak Pekiştirmeli öğrenmede bu limit yoktur (Nissen 2007).

Fonksiyon tahminleyicileri, yapay sinir ağları gibi, en uygun bir davranışı örneklere bakarak öğrenir. Bu yaklaşım en uygun davranışın mevcut olduğu zamanlarda kullanışlıdır. Ancak birçok problem için en uygun davranış mevcut değildir (Nissen 2007).

Pekiştirmeli öğrenme modeli bir ajan ve bir çevre içerir. T zamanında çevre, ajana bir durum sunar ve ajan da buna bağlı olarak bir hareket gerçekleştirir. Ajanın hareketinden sonra çevre ve durum güncellenir. Bunlara ek olarak, çevre bir nümerik ödül de sunar. Bu da anlık bir ödül ya da cezalandırma şeklindedir. Ajanın seçeceği hareket poliçeye bağlıdır. Pekiştirmeli öğrenmenin amacı poliçeyi uzun dönem ödülü maksimize etmesi için modifiye etmektir (Nissen 2007).

Pekiştirmeli öğrenme modeli, ayrık zaman adımları *(t)* ve ayrık durum setlerinden *(s)* ve ayrık aksiyon setlerinden *(a)* oluşur. Poliçeye göre ajan aksiyon seçer. Genel olarak iki çeşit poliçe fonksiyonu vardır, stokastik poliçe fonksiyonu; ajanın s durumundayken a aksiyonunu seçme olasılığı ve deterministik; ajanın s durumunda a aksiyonunu seçmesi. Olasılıksal poliçe fonksiyonu açgözlü bir strateji seçilerek deterministik yapılabilir (Nissen 2007).

Ajan aksiyon seçerken sadece şimdiki duruma bakar, önceki aksiyon ve durumlar irdelenmez. Çevre sabittir. A aksiyonunu seçip s' durumuna gidip r ödülünü kazanma olasılığı hep aynıdır. Pekiştirmeli öğrenme problemi aralıklı olabilir. S'teki bazı durumlar hedef durumlardır, ajan bunlara ulaştığında yeniden başlatılmalıdır. Aralıklı olmayan pekiştirmeli öğrenme problemleri de vardır. S hedef durum içermez. Eğer standart pekiştirmeli öğrenme modeli gerçek yaşam problemlerine uygulanacaksa, şu adımlarla yapılabilir (Nissen 2007);

- Eğer bir problem önceki adımların hatırlanmasını gerektiriyorsa, durum bu bilgiyi

içerecek şekilde değiştirilebilir.

- Eğer bir problem gerçek değerli bir parametre gerektiriyorsa, hız gibi, bu parametre ayrık değerlere bölünebilir.
- Gerçek değerli aksiyonlar da ayrık hale getirilebilir.
- Eğer çevre tamamen statik değilse bu state bu statik olmayan parçayı da içerecek şekilde modifiye edilebilir, bu çevreyi de statik yapar.

Pekiştirmeli öğrenmede iki element vardır. Bunlar, öğrenme tahmini ve öğrenme kontrolüdür. Öğrenme kontrolünde amaç, modelini bilmeden Markov karar verme sürecinin en uygun poliçesini veya en uygun değer fonksiyonunu tahmin etmektir. Öğrenme tahmini, hareketsiz poliçe bir Markov karar verme sürecinin poliçe değerlendirme problemini öncül bir model olmadan çözmeyi hedefler ve öğrenme kontrolünün alt bir problemi olarak kabul edilebilir (Xu *et al.* 2002).

Ajan eskiden iyi olduğu kanıtlanmış davranışları mı gerçekleştirmelidir yoksa keşfedip gelecekte daha büyük ödüller almak uğruna şimdi düşük bir ödüle mi razı olmalıdır? Bu probleme keşif-istismar *(exploration-exploitation)* denmektedir (Bloembergen 2010).

Pekiştirmeli öğrenmede ödül toplamına dair üç ana model vardır; bunlar, finite-horizon model, infinite-horizon discounted model ve average-reward modeldir. Finite-horizon modelinde herhangi bir zamanda ajan beklenen ödülü belirlenmiş geçici bir aralıkta optimize etmelidir. Sonraki n adım, n ufuk uzunluğu, sonlu ufuğu tanımlamakta kullanılabilir. Bu model ajanın aktivite-zaman uzunluğu önceden biliniyorsa kullanışlıdır (Faustino 2011).

$$R_t = r_{t+1} + r_{t+2} + \cdots + r_{t+n} = \sum_{i=0}^{n} r_{t+i} \tag{2.1}$$

Average-reward modelinde uzun dönem ortalamasının maksimize edilmesi hedeflenir. Bu modelin problemi, başlangıçta daha kazançlı olan poliçelerle sonlarda daha kazançlı olan poliçeler arasında ayrım yapma zorluğudur (Faustino 2011).

$$R_t = \lim_{n\to\infty}\left(\frac{1}{n}\sum_{i=1}^{n} r_{t+i}\right) \quad (2.2)$$

$$R_t = r_{t+1} + \gamma * r_{t+2} + \gamma^2 * r_{t+3} + \gamma * r_{t+4} + \ldots = \sum_{i=1}^{\infty} \gamma^{i-1} * r_{t+i} \quad (2.3)$$

Infinite-horizon discounted modelinde uzun dönem ödülü önemlidir, gelecekte elde edilen ödüller geometrik olarak hesaptan düşülür. Bu model ajan aktivitesinde sonlu bir zaman ufku empoze etmez (Faustino 2011).

Çevre hakkında önceden bilgisi olmayan ajanların çevreyle etkileşip deneme yanılma yöntemiyle eğitilmesidir (Faustino 2011).

2.1.1. Markov karar süreci

Olasılıksal bir çevrede, sonraki durumların şartlı olasılık dağılımları yalnızca şimdiki duruma bağlıysa ve geçmiş durumlara bağlı değilse buna Markov özelliği denir. Markov özelliğini sağlayan pekiştirmeli öğrenme görevine Markov karar süreci denir. Pekiştirmeli öğrenme probleminin Markov çevresinde gerçekleşeni Markov karar sürecidir. İlk olarak Bellman tarafından tanımlanmıştır ve dinamik programlamada yoğun olarak çalışılır. Durum ve aksiyon sonluysa, Markov karar süreci olasılığı şöyle tanımlanabilir (Nissen 2007; Hwang *et al.* 2013);

$$P^{a}_{ss'} = \Pr\{s_{t+1} = s' \mid s_t = s, a_t = a\} \quad (2.4)$$

$$R^{a}_{ss'} = E\{r_{t+1} \mid s_t = s, a_t = a, s_{t+1} = s'\} \quad (2.5)$$

2.1.2. Markov özelliği

Pekiştirmeli öğrenme modeli Markov özelliğiyle formalize edilebilir. Bir ajan t zamanındayken s durum bilgisi verildiğinde, farklı aksiyonlar seçmenin sonuçlarını tahmin için bu bilgiyi kullanmalıdır. Eğer bir aksiyonu seçmenin çıktısı sadece şimdiki

duruma bağlıysa ve önceki durumlara, aksiyonlara, ödüllere bağlı değilse Markov özelliğindedir. Eğer çevredeki bütün durumlar bu özelliğe sahipse, çevre Markov özelliğine sahiptir (Nissen 2007).

$$\Pr\{s_{t+1} = s', r_{t+1} = r \mid s_t, a_t\} = \Pr\{s_{t+1} = s', r_{t+1} = r \mid s_t \ldots s_0, a_t \ldots a_0, r_t \ldots r_0\} \quad (2.6)$$

Markov özelliği bütün pekiştirmeli öğrenme metodları için hayati özelliktir, çünkü ajana verilen tek bilgi şu anki durumdur ve bu temelde bir aksiyon seçip öğrenmesi beklenir. Çoğu gerçek yaşam çevreleri %100 Markov olmayacaktır, ama Markov'a yakın tahminler olacaktır. Örneğin, tavla oyunu Markov özelliğine sahip değildir, çünkü önceki durumlar çözüme katkı sağlayabilir. Ancak, tavla tahtasının şimdiki durumu Markov durumuna iyi bir yaklaşımdır ve iyi bir tavla oynayan ajan pekiştirmeli öğrenme metodları kullanılarak yapılabilir. Eğer problemin iyi bir Markov tahmini yoksa ajanların sonuç bulması zorlaşır (Nissen 2007).

2.1.3. Ödül fonksiyonu

Kısa dönemli ödülü tanımlamak kolaydır, çünkü bir sonraki adımda elde edilecek örnektir, poliçe sadece kısa dönemli ödülü maksimize etmeye ihtiyaç duyar. Daha sonra bu anlık ödülü maksimize eder. Eğer poliçe uzun dönemli ödülü optimize etmeyle ilgileniyorsa bütün gelecek ödüller optimize edilmelidir (Nissen 2007).

$$R_t = r_{t+1} \quad (2.7)$$

$$R_t = \sum_{k=0}^{T} r_t + k + 1 \quad (2.8)$$

T, zaman adımlarının sayısındaki üst sınırı belirler. Eğer üst sınır yoksa t sonsuz olur ve r de sonsuz olabilir. Bu ödül sadece periyodik görevlerde ne kadar zaman adımı kaldığını bildiğinde anlam kazanır. Eğer zaman-adım sayısı önceden bilinmiyorsa bu yaklaşım sorunludur, çünkü gelecekteki t+1 adımda ödül görülemeyebilir ve eğer hedef

bir adımda erişilirse aynı ödül (r) değerini alır, sanki hedefe t adımda erişilmiş gibi olur. Eğer poliçe kısa ve uzun dönem ödüllerini optimize etmeyle ilgiliyse gelecekteki ödülleri hesaptan düşmeyle yapılabilir (Nissen 2007);

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_t + k + 1 \tag{2.9}$$

2.1.4. Değer fonksiyonu

Devam ettirilmemiş gelecek ödülü en genel r hesaplamasıdır. Optimal poliçe durumlar için daha yüksek bir ödül olmadığı yerde tanımlanır. Bütün s durumları için, p poliçesini izleyen yarıda bırakılmış gelecek ödül optimize edilmelidir. Bir s durumunda p poliçesini takip ederek elde edilen devam ettirilmemiş gelecek ödüle değer fonksiyonu denir (Nissen 2007).

$$V^{\pi}(s) = E_{\pi}\{R_t \mid s_t = s\} = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_t + k + 1 \mid s_t = s\right\} \tag{2.10}$$

Bu değer fonksiyonu durum-değer fonksiyonu adını alır, çünkü beklenen gelecek ödülü spesifik bir durumdan verir. Ajanın bir s durumunda iken bir a aksiyonu seçmesinin geri dönüşünü bilmek de önemlidir. Bu değer fonksiyonu, aksiyon-değer fonksiyonu Q(s,a) olarak yönlendirilir. Aksiyon-değer fonksiyonu şu şekilde tanımlanabilir (Nissen 2007).

$$Q^{\pi}(s,a) = E_{\pi}\{R_t \mid s_t = s, a_t = a\} = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_t + k + 1 \mid s_t = s, a_t = a\right\} \tag{2.11}$$

İki değer fonksiyonunun önemli bir özelliği optimal bir altyapıya sahip olmasıdır, yani optimal çözüm yerel olarak hesaplanıp sonradan birleştirilebilir. Optimal altyapı dinamik programlamanın ve aksiyon-değer fonksiyonunun temelidir. Şu şekilde tanımlanabilir (Nissen 2007).

$$Q^{\pi}(s,a) = E_{\pi}\{\sum_{k=0}^{\infty}\gamma^{k}r_{t} + k + 1 \mid s_{t} = s, a_{t} = a\}$$
$$E_{\pi} * \{r_{t+1}\gamma\sum_{k=0}^{\infty}\gamma^{k}r_{t} + k + 2 \mid s_{t} = s, a_{t} = a\} \quad (2.12)$$

2.1.5. Optimal poliçenin (π^*) tanımı

Optimal state-value fonksiyonu ve optimal action-value fonksiyonu optimal poliçe tanımlamak için kullanılabilir. Optimal state-value fonksiyonu şöyle tanımlanır:

$$V^{*}(s) = \max_{\pi} V^{\pi}(s) \quad (2.13)$$

$$Q^{*}(s,a) = \max_{\pi} Q^{\pi}(s,a) \quad (2.14)$$

Optimal durum-değer fonksiyonu, optimal bir aksiyonun seçilmesinin sonucu olduğundan optimal durum-değer fonksiyonu aksiyon-değer fonksiyonunun bir fonksiyonu olarak da tanımlanabilir (Nissen 2007).

$$V^{*}(s) = \max_{\alpha \in A(s)} Q^{*}(s,a) \quad (2.15)$$

$\pi^*(s)$'e, en yüksek aksiyon-değer değerini veren a'dır denilerek $\pi^*(s)$, $Q^*(s,a)$'nın bir fonksiyonu olarak tanımlanabilir;

$$\pi^{*}(s) = \arg\max_{\alpha \in A(s)} Q^{*}(s,a) \quad (2.16)$$

$\pi^*(s)$ için bu tanımlama kendi içinde bir problem değildir ama $Q^*(s,a)$'yı kullanmaktadır, bu da $\pi^*(s)$'i hesaplamak için tek opsiyonun bütün olası değerleri π için denemek demektir. Bu da genellikle olası değildir. Bu sebeple Q için hesaplaması kolay daha yeni bir tanımlama gereklidir (Nissen 2007).

$$
\begin{aligned}
V^*(s) &= V^{\pi^*}(s) \\
&= \max_{\alpha \in A(s)} Q^{\pi^*}(s,a) \\
&= \max_{\alpha \in A(s)} E_{\pi^*}\{\textstyle\sum_{k=0}^{\infty} \gamma^k r_t + k + 1 \mid s_t = s, a_t = a\} \\
&= \max_{\alpha \in A(s)} E_{\pi^*}\{r_t + 1 + \gamma \textstyle\sum_{k=0}^{\infty} \gamma^k r_t + k + 2 \mid s_t = s, a_t = a\} \\
&= \max_{\alpha \in A(s)} E_{\pi^*}\{r_t + 1 + V^*(s_{t+1}) \mid s_t = s, a_t = a\} \\
&= \max_{\alpha \in A(s)} \textstyle\sum_{s' \in S} P^a_{ss'} [R^a_{ss'} + \gamma V^*(s')]
\end{aligned}
\tag{2.17}
$$

$$
\begin{aligned}
Q^{\pi^*}(s,a) &= E_{\pi^*}\{\textstyle\sum_{k=0}^{\infty} \gamma^k r_t + k + 1 \mid s_t = s, a_t = a\} \\
&= E_{\pi^*}\{r_{t+1} + \gamma \textstyle\sum_{k=0}^{\infty} \gamma^k r_t + k + 2 \mid s_t = s, a_t = a\} \\
&= E_{\pi^*}\{r_{t+1} + V^*(s_{t+1}) \mid s_t = s, a_t = a\} \\
&= E_{\pi^*}\left\{r_{t+1} + \gamma \max_{\alpha' \in A(s_{t+1})} Q^*(s_{t+1}, a') \,\middle|\, s_t = s, a_t = a\right\} \\
&= \textstyle\sum_{s \in S} P^a_{ss'} \left[R^a_{ss}{}' + \gamma \max_{a' \in A(s')} Q^*(s', a')\right] \\
&= \pi^*(s) = \operatorname*{argmax}_{\alpha \in A(s)} \textstyle\sum_{s' \in S} P^a_{ss'} \left[R^a_{ss'} + \gamma V^*(s')\right] \\
&= \pi^*(s) = \operatorname*{argmax}_{\alpha \in A(s)} \textstyle\sum_{s' \in S} P^a_{ss'} \left[R^a_{ss'} + \gamma \max_{a' \in A(s')} Q^*(s', a')\right]
\end{aligned}
\tag{2.18}
$$

2.1.6. $P^a_{ss'}$ ve $R^a_{ss'}$'dan optimal poliçe bulmak

(2.17) ve (2.18) denklemleri, π*(s) için Q*(s) veya V*(s)'in bilinmesini gerektirmektedir. π için bütün olası değerleri kullanılarak hesaplama yapılabilir ki bu çok pratik değildir. Bu konuda birkaç dinamik programlama algoritması geliştirilmiştir, bunu hızlandırmak için $P^a_{ss'}$ve$R^a_{ss'}$ bilindiği durumda Bellman'ın eşitlik denkleminin özyinelemeli tanımlarını kullanmaktadırlar. π*(s)'i bulmak için iki farklı dinamik programlama metodu bulunmaktadır, bunlar değer iterasyonu ve poliçe iterasyonudur (Nissen 2007).

2.1.7. Değer iterasyonu

Değer iterasyonu, optimal deterministik poliçeyi optimal değer fonksiyonunu hesaplayarak, sonra da optimal poliçeyi bularak bulur (Nissen 2007). Değer iterasyonu

algoritması **Ek 1.1**'de verilmiştir.

2.1.8. Poliçe iterasyonu

Değer iterasyonu v'yi bulduktan sonra poliçeyi bir kez güncellerken, poliçe iterasyonu her iterasyonuda poliçeyi günceller ve bu modifiye poliçeyi gelecek iterasyonda kullanır (Nissen 2007). Poliçe iterasyonu algoritması **Ek 1.2**'de verilmiştir.

2.1.9. Temporal-Difference öğrenmesi

Pekiştirmeli öğrenmedeki en büyük problem, geçici kredi atamasıdır. Şu anda seçilen bir aksiyon iyi midir, yoksa uzun dönem etkileri var mıdır? Gecikmeli ödülleri düşünürsek, sadece en sonunda kayda değer bir ödül değeri elde ediliyorsa, anlık aksiyonları nasıl sınıflandırabiliriz? Sutton bu noktada tahmin etmek için öğrenme kavramını geliştirmiştir. Temporal-Difference (TD), geçmiş deneyimi kullanarak tamamı bilinmeyen bir sistemde gelecek davranışları tahmin etmektir. Geleneksel tahmin öğrenmede, tahmin edilen değerle elde edilen sonuç arasındaki hata (fark) ilişkilidir. Temporal difference'da ise, tahmin edilen değerle bir sonraki başarılı bir biçimde tahmin edilen değer ilişkilidir. Bu yolla, zamanla tahminde bir değişim olduğunda öğrenme gerçekleşir (Faustino 2011).

Bu yolla gerçekleşen öğrenmenin geleneksel öğrenmeye göre iki avantajı vardır; birincisi, daha artan biçimdedir ve daha kolay hesaplanır. İkincisi, tecrübeden daha iyi yararlanır, daha hızlı ve iyi tahminler üretir. Ek olarak, daha az kaynak gereklidir, çünkü sonuca kadar olan tahmin değerlerinin tamamını depolamak gerekmez, sadece önceki tahmin tutulmalıdır, böylece modele gerek yoktur. Sonuç bir sonraki aşamada gelebileceği gibi birkaç adım sonra da gelebilir (Faustino 2011).

TD'ın ayırt edici özelliği, son çıktıyla tahminlerin toplam hata farkına bakmaktansa, başarılı tahminlerdeki değişime olan tepkisidir. TD'ın özyüklemeli olduğu söylenir, çünkü onun öğrenmesi, final bir çıktı beklemeye gerek kalmadan, diğer tahminler

temelinde bir tahmindir (Faustino 2011).

$$V(s_t) \leftarrow V(s_t) + \alpha[r_t + 1 + \gamma V(s_t + 1) - V(s_t)] \quad (2.18)$$

$$r_t + 1 + \gamma V(s_t + 1) \quad (2.19)$$

TD metodları öğrenilen bilgilerin modelsiz yolla optimal poliçe oluşturmak için nasıl kullanılacağını gösteren metodlardır. TD, iki farklı probleme bölünmüştür, bunlar tahmin ve kontroldür. Tahmin problemi s durumunda p poliçesinde düşülmüş gelecek ödülü tahmin etme problemidir. Yani v'nin durum-değer fonksiyonunu tahmin etmektir. Kontrol problemi ise p poliçesinde s durumunda bir a aksiyonu seçerken düşülmüş gelecek ödülü öğrenme problemidir, yani Q'nun aksiyon-değer değeridir (Nissen 2007). Temporal difference algoritması **Ek 1.3**'de verilmiştir.

2.1.10. Çevre modelleri olmadan öğrenme

Pekiştirmeli öğrenme, modelsiz ya da model bazlı algoritmalar kullanılarak uygulanabilir. Bunların farkı ise, optimal poliçe bulmak için çevre modellerinin bulunup bulunmamasıdır. Bir çevre modeli, ajanın davranışlarına vereceği tepkiyi tahmin etmesine yarayan herhangi bir şeydir. Modellerin olmaması ve modellerin bakımının olmaması, model bazlı algoritmalara nazaran hesaplama zamanını azaltır. Ancak ajan tarafından elde edilen veri oldukça etkisiz kullanılır ve iyi bir performans için yüksek tecrübe gerekir. Bunun sebebi durumların çok defa ziyaret edilmesi gerekliliğidir. Modelsiz algoritmalar az kaynak kullanan ya da hesaplamanın gerçek dünya tecrübelerinden daha maliyetli olduğu zamanlarda kullanışlıdır (Faustino 2011).

Modelsiz yaklaşım açgözlü bir yaklaşım olabilir, basitçe en yüksek beklenen ödülü olan aksiyonu seçerek mümkün mertebe en çok ödülü toplamaya çalışır. Bu yaklaşımın faydası, bütün toplanan bilgilerin anlık olarak hızlıca ödül toplamak için kullanılıp umutsuz çözümler üzerinde fazla zaman harcamamasıdır. Bu yaklaşım Sutton ve Barto tarafından pekiştirmeli öğrenme algoritmasının anahtar özelliklerinden biri olarak

görülür ve diğer yaklaşımlardan ayırır. Bu açgözlü yaklaşım genellikle bir ajanın sub-optimal çözüm bulmasına sebep olur, diğer optimal çözümleri bulmadan o çözüme yapışmak π*'yi bulma şansını azaltır (Nissen 2007).

2.1.10.a. Adaptif sezgisel kritik (ASK)

Hemen hemen bütün pekiştirmeli algoritmaları değer fonksiyonlarını tahmin etme temellidir, durum fonkisyonları (ya da durum-aksiyon çiftleri) bir durumda bulunmanın ajan için ne kadar iyi olduğunu tahmin eder. Ne kadar iyi sözü beklenen gelecek ödüllere bağlıdır. Değer fonksiyonunun bir örneği Adaptif Sezgisel Kritik'tir *(Adaptive Heuristic Critic).* Barto'nun Adaptive Critic Element ve Adaptive Selective Element iskeletinin bir soyutlamasıdır. Bir fonksiyon tahminleyicisi, poliçe iterasyonundaki lineer denklemleri çözerek hesaplamak yerine, değer fonksiyonu değerini tahmin etmede kullanılır. İskeletin temel blokları girdi ve çıktı arasındaki doğru ilişkiyi belirlemek için stokastik bir metod kullanan Associative Search Element ve gelecekteki ödül ve cezaya doğru tahmin vermeyi öğrenen Adapative Critic Element'tir (Faustino 2011).

Adaptif Sezgisel Kritik *(Adaptive Heuristic Critic)*, TD bazlı modelsiz öğrenmede ön plana çıkan ilk metoddur. Barto et al bu tekniği cart-pole problemini çözmekte kullanmıştır. Adaptif Sezgisel Kritik arkasındaki mantık, poliçe değerlendirme adımını bir online aksiyon-seçme stratejisiyle birleştirmektir, böylece modelsiz bir poliçe iterasyon mekanizması sunar (Faustino 2011).

Adaptif Sezgisel Kritik mimarisi iki ana element içerir; bir değerlendirme modülü ve bir poliçe modülü. İkincil elementler ise durum hissetme ve aksiyon seçimidir (Faustino 2011).

- **Durum hissetme:** Çevre bilgisi verildiğinde bu modül şu anki durumu hesaplar (Faustino 2011).

• **Değerlendirme modülü:** Her bir giriş durumu için değer fonksiyonu tahminini hesaplar. Beklenen maliyet ile şimdiki çıktıyı karşılaştırıp şu anki parametreleri güncelleyerek yapılır. Bu modül s(t+1) için aksiyon sağlamaz ama poliçe modülüne bu aksiyonu hesaplamak için bilgi sağlar (Faustino 2011).

• **Poliçe modülü:** İteratif olarak optimal bir aksiyon-poliçesi tahmin etmeye yarar. Bu modülün parametreleri şimdiki aksiyonu cesaretlendiren veya caydıran gradient metoduna göre güncellenir. Bu da güncellenen v'nin önceki zaman adımındaki v'den büyük mü küçük mü olduğuna bağlı olarak yapılır. Eğer yeni maliyet öncekinden az ise aksiyon cesaretlendirilmelidir, çünkü ziyaret edilen durumun beklenen maliyetini düşürür. Poliçe ağı güncellenmemelidir, çünkü onların değeri şimdiki tecrübeyle henüz bilinmemektedir (Faustino 2011).

• **Aksiyon seçimi:** Aksiyon poliçesinden işletmek için bir aksiyon seçer. Başlangıçta keşif için rastgele aksiyonlar stokastik olarak oluşturulur (Faustino 2011).

2.1.10.b. Q-öğrenmesi *(Q-learning)*

Watkins, optimal poliçeyi çevre modeli olmadan öğrenen bir metod sunmuştur. Bu metod basit bir yapıdaydı, aksiyon-değer fonksiyonu, Q, a aksiyonunu s durumunda işletmenin değerini saklayan, değer fonksiyonunun bir genişletilmişiydi. En iyi poliçeye tekabül eden aksiyon-değer Q* olarak gösterilir. Optimal poliçe doğrudan Q* aksiyon-değer fonksiyonundan her bir durum için optimal aksiyonu seçerek (en yüksek değerli olan) elde edilebilir (Faustino 2011).

$$\pi^*(\mathrm{s}) = \mathrm{argmax}_{\mathrm{a}} \mathrm{Q}^*(\mathrm{s},\mathrm{a}) \tag{2.20}$$

$$\mathrm{Q}(\mathrm{s},\mathrm{a}) \leftarrow (1-\alpha)\mathrm{Q}(\mathrm{s},\mathrm{a}) + \alpha(\mathrm{r} + \gamma \mathrm{max}_{\mathrm{a}'}\mathrm{Q}(\mathrm{s}',\mathrm{a}')) \tag{2.21}$$

Öğrenme oranı 0 ile 1 arasındadır. Bu oran, optimal değerlerin yakınsaması 1'e

yaklaştıkça hızlı, 0'a yaklaştıkça daha yavaş bir biçimde Q değerlerinin güncellenmesini etkiler.

Watkins göstermiştir ki, her bir aksiyon her bir durumda sonsuz defa işletildiğinde ve learning rate azaltıldığında, Q değerleri Q*'ya olasılık 1 olacak şekilde yakınsar ve optimal poliçe bulunmuş olur. Q-öğrenmesi keşif duyarsızdır, yani optimale yakınsama garantisi vardır, keşif metodundan bağımsızdır (Faustino 2011). Q-öğrenmesi algoritması **Ek 1.4**'de verilmiştir.

Q-öğrenmesi, TD fikri üzerine kurulu off-policy bir pekiştirmeli öğrenme algoritmasıdır. Poliçe bazlı TD'dan farkı, gerçek aksiyon-değer fonksiyonunu poliçeden bağımsız tahmin etmesidir (Sutton and Barto, 1998). Q-öğrenmesi'in standart formu, Q'yu rafine etmek için her zaman adımında $Q_{a_i}(t+1) \leftarrow Q_{a_i}(t) + \alpha[r_i(t+1) + \gamma \max_j Q_{a_j}(t) - Q_{a_i}(t)]$ aksiyon-değer güncelleme fonksiyonunu kullanır. a_i, t zamanda yapılan hareket, α adım büyüklüğü parametresi, γ discount faktörüdür. Sadece seçilen aksiyonun değeri güncellenir, diğer bütün aksiyonlar için $a_j, Q_{a_j}(t+1) \leftarrow Q_{a_j}(t)$ (Bloembergen 2010). Bu güncelleme sürecine poliçenin rolü yoktur, sadece hangi aksiyonun seçileceğini belirlemekte kullanılır. Aksiyon-değer güncellemesi sadece ödüle ve sonraki iterasyonun beklenen değerine bağlıdır. Q'nun her bir güncellemesinden sonra, yeni optimal poliçe, aksiyon-değer fonksiyon Q'yu olasılık dağıtımı x'e çeviren seçme mekanizmasıyla belirlenir. Sıklıkla kullanılan iki aksiyon seçme mekanizması vardır, bunlar Q-Greedy ve Boltzmann keşifleridir. Q-Greedy, olasılık (1-ϱ) iken, en iyi aksiyonu (en yüksek Q değeri) seçer, olasılık ϱ iken Q değerlerinden bağımsız rastgele bir değer seçer. Boltzmann ise keşif-istismar arasında kontrol sağlayan bir t parametresi ile çalışır (Bloembergen 2010).

2.1.11. Çevre modelleriyle öğrenme

Modelsiz algoritmalar iyi performans göstermek için epey tecrübeye ihtiyaç duyar. Bu, bütün durumlara birçok defa uğranması gerektiğinden kaynaklanır. Model bazlı

algoritmalar bu kısıtlamayı ortadan kaldırmak için bazı durumları daha fazla ziyaret etmesine olanak tanıyıp yakınsamayı hızlandırmıştır. Optimal poliçeye bir geçiş modeli ve ödül modeli kullanarak ulaşır. Geçiş modeli, s durumundan s' geçiş bilgisini, s durumundan a aksiyonunu işleterek tutar. Ödül modeli s durumundaki a aksiyonunun ödül bilgisini tutar (Faustino 2011).

Başlangıçta ajan çevre hakkında bilgi sahibi değildir, dolayısıyla modeller boştur, ama devam ettikçe bilgi artar (Faustino 2011).

Model-bazlı algoritmalar her adımda gerçek tecrübeyle beraber simüle edilmiş tecrübe serisi işletir. Gerçek tecrübe modelleri güncellemede kullanılır. Daha sonra bu modeller çoklu simülasyon deneylerinde kullanılır. Model-bazlı algoritmalar kaynak sıkıntısı olmayan, hesaplamanın daha ucuz olduğu durumlara uygundur (Faustino 2011).

Model bazlı algoritma çevreyi keşfederek P_{ss}^{a}, ve R_{ss}^{a},'yı tahmin etmeye çalışacaktır ve tahminler yakınsadığında, bir dinamik programlama algoritması optimal poliçeyi bulmak için kullanılacaktır. Bu algoritma mümkün olduğunca çok keşif yapmalıdır, rastgele bir aksiyon seçebilir, en az denenmiş bir aksiyonu seçebilir, ya da daha yönlendirilmiş bir keşfetme poliçesi kullanabilir (Nissen 2007).

Bu yaklaşımın bazı problemleri vardır:

- Algoritma çevreyi keşfederken sadece keşfetmeye çalışacak, elde ettiği bilgileri değerlendirmeye çalışmayacak, dolayısıyla çok az ödül elde edilecektir. Gerçek hayat pekiştirmeli öğrenme problemlerinin çoğunda bu tecrübeyi edinmek için bir tür maliyet vardır, öğrenme aşamasının başında da olsa iyi bir ödül almak istenir (Nissen 2007).
- Eğer durum-aksiyon uzayı çok büyükse keşfetme stratejisi hepsini keşfetmeye çalışacaktır, bunu makul bir zamanda yapmak mümkün olmayabilir (Nissen 2007).
- P_{ss}^{a}, ve R_{ss}^{a},'yı tahmin ettikten sonra value ya da poliçe iterasyonunu çalıştırmak zaman alıcıdır, çünkü bütün durum-aksiyon çiftlerini keşfetmelidir (Nissen 2007).

2.1.12. Keşif-istismar

Modelli ve modelsiz metodların keşif ve istismar arasında denge kurabilmesi için bir tür aksiyon seçme stratejisine ihtiyacı vardır. Problem, doğru bir strateji bulmadadır. Bu ikisinin kararını vermek zordur, çünkü bazı problemlerde tamamen keşif veya tamamen istismar uygun olabilir (Nissen 2007).

2.1.12.a. Q-Greedy seçimi

Keşfi açgözlü yaklaşıma dahil etmenin basit bir yöntemi, beklenen en yüksek ödülü olan aksiyonu çoğunlukla seçip, geri kalan zamanda da rastgele bir aksiyon seçmektir. Q-greedy seçimi bu yaklaşımı benimser ve $1-\varrho$ zamanda açgözlü aksiyonu seçer ve geri kalan zamanda $0 \leq \varrho \leq 1$ olduğunda rastgele bir aksiyon seçer (Nissen 2007). Problemleri ise şöyledir:

- Bir zamanda optimal sonuç bulursa eğer hala random aksiyon seçmeye devam edecektir, optimal poliçeyi bulsa da takip etmeyecektir (Nissen 2007).
- Rastgele bir aksiyon seçildiğinde yüksek Q(s,a) değerine sahip bir aksiyon olabileceği gibi düşük Q(s,a) değerine de sahip olabilir. Daha önce birçok defa denenmiş bir aksiyon da seçilebilir, hiç denenmemiş olan da seçilebilir (Nissen 2007).
- Optimal yol ajanın yüksek derecede negatif ödül içeren bir alana girmesini gerektiriyorsa optimal poliçeyi bulmak greedy olmayan birçok aksiyon gerektirir. Q-greedy seçimi bunu yapmadığından optimal poliçeyi bulmak uzun zaman alır (Nissen 2007).

İlk problem Q'yu, ikinci problem rastgele seçinden öte bir yöntemi gerektirir (Nissen 2007).

2.1.12.b. Boltzmann-Gibbs seçimi

Sonraki Q(s,a) değerlerine bakmadan rastgele bir aksiyon seçmenin aksine, Boltzmann-Gibbs dağılımı Q'ları kişisel aksiyonlar için kullanır, aksiyonları seçmenin olasılığını hesaplamak için;

$$P(s \mid a) = \frac{e^{Q(s,a)/\tau}}{\Sigma_{a \in A(s)} e^{Q(s,a)/\tau}} \tag{2.21}$$

formülü kullanılır. P (s|a), s durumunda a aksiyonunu seçme olasılığıdır. τ derece değişkenidir, sıfırdan büyüktür. Düşük bir derece seçimin daha açgözlü olacağını, yüksek bir derece daha rastgele olacağını gösterir. Bu metodun en büyük avantajı kişisel Q'ların aksiyon seçerken dikkate alınmasıdır. Bu da Q(s,a) değerleri benzer olduğunda daha çok keşif yapılacağını, yüksek bir Q değerine sahip bir aksiyon olduğunda daha az keşif yapılacağını gösterir (Nissen 2007).

Bu yaklaşımın bir dezavantajı vardır, çoğu pekiştirmeli öğrenme problemi büyük alanlara sahiptir, Q(s,a) değerleri de benzerdir. Bu gözlem, hedefe bir adım daha yakın olan aksiyon, hedeften bir adım uzaklaştıran aksiyondan sadece birazcık fazla Q(s,a) değerine sahiptir. Bu yüzden bu ikisini seçme olasılığı birbirine yakın olacaktır (Nissen 2007).

2.1.12.c. Max-Boltzmann seçimi

Bu yaklaşım Q-Greedy ile Boltzmann-Gibbs seçiminin birleşmesidir. Bu yaklaşım da ilk baştaki tecrübe temelinde durum-aksiyon çiftlerinin kıtlığı problemini yaşar. Bu kıtlığı aşmanın bir yolu τ ya da ϙ'i yumuşatmaktır, ancak bu çok çabuk yapılırsa problem hala ortada olacaktır (Nissen 2007).

2.1.12.d. Belirsizlik durumunda iyimserlik

Basit bir sezgiseldir, daha önce denenmemiş aksiyonlarla elde edilen ödüllerle ilgili oldukça iyimser inançlara sahiptir. Bu yaklaşım yaygın olarak kullanılır, çünkü basittir ve açgözlü poliçe ile keşifsel poliçe arasında bir kombinasyon sayılabilir. Başlangıçta keşifsel başlayıp istismarcı olarak biter. Bu strateji diğer stratejilerle de kombine edilebilir (Nissen 2007).

2.1.12.e. Yöneltilmiş keşfetme

Arkasındaki fikir basittir. Ödül-değer fonksiyonunu maksimize etmektense tecrübe-değer fonksiyonu maksimize edilir. Amacı stratejiyi değiştirip bilgisini istismar etmeye başlamadan önce durum-aksiyon uzayını mümkün olduğunca keşfetmektir. En basit keşfetme teknikleri açgözlü teknikleridir, her durumda açgözlü en yüksek tecrübe değerini seçer. S durumundayken a aksiyonunu seçmenin anlık kazanılan tecrübe değeri R^{exp}, beklenen düşülmüş gelecek tecrübe Q^{exp}'dır (Nissen 2007).

2.1.12.f. Frekans bazlı keşfetme

Bütün durum-aksiyon çiftlerini tecrübe-değer fonksiyonuyla tek biçimli olarak keşfetmeye çalışacaktır (Nissen 2007).

$$R^{exp}(s,a) = \frac{-C(s,a)}{K} \tag{2.22}$$

C yerel sayıcı, a'nın s'deyken kaç kez seçildiğini tutar, k da ölçeklendirme sabitidir. Bu strateji daima daha az ziyaret edilen durum-aksiyon çiftini keşfetmeye çalışır. Tecrübe değeri daima negatiftir, ziyaret edilmemiş bir durum-aksiyon çifti önceki ziyaret edilmiş durum-aksiyon çiftlerine oranla daima daha yüksek R^{exp} değerine sahip olacaktır. Bu yöntemin avantajı, daima daha az keşfedilmiş aksiyonu seçmesidir. Bu da durum-aksiyon uzayının tek biçimli keşfini sağlar (Nissen 2007).

2.1.12.g. Yenilik bazlı keşfetme

Uzun süredir ziyaret edilmemiş durum-aksiyon çiftlerini keşfetmeye çalışacaktır. Tecrübe-değer fonksiyonu şu şekildedir:

$$R^{exp}(s,a) = \frac{-t(s,a)}{K} \tag{2.23}$$

t en son (s,a)'nın ziyaret edildiği zaman ve K ölçeklendirme sabitidir. Sutton tarafından tanıtılmıştır. Daha gelişmiş bir fonkisyonla onun yaklaşımını Pchelkin kullanmıştır. Wiering ve Kolle, bunu t(s,a) yerine şimdiki t'ye bağlı bir keşif değer fonksiyon olarak tanımlar (Nissen 2007).

2.1.12.h. Hata bazlı keşfetme

Durum-aksiyon uzayında Q(s,a) tahminleri kuvvetli arttığı yerleri keşfetmeye çalışır. Burası ajanın poliçeyi geliştirmek için pozitif bilgi kazanabileceği bir alan olabilir (Nissen 2007).

$$R^{exp}(s,a) = Q_{t+1}(s_t, s_t) - Q_{t+1}(s_t, a_t) + K \tag{2.24}$$

2.1.12.i. R-Max keşfetme

Belirsizlik durumunda iyimserlik notasyonunu formalize eder, ziyaret edilmemiş durum-aksiyon çiftleri maksimum durum ödülüyle aynı ödülü verecektir, hepsi hedef bir duruma iletir mantığındadır. R-max sonunda t-step'i optimize eden keşfetme poliçesini hesaplamak için bu iyimser yaklaşım üzerine kurulmuş olan P_{ss}^{a}, ve R_{ss}^{a},'nin tahminlerini kullanır. Ya t-step poliçesi yüksek ödül verecektir, ya da yüksek öğrenme sağlayacaktır (Nissen 2007).

2.1.13. Seçim stratejilerini birleştirme

R-max, model-bazlı kombinasyonun bir örneğidir. Kearns ve Singh'in çalışmalarına dayanır. Eğer küçük problemleriniz varsa, başlangıçta keşif sonlara doğru istismar yapan bir kombinasyon yararlı olur. Daha büyük problemler için bu iyi olmayabilir, çünkü yüksek ödüllü durum-aksiyon çiftlerine ulaşmakta problem yaşanabilir, eğer bir tür istismarcı yaklaşım kullanmıyorsanız. Eğer oyun oynuyorsanız, sadece keşifsel anlayışla kazanmanız çok zordur. Hedefin büyük bir negatif ödülün arkasında olduğu bir problemde istismarcı yaklaşımlar zorlanacaktır (Nissen 2007).

2.1.14. Karşılaştırma

Bu algoritmalar iki ekstremlerdir, yani algoritma ya keşifseldir ya istismarcıdır. İki algoritma da optimal çözüm bulmak için zorlanır. Birincisi yeterince yönlendirilmemiştir, ikincisi çok fazla yerel ödüllere yönlendirilmiştir.

2.1.15. Öğrenme otomatı

Öğrenme otomatları Q-öğrenmesinden farklıdır, çünkü bir değer fonksiyonu tahmin etmezler ama doğrudan poliçe uzayında öğrenirler. Geleneksel olarak bu otomatlar sonlu bir aksiyon seti tahmin ederler. Daha özenli, parametrik, genelleştirilmiş ve devamsal aksiyon seti öğrenicileri bulunmaktadır (Bloembergen 2010).

$$\begin{gathered} x_i(t+1) \leftarrow x_i(t) + \alpha r_i(t+1)\big(1 - x_i(t)\big) - \beta\big(1 - r_i(t+1)\big)x_i(t) \\ \text{eğer } t \text{ zamanında seçilen aksiyon } a_i \text{ ise} \\ x_j(t+1) \leftarrow x_j(t) + \alpha r_i(t+1)\Big(x_j(t)\Big) + \beta\big(1 - r_i(t+1)\big)\big[(k-1)^{-1} - x_j(t)\big] \\ a_j \neq a_i \end{gathered} \tag{2.25}$$

2.1.16. Pişmanlık minimizasyonu

Pişmanlık minimizasyonu, poliçe uzayında doğrudan öğrenir. Ajan tüm zamanlarda

optimal olarak çalışmadığı için zarara uğrar. Bu genel olarak, çevrenin belirsizliğinden kaynaklanır, ajan başlangıçta çevreyi gözlemleyerek daha iyi bir poliçe bulmaya çalışır, dolayısıyla başlangıçta optimal değildir. Ajan, farklı davranmadığı için pişman olabilir. Ajan, her bir zaman-adımında bütün aksiyonlarının değerlerini bildiğinde, her bir hareket için en iyi harekete nispetle kaybını anlayıp olasılıkları ona göre güncelleyebilir (Bloembergen 2010).

2.1.17. Çoklu ajan

Çoklu ajan pekiştirmeli öğrenme problemi rekabetçi, işbirlikçi veya bunların kombinasyonu olabilir. Rekabetçi bir çevrede bir ajanın başarısı diğerinin başarısızlığı olabilir. İşbirlikçi bir çevrede ise ajanlar ortak bir hedef için birlikte çalışırlar. Bu da öğrenme problemini daha karmaşık hale getirir, çünkü bireylerin ödüllerini maksimize etme çabası genel çözüm için iyi olmayabilir. Keşif-istismar konusu da ayrıca burada daha belirgin hale gelir, keşif diğer ajanın varlığından dolayı sadece başlangıçta değil daha uzun süre önemli hale gelir (Bloembergen 2010).

2.1.18. Çoklu ajan ortamında tekil ajan

Tekil ajan algoritmaları çoklu ajan öğrenmesine uygulanabilir ama bazı sonuçları olacaktır. Ortak aksyion uzayı *(joint action space)* yaklaşımı ve bağımsız öğrenme şeklinde iki ekstrem yaklaşım seçilebilir. Joint action space yaklaşımında bir ajanın diğerleri üzerindeki etkisi açıkça modellenir. Öğrenme tüm ajanların ortak uzayında bütün aksiyon setleriyle gerçekleştirilir. Durum geçiş fonksiyonu her bir ajanın bir durum ve aksiyonunun bir durum seti üzerinden olasılık dağılımına doğrudan haritalanmasıdır. Bu şekilde, şimdiki durum geçmiş tüm durum bilgisini içerir ve Markov özellikler hala vardır. Bütünsel bir bilgi için özgür ve anlık iletişim gereklidir, genel manada uygulanabilir değildir ve dağıtık kontrol için alan bırakmaz. Diğer yaklaşımda bağımsız ajanlar kullanılır. Bu ajanlar diğerleriyle iletişim kurmazlar, yokmuş gibi davranırlar. Burada çevre statik değildir ve Markov özellikleri bulunmaz (Bloembergen 2010).

2.1.19. Hoşgörülü Q-öğrenmesi

Hoşgörülü *(lenient)* Q-öğrenmesi, işbirlikçi çoklu ajan çevreleri için oluşturulmuş bir algoritmadır. Çoklu ajanlar birlikte öğrendiklerinde, alt basamak çözümleri birleştirebilirler. Böylece bir ajan optimumu bulmakta zorlanmaz. Diğer ajanlara hoşgörü bazı düşük ödülleri görmezden gelip birkaç örneğin en yükseğini düşünmekle olur. Ajan, başlangıçta takım arkadaşlarına daha hoşgörülü yaklaşıp düşük ödülleri görmezden gelir, ama zaman geçtikçe daha kritik olur ve poliçesini devamlı günceller (Bloembergen 2010).

2.1.20. Tekil ajan öğrenmesi

Tek ajanlı pekiştirmeli öğrenmede çevre sabittir ve Markovian olarak kabul edilebilir, şu andaki durum geçmiş tüm durumlar kadar bilgi verebilir. Bu da öğrenme problemini kolaylaştırmaktadır. Değer tabanlı ve poliçe tabanlı iki türlü pekiştirmeli algoritması vardır. Değer tabanlı olanlar bir aksiyonun uzun dönem ödülünü alan bir aksiyon-değer Q fonksiyonu tahmin ederler ve bundan poliçe oluştururlar. Poliçe tabanlı olanlar ise poliçelerini doğrudan alınan ödüle göre güncellerler (Bloembergen 2010).

2.1.21. Dyna-Q

Q-öğrenmesinde, ajan durumları ziyaret ederek öğrenmeyi gerçekleştirir. Eğer uzun adımlar işletilirse ve sadece son adım manalı bir ödül sunulursa, önceki adımlar boşa geçmiş olacaktır. Dyna algoritması Sutton tarafından bu sorun düşünülerek ortaya atılmıştır. Bu yapı, planlamayla pekiştirmeli öğrenmeleri birleştirir ve gerçek dünya maliyetleri, planlama için gereken hesaplama maliyetinin yerini aldığında avantajlıdır. Q-öğrenmesine uygulandığı zaman Dyna-Q olarak adlandırılır (Faustino 2011).

Dyna-Q mimarisinde ajanın ekstra adımları yapabilmesi için bir iç model kullanılır, bu da Q değerlerinin daha çabuk yakınsamasını sağlar. Model, her bir gerçek adımdan sonra güncellenir. Genel olarak bu ajanın daha hızlı öğrenmesini sağlar (Faustino 2011).

Dyna-Q algoritması **Ek 1.5**'de verilmiştir.

$$Q(s,a) = \hat{R}(s,a,s') + \gamma \sum_{s' \in S} \hat{T}(s,a,s') \max_{a'} Q(s',a') \quad (2.26)$$

Deterministik bir çevrede, iyi bir sonuç alınacağı bilinen bir tahmini tekrarlamak, yüksek ihtimalle iyi bir sonuç üretecektir, iyi bir aksiyon işletmenin olasılığını güçlendirecektir. Öte yandan, çevre stokastik ise ve bir aksiyonu işletmenin belli bir duruma ulaştıracağına dair kesin bir olasılık varsa, tekrar tekrar bir tahmini tekrarlamak yanıltıcı olabilir, çünkü o durumda en etkili aksiyon olmasa dahi yapay olarak Q değerleri şişirilmektedir. Dyna-Q algoritması ajanın tecrübelerini kullanarak aynı anda bir internal çevre modeli oluşturur (Faustino 2011).

2.1.22. On-Policy SARSA öğrenmesi

Takip ettiği poliçeyi günceller. Açgözlü seçimler yerine p poliçesi a' seçer ve bunu Q'yu güncellemek ve sonraki adımı atmak için yapar (Nissen 2007).

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma Q(s',a') - Q(s,a)) \quad (2.27)$$

Q-öğrenmesine benzer fakat Q'yu atılacak sonraki adımın temelinde günceller. Adı aksiyon-değeri güncellemek için gereken değerlerden <s,a,r,s',a'> gelir. On-policy metodun avantajı, takip ettiği poliçeyi optimize edip optimal hale getirmesidir. Böylece öğrenme daha kolay gerçekleşir. Dezavantajı ise öğrenme safhasından sonraki aşamada optimize ettiği poliçeden farklı bir poliçe kullanmasıdır (Nissen 2007). One-Step SARSA öğrenmesi algoritması **Ek 1.6**'da verilmiştir.

2.1.23. Off-Policy ile On-Policy öğrenmenin karşılaştırması

Aralarındaki temel fark, on-policy algoritmaları keşif maliyetlerini ve Q(s,a) tahminlerini içerirken off-policy bunları içermez.

2.1.24. Q-SARSA

SARSA ve Q, benzer olmasına rağmen, farklı avantaj ve dezavantajları vardır. Bu ikisinin kombinasyonu lineer bir ağırlık parametresi kullanıp (0≤σ≤1), güncelleme kuralı 0 olduğunda Q-öğrenmesi, 1 olduğunda SARSA öğrenmesi olur (Nissen 2007).

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left(r + \gamma\left((1-\sigma)\max_{a'' \in A(s')} Q(s',a'') + \sigma Q(s',a')\right) - Q(s,a)\right) \quad (2.28)$$

2.1.25. Ayrıcalık izleri

R_t^{λ} hesaplanmak yerine, devamlı olarak her bir adımda tahmin edilebilir. Bunu yapmak için her bir durum-aksiyon çifti için bir ayrıcalık izi *(eligibility trace)* $e_t(s, a) \in \Re+$ kullanılır. Bir s durumunun ve a aksiyonunun ayrıcalık izi, şimdiki ödülün Q(s,a)'yı t zamanında ne kadar etkilemesi gerektiğinin göstergesidir. En son ziyaret edilen durumlar şimdiki ödülden daha etkilenmesi gerektiğinden, durum-aksiyon çiftinin ayrıcalık izi her adımda γλ ile bozulmaya uğratılır. γ discount oranıdır. Şimdi ziyaret edilen durumun ayrıcalık izi artırılır (Nissen 2007).

$$e_t(s,a) = \begin{cases} \gamma\lambda e_{t-1}(s,a) + 1 & \text{if}(s,a) = (s_t,a_t) \\ \gamma\lambda e_{t-1}(s,a) & \text{if}(s,a) \neq (s_t,a_t) \end{cases} \quad \forall s \in S, \forall a \in A(s) \quad (2.29)$$

2.2. Robotikte Pekiştirmeli Öğrenme Kullanımı

Robotikteki problemler genelde çok boyutlu, sürekli durumları ve aksiyonları olan şekilde gösterilirler. Robotikte gerçek durumun tamamen gözlemlenebilir ve gürültüsüz olduğunu düşünmek mümkün değildir. Çok farklı durumlar bile çok benzer olabilir ve sistem kesin olarak hangi durum olduğunu bilmeyecektir. Dolayısıyla pekiştirmeli öğrenmede robotlar genelde kısmen gözlemlenebilir olarak modellenir. Sistem bu yüzden filtreler kullanarak doğru durumu tahmin eder (Kober *et al.* 2013).

Fiziksel sistemlerdeki tecrübeler maliyetli ve tekrar üretilmesi zor tecrübelerdir. Buna rağmen, iş tamamen simülasyonlara bırakılamaz. Bütün pekiştirmeli öğrenme metotları robotik alanında uygun değildir. Birçok metodun (model temelli) zor problemleri olduğu görülmüştür. Robot öğrenme sistemleri genelde değer fonksiyonu temelli yaklaşımlardansa arama metotları kullanır (Kober *et al.* 2013).

Robotların birçok davranışı devamsal olduğu için, temsil edilecekleri çözünürlüğü düşünmemiz gerekmektedir. Robot üzerinde kontrolümüzün ne kadar hassas olacağını belirlememiz gerekir. Bunu ya ayrıksılaştırma ile ya da fonksiyon tahminleme kullanarak ve hangi zaman adımı kullanacağımızı belirleyerek yapabiliriz. Robotlarda durumların ve aksiyonların boyutları yüksek olduğundan boyutsallık belası *(curse of dimensionality)* ile karşılaşırız (Kober *et al.* 2013).

Bellman, ayrık çok boyutlu uzaylarda optimal kontrolü keşfettiğinde, durumların ve aksiyonların eksponansiyel bir artışıyla karşılaştı ve buna boyutsallık belası dedi. Boyutların sayısı arttıkça, verilerin ve hesaplanma sürelerinin de eksponansiyel arttığını gördü. Örneğin, her bir durum-uzaya ait boyutun 10 seviyeye ayrıksandığını düşünürsek, bir boyutlu uzay için 10 durumumuz olur, 3 boyutlu durum-uzay için 10^3 = 1000 tekil durumumuz olur ve n boyut için 10^n durumumuz olur (Kober *et al.* 2013).

Robotik sistemler bu çok boyutlu durum ve aksiyonlarla başetmek zorundadır, çünkü modern antromorfik robotların özgürlük derecesi çoktur. Pekiştirmeli öğrenme topluluğu bu problemle uzun yıllar başetmeye çalışmıştır, bunu da hesaplama soyutlamalarıyla, adaptif ayrıksılaştırmalarla ve fonksiyon tahminleme yaklaşımlarıyla yapmıştır (Kober *et al.* 2013).

Sıklıkla, erken öğrenme periyotlarındaki çevresel ayarlar tekrar gerçekleştirilemez. Dış faktörler her zaman net değildir, örneğin ışık sistemin görüş performansını nasıl etkileyecektir? Bu gibi problemler algoritmaların karşılaştırılmasını zorlaştırmaktadır. Çoğu robot öğrenme görevleri bir tür insan süpervizyonu gerektirmektedir, örneğin çubuk sabitlemede çubuğu tekrar robota yerleştirmek gibi. Robotik pekiştirmeli

öğrenmede genellikle gerçek dünyayla ilişkiyi sınırlamak hafıza kullanımını veya hesapsal karmaşıklığı sınırlamaktan daha önemli kabul edilir (Kober *et al.* 2013).

Pekiştirmeli öğrenme algoritmaları bilgisayarlarda uygulandıkça, fiziksel sistemlerdeki devamsal zamanın ayrıksılaştırılması kaçınılmaz olmuştur. Zamanın ayrıksılaştırılması da istenmeyen problemlere yol açabilir. Daha kritik olarak, hareket ettirmede iletişim gecikmeleri ve motorların, dişlilerin hareketlerinin ani olmaması gibi problemler vardır. Bu gecikmelerden dolayı hareketlerin etkileri ancak adımlar sonra gözlemlenebilir olabilir. Birçok tanım kümesinde, amaca ulaşıldığında ödülleri sağlamak doğaldır. Örneğin, bir masa tenisi robotu maçı kazandığında ödül verilebilir. Ancak bu ödülü robot çok az elde ediyor olabilir, öyle ki gerçek bir sistemde bu ödüle hiç ulaşamayabilir. Basit binary ödüllere dayanmak yerine, sıklıkla ortanca ödüller bulundurmak gereklidir. Buna ödül şekillendirme *(reward shaping)* denmektedir (Kober *et al.* 2013).

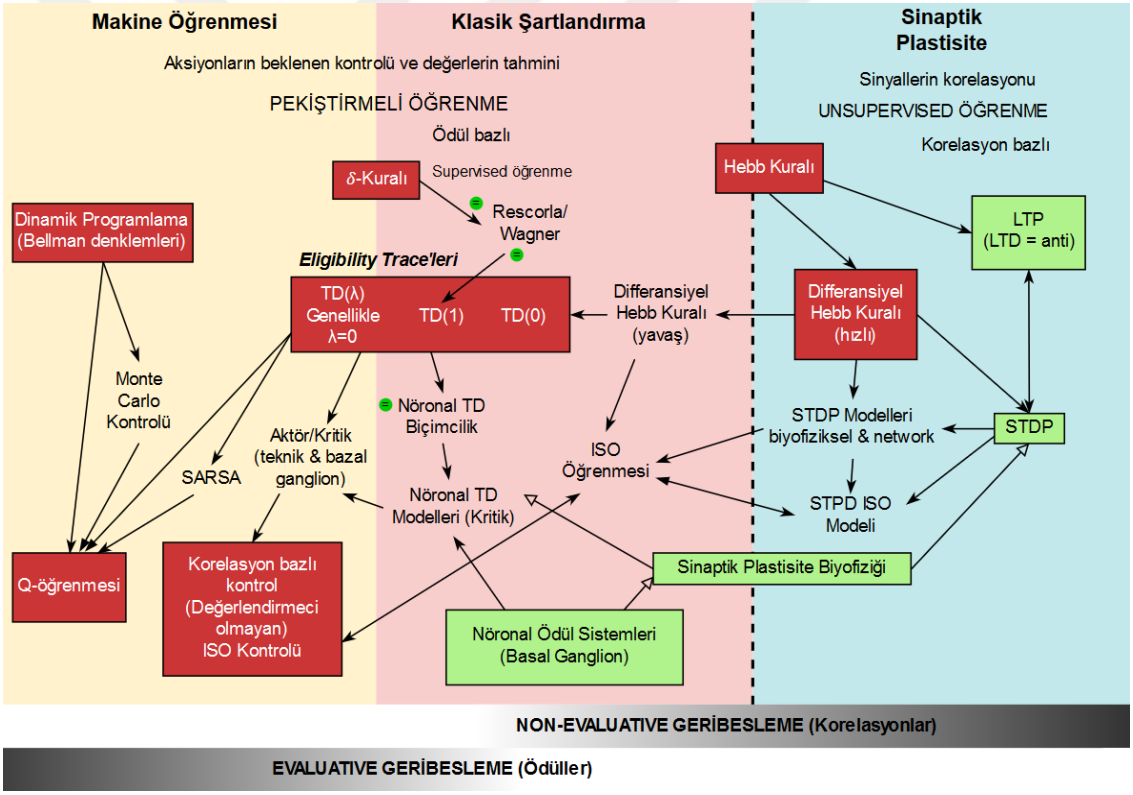
Bahsettiğimiz nedenlerden dolayı robotik alanında pekiştirmeli öğrenmenin başarısız olması muhtemeldir. Ancak önemli başarılar birkaç temel prensip ile elde edilebilir. Bunlar, efektif temsiller, doğru modeller ve önceden bilgi ve tecrübe aktarımıdır. Durumların ve aksiyonların boyut sayısını düşürmek, poliçe aramasını ve değer fonksiyonu temelli metodları iyileştirebilir (Kober *et al.* 2013).

2.3. Pekiştirmeli Öğrenmenin Literatürde Örnekleri

Pekiştirmeli öğrenme, temelde iki koldan gelişen araştırma alanlarının birleşmesiyle ortaya çıkmıştır. Bunlardan biri optimal kontrol alanı, diğeri ise deneme yanılma öğrenmesidir. Optimal kontrol problemleri, daha çok dinamik programlama metodları (Bellman denklemi) tarafından işaret edilmiştir. Deneme yanılma öğrenmeleri ise, psikoloji alanında köklere sahiptir. İlk akım, başından beri algoritmalar ve matematiksel yaklaşımlarla sürerken, ikincisinin matematiksel ifade edilmesi daha uzun sürmüştür. Bu iki alanın ortak çalışmalarından, Temporal Difference *(TD)* isimli, etkili ve ilham verici bir yöntem doğmuştur (Witten 1977; Sutton and Barto 1981). Watkins, 1989'da

TD bazlı Q-öğrenmesi'ni geliştirdi. TD, orjinali itibariyle hayvansal öğrenmeyle ilişkiliydi. Klopf'un çalışmalarıyla (1972, 1975, 1982, 1988), TD metodları hayvansal öğrenme teorileriyle birlikte kullanılmaya başlandı. Daha sonra optimal kontrol problemlerini çözmekte de kullanıldı. Pekiştirmeli öğrenmenin temelinin dayandığı alanlar Şekil 2.1'de gösterilmektedir (Anonymous 2016a).

Pekiştirmeli öğrenme, literatürde birçok alanda kullanılmakta olup, bu bölümde, robotik ve bilgisayar bilimleri alanlarında kullanımıyla ilgili, önemli birtakım çalışmalardan örnekler verilmiştir.



Şekil 2.3. Pekiştirmeli öğrenme bağlamı (Anonymous 2016a)

Samuel (1957), bugünkü Temporal Difference metodunun temeli olan, satranç oynamayı öğrenen bir program geliştirmiştir. Mahadevan ve Connel (1992), Obelix isimli bir robota kutuları ittirmeyi pekiştirmeli öğrenme ile öğretmeye çalışmıştır. Mataric (1994), pekiştirmeli öğrenme kullanarak robotlara tutma, bırakma, çarpışma

önleme gibi görevler öğretmeye çalışmıştır. Tesauro (1995), temporal difference yöntemiyle tavla oynayan program geliştirdi. Asada *et al.* (1996), bir robota topu kaleye götürerek gol atmasını gerçek zamanlı görüntüler kullanarak öğretmeye çalışmışlardır. Gaskett *et al.* (2000), Q-learning kullanarak, kamera bağlı bir robota etraftaki objelere çarpmamayı öğretmeye çalışmıştır. Paletta ve Pinz (2000), obje tanıma için pekiştirmeli öğrenme kullanmıştır. Porta ve Celaya (2001), robotun eklemlerinin hareketini ve yürüyüşünü oluturmak için pekiştirmeli öğrenme kullanmıştır. Bagnell ve Schneider (2001), pekiştirmeli öğrenme poliçe arama metodları kullanarak kendi kendine uçabilen bir helikopter üzerinde çalışmışlardır. Morimoto ve Do (2001), üç linkli bir robotun kendi kendine ayağı kalkıp durabilmesi için pekiştirmeli öğrenme kullanmışlardır. Rosenstein ve Barto (2004), çeşitli robotlar ve simülasyonlar kullanarak, gözetimli öğrenme ile pekiştirmeli öğrenmeyi birleştirerek kullanmıştır. Kohl ve Stone (2004), dört ayaklı Sony Aibo üzerinde robotun eklem hareketi ve yürüyüşü için pekiştirmeli öğrenme kullanmışlardır. Tedrake *et al.* (2005), iki ayaklı bir robotta bipedal yürümeyi pekiştirmeli öğrenme ile öğretmeye çalışmıştır. Marin ve Duckett (2005), Q-öğrenmesi'ne ek olarak yeni geliştirdikleri bir algoritmayı kullanarak kameralı ve gripperlı bir robotun objeye yanaşmasını sağlamıştır. Latzke *et al.* (2006), futbol oynayan robot üzerinde imitatif pekiştirmeli öğrenmeyi kullanmıştır. Hafner ve Riedmiller (2007), robotların tekerleklerinin hız kontrolü için, motorlar üzerine sinirsel pekiştirmeli öğrenme kullanmıştır. Birdwell *et al.* (2007), Q-öğrenmesi kullanarak radyo frekans ve ultrason sinyalleri yardımıyla Sony Aibo robotun bir doğrultuda yürümesini öğretmeye çalışmıştır. Peters ve Schaal (2008), robotlarda beceri öğrenimi için pekiştirmeli öğrenme kullanmıştır. Bir robota beyzbol sopasıyla gelen topa vurmayı öğretmeye çalışmıştır. Yasuda ve Ohkura (2008), birden çok robotun birlikte hareket ederek bir objeyi bir noktaya taşıması için pekiştirmeli öğrenme kullanmıştır. Kober ve Peters (2009), iple kupaya bağlı bir topu robot kol hareketi sağlayarak kupanın içine sokmayı pekiştirmeli öğrenme ile öğretmeye çalışmıştır. Rao *et al.* (2009), sanal makinaların konfigürasyonunu pekiştirmeli öğrenme ile yapmıştır. Hester *et al.* (2010), karar ağaçlarıyla birlikte pekiştirmeli öğrenmeyle robota penaltı atmayı öğretmeye çalışmıştır. Konidaris *et al.* (2011), pekiştirmeli öğrenme kullanarak bir robota, bir odadaki butona basıp kapıyı açma, bir anahtarı açıp odanın kapısını açma gibi görevler öğretmeye çalışmıştır. Buchli *et al.* (2011), değişken empedans kontrolü için

pekiştirmeli öğrenme kullanmış, bunu da simülasyon ve gerçek robotta uygulamıştır. Nemec and Ude (2011), iple bağlı bulunan bir topu sallandırarak robota bağlı bir kupanın içine düşürmeyi öğretmeye çalışmıştır. Kalakrishnan *et al.* (2011), dört ayaklı bir robota engebeli arazilerde hareket etmeyi öğretmeye çalışmıştır.

Literatürde çalışılan örneklerinden görüldüğü üzere, pekiştirmeli öğrenme, yaygın çalışılan, birçok duruma uyarlanabilen ve yönelimin kendisine doğru olduğu bir makine öğrenmesi metodudur.

2.4. Ters Sarkaç

Ters sarkaç *(inverted pendulum)*, klasik bir kontrol teoremi problemidir. Amaç, ters sarkacı kontrol edecek bir sistem tasarlamaktır. En sık olarak, sağa ve sola hareket edebilen bir mobil platform üzerine konumlandırılan bir çubuk ile sağlanır. Bazen, çubuk üzerine çeşitli ağırlıklar koyulabilir. Genellikle üç ana sistemden oluşur, bunlar, mekanik sistem, geribesleme ve kontroldür. Benchmark problemi olarak robotikte sıklıkla kullanılmaktadır. İlk olarak sismolojistler tarafından, 1844'te İngiltere'de sismometre yapımında kullanılmıştır. Bugün de roketlerde ve binaların antisismik çalışmalarında kullanılmaktadır (Kumar *et al.* 2013).

1950'li yıllardan beri ters sarkaç, stabil olmayan açık devre sistemleri stabilize etmek için lineer geribesleme kontrol teorisini öğretmekte kullanıldı. Bu probleme ilk çözüm 1960'ta Roberge tarafından bulundu. Daha sonra Schaefer ve Cannon tarafından 1966'da problemin çözümü geliştirildi (Boubaker 2012).

Ters sarkaç problemini bir elde avuç içinde sopayı dengede tutmaya benzetebiliriz. Dengede tutmak problemi, mobil platformun sınırlara ulaşmadan hareket etmesini ve bu esnada çubuğu düşürmeden dengede tutmasını sağlayacak bir yöntemin bulunmasını içerir (Kumar *et al.* 2013).

İlk kez model bağımsız aksiyon poliçe tahminine dayalı bir algoritmanın başarıyla

uygulandığı uygulamalardan biridir. 1983'te Barto ve Sutton tarafından gerçekleştirilmiştir (Faustino 2011).

Klasik algoritmada, ters sarkacı yöneten ajan, +f ve -f olmak üzere mobil platforma kuvvet uygulamaktadır. Uygulanan her aksiyon için, ajan bir ödül kazanmaktadır. Eğer çubuk düşmüşse ceza olarak ödül -1, dengede ise 0 olarak verilmektedir. Bu ödül modifiye edilebilir (Faustino 2011).

Ters sarkacın matematiksel modeli, Barto et al tarafından geliştirilmiştir. Bu model bugün birçok çalışmada temel olarak kabul edilmektedir (Faustino 2011).

Hougen, Fischer, Johnam tarafından kendi kendini organize eden yapay sinir ağları gerçek bir robotta uygulanmıştır. Widrow ve Smith tarafından kontrol teoretik teknikleri kullanılarak bir lineer kontrol modeli tasarlanmıştır. Lineer bir yapay sinir ağı (Adaline) bu kontrol ediciyi yeniden üretmek için eğitilmiştir. Bu çalışma Guez ve Selinsky tarafından genişletilmiştir, iki katmanlı sinir ağı kullanmışlardır. Michie ve Chambers, Boxes yaklaşımını kullanmışlardır. Sistemin durum uzayı, kutular adı verilen ayrık alanlara bölünüp problemi algoritma için daha kontrol edilebilir hale getirmiştir. Tecrübeyle ve problemin dinamikleriyle, algoritma her bir bölme için aksiyonları güncellemiştir. Barto, Sutton ve Anderson, Adaptive Heuristic Critic algoritması kullanarak sistemin ne zaman başarısız olacağını tahmin etmeye çalışmışlardır. Bununla birlikte, Boxes algoritmasını kullanmışlardır. Bu çalışma Anderson tarafından gerçek değerli girişlerin iki katmanlı sinir ağında kullanılmasıyla genişletilmiştir. İki ağ içeren bir pekiştirmeli öğrenme yaklaşımı kullanılmıştır, birincisi sistem durumunu tahmin etmek için durum değişkenleri kullanılarak eğitilen, diğeri de aksiyon-ağ olarak tahmin çıkışında, bir karar almak için ve araca uygulamak için eğitilendir. Bu yaklaşım “Temporal Difference” olarak adlandırıldı ve değerlendirme ağını eğitmek için kullanıldı. Daha sonra bu çalışma, Anderson tarafından rafine edildi. Dominic bu çalışmalarda sinir ağları mimarisini yeniden kullandı, ancak eğitim genetik algoritmayla yapıldı. Sonuçlar gösterdi ki, algoritma yaklaşık aynı denemelerde bir kontrol edici oluşmuş oldu, ancak %10 kadar daha başarılıydı. Fogel tekli bir dengeleyiciye kontrol

sistemi eğitmek için genetik bir algoritma uyguladı. Bu Barto'nun Adaptive Critic'ine benziyordu. Tolat ve Widrow sistemin şimdiki durumunu gösteren bir pixel resminden girdi alan bir kontroller geliştirdi. Bu yaklaşım piksel girdileriyle denetlenen bir ADALINE nöronu kullandı ve başarılı oldu. Troudet ve Merril sistem sinyallerinin gürültülü veya bozuk olduğu geriye yayılım sinir ağı kullandı. Ağ kontrolü çubuğu dengeleyip gürültüyü elemeyi başardı. Evrimsel algoritma yaklaşımları sinir ağlarıyla ve pekiştirmeli öğrenme ile birleştirilip bu problemin çözümünde farklı şekillerde kullanıldı. Wieland tekrarlı sinir ağlarının ağ ağırlıklarının evrilmesi için, tek çubukta, çift çubukta, değişik uzunluktaki çubuklarda genetik algoritmaları başarılı bir biçimde uyguladı. Genetik algoritma simülasyon zamanını maksimize etmek için, bir uygunluk *(fitness)* fonksiyonu kullandı, böylece evrimleşmiş bir ağ, kontrol edici çubuğu dengeleyebilirdi. Dieckmann ve Pasemann da tekrarlayan sinir ağlarını evriltip kullanmak için başarıyla genetik algoritmayı uyguladı. Goldberg, öğrenme sınıflandırıcı sistem adında evrimsel bir teknik kullandı. Uygunluk fonksiyonu, aracı başlangıç pozisyonundan alıp pistin ortasına bıraktığı için ve çubuğu dengede tutarken aracı durdurabildiği için kontrol ediciyi ödüllendirdi. Koza tarafından bir başka ilginç kontroller tasarımı yapıldı. Koza ve Keane iki ve üç boyutlu versiyon için LISP ifadelerini kullanarak genetik programlamayla başarıya ulaştı. Bulanık mantık kontrol edicileri de hem genetik hem de sinirsel bulanık teknikleri kullanılarak bu problem için kullanıldı. Sammut, Boxes kullanarak görsel bir Java applet'i oluşturdu. Gomez ve Miikkulainen, dijital simülasyon videoları oluşturdu (Qamaar 2012).

Pendulum stabilizasyonu, yapay sinir ağları, pekiştirmeli öğrenme, LQR *(Linear Quadratic Regulation)*, PID *(Proportional-Integral-Derivative)* kontrol edici ile yapılabilir.

Sıklıkla kullanılan robotik benchmarklardan örnek vermek gerekirse, Acrobot, Pendubot, Furuta Pendulum, Reaction Wheel Pendulum, Bicycle, VTOL Aircraft, the Beam and Ball system, TORA verilebilir (Boubaker 2012).

3. MATERYAL ve YÖNTEM

3.1. Problem Tanımı

Bu bölümde, uyguladığımız üç problemin, *(ters sarkaç, top düşürmeme, top fırlatma)* matematiksel modelleri ve durum uzay modelleri anlatılmaktadır. Problemler şekilsel olarak gösterilmektedir.

3.1.1. Ters sarkaç

Ters sarkacın matematiksel modeli gösterilen şekildedir;

$$\ddot{\theta} = \frac{g\sin\theta+\cos\theta\left[\frac{-F-ml\dot{\theta}^2\sin\theta}{m_c+m}\right]}{l\left[\frac{4}{3}-\frac{m\cos^2\theta}{m_c+m}\right]} \tag{3.1}$$

$$\ddot{x} = \frac{F+ml[\dot{\theta}^2\sin\theta-\ddot{\theta}\cos\theta]}{m_c+m} \tag{3.2}$$

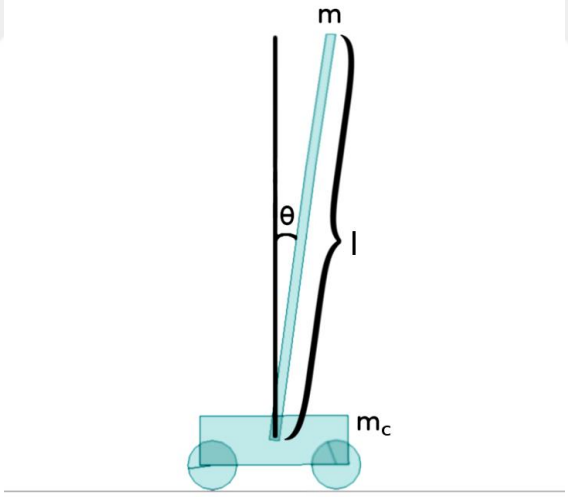
F, sağa ya da sola uygulanacak kuvvet, θ çubuğun normalle yaptığı açı, l çubuğun uzunluğunun yarısı, m_c aracın ağırlığı, m çubuğun ağırlığıdır. Burada amaç, araç hareket ederken çubuğun açısını minimuma yakın tutmak, mümkünse sıfır derecede dengede tutmaktır.

Durum-uzayı tanımlamak için, Boxes algoritması tercih edilmiştir. Boxes algoritması, durum-uzayı ayrık modellemeye yarayan, pekiştirmeli öğrenmenin ilk örneklerinden biridir. Her bir ayrık durum, kutu adını alır. Algoritma ismini burdan almaktadır. Ters sarkaç için tanımlanan sistem durum değişkenleri şu şekildedir;

$$S = [x, \dot{x}, \theta, \dot{\theta}] \tag{3.3}$$

x, aracın pozisyonu, ẋ aracın hızı, θ çubuğun açısı, θ̇ çubuğun açısal hızıdır. Sistem, durum değişkenlerini gözlemleyerek, bu değişkenlere göre ayrılmış ilgili kutucuğun alanına girdiğinde, duruma göre çalıştırılacak aksiyonu çalıştırır. Durumların ve aksiyonların seçildiği Boxes algoritması önceki bölümde verilmiştir. Durumlar Boxes algoritmasına anlık parametreler olarak gönderilmektedir. Buna karşılık algoritma, sistemin konumunu belirlemektedir.

Durum değişkenleri güncellenirken, 0.02 değerinde bir Tau (τ) değişkeni kullanılmıştır. Bu değişkenle çarpılan ẋ, x'e eşitlenmektedir. Aynı şekilde θ̇, bu değerle çarpılarak θ'ya eşitlenmektedir. (1) denklemi de çarpılıp, θ̇'ya eşitlenmektedir. Bu yaklaşım, ilk olarak Sutton ve Barto tarafından kullanılmış olup, bu çalışmada da Sutton ve Barto'nun kullandığı haliyle ters sarkaç için kullanılmıştır.



Şekil 3.1. Ters sarkaç

3.1.2. Top düşürmeme

3.1.2.a. Tek-Link

Tek link ile yapılan top düşürmeme, sabit bir duvara bağlı yine sabit bir kaldıracın

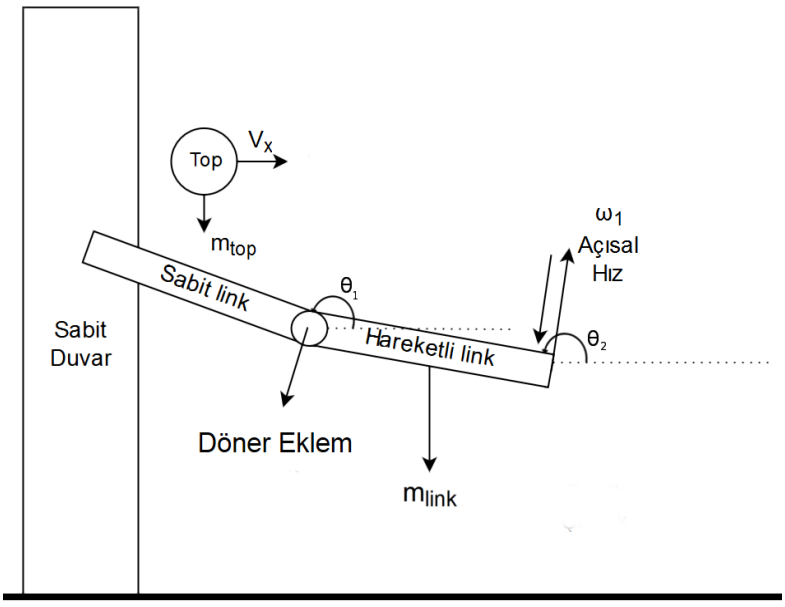
ucuna, hareketli bir kaldıraç şeklinde eklemlenmiş bir linkin, yer ile açılı olarak yerleştirilmesi ile üzerine bırakılan topu eğimden dolayı aşağıya düşmeden tutabilmesini sağlayan bir sistemdir. Durum değişkenleri olarak, topun pozisyonu, topun lineer hızı, linkin açısı ve linkin açısal hızı tanımlanmıştır.

$$\omega = \frac{V}{r} \tag{3.4}$$

Durum-uzayı temsil etmek için Boxes algoritması bu çalışmada kullanılmıştır. Durum-uzayı dört boyutlu vektör olarak olarak aşağıdaki gibi tanımlanmıştır;

$$S = [x, \dot{x}, \theta, \dot{\theta}] \tag{3.5}$$

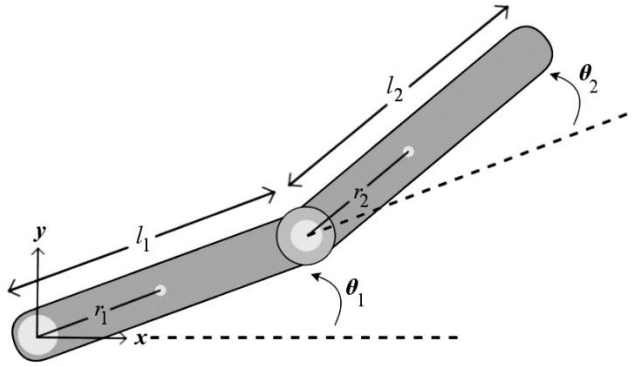
x topun pozisyonu, $\dot{x}$ topun lineer hızı, θ kaldıracın açısı, $\dot{\theta}$ kaldıracın açısal hızıdır. Algoritma 1'de durumları ve aksiyonları Boxes algoritmasına göre nasıl seçtiğimiz gösterilmektedir. Algoritmada, durumlar fonksiyona parametre olarak anlık bir biçimde gönderilmektedir. Algoritma da bu durumların değerlerine göre, sistemin konumunu box değişkenini değiştirerek belirlemektedir.



Şekil 3.2. Tek-Link top düşürmeme sistemi

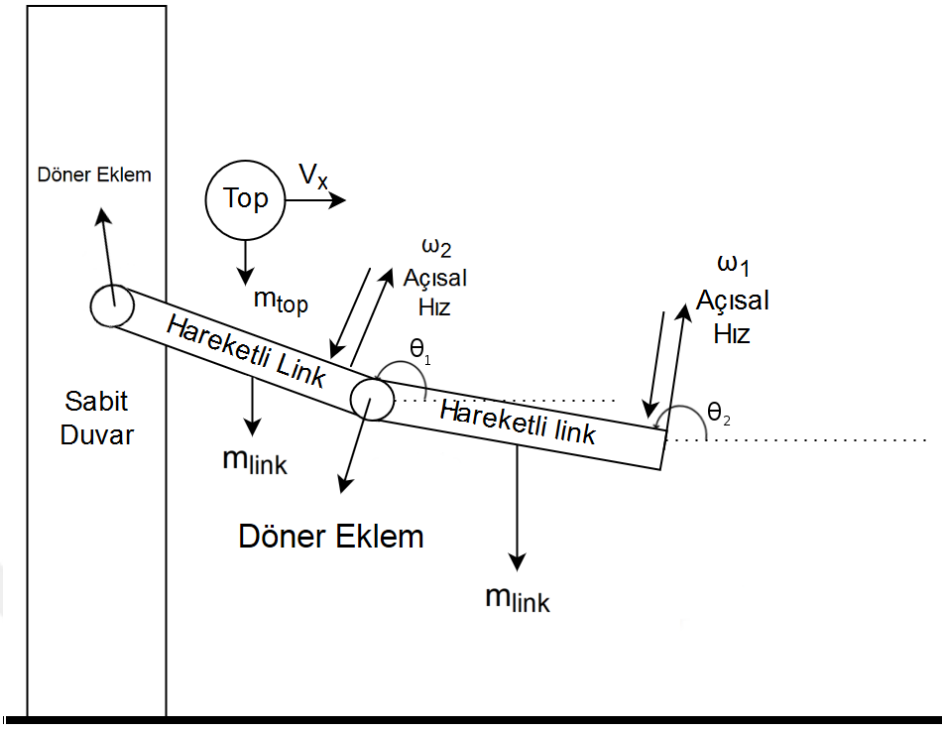
3.1.2.b. Çift Link

Çift linke sahip sistemde, sabit duvara bağlanmış bir hareketli link ve onun ucuna bağlanmış ikinci bir hareketli link bulunmaktadır. Bu linkler birbirine döner eklem ile bağlanmıştır. Bu yönüyle iki link planar manipülatöre benzetebiliriz. Linklerin kinematik çözümlemeleri simülatör tarafından yapılmaktadır.



Şekil 3.3. İki-Link planar manipülatör

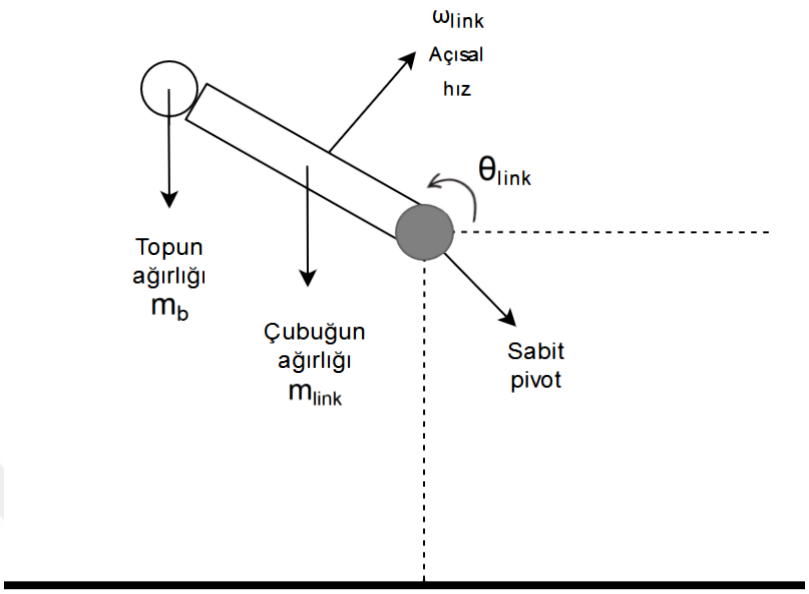
Her iki link için de ayrı Boxes algoritması, ayrı Q-öğrenmesi tabloları, ayrı Adaptif Sezgisel Kritik *(Adaptive Heuristic Critic)* vektörleri oluşturulmuştur. Kullanılan durum-uzay değişkenleri tek link ile aynıdır. Kullanılan Boxes algoritmaları metodoloji bölümünde gösterilmiştir. Sistemin kinematik hesaplamaları, Box2d simülatörü tarafından gerçekleştirilmektedir. Duvara bağlı link için uygulanan açısal hız, yukarı yönde 10 kat artırılıp, aşağı yönde 10 kat azaltılmıştır. İkinci link birinci linki de kaldırdığından, bu değerler seçilmiştir.



Şekil 3.4. Çift-Link top düşürmeme sistemi

3.1.3. Top fırlatma

Top fırlatma sistemi, sabit bir pivot noktası üzerine yerleştirilmiş bir linkin, devamlı olarak belirli bir açısal hızla dönerek, üzerinde ona bağlı olan bir topu en uzak mesafeye fırlatmayı öğrenmesi için tasarlanmıştır. Çubuk, saat yönünde dönmekte ve top sağ ya da sol tarafa fırlatılmaktadır. Sağ tarafa fırlatıldığında, mesafe pozitif olarak ölçülüp, sol tarafa fırlatıldığında ise mesafe negatif olarak kabul edilmektedir. Pivot noktası ise tam sıfırda konumlandırılmıştır.



Şekil 3.5. Top fırlatma sistemi

Bu sistemde, durum değişkenleri olarak, topun yataydaki x konumu, topun dikeydeki y konumu ve linkin açısı kullanılmaktadır.

$$S = [x, y, \theta] \tag{3.6}$$

Bu sistem için kullanılan Boxes algoritması metodoloji bölümünde gösterilmiştir. Topa uygulanan kuvvet, eğer sağa fırlatılacaksa Box2d fonksiyonu olan *linearImpulse* ile x yönünde 0.5 olarak seçilmiştir. Eğer top sola fırlatılacaksa kuvvet bunun negatifi, -0.5 olarak seçilmektedir. Topun düştüğü konum her seferinde kaydedilmektedir. Sistem, hiç kesintiye uğramadan topları fırlatmaktadır. Link, saat yönünde sabit bir açısal hızla devamlı dönmektedir. Fırlatılan bir önceki top yere düştüğü anda, sitem yenisini fırlatabilecek konuma getirilmektedir. Eğer atış başarısız ise, sistem topyekün olarak yeniden başlatılmamakta, sadece ödül olarak bu başarsızlık yansıtılmaktadır.

3.2. Metodoloji

3.2.1. Boxes algoritması

Boxes algoritması, dinamik kontrol sistemleri için etkili ve esnek bir metod olduğunu kanıtlamıştır. Algoritma, orjinal haliyle birçok kontrol probleminde benchmark olarak kullanılmaktadır. Algoritma adını durum-uzayı bölme şeklinden almaktadır. Durum-uzay, her bir boyutun uzunluğu kutulara ayırarak bölümlendirilir. Böylece bütün uzay dört parçaya bölünür. Sistemin durumunu bir kutu belirler. Her bir kutu, sistem o kutuya girdiğinde yapılacak olan programlanmış aksiyonu içerir. Kutunun işlevi, uzayın o bölümünde uygun olan aksiyonu öğrenmektir. Öğrenmeyi yapabilmek için, her bir kutu performansına ait istatistikler içerir. Bunlar;

- Her bir aksiyon kaç kez gerçekleştirilecektir?
- Bir aksiyondan sonra sistemin dengede olduğu sürelerin toplamı

Boxes algoritması başlangıçta rastgele bir ayarlama yapar. Bu ayarlarla sistemi kontrol etmek için bir deneme yapılır. Eğer başarısız olursa, kutulardaki ayarlar gözden geçirilir ve değiştirilir. Bu yeni ayarlarla yeni denemeler yapılır. Sistem belirli bir süre başarıyla kontrol altında tutulana kadar bu işlem tekrar eder (Michie and Chambers 1968; Sammut 1994).

Bir başarısızlık gerçekleştiğinde, her bir kutuda o kutuda bulunan aksiyonu “sağa ittir” veya “sola ittir” şeklinde belirlemek için bir karar verilir. Bu karar verilirken, keşif ve istismar arasında bir seçim yapılmalıdır. Bunun için aşağıdaki formülle karar verilir (Michie and Chambers 1968; Sammut 1994);

if valueL > valueR **then solu** seç
if valueL < valueR **then** sağı seç
if valueL = valueR **then** rastgele seçim yap

Bir aksiyonun değerinin hesaplanmasında ödünleşme hesaba katılmalıdır. Bu hesaplamada, Michie ve Chambers algoritması, Cribb's local merit, Laws algoritması gibi algoritmalar kullanılmaktadır. Tezde kullanılan Boxes algoritmaları, **Ek 2.1**, **Ek2.2**, **Ek 2.3**, **Ek 2.4**, **Ek 2.5**'de verilmiştir.

3.2.2. Adaptif sezgisel kritik algoritması

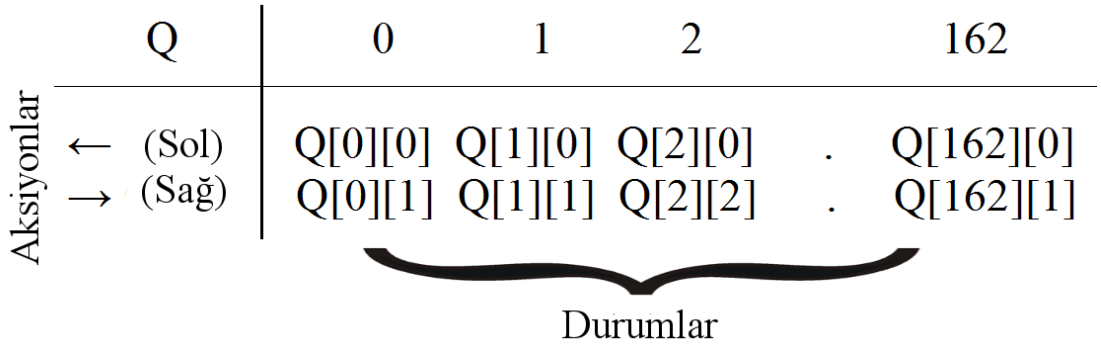
Adaptif Sezgisel Kritik algoritmasında, başlangıçta dört adet vektör bulunmaktadır. Bu vektörler, aksiyon ağırlıkları, kritik ağırlıkları, aksiyon ağırlıkları ayrıcalıkları ve kritik ağırlıkları ayrıcalıkları *(eligibility)* şeklindedir. Q-öğrenmesi'nin bu vektörler aksine başlangıçta sıfır değeriyle doludur. Algoritma çalıştıkça, bu vektörleri uygun değerlerle doldurarak, çubuğu dengede tutacak vektörleri oluşturur. Yine Q-öğrenmesi'nden farklı olarak, LambdaV ve LambdaW isimli iki değişken, ayrıcalık izleri *(eligibility trace)* için bozulma oranı *(decay rate)* olarak kullanılmaktadır. Bu tezde, kullanılan vektörlerin boyutu 162x1 şeklindedir.

	w	v	e	xbar
0	w[0]	v[0]	e[0]	xbar[0]
1	w[1]	v[1]	e[1]	xbar[1]
2	w[2]	v[2]	e[2]	xbar[2]
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
162	w[162]	v[162]	e[162]	xbar[162]
	Aksyion ağırlıkları	Kritik ağırlıkları	Aksiyon ağırlıkları üstünlükleri	Kritik ağırlıkları üstünlükleri

Şekil 3.6. Adaptif Sezgisel Kritik algoritması vektörleri

3.2.3. Q-öğrenmesi ve SARSA

Q-öğrenmesi algoritmasında, başlangıçta bir Q dizisi, rastgele değerlerle doldurulmaktadır. Algoritma bu dizideki rastgele değerlere göre bir aksiyon seçip çalışmaya başlamaktadır. Daha sonra öğrenme gerçekleştikçe bu dizideki değerleri güncellemektedir. SARSA algoritması da aynı şekilde çalışmakta olup, farklılık gösteren kısımı, aksiyon seçme ve öğrenme formülündedir. Bu tezde kullanılan Q dizileri, 162x2 boyutundadır.



Şekil 3.7. Q-öğrenmesi tablosu

4. ARAŞTIRMA BULGULARI ve TARTIŞMA

Bu bölümde, simülasyonda kullanılan simülatörden, her üç problem için kullanılan parametrelerden ve simülasyon sonuçlarından bahsedilmiştir.

4.1. Box2d Simülatörü

Box2d, ücretsiz bir iki boyutlu fizik simülatörü motorudur. Erin Catto tarafından C++'da yazılmıştır. Zlib lisansı altında yayınlanmıştır. Angry Birds gibi birçok popüler oyunlarda kullanılmıştır. İlk olarak Box2d Lite adında 2006'da sunulmuştur (Anonymous 2016b).

Box2d platform bağımsızdır. C++ derleyicisi olan herhangi bir sistemde çalışabilir. Nintendo DS, Wii, Android, iPhone gibi birçok sistemde kullanılmıştır (Anonymous 2016b).

Box2d Java, Flash, C#, Lua, Javascript gibi birçok farklı dile aktarılmıştır. Bazı özellikleri şu şekildedir (Anonymous 2016b);

- Çarpışma tanılama
- Konveks poligon ve daireler
- Her bir obje için birden fazla şekil
- Temas, friksiyon çözümlemeleri
- Revolute, prismatic, distance, pulley, gear, mouse joint tipleri
- Motorlar, impulse ve forcelar
- OpenGL

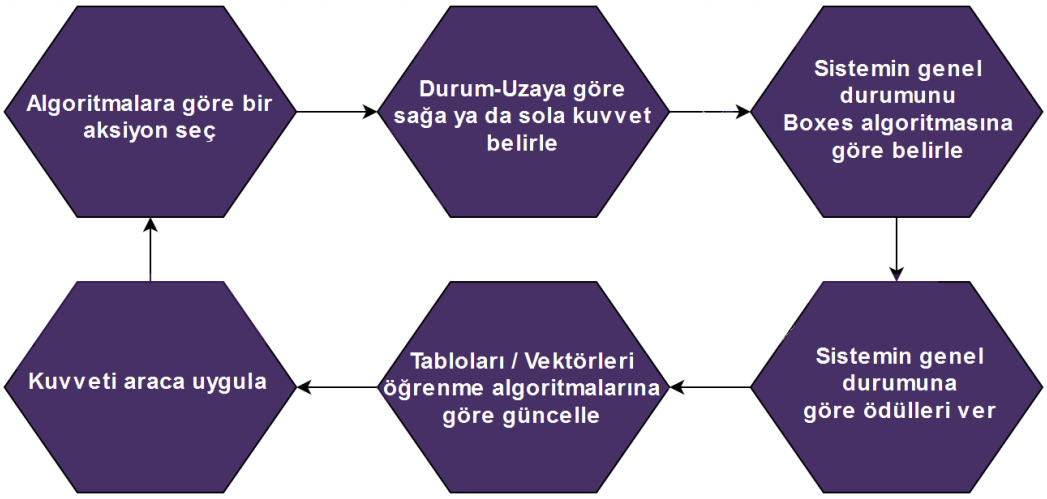
Box2d karmaşık fizik hesaplamaları gerektiren uygulamalar için oldukça uygun olup, bunları iki boyutlu olarak ekrana yansıtmakta başarılı bir kütüphanedir. Kolay ve anlaşılabilir yapısı, optimize edilmiş kodları ve hızıyla tercih edilebilir. Bu nedenle bu

tezdeki simülasyonlarda kullanılmıştır.

4.2. Ters Sarkaç Simülasyonu ve Sonuçları

Üç farklı pekiştirmeli öğrenme algoritması, Box2d adlı simülatör üzerinde çalıştırılıp, performansları incelenmiştir. Box2d, C++ ile yazılmış açık kaynak kodlu bir fizik motorudur. İki boyutlu fiziksel işlemleri hesaplayıp ekrana gerçekçi bir biçimde göstermeye yaramaktadır. Bazı bilimsel çalışmalarda ve birçok popüler mobil oyunda da kullanılmaktadır. Şekil 4.1'de sistemin çalışma şekli gösterilmiştir.

Ters sarkaç simülasyonunda kullanılan algoritmalar ve parametreler, temelde Sutton ve Anderson'ın (Anonymous 2015a, 2015b) yazdığı şeklinden örnek alınarak yazılmıştır. Simülasyonda kullanılan parametreler Çizelge 4.1'deki gibidir.



Şekil 4.1. Ters sarkaç sistemi akış şeması

Çizelge 4.1. Ters Sarkaç simülasyonunda kullanılan değerler

Çubuk ağırlığı (m)	0.1
Çubuk boyutları	0.02 x, 1 y
Araç (cart) ağırlığı (m_c)	1
Araç boyutları	0.3 x, 0.1 y
Alpha (Q-öğrenmesi) (α)	k/(k+n)
Alpha (ASK) (α)	k/(k+n)*1000
k	250
Beta (ASK) (β)	0.5
Gamma (γ)	0.9
Reward (r) (ASK) (failure)	-1
Reward (r) (ASK) (success)	0
Reward (r) (Q-öğrenmesi) (failure)	0.1
Reward (r) (Q-öğrenmesi) (success)	0.5
LambdaV (ASK)	0.8
LambdaW (ASK)	0.9
Kuvvet	10/-10
Velocity Iteration (Box2d)	8
Position Iteration (Box2d)	3
Hertz	60

$\dot{\theta}$ ve $\dot{x}$ durum parametreleri, (1) ve (2) denklemlerinde çubuğu dengelemekte kullanılmaktadır.

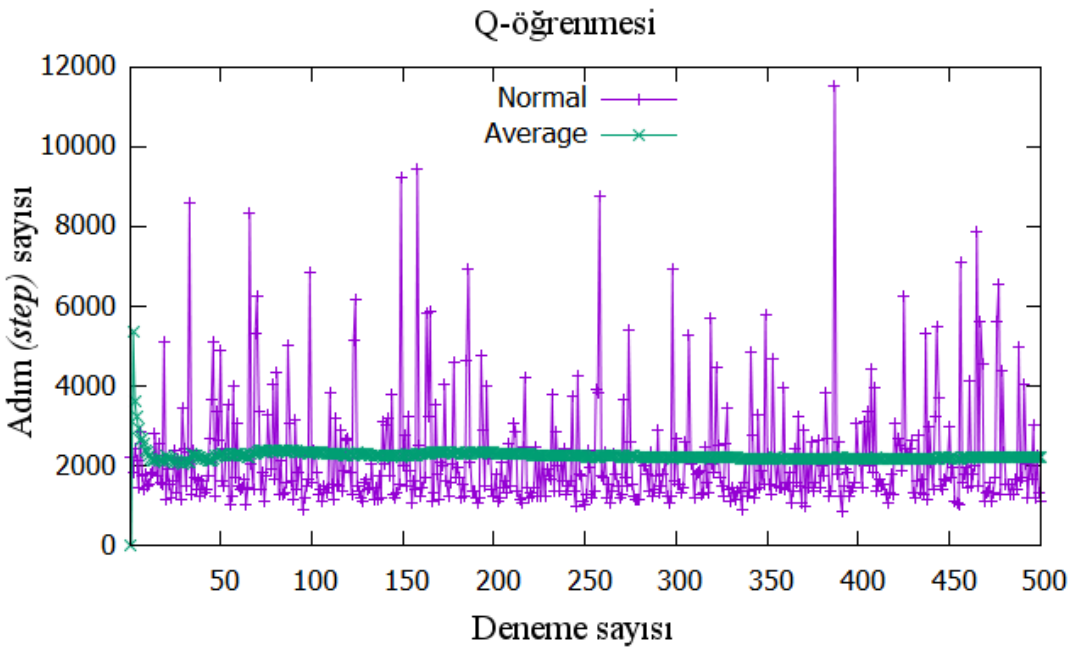
Araca uygulanacak F değeri 10 olarak seçilmiştir. 10 ya da -10 olarak, algoritmadan seçilerek, (3.7) denklemi ile hesaplanıp hedef ivmeye dönüştürülmektedir. (4.1) fonksiyonuyla bu değer, açısal hıza çevrilip, aracın tekerleklerine dönme kuvveti olarak gönderilmektedir. Her bir deneme için, araç rastgele bir yatay konumda başlatılmaktadır. Aracın ilk pozisyonunda, çubuk dengededir.

$$\omega = -(\text{hedefivme})/2 * \pi * \text{r} \tag{4.1}$$

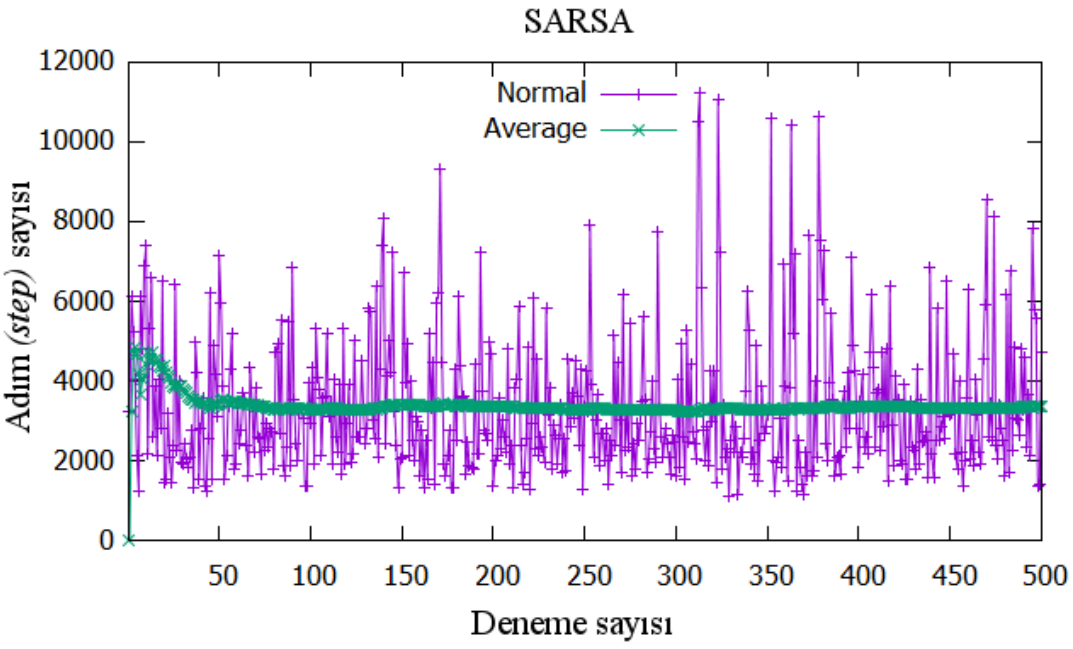
Algoritmalar için 500'er deneme yapılmıştır. Çubuğun düşürülmeden tutulduğu süre, her deneme için ayrı olacak şekilde adım *(step)* sayısı olarak hesaplatılmaktadır. 500 deneme bittiğinde, her denemede için adım *(step)* değeri, ortalama adım değeri, elde edilen Q, SARSA ve Adaptif Sezgisel Kritik dizi/vektörleri kaydedilmektedir. Denemeler sırasında, her adımdaki adım sayıları kümülatif olarak toplanıp, bulunulan deneme sayısına bölünerek ortalama adım sayısı hesaplanmaktadır.

Algoritmalarda kullanılan alpha değeri, k/(k+n) (Even-Dar and Mansour 2003) formülüne göre belirlenmektedir. Buradaki k değerimiz sabit olup, çalışmamızda deneme sayısının yarısı olarak, yani 250 olarak belirlenmiştir. N ise şimdiye kadarki deneme sayısıdır.

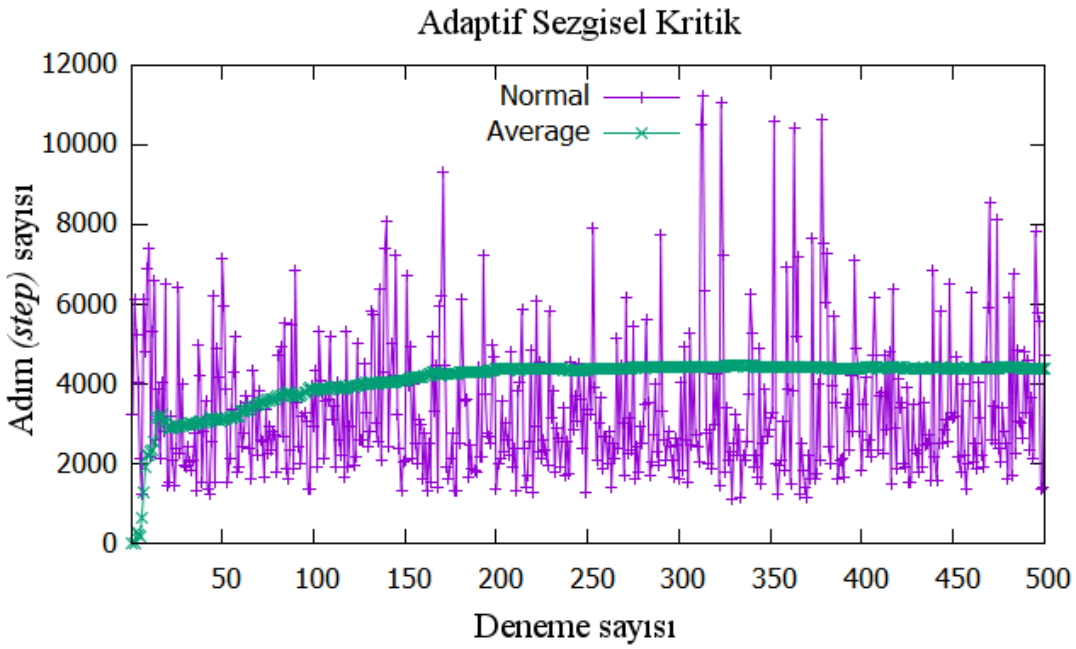
Şekil 4.2. Ters sarkaç simülasyon ekran görüntüleri



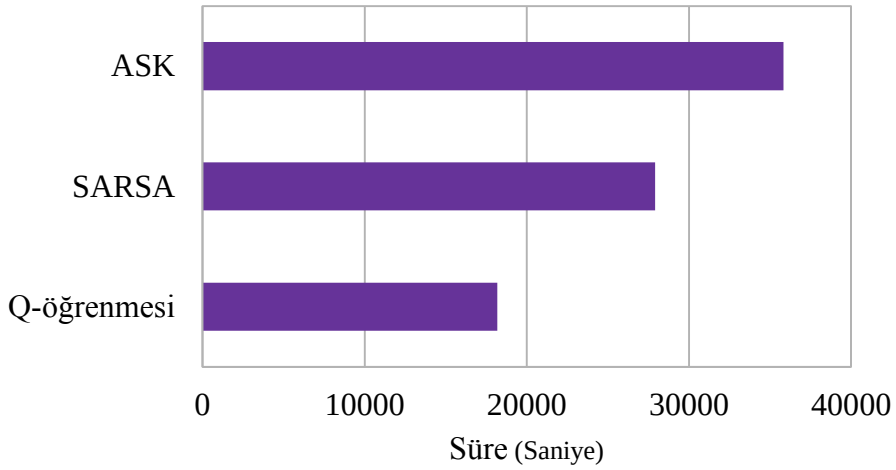
Şekil 4.3. Q-öğrenmesi algoritması performans sonuçları



Şekil 4.4. SARSA algoritması performans sonuçları



Şekil 4.5. Adaptif Sezgisel Kritik algoritması performans sonuçları



Şekil 4.6. Ters sarkaç denemelerinin toplam süresi

Şekil 4.3'te, Q-öğrenmesi algoritmasının her bir denemede elde ettiği adım sayısı ve ortalama adım sayısı gösterilmiştir. Yine Şekil 4.4'te, SARSA algoritması için elde edilen adım sayısı ve her bir deneme için hesaplanan ortalama adım sayısı gösterilmektedir. Şekil 4.5'te, Adaptif Sezgisel Kritik algoritmasının her bir denemede elde ettiği adım *(step)* sayısı ve her bir deneme için hesaplanan ortalama adım sayısı

gösterilmektedir. Şekil 4.6'da ise, bu üç algoritmanın denemelerinin toplam süresi saniye cinsinden karşılaştırılmıştır. Bu süre, denemeler başlatılırken ve bitirilirken saat ve dakika cinsinden kaydedilip, aralarındaki fark ile hesaplatılmıştır.

Her iki algoritmada kullanılan değerler tablolarda gösterilmiştir. En yüksek ortalama adım sayısına Adaptif Sezgisel Kritik algoritması ulaşmıştır. Bunu SARSA öğrenmesi izlemiştir. Q-öğrenmesi ise üçü arasından en az başarı gösteren algoritma olmuştur. Q-öğrenmesi algoritması, başlangıçta elde edilen rastgele diziye bağlı olduğundan, algoritmanın başarısı ve öğrenmesi, parametrelerin yanı sıra, başlangıçta rastgele dizinin oluşturulma şekline oldukça bağlıdır. Adaptif Sezgisel Kritik algoritmasında ise, bu durum söz konusu değildir, algoritmanın başarısı parametrelere bağlıdır.

4.3. Top düşürmeme Simülasyonu ve Sonuçları

4.3.1. Tek-Link

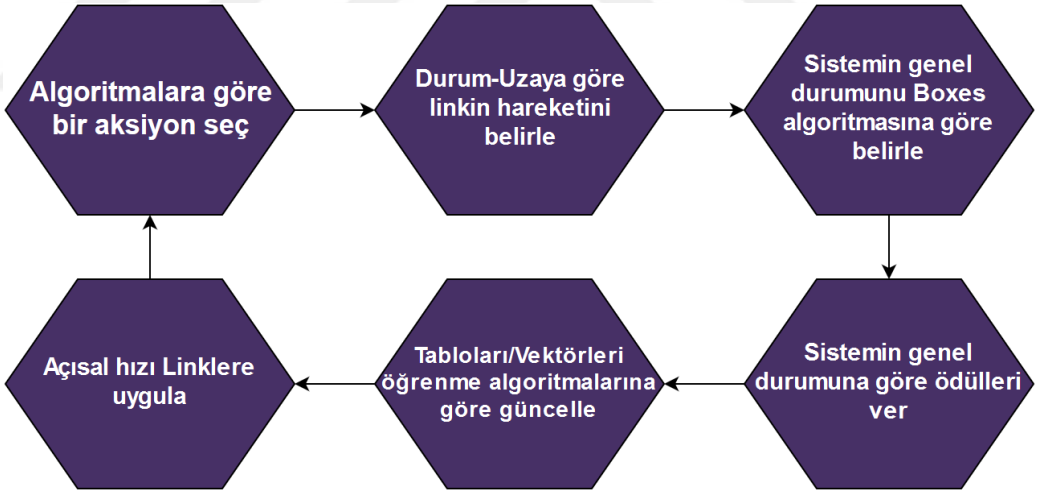
Bu çalışmamızda kullandığımız algoritmalar ve başlangıç parametreleri daha önce belirtildiği gibi Sutton ve Anderson tarafından yazılan halleri örnek alınarak oluşturulmuştur. Simülasyonda kullanılan parametreler Çizelge 4.2'deki gibidir;

Çizelge 4.2. Top düşürmeme tek link simülasyonunda kullanılan değerler

Kaldıraç ağırlığı (m)	0.5
Topun ağırlığı (m_b)	0.1
Kaldıracın boyutları	0.4 x, 0.03 y
Kaldıracın yoğunluğu	10
Topun yarıçapı	0.05
Topun yoğunluğu	10
Alpha (Q-öğrenmesi) (α)	k/(k+n)
Alpha (ASK) (α)	k/(k+n)*1000
k	250

Çizelge 4.2. (devam)

Beta (ASK) (β)	0.5
Gamma (γ)	0.9,0.6,0.3
Reward (r) (ASK) (başarı)	-1
Reward (r) (ASK) (başarısızlık)	0
Reward (r) (Q-öğrenmesi) (başarı)	0.1
Reward (r) (Q-öğrenmesi) (başarısızlık)	0.5
LambdaV	0.8
LambdaW	0.9
Açısal hız	12/-10
Velocity Iteration (Box2d)	8
Position Iteration (Box2d)	3
Hertz	60



Şekil 4.7. Top düşürmeme sisteminin akış şeması

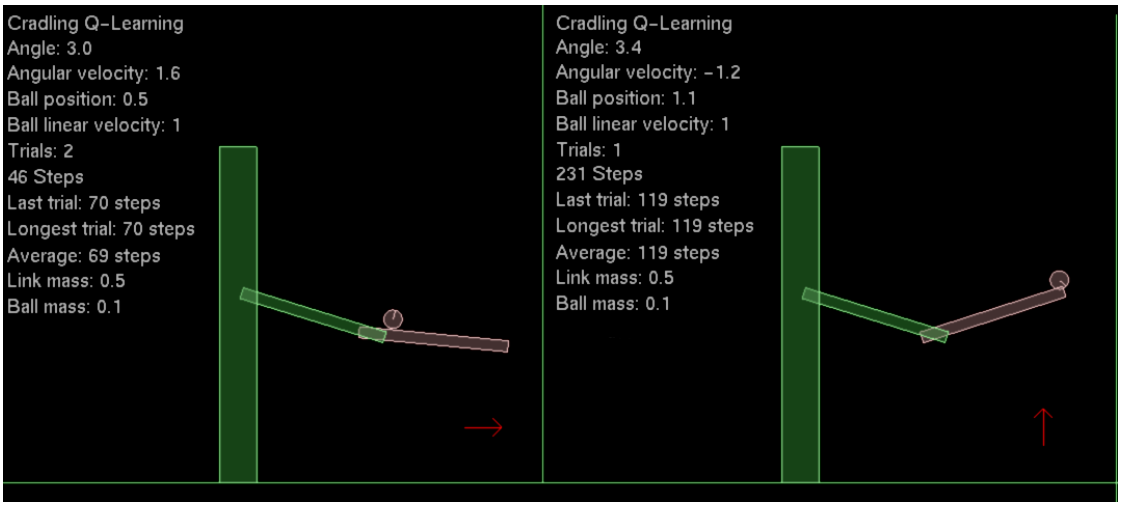
Durum parametreleri olan x, ẋ, θ, θ̇, değerleri simülasyonda anlık olarak Box2d'nin sunduğu fonksiyonlarla okutulmaktadır ve bu şekilde güncellenmektedir. Q-öğrenmesi, SARSA ve Adaptif Sezgisel Kritik algoritması daha önce ters sarkaç uygulaması bölümünde açıklandığı gibi çalışmaktadır. Top her denemede, kaldıraç boyutu üzerinde rastgele bir konumda bırakılmaktadır. Dört adet durum değişkeni, uygulanacak açısal

hızın belirlendiği fonksiyon içinde simülasyon üzerinden gerçek zamanlı olarak alınmaktadır.

Kaldıraca uygulanacak açısal hız, yukarıya doğru ya da aşağıya doğru olmasına göre farklılık göstermektedir. Yukarıya uygulanacak açısal hız 12 olarak belirlenmiştir. Aşağıya uygulanacak açısal hız ise -10 olarak seçilmiştir. Bunun sebebi, kaldıraca uygulanan yer çekimi kuvvetidir. Yer çekimi kuvveti zaten kaldıraca daima etki ettiğinden, aşağı kuvvet yukarı kuvvetten daha az seçilmiştir. Bu kuvvet değeri, doğrudan açısal hız olarak kaldıraca uygulanmaktadır.

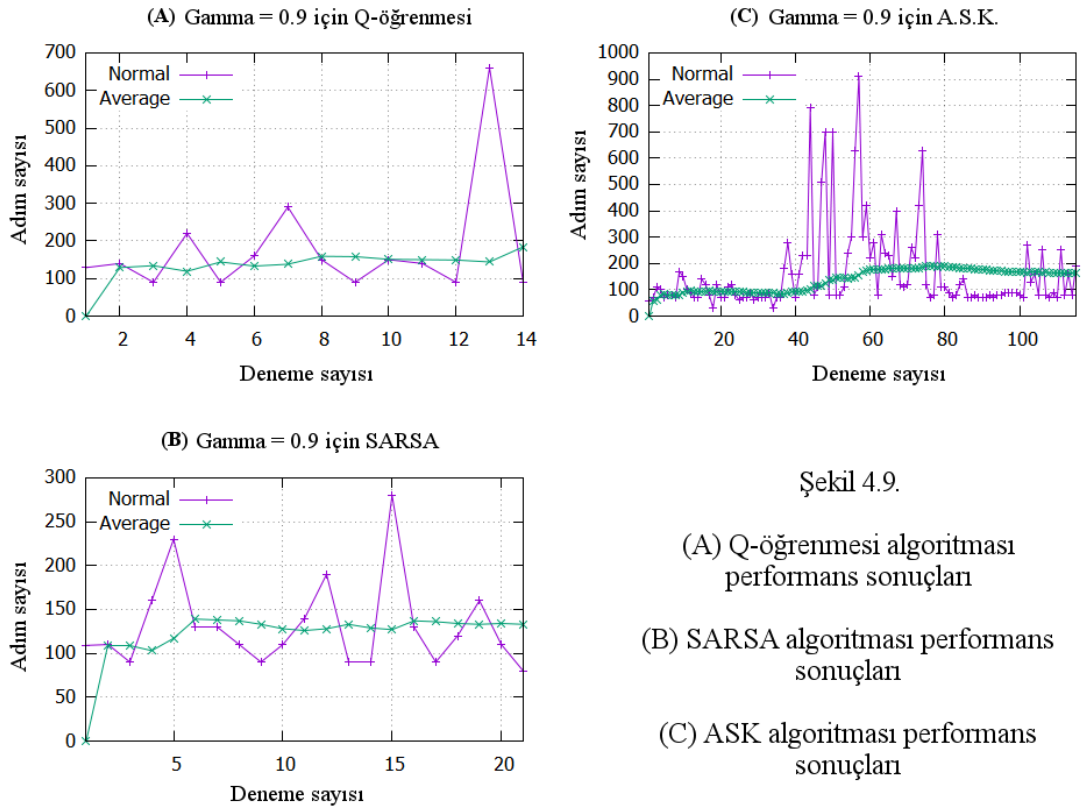
Her bir algoritma için 500 deneme yapılmıştır. Her bir deneme için, topun düşmeden durdurabildiği süre step olarak hesaplanmaktadır. Denemelerin sonunda, her bir deneme için elde edilen adım *(step)* değeri ve ortalama adım değerleri dizi halinde kaydedilmektedir. Tek link için çalışan algoritmalar, belirli bir deneme sayısından sonra topu hiç düşürmeden tutmayı başarabilmektedir. 500 deneme sayısına ulaşmadan top dengede tutulduğundan, başarıya ulaşılan deneme sayısına kadar grafikler çizdirilmiştir.

Şekil 4.8'de, kullandığımız ajanın pekiştirmeli öğrenme algortimalarıyla öğrenme işlemini gerçekleştirirken Box2D çalışma ekranından örnekler gösterilmiştir.



Şekil 4.8. Top düşürmeme tek link ekran görüntüleri

4.3.1.a. Gamma değeri 0.9



Şekil 4.9.

(A) Q-öğrenmesi algoritması performans sonuçları

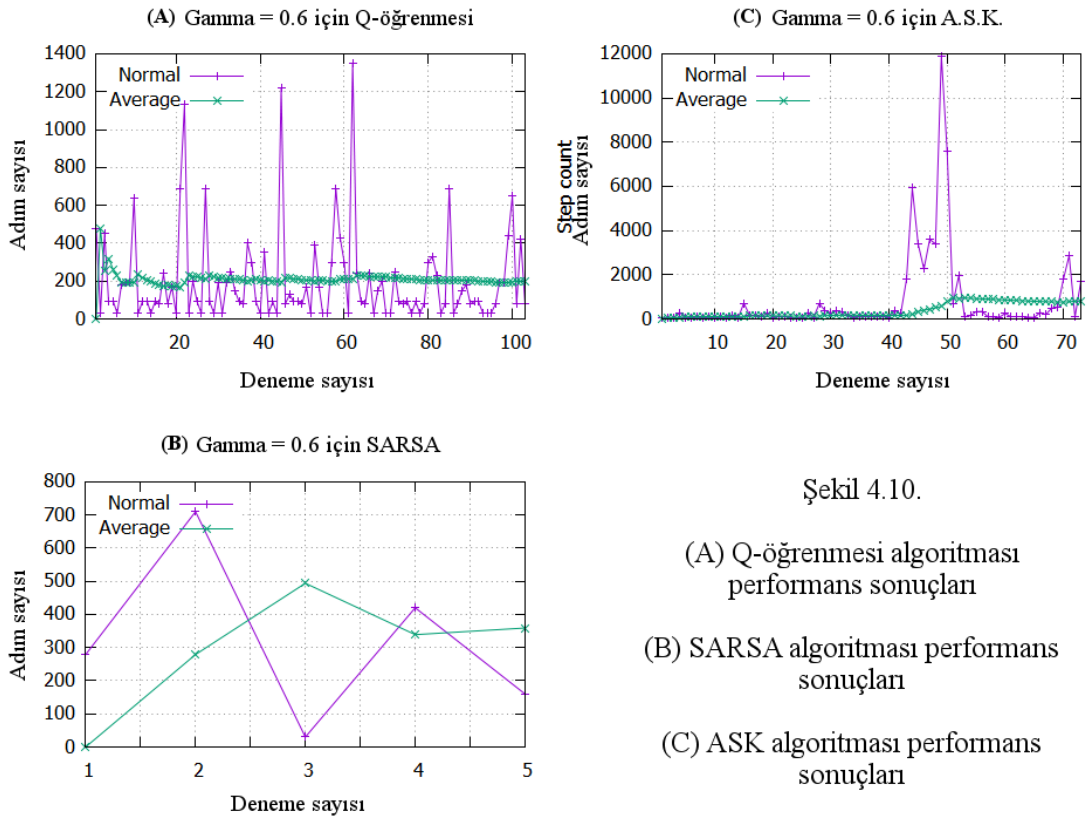
(B) SARSA algoritması performans sonuçları

(C) ASK algoritması performans sonuçları

Şekil 4.9. (A) Q-öğrenmesi algoritması performans sonuçları (B) SARSA algoritması performans sonuçları (C) ASK algoritması performans sonuçları

Şekil 4.9-A'da, Q-öğrenmesi algoritması için her bir denemede elde edilen adım sayısı ve ortalama adım sayısı gösterilmiştir. Şekil 4.9-B'de SARSA algoritması için her bir denemede elde edilen adım sayısı ve ortalama adım sayısı gösterilmiştir. Şekil 4.9-C'de Adaptif Sezgisel Kritik algoritması için her bir denemede elde edilen adım sayısı ve ortalama adım sayısı gösterilmiştir.

4.3.1.b. Gamma değeri 0.6



Şekil 4.10.

(A) Q-öğrenmesi algoritması performans sonuçları

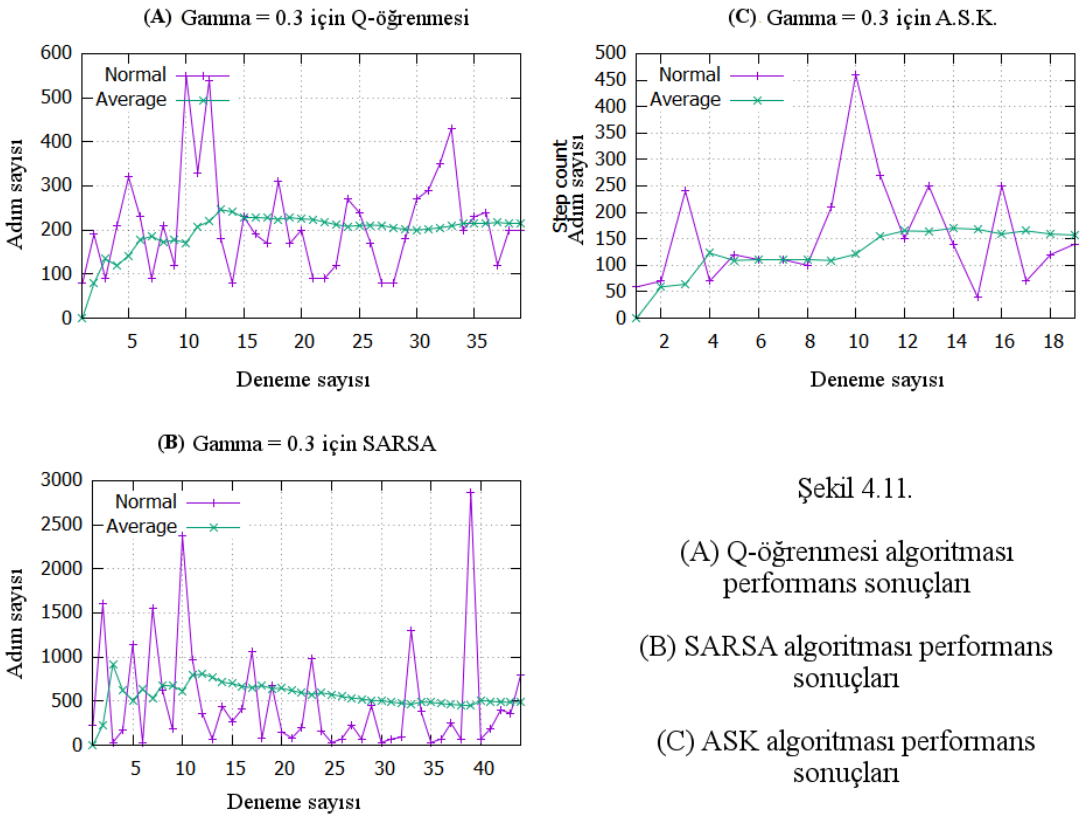
(B) SARSA algoritması performans sonuçları

(C) ASK algoritması performans sonuçları

Şekil 4.10. (A) Q-öğrenmesi algoritması performans sonuçları (B) SARSA algoritması performans sonuçları (C) ASK algoritması performans sonuçları

Şekil 4.10-A'da, Q-öğrenmesi algoritması için her bir denemede elde edilen adım sayısı ve ortalama adım sayısı gösterilmiştir. Şekil 4.10-B'de SARSA algoritması için her bir denemede elde edilen adım sayısı ve ortalama adım sayısı gösterilmiştir. Şekil 4.10-C'de Adaptif Sezgisel Kritik algoritması için her bir denemede elde edilen adım sayısı ve ortalama adım sayısı gösterilmiştir.

4.3.1.c. Gamma değeri 0.3



Şekil 4.11.

(A) Q-öğrenmesi algoritması performans sonuçları

(B) SARSA algoritması performans sonuçları

(C) ASK algoritması performans sonuçları

Şekil 4.11. (A) Q-öğrenmesi algoritması performans sonuçları (B) SARSA algoritması performans sonuçları (C) ASK algoritması performans sonuçları

Şekil 4.11-A'da, Q-öğrenmesi algoritması için her bir denemede elde edilen adım sayısı ve ortalama adım sayısı gösterilmiştir. Şekil 4.11-B'de SARSA algoritması için her bir denemede elde edilen adım sayısı ve ortalama adım sayısı gösterilmiştir. Şekil 4.11-C'de Adaptif Sezgisel Kritik algoritması için her bir denemede elde edilen adım sayısı ve ortalama adım sayısı gösterilmiştir.

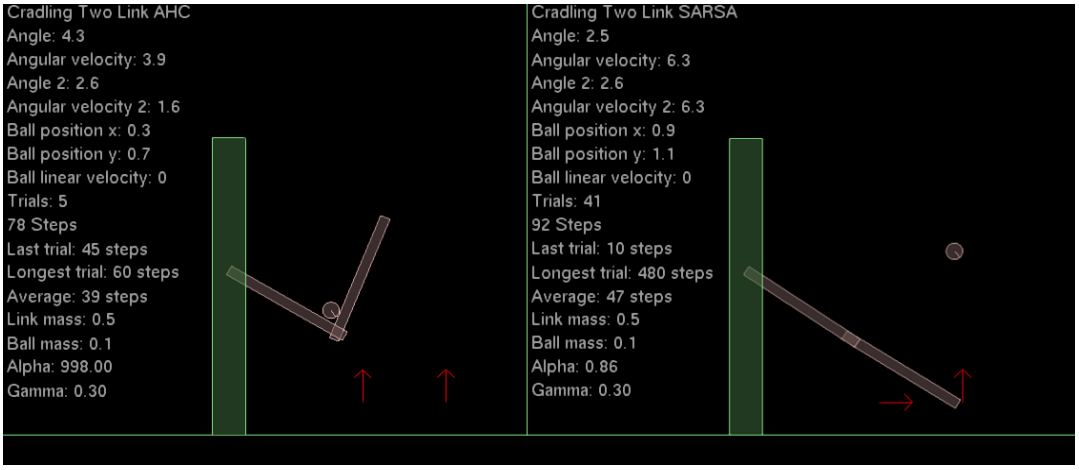
4.3.2. Çift-Link

Çift link için başlangıç parametreleri olarak uygulanan değerler Çizelge 4.3'deki gibidir;

Çizelge 4.3. Top düşürmeme çift link simülasyonunda kullanılan değerler

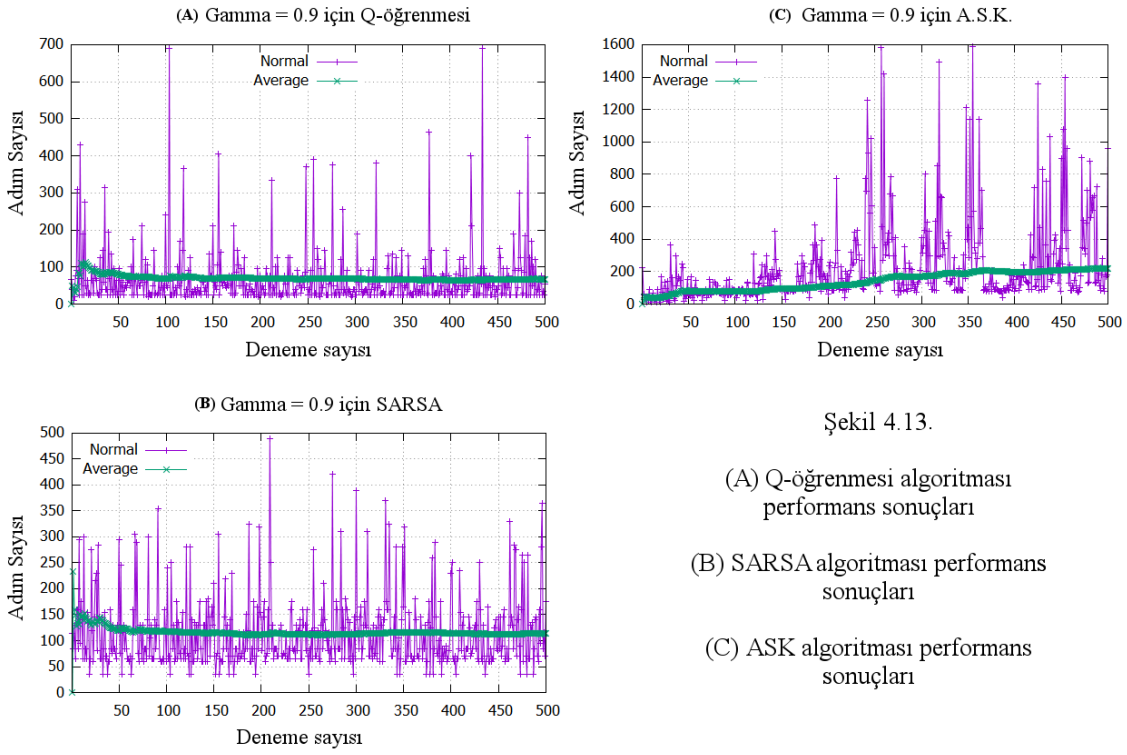
Kaldıraç ağırlığı (m)	0.5
Topun ağırlığı (m_b)	0.1
Kaldıracın boyutları	0.4 x, 0.03 y
Kaldıracın yoğunluğu	10
Topun yarıçapı	0.05
Topun yoğunluğu	10
Alpha (Q-öğrenmesi) (α)	k/(k+n)
Alpha (ASK) (α)	k/(k+n)*1000
k	250
Beta (ASK) (β)	0.5
Gamma (γ)	0.9,0.6,0.3
Reward (r) (ASK) (başarı)	-1
Reward (r) (ASK) (başarısızlık)	0
Reward (r) (Q-öğrenmesi) (başarı)	0.1
Reward (r) (Q-öğrenmesi) (başarısızlık)	0.5
LambdaV (ASK)	0.8
LambdaW (ASK)	0.9
Açısal hız (link 1)	12/-10
Açısal hız (link 2)	120/-1
Velocity Iteration (Box2d)	8
Position Iteration (Box2d)	3
Hertz	60

Şekil 4.12'de, kullandığımız ajanın pekiştirmeli öğrenme algortimalarıyla öğrenme işlemini gerçekleştirirken Box2d çalışma ekranından örnekler gösterilmiştir.



Şekil 4.12. Çift-Link top düşürmeme ekran görüntüleri

4.3.2.a. Gamma değeri 0.9



Şekil 4.13.

(A) Q-öğrenmesi algoritması performans sonuçları

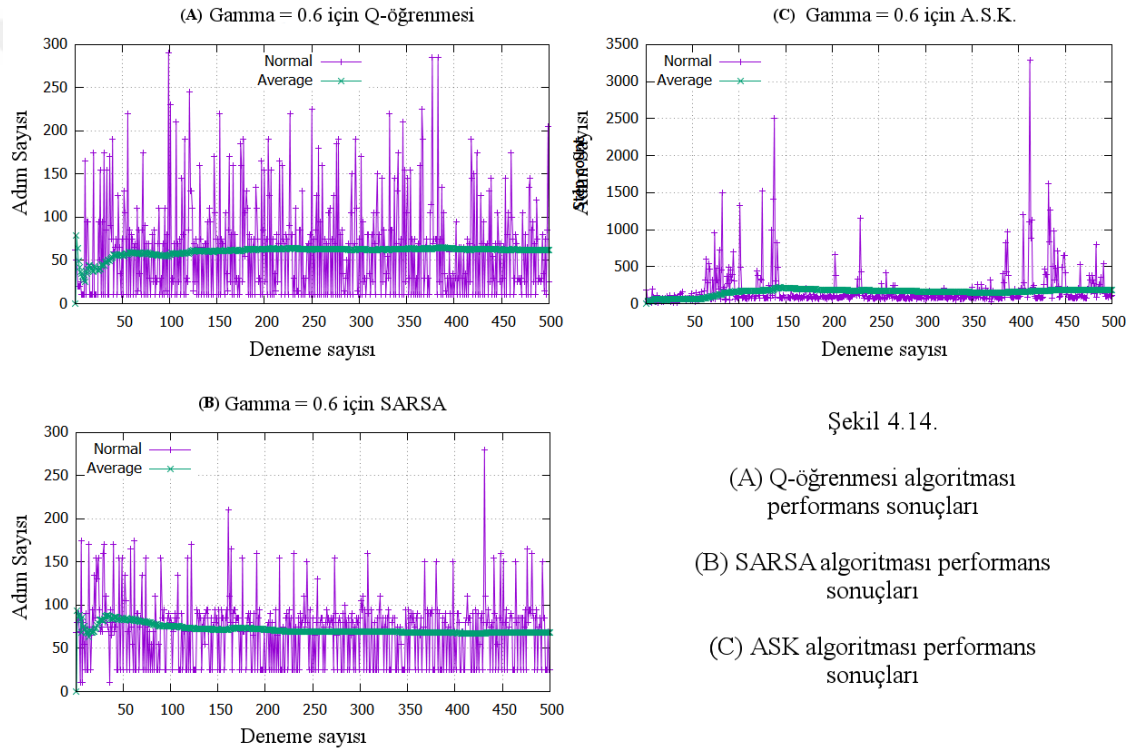
(B) SARSA algoritması performans sonuçları

(C) ASK algoritması performans sonuçları

Şekil 4.13. (A) Q-öğrenmesi algoritması performans sonuçları (B) SARSA algoritması performans sonuçları (C) ASK algoritması performans sonuçları

Şekil 4.13-A'da, Q-öğrenmesi algoritması için her bir denemede elde edilen adım sayısı ve ortalama adım sayısı gösterilmiştir. Şekil 4.13-B'de SARSA algoritması için her bir denemede elde edilen adım sayısı ve ortalama adım sayısı gösterilmiştir. Şekil 4.13-C'de Adaptif Sezgisel Kritik algoritması için her bir denemede elde edilen adım sayısı ve ortalama adım sayısı gösterilmiştir.

4.3.2.b. Gamma değeri 0.6



Şekil 4.14.

(A) Q-öğrenmesi algoritması performans sonuçları

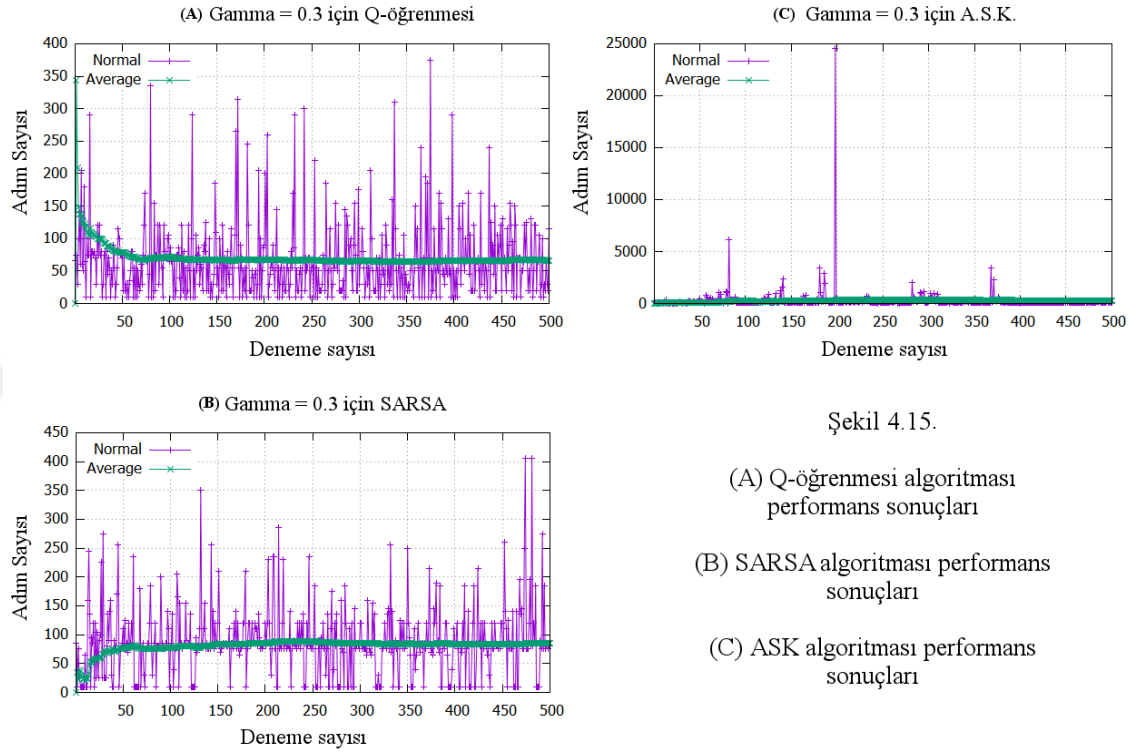
(B) SARSA algoritması performans sonuçları

(C) ASK algoritması performans sonuçları

Şekil 4.14. (A) Q-öğrenmesi algoritması performans sonuçları (B) SARSA algoritması performans sonuçları (C) ASK algoritması performans sonuçları

Şekil 4.14-A'da, Q-öğrenmesi algoritması için her bir denemede elde edilen adım sayısı ve ortalama adım sayısı gösterilmiştir. Şekil 4.14-B'de SARSA algoritması için her bir denemede elde edilen adım sayısı ve ortalama adım sayısı gösterilmiştir. Şekil 4.14-C'de Adaptif Sezgisel Kritik algoritması için her bir denemede elde edilen adım sayısı ve ortalama adım sayısı gösterilmiştir.

4.3.2.c. Gamma değeri 0.3



Şekil 4.15.

(A) Q-öğrenmesi algoritması performans sonuçları

(B) SARSA algoritması performans sonuçları

(C) ASK algoritması performans sonuçları

Şekil 4.15. (A) Q-öğrenmesi algoritması performans sonuçları (B) SARSA algoritması performans sonuçları (C) ASK algoritması performans sonuçları

Şekil 4.15-A'da, Q-öğrenmesi algoritması için her bir denemede elde edilen adım sayısı ve ortalama adım sayısı gösterilmiştir. Şekil 4.15-B'de SARSA algoritması için her bir denemede elde edilen adım sayısı ve ortalama adım sayısı gösterilmiştir. Şekil 4.15-C'de Adaptif Sezgisel Kritik algoritması için her bir denemede elde edilen adım sayısı ve ortalama adım sayısı gösterilmiştir.

Algoritmalar, tek link sistem için başarı göstermiş olup, çift link sistemde ise tam bir öğrenme gerçekleşmemiştir. Bunun sebebi, iki linkin birbiriyle haberleşmeden bağımsız hareket etmesine bağlanabilir. İleriki çalışmalarda, bu iki linkin birbiriyle bilgilerini paylaşacak şekilde tasarlanması ile çalışma genişletilebilir.

4.4. Top fırlatma Simülasyonu ve Sonuçları

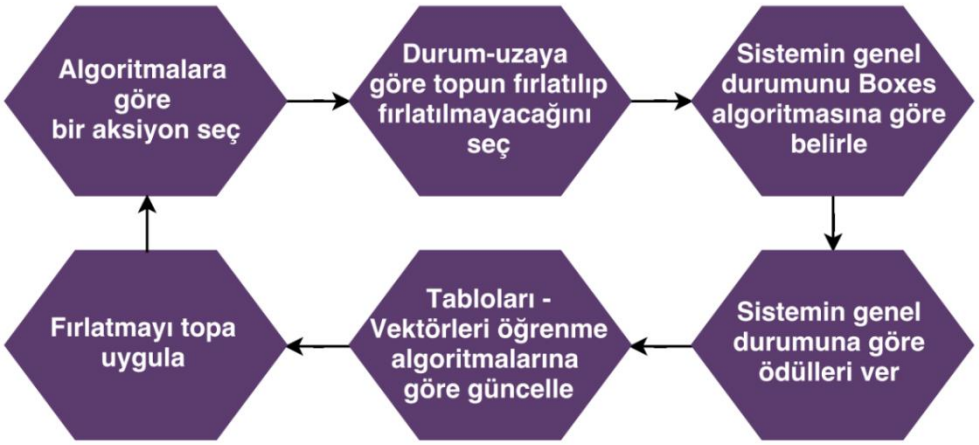
Top fırlatma sistemi, sabit bir nokta etrafında sabit açısal hızla dönen bir çubuğun ucuna yerleştirilmiş bir topun, en uzak mesafeye fırlatılmasını amaçlayan bir sistemdir. Çubuk, sabit bir açısal hızla hep saat yönünde dönmekte, iki boyutlu bir düzlemde topu ya sağa ya da sola doğru fırlatmaktadır. Sağa doğru fırlatılan topun aldığı mesafe pozitif, sola doğru aldığı mesafe ise negatiftir.

Top fırlatma için kullanılan başlangıç değerleri Çizelge 4.4'te gösterilmiştir. Her bir algoritma için 500 deneme yapılmıştır.

Durum değişkenleri olan x, y ve θ, simülasyondan anlık olarak okunmaktadır. θ derece cinsindendir. Boxes algoritmasına göre, top eğer belirli bir yükseklikte, kaldıracın açısı da belirli bir açıda ise ve topun x konumu da -0.3'ten büyükse, optimum uzaklığa ulaştıracak atılma konumuna geldiği kabul edilmektedir. Bu durumlar, Box2d simülatöründe belirli fonksiyonlar yardımıyla anlık olarak güncellenmektedir. Sistem, linke sürekli olarak -5 değerinde bir açısal hız uygulamaktadır. Boxes algoritmasına göre, aksiyon olarak fırlat ya da fırlatma komutu verilmektedir.

Her bir denemede top yere düştüğü anda katettiği mesafe simülatörün ölçtüğü mesafe cinsinden kaydedilmektedir. Şekilde Simülasyondan ekran görüntüleri görülmektedir.

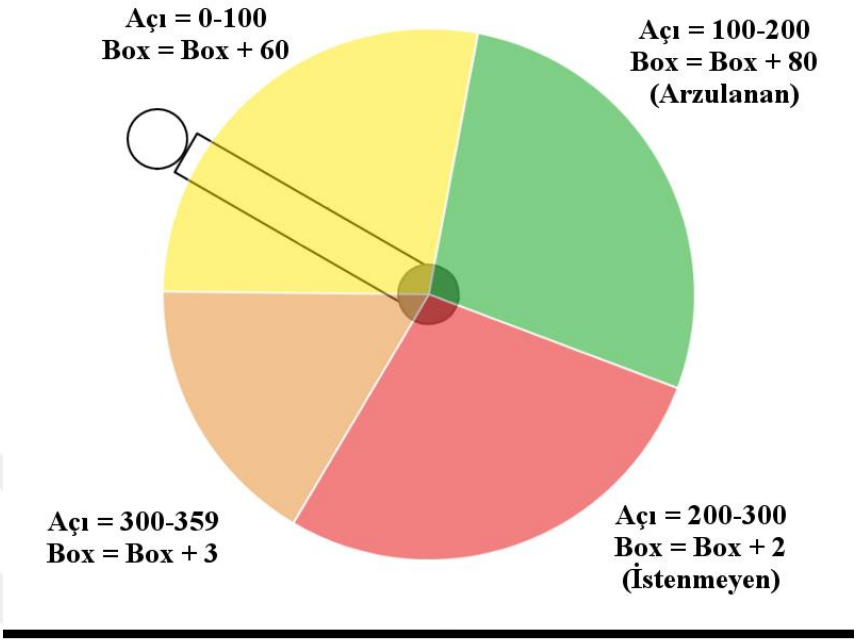
Bütün algoritmalar için 500 deneme yapılmıştır. Her bir denemede topun fırlatıldığı mesafe kaydedilmiştir. Her bir deneme sonunda bu mesafeler toplanmış, sonra bulunulan deneme sayısına böldürülerek her bir adım için ortalama fırlatılan mesafe hesaplanmıştır. Top fırlatıldıktan sonra düştüğü yerdeki mesafe kaydedilmektedir, sistem ise dönmeye devam etmekte, herhangi bir yeniden başlatma olmamaktadır.



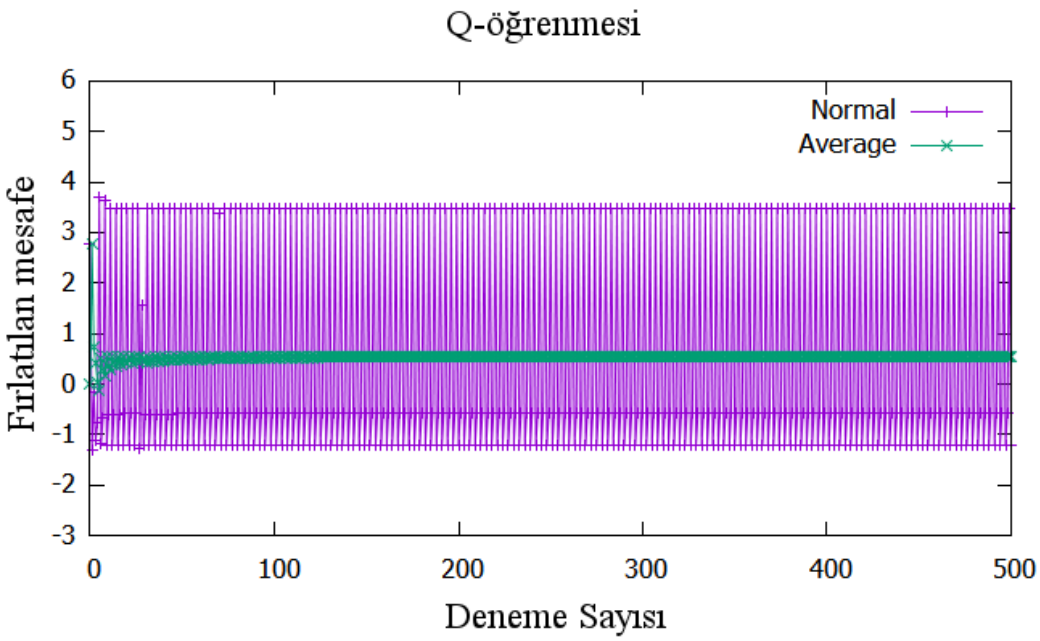
Şekil 4.16. Top fırlatma sistemi akış şeması

Çizelge 4.4. Top fırlatma simülasyonunda kullanılan değerler

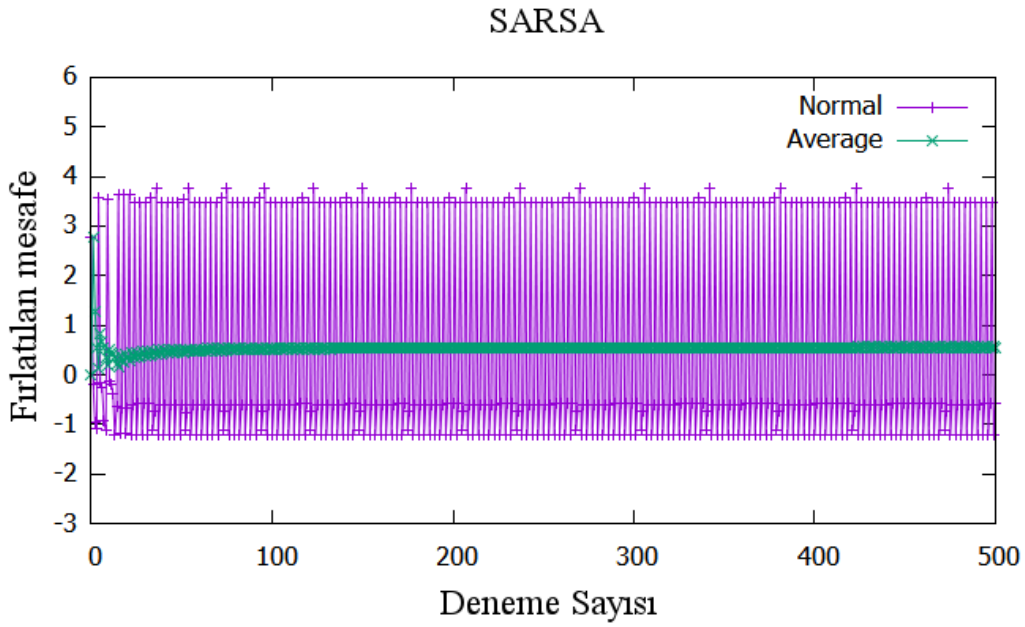
Kaldıraç ağırlığı (m)	0.5
Topun ağırlığı (m_b)	0.1
Kaldıracın boyutları	0.4 x, 0.03 y
Kaldıracın yoğunluğu	10
Topun yarıçapı	0.05
Topun yoğunluğu	10
Alpha (Q-öğrenmesi) (α)	k/(k+n)
Alpha (ASK) (α)	k/(k+n)*1000
k	250
Beta (ASK) (β)	0.5
Gamma (γ)	0.9
Reward (r) (failure)	-1
Reward (r) (success)	0
LambdaV (ASK)	0.8
LambdaW (ASK)	0.9
Açısal hız	-5
Fırlatma hızı (Vektör)	0.5x, 0y / -0.5x, 0y
Velocity Iteration (Box2d)	8
Position Iteration (Box2d)	3
Hertz	60



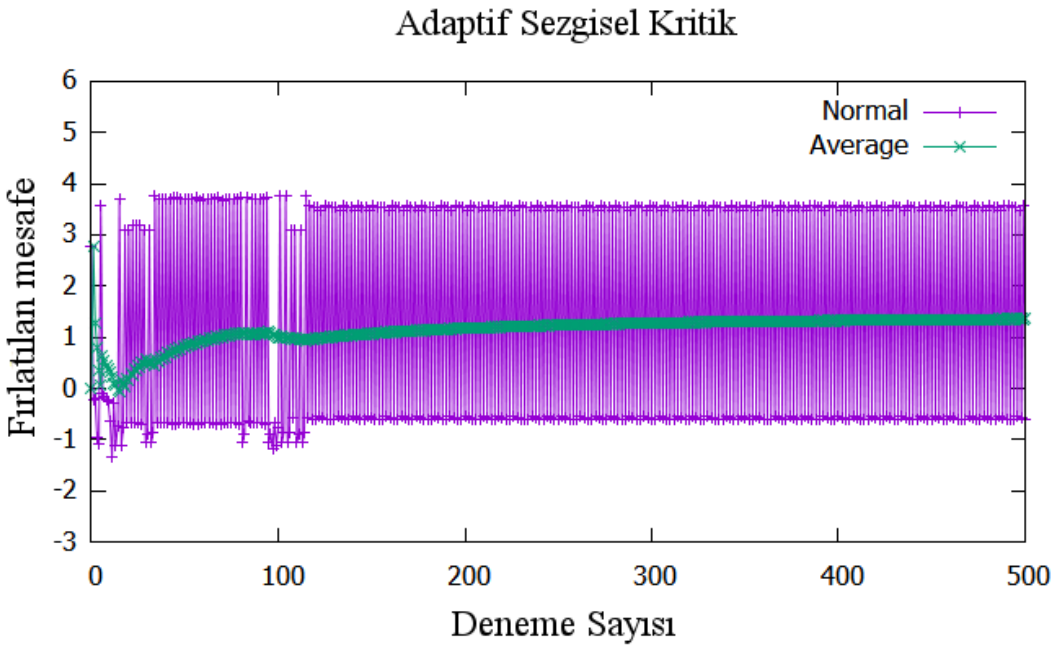
Şekil 4.17. Top fırlatma Boxes algoritmasının şekilsel gösterimi



Şekil 4.18. Q-Öğrenmesi algoritması performans sonuçları



Şekil 4.19. SARSA algoritması performans sonuçları



Şekil 4.20. ASK algoritması performans sonuçları

Üç farklı algoritmanın belirtilen parametreler ile gösterdiği performansın grafikleri Şekil 4.18, 4.19 ve 4.20'de görülmektedir. Şekil 4.18; Q-öğrenmesi, Şekil 4.19; SARSA, Şekil 4.20; Adaptif Sezgisel Kritik algoritmasına aittir.

Dikey eksen fırlatılan mesafe uzaklığını, yatay eksen ise deneme sayısını göstermektedir. Mor renkteki grafikler, her adım için fırlatma mesafesini göstermekte, yeşil renkteki grafikler ise her adım için ortalama fırlatma mesafesini göstermektedir.

Grafik sonuçlarına göre, en başarılı olan ve öğrenmenin en bariz gözlendiği algoritma Adaptif Sezgisel Kritik algoritması olmuştur. Q-öğrenmesi ve SARSA benzer performanslar göstermiş olup, belirgin bir öğrenme sağlanamamıştır.

Q-öğrenmesi ve SARSA algoritmaları, başlangıçta rastgele dizilerle doldurulduklarından için, bu dizilerin oluşturulma şekli performansı oldukça etkilemektedir. Yapılacak bir çalışmada, rastgele dizinin bir dağılımla oluşturulması ve deneme sayısının artırılması, daha başarılı bir performans göstermesini sağlayabilir.

4.5. Kullanılan Sistem ve Araçlar

Bu çalışma, Visual Studio 2015 ortamında, Sony Vaio marka bir dizüstü bilgisayarda hazırlanmıştır. Sistemin özellikleri şu şekildedir; Intel Core i5 3230M 2.60 Ghz CPU, 8 GB RAM, AMD HD 7500M/7600M Ekran Kartı, 700 GB HDD. Yardımcı araç olarak kodların sürüm takibinin yapılabilmesi için SmartGit adlı yazılım kullanılmıştır. SmartGit, Syntevo firması tarafından geliştirilmiş, ücretsiz sunumu da bulunan, üzerinde çalışılan kod dosyalarında yapılan değişiklikleri tutarak, zamanla yapılan değişiklikleri inceleme, takım halinde aynı proje üzerinde çalışma, dosyayı eksi haline getirme gibi fonksiyonlar içeren bir Git (kaynak kod yönetim sistemi)'dir.

5. SONUÇ

Çalışmamızda ters sarkaç sisteminin kendi kendine dengede kalmasını, top düşürmeme sisteminin topu düşürmeden tutabilmesini, top fırlatma sisteminin topu en uzağa fırlatmayı öğrenebilmesi için kullanılan üç farklı pekiştirmeli öğrenme algoritmasının performans sonuçları incelenmiştir. Her üç algoritma da belirtilen parametreler ile Box2d simülatörüyle çalıştırılmış ve birbirleri ile farklı sonuçlar elde edilmiştir. Q-öğrenmesi ve SARSA algoritması, başlangıçta elde edilen rastgele diziye bağlı olduğundan, algoritmanın başarısı başlangıçta seçilen rastgele dizinin oluşturulma şekline oldukça bağlıdır. Adaptif Sezgisel Kritik algoritmasında ise, sonuçlar seçilecek parametre değerlerine daha fazla bağlıdır. Adaptif Sezgisel Kritik algoritmasında öğrenme daha belirgin gözlemlenmiştir. Top düşürmeme sistemi ve top fırlatma sistemi, deneysel problemler olup, top düşürmeme sisteminde denenen farklı γ değerleri arasında anlamsal bir ilişki kurmak güçtür.

Q-öğrenmesi, daha fazla keşfetmeye ihtiyaç duymaktadır. Bu yüzden, genellikle Q-öğrenmede, başlangıçtaki dizi modelin öğrenme geliştirmesine uygun olarak verilmelidir. Veyahutta, Q-öğrenmesine yardımcı olabilecek başka algoritmalar ile birlikte kullanılması düşünülebilir.

Biz bu çalışmamızda, farklı pekiştirmeli öğrenme algoritmalarını çalıştırıp performanslarını ölçebilecek bir iskelet tasarladık. Bu iskelete farklı algoritmalar ve özellikler dahil edilmesi oldukça kolaydır.

Yapılacak ileriki çalışmalarda, bu algoritmaların farklı parametreler ve tablo değerleriyle çalıştırılıp daha detaylı ve derin analizler yapılması düşünülebilir. Bu çalışmalardan elde edilen sonuçlar, gerçek mobil robotlar üzerinde uygulanıp çalışma genişletilebilir. Daha farklı algoritmalar da kıyaslamaya katılıp kapsamlı bir pekiştirmeli öğrenme test düzeneği oluşturulup, bu düzenek diğer kullanıcılarla paylaşıma açılabilir.

KAYNAKLAR

Adam, S., Busoniu, L., & Babuska, R. 2012. Experience replay for real-time reinforcement learning control. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 42(2), 201-212.

Anders, A. A. S. 2014. Learning a strategy for whole-arm grasping (Doctoral dissertation, Massachusetts Institute of Technology).

Anonymous 2015a. http://webdocs.cs.ualberta.ca/~sutton/book/code/pole.c (27.06.2015)

Anonymous 2015b. http://www.cs.cmu.edu/afs/cs/project/lri/members/lalit/Phd/course/nn/ asg4/qpole.c (27.06.2015)

Anonymous, 2016a. http://www.scholarpedia.org/article/Reinforcement_learning (04.09.2016)

Anonymous, 2016b. https://en.wikipedia.org/wiki/Box2D (23.05.2016)

Asada, M., Noda, S., Tawaratsumida, S., & Hosoda, K. 1996. Purposive behavior acquisition for a real robot by vision-based reinforcement learning. In Recent Advances in Robot Learning (pp. 163-187). Springer US.

Bagnell, J. A., & Schneider, J. G. 2001. Autonomous helicopter control using reinforcement learning policy search methods. In Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on (Vol. 2, pp. 1615-1620). IEEE.

Barto, M. T. R. A. G. 2004. J. 4 supervised actor-critic reinforcement learning. Handbook of learning and approximate dynamic programming, 2, 359.

Becerikli, Y., & Celik, B. K. 2007. Fuzzy control of inverted pendulum and concept of stability using Java application. Mathematical and Computer Modelling, 46(1), 24-37.

Birdwell, N., Livingston, S., & Elhanany, I. 2007. Reinforcement learning in sensor-guided aibo robots. Technical report, University of Tennesse, Knoxville.

Bloembergen, D., & Westra, R. 2010. Analyzing reinforcement learning algorithms using evolutionary game theory.

Boubaker, O., 2012. The inverted pendulum: a fundamental benchmark in control theory and robotics. In Education and e-Learning Innovations (ICEELI), 2012 international conference on (pp. 1-6). IEEE.

Brownlee, J. 2005. The pole balancing problem: A benchmark control theory problem.

Buchli, J., Stulp, F., Theodorou, E., & Schaal, S. 2011. Learning variable impedance control. The International Journal of Robotics Research, 30(7), 820-833.

Chen, Y. F., Huang, A. C. 2014. Adaptive control of rotary inverted pendulum system with time-varying uncertainties. Nonlinear Dynamics, 76(1), 95-102.

Coggan, Melanie. Exploration and exploitation in reinforcement learning. Research supervised by Prof. Doina Precup, CRA-W DMP Project at McGill University 2004.

Even-Dar, Eyal, and Yishay Mansour. Learning rates for Q-learning. The Journal of Machine Learning Research 5 2004: 1-25.

Fang, X., He, H., Ni, Z., & Tang, Y. 2012. Learning and control in virtual reality for machine intelligence. In Intelligent Control and Information Processing (ICICIP), 2012 Third International Conference on (pp. 63-67). IEEE.

Faustino, P. F. P., 2011. Dynamic Equilibrium Through Reinforcement Learning. M.S. thesis, Computer and Information Engineering, ISEL, Lisbon, Portugal, 2011

Gaskett, C., Fletcher, L., & Zelinsky, A. 2000. Reinforcement learning for a vision based mobile robot. In Intelligent Robots and Systems, 2000. (IROS 2000). Proceedings. 2000 IEEE/RSJ International Conference on (Vol. 1, pp. 403-409). IEEE.

Gaskett, C., Wettergreen, D., Zelinsky, A. 1999. Q-learning in continuous state and action spaces. In Australasian Joint Conference on Artificial Intelligence (pp. 417-428). Springer Berlin Heidelberg.

Hafner, R., Riedmiller, M. 2007. Neural reinforcement learning controllers for a real robot application. In Proceedings 2007 IEEE International Conference on Robotics and Automation (pp. 2098-2103). IEEE.

Hehn, M., & D'Andrea, R. 2011. A flying inverted pendulum. In Robotics and Automation (ICRA), 2011 IEEE International Conference on (pp. 763-770). IEEE.

Hester, T., Quinlan, M., Stone, P. 2010. Generalized model learning for reinforcement learning on a humanoid robot. In Robotics and Automation (ICRA), 2010 IEEE International Conference on (pp. 2369-2374). IEEE.

Huang, C. H., Wang, W. J., & Chiu, C. H. 2011. Design and implementation of fuzzy control on a two-wheel inverted pendulum. IEEE Transactions on Industrial Electronics, 58(7), 2988-3001.

Huang, J., Guan, Z. H., Matsuno, T., Fukuda, T., Sekiyama, K. 2010. Sliding-mode velocity control of mobile-wheeled inverted-pendulum systems. IEEE transactions on robotics, 26(4), 750-758.

Hwang, K. S., Jiang, W. C., Chen, Y. J. 2013. Adaptive Model Learning Based on Dyna-Q Learning. Cybernetics and Systems, 44(8), 641-662.

Irodova, Marina, and Robert H. Sloan. 2005 Reinforcement Learning and Function Approximation. FLAIRS Conference.

Kaelbling, L. P., Littman, M. L., Moore, A. W. 1996. Reinforcement learning: A survey. Journal of artificial intelligence research, 4, 237-285.

Kajita, S., Morisawa, M., Miura, K., Nakaoka, S. I., Harada, K., Kaneko, K., Kanehiro, F., Yokoi, K. 2010. Biped walking stabilization based on linear inverted pendulum tracking. In Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on (pp. 4489-4496). IEEE.

Kalakrishnan, M., Buchli, J., Pastor, P., Mistry, M., Schaal, S. 2011. Learning, planning, and control for quadruped locomotion over challenging terrain. The International Journal of Robotics Research, 30(2), 236-258.

Katahira, Kentaro. 2015. The relation between reinforcement learning parameters and the influence of reinforcement history on choice behavior. Journal of Mathematical Psychology 66. 59-69.

Khansari-Zadeh, S. M., Kronander, K., Billard, A. 2012. Learning to play minigolf: A dynamical system-based approach. Advanced Robotics, 26(17), 1967-1993.

Kober, J., Bagnell, J. A., Peters, J. 2013. Reinforcement learning in robotics: A survey. The International Journal of Robotics Research, 0278364913495721.

Kober, J., Oztop, E., Peters, J. 2011. Reinforcement learning to adjust robot movements to new situations. In 2010 Robotics: Science and Systems Conference (RSS 2010) (pp. 33-40). MIT Press.

Kober, J., Peters, J. R. 2009. Policy search for motor primitives in robotics. In Advances in neural information processing systems (pp. 849-856).

Kohl, N., Stone, P. 2004. Policy gradient reinforcement learning for fast quadrupedal locomotion. In Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on (Vol. 3, pp. 2619-2624). IEEE.

Konidaris, G., Kuindersma, S., Grupen, R. A., Barto, A. G. 2011. Autonomous Skill Acquisition on a Mobile Manipulator. In AAAI.

Koujaku, S., Watanabe, K., Igarashi, H. 2011. Adaptive Profit Sharing Reinforcement Learning Method for Dynamic Environment. In Machine Learning and Applications and Workshops (ICMLA), 2011 10th International Conference on (Vol. 1, pp. 462-465). IEEE.

Kronander, K., Khansari-Zadeh, M. S., Billard, A. 2011. Learning to control planar hitting motions in a minigolf-like task. In 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (pp. 710-717). IEEE.

Kumar, R., Singh, R. B., Das, J. 2013. Modeling and simulation of inverted pendulum system using matlab: overview. Int J Mech Prod Eng, 1(4), 2320-2092

Latzke, T., Behnke, S., Bennewitz, M. 2006. Imitative reinforcement learning for soccer playing robots. In Robot Soccer World Cup (pp. 47-58). Springer Berlin Heidelberg.

Mahadevan, S., Connell, J. 1992. Automatic programming of behavior-based robots using reinforcement learning. Artificial intelligence, 55(2-3), 311-365.

Martínez-Marín, T., Duckett, T. 2005. Fast reinforcement learning for vision-guided mobile robots. In Proceedings of the 2005 IEEE International Conference on Robotics and Automation (pp. 4170-4175). IEEE.

Mataric, M. J. 1994. Reward functions for accelerated learning. In Machine Learning: Proceedings of the Eleventh international conference (pp. 181-189).

Matignon, Laëtitia, Guillaume J. Laurent, and Nadine Le Fort-Piat. 2006. Reward function and initial values: better choices for accelerated goal-directed reinforcement learning. Artificial Neural Networks–ICANN 2006. Springer Berlin Heidelberg, 840-849.

Michie, D., Chambers, R. A. 1968. BOXES: An experiment in adaptive control. Machine intelligence, 2(2), 137-152.

Morimoto, J., Doya, K. 2001. Acquisition of stand-up behavior by a real robot using hierarchical reinforcement learning. Robotics and Autonomous Systems, 36(1), 37-51.

Muralidharan, V., Mahindrakar, A. D. 2014. Position stabilization and waypoint tracking control of mobile inverted pendulum robot. IEEE Transactions on Control Systems Technology, 22(6), 2360-2367.

Najafi, E., Lopes, G. A., Babuška, R. 2013. Reinforcement learning for sequential composition control. In 52nd IEEE Conference on Decision and Control (pp. 7265-7270). IEEE.

Nakada, M., Allen, B., Morishima, S., Terzopoulos, D. 2010. Learning arm motion strategies for balance recovery of humanoid robots. In Emerging Security Technologies (EST), 2010 International Conference on (pp. 165-170). IEEE.

Nemec, B., & Ude, A. 2011. Reinforcement learning of ball-in-a-cup playing robot. In Robotics and Biomimetics (ROBIO), 2011 IEEE International Conference on (pp. 2682-2987). IEEE.

Neumann, G. 2006. Reinforcement Learning Toolbox Tutorial Institute for Theoretical Computer Science.

Nissen, S. 2007. Large scale reinforcement learning using q-sarsa (λ) and cascading neural networks. Unpublished masters thesis, Department of Computer Science, University of Copenhagen, København, Denmark.

Paletta, L., Pinz, A. 2000. Active object recognition by view integration and reinforcement learning. Robotics and Autonomous Systems, 31(1), 71-86.

Peters, J., Schaal, S. 2008. Reinforcement learning of motor skills with policy gradients. Neural networks, 21(4), 682-697.

Peters, J., Vijayakumar, S., Schaal, S. 2003. Reinforcement learning for humanoid robotics. In Proceedings of the third IEEE-RAS international conference on humanoid robots (pp. 1-20).

Porta, J. M., Celaya, E. 2001. Efficient gait generation using reinforcement learning. In Proceedings of the Fourth International Conference on Climbing and Walking Robots (pp. 411-418).

Qamar, T., 2012. Inverted Pendulum Control Using Soft Computing Techniques, Phd. Synopsis, Faculty of Engineering, Dayalbagh Educational, Institute, Dayalbagh, Agra.

Rao, J., Bu, X., Xu, C. Z., Wang, L., Yin, G. 2009. VCONF: a reinforcement learning approach to virtual machines auto-configuration. In Proceedings of the 6th international conference on Autonomic computing (pp. 137-146). ACM.

Rückstiess, Thomas, et al. 2010. Exploring parameter space in reinforcement learning. Paladyn, Journal of Behavioral Robotics 1.1:14-24.

Sammut, C. 1994. Recent progress with BOXES. In Machine intelligence 13 (pp. 363-383).

Sutton, R. S., Barto, A. G. 1998. Reinforcement learning: An introduction (Vol. 1, No. 1). Cambridge: MIT press.

Tanwani, A. K., Billard, A. 2013. Transfer in inverse reinforcement learning for multiple strategies. In 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (pp. 3244-3250). Ieee.

Tedrake, R., Zhang, T. W., Seung, H. S. 2005. Learning to walk in 20 minutes. In Proceedings of the Fourteenth Yale Workshop on Adaptive and Learning Systems (Vol. 95585).

Troniak, D. M. Optimal Control Learning in Noisy Environments.

Tsai, Y. Y., Lin, W. C., Cheng, K. B., Lee, J., Lee, T. Y. 2010. Real-time physics-based 3d biped character animation using an inverted pendulum model. IEEE transactions on visualization and computer graphics, 16(2), 325-337.

Van Hasselt, H. 2005. Reinforcement learning in continuous state and action spaces. Ms Thesis. University of Utrecht

Xu, J. X., Guo, Z. Q., Lee, T. H. 2014. Design and implementation of integral sliding-mode control on an underactuated two-wheeled mobile robot. IEEE Transactions on Industrial Electronics, 61(7), 3671-3681.

Xu, X., He, H. G., Hu, D. 2002. Efficient reinforcement learning using recursive least-squares methods. Journal of Artificial Intelligence Research, 16(1), 259-292.

Xu, X., Lian, C., Zuo, L., He, H. 2014. Kernel-based approximate dynamic programming for real-time online learning control: an experimental study. IEEE Transactions on Control Systems Technology, 22(1), 146-156.

Yang, Q., Jagannathan, S. 2012. Reinforcement learning controller design for affine nonlinear discrete-time systems using online approximators. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 42(2), 377-390.

Yasuda, T., Ohkura, K. 2008. A reinforcement learning technique with an adaptive action generator for a multi-robot system. In International Conference on Simulation of Adaptive Behavior (pp. 250-259). Springer Berlin Heidelberg.

Zheng, Y., Luo, S., Lv, Z. 2006, August. Control double inverted pendulum by reinforcement learning with double cmac network. In 18th International Conference on Pattern Recognition (ICPR'06) (Vol. 4, pp. 639-642). IEEE.

ÖZGEÇMİŞ

23.11.1989'da Erzurum'da doğdu. İlk ve Orta öğrenimini Kültür Kurumu İlköğretim Okulu'nda tamamladı. Liseyi Merkez Erzurum Anadolu Lisesi'nde okudu. 2007 yılında Kocaeli Üniversitesi Bilgisayar Mühendisliği Bölümü'nü kazandı. Aynı bölümden 2011 yılında mezun oldu. 2012 yılında Nanomagnetics Instruments firmasında yazılım mühendisi olarak çalıştı. Aynı yıl Hacettepe Bilişim Fakültesi Yönetim Bilişim Sistemleri bölümünde yüksek lisansa başladı. 2013 yılında Erzurum Teknik Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümü'nde araştırma görevlisi olarak göreve başladı. Aynı yıl Atatürk Üniversitesi Bilgisayar Mühendisliği Yüksek Lisans Programına başladı. Halen ETÜ'de araştırma görevlisi olarak devam etmektedir.