Kısmi Gözlemlenebilir Takviye Öğrenme için Dolaysız Soyutlama

Program Kodu: 1001

Proje No: 113E239

Proje Yürütücüsü:

Prof. Dr. Faruk POLAT

Araştırmacı(lar):

Dr. Erkin ÇİLDEN

Bursiyer(ler):

Coşkun ŞAHİN

EYLÜL 2015

ANKARA

Önsöz

Bu projede, genişletilmiş dizi ağacı isimli otomatik zamansal soyutlama tekniğinin kısmi gözlemlenebilir problem ortamları için genişletilmesi sağlanmıştır. Probleme hem model tabanlı, hem de modelden bağımsız bakış açısıyla yaklaşılmış, önerilen yöntemlerin her iki bakış açısının önde gelen temsilcilerinde hızlanma sağladığı gösterilmiştir. Yöntemlerin etkinliği, yaygın kabul gören problemler üzerinde deneylerle gösterilmiştir.

Bu çalışma TÜBİTAK-ARDEB Bilimsel ve Teknolojik Araştırma Projeleri 1001 Programı (Proje No: #113E239) kapsamında gerçekleştirilmiş olup TÜBİTAK-BİDEB Yurtiçi Yüksek Lisans Burs Programı tarafından da desteklenmiştir. Bu çalışmanın ortaya çıkmasında ve dolayısıyla niteliği yüksek bir doktora tezi (Çilden, 2014) ve bir yüksek lisans tezinin (Şahin, 2013) tamamlanmasına olanak sağlaması sebebiyle TÜBİTAK'a teşekkürlerimizi sunarız.

İçindekiler

Tablolar Listesi

Şekiller Listesi

ÖZET

KISMİ GÖZLEMLENEBİLİR TAKVİYE ÖĞRENME İÇİN DOLAYSIZ SOYUTLAMA

Pekiştirmeli öğrenme, özerk etmen bakış açısıyla, makine öğrenme yöntemleri arasında önde gelen bir yönlendirmesiz yöntem ailesi tanımlar. Markov karar süreci modeli, pekiştirmeli öğrenme algoritmaları için sağlam bir biçimsel temel oluşturur. Pekiştirmeli öğrenme yöntemlerinin üstüne zamansal soyutlama mekanizmaları inşa edilerek başarımlarında kayda değer artış elde edilebilmektedir. Eğer Markov karar süreci modelinin tam gözlemlenebilirlik varsayımı esnetilirse, ortaya çıkan kısmi gözlemlenebilir Markov karar süreci modeli, daha gerçekçi, ancak zor bir problem alanı tanımlar. Kısmi gözlemlenebilirlik altında pekiştirmeli öğrenme araştırmaları, algısal aynılık ve çok büyük durum uzayı sorunlarının yol açtığı olumsuz etkileri azaltacak tekniklere odaklanmıştır. Genel olarak, bu çalışmalar iki kategoriye ayrılabilir. Model tabanlı yaklaşımlar durum geçiş modelinin etmen tarafından erişilebilir olduğu varsayımına dayanır. Modelden bağımsız yaklaşımlarda ise durum bilgileri etmenden tamamen saklıdır.

Bu projede, bilinen bir sıralama tabanlı otomatik zamansal soyutlama tekniğini (genişletilmiş dizi ağacı metodu) kısmi gözlemlenebilir problemler için genelleştiren yöntemler önerilmektedir. Probleme hem model tabanlı, hem de modelden bağımsız bakış açısıyla yaklaşılmış, önerilen yöntemlerin her iki bakış açısının önde gelen temsilcilerinde hızlanma sağladığı gösterilmiştir. Yöntemlerin etkinliği, yaygın kabul gören problemler üzerinde deneylerle gösterilmiştir.

Anahtar Kelimeler: Pekiştirmeli Öğrenme, Kısmi Gözlemlenebilir Markov Karar Süreci, Zamansal Soyutlama, Genişletilmiş Dizi Ağacı

ABSTRACT

ABSTRACTION IN REINFORCEMENT LEARNING IN PARTIALLY OBSERVABLE ENVIRONMENTS

Reinforcement learning defines a prominent family of unsupervised machine learning methods in autonomous agents perspective. Markov decision process model provides a solid formal basis for reinforcement learning algorithms. Temporal abstraction mechanisms can be built on reinforcement learning and significant performance gain can be achieved. If the full observability assumption of Markov decision process model is relaxed, the resulting model is partially observable Markov decision process, which constitutes a more realistic but difficult problem setting. Reinforcement learning research for partial observability focuses on techniques to reduce negative impact of perceptual aliasing and huge state-space. In the broadest sense, these studies can be divided into two categories. Model based approaches assume that the state transition model is available to the agent. In the model free approaches, states are completely hidden from the agent.

In this project, we propose methods to generalize a known sequence based automatic temporal abstraction technique –namely, extended sequence tree method– to partial observability. We attack the problem in both model based and model free approaches, showing that our methods accelerate well known representatives of each perspective. Effectiveness of our methods are demonstrated by conducting experimentation on widely accepted benchmark problems.

Keywords: Reinforcement Learning, Partially Observable Markov Decision Process, Temporal Abstraction, Extended Sequence Tree

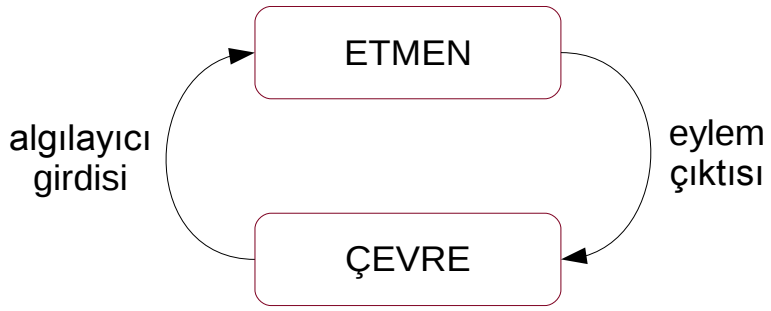
Kısaltmalar

b-GDA	Bellek tabanlı genişletilmiş dizi ağacı
BGDA	Bölümlenmiş genişletilmiş dizi ağacı
BMKS	Bölümlenmiş Markov karar süreci
CPU	Central processing unit (İng.)
DBA	Dinamik Bayes ağı
EOD	En olası durum
GDA	Genişletilmiş dizi ağacı
HSM	Hiyerarşik soyut makine
i-GDA	İnanç tabanlı genişletilmiş dizi ağacı
KGMKS	Kısmi gözlenebilir Markov karar süreci
KOT	Koşullu olasılık tablosu
KSD	Koşullu sonlanan dizi
MKS	Markov karar süreci
SPOVA	Smooth partially observable value approximation (İng.)
t-GDA	Tepkisel genişletilmiş dizi ağacı
TG-GDA	Tarihçe güdümlü genişletilmiş dizi ağacı
VAPS	Value and policy search (İng.)
YAD	Yasaklı alt-çözüm deposu
YAK	Yanıltıcı alt-çözüm kaldırma
YDP	Yapılandırılmış dinamik programlama
YDY	Yapılandırılmış değer yineleme
YHTY	Yapılandırılmış hareket tarzı yineleme
YMKS	Yarı-Markov karar süreci
YSB	Yararlı sonek belleği
ZF	Zamansal fark

1 GİRİŞ

Takviye öğrenme, memelilerin öğrenmesi alanındaki güçlü psikolojik ve bilişsel temellerine dayanmasının yanı sıra, makine öğrenme literatüründe de deneme-yanılma tabanlı yöntemlerden (öğrenme otomata, sınıflandıran sistemler, vb.) ve optimal kontrol (dinamik programlama) metotlarından temel alan, kısa ve sağlam bir tarihsel cebirsel birikime sahiptir. Takviye öğrenme problem, yaygın olarak, öğrenen *etmen* (İng. agent) kavramı ile birlikte tarif edilir. Bunun nedeni, muhtemelen sadece ilham alınan bilim dalları değildir; aynı zamanda 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.

"Etmen" kavramı için farklı tanımlar mevcuttur. Bu tanımlar, genel olarak soyutlama seviyesine veya çözülmeye çalışılan probleme göre değişir. En geniş anlamıyla, 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.

Wooldridge ve Jennings (1995) "etmen" teriminin iki farklı kullanım eğilimine işaret eder. Zayıf kullanım eğiliminde etmen, *özerklik (İng. autonomy)*, *sosyal kabiliyet*, *tepkisellik* ve *ön-etkinlik (İng. proactiveness)* gibi görece daha somut özellikleri kullanılarak tanımlanır. Zayıf kullanım eğilimiyle etmen terimi, temellerini esas olarak yazılım mühendisliği bakış açısından alır ve adından giderek daha çok söz edilen *etmen-tabanlı yazılım mühendisliği* disiplini çerçevesinde kullanılır. Güçlü kullanım eğilimi çerçevesinde ise etmen kavramı temellerini *yapay zeka (İng.*

artificial intelligence) araştırmalarından alır. Bu eğilim, zayıf eğilimli etmen kullanımını Shoham (1993) tarafından tanımladığı gibi *bilgi*, *inanç*, *niyet* ve *yükümlülük* ve hatta Bates (1994) tarafından tanımlandığı şekliyle bazı *duygusal* özellikler gibi çok daha zeka yönelimli kabiliyetlerle genişletir.

Etmen bakış açısıyla, yapay zeka araştırmaları, bir etmenin inançları doğrultusunda hedeflerine (İng. goal) ulaşmasını sağlayan yöntemlerin geliştirilmesine odaklanır. Bu bakış açısı, *akılcılık* özelliğini merkeze alır. Bazen gerçekleştirilecek en akılcı eylem mevcut olmasa bile, etmen yine de anlamlı bir karar verebilmelidir (Sutton ve Barto, 1998).

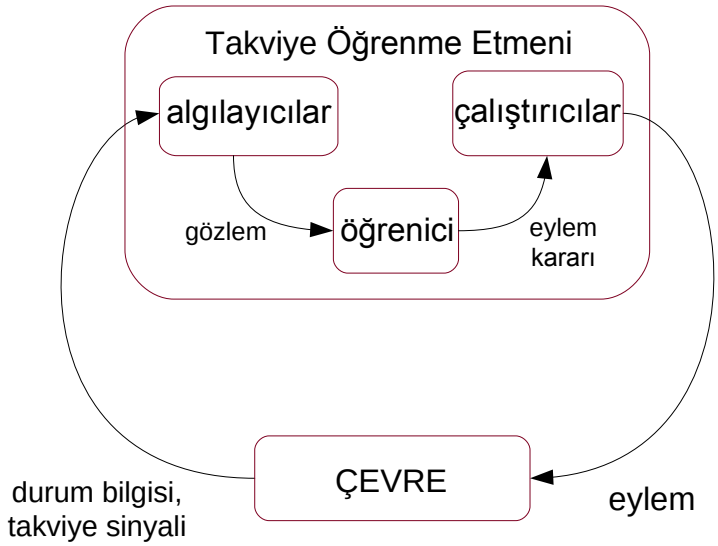
Makine öğrenme (İng. machine learning), yapay zeka alanının bir alt disiplini olarak, deneyim yoluyla akılcı davranış kalıplarının oluşturularak belli bir seviyede özerklik inşa edilmesine odaklanır. Makine öğrenmede temel bir ayrım, bu deneyimin denetimli (İng. supervised) veya denetimsiz (İng. unsupervised) 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) (MKS) 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) (YMKS) modeli, MKS modelindeki tek adımlı eylem varsayımının esnetildiği, MKS modelinin genişletilmiş bir halidir. Diğer bir deyişle, YMKS modelinde bir eylem bir yerine birden fazla ve belirsiz sayıda zaman aralıklarında devam edebilir. Neyse ki, takviye öğrenme yöntemleri de YMKS modelini kapsayacak şekilde genişletilebilmektedir.

Daha gerçekçi problemlerin özellikleri MKS modelindeki bazı kurallarının esnetilmesini gerektirmektedir. Kısmi gözlenebilir Markov karar süreci (İng. partially observable Markov decision process) (KGMKS), durumların ve durum geçiş dinamiklerinin etmen tarafından artık tümüyle gözlemlenebilir olmadığı, MKS modelinin genelleştirilmiş bir halidir. Etmen, eksiksiz durum bilgisi yerine sınırlı *gözlem (İng. observation)* bilgisi ile donatıldığından, KGMKS, takviye öğrenme algoritmaları için zor bir problem kategorisi tanımlar. Diğer bir deyişle, KGMKS modelinde kesin durum bilgisi *gizlidir (İng. hidden)*. Şekil 1.2 tekrar incelenecek olursa, MKS

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. KGMKS 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 vd., 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; McCallum, 1996).

Bir KGMKS problemindeki öğrenme sürecinde özel bir uygulama olarak, etmene altta yatan MKS modeli sağlanabilir. Bu bilgi önceden mevcut ise, kısmi gözlenebilirlik dahili bir *inanç durumu (İng. belief state)* olarak modellenebilmektedir. İnanç durumu, durum uzayı üzerinde bir olasılık dağılımı olarak tanımlanır (Kaelbling vd., 1996, 1998). Model güdümlü bu yaklaşımda inanç durumu gösterimi, anlamsal olarak salt gözlemlerden oluşan bir gösterimden daha zengindir. Bununla birlikte, elde edilen inanç durum uzayı pratik bir çözüme ulaşmak için fazla büyüktür. Klasik yöntemler makul bir süre içinde anlamlı bir çözüm bulamayabilmekte, ve hatta hiç bir çözüme ulaşamayabilmektedirler. Bu nedenle, inanç tabanlı kısmi gözlemlenebilir takviye öğrenme araştırmaları, ağırlıklı olarak durum uzayının küçültülmesi ve durum tahminini kolaylaştırıcı yöntemler üzerine yoğunlaşmıştır.

Literatürde genel kabul görmüş ortak bir adlandırma bulunmamakla beraber (Crook, 2006), bu çalışmada, KGMKS bağlamında altta yatan MKS modelinin mevcut olmadığı yöntemler *modelden bağımsız (İng. model free)* olarak anılacaktır. Benzer bir isimlendirme yaklaşımıyla, inanç tabanlı takviye öğrenme yöntemleri *model tabanlı (İng. model based)* olarak adlandırılacaktır.

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 vd. (1999) tarafından tanımlanmıştır. Yaygın kabul gören bu çerçeve, MKS modeli üzerine bir YMKS inşa ederek, takviye öğrenme kuramını zamansal soyutlama eylemleri içerecek şekilde genişletir.

Takviye öğrenme için zamansal soyutlama yöntemleri, ilk olarak etmene soyutlama bilgisinin öğrenme sürecinden önce ipucu olarak sağlanması şeklinde başlamıştır (Dietterich, 2000; Parr ve Russell, 1998). Daha sonra, takviye öğrenmeye paralel olarak soyutlama örüntülerinin (İng. pattern) otomatik olarak ortaya çıkarılması üzerinde odaklanılmıştır (Girgin vd. 2010; Hengst, 2002; Mannor vd., 2004; McGovern ve Barto, 2001; McGovern, 2002; Şimşek ve Barto, 2004; Stolle ve Precup, 2002). Şüphesiz, otomatik yöntemler felsefi tutarlılık bağlamında takviye öğrenmenin "çevrim içi" dokusuna daha uygundur.

Yakın geçmişte, az sayıda araştırmacı takviye öğrenme ile KGMKS problemi çözmede zamansal soyutlama tekniği kullanmayı denemiştir. Çok az sayıda çalışma, inanç tabanlı kısmi gözlemlenebilir takviye öğrenme için zamansal soyutlama konusunda öncülük etmiştir (Dung vd., 2007; Theocharous ve Kaelbling, 2004; Wiering ve Schmidhuber, 1997). Bu çalışmalar, temel olarak otomatik olmayan veya yarı otomatik soyutlama stratejileri önermişler, ağırlıklı olarak da bunu alt-hedef (İng. sub-goal) tespit ederek sağlamaya çalışmışlardır. Belleksiz kısmi gözlemlenebilir takviye öğrenme algoritmaları için otomatik soyutlama konusu ise çok daha az araştırılmış bir alandır (Yoshikawa ve Kurihara, 2006). Mevcut durumda, bellek kullanan kısmi gözlemlenebilir takviye öğrenme yöntemleri için tasarlanmış otomatik bir soyutlama çalışması ise yoktur.

Bu çalışmada, kısmi gözlemlenebilir problemler için takviye öğrenme sırasında tamamen otomatik şekilde zamansal soyutlama yapabilmek için yöntemler geliştirmesi amaçlanmıştır. Bu amaçla, model tabanlı (inanç tabanlı) yöntemler, belleksiz (veya reaktif) modelden bağımsız yöntemler ve bellek tabanlı modelden bağımsız yöntemler olmak üzere üç temel KGMKS araştırma akımı üzerinde çalışmalar yapılmıştır (Çilden, 2014). Geliştirilen yöntemler, esasen MKS için tanımlanmış, sağlam teorik altyapıya sahip bir otomatik zamansal soyutlama yöntemi olan *genişletilmiş dizi ağacı (İng. extended sequence tree)* yöntemini (Girgin vd., 2010) temel almaktadır. Son olarak, tam gözlemlenebilir BMKS için umut vaat eden bir zamansal soyutlama mekanizması önerilmiş (Şahin, 2015), kısmi gözlemlenebilir problemlerde durum uzayı bölümlemesi bilgisinin varlığı durumunda zamansal soyutlamaya katkı potansiyeline işaret eden öncül bir çalışma olarak literatüre kazandırılmıştır.

2 BAĞLANTILI ÇALIŞMALAR

Bu bölümde, gerekli biçimsel altyapının eksiksiz olarak tamamlanması için gerekli literatür özetlenmiştir. İlgili konular vurgulanmış ve literatürdeki bağlantılı yayınlar ayrıca işaret edilmiştir. İlk önce, Markov karar süreci, yarı-Markov karar süreci, kısmi gözlemlenebilir Markov karar süreci ve bölümlenmiş Markov karar süreci olmak üzere, ilgili karar süreci modelleri tanımlanmıştır. Ardından, Q-Öğrenme, Tekrarlanan Q-Öğrenme, Doğrusal Q-Öğrenme, Sarsa(λ) ve Yararlı Sonek Belleği gibi, bu çalışmada kullanılan takviye öğrenme algoritmaları özetlenmiştir. Son olarak, zamansal soyutlamalar konusundaki literatür özetlenmiş, ve kısmi gözlemlenebilir durumlar için takviye öğrenmedeki kayda değer soyutlama yöntemleri tartışılmıştır.

2.1 Markov Karar Süreçleri

Eğer bir problem, *dizi* şeklinde sunulan çeşitli *karar problemlerinden* oluşuyorsa, 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 vd., 1996). *Markov karar süreci* (MKS), *belirsizlik içeren dizisel karar problemleri* kapsamında Markov özelliğine sahip olanlar için biçimsel bir çerçeve tanımlar ve aşağıdaki şekilde tanımlanır:

Tanım 2.1. MKS modelinin bileşenleri şunlardır:

- durum sonlu kümesi, S,
- 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), ve
- ödül işlevi, $R: S \times A \to \mathcal{R}$.

$T(s, a, s')$, 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 MKS 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 MKS 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 MKS 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).

MKS çatısı ve değer işlevlerine odaklanan çözüm teknikleri takviye öğrenme için bir temel oluşturur. Bu yöntemler Bölüm 2.4'te özetlenmiştir.

2.2 Yarı-Markov Karar Süreçleri

MKS 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* (YMKS) olarak adlandırılır.

Matematiksel olarak bir YMKS, MKS tanımını aşağıdaki şekilde genelleştirir:

Tanım 2.2. YMKS 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 1994).

■

Kolayca anlaşılacağı üzere, MKS, YMKS'nin eylem adımı 1 olan özel bir halidir.

Hemen her gerçekçi MKS 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. YMKS modelinin önemi, bu tür *yetenek* veya *soyut eylemlerin* MKS 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 vd., 2005).

YMKS modeli ile sağlanan ve MKS üzerine inşa edilen zamansal eklentiler, zamansal soyutlama (ya da görev hiyerarşisi) yoluyla performans iyileştirme olanakları sunmaktadır. İlgili yöntemler Bölüm 2.6'da anlatılmaktadır.

2.3 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* (KGMKS), MKS modelinin genelleştirilmiş bir hali olarak bu ihtiyacı karşılamak amacıyla tasarlanmıştır.

Tanım 2.3. KGMKS 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.

■

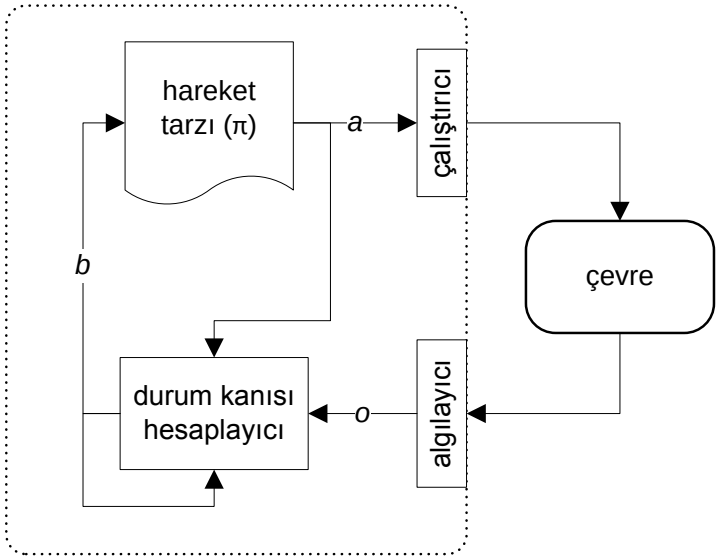
Kabaca, KGMKS, etmenin o anki durumunu mutlak olarak algılayamadığı bir MKS'dir. Bunun yerine, gerçekleşen eyleme ve varılan duruma bağlı olarak bir *gözlem* bilgisi oluşturulur (Kaelbling vd., 1998). Etmen tabanlı yaklaşım bakış açısıyla, KGMKS çerçevesi, etmen için sınırlı bir algılayıcı modeli oluşturmaya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

Kolayca anlaşılabileceği üzere, MKS, KGMKS modelinin $\Omega = S$ ve $\forall s \in S, O(s,a,s) = 1$ şartlarını sağlayan özel bir durumudur. KGMKS 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 KGMKS 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 KGMKS 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 MKS 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 KGMKS 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 vd., 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 vd., 1998; Sigaud ve Buffet, 2010).

Bir KGMKS 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 KGMKS çözmek, aşağıdaki özel yapıyı çözmekle aynıdır:

Tanım 2.4. İnanç-MKS 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

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

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

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

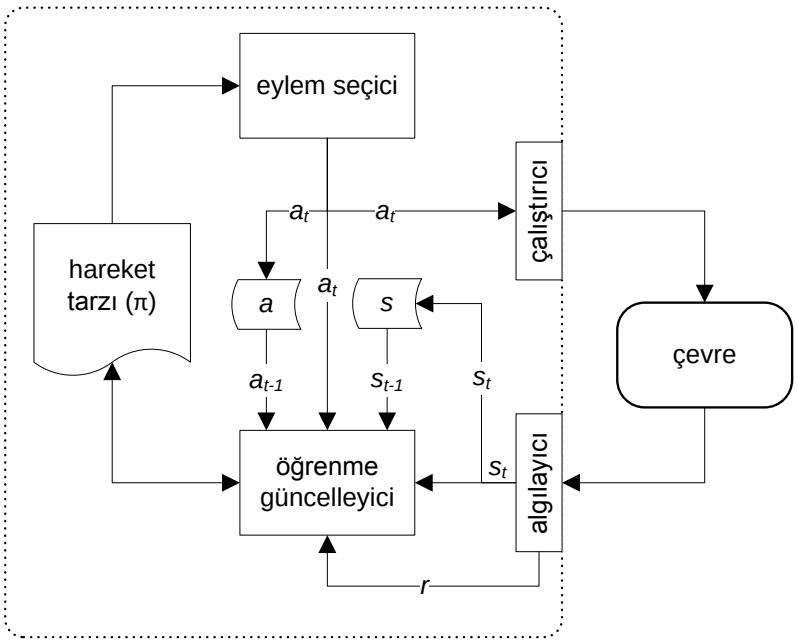
■

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

Neyse ki, inanç-MKS 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 vd., 1994). Böylece, sürekli-uzay MKS 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, KGMKS için yaklaşık sonuçlar arayan algoritmalar tasarlanagelmiştir.

Makine öğrenme bakış açısıyla, dikkate değer çalışmalar inanç uzayının işlemden geçirilerek bir MKS benzerinin oluşturması fikrine dayanır. Hauskrecht (2000), çalışmasında yaygın olarak kullanılan yöntemleri özetlemiştir. 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 KGMKS probleminin boyutunun benzer bir MKS modeline etkin bir şekilde indirgenip, MKS öğrenme yöntemleri kullanarak iyiye yakın bir çözüm aranması prensibine dayanmaktadır.



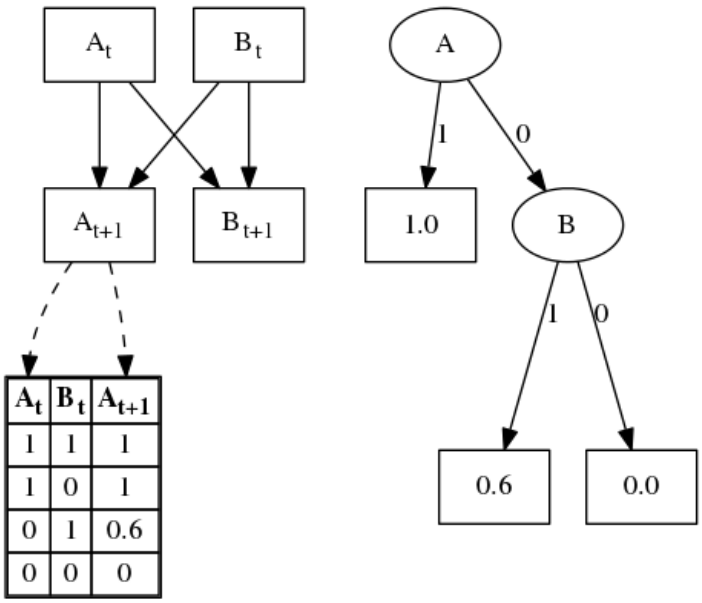
Şekil 2.2. Öğrenen etmen perspektifiyle takviye öğrenmenin genel yapısal görünümü.

2.4 Bölümlenmiş Markov Karar Süreçleri

Birçok öğrenme probleminde, durumlar çevrenin bazı özelliklerinin kombinasyonlarından oluşmaktadır. Bu nedenle durumları bu özelliklerin tutulduğu bir vektörle gösterilebilir. Fakat durum uzayı özellik sayısına bağlı olarak üstel (İng. exponential) olarak artar. Bölümlenmiş (İng. factored) Markov karar süreçleri (BMKS), bu tür problemlerde, yeni durum gösterimini kullanarak, daha kompakt geçiş, hareket tarzı ve ödül fonksiyonları tanımlamada kullanılır. Bölümlenmiş formda durumlar, her x_i değişkeninin kendi değer uzayı $Dom(x_i)$'den herhangi bir değer alabildiği $s =< x_1, x_2, \ldots, x_n >$ formatıyla gösterilirler. Böylece, durumlar arasında değişken değerleri açısından benzerlikler bulup bu durumları guruplayarak problemler daha kolay çözülebilir.

Dinamik Bayes ağları (DBA, İng. dynamic Bayesian networks), değişkenler arasındaki geçişleri ve koşullu olasılıkları modellemek için kullanılır. Eylem a için oluşturulmuş bir DBA, bu eylem uygulandığında değişkenlerin birbirlerini etkileme durumlarını gösteren bir yönlü döngüsüz çizgedir (İng. directed acyclic graph). Her eylem farklı değişkenleri farklı yönlerden etkilediğinden DBA'lar eyleme özel yaratılır. Verilen her x değişkeni için ilgili eylemin DBA'sından x'in bir sonraki adımdaki değerini etkileyen değişkenler bulunup, bunların alabilecekleri her değer için

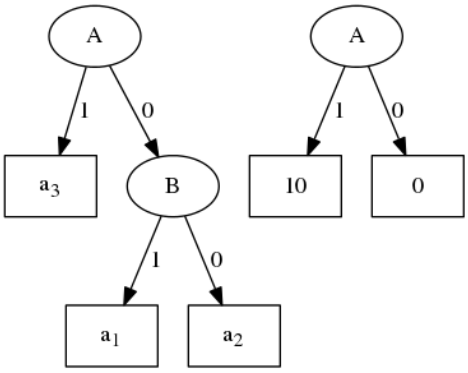
x'in yeni değerinin olasılık dağılımı koşullu olasılık tablolarında (KOT) (İng. conditional probability table) gösterilebilir. Fakat böyle bir yapı da değişken sayısına bağlı olarak üstel olarak büyür. Daha uygun bir gösterim ise farklı değer kombinasyonlarından, x'in yeni değeri için aynı olasılık dağılımına sahip olanlar arasındaki değişken değerleri benzerlikleriyle ağaç yapısı kurmakla mümkündür.



A_t	B_t	A_{t+1}
1	1	1
1	0	1
0	1	0.6
0	0	0

Şekil 2.3. Bölümlenmiş yapıda geçiş fonksiyonu örneği.

1 ve 0 değerleri alan A ve B değişkenlerinden oluşan bir çevreyi göz önüne alırsak, Şekil 2.3 eylemlerden birinin geçiş fonksiyonu için DBA, bu DBA'ya bağlı olarak A değişkeni için KOT ve onun ağaç yapısındaki karşılığını göstermektedir. DBA'ya göre, hem A, hem de B'nin bir sonraki değeri her ikisinin de şu anki değerlerine bağlıdır. KOT'ta ise A'nın $t+1$ zamanındaki değerinin 1 olmasının t anındaki değişken değerlerine göre hangi olasılıklarla gerçekleşeceği ifade edilmiştir. Örneğin; t anında A 0 ve B 1 ise, eylem uygulandığında A, 0.6 olasılıkla 1 olacaktır. Aynı tablodan $< A_t = 1, B_t = 1 >$ ve $< A_t = 1, B_t = 0 >$ durumlarında aynı eylem uygulandığında A'nın kesinlikle 1 olacağı sonucu da çıkarılabilir. Bu iki durum, A aynı değerde iken (1) B'nin kendi değer uzayındaki bütün değerleri $(0$ ve $1)$ aldığı durumları kapsadığı için A_{t+1}'in $A_t = 1$ iken aslında B'den bağımsız olduğu görülebilir. Bu nedenle, Şekil 2.3'teki ağaçtaki gibi sadece A'ya bağlı olarak ifade etmek mümkündür. Yeni yapının sağladığı kazanç, büyük çevrelerde daha kolay görülebilir.



Şekil 2.4. Bölümlenmiş yapıda hareket tarzı ve ödül fonksiyonları örnekleri.

Şekil 2.4'te ise ödül ve hareket tarzı fonksiyonlarının aynı yolla gösterildiği örnekler verilmiştir. Verilen hareket tarzı tabloda 4 girdi gerektirirken 3 ağaç yaprağıyla, ödüller ise sadece A değişkeniyle ifade edilebilir.

2.5 Tam Gözlemlenebilirlik Durumunda Takviye Öğrenme

Takviye öğrenme 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 MKS 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.

MKS problemlerinin Bölüm 2.1'de anlatılan yöntemler kullanılarak (hareket tarzı yineleme ve değer yineleme) çözülmesi, büyük ve karmaşık MKS'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 MKS 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 MKS 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 MKS 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:

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

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.

2.5.1 Zamansal Fark Yöntemi

Zamansal fark (ZF) 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.

2.5.2 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 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).

Algoritma 1 Q-Öğrenme

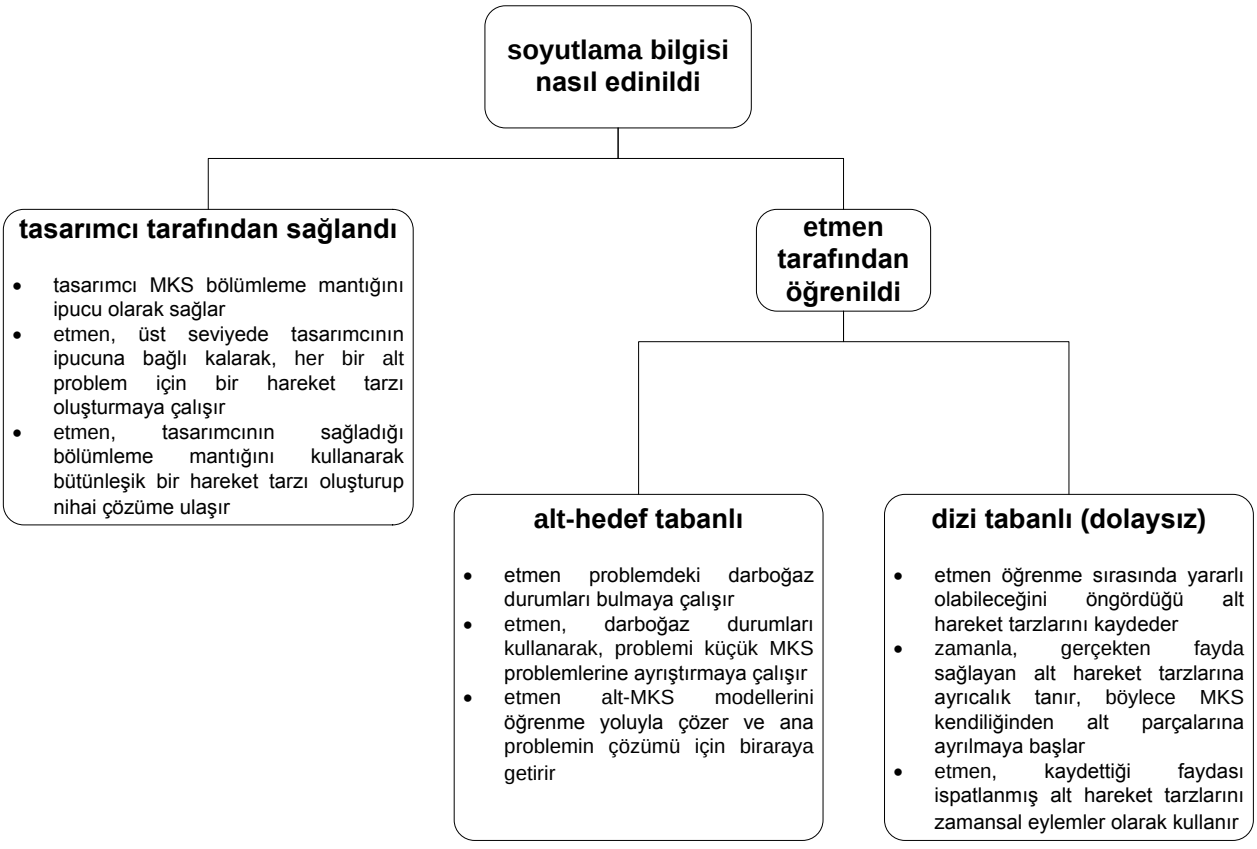
1: $Q(s, a)$ işlevine isteğe bağlı olarak ilk değerler ata
2: **repeat**
3: s mevcut durum olsun
4: **repeat**
5: Q işlevinin tanımladığı hareket tarzını kullanarak (ϵ-hırslı gibi bir strateji ile) s için a seç
6: a eylemini gerçekleştir, r değerini ve s' sonraki durumunu gözlemle
7: $Q(s, a) \leftarrow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

2.6 Takviye Öğrenmede Zamansal Soyutlama

Takviye öğrenme teorisi ve uygulamaları zamanla oturdukça, araştırmacılar çözüm kalitesi, öğrenme hızı, bellek alanı gereksinimleri gibi konularda iyileştirme yapmanın yollarını bulmaya odaklandılar. Öğrenme hızında elde edilen önemli iyileştirme olanaklarından biri de, çözüm uzayında sıkça tekrarlayan örüntülerin tespit edilerek etmenin tekrar tekrar aynı alt hareket tarzlarını keşfetmeye çabalamasının önüne geçilmesi olmuştur.

Literatürde yaygın olarak kabul gören ortak bir adlandırma mevcut değilse de, *zamansal soyutlama* (İng. temporal abstraction) bu kavramı tanımlamak için sıkça kullanılan bir terimdir. Bu terim, çözüm uzayındaki bir alt hareket tarzının, belirli bir ayrık zaman adımı boyunca uygulanan bir soyut eylem (veya makro eylem) olarak tanımlanabilmesi kavramı için kullanılır.

Şekil 2.5, takviye öğrenmede zamansal soyutlama yöntemlerini, soyutlama bilgisinin etmen tarafından nasıl elde edildiği bakış açısıyla sınıflandırmaktadır. Bu alandaki literatür özetinden önce, Bölüm 2.6.1'de konu ile ilgili önemli bir tanım olan *tercih çatısı* (İng. options framework) anlatılmıştır. Bölüm 2.6.2, soyutlama bilgisinin tasarımcı tarafından etmene sağlandığı yöntemlere odaklanmıştır. Son olarak, Bölüm 2.6.3 soyutlamaların otomatik olarak nasıl elde edilebileceğini açıklamaktadır. Bu kısımların tamamı, MKS modeli ile tanımlanmış bir problemin takviye öğrenme kullanılarak çözülmeye çalışıldığını varsaymaktadır.



Şekil 2.5: Takviye öğrenmede zamansal soyutlama yöntemlerinin, soyutlama bilgisinin etmen tarafından nasıl elde edildiği bakış açısıyla sınıflandırılması.

2.6.1 Tercih Çatısı

Bölüm 2.2'de özetlendiği üzere, YMKS modelinde eylemler, içeriğine bakılmaksızın olduğu haliyle kullanılan kesintisiz eylem akışları olarak varsayılır. Tüm zamansal soyutlama mekanizmaları, her bir durum-eylem alt dizisini, sanki bölünmez ve tek parçadan oluşan sıradan bir eylemmiş gibi kullanırlar.

Tercih çatısı (Sutton vd., 1999), zamansal olarak genişletilmiş eylemler içeren YMKS modelini, MKS üzerine ve yine MKS modelinin bileşenlerini kullanarak inşa edip takviye öğrenmeyi zenginleştirmeyi amaçlar. Bu yeni yapıda, MKS modelinin birim zamanda gerçekleşen durum geçiş dinamikleri değişmemiştir; ancak eylemler, birden fazla zaman adımı sürebileceği varsayımı ile genelleştirilir ve artık *tercih* (İng. option) olarak adlandırılır. Yani "tercih", aslında "eylem"in zamansal olarak genişletilmiş halidir.

Tanım 2.5. Bir tercih, şu bileşenlerden oluşur:

- başlatma kümesi, $\mathcal{I}$, tercihin başlatılabileceği durumların kümesini tanımlar
- tercih için yerel hareket tarzı, π_{tercih}, ve
- tercihin sonlanma koşulunu tanımlayan olasılık dağılım işlevi, β.

Bir tercih, etmen tarafından $s \in \mathcal{I}$ durumundayken başlatıldığında, tercihin yerel hareket tarzı olan π_{tercih}, β olasılık işlevi ile belirlenen koşul gerçekleşene kadar uygulanır. Bu koşul gerçekleştiğinde π_{tercih} sonlandırılır.

■

Tercih kavramının detaylarında farklı yorumlar söz konusu olabilmektedir. Bir yoruma göre, tercih içerisindeki eylem seçimi, sadece o anki durum dikkate alınarak yapılabilir, ki bu yorum *Markov tercih* olarak da adlandırılır. Bir diğer yorumda ise bu kural esnetilir. Bu alternatif yorumda π_{tercih} ve/veya β, sadece o anki durumu değil, tercihin başlangıcından o ana kadar gerçekleşen tüm adımları dikkate alır. Bu yoruma *yarı-Markov tercih* adı verilir. Tanım 2.6, tercih çatısı kullanan tüm yöntemler için gerekli olan ardışık olaylar kavramının tanımını vermektedir.

Tanım 2.6. Bir tarihçe aşağıdaki şekilde tanımlanır:

$$h_{t\tau} = s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1}, \dots, r_\tau, s_\tau$$

Bu tanımda etmen tarafından deneyimlenen durumlar, ödüller ve eylemler, t zamanından τ zamanına kadar ardışık olarak sıralanır.

■

β ve π, S kümesi yerine olası tüm tarihçeler kümesi üzerinden tanımlandığında, her eylemin bir tercih olduğu bir YMKS tanımlamak mümkün olmaktadır. Böylece, eylem kümesi yerine tercih kümesi kullanarak, YMKS için geçerli olan takviye öğrenme yöntemlerinin uygulanabilir kılınması mümkün olmaktadır.

Bir tercih, aslında sabit bir sınırlı hareket tarzıdır. Tercih çatısında sistem, durum uzayının her ayrık alt kümesi için bir tercih kümesi ile sınırlandırılmıştır. Etmenin, bir durumda iken, söz konusu durum ile başlatılabilen tercihler kümesinden birini seçmesi beklenir. Sistemin durum uzayının yeni bir alt kümesine girmesi durumunda, yeni bir tercih kümesi kullanılabilir hale gelir. Her tercih, sabit bir hareket tarzı olduğundan, ilkel eylemlerden oluşur. Daha genel bir anlamda, böyle bir düzenlemenin altında yatan mantık, durum uzayının her alt kümesi için geçerli sınırlandırılmış hareket tarzları arasında aramalar yapmaktır.

θ tercihinin s durumunda iken seçilmesi durumunda uygulanacak Q-Öğrenme güncelleme kuralı aşağıdaki gibidir:

$$Q(s,\theta) \leftarrow Q(s,\theta) + \alpha\left[r + \gamma^t \max_{\theta' \in \Theta_s} Q(s',\theta') - Q(s,\theta)\right] \tag{2.12}$$

Bu kuralda Θ_s, s durumunda iken başlatılabilecek tercih kümesi, $\theta \in \Theta$ (Θ, olası tüm tercihler kümesi olacak şekilde), ve diğer semboller temel Q-Öğrenme yöntemindeki gibidir. Q-Öğrenmenin bu uyarlaması, temel Q-Öğrenme yöntemindekine benzer koşullar altında, $s \in S$ ve $\theta \in \Theta$ için optimal Q-değerlerine yakınsar (Girgin, 2007).

Tercih çatısının önemi, takviye öğrenmede zamansal soyutlama yöntemleri için biçimsel bir temel oluşturmasıdır. Dahası, bazı otomatik zamansal soyutlama yöntemleri, tercih çatısı ile tanımlanan altyapıyı kullanır.

2.6.2 Hiyerarşik Takviye Öğrenme

MKS bakış açısıyla incelendiğinde, bazı problemlerin, (Forestier ve Varaiya, 1978) tarafından *hiyerarşik ayrıştırma* (İng. hierarchical decomposition) olarak adlandırılan bölme işlemine uygun, özel bir yapıya sahip olduğu görülür. Bu tür bir ayrıştırma sayesinde, bir MKS problemi, alt seviyelerdeki küçük çözüm uzaylarının üst seviyedeki kontrol optimizasyon probleminde kullanılması ile çözülebilmektedir. Diğer bir deyişle, tüm MKS problemi ayrık kümelere parçalanarak her bir kümedeki MKS probleminin, başka kümelerdeki MKS problemi dikkate alınarak veya alınmayarak, çözülmektedir. Bu çözümler, üst seviyedeki problemlere girdi

oluşturmakta, böylece üst seviye problemdeki bazı kararların oluşturulmasına artık gerek kalmamaktadır. Takviye öğrenme için hiyerarşik yöntemler geliştirme konusunda hatırı sayılır bir araştırma birikimi mevcuttur. Bu fikir halen takviye öğrenme araştırmalarının zorlu, ilgi çeken ve ümit veren bir alanını oluşturmaktadır (Gosavi, 2009).

Bu çalışmada, tarihsel nedenlerle, etmene soyutlama stratejisindeki tasarım ipucunu öğrenme öncesinde sağlayan zamansal soyutlama yöntemleri için *hiyerarşik takviye öğrenme* (İng. hierarchical reinforcement learning) algoritmaları/yöntemleri adlandırması kullanılmıştır. Doğrudan bu çalışmanın içeriği ile ilgili olmamasına rağmen, zamansal soyutlama stratejilerinin eksiksiz bir resmini vermek amacıyla, iki önemli hiyerarşik takviye öğrenme algoritması, izleyen alt kısımlarda kısaca anlatılmıştır.

MAXQ MAXQ, hedef MKS modelinin elle (etmen tasarımcısı tarafından) küçük MKS modellerine ayrıştırılarak, hedef MKS modeline ait değer işlevinin, alt-MKS modellerinin değer işlevlerinin hiyerarşik olarak birbirine eklenerek oluşturulmasını temel alan bir hiyerarşik takviye öğrenme yöntemidir. Yöntem, tasarımcının (programcının) yararlı alt hedefleri tespit ederek bu alt hedeflere ulaşmak için gerekli alt görevleri tanımlayabileceği varsayımına dayanır. Böylece, programcı takviye öğrenme için dikkate alınması gereken hareket tarzları kümesini sınırlayarak öğrenme süresini azaltabilir.

MAXQ yöntemi, çözülecek M MKS modelinin alt görevlerden (veya alt MKS'lerden) oluşan sonlu bir kümeye ayrıştırılmasını gerektirir. Bu alt görevler kümesi, M_0, tüm M MKS modelinin çözümü olan kök görev olacak şekilde, $\{M_0, M_1, \dots, M_n\}$ olarak gösterilir. Sonuçta oluşan çizge (İng. graph) MAXQ çizgesi olarak adlandırılır. Çözüm, çizgedeki her düğüm (İng. node) tarafından simgelenen alt görevler için ayrı birer hareket tarzının tanımlandığı hiyerarşik bir hareket tarzıdır, ve $\pi = \{\pi_0, \pi_1, \dots, \pi_n\}$ kümesi ile gösterilir.

MAXQ yöntemi, Q-Öğrenme yönteminin, MAXQ çizgesi kullanarak hiyerarşik bir hareket tarzı öğrenecek şekilde uyarlanmış halidir. Bu uyarlanmış yönteme MAXQ-Q adı verilir, ve çözüme yakınsadığı ispat edilmiştir (Dietterich, 2000).

Hiyerarşik Soyut Makineler *Hiyerarşik soyut makineler* (HSM, İng. hierarchical abstract machines) çatısında, sistem durumu ve eylemlere ek olarak, bir makine hiyerarşisinden bahsedilir. Makine hiyerarşisi, etmen tarafından herhangi bir durum için çalıştırılabilecek eylemleri sınırlandıran bir programdır. HSM makineleri, bir durum kümesi, geçiş işlevi ve

makinenin ilk durumunu belirten bir başlangıç işlevi ile tanımlanır. Makinenin içinde bulunabileceği dört durum tipi vardır:

- *eylem* durumu çevrede bir eylem gerçekleştirir,
- *çağrı* durumu başka bir makineyi alt program olarak çalıştırır,
- *seçim* durumu sonraki makine durumunu öngörülebilir olmayan bir yöntemle (İng. nondeterministically) seçer,
- *durma* durumu makinenin çalışmasını durdurur ve akış kontrolünü önceki çağrı durumuna geçirir.

Sonraki makine durumu, bir eylem veya bir çağrı durumundan sonra uygulanan mevcut makine durumuna bağlı bir geçiş işlevi ve ulaşılan çevre durumu ile belirlenir (Parr ve Russell, 1998).

HSM yöntemi, bazı kısıtlamalar altında HSM yapısı ile genişletilmiş bir MKS modelinin, problem karmaşıklığını azaltarak, en iyi çözüme ulaşılmasa imkan verecek indirgenmiş bir YMKS modelini tanımlamakta kullanılabileceği gerçeğine dayanır.

Bu nedenle, HSM, problemin büyüklüğü arttıkça zorlaşabilecek çözüm bulma eforunu azaltmakta kullanılabilmektedir. HSM kısıtlamaları, durum uzayının keşfine odaklanmak suretiyle, takviye öğrenmenin yeni bir çevre öğrenirken sıkça maruz kaldığı *kör arama* (İng. blind search) aşamasının olumsuz etkilerini azaltabilmektedir. Dahası, indirgenmiş bir durum uzayında işlem yapmak öğrenme hızını etkili şekilde arttırabilmektedir (Parr, 1998).

2.6.3 Zamansal Soyutlamaların Öğrenilmesi

Hiyerarşik takviye öğrenme çalışmalarının (Bölüm 2.5.2) ardından, bazı araştırmacılar, soyutlamaların öğrenilmesi konusuna, bir başka deyişle, takviye öğrenme prosedürü ile eşzamanlı olarak zamansal soyutlamaların otomatik olarak türetilmesine odaklanmışlardır (Barto ve Mahadevan, 2003). Bu alandaki araştırmalar iki alt başlıkta toplanabilir: *alt-hedef* tabanlı (İng. sub-goal based) çalışmalar ve *dizi tabanlı* (sequence based) çalışmalar. Dizi tabanlı çalışmalar dolaysız (İng. direct) olarak da adlandırılırlar. Alt hedef tabanlı yöntemler, alt hedeflerin bulunması ve bunlar kullanılarak yararlı bölümlemeler elde edilmesi esasına dayanır (Hengst, 2002; Mannor vd., 2004; McGovern ve Barto, 2001; Şimşek ve Barto, 2004; Stolle ve Precup, 2002). Dizi tabanlı (veya dolaysız) yöntemlerde ise, alt hedef tespiti yapılmadan, başarılı tarihçe dizilerinin analizi gerçekleştirilir (Girgin vd., 2010; McGovern 2002).

Takip eden iki alt bölümde, bu konuyla ilgili literatür özetlenmektedir. Bu çalışmada ağırlıklı olarak dizi tabanlı yöntemler kullanıldığından, dizi tabanlı yöntemler daha detaylı anlatılmıştır.

2.6.3.1 Alt-hedef Tabanlı Yöntemler

Stolle ve Precup (2002), takviye öğrenme yöntemlerine, tercihlerin otomatik olarak keşfedilmesine imkan veren istatistiksel bir eklenti önermiştir. Bu yöntemde, takviye öğrenme etmenin ilk olarak çevreyi keşfetmesi ve istatistiksel veri toplaması beklenir. Toplanan veriler daha sonra potansiyel alt hedefleri tespit etmek ve başlangıç durumlarını tanımlamak için kullanılır. Etmen daha sonra, klasik takviye öğrenme yöntemlerini kullanarak, tespit edilen tercihlerin iç hareket tarzlarını öğrenir. Son olarak, etmen, öğrenilen davranışları uygular. Yöntem, rastsal görevlerin çözüm rotalarında sıkça tekrar eden durumların önemli olabileceği, ve tercihler belirlemekte kullanılabileceği gibi basit bir varsayıma dayanır. Bu yaklaşım *darboğaz durumu* (İng. bottleneck state) fikrinin belki de en basit uygulaması olarak kabul edilebilir. Yöntemin bir dezavantajı ise, değer işlevinin bir matris veya tablo ile temsil edildiği varsayımına dayanmasıdır.

McGovern ve Barto (2001), etmenin yararlı alt hedefleri bir veri madenciliği (İng. data mining) yöntemi ile otomatik olarak keşfetmesine olanak veren bir yöntem önermişlerdir. Bu yöntem, öğrenen etmenin, çevresi ile etkileşimde bulundukça biriktirdiği davranışsal rotaların bir araya getirilerek analiz edilmesine dayanır. Bu çalışma, arama uzayında darboğaz durumların taranarak alt hedeflerin tespitine odaklanır. Araştırmacılar Dietterich vd. (1997) tarafından tanımlanan çoklu-örnek (İng. multiple-instance) öğrenme probleminden esinlenilerek, söz konusu darboğaz bölgeleri keşfetmek için Maron (1998) tarafından önerilmiş olan *ayrıştırıcı yoğunluk* (İng. diverse density) kavramının bir uyarlamasını kullanırlar. Burada önemli olan, ayrıştırıcı yoğunluk kavramı ile takviye öğrenmede alt hedef tanımındaki darboğaz bölge kavramı arasındaki benzerliktir. Dolayısı ile bu yöntemde temel olarak, maksimum ayrıştırıcı yoğunluk ile etmeni başarıya ulaştıran darboğaz bölge kavramları arasında bir bağlantı kurulmuştur.

Mannor vd. (2004), takviye öğrenmede alt hedef tespiti için durum uzayı üzerinde kümeleme (İng. clustering) tekniklerini kullanmayı dener. Yöntemdeki temel fikir, durum kümelerinin sıradan durumlar olarak değil, öğrenme sürecinin bir ara aşaması olarak ele alınmasıdır. Böylece, bir alt hareket tarzı tercih olarak tanımlanarak, bir durum kümesinden diğerine geçişlerin etkin olarak gerçekleştirilmesi sağlanır. Yöntem, öğrenme ve kümelenme olmak üzere iki kısımdan oluşur.

Şimşek ve Barto (2004) takviye öğrenme sırasında zamansal soyutlamaları *erişim durumları* (İng. access states) olarak adlandırdıkları durumları kullanarak belirlemeye çalışır. Bu çalışmada erişim durumu, etmenin, durum uzayının belirli bir kısmına geçişini sağlayan, mevcut olmaması halinde ise etmenin o anki durumundan hareketle söz konusu durum uzayı kısmına ulaşmasının zor veya imkansız olduğu alt hedef durumu olarak tanımlanır (örneğin, iki oda arasındaki kapı). Yöntem bu geçişi, "bir durumun etmene kısa vadede sağladığı yenilik" olarak tanımlanan *göreceli yenilik* (İng. relative novelty) kavramını kullanarak tespit eder. Bu amaçla, bir alt hedef tespit ederek etmeni o alt hedef durumuna ulaştırmak amacıyla zamansal olarak uzatılmış eylem tanımlayabilen göreceli yenilik algoritması tanımlanmıştır. Göreceli yenilik, matematiksel olarak, bir durumu takip eden durumların (kendisi dahil) yeniliği ile, önceki durumların yeniliğinin oranı olarak tanımlanır; ve göreceli yenilik algoritması bu sayıları kullanır. Buradaki kritik gözlem, hedef durumların diğerlerine göre daha yüksek göreceli yenilik skorlarına sahip olmasıdır. Yöntemin bir eksikliği, göreceli yenilik algoritmasının parametrelerinin deneysel ve sezgisel olarak probleme göre ayarlanması gerekliliğidir.

Hengst (2002) tarafından yapılan çalışma, durum uzayının hiyerarşik alt MKS alanlarına ayrıştırılarak görev hiyerarşisi oluşturulmasına güzel bir örnektir. Ayrıştırma, aşağıdaki şartlar altında mümkün olabilmektedir:

- durum vektörünün bazı değişkenlerinin sık değişmeyen çevre özelliklerini temsil ediyor olması,
- değerleri daha sık değişen değişkenlerin, daha kalıcı değişkenler bağlamında geçiş özelliklerini koruyor olmaları, ve
- alanlar arasındaki arayüzlerin kontrol edilebilir olması.

Önerilen algoritma HEXQ olarak adlandırılmıştır, ve hiyerarşi inşa etmek için durum değişkenlerini kullanır. Öngörülebilir (İng. deterministic) en kısa yol bulma problemleri için HEXQ evrensel bir en iyi hareket tarzı bulabilmektedir. HEXQ, MAXQ gibi, stokastik eylemler durumunda özyinelemeli optimaldir (İng. recursively optimal). Ancak HEXQ, aynı takip eden duruma erişen tercih çıkışlarını otomatik olarak birleştirme yeteneğinden yoksundur.

2.6.3.2 Dolaysız (Dizi Tabanlı) Yöntemler

Dolaysız veya *dizi tabanlı* zamansal soyutlama öğrenme yöntemleri göreceli olarak daha az çalışılmış alanlardır, ve temelleri tercih çatısına dayanmaktadır (Bölüm 2.6.1). Bu kategorideki

önemli iki yöntem *acQuire-macros* yöntemi ve ***genişletilmiş dizi ağacı*** (İng. extended sequence tree) yöntemleridir.

acQuire-macros Tercihlerin doğrudan öğrenilmesini sağlayan önemli yöntemlerden biri *acQuire-macros* algoritmasıdır (McGovern, 2002). Bu yöntemde, düzenli aralıklarla ortak eylem dizilerinin belirlenmesi için şu eylemler gerçekleştirilir:

- tekrarlayan başarılı rotaların belirlenmesi,
- benzer sonuçların elenmesi, ve
- kalan rotalar için tercihler yaratılması.

acQuire-macros algoritması, *koşullu sonlanan dizi* (KSD, İng. conditionally terminating sequence) olarak adlandırılan, özel bir semi-Markov tercih modeline dayanır. KSD, $C_i \subseteq S$ *devam kümesi* (İng. continuation set) ve $a_i \in A$ olmak üzere, sıralı n adet ikiliden oluşan $\sigma = \langle C_1, a_1 \rangle \langle C_2, a_2 \rangle \ldots \langle C_n, a_n \rangle$ dizisi ile tanımlanır. i adımında, a_i eylemi seçilerek uygulandığında, eğer mevcut durum s, C_i kümesine dahilse dizide bir sonraki zaman adımına ilerlenir, değilse dizi akışı sonlandırılır.

KSD'lerin en önemli özelliği, bir takviye öğrenme probleminde sıkça tekrarlayan faydalı eylemleri kompakt bir şekilde temsil edebilmeleridir. Bu gösterim, KSD'lerin çalıştırılan takviye öğrenme yöntemi için tercih olarak kullanılabilmesine olanak vermektedir.

Ancak, KSD doğrusal bir eylem akışını temsil ettiğinden, gözlem tarihçesine bağlı olarak farklı davranış biçimlerinin seçilebileceği hallerde kullanılamamaktadır. Halbuki, pek çok gerçekçi problem, doğası gereği, öğrenilmiş soyutlamaların bazı karar noktalarında çatallanarak tüm problem için bir çözüm hiyerarşisi oluşturduğu alt görevlerden oluşur.

Genişletilmiş Dizi Ağacı Yöntemi Genişletilmiş dizi ağacı (GDA, İng. etended sequence tree) yöntemi (Girgin vd., 2010), acQuire-macros algoritmasının geliştirilmesiyle oluşturulmuştur. Bu yöntem, yararlı tarihçeleri bir ağaç veri yapısına dönüştürerek, eylem seçiminde koşullu dallanma (İng. conditional branching) yapılmasına, dolayısıyla mevcut soyutlamaların daha sıkıştırılmış ve verimli şekilde yönetilmesine olanak verir.

Yöntem temelde, kayıt edilen tarihçeler kullanılarak alt hareket tarzlarının ezberlenmesi prensibine dayanır. GDA ile genişletilmiş tipik bir takviye öğrenme algoritmasının basitleştirilmiş betiğimsisi Algoritma 2 ile verilmiştir.

Algoritma 2 $GDA\_ILE\_TAKVIYE\_OGRENME$

1: T değişkenine boş GDA ilk değeri ata
2: π hareket tarzına ilk değer ata ▷ başlangıç hareket tarzı olarak isteğe bağlı herhangi bir işlev atanabilir
3: **repeat**
4: s durumunu gözlemle ve boş e tarihçesine ekle
5: **repeat**
6: $a \leftarrow EYLEM\_SEC(s, T)$ ▷ hem GDA hem de altta çalışan takviye öğrenme algoritmasını ele alacak şekilde güncellenmiş olmalıdır
7: a eylemini uygula, s' durumunu and r ödülünü gözlemle ▷ r anlık ödüldür
8: $\pi \leftarrow TAKVIYE\_OGRENME\_GUNCELLE(s, s', a, r)$
9: a, r ve s değerlerini e tarihçesinin sonuna ekle
10: $s \leftarrow s'$
11: **until** s sonlandırma durumu ise
12: $T \leftarrow DIZI\_AGACI\_GUNCELLE(T, e)$
13: **until** belli bir yakınsama ölçütü karşılanıyor ise

GDA yönteminin üç ana bileşeni vardır.

- İlk bileşen, GDA veri yapısıdır. GDA veri yapısı, ağaç formunda bir tercih deposu olarak kullanılır. Ağacın düğümleri devam kümeleri içerirken, ayrıtları (İng. edge) eylemlerle adlandırılmıştır. GDA veri yapısının biçimsel tanımı Tanım 2.7 ile verilmiştir (bu dokümanda genellikle açıkça belirtilmekle birlikte, "genişletilmiş dizi ağacı" terimi hem veri yapısını hem de yöntemin tamamını anlatmakta kullanılabilmektedir).

  Tanım 2.7. Genişletilmiş dizi ağacı, düğümler kümesi N ve ayrıtlar kümesi E bileşenlerinden oluşur. Her düğüm, o düğüme ulaşma yolunda deneyimlenmiş bir eylem dizisini temsil eder ve bu eylem dizisinin ağaçta bir tekrarı yoktur. Kök düğüm $\emptyset$ sembolü ile gösterilir ve boş eylem kümesini temsil eder. Eğer q düğümünün eylem dizisi, a eyleminin p düğümü ile temsil edilen eylem dizisine eklenmesi ile elde edilebiliyorsa, bu durumda p düğümü q düğümüne $\langle a, \psi \rangle$ etiketli bir ayrıt ile bağlıdır; ve bu bağ $\langle p, q, \langle a, \psi \rangle \rangle$

ile gösterilir. ψ, q düğümünün temsil ettiği eylem dizisinin hangi sıklıkla uygulandığını anlatan ayrıt nitelik (İng. eligibility) değeridir. Dahası, q düğümü, eğer gözlemlenen mevcut durum $\{s_1, \dots, s_k\}$ durumlarından biri ise a eyleminin p düğümünde seçilebileceğini anlatan $\langle s_1, \xi_{s_1}, R_{s_1} \rangle, \dots, \langle s_k, \xi_{s_k}, R_{s_k} \rangle$ yapısını barındırır. Bu yapıya q düğümünün devam kümesi adı verilir ve $cont_q$ ile gösterilir. R_{s_i}, p düğümü ile temsili edilen eylem dizisinin uygulanmasının ardından, etmenin s_i durumunda a eylemini seçmesi ile elde etmesi beklenen toplam birikmiş ödüldür. ξ_{s_i}, q düğümündeki s_i durumunun nitelik değeridir, ve a eyleminin s_i durumunda gerçekte ne sıklıkta seçildiğini belirtir.

■

GDA veri yapısı değerli tarihçelerin deposu olarak tasarlanmıştır. Bu yapıda saklanan her tarihçe, ayırt edici bir başlangıç durumu ile başlayan bir durum-eylem dizisini temsil eder. Bu noktada, *tarihçe* tanımını (Tanım 2.6), bir durum ile başlatılacak ve π hareket tarzı ile oluşturulacak şekilde yenide tanımlamak uygun olacaktır:

Tanım 2.8. Eğer bir tarihçe, s durumu ile başlar ve bir bölümün (İng. episode) sonuna ulaşana kadar (veya, en yüksek ödülün elde edilmesi gibi, başka belirli koşullar oluşana kadar) π hareket tarzı takip edilerek elde edilirse, bu tarihçeye s durumunun π-tarihçesi adı verilir.

■

Kısaca, her bölüm sonunda bir prosedür çalıştırılır ve –başlangıçta boş olan- GDA veri yapısını, başarılı π-tarihçeleri tespit edip eklemeye çalışarak günceller. Bu şekilde, altta yatan takviye öğrenme algoritmasının uygulaması sırasında başarıya ulaştığı belirlenen çözüm rotalarından oluşturulmuş bir rota ağacı, daha kazançlı olabileceği durumlarda işleme konmak üzere hazır hale getirilir.

Şekil 2.6 örnek bir GDA veri yapısını göstermektedir. Bu şekilde kullanılan problem, Littman vd. (1998) tarafından tanımlanan küçük gezinti ortamının (İng. tiny navigation environment) tam gözlemlenebilir versiyonudur. Ağaç, GDA yönteminin Q-Öğrenme ile birlikte çalıştırılmasıyla otomatik olarak oluşturulmuştur. Kök düğümün hemen altındaki düğümlerden başlayarak, her bir düğüm ve o düğümü bir üst düğüme bağlayan ayrıt, tercihte bir uygulama adımı tanımlamaktadır. Geçerli düğüm ve onun üst (İng. parent) ayrıtı şu şekilde yorumlanır: “eğer bu düğümdeki bir durum etmen tarafından gözlemlendi ise, etmen üst ayrıtta belirtilen eylemi gerçekleştirmelidir.” Bir uygulama adımından sonra, kontrol geçerli düğümün alt düğümlerinin bulunduğu düğüm seviyesine geçer.

Amaç, ağacı yapısını takip etmek suretiyle, kök düğümden bir yaprak (İng. leaf) düğüme ulaşana kadar bir eylem dizisi icra etmektir. Burada ağacın tanımladığı her yol (İng. path), çözüm uzayında tercih olarak kullanılabilecek başarılı bir alt hareket tarzıdır.

- İkinci bileşen, eylem seçim prosedürüdür (Algoritma 2'nin 6. satırında çağırılmaktadır). Eylem seçim prosedürü, kontrol akışını, altta yatan takviye öğrenme algoritmasının eylem seçim adımı ile GDA yönteminin eylem seçim adımı arasında her seferinde seçim yapacak şekilde değiştirilir. Yeni eylem seçim prosedürü, bu seçimi mevcut durumun takviye öğrenmedeki hareket tarzı işlevi ile hesaplanan beklenen değeri ile GDA veri yapısında biriktirilmiş deneyimlere ait beklenen değeri karşılaştırarak yapar.

  Değiştirilmiş eylem seçim mekanizması, her ayrık zaman adımında, bir tercih icrasını başlatıp başlatmamaya karar verecek şekilde tasarlanmıştır. Eğer bir tercih başlatılmamışsa, olağan takviye öğrenme eylem seçim mekanizması devrededir. Eğer bir tercih başlatılmışsa, kontrol akışı GDA veri yapısına geçirilerek kök düğümden akış başlatılır. Her zaman adımında, kontrol akışı, gözlemlenen mevcut durumu en iyi şekilde temsil eden devam kümesi elemanının bulunduğu alt düğüme geçer. Bu geçiş sırasında da üst ayrıtın etiketi olan eylem gerçekleştirilir.

  Bu yordam bir yaprak düğüme ulaşılana, veya geçerli düğüm o anki durumu iyi temsil edemeyene kadar tekrarlanır. Her iki durumda da, tercih icrası sonlandırılır. Bu arada, GDA izlenirken uygulanan her eylemden sonra, altta çalışan takviye öğrenme algoritmasının olağan tek adım güncellemesi gerçekleştirilerek, takviye öğrenme hareket tarzının deneyimlenen tercih uygulamasına göre güncellenmesi sağlanır.
- Üçüncü bileşen ağaç güncelleme mekanizmasıdır (Algoritma 3). Bu mekanizma, bir hedef durumuna ulaşıldığı (alternatif olarak, yüksek bir ödül edinildiği) her seferde devreye girer (Algoritma 2, satır 12). Önce, ümit vaat eden tüm π-tarihçeler, durum eşitlikleri kullanılarak, bölümün tüm tarihçesi içerisinden ayıklanır (Algoritma 3, satır 1). Sonra, oluşturulan tüm alt tarihçeler, dizileri kompakt olarak temsil edecek şekilde GDA veri yapısına kaynaştırılır. Bu kaynaştırmada, kök düğümden yaprak düğüme kadarki her yol, bir başarılı alt hareket tarzını temsil eder (Algoritma 3, satırlar 2-4). Güncelleme mekanizması, aynı zamanda nispeten daha az kullanılan, veya neredeyse hiç kullanılmayan yolların ağaçtan budanmasından (İng. prune) da sorumludur (Algoritma 3, satır 5). Budama işlemi, bir çürüme (İng. decay) ve eşik (İng. threshold) mekanizmasının GDA veri yapısındaki ψ ve ξ nitelik bilgilerini yöneterek, ağaçta aralıklarla kontrollü temizlik yapılması ile gerçekleştirilir.

Sonuç olarak, bu yöntemle ortaya çıkan genişletilmiş takviye öğrenme algoritması, GDA veri yapısını bir meta-eylem kılavuzu gibi kullanarak, yararlı zamansal soyutlamalar inşa edip onlardan uygun şekilde faydalanabilmektedir.

Algoritma 3 $DIZI\_AGACI\_GUNCELLE(T, e)$

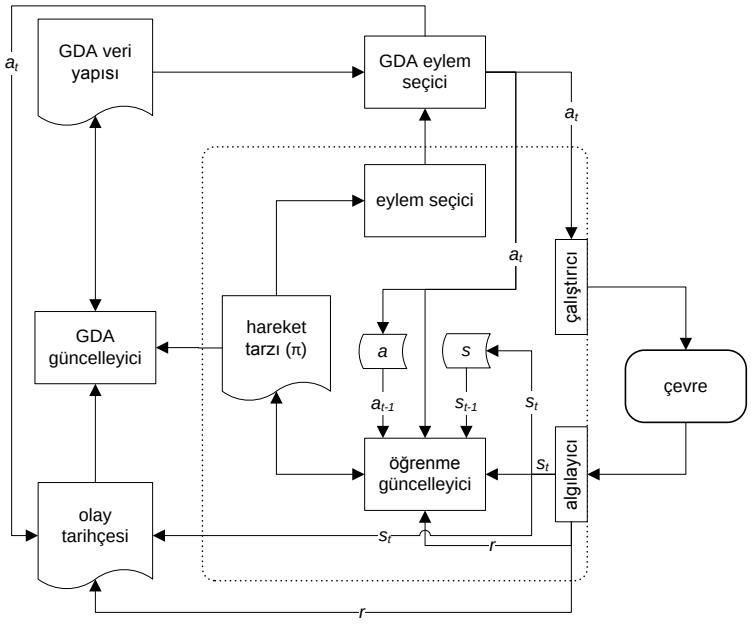
Require: T bir GDA veri yapısıdır
Require: e, etmen tarafından belirli bir zaman aralığında gözlemlenmiş, $s_1 a_1 r_2 \ldots s_{t-1} a_{t-1} r_t s_t$ formunda bir tarihçedir.
Ensure: güncellenmiş T
    $H \leftarrow OLASI\_TARIHCELER\_OLUSTUR(e)$
    for all $h \in H$ do
        $TARIHCE\_EKLE(h, T)$
    end for
    $DUGUM_GUNCELLE(T'nin\ kök\ d$üğü$m$ü$)$ ▷ bakım amacıyla ağacı özyinelemeli olarak tara
    return T

Şekil 2.6. Tam gözlemlenebilir küçük gezinti ortamı problemi için bir genişletilmiş dizi ağacı veri yapısı. F ileri, RR sağa-dön ve RL sola-dön eylemlerini simgelemektedir.

Altta çalışan takviye öğrenme algoritması değişmeden kaldığı için, eylem seçim mekanizmasının yeterli keşif imkanı sağlaması durumunda (yani, her durum-eylem ikilisinin sonsuz sıklıkta ziyaret edildiği durumda), GDA ile genişletilmiş bu öğrenme modeli, klasik takviye öğrenmedeki pek çok kuramsal özelliği –en iyi değer işlevine veya hareket tarzına yakınsamak gibi- muhafaza eder. Rapor edilen test sonuçları GDA yönteminin literatürdeki diğer bazı yöntemlere göre avantajlarını göstermektedir (Girgin vd., 2010).

Algoritma 2'yi bölümlerden oluşmayan (İng. non-episodic) problemler için uyarlamak kolaydır. Bu tür problemler için dizi ağacı, belirlenmiş bir "ödül tepe noktasına" ulaşana kadarki tarihçe kullanılarak güncellenir. Hemen ardından tarihçe silinir ve tüm GDA işlemleri yeni bölüm başlamış gibi yeniden başlatılır.

Şekil 2.2 ile resmedilen takviye öğrenme algoritmasının genel yapısal görünümünden hareket edersek, GDA mekanizması bu yapının etrafında bir kaplık (İng. wrapper) gibi tasvir edilebilir. Bu bakış açısıyla, Algoritma 2 ile tanımlanan yapısal görünüm Şekil 2.7'de verilmiştir. Burada eylem seçimi bir GDA filtresinden geçmektedir. Bölüm tarihçesi bir tarihçe veritabanında tutulmakta ve GDA veri yapısını oluşturmakta kullanılmaktadır. Buradaki önemli hususlardan biri de, noktalı dörtgenin dışında kalan kısmın çevrim dışı (İng. off-line) olarak çağırılmasıdır (yani, takviye öğrenme sırasında çağırılmamaktadır). Bu yapı, paralel veya boruhattında (İng. pipeline) işleme gibi mimari iyileştirmelere olanak sağlamaktadır.



Şekil 2.7. GDA ile genişletilmiş takviye öğrenme algoritmasının yapısal görünümü.

2.7 Kısmi Gözlemlenebilirlik Durumunda Takviye Öğrenme

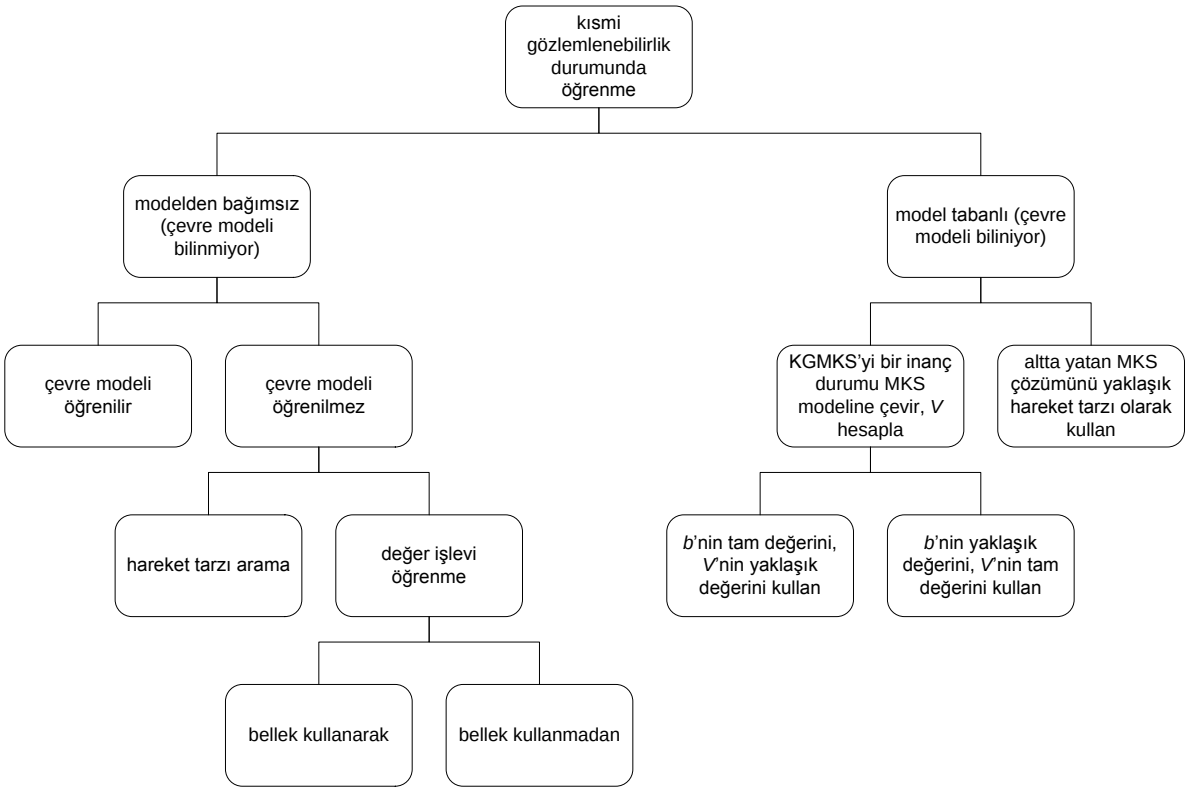
Takviye öğrenme yöntemleri her ne kadar doğru çözüme yakınsamak için problemin MKS biçiminde tanımlanmasını gerektirse de, bu yöntemlerin kısmi gözlemlenebilirlik içeren problemlerde kullanılmasına teknik bir engel yoktur. Bu anlamda, akla gelebilecek en basit uygulama, takviye öğrenmedeki durumları gözlemlerle değiştirmek olacaktır. Ne yazık ki, çoğu zaman bu yaklaşım en iyi çözüme yakınsamayı başaramaz. Genellikle bu başarısızlığın nedeni *algısal çakışma* (İng. perceptual aliasing) olarak adlandırılan olgudur.

Tam gözlemlenebilir (MKS) bir çevrede etmen herhangi bir zamanda eksiksiz durum bilgisine sahiptir. Eylemler öngörülebilir olmasa bile, ulaşılan durumun bilgisi etmen tarafından mutlaka algılanır. Bu senaryo, etmenin algılayıcılarının çevredeki her şeyi herhangi bir kısıtlama olmaksızın algıladığı varsayımına dayanır. Gerçek dünyada ise, bu varsayım neredeyse hiçbir zaman geçerli değildir.

Etmenin algılayıcılarının hatalı ve gürültülü (İng. noisy) olduğu daha gerçekçi bir senaryoda, etmenin farklı iki durumu benzer, ve hatta aynı algılaması son derece olasıdır. Daha biçimsel bir anlatımla, eğer etmen farklı iki durumda iken aynı gözlemi elde ediyorsa, bu duruma algısal çakışma adı verilir (Whitehead ve Ballard, 1991). Burada asıl problem, farklı iki durum farklı iki eylem gerektiriyorsa, etmenin bu gereksinimi salt gözlem mekanizması kullanarak karşılamasının imkansız olmasıdır.

Ayrıca, inanç durumu yapısı (Bölüm 2.3), inanç-MKS inşa ederek sürekli (İng. continuous) bir problem uzayı tanımlar. Bu da, Bölüm 2.4'te değinilen çok boyutluluk sorununun çözümü zor, hatta çözülemez hale gelmesine neden olabilmektedir.

KGMKS literatüründe takviye öğrenme çalışmaları, temelde, yaygın olarak kullanılan klasik takviye öğrenme algoritmalarının varyantlarıdır, ve genellikle algısal çakışma ve büyük durum uzayı sorunlarının yol açtığı olumsuz etkileri azaltmaya odaklanırlar.



Şekil 2.8. KGMKS problemleri için öğrenme yaklaşımlarının genel hatlarıyla sınıflandırılması.

KGMKS problemleri için mevcut öğrenme yaklaşımlarına genel bir bakış Şekil 2.8 ile gösterilmiştir. Pek çok araştırmacı KGMKS problemlerini çözmek için dinamik programlama ve planlama tabanlı yöntemlere odaklanmış olsa da, bu bölüm yalnızca takviye öğrenme bakış açısını, ve bu çalışmaya yararı dokunabilecek pratik destekleyici (İng. heuristic) yöntemleri anlatmaktadır. Bu çalışmada özelikle aşağıdaki yöntemler ele alınmaktadır:

- model tabanlı yöntemler (çevre modelinin bilindiği durum)
- modelden bağımsız yöntemler (çevre modelinin bilinmediği durum)
    - hafıza kullanmayan yöntemler: etmen tepkisel (İng. reactive) bir hareket tarzı oluşturmaya çalışır
    - hafıza tabanlı yöntemler: etmen bir çeşit hafıza kullanır

Her kategoride, en yaygın kullanılan veya atıfta bulunulan algoritmalar vurgulanmıştır. Daha farklı KGMKS çözüm teknikleri için, konunun daha geniş bir yelpazede ele alındığı şu kaynaklara başvurulabilir: (Cassandra, 1998; Murphy, 2000; Shani, 2007).

Takip eden alt bölümlerde ilgili literatür özetlenmiştir, anlatılan yöntemlerin avantajları ve kısıtlamaları belirtilmiştir.

2.7.1 Model Tabanlı Yöntemler: Çevre Modelinin Bilindiği Durum

Çoğu zaman, etmen tasarımına alttaki durum geçişleri ile ilgili ek bilgi sağlamak mümkün olabilmektedir. Etmen, algısal çakışma olan iki durumu halen net olarak ayırt edemese de, örneğin bir durum geçiş haritası, tek başına gözlem uzayından daha çok bilgi içerir. Bir KGMKS probleminin eksiksiz olarak çözülmesi zor olduğundan, önce bilinen modelin çözülüp KGMKS çözümü ile birleştirilmesi pratik olarak yararlı olabilmektedir. Diğer bir deyişle, inanç durumları veya inanç değerleri üzerinden yaklaştırma (İng. approximation) yöntemleri kullanarak kabul edilebilir kalitede bir çözüm bulunması mümkündür. KGMKS için takviye öğrenme literatüründe, bilinen çevre modeli bilgisini kullanan çözüm yöntemleri önemli bir kategori tanımlamaktadır. Bu yöntemler takip eden alt bölümlerde özetlenmiştir.

2.7.1.1 Altta Yatan MKS Modelinin Çözümünü Kullanan Yöntemler

Bir KGMKS probleminin çevre modelinin bilindiği, ve bu modelin MKS olarak modellenebildiği varsayıldığında, bu MKS modeline literatürde problemin *altta yatan* (İng. underlying) MKS modeli adı verilir. Bu tür bir KGMKS problemini çözmenin bir yolu, önce altta yatan MKS modelini çözüp, sonra bu çözümü destekleyici yaklaştırma yöntemleri kullanarak inanç durumu üzerinden birleştirmektir.

Cassandra (1998), çalışmasında bu yaklaşımın farklı örneklerini açıklamıştır. Bu yöntemlerden biri *en olası durum* (EOD, İng. most likely state) yaklaştırmasıdır. Bu sezgisel yaklaştırma yönteminde, etmen, altta yatan MKS modelini kullanarak içinde bulunma olasılığı en yüksek olan durumu seçer. Benzer bir başka yöntem ise Q_{MDP} yaklaştırmasıdır. Q_{MDP} yaklaştırması, hesaplanmış inanç durumundaki olasılık dağılımına göre ağırlıklandırılan inanç-MKS değer işlevinin tahminine dayanır.

2.7.1.2 İnanç Değeri Yaklaştırma Yöntemleri

KGMKS'leri takviye öğrenme ile yaklaşık olarak çözmenin bir diğer yolu ise inanç durumunu mutlak olarak takip ederken, inanç durumu MKS modelinin -parçalı doğrusal ve dışbükey olduğu bilinen- değer işlevini yaklaştırmalı olarak hesaplamaktır.

Bölüm 2.3'te kısaca açıklandığı üzere, sınırlı gözlem sorunuyla Markov özelliğini kaybetmeden başa çıkmanın yaygın bir yolu, etmene altta yatan MKS modelini sağlayarak problem tanımını genişletmek, böylece etmenin T and O işlevlerini kullanarak dahili bir inanç durumu oluşturmasına imkan vermektir. Bir b inanç durumu, S kümesi üzerinde bir olasılık dağılımıdır. $b(s)$, s çevre durumunun b inanç durumu tarafından atanmış olasılığı olsun ($\forall s \in S, 0 \leq b(s) \leq 1; \sum_{s \in S} b(s) = 1$ şartlarını sağlayacak şekilde). Her zaman adımında, b eski inanç durumu, a gerçekleşen eylemi, ve o gözlemi kullanılarak b' yeni inanç durumu tahmini hesaplanmalıdır. s' durumundaki yeni inanç durumu $b'(s')$ ile gösterilir ve aşağıdaki denklemle hesaplanır:

$$b'(s') = \frac{O(s', a, o) \sum_{s \in S} T(s, a, s') b(s)}{\Pr(o|a, b)} \tag{2.13}$$

Bir KGMKS modelini inanç-MKS modeline dönüştürmek mümkündür. Böylece, amaç bir sürekli uzay MKS problemini çözmeye dönüşür. Her ne kadar sürekli uzay MKS problemlerini çözmek çok zorsa da, inanç-MKS tabanlı (veya model tabanlı) takviye öğrenme yaklaşımları için pratik bir çözüm yaklaşımı mevcuttur. Bu yaklaşımda mutlak inanç durumu hesaplanır ve takip edilir, fakat inanç-MKS modelinin değer işlevi mutlak olarak değil, yaklaştırmalı olarak hesaplanır.

Parr ve Russell (1995) çalışmalarında SPOVA-RL (Smooth Partially Observable Value Approximation-reinforcement learning) adını verdikleri, değer işlevinin sürekli ve türevlenebilir bir gösterimini temel alan yeni bir takviye öğrenme yaklaşımı önermişlerdir.

Littman vd. (1998) çalışmasında aynı kategoride iki benzer takviye öğrenme algoritması tanıtır. İnanç tabanlı KGMKS modeli üzerinden tanımlanan bu algoritmalar, *Tekrarlanan Q-Öğrenme* (İng. Replicated Q-Learning) ve *Doğrusal Q-Öğrenme* (İng. Linear Q-Learning) yöntemleridir. Her iki yöntem de Q-Öğrenme algoritmasının vektörel tanımlanmış durumlar için uyarlamış halidir. Q değer işlevi, her bir a eylemi için ayrı bir q_a vektörü kullanılarak $Q_a(b) = q_a \cdot b$ denklemi ile yaklaştırmalı olarak hesaplanır. Her ne kadar eylem başına tek vektör gösterimi çoğu problem için yetersiz kalsa da, bu basit yaklaştırma yönteminin deneysel olarak etkin olduğu gösterilmiştir (Chrisman, 1992). Tekrarlanan Q-Öğrenme için güncelleme kuralı şu şekildedir:

$$\Delta q_a(s) = \alpha b(s)(r + \gamma \max_{a'} Q_{a'}(b') - q_a(s)) \tag{2.14}$$

Bu kuralda α öğrenme hızı, b inanç durumu, a gerçekleşen eylem, r anlık ödül ve b' ulaşılan inanç durumudur. Bu güncelleme kuralı, her durum geçişinin ardından her $s \in S$ için hesaplanır. Doğrusal Q-Öğrenmede ise, q_a vektörünün bileşenleri, Q değerlerinin tahmini için kullanılan

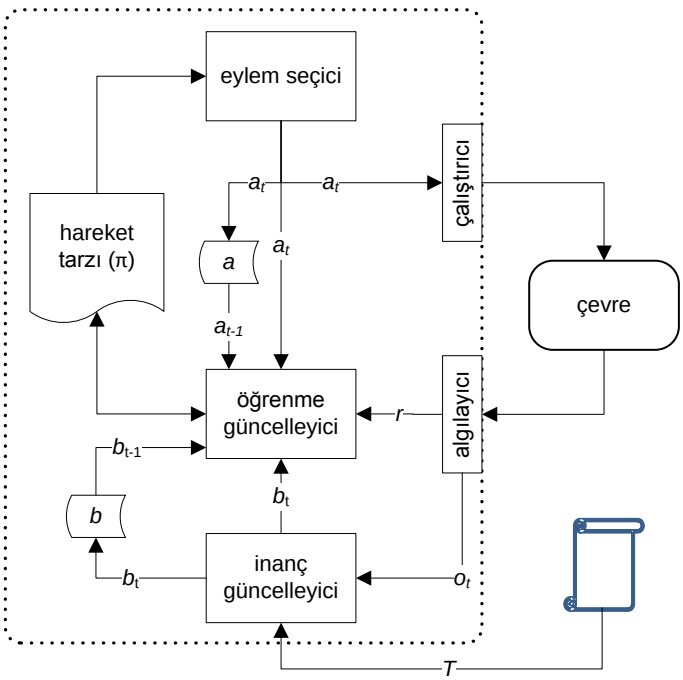
doğrusal işlevin katsayılarını yakalamak amacıyla ayarlanır. Denklem (2.14) üzerinde küçük bir değişiklikle, güncelleme kuralı aşağıdaki hale gelir:

$$\Delta q_a(s) = \alpha b(s)(r + \gamma \max_{a'} Q_{a'}(b') - q_a \cdot b) \tag{2.15}$$

Bu noktada, inanç durumu öngörülebilirse, (2.14) ve (2.15) denklemlerinin sıradan Q-Öğrenme güncellemesine indirgendiğini belirtmek yerinde olacaktır.

Şekil 2.2 ile verilen yapısal görünüme benzer şekilde, Tekrarlanan Q-Öğrenme ve Doğrusal Q-Öğrenme algoritmaları Şekil 2.9'deki gibi resmedilebilir. Burada önemli olan temel fark, etmenin artık tüm durum bilgisi yerine bir gözlem algılamasıdır. Ayrıca etmen, öğrenme güncelleme bileşenine yönlendirmeden önce kendi inanç durumunu güncellemek üzere, T durum geçiş işlevi ile donatılmıştır.

Bu çalışmada, Tekrarlanan Q-Öğrenme ve Doğrusal Q-Öğrenme yöntemleri, model tabanlı kısmi gözlemlenebilir takviye öğrenme kategorisini temsil etmek üzere seçilmiştir.



Şekil 2.9. KGMKS problemleri için inanç durumu tabanlı takviye öğrenme algoritmasının yapısal görünümü.

2.7.1.3 İnanç Durumu Yaklaşıklaştırma Yöntemleri

Alternatif bir yol olarak inanç-MKS için mutlak değer işlevi hesaplayıp, inanç durumunu yaklaştırmalı olarak takip etmek mümkündür. Boyen-Koller algoritması (Boyen ve Koller, 1998), Dinamik Bayes Ağı (DBA, İng. Dynamic Bayesian Network) ile kompakt şekilde temsil edilen bir çevre için gerçekleştirebilmektedir. Bu yöntemde, yaklaşıklaştırmalı inanç durumu güncellemesi için örnekleme yöntemi kullanılır.

2.7.2 Modelden Bağımsız Yöntemler: Bilinmeyen Çevre Modeli

Bu KGMKS problem kategorisi, zor bir senaryo tanımlar. Bu senaryoya göre, etmenin çevre hakkında herhangi bir ön bilgisi yoktur. Etmen deneme-yanılma döngüleri ile, sınırlı gözlemlenebilirlik çerçevesinde çevresine adapte olmaya çalışır.

2.7.2.1 Belleksiz Yöntemler

KGMKS problemlerinde takviye öğrenme uygulayabilmenin akla ilk gelen yolu, öğreniciden gizli olan "durumlar" yerine, öğrenicinin edinebildiği "gözlemler" kullanmaktır. Örneğin, etmen $Q(s,a)$ değerleri yerine $Q(o,a)$ değerlerini kullanabilir. Algısal çakışmanın olumsuz etkisinin şiddetine bağlı olarak (yeterince açıklayıcı olmamakla birlikte, bu şiddet bir problem için $|S|/|\Omega|$ oranı ile kabaca belirlenebilir), bu yöntemin başarı garantisi olduğu söylenemez.

Littman (1994) çalışmasında Markov olmayan çevrelerde, etmene önişleme için tüm çevrenin bir haritası sağlanmış olsa bile, başarılı, hatta kimi zaman sadece kabul edilebilir seviyede öngörülebilir belleksiz hareket tarzları bulmanın çok zor olduğunu göstermiştir.

Yine de, belleksiz yöntemler için birkaç iyileştirme mevcuttur. Loch ve Singh (1998), çalışmalarında, belli bir takviye öğrenme algoritma ailesine niteliklilik takibi (İng. eligibility trace) yöntemi ekleyerek (özellikle SARSA(λ) algoritması öne çıkmaktadır), belleksiz hareket tarzı çözümüne sahip kısmi gözlemlenebilir problemlerde başarılı deneysel sonuçlar elde etmişlerdir. Benzer şekilde, Baird ve Moore (1999), VAPS (İng. kısaltma: value and policy search) adını verdikleri belleksiz bir takviye öğrenme algoritması önermişlerdir. Bu algoritma, hem durum işlevi uzayında, hem de hareket tarzı uzayında arama yaparak, belleksiz bir hareket tarzı çözümüne sahip kısmi gözlemlenebilir çevrelerde, bir çözüme yakınsamaktadır.

Bu çalışmada, SARSA(λ) algoritması, KGMKS problemleri için modelden bağımsız takviye öğrenmede belleksiz algoritmaları temsilen, yaygın kullanımı nedeniyle tercih edilmiştir.

SARSA(λ) Algoritması Q-Öğrenme (Algoritma 1) algoritmasının önemli bir varyantı, SARSA algoritmasıdır (Sutton ve Barto, 1998). SARSA'da Q-Öğrenmedeki güncelleme kuralı (Denklem (2.11)) aşağıdaki şekilde yeniden düzenlenmiştir:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \quad (2.16)$$

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 4). *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 MKS yapısını varsayarak tasarlanmışsa da, niteliklilik takibinin belleksiz çözümler içeren KGMKS problemleri için başarılı sonuçlar verdiği deneysel olarak gösterilmiştir (Loch ve Singh, 1998).

Algoritma 4 $SARSA(\lambda)$

repeat
  $Q(s,a)$ işlevine isteğe bağlı ilk değerler, ve $e(s,a) = 0, \forall s,a$ şeklinde ilk değerler ata
  s,a için ilk değerler ata
  repeat
    a eylemini gerçekleştir, r ve s' gözlemle
    Q işlevinin tanımladığı hareket tarzını kullanarak s' için a' seç (ϵ-hırslı gibi bir strateji ile)
    $\delta \leftarrow r + \gamma Q(s',a') - Q(s,a)$
    $e(s,a) \leftarrow e(s,a) + 1$
    for all s,a **do**
      $Q(s,a) \leftarrow Q(s,a) + \alpha\delta e(s,a)$
      $e(s,a) \leftarrow \gamma\lambda e(s,a)$
    end for
    $s \leftarrow s'$
    $a \leftarrow a'$
  until s son durum olana kadar
until bir yakınsama kriteri gerçekleşene kadar

2.7.2.2 Dahili Bellek Kullanan Yöntemler

Eğer bir takviye öğrenme etmeni bir dahili bellekle donatılırsa, o anki gözlem ve tarihçeye bakarak, eksiksiz bilgi durumunu yaklaşık olarak elde etmesi mümkün olabilmektedir. Ardından bu bilgi, takviye öğrenmedeki "durum" bilgisinin yerine etkin bir şekilde kullanılabilmektedir. Bu sayede etmen, sürekli olarak yaklaştırma işlevini ve değer işlevini güncelleyerek, belleksiz yöntemlerle karşılaştırıldığında çok daha iyi hareket tarzlarını daha süratli bir şekilde öğrenebilmektedir. Şurası açıktır ki, bu iyileşme algısal olarak çakışan durumların birbirinden önemli oranda ayırt edilebilir hale gelmesi ile mümkün olmaktadır. Araştırmacılar bu fikrin çeşitli farklı uygulamalarını denemişlerdir.

Sabit boyutlu tarihçe (İng. finite size history) kullanmak, dahili bilgi durumunu iyileştirmek için izlenebilecek belki de en basit yoldur. Burada temel motivasyon, bilgi durumu olarak son n adımdan oluşan bir tarihçe kullanmaktır. Lin ve Mitchell (1992) çalışmalarında, yakın geçmişteki gözlemler ve eylemlerden oluşan sabit boyutlu bir tarihçe penceresi kullandıkları yöntemlerinin

sonuçlarını paylaşmışlardır. Bu yöntemde, etmen tasarımcısının, çevreyi doğru şekilde modellemeye yetecek hafıza miktarını, öğrenme başlamadan önce analiz ederek belirlemesi gerekmektedir.

Peshkin vd. (1999) tarafından tanıtılan *bellek ikilleri* (İng. memory bits) yöntemi, *dolaylı iletişim* (İng. stigmergy: literatürde bu kavram, bir böceğin eyleminin, başka bir böceğin eylemine dolaylı uyarıcı etkisi için kullanılır) kavramından esinlenilerek, harici "bellek ikilleri" kullanan bellek tabanlı bir bir yöntemdir. Bu yöntemin eksiksiz bilgi durumunu yaklaşık olarak hesaplama kabiliyeti sınırlı olup, eylem uzayında harici belleğin işlenmesi amacıyla özel bir yapılanma gerektirmektedir.

Uzun kısa-süreli bellek (İng. long short-term memory), Hochreiter ve Schmidhuber (1997) tarafından Geri-beslemeli-Q (İng. recurrent-Q) algoritmasının uzantısı olarak sunulan, yön-türevi (İng. gradient) tabanlı bir yaklaşımdır. Uzun kısa-süreli bellek, problem uzayının ilgili özelliklerini temsil etmek üzere tasarlanmış özel bir Geri-beslemeli Sinir Ağı (İng. recurrent neural network) kullanır. Geri-beslemeli-Q algoritması, sinir ağını aynı zamanda algısal çakışma yaşanan durumları ayırt etmek amacıyla da kullanır. Algoritma, bellek girdi ve çıktılarını açıp kapatmayı öğrenmek amacıyla karmaşık yapay sinir hücrelerinden yararlanır.

Değişken boyutlu tarihçe (İng. variable length history) yöntemleri, ihtiyaç halinde genişletilebilen dahili bellek yapıları oluşturmaya odaklanır. McCallum (1996) etmenin çevre ile geçmişteki etkileşiminin eylem-ödül-gözlem dizisi formunda tutulduğu, *örnek temelli* (İng. instance based) yöntemler sunmuştur. En yakın dizi belleği (İng. nearest sequence memory), yararlı sonek belleği (İng. utile suffix memory), ve U-Ağacı (İng. U-Tree) algoritmaları, aynı örnek tabanlı takviye öğrenme fikrinin farklı uygulamalarıdır.

Yararlı Sonek Belleği Agoritması Yararlı sonek belleği (YSB) algoritması (McCallum, 1996) 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.

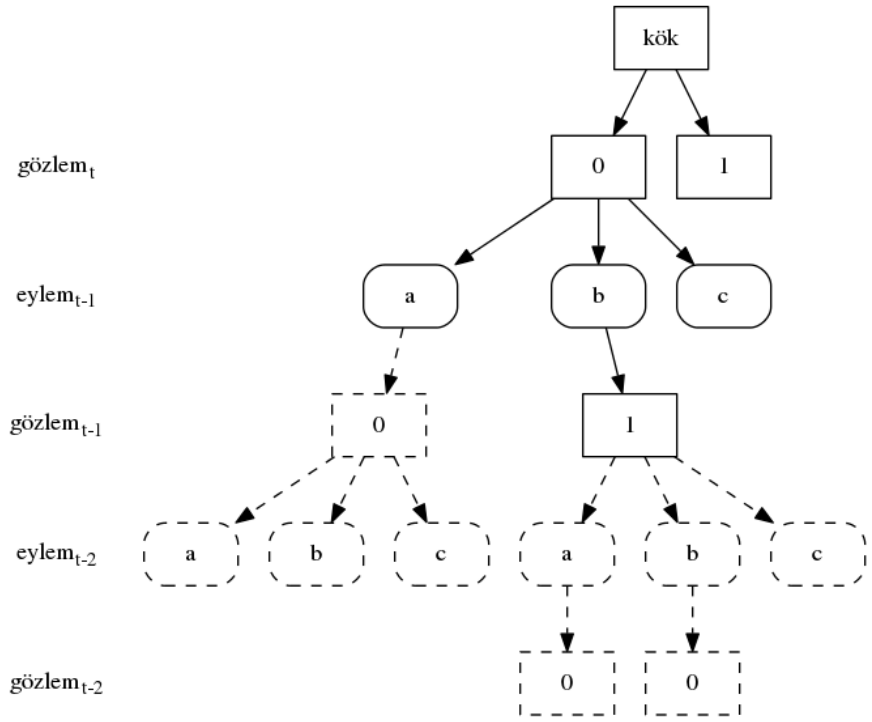
YSB algoritmasının merkezinde *sonek ağacı* (İng. suffix tree) veri yapısı vardır. Sonek ağacı, ham deneyimlerden türetilmiş kısa tarihçelerden oluşan bir depo olarak kullanılır. Bu kısa tarihçelere *örnek* (İng. instance) adı verilir. Ağacın derinlik kısıtlaması yoktur. Derinlik, öğrenme

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 YSB ağaç veri yapısı Şekil 2.10 ile gösterilmiştir. Şekildeki ağaç, t anındaki gözlemi ayırt etmek için faydalı olabileceği tespit edilmiş kısa vadeli ($t-2$ zamanına kadarki) geçmiş örneklerinin deposudur.

Aslında, YSB 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. GDA yöntemi problem uzayının farklı boyutlarını ayırt edecek şekilde tasarlanmadığından, U-Ağacı algoritması bu çalışmanın kapsamı dışındadır.



Şekil 2.10. Temsili YSB sonek ağacı veri yapısı. Gözlemler sayılarla, eylemler harflerle gösterilmiştir. Kesik çizgili çerçevesi olan düğümler saçak düğümlerdir.

2.8 Bölümlenmiş Markov Karar Süreçlerinde Takviye Öğrenme

BMKS'de geçiş ve ödül fonksiyonları bilindiğinde, klasik MKS'deki dinamik programlama teknikleri gibi, yapılandırılmış değer yineleme (YDY, İng. structured value iteration) ve yapılandırılmış hareket tarzı yineleme (YHTY, İng. structured policy iteration) (Boutilier, 2000) yapılandırılmış dinamik programlama (YDP) teknikleri optimum hareket tarzına ulaşabilmek için kullanılabilir. Bu metotlar, hesaplamalar sırasında ağaç yapılarına ekleme, birleştirme ve basitleştirme teknikleri uygularlar. SDYNA (Degris, 2006) ise problem yapısı tam olarak bilinmediğinde kullanılan bir bölümlenmiş takviye öğrenme (BTÖ, İng. factored reinforcement learning) algoritmasıdır. Bu algoritma "uygulama", "öğrenme" ve "planlama" olmak üzere 3 fazdan oluşmaktadır. Ödül ve geçiş fonksiyonları yinelemeli bir şekilde her bölümde oluşturulup bunlara bağlı olarak hareket tarzı ve durum değerleri fonksiyonları güncellenir. Sonrasında da uygulanacak olan eylem seçilir. SPITI, eylem seçiminde ϵ-Hırslı (İng. ϵ-Greedy), öğrenme kısmında artırımlı ağaç tümevarımı (İng. incremental tree induction) (Utgoff, 1996), ve planlama fazında YDY metotlarını kullanan bir SDYNA örneğidir.

YDP ve BTÖ tekniklerine ek olarak, bölümlenmiş çevrelerde zamansal soyutlamaları öğrenmek için bazı yaklaşımlar mevcuttur. Bu yaklaşımlar önceden bulunmuş alt-çözümleri kullanarak problemlerdeki zaman ve hafıza karmaşıklıklarını azaltmayı hedeflerler. HEXQ (Hengst, 2002) ve TeXDYNA (Kozlova, 2010), zamansal soyutlamaları, problemi hiyerarşik alt gruplara ayırarak bulmayı amaçlayan algoritmalara örnektir.

2.9 Kısmi Gözlemlenebilirlik Durumunda Takviye Öğrenme için Zamansal Soyutlama

KGMKS kapsamında, takviye öğrenme için zamansal soyutlama çalışmaları sınırlıdır. İlgili çalışmaların çoğu planlama literatürü kaynaklıdır (Charlin vd., 2007; He vd., 2010; Pineau ve Thrun, 2002; Theocharous vd., 2001), ve takviye öğrenme çerçevesinin dışında kalmaktadır.

Takip eden iki alt bölümde, kısmi gözlemlenebilirlik durumunda takviye öğrenme için az sayıdaki zamansal soyutlama çalışması özetlenmiştir.

2.9.1 Model Tabanlı Yöntemler

Mevcut MKS çatısı ile inanç durumu yaklaştırma yöntemlerini birleştiren umut verici bir çalışma Theocharous ve Kaelbling (2004) tarafından yapılmıştır. Çalışmada, inanç uzayında oluşturulan ızgara noktaları üzerinden model tabanlı bir takviye öğrenme algoritması önerilmiştir. Algoritma, makro-eylemler ve Q-değerlerinin Monte Carlo güncellemelerini temel alır. Yöntem, büyük ölçekli bir robot gezinti problemine uygulanarak makro-eylemlerin KGMKS modeli için avantajları sergilenmiştir. Deney sonuçları, makro-eylemler sayesinde, etmenin inanç uzayının çok küçük bir kısmını deneyimlemekle yetindiğini, sadece ilkel eylemler kullanıldığında (makro-eylemler olmadığında) gereken deneyimin ise görece çok daha fazla olduğunu göstermiştir. Ayrıca, öğrenme hızı da artmıştır, çünkü etmen artık daha uzak geleceğe bakabilmekte, ve inanç noktalarının değerlerini daha hızlı yayabilmektedir. Son olarak, iyi tasarlanmış makro eylemler (örneğin, etmeni yüksek entropiye sahip bir inanç durumundan düşük entropiye sahip bir inanç durumuna taşıyabilecek makro eylemler) etmenin bilgi edinmesini kolaylaştırmaktadır. Bu yöntemin eksik yanı ise, makro-eylem mekanizmasının zekice tasarlanmasını gerektirmesidir. Yani, yöntem takviye öğrenme sırasında makro-eylemleri otomatik olarak öğrenememektedir.

Dung vd. (2007), kısmi gözlemlenebilir problemler için takviye öğrenmede, yararlı alt-hedeflerin otomatik olarak keşfedilip kullanılmasını sağlayan bir yol sunmuşlardır. Çalışmalarında, bir

durum, eğer başarılı rotalar üzerinde sıklıkla ziyaret ediliyorsa, alt-hedef olarak değerlendirilir. Alt-hedefler yaratıldıktan sonra onlara ulaşmak için geri-beslemeli sinir ağı kullanılır. Eğitilmiş sinir ağları, sonrasında ana sinir ağına *uzman* (İng. expert) olarak entegre edilir. Bu yöntem, alt-hedef tabanlı otomatik zamansal soyutlama yöntemi olarak değerlendirilebilse de, zamansal soyutlama işlevi alt-hedef tespiti aşamasına kadar tanımlanmıştır, yani tercih (veya makro-eylem) inşa ve kullanım mekanizmalarından yoksundur.

2.9.2 Modelden Bağımsız Yöntemler

Belleksiz yöntemlerin önemli temsilcilerinden biri *HQ-Öğrenme* algoritmasıdır (Wiering ve Schmidhuber, 1997). HQ-Öğrenme, Q-Öğrenmenin alt-hedef yönelimli şekilde, hiyerarşik olarak genişletilmiş halidir. HQ-Öğrenmede KGMKS modeli sabit sayıda tepkisel (belleksiz) hareket tarzına ayrıştırılır, ve her bir alt-hedef ikmalinin tamamen gözlemlenebilir olduğu varsayılır. Göreceli olarak çok sayıda duruma sahip kısmi gözlemlenebilir labirent problemlerinde iyi sonuçlar elde edilmiştir. Yöntemdeki sorunlardan biri, gürültü durumunda, alt hareket tarzları arasında kontrolün aktarımının yönetiminde sorun yaşanmasıdır. Bir diğer önemli sorun ise, önerilen mimarinin özünde bir çoklu-etmen (İng. multi-agent) tasarımı gerektirmesidir. Çoklu-etmen tasarımının tek etmene uyarlanabileceği iddia edilebilirse de, yöntem problemi parçalamak için etmen sayısı bilgisine ihtiyaç duymaktadır, ve bu sayının öğrenme öncesinde sağlanmalıdır. Bu nedenle, öğrenmede sorun yaşanmaması için problemin doğasına uygun bir etmen sayısı tahmini yapılması gerekmektedir.

Yoshikawa ve Kurihara (2006) Macro/SARSA(λ) adını verdikleri bir yöntemle belleksiz takviye öğrenmede zamansal soyutlamaların otomatik olarak çıkarılmasını denemişlerdir. Macro/SARSA(λ), klasik SARSA(λ) ile kısa dönem deneyimlerden makro eylemler çıkarmayı hedefleyen basit bir rutinin bir araya gelmesinden oluşmuştur. Ancak, bu yöntem, algısal çakışmaları tespit edecek bir mekanizmaya sahip olmadığı için, ilgisiz makro eylemler üretebilmektedir.

Bu çalışmanın yapıldığı dönem itibarı ile, kısmi gözlemlenebilir problemler için bellek tabanlı takviye öğrenme algoritmalarını iyileştirebilecek zamansal soyutlama çalışmaları henüz literatürde mevcut değildir.

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

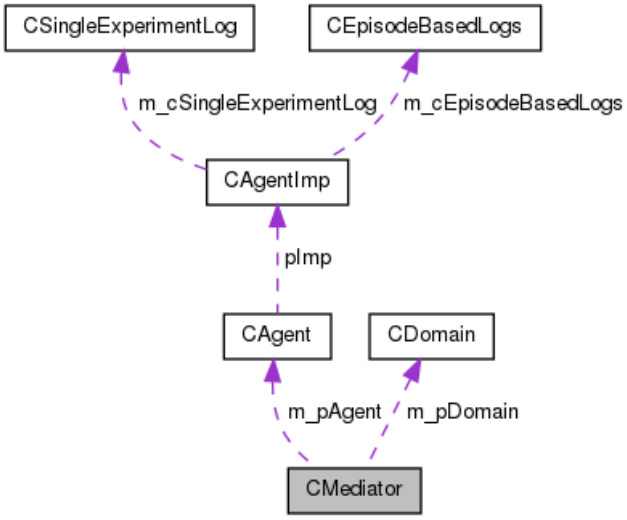
Proje kapsamında oluşturulacak çözümlerin en az ek eforla test edilebilmesi amacıyla bir yazılım geliştirme altyapısı tasarlanarak gerçeklenmiştir. Yazılım, tek parça (İng. monolythic) ve tek iş parçacığı tabanlı (İng. single threaded) olarak tasarlanmıştır.

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

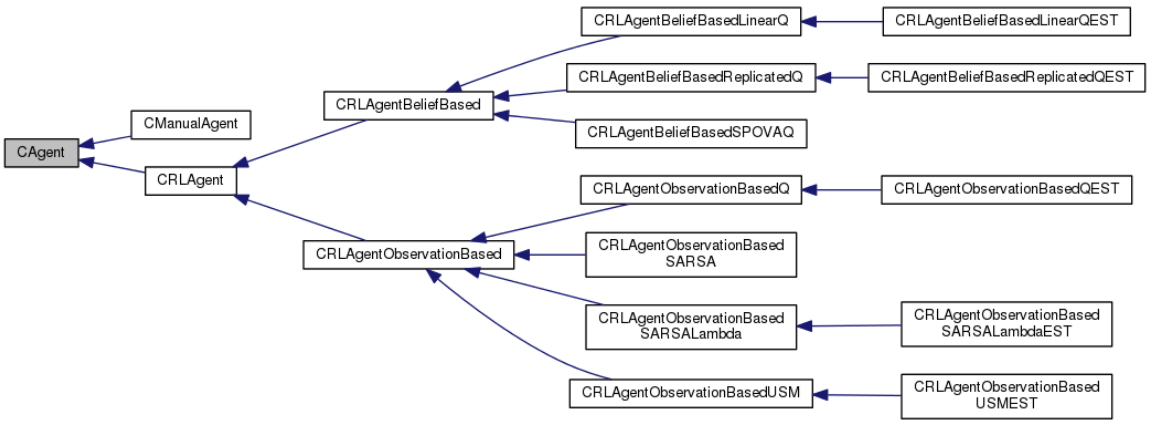


Şekil 3.1. Altyapının temelinde yer alan bağlantı sınıfına (CMediator) ait işbirliği şeması

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 3.5'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.

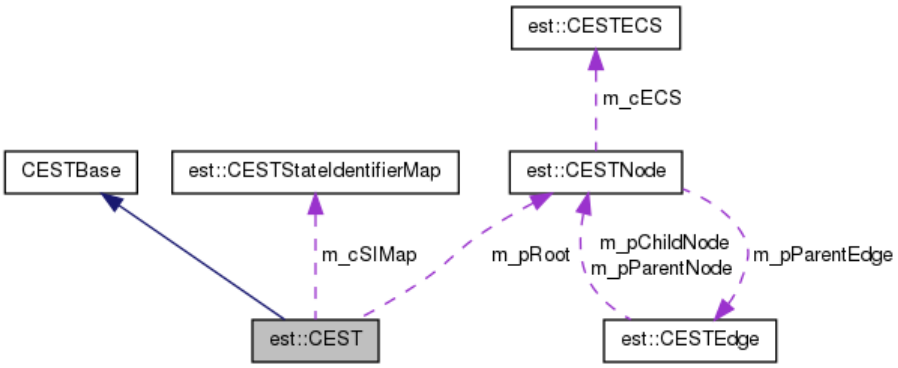
Projede zamansal soyutlama için tercih edilen GDA yöntemi, orijinal şekliyle gerçeklenmiştir. Bu gerçekleştirimi özetleyen işbirliği şeması Şekil 3.3 ile gösterilmiştir.

GDA yapısının bu şekilde tasarlanmış olması, aynı zamanda yeni GDA denemelerinin de daha kolay yapılabilmesine olanak sağlamaktadır. Her türlü GDA veri yapısı için ortak olması muhtemel metot ve parametreler CESTBase sınıfında toplanmış olup, proje kapsamında geliştirilen her GDA varyantı, bu sınıf miras alınarak daha kolay tasarlanabilmiştir.

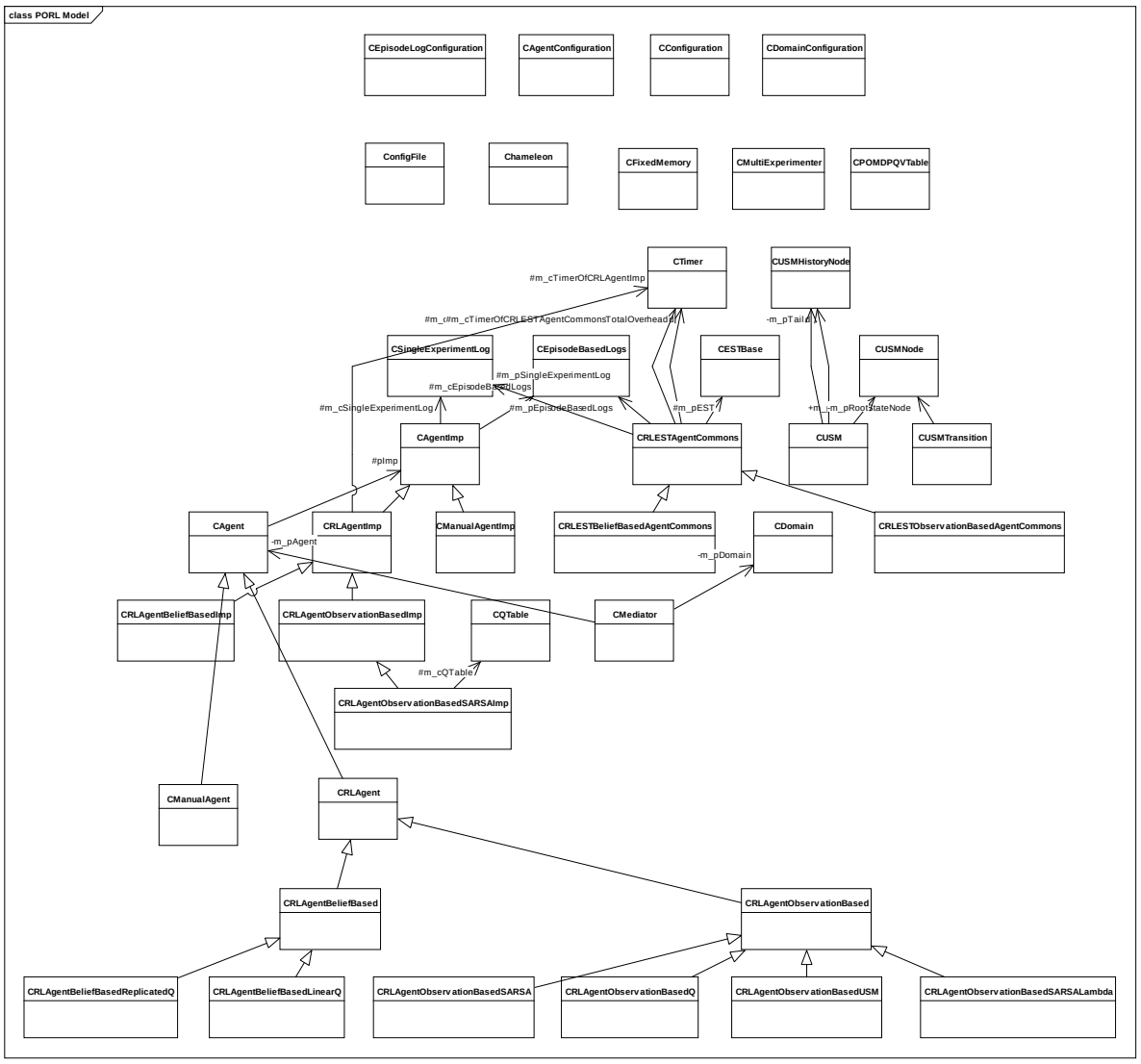
Altyapıda, problem çeşitliliği, Anthony R. Cassandra'nın kullandığı dosya yapısı[4] kullanılarak sağlanmıştır. Bu amaçla, ilgili dosya formatını ayrıştırarak (İng. parse) iç veri yapılarına aktaran bir kütüphane yazılmıştır. Böylece, Cassandra'nın formatı kullanılarak tasarlanmış herhangi bir problemin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

Yazılım altyapısının tamamımın genel olarak yapısını gösteren basitleştirilmiş sınıf şeması Şekil 3.4'te gösterilmiştir.

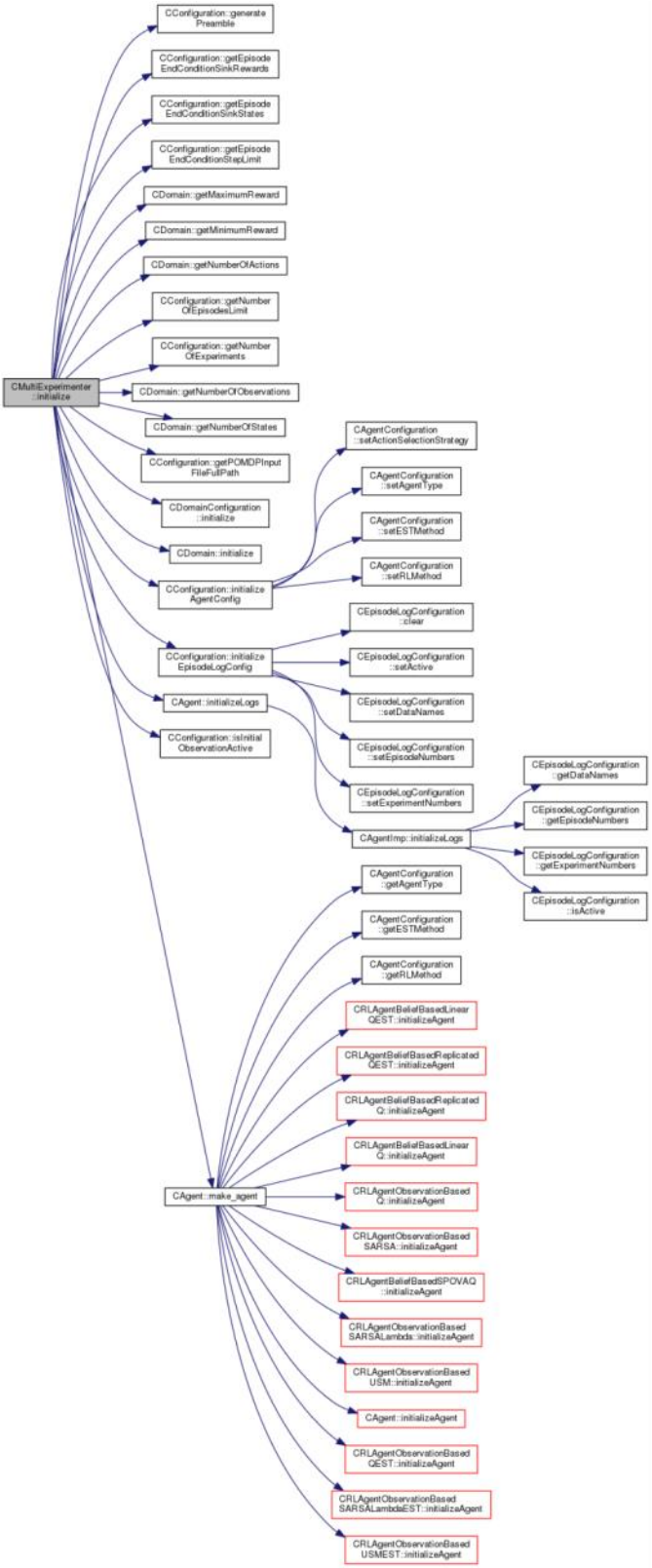
[4] http://www.pomdp.org/



Şekil 3.3. GDA için basitleştirilmiş işbirliği şeması



Şekil 3.4. Deney ortamı için oluşturulan yazılımın basitleştirilmiş sınıf şeması



Şekil 3.5. CMultiExperimenter sınıfının initialize işlevi için çağrı şeması

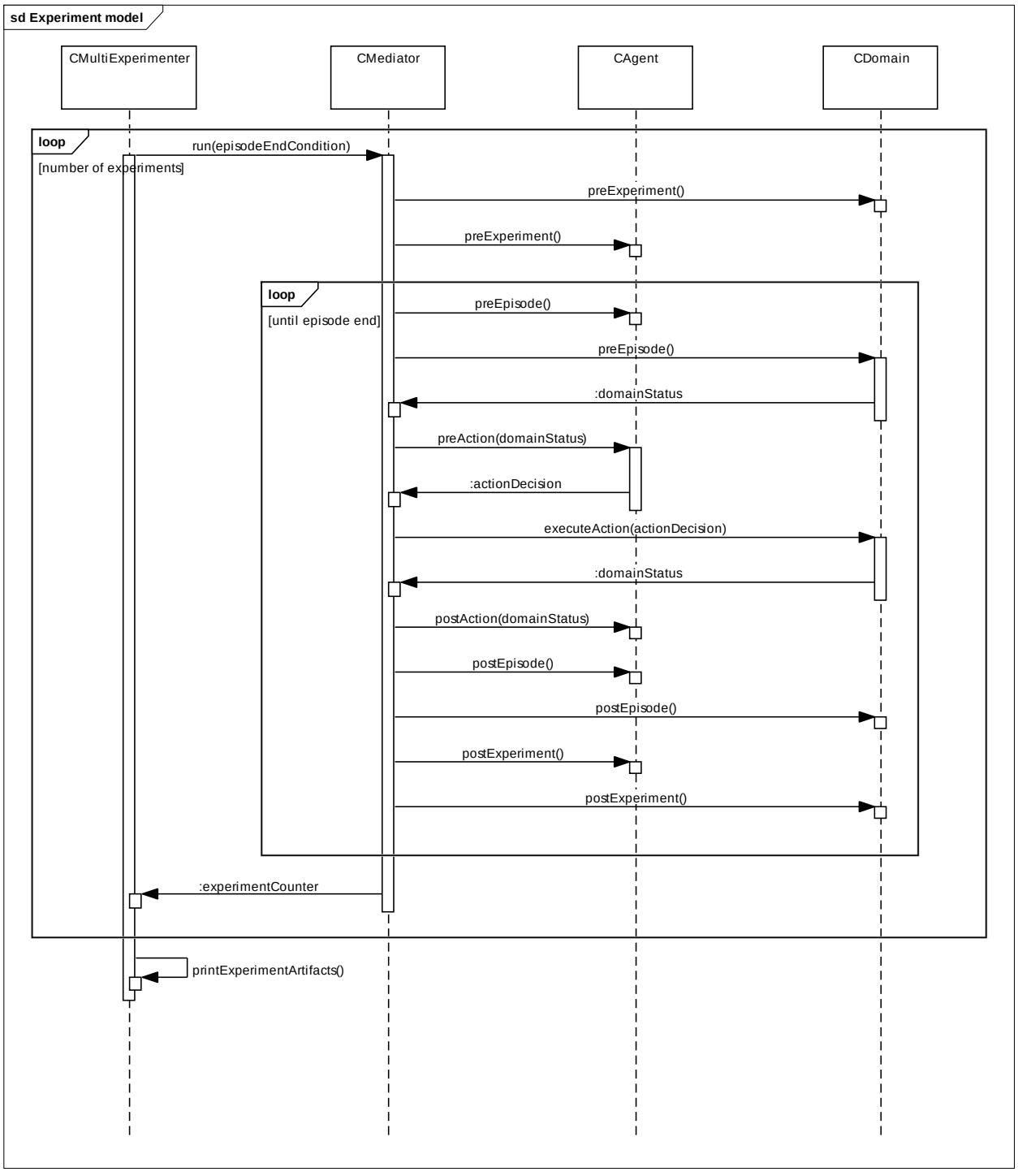
3.3 Mukayese için Kullanılan Takviye Öğrenme Yöntemlerin Geliştirilmesi

Proje kapsamında inanç tabanlı yöntemleri temsilen Tekrarlanan-Q ve Doğrusal-Q öğrenme algoritmaları, reaktif hareket tarzlarını bulan yöntemleri temsilen SARSA(λ), bellek tabanlı yöntemleri temsilen YSB öğrenme algoritmaları tercih edilmiştir. Bu algoritmaların gerçekleştirimleri, oluşturulan yazılım altyapısı kullanılarak tamamlanmıştır.

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 3.6'da gösterilmiştir.



Şekil 3.6. Takviye öğrenme yazılım altyapısının basitleştirilmiş akış şeması

4 MODEL TABANLI KISMİ GÖZLEMLENEBİLİR TAKVİYE ÖĞRENMENİN HIZLANDIRILMASI

Bu bölümde, kısmi gözlemlenebilir ortamlarda kullanılan model tabanlı takviye öğrenme algoritması için dolaysız otomatik zamansal soyutlama yöntemi olarak, bir inanç tabanlı GDA yöntemi önerilmektedir (Çilden ve Polat, 2015).

Altta yatan model tabanlı takviye öğrenme algoritması olarak seçilen yöntemler, inanç durumlarının mutlak olarak hesaplandığı, değer işlevinin ise yaklaşıklaşltırma ile hesaplandığı yöntemlerdir. Bu yöntemler Bölüm 2.7.1.2'de özetlenmiştir. Bu kapsamda, GDA'da yapılacak basit bir değişiklikle, s durum bilgileri yerine b inanç bilgileri yerleştirilerek ilgili tüm prosedürler de uygun şekilde değiştirilebilir. Ancak inanç durum uzayı sonsuz olduğundan, bu basit çözümün işe yaraması maalesef pratikte mümkün değildir.

Ayrıştırma (İng. discretization) yöntemleri sayesinde, sonsuz büyüklükteki inanç durum uzayının sonlu büyüklükteki yaklaşıklaştırmalarını elde etmek mümkün olabilmektedir. Gerçek inanç durumları yerine inanç durumu ayrıştırmaları kullanılarak, model tabanlı bir takviye öğrenme algoritması üzerinde GDA uygulanması pratik olarak mümkün hale gelmektedir.

4.1 İnanç Ayrıştırma Yöntemleri

İnanç tabanlı GDA yöntemini tanıtmadan önce, yeni GDA yaklaşımımızda kullanılabilecek farklı inanç durum ayrıştırma yöntemleri, takip eden üç alt bölümde sunulmuştur. Bölüm 4.2'de inanç tabanlı yeni GDA yöntemi anlatılmış, takip eden bölümde ise ilgili deneyler ve elde edilen sonuçlar sunulmuştur.

4.1.1 Sabit Çözünürlüklü Düzenli Izgara Ayrıştırması

Izgara tabanlı ayrıştırma, KGMKS'lerde boyut sorununun üstesinden gelmenin etkin yollarından biridir. Bu yöntem ailesi, inanç uzayından, KGMKS'nin çekirdek durumları üzerindeki tüm olasılık dağılımlarını içeren bir durum uzayı ile tanımlanan, tam gözlemlenebilir bir MKS'ye bir dönüşüm tanımlar. n çekirdek duruma sahip bir KGMKS için, dönüştürülen durum uzayı n-boyutlu simpleks (İng. simplex), veya *inanç simpleksi* olarak adlandırılır (Zhou ve Hansen, 2001). *Sabit çözünürlüklü düzenli ızgara* yöntemi (Lovejoy, 1991) bu kategorideki en popüler yöntemlerden

biridir. Bu yöntemde ızgara noktaları düzenli bir örüntü ile aralıklandırılarak, inanç simpleksi eşit boyutlu alt simplekslere bölünürler.

Algoritma 5 ile tanımlanan $D_{IZGARA}(b, M)$ işlevi orijinalinin çok az değiştirilmiş versiyonudur. Bu versiyonda, yöntemin Lovejoy'un tanımladığı orijinal versiyonundaki son iki adım kullanılmamaktadır. Algoritma 5, M çözünürlüğündeki bir düzenli ızgarada b inanç durumunun ayrıştırmasına karşılık gelen ızgara koordinatını temsil eden $|S|$ boyutlu bir tamsayı vektörü hesaplayarak döner. Orijinal algoritmanın barisentrik koordinat hesaplayan son kısmı burada kullanılmamıştır.

Algoritmanın en zayıf yönü ise, M büyüdükçe ızgaranın boyutunun üstel (İng. exponential) olarak artmasıdır.

Algoritma 5 $D_{IZGARA}(b, M)$

Require: S üzerinde tanımlı bir inanç durumu b

Require: ızgara çözünürlüğü $M \in \mathbb{N}^+$

Ensure: b inanç durumunu temsil eden bir ızgara koordinatı

1: $1 \leq s \leq |S|$ için $x(s) = M \sum_{i=s}^{|S|} b(i)$ olacak şekilde, x adında bir $|S|$-vektörü oluştur

2: v, tüm $s \in S$ değerleri için $v(s) < x(s)$ eşitsizliğini sağlayan en büyük tamsayılarla oluşturulmuş $|S|$-vektörü olsun.

3: d, tüm $s \in S$ değerleri için $d(s) = x(s) - v(s)$ ile hesaplanan $|S|$-vektörü olsun.

4: p, d vektörünün içeriğinin, azalan şekilde, yani $d\big(p(1)\big) \geq d\big(p(2)\big) \geq \cdots \geq d\big(p(|S|)\big)$ eşitsizliğine uyacak şekilde sıralandığı, $1,2,\ldots,|S|$ tamsayılarının bir permütasyonunu içeren bir $|S|$-vektörü olsun.

5: x noktasını içeren alt-simpleksin $\{v^i : 1 \leq s \leq |S|\}$ köşe noktalarını aşağıdaki yöntemle hesapla:

$$v^1(s) = v(s), \text{for } 1 \leq s \leq |S|$$

$$v^{i+1}(s) = \begin{cases} v^i(s) + 1, \text{eğer } s = p(i)\ ise \\ v^i(s), \text{aksi durumda} \end{cases}$$

6: return v.

4.1.2 Durum Öncelik Sırası Ayrıştırması

En olası durum (EOD) yaklaştırması (Cassandra vd., 1996), belki de literatürdeki en sezgisel ve basit inanç ayrıştırma yöntemidir. Çilden ve Polat (2012), çalışmalarında, EOD yaklaşımını, istatistiksel sıralama (İng. statistical ranking) fikrinden esinlenerek, durumlar arasında öncelik sırası gözeten bir yöntemle genelleştirmişlerdir.

Bu çalışmada, Algoritma 6'da verilen $D_{SIRALAMA}(b, p_{eşik})$ işlevi ile, orijinal sıralama tabanlı algoritmamızın bir adım daha genelleştirilmiş versiyonu önerilmiştir. Yeni yöntemde, olasılık değerlerine göre ters sıralanmış bir önceliklendirme vektörü, tanımlı bir olasılık eşiğine erişilene kadar inşa edilir. Oluşturulan vektör, inanç durumunun ayrıştırılmış bir yaklaşık değeri olarak kullanılabilir.

Örneğin, bir b inanç durumunun $0.3, 0.15, 0.0, 0.15, 0.4$ listesi ile temsil edildiğini, bu listedeki değerlerin sırasıyla s_1, s_2, s_3, s_4, ve s_5 durumlarının olasılıkları olduğunu varsayalım. s_2 ve s_4 değerleri aynı olduğundan, sıralama kriteri olarak durumların indisleri kullanılarak çakışma çözülebilir ($2 < 4$). $D_{SIRALAMA}$ işlevi, $p_{eşik}$ parametresinin 0.0, 0.5 and 1.0 değerleri için sırasıyla $\langle s_5 \rangle$, $\langle s_5, s_1 \rangle$, ve $\langle s_5, s_1, s_2, s_4 \rangle$ yaklaşık durum vektörlerini üretir.

Bu aşamada, $p_{eşik} = 0.0$ için $D_{SIRALAMA}$ algoritmasının EOD yaklaştırması ile aynı olduğunu belirtmek gerekir. Bu parametre değeri ile çalıştırılan algoritma b inanç durumu için en yüksek olasılık değerine sahip dünya durumunu hesaplar:

$$EOD(b) = \arg\max_{s \in S} b(s) \tag{4.1}$$

$$= D_{SIRALAMA}(b, 0.0) \tag{4.2}$$

$D_{SIRALAMA}$ algoritmasının bir diğer özelliği de, imkansız durumları (yani $\{s | s \in S \wedge b(s) = 0.0\}$ durumlarını) yok saymasıdır. Her ne kadar bir inanç durumu için imkansız durumların da bir anlamı olsa da, sıralama tabanlı bir ayrıştırma yöntemi için bu bilginin gereksiz olduğu açıktır.

Algoritma 6 $D_{SIRALAMA}(b, p_{eşik})$

Require: S üzerinde tanımlı bir inanç durumu b

Require: birikimli (İng. cumulative) olasılık için bir eşik değeri, $p_{eşik} \in [0.0,1.0]$

Ensure: b inanç durumunun sıralama ayrıştırmasını temsil bir durum isimleri vektörü

1: $b_{eşleme}$, $s_i \in S, 1 < i < |S|$ için $b_{eşleme}[s_i] = b(s_i)$ eşlemesinin sağlandığı bir eşleme (İng. hash) tablosu olsun

2: v, $s_i \in S$ olacak şekilde bir durum isimleri vektörü olsun

3: $v \leftarrow \langle \quad \rangle$ ▷ boş vektör

4: $p_{toplam} \leftarrow 0.0$

5: **repeat**

6: $s_{max} \leftarrow \arg\max(b_{eşleme})$ ▷ eşit değerli isimler, alfabetik karşılaştırma gibi öngörülebilir bir yöntemle çözülmelidir

7: s_{max} değerini v vektörünün sonuna ekle

8: $p_{toplam} \leftarrow p_{toplam} + b_{toplam}[s_{max}]$

9: $b_{eşleme}$ tablosundan s_{max} ile indislenen kaydı sil

10: **until** $p_{toplam} \geq p_{eşik}$

11: return v

4.1.3 Eklentili Durum Öncelik Sırası Ayrıştırması

Roy (2003), inanç durumunun ayrıştırılması amacıyla eklentili MKS (İng. augmented Markov decision process) adını verdiği bir yapı kullanmıştır. Eklentili MKS yaklaşımı, çoğu KGMKS problemi için sistemdeki belirsizliğin alana özel ve yerel olduğu gözlemi üzerine kurgulanmıştır. Bu varsayım altında, bir inanç durumunu, EOD ve entropiden oluşan ikili ile özetlemek genellikle mümkün olabilmektedir. İnanç durumunun EOD durumu algısal çakışmadan mustarip olduğunda, entropi değeri ile temsil edilen belirsizlik düzeyi, bahsi geçen inanç durumunu diğer inanç durumlarından ayırt edebilmektedir.

b inanç durumunun entropi değeri alağıdaki denklemlerle hesaplanır:

$$H(b) = -\sum_{i=1}^{|S|} b(s_i) \log_2 b(s_i) \tag{4.3}$$

$$\overline{H}(b) = \frac{H(b)}{H(b_u)} \tag{4.4}$$

Bu denklemlerde b_u tüm durumlar üzerindeki düzgün dağılım (İng. uniform distribution), $\overline{H}(b)$ ise $[0,1]$ aralığına düşmek amacıyla normalleştirilmiş entropi değeridir. EOD ve normalleştirilmiş entropi bilgileri birleştirildiğinde, b inanç durumunun $\tilde{b}$ ile gösterilen düşük boyutlu gösterimi, aşağıdaki şekilde olur:

$$\tilde{b} = \langle MLS(b); \overline{H}(b) \rangle \tag{4.5}$$

Doğal olarak, entropi değeri $\tilde{b}$ ikilisini sürekli (İng. continuous) bir değişken haline getirmektedir. Roy (2003), eklentili MKS uzayını sonlu büyüklüğe çekmek amacıyla entropi değerini sabit sayıda hücreye ayırır.

$\tilde{b}$'nin aynı zamanda $\tilde{b} = \langle D_{SIRALAMA}(b, 0.0); \overline{H}(b) \rangle$ olarak hesaplanabilmesi, orijinal eklentili MKS yaklaşımının, Bölüm 0'de anlatılan durum öncelik sıralaması ayrıştırma mekanizması kullanılarak genelleştirilebileceğini fikrine yol açmaktadır. Algoritma 7'de, eklentili MKS yaklaştırma yönteminin, $D_{SIRALAMA}$ ile entropi ayrıştırmasını birleştiren genel versiyonu sunulmuştur. Bu genelleme kapsamında, eklentili MKS yöntemi, $D_{EKLENTI}$ algoritmasının $p_{eşik} = 0$ alınan özel bir halidir.

$D_{EKLENTI}$ yöntemi, entropi eklentisinin ayırt edici etkisinin yaralı olabileceği problemlerde, $D_{SIRALAMA}$ yönteminin çözünürlüğünü arttırmanın etkin bir yoludur.

Algoritma 7 $D_{EKLENTI}(b, p_{eşik}, n)$

Require: S üzerinde tanımlı bir inanç durumu b

Require: birikimli (İng. cumulative) olasılık için bir eşik değeri, $p_{eşik} \in [0.0, 1.0]$

Require: normalleştirilmiş entropi ayrıştırması için üleşim (İng. partition) sayısı, $n \in \mathbb{N}^+$

Ensure: bir durum isim vektörü ve bir pozitif tamsayıdan oluşan, b inanç durumunun ayrıştırmasını temsil eden ikili

1: v, $s_i \in S$ olacak şekilde bir durum isimleri vektörü olsun

2: $v \leftarrow D_{SIRALAMA}(b, p_{eşik})$ ▷ Algoritma 6 kullanılarak

3: $H_d(b) \leftarrow \lfloor n\overline{H}(b) \rfloor$ ▷ (4.3) ve (4.4) denklemleri kullanılarak

11: return $\langle v, H_d(b) \rangle$

4.2 İnanç Tabanlı Genişletilmiş Dizi Ağacı Algoritması

İnanç ayrıştırma yöntemleri sayesinde, model tabanlı Doğrusal Q-Öğrenme ve Tekrarlanan Q-Öğrenme takviye öğrenme algoritmaları için GDA yöntemini yeniden tasarlamak mümkün olabilmektedir.

D, inanç durum uzayı üzerinde tanımlı bir ayrıştırma işlevi olsun. Bu bölümde, GDA yönteminin yapıtaşları, inanç durumu ayrıştırma kullanarak yeniden tanımlanmaktadır. İlk olarak, tarihçe tanımı (Tanım 2.6) D işlevi üzerinden yeniden tanımlanır:

Tanım 4.1. Bir D-tarihçe, aşağıdaki şekilde tanımlanır:

$$h_{D_{i\tau}} = D(b_t), a_t, r_{t+1}, D(b_{t+1}), a_{t+1}, \dots, r_\tau, D(b_\tau)$$

Bu tanımda etmen tarafından deneyimlenen durumlar, eylemler ve ödüller, t zamanından τ zamanına kadar ardışık olarak sıralanır.

■

Ardından, GDA tanımı (Tanım 2.7), ayrıştırılmış inanç durumlarını kullanacak şekilde güncellenir.

Tanım 4.2. D-genişletilmiş dizi ağacı (D-GDA), düğümler kümesi N ve ayrıtlar kümesi E bileşenlerinden oluşur. Her düğüm, o düğüme ulaşma yolunda deneyimlenmiş bir eylem dizisini temsil eder ve bu eylem dizisinin ağaçta bir tekrarı yoktur. Kök düğüm $\emptyset$ sembolü ile gösterilir ve boş eylem kümesini temsil eder. Eğer q düğümünün eylem dizisi, a eyleminin p düğümü ile temsil edilen eylem dizisine eklenmesi ile elde edilebiliyorsa, bu durumda p düğümü q düğümüne $\langle a, \psi \rangle$ etiketli bir ayrıt ile bağlıdır; ve bu bağ $\langle p, q, \langle a, \psi \rangle \rangle$ ile gösterilir. ψ, q düğümünün temsil ettiği eylem dizisinin hangi sıklıkla uygulandığını anlatan ayrıt nitelik (İng. eligibility) değeridir. Dahası, -D inanç durum uzayı üzerinde bir ayrıştırma işlevi olmak üzere, $D(b_i)$'nin, kısaca d_i olarak gösterildiğini varsayarsak-, q düğümü, eğer gözlemlenen mevcut inanç durumu ayrıştırması $\{d_1, \dots, d_k\}$ kümesindeki yaklaşık durumlardan biri ise a eyleminin p düğümünde seçilebileceğini anlatan $\langle d_1, \xi_{d_1}, R_{d_1} \rangle, \dots, \langle d_k, \xi_{d_k}, R_{d_k} \rangle$ yapısını barındırır. Bu yapıya q düğümünün devam kümesi adı verilir ve $cont_q$ ile gösterilir. R_{d_i}, p düğümü ile temsili edilen eylem dizisinin uygulanmasının ardından, etmenin d_i ile temsil edilen ayrıştırılmış inanç durumunda a eylemini seçmesi ile elde etmesi beklenen toplam birikmiş ödüldür. ξ_{d_i}, q düğümündeki d_i ile temsil edilen ayrıştırılmış inanç durumunun nitelik değeridir, ve a eyleminin d_i yaklaşık inanç durumun gerçekte ne sıklıkta seçildiğini belirtir.

■

Son olarak, model tabanlı takviye öğrenme algoritması, yeni GDA mekanizmasını kullanacak şekilde değiştirilir. Yeni yöntem *İnanç tabanlı GDA*, veya i-GDA olarak adlandırılmış olup, ayrındıları Algoritma 8 ile verilmiştir. Bu yöntem, orijinal GDA yöntemini, altta yatan takviye öğrenme metodunu model tabanlı olacak şekilde (Doğrusal Q-Öğrenme veya Tekrarlanan Q-Öğrenme), GDA eklentisi bileşenlerini ise Tanım 4.1 ve Tanım 4.2 ile değiştirilmek suretiyle güncellemektedir.

D ayrıştırma işlevi, D-GDA için sonlu sayıda yaklaşık durum içeren bir çözüm kümesi sağladığı sürece, herhangi bir ayrıştırma metodunu temel alabilir. Bahsi geçen ayrıştırma yöntemi, altta yatan model tabanlı öğrenme prosedürü için geçerli değildir; yalnızca i-GDA için sonlu sayıda temsili yaklaşık durum üretebilmek amacıyla kullanılmaktadır.

Eylem seçim mekanizması (10-14 satırlar), doğrudan durum-eylem değeri yerine, satır 13'teki $Q_a(b) = q_a.b$ yaklaştırmasını kullanmaktadır. Ayrıca, eğer eylem seçim stratejisi kontrol akışını T_D veri yapısını takibe yönlendirirse (satır 12), mekanizmanın düğümlerde mevcut olan inanç yaklaştırmalarını karşılaştırması gerekmektedir.

Bir açıdan, i-GDA, GDA'nın daha genel bir takviye öğrenme ailesini kapsamasını sağlayan bir genelleştirme olduğu söylenebilir, çünkü model tabanlı takviye öğrenme, klasik takviye öğrenmenin genel bir halidir. Yani, Q-Öğrenmenin bölüm 2.3'te tanıtılan algoritmaların özel bir hali olması gibi, GDA da, $D = MLS$ iken i-GDA'nın özel bir halidir.

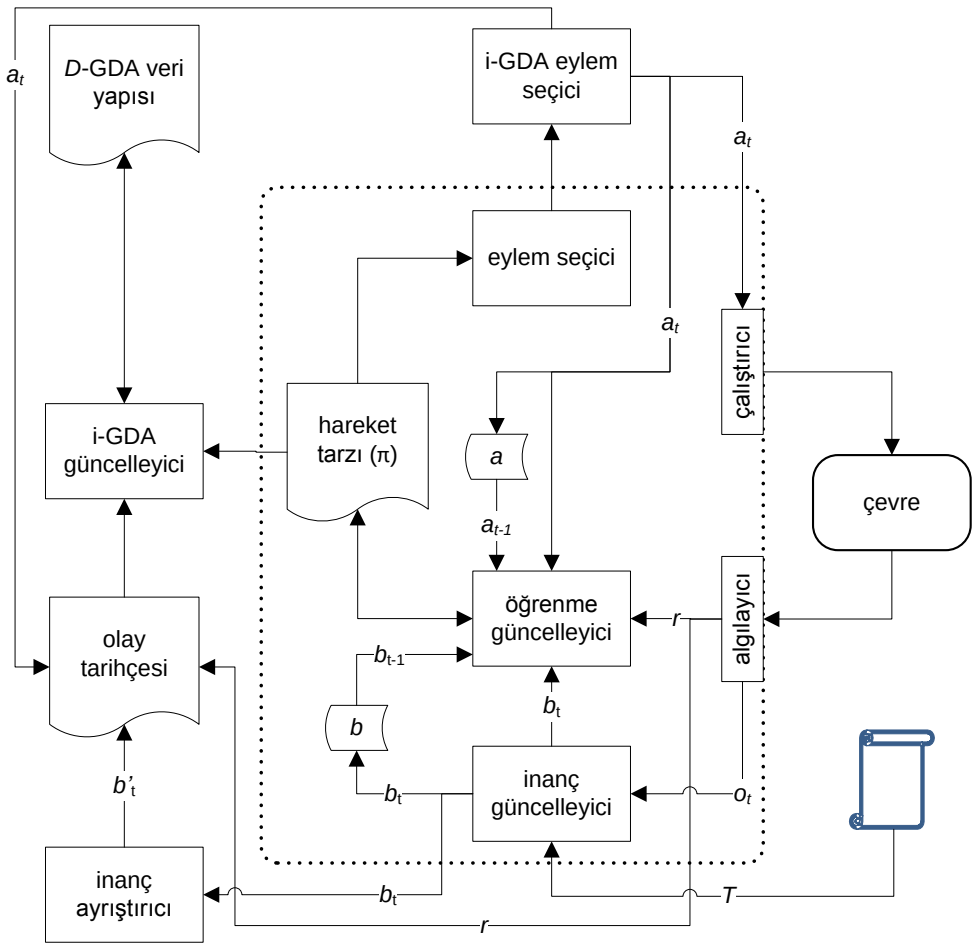
Şekil 4.1, Şekil 2.9'deki gösterime benzer şekilde, i-GDA yönteminin takviye öğrenme algoritması üzerine nasıl inşa edildiğini yapısal olarak özetelemektedir.

Takip eden bölümde, i-GDA yönteminin, uygun inanç ayrıştırma metodları kullanıldığında, altta yatan model tabanlı takviye öğrenme algoritmasının öğrenme performansını arttırdığı deneylerle gösterilmiştir.

Algoritma 8 $i\text{-}GDA(D)$

Require: inanç durum uzayı üzerinde tanımlı bir ayrıştırma işlevi, D

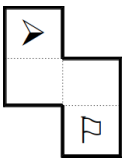
T_D bir D-GDA olsun
$T_D \leftarrow$ tek bir boş kök düğümden oluştan ağaç
repeat
    c, T_D üzerindeki aktif düğümü simgelesin
    $c \leftarrow T_D$'nin kök düğümü
    b, mevcut inanç durumu olsun
    $h_D \leftarrow D(b)$ ▷ bölüm D-tarihçesine ilk değer olarak mevcut inanç durumunun ayrıştırılmış halini ekle
    $aktif \leftarrow false$
    repeat
        if $aktif = true$ **then**
            $a \leftarrow D(b)$ ve T_D kullanarak eylem seç, yan etki olarak $aktif$ değişkenini ve c değişkenini güncelle
        else
            $a \leftarrow$ mevcut değer işlevini kullanarak eylem seç, yan etki olarak $aktif$ değişkenini güncelle
        endif
        a eylemini gerçekleştir, r gözlemle ve sonraki inanç durumunu (b') oluştur
        altta yatan takviye öğrenme güncelleme kuralını uygula (Denklem (2.14) veya (2.15))
        h_D sonuna r, a ve $D(b')$ ekle
        $b \leftarrow b'$
    until çevreden bir son durum sinyali gelene kadar
    T_D veri yapısını h_D bölüm tarihçesi ile güncelle
until bir bitiş kriteri gerçekleşene kadar



Şekil 4.1. KGMKS problemleri için inanç durumu tabanlı takviye öğrenme algoritması ile i-GDA eklentisinin yapısal tasarımı.

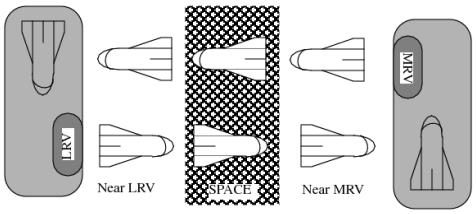
4.3 Deneyler

i-GDA eklentisinin kısmi gözlemlenebilir takviye öğrenme yöntemlerinin performansını iyileştirebileceğini göstermek amacıyla deneyler gerçekleştirilmiştir. Deneylerde farklı boyut ve zorlukta yapay problem çevreleri kullanılmıştır. Bu çevreler, KGMKS literatüründe sıkça kullanılan referans çevreler olup, aşağıdaki paragraflarda özellikleri açıklanmıştır:



Şekil 4.2. Cassandra'nın küçük gezinti ortamı problemi (*Mini-hall*).

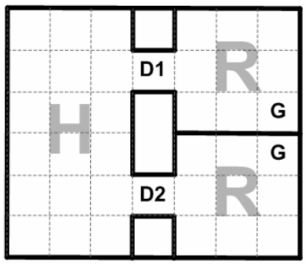
- Küçük gezinti ortamı (*Mini-hall*) (Littman vd., 1998) öngörülebilir eylemler ve sınırlı öngörülebilir algılarla tanımlanmış bir koridor gezintisi problemidir. (Şekil 4.2).
  The problem is intended to model a robot navigating in a simple office environment. Problem, bir robotun basit bir ofis ortamında gezinmesini modellemektedir. Ortam dört hücreden oluşur. Hücrelerden biri hedef hücre (bayrak işareti ile gösterilmiştir) olup, başlangıçta etmen hedef hücre dışında herhangi bir hücrede, ve yüzü dört temel pusula yönünden herhangi birine dönük olabilir. Etmen, etrafındaki duvarların, kendi bakış açısına göre bağıl konumlarını, ve kendisinin hedef hücrede olup olmadığını, mükemmel şekilde algılayabilir. Gerçekleştirebileceği eylemler ise, *ilerle*, *sola dön* ve *sağa dön* eylemleridir.
  Öğrenen etmenin amacı, bu hücre tasarımlı yapay dünyada hedef hücreye en az sayıda adımla ulaşmasını sağlayacak hareket tarzını geliştirmektir. Bu küçük problem, öngörülebilir yapısıyla, problemlerimiz arasında en basitidir.



Şekil 4.3. Chrisman'ın uzay mekiği kenetlenme problemi (*Shuttle*) (Chrisman, 1992).

- Uzay mekiği kenetlenme problemi (*Shuttle*) (Chrisman, 1992), iki uzay istasyonu arasında malzeme taşıyan bir uzay gemisinin basitleştirilmiş bir ayrık benzetimidir (İng. discrete simulation) (Şekil 4.3). Bu problemde, etmenin, yakın zamanda en az ziyaret edilmiş istasyona gitmeyi öğrenerek, süreçten elde edebileceği toplam ödülü maksimuma çıkarması beklenir.

Gerçekleştirilebilecek eylemler *ileri git*, *etrafında dön* ve öngörülebilir olmayan *yükle* eylemleridir. Gemi, belirli olasılıklar dahilinde, önünde istasyon olup olmadığını, istasyonlardan birine kenetli olup olmadığını görebilmekte, veya hiç bir şey algılayamayabilmektedir. Bu problemi diğerlerinden ayıran, bir hedef durumu olmamasıdır, yani problem doğal olarak bölümlere ayrılamamaktadır. Ayrıca, problem çok büyük olmamakla birlikte, algılayıcıların ve çalıştırıcıların güvenilir olmaması nedeniyle, *Mini-hall* probleminden daha zor bir deney ortamıdır.



Şekil 4.4. Dung'un sanal ofis (*Virtual Office*) problemi (Dung vd., 2007).

- Sanal ofis (*Virtual Office*) (Dung vd., 2007) problemi, iki darboğaz durumu (D1 ve D2) ile bu darboğaz durumlarından geçerek ulaşılabilecek iki hedef durumu (G) içeren bir gezinti problemidir. (Şekil 4.4).

  Etmenin, H koridorunda (sol oda) rastsal olarak seçilecek herhangi bir başlangıç durumundan başlayarak, sağ odalardan birindeki hedef durumuna ulaşması beklenir. Etmen, etrafındaki dört temel pusula yönünde duvar olup olmadığını kesin olarak gözlemleyebilir. Bu gözlem mantığı nedeniyle, sağ üst odadaki gözlemlerle, sol üst odaki gözlemler, hedef durumları hariç, aynıdır. Eylem uzayı, dört temel pusula yönüne doğru tek adımlık hücre geçişinden oluşur. Etmen eğer hedefe ulaşırsa $+10$ ödül alır. Diğer eylemler, etmenin -1.0 ödül alması ile sonuçlanır. D1 ve D2 kapı durumları, ortamın çözüm hareket tarzında hiyerarşik bir yapı tanımlarlar. T ve O işlevleri ise öngörülebilirdir.

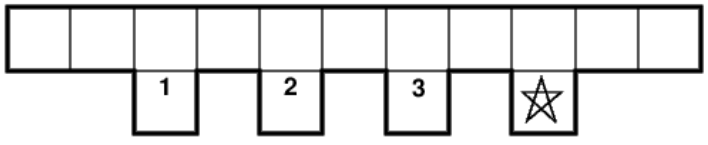
S0 O1	S1 O2	S2 O3	S3 O2	S4 O4
S5 O5		S6 O5		S7 O5
S8 O6		S10 O7		S9 O6

Şekil 4.5. Pineau's peynir-taksi (*Cheese-Taxi*) probleminde durum/gözlem uzayı (Pineau, 2004).

- Peynir-taksi (*Cheese-Taxi*) problemi (Pineau, 2004) bilinen yaygın iki problem olan peynir labirenti (İng. *Cheese Maze*) (McCallum, 1996) ve taksi ortamı (İng. *Taxi Domain*) (Dietterich, 2000) problemlerinin bir karmasıdır. *Cheese-Taxi*, belirsizlikle ilgili özellikleri *Cheese Maze*, hiyerarşik yapı özelliğini de *Taxi Domain* problemlerinden almaktadır (Şekil 4.5).

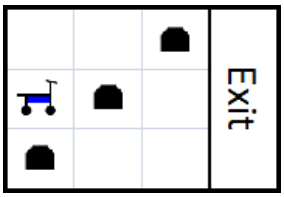
  Problem, bir taksi etmeninin basit bir benzetimidir. Etmenin, S10 hücresinde beklemekte olan yolcuyu alıp, S0 veya S4 hedef hücresine en kısa sürede taşıması beklenir. Etmenin uygulayabileceği yedi eylem vardır: *Kuzey, Güney, Doğu, Batı, Sorgula, Al, Bırak*. Etmen, on ayrı gözlem yapabilir: *O1, O2, O3, O4, O5, O6, O7, hedefS0, hedefS4*, ve *geçersiz*. O1-O7 arası gözlemler etmenin bulunduğu hücreyi çevreleyen duvarlara göre hesaplanır. S5, S6, S7 durumları, S1, S3 durumları ve S8, S9 durumları, grup olarak birbirleri ile aynı algılanırlar (sırasıyla O5, O1 ve O6 olarak). Diğer durumlar ise ayrı gözlemlerle tanımlanırlar. Hedef gözlemler, eğer yolcu taksinin içindeyse, *Sorgula* eylemi kullanılarak kesin olarak algılanabilmektedir. *Al* ve *Bırak* eylemlerinden sonra etmen *geçersiz* gözlemini edinir.

  Etmenin amacı, yolcuyu doğru konuma taşıyıp $+20$ ödülü almaktır. *Al* ve *Bırak* eylemlerinin yanlış kullanımında etmen -10 ile cezalandırılır. Bu problemde belirsizliğin iki kaynağı vardır. Taksinin ilk konumu rastsal olarak belirlenmektedir, ve ancak bir dizi hareket eylemi gerçekleştikten sonra ayırt edilmeye başlanmaktadır.



Şekil 4.6. Cassandra'nın koridor (*Hallway)* gezinti problemi (Cassandra 1998).

- Koridor (*Hallway*), (Cassandra, 1998) tarafından önerilen bir dizi koridor gezintisi probleminden biridir ve orta büyüklüktedir (Şekil 4.6). Gözlem ve eylem mantığı, Mini-hall problemi ile aynıdır. Ek olarak, etmenin mutlak konumunu algılayabildiği üç adet ayırt edici durum bulunmaktadır. Böylece, etmen eğer bu ayırt edici durumları dikkate alan bir hareket tarzı geliştirebilirse, önemli bir avantaj elde edebilmektedir.
  Bu problem, önceki problemlere göre daha zordur, çünkü hem göreceli olarak daha büyüktür, hem de son derece gürültülü bir ortam tanımlar.



Şekil 4.7. Taş toplama (Rock sampling) problemi, 3x3 ızgara modeli ve 3 taş ile (Smith ve Simmons, 2004).

- Taş toplama (İng. Rock sampling) problemi (*RockSample[x, y]*) (Smith ve Simmons 2004) bilimsel keşif robotunu modellemektedir. Bu modelde x kare biçimli bir ızgara dünyasını, y ise ortamdaki taş sayısını tanımlar (Şekil 4.7).
  Robot, dört temel pusula yönüne gerçekleştirilebilecek tek adımlık hareket eylemlerinin dışında, $Ornek\_topla$, $Kontrol_1$, $Kontrol_2$, ... $Kontrol_y$ eylemleri de yapabilir. Ortamdaki her taş, örnekleme için "değerli" veya "değersiz"dir. Etmenin amacı, "değerli" tüm taşları örnek olarak toplamak, ve sonrasında hedef durumuna ulaşmaktır. Etmen, kendi koordinatlarını ve taşların yerlerini bilir, fakat hangi taşların "değerli" (toplamaya değer) olduğunu bilmez. Taşların değerini anlamak amacıyla, her taş için $Kontrol_i$ eylemini çalıştırması gerekir. Bir taşın değerli olup olmadığını, uzaklığına bağlı olarak gürültülü şekilde algılayabilir. Bir taştan örnek topladığında ise, taş "değersiz" hale gelir.
  RockSample, iki önemli özelliği nedeniyle problemlerimiz arasında en zorlu olanıdır: (1) Ortam sabit değildir, değişkendir (İng. non-stationary), çünkü bir taştan örnek alındığında taş değersiz hale gelmektedir. Dahası, rastsal olarak oluşturulmuş her başlangıç konfigürasyonu, toplanabilecek maksimum ödül miktarını belirlemektedir. (2) Gözlem uzayı hem çok kısıtlıdır, hem de güvenilirliği azdır.

4.3.1 Deney Düzeneği

Deneyde kullanılan problemlerin durum, eylem ve gözlem uzayı büyüklükleri, problemin geçiş işlevinde ve gözlem işlevinde gürültü olup olmadığını, ve ilgili problem tanımının yapıldığı referans yayını, Tablo 3.1'de özetlenmiştir.

Her problem Tablo 4.2'de listelenen farklı $p_{eşik}$, M ve n değerleri ile test edilmiştir. Bu değerler, bir dizi deneme-yanılma deneyimiyle elde edilmiş olup, ilgili problem için soyutlama olmaksızın gerçekleştirilen koşuma göre performans artışı sağladıkları tespit edilmiştir. Ayrıca, merkezi işlem birimi (İng. Central Processing Unit, CPU) zamanı ve hafıza kullanımı gibi performans parametreleri açısından da göreceli olarak iyi sonuçlar sağlamışlardır.

Tablo 4.1. Kullanılan problemlerin özellikleri

Problem	**Boyutlar**			**Gürültü**	**Referans**
	$\|S\|$	$\|A\|$	$\|\Omega\|$		
Mini-hall	13	3	9	-	(Littman vd., 1998)
Shuttle	8	3	5	T/O	(Chrisman, 1992)
Virtual Office	38	4	12	-	(Dung vd., 2007)
Cheese-Taxi	35	7	10	T	(Pineau, 2004)
Hallway	60	5	21	T/O	(Cassandra, 1998)
RockSample[3,3]	257	9	2	O	(Smith ve Simmons,2004)

Tablo 4.2. Deney ayarları

Problem	D_{IZGARA}	$D_{SIRALAMA}$	$D_{EKLENTI}$	
	M	$p_{eşik}$	n	$p_{eşik}$
Mini-hall	7	0.2	10	0.0
Shuttle	3	0.1	5	0.0
Virtual Office	3	0.6	5	0.0
Cheese-Taxi	15	1.0	15	1.0
Hallway	10	1.0	20	1.0
RockSample[3,3]	5	1.0	10	1.0

Tablo 4.3. Öğrenme paremetrelerinin değerleri

Parametre	**Değer**
α	0.125
γ	0.9
ϵ	0.1
$\psi_{\text{çürüme}}$	0.95
$\xi_{\text{çürüme}}$	0.99
$\psi_{eşik}$	0.01
$\xi_{eşik}$	0.01

Öğrenme parametrelerinin varsayılan değerleri Tablo 4.3'te listelenmiştir. Tek istisna, ϵ değerinin keşif davranışını teşvik ederek daha hızlı yakınsamaya olanak sağlaması amacıyla 0.3 olarak alındığı *RockSample* problemidir. Bu problemde varsayılan değerlerle makul sürede sonuca yakınsamak mümkün olamamaktadır.

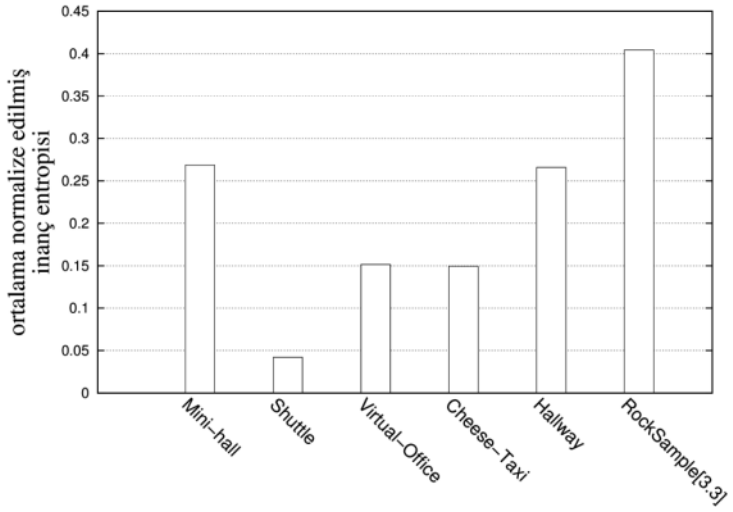
Tüm deneylerde Girgin vd. (2010) tarafından önerilen Değiştirilmiş-ϵ-Hırslı (İng. Modified-ϵ-Greedy) yöntemi, eylem seçim mekanizması Bölüm 2.6.3.2 and Bölüm 4.2'te tartışıldığı şekilde güncellenerek kullanılmıştır.

Her problem, yalın olarak bir model tabanlı takviye öğrenme algoritmasının (i-GDA olmadan, Doğrusal Q-Öğrenme veya Tekrarlanan Q-Öğrenme), aynı algoritmanın i-GDA (Algoritma 8) ile genişletilmiş hali ile karşılaştırılması amacıyla, Tablo 4.2'de verilen değerler kullanılarak ve her bir ayrıştırma işlevi ayrı deneylerle test edilmiştir. Yakınsama davranışına bağlı olarak, deneyler her problem için farklı sayıda bölüm için koşturulmuştur. Her test koşumu 250 kez tekrarlanarak her koşumun aynı bölümlerinin performansının ortalaması alınmıştır. Temel performans kriteri, etmen tarafından zaman adımı başına elde edilen ortalama ödüldür (İng. reward-per-step). Tek istisna, *RockSample* problemidir. Bu problemde ise, değişken yapısı nedeniyle, ortalama birikimli ödül (veya toplam ödül) ölçülmüştur.

Bölüm tabanlı olmayan yapısı nedeniyle, *Shuttle* problemi için bir bölüm 250 zaman adımı olarak kabul edilerek, bu zaman sonunda yeni bir bölüm başlatılmıştır. Bu problemde, yine aynı nedenle, Algoritma 8'de GDA güncellemesinin tetiklenmesi için bölüm sonu sinyali yerine alınabilecek en yüksek ödül sinyali kullanılmıştır.

4.3.2 Sonuçlar ve Tartışma

Deney sonuçlarını sunmadan önce, seçilen problemlerin büyüklüğüne ilişkin bir fikir sahibi olmak yararlı olacaktır. Bir etmenin bulunduğu çevre içerisindeki deneyimindeki belirsizlik miktarını tanımlamak pek kolay olmasa da, inanç durumlarının normalize edilmiş entropi değeri (Denklem (4.4)), etmenin tanımakta olduğu çevreyi ne kadar bulanık algıladığı ile ilgili kaba bir fikir verebilir. Şekil 4.8, Tekrarlanan Q-Öğrenme çalışırken (i-GDA olmadan) ölçülen normalize edilmiş inanç entropi değerlerinin ortalamasını, her problem için ayrı ayrı göstermektedir. Bu grafikte düşük değer daha az belirsizlik anlamına gelmektedir. Grafikten de açıkça görüldüğü gibi, *RockSample[3,3]* problemi, problem kümemizin en çok belirsizlik içeren çevresini tanımlamaktadır. *Shuttle* ise, ortalamada etmene en eksiksiz bilgi durumunu sağlayan çevre olarak ortaya çıkmaktadır.



Şekil 4.8. Problemlerin ortalama normalize edilmiş inanç entropi seviyeleri.

Şekil 4.9 ile Şekil 4.14 arasındaki şekiller, Tablo 4.2 ile verilen deney ayarlarının her biri ile, deney bölümleri ilerledikçe elde edilen ortalama performansı göstermektedir. Çizimler, görsel netlik elde etmek amacıyla düzleştirme (İng. smoothing) işleminden geçirilmiştir.

i-GDA belirli bir ek CPU zamanı maliyeti getirmektedir. Bu maliyetler Şekil 4.15 ile özetlenmiştir. Bu grafikler her bir problem ve her bir ayrıştırma yöntemi için, i-GDA yönteminin altta yatan takviye öğrenme algoritmasına yüzde kaç ek CPU zamanı maliyeti eklediğini göstermektedir.

Bir diğer önemli performans parametresi ise yöntemin kullandığı bellek miktarıdır. Tablo 4.4, altta koşturulan her bir takviye öğrenme algoritması ve ayrıştırma yöntemi için i-GDA tarafından oluşturulan düğüm sayısını göstermektedir.

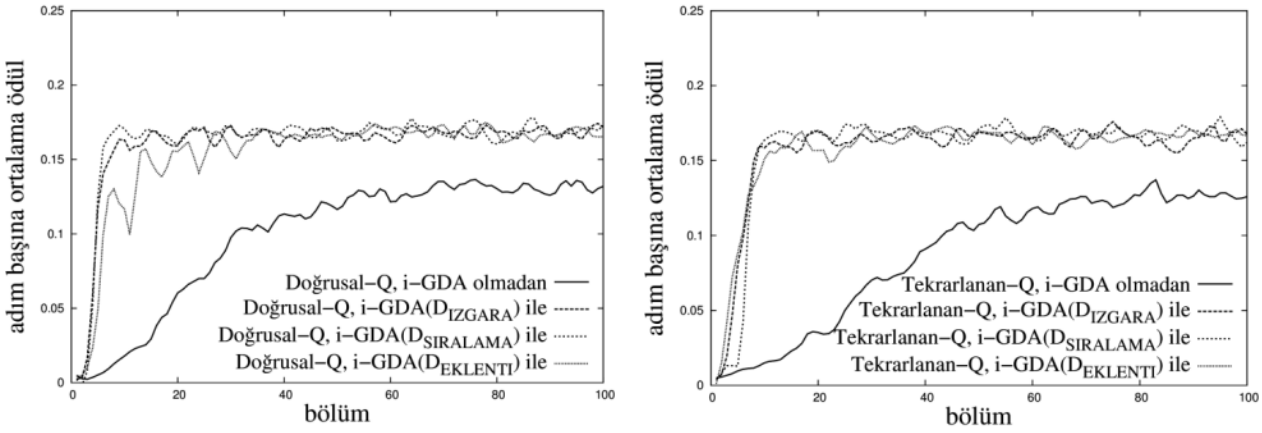
Şekil 4.16, öğrenme sırasında uygulanan eylemler arasında soyut eylemlerin oranını göstermektedir. Diğer bir deyişle, "tercihlerin" ortalama kullanımı, uygulanan tüm "ilkel eylemlere" göre yüzdelik bir değer olarak verilmiştir.

Son olarak, Tablo 4.5, i-GDA tarafından her bir ayrıştırma yöntemi ve her bir problem için üretilen ayrıştırılmış durum uzaylarının büyüklüğünü özetlemektedir.

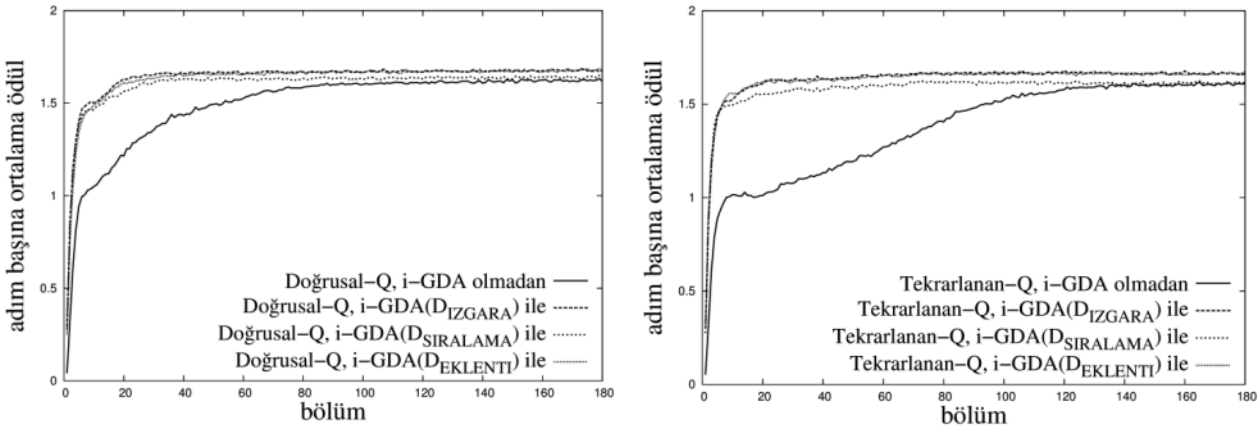
Hemen hemen tüm koşumlarda, model tabanlı takviye öğrenme yönteminin üzerinde çalıştırılan i-GDA metodu, kayda değer performans iyileştirmesi sağlamıştır. Ancak, performans artışındaki "kayda değerlik" derecesini belirleyen parametrelerin yorumlanması kolay değildir.

Mini-hall problemi, problem kümemizdeki en basit problem olarak, i-GDA yönteminden diğer problemlere göre daha çok yararlanmış görünmektedir. Uygun parametre değerleri kullanıldığında tüm ayrıştırma yöntemleri, i-GDA eklentisinin öğrenmeyi hızlandırmasını sağlamaktadır (Şekil 4.9). Tablo 4.2'deki gibi çok küçük M, n and $p_{eşik}$ değerleri için bile i-GDA alttaki öğrenme algoritmasını hızlandırmayı başarmaktadır. Bunun büyük oranda sebebi, ortalama olarak eylemlerin yaklaşık %50'sinin "tercih" olarak kullanılmak üzere soyutlanabiliyor olmasıdır (Şekil 4.16). Bu durum, i-GDA yönteminin bu kadar küçük bir problem için bile başarılı olabilmesinin temel nedenidir.

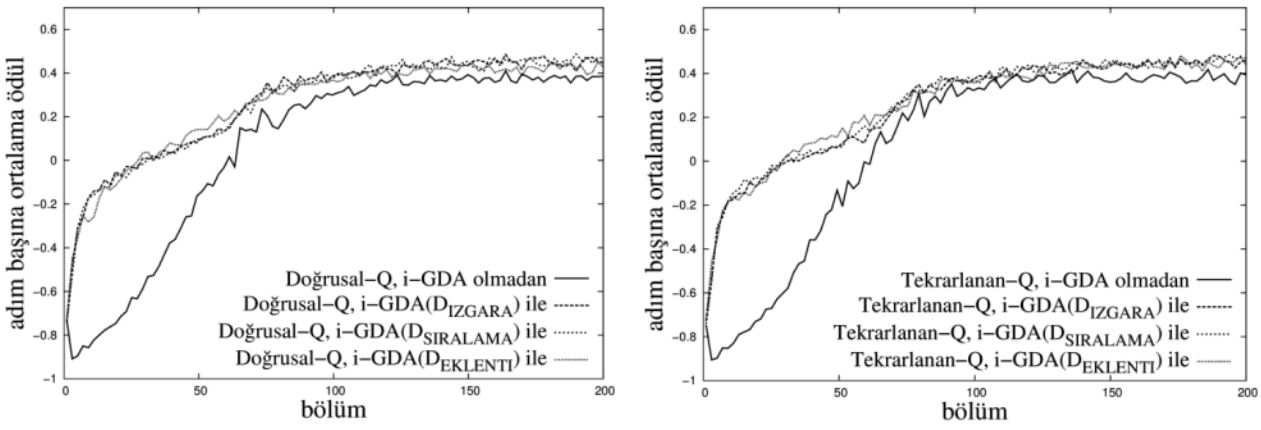
Problem kümemizin diğer bir küçük problem olan *Shuttle* problemi de i-GDA ile kayda değer performans kazanımı elde etmiştir. Soyutlama potansiyeli diğer problemlere göre daha az olmasına rağmen (Şekil 4.16 ile verilen düşük tercih kullanım yüzdelerinin işaret ettiği üzere), öngörülebilir doğası (Şekil 4.8 ile verilen düşük belirsizlik değerinin anlattığı üzere) performans artışına izin vermekte, küçük ayrıştırma parametre değerleri ile dahi bu artış elde edilebilmektedir (Tablo 4.2).



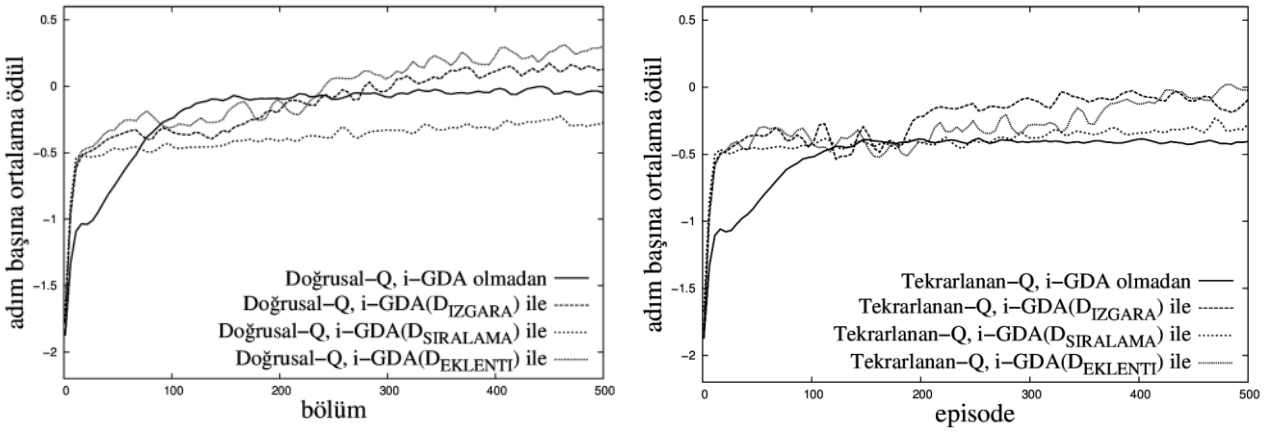
Şekil 4.9. Mini-hall problemi için deney sonuçları



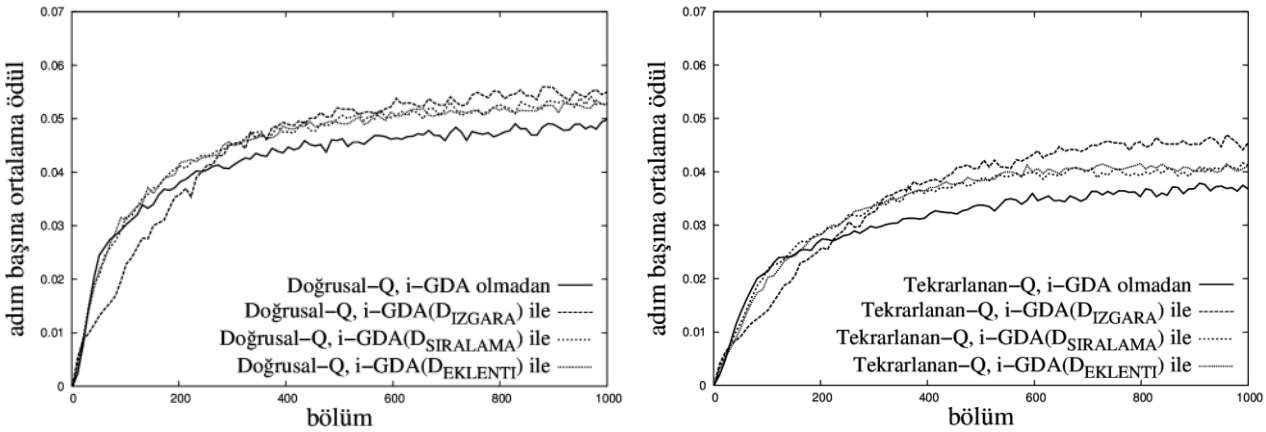
Şekil 4.10. Shuttle problemi için deney sonuçları



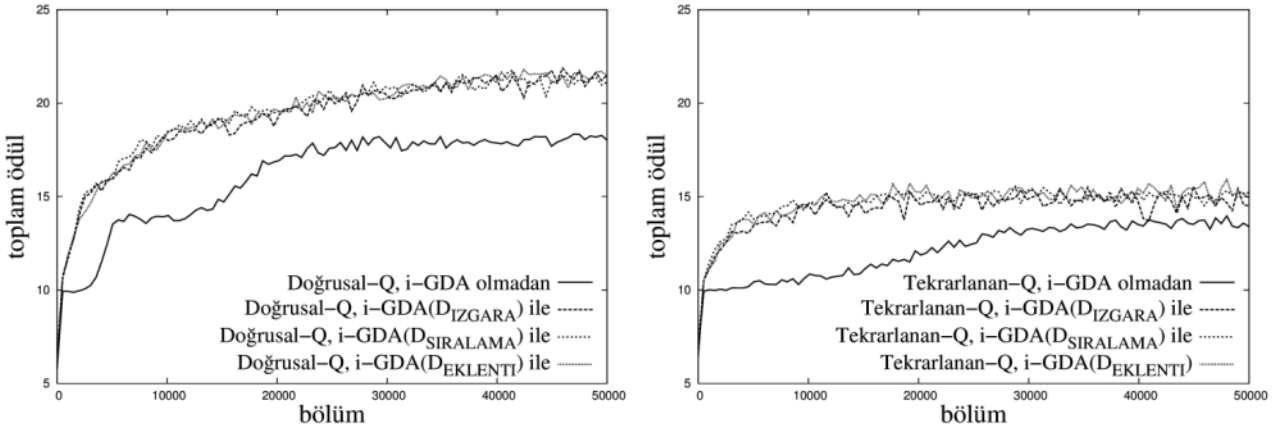
Şekil 4.11. Virtual Office problemi için deney sonuçları



Şekil 4.12. Cheese-Taxi problemi için deney sonuçları



Şekil 4.13. Hallway problemi için deney sonuçları



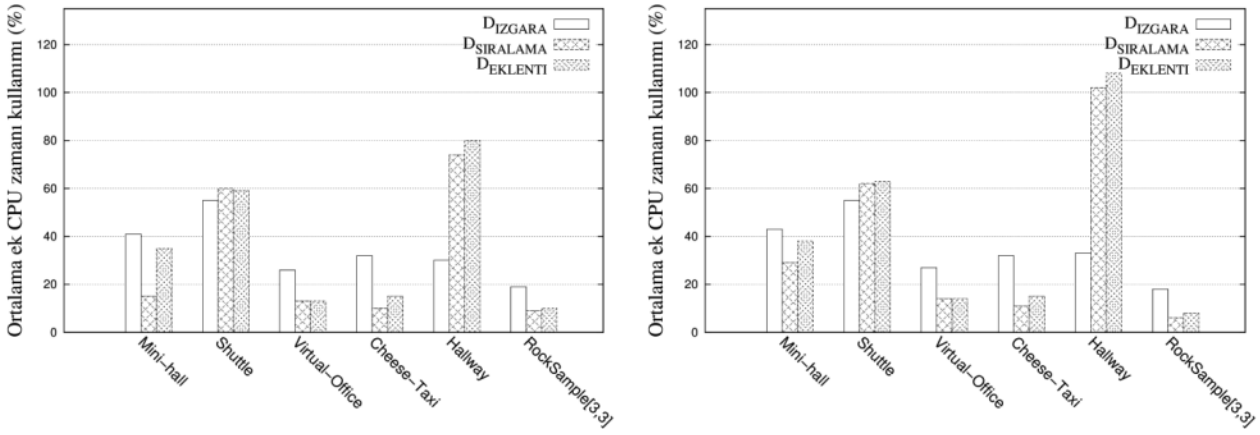
Şekil 4.14. RockSample[3,3] problemi için deney sonuçları

Şekil 4.11 ile gösterildiği gibi, i-GDA, zamansal soyutlamaya uygun şekilde özel olarak tasarlanmış problemlerden biri olan *Virtual Office* problemi için de iyi bir performans göstermiştir. Bu problem için Şekil 4.16 ile verilen tercih kullanım yüzdelerinin yüksekliği dikkat çekicidir.

Zamansal soyutlama için tasarlanmış diğer problem olan *Cheese-Taxi* problemi de i-GDA yönteminden yararlanabilmektedir (Şekil 4.12). Özellikle öğrenmenin ilk aşamalarında i-GDA altta çalışan öğrenme algoritmasını etkin olarak desteklemektedir. Bu problem için, $D_{SIRALAMA}$ ilk bölümlerde performansı arttırsa da, ilerleyen bölümlerde problemin gürültülü (İng. noisy) doğasını taşımakta başarısız olmaktadır. Uygun parametreler kullanıldığında $D_{EKLENTİ}$ ve D_{IZGARA} yöntemleri, i-GDA için taha çok parçalı bir ayrıştırılmış durum uzayı tanımladıklarından, daha iyi sonuçlar vermektedirler (Tablo 4.5).

Hallway gibi son derece gürültülü problemlerde, i-GDA yönteminin, aslında doğru olmayan, ancak gürültü nedeniyle kazara başarıya ulaşmış, başarılı olarak kaydedilmiş ve D-GDA veri yapısına kaydedilmiş tercih rotaları keşfetmesi olasılığı yüksektir. Bu rotalar daha sonra D-GDA veri yapısından silinse de, Tablo 4.3 ile verilen sınırlı çürüme ve eşik parametre değerleri nedeniyle budama mekanizması biraz gecikmeyle gerçekleşmekte, bu da D_{IZGARA} eğrilerindeki geç iyileşme eğilimini açıklamaktadır. Ayrıca, problemdeki durum sayısı ve ortalama inanç entropi değeri arttıkça, gereken ayrıştırma sayısı da azalmaktadır. Bunun sonucu olarak, görece çok sayıda oluşan ayrıştırılmış inanç durumları nedeniyle (Tablo 4.5), *Hallway* problemi için olduğu gibi, yüksek CPU zaman kullanımı gözlemlenebilmektedir (Şekil 4.15).

Ayrıştırma yöntemleri karşılaştırıldığında, ortalama düğüm sayısındaki en dikkat çekici fark *Hallway* problemi için oluşmaktadır (Tablo 4.4). $D_{SIRALAMA}$ ve $D_{EKLENTİ}$ yöntemleri, D_{IZGARA} yöntemine kıyasla önemli ölçüde daha çok düğüme ihtiyaç duymaktadır. Bu sonuç, D_{IZGARA} ayrıştırmasının *Hallway* problemi kapsamında, görece az sayıda ayrıştırılmış durum kullanarak (Tablo 4.5) daha çok sayıda "sık kullanılan" (veya yararlı olan) tercih oluşturma konusunda, uzun erimde daha etkin olmasına bağlanabilir (Şekil 4.16). Genel olarak, bu tür farkların *Hallway* gibi son derece gürültülü ve görece büyük problemlerde ortaya çıkması olasılığı daha yüksektir. Bu tür durumlar için, $D_{SIRALAMA}$ ve $D_{EKLENTİ}$ yöntemleri, sürekli inanç durum uzayını daha çok sayıda ayrıştırılmış durum kullanarak temsil etmeye meyillidirler. Dolayısı ile, benzer inanç durumları için farklı ayrıştırmaların üretilmesi tetiklenebilir, ve buna bağlı olarak aynı alt-göreve karşılık gelen daha fazla sayıda tercih üretilmesi söz konusu olabilir.



Doğrusal Q-Öğreme Tekrarlanan Q-Öğrenme

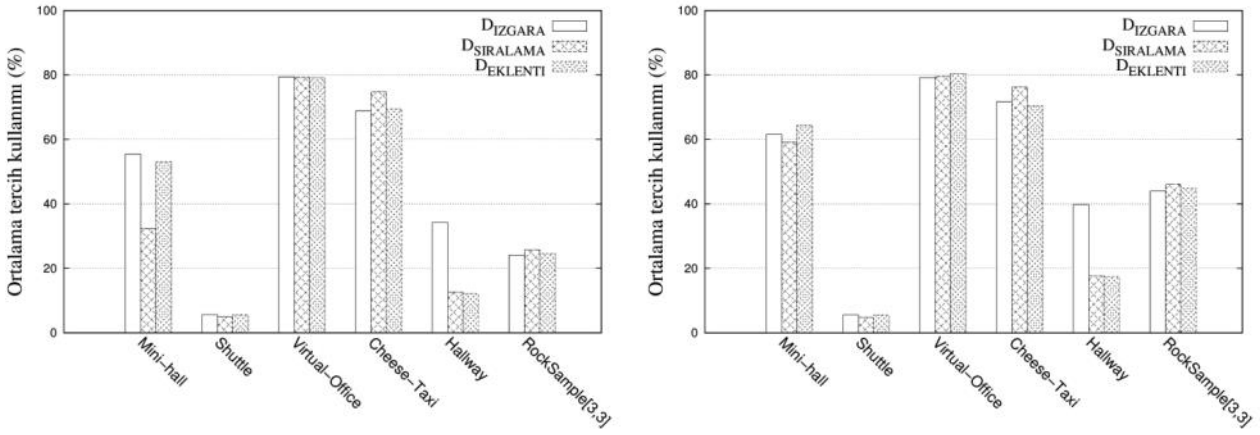
Şekil 4.15. Ortalama ilave CPU zamanı yüzdeleri

Son olarak, *RockSample[3,3]* problemi söz konusu olduğunda, i-GDA önerdiğimiz herhangi bir ayrıştırma yöntemi ile model tabanlı takviye öğrenme yöntemini hızlandırmaktadır. Bu problem, problem kümemizdeki en yüksek belirsizlik düzeyine sahip olmasına karşın (Şekil 4.8), i-GDA altta çalışan takviye öğrenme algoritmasını, episode başına görece ortalama bir tercih kullanım yüzdesi ile, başarılı bir şekilde desteklemektedir (Şekil 4.16).

Bu bölümde tanıtılan üç ayrıştırma yönteminin de karmaşıklık (İng. complexity) düzeyi $O(|S|\log|S|)$ dir. Bu nedenle, seçilen bir ayrıştırma yönteminin tüm hesapalama maliyetine olan etkisini saptamak zordur. Ancak, problem doğasına bağlı olarak, kullanılan ayrıştırma yönteminin ayrıştırılmış inanç durum uzayının büyüklüğünü etkilediği, dolayısı ile oluşturulan D-GDA düğüm sayısında ve CPU zamanındaki artışıta etkisi olabileceği söylenebilir.

Tablo 4.4. D-GDA veriyapısındaki ortalama düğüm sayısı

Problem	**Doğrusal Q-Öğrenme**			**Tekrarlanan Q-Öğrenme**		
	D_{IZGARA}	$D_{SIRALAMA}$	$D_{EKLENTI}$	D_{IZGARA}	$D_{SIRALAMA}$	$D_{EKLENTI}$
Mini-hall	103.47	62.17	104.83	105.37	62.69	105.36
Shuttle	51.19	43.43	50.65	55.45	45.83	53.71
Virtual Office	419.80	427.03	381.76	431.14	424.25	374.68
Cheese-Taxi	886.78	525.93	794.50	828.06	505.79	788.09
Hallway	9959.80	48588.29	54999.75	12470.42	79783.73	88499.95
RockSample[3,3]	1518.97	1289.17	1382.53	691.71	555.63	612.44



Doğrusal Q-Öğrenme Tekrarlanan Q-Öğrenme

Şekil 4.16. Tüm eylem adımları içindeki ortalama tercih eylemi kullanımı yüzdesi

Model tabanlı takviye öğrenme, i-GDA ile birlikte çalıştırıldığında tek başına olduğundan daha erken bir bölümde ortalama ödül zirvesine ulaşabilmekte, dolayısı ile toplam ödül edinimini de tercih kullanımları ile arttırabilmektedir. Örneğin, *Shuttle* problemi için model tabanlı takviye öğrenme yaklaşık 100üncü bölümde zirveyi yakalarken, aynı algoritmanın i-GDA ile birlikte çalıştırıldığında benzer değerleri 20nci bölüm dolaylarında eriştiği görülmektedir.

Bu nedenle, erken yakınsama ve ortalama ödüldeki belirgin artış göz önüne alındığında, CPU zamanındaki ek yük genel anlamda kabul edilebilir seviyede kalmaktadır. Dahası, çevrim dışı doğası nedeniyle, i-GDA yönteminin alttaki öğrenme algoritması ile –belirli sayıda bölüm gecikmesi belirlenerek- paralel olarak çalıştırılması her zaman mümkündür.

Tabii ki, i-GDA yönteminde iyileştirme gerektiren kısımlar da mevcuttur. Belki de bunlardan en önemlisi, i-GDA yönteminde soyutlama ve ayrıştırma parametrelerinin önceden doğru şekilde seçilmesi veya ayarlanmasını gerekliliğidir. Ne yazık ki, deneyimle elde edilecek tahminler dışında, bu parametrelerin belirli bir problem için ne şekilde doğruya yakın bulunabileceğine ilişkin bir uygulama pratiği önerilememektedir.

Tablo 4.5. Oluşturulan ayrıştırılmış inanç durum uzayının ortalama büyüklüğü

Problem	Doğrusal Q-Öğrenme			Tekrarlanan Q-Öğrenme		
	D_{IZGARA}	$D_{SIRALAMA}$	D_{IZGARA}	$D_{SIRALAMA}$	D_{IZGARA}	$D_{SIRALAMA}$
Mini-hall	19.13	13.04	18.18	19.19	12.95	18.01
Shuttle	16.71	7.95	13.11	16.78	7.96	13.01
Virtual Office	49.52	46.88	38.32	49.91	46.99	38.55
Cheese-Taxi	68.08	49.58	68.3	65.4	48.84	67.72
Hallway	2774.11	14049.06	14442.85	3064.65	16999.05	17743.92
RockSample[3,3]	1156.36	889.17	1430.64	1046.57	751.03	1124.92

5 SAKLI DURUMLAR İÇEREN TAKVİYE ÖĞRENMEDE REAKTİF HAREKET TARZLARININ DAHA HIZLI ÖĞRENİLMESİ

Saklı durumlar içeren problemler üzerinde yapılan erken dönem takviye öğrenme çalışmaları, bir KGMKS problemini salt gözlemleri kullanarak kabul edilebilir kalitede çözebilen belleksiz hareket tarzları üretmeye odaklanmıştı. Bu bölümde, bahsi geçen kategoriyi temsilen, belleksiz bir takviye öğrenme metodu olan SARSA(λ) algoritmasını, genişletilmiş dizi ağacı soyutlama yöntemi kullanarak iyileştiren bir yöntem önerilmektedir (Çilden ve Polat, 2013).

Bölüm 2.7 ile anlatıldığı üzere, saklı durumlar içeren problemler için takviye öğrenme uygulamalarındaki temel problem *algısal çakışmadır* (İng. perceptual aliasing). Bu durum, altta yatan MKS modeline etmenin erişiminin olmaması ile ortaya çıkmaktadır.

GDA yöntemi MKS modeli üzerine kurgulandığından, döngü içermeyen kullanışlı tarihçelerin sıkıştırılmış bir şekilde ağaç veri yapısı ile temsil edilmesi gibi basit bir fikre dayanan GDA veri yapısının inşasında, algısal çakışma olmadığını varsayar. Bu nedenle, GDA yöntemi kısmi gözlemlenebilirlik varsayımı altında son derece kırılgandır. Kırılganlığın kökeninde algısal çakışma oluşan iki durumun sanki aynıymış gibi ele alınmasının etmeni çözüm uzayında tamamen yanlış bir tarafa yönlendirebileceği gerçeği yatmaktadır.

Bu problemin üstesinden gelmek için, algısal çakışma yaşanan durumların GDA veri yapısından ayıklanmasını sağlayacak bir yönteme ihtiyaç vardır. Bu noktada, GDA veri yapısının temel olarak, -GDA'nın daima başarılı olmuş tarihçeler barındırdığı varsayımına dayanarak- başarılı alt hareket tarzlarının "işletimi" için kullanıldığını hatırlatmak gerekir. Bu gerçeğe dayanarak, bir *yanıltıcı işletim* dizisinin (yani, değerli bir hedefe varmayan bir GDA rotasının) "yanlış pozitif" rotaların tespitinde kullanılabileceğini farketmek zor değildir.

Bu bölümde, bahsi geçen tespiti, GDA veri yapısı üzerinde titiz bir budama stratejisi kullanarak gerçekleştirebilecek bir yöntem önerilmektedir. Yöntem, altta yatan takviye öğrenme algoritması olarak SARSA(λ) kullanmakta, ve bölüm tarihçelerini işleyerek yararlı alt hareket tarzları biriktirilmesini sağlayabilmektedir.

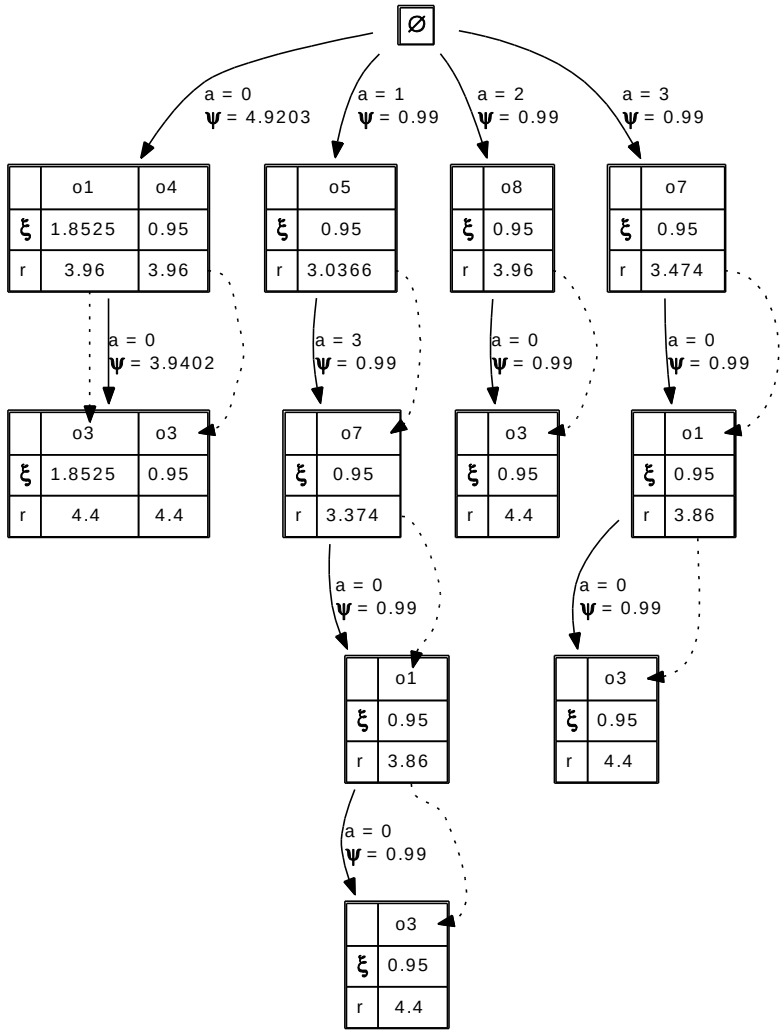
5.1 Tarihçe Güdümlü Genişletilmiş Dizi Ağacı Veri Yapısı

GDA yönteminin prosedürleri üzerinde değişiklik yapmadan önce, GDA veri yapısı aşağıdaki şekilde yeniden tanımlanmıştır:

Tanım 5.1. Tarihçe güdümlü genişletilmiş dizi ağacı (TG-GDA) düğümler kümesi N ve ayrıtlar kümesi E bileşenlerinden oluşur. Her düğüm, o düğüme ulaşma yolunda deneyimlenmiş bir eylem dizisini temsil eder ve bu eylem dizisinin ağaçta bir tekrarı yoktur. Kök düğüm $\emptyset$ sembolü ile gösterilir ve boş eylem kümesini temsil eder. Eğer q düğümünün eylem dizisi, a eyleminin p düğümü ile temsil edilen eylem dizisine eklenmesi ile elde edilebiliyorsa, bu durumda p düğümü q düğümüne $\langle a, \psi \rangle$ etiketli bir ayrıt ile bağlıdır; ve bu bağ $\langle p, q, \langle a, \psi \rangle \rangle$ ile gösterilir. ψ, q düğümünün temsil ettiği eylem dizisinin hangi sıklıkla uygulandığını anlatan ayrıt nitelik (İng. eligibility) değeridir. Dahası, q düğümü, eğer mevcut gözlem ve önceki adımdaki devam kümesi elemanı ikilisi $\{\langle o_1, \Pi_1^{\mathrm{p}} \rangle, \dots, \langle o_k, \Pi_{\mathrm{k}}^{\mathrm{p}} \rangle\}$ kümesine dahil ise a eyleminin p düğümünde seçilebileceğini anlatan $\langle \langle o_1, \Pi_1^{\mathrm{p}} \rangle, \xi_{\langle o_1, \Pi_1^{\mathrm{p}} \rangle}, R_{\langle o_1, \Pi_1^{\mathrm{p}} \rangle} \rangle, \dots, \langle \langle o_k, \Pi_{\mathrm{k}}^{\mathrm{p}} \rangle, \xi_{\langle o_k, \Pi_{\mathrm{k}}^{\mathrm{p}} \rangle}, R_{\langle o_k, \Pi_{\mathrm{k}}^{\mathrm{p}} \rangle} \rangle$ yapısını barındırır. Bu yapıya q düğümünün devam kümesi adı verilir ve $cont_q$ ile gösterilir. $\Pi_{\mathrm{i}}^{\mathrm{p}}$, mevcut devam kümesi elemanının bir zaman adımı öncülü olan $cont_q$ elemanını belirtir. Diğer bir deyişle, $\Pi_{\mathrm{i}}^{\mathrm{p}}$, bir önceki tercih eylem adımında seçilmiş olan devam kümesi elemanıdır. Bir devam kümesi elemanı, $\langle o_i, \Pi_i^{\mathrm{p}} \rangle$ ikilisi ile tanımlanır ve indislenir. $R_{\langle o_i, \Pi_{\mathrm{i}}^{\mathrm{p}} \rangle}$, p düğümü ile temsili edilen eylem dizisinin uygulanmasının ardından, etmenin $\langle o_i, \Pi_i^{\mathrm{p}} \rangle$ ikilisine ulaşması durumunda a eylemini seçmesi ile elde etmesi beklenen toplam birikmiş ödüldür. $\xi_{\langle o_i, \Pi_i^{\mathrm{p}} \rangle}$, q düğümündeki $\langle o_i, \Pi_i^{\mathrm{p}} \rangle$ ikilisinin nitelik değeridir, ve a eyleminin $\langle o_i, \Pi_i^{\mathrm{p}} \rangle$ ikilisine karşılık gelen bir durumda gerçekte ne sıklıkta seçildiğini belirtir.

■

Şekil 5.1, Tanım 5.1 ile verilen tanıma bir örnektir. Bir düğümdeki devam kümesi elemanları, bundan böyle "gözlem" ve "önceki devam kümesi elemanından" oluşan ikililerdir (noktalı oklarla gösterilmiştir). Bu nedenle, TG-GDA veri yapısının bir düğümü, artık aynı gözlemi birden çok elemanda barındırabilir. Ayrıca, bu şekilde, bir düğümdeki her bir devam kümesi elemanı, kök düğümden bir yaprak düğüme kadarki tekil bir rotada bir adım teşkil eder, ki bu özelliğin orijinal GDA veri yapısı ile sağlanması mümkün olmamaktadır. Diğer bir deyişle, herhangi bir devam kümesi elemanı ele alındaığında, o elemandan kök noda ulaşan tek bir rota vardır. Bu özellik, tanımlayacağımız budama mekanizmasına hizmet etmesi açısından önemli bir özelliktir.



Şekil 5.1. Örnek TG-GDA veri yapısı

TG-GDA veri yapısının inşası, orijinal GDA prosedürlerinde (Algoritma 2 ve Algoritma 3) oldukça az değişiklik gerektirmektedir. Olası yararlı alt-tarihçeler oluşturan işlevde (Algoritma 3, satır 1 ile çağrılan işlev), tek değişiklik “durumlar” yerine “gözlemler” kullanılmasıdır. Ağaç güncelleme aşamasında (Algoritma 3) ise, “durum” bilgisi, “gözlem” ile “o gözlemden önceki adım işareti” ikilisi şeklinde özetleyebileceğimiz, Tanım 5.1 içerisinde $\langle o_i, \Pi_i^{\mathrm{p}} \rangle$ olarak gösterilen bilgi ile değiştirilmiştir. Tarihçe ekleme ve eylem seçimi mekanizmalarındaki değişiklikler, takip eden bölümde özetlenmiştir.

5.2 Yanıltıcı Alt-Çözüm Kaldırma Yöntemi ile Genişletilmiş Dizi Ağacı Soyutlama

TG-GDA ile temsil edilen tüm tarihçeler potansiyel olarak belirsizdir (İng. ambiguous). Diğer bir deyişle, kök düğümden bir yaprak düğüme kadar takip edilecek herhangi bir rota, devam kümesi elemanları ayrık durumlara karşılık gelen aynı gözlemleri içerebilir. Bu rotaların ağaçtan silinerek, zamanla yalnız belirsiz olmayan alt-çözüm rotalarının kalması sağlanmalıdır. Bir problem için algısal çakışma oluşturan durumların sayısı çok olsa dahi, birkaç "ayırt edici" gözlem (yani algısal çakışmaya maruz olmayan gözlemler) başarılı olacağı kesin olan tarihçeler için başlangıç noktası teşkil edebilir.

Bu amaç doğrultusunda, GDA yönteminin "tarihçe ekleme" (Algoritma 3, satır 3) ve "eylem seçim" (Algoritma 2, satır 6) işlevlerine bir budama (İng. pruning) mekanizması geliştirilerek entegre edilmiştir.

Öncelikle, tarihçeleri gözlem-eylen dizileri şeklinde saklamaya yarayan, *Yasaklı Alt-Çözüm Deposu (YAD)* (İng. Forbidden Sub-policy Repository, FSR) veri yapısı tanımlanmıştır. Bu tür bir deponun gerçekleştirilmesi için, zaman ve yer açısından farklı etkinlik seviyelerinde, alternatif yöntemler mevcuttur. Bu çalışmada YAD, kabul edilebilir etkinlik seviyesine sahip olduğunu değerlendirdiğimiz bir form olan *ağaç* veri yapısı olarak gerçeklenmiştir. YAD veri yapısının amacı, amaca giden yolda etmeni yanlış yönlendirdiği deneyimlenmiş olan alt-dizileri tutmaktır.

Algoritma 9, YAD veri yapısına daha önceden eklenmiş alt-dizilerin yeniden eklenmesini önleyen, ve Tanım 5.1 ile verildiği şekliyle devam kümesi elemanları arasındaki bağlantıları inşa eden, değiştirilmiş tarihçe ekleme algoritmasını özetlemektedir.

Algoritma 9 $TARİHÇE\_EKLE(h, T)$

Require: h $o_1 a_1 r_2 \dots o_{t-1} a_{t-1} r_t o_t$ formunda bir tarihçedir

Require: T bir TG-GDA'dır

$h' \leftarrow o_1 a_1 \dots o_{t-1} a_{t-1} o_t$ ▷ ödüller çıkarılmıştır

if YAD içindeki herhangi bir rota h' dizisinin bir öneki ise **then** ▷ tarihçe belirsizlik içerir

    exit

end if

$\Pi \leftarrow NULL$ ▷ önceki devam kümesi elemanına bağlantı

$R[t] \leftarrow r_t$ ▷ zaman bilgisi ile indislenmiş azaltımlı ödül dizisi

for $i \leftarrow t-1$ to 1 **do**

    $R[i] \leftarrow r_i + \gamma R[i+1]$

end for

$n_{şimdiki} \leftarrow T$'nin kök düğümü

for $i \leftarrow 1..t-1$ **do**

    if $\exists$ n düğümü: öyle ki $n_{şimdiki}$, n düğümüne $\langle a_i, \psi \rangle$ etiketli bir ayrıtla bağlıdır **then**

        Arttır ψ

        if n düğümü $\langle o_i, \Pi \rangle$ ile etiketlenmiş bir devam kümesi elemanı içerir **then**

    ▷ bulunan devam kümesi elemanının değerlerini güncelle

            Arttır $\xi_{\langle o_i, \Pi \rangle}$

            $R_{\langle o_i, \Pi \rangle} \leftarrow R_{\langle o_i, \Pi \rangle} + \alpha\left(R[i] - R_{\langle o_i, \Pi \rangle}\right)$

        else ▷ n içinde yeni bir devam kümesi elemanı yarat

            n düğümüne $\langle \langle o_i, \Pi \rangle, 1, R[i] \rangle$ ekle

        end if

    else

        $\langle \langle o_i, \Pi \rangle, 1, R[i] \rangle$ içeren yeni bir n düğümü yarat

        $n_{şimdiki}$ düğümünü $\langle a_i, 1 \rangle$ etiketli bir ayrıtla n düğümüne bağla

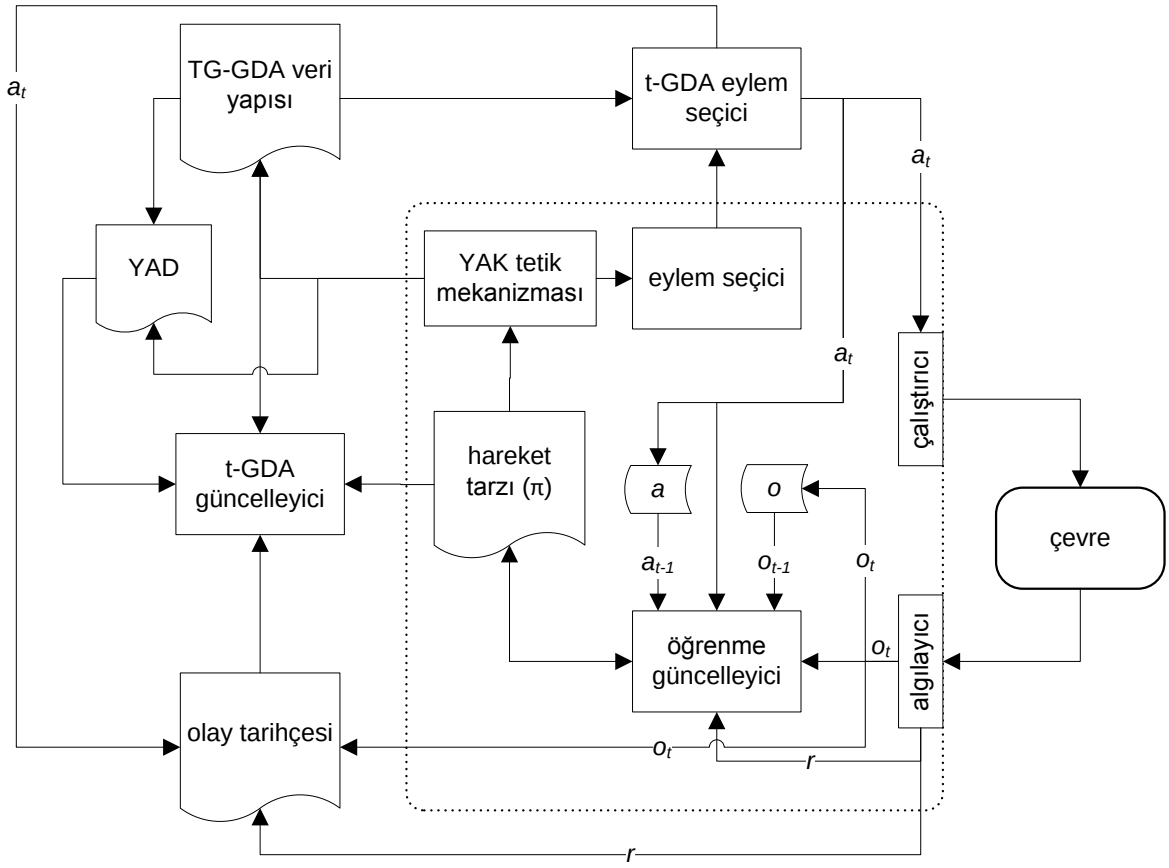
    end if

    $\Pi \leftarrow n$ düğümünde $\langle o_i, \Pi \rangle$ içeren devam kümesi elemanına bağlantı

    ▷ sonraki yineleme için önceki adıma bağlantı hazırla

    $n_{şimdiki} \leftarrow n$

end for



Şekil 5.2. Saklı durumlar içeren takviye öğrenme algoritmasının, t-GDA eklentisi ile birlikte yapısal görünümü

Eylem seçim mekanizması, güncelleme gerektiren son kısımdır. GDA yönteminde kullanılan orijinal eylem seçim mekanizması, GDA ağacı veri yapısını kök düğümden başlayarak, bir yaprak düğüme ulaşılana, veya seçili düğüm o anki durumu yansıtmayana kadar taramaktadır. Her iki durumda da, aktif tercih koşumunun sonlandırılması prosedürü tetiklenmekte, ve akış kontrolü altta çalışan takviye öğrenme algoritmasına iade edilmektedir. Çalışılan problemin sabit ve tamamen öngörülebilir olduğu varsayımı altında, TG-GDA belirsizlikleri yakalamak adına yararlı bir özelliğe sahiptir (Öngörülebilirlik, verilen herhangi bir durum için, bir eylemin her zaman aynı durum geçişine neden olmasıdır. Bir problemin sabit olması, öğrenme sürecinde durum uzayının değişmemesi anlamına gelir.). Bu şartları sağlayan bir problem için aşağıdaki

olaylardan herhangi birinin gerçekleşmesi, ağaçta takip edilen rotanın etmeni yanlış yönlendirdiği anlamına gelir:

- kontrol akışının belirli bir TG-GDA düğümü üzerinde iken, eğer o anki gözlem herhangi bir alt düğümde mevcut değilse (bir sonraki zaman adımında kontrolün tercih uygulamasından çıkacağı anlamına gelir)
- ilgili eylem, alt düğümü olmayan bir devam kümesi elemanı için gerçekleştirilmiş, fakat hedef duruma ulaşılamamışsa.

Yukarıdaki şartların herhangi birinin sağlanması, o noktaya kadar takip edilmiş rotanın içinde en az bir tane algısal çakışmaya neden olan saklı durum olduğunu gösterir. Diğer bir deyişle, bahsedilen varsayımlar altında, TG-GDA üzerinde tanımlı herhanbi bir tercih, ya hedef duruma ulaşmalı, ya da ulaşmıyorsa silinmelidir.

Bu olaylarda biri gerçekleştiğinde, değiştirilmiş eylem seçim mekanizması son işlem gören devam kümesi elemanını tespit eder, o eleman üzerinden ulaşılabilen tüm yaprak devam kümesi elemanlarını bulur, ve son olarak, bulunan yaprak devam kümesi elemanlarından başlayarak, üst devam kümesi eleman bağlantıları üzerinden kök düğüme kadar üzerinden geçilebilen tüm devam kümesi elemanlarını ağaçtan siler. Silinen tüm rotalar, ileride yeniden TG-GDA veriyapısına eklenmelerini engellemek amacıyla, YAD veriyapısına eklenir.

Algısal çakışma problemini TG-GDA üzerinde çözmeye çalışan kullanan bu budama yöntemine, *Yanıltıcı Alt-Çözüm Kaldırma (YAK)* (İng., Misleading Sub-policy Removal) adı verilmiş, GDA ile birlikte tüm metot *tepkisel GDA* (t-GDA) olarak adlandırılmıştır.

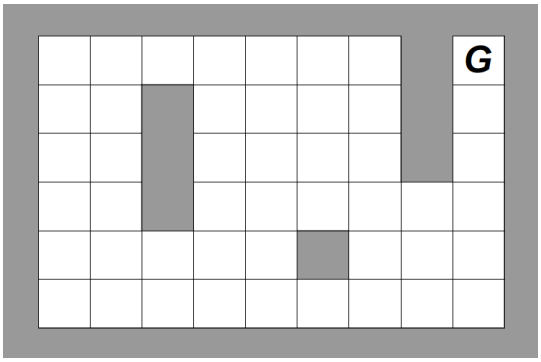
Şekil 2.2'dekine benzer bir yaklaşımla, Şekil 5.2, t-GDA yönteminin, saklı durumlar içeren problemlerde reaktif hareket tarzları üreten takviye öğrenme algoritmalarına yapısal olarak nasıl eklandiğini özetlemektedir. Burada, Şekil 2.2 içerisindeki tüm durumlar (s), gözlemlerle (o) değiştirilmiştir.

Belirtmekte yarar olan son bir nokta ise, çözülmeye çalışılan problemin tam olarak gözlemlenebilir ve öngörülebilir olduğu durumda YAK mekanizmasının hiç bir zaman tetiklenmediği, ve t-GDA yönteminin klasik GDA yöntemi ila aynı hale geldiğidir.

5.3 Deneyler

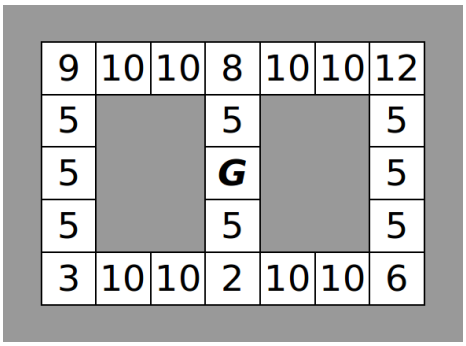
t-GDA yöntemi, iki varsayımı karşılayan problemlerde etkilidir. İlk varsayım, önceki bölümde anlatıldığı gibi, öngörülebilirlik ve değişmeyen bir çevre tanımıdır. Bu nedenle, deney için seçilen tüm problemler öngörülebilir ve değişken olmayan bir doğaya sahiptir. Diğer varsayım ise, seçilen problemin en az bir belleksiz hareket tarzı çözümüne sahip olmasıdır ki, bu varsayım, üstü kapalı olarak SARSA(λ) algoritması tarafından empoze edilmektedir. Yani, öğrenme algoritmasının başarılı olabilmesi için, salt gözlemler üzerinden öğrenilebilecek bir çözümün seçilen problem için mevcut olması gereklidir.

- Seçilen problemlerden biri, Bölüm 4.3 kapsamında tanıtılan küçük gezinti ortamıdır (Mini-hall) (Şekil 4.2) (Littman vd., 1998).



Şekil 5.3. Sutton'ın ızgara dünyası problemi. *G*, hedef durumunu temsil etmektedir (Littman, 1994).

- Biraz daha büyük bir problem, *Sutton'ın ızgara dünyası* problemidir *(İng. Sutton's grid world)* (Sutton 1990). Bu problem, çevreleyen duvarlar ve bazı iç duvarlar bulunan 9×6 boyutlarında bir ızgara günyasıdır (Şekil 5.3). Etmen, herhangi bir ızgara hücresinde başlar, ve her zaman adımında dört pusula yönündeki hücrelerden birine hareket ederek, kuzey-doğu köşedeki hedef hücreye ulaşmaya çalışır. Problem, hedef hücreye ulaştığında $+1$, diğer tüm eylemlerinde ise 0 takviye sinyali alır.
  Problemin orijinali tam gözlemlenebilir olup, kısmi gözlemlenebilir versiyonu Littman (1994) çalışmasında tanıtılmıştır. Kısmi gözlemlenebilirlik, yalnızca komşu 8 hücrenin gözlemlenmesi ile tanımlanır.



Şekil 5.4. McCallum'un labirent problemi. *G* hedef durumunu belirtir. Her sayı, bulunduğu duruma karşılık gelen, etmen tarafından gerçekleştirilen gözleme karşılık gelir (McCallum, 1996).

- Son problem ise, *McCallum'un labirenti* problemidir *(İng. McCallum's maze)* (McCallum, 1996). Bu problem, etmenin yalnızca bulunduğu hücreyi çevreleyen duvarları algılayabildiği, dar geçitlerden oluşan 7×5 büyüklüğünde basit bir labirenttir. Etmen, rastsal olarak dört köşe hücreden birinde oyuna başlar. Eylemler, *Sutton's grid world* probleminde olduğu gibi, dört pusula yönünden birine doğru tek adım atılarak gerçekleştirilir. Ortam etmene, etmen eğer duvara doğru hamle yaparsa -1.0, hedef hücreye ulaşırsa $+5.0$, diğer her eyleme karşılık ise -0.1 takviye sinyali gönderir. Şekil 5.4, durumlara karşılık gelen gözlemleri tanımlayan sayılarla birlikte, problemin bir çizimini göstermektedir.

  McCallum's maze yüksek belirsizlik içeren doğası nedeniyle yaygın olarak kullanılan bir referans problemdir. Bu problemde, kuzey-güney ve batı-doğu doğrultusundaki koridorlar etmen tarafından (sırasıyla 5 ve 10 gözlem tanımları ile) aynı şekilde algılanırlar.

  Bu problemin optimal bir belleksiz hareket tarzı çözümü olmamasına karşın, ϵ-Hırslı gibi stokastik bir eylem seçimi mekanizması kullanıldığında, optimale yakın sonuç üretebilecek hareket tarzları elde edilmesi mümkün olmaktadır.

5.3.1 Deney Düzeneği

Tablo 5.1 test edilmiş problemlerin durum, eylem ve gözlem uzayları cinsinden bir özetini, ve ilgili referans dokümanı listelemektedir.

Deneylerde kullanılan öğrenme ayarları Tablo 5.2 ile verilmiştir. Her problem için, 250 deney gerçekleştirilmiş, ve sonuçlar koşulmarın her bir bölümünün ortalaması alınarak oluşturulmuştur. Tüm problemlerde keşif stratejisi, eylem seçiminde sabit değerli ϵ-Hırslı kullanılarak gerçekleştirilmiştir.

Mini-hall ve *McCallum's maze* problemlerinde başarım kriteri olarak "adım başına ortalama ödül" kullanılmıştır. *Sutton's grid world* problemi için doğru değerlendirme ölçeği ise "adım sayısıdır." Çizimler, görsel netlik elde etmek amacıyla düzleştirme işleminden geçirilmiştir.

Tablo 5.1. Kullanılan problemlerin özellikleri

Problem	**Boyutlar**			**Referans**
	$\|\boldsymbol{S}\|$	$\|\boldsymbol{A}\|$	$\|\boldsymbol{\Omega}\|$	
Mini-hall	13	3	9	(Littman vd., 1998)
Sutton's grid world	47	4	31	(Littman, 1994)
McCallum's maze	23	4	9	(McCallum, 1996)

Tablo 5.2. Öğrenme paremetrelerinin değerleri

Parametre	**Değer**
α	0.01
γ	0.9
λ	0.9
ϵ	0.1
$\psi_{\text{çürüme}}$	0.95
$\xi_{\text{çürüme}}$	0.99
$\psi_{eşik}$	0.01
$\xi_{eşik}$	0.01

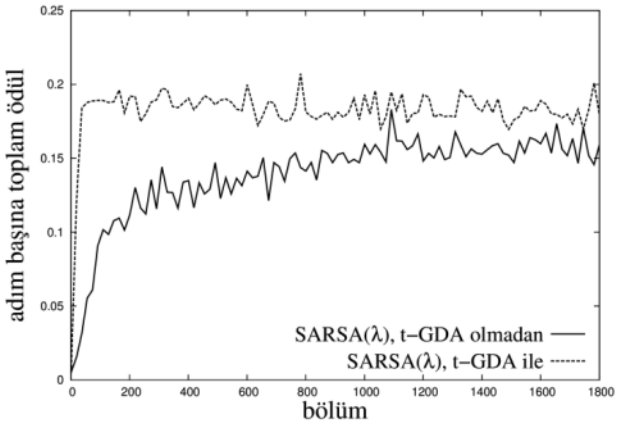
5.3.2 Sonuçlar ve Tartışma

Elde edilen sonuçlar, genel hatlarıyla t-GDA yönteminin seçilen problemlerde iyi sonuçlar verdiğini, ve öğrenme performansını iyileştirdiğini göstermektedir.

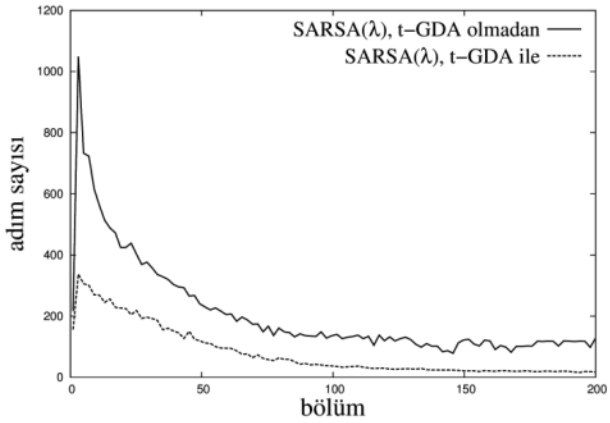
Mini-hall probleminde, t-GDA öğrenme kabiliyetini deneyin ilk aşamalarından itibaren desteklemektedir. Dahası, uzun vadede, öğrenilen soyutlamalar edinilen ortalama ödül miktarını da artırmaktadır. Bunun temel nedeni, rafine edilen soyut eylemlerin gözlemsel belirsizlik içermemesi, dolayısı ile altta yatan takviye öğrenme yönteminin stokastik eylem karar mantığına takılmadan, kısa yoldan etmeni sonuca götürebilmesidir (Şekil 5.5).

Sutton's grid world probleminde de, t-GDA altta çalışan SARSA(λ) algoritmasını destekleyerek bölüm başlarından itibaren öğrenme performansını artırmaktadır (Şekil 5.6). Hedef duruma yakın koridor (bkz. Şekil 5.3), olası tüm çözümlerde bulunması gereken, yararlı bir alt-çözüm örüntüsü oluşturmaktadır. Çünkü etmen, hedefe ulaşmak için –her bir hücresi ayırt edici bir gözleme karşılık gelen- bu koridordan geçmek zorundadır. Bu nedenle etmen, TG-GDA veri yapısının ilk inşasından itibaren yararlı soyutlamalar oluşturmuş durumdadır. Budama işlemi görevini iyi yapmakta, TG-GDA veri yapısını yararlı ve belirsizlikten arınmış alt çözüm rotaları tutacak şekilde güncellemektedir.

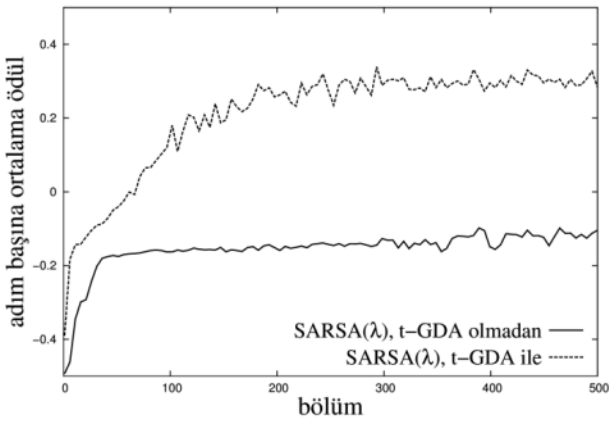
t-GDA yöntemi *McCallum's maze* için, problemin belleksiz bir çözümü olmamasına rağmen, şaşırtıcı seviyede başarılı olmuştur (Şekil 5.7). Elbette, bölümlerdeki başarı, uygulanan eylem seçim mekanizmasının, yani ϵ-Hırslı'nın, stokastik doğasından kaynaklanmaktadır. t-GDA yönteminin bu problem için nasıl daha iyi sonuç verdiğini anlamak için, Şekil 5.8 ile verilen TG-GDA veri yapısı örneğini, Şekil 5.4'teki şekil ve rakamları referans alarak incelemek yararlı olabilir. TG-GDA, çok sayıda bölümün ardından doygunluğa ulaşmış ve rafine bir hale gelmiştir. Bu şekil bize, problemde yararlı ve belirsizlik içermeyen yegane iki belleksiz alt hareket tarzının o2-N-o5-N-G ve o8-S-o5-S-G olduğunu anlatmaktadır ("oi" i sayısı ile etiketlenmiş gözlem, "N" kuzey doğrultusunda bir adım, "S" ise güney doğrultusunda bir adım demektir). Eğer bu rehberlik mevcut olmasaydı, o5 belirsizlik içeren bir gözlem olduğundan, tek bir eylem kullanarak hem o2 hem de o8 üzerinden hedefe ulaşan bir rota çizilemezdi. t-GDA sayesinde, etmen artık o2 ve o8 gözlemlerini ayırt edici gözlemler olduğunun, ve bu gözlemlerden geçerek yararlı alt-çözümlere ulaşılabildiğinin farkındadır.



Şekil 5.5. *Mini-hall* problemi için deney sonuçları



Şekil 5.6. *Sutton's grid world* problemi için deney sonuçları



Şekil 5.7. *McCallum's maze* problemi için deney sonuçları

Tablo 5.3, problemlerde deneyler sırasında toplanan ve her bölüm için ortlaması alınan farklı ölçümleri sıralamaktadır.

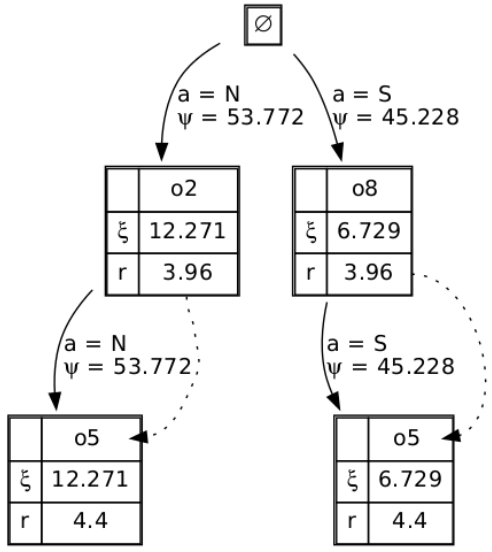
- Ortalama tercih kullanımı ölçümü, tercih işletimleri (yani soyut eylemler) içinde gerçekleştirilen, eylemlerin tüm bölüm boyunca uygulanan eylemlere oranını yüzde olarak vermektedir.

  Mini-hall problemi, diğer problemlerle karşılaştırıldığında zamansal soyutlamadan en fazla yararlanan problem olarak dikkat çekmektedir. Bunun nedeni, hedef duruma yakın yararlı soyutlamanın görece daha uzun olması ile açıklanabilir. Etmen, t-GDA bu soyutlamayı farkedip kullanmaya başladıktan sonra, önemli bir avantaj yakalamış olmaktadır.
- Ortalama TG-GDA düğüm sayısı, bölüm başına ortalama bellek kullanım miktarını göstermektedir.

  McCallum's maze problemi, daha önce belirtildiği gibi, görece daha az soyutlama potansiyeli içeren problemdir, ve problem kümemizde en küçük bellek izine sahip olanıdır. *Sutton's grid world* ise,problem kümemizdeki en çok sayıda ayrık gözleme sahip problem olduğundan, TG-GDA veri yapısında daha derin ve geniş bir dallanma tetiklemektedir.
- Ortalama CPU kullanımı ölçümü, deneyin tamamlanması için geçen ortalama CPU zamanını milisaniye cinsinden vermektedir.

  Genel olarak, t-GDA yöntemi öğrenme adım sayısını belirgin şekilde azaltmayı başardığında, reaktif bir hareket tarzının öğrenilmesi için gereken toplam CPU zamanı kullanımını azaltabilmektedir. Aslında, CPU kullanımındaki olası kazanç, t-GDA tarafından gelen ilave hesap yükü ile öğrenme adım sayısında elde edilecek iyileşme arasındaki dengeye bağlıdır. Örneğin, *Mini-hall* probleminde, t-GDA mekanizmasının ilave yükü, elde edilen öğrenme adımı kazancını aşmış görünmektedir.

t-GDA, etmenin kısa geçmişindeki deneyimlerden yararlanarak, YAK mekanizması ile birlikte, yanlış soyutlamalara yol açabilecek alt-rotaları hızla budamaktadır. Bu arada, yararlı olabilecek soyutlamaları öğrenme sürecine ekleyerek etmenin öğrenmesine destek olmaktadır. Dahası, altta yatan takviye öğrenme algoritması olarak, kategorisinin güçlü bir temsilcisi olduğu için seçilen SARSA(λ) algoritması yerine, küçük değişikliklerle herhangi bir belleksiz takviye öğrenme algoritması tercih edilebilir (Q-Öğrenme, Q(λ), SARSA, TD(λ) gibi).



Şekil 5.8. Bir deneyin sonunda *McCallum's maze* problemi için elde edilen TG-GDA veri yapısı

t-GDA yönteminin iyileştirme gerektiren bazı kısıtlamaları vardır. Herşeyden önce, daha önce belirtildiği gibi, yöntem öngörülebilir ve sabit problemlerde etkilidir. Çevredeki öngörülemezlik faktörü, kolayca TG-GDA veri yapısının tümden budanmasına yol açarak, etkisiz bir soyutlama mekanizmasına yol açabilir.

Bir diğer sorun ise, t-GDA yönteminin problemdeki ara soyutlamaları (yani, algısal çakışma problemine yol açmayan, ancak doğrudan bir hedef duruma ulaşmayan alt-çözümler) yakalayamamasıdır. Diğer bir deyişle yöntem, operasyonel doğası gereği, hedef duruma yakın soyutlamalara odaklanmakta, hedef bölgesinden uzak makroları elemeyi tercih etmektedir.

Tablo 5.3. Bazı deney ölçümlerinin karşılaştırılması

Problem	**ortalama tercih kullanımı (%)**	**ortalama TG-GDA düğüm sayısı**	**ortalama CPU kullanımı (milisaniye)**	
			t-GDA olmadan	**t-GDA ile**
Mini-hall	59	28.83	351	471
Sutton's maze	18	407.65	13065	5005
McCallum's maze	11	5.18	23329	6038

6 YARARLI SONEK BELLEĞİ ALGORİTMASININ İYİLEŞTİRİLMESİ

Bu bölümde, t-GDA yöntemi temel alınarak, gözlemler üzerinden otomatik zamansal soyutlama yöntemi ile Yararlı Sonek Belleği (YSB) (İng. Utile Suffix Memory, USM) algoritmasının nasıl hızlandırılabileceği anlatılmaktadır. Her ne kadar t-GDA yönemi, belleksiz takviye öğrenme yöntemlerini iyileştirmek amacıyla tasarlanmış olsa da, t-GDA yönteminin YSB gibi bir bellek tabanlı yönteme uyarlanması mümkündür. Bu çerçevede, t-GDA yönteminin YSB ağacı veri yapısı ile entegre şekilde çalışması için bir yöntem önerilmekte, bu yöntem kullanılarak, normal şartlarda t-GDA yönteminin hemen budaması gereken ara soyutlamaların nasıl hatırlanıp işe yarar hale getirileceği anlatılmaktadır (Çilden ve Polat, 2014).

Tanımlanan yeni yöntem *bellek tabanlı GDA (b-GDA)* olarak adlandırılmıştır.

6.1 t-GDA Yönteminin Yararlı Sonek Belleği Algoritmasına Uyarlanması

t-GDA yönteminin bazı kısıtlamaları vardır. Bölüm 5.3.2'de açıklandığı gibi, belki de en önemli kısıtlama, yöntemin öğrenmenin başında oluşturduğu bazı ara soyutlamaları, "ya hep ya hiç" yapısı nedeniyle sonradan budamasıdır. Bu nedenle, t-GDA ara soyutlamalar üretemez.

b-GDA yönteminin detaylarına geçmeden önce, YSB algoritmasının sonek belleği mekanizmasını hatırlatmakta yarar vardır.

t zamanında YSB veriyapısındaki bir *tarihçe aşaması* (İng. history instance), $I_t = \langle I_{t-1}, a_{t-1}, o_t, r_t \rangle$ geçişi (İng. transition) ile tanımlanır. YSB veriyapısında bu geçiş, σ ile gösterilen sonekinin, zamanda I_t geçişinin öncülü olan geçiş aşamalarının eylem ve gözlem sonekleri ile örtüştüğü yaprak düğümünde bulunur. Başak bir deyişle, I_t geçişi etiketi $[\ldots o_{t-3} a_{t-3} o_{t-2} a_{t-2} o_{t-1}]$'nin bir soneki olan yaprak düğümündedir. σ ile etikentlenmiş yaprak düğümle ilişkili aşamaların kümesi $I_t(\sigma)$ ile gösterilir. Ağaçta I_t aşamasının ait olduğu yaprak düğüm $L(I_t)$ ile gösterilir.

Örnek olarak, Şekil 2.10 ile verilen YSB ağacını ele alalım. Bu ağaç aslında, $a0, 1b0, c0, 1$ sonekleri ile temsil edilen durum tahminlerinden oluşan bir Q tablosundan başka bir şey değildir. Örneğin, etmenin bir andaki tarihçesinin $l_{current} = [\ldots 0a1b0]$ geçişi ile gösterildiğini varsayalım. Bu durumda, etmenin dahili bilgi durumu, $L(I_{current}) \equiv 1b0$ sonekini temsil eden yaprak düğümdür.

Algoritma 10 $OLASI\_TARİHÇELER\_OLUŞTUR(h)$

Require: h, $o_1 a_1 r_2 \dots o_{t-1} a_{t-1} r_t o_t$ yapısında bir tarihçedir

1: $best[L_{t-1}] \leftarrow L_{t-1} a_{t-1} r_t L_t$ ▷ $best$, mevcut durumdaki en çok ümit vaadeden tarihçe adaylarını tutar

2: $R[L_{t-1}] \leftarrow r_t$ ▷ $R[L_t]$, $best[L_t]$ için toplam birikimli ödül miktarıdır

3: **for** $i \leftarrow t-2$ down to 1 **do** ▷ sondan başa doğru

4: **if** $R[L_i]$ is not set or $r_{i1} + \gamma R[L_{i+1}] > R[L_i]$ **then** ▷ L_i daha önce ele alınmadı ise, veya daha düşük bir getiri tahmini yapıldıysa

5: $best[L_i] \leftarrow L_i a_i r_{i+1} {}^{\circ} best[L_{i+1}]$ ▷ aday tarihçeyi oluştur veya güncelle

6: $R[L_i] \leftarrow r_{i+1} + \gamma R[L_{i+1}]$ ▷ maksimum ödülü güncelle

7: **end if**

8: **end for**

9: let $best_o$ bir çoklu eşleme tablosu olsun (İng. multimap) ▷ tüm tarihçe girişlerini tek gözleme indirge

10: **for** l ile indislenen her bir $best$ elemanı için **do**

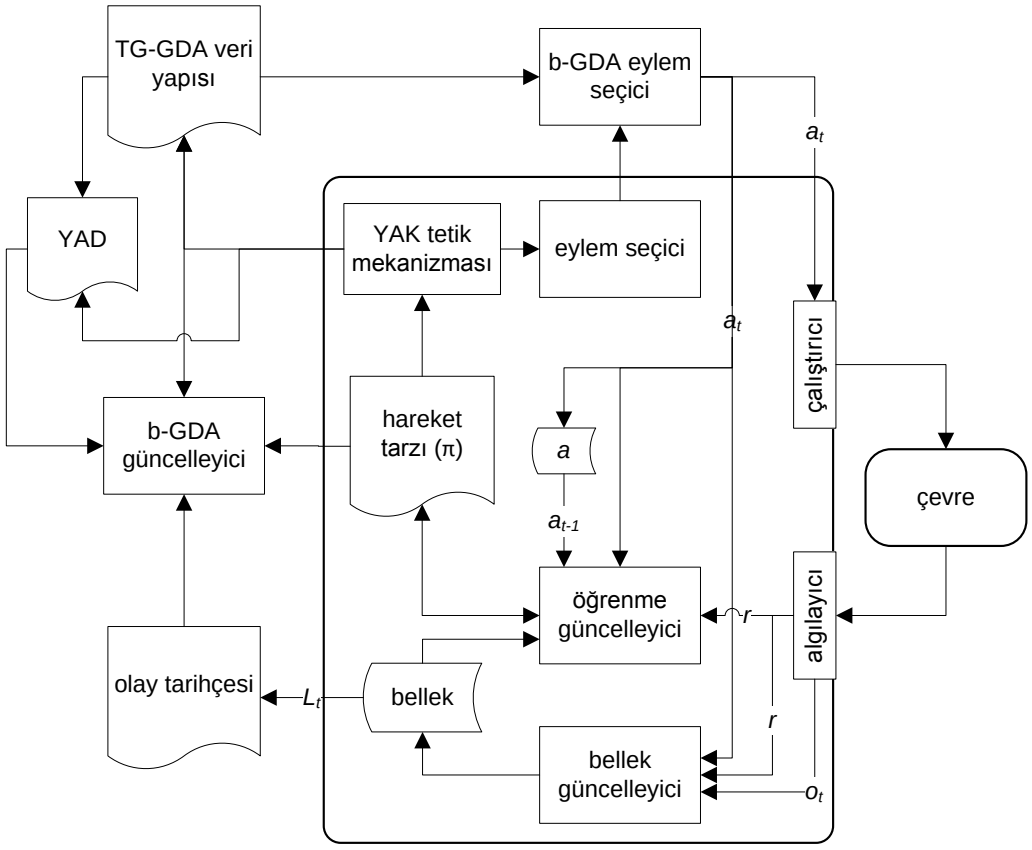
11: $h_o \leftarrow best[l]$ dahilinde tüm geçişlerdeki durum kayıtlarına ω uygula

12: $best_o[\omega(l)] \leftarrow h_o$

13: **end for**

14: **return** $best_o$

Orijinal GDA mekanizmasında, deneyimlere bağlı olarak oluşturulan yararlı tarihçe parçacıklarının ortaya çıkarılması için durum eşitlikleri dikkate alınmaktadır. t-GDA, bu eşitlikleri test etmek için, durumlar yerine doğrudan gözlemleri kullanır, ve sonradan etmeni yanlış yönlendirdiği deneyimlenen rotaları ağaçtan budar. Diğer yandan, YSB veri yapısı gözlem örneklerini ayırt edebilmekte ve bu ayrım öğrenme sırasında süreki güncellenmektedir. YSB yöntemi, çoğu zaman bazı gözlemleri öğrenmenin daha başlarında ayırt edebilmektedir. YSB veri yapısı sayesinde GDA mekanizmaının durum eşitliği testi tarihçe örneklerindeki YSB durumları üzerinden gerçekleştirilebilir. Böylece, öğrenmenin daha başlarında, olası tarihçe üretimi aşamasında *sonekler*, yani YSB veriyapısının yaprak düğümleri kullanılarak, öğrenme etkinliği artırılabilir.



Şekil 6.1. YSB algoritmasının b-GDA yöntemi ile birlikte yapısal gösterimi

Bu amaçla, t-GDA yönteminin olası tarihçe üretimi ile ilgili prosedüründeki durum eşitliği testi, salt gözlem karşılaştırması yerine YSB yaprak düğümü karşılaştırması ile gerçekleştirilecek şekilde yeniden düzenlenmiştir. Algoritma 10, değiştirilen tarihçe üretim prosedürünü göstermektedir. L_i, $L(I_i^h)$ notasyonunun kısaltılmış şekli olup, burada I_i^h, h tarihçesinin i zaman adımındaki örneğini belirtir. $\omega(L)$, L ile temsil edilen örnek için edinilmiş gözlemi anlatır. Dikkat edilirse, TG-GDA veri yapısındeki devam kümesi elemanları, soneklerden veya gözlem-eylem dizilerinden değil, hala sadece gözlemlerden oluşmaktadır. Bu durum, b-GDA mekanizmasının, özellikle öğrenmenin başlarında, daha sık devreye girmesine neden olmaktadır. t-GDA yönteminin diğer tüm mekanizmaları değişmeden korunmuştur.

b-GDA ile, daha derin TG-GDA ağaçları üretilebilmekte ve tercih işletimi amacıyla kullanılabilmektedir. Hedefe giden yolda yararlı olabilecek tekrarlayan gözlem dizilerinin yakalanarak kullanılması olasılığı artmaktadır. Bu durum, orijinal GDA veya t-GDA kullanıldığında kısmi gözlemlenebilir problemler için geçerli olamamaktadır. Ortalamada, uzun

vadede tercih yollarının sayısı arttığı için (çünkü, daha çok sayıda TG-GDA rotası hedefe ulaşmakta başarılı olacağından, budama mekanizması daha isteksiz çalışacaktır), altta koşturulan takviye öğrenme algoritmasının YSB olması durumunda, b-GDA yönteminin t-GDA yöntemine göre daha iyi sonuçlar vermesini beklemek yanlış olmaz.

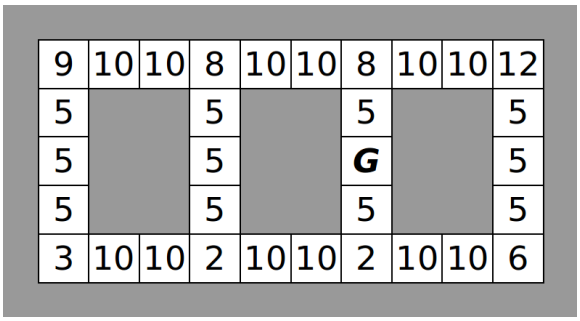
Eğer problem tam gözlemlenebilir ve öngörülebilir ise, ve YSB saçak derinliği parametresi sıfır olarak belirlenirse, b-GDA yöntemi orijinal GDA yöntemi ile aynıdır. Bu genelleştirme özelliği, b-GDA yöntemini önemli kılan özelliklerden biridir.

YSB algoritmasının üzerine inşa edilmiş b-GDA yönteminin yapısal görünümü Şekil 6.1 ile verilmiştir. TG-GDA ve YAD veri yapıları ile YAK tetik mekanizması t-GDA yönteminden miras alınmıştır. Bu yöntemde farklı olarak gözlem akışları, YSB bellek yapısının desteği ile, tarihçe üretim prosedürünü zenginleştirmektedir.

6.2 Deneyler

t-GDA gibi, b-GDA da öngörülebilir nitelikteki problemler için etkindir. YSB algoritmasının hem t-GDA ile, hem de b-GDA ile birlikte çalıştırıldığındaki etkinliği, tek başına çalıştırıldığında alınan sonuçlarla karşılaştırılmış, testlerde aşağıda sıralanan öngörülebilir doğaya sahip problemler kullanılmıştır:

- Kullanılan problemlerin ilki yine, Bölüm 4.3'te tanıtılan *küçük gezinti ortamıdır* (*Mini-hall*) (Littman vd., 1998) (Şekil 4.2). Bu problem, en basit problemimiz olmakla kalmayıp, t-GDA yönteminin doğasının tüm avandajını kullanabileceği bir yapısal özelliğe sahipdir: tek bir zamansal soyutlama paketi vardır, ve hedef durumun hemen öncesindedir.
- *Sanal ofis* (*Virtual Office*) (Dung vd., 2007), problem kümemizin sonraki üyesidir. Problem karakteristikleri Bölüm 4.3'te kısaca açıklanmıştır (Şekil 4.4). Bu problemin önemi, görece büyük olması, birden fazla hedef ve darboğaz durumuna sahip olması, dolayısı ile hiyerarşik bir çözüme uygun bir yapıya sahip olmasıdır.
- Sıradaki problem *McCallum'un labirenti (McCallum's maze)* problemidir ve Bölüm 4.3'te tanıtılmıştır (McCallum, 1996) (Şekil 5.4). *Mini-hall* probleminde olduğu gibi, hedef kümesinin yakınında yararlı zamansal soyutlama olanakları vardır, ve bu soyutlamaların t-GDA tarafından yakalanabildiği bilinmektedir (Şekil 5.8). Ancak, problemde başka soyutlama olanakları da vardır ve b-GDA yönteminin bu t-GDA tarafından keşfedilemeyen bu soyutlamaları da yakalayarak fark yaratması beklenmektedir.



Şekil 6.2. *McCallum's maze* probleminin, b-GDA yönteminin testi için özel olarak genişletilmiş versiyonu

- Son olarak, *McCallum's maze* problemine, b-GDA yönteminin etkinliğini test etmek için özel olarak tasarlanmış bir genişletme önerisi Şekil 6.2 ile sunulmuştur. Bu yeni versiyonda geçiş ve gözlem mantığı, orijinal problemdeki ile aynıdır. Temel fark ise, orijinal problemin aksine, hedef duruma yakın ayırt edici bir gözlemin bulunmamasıdır. Orijinal problemde 2 ve 8 etiketleri ile belirtilen gözlemler, karşılık gelen durumları tek başlarına ayırt etmeye yeterken, problemin bu versiyonunda 2 ve 8 gözlemleri ile algılanabilecek ikişer durum bulunmaktadır. Yani, bu problem için başlangıç durumları hariç (dört farklı köşedeki durumlar) tüm durumlar, gözlemsel belirsizliğe maruzdurlar. Diğer bir deyişle, hedef duruma yakın ayırt edici soyutlama paketleri mevcut değildir. Bu durumda, soyutlama prosedürünün hızlanma sağlaması, ara soyutlamaları yakalama becerisine bağlı olacaktır.

6.2.1 Deney Düzeneği

Tablo 6.1, test edilen problemleri, durum, eylem ve gözlem uzaylarının büyüklüğü cinsinden özetlemekte, ve ilgili kaynak yayını vermektedir.

Deneylerde kullanılan öğrenme parametrele değerleri Tablo 6.2 ile verilmiştir. Tüm problemlerde eylem seçim mekanizması olarak ϵ-Hırslı kullanılmış, ϵ değeri sabit alınmıştır. YSB algoritmasında her zaman adımının ardından McCallum'un tanımladığı şekliyle Kolmogorov-Smirnov (K-S) testi kullanılmıştır (McCallum, 1996).

Her bir problem için 100 deney yapılmış, her koşumun bölümleri üzerinden ortalamalar alınmıştır. Başarı kriteri olarak "adım başına ortalama ödül" kullanılmıştır. Çizimler, görsel netlik elde etmek amacıyla düzleştirme işleminden geçirilmiştir.

Tablo 6.1. Kullanılan problemlerin özellikleri

Problem	Boyutlar			Referans
	$\|S\|$	$\|A\|$	$\|\Omega\|$	
Mini-hall	13	3	9	(Littman vd., 1998)
Sutton's maze	47	4	31	(Littman, 1994)
McCallum's maze	23	4	9	(McCallum, 1996)
McCallum's maze extended	32	4	9	-

6.2.2 Sonuçlar ve Tartışma

Genel olarak sonuçlar, seçilen problemler için YSB algoritmasının t-GDA ve b-GDA yöntemlerinden yararlanabileceğini, öğrenme performansını arttırabileceğini göstermektedir.

Mini-hall probleminde, her iki soyutlama mekanizması da öğrenmeyi daha deneyin ilk bölümlerinden başlayarak hızlandırmıştır (Şekil 6.3). b-GDA, deneyin başında daha iyi performans göstermekte, ancak sonrasında t-GDA'nın gerisine düşmektedir. Bu durum, b-GDA yönteminin bazen kısa öngörülü olmasından kaynaklanmaktadır: Öğrenmenin başlarında, b-GDA henüz görece olgunluktan uzak bir YSB yapısını durum tahmini için kullanmakta, bu da gözlemler üzerinden anlamlı bir ayrım yapmak için bazen yeterince detaylı olmayabilmektedir. Ayrıca b-GDA, olgun olmayan tahminler kullanıldığında, bazen olması gerekenden çok daha uzun gözlem-eylem dizileri üreterek hedef duruma ulaşmayı sağlayabilmektedir. Bunun da nedeni, olgun olmayan tahminlerle oluşturulan rotaların, aynı gözlemle sonlanan gereksiz rota döngüleri içerebilmesidir. Bu uzun rotalar, öğrenmenin başlarında etmeni doğru yönlendirse de, b-GDA döngüleri tespit edemediğinden, bu gereksiz tercih adımlarını atmak için etmen fazladan zaman harcamaktadır. *Mini-hall* probleminde de bu sorun kendini göstermiştir.

Öğrenilmiş soyutlamalar her iki yöntemde de uzun vadede etmenin ortalama ödülünde artış sağlamıştır. Rafine edilmiş soyut eylemler gözlemsel belirsizlik içermemekte, böylece YSB algoritmasında kullanılan stokastik eylem karar mantığına (ϵ-Hırslı) destek olmaktadırlar.

Tablo 6.2. Öğrenme parametrelerinin değerleri

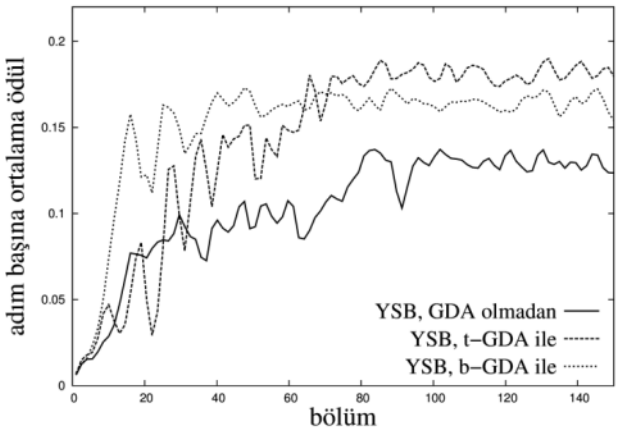
Parametre	**Değer**
α	0.125
γ	0.9
ϵ	0.1
K-S test eşik değeri	0.01
YSB için maksimum saçak derinliği	4
$\psi_{\text{çürüme}}$	0.95
$\xi_{\text{çürüme}}$	0.99
$\psi_{eşik}$	0.01
$\xi_{eşik}$	0.01

Virtual Office probleminde, b-GDA yöntemi t-GDA yöntemine göre belirgin şekilde daha iyi sonuç üretmiştir (Şekil 6.4). Problemin geçiş işlevi dinamikleri incelendiğinde, iki hedef duruma belirsizliklerle dolu uzun gözlem-eylem dizileri aracılıyla erişilebildiği görülür. b-GDA, uzun vadede bu tip soyutlamaları ortaya çıkarmakta daha başarılı olmuştur.

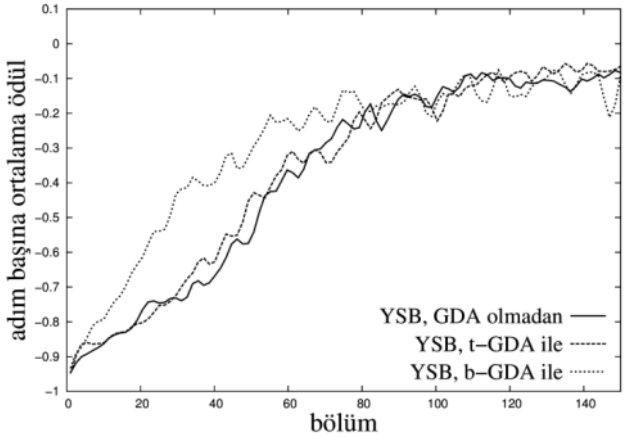
Her iki soyutlama mekanizması da YSB algoritmasının gücünü göstermek için özel olarak tasarlanmış *McCallum's maze* problemi için iyi sonuçlar vermiştir (Şekil 6.5). Diğer bir deyişle, bu problem YSB algoritmasının kendini kanıtladığı bir problemdir. Bu problemde dahi, t-GDA ve b-GDA yöntemleri YSB algoritmasının performansını başarılı bir şekilde arttırmıştır. Ancak iki soyutlama yöntemdeki performans artışları arasında belirgin bir fark yoktur.

McCallum's maze extended probleminde ise, daha önce açıklanan tasarım özelliği nedeniyle t-GDA yöntemi etkinliğini kaybetmiştir. Dahası, YSB algoritmasının tek başına çalıştırılması ile t-GDA eklentisi kullanılarak çalıştırılması arasında neredeyse hiç fark yoktur. Bu problemde b-GDA, özellikle öğrenmenin başlarında, diğer yöntemlerden daha iyi sonuçlar vermiştir.

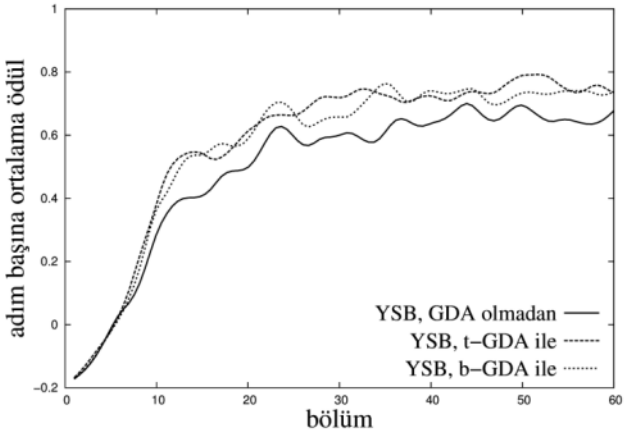
Tablo 6.3 ile sıralanan ölçümler b-GDA yönteminin t-GDA'ya göre daha çok sayıda soyutlama üretip kullandığı argümanını desteklemektedir. b-GDA, ortalamada sadece daha çok sayıda TG-GDA düğümü üretmekle kalmamış (yaklaşık 150 katı), aynı zamanda bu soyutlamaları doğru şekilde kullanmak suretiyle faydaya çevirmiştir. İlginç bir sonuç değeri *McCallum's maze extended* probleminde t-GDA yönteminin, beklendiği şekilde, hemen hemen hiç soyutlama kullanmamış olması olarak not edilebilir.



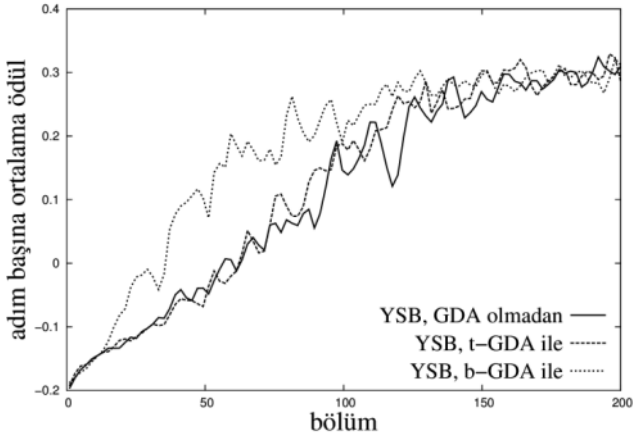
Şekil 6.3. *Mini-hall* problemi için deney sonuçları



Şekil 6.4. *Virtual Office* problemi için deney sonuçları



Şekil 6.5. *McCallum's maze* problemi için deney sonuçları



Şekil 6.6. *McCallum's maze extended* problemi için deney sonuçları

Tablo 6.3. Bazı deney ölçümlerinin karşılaştırılması

	ortalama tercih kullanımı (%)		**ortalama TG-GDA düğüm sayısı**	
Problem	**t-GDA ile**	**b-GDA ile**	**t-GDA ile**	**b-GDA ile**
Mini-hall	43	68	26.46	65.80
Sutton's maze	6	38	3.09	515.49
McCallum's maze	18	51	5.97	33.18
McCallum's maze extended	1	43	1.43	207.01

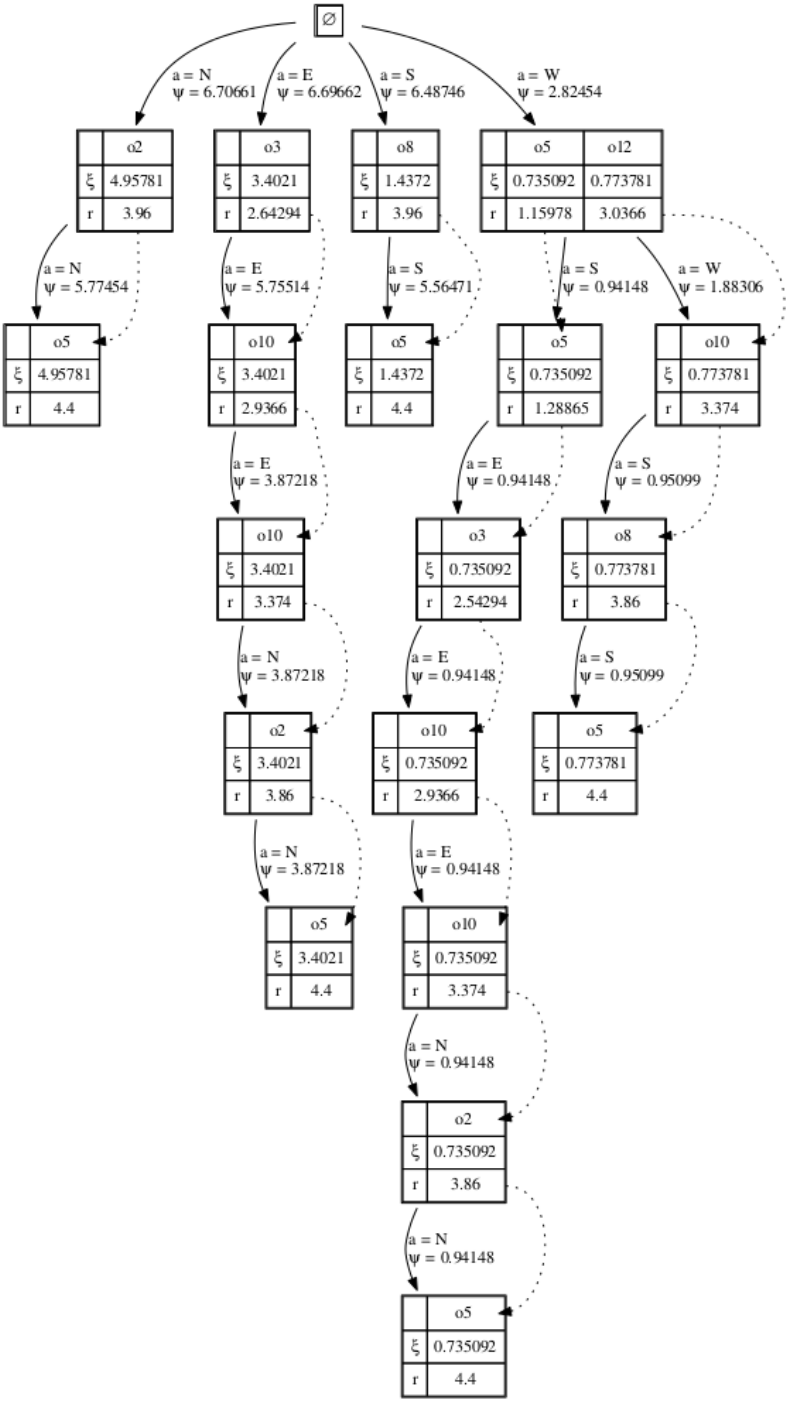
Deneylerde kullanılan ortalama toplam CPU zamanları Tablo 6.4 ile verilmiştir. Neredeyse tüm deneylerde, GDA kullanılmayan YSB uygulaması, soyutlama kullanıldığı duruma göre daha çok zaman almıştır. Aslında bu sonuçlar kullanılan YSB öğrenme parametre değerleriyle ve problemin doğası ile yakından ilintilidir. Bu problemler ve bu ayarlarla, YSB ağacının yönetim maliyeti (istatistiksel testler, saçak düğümlerin terfisi, tarihçe bağlantıları vb.), GDA ağacının yönetim maliyetini aşmış görünmektedir. Böylece, t-GDA ve b-GDA sayesinde toplam adım sayısı cinsinden elde edilen iyileşme, harcanan toplam CPU zamanını önemli oranda düşürmektedir. Doğal olarak, bu sonuçlar, başka bir YSB öğrenme ayarı kullanıldığında farklı olabilirdi. Yine de, elde edilen sonuçlar, t-GDA ve b-GDA yöntemlerinin CPU zamanı cinsinden önemli kazanç sağlayabileceğini göstermektedir.

McCallum's maze problemi için örnek bir TG-GDA veri yapısı Şekil 6.7 ile gösterilmiştir (gözlem etiketleri için Şekil 5.4'e bakınız). Noktalı oklar önceki devam kümesi elemanına bağlantıyı temsil etmektedir. Görüldüğü gibi, bu ağaç t-GDA tarafından üretilen TG-GDA ağacından (Şekil 5.8) çok daha büyük ve derin olup, ara soyutlamalara da yer vermektedir. Ancak, bu özelliğinin bir yan etkisi olarak, elde edilen uzun tercihlerin içindeki bazı gereksiz döngüler bulunmakta, bu göndülerin tespiti yapılamamaktadır.

YAK, GDA yönteminin yanlış soyutlamalara yol açabilecek alt rotaları, etmenin deneyimlerine dayanarak hızlıca budamasını sağlayan bir yöntemdir. t-GDA potansiyel olarak yararlı soyutlamaları bulup test ederek etmeni başarı ile desteklerken, b-GDA, özellikle t-GDA'nın gözden kaçırdığı ara soyutlamaları hedef alarak, bulunan tercihin kalitesi üzerinde bir iyileştirme vaat etmektedir.

Tablo 6.4. Milisaniye cinsinden ortalama CPU kullanımı

Problem	**YSB**	**YSB, t-GDA ile**	**YSB, b-GDA ile**
Mini-hall	2498	2792	1917
Sutton's maze	46621	43241	34146
McCallum's maze	997	823	755
McCallum's maze extended	234434	238939	106960



Şekil 6.7. TG-GDA veriyapısının, *McCallum's maze* problemi için çalıştırılan b-GDA yönteminin ara bir adımındaki durumuna örnek. En sağ kanattaki rota (o12-W-o10-W-o8-S-o5-S) etmeni yanlış yönlendirmektedir (o10 iki kez gözlemlendiği için). Dolayısı ile bu rota ilk işletiminin ardından ağaçtan budanacaktır. Ağaçtaki diğer tüm rotalar kalacaktır.

7 TAKVİYE ÖĞRENME İÇİN HAFIZA TASARRUFLU BÖLÜMLENMİŞ SOYUTLAMA

Bu bölümde, GDA yönteminin bellek kullanım performansını iyileştirmek maksadıyla bölümlendirilmiş MKS modelini esas alan bir çözüm önerilmektedir (Şahin vd., 2015). Bu çözümün, kısmi gözlemlenebilir problemler için de genişletilme potansiyeli mevcuttur.

Klasik takviye öğrenme teknikleri, çoğu zaman büyük durum uzayına sahip problemler için yetersiz kalırlar. Çünkü bu yaklaşımlar, bir çözüm üretebilmek için bütün olası durumların hesaplanmasını gerektirirler. Fakat durumlar, çevredeki faktörlerin değerlerinin listesiyle ifade edilebildiğinde, durum uzayı, faktör sayısına bağlı olarak üstel olarak büyür. Buna *çok boyutluluğun laneti (İng. curse of dimensionality)* denir. Yeni ifade biçiminin avantajlarını kullanarak çevreyi daha kompakt bir şekilde modellemek mümkündür. Bölümlenmiş Markov Karar Süreçleri (BMKS) bu amaç için kullanılır ve bu yapıya bölümlenmiş takviye öğrenme metotları uygulanarak yeni modelin faydalarından istifade edilir. Fakat bu yaklaşım da büyük ölçekli problemler için yeterli olmayabilir. Değerlendirilmesi gereken durum ve eylemlerin sayılarının çok büyük olması nedeniyle öğrenme süreci fazla zaman ve kaynak gerektirir. Bu başlıkta, bu sorunun çözümü için bölümlenmiş kompakt bir yapı önerilmektedir.

Zamansal soyutlamaların otomatik olarak tespit edilip kullanılmasının öğrenme hızını artırdığı ispatlanmıştır. Bu yolla, problemin farklı bölümlerinde tekrar eden şablonlar bulunup ortak bir hareket tarzı hepsinde uygulanarak aynı çözümü tekrar tekrar hesaplamanın önüne geçilmektedir. Genişletilmiş dizi ağaçları (GDA) algoritması, durum ve eylemlerin tarihçelerini, sıklıkla kullanılan şablonları yapısal bir şekilde kaydedip, altta yatan takviye öğrenme algoritmasına alternatif eylemler öneren bir otomatik zamansal soyutlama tespit tekniğidir. Bu çalışmada, GDA'lara dayanan, birbirlerini takip eden durumlardaki değişken değerlerinin farklılıklarını kullanan, bölümlenmiş bir otomatik zamansal soyutlama metodu önerilmektedir. Çalışmadaki amaç, durum ve eylem tarihçelerini daha kompakt bir sekilde saklayıp yüksek hafıza kullanımından kaçınmaktır. Metodun önemli ölçüde hafıza kazanımı sağladığı yaygın bir şekilde kabul gören problemler üzerinde gösterilmiştir.

Tanımlanan yeni yöntem bölümlenmiş *GDA (BGDA)* olarak adlandırılmıştır.

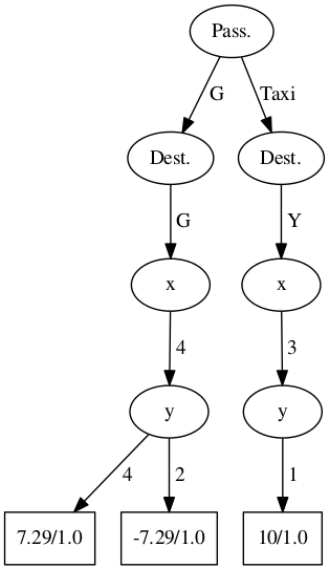
7.1 Bölümlenmiş Takviye Öğrenmede Bölümlenmiş Genişletilmiş Dizi Ağacı Kullanımı

7.1.1 Karar Ağaçları

Karar ağaçları, bir çevreyi, durum değişkenleri arasındaki koşullu olasılıkları kullanıp etiket-değer ikilileri biçiminde göstermek için kullanılan veri yapılarıdır. Bir karar ağacında üç çeşit eleman bulunur:

- *iç düğümler,* değişkenler üstündeki kontrolleri tanımlar.
- *ayrıtlar,* başladığı düğümlerde gösterilen değişkenlerin değerlerini kontrol edip, onları çocuk düğümlerden birine bağlar.
- *yaprak düğümler,* kök düğümden itibaren uygulanan bütün kontrolleri geçerek elde edilen olasılık değerlerini gösterir.

Karar ağaçları, BMKS'leri modellemek için kullanılabilir. Bu yapıda, iç düğümler değişkenleri gösterirken, ayrıtlar bu değişkenlerin kendi değer uzaylarından aldıkları değerleri verir. Yaprak düğümler ise kök düğümden başlayıp değişken değerlerini takip edip elde edilen durumlar için ödül ve niteliklilik değerlerini tutar. Örnek olarak, taksi probleminin (Dietterich, 2000) (Şekil 7.3(b)) $\{Yolcu, G\ddot{u}zergah, x, y\}$ değişkenleriyle gösterildiği bir modelde, Şekil 7.1, $s_1 =< G, G, 4, 4 >$ $s_2 =< G, G, 4, 2 >$ ve $s_3 =< Taksi, Y, 3, 1 >$ örnek durumlarını bir karar ağacında göstermektedir.



Şekil 7.1. BMKS durumlarını gösteren karar ağacı örneği.

7.1.2 Bölümlenmiş Olaylar Tarihçesi

Faydalı durum-eylem alt dizilerini kompakt bir şekilde tanımlayıp bunlara erişebilmek için, bölümlenmiş modele uygun uygun hale getirmek amacıyla GDA yapısında bazı değişiklikler yapılmıştır. Bu değişikliklerin çıkış noktası, bölümlenmiş bir çevrede, her eylemin değişkenlerin sadece küçük bir bölümünü etkilemesidir. Bu nedenle, bir olay tarihçesinde, ilk durum haricindeki bütün durumlar, bir önceki durum ile eylemden etkilenen değişkenlerin yeni değerlerini kullanarak tanımlanabilir. Eğer bir olay tarihçesi klasik durum modelinde $h: s_1 a_1 r_2 s_2 a_2 \dots s_i a_i r_{i+1}$ şeklinde gösterilirse, bölümlenmiş modelde durumlar değişken dizisiyle değiştirilip aynı tarihçe $h': (x_{1_1}, \dots, x_{k_1}) a_1 r_2 (x_{1_2}, \dots, x_{k_2}) a_2 \dots (x_{1_i}, \dots, x_{k_i}) a_i r_{i+1}$ şeklinde ifade edilebilir. Bu gösterimde, x_{j_i}, klasik gösterimdeki i'inci durumun j'inci değişkeninin değerini vermektedir. Eğer t anında s_t durumundan s_{t+1} durumuna geçerken değişen durum değişkenlerinin yeni değerlerinin $diff(t)$ işlevi kullanılarak alınabileceği varsayılırsa, h' daha kompakt olarak $h': (x_{1_1}, \dots, x_{k_1}) a_1 r_2 (diff(2)) a_2 \dots (diff(i)) a_i r_{i+1}$ şeklinde tanımlanabilir. $diff(t)$ işlevi, birbirini izleyen durumlar aynı olduğunda boş küme dönmektedir.

Aşağıda, taksi probleminde (Şekil 7.3(b)) bölümlenmiş olaylar tarihçesini gösteren bir örnek verilmiştir. Bu problemde taksi şoförü olan etmen, bir yolcuyu önceden tanımlanmış *G* noktasından alıp *Y* noktasına götürmeye çalışmaktadır (Dietterich, 2000). Sol alt köşedeki nokta (1,1) koordinatıyla gösterildiğinde, olası bir bölümlenmiş olaylar tarihçesi,

$$h_1 : < G, G, 4, 4 >, batı, 0, (x: 5), kuzey, 0, (y: 5), al, 20, (Yolcu: Taksi, Güzergah: Y)$$

şeklinde gösterilebilir. Problemdeki durumları tanımlamak için kullanılan değişkenler şu şekilde listelenebilir:

- *Güzergah:* Etmenin şuanki güzergahıdır. Değer kümesi $Dom(Güzergah): \{R, G, B, Y\}$'dir.
- *Yolcu:* Yolcunun şuan bulunduğu noktadır. Değer kümesi $Dom(Yolcu): \{R, G, B, Y, Taksi\}$'dir.
- *x:* Etmenin şuanki yatay koordinatıdır. Değer kümesi $Dom(x): \{1,2,3,4,5\}$'dir.
- *y:* Etmenin şuanki dikey koordinatıdır. Değer kümesi $Dom(y): \{1,2,3,4,5\}$'dir.

Bu bölümdeki örneklerin hepsinde, değişkenlerin durumlardaki gösterim sırası aynı kabul edilmiştir. Yukardaki örnekte h_1'in başlangıç durumu $< G, G, 4, 4 >$ alındığında, etmen $t = 1$ anında batıya gidip sadece x değişkeninin, sonrasında kuzeye gidip sadece y değişkeninin değerini değiştirmiştir. Yolcunun bulunduğu noktaya ulaşılıp yolcu taksiye alındığında ise

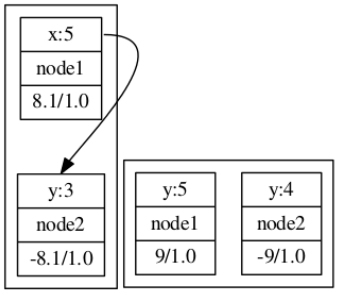
$+20$ ödül kazanıp yolcunun güncel lokasyonunu $Taksi$'ye, yeni güzergahı da yolcunun indirileceği nokta olan Y'ye çevirmiştir.

7.1.3 Durum Fark Çizgesi

BGDA'larda, her zaman adımında, durumlar arasındaki farkları kaydedebilmek için, yönlü döngüsüz çizgeler kullanılmaktadır. Bu yapıda, her çizge düğümü $< x_i, v_i, D_i >$ değişkenler gurubundan oluşur, x_i değişken etiketini ve v_i onun değerini gösterir. D_i ise $< l_m, R_m, \xi_m >$ değerler listesinden oluşan bir liste içerir. Listedeki l_m değeri, sözkonusu olaylar tarihçesinin başlangıç durumunun ağaçta gösterildiği noktaya işaret eder. R_m ve ξ_m ise orijinal GDA yapısındaki ödül ve niteliklilik değerleridir. Fakat klasik algoritmada her durum sadece bir tane ödül ve niteliklilik değeri içerirken, bölümlenmiş yapıda her durum fark kümesi bu değerlerden birden fazla içerebilir. Yukardaki örnekteki h_1 geçmişine ek olarak yeni bir $h_2 : < G, G, 4, 2 >, batı, 0, (x: 5, y: 3), kuzey, 0, (y: 4), al, -10, \emptyset$ verildiğini varsayalım. Bu örnekte, başlangıç anında etmen batıya gitmesine rağmen h_1'dekinin aksine, aynı anda y değişkeninin de değeri değişmektedir. Öngörülemez çevrelerde bu tür davranışlar gözlemlenmesi mümkündür.

Durum fark çizgeleri, durumlar arasındaki farkları tuttuğundan, bölümlenmiş GDA'larda bu yapılar ikinci adımdan itibaren kullanılmaya başlar. Çünkü bölümlenmiş olaylar tarihçesinde bütün durumlar dizisini çıkarabilmek için başlangıç durumunun olduğu gibi tutulması gerekmektedir. Bu nedenle, başlangıç durumları Şekil 7.1'de gösterildiği gibi karar ağaçlarında saklanmaktadır.

h_1 ve h_2 nin eylem sıraları incelendiğinde, aynı eylemlerin aynı sırada uygulandığı görülebilir. Bu durumda, klasik GDA'larda olduğu gibi, BGDA'larda da her adımdaki durum (bölümlenmiş yapıda durum fark kümeleri) aynı ağaç düğümünde tutulmaktadır. Şekil 7.2'de h_1 ve h_2'nin $t = 2$ ve $t = 3$ anlarındaki düğüm elemanlarının içinde bulunduğu durum fark çizgeleri verilmiştir.



Şekil 7.2. $t = 2$ ve $t = 3$ anları için h_1 ve h_2'nin ilgili elemanlarını gösteren durum fark çizgeleri.

Durum fark kümeleri oluşturulurken değişkenlerin sıralarının aynı olmasına dikkat edilmelidir. Çünkü kümede birbirini takip eden elemanların eklendiği çizge düğümleri arasında bağlantı kurulur. Döngüsüz bir çizge yapısı elde edebilmek için her ekleme işleminde, düğümler arasındaki bağlar $< x_i, v_i >$'nin içinde bulunduğu çizge düğümünden $< x_{i+1}, v_{i+1} >$'nin bulunduğu çizge düğümüne olacak şekilde oluşturulur. Algoritma 11'de, verilen durum fark kümesinin varolan bir durum fark çizgesine eklenme adımları gösterilmiştir.

Algoritma 11 $DURUM\_FARKI\_EKLE(G, L, S, R)$

Require: S, eylem uygulandıktan sonra değerleri değişen durum değişkenlerinin $\{x_1 : d_1, \ldots, x_k : d_k\}$ yapısındaki kümesidir

Require: G, S'nin ekleneceği durum fark çizgesidir

Require: L, üzerinde işlem yapılan tarihçenin başlangıç durumunu tutan karar ağacının ilgili yaprak düğümüne işaret eder

Require: R, durumun toplam birikimli ödül miktarıdır

1: $current \leftarrow nil$ ▷ $current$ şu anki çizge düğümü
2: **for** $i \leftarrow 1..k$ **do**
3: **if** $\exists$ $\mathrm{x_i : d_i}$ etiketine sahip bir n düğümü var ise **then**
4: $current \leftarrow n$
5: **else**
6: G'ye $n(\ \mathrm{x_i : d_i})$ düğümünü ekle
7: $current \leftarrow n$
8: **end if**
9: **if** $\mathrm{i} < k - 1$ ve $\nexists$ bir $current$ komşuluğu $\mathrm{x_{i+1} : d_{i+1}}$ etiketine sahip ise **then**
10: $current$ 'e $\mathrm{x_{i+1} : d_{i+1}}$ içeren komşu ekle
11: **end if**
12: **end for**
13: **İf** $current$ is nil **then**
14: G'ye $n(\emptyset)$ düğümünü ekle
15: $current \leftarrow n$
16: **end if**
17: $current$ değişkeninin Dkümesine $< L, R, 1.0 >$ ekle

Her yeni bir olaylar tarihçesi BGDA'ya eklendikten sonra bütün ağaç düğümleri, niteliklilik değerlerini çürüterek ve verilen eşik değerinden düşük niteliklilik değerine sahip elemanları silinerek güncellenir. Kök düğümün çocukları karar ağaçlarından oluştuğundan bu düğümlerde yaprak silme algoritması uygulanır. Durum fark çizgelerinden oluşan diğer düğümlerin budanması ise, karar ağaçlarına işaret eden bağlantıların varlığı nedeniyle daha maliyetlidir. Karar ağaçlarındaki yaprak düğümlerini silmeden önce, bu düğümlerle çizgeler arasında yapılan bağlantılar koparılır. Sonrasında ise, her adımda, yapının geri kalanından kopmuş olan çizge düğümleri silinir. Durum fark çizgelerini güncelleyen yöntem Algoritma 12 ile verilmiştir.

Algoritma 13 ÇİZGE_GÜNCELLE(N, LS)

Require:LS, karar ağaçlarından silinecek düğümlerin kümesidir

Require:N, güncellenecek çizgedir

$E \leftarrow \emptyset$ ▷silinecek çizge düğümleri kümesi
for all $n = < x, v, D > \in G$ **do**
  for all $d = < l, R, \xi > \in D$ **do**
    if d $\in$ LS **then**
      d'yi D'den çıkar
    end if
    $\xi \leftarrow \xi \times \xi_{\text{çürüme}}$ ▷niteliklilik değerini güncelle
    if $\xi < \xi_{eşik}$ **then**
      d'yi D'den çıkar
    end if
  end for
  H, n ile bağlantılı çizge düğümleri olsun
  if $D = \emptyset$ ve $H = \emptyset$ **then** ▷ n ile bağlantılı eleman kalmamışsa
    $E = E \cup \{n\}$
  end if
end for
while E updated ▷ E güncellendiği sürece
  for all $n = < x, v, D > \in G$ **do**
    if n $\in$ E veya $D \neq \emptyset$ **then** ▷ düğüm silinmek için işaretlendiyse veya ödül ve niteliklilik değerli içeriyorsa
      continue

21: **end if**
22: H, n ile bağlantılı çizge düğümleri olsun
23: **if** $H - E = \emptyset$ **then**
24: $E = E \cup \{n\}$
25: **end if**
26: **end for**
27: **end while**
28: **for all** $n \in E$ **do**
29: n'yi N'den çıkar
30: **end for**

7.1.4 BGDA'ların Kullanımı

Temel olarak, BGDA'ları kullanma yöntemi orjinal olanla aynıdır. Öğrenme tarihçesi belli aralıklarla algoritmaya aktarılır ve yapı, öğrenme sırasında dinamik olarak güncellenir. Kök düğümden başlayarak, BGDA algoritması gözlemlenen durumu şimdiki düğümün çocuklarında bulmaya çalışır. Eğer böyle bir düğüm bulunursa aradaki bağlantıyı sağlayan ayrıtın içerdiği eylem, altta yatan takviye öğrenme algoritmasına verilir. Aynı prosedür bir sonraki adımda bu sefer çocuk düğüm kullanılarak tekrar eder. BGDA önerecek eylem bulamadığı durumda ise kök düğümden çalışmaya başlar. Bölümlenmiş yapının, klasik GDA'da yaptığı değişikliklerin kullanılan hafızayı azalttığı bir sonraki bölümdeki deneylerde gösterilmiştir.

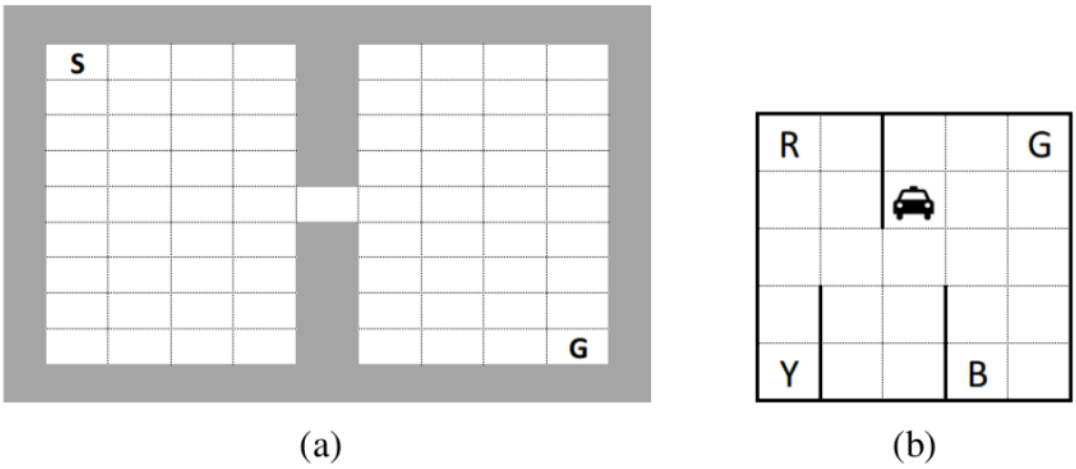
7.2 Deneyler

BGDA'nın klasik GDA'dan daha az hafıza kullandığını ve aynı zamanda da öğrenme hızını etkilemediğini test etmek için birtakım deneyler yapılmıştır. Bunun için literatürde kabul görmüş çevreler seçilmiştir. Bu çevreler aşağıda anlatılmıştır.

Seçilen ilk problem kahve-robot (Boutilier, 2000) çevresinde tanımlanmaktadır. Bu problemde bir robot ofisten kahveciye gidip yine ofiste bulunan sahibine kahve getirmeye çalışmaktadır. Eğer hava yağmurluysa, robotun yanında şemsiye alması beklenmektedir. Aksi takdirde kahve tesliminde alınan ödül 1.0'dan 0.9'a düşmektedir. Çevre, $\{hareket\_et, şemsiye\_al, kahveyi\_ver, kahve\_al\}$ eylem kümesi ve $\{lokasyon, robot\_ıslak, yağmur\_yağıyor, şemsiye\_alındı, kahve\_robotta, kahve\_teslim\_edildi\}$

değişken kümesiyle gösterilebilir. Bu değişkenlerden $lokasyon$'un değer kümesi $\{ofis, kahveci\}$ iken geri kalanlar $\{doğru, yanlış\}$ değerlini alabilirler.

İkinci problem dikey olarak iki odaya ayrılmış 9x9 ızgara dünyasında tanımlanan oda problemidir (Şekil 7.3(a)). Çevrede bu iki odayı birbirine bağlayan tek bir koridor bulunmaktadır. Etmen doğu, batı, kuzey ve güney yoönlerinde hareket edip, ilk odanın sol üstündeki noktadan başlayıp ikinci odanın sağ altındaki noktaya ulaşmaya çalışmaktadır. Problemde etmenin şimdiki koordinatlarını göstermek için x ve y değişkenleri kullanılmaktadır. Ortamdaki engellerden herhangi birine çarpmak lokasyonu değiştirmezken, diğer durumlarda da istenilen yöne gitme 0.8 olasılıkla gerçekleşmektedir. 0.2 olasılıkla ise etmen diagonal hücrelerden birine gidecektir. Yapılan her hareket -0.01 ödül getirirken hedef noktaya ulaşınca $+1.0$ ödül verilmektedir.



Şekil 7.3. (a) Oda problemi (b) Taksi problemi örnekleri.

Seçilen son problem taksi ortamıdır (Dietterich, 2000) (Şekil 7.3(b)). Bu problemde taksi şoförü 5x5 ızgara dünyasında yine 4 yönde hareket edip yolcuyu taksiye aldıktan sonra istenilen noktaya bırakmaya çalışmaktadır. Yolcuyu, bulunduğu yere ulaşmadan almaya çalışmak veya yolcu takside değilken indirmeye çalışmak -10 ödül verirken başarılı teslimat $+20$ getirir. Durumları tanımlamak için kullanılan değişkenler Bölüm 7.1.2'de verilmiştir.

7.2.1 Deney Düzeneği

Deneylerde, BGDA ve klasik GDA'nın üstünde Q-öğrenme algoritması, Girgin vd., (2010) tarafından önerilen Değiştirilmiş-ϵ-Hırslı (İng. Modified-ϵ-Greedy) yöntemi ile koşulmuştur.

Kullanılan öğrenme parametre değerleri Tablo 7.1'de verilmiştir. Her bir problem için 100 deney yapılmış, her koşumun bölümleri üzerinden ortalamalar alınmıştır.

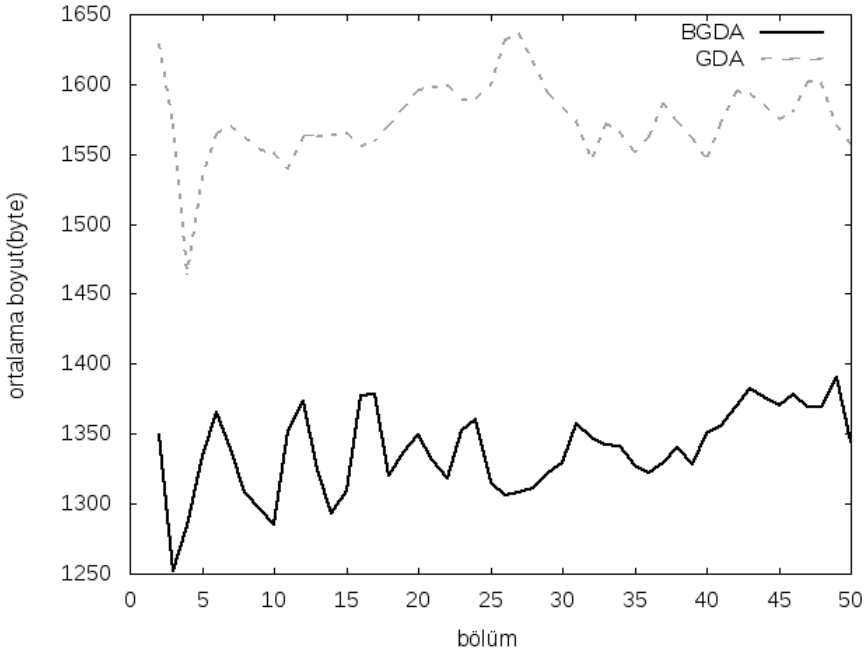
Tablo 7.1. Öğrenme paremetrelerinin değerleri

Parametre	**Değer**
α	0.125
γ	0.125
ϵ	0.1
$\psi_{\text{çürüme}}$	0.9
$\xi_{\text{çürüme}}$	0.9
$\psi_{eşik}$	0.7
$\xi_{eşik}$	0.7

Deneylerde, BGDA ve GDA versiyonları hafıza kullanımı açısından karşılaştırılmıştır. Ek olarak, hedef duruma ulaşmak için gereken adım sayısı BGDA ve GDA'nın yanısıra salt Q-öğrenme algoritmasında da ölçülüp yeni modelin öğrenme hızında herhangi bir negatif etkisi olup olmadığı araştırılmıştır. İlk adımlarda BGDA ve GDA'ya henüz olaylar geçmişi verilmediği için, bu adımların grafiklerden çıkarıldığı göz önünde bulundurulmalıdır. Her bölüm sonunda, yapıların güncellenme için harcadıkları zamanlar da çizgelerde gösterilmiştir.

7.2.2 Sonuçlar ve Tartışma

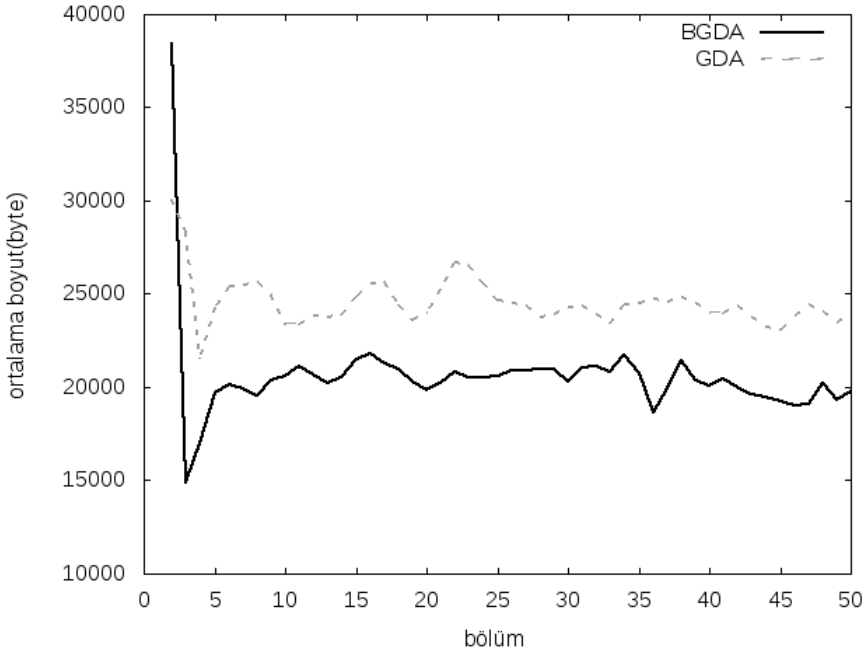
Deney sonuçlarına göre, BGDA, GDA'ya oranla daha az hafıza kullanırken öğrenme hızını değiştirmemektedir. Buna rağmen, yeni modeli güncellemek orijinalinden daha maliyetli olmaktadır. Bunun nedeni, öğrenme tarihçesinden bölümlenmiş olaylar tarihçesi elde etmek için yapılan işlemler ve düğümler içinde uygulanan ağaç ve çizge gezme algoritmalarıdır.



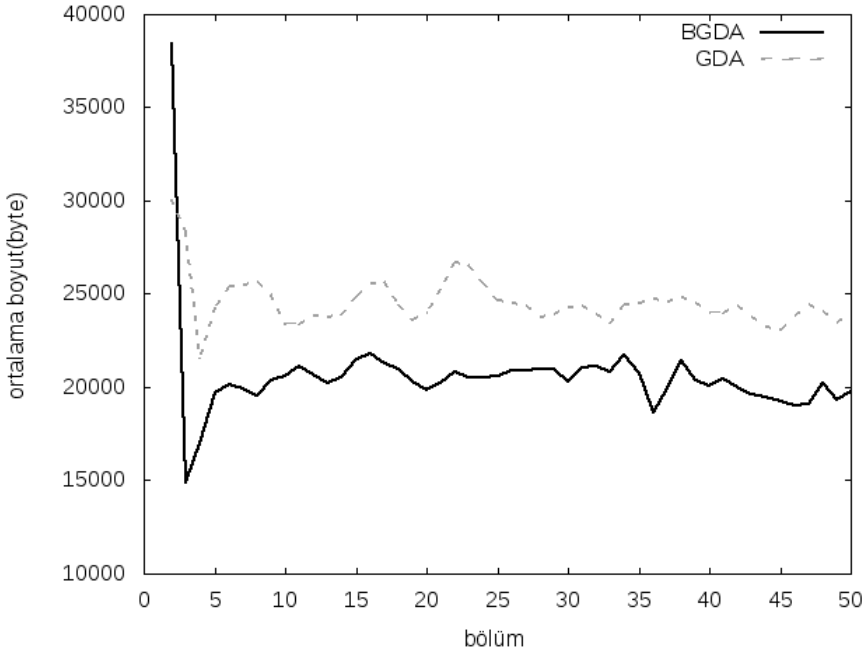
Şekil 7.4. Kahve-robot problemi için ortalama ağaç boyutları.

Kahve-robot probleminde BGDA hafıza kullanımında GDA'ya oranla daha etkin davranmıştır (Şekil 7.4). Grafiklerdeki keskin inişler ise yüksek niteliklilik eşik değerlerinin sonucudur. Bu nedenle, güncelleme anında her düğümden silinen eleman sayısı fazla olmaktadır. Tablo 7.2'de gösterildiği gibi hedef durumlara ulaşmak için yapılan hamle sayılarında büyük farklar bulunmamaktadır. Bu problem için, herhangi bir yapıyla desteklenmeyen Q-öğrenme algoritması da hızlı bir şekilde öğrenmektedir. Aynı tablodan, BGDA'nın güncellenme zamanı açısından geride kaldığı da gözlemlenebilir.

Oda ve taksi problemlerinde, BGDA ilk bölümlerde GDA'dan daha fazla hafıza kullanırken, budama işlemlerinden sonra avantajlı duruma geçmiştir (Şekil 7.5, Şekil 7.6). Yine problemleri çözmek için gereken adım sayısı iki GDA versiyonunda da yakın çıkarken Q-öğrenme metodu diğer algoritmalar kadar iyi sonuçlar vermemiştir. Klasik DGA yapısı, güncellenme zamanı açısından bu deneylerde de BGDA'dan daha iyi durumdadır.



Şekil 7.5. Oda problemi için ortalama ağaç boyutları.



Şekil 7.6. Taksi problemi için ortalama ağaç boyutları.

Tablo 7.2. Bazı deney ölçümlerinin karşılaştırılması

	ortalama problemi çözme adım sayısı			**ortalama ağaç güncelleme süresi**	
Problem	**GDA'sız**	**GDA ile**	**BGDA ile**	**GDA ile**	**BGDA ile**
Kahve-robot	14.69	7.64	7.77	0.14	0.17
Oda	121.027	23.68	23.15	1.1	1.4
Taksi	199.586	16.08	16.06	0.87	1.09

8 SONUÇLAR VE GELECEK ÇALIŞMALAR

Bu çalışmada, orijinal GDA soyutlama yöntemi değiştirilerek, kısmi gözlemlenebilir ortamlar için tasarlanmış bilinen takviye öğrenme algoritmalarının hızlandırılmasını amaçlayan yöntemler önerilmiştir.

i-GDA, inanç uzayının ayrıştırılmasına dayanarak, mevcut GDA soyutlama yöntemini KGMKS için tasarlanmış model tabanlı takviye öğrenme algoritmalarını kapsayacak şekilde genişletmektedir. i-GDA için kullanıma uygun bazı inanç tabanlı ayrıştırma yöntemleri de önerilmiştir. i-GDA yönteminin önerilen ayrıştırma yöntemleriyle etkinliği, altı farklı kısmi gözlemlenebilir problem üzerinde deneylerle gösterilmiş, sonuçlar detaylı olarak tartışılmıştır.

i-GDA için yeni bir araştırma alanı, ayrıştırma parametrele değerlerinin otomatik olarak bulunması veya artırımlı şekilde ayarlanması olabilir. Ayrıca, yeni inanç ayrıştırma yöntemlerinin uygulanması değerli olabilecektir. Bu çalışmanın, model tabanlı farklı kısmi gözlemlenebilir takviye öğrenme algoritmalarını (SPOVA gibi) kapsayacak şekilde genişletilmesi de, başka bir zorlu araştırma yönü olarak belirlenebilir.

Bu çalışmada odaklanılan bir başka takviye öğrenme paradigması ise, modelden bağımsız takviye öğrenmedir. Belleksiz etmen varsayımı çerçevesini temsilen seçilen SARSA(λ) algoritması, zamansal soyutlama ile iyileştirilmiştir. Önerilen yöntem olarak t-GDA, belirli varsayımlar altında iyi performans sergilemekte, ve öğrenmeyi iyileştirebilmektedir. t-GDA, saklı durum içeren üç farklı problemle test edilmiş, ve sonuçlar tartışılmıştır.

Çalışmanın son bölümünde, bellek tabanlı modelden bağımsız bir çerçeve varsayıldığında, kısmi gözlemlenebilir problemler için takviye öğrenmeyi hızlandıran bir başka GDA varyantı önerilmiştir. t-GDA ara soyutlamaları bulmakta yetersiz kaldığından, kendi alanında iyi bilinen bir algoritma olan YSB algoritmasının hızlandırılması için t-GDA üzerinde bir değişiklik önerilmiştir. b-GDA olarak adlandırılan yeni soyutlama yöntemi, YSB tarafından üretilen durum bilgilerini de kullanarak YSB algoritmasını hızlandırabilmektedir. Tek başına YSB, t-GDA ile genişletilmiş YSB ve b-GDA ile genişletilmiş YSB algoritmalarının performansları dört farklı problem üzerinde denenmiş ve sonuçlar tartışılmıştır.

b-GDA yönteminin en önemli kusuru, gözlem-eylem dizilerindeki gereksiz tekrarları yakalayamamasıdır. Bu problemin üstesinden gelmek amacıyla, tarihçe üretimi aşamasında

YSB'nin yaprak düğümleri yerine saçak düğümleri kullanılabilir. Ancak bu çözüm, tahmin edilen durumların gereğinden fazla ayrıştırılmasına, dolayısı ile uzun vadede tercih kalitesinin düşmesine neden olabilir.

Maalesef, gerek t-GDA, gerekse b-GDA yöntemleri belirsizlik içeren ve değişken ortmalarda başarılı olamamaktadır. Gelecek vaadeden bir araştırma yönü, bu gereksinimi, budama mekanizmasında bazı istatistiksel yöntemler kullanarak ortadan kaldırmaktır.

Bu çalışmada önerilen tüm kısmi gözlemlenebilir GDA varyantları, orijinal yöntemin genelleştirilmiş versiyonlarıdır. Bu durum, KGMKS'nin MKS modelinin daha genel bir hali olmasına benzetilebilir. Başka bir deyişle, önerilen yöntemler belli koşullar altında GDA'ya indirgenmektedir. Bu bağlamda, çözümün tamamı oluşturulmamış olsa da, bu çalışma, otomatik zamansal soyutlamanın kısmi gözlemlenebilir ortamlarda takviye öğrenme için genelleştirilmesi için atılan ilk kapsamlı adım olarak değerlendirilebilir.

Nihai olarak, "bölümlenmiş KGMKS için zamansal soyutlama" probleminin çözümüne ışık tutabilecek BGDA yöntemi, tam gözlemlenebilir ortamlar için geliştirilmiş ve literatüre kazandırılmıştır. Yöntemin hafıza kullanımına katkısı ve diğer yan etkileri örnek problemler üzerinde tartışılmıştır.

9 KAYNAKÇA

Alpaydın, E., 2004. Introduction to Machine Learning. The MIT Press.

Baird, L., Moore, A., 1999. Gradient descent for general reinforcement learning. In Cambridge, MA, USA: MIT Press, pp. 968–974.

Barto, A.G., Mahadevan, S., 2003. "Recent advances in hierarchical reinforcement learning", Discrete Event Dynamic Systems, 13(4), 341–379.

Bates, J., 1994. "The role of emotion in believable agents", Communications of the ACM, 37(7), 122–125.

Bellman, R.E., 1957. Dynamic Programming, Princeton, NJ: Princeton University Press.

Boutilier, C., Dearden, R., Goldszmidt, M., 2000. "Stochastic dynamic programming with factored representations", Artificial Intelligence, 121(12), 49–107.

Boyen, X., Koller, D., 1998. Tractable inference for complex stochastic processes. In Madison, Wisconsin, USA: Morgan Kaufmann, 33–42.

Bradtke, S.J., Duff, M.O., 1994. "Reinforcement learning methods for continuous-time markov decision problems", NIPS 1994. Cambridge, MA: MIT Press, 393–400.

Cassandra, A.R., 1998. *Exact and approximate algorithms for partially observable markov decision processes*. Ph.D. thesis. Providence, RI, USA: Brown University.

Cassandra, A.R., Kaelbling, L.P., Kurien, J.A., 1996. "Acting under uncertainty: discrete Bayesian models for mobile-robot navigation", IROS '96, 963–972.

Cassandra, A.R., Kaelbling, L.P., Littman, M.L., 1994. "Acting optimally in partially observable stochastic domains", AAAI'94. Menlo Park, CA, USA: AAAI Press, 1023–1028.

Charlin, L., Poupart, P., Shioda, R., 2007. "Automated hierarchy discovery for planning in partially observable environments" NIPS 2006. MIT Press, 225–232.

Chrisman, L., 1992. "Reinforcement learning with perceptual aliasing: the perceptual distinctions approach", AAAI'92. AAAI Press, 183–188.

Crook, P.A., 2006. *Learning in a State of Confusion: Employing Active Perception and Reinforcement Learning in Partially Observable Worlds*. Ph.D. thesis. UK: University of Edinburgh.

Çilden, E., 2014. *Abstraction in Reinforcement Learning in Partially Observable Domains*, Ph.D. thesis, Ankara, Turkey: Middle East Technical University.

Çilden, E., Polat, F., 2012. "Abstraction in Model Based Partially Observable Reinforcement Learning using Extended Sequence Trees", In Macau, China, 348–355. Available at: http://dx.doi.org/10.1109/WI-IAT.2012.161.

Çilden, E., Polat, F., 2013. “Generating memoryless policies faster using automatic temporal abstractions for reinforcement learning with hidden state”, ICTAI ’13. Washington, DC, USA, 719–726. Available at: http://dx.doi.org/10.1109/ICTAI.2013.111.

Çilden, E., Polat, F., 2014. “Employing automatic temporal abstractions to accelerate utile suffix memory algorithm”, In J. P. Müller, M. Weyrich, A. L. C. Bazzan, eds. *Multiagent System Technologies*. Lecture Notes in Computer Science. Springer International Publishing, 156–169. Available at: http://dx.doi.org/10.1007/978-3-319-11584-9_11.

Çilden, E., Polat, F., 2015. “Toward Generalization of Automated Temporal Abstraction to Partially Observable Reinforcement Learning”, IEEE Transactions on Cybernetics, 45(8), 1414-1425. Available at: http://dx.doi.org/10.1109/TCYB.2014.2352038.

Degris, T. , Sigaud, O., Wuillemin, P.-H., 2006. “Learning the structure of factored markov decision processes in reinforcement learning problems”, Proceedings of the 23rd International Conference on Machine Learning, ICML ’06. New York, NY, USA, 257–264.

Dietterich, T.G., 2000. “Hierarchical reinforcement learning with the MAXQ value function decomposition”, Journal of Artificial Intelligence Research, 13, 227–303.

Dietterich, T.G., Lathrop, R.H., Lozano-Pérez, T., 1997. “Solving the multiple instance problem with axis-parallel rectangles”, Artificial Intelligence, 89(1-2), 31–71.

Dung, L.T., Komeda, T., Takagi, M., 2007. “Reinforcement learning in non-markovian environments using automatic discovery of subgoals”. Proceedings of SCE’07, 2601–2605.

Forestier, J.-P., Varaiya, P., 1978. “Multilayer control of large markov chains”, IEEE Transactions on Automatic Control, 23(2), 298 – 305.

Girgin, S., 2007. *Abstraction in reinforcement learning*. Ph.D. thesis. Ankara, Turkey: Middle East Technical University.

Girgin, S., Polat, F., Alhajj, R., 2010. “Improving reinforcement learning by using sequence trees”, Machine Learning, 81(3), 283–331.

Gosavi, A., 2009. “Reinforcement learning: a tutorial survey and recent advances”, INFORMS Journal on Computing, 21(2), 178–192.

Hauskrecht, M., 2000. “Value-function approximations for partially observable markov decision processes”, Journal of Artificial Intelligence Research, 13, 33–94.

Hengst, B., 2002. “Discovering hierarchy in reinforcement learning with HEXQ”, ICML’02. Morgan Kaufmann Publishers Inc., 243–250.

He, R., Brunskill, E., Roy, N., 2010. “PUMA: Planning under uncertainty with macro-actions”, In M. Fox, D. Poole, eds. Twenty-Fourth AAAI Conference on Artificial Intelligenc. Atlanta, Georgia, USA: AAAI Press.

Hochreiter, S., Schmidhuber, J., 1997. “Long short-term memory”, Neural Computation, 9, 1735–1780.

Jaakkola, T., Singh, S.P., Jordan, M.I., 1995. “Reinforcement learning algorithm for partially observable markov decision problems”, NIPS ’94, Cambridge, MA: MIT Press, 345–352.

Kaelbling, L.P., Littman, M.L., Cassandra, A.R., 1998. “Planning and acting in partially observable stochastic domains”, Artificial Intelligence, 101(1-2), 99–134.

Kaelbling, L.P., Littman, M.L., Moore, A.P., 1996. “Reinforcement learning: a survey”, Journal of Artificial Intelligence Research, 4, 237–285.

Kozlova, O., Sigaud, O., Meyer, C., 2010. “Texdyna: Hierarchical reinforcement learning in factored mdps”, In From Animals to Animats 11, 11th International Conference on Simulation of Adaptive Behavior, SAB 2010, Paris - Clos Lucé, France, August 25-28, 489–500.

Lin, L.-J., 1991. “Programming robots using reinforcement learning and teaching”, AAAI’91. Anaheim, California: AAAI Press, 781–786.

Lin, L.-J., Mitchell, T.M., 1992. *Memory approaches to reinforcement learning in non-markovian domains*, Technical Report, Pittsburgh, PA, USA: Carnegie Mellon University.

Littman, M.L., 1994. “Memoryless policies: theoretical limitations and practical results”, SAB’94. Cambridge, MA: MIT Press, 238–245.

Littman, M.L., Cassandra, A.R., Kaelbling, L.P., 1998. “Learning policies for partially observable environments: Scaling Up”. In M. N. Huhns, M. P. Singh, eds. *Readings in Agents*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 495–503.

Loch, J., Singh, S.P., 1998. “Using eligibility traces to find the best memoryless policy in partially observable markov decision processes”. ICML ’98. San Francisco, CA: Morgan Kaufmann Publishers Inc., 323–331.

Lovejoy, W.S., 1991. “Computationally feasible bounds for partially observed Markov decision processes”, Operational Research, 39(1), 162–175.

Mannor, S. vd., 2004. “Dynamic abstraction in reinforcement learning via clustering”, ICML ’04. ACM, 71–78.

Maron, O., 1998. *Learning from Ambiguity*. Ph.D. thesis. Cambridge, MA, USA: Massachusetts Institute of Technology.

McCallum, A.K., 1996. *Reinforcement Learning with Selective Perception and Hidden State*. Ph.D. thesis. NY: University of Rochester.

McGovern, A., Barto, A.G., 2001. “Automatic discovery of subgoals in reinforcement learning using diverse density”, ICML’01. Morgan Kaufmann Publishers Inc., 361–368.

McGovern, E.A., 2002. *Autonomous Discovery of Temporal Abstractions from Interaction with an Environment*. Ph.D. thesis. Amherst, MA, USA: University of Massachusetts Amherst.

Murphy, K.P., 2000. *A survey of POMDP solution techniques*, Technical Report, Berkeley, California, USA: University of California, Berkeley.

Parr, R.E., 1998. *Hierarchical Control and Learning for Markov Decision Processes*. Ph.D. thesis. Berkeley, CA, USA: University of California.

Parr, R., Russell, S., 1995. Approximating optimal policies for partially observable stochastic domains. In IJCAI'95. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1088–1094.

Parr, R., Russell, S., 1998. "Reinforcement learning with hierarchies of machines", NIPS '97. Cambridge, MA, USA: MIT Press, 1043–1049.

Pendrith, M.D., McGarity, M., 1998. "An analysis of direct reinforcement learning in non-Markovian domains" ICML '98. San Francisco, CA: Morgan Kaufmann Publishers Inc., 421–429.

Peshkin, L., Meuleau, N., Kaelbling, L.P., 1999. Learning policies with external memory. ICML '99. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 307–314.

Pineau, J., 2004. *Tractable planning under uncertainty: exploiting structure*. Ph.D. thesis. Pittsburgh, PA, USA: Carnegie Mellon University.

Pineau, J., Thrun, S., 2002. *An integrated approach to hierarchy and abstraction for POMDPs*, CMU-RI-TR-02-21, Pittsburgh, PA, USA: The Robotics Institute at Carnegie Mellon University.

Roy, N., 2003. *Finding Approximate POMDP Solutions through Belief Compression*. Ph.D. thesis. Pittsburgh, PA, USA: Carnegie Mellon University.

Rummery, G.A., Niranjan, M., 1994. *On-line Q-learning using connectionist systems*, Cambridge, UK: Cambridge University Engineering Department.

Russell, S.J., Norvig, P., 2003. Artificial Intelligence: A Modern Approach. (Second Edition). Pearson Education.

Shani, G., 2007. *Learning and Solving Partially Observable Markov Decision Processes*. Ph.D. thesis. Israel: Ben-Gurion University of the Negev.

Shoham, Y., 1993. "Agent-oriented programming", Artificial Intelligence, 60(1), 51–92.

Sigaud, O., Buffet, O., 2010. Markov Decision Processes in Artificial Intelligence, ISTE - Wiley.

Smith, T., Simmons, R., 2004. "Heuristic search value iteration for POMDPs", UAI '04. Banff, Canada: AUAI Press, 520–527.

Sondik, E.J., 1971. *The optimal control of partially observable Markov decision processes*. Ph.D. thesis. Stanford, CA, USA: Stanford University.

Stolle, M., Precup, D., 2002. Learning options in reinforcement learning. In S. Koenig, R. C. Holte, eds. *Abstraction, Reformulation, and Approximation*. London, UK: Springer Berlin / Heidelberg, 212–223.

Stone, P., Sutton, R.S., Kuhlmann, G., 2005. “Reinforcement learning for RoboCup soccer keepaway”, Adaptive Behavior, 13(3), 165–188.

Sutton, R.S., 1990. Integrated architecture for learning, planning, and reacting based on approximating dynamic programming. In San Francisco, CA: Morgan Kaufmann Publishers Inc., pp. 216–224.

Sutton, R.S., 1988. “Learning to predict by the methods of temporal differences” Machine Learning, 3(1), 9–44.

Sutton, R.S., Barto, A.G., 1998. Reinforcement Learning: An Introduction, MIT Press.

Sutton, R.S., Precup, D., Singh, S., 1999. “Between MDPs and semi-MDPs: a framework for temporal abstraction in reinforcement learning”, Artificial Intelligence, 112(1-2), 181–211.

Şahin, C., 2015. *Factored reınforcement learnıng usıng extended sequence trees*, M.S. thesis, Ankara, Turkey: Middle East Technical University.

Şahin, C., Çilden, E., Polat, F., 2015. “Memory efficient factored abstraction for reinforcement learning”, Cybernetics (CYBCONF), 2015 IEEE 2nd International Conference on , vol. 18, no. 23, pp. 24-26. Available at http://doi.acm.org/10.1109/CYBConf.2015.7175900.

Şimşek, Ö., Barto, A.G., 2004. “Using relative novelty to identify useful temporal abstractions in reinforcement learning”. ICML’04. ACM, pp. 95–102.

Theocharous, G., Kaelbling, L.P., 2004. “Approximate planning in POMDPs with macro-actions” NIPS 2003. MIT Press, pp. 775–782.

Theocharous, G., Rohanimanesh, K., Mahadevan, S., 2001. “Learning hierarchical partially observable markov decision process models for robot navigation”, In ICRA. 511–516.

Utgoff, P. E., Berkman, N. C., Clouse, J. A., Fisher, D., 1996. “Decision tree induction based on efficient tree restructuring”, Machine Learning, 29(1), 5–44.

Watkins, C., 1989. *Learning from Delayed Rewards*. Ph.D. thesis. UK: Cambridge University.

Whitehead, S., Ballard, D., 1991. “Learning to perceive and act by trial and error” Machine Learning, 7(1), 45–83.

Wiering, M., Schmidhuber, J., 1997. “HQ-Learning”, Adaptive Behavior, 6, 219–246.

Wooldridge, M., Jennings, N.R., 1995. “Intelligent Agents: Theory and Practice”, *Knowledge Engineering Review*, 10, 115–152.

Yoshikawa, T., Kurihara, M., 2006. “An acquiring method of macro-actions in reinforcement learning”, SMC ’06, 4813–4817.

Zhou, R., Hansen, E.A., 2001. “An improved grid-based approximation algorithm for POMDPs”, IJCAI’01. Seattle, Washington, USA: Morgan Kaufmann Publishers Inc., 707–716.

TÜBİTAK
PROJE ÖZET BİLGİ FORMU

Proje Yürütücüsü:	Prof. Dr. FARUK POLAT
Proje No:	113E239
Proje Başlığı:	Kısmi Gözlemlenebilir Takviye Öğrenme Için Dolaysız Soyutlama
Proje Türü:	1001 - Araştırma
Proje Süresi:	24
Araştırmacılar:	
Danışmanlar:	
Projenin Yürütüldüğü Kuruluş ve Adresi:	ORTA DOĞU TEKNİK Ü. MÜHENDİSLİK F. BİLGİSAYAR MÜHENDİSLİĞİ B.
Projenin Başlangıç ve Bitiş Tarihleri:	01/09/2013 - 01/09/2015
Onaylanan Bütçe:	138870.0
Harcanan Bütçe:	101499.16
Öz:	Pekiştirmeli öğrenme, özerk etmen bakış açısıyla, makine öğrenme yöntemleri arasında önde gelen bir yönlendirmesiz yöntem ailesi tanımlar. Markov karar süreci modeli, pekiştirmeli öğrenme algoritmaları için sağlam bir biçimsel temel oluşturur. Pekiştirmeli öğrenme yöntemlerinin üstüne zamansal soyutlama mekanizmaları inşa edilerek başarımlarında kayda değer artış elde edilebilmektedir. Eğer Markov karar süreci modelinin tam gözlemlenebilirlik varsayımı esnetilirse, ortaya çıkan kısmi gözlemlenebilir Markov karar süreci modeli, daha gerçekçi, ancak zor bir problem alanı tanımlar. Kısmi gözlemlenebilirlik altında pekiştirmeli öğrenme araştırmaları, algısal aynılık ve çok büyük durum uzayı sorunlarının yol açtığı olumsuz etkileri azaltacak tekniklere odaklanmıştır. Genel olarak, bu çalışmalar iki kategoriye ayrılabilir. Model tabanlı yaklaşımlar durum geçiş modelinin etmen tarafından erişilebilir olduğu varsayımına dayanır. Modelden bağımsız yaklaşımlarda ise durum bilgileri etmenden tamamen saklıdır. Bu projede, bilinen bir sıralama tabanlı otomatik zamansal soyutlama tekniğini (genişletilmiş dizi ağacı metodu) kısmi gözlemlenebilir problemler için genelleştiren yöntemler önerilmektedir. Probleme hem model tabanlı, hem de modelden bağımsız bakış açısıyla yaklaşılmış, önerilen yöntemlerin her iki bakış açısının önde gelen temsilcilerinde hızlanma sağladığı gösterilmiştir. Yöntemlerin etkinliği, yaygın kabul gören problemler üzerinde deneylerle gösterilmiştir.
Anahtar Kelimeler:	Pekiştirmeli Öğrenme, Kısmi Gözlemlenebilir Markov Karar Süreci, Zamansal Soyutlama
Fikri Ürün Bildirim Formu Sunuldu Mu?:	Hayır
Projeden Yapılan Yayınlar:	1- Toward Generalization of Automated Temporal Abstraction to Partially Observable Reinforcement Learning (Makale - Diğer Hakemli Makale), 2- Memory Efficient Factored Abstraction for Reinforcement Learning (Bildiri - Uluslararası Bildiri - Sözlü Sunum), 3- Employing Automatic Temporal Abstractions to Accelerate Utile Suffix Memory Algorithm (Bildiri - Uluslararası Bildiri - Sözlü Sunum), 4- Generating Memoryless Policies Faster Using Automatic Temporal Abstractions for Reinforcement Learning with Hidden State (Bildiri - Uluslararası Bildiri - Sözlü Sunum), 5- Abstraction in Reinforcement Learning in Partially Observable Domains (Tez (Araştırmacı Yetiştirilmesi) - Doktora Tezi), 6- FACTORED REINFORCEMENT LEARNING USING EXTENDED SEQUENCE TREES (Tez (Araştırmacı Yetiştirilmesi) - Yüksek Lisans Tezi),