Kısmi Gözlemlenebilir Ardışık Karar Vermede Alt Hedef Tespiti

Program Kodu: 1001

Proje No: 215E250

Proje Yürütücüsü:

Prof. Dr. Faruk POLAT

Araştırmacı(lar):

Dr. Erkin ÇİLDEN

Bursiyer(ler):

Alper DEMİR

Hüseyin AYDIN

Nisan 2018

ANKARA

Önsöz

Bu raporda, ardışık karar verme problemlerinin kısmi gözlemlenebilir olması durumu ele alınarak bu tür problemler için otomatik alt hedef tespiti konusunda proje ekibi tarafından gerçekleştirilen faaliyetler raporlanmıştır. Markov karar süreç modelleri çerçevesinde takviye öğrenme yaklaşımı temel alınarak gerçekleştirilen araştırma faaliyetleri, öncelikle araştırma konusu ile ilgili literatürün taranması, gerekli olabilecek yöntemlerin yazılım altyapısı üzerinde kodlanması, ve sonuca yönelik araştırma adımlarının olgunlaştırılarak akademik dünya ile paylaşılması şeklinde yürütülmüştür.

Bu çalışma TÜBİTAK-ARDEB Bilimsel ve Teknolojik Araştırma Projeleri 1001 Programı (Proje No: #215E250) kapsamında gerçekleştirilmiştir.

Proje çalışmalarına başından sonuna kadar destek veren Dr.Erkin Çilden ve bursiyerlerimiz Alper Demir ve Hüseyin Aydın'a teşekkürlerimi ifade etmek isterim. Bu çalışmanın ortaya çıkma sürecinde her türlü desteği veren TÜBİTAK ARDEB-EEEAG'na da teşekkür ediyorum.

Faruk Polat
Nisan 2018

İçindekiler

Tablolar Listesi

Şekiller Listesi

ÖZET

KISMİ GÖZLEMLENEBİLİR ARDIŞIK KARAR VERMEDE ALT HEDEF TESPİTİ

Kısmi gözlemlenebilirlik durumunda ardışık karar verme, algısal aynılığın ve büyük boyutluluğun getirdiği sorunlar nedeniyle zor bir problem olarak bilinmektedir. Öğrenme algoritmaları, ardışık karar verme problemine adaptif etmen bakış açısıyla yaklaşmaya çalışır, ve bazı yaklaşıklaştırma yöntemleri kullanarak söz konusu problemle başa çıkmayı dener.

Takviye öğrenme (RL), özerk etmen modeline uyumluluğu, gerçekleştiriminin göreceli olarak kolay olması ve gerçek dünyadaki durumlara adaptasyonunun rahatlığı gibi bilinen bazı özellikleri nedeniyle, güçlü bir çevrim-içi öğrenme yöntemi olarak kabul görür. Teorik olarak Markov karar süreci (MDP) modelini temel alan RL yöntemlerinin, bazı varsayım ve kısıtlamalar çerçevesinde kısmi gözlemlenebilir MDP (POMDP) versiyonları mevcuttur.

Literatürde, MDP problemlerinin küçük alt problemlere bölünerek her bir problemin daha az eforla çözüldüğü ve bu çözümlerin sonradan birleştirilip problemin bütünü için büyük çözümün üretildiği yöntemler vardır. Bu yöntemler arasında popüler olan bir yaklaşım, problemi doğal olarak parçalara ayıran alt-hedeflerin tespitidir. Bu kapsamda MDP-RL yöntemleri için yöntemler önerilmişse de kısmi gözlemlenebilir problemler için alt-hedef tespiti konusu halen olgunluğa ulaşmamıştır.

Bu projede, POMDP-RL için alt-hedef tespiti alanında henüz hiçbir çalışma yapılmamış olan, “gizli durumlar içeren problemler için bellek tabanlı RL algoritmaları” konusunda yeni yöntemler üzerinde araştırmalar gerçekleştirilmiştir. Bu çalışma, hal-i hazırda MDP-RL için mevcut olan çevrim-içi alt-hedef tespit yöntemlerinin POMDP-RL modeline adaptasyonuna veya yeniden tasarlanmasına odaklanmakta, böylece öğrenme performansının herhangi bir çevrim-dışı müdahaleye gerek kalmaksızın artırılmasını amaçlamaktadır. Bu amaca ulaşmak için, bellek tabanlı POMDP-RL algoritması tarafından üretilen durum tahmin (veya ayrım) şeması, faydalı yaklaşık alt-hedefler üretmek için kullanılmıştır.

Öncelikle, gerek MDP-RL, gerekse POMDP-RL yöntemleri için mevcut alt-hedef tespit yaklaşımları -öğrenme çıktılarını kullanan yöntemlere ağırlık verilerek- analiz edilmiş ve çalışılmıştır. Ardından, olgun bir POMDP-RL yöntem ailesi olan bellek tabanlı algoritmalar analiz edilmiştir. Araştırmaların sonucunda, bellek yapısını temel alan bir alt-hedef tespit yöntemi geliştirilmiştir. Son olarak, literatürde yaygın kabul gören farklı problemler üzerinde karşılaştırmalı koşumlarla, önerilen yöntemlerin etkinliği doğrulanmıştır.

Anahtar Kelimeler: Pekiştirmeli Öğrenme, Kısmi Gözlemlenebilir Markov Karar Süreci, Alt-Hedef Tespiti

ABSTRACT

SUBGOAL IDENTIFICATION IN SEQUENTIAL DECISION MAKING UNDER PARTIAL OBSERVABILITY

Sequential decision making under partial observability is a hard problem mainly due to perceptual aliasing and dimensionality issues. Learning algorithms try to handle the sequential decision making problem through an adaptive agent perspective, trying to cope with the problem using some approximation methods.

Reinforcement learning (RL) is a strong on-line learning method widely known for its fitness to autonomous agent model, relatively simple implementation and ease of adaptation to real-world phenomena. Although RL methods are theoretically based on Markov decision process (MDP) model, partially observable MDP (POMDP) variants exist, together with some assumptions and limitations.

Significant effort has been spent to divide MDP problems into smaller problems, so that every sub-problem can be solved with less effort, and solutions of all sub-problems can be combined later on for the grand solution. One of the popular ways to do this is the identification of sub-goals which naturally clusters the problem into pieces. Although there are sound methods for MDP-RL case, the sub-goal identification literature for partial observable case is still immature.

The aim of this project is to attack a definitely unexplored area in terms of sub-goal identification for POMDP-RL, which is the memory based RL algorithms for problems with hidden state. This study focuses on adaptation or re-design of existing on-line sub-goal identification methods already available for MDP-RL to POMDP-RL algorithms, so that learning performance can be improved without an off-line intervention. In order to do this we rely on the state estimation (or discrimination) scheme generated by the memory based POMDP-RL algorithm, to generate approximate but useful sub-goals.

We first extensively analyze and study the existing sub-goal identification approaches for both MDP-RL and POMDP-RL, with emphasis on methods making use of learning outcomes. Then

we focus on one of the mature family of POMDP-RL algorithms, namely memory based algorithms. Since the nature of the selected POMDP-RL algorithm(s) will determine the solution method, we devise a sub-goal identification method that makes use of the used/generated memory. Finally, in order to verify effectiveness of new method(s), extensive comperative test runs are executed and reported using various different benchmark problems from the literature.

Keywords: Reinforcement Learning, Partially Observable Markov Decision Process, Sub-Goal Identificaltion

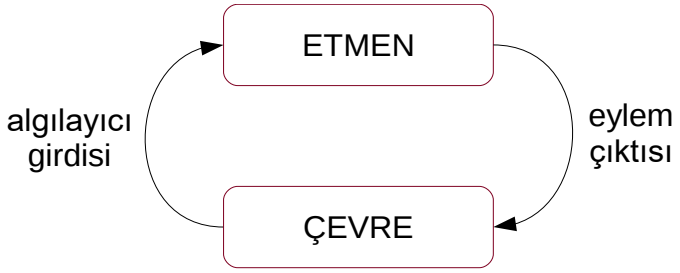
Kısaltmalar

CPU	Merkezi İşlem Ünitesi (İng., Central Processing Unit)
DD	Farklı Yoğunluk (İng., Diverse Density)
ER	Yeniden Deneyim Oynatma (İng., Experience Replay)
GCC	GNU Compiler Collection (İng.)
HSM	Hiyerarşik soyut makine
IDE	Integrated Development Environment (İng.)
KB	Kilobytes (İng.)
kNN	En yakın k komşu (İng., k-Nearest Neighbor)
L-Cut	Yerel Kesitler (İng., Local Cuts)
LoBet	Yerel Aradalık (İng., Local Betweenness)
LoRoots	Yerel Kökler (İng., Local Roots)
MDP	Markov Karar Süreci (İng., Markov Decision Process)
MI	Çoklu Örnek (İng., Multiple Instance)
NSM	En Yakın Dizi Belleği (İng., Nearest Sequence Memory)
POMDP	Kısmi Gözlemlenebilir Markov Karar Süreci (İng., Partially Observable MDP)
RL	Takviye Öğrenme (İng., Reinforcement Learning)
RN	Göreceli Yenilik (İng., Relative Novelty)
SARSA	State-Action-Reward-State-Action (İng.)
SMDP	Yarı Markov Karar Süreci (İng., Semi-Markov Decision Process)
STL	Standard Template Library (İng.)
UDM	Yararlı Ayrıştırıcı Belleği (İng., Utile Distinction Memory)
USM	Yararlı Sonek Belleği (İng., Utile Suffix Memory)
ZF	Zamansal Fark

1 GİRİŞ

Takviye öğrenme, makine öğrenme literatüründe de deneme-yanılma tabanlı yöntemlerden (öğrenme otomata, sınıflandıran sistemler, vb.) ve optimal kontrol (dinamik programlama) metotlarından temel alan, kısa ancak etkin bir tarihsel cebirsel birikime sahiptir. Takviye öğrenme problemi, yaygın olarak, öğrenen *etmen* (İng. agent) kavramı ile birlikte tarif edilir. Bunun temel nedeni, etmen modelinin, dış dünyadan doğrudan denetimin mümkün olmadığı, öğrenen birimin bilinmeyen çevre koşullarına tek başına uyum sağlaması gereken problemler için daha uygun bir model olmasıdır.

En geniş anlamıyla, (S. J. Russell ve Norvig 2003), bir etmeni *"algılayıcılarıyla bulunduğu çevreyi algılayabilen ve işleticileriyle aynı çevre üzerinde eylem yapabilen herhangi bir şey"* olarak tanımlar (Şekil 1.1).



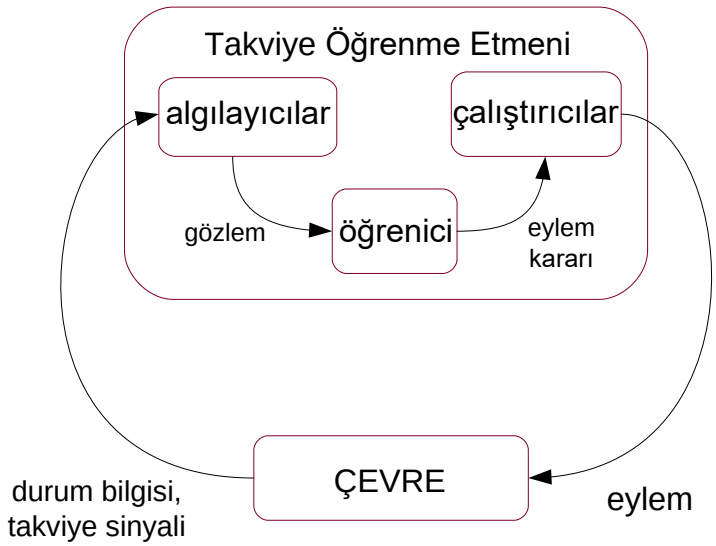
Şekil 1.1. Çevresiyle etkileşim halindeki bir etmen.

Makine öğrenme (İng. machine learning), yapay zeka alanının bir alt disiplini olarak, deneyim yoluyla akılcı davranış kalıplarının oluşturularak belli bir seviyede özerklik inşa edilmesine odaklanır. Makine öğrenmede temel bir ayrım, bu deneyimin denetimli (İng. supervised) veya denetimsiz (İng. unsupervised) bir şekilde elde edilmesi üzerinden yapılır. Denetimli öğrenmede, etmen dışarıdan bir uzman tarafından sağlanan, doğru şekilde sınıflandırılmış örnekler aracılığıyla bir davranış haritası geliştirmeye çalışır. Denetimsiz öğrenmede ise, etmen önceden sınıflandırılmış örneklere sahip olmadığından, girdilerdeki düzen ve kuralları bulmaya çalışır (Alpaydın 2004).

Takviye öğrenme, bir makine öğrenme yöntemi olarak, etmen tabanlı yaklaşıma çok uygundur ve denetimli ile denetimsiz yöntemler arasında bir noktada yer almaktadır. Takviye öğrenme çevrim içi (İng. on-line) bir yöntem olduğundan, iki önemli kabiliyet bir arada tanımlanmalıdır: deneyim yoluyla öğrenmeyi sağlayan bir yöntem (keşif, İng. exploration), ve öğrenilmiş çözümün uygulanmasını sağlayacak bir yöntem (işletme, İng. exploitation).

Takviye öğrenmede ana fikir, kendi işletici mekanizmaları (*eylem*) aracılığı ile bulunduğu ortamla (*çevre*) etkileşimde bulunan ve çevreden geri bildirim (*ödül*) alan bir öğrenici (*etmen*) kurgusudur. Genel anlamda, takviye öğrenme yöntemleri ile yapılan, etmenin gelecekte edinmeyi beklediği ödülleri optimize ederek daha *uyumlu (İng. adaptive)* hale gelmesi amacıyla, çevreden edindiği ödül (veya ceza) bilgisini temsil eden dahili bir durum-eylem işlevi tutmaktır. Daha biçimsel bir tanım yapmak gerekirse, bir takviye öğrenme algoritması, bir etmenin bir çevre içinde, edinilen ödül üzerinden tanımlı bir hedef işlevinin çıktısını maksimuma çıkarmak için gerçekleştirmesi gereken eylemlerden oluşan hareket tarzını (İng. policy) bulmaya çalışır.

Takviye öğrenme problemleri genellikle çevredeki durum (İng. state) bilgilerinin *Markov özelliğine (İng. Markov property)* sahip olduğu varsayımıyla, Markov karar süreci (İng. Markov decision process, MDP) olarak biçimlendirilir. Bir durumun Markov özelliğine sahip olması için, bir optimal hareket tarzı oluşturabilmek için gerekli tüm bilgiyi içermesi gerekir. Diğer bir deyişle, mevcut duruma bakarak verilecek bir sonraki eylem kararı, önceden deneyimlenmiş durumlardan bağımsız olarak verilebiliyor olmalıdır. Şekil 1.1 ile sunulan etmen yönelimli görünüme binaen, bir takviye öğrenme etmeninin genel dahili görünümü Şekil 1.2 ile kabaca resmedilmiştir.



Şekil 1.2. Bir takviye öğrenme etmeninin genel görünümü.

Yarı-Markov karar süreci (İng. semi-Markov decision process, SMDP) modeli, MDP modelindeki tek adımlı eylem varsayımının esnetilmesi ile oluşur. SMDP modelinde bir eylem bir yerine birden fazla ve belirsiz sayıda zaman aralıklarında devam edebilir. Takviye öğrenme yöntemleri de SMDP modelini kapsayacak şekilde genişletilebilmektedir.

Daha gerçekçi problemlerin özellikleri MDP modelindeki bazı kurallarının esnetilmesini gerektirmektedir. Kısmi gözlenebilir Markov karar süreci (İng. partially observable Markov decision process, POMDP) modelinde, durumların ve durum geçiş dinamiklerinin etmen tarafından artık tümüyle gözlemlenebilir değildir. Etmen, eksiksiz durum bilgisi yerine sınırlı *gözlem (İng. observation)* bilgisi ile donatıldığından, POMDP, takviye öğrenme algoritmaları için zor bir problem kategorisi tanımlar. Diğer bir deyişle, POMDP modelinde kesin durum bilgisi *gizlidir (İng. hidden)*. Şekil 1.2 tekrar incelenecek olursa, MDP modeli özelinde, kontrol akışındaki "gözlem" bileşeninin, kesin durum bilgisi ile aynı olduğu açıktır; ki bu durum algılayıcıların mutlak durum bilgisini eksiksiz olarak elde edebildiği gibi gerçekçi olmayan bir varsayıma dayanmaktadır. POMDP modelinde ise, gözlemin anlamsal karşılığı, etmen algılayıcılarındaki bazı sınırlamalardır ve çevrenin kısmi ve sınırlı olarak idrak edilmesini betimler.

Pek çok problem için, etmenin herhangi bir gözleme karşılık en uygun eylemi seçmeyi öğrenmesi söz konusu olduğunda, en iyi (hatta en iyiye yakın) hareket tarzının salt gözlem mantığı üzerinden elde edilmesi mümkün değildir. Bu tür "gözleme karşılık eylem" basitliğindeki hareket tarzlarına *belleksiz (İng. memoryless)* veya *tepkisel (İng. reactive)* adı verilir. Bu kategorideki yöntemler, araştırmacılar arasında epey ilgi görmüştür (Crook 2006; Jaakkola, Singh, ve Jordan 1995; Littman 1994; Loch ve Singh 1998; Pendrith ve McGarity 1998). Belleksiz takviye öğrenme yöntemlerinin bazı zorluklarını aşmak için bariz bir alternatif, dahili durum kanıları (İng. state estimations) oluşturmak amacıyla bir çeşit bellek kullanmaktır. Bu kategori, bir diğer yaygın araştırma alanını tanımlamaktadır (Chrisman 1992; Hochreiter ve Schmidhuber 1997; A. McCallum 1996a).

Tam gözlemlenebilir problemler için yürütülen takviye öğrenme araştırmalarındaki gelişmeler, öğrenme performansının bazı destekleyici yöntemlerle iyileştirilmesi için yöntemler ortaya

koymuştur. Bu yöntemlerden biri de, etmen tepkilerinin durum-eylem dizileri biçimi elde edilecek şekilde, *zamansal olarak soyutlanmasıdır (İng. temporal abstraction)*.

Pek çok problemde, öğrenen etmen aslında hiyerarşik olarak bölümlenmiş alt görevlerden müteşekkil bir görevi başarmaya çalışır. Bir alt görev, çoğu zaman çözüm uzayının farklı bölgelerinde tekrarlar, ancak etmen bu tekrarlayan durumları ayrı ayrı, yeniden keşfetmek durumunda kalır. Bu da, öğrenme performansını olumsuz olarak etkileyen, ve makul bir süre içinde en iyi çözüme yakınsamayı zorlaştıran bir durumdur. Bu anlamda, zamansal soyutlama fikrinin temel hedefi daha iyi bir çözüm bulmak değil, daha hızlı öğrenmektir.

Bu kapsamdaki bütünleşik bir bakış açısı *tercih çatısı* (İng. options framework) ile (Sutton, Precup, ve Singh 1999) tarafından tanımlanmıştır. Yaygın kabul gören bu çerçeve, MDP modeli üzerine bir SMDP inşa ederek, takviye öğrenme kuramını zamansal soyutlama eylemleri içerecek şekilde genişletir.

Zamansal soyutlama yöntemlerinin tercih çatısı ile formalize edilmesi ile önemi artan bir kavram da, alt hedef bulma yaklaşımıdır. Alt hedefler, bir MDP'yi alt problemlere bölmek için doğal bir ayraç işlevi görürler. Dahası, bulunan alt hedefler, tercih çatısındaki soyut eylemler için de doğal sonlandırma koşulları olduğundan, alt hedef tespiti tercih çatısının etkinliğini sağlamada önemli bir araç haline gelmiştir. Alt hedef tespitinin ve hemen sonrasında tercih oluşturmanın otomatik olarak öğrenme sırasında gerçekleştirildiği ve takviye öğrenme hızının kayda değer oranda hızlandığının gösterildiği çalışmalar çoktur (Menache, Mannor, ve Shimkin 2002; E. A. McGovern 2002; Şimşek 2008; Kazemitabar ve Beigy 2008; Taghizadeh ve Beigy 2013; Chen vd. 2007; Goel ve Huber 2003; Stolle ve Precup 2002; Kheradmandian ve Rahmati 2009; Mannor vd. 2004). Bu anlamda, ardışık karar verme problemlerinde etkin alt-hedef tespiti, kendi başına bir meydan okuma olarak karşımıza çıkar.

Yakın geçmişte, az sayıda araştırmacı takviye öğrenme ile POMDP problemi çözmede zamansal soyutlama tekniği kullanmayı denemiş olup, bunlar daha çok inanç tabanlı (İng., belief based) kısmi gözlemlenebilir takviye öğrenme için zamansal soyutlama konusunda öncülük etmiştir (Dung, Komeda, ve Takagi 2007; Theocharous ve Kaelbling 2004; Wiering ve Schmidhuber 1996).

Literatürde, çok basit ve genelleştirilebilirliği tartışmalı denemeler dışında (Wiering ve Schmidhuber 1996; Kamaya, Lee, ve Abe 2000), POMDP ile modellenmiş kısmi gözlemlenebilir ardışık karar verme problemlerinde otomatik olarak bir alt-hedef tespit etme amacını taşıyan

çalışma yoktur. Öngörümüz, POMDP kapsamında bellek kullanan takviye öğrenme algoritmalarının elde ettiği bilgiler kullanılarak alt-hedef tespitinin mümkün olabileceği yönündedir.

Bu çalışmada, kısmi gözlemlenebilir problemler için takviye öğrenme sırasında tamamen otomatik şekilde alt-hedef tespit edebilmek amacıyla yöntemler geliştirmesi amaçlanmıştır. Bu raporda kapsamlı bir literatür taraması, yazılım altyapısının oluşturulması, uygun bazı algoritmaların gerçekleştiriminin yapılması ve amacımız doğrultusunda yararlı olabileceğine inandığımız fikirlerin yazılım altyapımızda test edilerek sonuçlarının uluslararası akademik camia ile paylaşılması yer almaktadır. Bununla birlikte POMDP-RL için alt hedef tespitine yönelik yeni yöntemler geliştirilmiş (Raporun 6, 7, 8 ve 9. bölümlerinde yer almaktadır) olup ilgili algoritmaların performans testleri için deneysel çalışmalar tamamlanmıştır.

2 BAĞLANTILI ÇALIŞMALAR

Bu bölümde, dokümanın anlaşılırlığı ve gerekli bilgi altyapısının eksiksiz olarak tamamlanması için ilgili literatür özetlenmiştir. Konuya ilişkin hususlar vurgulanmış ve literatürdeki bağlantılı yayınlar ayrıca işaret edilmiştir. İlk önce, Markov karar süreci, yarı-Markov karar süreci ve kısmi gözlemlenebilir Markov karar süreci olmak üzere, ilgili karar süreci modelleri tanımlanmıştır. Ardından, Takviye Öğrenme yöntemi özetlenmiş, SARSA(λ), En Yakın Dizi Belleği ve Yararlı Sonek Belleği gibi, bu çalışmada kullanılmakta olan takviye öğrenme algoritmaları anlatılmıştır. Son olarak, alt-hedef bulma konusundaki literatür özetlenmiş, ve kısmi gözlemlenebilir karar süreçleri için alt-hedef tespitine yardımcı olabilecek mevcut literatür sunulmuştur.

2.1 Ardışık Karar Verme Modelleri

Eğer bir problem, dizi şeklinde sunulan çeşitli karar problemlerinden oluşuyorsa, karar verici etmenin, uygulanacak eyleme, çözümün bu dizinin gelecek adımlara etkisini de dikkate alacak şekilde karar vermesi beklenir. Ayrıca, alınacak kararın etkileri etmen tarafından önceden bilinemez, ki bu da belirsizlik faktörünü beraberinde getirir (Sigaud ve Buffet 2010).

Genellikle, bir karar sürecini, mevcut durumun yalnızca bir önceki adımdaki duruma bağımlı olduğu varsayımı ile modellemek hem daha kolaydır, hem de sezgisel olarak daha uygun kabul edilir. Diğer bir deyişle, daha önceki durumlara atıf yapmaya gerek olmaksızın, modeldeki her durum, çevrenin mevcut durumu ile ilgili tüm bilgiyi, bir önceki adımdaki durum cinsinden özetleyebilmelidir. Bu özelliğe sahip süreçler, Markov özelliğine sahip süreçler olarak anılır (Kaelbling, Littman, ve Moore 1996). Yaygın kabul görmüş bu kavram, yararlı bazı karar süreç modellerini de beraberinde getirir.

2.1.1 Markov Karar Süreçleri

Markov karar süreci (İng., Markov Decision Process, MDP), *belirsizlik içeren dizisel karar problemleri* kapsamında Markov özelliğine sahip olanlar için biçimsel bir çerçeve tanımlar ve aşağıdaki şekilde tanımlanır:

Tanım 2.1. MDP modelinin bileşenleri şunlardır:

- durum sonlu kümesi, S,
- eylem sonlu kümesi, A,

- durum geçiş işlevi, $T: S \times A \times S \rightarrow [0,1]$ ($\forall s \in S, \forall a \in A, \sum_{s' \in S} T(s,a,s') = 1$ şartlarını sağlamalıdır), ve
- ödül işlevi, $R: S \times A \rightarrow \mathcal{R}$.

$T(s,a,s')$, s durumundan s' durumuna a eylemi ile geçiş yapma olasılığını tanımlar. $R(s,a)$ ise s durumunda a eylemi gerçekleştirildiğinde doğrudan elde edilecek ödül miktarını tanımlar.

■

Durum geçiş işlevi, çevrenin bir adım sonraki durumunu, mevcut durumun ve etmenin eyleminin bir işlevi olarak olasılıksal olarak tanımlar. Ödül işlevi ise, mevcut durumun ve uygulanan eylemin bir işlevi olarak beklenen doğrudan ödül miktarını tanımlar.

Bir MDP için S kümesindeki durumları A kümesindeki eylemlere bağlayan ve tüm işlevlerin kümesine *hareket tarzı* (İng. policy) adı verilir ve π ile gösterilir:

$$\pi : s \in S \rightarrow \pi(s) \in A \tag{2.1}$$

Hareket tarzı, etmen tarafından erişilebilecek olası her durumu kapsayacak şekilde, karar sürecinin her bir adımında hangi eylemin gerçekleştirileceğini tanımlar.

Bir dizisel karar problemi MDP olarak tanımlandığında, çözüm bulmak en iyi hareket tarzını bulmakla aynı şey haline gelir. Çözüm yöntemleri, tercih edilen ödül birikimi tabanlı performans ölçütüne göre farklılık gösterse de (yani, ödül dizisinin bir işlevine bağlı olarak en iyi dizinin nasıl tanımlandığı), iyi hareket tarzları bulabilmek için yaygın olarak kullanılan iki yöntemden biri *değer yineleme* (İng. value iteration) yöntemi, diğeri ise *hareket tarzı yineleme* (İng. policy iteration) yöntemidir.

Her iki yöntem de, herhangi bir s durumu ile *toplam*, *indirimli* veya *ortalama* ölçütleri çerçevesinde oluşturulmuş bir beklenen ödül bilgisi arasında bir eşleme tanımlayan, *değer işlevi* (İng. value function) kavramına dayanmaktadır. Değer yineleme yöntemi bir MDP problemini, en iyi hareket tarzına doğru yakınsayan bir dizi işlev hesaplayarak çözmeye çalışır. Hareket tarzı yineleme yöntemi ise, bir dizi hareket tarzını giderek iyileşecek şekilde oluşturmaya odaklanır.

Bu yöntemler sayesinde, en iyi hareket tarzı arayışı sorunu, değer işlevleri cinsinden ifade edilen bir optimizasyon problemine doğrudan dönüştürülebilmektedir (Sigaud ve Buffet 2010).

2.1.2 Yarı-Markov Karar Süreçleri

MDP modeli eylem modelinin tek adımlık ayrık işlemlerden oluştuğunu varsayar. Bu varsayım zamansal olarak uzatılmış eylemler de gerçekleştirecek şekilde genişletilirse (atomik bir eylemin değişen sürelerde devam edebileceği şekilde) ortaya çıkan model *yarı-Markov karar süreci* (İng., Semi Markov Decision Process, SMDP) olarak adlandırılır.

Matematiksel olarak bir SMDP, MDP tanımını aşağıdaki şekilde genelleştirir:

Tanım 2.2. SMDP modelinin bileşenleri şunlardır:

- durum sonlu kümesi, S,
- bir eylem sonlu kümesi, A,
- durum geçiş işlevi, $T: S \times A \times S \to [0,1]$ ($\forall s \in S, \forall a \in A, \ \sum_{s' \in S} T(s,a,s') = 1$ şartlarını sağlamalıdır),
- ödül işlevi, $R: S \times A \to \mathcal{R}$, ve
- her durum-eylem ikilisi için geçiş zamanı olasılığını veren bir işlev, F.

$T(s,a,s')$, s durumundan s' durumuna a eylemi ile geçiş yapma olasılığını tanımlar. $F(t|s,a)$, s durumunda iken başlayan a eyleminin t zaman dilimi süresinde tamamlanma olasılığını tanımlar. $R(s,a)$, bir durumdan başka bir duruma geçiş sürecinde edinilmesi beklenen ödül miktarıdır, ve aşağıdaki formülle tanımlanır:

$$R(s,a) = k(s,a) + \int_0^\infty \int_0^t \rho(s,a,t) dt dF(t|s,a) \tag{2.2}$$

Burada, $k(s,a)$, s durumunda iken a eylemi uygulandığında elde edilecek sabit ödülü verir. $\rho(s,a,t)$ ise, geçişin t zaman birimi sürmesi durumundaki ödül oranıdır (Bradtke ve Duff 1994a).

■

Kolayca anlaşılacağı üzere, MDP, SMDP'nin eylem adımı 1 olan özel bir halidir.

Hemen her gerçekçi MDP probleminde, problemin yapısı, etmenin doğal olarak bir *alt hareket tarzı* alanlarına (kabiliyet veya alt-hedef olarak da adlandırılır) geçiş yaptığı şartlar içerir. Belli bir sayıdaki adımdan oluşan bu alanlarda, çözüme katkıda bulunacak yeni bir karar fırsatı bulunmaz. SMDP modelinin önemi, bu tür *yetenek* veya *soyut eylemlerin* MDP modelinin üzerine, geçiş zamanı olasılık işlevi F vasıtasıyla, inşa edilebilmesinden kaynaklanmaktadır. Genellikle soyut eylem, bir *ilkel eylem*ler (İng. primitive action) kümesi üzerinde tanımlanır. Örneğin, robotik futbol dünyasında, *pas verme* eylemi, topun istenen vuruş hizasına getirilmesi

için gereken bir dizi *topa vurma* eyleminden ve topu hedeflenen uzak noktaya gönderebilmek için gereken bir *topu ivmelendirme* eyleminden müteşekkildir (Stone, Sutton, ve Kuhlmann 2005).

2.1.3 Kısmi Gözlemlenebilir Markov Karar Süreçleri

Gerçekçi problemlerde etmen, çoğunlukla karar sürecinin o anki gerçek durumunu çıkarımsamaya yetecek bilgiye sahip değildir. Süreci kısmen gözlemleyebilir, fakat mutlak durumundan haberdar değildir. *Kısmi gözlenebilir Markov karar süreci* (POMDP), MDP modelinin genelleştirilmiş bir hali olarak bu ihtiyacı karşılamak amacıyla tasarlanmıştır.

Tanım 2.3. POMDP modelinin bileşenleri şunlardır:

- durum sonlu kümesi, S,
- eylem sonlu kümesi, A,
- durum geçiş işlevi, $T: S \times A \times S \rightarrow [0,1]$ $(\forall s \in S, \forall a \in A, \sum_{s' \in S} T(s,a,s') = 1$ şartlarını sağlamalıdır),
- ödül işlevi, $R: S \times A \rightarrow \mathcal{R}$,
- etmenin kendisini çevreleyen dünyada edinebildiği gözlemlerin sonlu kümesi, Ω, ve
- gözlem işlevi, $O : S \times A \rightarrow \mathcal{P}(\Omega)$. Gözlem işlevi, her bir eylem ve varılan durum ikilisi için gözlem kümesi üzerinde bir olasılık dağılımı tanımlar. $O(s,a,o)$, etmenin a eylemini gerçekleştirmesinin ardından s durumuna varması halinde o gözlemini edinmesi olasılığını verir.

■

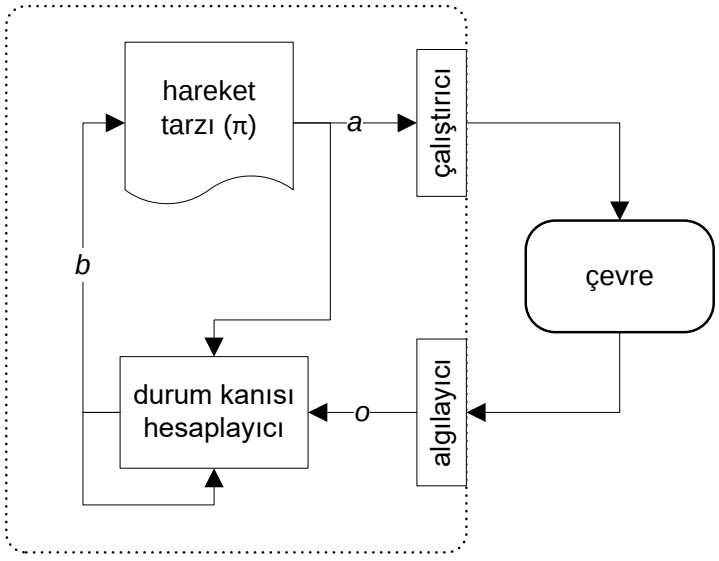
Kabaca, POMDP, etmenin o anki durumunu mutlak olarak algılayamadığı bir MDP'dir. Bunun yerine, gerçekleşen eyleme ve varılan duruma bağlı olarak bir *gözlem* bilgisi oluşturulur (Kaelbling, Littman, ve Cassandra 1998). Etmen tabanlı yaklaşım bakış açısıyla, POMDP çerçevesi, etmen için sınırlı bir algılayıcı modeli oluşturmaya olanak sağlar. Böylece, etmen çevresini yalnızca durum uzayının bazı özellikleri ile sınırlanmış haliyle, kısıtlı bir şekilde algılayabilir.

Kolayca anlaşılabileceği üzere, MDP, POMDP modelinin $\Omega = S$ ve $\forall s \in S, O(s,a,s) = 1$ şartlarını sağlayan özel bir durumudur. POMDP modelinde gözlem işlevinin esnekliği, modelin genelleştirmedeki gücünün de kaynağıdır. Ancak, bu ifade gücü, bazı sakıncaları da beraberinde getirmektedir. Model o kadar geneldir ki, bilgi içeriği son derece kısıtlı gözlem işlevleri dahi

tanımlanabilmektedir. Bu nedenle, gözlem işlevi genellikle tek başına bir POMDP problemini çözmek için yeterli olmamaktadır.

Her ne kadar etmen mutlak durumunu bilmiyor olsa da, gözlem ve eylemlerden oluşan bir tarihçe (İng. history) tutarak farkındalığını zenginleştirmesi mümkündür. Aslında, aşağıda sıralanan bilgiler, bir etmenin *eksiksiz bilgi durumu* (İng. complete information state) oluşturması, dolayısı ile problemin Markov özelliğini sağlar hale getirilmesi için yeterlidir:

- s_0 durumu için başlangıç olasılık dağılımı
- geçmiş tüm gözlemler ve şu anki gözlem $(o_0, \dots, o_t)$
- geçmiş tüm eylemler $(a_0, \dots, a_{t-1})$



Şekil 2.1. İnanç tabanlı bir POMDP etmeninin genel yapısal görünümü. Yapı, inanç durumunu hesaplayan durum kanısı oluşturucu ve hareket tarzı (π) bileşenlerinden oluşmaktadır.

Eğer etmen yukarıda sıralanan bilgilere sahipse, bilgi durum uzayı eksiksiz hale gelir ve problem bir MDP problemine dönüşür. Ancak, uygulamada, bilgi durum uzayının boyutu sürecin her zaman adımında büyür ve durum bilgisinin gösterimi ve saklanması çok hızlı bir şekilde pratik olmaktan çıkar. Dahası, sürecin belirsiz uzunlukta olduğu problemler için bu çözüm uygulanabilir olmaktan çok uzaktır.

Bu sorunun yaygın olarak kullanılan bir çözümü, Şekil 2.1 ile gösterildiği gibi, bir *inanç durumu* (İng. belief state) bilgisi tutmaktır. Öğrenen etmen, gözlem yönelimli POMDP mantığının

gerektirdiği şekilde, kararlarını gözlemlerine göre vererek uygun davranışı sergiler. Ancak, tüm geçmiş deneyimlerini temsil eden, b ile göstereceğimiz bir inanç durumu bilgisini dahili olarak tutar. Durum kanısı oluşturucu bileşeni, önceki eylem, son elde edilen gözlem ve önceki inanç durumu bilgilerini kullanarak, güncel inanç durumu bilgisini hesaplar. π, önceki gibi, hareket tarzı işlevini simgelemektedir. Fakat bu kez π, mutlak durum bilgisinin değil, inanç durum bilgisinin bir işlevidir (Kaelbling, Littman, ve Cassandra 1998).

Bu modelin, süreçte Markov özelliğini etkin olarak koruyabilmesi için, inanç durumu tanımı, sürecin kontrol mekanizmasına bağlı olarak *yeterli istatistik* (İng. sufficient statistics) sağlamalıdır. Ancak bu şekilde eksiksiz bilgi durumu gereksinimi karşılanmış olur.

İnanç durumu kavramı, etmenin geçmişini ve ilk inanç durumunu kullanarak, etkin bir şekilde yeterli istatistik oluşturmayı garanti etmektedir. Diğer bir değişle, etmenin o anki inanç durumuna ilave edilmek üzere kullanılacak herhangi bir geçmiş eylem veya gözlem verisi, mevcut durum hakkında etmene ek bir bilgi sağlamaz. Bu durum, inanç durumları üzerinden gerçekleştirilecek bir karar sürecinin Markov özelliğine sahip olmasını garanti etmektedir (Kaelbling, Littman, ve Cassandra 1998; Sigaud ve Buffet 2010).

Bir POMDP etmenindeki Şekil 2.1'de gösterildiği gibi bir hareket tarzı (π) bileşeni, mevcut inanç durumunu bir eyleme dönüştürmelidir. İnanç durumu bir yeterli istatistik olduğundan, bir POMDP çözmek, aşağıdaki özel yapıyı çözmekle aynıdır:

Tanım 2.4. İnanç-MDP modelinin bileşenleri şunlardır:

- inanç durumları kümesi (inanç durum uzayı), $\mathcal{B}$,
- eylemler kümesi, A, (Tanım 2.1)
- detayları aşağıda verilen durum geçiş işlevi, $\tau(b, a, b')$,

$$\tau(b, a, b') = Pr(b'|a, b) = \sum_{o \in \Omega} Pr(b'|a, b, o)\, Pr(o|a, b) \tag{2.3}$$

ve

$$Pr(b'|b, a, o) = \begin{cases} 1 & if\ SE(a, b, o) = b' \\ 0 & otherwise \end{cases} \tag{2.4}$$

- MDP ödül işlevi kullanılarak aşağıdaki şekilde inşa edilen ödül işlevi, $\rho(b, a)$,

$$\rho(b, a) = \sum_{s \in S} R(s, a) \tag{2.5}$$

■

Tanım 2.4 ile tanımlanan inanç-MDP için bulunacak en iyi hareket tarzı, orijinal POMDP için de en iyi hareket tarzını oluşturacaktır. Bu durumda geriye sadece bu MDP problemini kabul edilebilir bir sürede çözebilmek kalmış görünmektedir. Ancak, maalesef, inanç-MDP modelini çözmenin genel anlamda çok zor bir problem olduğu bilinmektedir (Kaelbling, Littman, ve Cassandra 1998).

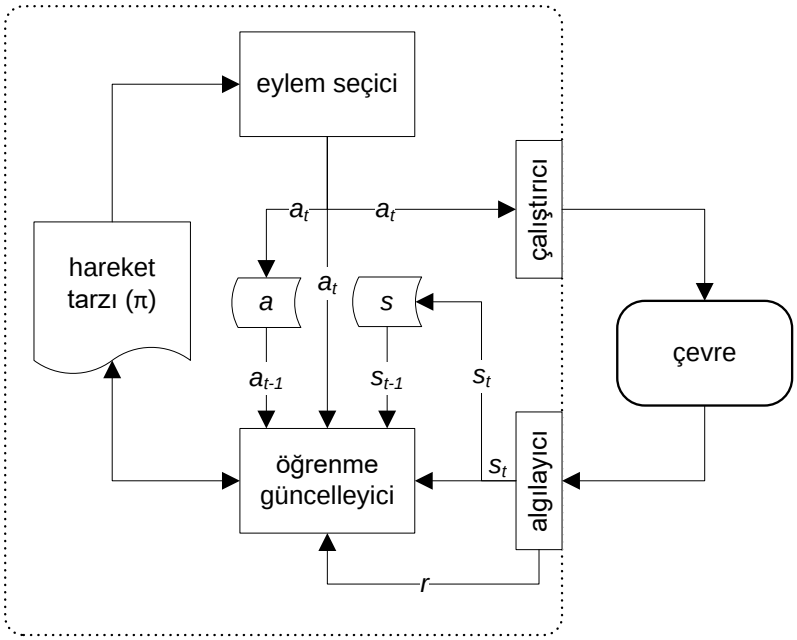
Neyse ki, inanç-MDP için tanımlanan değer işlevinin önemli bir özelliği, parçalı doğrusal ve dışbükey (İng. piecewise linear and convex) olmasıdır (Sondik 1971). Bu özellik sayesinde, değer işlevi sonlu sayıda parametre kullanılarak temsil edilebilmekte, dolayısı ile değer işlevinin bir *tutumlu gösterimi* (İng. parsimonious representation) inşa edilebilmektedir. Değer işlevinin tutumlu gösterimini inşa etmenin yaygın bir yolu, WITNESS algoritması kullanmaktır (Cassandra, Kaelbling, ve Littman 1994). Böylece, sürekli-uzay MDP modeli, etkin şekilde küçültülerek mutlak değer yineleme yöntemleri kullanılabilmektedir.

Ancak, yukarıda özetlenen kuramsal çalışmalar yalnızca küçük problemler üzerinde etkindir, ve pratik durumlara uygulanabilirlikleri sınırlıdır. Bu nedenle, POMDP için yaklaşık sonuçlar arayan algoritmalar tasarlanagelmiştir.

Makine öğrenme bakış açısıyla, dikkate değer çalışmalar inanç uzayının işlemden geçirilerek bir MDP benzerinin oluşturması fikrine dayanır. Bir çalışmasıda yaygın olarak kullanılan yöntemler özetlenmiştir (Hauskrecht 2000). Bu yöntemler arasında ızgara (İng. grid) tabanlı ara değerleme (İng. interpolation) ve dış değerleme (İng. extrapolation) yöntemleri, eğri hizalama (İng. curve fitting) yaklaşımları, ızgara tabanlı doğrusal işlev yöntemleri gibi yöntemler bulunmaktadır. Tüm bu yöntemler, bir POMDP probleminin boyutunun benzer bir MDP modeline etkin bir şekilde indirgenip, MDP öğrenme yöntemleri kullanarak iyiye yakın bir çözüm aranması prensibine dayanmaktadır.

Bir POMDP'nin MDP'ye dönüştürülmesi çabası, teorik çözüm taahhütünden kısmen ödün vermeden pratik uygulamalar açısından olası görünmemektedir. POMDP alanının bir diğer dalı, soruna farklı bir bakış açısıyla yaklaşır. Teorik çözüm taahhütü yerine "kabul edilebilir" çözümlerden söz edilmeye başlandığında, eksiksiz bilgi durumu kısıtlaması, yerini "sınırlı bellek" kullanımına bırakabilmektedir. Bir POMDP problemi, gözlemler üzerinden "yeterli" boyutta sınırlı bellek kullanılması durumunda, modeli en az bir tepkisel hareket tarzı (İng., reaktif policy) çözümü olan, algısal çakışmalar (İng., perceptual aliasing) içeren bir MDP modeline dönüştürülebilmektedir. Literatürde konuya bu yaklaşımla bakan yöntemlerin hemen hepsi,

bellek boyutunun “yeterliliğini” optimize etmeyi hedefler (A. McCallum 1996a; Loch ve Singh 1998; Lin ve Mitchell 1992; Peshkin, Meuleau, ve Kaelbling 1999; Hochreiter ve Schmidhuber 1997).



Şekil 2.2. Öğrenen etmen perspektifiyle takviye öğrenmenin genel yapısal görünümü.

2.2 Takviye Öğrenme

Takviye öğrenme (İng., Reinforcement Learning, RL) algoritmaları, hayvan ve insan bilişsel yapılarını inceleyen çalışmalardan esinlenilen basit prensiplere dayanmaktadır. Bunların başında, mevcut şartlar altında sonuçları genellikle faydalı olan eylemlerin uygulanması eğiliminde artış prensibi gelir (Sigaud ve Buffet 2010).

Makine öğrenme bakış açısıyla, takviye öğrenmenin matematiksel altyapısı, optimal kontrol alanına dayanır (Bellman 1957), ve MDP ile modellenir (Sutton ve Barto 1998).

Klasik bir takviye öğrenme algoritmasının genel yapısal görünümü Şekil 2.2 ile gösterilmiştir. Aşağı yukarı tüm takviye öğrenme algoritmaları bu yapısal şablona uyar. Bu şablonda, öğrenme işlevi (noktalı dörtgen ile çevrelenmiş alan) bir *öğrenme güncelleyici* (İng. learning update) ve *eylem seçici* (İng. action selection) modülle donatılmıştır. Bunlara ek olarak, bir *hareket tarzı* (İng. policy) veritabanı, bu modüllerin yinelemeli olarak çalıştırılması sonucunda düzenli

aralıklarla güncellenir. Öğrenme güncelleyici modül, çevreden edinilen durum (s) ve ödül (r) bilgileri ile, ve ayrıca eylem seçici modülün karar verdiği eylem (a) ile beslenir. Etmen, dış dünya ile (*çevre,* İng. environment) ***algılayıcı*** (İng. sensor) ve ***çalıştırıcı*** (İng. actuator) bileşenleri aracılığı ile etkileşir.

MDP problemlerinin Bölüm 2.1.1'de anlatılan yöntemler kullanılarak (hareket tarzı yineleme ve değer yineleme) çözülmesi, büyük ve karmaşık MDP'ler için, zaman ve bellek gereksinimlerinin geçiş matrislerini tutmak ve hesaplamakta yetersiz kalması nedeniyle, pratik olmaktan uzaktır.

Modern takviye öğrenme teknikleri, dört farklı çalışma alanının sentezi ile oluşturulmuştur: klasik dinamik programlama (İng. dynamic programming), yapay zeka alanındaki zamansal farklar yöntemi (İng. temporal differences), stokastik benzetim (İng. stochastic approximation), ve işlevsel benzetim (regresyon teknikleri, Bellman hatası yöntemi ve yapay sinir ağları) (Gosavi 2009).

Takviye öğrenme yöntemleri ile dinamik programlama yöntemlerinin farklı olduğuna karşı çıkan ve bu yöntemleri aynı MDP çözüm yaklaşımının farklı görünümleri olarak ele alan araştırmacılar olmakla birlikte, diğer bir araştırma akımı takviye öğrenmenin dinamik programlama ile makine öğrenme arasında bir yerde olduğunu, ve her iki yöntemden bazı özellikler miras aldığını söylemektedir. Ancak her iki bakış açısında da, takviye öğrenmeyi dinamik programlamadan ayırıp makine öğrenmeye yaklaştıran temek özellik, takviye öğrenme yöntemlerinde problem dinamikleri ile ilgili herhangi bir ön bilgi kullanılmamasıdır.

Diğer bir deyişle, bir takviye öğrenme etmeni altta yatan MDP modeline ait geçiş ve ödül işlevlerini öğrenme öncesinde bilmemekte, en iyi çözümü, ***değer işlevi*** (V) tahminini yinelemeli olarak iyileştirerek bulmaya çalışmaktadır. Değer işlevi, bir durumda olmanın hedefe giden yoldaki değerini verir. Alternatif olarak, Q adı verilen, bir durumdaki olası her eylemin değerini veren işlev de kullanılabilir. Esasen, takviye öğrenme, klasik MDP çözme sorununa farklı bir bakış açısıyla yaklaşmaktadır. Bu yaklaşım, robotik ve akıllı etmen sistemleri gibi, endüstriyel problem alanları açısından çok daha gerçekçidir.

Değer işlevi tahmininin gerçekleştirilmesi için kolay bir yol, π hareket tarzı ile tanımlı tüm rotalardan (İng. trajectory) toplanan ortalama toplam ödül miktarının kullanılmasıdır. $R_k(s)$, k sıralı adım üzerinde s durumundan beklenen toplam yarar olsun. Bu durumda V değer işlevinin $k+1$ sıralı adımdan sonraki ortalama tahmini şu şekildedir:

$$\forall s \in S, V_{k+1}(s) = \frac{R_1(s) + R_2(s) + \cdots + R_k(s) + R_{k+1}(s)}{k+1} \quad (2.6)$$

Tüm ödül bilgilerini saklamaktan kaçınmak amacıyla, bu işlem yinelemeli şekilde yeniden formüle edilebilir:

$$\forall s \in S, V_{k+1}(s) = V_{k+1}(s) + \frac{1}{k+1}[R_{k+1}(s) - V_k(s)] \quad (2.7)$$

$V_{k+1}(s)$ değerini elde etmek için, $V_{k+1}(s)$ ve k değerlerinin saklanması gerekmektedir. Ancak daha genel bir denklem kullanarak k değerinin saklanmasından kaçınılabilir:

$$\forall s \in S, V_{k+1}(s) = \alpha[R_{k+1}(s) - V_k(s)] \quad (2.8)$$

Bu denklemde α pozitiftir ve zamanla azalmalıdır. Bu yinelemeli formül, bir optimal (en iyi) değer işlevine yakınsar:

$$\lim_{k \to \infty} V_k(s) = V^{\pi}(s) \quad (2.9)$$

Yinelemeli tahmin formülü (Denklem (2.8)), çoğu takviye öğrenme yönteminin özünü oluşturur (Sigaud ve Buffet 2010). Takip eden kısımlarda, literatürdeki en önemli iki takviye öğrenme algoritması özetlenecektir.

2.2.1 Zamansal Fark Yöntemi

Zamansal fark (ZF, İng., Temporal Difference) fikri (Sutton, 1988), muhtemelen modern takviye öğrenme yöntemlerinin en merkezinde bulunan kavramdır. *Azaltımlı ödül* (İng. discounted reward) varsayımı altında, zamansal fark yöntemi yinelemeli tahmin formülünü (Denklem (2.8)) aşağıdaki güncelleme kuralına çevirir:

$$V(s_t) \leftarrow V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)] \quad (2.10)$$

Burada α *öğrenme hızı* (İng. learning rate), γ *azaltım hızı* (İng. discount rate) olarak adlandırılır. Bu kuralda, R ödül değişkeni artık sahnede değildir. Onun yerine, etmen $t+1$ zaman adımında r_{t+1} ödülünü kullanarak $V(s_{t+1})$ azaltımlı değer tahmini işlevinde yararlı bir güncelleme yapar. Bu güncellemeye bir de kuraldaki zamansal fark kısmı olan $-V(s_t)$ düzeltmesini ekler. Etmen artık geçilen rota üzerindeki tüm ödül bilgilerinin toplanmasını hedef duruma ulaşana kadar (veya öğrenme zamanı dolana kadar) beklemek zorunda değildir. Güncellemeler artık ziyaret edilen durumların değeri *öğrenilirken* yapılabilmektedir.

2.2.2 Q-Öğrenme Algoritması

Q-Öğrenme algoritması ZF yöntemini temel alsa da, iki önemli farkı vardır. Birincisi, durumlar yerine durum-eylem ikilileri üzerinde çalışır (yani Q işlevi). İkincisi, güncellemeleri hesaplamak için sonraki adımdaki eylemin belirlenmesine gerek yoktur. Q-Öğrenme için güncelleme kuralı şu şekildedir:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right] \quad (2.11)$$

Bu kuralda, öğrenilen eylem-değer işlevi, yani Q, takip edilen hareket tarzından bağımsız olarak, doğrudan Q^* değerine, yani optimal (en iyi) eylem-değer işlevine yakınsar.

Bu güncelleme kuralını kullanan Q-Öğrenme, takviye öğrenme araştırmalarındaki en önemli dönüm noktalarından biridir (Watkins 1989). Q-Öğrenme, basitliği nedeniyle muhtemelen en yaygın olarak kullanılan takviye öğrenme algoritmasıdır. Algoritmanın betiğimsisi (İng. pseudo-code) Algoritma 2.1 ile verilmiştir.

Sezgisel olarak, Q-Öğrenme algoritmasında Q değerlerinin tablolarda saklanması tercih edilir. Ancak bu yöntem çok fazla sayıda boyutla tanımlanmış durum uzayları için uygulanabilir değildir. Bu durumda tablolar yerine, *yapay sinir ağları* gibi işlev benzetim teknikleri kullanılabilmektedir (Lin 1991).

Q-Öğrenme

1: $Q(s, a)$ işlevine isteğe bağlı olarak ilk değerler ata
2: **Repeat**
3: s mevcut durum olsun
4: **Repeat**
5: Q işlevinin tanımladığı hareket tarzını kullanarak (ϵ-hırslı gibi bir strateji ile) s için a seç
6: a eylemini gerçekleştir, r değerini ve s' sonraki durumunu gözlemle
7: $Q(s, a) \leftarrow Q(s, a) + \alpha \left[r + \gamma \max_{a'} Q(s', a') - Q(s, a) \right]$
8: $s \leftarrow s'$
9: **until** s son durum olana kadar
10: **until** bir yakınsama kriteri gerçekleşene kadar

Algoritma 2.1 Q-Öğrenme algoritması

2.2.3 SARSA(λ) Algoritması

Q-Öğrenme (Bölüm 2.2.2) algoritmasının önemli bir varyantı, SARSA algoritmasıdır (Sutton ve Barto 1998). SARSA'da Q-Öğrenmedeki güncelleme kuralı (Denklem (2.11)) aşağıdaki şekilde yeniden düzenlenmiştir:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \quad (2.12)$$

Q-Öğrenme algoritması Q-değerlerini en iyi tahmine göre güncellerken, SARSA bu güncellemeyi gerçekleşen eylemin Q-değerine göre yapar, ve bu Q-değerinin olası en iyi eyleme ait olup olmadığı ile ilgilenmez. Bu anlamda, SARSA güncellemelerinin hareket tarzını takip ettiği (İng. on-policy) söylenir.

Klasik SARSA algoritmasının, kısmi gözlemlenebilirlik durumunda belleksiz hareket tarzları üretmedeki görece daha iyi olan performansı nedeniyle kabul gören bir biçimi, SARSA(λ) algoritmasıdır (Algoritma 2.2). *Niteliklilik takibi* adı verilen mekanizma sayesinde, eğer verilen problem için gözlemleri doğrudan eylemlerle eşleyen başarılı hareket tarzları mevcutsa, SARSA(λ) bu hareket tarzlarını etkin bir şekilde bulabilmektedir (Loch ve Singh 1998) (Rummery ve Niranjan 1994). Niteliklilik takibi mekanizması özünde öğrenme prosedürünün içindeki basit bir tarihçe takip yöntemidir. Bu yöntem, her gözlem-eylem ikilisi için, ikilinin ne kadar nitelikli olduğuna dair bir değer tutar. Niteliklilik takibi güncellemeleri, öğrenme prosedürüyle eşzamanlı olarak çalışır, ve λ ile gösterilen bir çürüme parametresi kullanır $(0 \leq \lambda \leq 1)$.

SARSA(λ) her ne kadar MDP yapısını varsayarak tasarlanmışsa da, niteliklilik takibinin belleksiz çözümler içeren POMDP problemleri için başarılı sonuçlar verdiği deneysel olarak gösterilmiştir (Loch ve Singh 1998).

$SARSA(\lambda)$

1: **Repeat**
2: $Q(s, a)$ işlevine isteğe bağlı ilk değerler, ve $e(s, a) = 0, \forall s, a$ şeklinde ilk değerler ata
3: s, a için ilk değerler ata
4: **Repeat**
5: a eylemini gerçekleştir, r ve s' gözlemle
6: Q işlevinin tanımladığı hareket tarzını kullanarak s' için a' seç (ϵ-hırslı gibi bir strateji ile)
7: $\delta \leftarrow r + \gamma Q(s', a') - Q(s, a)$

8: $e(s,a) \leftarrow e(s,a) + 1$

9: **for** all s, a **do**

10: $Q(s,a) \leftarrow Q(s,a) + \alpha\delta e(s,a)$

11: $e(s,a) \leftarrow \gamma\lambda e(s,a)$

12: **end for**

13: $s \leftarrow s'$

14: $a \leftarrow a'$

15: **until** s son durum olana kadar

16: **until** bir yakınsama kriteri gerçekleşene kadar

Algoritma 2.2 SARSA(λ) algoritması

2.3 Tercihler Çatısı

RL'nin etkili bir çevrim-içi öğrenme tekniği olduğu kanıtlanmıştır (Sutton ve Barto 1998). Temel olarak, RL, çevresel geri bildirim kullanan öğrenen bir etmenin kararlarında kendi kendini geliştirmesi ile ilgilidir. RL'deki son gelişmelerden biri, öğrenme hızını yavaşlattığı bilinen büyük durum uzayının çeşitli etkilerini azaltmaya çalışmaktadır. Olası bir çözüm, problemin yönetilebilir parçalara bölünmesi ve önce her birinin çözülmeye çalışılması, daha sonra genel sonuç elde etmek için çözümlerin bir araya getirilmesidir. Son zamanlarda, genellikle tercihler çatısında birleştirilen alt hedef bulma yöntemleri bu amaç için ön plana çıkarılmıştır.

Ardışık karar verme modeli için örtük bir varsayım, bir eylemin tek zaman adımı kadar sürdüğüdür. Bununla birlikte, bu varsayımı gevşetmek için kabul edilebilir gerçekçeler vardır. Açıktır ki, bir davranış (beceri) kalıbının problem uzayındaki farklı durumlarda yeniden kullanımı kolaylık sağlayacaktır. Bu soyutlama fikri, çeşitli araştırmacılar tarafından ön plana çıkarılmış ve buradan birkaç yaygın yaklaşım ortaya çıkmıştır (Dietterich 2000; Parr ve Russell 1998; Sutton, Precup, ve Singh 1999).

SMDP modeline dayanan bir soyutlama biçimi olan tercihler çatısı (Sutton, Precup, ve Singh 1999), bir MDP modelinin üzerine çok adımlı soyut eylemlerin dahil edilmesi yoluyla zamanlı eylemler tanımlamak ve uygulamak için bir yol sağlar. Soyut eylemlerin (tercihlerin) yaratılmasına ve serbest zamanlı adımların sonlu sayıda süren ilkel eylemleri kullanarak kullanılmasına, olanak tanır. Kısaca, bir tercih üç bileşenle tanımlanır: (1) başlatma kümesi, (2) tercihin yerel politikası ve (3) sonlandırma koşuluyla belirlenen bir olasılık dağılımı.

2.3.1 Makro-Q Öğrenme

Q-Öğrenme'nin, tercihlerin dahil edildiği doğal bir uzantısı, her ilkel eylemin değerinin tekrarlanan Q-Öğrenme (güncellenme kuralı (2.11) ile verildiği şekilde) göre yeniden güncellendiği Makro-Q Öğrenme (Mcgovern, Sutton, ve Fagg 1997) olup, bir tercihin değeri aşağıdaki kurala göre güncellenir:

$$Q(s_t, o_t) \leftarrow Q(s_t, o_t) + \alpha \times (\gamma^n \times \max_{o'} Q(s_{t+n}, o') - Q(s_t, o_t) + r_{t+1} + \gamma r_{t+2} + \ldots + \gamma^{n-1} r_{t+n}) \quad (2.13)$$

Burada, s_t, o_t tercihin başlangıç durumunu, n tercihi kullanılırken atılan adım sayısını, s_{t+n} tercihin bittiği durumu, o', s_{t+n} durumundaki en yüksek değere sahip olan tercihi ve r_{t+i}, $t+i$ zamanında alınan ödülü temsil eder. Ödül, alındığı zamana göre azaltılır.

Bununla birlikte, tercihler çatısı, tasarımcının faydalı soyutlamaları yakalamasına yardımcı olmaz. Bu nedenle, bu soyutlamaların otomatik olarak üretilmesi, kendi çeşitliliğine sahip olan ilginç bir araştırma konusudur. Yaygın olarak kullanılan yaklaşım, metodun üretilecek tercihlerin sonlanma noktaları olarak kullanılacak yapay alt hedefleri türetmek için problem alanındaki tıkanık durumları veya bölgeleri keşfetmesidir.

2.4 Kısmi Gözlemlenebilir Problemler için Bellek Tabanlı Takviye Öğrenme Yaklaşımları

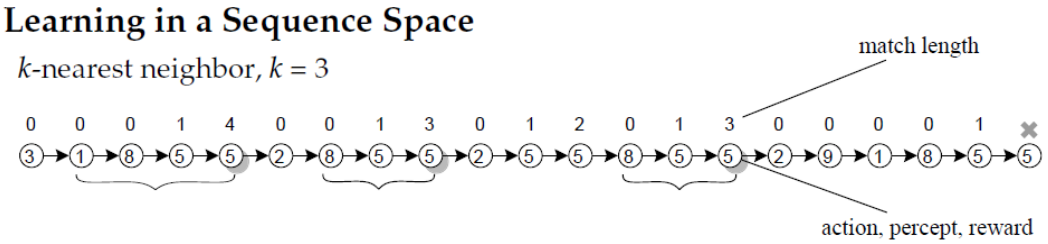
2.4.1 En Yakın Dizi Belleği Algoritması

En Yakın Dizi Belleği (İng., Nearest Sequence Memory, NSM), etmenin yapmak üzere olduğu bir eylemin faydasını hesaplamak için en yakın k komşu (İng., k-Nearest Neighbor, kNN) fikrini kullanan örneğe dayalı bir yöntemdir. NSM, sürekli alanlar üzerinde örneğe dayalı algoritmanın kalıtsal gücü ile (Atkeson 1992a; Andrew William Moore 1990; Andrew W. Moore, Atkeson, ve Schaal 1995; Schneider 1995), Takviye Öğrenme'ye yeni bir yaklaşım olarak McCallum tarafından sunulan NSM (A. McCallum 1996a), Yararlı Ayrıştırıcı Belleği (İng., Utile Distinction Memory, UDM)'nin çok yavaş öğrenme ve bir öğrenme adımından daha uzun veriye sahip kaydedilmiş deneyimin faydasını keşfetmede başarısızlık problemini çözmektedir.

NSM Algoritması'nda, etmenin mevcut durumuna en benzer geçmiş deneyimleri bulmak için kNN algoritması kullanılır. NSM Algoritması çoğu kNN uygulamasında olduğu gibi üç ana kısımdan oluşmaktadır:

- Etmenin tüm öğrenme deneyimlerini kaydetme
- Etmenin mevcut durumu ile geçmiş deneyimler arasındaki komşuluk değerini hesaplama
- Etmenin yapabileceği eylemlerin her biri için seçilen k-tane komşudan bu eylemlerin faydalılık bilgisini çıkarma

Şekil 2.3, bir dizi alanında etmenin mevcut durumu ile geçmiş deneyimler arasında nasıl bir kıyaslama yapılabileceğini göstermektedir. Eylem a, gözlem o, ödül r üçlüsünü kullanarak bakılan eşleşme uzunluğu etmenin geçmişindeki herhangi iki düğüm arasındaki komşuluk değerini belirler. Burada 5 no'lu bir düğümde a5 eylemini gerçekleştirip o5 gözlemine ve r5 ödülüne ulaştığı varsayılmıştır.



Şekil 2.3. Bir dizi alanınında öğrenme gerçekleşirken eylem, gözlem, ödül üçlüsü kullanılarak eşleşme uzunluğunu hesaplama

Mevcut durumda yapılabilecek her bir eylem için k tane en yakın komşu belirlendikten sonra, bu komşuların q değerlerinin ortalaması alınarak her bir eylemin faydalılığı hesaplanır. Daha sonra en yüksek faydaya sahip ya da ϵ-hırslı (İng., ϵ-greedy) keşif yöntemi ile rastgele bir eylem seçilir.

Birçok örneğe dayalı yöntem denetimli öğrenmeyi uygulamaktayken NSM dinamik programlama da içermektedir. NSM bir sonraki adımda yapılacak olan eylem için seçilen komşuların bu eylem için olan q değerini yine kendi ödüllerini kullanarak günceller, bu sayede etmenin daha güncel olan ödüllere karşı önyargılı olması engellenir.

2.4.2 Yararlı Sonek Belleği Algoritması

Yararlı sonek belleği (İng., Utile Suffix Memory, USM) algoritması (A. McCallum 1996a) kısmi gözlemlenebilir problemlerde modelden bağımsız öğrenmeyi sağlayan bellek tabanlı temel takviye öğrenme algoritmalarından biridir. Etmen, gözlem-eylem-ödül tarihçesinden oluşan bir veritabanı kullanarak, farklı tarihçelere sahip aynı gözlemler arasında istatistiksel mesafeleri tespit etmek suretiyle, zamanla altta yatan (gizli) durumları birbirinden ayırt etmeyi öğrenir. Bu sayede etkin bir şekilde algısal çakışma probleminin üstesinden gelebilmektedir.

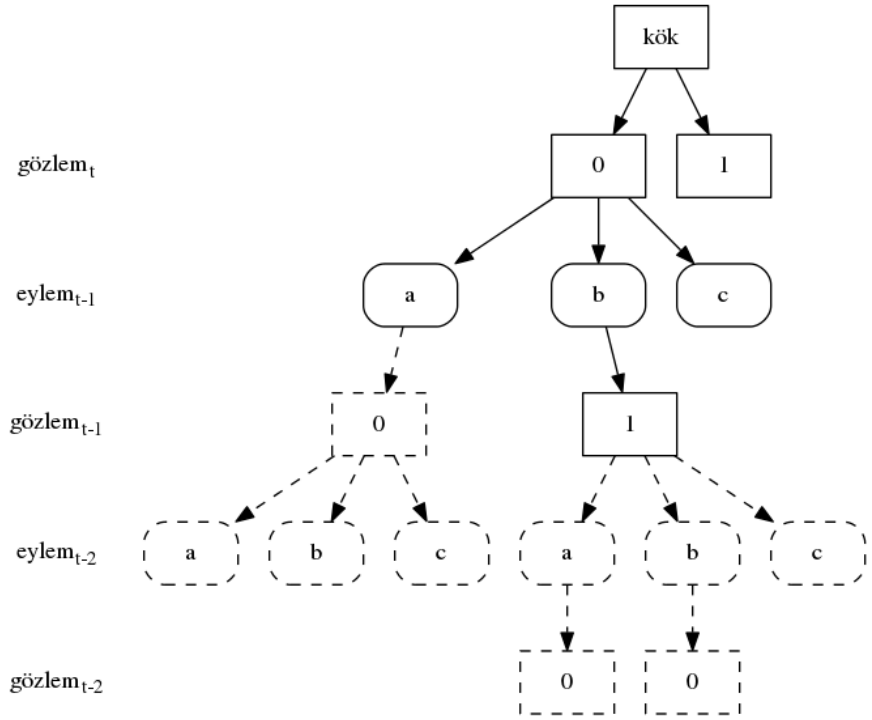
USM algoritmasının merkezinde *sonek ağacı* (İng. suffix tree) veri yapısı vardır. Sonek ağacı, ham deneyimlerden türetilmiş kısa tarihçelerden oluşan bir depo olarak kullanılır. Bu kısa tarihçelere *örnek* (İng. instance) adı verilir. Ağacın derinlik kısıtlaması yoktur. Derinlik, öğrenme süresince, algısal çakışma yaşanan durumları ayırt etmek için gerekli olduğu ölçüde, dinamik olarak arttırılır. Derinlik arttırımı, önceden belirlenmiş bir ek derinlik çerçevesinde sürekli güncellenerek takip edilen saçak düğümler (İng. fringe nodes) kullanılarak gerçekleştirilir. Bir saçak düğümün ve ilgili akrabalarının terfisi (yani normal bir yaprak düğümüne dönüştürülmesi), ayrıştırma (İng. distinction) gerekliliğinin istatistiksel testlerle ortaya çıkması ile gerçekleştirilir. Temsili bir USM ağaç veri yapısı Şekil 2.4 ile gösterilmiştir. Şekildeki ağaç, t anındaki gözlemi ayırt etmek için faydalı olabileceği tespit edilmiş kısa vadeli ($t-2$ zamanına kadarki) geçmiş örneklerinin deposudur.

Aslında, USM sonek ağacı, etmenin ham deneyimlerinin (eylem-gözlem-ödül bilgilerinin) kümelendirilmiş bir halidir. Bu kümelemede, ağacın katmanlarında derinlere indikçe, önceki gözlem ve eylemler cinsinden alternatif ayrıştırmalar yapılmaktadır. Ayrıştırmanın anlamı cinsinden üç tip düğüm vardır:

- *Dahili düğümler* eski yaprak düğümlerdir, ve artık kökten bir düğüme giden yolların tanımlanmasının dışında bir görevleri yoktur.
- *Yaprak düğümler* o anki Q tablosunu oluştururlar, çünkü her biri, bir eylem ve ayıştırıcı durum ikilisi için Q değeri tutmaktadır.
- *Saçak düğümler* potansiyel müstakbel yaprak düğümlerdir ve istatistiksel testler kullanılarak mevcut yaprak düğümlerle düzenli olarak karşılaştırılıp yeni bir ayrıştırmaya, dolayısı ile terfiye değer olup olmadıkları değerlendirilir.

YSB, gizli durumlar içeren problemler için geliştirilmiş takviye öğrenme algoritmaları arasında, belki de en etkin olanıdır. Dolayısı ile, bu çalışmada bellek tabanlı takviye öğrenme yöntem

ailesini temsilen kullanılmıştır. McCallum (1996) görece daha popüler bir algoritma olan U-Ağacı (İng. U-Tree) yöntemini de tanıtmıştır. U-Ağacı yöntemi, YSB algoritmasının gözlem ve eylem uzaylarının bileşenlerine ayrıştırıldığı, genelleştirilmiş bir halidir. Diğer bir değişle, YSB, U-Ağacı yönteminin tek boyuta indirgenmiş halidir.



Şekil 2.4. Temsili USM sonek ağacı veri yapısı. Gözlemler sayılarla, eylemler harflerle gösterilmiştir. Kesik çizgili çerçevesi olan düğümler saçak düğümlerdir.

2.5 Alt Hedef Tespit Yöntemleri

Alt hedeflerin otomatik olarak keşfedilmesi, bir MDP içindeki doğal olarak sorunu parçalayan ara noktaların veya bölgelerin belirlenmesi sorunu ile ilgilidir. Konseptin belirsizliğinden dolayı, RL bağlamında alt hedef bulma için farklı yaklaşımlar geliştirilmiştir.

Bir yaklaşım, ödül sinyalinin düzensizliklerini veya zirve noktalarını takip eder ancak takviyenin geç edinildiği problemler için uygun değildir (Digney 1998).

Bazı yöntemler, deneyim geçmişini bir geçiş çizgesine dönüştürür ve problemi bölen en uygun darboğaz bölgelerini bulmak için analiz eder (Kazemitabar ve Beigy 2008; Menache, Mannor, ve Shimkin 2002; Şimşek, Wolfe, ve Barto 2005; Taghizadeh ve Beigy 2013). Çizge temelli

yöntemlerde, deneyimleri kullanarak önce bir durum geçiş diyagramı oluşturulur. Ardından, kuvvetle bağlı bölgelerin birbirinden ayrılması için farklı teknikler birleştirilir ve aralarındaki durumlar alt hedefler olarak işaretlenir. Bu yöntemler ayrıca problemin kapsamlı bir şekilde araştırılmasını gerektirir ve temel çizge algoritmalarının yetenekleriyle sınırlıdır.

Sıklıkla ziyaret edilen durumların hedefe giden yolda bir alt hedef olma olasılığının daha yüksek olduğu gözlemine dayanan diğer bazı yöntemler sık ziyaret edilen durumları bulmak için durum ziyaret istatistiklerine dayanmaktadır (Chen vd. 2007; Goel ve Huber 2003; A. McGovern ve Barto 2001; Şimşek ve Barto 2004; Stolle ve Precup 2002). Bu yöntemler genelde probleme özgü istatistiksel parametre değerlerini ve problemin kapsamlı bir şekilde keşfedilmesini gerektirir.

Yine farklı bir yaklaşım, aynı konuyu bir kümeleme problemi olarak yorumlamakta, durum alanlarında deneyimleri kullanarak ayrı bölgeler bulmaya çalışıp, daha sonra bölgeler arasındaki erişim noktalarını alt hedef olarak tanımlamaktadır (Kheradmandian ve Rahmati 2009; Mannor vd. 2004). Açıkça alt-hedef tabanlı olmasa da, ilgili bir başka yöntem ailesi, alt hedeflerin ödül zirveleri vasıtasıyla sinyallendirildiği varsayımıyla bölüm geçmişleri üzerinde dizi analizi üzerine odaklanmaktadır (Girgin, Polat, ve Alhajj 2010; A. McGovern 1998).

Bununla birlikte, bir durumun bir alt hedef olup olmadığını kesin olarak belirlemek güçtür. İdeal durumda, doğru bir karar vermek için tam bir geçiş fonksiyonu T'ye ihtiyaç duyulur ve bu da pratik olarak mümkün değildir. Yine de, alt hedef bulma yöntemlerinin çoğu, "yaklaşık" T'nin RL deneyimi boyunca toplanabileceği varsayımına dayanmaktadır. Bu yaklaşımın bir dezavantajı, T'nin yaklaşıklığının alt hedef keşfi için kullanılacak kadar doğru olduğuna karar vermenin mümkün olmayabileceğidir. Alternatif olarak, bazı az sayıda yöntem, yerel olarak toplanan geçiş bilgisini ve alt hedef bulma için yeterliliğini istatistiksel olarak test etmenin bir yolunu bir araya getiren karma bir yaklaşım kullanmaktadır (Şimşek 2008).

Diğer yandan, kısmi gözlemlenebilir problemler için, çoğunlukla alt hedef tespiti yapılmadan zamansal soyutlama mekanizmalarına odaklanan çok az sayıda ilgili çalışma bulunmaktadır (Yoshikawa ve Kurihara 2006).

Bu kısımda, önde gelen alt hedef tespit yöntemleri açıklanmıştır.

2.5.1 Farklı Yoğunluk Yöntemi

Farklı Yoğunluk (İng., Diverse Density, DD), çoklu örnek (İng., Multiple Instance, MI) problemlerini çözmek için geliştirilmiş bir yöntemdir. Bir MI problem yapısında, pozitif ve negatif örnek torbaları bulunur. Her pozitif torbanın, hedef konseptten en az bir pozitif örnek içerdiği kabul edilir, ancak birçok olumsuz örnek içerebilir. Öte yandan her negatif torbanın yalnızca olumsuz örnekler içerdiği varsayılır.

Buradaki problem, torbalardaki örneklerin tek tek etiketlenmemiş olmasıdır, bu nedenle hedef konsepti farklı çantalar tarafından sunulan kanıtlardan öğrenmemiz gerekmektedir. (Maron ve Lozano-Pérez 1998), c_t konseptinin farklı yoğunluğunu, aşağıdaki gibi tanımlar;

$$DD(c_t) = Pr(c_t \mid B_1^+, \ldots, B_n^+ B_1^-, \ldots, B_m^-)$$

Burada, $Pr(c_t)$, t.konseptin doğru konsept olma olasılığını, B_i^+, i. pozitif torbayı ve B_i^-, i. negatif torbayı temsil eder. En yüksek DD değerine sahip kavram, bir aramanın $DD(c_t) = \prod_{1 \le i \le n} \Pr(c_t \mid B_i^+) \prod_{1 \le i \le m} \Pr(c_t \mid B_i^-)$ uzayında bir aramanın sonucudur. Bu aramayı gerçekleştirmek amacıyla, t. konsepti doğru konsept yapan etmeni aramak için aşağıdaki gürültülü-veya (İng., noisy-or) modeli önerilmiştir.

$$Pr(c_t \mid B_i^+) = 1 - \prod_{1 \le j \le p} \left(1 - Pr\left(B_{ij}^+ \in c_t\right)\right)$$

$$Pr(c_t \mid B_i^-) = 1 - \prod_{1 \le j \le n} \left(1 - Pr\left(B_{ij}^+ \in c_t\right)\right)$$

Burada, B_{ij}, i'nci torbanın j'nci örneğini, p ve n pozitif ve negatif torba sayılarını temsil eder. $Pr\left(B_{ij} \in c_t\right)$, bir örneğin konsepte olan uzaklığını kullanan bir normal dağılım olarak tanımlanmıştır.

2.5.2 Erişim Durumları Yöntemi

Otomatik alt hedef tanımlamadaki en önemli çalışmalardan biri olan (Şimşek 2008) yerel bilgi tabanlı yöntemleri önermektedir. İdeal durumda, bir durumun bir alt hedef olup olmadığına doğru bir şekilde karar vermek için geçiş fonksiyonu T'nin tamamı elde olmalıdır. Bununla birlikte, öğrenen etmenin genellikle yalnızca belirli bir süre içinde elde edilen deneyimlerden oluşan yerel bilgileri vardır. Şimsek'in çalışması, erişim durumu (İng., access state) kavramını, aralarında az

geçiş bulunduran iki veya daha fazla kendi içinde bağlı bölgeyi birbirine bağlayan bir durum olarak tanımlar. Temel fikir, bir erişim durumunu araştıran bir yöntemin, bir durumu bir hedef durum (erişim durumu) olarak sınıflandırmak için deneyim boyunca yalnızca yerel istatistiklere sahip olmasına izin verildiğidir. Bir durum için bu şekilde toplanan gözlemler, daha sonra aşağıdaki Karar Kuralı'nda kullanılır:

$$\frac{n_+}{n} > \frac{\ln\frac{1-q}{1-p}}{\ln\frac{p(1-q)}{q(1-p)}} + \frac{1}{n}\frac{\ln\left(\frac{\lambda_{fa}}{\lambda_{miss}}\frac{p(N)}{p(T)}\right)}{\ln\frac{p(1-q)}{q(1-p)}} \tag{2.14}$$

Burada n bir durum için toplam gözlem sayısı, n^+ bir durum için toplam pozitif gözlem sayısı, p bir hedef duruma (erişim durumu) verilen olumlu gözlem olasılığı, q bir hedef olmayan bir duruma verilen olumlu gözlem olasılığı, λ_{fa} yanlış alarmın maliyeti, λ_{miss} ıskalama maliyeti, $p(N)$ hedef olmayan durumların önsel olasılığı ve $p(T)$ hedef durumların önsel olasılığıdır.

Eşitsizlik tutarsa, durum erişim hali olarak sınıflandırılır. Bu karar kuralı, toplanan gözlemlerin bir durumun etiketi hakkında karar vermek için yeterli olduğu zaman adımını belirler. Bu iki seviyeli mekanizma, yöntemlerin tüm problem alanını gezmenin zaman maliyetinden kaçınmasını sağlar. Öte yandan, parametreler probleme özgüdür ve deney yoluyla tahmin edilmesi veya tahmin edilmesi gerekir.

Aynı çalışmada, üç erişim durumu keşfetme yöntemi önerilmiştir. Göreceli Yenilik (RN), bir erişim durumunun, etmenin, çokça ziyaret edilen bir bölgeden durum uzayındaki yeni bir bölgeye geçmesini sağladığı fikrini temel alan, frekans tabanlı bir alt hedef keşfetme yöntemidir. Yerel Kesitler (L-Cut), yerel etkileşim grafiğini, bloklar arası geçiş olasılığı düşük bloklara bölen, iyi bir kesit bulmayı amaçlayan grafik tabanlı bir yöntemdir. Yerel Aradalık (LoBet), aynı zamanda, grafik kuramında kullanılan bir merkeziyet ölçeği olan bir aradalık ölçüsü (Freeman 1977) kullanan grafik tabanlı algoritmadır.

Keşfedilen alt hedefler, büyük durum uzayının olumsuz etkilerini azaltmak için etkili bir şekilde kullanılmadıkça faydasızdır. Genellikle, tercihler çatısı bu amaca ulaşmak için kullanılır. Tanımlanan her bir alt hedef için bu duruma yönelik bir tercih oluşturulur. Tercih için belirlenen başlatma kümesi, her güzergahtaki alt hedef öncesinde gözlemlenen durumların eklenmesiyle oluşturulur.

Bir alt hedefe ulaşmak için bir politika üretmenin yaygın bir yolu, Yeniden Deneyim Oynatma (ER) mekanizmasıdır (Lin 1992a). ER, aracın geçmiş tecrübelerini, çevre tarafından verilen

gerçek ödül yerine yapay ödüller sağlayarak tanımlanan alt hedefe ulaşmada kullanılacak bir politika bulmak için yeniden kullanır. Genel bir yaklaşım, alt hedef duruma ulaştığında olumlu bir ödül ve diğer herhangi bir geçiş için olumsuz bir ödül sağlamaktır.

2.5.2.1 Göreceli Yenilik

Göreceli Yenilik (İng., Relative Novelty, RN), erişim durumlarının etmenin çokça ziyaret edilen bir bölgeden durum alanındaki başka bir yeni bölgeye gitmesine izin verme olasılığının yüksek olduğu fikrine odaklanmaktadır. Yöntem istatistiksel bir yaklaşım kullanmaktadır. Etmenler, ziyaret ettiği durumların geçmişini tutar ve durumlar üzerindeki göreceli yenilik değerini hesaplar.

Bir dizi durumun yeniliği, $n^{-\frac{1}{k}}$ olarak tanımlanır; burada, n, setteki durumlara ortalama ziyaret sayısıdır ve $k > 0$, yenilik katsayısı olarak adlandırılan bir parametredir. Bu parametreyle, bir setin yeniliği artan ziyaret sayısıyla azalır.

Bir durumun göreceli yeniliği, takip eden durumların yeniliğinin (kendisi de dahil) önceki durumların yeniliğine oranıdır. Bir sette dikkate alınması gereken durumların sayısı, yenilik gecikmesi (l_n) adlı bir parametreyle belirlenir.

Bir durumun, göreceli yenilik değeri, göreceli yenilik eşiği (t_{RN}) olarak adlandırılan tahmini eşikten daha büyük olduğunda göreceli yenilik yarattığı söylenir. Bir durumun değeri eşikten büyükse, algoritma durum için pozitif bir gözlem oluşturur aksi halde olumsuz bir gözlem oluşturur. Ardından Karar Kuralı eşitsizliği kontrol edilir ve durum buna göre sınıflandırılır.

2.5.2.2 Yerel Aradalık

Yerel Aradalık (İng., Local Betweenness, LoBet), Şimşek tarafından önerilen, deneyimlenmiş geçişler yoluyla türetilen yerel etkileşim grafikleri kullanan grafik tabanlı yöntemlerden biridir, yani yerel olarak toplanan bilgileri kullanılarak alt hedef tespiti yapılır. Yöntem, genellikle sosyal ağ analizinde kullanılan bir grafik aradalık ölçüsüne dayanır. Bir grafikte bir düğümün aradalığı, grafikte, olası tüm kaynakların ve hedeflerin arasındaki en kısa yolların, düğümün üzerinden geçenlerinin bütün sayıya oranı şeklinde tanımlar. Yerel aradalık, bir durumun yerel etkileşim grafiğindeki değeridir. Resmi olarak, bir düğümün aradalık değeri $\sum_{s \neq t \neq v} \frac{\sigma_{st}(v)}{\sigma_{st}} w_{st}$ şeklinde tanımlanır. Burada σ_{st}, s düğümünden t düğümüne giden kısa yolların sayısını, $\sigma_{st}(v)$ bu yolların v düğümü üzerinden geçenlerinin sayısını, w_{st} ise s düğümünden t düğümüne giden

yolun ağırlığını temsil eder. Her durum için elde edilen sonuç daha sonra Karar Kuralına (2.14) gönderilir.

2.5.2.3 Yerel Kesimler Yaklaşımı

Yerel Kesimler (İng., Local Cuts, L-Cut), (Şimşek 2008) tarafından geliştirilmiş, yerel etkileşim grafiğini (Shi ve Malik 2000) tarafından tanımlanmış normalleştirilmiş kesim (İng., normalized cut, NCut) adlı bir ölçüye göre ikiye bölmeyi amaçlayan başka bir yerel grafik tabanlı yöntemdir. Ncut şu şekilde tanımlanır: $NCut(A,B) = \frac{cut(A,B)}{vol(A)} + \frac{cut(B,A)}{vol(B)}$. Burada A ve B iki bölümdür, $cut(A,B)$, A'da başlayan ve B'de biten ayrıtların ağırlıkları toplamını ve $vol(A)$, A'da başlayan ayrıtların ağırlıkları toplamını temsil eder. Ayrıt ağırlıkları geçiş frekanslarını ile belirlenir. Algoritma, yerel etkileşim grafiğinin bölünmesi için spektral kümeleme kullanmaktadır. Ardından kesim kalitesini belirlemek için bölümlemenin Ncut değerini hesaplar. Ncut değeri, bölümlerin iyi ayrılmış olduğunu belirten önceden belirlenmiş bir kesme eşiği (t_c) değerinden düşükse, bu kesimin sınır durumları pozitif bir gözlem alırken, diğerleri olumsuz gözlem alırlar. Yine, bu gözlemler, daha fazla eleme için Karar Kuralına gönderilir.

2.5.3 Q-Kesim ve Bölünmüş Q-Kesim Yöntemleri

(Menache, Mannor, ve Shimkin 2002), tarafından önerilen Q-Kesim (Q-Cut) ve Bölünmüş Q-Kesim (Segmented Q-Cut), etmenin tüm etkileşim grafiği üzerinde maksimum akış / minimum kesim (Max-Flow/Min-Cut) yöntemini uygulayarak alt hedef bulmayı amaçlayan bütünsel (global) grafik tabanlı bir yöntemlerdir. Q-Cut etkileşim grafiği üzerine uygulandığında ve kesim kalitesi eşiğini aşan tek kesim gerçekleştirirken, Segmented Q-Cut bu grafik üzerinde kesim kalitesi korunduğu sürece bölünmüş grafik parçaları üzerinde özyinelemeli bir şekilde kesim uygulamaya devam eder.

Kesim kalitesi aşağıdaki gibi tanımlanmıştır:

$$Q[N_s, N_T] \triangleq \frac{|N_s||N_t|}{A(N_s, N_t)} \tag{2.15}$$

Buradaki N_s ve N_t kesim gerçekleştikten sonra elde edilen kaynak ve hedef bölümleri; $|N_s|$ ve $|N_t|$ bu bölümlerde kalan düğüm sayıları ve son olarak $A(N_s, N_t)$ ise bu iki bölüm arasındaki ayrıt sayısına karşılık gelmektedir. Grafikte i düğümünden j düğümüne olan ayrıtın ağırlığı i düğümünden j düğümüne olan geçiş sayısının i düğümünden olan tüm geçişlere oranı şeklinde belirlenir.

3 YAZILIM GELİŞTİRME VE MİMARİ TASARIM

Bu bölümde, proje kapsamında gerçekleştirilen yazılım geliştirme faaliyetlerinde kullanılan araçlar, yöntemler ve nihai durum özetlenmiştir.

3.1 Kullanılan Araçlar

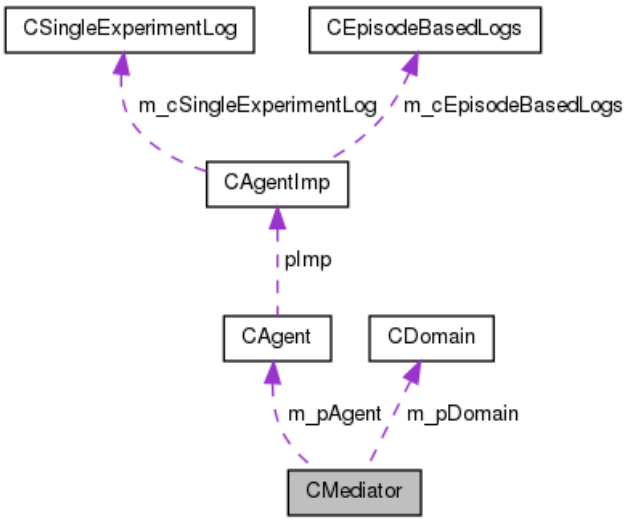
Yazılım geliştirme faaliyetleri C++ programlama dili kullanılarak gerçekleştirilmiştir. C++ diline ait standart STL kütüphaneleri ile Boost[1] kütüphanesinin küçük bir kısmı da geliştirmeye dahil edilmiştir. Geliştirme platformu olarak Linux (Ubuntu) işletim sistem üzerinde Eclipse-CDT IDE ortamı ve GNU Compiler Collection (GCC) derleyici yazılımı kullanılmıştır.

3.2 Yazılım Geliştirme Altyapısı

Proje kapsamında oluşturulacak çözümlerin en az ek eforla test edilebilmesi amacıyla, yürütücünün önceki TÜBİTAK destekli projesinde (No: 113E239) geliştirilmiş olan yazılım geliştirme altyapısı kısmen yeniden kullanılmaktadır (İng., reuse). Bu yazılım, tek parça (İng. monolythic) ve tek iş parçacığı tabanlı (İng. single threaded) olarak tasarlanmıştır.

Altyapının mimari temelinde *arabulucu (İng. mediator)*[2] ve *köprü (İng. bridge)*[3] tasarım örüntülerinin (İng. design pattern) birleştirilmiş bir yorumu yer almaktadır. Bu tasarım örüntülerinden ilki, altyapıda ihtiyaç duyulan iki temel bileşen olan etmen ve problem modülleri arasındaki bağlantıyı sağlamakta, ikincisi ise etmen çeşitliliği ihtiyacını, altyapıdaki deney oluşturma ortak kod havuzunu bozmadan ve etmen tiplerinde kod tekrarını en aza indirecek şekilde karşılamaktadır.

[1] http://www.boost.org
[2] http://en.wikipedia.org/wiki/Mediator_pattern
[3] http://en.wikipedia.org/wiki/Bridge_pattern

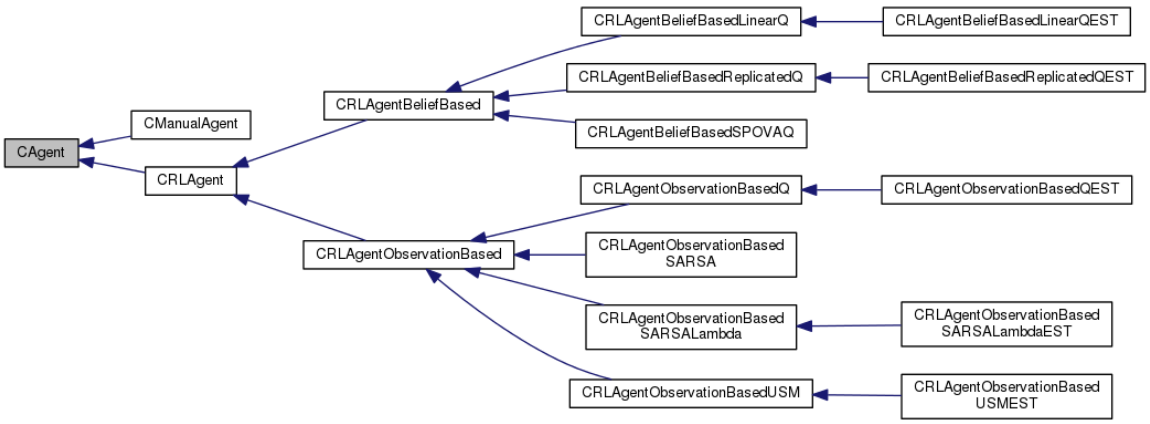


Şekil 3.1. Altyapının temelinde yer alan bağlantı sınıfına (CMediator) ait işbirliği şeması

Altyapının temelinde yer alan bağlantı sınıfına (CMediator) ait işbirliği şeması (İng. collaboration diagram) Şekil 3.1'de, detaylardan arındırılmış olarak gösterilmiştir. Bu şemadaki CMediator sınıfı, deney altyapısının temelini oluşturmaktadır. Bu sınıf bir etmen (CAgent) ile bir problem tanımını (CDomain) bir araya getirmektedir. Dahası, CMediator sınıfı yaratılış aşamasında etmenin farklı gerçekleştirimleri kullanılarak yaratılabilmektedir. Her etmen gerçekleştirimi ise, ortak seyir defteri (İng. log) sınıflarını kullanarak deneyi kaydedebilmektedirler. Bu kayıtlar deney sonrasında analiz amacıyla kullanılabilmektedir.

Bir üst seviyede, CMediator sınıfından çok sayıda kopyalayarak bir deney tekrar edilebilmekte, böylece aynı deney düzeneği için istatistiksel çokluk yaratılabilmektedir. Bu işlevi CMultiExperimenter adlı bir sınıf gerçekleştirmektedir. CMediator sınıfı iş parçacığı açısından güvenli (İng. thread safe) değildir. Dolayısı ile, olası bir çoklu iş parçacığı tasarımı CMultiExperimenter seviyesinden başlamalı, ve CMediator sınıfını bölünmez (İng. atomic) kabul etmelidir.

Örnek olarak, deney oluşturma, etmen yaratma ve problem hazırlama görevlerini çağıran CMultiExperimenter sınıfının başlatma işlevi Şekil 3.4'teki çağrı şeması (İng. call graph) ile gösterilmiştir.



Şekil 3.2. Etmen çeşitliliğini sağlayan hiyerarşiyi gösteren kalıtım şeması

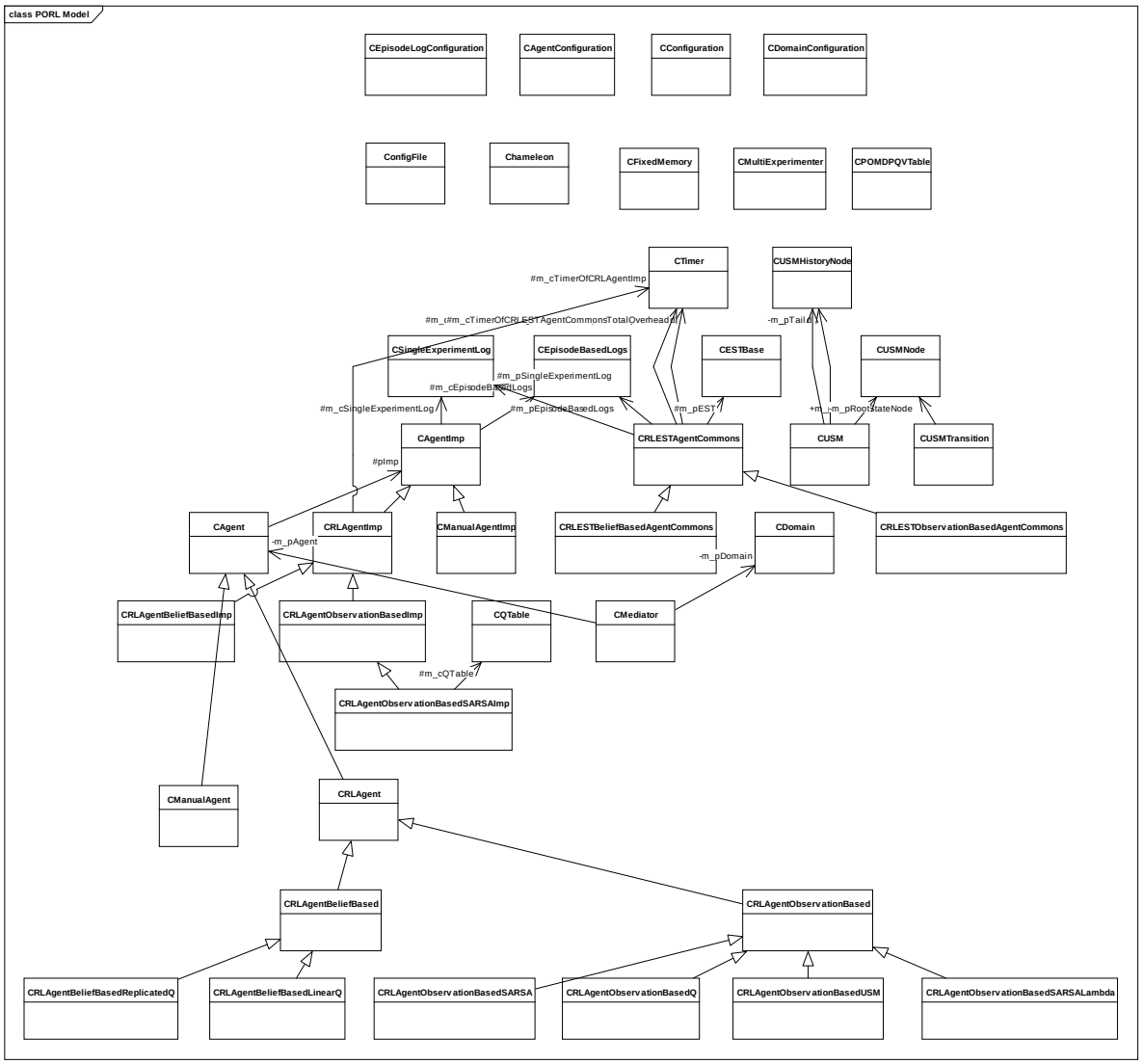
Etmen çeşitliliği kalıtım (İng. inheritence) mekanizması kullanılarak sağlanır. Şekil 3.2, bu hiyerarşinin mevcut durumunu özetleyen basitleştirilmiş bir kalıtım şeması (İng. inheritence diagram) göstermektedir. Geçekleştirimi tamamlanan takviye öğrenme etmenleri CRLAgent sınıfından türemekte, inanç tabanlı ve gözlem tabanlı olarak iki kategoriye ayrılmaktadır. Ayrıca, deneyleri manuel olarak da yaparak test etmek amacıyla, CManualAgent sınıfı da gerçekleştirilmiştir.

Altyapıda, problem çeşitliliği, Anthony R. Cassandra'nın kullandığı dosya yapısı[4] kullanılarak sağlanmıştır. Bu amaçla, ilgili dosya formatını ayrıştırarak (İng. parse) iç veri yapılarına aktaran bir kütüphane yazılmıştır. Böylece, Cassandra'nın formatı kullanılarak tasarlanmış herhangi bir problemin projede kullanılabilmesi sağlanmıştır.

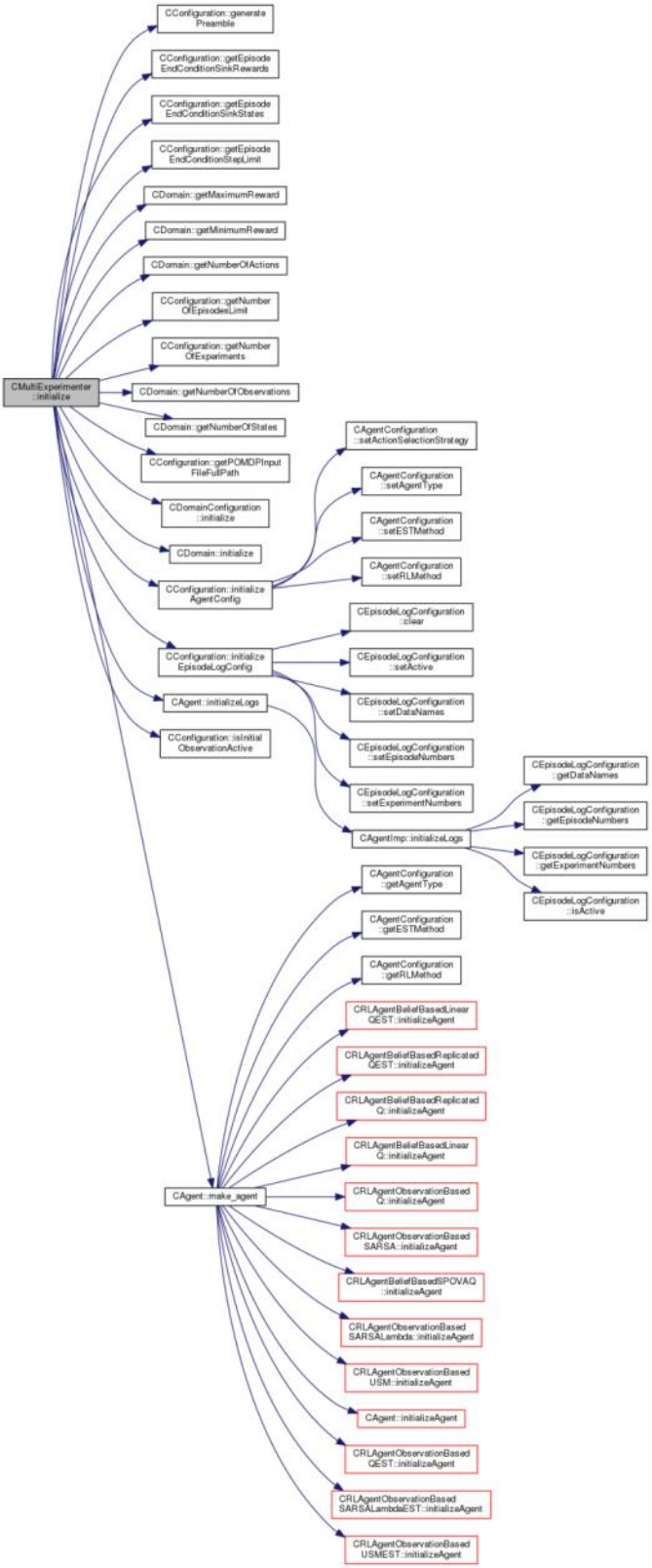
Ayrıca tekrarlayan deneylerin daha hızlı yapılabilmesi amacıyla OpenMP[5] kütüphanesi kullanılarak paralel koşum desteği eklenmiştir.

Yazılım altyapısının tamamımın genel olarak yapısını gösteren basitleştirilmiş sınıf şeması Şekil 3.3'te gösterilmiştir.

[4] http://www.pomdp.org/
[5] http://www.openmp.org/



Şekil 3.3. Deney ortamı için oluşturulan yazılımın basitleştirilmiş sınıf şeması



Şekil 3.4. CMultiExperimenter sınıfının initialize işlevi için çağrı şeması

3.3 Projede Kullanımı Öngörülen Yöntemlerin Geliştirilmesi

Proje kapsamında bellek tabanlı yöntemleri temsilen SARSA(λ), USM ve NSM algoritmaları mevcut yazılım altyapısı kullanılarak tamamlanmıştır.

Ayrıca, karşılaştırma amacıyla literatürde sık referans verilen alt-hedef tespit algoritmalarından aşağıda sıralananlar gerçekleştirilmiştir:

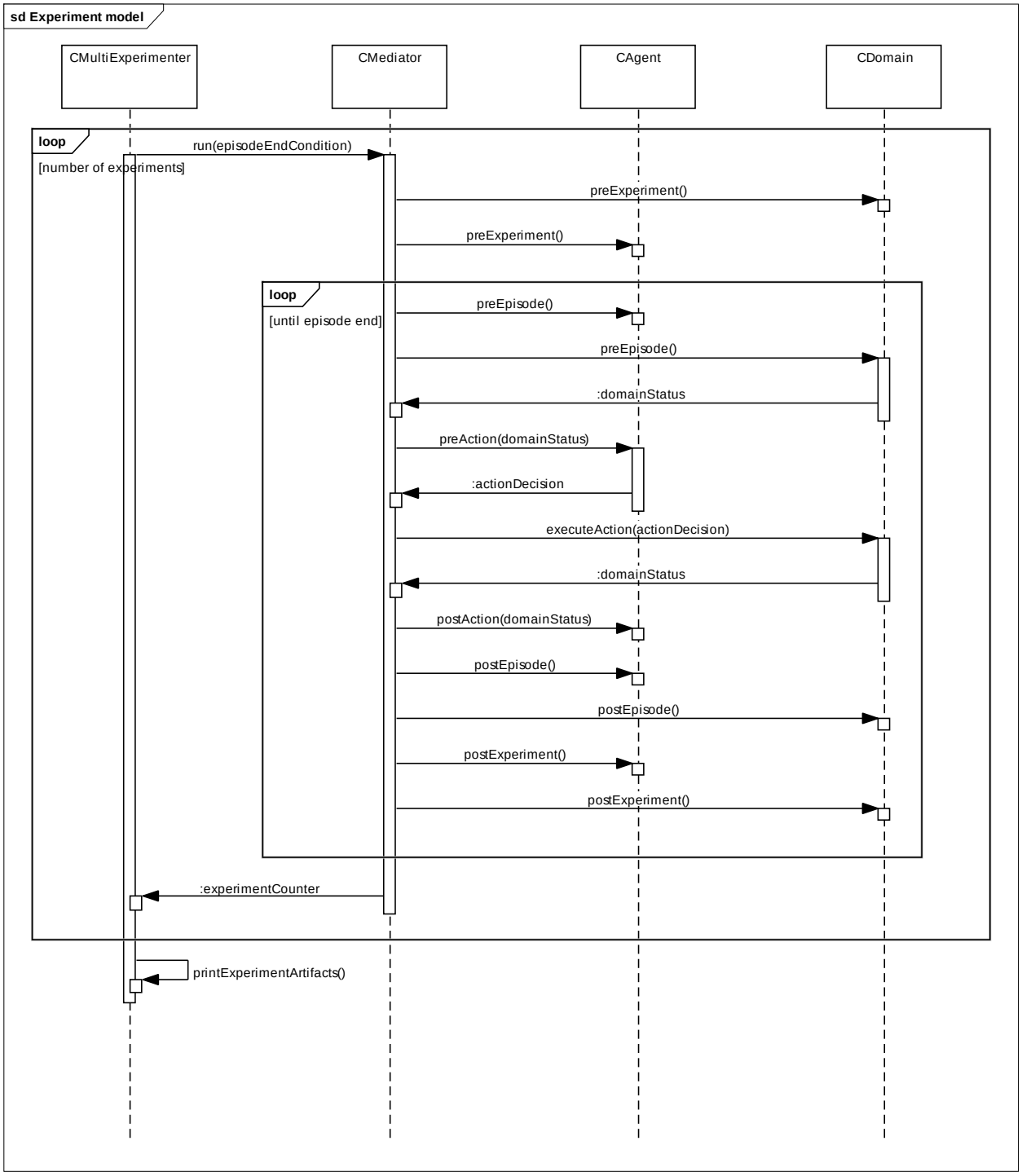
- Farklı Yoğunluk (İng., Diverse Density) (A. McGovern ve Barto 2001)
- Göreceli Yenilik (İng., Relative Novelty) (Şimşek 2008)
- Yerel Aradalık (İng., Local Betweenness) (Şimşek 2008)
- Yerel Kesimler (İng., Local Cuts) (Şimşek 2008)
- Q-Kesim (İng., Q-Cut) (Menache, Mannor, ve Shimkin 2002)
- Bölünmüş Q-Kesim (İng., Segmented Q-Cut) (Menache, Mannor, ve Shimkin 2002)

Bunların dışında, tercih çatısı kullanarak takviye öğrenme işleminin gerçekleştirilebilmesi için kullanılan Makro-Q Öğrenme (Mcgovern, Sutton, ve Fagg 1997) yöntemi de yazılım altyapısına dahil edilmiştir.

Takviye öğrenme algoritmalarının işleyişi, yazılım altyapısında, bu tip algoritmalarda sıkça gözlemlenen bir akış modeli ile tasarlanmıştır. Bu model, temel alınan üç farklı seviyedeki şu işlemlerin ayrıştırılması ile oluşturulmuştur:

- en üst seviyede, deney kümesinin hazırlanması, ve deney sonuçlarının toplanması,
- orta seviyede, bölüme (İng. episode) özel hazırlıkların yapılması ve bölüm sonuçlarının toplanması,
- en alt seviyede ise, etmenin eylem öncesi hazırlıklarının yapılması, eylemin gerçekleştirilmesi ve gerçekleşen eylemin sonuçlarının toplanarak etmen durumunun güncellenmesi.

Bu model, gerçekleştirilen tüm takviye öğrenme algoritmaları için ortaklaşa kullanılmış, en alt seviyedeki işlevlerin, gerçekleştirilen algoritmanın işleyiş tarzına göre düzenlenmesi ile ortak bir çatıda farklı varyasyonlar oluşturulması sağlanmıştır. Bu modeli basitleştirilmiş şekilde anlatan akış şeması (İng. sequence diagram) Şekil 3.5'da gösterilmiştir. Alt-hedef tespit yöntemleri, ya her eylem sonrasında (postAction), ya da bölüm sonunda (postEpisode) çalıştırıldığından, gerçekleştirilen yazılım akış mimarisi bu eklentileri de kapsayacak şekildedir.



Şekil 3.5. Takviye öğrenme yazılım altyapısının basitleştirilmiş akış şeması

4 YEREL KÖKLER: TAKVİYE ÖĞRENMEYİ HIZLANDIRMAK İÇİN AĞAÇ TABANLI BİR ALT HEDEF BULMA YÖNTEMİ

Takviye öğrenmede alt hedef tespiti, büyük durum uzayına sahip problemleri parçalamanın etkili bir yoludur. Son zamanlarda yapılan araştırmalar, esas olarak bu tür alt hedeflerin otomatik olarak bulunmasına odaklanarak tespit sırasında toplanan durum geçiş bilgilerini kullanmaktadır. Çoğunlukla tercihler çatısı (İng., options framework) yaklaşımı kullanılarak, tanımlanmış bir alt hedef, öğrenen etmeni hedefe giden yolda faydalı olduğu bilinen bir ara bölgeye yönlendirir. Bu bölümde, tecrübelerden türetilmiş tahmini kısa yol geçmiş parçalarının analizine dayanan ve sonrasında öğrenmeyi hızlandırmak için yararlı tercihler üretmek için kullanılan yeni bir otomatik alt hedef bulma yöntemi önerilmektedir (Demir, Çilden, ve Polat 2016b). Yöntem, mevcut benzer yöntemlere kıyasla, çözüm niteliğinden ödün vermeden, tanımlı alt hedeflerin tespitinde zaman karmaşıklığı ve kullanışlılığı açısından çok daha iyi performans sergilemektedir. Metodun etkinliği, iyi bilinen alt hedef bulma yöntemlerine kıyaslanarak, yaygın kabul gören farklı problemler üzerinde deneylerle gösterilmiştir.

4.1 Motivasyon

Alt hedef tespiti, RL'de ölçeklenebilirlik sorununun üstesinden gelmenin önemli bir yoludur. Problem için bir alt hedef, onu alt problemlere ayırmak için doğal bir ipucudur. Böylece etmen, daha küçük görevlerin öğrenilmesine odaklanabilir; bu durum öğrenilen davranışın benzer bir probleme aktarılması gibi fırsatlar doğurur ve daha da önemlisi, öğrenme performansını arttırır.

Alt hedef tespiti, çoğu zaman bir zamansal soyutlama (İng., temporal abstraction) mekanizması ile birleştirilir. Bu mekanizma ile, tanımlanan durum, etmenin çözmeye çalıştığı problem bölümü için yapay bir hedef görevi görür. Yaygın olarak kabul edilmiş geçici bir soyutlama biçimi, tercihler çatısıdır (Sutton, Precup, ve Singh 1999). Esasen durumlar üzerinde tanımlı ilkel eylemlerden oluşan soyut bir eylem olan bir tercih, öğrenme performansını artırma potansiyeline sahip olduğunu varsayılarak yararlı bir ara durum yolunu takip ettirerek öğrenen etmenin nasıl yönlendirileceğini tanımlar. Bununla birlikte, bu çatı, ara bir durumun nihai hedefe giden yolda yararlı olduğuna nasıl karar vereceği ile ilgilenmez. Bu gereklilik, alt hedef bulma teknikleri ile etkili bir şekilde yerine getirilebilir.

RL'de alt hedef keşfi sorununa odaklanan farklı yaklaşımlar vardır. Bazı yöntemler çizge teorisine (İng., graph theory) dayanmakta (Kazemitabar ve Beigy 2008; Menache, Mannor, ve Shimkin 2002; Şimşek, Wolfe, ve Barto 2005; Taghizadeh ve Beigy 2013), bazıları istatistiksel yöntemler (Chen vd. 2007; A. McGovern ve Barto 2001; Şimşek ve Barto 2004; Stolle ve Precup 2002) kullanırken, bazıları veri madenciliği yaklaşımına dayanır (Kheradmandian ve Rahmati 2009; Mannor vd. 2004).

RL'nin asıl odak noktası öğrenme sırasındaki performans olduğundan, alt hedeflerin tanımlanmasının altta yatan öğrenme yöntemiyle uyumlu olmasını beklemek mantıklıdır. Bazı yöntemler bu yaklaşımı doğal olarak desteklemektedir (Girgin, Polat, ve Alhajj 2010; Şimşek 2008; Şimşek ve Barto 2004), bazıları ise ek kurgu gerektirebilir.

Bu kısımda, sıra ağacı kavramını temel alan, bölüm (İng., episode) geçmişi analizine dayalı bir alt hedef bulma yöntemi önerilmektedir. Yöntem her bölümün ardından ilk önce her ziyaret edilen durum için birkaç başarılı kısa yol politikası üretmeye, oluşturulan kısa yol hareket tarzlarından geçiş ağacı oluşturmaya ve ardından ağacı alt hedef durumları çıkarmak için analiz etmeye çalışır. Bu yöntem, altta yatan öğrenme algoritması ile eş zamanlı olarak çalışır ve çözüm kalitesi açısından mevcut benzer yöntemlerden daha kötü sonuç vermez. Öte yandan, yöntem daha az işlemci zamanı kullanır ve istatistiksel karar verme parametreleri dışında herhangi bir harici probleme özgü değişkene bağımlı değildir. Algoritmanın zaman karmaşıklığı, $O((log_b(n))^2)$'dir. Burada n üretilen ağaçtaki düğümlerin sayısını, b ise ağaçın ortalama dallanma katsayısıdır. En kötü senaryo, etmen, her duruma yalnızca bir kez uğradığı ve dallanma katsayısının 1 olduğu (öğrenmenin başlangıç aşamalarında oluşması pek mümkün olmayan) bir yol izlediğinde gerçekleşir. Bu durumda algoritmanın zaman karmaşıklığı, $O(n^2)$'dir.

4.2 Alt Hedef Tespiti için Yerel Kökler Yöntemi

Bu çalışmada, bir alt hedef, her bir durumdan hedef durumuna kadar bilinen kısa yol güzergahlarının bir birleşim noktası olarak görev yapan durum ya da bölge şeklinde tanımlanır. Bu tanım, aslında muhtemel bir darboğazlık göstergesidir. Bu sezgisel tanımlamayı takiben, önerilen yaklaşım başarılı güzergâh kavramına, yani sonunda ayırt edici bir ödül olan bir güzergahın varlığına dayanır.

Yerel Kökler (İng., Local Roots, LoRoots) adını verdiğimiz bu yöntem, ziyaret edilen durumlar için olumlu ve olumsuz gözlemler üretir ve onları Karar Kuralı (2.14)'e yönlendirir. Bu, alt hedef

tespitinde yerel yaklaşımlarda yaygın bir modeldir, çünkü bölüm tarihçesinden toplanan yerel bilgiler, durumların ziyaret edilme sırasına bağımlı olabilir ve bu nedenle, yüksek düzeyde bir karar filtresi olmadan gürültülü (İng., noisy) sonuçlar (özellikle yanlış pozitifler) üretebilir. Alt hedeflerin daha doğru bir şekilde ayrılması hedeflendiğinden, Karar Kuralı (2.14) ziyaret edilen her bir durum için ayrı ayrı hesaplanır.

Yerel Kökler yöntemi, her bölüm için geçiş geçmişini (İng., transition history) kaydeder (Algoritma 4.1, satır 3). Bir bölümü tamamladıktan sonra, ilk önce son geçişin o bölüm için maksimum ödül getirip getirmediğini denetler. Eğer öyleyse, bir durum üzerinden yapılan ortalama geçiş sayısını hesaplar (nt_{avg}) ve her ziyaret edilen durumdan hedef durumuna giden en iyi "ezberlenmiş" güzergahları kullanarak türetilen kısa yollarla bir ağaç oluşturur. Hedef durumu (Algoritma 4.1, 6. satır). En iyi güzergahlar, ağacın bir yaprağından köküne doğru gezerek, geçişleri en iyi değerlerle güncelleyerek hesaplanır. Bir düğümden ağacın köküne giden yol, ilgili durumdan bölümdeki son duruma en kısa yolu oluşturur. Ağaç oluşturma yöntemi, Algoritma 4.2'de verilmiştir. Ortaya çıkan ağaç, gözlemlenen bütün durumlardan çıkan ve deneyimlerden türetilen yerel geçiş grafiğine dayanan (döngüsüz) kısa yolların bir kümesidir.

Algorithm 1 *LOCAL_ROOTS*

Require: p, q, $\frac{\lambda_{fa}}{\lambda_{miss}}$, $\frac{p(N)}{p(T)}$

$o_i \leftarrow 0$, $o_i^+ \leftarrow 0$
for each episode **do**
  $h \leftarrow$ Interact with the environment ▷ record episode history
  if h ended with a peak reward **then**
    $nt_{avg} \leftarrow$ calculate average number of distinct transitions in h
    $T \leftarrow CREATE\_TREE(h)$
    $CALCULATE\_ROOTING\_FACTORS(T, nt_{avg})$ ▷ for each vertex
    for $s \in V_T$ **do**
      $o_s \leftarrow o_s + 1$
      if s is a local maximum on T **then**
        $o_s^+ \leftarrow o_s^+ + 1$
      end if
      if the decision rule is satisfied **then** ▷ use Decision Rule 3
        Classify s as a subgoal
      end if
    end for
  end if
end for

Algoritma 4.1 Yerel Kökler

Önerilen yöntemin temel fikri, doğadaki bir ağacın kök yapısına görsel olarak benzemekten ötürü *köklenme faktörü* (İng., rooting factor) olarak adlandırdığımız bir durum ölçeğinde yatmaktadır. Fikri açıklığa kavuşturmak için Şekil 4.1(a), bitişik odalar arasında geçiş yolları barındıran üç odadan oluşan örnek bir 2 boyutlu şebeke alanını ve Şekil 4.1(b), bir bölüm geçmişi kullanarak sorun için oluşturulan bir ağacı göstermektedir. Hedef durum, odanın güneydoğu köşesinde yer alan s_{76}'dır ve etmen, kuzey-batı köşesinden, yani s_0'dan başlar. Etmen, duvarlara ve odanın sınırlarına bir hareket girişimi sonrasında hareketsiz olarak kalması dışında, her adımında dört pusula yönünün herhangi birine geçebilir.

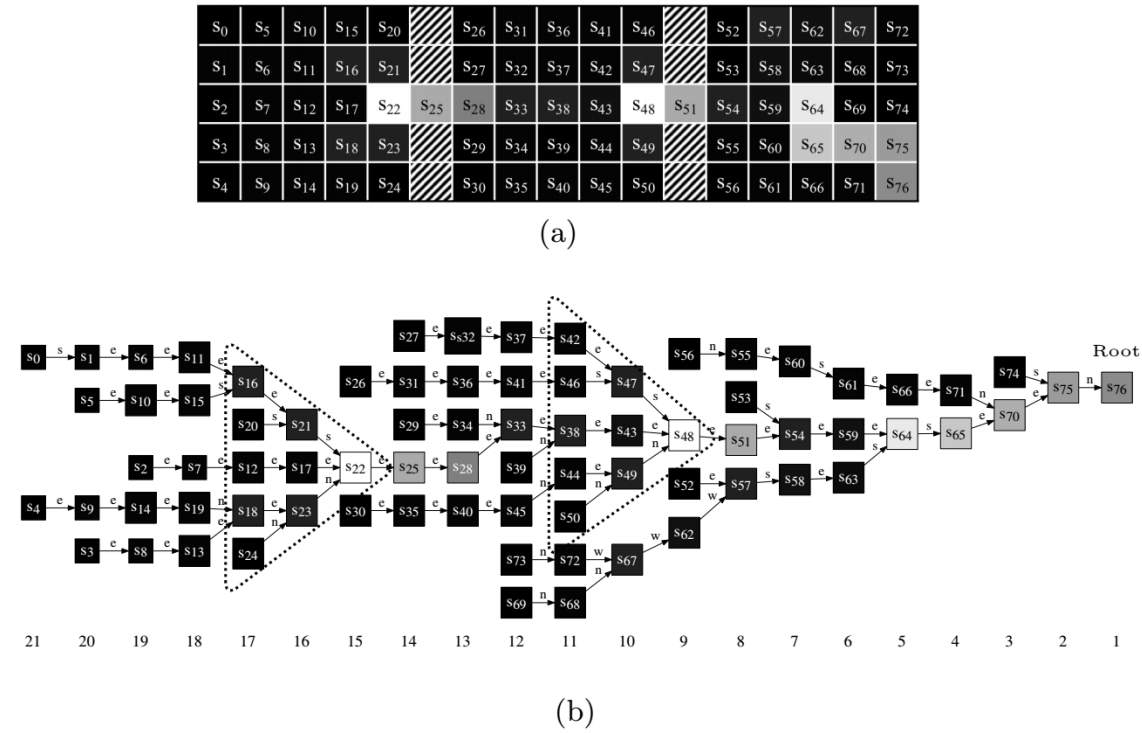
Algorithm 2 $CREATE\_TREE$

Require: a successful episode trajectory h
Ensure: a tree T representing shortcut histories to goal from each state

$t \leftarrow length(h) - 2$
$V_{s_{t+1}} \leftarrow 0$, $best_{s_{t+1}} \leftarrow null$
while $t \geq 0$ **do**
    if V_{s_t} is undefined $\vee$ $(r_{t+1} + \gamma * V_{s_{t+1}}) > V_{s_t}$ **then**
        $V_{s_t} \leftarrow r_{t+1} + (\gamma * V_{s_{t+1}})$
        $best_{s_t} \leftarrow s_{t+1}$
    end if
    $t \leftarrow t - 1$
end while
$V \leftarrow \{s_l\}, E \leftarrow \emptyset$
for each state $s \neq s_l$ **do**
    $V \leftarrow V \cup \{s\}$
    $E \leftarrow E \cup (s, best_s)$
end for
return (V, E)

Algoritma 4.2 Ağaç oluşturma

Şekil 4.1'de verilen ızgara dünyası ve ağaç örneği, siyahtan beyaza doğru ölçeklendirilen durumların köklenme faktörü değerlerine göre renklendirilmiştir ve daha parlak renk yüksek bir değer anlamına gelir. Okuyucunun dikkatini, kapıların içindeki durumların köklenme faktörü değerlerinin yüksek olduğuna çekmek isteriz. Seviye 9'da bulunan s_{48} göz önüne alındığında, köklenme faktörü ölçeği, s_{48} durumunda köklenmiş olan alt ağacın üzerine odaklanmaktadır. s_{48} durumunun köklenme faktörünü hesaplamak için öncelikle, s_{48}'in kök olduğu alt ağacın genişlikteki ilk zirve noktasına karşılık gelen en geniş seviye bulunmalıdır. Bu alt ağaçtaki her bir seviyenin genişliği sırasıyla 1,3,5,5,5,5,3 şeklindedir ve genişlik bakımından ilk tepe değeri 5'dir. Yöntem, seviye 11'i ilk seviye genişliği zirve noktası olduğu için en geniş seviye kabul eder ve seviye 17'de başlayan ikinci tepe noktasını göz ardı eder. Böylece, aşağıda daha geniş bir seviye olsa bile, bu seviye s_{48} için dikkate alınmaz. Dolayısıyla, ağaçtaki her tıkanıklık durumu -göreceli alt ağacının en geniş seviyesi 17 olan bir başka alt hedef s_{22} için olduğu gibi- kendi ilgili alt ağacına sahiptir.



Şekil 4.1. (a) Köklenme faktörüne göre renklendirilmiş 3 odalı örnek ızgara problemi (b) Bu problemin Algoritma 4.2'ye göre oluşturulmuş ağaç yapısı

s_{48} durumu için en geniş seviyenin belirlenmesiyle, en geniş seviyedeki köşelerin tabanını oluşturduğu ve s_{48}'in üst köşesini oluşturduğu sanal bir üçgen (Şekil 4.1(b)'de kesikli çizgilerle belirtilmiş üçgen) düşünülebilir. Bu üçgenin şekli, ağaçta s_{48}'in "öneminin" bir göstergesidir. Daha geniş bir üçgen, nispeten daha fazla durum için, etmenin kök durumuna (yani hedefe) ulaşmak için s_{48}'den geçmesi gerektiğini işaret etmektedir. Öte yandan, üçgenin yüksekliği, birleştirme yollarının "ilk" bağlantı noktası olan bir durumu belirler. Bu nedenle, s_{48}'in köklenme faktörü s_{51}'den daha yüksektir.

Yukarıdaki özelliklerin matematiksel bir yorumu olarak s durumunun köklenme faktörü şu şekilde tanımlanabilir:

$$r_s = \frac{(n_{widest})^{nt_{avg}}}{d_{widest} - d_s} \tag{4.1}$$

Burada, d_s, s urumunun ağaçtaki derinliğini; nt_{avg}, ağaçtaki durumların ortalama geçiş sayılarını; n_{widest}, en geniş seviyedeki düğüm sayısını ve d_{widest} bu seviyedeki durumların ağaçtaki

derinliğini temsil eder. Bir düğümün sahip olabileceği olası bağlantıların etkisini güçlendirmek için, en geniş seviyedeki düğüm sayısı, ortalama geçiş sayısı nt_{avg} ile güçlendirilir.

Ağaç Algoritma 4.2 kullanılarak oluşturulduktan sonra, köklenme faktörü hesaplaması her düğüm için gerçekleşir (Algoritma 4.1, satır 7). Algoritma 4.3, öncelikle ağaçtaki her düğümün derinliğini bulmak için bir ağaç gezimi (İng., tree traversal) yaptırılarak, bu amaç için kullanılır. Daha sonra, her derinlikteki düğüm sayısını bulmak amacıyla, en derin durumlardan köke doğru giden bir gezim daha uygulanır (Algoritma 4.3, 3-7 arası satırlar). Bu bilgiyi kullanarak, her bir durumun köklenme faktörü, göz önüne alınan durumdan daha derin seviyelere gezilerek hesaplanabilir.

Algorithm 3 *CALCULATE_ROOTING_FACTORS*

Require: a successful history tree T, nt_{avg}
1: calculate the depth of each state in T ▷ use BFS
2: $d_{max} \leftarrow \max_{s \in V_T}(\text{depth}(s))$ ▷ find maximum depth
3: **for** every $s \in V_T$ **do**
4: $n_i(s) \leftarrow$ number of nodes at depth $i \geq \text{depth}(s)$ in the subtree rooted at s
5: **end for**
6: **for** each state s in V_T **do** ▷ rooting factor calculation for every vertex
7: $d_s \leftarrow \text{depth}(s)$
8: $i \leftarrow d_s, n_{widest} \leftarrow 1, d_{widest} \leftarrow d_s$
9: **while** $i \leq d_{max}$ and $n_i(s) \geq n_{widest}$ **do**
10: **if** $n_i(s) > n_{widest}$ **then**
11: $n_{widest} \leftarrow n_i(s)$
12: $d_{widest} \leftarrow i$
13: **end if**
14: $i \leftarrow i + 1$
15: **end while**
16: $r_s \leftarrow$ calculate the rooting factor of s using Equation 4
17: **end for**

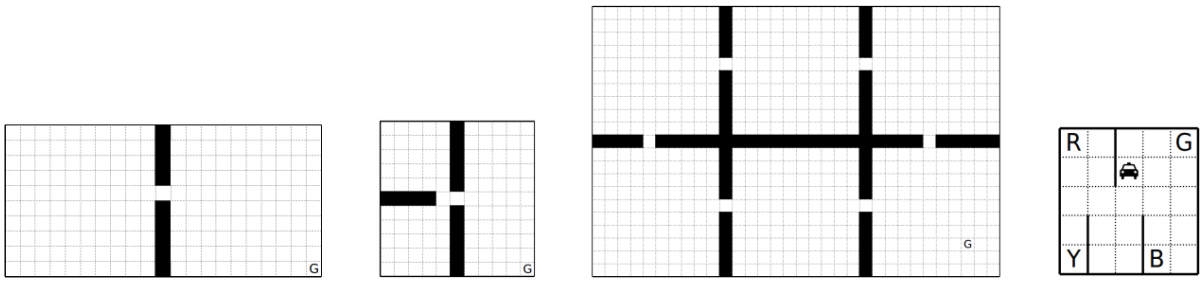
Algoritma 4.3 Köklenme faktörü hesaplama

Her ziyaret edilen durum için köklenme faktörü değerleri hesapladıktan sonra, her durumun, ağaçtaki alt komşuları ve üst komşusu arasında köklenme faktörleri açısından yerel bir maksimum olup olmadığı kontrol edilir. Söz konusu durum, maksimum tespit edildiyse olumlu bir gözlem, aksi takdirde olumsuz bir gözlem olarak etiketlenir. Tüm ağacın kökü olumlu bir gözlem

almaz, çünkü olası bir hedef durumudur ve doğal olarak bir alt hedef değildir. Her durum için yapılan gözlemler, daha ileri bir sınıflandırma amacıyla Karar Kuralı (2.14)'e gönderilir.

Yerel Kök algoritmasının en çok zaman harcayan kısmı, en kötü $O(n^2)$ zaman karmaşıklığına sahip olan her durum için köklenme faktörü değerinin hesaplandığı kısımdır. Bununla birlikte, en kötü durum ağacın doğrusal olduğu (keşif bileşeninden ötürü öğrenmenin başlangıç durumlarında genellikle mümkün olmamakla birlikte, etmen her bir durumu yalnızca bir kez ziyaret edebilir) ve dallanma faktörünün (b) 1 olduğu zamandır, yani tanımımıza göre bulunmayı gerektiren yeni bir alt hedef yoktur. Böylece, en kötü durum, oluşturulan ağacın şekli üzerine sezgisel bir kontrolle önlenebilir. Öte yandan, algoritma ortalama olarak $O((log_b(n))^2)$ zaman karmaşıklığına sahiptir; burada n ağaçdaki düğümlerin sayısıdır. Algoritma yerel bölüm güzergahlarını kullandığından, ağaçdaki düğüm sayısı (n) doğrudan tüm problem uzayındaki durum sayısıyla ilişkili değildir.

4.3 Deneyler



Şekil 4.2. Deneylerde kullanılan problemler

Algoritmalar, ilgili literatürde üç tanesi iyi bilinen kriter problemleri olan dört şebeke dünyasında test edilmiştir (Şekil 4.2). Durum ve eylem sayıları ve karşılık gelen referanslar Tablo 4.1'de verilmektedir. Yeni bir 3 rooms ızgara problemi (Şekil 4.2(b)), 3 yönlü kavşak durumunda, yöntemlerin alt hedef tanımlama davranışını araştırmak üzere tasarlanmıştır. Yerel Kökler, diğer benzer yöntemler gibi problemi bir geçiş grafiğine dönüştürür. Dolayısıyla, yöntem esasen alana özgü yapıdan bağımsızdır. Şebeke dünya alanlarında deneme yapma motivasyonu, okuyucu için darboğaz fikrini daha iyi görselleştirmektir.

Tablo 4.1. Deneylerde kullanılan parametreler

Problem	Size		Parameters used						
	$\|S\|$	$\|A\|$	Method	p	q	t_c	t_{RN}	k	l_n
2 rooms	201	4	RN	0.06	0.01	-	2.0	2	7
			L-Cut	0.3	0.01	0.05	-	-	-
			LoBet	0.7	0.07	-	-	-	-
			LoRoots	0.6	0.06	-	-	-	-
3 rooms	106	4	RN	0.05	0.01	-	2.0	2	7
			L-Cut	0.1	0.01	0.05	-	-	-
			LoBet	0.6	0.06	-	-	-	-
			LoRoots	0.6	0.06	-	-	-	-
6 rooms	605	4	RN	0.5	0.008	-	2.0	2	7
			L-Cut	0.2	0.01	0.05	-	-	-
			LoBet	0.5	0.05	-	-	-	-
			LoRoots	0.75	0.05	-	-	-	-
Taxi	500	6	RN	0.712	0.01	-	2.0	2	7
			L-Cut	0.04	0.002	0.05	-	-	-
			LoBet	0.3	0.03	-	-	-	-
			LoRoots	0.24	0.03	-	-	-	-

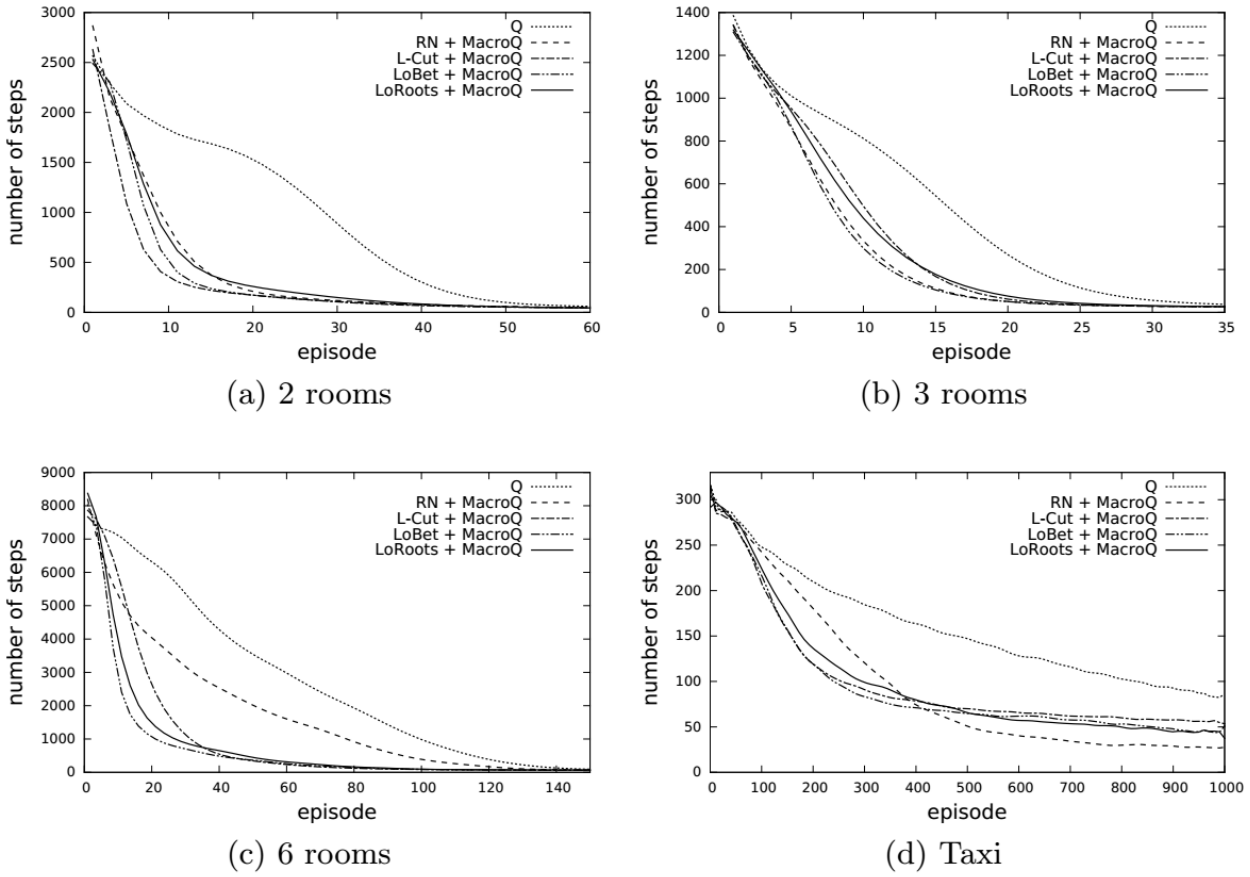
Etmen, 2 rooms, 3 rooms ve 6 rooms problemlerindede kuzey, doğu, güney ve batı olmak üzere dört hareket eylemi gerçekleştirebilir. Etmen, 0.9 ihtimalle amaçlanan yönde hareket ettiği ve 0.1 olasılıkla hareket yönlerinden herhangi birinde rastgele hareket ettiği için ortam öngörülemezdir. Hedef duruma ulaşma ödülü 1.0 olup, diğer tüm geçiş ödülleri 0'dır. 2 rooms ve 3 rooms problemlerindede, etmen sol odalardaki herhangi bir noktadan probleme başlarken 6 rooms probleminde sol üstteki odada herhangi bir noktadan başlar.

Son problem bir taksinin konumundan bir yolcu seçip onu 5x5'lik şebeke dünyasında belirli bir yere bıraktığı ünlü Taksi problemidir (Dietterich 2000) (Şekil 4.2(d), 4). Taksi etmeni 6 eylem gerçekleştirebilir: Hareket eylemleri kuzey, doğu, güney, batı; Yolcu alma eylemi ve yolcuyu bırakmak eylemi. Yolcu başlangıçta R, Y, G ve B olarak işaretlenmiş 4 farklı noktada bulunur ve yolcunun varış yeri bu dört belirlenmiş hücreden biridir. Eylemlerin gürültülü olması, etmeni 0.8 ihtimaliyle amaçlanan yönde ve 0,1 olasılıkla hedeflenen yönün soluna veya sağına hareket ettirir. Etmen, yanlış yolcu alma ve bırakma eylemleri için -10 ile cezalandırılırken doğru noktadaki yolcu bırakma eylemi için 20 ile ödüllendirilir. Diğer tüm geçişlere -1 ödül verilir.

4.3.1 Deney Düzeneği

Yerel Kökler (LoRoots) yöntemi, aynı karar kuralını kullandıkları ve öğrenme süresince kullanılabildikleri için RN, LoBet ve L-Cut yöntemleri (Bölüm 2.5.2) ile karşılaştırılmıştır. Karar kuralı parametreleri, öğrenmenin erken evrelerinde alt hedefler bulmaları ve gürültüyü uygun bir şekilde ortadan kaldırmaları için her yöntem ve problem için ayrı ayrı en iyi hallerine getirilmiştir. Karar Kuralı (2.14)'ün maliyet oranı ($\lambda_{fa}/\lambda_{miss}$) ve öncelik oranı ($p(N)/p(T)$) parametreleri tüm deneyler için 100 olarak belirlenmiştir. RN tarafından kullanılan ziyaret sayıları, her bir bölümün sonunda sıfıra eşitlenmiştir. Alt hedef tanımlama yöntemleri tarafından kullanılan kalan parametreler Tablo 4.1'de verilmiştir. Ancak, deneme yanılma haricinde doğru değerleri bulmanın pratik yolu bilinmemektedir. Özellikle, p ve q değerlerini ayarlamak için kullanılan pratik destekleyici yöntem, kullanılan alt hedef tespit yöntemlerinin çıktılarını incelemek ve elle belirlediğimiz alt hedeflere göre ayarlamak şeklinde olmuştur. Diğer parametreler çoğunlukla (Şimşek 2008)'den devralınmıştır ve Şimşek'in çalışmasında bu bakış açısıyla daha derin bir analiz bulunabilir.

Bir alt hedef tespit edildiğinde, etmen alt hedefe ulaşmak için bir tercih oluşturur. Yeni tercihin başlangıç kümesi, önceki bölümlerin her birinde alt hedefin oluşumundan önceki durumları içerir. Tercih gecikmesi (l_o), yani başlangıç kümesine eklemek için yapılan durum aramasında atılacak adım sayısı, 10 olarak belirlenmiştir. Başlangıç kümesindeki her bir durum için sonlanma olasılığı 0.0, alt hedef için 1.0 şeklinde kullanılmıştır. Tercihin hareket tarzı (İng., policy), başlangıç kümesinden çıkma için -10 ceza, alt hedefe erişme için 100 ödül ve diğer geçişler içinse -1 ceza verilerek ER aracılığıyla oluşturulmuştur. ER'ın politika öğrenme kısmı için öğrenme parametreleri olarak $\alpha = 0.125$ ve $\gamma = 0.9$ kullanılmıştır. Hızlı yakınsama için tekrar oynatma 10 kez tekrarlanmıştır.



Şekil 4.3. Yöntemlerin adım sayısı - bölüm sayısı grafikleri

Öğrenme için, bir tercihin Q değerlerinin Makro-Q öğrenme yöntemi ile ve ilkel eylemlerin Q değerlerinin ise olağan Q öğrenme yöntemi ile güncellendiği Makro-Q öğrenme algoritması kullanılmıştır. $\varepsilon = 0.1$, $\alpha = 0.05$ ve $\gamma = 0.9$ parametreleri ile, ϵ-hırslı (İng., ϵ-greedy), bir tercih seçme yöntemi olarak uygulanmıştır. Algoritma 4.2'de de aynı γ değeri kullanılmıştır. Tüm sonuçların ortalaması 200 deney üzerinden alınmıştır.

4.3.2 Sonuçlar ve Tartışma

Hedef duruma ulaşmak için atılan ortalama adım sayısı yöntemler arasında karşılaştırılmış ve sonuçlar Şekil 4.3'te gösterilmiştir. Grafikler görsel netlik için düzleştirilmiştir. LoRoots da dahil olmak üzere tüm alt hedef tespit yöntemleri, etmeni daha önce hedef duruma getirerek öğrenme hızını arttırmaktadır ve önerilen yöntemin (LoRoots) başarımı diğer yöntemlerin performansına denktir. Genel olarak, LoRoots tarafından keşfedilen alt hedefler, L-Cut, LoBet ve RN tarafından

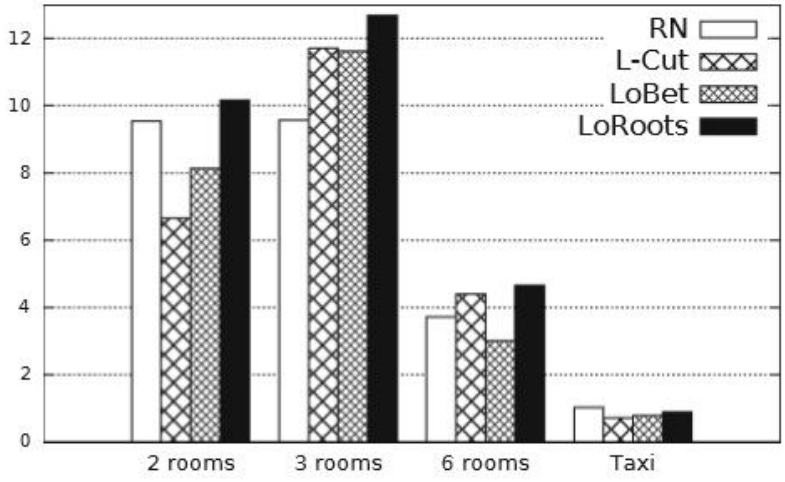
bulunanlar kadar faydalı görünmektedir. Yerel Kökler, yanı LoRoots yönteminin çözüm kalitesinin diğerlerinden daha kötü olmadığı sonucuna varabiliriz.

LoBet algoritması, yerel etkileşim grafiğinde n düğüm ve m ayrıt bulunduğu durumda, ağırlıksız ve ağırlıklı grafikler üzerinde sırasıyla $O(n.m)$ ve $O(n.m + n^2.logn)$ zaman karmaşıklığına sahipken; L-Cut algoritması ise $O(n^3)$ zaman karmaşıklığına sahiptir. Öte yandan, RN algoritması karmaşıklığı $O(1)$'dir. Önerilen yöntem olan Yerel Kökler'in zaman karmaşıklığı, bir durumun oluşturduğu ağaçta alabileceği maksimum derinliğine bağlıdır. $O((log_b(n))^2$ zaman gerektirir; burada b ortalama dallanma faktörü ve n ağaçdaki düğüm sayısıdır.

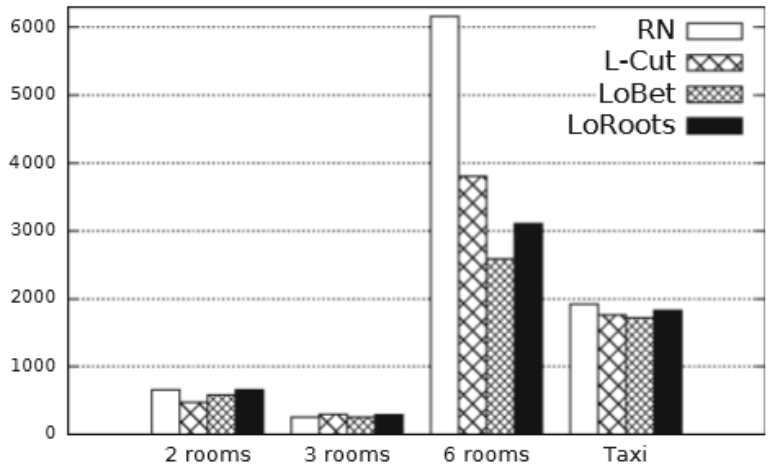
Tablo 4.2. Yöntemlerin kullandıkları süreler

Problem	**RN**	**L-Cut**	**LoBet**	**LoRoots**
2 rooms	0.63	5.18	0.65	0.45
3 rooms	0.33	1.78	0.32	0.24
6 rooms	11.00	289.30	7.10	4.08
Taxi	0.85	1.68	0.79	0.81

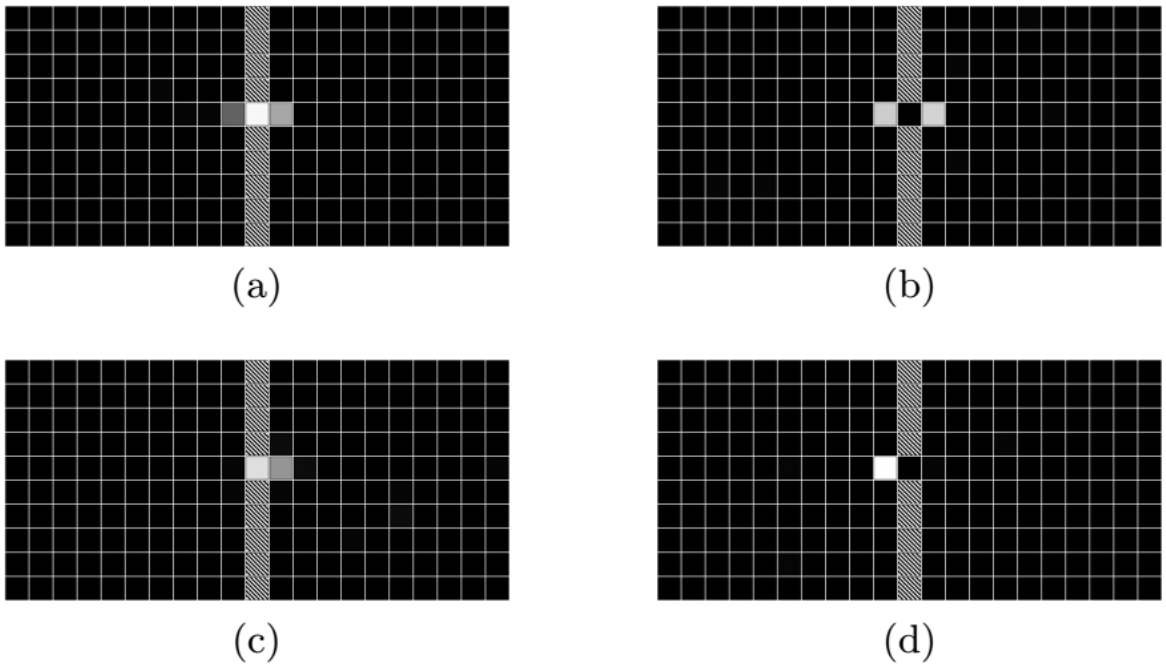
Altta yatan Makro-Q algoritmasını hariç tutan her alt hedef bulma yöntemiyle kullanılan CPU sürelerini gösteren CPU zamanı ölçümleri Tablo 4.2'de verilmiştir. Sonuçlar, LoBet ve L-Cut'ın daha yüksek zaman karmaşıklıkları nedeniyle Yerel Köklere göre çok daha fazla zamana ihtiyaç duyduğunu göstermektedir. Tek istisna, Taksi problemi için LoBet'tir; bu problem için zaman tüketimi, Yerel Köklerdeki ile neredeyse aynıdır. Öte yandan, RN'nin kullandığı zaman, zaman karmaşıklığı $O(1)$ olmasına rağmen, atılan adım sayısıyla ilişkilidir, çünkü işlem için bölüm sonunu bekleyen diğer yöntemlerin aksine, her zaman adımında çağrılır. Bir deneyin daha önceki evrelerinde gerçekleşen daha uzun bölümler, RN'nin Yerel Köklerden daha fazla zaman almasına neden olur. Tablo 4.2 Yerel Kökler yaklaşımının diğer yöntemlere kıyasla CPU zamanı açısından önemli bir avantajı olduğunu göstermektedir.



Şekil 4.4. Yöntemlerin buldukları alt hedeflerin etkinlik ortalamaları (alt hedef başına ortalama tercih takibi yüzdesi)



Şekil 4.5. Yöntemlerin bölüm başına kullandıkları ortalama hafıza boyutu (KB)



Şekil 4.6. Yöntemlerin 2 rooms probleminde buldukları alt hedefler

Şekil 4.6, belirleme sıklığı yüksek olanları parlak renklerle işaretlemiş bir şekilde, 2 rooms problemi için dört yöntemle keşfedilen alt hedefleri gösterir. L-Cut, LoBet ve RN, kapıyı da içeren birden fazla alt hedef bulur. Öte yandan LoRoots, sol odadaki durumların sağ odadaki hedef duruma ulaşması için en kısa yoldaki ilk buluşma noktası olan yalnızca kapı önündeki durumu bulur. Bu özellik, LoRoots yönteminin, özellikle 2 rooms, 3 rooms ve 6 rooms problemlerinde, diğer algoritmalara göre daha az sayıda alt hedef bulmasına neden olur. Bununla birlikte, Şekil 4.4'te görüldüğü gibi, keşfedilen alt hedeflerin etkinliği, LoRoots yönteminde diğerlerine kıyasla daha yüksektir. Bir alt hedefin etkinliği $(100 \times n_{steps(option)}/n_{steps(episode)})/n_{subgoal}$ olarak tanımlanmıştır; burada $n_{steps(option)}$ tercih süresince atılan toplam adım sayısı, $n_{steps(episode)}$ bölüm boyunca atılan toplam adım sayısı ve $n_{subgoal}$ bir bölümün sonunda tanımlanan alt hedeflerin sayısıdır. Alt-hedef etkinliği, bir alt-hedefin yararlı bir tercihi tetikleme yeteneği olarak yorumlanabilir. Diğer üç yöntemle bulunan bazı ilave alt hedeflerin katkısı genel olarak LoRoots yönteminde önerilenler kadar önemli değildir.

Son olarak, LoRoots yönteminin ortalama bellek kullanımı, diğer yöntemlerle kullanılan grafiklerden daha az sayıda ayrıt içeren bir ağaç kullandığı için, genel olarak grafik temelli

yöntemleri (ör. LoBet ve L-Cut) aşmaz. Hafıza kullanım ölçümlerine ER veritabanlarının da dahil olduğuna okuyucunun dikkatini çekmek isteriz.

Karar Kuralı (2.14) parametrelerine ek olarak, L-Cut bir, RN ise dört adet ilave parametre gerektirmektedir; LoBet ve LoRoots için ise ilave parametre ihtiyacı yoktur. Bu ilave parametreler, L-Cut ve RN tarafından bulunan alt hedeflerin kalitesini belirler ve onları problemin yapısına bağımlı hale getirir. Bu anlamda, LoBet gibi LoRoots yöntemi de, L-Cut ve RN'ye kıyasla problem özelliklerine daha az bağımlıdır. Dahası, Şekil 4.4'te görüldüğü gibi, LoRoots, LoBet'e göre alt hedef kalitesi bakımından daha iyi performans sergilemektedir.

5 TAKVİYE ÖĞRENMEDE TERCİHLER İÇİN DAHA İYİ BAŞLATMA KÜMESİ OLUŞTURMA AMAÇLI TARİHÇE AĞACI TEMELLİ SEZGİSEL BİR YÖNTEM

Tercihler çatısı, tercih (İng., option) adı verilen zamansal olarak genişletilmiş eylemler yoluyla öğrenme hızını artırmak için kullanılan önemli bir yöntemdir (Bölüm 2.3). Tercihler için yüksek kaliteli sonlandırma koşullarının nasıl türetileceğine odaklanan çeşitli çalışmalar mevcut olmakla birlikte, bir tercihin başlatma kümesinin etkisi henüz çalışılmamış bir alandır. Bu bölümde olayların yakın tarihinin bir analizi yoluyla yararlı başlatma kümesi unsurlarını türetmek için etkin bir sezgisel (İng., heuristic) yöntem önerilmektedir (Demir, Çilden, ve Polat 2016a).

5.1 Motivasyon

Literatürde RL yaygın olarak Markov Karar Süreci (MDP) modelinde tanımlanmaktadır. Q-Öğrenme (Watkins 1989) muhtemelen en popüler RL algoritmasıdır, çoğunlukla basitlik ve kullanım kolaylığı açısından önemlidir (Bölüm 2.2.2).

Tercihler çatısı (Sutton, Precup, ve Singh 1999) SMDP modeline dayanan önde gelen soyutlama modellerinden biridir ve MDP modelinin üstünde zamanlı eylemler tanımlamak ve çağırmak için parçala ve yönet (İng., divide-and-conquer) yaklaşımıyla performansı artırmak için kullanır. MDP'lerin birim zaman geçiş dinamiklerini hâlâ koruyan bir eylem, artık bir takım ayrık zaman adımları için sürdürebileceği ve bir tercih olarak anılacak olması anlamında genelleştirilebilir. Diğer bir deyişle, bir "tercih", bir "eylem" in geçici olarak genişletilmiş bir karşılığıdır (Bölüm 2.3).

Bir tercih üç bileşenden oluşur: (1) bir başlatma kümesi (bir tercihin başlayabileceği durumlar), (2) tercihin yerel politikası ve (3) bir sonlandırma koşulu (bir tercihin sonlandırılması). Q-Öğrenme'nin tercihlerin dahil edilmesi için doğal bir uzantısı, her bir ilkel eylemin değerinin düzenli Q-Öğrenme'de olduğu gibi güncellendiği Macro-Q Öğrenme'dir (Mcgovern, Sutton, ve Fagg 1997); bir tercihin değeri ayrı bir tercih seviyesi güncelleme kuralı üzerinden güncellenir.

Tercihler çatısı anlamlı veya kullanışlı tercihler tasarlamak için hiçbir strateji önermez. Yine de, otomatik olarak tercihler üretmeye çalışan yöntemler vardır. Öne çıkan algoritma ailelerinden biri, darboğazlara dayalı bir bölme mekanizmasına dayanır. Bu yöntemler, doğal olarak bir

bölme durumunu veya bir bölgeyi sonlandırma kriteri olarak yorumlayacağı için, sonlanma koşuluna odaklanan alt hedef bulma tekniklerine (Şimşek 2008) dayalıdır (Bölüm 2.5.2).

Öte yandan başlatma kümeleri (İng., initiation sets), genellikle, “eylem sonlandırma durumları dışındaki tüm durumlar” gibi daha basit bir şekilde tanımlanır. (Jong, Hester, ve Stone 2008) eylem seçim mekanizmasının, daha sonra öğrenme sırasında seçimleri sınırlamasını umut eder. “Tercih gecikmesi” gibi kısıtlayıcı parametreler, tercih uzunluğu için belirli bir sınır getirmek için kullanılabilir (Şimşek ve Barto 2004). Bazı yöntemler, durumlar için ulaşılabilirlik ölçütlerini, sonlandırma koşullarında buluşsal bir ölçüt olarak içermektedir (Stolle ve Precup 2002). Mevcut çalışmaların neredeyse hiçbiri üretilen tercihlerin kalitesini artıracağına inandığımız “başlangıç kümesinin” potansiyeline odaklanmamaktadır.

Bu kısımda, tercihlerin başlatma kümelerini üretmek için yararlı durumları tanımlamak amacıyla kullanılan tarihçe ağacı tabanlı sezgisel bir yöntem önerilmektedir. Bir tercihin başlangıç kümesi soyutlamanın ayrılmaz bir parçası olduğundan, iyi bir sezgisel yöntemle öğrenmenin genel performansının artırılabileceği deneysel olarak gösterilmiştir.

5.2 Sezgisel Tarihçe Ağacı Yöntemi

Tercih üretme sürecinin ayrılmaz bir parçası olarak, başlatma kümesi için durum seçimi önemlidir. Tercihi, ödül zirvelerinin (İng., reward peak) göreceli yönlendirmesi gibi çevre özellikleri hakkında bazı bilgilerle yönlendirme, özellikle de öğrenmenin ilk aşamalarında tercih performansı üzerinde olumlu etkiye sahip olacaktır.

Bu çalışma, bir bölüm (İng., episode) içinde son durumundan önce ziyaret edilenler arasında başlatma kümesi durumlarını seçen başlatma kümeleri için yaygın olarak kullanılan hırslı (İng., greedy) yaklaşımın iyileştirilmesini amaçlamaktadır. Tercihin son durumundan önce kontrol edilecek geçiş sayısı, tercih gecikmesi olarak adlandırılan harici olarak sağlanan bir parametredir.

Öngörümüz, bir son durumun, durum alanının bazı ayırt edici durumlarına göre kısa vadeli fayda bakımından göreceli bir yönelime sahip olacağıdır. Önerilen yöntem, bir başlatma kümesi oluşturmak için bir tarihçe ağacından yararlanan sezgisel bir yöntemdir. Amaç durumun göreli yönlendirmesi yoluyla tercihleri inşa etmektir. Yani, yeni bir tercihin başlatma kümesi, örtülü bir “yön” ile inşa etme ihtimali yüksek olan durumlarla kısıtlanmaktadir. Yöntem aynı zamanda,

gereksiz döngülerin ortadan kaldırılması sayesinde, tercih gecikmesi yöntemine kıyasla başlatma kümesi olarak daha büyük bir dizi kümeyi dahil etmeyi amaçlamaktadır.

Bir son s_t durumu verildiğinde, yöntem kök düğüm olarak s_r değerine sahip bir ağaç oluşturur; bu, muhtemelen bir hedef durum ya da bir ödül zirvesi veren bir durumdur ve yaratılacak tercih için doğru bir yön sağlar. Algoritma, s_t ile biten tüm alt tarihçeleri (veya bölümleri) geçerek, her ziyaret edilen düğümden s_r'ye giden en kısa yollarla bir ağaç üretir. s_r ile biten her bölüm, her durumun üst durumu, s_r'ye giden en iyi yolu temsil etmek üzere belirlenerek en son ziyaret edilen durumdan başlayarak birinci duruma kadar gezilir. Sonuç olarak, geçiş esnasında gözlemlenen tüm durumlardan oluşan düğüm kümesi ve her durumdan üstüne geçişlerden oluşan ayrıt kümesi olmak üzere bir tarihçe ağacı oluşturulur.

Yöntem, daha sonra, her ziyaret edilen durumu (s_t hariç) başlatma kümesine eklemek için son durum s_t'den başlayan bir gezim (İng., traversal) uygular. s_t son durumu 0 derinliğinde olmak üzere, gezimin gideceği maksimum derinlik, seçenek derinliği adlı parametre ile belirlenir.

5.3 Deneyler

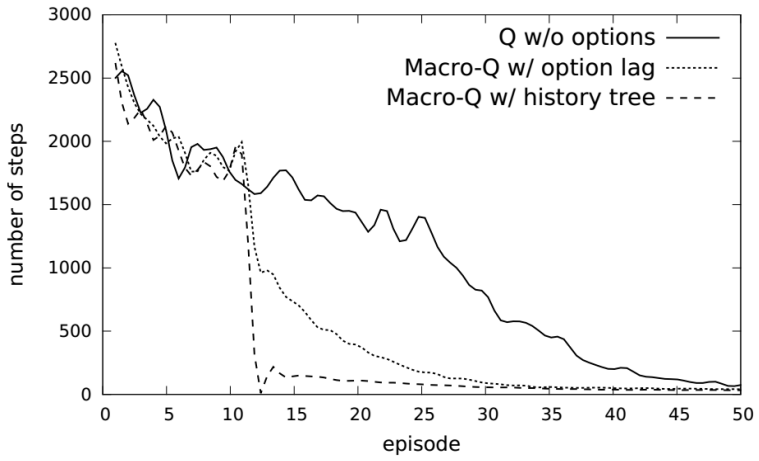
5.3.1 Deney Düzeneği

Deneylerde tek bir tıkanıklık durumu ile problemin iki alt probleme ayrıldığı klasik bir problem olan, 2 oda arasında bir kapının olduğu 2 rooms problemi kullanılmıştır. Tarihçe ağacı yöntemi ve tercih gecikmesi yöntemi arasında adil bir karşılaştırma yapmak amacıyla, son durum (kapıdaki durum) önceden etmene sunulmaktadır. Etmen, deneyde için 10 olarak belirlenmiş yeterli sayıda bölüm deneyimledikten sonra tercih oluşturulmasını başlatır.

Başlangıç kümeleri oluşturulduktan sonra, Deneyim Yeniden Oynatma (İng., Experience Replay, ER) (Lin 1992a), son duruma erişmek için başlatma durumlarından başlayarak politikayı oluşturmak amacıyla etmen tarafından kullanılır. ER mekanizması için, son duruma ulaşıldığında büyük bir ödül verilirken, başlangıç kümesinden ayrılmak için büyük bir ceza ve her geçiş için küçük bir ceza uygulanır. Öğrenme parametreleri $\alpha = 0.125$ ve $\gamma = 0.9$, her seansta 10 kez tekrarlanan deneyim tekrarlama oturumları için kullanılır. Bir tercih, 1.0 olasılığı ile son durumuna ulaştığında sona erer, ancak başlatma kümesindeki herhangi bir durumda sonlanması yasaktır.

Tercih gecikmesi ve tercih derinliği parametreleri, ortaya çıkan tercihlerin her iki yaklaşımın ortalamasında yaklaşık aynı büyüklükte başlatma kümesine sahip olacağı şekilde ayarlanır.

Öğrenme parametreleri ile birlikte, Makro-Q öğrenme, $\varepsilon = 0.1$, $\alpha = 0.05$ ve $\gamma = 0.9$ ile kullanılır. Her iki yöntem de tercihsiz aynı öğrenme parametrelerini kullanan normal Q-Öğrenme ile karşılaştırılmıştır. Deney sonuçları, 200 deneyin ortalaması olarak sunulmuştur.

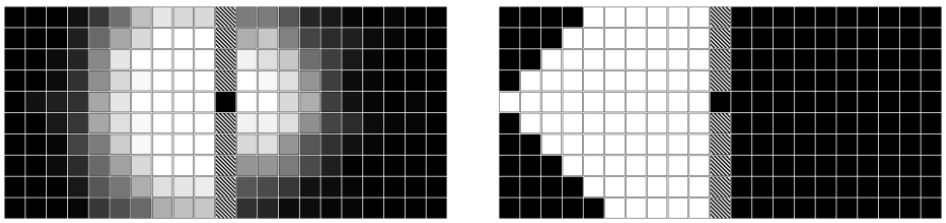


Şekil 5.1. Yöntemlerin adım sayısı - bölüm sayısı grafikleri

5.3.2 Sonuçlar ve Tartışma

Şekil 5.1'de verilen sonuçtan açıkça görüleceği üzere, başlatma kümesi için tarihçe ağacı yöntemi kullanılarak üretilen tercihlerde Makro-Q öğrenme yöntemi, hedefe ulaşmak için atılan adım sayısı açısından bakıldığında tercih gecikmesi yöntemine sahip olana göre daha iyi performans sergilemektedir. Tarihçe ağacı, öğrenme sürecinin ilk aşamalarından başlamak üzere, Makro-Q öğrenme algoritmasına açık bir avantaj sağlamaktadır.

Metodumuzun bu iyileştirmeyi nasıl gerçekleştirdiğine ilişkin daha derin bir kavrayış elde edebilmek içinŞekil 5.2, her bir durumun ortalamada bir başlatma kümesine dahil olma sıklığı görselini sunmaktadır. Daha parlak renk, durumun başlatma kümesi öğesi olarak daha fazla kez seçildiğini, daha koyu renk daha az defa seçildiğini gösterir.



(a) option lag (b) history tree

Şekil 5.2. 2 rooms probleminde oluşturulan tercihlerin başlatma kümeleri

Her iki yöntemin de başlatma kümelerindeki durum sayısının neredeyse eşit olmasına rağmen, etmen kapıya geri dönen döngülere takılı kalabileceğinden, tercih gecikme mekanizması, kapının sağ tarafında bulunan pek çok durumu seçme ve başlatma kümesine ekleme eğilimindedir. Bununla birlikte, tarihçe ağacı yöntemi tarafından oluşturulan başlatma kümesi, yalnızca sol odanın durumlarından oluşur. Bu farklılaşma, etmenin sağ taraftaki durumlar arasında bir tercih keşfetmesini engeller ve bu durum daha iyi öğrenme performansı sağlar.

6 TAKVİYE ÖĞRENMEDE ALT HEDEF TESPİT YÖNTEMİ OLAN FARKLI YOĞUNLUK İÇİN BİR KAVRAM SÜZME YAKLAŞIMI

Takviye öğrenmede (RL), alt-hedef tespit yöntemleri durum uzayındaki darboğazları bulmayı hedeflemekte ve böylece problem doğal olarak daha küçük alt problemlere ayrılabilmektedir. Bu bölümde, mevcut bir alt hedef bulma metodu olan Farklı Yoğunluk (İng., Diverse Density, DD) yöntemini, tam ve kısmi gözlemlenebilir RL problemleri için kullanılacak şekilde genişleten bir kavram süzme (İng., concept filtering) metodu önerilmektedir. Önerilen yöntem, çoklu örnekli öğrenme yardımı ile yararlı alt hedefleri keşfetmekte başarılıdır (Demir, Çilden, ve Polat 2017b). Orijinal algoritma ile karşılaştırıldığında, ortaya çıkan yaklaşım, çözüm kalitesinden ödün vermeden çok daha hızlı çalışmaktadır. Dahası, algısal çakışma barındıran problemlerde gözlemsel darboğazları bulmak için etkili bir şekilde kullanılabilir.

6.1 Motivasyon

RL (Sutton ve Barto 1998), diğer pek çok makine öğrenme çerçevesi gibi, durum uzayının boyutu arttıkça sorun yaşamaktadır. Bu sorunun geleneksel çözümü, parçala ve yönet yaklaşımıyla problemin bölünmesi, üretilen alt problemlerin tek tek çözülmesi ve son olarak bütün alt çözümlerin genel çözümü inşa edecek şekilde bir araya getirilmesidir. Fikir basit de olsa, RL bağlamında gerçekleştirilmesi o kadar kolay değildir: eldeki problemin en etkin şekilde nasıl parçalanacağı kesin olarak bilinemez.

Dikkate değer bir yöntem, problemi kümelemesi beklenen darboğazları durum uzayında araştırmak, tespit edilen darboğazlara nasıl ulaşılacağını öğrenmek ve o ana kadar öğrenilen alt hareket tarzlarını kullanarak kümeler arasında soyut bir öğrenme stratejisi geliştirmektir. Darboğazları otomatik olarak keşfetmeyi amaçlayan çeşitli metotlar vardır. Farklı Yoğunluk (İng., Diverse Density, DD), bu konuda etkili bir yöntemidir (A. McGovern ve Barto 2001), ancak hesaplama karmaşıklığı gerçekçi problemler için kullanımını sınırlar.

Bu bölümde, McGovern'ın DD yaklaşımı için hesaplama süresini önemli ölçüde iyileştiren Kavram Filtreleme (İng., Concept Filtering) olarak adlandırdığımız bir yöntem önerilmektedir. Yöntem köprülük (İng., bridgeness) ve kümelenme (İng., clustering) katsayılarını kullanan ve geliştiren Tıkanıklık Oranı (İng., Congestion Ratio, CR) olarak adlandırmayı uygun gördüğümüz yeni bir yerel çizge ölçüsünü kullanır. Bu yaklaşımı kullanarak güncellenmiş RL algoritmasının

etkinliği hem tam gözlemlenebilir, hem de kısmi gözlenebilir problemler için deneylerle gösterilmiştir.

6.1.1 Gizli Durumlar Barındıran RL Problemleri

Durum-eylem çiftleri yerine gözlem-eylem çiftleri üzerinde Q-Öğrenme uygulayarak kısmen gözlemlenebilir bir problemi çözmeye çalışmak ve sonucun iyi çıkmasını ümit etmek mümkündür. Maalesef, algısal çakışma (İng., perceptual aliasing) olarak anılan bir olgu nedeniyle gözlemin anlamı genellikle tek başına problemi çözmek için yeterli değildir. Aynı gözlem etmen tarafından (optimum eylemlerin muhtemelen farklı olduğu) iki farklı durumda gözlemleniyorsa, durumların algısal olarak örtüştüğü söylenir (Chrisman 1992).

Algısal çakışma, gözlem yapısı vasıtasıyla POMDP modelinde (Bölüm 2.1.3) doğal olarak vardır, ve çoğu zaman, durumların bazıları bu çakışmayı gösteriyorsa model Markov değildir. (Littman 1994) çalışmasında, Markov olmayan ortamlarda öngörülebilir belleksiz politika çözümleri bulmakta zorlanıldığını göstermiştir. Bununla birlikte kategori, göreceli olarak gerçekçi doğası nedeniyle zorlu ve cazip olagelimiştir, ve bu alanda birkaç etkili yöntem vardır (Sutton ve Barto 1998; Loch ve Singh 1998).

6.1.2 Köprülük ve Kümelenme Katsayıları

Köprülük katsayısı $BC(v)$, v düğümünün bulunduğu çizgedeki komşularını bağlama derecesini gösteren yerel bir ölçüttür:

$$BC(v) = \frac{d^{-1}(v)}{\sum_{i \in N(v)} d^{-1}(i)}$$

Burada, $d(v)$, v düğümünün derecesini, $N(v)$ ise doğrudan komşularının kümesini gösterir.

Kümelenme katsayısı $CC(v)$, v düğümünün komşularının ne ölçüde bağlı olduklarını gösteren bir başka yerel ölçüttür (Watts ve Strogatz 1998). $CC(v)$'nin bir uzantısı, $CC^{(k)}(v)$ şeklinde gösterilen ve k derinliğindeki bütün komşuları göz önüne alan k-Kümelenme katsayısıdır (Jiang ve Claramunt 2004). Bu ölçüt, v düğümünün bir kümeye ne kadar bağlı olduğunu gösterme eğilimindedir ve şu şekilde tanımlanmıştır;

$$CC^{(k)}(v) = \frac{2e^{(k)}}{n^{(k)}(n^{(k)} - 1)}$$

Burada $e^{(k)}$, k komşu arasındaki ayrıt sayısını, $n^{(k)}$ ise v düğümünün k derinlik komşuluğundaki düğüm sayısını temsil eder. Açıktır ki, $CC(v) = CC^{(1)}(v)$.

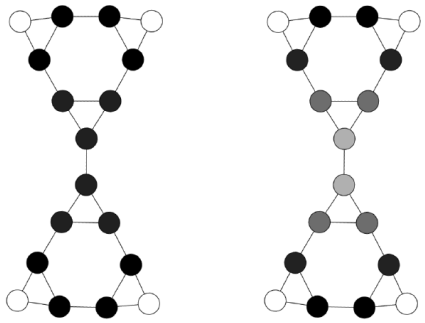
Her iki ölçümün de yerel olduğunu ve yerellik önyargısı ile global ölçekte yanlış sonuçlar üretme potansiyeli olduğuna dikkat etmek gerekir. Örneğin, yüksek bir BC değeri, düğümün genel bir köprü olduğunu göstermez. Bununla birlikte, genel varyantlarına (Hwang vd. 2006; Yang ve Liu 2008) göre hesaplama açısından bu yöntemler çok ucuzdurlar ve dolayısıyla genel kararlar için destekleyici bilgi olarak kullanılabilirler. İlginçtir ki BC ayrıklığı ölçerken, $CC^{(k)}$ birleştiricilik ile ilgilidir; Dolayısıyla bu iki ölçüt, bir düğüm hakkında tamamlayıcı bilgi verme eğilimindedir.

6.2 Kavram Filtrelemeli Farklı Yoğunluk

(A. McGovern ve Barto 2001), RL problemleri için durum uzayındaki darboğaz bölgelerini belirlemek amacıyla ilk kapsamlı DD kullanımını gerçekleştirimişlerdir. Her kavramın - RL bakış açısıyla - çevreden toplanan bir gözlem olduğu hedef kavram araması fikrini kullanır. Fikir, bir alt hedefin başarılı bölümlerde sıklıkla ziyaret edildiğinin gözlenmesi, ancak başarısız olan bölümlerde gözlenmemesine dayanır. Bu yöntem, bir alt hedefin özelliklerinden hiçbirine bağlı değildir, ancak yalnızca etmenin geçmiş deneyimlerine bağlıdır. Bu nedenle, etmen, özelliklerini analiz ederek bir alt hedef belirleyemediği durumda, kısmi gözlenebilen problemlerde alt hedef tespiti yapabilmesi için umut vaat eden bir adaydır.

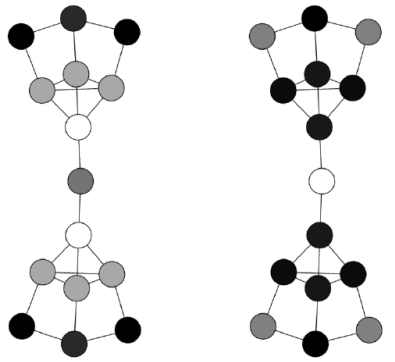
DD aramaları için kavramlar arasında bir benzerlik hesaplaması gereklidir, bu nedenle algoritma kavramlar arasındaki en kısa yol mesafelerini kullanır. Çevre modeli etmen tarafından bilinmediğinden mesafeler yönteme önceden sağlanmalıdır. Bununla birlikte, gözlem geçiş çizgesini aşamalı olarak oluşturarak ve DD sırasında uzaklıkları hesaplayarak benzer bir performans elde etmek mümkündür.

DD'yi bir alt hedef tespiti için kullanırken dikkate alınması gereken bir yönü, verimliliğine karşılık bir bedel olarak ödenen hesaplama karmaşıklığıdır. DD, her bir gözlemi bir kavram olarak görür ve her bölümün sonunda her biri için bir artımlı hesaplama gerektirir. Bununla birlikte, bazı çizge özelliklerinin değerlendirilmesi yoluyla bir takım kavramların ortadan kaldırılması yoluyla iyileştirme yapılabilir.



(a) BC (b) $CR^{(3)}$

Şekil 6.1. BC-CR kıyaslaması



(a) $1/CC^{(2)}$ (b) $CR^{(2)}$

Şekil 6.2. CC-CR kıyaslaması

6.2.1 Tıkanıklık Oranı Ölçütü

Bir çizge düğümü v için tıkanıklık oranı $CR^{(k)}(v)$ şu şekilde tanımlanır;

$$CR^{(k)}(v) = \frac{BC(v)}{CC^{(k)}(v)}$$

Burada, $BC(v)$, v düğümünün köprülük katsayısını, $CC^{(k)}(v)$ k-Kümelenme katsayısını ve k ise k-Kümelenme Katsayısı için komşuluk mesafesini tanımlar. Bu ölçüt, çizgenin farklı bölgelerini birbirine bağlayan ancak görünüşte bir kümeye ait olmayan düğümleri vurgular. Dolayısıyla, $CR^{(k)}(v)$ en çok köprülük sağlayan ve bir kümeye bağlılığı en az olan düğümlerini saptar denebilir.

$CR^{(k)}(v)$ ölçütüne BC ve $CC^{(k)}(v)$ değerlerinin bazı şartlar altında yetersiz olmasından dolayı ihtiyaç duyuldu. Örneğin, düğümler arasında hemen hemen eşit bir derece dağılımı olan bir çizge için, BC değerinin her düğüm için az ya da çok aynı olması beklenir. Bu nedenle, bu özel yapı sebebiyle kümeler arasındaki belirgin darboğaz düğümünü dikkatten kaçırmak olasıdır (Şekil 6.1a). Diğer yandan $CR^{(3)}(v)$ hesaplaması, köprü kuran düğümleri daha doğru bir şekilde tanımlar (Şekil 6.1b).

Benzer şekilde, Şekil 6.2a'da verilen gibi bir çizgede her düğüm için $1/CC^{(k)}(v)$ değeri hesaplanır, böylece en az kümelenme gösteren düğümün daha yüksek bir değer elde etmesi sağlanır. Dikkat edilmelidir ki; görünür darboğaz düğümünün çevresindeki düğümlerin (yani, bu çizgenin merkezi düğümü), darboğaz durumuna kıyasla daha az kümelenmiş oldukları anlamına gelen, $1/CC^{(2)}(v)$ değerleri yanlış bir yönlendirmeye sebep olur. Bununla birlikte, $CR^{(2)}(v)$, herhangi bir kümeye ait olmayan düğümü açıkça saptamaktadır (Şekil 6.2b). Şekil 6.1 ve Şekil 6.2'de daha parlak renkler daha yüksek değer anlamına gelmektedir.

6.2.2 Tıkanıklık Oranı Kullanarak Kavram Filtreleme

Alt hedef tespiti için McGovern'ın Farklı Yoğunluk yöntemi (MDD), her bölümün olumlu ya da olumsuz bir torba olarak sınıflandırılmasını sağlar (A. McGovern ve Barto 2001). Bu karar, bölümün bir hedef durumla bitip bitmediğine bağlı olarak yapılabilir. Ardından, algoritma her kavram için DD değerlerini hesaplar ve tepe değeri olanların ρ ortalamalarını günceller. Son olarak eşik değerini geçen kavramlar alt hedef olarak işaretlenir. Sonuçlardaki olası gürültü, işlem boyunca tüm kavramların ortalamaları üzerindeki bozunma mekanizması yoluyla ortadan kaldırılır.

Bu bölümde önerilen yöntemin sunduğu ilk iyileştirme, kısa yol mesafelerini aşamalı olarak hesaplamak için bir rutin ekleyerek, DD değerlerini hesaplamak için kavramlar arasındaki en kısa yol mesafelerine baştan sahip olma gerekliliğini ortadan kaldırmasıdır. Yöntem, öğrenme prosedürüyle eşzamanlı olarak, etkileşim çizgesi G yoluyla etmenin geçişlerini (Şimşek 2008) takip eder. Çizgede bir güncelleme olduğunda, Dijkstra'nın en kısa yol algoritması (Dijkstra 1959) aracılığıyla en kısa yol mesafesi matrisi D_G de güncellenir (Algoritma 6.1 satır 8). Bu sayede yöntem, problem uzayının geçiş özelliklerine olan bağımlılığından kurtulur. İlk etkileşim çizgeleri eksik, ve başlangıçtaki en kısa yol mesafeleri gürültülü veya yetersiz olsa bile, D_G, öğrenme sürecinin başlangıcında, çizgenin keşifler boyunca tamamlanmasına bağlı olarak, hızlı bir şekilde gerçek değerine evrilir.

Algorithm 1 $DD\_WITH\_CONCEPT\_FILTERING$

Require: λ, θ, k

Initialize full trajectory database to $\emptyset$
Initialize running averages ρ_c to 0
$G \leftarrow \emptyset$, $D_G \leftarrow \emptyset$
$C \leftarrow \emptyset$
for each episode **do**
    Interact with environment / Learn using RL
    Add observed full trajectory to database
    Update G and D_G if necessary
    Create pos. or neg. bag from filtered trajectory
    Update the concept set C with new observations
    $C_f \leftarrow FILTER\_CONCEPTS(C, G, D_G, k)$
    Search for diverse density peaks in C_f
    for each peak concept c found **do**
        Update the running average by $\rho_c \leftarrow \rho_c + 1$
        if ρ_c is above threshold θ **then**
            if c passes static filter and $c \in C_f$ **then**
                Mark c as a subgoal
            end if
        end if
    end for
    Decay all running averages by $\rho_c \leftarrow \lambda \rho_c$
end for

Algoritma 6.1 Kavram filtreleme ile Farklı Yoğunluk hesaplaması

Algorithm 2 $FILTER\_CONCEPTS$

Require: C, G, D_G, k

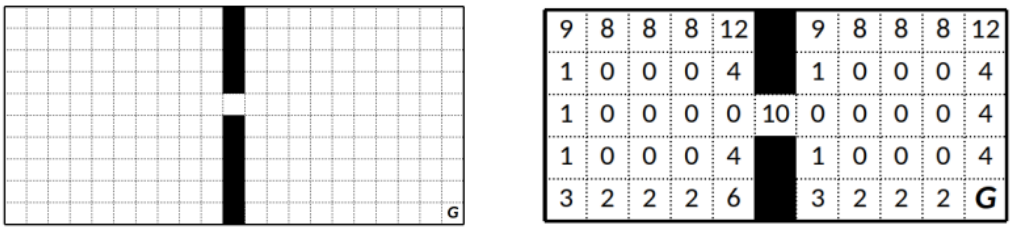
$C_f \leftarrow \emptyset$
for each concept $c \in C$ **do**
    Calculate $BC(c)$ by using D_G ▷ Eqn. 1
    Calculate $CC^{(k)}(c)$ by using G, D_G and k ▷ Eqn. 2
    $CR^{(k)}(c) \leftarrow \frac{BC(c)}{CC^{(k)}(c)}$ ▷ Eqn. 3
end for
for each concept c **do**
    $N(c) \leftarrow$ immediate neighbors of c
    if c has a peak value in $N(c)$ **then**
        $C_f \leftarrow C_f \cup \{c\}$
    end if
end for
return C_f

Algoritma 6.2 Tıkanıklık oranı kullanarak kavram filtreleme hesaplaması

Önerilen yöntemin diğer katkısı, DD tepe noktalarının araştırılmasının yalnızca kavram filtreleme yordamından sağ kalan kavramlar arasında yapılmasını ve filtreyi geçemeyen diğer kavramların yok sayılmasını sağlar. Filtreleme prosedürü Algoritma 6.2 tarafından verilir. Bu yaklaşım, DD yönteminin, yalnızca küçük sayıda kavram içinde arama yapmasından dolayı (Algoritma 6.1 satır 12), ortalama olarak kavramların önemli ölçüde daha azıyla uğraşmasına sebep olur. Genel yöntem, Kavram Filtrelemeli Farklı Yoğunluk (İng., diverse density with concept filtering) olarak adlandırılır ve DDCF olarak kısaltılır.

Orijinal DD algoritmasındaki tercih oluşturma (İng., option generation) aşaması iki nedenle bu çalışmada ele alınmamıştır: (1) Bu çalışmadaki amaç, daha yüksek kalitede alt hedefleri tespit etmek, ya da daha iyi tercihler (İng., options) oluşturulması değildir. Beklenen katkı, alt hedef bulma hızının artırılmasıdır. (2) Önerilen yöntem gizli durumlar barındıran problemler için etkin bir şekilde kullanılabilir, ancak Markov olmayan ortamlar için tercih (İng., option) üreten bir yöntem henüz mevcut olmadığından, seçenek oluşturma deneyi tasarlamak mümkün değildir.

Bu çalışmada, kısmi gözlemlenebilirlik durumunda, DDCF'nin çalışması için iki varsayım yapılmaktadır. Birincisi, DD yönteminin doğru DD değerlerini hesaplamak için pozitif torbalar üzerinde yüksek kalite gerektirmesi nedeniyle, etmen, sorunu en azından kabul edilebilir bir seviyede öğrenebilir olmalıdır. İkincisi, okuyucu tarafından aşikar bulunacağı düşünülen nedenlerden, alt hedef durumun, DDCF tarafından tanımlanabilmesi için ayrık bir gözlem verdiği varsayılmaktadır (genelde pratik durumlarda olduğu gibi, örneğin, bir "kapı" görsel/duyusal bilgi ile ayırt edilebilmektedir).



(a) 2rooms1door$_F$ (b) 2rooms1door$_P$

Şekil 6.3. Deney problemleri

6.3 Deneyler

6.3.1 Deney Düzeneği

Yöntemin etkinliğinin ölçülmesi için tasarlanan deneyler, 2 oda 1 kapı ızgara dünya (İng., grid world) probleminin iki versiyonunda gerçekleştirilmiştir. Şekil 6.3a'da gösterilen ilk versiyonda ($2rooms1door_F$), etmen durumları tam olarak gözlemleyebilir ve alt hedef durumunu tespit etmeyi amaçlamaktadır. Şekil 6.3b'de verilen ikinci problemde ise ($2rooms1door_P$) etmen, sınırlı gözlem yapısı aracılığıyla alt hedef durumunu bulmaya çalışır.

Deney sonuçları, McGovern'ın DD yöntemi (MDD) ve Yerel Aradalık (İng., Local Betweenness, LoBet) (Şimşek 2008) adı ile bilinen alt hedef tespit algoritması ile karşılaştırılmıştır. LoBet, etmenin etkileşim grafiğinde tıkanıklık durumlarını bulmak için aradalık (İng., betweenness) adlı bir çizge ölçütü içerir (Freeman 1977) ve tam gözlenebilirlik altında iyi performans gösterdiği bilinmektedir.

Deneylerde MDD ölçütü, gözlemler arasındaki en kısa yol mesafesi ile sağlanır. Her iki versiyon için de her kavramın tek bir gözlem olduğu varsayılmaktadır.

DD parametreleri hem MDD hem de DDCF için $\theta = 0.8$ ve $\lambda = 0.9$ olarak ayarlanmıştır. DDCF için, kümeleme katsayısı derinliği tam gözlenebilir durumda $k = 3$, kısmen gözlemlenebilir durumda ise $k = 2$ olarak belirlenmiştir. Tam gözlemlenebilirlik için, (A. McGovern ve Barto 2001) tarafından önerilen şekilde, tüm torbalar pozitif olarak kabul edilmiştir. Öte yandan, eşit sayıda pozitif ve negatif çanta bulunduğunu garanti etmek için, kısmi gözlenebilir problemde bölüm başına bir adım sınırı kullanılmıştır. Her iki algoritma, bir kavramın dinamik ortalaması ρ, θ'yı geçtiğinde, kavramı bir alt hedef olarak işaretlemektedir. Alt hedef tespit işlemi sırasında etmen, $\varepsilon = 0.1$, $\alpha = 0.05$ ve $\gamma = 0.9$ parametre değerlerini kullanarak ε–hırslı (İng., ε–greedy) eylem seçme mekanizması ile Q-Öğrenme kullanmıştır. Her deney 100 bölüm sürmüş ve sonuçlar 50'den fazla deneyin ortalaması alınarak elde edilmiştir.

Her iki problem için de etmen, kuzey, doğu, güney, batı olmak üzere dört yöne tek adımlık hareket eylemi kullanabilmektedir. $2rooms1door_P$ problemi öngörülebilirdir (İng., deterministic). Diğer yandan, $2rooms1door_F$ probleminde etmen, 0.9 olasılıkla istenilen yöne doğru, 0.1 olasılıkla rastgele bir yöne ilerler. Etmen başlangıçta, probleme sol odanın herhangi bir hücresinde başlar. $2rooms1door_F$ problemindeki hedef duruma ulaştığında positif bir ödül alır ve başka herhangi bir hareket için ödül veya ceza almaz. Öte yandan, $2rooms1door_P$ problemi için,

her eylem için küçük bir ceza, bir duvara çarpmak için daha yüksek bir ceza ve hedef duruma ulaşmak için pozitif bir ödül vardır. Etmenin $2rooms1door_P$ problemindeki gözlem yapısı, çevreleyen duvarları (hemen komşu hücreleri duvar olup olmadığı) gürültülü algılaması üzerine oluşur.

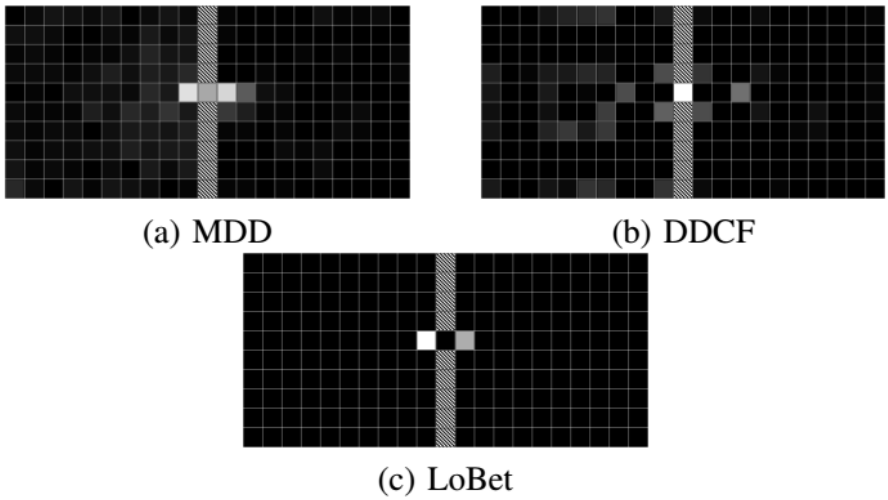
6.3.2 Sonuçlar ve Tartışma

Tablo 6.1. Ortalama adım başına ödül miktarları

Problem	MDD	DDCF
2rooms1door$_F$	0.0018	0.0019
2rooms1door$_P$	-0.0046	-0.0049

Tablo 6.2. Kullanılan kavram sayısı ve harcanan zaman

Problem	# concepts used		time (msec)	
	MDD	DDCF	MDD	DDCF
2rooms1door$_F$	198.96	**55.90**	252885.18	**80170.94**
2rooms1door$_P$	9.94	**4.93**	1436.82	**939.98**



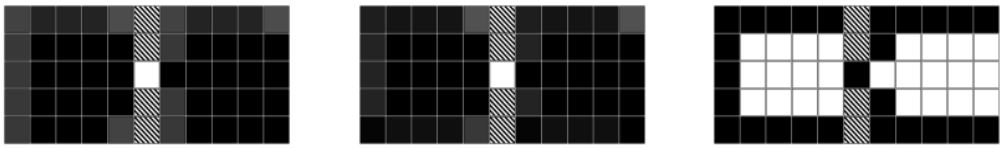
(a) MDD (b) DDCF

(c) LoBet

Şekil 6.4. $2rooms1door_F$probleminde bulunan alt hedefler

Şekil 6.4, $2rooms1door_F$ problemi için alt hedef tespit performansını gösterir. Tüm algoritmalar, kapı önündeki durumları başarıyla bulmaktadır. LoBet en az gürültülü sonuçlar verirken, DDCF,

MDD algoritmasına kıyasla daha düşük gürültüye sahiptir. DDCF, daha düşük $CR^{(k)}$ değerine sahip kavramlar dikkate alınmadığından, dar ve anlamlı bir konsept kümesinde Farklı Yoğunluk zirvesi arar ve bu durum daha az gürültülü sonuç elde edilmesine yol açar. Dahası, bu iyileşme, daha az sayıda aday kavramıyla (Tablo 6.2) zirve DD değerlerini bulmak için gereken süreyi önemli ölçüde azaltır.



(a) MDD (b) DDCF (c) LoBet

Şekil 6.5. $2rooms1door_P$ probleminde bulunan alt hedefler

Kısmi gözlemlenebilir durumda, DD algoritmasının, son derece belirsiz gözlemleri alt hedef durumlarına karşılık gelen gözlemlerden ayırt etmek için en az bir negatif çanta gerektirdiği ortaya konulmuştur. Aksi halde, belirsiz gözlemler ($2rooms1door_P$ problemindeki 0 numaralı gözlem gibi), etmenin bu gözlemleri daha sık almasından dolayı yüksek DD değerlerine sahip olabilir.

Keşfedilen alt hedefler Şekil 6.5'te gösterilmiştir. Etmenin etkileşim çizgesi (İng., interaction graph), algısal çakışma nedeniyle durum geçiş çizgesinden (İng., state transition graph) tamamen farklı olduğu için, LoBet gerçek alt hedef durumlarını bulamamaktadır. Aradalık ölçütü, hatalı şekilde, etkileşim çizgesi için merkezi düğümler gibi davranan belirsiz gözlemleri vurgular. Bu bağlamda, çizge tabanlı özellikleri kullanan algoritmaların, gizli durumlar içeren problemlerde belirgin bir dezavantaja sahip olduğu söylenebilir.

Öte yandan, daha güvenli bir yaklaşım kullanan DD algoritması genel olarak kısmi gözlenebilirlik altında iyi performans göstermektedir. Belirsiz gözlemler negatif çantalarda da sıklıkla gözlenmekte olduğundan, alt hedeflere karşılık gelen gözlemlere kıyasla daha düşük bir DD değeri elde etmektedirler. Bu ayrım kısmen gözlemlenebilir bir ortamın alt hedeflerini tanımlamak için DD yöntemine yardımcı olur.

Tam ve kısmen gözlemlenebilir olan problemlerde Tablo 6.1 ve Tablo 6.2, DDCF'nin daha az sayıda kavram ürettiğini ve genel öğrenmeyi etkilemeden MDD ile hemen hemen aynı alt hedefleri keşfetmek için daha az zaman harcadığını göstermektedir.

7 EN YAKIN DİZİ BELLEĞİ ALGORİTMASINDA ÖĞRENİMİ GELİŞTİRMEK İÇİN GEÇİŞSEL DARBOĞAZLARIN KULLANIMI

Örneğe dayalı (İng., instance based) yöntemler, gizli durumlar barındıran pekiştirmeli öğrenme problemlerinin çözümünde kullanılmaktadır. En Yakın Dizi Belleği (NSM), En Yakın k-Komşu (kNN) algoritmasını temel alan, yaygın olarak kabul görmüş bir örneğe dayalı yaklaşımdır. NSM, etmenin geçmişini eylem-gözlem-ödül üçlüleri cinsinden hafızasında tutar ve sıradaki en iyi eylemin seçimini oylamak için kullanır. Bu çalışmada, NSM algoritması için etmene ilave bir ön bilgi sağlayan, diğer bir deyişle hedefe giden yoldaki geçişsel darboğazları bildiren sezgisel (İng., heuristic) bir iyileştirme önerilmiştir (Aydın, Çilden, ve Polat 2017). Ek olarak, NSM'in üçlü tanımına, geçişsel darboğazlardaki belirsizliği azaltarak bu iyileştirmeyi geliştirecek bir ekleme örüntüsü gösterilmiştir.

7.1 Motivasyon

RL (Sutton ve Barto 1998), etmenin durum uzayının son durumundan habersiz oluşu, hem sürekliliğin olduğu problemler hem de durum bilgisinin gizli olduğu problemler için sorun teşkil etmektedir. Bu da bizi *örneğe dayalı* olarak da adlandırılan genel geçer hafıza temelli çözümlere yönlendirir. Burada temel fikir, etmenin öğrenme boyunca deneyimlediği ham geçmiş bilgisini hafızada tutmak ve bu veriyi daha sonra örtüşen durumları ayırt etmek için kullanabilmektir.

En Yakın Dizi Belleği (NSM), yaklaşmakta olan bir eylemin gerçek faydasını hesaplamak için k-en yakın komşu (kNN) (Altman 1992) yaklaşımını kullanan örnek tabanlı bir RL algoritmasıdır. NSM, McCallum (A. McCallum 1996a) tarafından RL'ye yeni bir yaklaşım olarak tanıtılmıştır. Örneğe dayalı yöntemlerin sürekliliğin olduğu problemlerdeki gücünden (Andrew William Moore 1990; Atkeson 1992b; Schneider 1995; Andrew W Moore 1995) yararlanarak, etmenin şu andaki gözlemini eğer hedefe ulaşmasında yararlı olacaksa parçalayan Fayda Ayırt Etme Belleği (İng., Utile Distinction Memory, UDM) (R. A. McCallum 1993) yönteminin iki ana dezavantajını çözmeyi hedeflemiştir. NSM, UDM'in yavaş öğrenme hızı ve bir adımdan daha fazla süren bir deneyimin faydasını tespit etmedeki başarısızlığının üstesinden gelmeye çalışır.

NSM etmenin geçmişini eylem-gözlem-ödül üçlülerinin zinciri halinde tutar. Etmenin şu andaki durumunun en yakın komşularını belirleyebilmek için bu zincirdeki her düğümü yinelemeli bir

şekilde dolaşır ve bu komşuları kullanarak bir sonraki adımda yapılması gereken eylemi daha iyi tahmin etmeye çalışır. NSM yönteminin temel prensipleri Bölüm 2.4.1'de özetlenmiş olmakla birlikte, bu bölümde önerilen eklentilerin daha kolay formüle edilmesi için betiğimsi (İng., pseudocode) olarak yeniden ele alınmıştır.

7.2 En Yakın Dizi Belleği (NSM) Betiğimsisi

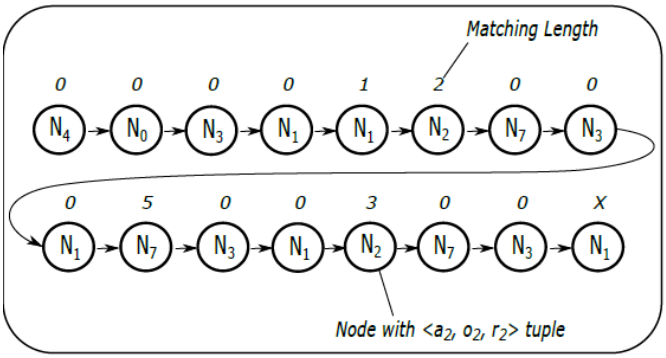
NSM Algoritması, etmenin mevcut durumuna en çok benzeyen geçmiş deneyimlerini saptamak için kNN kullanılan hafıza tabanlı yaklaşımlardan biridir. kNN uygulamalarında genelde gerçekleştiği gibi, NSM algoritması da üç karar aşamasından oluşur: (1) her tecrübeyi kaydetme, (2) bu deneyimler mevcut duruma komşuluğunu hesaplama ve (3) etmenin yapabileceği her eylemin yararını aynı eylem için seçilen komşulardan gelen fayda ile hesaplama.

Algorithm 1 Computing the neighborhood between nodes

```
1: function NEIGHBORHOOD(N_i, N_j)
2:     if ⟨a_i, o_i, r_i⟩ = ⟨a_j, o_j, r_j⟩ then
3:         return 1 + NEIGHBORHOOD(N_{i-1}, N_{j-1})
4:     else
5:         return 0
6:     end if
7: end function
```

Algoritma 7.1 Düğümler arası komşuluk değeri hesaplama

NSM, eylem-gözlem-ödül üçlülerinin (NSM terminolojisinde durum olarak da adlandırılır) bir diziden (geçmişten) oluşan bir arama alanındaki deneyimleri mevcut düğüm ve önceki düğümler arasındaki doğrudan eşitlik kontrolü ile yinelemeli bir biçimde karşılaştırır (Algoritma 7.1). İki düğümün eşleşen uzunluğu komşuluğun derecesini belirler. McCallum, yaygın gösterimin aksine, eylemi takip eden gözlem ve ödül için eylem ile aynı zaman indisini kullanmıştır, bu nedenle biz de bu bölümde de aynı gösterim kullanılmaktadır.



Şekil 7.1. Örnek bir dizi için NSM algoritmasının komşuluk hesaplaması

Şekil 7.1, komşuluğu hesaplamak için bir dizideki deneyimlerin nasıl karşılaştırıldığını örnek üzerinden göstermektedir. Bu dizideki her düğüm bir eylem gözlem-ödül kümesini temsil eder. İki düğüm arasındaki komşuluk, eşleşmeyen bir düğüme erişilinceye kadar bir eşitlik kontrolü yoluyla hesaplanır ve eşleşen uzunluk cinsinden elde edilir.

NSM'yi anlatan ayrıntılı bir betiğimsi, Algoritma 7.2'de verilmiştir. İlginçtir ki, bildiğimiz kadarıyla, ilgili literatürde NSM'ye ait ayrıntılı bir betiğimsi bulunmamaktadır. Dolayısıyla, Algoritma 7.2'nin bu alanda çalışan kişilere yardımcı olacağı değerlendirilmektedir. Her eylem için k-en yakın komşu düğümler, Algoritma 7.1'deki gibi karşılaştırılarak belirlenir. Daha sonra her eylemin faydası, bu komşulardan alınan eylemlerin gerçek Q-değerlerinin ortalaması alınarak hesaplanır. Eşitlik durumunda son düğümlere öncelik verilir. Etmenin yapacağı eylem, rastgele keşif stratejisi (ϵ-hırslı gibi) kullanıyorsa, ya rastgele bir şekilde ya da faydayı azami hale getirecek şekilde seçilir.

Diğer örneğe dayalı algoritmalarının çoğu denetimli öğrenme yöntemi olduğu halde, NSM kendi Q-öğrenme uygulamasıyla dinamik programlama içerir. Son seçilen ödüllere yönelik önyargıları önlemek için, azami fayda sağlayan eylem için seçilen her komşusunu, kendi ödülü ile günceller.

Algorithm 2 Nearest Sequence Memory Algorithm

procedure NSM
  require: $\alpha,\ \gamma$
  $t \leftarrow 0$
  $H \leftarrow \emptyset$ ▷ initialize node history
  while learning continues **do**
    take action a_t, observe o_t and r_t
    $N_t \leftarrow \langle a_t, o_t, r_t \rangle$ ▷ a node tuple, or "state"
    $NN \leftarrow \emptyset$ ▷ neighboring states of the node
    for each node N_i in H **do**
      $n \leftarrow NEIGHBORHOOD(N_t, N_{i-1})$
      add $\langle N_i, n \rangle$ pair to NN
    end for
    sort NN in decreasing order w.r.t. neighborhood
    for each action a in A **do** ▷ trace the action set
      $C_a \leftarrow \emptyset$ ▷ sets of selected nodes partitioned by action
    end for
    for each node N_i in NN **do**
      if k nodes have been selected for each action **then**
        break
      end if
      if k nodes have not been selected for a_i **then**
        add N_i to C_{a_t}
      end if
    end for
    for each action a in A **do**
      $Q_{avg}(a) = \sum_{N_t \in C_a} \frac{q_{N_t}(a)}{||C_a||}$ ▷ calculating the utility
    end for
    $U_t \leftarrow max_a\ Q_{avg}(a)$
    $a_{t+1} \leftarrow argmax_a\ Q_{avg}(a)$
    for each node N_i in $C_{a_{t+1}}$ **do**
      $q_{N_t}(a_{t+1}) \leftarrow (1-\alpha) q_{N_t}(a_{t+1}) + \alpha(r_i + \gamma U_t)$
    end for
    add N_t to H
    if memory limit has been reached **then** ▷ if the memory is fixed length
      delete oldest node in H
    end if
    $t \leftarrow t+1$
  end while
end procedure

Algoritma 7.2 NSM Algoritması

Etmenin belleği sabit büyüklükte bir bellek penceresi ile sınırlandırılabilir (Albus 1991; Lin 1993). Bu durumda, geçmiş zincirine yeni bir düğüm eklendiğinde, zincirdeki en eski düğüm silinir.

7.3 Geçişsel Darboğazlı NSM Algoritması

Gizli durumlar barındıran problemlerde, darboğaz fikrinden yararlanarak NSM performansı daha da artırılabilir. Bununla birlikte, algısal örtüşme nedeniyle darboğazları göstermede tek başına bir gözlem yeterli olmadığından ve NSM'nin atomik birimi olan NSM durumu (başka bir deyişle eylem gözlemi ödül üçlüsü) olduğundan, farklı bir darboğaz tanımına ihtiyaç duyulmuştur. İlk adımda NSM durumu bir dar boğaz olarak kullanmaya çalışılmıştır. Önceki zaman adımındaki eylemi içerdiğinden burada tanımlanan darboğaz geçişseldir, dolayısıyla, problemi anlamlı alt bölgelere ayırabilmek için alt hedef olarak kullanılabilen geçişlerin tanımlanması gerekmektedir. İkinci adım olarak, seçilen geçişsel darboğazlara ulaşmayı teşvik etmek için aracıya yapay bir ödül verilir. Yapay ödül, ortalama "varsayılan" ödülden daha yüksek, ancak ortamdan alınan en yüksek ödülden daha az olmalıdır (problem özelinde tuzaklar ortaya çıkmasına engel olacak ölçüde). Anlatılan prensiplere göre değiştirilmiş NSM algoritması Algoritma 7.3'te verilmiştir. Yıldız (*) sembolü önceki orijinal NSM algoritması olan Algoritma 7.2'de değiştirilen ya da eklenen satırları göstermektedir.

Algorithm 3 NSM with Transitional Bottlenecks

```
 1: procedure NSM_TB
*2:     require: α, γ, r_artificial, bottlenecks
3-4:    ...                                          ▷ unchanged
 5:     while learning continues do
 6:         take action a_t, observe o_t and r_t
 7:         N_t ← ⟨a_t, o_t, r_t⟩
 *:         if N_t ∈ bottlenecks then
 *:             r_t ← r_artificial
 *:         end if
8-36:       ...                                      ▷ unchanged
37:         t ← t + 1
38:     end while
39: end procedure
```

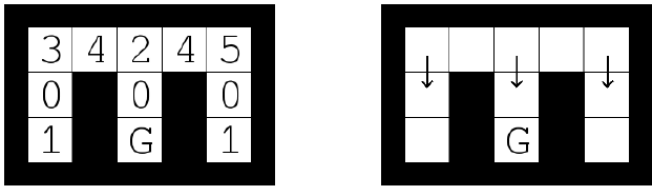
Algoritma 7.3 Geçişsel Darboğazlı NSM Algoritması

Algorithm 4 NSM with Transitional Bottlenecks and Extended State

1: **procedure** NSM_{TBx}
⋆2: **require:** α, γ, $r_{artificial}$, $bottlenecks$
3-4: ... ▷ unchanged
5: **while** learning continues **do**
6: from o_{t-1}, take action a_t, observe o_t and r_t
7: $N_t \leftarrow \langle o_{t-1}, a_t, o_t, r_t \rangle$ ▷ extended NSM state
⋆: **if** $N_t \in bottlenecks$ **then**
⋆: $r_t \leftarrow r_{artificial}$
⋆: **end if**
8-36: ... ▷ unchanged
⋆ $o_{t-1} \leftarrow o_t$
37: $t \leftarrow t + 1$
38: **end while**
39: **end procedure**

Algoritma 7.4 Geçişsel Darboğazlı ve Geliştirilmiş Durumlu NSM algoritması

Bu yaklaşım NSM'nin öğrenme performansını geliştirse de, problemlerde deneyimlenen geçişlerin belirsizliğinden dolayı yararlı bir geçişsel darboğaz tanımlamak zordur. Şekil 7.2'de verilen problemi (A. McCallum 1996a) düşünelim. Şekil 7.2a, problemin durum-gözlem eşleşmesini gösterir; burada durumlar, konumsal olarak çevresindeki duvarların varlığına dayanılarak verilen için farklı bir gözlem oluşturur ve buna göre numaralandırılmıştır. Bu durumda, Şekil 7.2b'deki geçişler, farklı gözlemler üreten durumlardan başlatıldığı ve açıkça hedefe giden yolda farklı önemler teşkil ettiği halde orijinal NSM durumu tanımında ayırt edilememektedir.



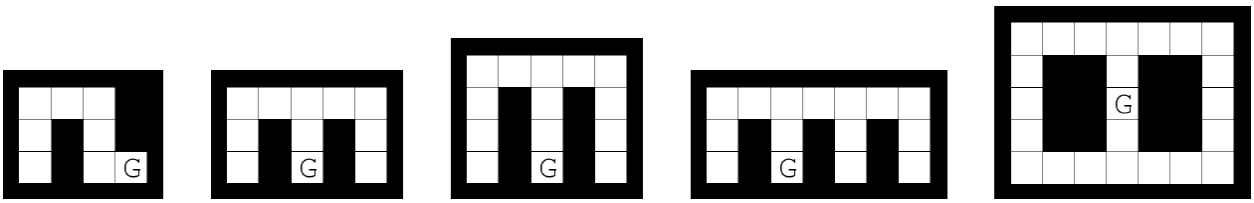
(a) (b)

Şekil 7.2. a) McCallum'un maze problemlerinden biri için gözlem numaralandırması b) Orijinal NSM algoritmasına göre aynı olduğu düşünülen fakat bir önceki gözlemin kaydedilmesi ile kolaylıkla ayırt edilebilecek geçişler

Bu sorunun üstesinden gelmek için, NSM'nin genel eylem seçimi ve oy verme performansı üzerinde belirgin bir olumsuz etkisi olmaksızın, durum üçlüsünün her birinde geçiş bütünlüğünü

kolaylaştırmak için yeni bir NSM durum formu önerilmiştir. Yeni formda eylem öncesi gözlem de NSM durum konseptine de dahil edilmiştir. Diğer bir deyişle, etmen t anında harekete geçtiğinde, $< o_{t-1}, a_t, o_t, r_t >$ deneyimine eklenir. Bir önceki gözlemin de hafızada tutulması sayesinde, bir NSM durumunun kendisi, etmenin belirli bir geçiş hakkında daha kapsamlı bir bilgi sunmaktadır. Darboğaz tanımlama açısından, Şekil 7.2b'deki gibi belirsiz geçişler kaydedilen eski gözlemlerle ayırt edilebilir. Ortaya çıkan yeni NSM yöntemi Algoritma 7.4'te verilmiştir. Açıktır ki, yeni oluşturulmuş NSM durum fikri Algoritma 7.1'deki 2. satırına da uygulanmalıdır.

Geçişsel darboğaz fikri bir NSM durumunun bir alt görev ayırıcı olarak kullanılabilecek bir geçiş bilgisine sahip olduğu sezgisine dayanır. Bununla birlikte, geçiş bilgileri hala gizli durumların algısal örtüşmesi probleminin belirsizliğini miras aldığından, NSM_{TBx} algoritması orjinal algoritmada eksik olan önceki gözlem bilgisi ile NSM durumunu genişleterek boşluğu doldurmaktadır.



Şekil 7.3. Deneylerde kullanılan McCallum'ın maze ve hallway problemleri

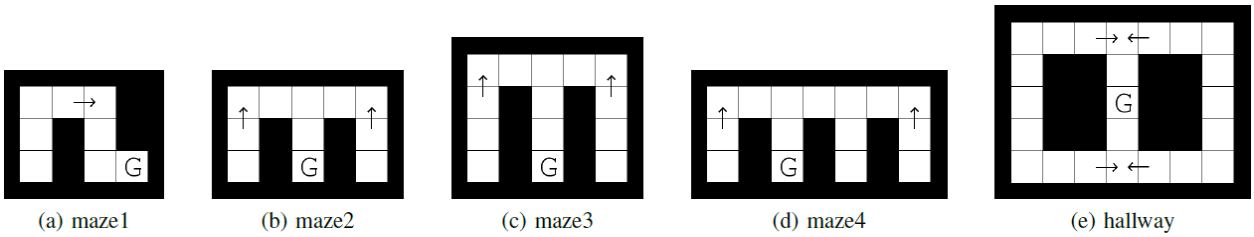
7.4 Deneyler

7.4.1 Deney Düzeneği

Deneyler, orijinal NSM, Geçişsel Darboğazlı NSM (NSM_{TB}) ile Geçişsel Darboğazlı ve Genişletilmiş Durumlu NSM (NSM_{TBx}) kullanılarak McCallum'ın ünlü problemleri (A. McCallum 1996a) için (Şekil 7.3) yapılmıştır. Tüm algoritmalar için bellek sınırsız olarak kabul edilmiştir. Çevre, bir duvara çarpmak, komşu bir duruma geçmek ve hedef duruma ulaşmak için maze problemleri için sırasıyla -1.0, -0.01, 1.0 ve hallway problemi için -1.0, -0.1 ve 5.0 ödüllerini vermektedir.

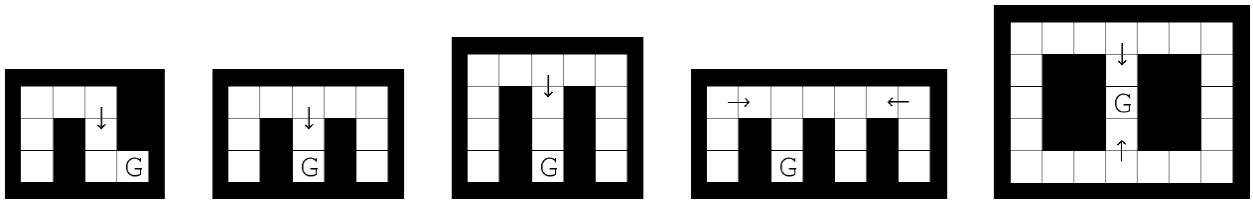
Geçişsel darboğazlar, Algoritma 7.3 ve Algoritma 7.4 için, yapılan deneylerden önce etmene sağlanmıştır. NSM_{TB} için sağlanacak olan darboğazlar, öğrenme esnasındaki tuzaklardan

kaçınmak için dikkatle seçilmiştir ve Şekil 7.4'te gösterilmiştir. Bu geçişler deneyimlendiğinde, etmen 0.0 değerinde yapay bir ödül almıştır.



(a) maze1 (b) maze2 (c) maze3 (d) maze4 (e) hallway

Şekil 7.4. NSM_{TB}için verilen geçişsel darboğazlar



(a) maze1 (b) maze2 (c) maze3 (d) maze4 (e) hallway

Şekil 7.5. NSM_{TBx}için verilen geçişsel darboğazlar

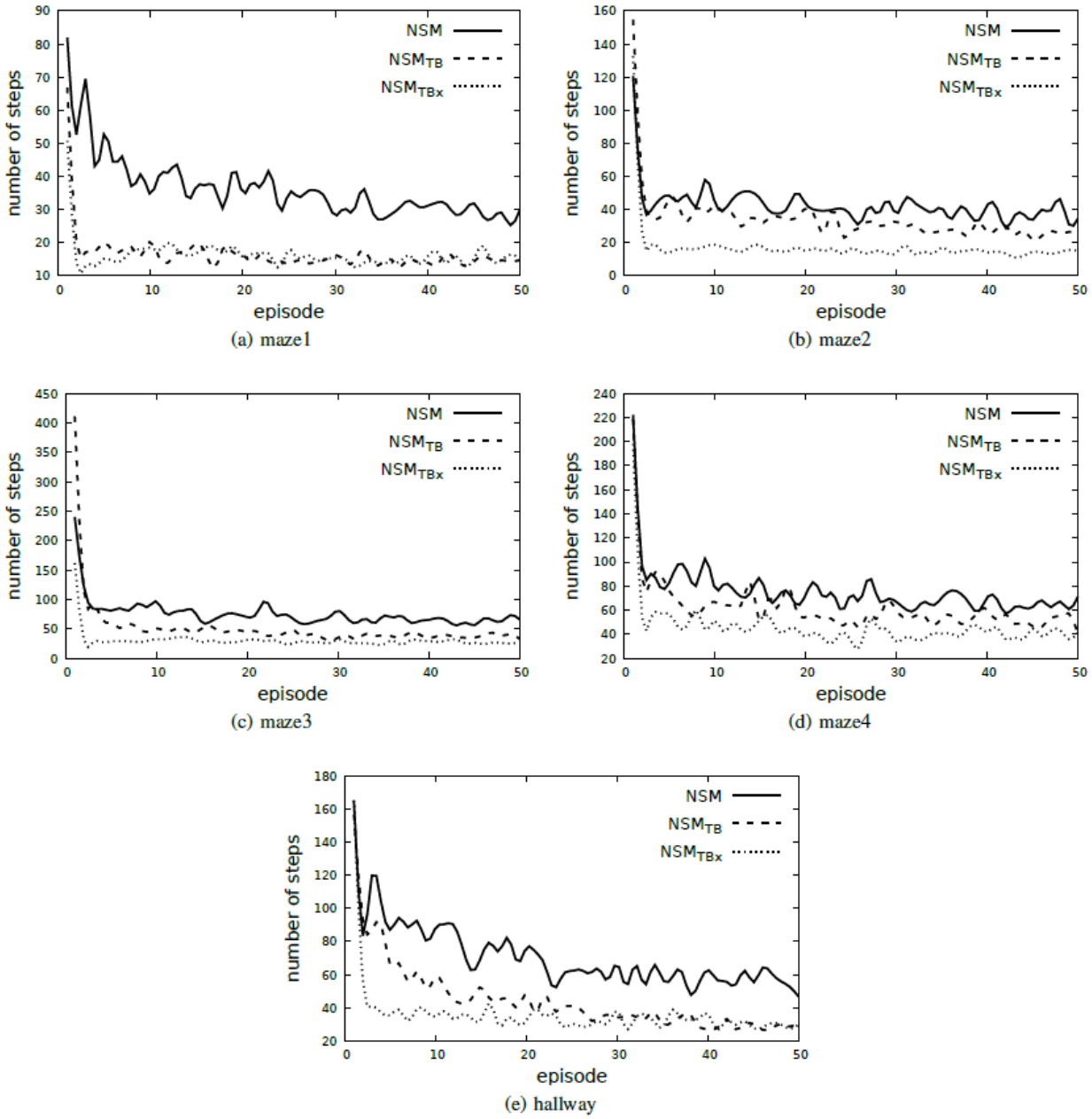
Bununla birlikte, genişletilmiş durum için, geçişsel darboğazlar Şekil 7.5'te gösterildiği gibi farklı bir şekilde seçilmiştir. maze4, maze2 ve maze3'ten farklı olarak, hedef durum koridorunun başlangıcındaki geçiş, benzer bir koridorun varlığı nedeniyle etmen için bir tuzağa neden olmaktadır. Dolayısıyla, bu geçiş yerine Şekil 7.5d'de gösterildiği gibi başka iki karmaşıklığa neden olmayacak geçiş sağlanmıştır. Yine, bu geçişlerin tümü 0.0 yapay bir ödüle sebep olmaktadır.

Tüm öğrenme yöntemleri için, k (komşuların sayısı), α (öğrenme oranı), γ (azaltım faktörü) ve ϵ (keşif olasılığı) 4, 0.9, 0.9 ve 0.1 olarak seçilmiştir. Başlangıçta, etmen, bir bölüm başlangıcında *G* ile gösterilen hedef durum haricinde rastgele bir duruma yerleştirilmiştir. Deney sonuçları, 200 koşumun ortalaması şeklinde verilmiştir.

7.4.2 Sonuçlar ve Tartışma

Şekil 7.6, NSM_{TBx}'in öğrenme performansının, özellikle belirsiz geçişlerin yoğun olduğu problemlerde, hedefe ulaşmak için gereken adımların sayısı bakımından hem NSM hem de NSM_{TB}'den daha iyi olduğunu göstermektedir. Şekil 6a'da her iki NSM_{TB} ve NSM_{TBx} yöntemi NSM'den daha iyi performans göstermesine rağmen aralarındaki fark o kadar da belirgin

değildir, çünkü bu problem daha az belirsiz geçiş içerdiğinden NSM_{TB} için de nitelikli bir darboğaz tanımlama imkanı mevcuttur. Bununla birlikte, diğer problemlerin sonuçlarında, NSM_{TBx} lehine iyileşme daha açıktır; bu da geçişsel darboğaz fikrinin, NSM gibi örneğe dayalı bir RL algoritmasında öğrenmeyi hızlandırmak için bir görev bölümlemesi yöntemi olarak kullanılabileceğini göstermektedir; ve bu fikir darboğazların kalitesi genişletilmiş durum tanımlaması ile arttığında çok daha iyi sonuçlar vermektedir.



Şekil 7.6. Her bir problem için yöntemlerin öğrenme performansları

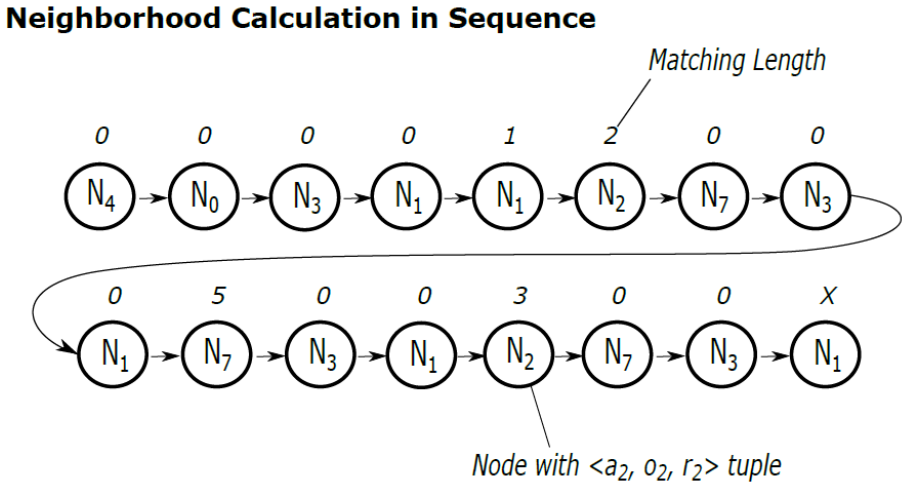
8 İNDEKSLEMELİ EN YAKIN DİZİ BELLEĞİ YÖNTEMİ

Örneğe dayalı metotların, saklı durumlu pekiştirmeli öğrenme problemlerini çözmekteki başarısı kanıtlanmıştır. En Yakın Dizi Belleği (*Nearest Sequence Memory, NSM*) yöntemi de En Yakın k-Komşu (kNN) yaklaşımını benimseyen ve yaygın bir şekilde kullanılan örneğe dayalı bir yöntemdir. Bu yöntem etmenin ortamdaki etkileşimlerini hafızada eylem, gözlem ödül üçlülerinin zinciri halinde saklar. Bulunduğu duruma en yakın geçmiş deneyimini bulmak için de zinciri yinelemeli bir şekilde taramaktadır. Bu çalışmada bir indeksleme mekanızması eklentisiyle bu taramanın gereksiz karşılaştırmalardan nasıl kurtulacağı sunulmaktadır. Deneyler yöntemin çalışma süresinde ciddi bir azalmaya işaret ederken, öğrenme performansı korunmaktadır.

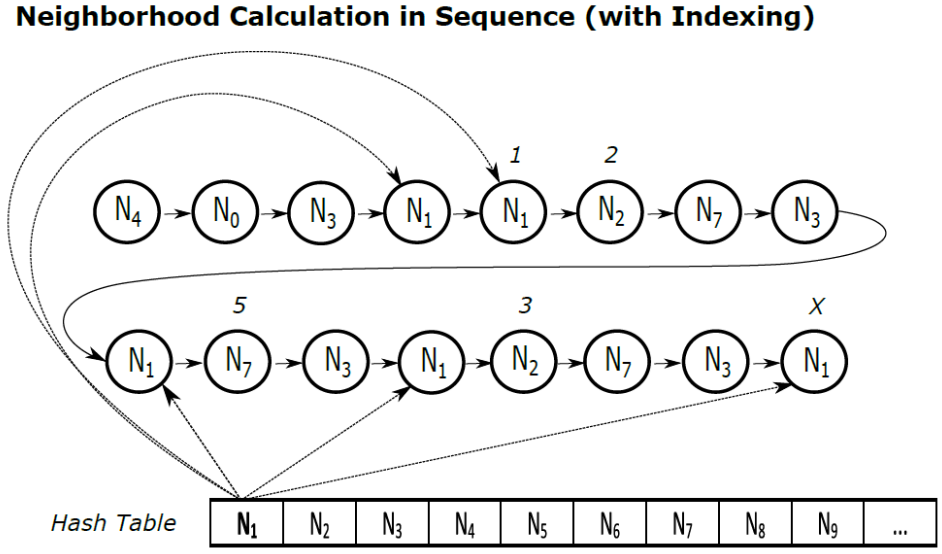
8.1 En Yakın Dizi Belleği Algoritmasında Durum İndekslemesi

En Yakın Dizi Belleği (NSM) algoritması etmenin geçmişini ham bir şekilde saklar (A. McCallum 1996a). Etmenin mevcut durumunun komşu düğümlerini bulmak için bu geçmişteki her bir düğümü tek tek kontrol etmektedir. Bu işlem, etmenin her bir adımı için tekrar tekrar yapılması gerektiğinden zaman bakımından çok masraflı olmaktadır. Çok uzun sürebilen bu işlem, yalnızca mevcut durum ile en az bir eylem, gözlem, ödül üçlüsü eşleşmesine sahip düğümleri hesaplamaya dahil edilerek hızlandırılabilir. Bahsedilen bu düğümlerin zincirdeki, yine bu üçlüleri anahtar olarak kullanan bir eşleme (hashing) mekanizması oluşturarak adreslenmesiyle kolayca tespit edilebilir. Öğrenme süreci boş bir eşleme haritası (hash-map) ile başlar ve etmen yeni bir üçlü ile karşılaştığı sürece büyür. Yani kullanılan eşleme haritasının büyüklüğü sınırlanmaz. Etmenin geçmiş zincirindeki her bir düğüm yine eylem, gözlem, ödül üçlüsü ile tanımlı olduğundan, bu indeksleme mekanizması verilen anahtar ile ilgili düğümleri bulmak için ayrıca bir hesaplama gerektirmemektedir.

Şekil 1, indeksleme mekanizmasının nasıl çalıştığını göstermektedir. Geçmiş zincirine yeni bir düğüm eklendiği zaman, aynı zamanda eşleme tablosunda (hash-table) da indekslenir. Böylece, tüm düğümleri gezmek yerine, dizide en yakın komşu olmaya aday düğümler ortalama O(nlogn) karmaşıklığıyla eşleme tablosundan kolayca çekilebilir. Şekil 1'de görüldüğü gibi indeksleme mekanizması, on beş düğümün tamamı yerine yalnızca ilgili dört düğümü gezerek komşuluk hesabını yapmaktadır. İndeksleme mekanizmasının bu komşuluk hesabının süresine olan etkisi, etmenin geçmiş zinciri öğrenme boyunca yeni düğümlerin eklenmesiyle uzadıkça ve daha da önemlisi birbiriyle eşleşmeyen düğümlerin sayısı arttıkça daha da belirgin hale gelmektedir.



(a) NSM dizisinde komşuluk hesabı



(b) NSM dizisinde indekslemeyle komşuluk hesabı

Şekil 8.1: Komşuluk hesabı için etmenin geçmişini yinelemeli bir şekilde tüm düğümleri gezen orijinal NSM algoritması ile (a), yalnızca ilgili 4 düğümün kıyaslayan İndekslemeli NSM algoritmasının (b) karşılaştırmalı örneği.

Algoritma 8.1, Durum İndekslemeli NSM Algoritmasını göstermektedir. Burada, "*" sembolü ile gösterilen satırlar Algoritma 7.2'de verilen NSM'in orijinal algoritmasından farklı olan kısımları belirtmektedir. İndeksleme mekanizması, komşuluğun nasıl hesaplanacağına müdahale etmemektedir. Dolayısıyla bu hesaplama daha önceden olduğu gibi Algoritma 7.1'de verilen NEIGHBORHOOD isimli prosedürün çağrılmasıyla gerçekleşmektedir. Üstelik, bu indeksleme mekanizması orijinal NSM algoritmasının eylem seçme mekanizmasını etkileyecek olan bir değişiklik de gerektirmez. Bu da öğrenme performansının neden korunduğunu açıklar. İndeksleme uzantısının gerektirdiği tek şey, kısıtlı hafıza durumunda zincirden en eski düğümün silinmesi sırasında, eşleme tablosundan da ayrıca silinmesidir. Fakat, bu işleme zaman bilgisi dahil edilerek ve eşleşen düğümler arasında en erken gözlenmiş olanı arayarak kolayca gerçekleştirilebilir.

Algorithm 1 NSM with State Indexing

```
 1: procedure NSM − i
2-4:     ...                                              ▷ unchanged
 5:     while learning continues do
6-8:         ...                                          ▷ unchanged
  ⋆          CandidateNodes ← ∅
  ⋆          for each node N_i in HashMap[N_t] do
  ⋆              add N_{i+1} to CandidateNodes
  ⋆          end for
  ⋆          for each node N_i in CandidateNodes do
  ⋆              n ← NEIGHBORHOOD(N_t, N_{i−1})
  ⋆              add ⟨N_i, n⟩ pair to NN
  ⋆          end for
  ⋆          append link to history entry of N_t to HashMap[N_t]
13-36:           ...                                      ▷ unchanged
37:          t ← t + 1
38:     end while
39: end procedure
```

Algoritma 8.1: İndekslemeli NSM yöntemi

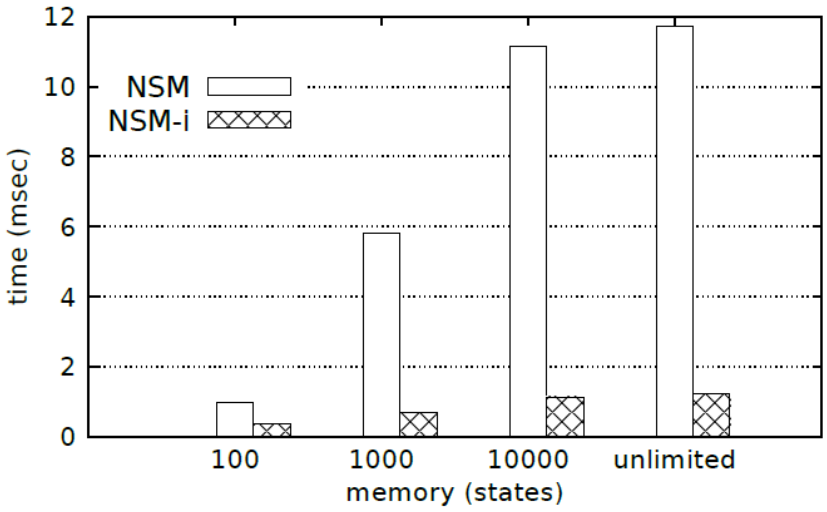
8.2 Deneyler

Deneyler; NSM ve İndeksli NSM (NSM-i) yöntemlerinin 100, 1000 ve 10000 düğüm ile sınırlandırılmış hafıza ve sınırsız hafızalı versiyonlarıyla McCallum tarafından oluşturulan Maze problemleriyle Hallway probleminde yürütülmüştür (A. McCallum 1996b). Her iki yöntem için de k (komşuların sayısı), α (öğrenme oranı), γ (azaltım faktörü) ve ε (keşif olasılığı) sırasıyla 4, 0.9, 0.9 ve 0.1 olarak seçilmiştir. Deneylerin her bir bölümünün başlangıcında, etmen hedef durum haricinde bir duruma rastgele yerleştirilmiştir.

	100		**1000**		**10000**		**Unlimited**	
	NSM	NSM-i	NSM	NSM-i	NSM	NSM-i	NSM	NSM-i
Maze 1	-0.0333	-0.0333	0.0045	0.0049	0.0128	0.0134	0.0119	0.0128
Maze 2	-0.0501	-0.0503	-0.0013	-0.0008	0.0069	0.0084	0.0069	0.0070
Maze 3	-0.0494	-0.0501	-0.0101	-0.0101	-0.0029	-0.0023	-0.0027	-0.0018
Maze 4	-0.0556	-0.0557	-0.0111	-0.0104	-0.0031	-0.0020	-0.0025	-0.0022
Hallway	-0.4562	-0.4560	-0.1706	-0.1624	-0.0859	-0.0861	-0.0878	-0.0824

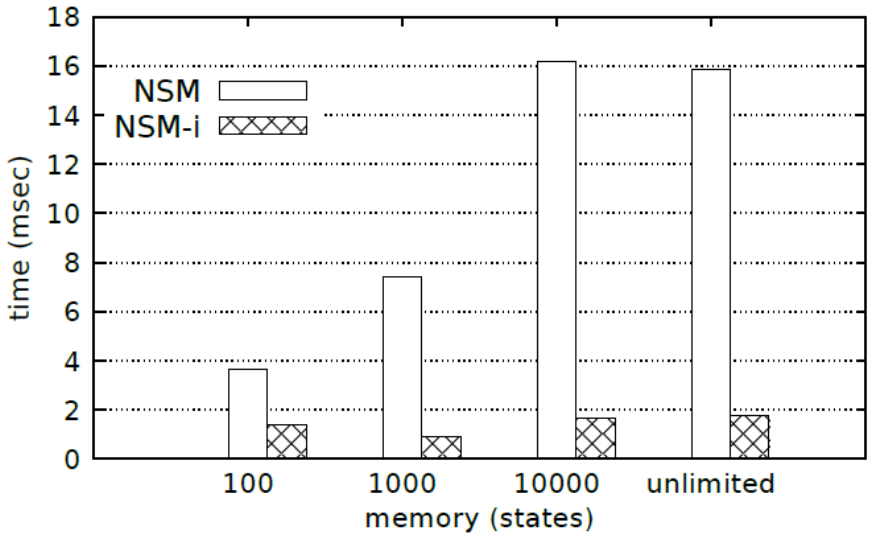
Tablo 8.1: Farklı hafıza düzenlemeleri ve problemler için yöntemlerin adım başına ödül bakımından öğrenme performansları

Tablo 8.1'de gösterilen sonuçlar ve Şekil 8.2'den Şekil 8.6'ya kadar çizilen tüm grafikler 100 bölüm halinde yürütülen 200 deneyin ortalaması alınarak elde edilmiştir. Tabloda gösterildiği üzere, adım başına düşen ortalama ödüller bakımından yöntemlerin öğrenme performansı arasında belirgin bir fark yoktur. Farklı hafıza düzenlemeleriyle verilen problemler üzerinde koşturulan bu iki yöntemden orijinal NSM algoritmasının adım başına düşen ortalama ödülü 0.10617 standart sapma ile -0.04898'dir. NSM-i algoritmasının ise benzer şekilde 0.10574 standart sapma ile -0.04797'dir. Bu da öğrenme performansı bakımından elde edilen sonuçların çok farklı olmadığını doğrulamaktadır.

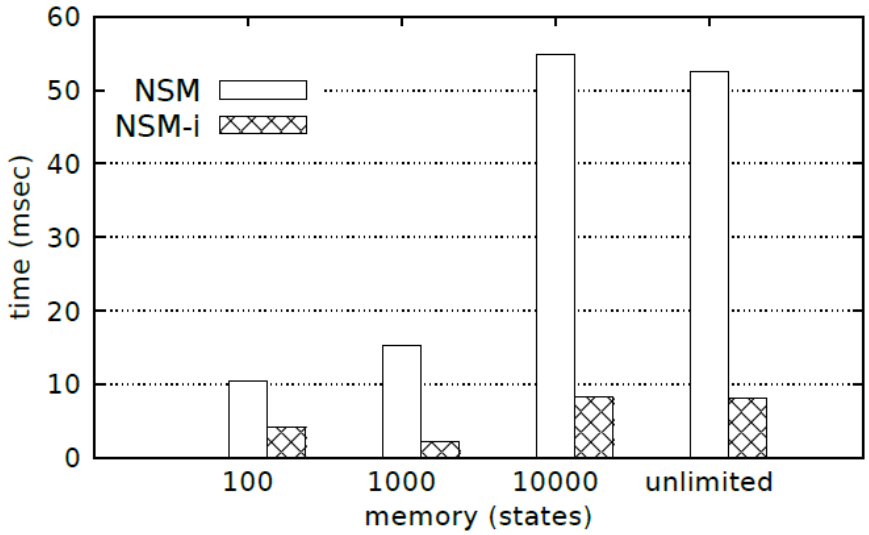


Şekil 8.2: NSM versiyonlarının Maze 1 problemindeki işlem süreleri

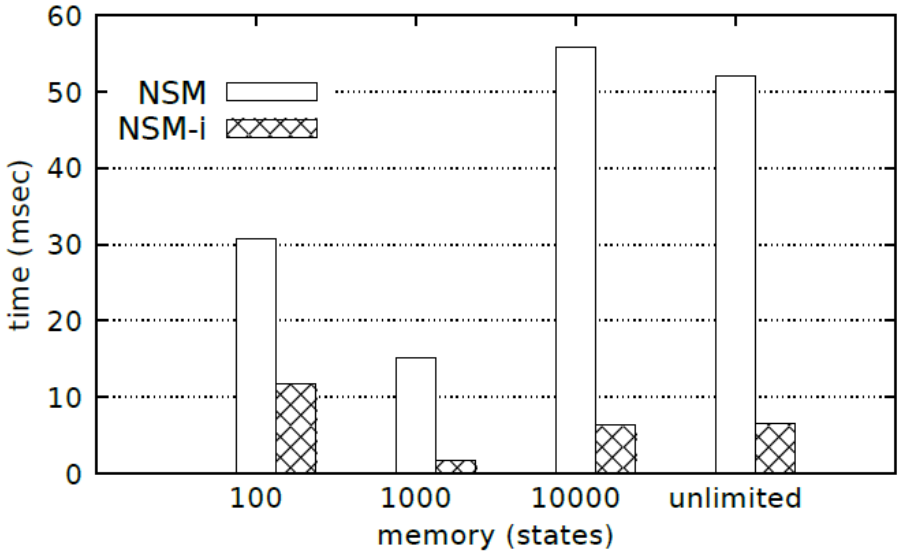
Şekil 8.2, NSM versiyonlarının Maze 1 problemindeki işlem sürelerini göstermektedir. Hafıza boyutu büyüdükçe bu sürelerin arttığı gözlenir. Fakat, NSM-i yönteminin tüm hafıza düzenlemelerinde orijinal NSM'den daha az zamana ihtiyaç duyduğu açıkça görülmektedir.



Şekil 8.3: NSM versiyonlarının Maze 2 problemindeki işlem süreleri



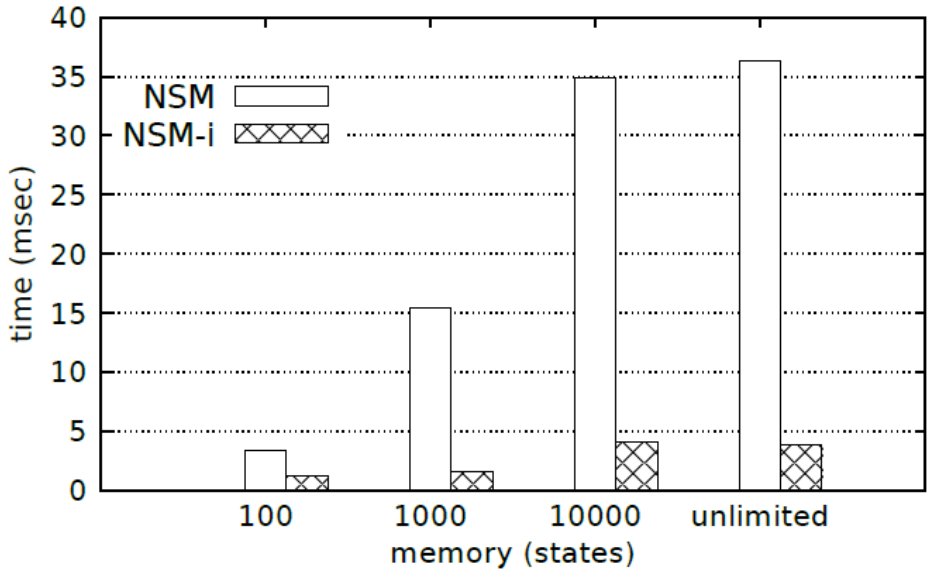
Şekil 8.4: NSM versiyonlarının Maze 3 problemindeki işlem süreleri



Şekil 8.5: NSM versiyonlarının Maze 4 problemindeki işlem süreleri

Şekil 8.3, Şekil 8.4 ve Şekil 8.5; NSM versiyonlarının sırasıyla Maze 2, Maze 3 ve Maze 4 problemlerinde ihtiyaç duyduğu süreleri göstermektedir. NSM-i yöntemi, bu problemlerde, farklı hafıza düzenlemelerinin tümüyle, orijinal NSM yöntemine göre daha hızlı çalışmaktadır. Her iki yöntem de hafızanın büyüklüğü değiştikçe benzer davranışlar sergilemektedir. Bu şekillerde gözleneceği üzere, NSM yöntemleri 10000 düğümlük hafıza ile, sınırsız hafıza ile çalıştırıldığından daha fazla zaman gerektirmektedir. Bu da NSM etmenlerinin bu problemler

üzerinde öğrenmesinin yakınsamasının 10000 düğümlük hafıza ile daha fazla zaman gerektirdiğini gösterir. Başka bir ifadeyle, etmenler bu problemleri 10000 düğümlük hafıza ile öğrenmesi daha uzun sürmektedir. Benzer bir çıkarım, Şekil 8.5'te görüşebileceği gibi, Maze 4 probleminde 100 ve 1000 düğümlük hafıza düzenlemesinin sonuçlarının karşılaştırılması için de yapılabilir.



Şekil 8.6: NSM versiyonlarının Hallway problemindeki işlem süreleri

Son olarak, NSM etmenlerinin Hallway problemindeki çalışma süreleri Şekil 8.6'da görülebilir. Bu problem için, NSM versiyonlarının ihtiyaç duydukları süreler, Maze 1 probleminde olduğu gibi kullanılan hafızanın büyüklüğüyle doğrudan paralellik göstermektedir. NSM-i yönteminin performansı harcanan süreler bakımından orijinal NSM yönteminden bu problem için de gelişme göstermiştir. Özetlemek gerekirse, indeksleme mekanizması, NSM algoritmasını hangi hafıza düzenlemesiyle uygulandığından bağımsız olarak hızlandırmakta ve bunu yaparken yöntemin öğrenme performansını da olumsuz yönde etkilememektedir.

9 ALT HEDEFE ULAŞMA AMAÇLI TERCİHLER İÇİN ETKİN BAŞLATMA KÜMESİ OLUŞTURMA

Tercihler çerçevesi, zamansal soyutlamalar aracılığıyla öğrenme hızını geliştirmek için temel teşkil eden önemli modellerden biridir. Bir tercih esas olarak üç unsurdan oluşur: başlatma kümesi, tercihin yerel politikası ve sonlandırma koşulu. Belirli bir problem için yüksek kaliteli sonlandırma koşullarının nasıl elde edileceğine odaklanan çeşitli girişimlerde bulunulmasına rağmen, başlangıç kümesi üretiminin etkisi nispeten araştırılmamıştır. Bu çalışmada, olayların yakın tarihinin bir analizi ile faydalı başlangıç kümesi unsurları türetmek için etkili bir hedef odaklı sezgisel yöntem öneriyoruz (Demir, Çilden, ve Polat 2017a). Yöntemin etkinliği çeşitli kıyaslama problemleri üzerinde denenmiş ve sonuçlar tartışılmıştır.

9.1 Giriş

Makine öğrenme çerçevelerinin çoğunda olduğu gibi, pekiştirme öğrenimindeki zorluklardan biri de, problem boyutunun çözüm prosedürünün zaman karmaşıklığına olan ters etkisidir (Sutton ve Barto 1998). Bu sorunla başa çıkmak için klasik bir yaklaşım, *böl ve fethet* stratejisidir. Temel olarak bu stratejiye dayanarak, boyutsallık sorununun olumsuz etkilerini azaltmaya çalışan birkaç yaklaşım vardır. Fikir şu ki, eğer problemi alt problemlere bölebilirse, alt problemleri çözmek daha kolay olabilir ve sonra bu alt çözümler sorunun cevabını elde etmek için birleştirilebilir. Çözüm açısından bakıldığında, hiyerarşik algoritmik yapılarla bu ayrıştırmanın yakın bir benzerliği vardır; alt-çözümleri, yararlı olduğunda, *soyutlama* ya da *makro* olarak ele alır. Bu girişimler alanda üç önemli çözüm ailesine yol açmıştır: tercihler çerçevesi (Sutton, Precup, ve Singh 1999)**,** değer fonksiyonu ayrıştırması (Dietterich 2000) ve hiyerarşik soyut makineler (Parr ve Russell 1998).

Yaklaşımlar arasında, *tercih* çerçevesi, genelliği ve uygulama kolaylığı nedeniyle dikkat çekerken, bir *tercih* , zamansal genişletilmiş bir *soyut eylemden* (veya *makro eylem*) farklı değildir. Tercihin orijinal tanımı, soyutlamaların öğrenmeden önce bir şekilde sağlandığını varsaydığından, *alt hedef tespiti* (Menache, Mannor, ve Shimkin 2002; Simsek 2008) gibi bir problem bölümleme şeması ile birleştirildiğinde daha etkili olur.

Bir tercihin tanımı, bir başlatma kümesini (bir tercihin başlayabileceği durumlar) ve bir sonlandırma koşulunu (bir tercihin nasıl sona erdiğini) içerir. Bölümleme mekanizmaları, alt

hedef keşfi gibi, genellikle bölümlendirme etkinliğini belirlediğinden, sonlandırma koşulunu doğru bir şekilde nasıl tanımlayacağına bakarlar. Bununla birlikte, çok az sayıda çalışma, başlangıç kümesiyle uğraşmaktadır ve tüm yükü ilgili eylem seçim stratejisine bırakmaktadır.

Bu çalışmada, başlatma kümesinin seçilme yolunun sonlandırma noktalarını nasıl belirlediğiniz kadar önemli olduğunu ve çözüm kalitesi üzerinde doğrudan bir etkiye sahip olabileceğini iddia ediyoruz. Böylece, daha yüksek kalite seçenekleri elde etmek ve öğrenme hızını arttırmak amacıyla etkin başlatma kümeleri oluşturmak için yeni bir yaklaşım önermekteyiz.

9.2 Arka Plan

Pekiştirmeli Öğrenme (RL), bir karar problemini çözmek için, problem alanından geçerken çevrim içi olarak alınan geri bildirimleri kullanan bir makine öğrenme çerçevesini tanımlar. Aynı zamanda, doğru kararları öğrenmeden önce sunmadığı için öğreticiyle öğrenme tekniklerinden de farklıdır, ancak etmenin, çevresel geri bildirimin işlenmesi yoluyla doğru kararları kendisi üretmesi gerekir (Sutton ve Barto 1998).

RL bağlamında, bir karar problemi genellikle $< S, A, T, R >$olarak tanımlanan bir *Markov Karar Süreci* (MDP) ile modellenir. Burada;

- S, durumlar kümesi,
- A, eylemler kümesi,
- $T: S\ x\ A\ x\ S \rightarrow [0,1]$, geçiş fonksiyonu ($\forall s\ \in S, \forall a \in A, \sum_{s' \in S}\ \ T(s,a,s') = 1$),
- $R: S\ x\ A\ \rightarrow \Re$, s durumunda a eyleminin alınması durumunda gelen ödülü belirleyen ödül fonksiyonudur.

Belirli bir karar davranış şeması, bir durumda bir eylem seçme olasılığı olan $\pi: S\ x\ A \rightarrow [0,1]$olarak tanımlanan ***hareket tarzı*** olarak adlandırılır. RL'nin amacı, öğrenme etmeni tarafından alınan toplam beklenen ödülü en yüksek seviyeye ulaştıran en iyi π^* hareket tarzını bulmaktır. RL'nin öğretmensiz özelliği, ödül ve geçiş fonksiyonlarının bilinmediği gerçeğidir, aksi takdirde en iyi hareket tarzı klasik dinamik programlama teknikleri kullanılarak kolayca bulunabilir. Bununla birlikte, π^*, etkin olarak, ***değer fonksiyonu*** (yani, hedefe gitme sürecinde bir durumda olmanın değerini veren fonksiyon) aşamalı olarak tahmin edilerek bulunabilir. *Artımlı tahmin* yaklaşımı, değer fonksiyonunu hesaplamak için bir hareket tarzının izlenmesiyle elde

edilen farklı gezinmelere göre ortalama birikimli ödülleri kullanır ve çoğu RL algoritmasının merkezi fikrini ortaya çıkarır, bu da *zamansal fark* (TD) olarak adlandırılır (Sutton 1988)**.**

Durum değerleri yerine eylem değerleri (yani Q değerleri) kullanan iyi bilinen bir TD algoritması, Q-Learning (Watkins 1989) olarak adlandırılır ve sadeliği ve kullanım kolaylığı nedeniyle tercih edilir. Q-Learning için güncelleme kuralı;

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha[r + \gamma \max_{a' \in A} Q(s',a')] \qquad (1)$$

Formül 9.1 Q-Learning

şeklindedir. Burada $\alpha \in [0,1]$*öğrenme adımı*, $\gamma \in [0,1)$ *azaltım hızı*'dır ve s etmenin a eylemini alarak s' durumuna ulaştığı durumdur. Q-Learning'in, standart olasılıksal yaklaşım varsayımları altında en iyi eylem değeri fonksiyonuna yaklaştığı gösterilmiştir.

9.2.1 Tercih Çerçevesi

MDP modeli, gerçek dünya problemleri için gerçekçi bir varsayım olmayan, her bir ayrı zaman adımında bir eylem gerçekleştirildiğini varsayar. Bu varsayımın bir gevşemesi olarak, *yarı Markov Karar Süreci* (SMDP) modeli olasılıksal zaman süresine sahip eylemlere izin verir. Aslında $< S, A, T, R, F >$olarak tanımlanan SMDP, MDP'nin bir uzantısıdır.

- S, A, T, R bir MDP tanımlar ve
- $F(t \mid s, a)$, s durumunda başlandığında a eyleminin t anında tamamlanma olasılığıdır.

Bariz bir şekilde, MDP, adım fonksiyonunun birer atladığı özel bir SMDP versiyonudur. SMDP modelinde, bir politika hâlâ durumlardan eylemlere kadar bir haritalamadır, bu yüzden Bellman denklemleri (Bellman 1957) hala en uygun politika için sağlanmaktadır (Bradtke ve Duff 1994b).

Tercihler çerçevesi (Sutton, Precup, ve Singh 1999)**,** MDP modelinin üstündeki birleşik eylemler aracılığıyla zamanlanmış eylemleri tanımlayıp çağırmanın bir yolunu tasarlayan SMDP modeline dayanan önemli soyutlama mekanizmalarından biridir. Halen MDP'lerin birim zaman geçiş dinamiklerini koruyarak, bir eylem, bir dizi ayrık zaman adımı için sürdürebileceği ve bir *tercih*

olarak adlandırılabileceği anlamda genelleştirilebilir. Diğer bir deyişle, “tercih” bir “eylem”in zamansal olarak genişletilmiş bir karşılığıdır.

Bir tercih $< I, \pi_{option}, \beta >$ şeklinde tanımlanır. Burada;

- I *başlatma kümesi* olup, seçeneğin başlatılabileceği durumlar kümesidir.
- π_{option} tercihin yerel politikasıdır ve
- β *sonlandırma koşulu* tarafından tetiklenen bir olasılık dağılımıdır.

Bir seçenek, I durumunda bir etmen tarafından başlatıldığında, seçenek β ile belirlenen belirli bir koşulda sona erene kadar seçeneğin yerel politikası π_{option} izlenir. Öte yandan, bir *Markov tercihi*'nde, eylem seçimi ve tercih sonlandırma kararı sadece mevcut durum temelinde yapılır. Bu çalışma, eski tercihlerin tanımına odaklanmaktadır.

Q-Learning’in doğal bir uzantısı, tercihleri içine alan Macro-Q Learning’dir (Mcgovern, Sutton, ve Fagg 1997). Burada her ilkel eylemin değeri, Q-Learning güncelleme kuralında verildiği gibi güncellenir. Bir tercihin değeri ise aşağıdaki kurala göre güncellenir:

$$Q(s_t, o_t) \leftarrow Q(s_t, o_t) + \alpha \times (\gamma^n \times \max_{o'} Q(s_{t+n}, o') - Q(s_t, o_t) + r_{t+1} + \gamma r_{t+2} + \ldots + \gamma^{n-1} r_{t+n}) \quad (2)$$

Formül 9.2 Macro Q-Learning

Burada s_t, $o_t$tercihinin başlangıç durumu olup, n tercih kullanılırken atılan adım sayısıdır. s_{t+n}, tercihin durduğu durumken o', s_{t+n}'den alınabilecek en yüksek değere sahip olan tercih ve r_{t+i},$t + i$ zamanında alınan ödüldür. Ödül, alındığı zamana göre indirilir.

9.2.2 Otomatik Tercih Üretimi

Tercihler çerçevesi, anlamlı veya yararlı tercihler tasarlamak için herhangi bir strateji getirmez. Çerçevenin dinamikleri hakkında biraz bilgi veren “yönergeler” olmasına rağmen, biçimin kendisi değerli bir tercihi tanımlamak için yöntemsel yollardan yoksundur. Genel bir kural, “bir tercihin başlatma kümesi ve sonlandırma koşulu, kendi uygulama alanını potansiyel olarak yararlı bir şekilde sınırlandırır” şeklindedir (Sutton, Precup, ve Singh 1999)**.**

Bununla birlikte, mevcut otomatik tercih oluşturma algoritmalarının birçoğu, (açık bir strateji olarak) bir bölümlendirme girişiminin, durum uzayında bölme noktaları veya bölgeleri bulması ve bu darboğaz durumlarını kullanarak tercihlerin sonlandırma koşullarının türetilmesini sağlaması nedeniyle, sonlandırma durumuna odaklanmaktadır.

Diğer yandan, başlangıç kümeleri genellikle daha sonra öğrenim sırasında eylem seçim mekanizmasının seçimleri kısıtlamasını umarak, “sonlandırma durumları dışındaki tüm durumlar” gibi daha basit bir öncül ile tanımlanır. “Diğer bütün durumlar” sezgisi, büyük olasılıkla başlatma kümelerinin inşası için en basit fikirlerden biridir (Jong, Hester, ve Stone 2008)**.** Bazı yöntemler bu fikri, problem uzayıyla ilgili dolaylı veya dolaysız bilgiler sunarak geliştirir. "Tercih gecikmesi" gibi kısıtlayıcı parametreler, tercih uzunluğu için bir problem bazlı bir özel sınır sağlamak için kullanılabilir (Simsek ve Barto 2004). Bazı metotlar, sonlandırma kümesindeki durumlar için bir ek *ulaşılabilirlik* kriteri kullanır (Stolle ve Precup 2002).

Göreceli olarak daha az incelenen bir yaklaşımda, alt sıradaki ortaklıklar, darboğazlar yerine araştırılmıştır (E. A. McGovern 2002; Girgin, Polat, ve Alhajj 2010). Bu esneklik, yöntemlerin geçmişlerin ortak kısımlarından yararlı tercihler oluşturmasına izin verir. McGovern’ın çalışması (E. A. McGovern 2002) bu amaçla bir geçmiş veri tabanı saklarken Girgin’in çalışması (Girgin, Polat, ve Alhajj 2010)**,** çalışmamıza ilham veren, aşırı bellek gereksinimini ortadan kaldıran bir kısayol geçmişi yöntemi ve sıkı bir ağaç yapısı sunar. İşlemler de dahil olmak üzere dizilere odaklanan Girgin’den (Girgin, Polat, ve Alhajj 2010) farklı olarak, çalışmalarımız, bir tercihin işe alınmasının yararlı olabileceği durumları tanımlamak için yalnızca durum bilgisiyle benzer tarihçe ağacı keşfini kullanır.

Klasik MDP modeline uymayan problemleri ele alan ve tercih kavramını genişleten çalışmalar da vardır. Sürekli durum alanı ile ilgili problemler için, keşfedilen becerilerin başlangıç kümesi, bir sonraki beceriye ait bir sonlandırma kümesi oluşturabilir ve bir beceri zinciri oluşturabilir (Konidaris ve Barreto 2009)**.** Ulaşılabilirlik sezgiseli de, farklı bir biçimde, faktörlü MDP modeli için (Jonsson ve Barto 2006) çalışılmıştır. Bu yöntemler ilgili genişletilmiş MDP modelleri için geçerlidir ve işimizle doğrudan ilgili değildir.

Bir tercihin üçüncü bileşeni olarak, *tercih politikası*'nı oluşturmanın yaygın bir yolu, *Tecrübe Tekrarı* (ER) mekanizmasıdır (Lin 1992b)**.** ER, geçmiş bölümlerin deposundan yararlanır ve etmen tekrar deneyimliyormuş gibi tekrar tekrar oynatır. Tercih üretmede, bir duruma ulaşma

politikası, ER tarafından ortamdan elde edilenler yerine yapay *içsel* ödüller kullanılarak etkili bir şekilde oluşturulabilir. Geleneksel olarak, ER ödülleri, hedef durumlarına ulaşmanın çok yüksek bir ödül almasını sağlayacak şekilde tasarlanırken, başlatma kümesinden ayrılmak cezalandırılır.

9.2.3 Otomatik Alt Hedef Tespiti

Bir önceki bölümde belirtildiği gibi, otomatik tercih oluşturma çalışmalarının çoğu, anlamlı bir sonlandırma koşulu oluşturmayı amaçlayan darboğazların veya alt hedeflerin tanımlanmasına odaklanmaktadır. Darboğaz durumlarını tespit etmenin pratik bir yolu, sorunun araştırılması sırasında ortaya çıkan geçiş grafiğinde alt hedef aramaktır. Burada, daha sonra önerilen algoritmamız için test yatakları olarak hizmet edecek üç yaklaşımı özetliyoruz.

Otomatik alt hedef tanımlamasında öne çıkan çalışmalardan biri olarak, Şimşek'in çalışmaları (Simsek 2008) yerel bilgi tabanlı yöntemler önerir. İdeal durumda, bir durumun bir alt hedef olup olmadığına kesin olarak karar vermek için geçiş fonksiyonu T tamamen bilinmelidir. Bununla birlikte, öğrenen etmen genellikle sadece yerel bilgilere, yani belirli bir süre içinde elde edilen deneyime sahiptir. Şimsek'in çalışması, bir *erişim durumu* kavramını, aralarında birkaç geçişi olan iki veya daha fazla bağlantılı bölgeyi birbirine bağlayan bir durum olarak tanımlar. Temel fikir şudur: Bir erişim durumu için arama yapan bir yöntemin, bir durumun bir hedef durum (bir erişim durumu) olarak sınıflandırılıp sınıflandırılmaması için deney boyunca sadece yerel istatistiklere sahip olmasına izin verilir. Bu şekilde bir durum için toplanan gözlemler daha sonra aşağıdaki Karar Kuralı'nda kullanılır:

$$\frac{n_+}{n} > \frac{\ln \frac{1-q}{1-p}}{\ln \frac{p(1-q)}{q(1-p)}} + \frac{1}{n} \frac{\ln(\frac{\lambda_{fa}}{\lambda_{miss}} \frac{p(N)}{p(T)})}{\ln \frac{p(1-q)}{q(1-p)}} \quad (3)$$

Formül 9.3 Karar Kuralı

Burada n bir durum için toplam gözlem sayısı, n^+bir durum için toplam pozitif gözlem sayısı, p bir hedef duruma (erişim durumu) verilen olumlu gözlem olasılığı, q bir hedef olmayan bir duruma verilen olumlu gözlem olasılığı, λ_{fa} yanlış alarmın maliyeti, λ_{miss}bir ıska maliyeti, $p(N)$ hedef olmayan durumların önsel olasılığı ve $p(T)$ hedef durumların önsel olasılığıdır.

Eşitsizlik tutarsa, o zaman durum bir erişim durumu olarak sınıflandırılır. Bu karar kuralı, toplanan gözlemlerin bir durumun etiketi hakkında karar vermek için yeterli olduğu zaman adımını belirler. Bu iki seviyeli mekanizma, tüm problem alanlarını geçmenin zaman maliyetini önleme yöntemlerini mümkün kılar. Diğer taraftan, parametreler probleme özgüdür ve deneylerle hesaplanmalı veya tahmin edilmelidir.

Yerel Aradalık, deneyimli geçişlerle elde edilen yerel etkileşim grafiklerini kullanan, Şimşek (Simsek 2008) tarafından önerilen çizge tabanlı yöntemlerden biridir. Bu, kararın yerel olarak toplanan bilgiler kullanılarak yapıldığı anlamına gelir. Bu yöntem, genellikle sosyal ağ analizinde kullanılmak üzere yatırılan bir çizge ölçüsüne, *aradalık* ölçüsüne dayanır (Brandes 2001) Bir çizgede bir düğümün aradalığı, grafikte, olası tüm kaynakların ve hedeflerin arasındaki en kısa yolların, düğümün üzerinden geçenlerinin bütün sayıya oranı şeklinde tanımlanır. Yerel aradalık, yerel bir etkileşim çizgesindeki bir durumun aradalığıdır. Resmi olarak bir çizge düğümü olan v'nin aradalığı;

$$\sum_{s\neq t\neq v} \frac{\sigma_{st}(v)}{\sigma_{st}} w_{st}$$

Formül 9.4 Aradalık

formülü ile hesaplanır. Burada σ_{st}, s düğümünden t düğümüne giden kısa yolların sayısını, $\sigma_{st}(v)$ bu yolların v düğümü üzerinden geçenlerinin sayısını, w_{st}ise s düğümünden t düğümüne giden yolun ağırlığını temsil eder. Her durum için elde edilen sonuç daha sonra Karar Kuralına gönderilir.

Yerel Kesimler, yine yerel etkileşimli çizgiyi iki parçaya bölmeyi amaçlayan bir başka yerel çizge tabanlı yöntemdir (Simsek 2008)**.** Burada çizge, normalleştirilmiş kesim (NCut) adlı bir ölçeğe göre ikiye bölünür (Shi ve Malik 2000). *Ncut* şu şekilde tanımlanır:

$$NCut(A,B) = \frac{cut(A,B)}{vol(A)} + \frac{cut(B,A)}{vol(B)}$$

Formül 9.5 NCut

Burada Ave B iki bölümdür, $cut(A,B)$, A'da başlayan ve B'de biten ayrıtların ağırlıkları toplamını ve $vol(A)$, A'da başlayan ayrıtların ağırlıkları toplamını temsil eder. Ayrıt ağırlıkları geçiş frekanslarını ile belirlenir. Algoritma, yerel etkileşim grafiğinin bölünmesi için spektral kümeleme kullanmaktadır. Ardından kesim kalitesini belirlemek için bölümlemenin *Ncut* değerini hesaplar.

Ncut değeri, bölümlerin iyi ayrılmış olduğunu belirten önceden belirlenmiş bir kesme eşiği (t_c) değerinden düşükse, bu kesimin sınır durumları pozitif bir gözlem alırken, diğerleri olumsuz gözlem alırlar. Yine, bu gözlemler, daha fazla eleme için Karar Kuralına gönderilir.

Q-Kesim, etkileşim çizgesindeki güçlü bağlanan alanların sınır durumlarını bulmayı temel alan bir başka otomatik alt hedef tanımlama yaklaşımıdır. Şimşek'in yöntemlerinden farklı olarak, *Q-Kesim* tüm durum geçişlerini görüntüleyerek darboğazları seçen genel bir kriter arar. Algoritma, bir göreceli ziyaret frekansı ölçeği ile hesaplanan geçiş kapasitelerine dikkat ederek, geçmiş kullanılarak oluşturulan geçiş grafiğinde bir *Max-Flow / Min-Cut* algoritması gerçekleştirir. Algoritmanın çalışması için bir kaynak durumu $s$ve bir hedef durum t gereklidir ve bu durumların nasıl seçileceği probleme özgüdür. Kesimin kalitesi, durum uzayının dengeli parçalarını ayırabilen az sayıda darboğaz durumunun belirlenmesiyle ortaya çıkar. Bu kalite, *oran kesim* bölme ölçeği (Wei ve Cheng 1991) ile sağlanabilir.

$$Q[N_s, N_t] \triangleq \frac{|N_s||N_t|}{A(N_s,N_t)}$$

Formül 9.6 Oran Kesim

$A(N_s, N_t)$, her iki kümeyi birleştiren geçiş sayısıdır ve kalite değeri probleme özgü kalite eşiği t_q'nun üzerinde olan kesintileri hesaba katar. Q-Kesim, bir darboğaz durumunun diğerine yol açtığı problemler için iyi çalışır, fakat birçok problem için, bir doğrusal bölümleme şeması uygun değildir. *Parçalı Q-Kesim* adlı genişletilmiş bir Q-Kesim sürümü, bir parçalama aracı olarak keşfedilen darboğazları kullanır ve böl-ve-yönet yaklaşımı ile parçalar üzerinde bu işlemi yineler.

9.3 Tarihçe Ağacı Sezgiseli

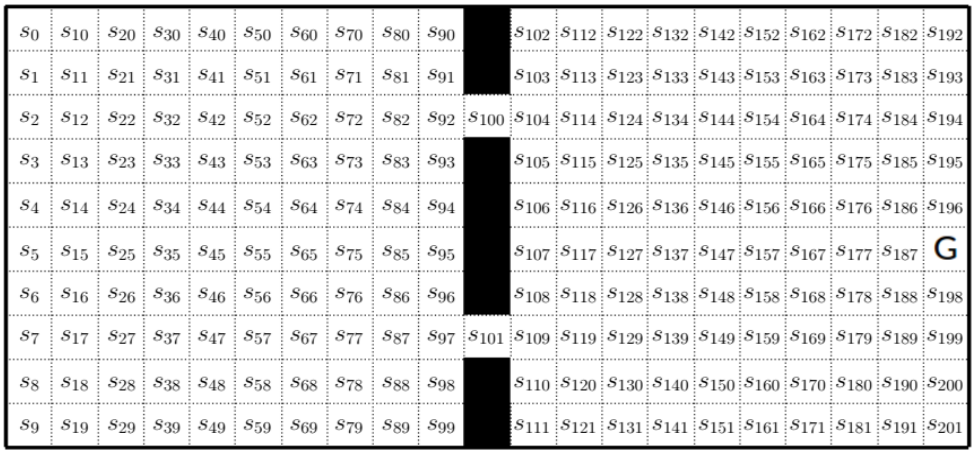
Bu çalışmada, geleneksel tercih yaklaşımında, tercih başlangıç kümelerini oluşturmacak yararlı durumları tanımlamak için sezgisel bir yöntem önermekteyiz. İlk bulgularımız (Demir, Çilden, ve Polat 2016)**,** yerel tarih analizine dayanan hedefe yönelik bir başlangıç kümesi oluşturmanın seçenek kalitesini artırabileceğini göstermektedir. İlgili literatürün çoğunda, başlangıç kümesi daha basit bir keşif kullanılarak tanımlanır ve odak genellikle iyi bir sonlandırma koşulunun belirlenmesi üzerinedir. Bununla birlikte, tarihçe ağacı tabanlı sezgisel yaklaşımımız, bir tercihin

başlatılmasının soyutlamanın ayrılmaz bir parçası olduğunu ve iyi bir sezginin, genel öğrenme performansını olumlu yönde etkileyebileceğini göstermektedir. Sezgisel metodumuzun etkinliğini çeşitli problemler üzerinde kapsamlı deneylerle göstermekteyiz.

Tercih oluşturma işleminin ayrılmaz bir parçası olarak, başlatma kümesinin durumlarının seçimi önemlidir. Tercihe ödül zirvelerinin göreceli yönelimi gibi ortam özelliklerine dair bazı bilgilerle yol göstermenin, özellikle öğrenmenin ilk aşamalarında, tercih performansı üzerinde olumlu etkisi olması muhtemeldir.

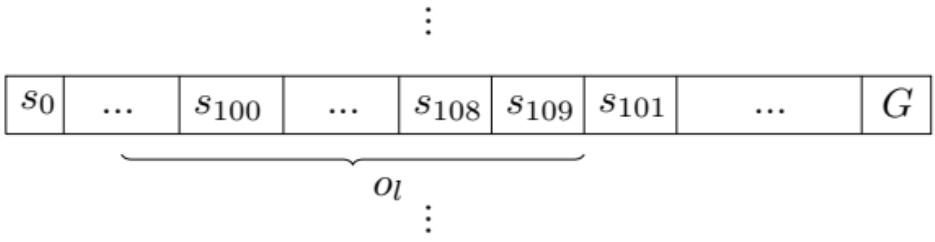
Bu çalışmanın amacı, başlangıç kümesini bir dizi kısıtlama kriterini kullanarak (muhtemelen bazı ek kısıtlama ölçütleri ile) basit bir "sonlandırma kümesinde olmayanlar" stratejisi kullanarak başlangıç kümesi üreten ve başlangıç kümeleri için yaygın olarak kullanılan hevesli yaklaşımı geliştirmeyi amaçlamaktadır. Sonlandırma koşullarının bir şekilde önceden belirlenmiş olduğu (alt hedef tanımlama, el ile tasarım, tercih aktarımı vb.) ve hedefe ulaşmada potansiyel olarak yararlı oldukları varsayımıyla, sezgimiz, bir sonlandırma durumunun, kısa bir süre için genellikle göreceli bir yönelime sahip olduğudur. Başka bir deyişle, durum uzayının belirli bir alt kümesinden bir (bir sonlandırma koşulu olarak atanmış) s durumuna ulaşılması, aslında ulaşılması gerekli olmayan ancak s'e ulaşan durum uzayının diğer bölümleriyle karşılaştırıldığında, hedefe giden yolda nispeten daha faydalı olabilir.

Popüler hevesli stratejilerden biri, bir bölüm içindeki son durumdan önce ziyaret edilenler arasından başlangıç kümesi durumlarını seçer. Tercihin son durumundan önce kontrol edilecek geçiş sayısı, *tercih gecikmesi* (o_l) olarak adlandırılan ve harici olarak sağlanan bir parametredir. Bu strateji, daha önce ziyaret edilen her bir durumdan son duruma ulaşan bir yol olduğu ve gelecekteki bölümlerde de benzer bir sıralamanın yeniden oluşacağı fikrine dayanıyor.

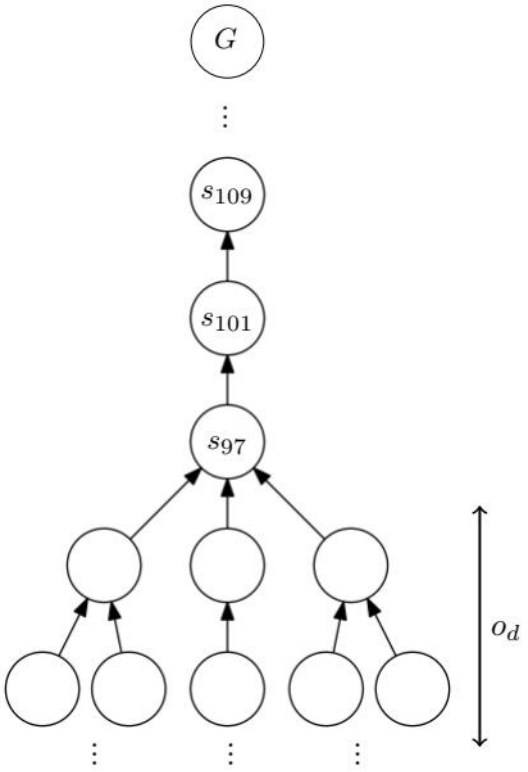


Şekil 9.1. 2 oda 2 kapı problemi

Ancak, bu her zaman böyle değildir. Son durumdan önce ziyaret edilen bazı durumların, son durumu ziyaret etmeksizin hedefe daha iyi bir rotayla ulaşması, yani uzun vadede daha yüksek bir faydaya sahip olmaları mümkündür. Şekil 9.1'de verilen *2 oda 2 kapı* probleminde böyle bir durumu inceleyelim. Şekil 9.2, etmenin alt hedef s_{100}aracılığıyla sağ odaya gittiği ve s_{101} alt hedefini ziyaret ederek sol odaya geri döndüğü örnek bir bölümdeki tercih gecikmesi yönteminin davranışını gösterir. Bu durumda, s_{101}'a ulaşma tercihi, sağ odadaki durumlar için kullanışlı değildir. Bununla birlikte, tercih gecikmeli sezgisel olan hevesli strateji bunu hesaba katmaz. Sadece bu durumları başlangıç kümesine koyar ve gelecekte gerekli ayrımı yapmak için eylem seçim mekanizmasına güvenir. Ne yazık ki, bu gereksiz tercih işlemi, öğrenme değerleri, öğrenmenin genel performansını olumsuz yönde etkileyen doygunluk seviyesine ulaşıncaya kadar öğrenme zamanını çalmaktadır. Ayrıca, son durumdan önce birkaç adımı kontrol etmek, durum tekrarlarını (veya döngülerini) dikkate almaz. Gözlemlenen geçmişte durumların tekrarı, genellikle tercihinin etkisini azaltarak üretilen başlatma setini kısaltır.



Şekil 9.2. Tercih Gecikmesi Yöntemi Örneği



Şekil 9.3. Tarihçe Ağacı Sezgiseli tarafından üretilen örnek bir ağaç

Algoritma 1'de verilen önerilen yöntem, bir son durumu göz önüne alındığında, bir başlatma kümesi oluşturmak için bir tarihçe ağacının kullanıldığı sezgisel bir yöntemdir. Hedef yönelimi ile tercih oluşturmayı amaçlamaktadır, bu da yeni bir tercihin başlangıç kümesinin, örtük bir "yön" ile inşa etme olasılığı daha yüksek olan durumlarla sınırlandırması anlamına gelmektedir. Bu yöntem aynı zamanda, gereksiz döngülerin ortadan kaldırılması nedeniyle, tercih gecikme sezgisi ile karşılaştırıldığında daha geniş bir durumlar kümesini de dahil etmeyi amaçlamaktadır. Şekil 9.3 yaklaşımımızın örnek bölümdeki durumu nasıl ele aldığını gösterir. Problemin yapısı sağ odaya doğru bir yön çizdiğinden, bizim sezgimiz, s_{101} durumuna ulaşan tercihin başlangıç kümesine sağ odadaki (s_{109} gibi) durumları dahil etmez.

Algorithm 1 *HISTORY_TREE_HEURISTIC*

Require: a history H
Require: a terminal state s_t ▷ we assume the terminal state is known in advance
Ensure: initiation set I

1: $s_r \leftarrow$ FIND_ROOT(H, s_t)
2: $parent(s_r) \leftarrow null$, $\mathcal{V}_{s_r} \leftarrow 0$,
3: **for** each episode $e \in H$ ending with s_r **do**
4: $\quad i \leftarrow length(e) - 2$
5: $\quad$ **while** $i \geq 0$ **do** ▷ calculate state values backwards in history
6: $\quad\quad$ **if** $\mathcal{V}_{s_i}$ is undefined $\vee$ $(r_{i+1} + \gamma * \mathcal{V}_{s_{i+1}}) > \mathcal{V}_{s_i}$ **then**
7: $\quad\quad\quad \mathcal{V}_{s_i} \leftarrow r_{i+1} + (\gamma * \mathcal{V}_{s_{i+1}})$
8: $\quad\quad\quad parent(s_i) \leftarrow s_{i+1}$
9: $\quad\quad$ **end if** ▷ get rid of loops
10: $\quad\quad i \leftarrow i - 1$
11: $\quad$ **end while**
12: **end for**
13: $V \leftarrow \{s_r\}, E \leftarrow \emptyset$
14: **for** each state $s \neq s_r$ **do**
15: $\quad V \leftarrow V \cup \{s\}$ ▷ vertices of the tree
16: $\quad E \leftarrow E \cup (s, parent(s))$ ▷ edges of the tree
17: **end for**
18: $I \leftarrow$ traverse sub-tree of (V, E) rooted by s_t down to depth o_d (excluding s_t) and collect each visited state
19: **return** I

Algoritma 9.1 Tarihçe Ağacı Sezgiseli

Bir son durum s_t verildiğinde, Algorithma 1 öncelikle Algorithm 2'yi kullanılarak uygun bir kök durumu s_r bulur. Algoritma 2 yeni bir tercihin yönlendirilebilmesi için olası bir hedef durumu bulur. Bir problemin birden fazla hedef durumu olabileceği için, algoritma etmenin bölüm geçmişini incelemekte, bir ödül zirvesi ile biten bölümleri kontrol etmekte ve s_t durumu kullanılarak en çok gidilen son durumu döndürmektedir. Buradaki amaç, yeni tercih için doğru bir yön belirlemektir.

Kök düğüm olarak s_t durumu seçildiğinde, algoritma s_r ile biten tüm alt öyküleri (veya bölümleri) geçerek her ziyaret edilen düğümden s_r'a giden en kısa yollara sahip bir ağaç oluşturur. Her durum, kök s_r'ya kadar olan yolda alınan indirimli ödüllere göre Vdeğerine sahiptir. Bu değerleri kullanarak, s_r ile biten her bölüm, en son ziyaret edilen durumdan birinciye gezilerek, her durum için üst durumları, hedef durum s_r'a ulaşmak için en iyi durumun bir temsilcisi olarak ayarlar (satır 3-12). Sonuç olarak, bölüm sırasında gözlenen tüm durumlar kümesi Vve her bir

durumdan üst duruma geçişleri temsil eden geçiş kümesi Eile bir tarihçe ağacı oluşturulur (satırlar 13-17).

Algorithm 2 $FIND\_ROOT$

Require: a history H
Require: a terminal state s_t
Ensure: a root state s_r
1: $E_{s_i} \leftarrow 0, \forall s_i \in H$ ▷ number of reward peaked episodes ending with s_i containing s_t
2: **for** each episode $e \in H$ **do**
3: Let s_n be the last state of e
4: Let r_n be the last reward achieved in e
5: **if** $s_t \in e$ and $r_j < r_n$ where $1 < j < n$ **then**
6: $E_{s_n} \leftarrow E_{s_n} + 1$
7: **end if**
8: **end for**
9: $s_t \leftarrow \arg\max E_{s_i}$
10: **return** s_t

Algoritma 9.2 Kök Bulma Algoritması

Şekil 9.3, *2 oda 2 kapı* örneğinde algoritma tarafından oluşturulan bir ağacın bir bölümünü gösterir. Algoritma 1, s_t hariç her ziyaret edilen durumu eklemek için son durum s_t'den başlayarak (örnekte s_{101}) bir gezim gerçekleştirir ve bu gezimde gözlenen her durumu başlatma kümesine dahil eder (satır 18). Son durumu s_t 0 derinliğinde olmak üzere, gezimin gideceği en fazla derinlik, *tercih derinliği* (o_d)olarak adlandırılan bir parametre olarak sağlanır.

Geri kalan tercih üretme mekanizmaları, tercih gecikmeli hevesli stratejiyle aynıdır. Başlatma kümesindeki durumların sonlandırma olasılığı, 0 olarak belirlenirken son durumlar için 1 olarak atanır. Son duruma ulaşma politikası ER aracılığıyla üretilir.

9.4 Deneyler

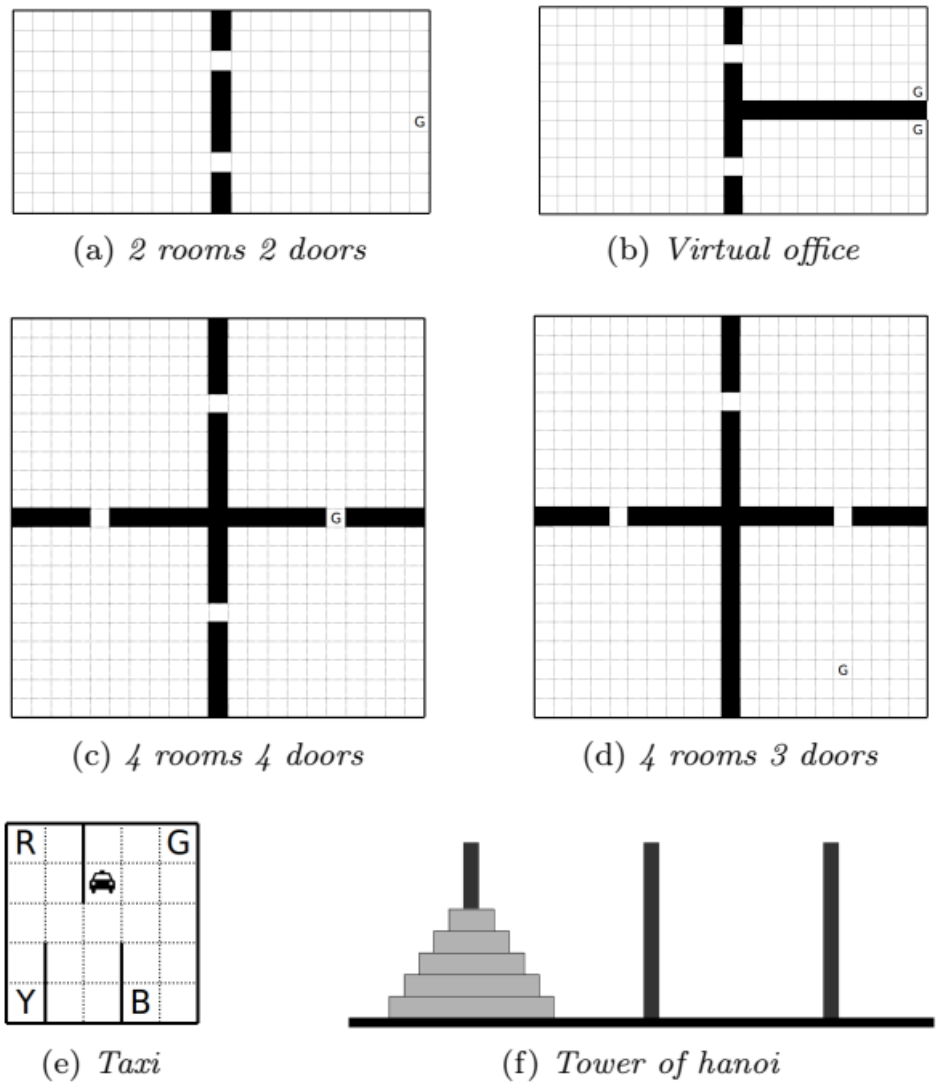
Deneyler, literatürdeki referansları ile birlikte Tablo 9.1'de listelenen altı farklı problem kullanarak yapılmıştır. Problemlerin taslakları Şekil 9.4'te verilmiştir. *Taxi* ve *Tower of Hanoi* dışındaki problemlerde, etmen, *kuzey, doğu, güney ve batı* şeklinde bir adımda dört pusula yönüne gidebilir. 0.9'lık olasılıkla hedeflenen yönde hareket eder ve 0.1'lık olasılıkla rastgele bir

doğrultuda komşu bir hücreye doğru yürür. Etmen, G hedef durumuna ulaştığında 1.0'lık bir ödül alırken, diğer herhangi diğer bir adım 0'lık bir ödül verir. Bu problemlerde bir bölüm, batıdaki odaların herhangi bir hücresinden rastgele başlar.

Tablo 9.1. Problem boyutları

Problem	**|S|**	**|A|**	**Ref.**
2 rooms 2 doors	202	4	[16]
Virtual office	212	4	[6]
4 rooms 4 doors	404	4	[13]
4 rooms 3 doors	403	4	[13]
Taxi	500	6	[5]
Tower of hanoi	243	6	[8]

Izgara problemlerinin duvar yapısı deney kümemizde amacını belirlemektedir. *2 rooms 2 doors* problemi (Şekil 9.4.a), iki darboğaz durumu ile klasik bir problemdir; bu, hem alt hedef tanımlama problemleri hem de tercih üretim mekanizmaları için zorlayıcı olabilir. Burada kullanılan *2 rooms 2 doors* problemi, hedef durumun konumu ile, daha uzun çözüm yolları elde etmek için değiştirmemizden dolayı orijinal versiyonundan farklıdır (Menache, Mannor, ve Shimkin 2002)**.** *Virtual Office* (Şekil 9.4.b) aslında her biri bir hedef durumu içeren iki ayrı oda ile kısmen gözlemlenebilir bir MDP ölçüt problemidir (Dung, Komeda, ve Takagi 2009). Ancak, bizim deney ortamımızda, problemin boyutunu arttırdık ve metotların, eşit derecede çekici olan birden fazla hedef durum olduğunda nasıl davrandığını test etmek için kısmi gözlem yapısını çıkardık. *4 rooms 4 doors* (Şekil 9.4.c), farklı kapılardan çoklu çözüm yollarıyla, alt hedef tanımlama yöntemlerinin etkinliğini göstermek için yaygın olarak kullanılan bir diğer ölçüt problemidir (A. McGovern ve Barto 2001). *4 rooms 3 doors* (Şekil 9.4.d), güney kapısının kaldırıldığı ve hedef durumunun değiştirildiği *4 rooms 4 doors* probleminin değiştirilmiş bir versiyonudur. Böylece, güneybatı odası artık etmen için bir tuzaktan başka bir şey değildir. Yararlı bir makro eylem oluşturma politikası, o odaya giren bir yol içermemelidir.



(a) *2 rooms 2 doors* (b) *Virtual office*

(c) *4 rooms 4 doors* (d) *4 rooms 3 doors*

(e) *Taxi* (f) *Tower of hanoi*

Şekil 9.4. Problem çizimleri

Taxi problemi (Şekil 9.4.e) deney kümemizde yer alan iyi bilinen bir referans problemdir. Bu problemde, etmen belirlenen bir yerden bir yolcuyu istediği yere transfer etmeye çalışır. Diğer ızgara problemlerinde olduğu gibi, etmen dört pusula yönüne gidebilir ve iki ek yolcuyla ilgili eylemi yapabilir: *yolcuyu al* ve *yolcuyu bırak*. Herhangi bir zamanda, yolcu R, G, B, Y ile işaretlenmiş hücrelerden birinde veya takside yer alabilir. *Yolcuyu al* ve *yolcuyu bırak* eylemleri yalnızca taksi etmeni belirlenen hücrelerden birinde olduğunda etkilidir. Başlangıçta, taksi etmeni, ızgara hücrelerinden herhangi birinde yer alır ve yolcu, belirlenmiş hücre konumlarından birindedir. Herhangi bir gezinme eylemi 0.8 olasılık ile başarılı olur, aksi takdirde etmen, her biri

0.1 olasılıkla hareket yönüne göre sola veya sağa yönlendirir. Etmen, yolcunun yanlış yere bırakılmasında -10 ceza, doğru yere bırakılmasında ise 20 ödül alır. Diğer tüm geçişler, -1'lik bir varsayılan ödül değeri verir.

Tower of Hanoi (Şekil 9.4.f) problemi, problem kümemizde klasik bulmaca oyunlarının temsilcisidir. Klasik tanımlarında, problem, farklı boyutlarda m çubuk ve n diskten oluşur. Problem, artan boyut sırasına göre çubuklardan birine yerleştirilen tüm disklerle başlar. Amaç, diskleri birer birer başka çubuğa hareket ettirmek, aynı sırayla bu çubukta istiflemektir. Bir diskin hareketi, yalnızca çubuktaki en üstteki diskse ve boş bir çubuğa veya daha büyük boyutlu bir diskin üzerine yerleştirilecekse geçerlidir. Deneylerimizde, m = 3 ve n = 5 olarak ayarladık. Bu ayar için 6 disk hareketi eylemi vardır ve etmen eylem adımı başına yalnızca bir disk taşıyabilir. Eylemler deterministiktir, geçersiz bir eylem ise çevrede hiçbir değişikliğe neden olmaz. Etmen, hedef duruma ulaşıldığında 1.0'lık bir ödülle ödüllendirilir ve geçerli olmayan bir eylemi uygularsa -0.1 ile cezalandırılır. Başka herhangi bir eylem, -0.01''lik bir varsayılan ceza verir.

9.4.1 Önceden Tanımlanmış Alt Hedeflerle Deneyler

Testlerimizdeki iki deney düzeneğinin ilki, öğrenme başlamadan önce yararlı son durumlar (alt hedefler) kümesini sağlayarak adil bir karşılaştırma yapmayı amaçlamaktadır. Deneyler, tarihçe ağacı sezgiseli performansını tercih gecikmesi yöntemi uygulayan hevesli tercih üretme stratejisi ile karşılaştırılmak için tasarlanmıştır. Problem dinamikleri incelenerek her problem için anlamlı alt hedefler önceden tanımlanmıştır.

9.4.1.1 Ayarlar

Oda problemleri için tercihler için bariz yararlı son durumlar kapılardır. *Taxi* problemi için, bunlar, gezinme darboğaz durumlarıdır ve taksinin yolcuyu aldıktan sonra vardığı durumlardır. *Tower of Hanoi* probleminde en büyük diskin tek başına hedef çubuğa yerleştirildiği veya en büyük ve ikinci en büyük disklerin birbirinin üzerine hedef çubukta yerleştirildiği durumlara karşılık gelen dört durum alt-hedef olarak sağlanmıştır. Etmen, yeterli sayıda bölüm tecrübe ettiğinde bir son durumuna (son durum dahil olmak üzere) ulaşmak için bir tercih oluşturur. Bu sayı, önceden tanımlanmış alt hedef deneylerinde yapılan tüm denemeler için 10 olarak ayarlanmış olup bu sayıya *bölüm eşiği* (t_e) denir.

Tablo 9.2. Öğrenme parametreleri

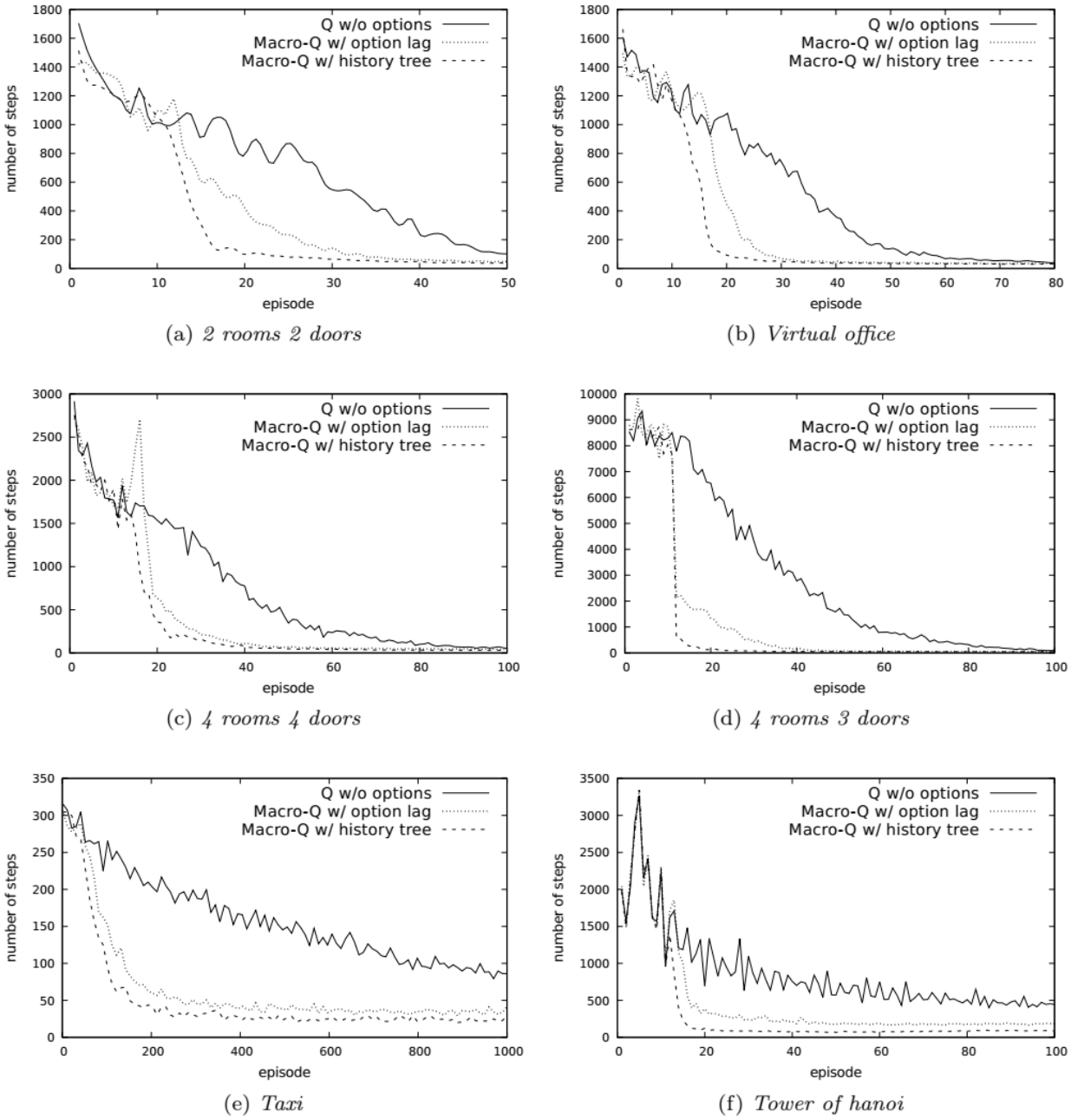
Algorithm	Parameter	Value
Q/Macro-Q Learning	α	0.05
	γ	0.9
	ϵ	0.1
Experience Replay	α	0.125
	γ	0.9
	r_{β}	+1000.0
	$r_{\tilde{\mathcal{I}}}$	−100.0
	$r_{default}$	−1.0

Son durumlar belirlendikten ve başlatma kümelerinin oluşumu tamamlandıktan sonra, ER, etmen tarafından, terminal durumuna ulaşmak için başlatma kümesindeki durumlar üzerinden politika üretmek için kullanılır. Bir geçiş, son duruma ulaştığında 1000'lik bir ödül kazanır, başlangıç kümesinden ayrıldığında -100.0 ile cezalandırılır ve başka geçişler için -1'lik bir ceza alır. ER'nin öğrenme parametreleri $\alpha = 0.125$ ve $\gamma = 0.9$'dur. Bir tercih son duruma ulaştığında 1.0'lık bir olasılıkla sonlandırılırken, başlangıç kümesindeki herhangi bir durumda sonlandırılmaz. ER parametreleri Tablo 9.2'de verilmiştir. ER işlemi, Q fonksiyonunun hızlı bir şekilde yakınlaşmasını sağlamak için 10 kez tekrarlanır.

Tercih gecikmesi ve tercih derinliği parametreleri, sonuçta ortaya çıkan tercihlerin, her iki yaklaşım için ortalamada yaklaşık olarak en fazla 3 durum farkı olacak şekilde aynı sayıya sahip başlatma kümelerine sahip olacak şekilde ayarlanır. Macro-Q Learning, Tablo 9.2'de verilen öğrenme parametreleriyle birlikte kullanılır. Tüm yöntemler, seçenekler olmadan aynı öğrenme parametrelerini kullanarak normal Q-Learning ile karşılaştırılmıştır. Test sonuçlarının 200 deney ortalaması ile alınmıştır.

9.4.1.2 Sonuçlar ve Yorumlar

Şekil 9.5'te verilen sonuçlardan açıkça anlaşılmaktadır ki, adil koşullar altında, başlangıç kümesi için tarihçe ağacı sezgisi kullanılarak oluşturulan tercihlerle Macro-Q Learning, tercih gecikmesi sezgisi kullanılarak oluşturulan tercihlerle olan haline göre hedef durumuna ulaşmak için gerekli adım sayısı bakımından ortalamada daha iyi performans göstermektedir.. Bizim deney kümemizdeki her problem için, tarihçe ağacı özellikle öğrenme sürecinin ilk aşamalarında Macro-Q Learning algoritmasına bir avantaj sağlar.



Şekil 9.5. Ortalama hedefe ulaşma adım sayısı

Oluşturulan tercihlerin kalitesini daha iyi analiz etmek için, iki ölçü ortaya atılmıştır: Bunlardan biri, başlangıç kümelerinin kesişim kümesi büyüklüğünün, birleşim kümesi büyüklüğüne oranı olarak verilen, *tercih çakışması*'dır. Bu ölçünün daha yüksek değerleri tercihler arasında potansiyel olarak gereksiz tekrarların bir göstergesidir. Diğer ölçü, *tercih önceliği* olup, başlatma kümesindeki durumlarda tercihi kullanmanı ne kadar cazip olduğunu gösterir.

Tablo 9.3. Tercih kalitesi ve zaman kullanımı

Problem	Option overlap (%)		Average time overhead (msec)	
	option lag	history tree	option lag	history tree
2 rooms 2 doors	13.99	**0.38**	21.82	**20.69**
Virtual office	9.43	**1.15**	11.83	**10.39**
4 rooms 4 doors	9.62	**0.03**	**31.58**	35.45
4 rooms 3 doors	12.85	**0.00**	**103.79**	105.31
Taxi	52.59	**51.24**	25.53	**23.31**
Tower of hanoi	30.61	**0.00**	142.11	**77.96**

Tercih önceliği aşağıdaki gibi tanımlanmıştır;

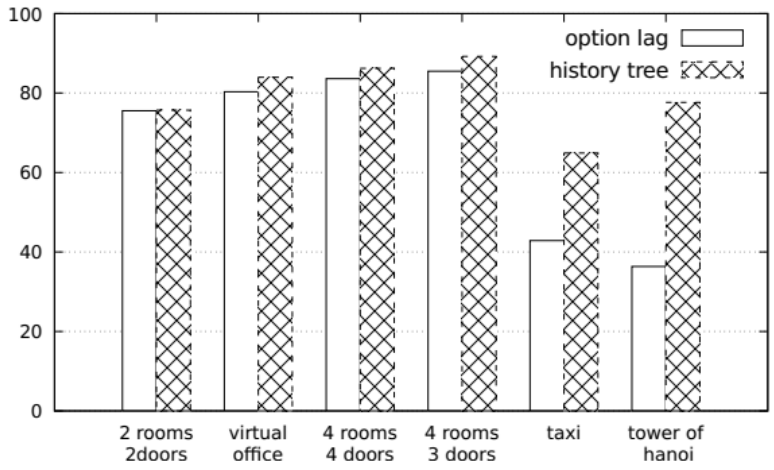
$$P(o) = \frac{\sum_{s \in I_o} p(s,o)}{|I_o|}$$

$p(s,o)$ ise şu şekilde tanımlanmıştır;

$$p(s,o) = \begin{cases} 1, & \max_{o'} Q(s,o') = Q(s,o), \\ 0, & \text{otherwise} \end{cases}$$

Burada o eldeki tercih, s I_o'daki bir durumu, o' s'den alınabilecek herhangi bir tercihi ve I_o ise o tercihinin başlatma kümesini temsil eder.

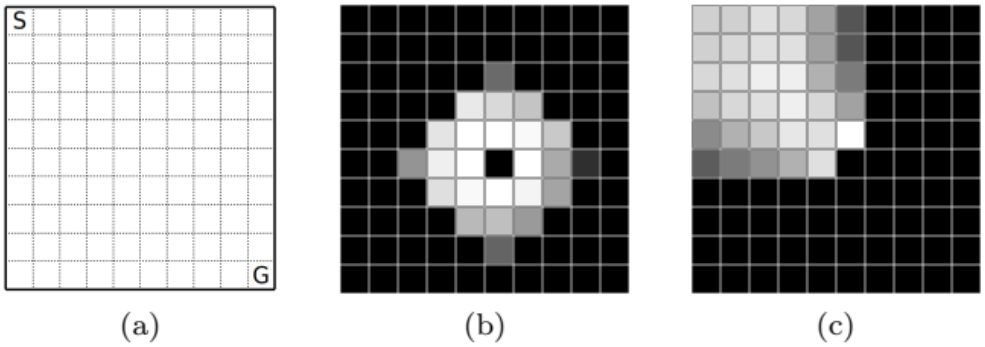
Tablo 9.3, iki sezgisel yöntem tarafından oluşturulan tercihlerin tercih çakışma değerlerini ve ortalama CPU zamanını listeler. Tercih çakışması açısından, tarihçe ağacı sezgiseli problemlerin çoğunda önemli bir iyileşme sağlar. Bu, tarihçe ağacı sezgiselinin, bir tercih için, gereksiz tekrarların azaltılacağı şekilde bir başlatma kümesi oluşturabileceği anlamına gelir. Dahası, bu yöntem, az çok aynı CPU zamanını kullanarak bunu başarır.



Şekil 9.6. Tercih önceliği değerleri

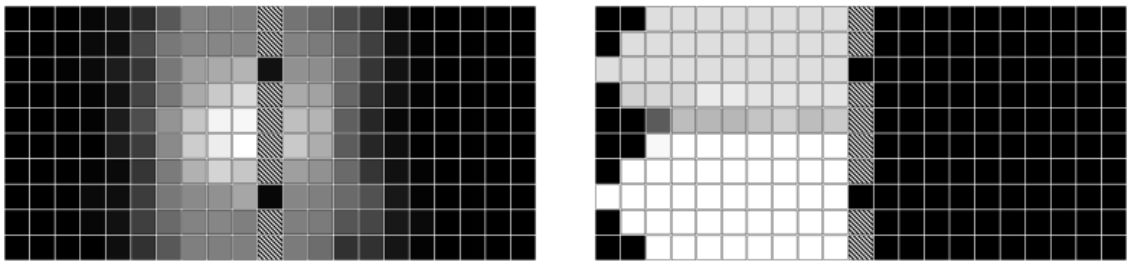
Tarihçe ağacı yöntemi ile oluşturulan tercihler, ortalamada, Şekil 9.6'da görüldüğü gibi, tercih gecikmesi yöntemi kullanılarak oluşturulanlardan daha fazla tercih edilir. Göreceli fark, problem boyutunun ve son durumlarının sayısının artmasıyla daha önemli hale geliyor; ve bu, tarihçe ağacı tabanlı başlatma kümelerinin, hedefe yönelik tercihlerin oluşturulmasını teşvik ederek, makrolar aracılığıyla eylem seçim sürecini yönlendirdiğini destekliyor.

Metodumuzun bu gelişmeyi nasıl gerçekleştirdiğine dair daha derin bir anlayışa sahip olmak için, aşağıdaki şekiller, başlatma kümesi içinde bulunan her bir durumun ortaya çıkış frekanslarını, ortalama alarak görselleştirir. Daha parlak renk, durumun daha çok seçildiği anlamına gelirken, karanlık renk ise daha az seçildiği anlamına gelir.



Şekil 9.7. Örnek problem

Ama önce, diğer oda problemleri ile aynı etki, geçiş ve ödül yapısına sahip ancak herhangi bir darboğaz duruma sahip olmayan basit bir 10x10 ızgara problemini (Şekil 9.7.a) incelemek yararlı olabilir. Bu, bir son durumunun bu problemde rastgele seçilmiş olması durumunda, problem doğal olarak bölümlenemeyeceği için öğrenme performansını geliştirmenin mümkün olmadığı anlamına gelir. Bununla birlikte, yine de, hedef durumun konumu, rastgele seçilen bir son duruma göre bir tercih yönelimi verebilir. Yapay son durum (ya da alt-hedef) olarak, ızgara probleminin ortasında bir yer seçilir ve başlatma kümeleri oluşturmak için tercih gecikmesi ve tarihçe ağacı yöntemleri çağrılır ve sonra tercihler oluşturulur. Her bir durum için başlatma kümesinde bulunma frekansları Şekil 9.7.b ve 9.7.c'de gösterilmiştir. Her iki yöntem de başlangıç kümelerinde yaklaşık 17 duruma sahip olmuştur. Rakamlarla gösterildiği gibi, tercih gecikmesi yaklaşımı, verilen son duruma çevre durumlardan ulaşmayı amaçlayan bir tercih oluşturur. Öte yandan, tarihçe ağacı yaklaşımı sadece başlangıç durumu ve son durumu arasındaki durumları içerir. Hedef durum güney-doğu köşesinde yer aldığından, başlangıç kümesi öğeleri olarak seçilen durumlar, hedef duruma doğru bir “yön”e sahip olacak şekilde seçeneği açıkça şekillendirmektedir. Elenen durumlar, son durumu ziyaret etmeye gerek kalmadan hedef durumuna ulaşman daha kısa yollara sahiptir.

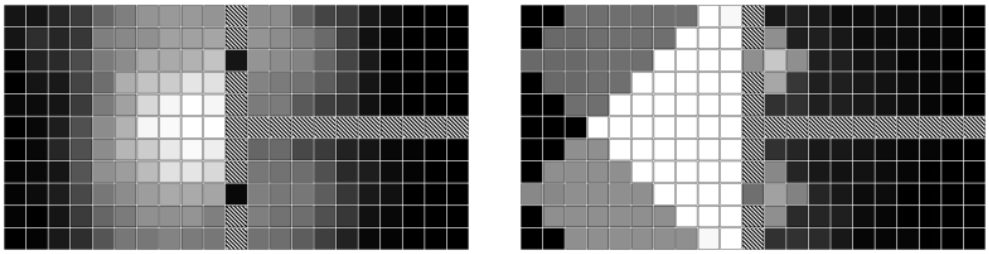


(a) option lag (b) history tree

Şekil 9.8. 2 rooms 2 doors problemindeki başlatma kümeleri

2 rooms 2 doors problemi, etmenin tek hedef durumuna ulaşması için iki alternatif yol sunar. Etmen, kapı girişlerine giden döngüler içinde dönebildiğinden, tercih gecikmesi mekanizması, kapının sağ tarafındaki durumların birçoğunu seçme eğilimindedir ve bunları başlatma kümelerine ekler. Bununla birlikte, tarihçe ağacı sezgiseli tarafından oluşturulan başlatma kümeleri sadece batıdaki odanın durumlarından ibarettir (Şekil 9.8.b). Ayrıca, batı odasındaki bir durum, kapı girişlerine giden tercihlerin sadece birinin başlangıç kümesine eklenirken, bu tercih gecikmesi sezgiseline kıyasla çok daha düşük bir seçenek çakışmasına yol açmaktadır (Tablo 9.3). Bunun nedeni, batıdaki odadaki bir durumun, oluşturulan ağaç içindeki bir kapı durumunun

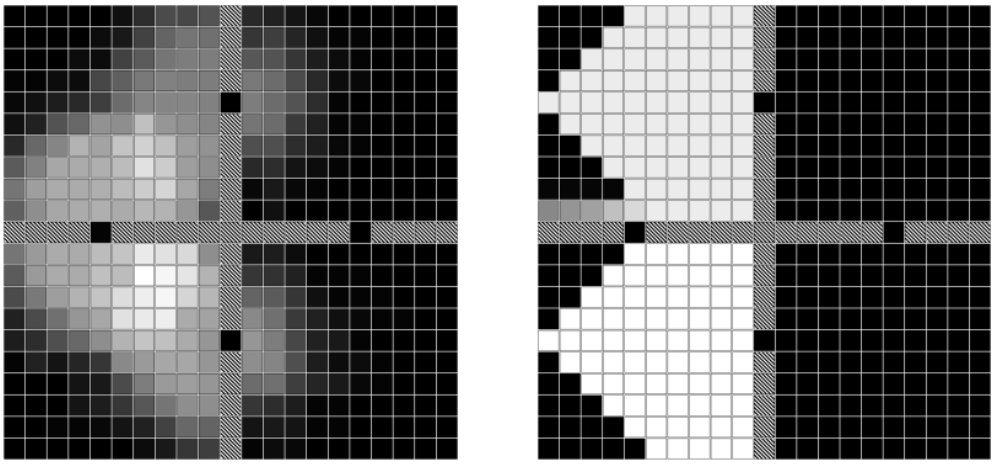
altındaki alt ağaçlardan sadece birine ait olmasıdır. Bu, tarihçe ağacı yönteminin yararlı bir tercih üzerinde yapılacak gereksiz keşifleri atlamasına ve öğrenme hızındaki performansını geliştirmesine izin verir (Şekil 9.5.a).



(a) option lag (b) history tree

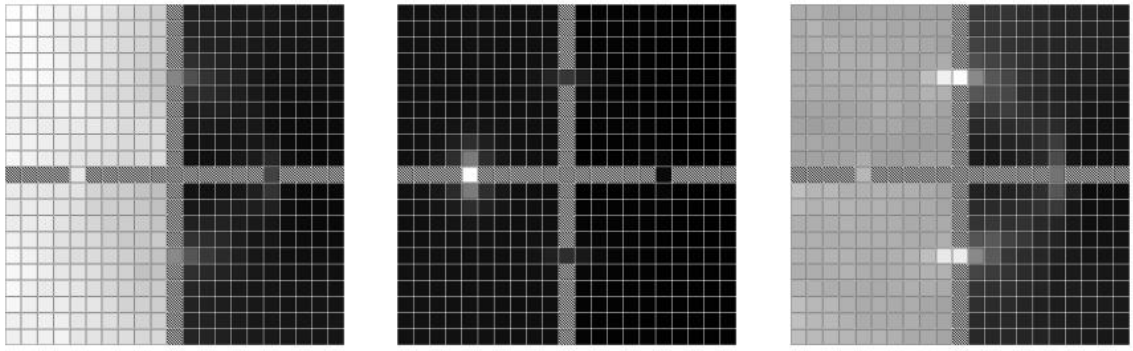
Şekil 9.9. Virtual Office problemindeki başlatma kümeleri

Virtual Office probleminde olduğu gibi, çoklu hedef durumları ortaya çıktığında, tarihçe ağacı sezgisi hala öğrenme hızı açısından tercih gecikmesinden daha iyi performans gösterir (Şekil 9.5.b). Darboğaz durumu içeren bir bölüm genellikle ilgili odada hedef durumla bittiğinden, tarihçe ağacı sezgiseli, hedef durumu kökü olarak seçer ve bu da o odaya girmek için bir tercih oluşturur. Bu seçim mekanizması, yöntemimizin bir etmeni hedef durumu olan bir odadan çıkaracak bir tercih oluşturmasını engeller. Başlangıç kümelerindeki durumların frekansları Şekil 9.9'da çizilmiştir, bu da tarihçe ağacı sezgiselinin, doğudaki amaca yönelik hedefe vekalet etmede daha hevesli tercihleri elde etme eğiliminde olduğunu göstermektedir.



(a) option lag (b) history tree

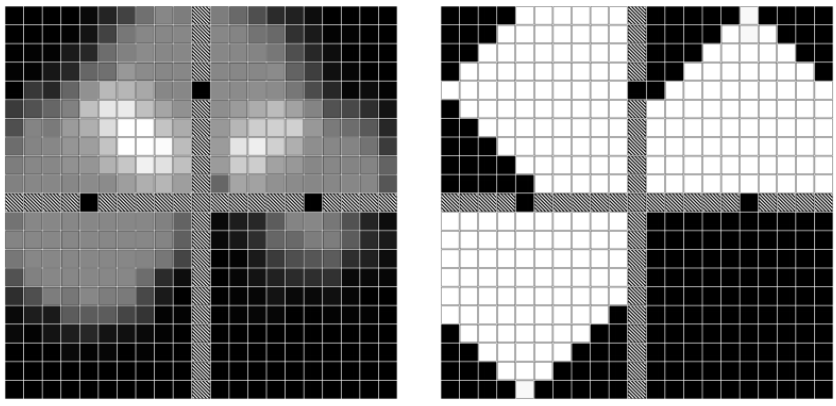
Şekil 9.10. 4 rooms 4 doors problemindeki başlatma kümeleri



(a) Q learning without options (b) option lag (c) history tree

Şekil 9.11. 4 rooms 4 doors problemindeki durum ziyaret edilme frekansları

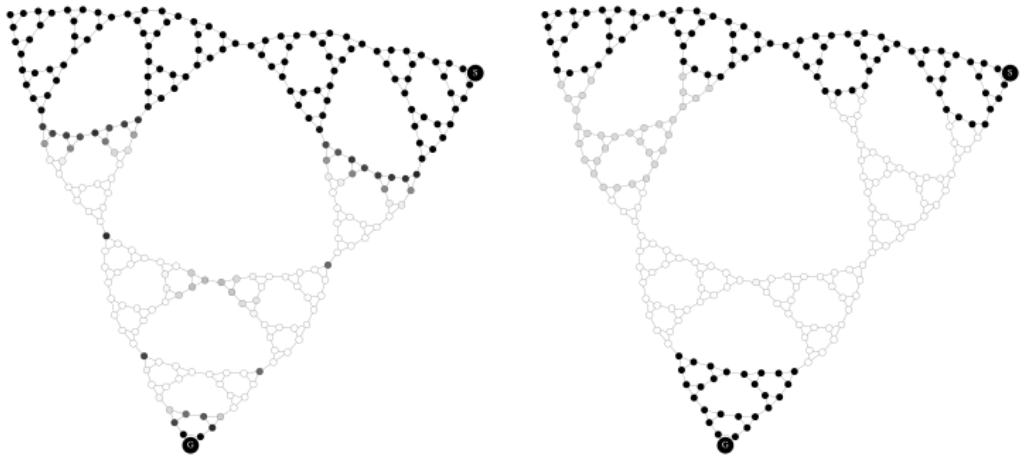
4 rooms 4 doors probleminde, sol odalar arasındaki kapı açık bir darboğaz olduğu için otomatik tanımlama yöntemleriyle bir alt hedef olarak tanımlanabilir. Bununla birlikte, etmeni doğru odalara yönlendiren alt hedefler, öğrenme sürecinde daha faydalıdır. Şekil 9.5.c, sol odalar arasında bulunanla birlikte üç kapı önceden sağlandığında, tercih gecikmesi sezgiselinin oluşturduğu tercihler, etmenin, ilgili alt hedefin gerçekte gereksiz olduğunu öğrenmek için zaman harcadığını göstermektedir. Ancak, bu problemi, tarihçe ağacı sezgiselinin oluşturduğu tercihler ile görmüyoruz. Tarihçe ağacı, kapının etrafındaki durumları kullanarak bir başlatma kümesi oluşturma eğiliminde olan tercih gecikmesi sezgiselinden farklı olarak, yalnızca birkaç durumu kullanan bir başlatma kümesi oluşturur (Şekil 10). Şekil 11 bu bulgunun desteklendiğini göstermektedir. Bu durum, etmenin tercih gecikmesi mekanizmasıyla kapı aralığını daha sık ziyaret ettiğini, ancak tarihçe ağacı sezgiseli ile hedef duruma daha erken ulaştığını göstermektedir.



(a) option lag (b) history tree

Şekil 9.12. 4 rooms 3 doors problemindeki başlatma kümeleri

4 rooms 4 doors probleminin değiştirilmiş versiyonu için, güney odalar arasındaki kapı olmadan (Şekil 9.4.d), sol odaları bağlayan kapıya ulaşmak için bir tercih ancak güneybatıdaki odadan başlatılıyorsa yararlıdır. Başlangıç kümesine dahil edilmek için, tarihçe ağacı yöntemi yalnızca tercihi oda çıkışına yönlendiren durumları seçer. Seçim stratejileri arasındaki fark, Şekil 9.12'de açıkça görülebilir.



(a) option lag (b) history tree

Şekil 9.13. Tower of Hanoi problemindeki başlatma kümeleri

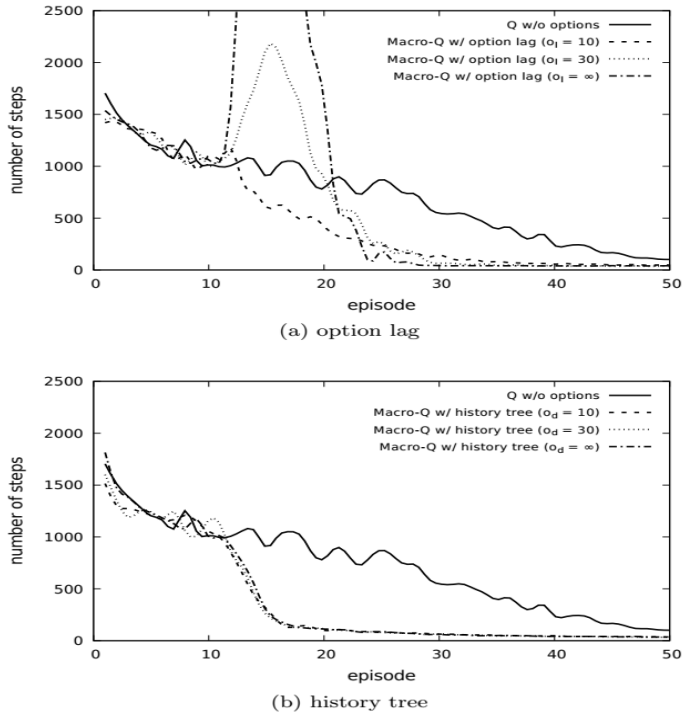
Tower of Hanoi gibi ızgara dışı bir problemde, tarihçe ağaç sezgiseli tarafından oluşturulan tercihler, diğer sezgisele göre, etmenin, daha hızlı bir şekilde daha iyi bir politikaya yaklaşmasına olanak tanır (Şekil 9.5.f) . Şekil 13, açık bir şekilde tarihçe ağacı sezgisinin, hedef duruma giden yolun alt hedefinden önceki durumları içerdiğini açıkça göstermektedir. Bu fark, öğrenmede daha iyi bir performansa yol açar.

Son olarak, *Taxi* probleminde, yolcunun hedefi doğal olarak hedef duruma doğru bir yönlendirme önerir. Tarihçe ağacı sezgisi, hedefe yönelik tercihler oluşturmaya yardımcı olduğundan, performansı, Şekil 9.5'te görüldüğü gibi, tercih gecikmesi performansından çok daha iyidir. Öte yandan, *Taxi* problemindeki alt hedefler ardışıktır, yani ilk durumdan başlayarak, hedefe giden yolda birden fazla alt-hedefler vardır. Bu, her iki yöntem için diğer sorunlara kıyasla daha yüksek bir tercih çakışmasına neden olur. Bununla birlikte, tarihçe ağacı sezgisi hala örtüşen tercihler açısından tercih gecikmesi sezgiselini geride bırakmaktadır (Tablo 9.3).

Genel olarak, tarihçe ağacı sezgisi yoluyla üretilen hedefe yönelik tercihler, tercih gecikmesi mekanizması aracılığıyla yaratılanlara göre daha fazla tercih edilir gibi görünmektedir. Problem kümemizdeki tüm problemlerde daha iyi performans göstermektedir (Şekil 9.5). Bu, tarihçe ağacı sezgisinin, geleneksel hevesli başlatma kümesi yaklaşımından daha iyi bir alternatif olabileceğini düşündürmektedir.

9.4.1.3 Parametre Analizi

Daha fazla analiz için, tercih üretme için kullanılacak iki alt hedefin bulunduğu *2 rooms 2 doors* probleminde farklı parametre değerleriyle her iki sezgiseli denedik. Deneyler, belirlenen başlangıç parametrelerinin farklı değerleri; yani, *tercih gecikmesi* için $o_l$ve *tarihçe ağacı* için o_d; başlangıç kümesi boyutlarında adil bir kontrol yapılması adına değiştirilerek yapılmıştır. o_l, son durumdan önce kontrol edilecek adım sayısını sınırlandırırken, o_d, son durumun altındaki ağaçtaki arama derinliğini belirler. Her iki sezgisel için de, 10, 30 ve sınırsız (pratik olarak, alabilecekleri maksimum değer), olmak üzere 3 parametre değeri ayrı ayrı uygulanmıştır.



Şekil 9.14. 2 oda 2 kapı probleminde farklı parametrelerle oluşturulmuş tercihlerle ortaya çıkan adım sayısı grafikleri

Tablo 9.4, *tercih gecikmesi* değerini artırmanın da ortalama başlatma kümesi boyutunu artırdığını gösterir. Sınırsız *tercih gecikmesi* ile, $|I|$neredeyse tüm durum uzayına ulaşır. Ancak, etmen hedef durumla aynı odadayken, kapılara geri dönmesi gerekmediğinden, bu durum, odanın sağ odadaki tercihlerin kullanımını ortadan kaldırmak için fazladan zaman harcamasına neden olur. Bu gözlem, tercihler etmene sunulduğunda, Şekil 14'te görülen hedefe ulaşmak için atılan adım sayısındaki artışla desteklenmektedir.

Tablo 9.4. 2 oda 2 kapı problemindeki tercih kalitesi ve ortalama başlatma kümesi büyüklükleri

o_l or o_d	**Option overlap (%)**		**Initiation set size** ($\|I\|$)	
	option lag	**history tree**	**option lag**	**history tree**
10	13.99	**0.38**	45.02	43.35
30	22.84	**0.58**	97.72	49.53
∞	48.60	**0.53**	171.68	50.17

Öte yandan, *tercih derinliği*'nin artırılması etmenin başlatma kümelerine yalnızca sol odayı eklemesini sağlar. Bizim sezgisel tercih çakışmasını önlediğinden (Tablo 9.4), soldaki oda iki tercih arasında paylaştırılır, bu da başlangıç kümelerini sınırsız *tercih derinliği* ile yaklaşık 50 durum barındıracak hale getirir. Başlangıç kümelerine sağ oda durumlarını dahil etmekten kaçınan yaklaşımımız, farklı *tercih derinliği* değerleri ile benzer bir performans gösterir.

9.4.2 Otomatik Alt Hedef Tanımlama Yöntemleri ile Deneyler

Tablo 9.5. Otomatik Alt Hedef Keşif Yöntemlerinde Kullanılan Parametreler

Problem	**Parameters used**					
	Method	p	q	t_c	t_q	t_e
2 rooms 2 doors	LoBet	0.7	0.07	-	-	-
	LCut	0.2	0.01	0.05	-	-
	SegQCut	-	-	-	1000	10
Virtual office	LoBet	0.8	0.08	-	-	-
	LCut	0.1	0.01	0.05	-	-
	SegQCut	-	-	-	1000	15
4 rooms 3 doors	LoBet	0.5	0.05	-	-	-
	LCut	0.2	0.01	0.015	-	-
	SegQCut	-	-	-	1000	10
4 rooms 4 doors	LoBet	0.5	0.05	-	-	-
	LCut	0.2	0.002	0.05	-	-
	SegQCut	-	-	-	2000	10
Taxi	LoBet	0.3	0.03	-	-	-
	LCut	0.04	0.002	0.05	-	-
	SegQCut	-	-	-	1000	200
Tower of hanoi	LoBet	0.9	0.09	-	-	-
	LCut	0.25	0.025	0.05	-	-
	SegQCut	-	-	-	200	10

Deneylerin ikinci kısmı, yerinde yapılan testler olarak yorumlanabilir. Bu nedenle, şimdi, yararlı alt hedefleri etkin olarak buldukları bilinen otomatik algoritmalarla birlikte, başlangıç kümesi oluşturma yöntemimizin etkinliğini araştırmaya çalışacağız. Önceden tanımlanmış alt hedeflerin verildiği durumdan farklı olarak, şimdi otomatik keşif yöntemleriyle bulunacak olan alt hedeflerin sayısı ve kalitesi üzerinde herhangi bir kontrol yoktur. Böylece, tercih oluşturma detayları açısından adil bir karşılaştırma yapmak artık kolay değil. Bu nedenle, bu deney durumu için, klasik öğrenme performans kriterine ek olarak (yani bölüm başına ortalama adımlar), toplam çalışma süresi ve ek bellek kullanımı gibi daha genel ölçümlerini karşılaştırarak daha geniş bir bakış açısı sağlamaktayız.

9.4.2.1 Ayarlar

Bu deneylerde, önceki ayarlardan farklı olarak, alt hedefler, yani Yerel Aradalık (LoBet), Yerel Kesimler (LCut) ve Parçalı Q-Kesim (SegQCut) gibi iyi bilinen bazı yöntemlerle, otomatik olarak keşfedilmektedir. Genel öğrenme parametreleri, önceden tanımlanmış alt hedefler durumundaki (Tablo 9.2) ile aynı olsa da, alt hedef bulma yöntemleri için kullanılan parametreler, Tablo 9.5 verilmiştir. Tablo 9.5 içindeki parametreler, bir dizi deneme-ve-yanılma çalışmasıyla belirlenmiştir, böylece her bir yöntem, öğrenmeyi etkili bir şekilde geliştirmek için yeterli sayıda yararlı alt-hedef bulmaktadır.*Tercih gecikmesi* ve *tarihçe ağacı* başlatma kümesi oluşturma yöntemleri için aynı ayarlar önceden tanımlanmış alt hedefler ile yapılan deneylerdekilerle aynı olacak şekilde kullanılır. Öte yandan, Parçalı Q-Kesim algoritmasının (bundan sonra parçalama olarak adlandıracağız) başlatma kümesi oluşturma sezgisi, başlatma küme boyutunun kontrol edilmesi için herhangi bir parametreye sahip değildir. Bu nedenle, önceden tanımlanmış alt hedefler durumundan farklı olarak, ilgili deneyler, "hemen hemen eşit başlatma küme büyüklüğü" durumuyla sınırlı değildir, sonuç olarak, ortaya çıkan tercihlerin, her bir sezgisel için başlangıç kümelerinde aynı miktarda durumları yoktur.

Parçalı Q-Kesim, etmenin problemi neredeyse kararlı (yani değişmeyen) bir durum geçiş çizgesi elde etmek için yeterince araştırdığını garanti etmek amacıyla bir bölüm eşiğine $t_e$ulaştığı zaman gerçekleştirilir. Parçalı Q-Kesim'de kullanılan ilk kaynak ve hedef durum çiftleri, her bir problem için elle verilir, böylece algoritma makul bir bölümleme yapabilir. Parçalı Q-Kesim tarafından parçalanan bölümlerini başlatma kümesine ekleyen sezgisel de diğer sezgisel yöntemler ile karşılaştırılmıştır. Her bir alt hedef keşif metodu, farklı başlatma kümesi oluşturma sezgisel teknikleriyle ayrı ayrı test edilir.

9.4.2.2 Sonuçlar ve Yorumlar

Tablo 9.6. Otomatik Alt Hedef Keşiflerinin Ortalama Hedef Duruma Ulaşma Adım Sayıları

Problem	Q	Macro-Q w/ LoBet		Macro-Q w/ LCut		Macro-Q w/ SegQCut		
		option lag	history tree	option lag	history tree	option lag	segmen-tation	history tree
2 rooms 2 doors	696.58	463.43	**317.16**	452.09	**318.68**	497.56	**311.96**	314.05
Virtual office	520.54	360.05	**266.89**	351.01	**294.41**	394.71	**222.43**	276.81
4 rooms 4 doors	751.44	537.97	**306.96**	425.71	**308.91**	472.81	**279.11**	281.98
4 rooms 3 doors	3043.17	1470.23	**614.71**	3140.24	**1090.57**	1411.21	943.55	**940.31**
Taxi	161.46	114.06	**78.02**	112.38	**92.72**	98.51	**81.86**	82.03
Tower of hanoi	856.82	831.59	**317.88**	941.60	**549.20**	966.24	729.09	**677.46**

Tablo 9.6'da görüldüğü gibi, tarihçe ağacı sezgisi tarafından oluşturulan tercihlerin öğrenme performansı, tüm alt hedef bulma yöntemlerinde ve tüm problemlerde tercih gecikmesi ile oluşturulan tercihlerden üstün performans gösterir. Bu sonuç, önceden tanımlanmış alt hedefler örneğine benzemektedir, çünkü tarihçe ağacı sezgiseli, yalnızca tercihin hedefe ulaşmada yararlı olduğu kanıtlanmış durumları içerir ve bu şekilde, etmeni gereksiz bir öğrenme süresinden kurtarır. Öte yandan, parçalama sezgiseli ile oluşturulan tercihlerin performansı, tarihçe ağacı sezgiseli ile oluşturulanlara çok yakındır. Parçalama sezgiseli kesme tarafından bölümlendirilen kaynak parçadaki tüm durumları içerdiğinden, oluşturulan tercihler daha büyük başlatma kümelerine sahip olma eğilimindedir (Tablo 9.7). Her durum için, tarihçe ağacı sezgisi, hedefe adım sayısı açısından parçalama sezgiselliği ile benzer bir performans gösterse de, bu performansı önemli ölçüde daha küçük başlatma kümeleriyle elde eder.

Tablo 9.7. Parçalı Q-Kesim Algoritması ile Elde Edilen Başlatma Küme Boyutları

Problem	option lag	segmentation	history tree
2 rooms 2 doors	25.61	97.51	43.25
Virtual office	46.99	155.28	116.79
4 rooms 4 doors	50.07	133.97	50.45
4 rooms 3 doors	71.05	99.66	78.19
Taxi	14.55	23.94	23.99
Tower of hanoi	17.23	25.02	28.23

Sezgisel yöntemlerin bellek tüketimi, alt hedef tanımlama ve tarihçe tutma için gereken ek bellek bakımından Tablo 9.8'de verilmiştir. Tarihçe ağacı sezgiseli, ek bir ağaç yapısını kullanmasına rağmen, bellek tüketimi tercih gecikmesinden daha iyidir, çünkü tarihçe ağacı sezgiselliğine sahip tercihler etmeni çok daha kısa bir hedefe yönlendirir ve kısa bölümlere neden olur.

Parçalama sezgiseli için gereken ek bellek kullanımı ise, bir başlatma kümesi oluşturmak için oluşturulan parçalar içindir. Benzer şekilde, öğrenme performansındaki artış genel bellek kullanımını azaltıyor gibi görünmektedir. Çoğunlukla bu avantajı paylaştıkları için, hem tarihçe ağacının hem de parçalama sezgisellerinin bellek tüketimleri birbirine çok yakındır.

Tablo 9.8. Alt Hedef Tespiti Yöntemlerinde Hafıza kullanımı

Problem	Macro-Q w/ LoBet		Macro-Q w/ LCut		Macro-Q w/ SegQCut		
	option lag	history tree	option lag	history tree	option lag	segmen-tation	history tree
2 rooms 2 doors	485.90	**352.66**	474.79	**355.60**	537.25	**373.45**	374.27
Virtual office	641.91	**490.75**	632.45	**538.71**	906.04	**547.21**	653.91
4 rooms 4 doors	1233.42	**725.03**	976.97	**728.04**	1127.46	706.20	**705.40**
4 rooms 3 doors	3482.72	**1552.94**	7188.94	**2658.60**	3493.59	2432.17	**2417.91**
Taxi	1997.84	**1447.97**	1918.37	**1642.15**	1926.61	**1717.06**	1720.00
Tower of hanoi	1484.26	**628.64**	1661.79	**1122.82**	1749.11	1319.82	**1242.94**

Tablo 9.9. Alt Hedef Tespiti Yöntemlerinde Çalışma Süreleri

Problem	Q	Macro-Q w/ LoBet		Macro-Q w/ LCut		Macro-Q w/ SegQCut		
		option lag	history tree	option lag	history tree	option lag	segmen-tation	history tree
2 rooms 2 doors	*83.62*	90.84	**62.96**	108.99	**81.59**	119.15	91.63	**89.61**
Virtual office	*74.64*	75.18	**56.50**	92.12	**77.86**	97.92	**62.56**	75.37
4 rooms 4 doors	*204.81*	216.06	**138.10**	225.47	**177.73**	234.08	**155.43**	170.34
4 rooms 3 doors	*740.84*	572.14	**274.49**	1206.60	**506.85**	700.33	548.07	**483.93**
Taxi	*51.71*	167.95	**121.80**	84.00	**63.62**	130.43	**127.30**	128.13
Tower of hanoi	*60.21*	221.61	**101.83**	288.11	**219.21**	206.04	**175.08**	242.53

Bölüm başına ortalama CPU zaman tüketimi sonuçları, Tablo 9.9'da verilmiştir. Alt hedef tanımlama metotları, genel öğrenme prosedürünü etkileyen çeşitli sonuçlar üretebildiğinden, metotlar arasında adil olabilmeleri için, bu çizelgenin toplam öğrenme süresini gösterdiğine dikkat etmek gerekir. Sonuçlar, tarihçe ağacına sahip tercihlerin, hem normal Q-Learning'e hem de tercih gecikmesi sezgisi ile öğrenmeye kıyasla CPU zamanını düşürdüğünü göstermektedir. Macro-Q Learning tarafından tarihçe ağacı tabanlı başlatma kümeleri oluşturularak kazanılan zamanın, gereken ek hesaplamadan daha fazla olduğu görülmektedir. Yine, öğrenme hızındaki benzer performanslar gibi, parçalama ve tarihçe ağacı sezgiselleri için zaman tüketimleri birbirine çok yakındır. CPU zamanı açısından, tarihçe ağacı sezgisi, tercih gecikmesinden daha iyidir ve parçalama sezgisiyle rekabet eder.

Her ne kadar parçalama sezgiseli ve tarihçe ağacı sezgiseli benzer bir performans sergiliyor olsa da, Parçalı Q-Kesim yönteminin, küresel bir yaklaşım kullanarak, anlamlı bir alt hedef tanımlaması yapılabilmeden önce tüm durum uzayının araştırılmasını gerektirdiğini hatırlatmak gerekir. Öte yandan, tarihçe ağacı sezgisi, bir alt-hedef tanımlandığı anda kullanılabilir. Ayrıca, Parçalı Q-Kesim'in çalışması için ilk kaynak ve hedef durumların önceden sağlanması gerekir, böylece makul bir bölümleme bulunabilir. Bu durumların seçimi problemlidir ve her zaman elle tanımlamak her zaman kolay olmayabilir. Öte yandan, tarihçe ağacı sezgiseli, problem uzayına özgü hiçbir ek bilgi olmadan çalışabilir.

9.5 Sonuç

Bu makalede, bir tarihçe ağacı sezgisel yöntemi kullanılarak hedefe yönelik bir tercih başlatma kümesi üretme yöntemi önerdik. Yöntem, bir tercihin başlatma kümesini, tercihi kullanan durumların daha önce deneyimlenen hedeflere giderken faydalı olacağı garantisiyle kısıtlamaktadır. Daha yüksek kalite tercihlere ulaşarak, yeni yaklaşım, diğer yöntemler ile karşılaştırıldığında, ek hesaplama zamanı ve bellek gerektirmeden, etmenin öğrenme performansını arttırmaktadır. Deneyler, önceden tanımlı alt hedefler ve iyi bilinen alt hedef tanımlama algoritmaları kullanılarak otomatik olarak oluşturulmuş alt hedeflerle birlikte yapılmıştır.

Gelecek için olası bir çalışma yönü, üretilen tercihlerin yönünü belirleyen kök durumun seçimi için iyileştirmeler olabilir. Tanımlanan bir alt hedefin ulaşılmasının en faydalı olduğu bir kök durumu seçmek faydalı olabilir veya farklı yönlerde aynı son durumuna gitmeyi amaçlayan çoklu tercihlere neden olan çoklu kökler seçilebilir. Ayrıca, oluşturulan ağaç, tercih politikasının inşası için de doğrudan kullanılabilir. Eksik olan tek parça, etmeni ağaçtaki yollarda yönlendiren eylemlerdir. Tecrübe tekrarı mekanizmasında kullanılan yapay ödüller, her bir durum için en iyi hedef ağacın o durumun üst durumuna karşılık gelecek şekilde değiştirilebilir.

10 SONUÇLAR VE GELECEK ÇALIŞMALAR

Bu raporda önerilen yöntemleden ilki, öğrenme etmeninin öğrenmenin erken evrelerinde hedef duruma giden yolda önemli durumları tanımlamasına yardımcı olan, Yerel Kökler adı verilen

ağaç tabanlı otomatik alt hedef bulma yöntemidir. Yerel Kökler yöntemi, genellikle hedef durumlar olan ödül zirvelerinde uygulanabilir. Tercihler çatısını kullanarak, öğrenen etmen belirlenen alt hedeflere ulaşmak için soyutlamalar hazırlayabilir. Yöntem, her ziyaret edilen durumdan hedef duruma doğru yönlendirilen kısayolların birleşme noktalarını yerel olarak tanımlamak için ağaç tabanlı bir metrik kullanmaktadır.

Öğrenme hızı açısından, Yerel Kökler, üzerinde deney yapılan tüm problemler için standart Q-Öğrenme'den daha iyi performans göstermektedir. Ayrıca, bulduğu alt hedeflerin diğer yöntemlerin bulduklarına göre daha kötü olmadığını göstererek diğer yerel yöntemler ile ortalamada benzer performansa sahiptir.

Test edilen diğer grafik temelli yöntemlerle karşılaştırıldığında, Yerel Kökler'in zaman karmaşıklığı daha düşüktür. Öte yandan, bölüm başına ortalama CPU süreleri karşılaştırıldığında, Yerel Kökler, en düşük teorik zaman karmaşıklığına sahip fakat her adımda çağrılması gereken RN dahil olmak üzere, diğer tüm yöntemlerden daha iyi performans göstermektedir. Yerel Köklerin genel olarak daha yüksek etkinliğe sahip daha az sayıda alt hedefi belirlediği de gösterilmiştir. Üstelik RN ve L-Cut'dan farklı olarak ek bir parametre gerektirmemektedir.

Yerel Kökler yöntemi çerçevesinde, olası bir yeni araştırma doğrultusu, problemin yapısına daha az bağımlı parametrelerle çalışan ve gürültüyü yerel alt hedef bilgisinden ayırabilecek alternatif bir yol bulmaktır. Ayrıca, bu parametrelerin otomatik algılanışı, burada sunulan tüm öğrenme sürecinde çalışan yöntemler için önemli bir gelişme olabilir.

Bu raporda önerilen bir diğer yeni yöntem ise, tarihçe ağacı kullanan bir hedef odaklı tercih üretim yöntemidir. Bu yöntem, bir tercihin başlatma kümesini, onun kullanılmasının faydalı olduğu durumlara sınırlar. Bu kapsamda yeni bir açılım, oluşturulan tercihlerin yönünü belirlemek için kök durum seçimi üzerinde odaklanabilir. Buna ek olarak, oluşturulan ağacı kullanarak tercih yerel politikasının oluşturulması tercih kalitesini artırma potansiyeline sahiptir. Raporda bu yöntem iki farklı başlıkta anlatılmış olup, yöntemin daha geniş bir analizini ikinci başlıkta sunulmaktadır.

Bir başka bölümde, McGovern'ın alt hedef bulma için kullanılan Farklı Yoğunluk algoritmasını hesaplama süresi açısından geliştirmek için yeni bir fikir olan kavram filtreleme yöntemi önerilmiştir. Yazarların bilgisine göre bu, aynı zamanda, tamamen gözlemlenebilir ve kısmen gözlemlenebilir problemleri de kapsayan otomatik alt-hedef bulma tekniğini açıkça işleyen ilk

çalışmadır. Deneysel sonuçlar önerilen yöntemin mevcut literatürdeki benzerlerinden daha geniş bir problem kategorisini kapsaması açısından önemli bir adım olduğunu göstermektedir. Gelecekteki bir çalışma olarak, yöntemin içindeki “tıkanıklık oranı” ölçütünün sosyal ağlar gibi diğer alanlarda da yararlı olabileceği değerlendirilmektedir.

Geçişsel darboğaz fikri ise, NSM algoritmasını hızlandırmak için önerilmiştir. Buna ek olarak, darboğaz iyileştirmesini daha da geliştirmek için orijinal NSM algoritmasının durum tanımlamasına bir ekleme önerilmiştir. Deneyler, önerilen yöntemlerin NSM'nin kendi mekanizmasının avantajlarından ödün vermeden öğrenmeyi hızlandırdığını göstermiştir. Geçişsel darboğazlarının tanımlanmasının nasıl otomatikleştirileceği ve problemlerdeki alt hedef benzerliklerine daha az hassas olan daha iyi NSM durum yapılanmasının nasıl bulunacağı konusunda daha fazla araştırmaya öncülük edebileceği değerlendirilmektedir. Öğrenme deneyimine dayalı yapay ödül mekanizmasının nasıl otomatik olarak nasıl oluşturulacağı konusu sonraki adım olarak ele alınabilir.

NSM algoritmasının performansını iyileştirmek için sunulan başka bir yöntem de algoritmaya dahil edilen indeksleme mekanizmasıdır. Bu mekanizma algoritmanın temel birimi olan eylem-gözlem-ödül üçlülerini anahtar olarak kabul eden akıllı bir adresleme sistemini, mevcut durumun en yakın komşularını hızlı bir şekilde bulması için kullanır. Deneyler, önerilen yöntemin, NSM algoritmasının öğrenme performansını koruduğu halde çalışma süresini azalttığını göstermiştir.

11 KAYNAKÇA

Albus, J. S. 1991. “Outline for a theory of intelligence”. *IEEE Transactions on Systems, Man, and Cybernetics* 21 (3): 473-509. https://doi.org/10.1109/21.97471.

Alpaydın, Ethem. 2004. *Introduction to Machine Learning*. The MIT Press.

Altman, N. S. 1992. “An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression”. *The American Statistician* 46 (3): 175-85. https://doi.org/10.1080/00031305.1992.10475879.

Atkeson, Christopher G. 1992a. “Memory-based approaches to approximating continuous functions”. *SANTA FE INSTITUTE STUDIES IN THE SCIENCES OF COMPLEXITY-PROCEEDINGS VOLUME-*, 12:521–521. ADDISON-WESLEY PUBLISHING CO. http://www-2.cs.cmu.edu/~awm/15781/readings/paper.ps.

———. 1992b. “Memory-based approaches to approximating continuous functions”. *Nonlinear Modeling and Forecasting*, 503–521.

Aydın, Hüseyin, Erkin Çilden, ve Faruk Polat. 2017. “Using Transitional Bottlenecks to Improve Learning in Nearest Sequence Memory Algorithm”. . 29th International Conference on Tools with Artificial Intelligence, ICTAI 2017. Boston, MA, USA.

Bellman, Richard Ernest. 1957. *Dynamic Programming*. Princeton, NJ: Princeton University Press.

Bradtke, Steven J., ve Michael O. Duff. 1994a. “Reinforcement learning methods for continuous-time markov decision problems”. *Advances In Neural Information Processing Systems*, 7:393-400. Cambridge, MA: MIT Press. http://books.nips.cc/papers/files/nips07/0393.pdf.

———. 1994b. “Reinforcement learning methods for continuous-time markov decision problems”. *Advances In Neural Information Processing Systems*, 7:393–400. NIPS ’94. MIT Press.

Brandes, Ulrik. 2001. “A Faster Algorithm for Betweenness Centrality”. *The Journal of Mathematical Sociology* 25 (2): 163–177.

Cassandra, Anthony R., Leslie Pack Kaelbling, ve Michael L. Littman. 1994. “Acting optimally in partially observable stochastic domains”. *Proceedings of the Twelfth National Conference on Artificial Intelligence*, 2:1023-28. Menlo Park, CA, USA: AAAI Press. http://portal.acm.org/citation.cfm?id=199480.199520.

Chen, Fei, Shifu Chen, Yang Gao, ve Zhenduo Ma. 2007. “Connect-based subgoal discovery for options in hierarchical reinforcement learning”. *Proceedings of the Third International Conference on Natural Computation*, 4:698–702. Haikou, Hainan, China: IEEE. https://doi.org/10.1109/ICNC.2007.312.

Chrisman, Lonnie. 1992. “Reinforcement learning with perceptual aliasing: the perceptual distinctions approach”. *Proceedings of the Tenth National Conference on Artificial Intelligence*, 183-88. AAAI Press. http://dl.acm.org/citation.cfm?id=1867135.1867164.

Crook, Paul Anthony. 2006. “Learning in a State of Confusion: Employing Active Perception and Reinforcement Learning in Partially Observable Worlds”. Ph.D. thesis, UK: University of Edinburgh.

Demir, Alper, Erkin Çilden, ve Faruk Polat. 2016a. “A History Tree Heuristic to Generate Better Initiation Sets for Options in Reinforcement Learning (extended abstract)”. *ECAI 2016 - 22nd European Conference on Artificial Intelligence*, 1644–1645.

———. 2016b. “Local Roots: A Tree-Based Subgoal Discovery Method to Accelerate Reinforcement Learning”. *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2016, Riva del Garda, Italy, September 19-23, 2016, Proceedings, Part II*, 361–376. https://doi.org/10.1007/978-3-319-46227-1_23.

———. 2017a. “Generating Effective Initiation Sets for Subgoal Driven Options”. *(Dergi makale taslağı, hakem değerlendirmesinde)*.
———. 2017b. “A concept filtering approach for diverse density to discover subgoals in reinforcement learning”. *Proceedings of the 29th IEEE International Conference on Tools with Artificial Intelligence*, 1–5. ICTAI ’17. Boston, MA. https://doi.org/10.1109/ICTAI.2017.00012.
Dietterich, Thomas G. 2000. “Hierarchical reinforcement learning with the MAXQ value function decomposition”. *Journal of Artificial Intelligence Research* 13: 227-303.
Digney, Bruce L. 1998. “Learning hierarchical control structures for multiple tasks and changing environments”. *Proceedings of the fifth international conference on simulation of adaptive behavior on From animals to animats*, 321–330. Cambridge, MA, USA: MIT Press. http://dl.acm.org/citation.cfm?id=299955.299998.
Dijkstra, Edsger W. 1959. “A note on two problems in connexion with graphs”. *Numerische mathematik* 1 (1): 269–271.
Dung, Le Tien, Takashi Komeda, ve Motoki Takagi. 2007. “Reinforcement learning in non-markovian environments using automatic discovery of subgoals”. *SICE, 2007 Annual Conference*, 2601-5. https://doi.org/10.1109/SICE.2007.4421430.
———. 2009. “Solving POMDPs with Automatic Discovery of Subgoals”. *Theory and Novel Applications of Machine Learning*, editör Meng Joo Er ve Yi Zhou, 229–238. InTech.
Freeman, Linton C. 1977. “A Set of Measures of Centrality Based on Betweenness”. *Sociometry* 40 (1): 35--41. https://doi.org/10.2307/3033543.
Girgin, Sertan, Faruk Polat, ve Reda Alhajj. 2010. “Improving reinforcement learning by using sequence trees”. *Machine Learning* 81 (3): 283-331.
Goel, Sandeep, ve Manfred Huber. 2003. “Subgoal discovery for hierarchical reinforcement learning using learned policies”. *In Proceedings of the 16th International FLAIRS Conference*, editör Ingrid Russell ve Susan M. Haller, 346-50. St. Augustine, Florida, USA: AAAI Press.
Gosavi, Abhijit. 2009. “Reinforcement learning: a tutorial survey and recent advances”. *INFORMS Journal on Computing* 21 (2): 178–192. https://doi.org/10.1287/ijoc.1080.0305.
Hauskrecht, Milos. 2000. “Value-function approximations for partially observable markov decision processes”. *Journal of Artificial Intelligence Research* 13: 33–94.
Hochreiter, Sepp, ve Jürgen Schmidhuber. 1997. “Long short-term memory”. *Neural Computation* 9: 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735.
Hwang, Woochang, Young-rae Cho, Aidong Zhang, ve Murali Ramanathan. 2006. “Bridging centrality: identifying bridging nodes in scale-free networks”. *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 20–23. http://alfred.cse.buffalo.edu/tech-reports/2006-05.pdf.
Jaakkola, Tommi, Satinder P. Singh, ve Michael I. Jordan. 1995. “Reinforcement learning algorithm for partially observable markov decision problems”. *Advances in Neural Information Processing Systems*, 7:345-52. Cambridge, MA: MIT Press.
Jiang, Bin, ve Christophe Claramunt. 2004. “Topological Analysis of Urban Street Networks”. *Environment and Planning B: Planning and Design* 31 (1): 151-62. https://doi.org/10.1068/b306.
Jong, Nicholas Kenneth, Todd Hester, ve Peter Stone. 2008. “The utility of temporal abstraction in reinforcement learning”. *Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems*, 1:299–306. Estoril, Portugal: International Foundation for Autonomous Agents and Multiagent Systems. http://dl.acm.org/citation.cfm?id=1402383.1402429.

Jonsson, Anders, ve Andrew Barto. 2006. “Causal graph based decomposition of factored MDPs”. *Journal of Machine Learning Research* 7 (Kasım): 2259–2301.

Kaelbling, Leslie Pack, Michael L. Littman, ve Anthony R. Cassandra. 1998. “Planning and acting in partially observable stochastic domains”. *Artificial Intelligence* 101 (1-2): 99-134. https://doi.org/10.1016/S0004-3702(98)00023-X.

Kaelbling, Leslie Pack, Michael L. Littman, ve Andrew P. Moore. 1996. “Reinforcement learning: a survey”. *Journal of Artificial Intelligence Research* 4: 237-85.

Kamaya, Hiroyuki, Haeyeon Lee, ve Kenichi Abe. 2000. “Switching Q-learning in partially observable Markovian environments”. *Intelligent Robots and Systems, 2000.(IROS 2000). Proceedings. 2000 IEEE/RSJ International Conference on*, 2:1062–1067. IEEE. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=893160.

Kazemitabar, Seyed Jalal, ve Hamid Beigy. 2008. “Automatic discovery of subgoals in reinforcement learning using strongly connected components”. *Proceedings of the 15th International Conference on Neuro- Information Processing*, Part I:829–834. Auckland, New Zealand: Springer-Verlag. http://dl.acm.org/citation.cfm?id=1813488.1813597.

Kheradmandian, Ghorban, ve Mohammad Rahmati. 2009. “Automatic abstraction in reinforcement learning using data mining techniques”. *Robotics and Autonomous Systems* 57 (11): 1119–1128. http://dx.doi.org/10.1016/j.robot.2009.07.002.

Konidaris, George, ve Andre S. Barreto. 2009. “Skill discovery in continuous reinforcement learning domains using skill chaining”. *Advances in Neural Information Processing Systems*, 1015–1023.

Lin, Long-Ji. 1991. “Programming robots using reinforcement learning and teaching”. *Proceedings of the ninth National conference on Artificial intelligence - Volume 2*, 781-86. Anaheim, California: AAAI Press. http://dl.acm.org/citation.cfm?id=1865756.1865798.

———. 1992a. “Self-Improving Reactive Agents Based on Reinforcement Learning, Planning and Teaching”. *Machine Learning* 8 (3-4): 293–321. http://dx.doi.org/10.1007/BF00992699.

———. 1992b. “Self-improving reactive agents based on reinforcement learning, planning and teaching”. *Machine Learning* 8 (3): 293–321.

———. 1993. “Reinforcement Learning for Robots Using Neural Networks”. Pittsburgh, PA, USA: Carnegie Mellon University, School of Computer Science.

Lin, Long-Ji, ve Tom M. Mitchell. 1992. “Memory approaches to reinforcement learning in non-markovian domains”. Technical Report. Pittsburgh, PA, USA: Carnegie Mellon University.

Littman, Michael Lederman. 1994. “Memoryless policies: theoretical limitations and practical results”. *From Animals to Animats 3: Proceedings of the third International Conference on Simulation of Adaptive Behavior*, 238-45. Cambridge, MA: MIT Press. http://portal.acm.org/citation.cfm?id=189829.189893.

Loch, John, ve Satinder P. Singh. 1998. “Using eligibility traces to find the best memoryless policy in partially observable markov decision processes”. *Proceedings of the Fifteenth International Conference on Machine Learning*, 323-31. San Francisco, CA: Morgan Kaufmann Publishers Inc. http://portal.acm.org/citation.cfm?id=645527.657452.

Mannor, Shie, Ishai Menache, Amit Hoze, ve Uri Klein. 2004. “Dynamic abstraction in reinforcement learning via clustering”. *Proceedings of the Twenty-first International Conference on Machine Learning*, 71-78. ACM. http://doi.acm.org/10.1145/1015330.1015355.

Maron, Oded, ve Tomás Lozano-Pérez. 1998. “A framework for multiple-instance learning”. , 570–576. Cambridge, MA, USA: MIT Press. http://dl.acm.org/citation.cfm?id=302528.302753.

McCallum, Andrew. 1996a. “Reinforcement Learning with Selective Perception and Hidden State”. Ph.D. thesis, NY: University of Rochester.

———. 1996b. “Reinforcement Learning with Selective Perception and Hidden State”. Ph.D. thesis, NY: University of Rochester.
McCallum, R. Andrew. 1993. “Overcoming Incomplete Perception with Utile Distinction Memory”. *In Proceedings of the Tenth International Conference on Machine Learning*, 190–196. Morgan Kaufmann.
McGovern, Amy. 1998. “acQuire-macros: An algorithm for automatically learning macro-actions”. *The Neural Information Processing Systems Conference Workshop on Abstraction and Hierarchy in Reinforcement Learning*.
McGovern, Amy, ve Andrew G. Barto. 2001. “Automatic discovery of subgoals in reinforcement learning using diverse density”. *Proceedings of the Eighteenth International Conference on Machine Learning*, 361-68. MA, USA: Morgan Kaufmann Publishers Inc. http://dl.acm.org/citation.cfm?id=645530.655681.
Mcgovern, Amy, Richard S. Sutton, ve Andrew H. Fagg. 1997. “Roles of Macro-Actions in Accelerating Reinforcement Learning”. *In Grace Hopper Celebration of Women in Computing*, 13–18.
McGovern, Elizabeth Amy. 2002. “Autonomous Discovery of Temporal Abstractions from Interaction with an Environment”. Ph.D. thesis, Amherst, MA, USA: University of Massachusetts Amherst.
Menache, Ishai, Shie Mannor, ve Nahum Shimkin. 2002. “Q-Cut---Dynamic Discovery of Sub-goals in Reinforcement Learning”. *13th European Conference on Machine Learning Proceedings*, 295-306. Helsinki, Finland,: Springer-Verlag. https://doi.org/10.1007/3-540-36755-1_2.
Moore, Andrew W. 1995. “Memory-Based Learning for Control”. *Artificial Intelligence Review*.
Moore, Andrew W., Christopher G. Atkeson, ve Stefan A. Schaal. 1995. “Memory-Based Learning for Control.” DTIC Document. http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA311505.
Moore, Andrew William. 1990. “Efficient memory-based learning for robot control”. UCAM-CL-TR-209. University of Cambridge, Computer Laboratory. http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-209.pdf.
Parr, Ronald, ve Stuart Russell. 1998. “Reinforcement learning with hierarchies of machines”. *Advances In Neural Information Processing Systems*, 10:1043-49. Cambridge, MA, USA: MIT Press. http://papers.nips.cc/paper/1384-reinforcement-learning-with-hierarchies-of-machines.
Pendrith, Mark D., ve Michael McGarity. 1998. “An analysis of direct reinforcement learning in non-Markovian domains”. *Proceedings of the Fifteenth International Conference on Machine Learning*, 421-29. San Francisco, CA: Morgan Kaufmann Publishers Inc. http://dl.acm.org/citation.cfm?id=645527.657309.
Peshkin, Leonid, Nicolas Meuleau, ve Leslie Pack Kaelbling. 1999. “Learning policies with external memory”. *Proceedings of the Sixteenth International Conference on Machine Learning*, 307-14. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. http://portal.acm.org/citation.cfm?id=645528.657642.
Rummery, G. A., ve M. Niranjan. 1994. “On-line Q-learning using connectionist systems”. CUED/F-INFENG/TR 166. Cambridge, UK: Cambridge University Engineering Department.
Russell, Stuart J., ve Peter Norvig. 2003. *Artificial Intelligence: A Modern Approach*. Second. Pearson Education.
Schneider, Je G. 1995. “Robot skill learning through intelligent experimentation”. Citeseer. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.24.5702&rep=rep1&type=pdf.
Shi, Jianbo, ve Jitendra Malik. 2000. “Normalized cuts and image segmentation”. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 22 (8): 888–905.

Sigaud, Olivier, ve Olivier Buffet. 2010. *Markov Decision Processes in Artificial Intelligence*. ISTE - Wiley.

Simsek, Ozgur. 2008. “Behavioral Building Blocks for Autonomous Agents: Description, Identification, and Learning”. Ph.D. thesis, University of Massachusetts Amherst.

Simsek, Ozgur, ve Andrew G. Barto. 2004. “Using relative novelty to identify useful temporal abstractions in reinforcement learning”. *Proceedings of the Twenty-first International Conference on Machine Learning*, 95–102. ICML ’04. ACM.

Sondik, Edward Jay. 1971. “The optimal control of partially observable Markov decision processes”. Ph.D. thesis, Stanford, CA, USA: Stanford University.

Stolle, Martin, ve Doina Precup. 2002. “Learning options in reinforcement learning”. *Proceedings of the 5th International Symposium on Abstraction, Reformulation, and Approximation*, editör Sven Koenig ve Robert C. Holte, 2371:212-23. London, UK: Springer Berlin / Heidelberg.

Stone, Peter, Richard S. Sutton, ve Gregory Kuhlmann. 2005. “Reinforcement learning for RoboCup soccer keepaway”. *Adaptive Behavior* 13 (3): 165–188. https://doi.org/10.1177/105971230501300301.

Sutton, Richard S. 1988. “Learning to predict by the methods of temporal differences”. *Mach. Learn.* 3 (1): 9–44. https://doi.org/10.1023/A:1022633531479.

Sutton, Richard S., ve Andrew G. Barto. 1998. *Reinforcement Learning: An Introduction*. Adaptive Computation and Machine Learning. MIT Press. http://books.google.com/books?id=CAFR6IBF4xYC.

Sutton, Richard S., Doina Precup, ve Satinder Singh. 1999. “Between MDPs and semi-MDPs: a framework for temporal abstraction in reinforcement learning”. *Artificial Intelligence* 112 (1-2): 181-211. http://dx.doi.org/10.1016/S0004-3702(99)00052-1.

Şimşek, Özgür. 2008. “Behavioral Building Blocks for Autonomous Agents: Description, Identification, and Learning”. Ph.D. thesis, University of Massachusetts Amherst.

Şimşek, Özgür, ve Andrew G. Barto. 2004. “Using relative novelty to identify useful temporal abstractions in reinforcement learning”. *Proceedings of the Twenty-first International Conference on Machine Learning*, 95-102. ACM. http://doi.acm.org/10.1145/1015330.1015353.

Şimşek, Özgür, Alicia P. Wolfe, ve Andrew G. Barto. 2005. “Identifying useful subgoals in reinforcement learning by local graph partitioning”. *Proceedings of the 22nd international conference on Machine Learning*, 816–823. Bonn, Germany: ACM. http://doi.acm.org/10.1145/1102351.1102454.

Taghizadeh, Nasrin, ve Hamid Beigy. 2013. “A Novel Graphical Approach to Automatic Abstraction in Reinforcement Learning”. *Robotics and Autonomous Systems* 61 (8): 821-35. https://doi.org/10.1016/j.robot.2013.04.010.

Theocharous, Georgios, ve Leslie Pack Kaelbling. 2004. “Approximate planning in POMDPs with macro-actions”. *Advances In Neural Information Processing Systems*, 16:775-82. MIT Press.

Watkins, Chris. 1989. “Learning from Delayed Rewards”. Ph.D. thesis, UK: Cambridge University.

Watts, Duncan J, ve Steven H Strogatz. 1998. “Collective dynamics of ‘small-world’networks”. *nature* 393 (6684): 440–442.

Wei, Y.-C., ve C.-K. Cheng. 1991. “Ratio cut partitioning for hierarchical designs”. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 10 (7): 911–921.

Wiering, Marco, ve Juergen Schmidhuber. 1996. “HQ-Learning: discovering markovian subgoals for non-markovian reinforcement learning”. Technical Report IDSIA-95-96. Istituto Dalle Molle Di Studi Sull Intelligenza Artificiale.

Yang, Bo, ve Jiming Liu. 2008. “Discovering Global Network Communities Based on Local Centralities”. *ACM Transactions on the Web* 2 (1): 1-32. https://doi.org/10.1145/1326561.1326570.

Yoshikawa, Takeshi, ve Masahito Kurihara. 2006. “An acquiring method of macro-actions in reinforcement learning”. *IEEE International Conference on Systems, Man, and Cybernetics*, 6:4813-17. https://doi.org/10.1109/ICSMC.2006.385067.

TÜBİTAK
PROJE ÖZET BİLGİ FORMU

Proje Yürütücüsü:	Prof. Dr. FARUK POLAT
Proje No:	215E250
Proje Başlığı:	Kısmi Gözlemlenebilir Ardışık Karar Vermede Alt Hedef Tespiti
Proje Türü:	1001 - Araştırma
Proje Süresi:	24
Araştırmacılar:	
Danışmanlar:	
Projenin Yürütüldüğü Kuruluş ve Adresi:	ORTA DOĞU TEKNİK Ü. MÜHENDİSLİK FAKÜLTESİ BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ
Projenin Başlangıç ve Bitiş Tarihleri:	01/05/2016 - 01/05/2018
Onaylanan Bütçe:	246729.0
Harcanan Bütçe:	63065.07
Öz:	Kısmi gözlemlenebilirlik durumunda ardışık karar verme, algısal aynılığın ve büyük boyutluluğun getirdiği sorunlar nedeniyle zor bir problem olarak bilinmektedir. Öğrenme algoritmaları, ardışık karar verme problemine adaptif etmen bakış açısıyla yaklaşmaya çalışır, ve bazı yaklaşıklaştırma yöntemleri kullanarak söz konusu problemle başa çıkmayı dener. Takviye öğrenme (RL), özerk etmen modeline uyumluluğu, gerçekleştiriminin göreceli olarak kolay olması ve gerçek dünyadaki durumlara adaptasyonunun rahatlığı gibi bilinen bazı özellikleri nedeniyle, güçlü bir çevrim-içi öğrenme yöntemi olarak kabul görür. Teorik olarak Markov karar süreci (MDP) modelini temel alan RL yöntemlerinin, bazı varsayım ve kısıtlamalar çerçevesinde kısmi gözlemlenebilir MDP (POMDP) versiyonları mevcuttur. Literatürde, MDP problemlerinin küçük alt problemlere bölünerek her bir problemin daha az eforla çözüldüğü ve bu çözümlerin sonradan birleştirilip problemin bütünü için büyük çözümün üretildiği yöntemler vardır. Bu yöntemler arasında popüler olan bir yaklaşım, problemi doğal olarak parçalara ayıran alt-hedeflerin tespitidir. Bu kapsamda MDP-RL yöntemleri için yöntemler önerilmişse de kısmi gözlemlenebilir problemler için alt-hedef tespiti konusu halen olgunluğa ulaşmamıştır. Bu projenin amacı, POMDP-RL için alt-hedef tespiti alanında henüz hiçbir çalışma yapılmamış olan, gizli durumlar içeren problemler için bellek tabanlı RL algoritmaları konusunda yeni yöntemler üretmektir. Bu çalışma, hal-i hazırda MDP-RL için mevcut olan çevrim-içi alt-hedef tespit yöntemlerinin POMDP-RL modeline adaptasyonuna veya yeniden tasarlanmasına odaklanmakta, böylece öğrenme performansının herhangi bir çevrim-dışı müdahaleye gerek kalmaksızın artırılmasını amaçlamaktadır. Öncelikle, gerek MDP-RL, gerekse POMDP-RL yöntemleri için mevcut alt-hedef tespit yaklaşımları -öğrenme çıktılarını kullanan yöntemlere ağırlık verilerek- analiz edilmiştir. Ardından, olgun bir POMDP-RL yöntem ailesi olan bellek tabanlı algoritmalara odaklanılarak yeni bir alt-hedef tespit yöntemi geliştirilmiştir. Son olarak, literatürde yaygın kabul gören farklı problemler üzerinde karşılaştırmalı koşumlarla, önerilen yöntemlerin etkinliğinin doğrulanması sağlanmıştır.
Anahtar Kelimeler:	Pekiştirmeli Öğrenme, Kısmi Gözlemlenebilir Markov Karar Süreci, Alt-Hedef Tespiti
Fikri Ürün Bildirim Formu Sunuldu Mu?:	Hayır

Projeden Yapılan Yayınlar:	1- Local Roots: A Tree-Based Subgoal Discovery Method to Accelerate Reinforcement Learning (Bildiri - Uluslararası Bildiri - Sözlü Sunum), 2- A History Tree Heuristic to Generate Better Initiation Sets for Options in Reinforcement Learning (Bildiri - Uluslararası Bildiri - Sözlü Sunum), 3- A Concept Filtering Approach for Diverse Density to Discover Subgoals in Reinforcement Learning (Bildiri - Uluslararası Bildiri - Sözlü Sunum), 4- Using Transitional Bottlenecks to Improve Learning in Nearest Sequence Memory Algorithm (Bildiri - Uluslararası Bildiri - Sözlü Sunum), 5- Automatic Identification Of Transitional Bottlenecks In Reinforcement Learning Under Partial Observability (Tez (Araştırmacı Yetiştirilmesi) - Yüksek Lisans Tezi), 6- Effective Subgoal Discovery And Option Generation In Reinforcement Learning (Tez (Araştırmacı Yetiştirilmesi) - Yüksek Lisans Tezi),