İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

BİLİŞSEL ROBOTLARDA YAŞAM BOYU DENEYİMSEL ÖĞRENME İLE HATA KOTARMA

YÜKSEK LİSANS TEZİ

Sertaç KARAPINAR

Bilgisayar Mühendisliği Ana Bilim Dalı

Bilgisayar Mühendisliği Lisansüstü Programı

OCAK 2013

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

BİLİŞSEL ROBOTLARDA YAŞAM BOYU DENEYİMSEL ÖĞRENME İLE HATA KOTARMA

YÜKSEK LİSANS TEZİ

Sertaç KARAPINAR
(504101517)

Bilgisayar Mühendisliği Ana Bilim Dalı

Bilgisayar Mühendisliği Lisansüstü Programı

Tez Danışmanı: Yrd. Doç. Dr. Sanem SARIEL TALAY

OCAK 2013

İTÜ, Fen Bilimleri Enstitüsü'nün 504101517 numaralı Yüksek Lisans Öğrencisi **Sertaç KARAPINAR**, ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı **"BİLİŞSEL ROBOTLARDA YAŞAM BOYU DENEYİMSEL ÖĞRENME İLE HATA KOTARMA"** başlıklı tezini aşağıdaki imzaları olan jüri önünde başarı ile sunmuştur.

Tez Danışmanı :	**Yrd. Doç. Dr. Sanem SARIEL TALAY** İstanbul Teknik Üniversitesi	.....
Jüri Üyeleri :	**Yrd. Doç. Dr. Hülya Yalçın** İstanbul Teknik Üniversitesi	.....
	Yrd. Doç. Dr. Yusuf Yaslan İstanbul Teknik Üniversitesi	.....
		.....

Teslim Tarihi : **17 Aralık 2012**
Savunma Tarihi : **24 Ocak 2013**

ÖNSÖZ

Tez çalışması süresince gerçekleştirilen deneylerdeki önemli katkıları sebebiyle Melis Kapotoğlu ve İTÜ Artificial Intelligence and Robotics (AIR) laboratuvarında görevli diğer tüm öğrencilere,

Lisans döneminden itibaren beraber çalıştığım, birçok çalışmamda olduğu gibi bu çalışmamda da çok önemli katkı ve yardımlarda bulunan tez danışmanım Sayın Sanem Sarıel Talay'a teşekkürlerimi sunarım.

Bu tez çalışması TÜBİTAK tarafından desteklenen 111E-286 numaralı proje kapsamında gerçekleştirilmiştir. Ayrıca yüksek lisans eğitimim süresince TÜBİTAK kurumundan tarafıma burs sağlanmıştır. TÜBİTAK kurumuna değerli desteklerinden ötürü teşekkürlerimi sunarım.

Ocak 2013

Sertaç KARAPINAR
(Bilgisayar Mühendisi)

İÇİNDEKİLER

Sayfa

KISALTMALAR

ILP : Inductive Logic Programming
ID3 : Iterative Dichotomiser 3
KB : Knowledge Base
FOIL : First-order Inductive Logic
SIFT : Improve Scale Invariant Feature Transform
VPFH : ViewPoint Feature Histogram
PSP : Partial Satisfaction Planning
ROC : Receiver Operating Characteristic

ÇİZELGE LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

BİLİŞSEL ROBOTLARDA YAŞAM BOYU DENEYİMSEL ÖĞRENME İLE HATA KOTARMA

ÖZET

Bilişsel bir robot hedeflerini gerçekleştirebilmek için gerekli olan eylemler dizisini planlama ile üretebilir. Ancak robot, planda yer alan eylemleri yürütürken iç ve dış etkenlerden dolayı çeşitli hatalarla karşılaşabilir. Gürbüz görev yürütme; sürekli planlama, yürütme, gözlemleme, çıkarsama ve öğrenme süreçlerinin birbirleriyle entegre olarak çalışmasını gerektirir. Bu tez çalışmasında, eylem yürütme deneyimleri göz önüne alınarak görev yürütmede gürbüzlüğün nasıl sağlanabileceği araştırılmıştır. Hataları kotarmak için, bilişsel robotlar için bir gürbüz planlama sistemi önerilmiştir. Bu sistemde, görevlerin başarıyla yürütülebilmesi için planlama, yürütme, gözlemleme ve öğrenme süreçleri bir araya getirilmiştir.

Sistemin ilk adımı, planlama ile üst seviyede bir plan üretmektir. Önerilen yaklaşımda ileri zincir yöntemli, zamansal bir planlayıcı olan TLPlan kullanılmıştır. Bu planlayıcı, tekli ve çoklu robot sistemlerinde hem *süreli* hem de *anlık* eylemler için plan üretebilmektedir. TLPlan; tanım kümesi bilgisini (operatorler, olgular vs.), mevcut ortam durumunu, verilen hedefleri ve gürbüz görev yürütme için gereken geçerli planları üretmek amacıyla öğrenilen hipotezleri kullanır. Verilen durum için geçerli bir plan üretildiğinde, bu plan içinde yer alan her bir eylemin sırasıyla yürütülebilmesi için robotun modelinde saklanır. Yürütme süreci robotun eyleyicilerini (actuator) ve ortam etkileyicilerini (effector) kontrol ederek eylemleri sırasıyla yürütür.

Yürütme gözlemleme süreci güncellenen durumların ve olası hataların kontrol edilebilmesi için, plan yürütülürken sürekli olarak gözlenmesini sağlar. Eğer gözlenen durum beklenen bir durum değil ise, hata oluştuğuna karar verilir, yeniden planlama ve çıkarım mekanizmaları devreye sokulur. Hatanın farkedilmesiyle birlikte güncellenen ortam durumuna göre yeniden planlama yapılır ya da öğrenme süreci kullanılarak yeni hipotezler oluşturulur.

Öğrenme süreci; robotun eylemler, eylemlerin parametreleri ve ortamda bu eylemler tarafından etkilenen varlıklar hakkında kazandığı tecrübelere dayanarak kendini ortama adapte etmesini sağlar. Bu süreç, her eylemin yürütülmesi sonucunda etkinleştirilerek hipotez üretilmesini sağlar. Böylece her eylem için başarılı ve başarısız olduğu durumlar ile ilgili hipotezler üretilerek, bu eylemlerin daha sonra tekrarlanmaları durumunda robotun deneyimlerinden faydalanması sağlanır. Ayrıca, hata sezildiğinde ve yeniden planlama yapmanın mümkün olmadığı durumlarda da alternatif planlar üretilebilmesi için yine öğrenme sürecinin çıktılarından faydalanılır.

Problemin çözümü için kullanılan yaklaşım yaşam boyu deneyimsel öğrenmeye dayanmaktadır. Hem yürütme modelleri hem de hata durumları için hipotez üretmek amacıyla Tümevarımsal Mantıksal Programlama (Inductive Logic Programming)

yöntemi kullanılmıştır. ILP hipotezleri yüklem mantığı bilgi gösterim dili ile temsil etmektedir. Böylece üretilen hipotezler çıkarsama ve planlama birimleri tarafından kolaylıkla kullanılabilmektedir. Ayrıca ILP ile bilgi birikimi (background knowledge) kullanılabileceğinden daha gelişmiş hipotezler üretilebilir. Kısmi olarak gözlemlenen dünya durumları bu kurallar ile kolaylıkla temsil edilebilir. ILP'nin bütün bu avantajları, nitelik tabanlı öğrenme yöntemlerine karşı üstünlük kurmasını sağlar. Öğrenme ile elde edilen deneyimler robotu gelecek kararlarında yönledirir. ILP'nin başarımı hem nitelik tabanlı öğrenme yöntemleri ile karşılaştırılarak hem de Pioneer 3DX robot üzerinde kullanılarak analiz edilmiştir. Sonuçlar, hata durumları için üretilen hipotezlerin robotun gelecek görevlerinin güvenliğini sağladığını göstermiştir.

FAILURE RECOVERY BY LONG-TERM EXPERIENCE-BASED LEARNING FOR COGNITIVE ROBOTICS

SUMMARY

A cognitive robot should possess abilities to solve problems and plan to attain its goals, reason about dynamic cases and learn from experience as intelligent systems in nature. Problem solving and planning is crucial for achieving the given objectives. Automated planners are commonly used for finding a coarse of actions for a robot to achieve its goals. These planners usually take the domain information (initial/goal states and operators corresponding to real-world actions) to construct a plan. During the execution of actions in the constructed plan, a robot may face several types of failures some of which may be recovered by replanning. However, there may be gaps between the real-world representation of the domain and its symbolic counterpart. Especially when the real outcomes of actions are not completely represented, a planner may not be able to construct a valid plan in case of failures. Belief revision and reasoning tools are necessary to deal with these type of issues. Furthermore, the robot should be equipped with learning capabilities for the efficiency of its future decisions.

The main focus of this research is developing a robust planning framework against real-world failures. We propose a continual planning, execution, monitoring and learning framework for cognitive robots. The framework combines five main processes, namely, Planning, Scene Interpretation, Execution, Execution Monitoring and Learning for robust execution of tasks. All these processes have access to the Knowledge Base (KB). KB maintains the domain knowledge, the world state and the goals (for the planning problem), the plan and the gained experience in terms of the generated hypotheses. These processes use sensor and motor interfaces to sense and act, respectively.

The first step in the framework is constructing a high-level plan by planning. TLPlan, a forward chaining temporal planner is used in our proposed approach. This planner can construct plans of both TGP-style actions and instantaneous actions for both single and multirobot domains. The planner uses the domain knowledge (i.e., operators and facts), the current world state, the given goals and the hypotheses learned to generate a valid plan for robust execution. When a complete plan for a given state is found, it is maintained in KB so that Execution takes responsibility of taking each action in sequence. Execution process can control the actuators and the effectors of the robot.

Execution Monitoring process continuously monitors the execution of the plan to check the updated states and detect failures if any. If the observed state does not include the intended outcome, a failure is assumed, and the corresponding replanning and reasoning methods are activated. Upon detecting failures, depending on the updated world state, either the planner is invoked to replan or new hypotheses are generated by *Learning*.

Similar to human-level failure detection (Wolpert ve Ghahramani, 2000), our framework detects failures by confronting sensory predictor with the actual sensor inputs. For instance, the robot may fail during the execution of a *pick up* action due to several reasons. *Scene Interpretation* is responsible for updating the domain knowledge and the world state in KB, based on the gathered data from the environment using the sensors. After getting all sensory data, this process interprets the objects in the scene and their relations, and propositionalize these facts in KB. To detect and recognize objects in a scene (Sjoo ve diğ., 2011; Cubek ve Ertel, 2011; Hinterstoisser ve diğ., 2012), advanced 3D vision techniques are needed. Our ongoing work includes investigation of these techniques. We leave the details of these techniques out of the scope of this paper. For now, we assume that some observable properties of objects can be recognized but we also relax the assumption on full observability of these features. When the robot is interacting with known objects, it can use their predefined models (i.e., templates) and different physical/visual features. If the object models are not known in advance, we assume the robot can grab the object's some of the observable visual features (e.g., key descriptors such as SIFT (Lowe, 2004) and Viewpoint Feature Histogram (Rusu ve diğ., 2010) or size) and store them for further reference.

We investigate action execution failures and propose a method to derive hypotheses through learning. There are two types of hypotheses derived from execution monitoring. The first type corresponds to hypotheses on safe action types in different contexts, and the second one to hypotheses for failure cases. The former type of hypotheses are used to update the planning domain to guide future planning processes. Therefore, the robot gains experience on both correct execution types of actions and when executions fail. These hypotheses are framed by the observed features of the objects in interest and the relevant world states. Since the relevant facts in the domain are also represented in a hypothesis, both generalized and specialized conclusions can be made.

Our approach is based on a lifelong experience-based learning process. We use Inductive Logic Programming as the learning method to frame hypotheses for both efficient execution types and failure situations. ILP learning provides first-order logical representations of the derived hypotheses that are useful for reasoning and planning processes. Furthermore, this approach can use background knowledge to represent more advanced rules. Partially specified world states can also be easily represented in these rules. All these advantages of ILP make this approach superior to the attribute-based learning approaches. Experience gained through incremental learning is used as a guide to the future decisions of the robot for robust execution. Failure situations of actions on specific contexts are represented by hypotheses that are, then, used to compute the costs. Adaptive cost computations for the failed actions make the overall system robust by blocking the selection of actions that may fail.

The performance of the ILP process is analysed by setting up several experiment environments including real world and simulation experiments. The real world experiments are performed on a Pioneer 3DX robot. Hypotheses produced for the real world experiments by both the original and improved ILP are compared and discussed. In the simulation experiments, two different cases are considered. In the first case, random observations are generated for predefined hypotheses, and then, the performance of the learning system is analysed by dividing these observations into

training and test sets. The results of ILP are compared to that of the attributed-based learners, *ID3* and *Bayes Network* classifiers in the simulation experiments. In the second analysis, the powerful features of the ILP such as background knowledge usage are investigated. The results reveal that the ILP is better than attribute-based learners in several ways, and the hypotheses framed for failure cases are sound and ensure safety in future tasks of the robot.

1. GİRİŞ

Bilgi gösterimi, planlama, çıkarsama, makine öğrenmesi ve bilgisayarlarla görü alanlarındaki gelişmeler robotlar için daha önce zor olarak nitelendirilen görevlerin (Bicchi, 2000) başarıyla yerine getirilmesine sebep olmuştur. Günümüzde robotlar çok daha karmaşık görevlerde (Beetz ve diğ., 2010) yer almaktadır. Hem algılama hem de nesnelerle etkileşimdeki son dönemlerdeki gelişmeler bilişsel robotların hem insana özgü görevleri hem de uzun vadeli Mars keşfi gibi insan yeteneklerini aşan görevleri yürütmesine olanak sağlamıştır.

Bilişsel bir robot, problem çözmek ve hedeflerini gerçekleştirmek için plan yapmak, dinamik durumlarda kararlarını değiştirmek ve deneyimlerinden öğrenmek gibi yeteneklere sahip olmalıdır. Problem çözmek ve plan yapmak, verilen görevlerin gerçekleştirilebilmesi için son derece önemlidir. Literatürde bulunan planlayıcılar, robotların hedeflerine ulaşması için izlenmesi gereken eylem akışını bulmak için sıkça kullanılır. Bu planlayıcılar plan üretmek için tanım kümesi bilgisini (başlangıç/hedef durumları ve gerçek dünya eylemlerine karşılık düşen operatörler) kullanır. Planda yer alan eylemlerin robot tarafından gerçek dünyada yürütülmesi esnasında bir takım hatalar oluşabilir. Bu hatalar yeniden planlama ile kotarılabilir. Ancak tanım kümesinin gerçek dünya temsili ile sembolik temsili arasında bazı farklar olabilir. Özellikle eylemlerin gerçek etkileri tam olarak temsil edilmediğinde, bir hata oluştuğunda planlayıcı geçerli bir plan üretemeyebilir. Bu tür durumlarla başa çıkabilmek için robotun dünya modelini güncellemesi ve ortam ile ilgili çıkarımlar yapabilmesi gerekmektedir. Ayrıca robotun gelecekteki kararlarını daha verimli verebilmesi için öğrenme yeteneğine de sahip olması gerekir.

Gürbüz görev yürütme; sürekli planlama, yürütme, gözlemleme, çıkarsama ve öğrenme süreçlerinin birbirleriyle entegre olmalarını gerektirir. Bu tez çalışmasında çözülmeye çalışılan problem robotların görev yürütmesi esnasında gürbüzlüğü sağlamak için eylem yürütme hatalarından öğrenmeyi gerektirmektedir. Problemin çözümü

için kullanılan yaklaşım yaşam boyu deneyimsel öğrenmeye dayanmaktadır. Öğrenme ile elde edilen deneyimlerin robotu gelecek kararlarında da yönlendirmesi sağlanmıştır. Böylelikle robotlar uyarlamalı bir planlama stratejisi ile hataları kotarabilmektedir. Hem yürütme modelleri hem de hata durumları için hipotez üretmek amacıyla Tümevarımsal Mantıksal Programlama (Inductive Logic Programming) yöntemi kullanılmıştır.

1.1 Tez Çalışmasının Katkıları

Bu tez çalışmasında önerilen yaklaşım, öğrenme sürecinin uygulanması bakımından önceki çalışmalardan farklıdır. Önerilen yaklaşımda robotun gelecek görevlerindeki başarımını arttırmak için başarısızlığa neden olan durumlar geribesleme olarak kullanılır. Hipotez üretirken mevcut durumun içerikleri (eylem, ilgilenilen nesne ve ilişkileri) kullanıldığı için nitelik tabanlı bir öğrenme yöntemi yerine bilgi tabanlı bir öğrenim yöntemi kullanılmıştır. Bu yöntemde hipotezler, yüklem mantığı gösterim dili ile ifade edilebilirler ve öğrenme sürecinde önsel bilgileri işleyebilirler. Tez çalışmasının bir diğer katkısı da hipotezler üzerinde bilgi birikimi kullanılarak daha gerçekçi çıkarımlar yapılmasına imkan tanımasıdır. Üstelik model tabanlı hata belirlemenin mümkün olmadığı durumlar için de çözüm üretilebilmektedir. Deneyim tabanlı öğrenme ve hata durumlarında planlayıcıya alternatif çözümler sunmak için kullanılan öğrenme tabanlı yönlendirme ile yürütme görevlerinde gürbüzlük sağlanabileceği gösterilmiştir.

2. PLANLAMA

Planlama, eylemlerin sırasına karar vererek bir sistemi başlangıç durumundan istenen duruma getiren problem çözme tekniğidir (Schalkoff, 1990). Burada uğraşılan planlama problemleri çeşitlilik gösterir. Bir paketi bir şehirden bir başka şehire götürürken izlenmesi gereken yol ya da yatırım yaparken maksimum kazanç minimum risk için yapılması gereken tercihler birer planlama örneğidir.

Planlama probleminde amaç bir hedef kümesini gerçekleştirmek, soyut bir işi yapmak ya da bir amaç fonksiyonunu en iyilemek gibi birçok işi kapsayabilir. Planlama problemleri kendi içinde birçok özelleşmiş problem barındırır. Bu problemler yol planlaması, ardışıl planları sıralama, üretim planlaması ve zamanlama gibi problemlerdir (Smith ve diğ. 2000).

Bu bölümde öncelikle planlama ile ilgili kuramsal bilgiler verilecek, daha sonra planlama konusunda son dönemde yapılan çalışmalardan bahsedilecektir.

2.1 Planlama Ortamının Temsil Edilmesi

Uygun bir problem temsilinin geliştirilmesi planlama açısından oldukça önemlidir. Bu temsil şunları gerektirir (Schalkoff, 1990):

1. Problem ortamının tanımlanması

2. Yürütülen bir operatörün (eylemin) ortamı nasıl değiştireceğinin tanımlanması

Planlama ortamının temsili iki alt aşama içerir. İlk aşama planlama tanım kümesinin temsili, ikinci aşama ise planlama probleminin temsilidir. Bu temsiller ve temsilleri gerçeklemek için kullanılan planlama dilleri bu bölümde incelenecektir.

2.1.1 Planlama tanım kümesinin temsili

Tanım kümesi temsilinde genel olarak 4 farklı özellikten bahsedilir. Bu özellikler:

1. Nesneler
2. İfadeler
3. Durumlar
4. Operatörler

Bu özellikleri bir örnek üzerinde incelemek amacıyla, örnek ortam; boyama problemi (Yang, 1997) olarak seçilmiştir. Bu probleme göre ortamda yer alan kapı, merdiven gibi nesneler boyanmalı ve ayrıca merdiven kullanılarak tavan boyanmalı. Ortamdaki nesneler; kapı, merdiven, fırça, boya ve tavan. Ortamın durumunu temsil etmek için birinci-dereceden mantık gösterimi (first-order logic) kullanılabilir. Bu gösterime göre nesnelerin başlangıçtaki durumları şu şekilde ifade edilebilir: kuru(kapı), kuru(merdiven), kuru(tavan), ¬boyalı(kapı), ¬boyalı(merdiven) ¬boyalı(tavan). Ayrıca ortamda değişiklik yapmak için bazı operatörlerin tanımlanması gerekir. Her operatörün ortama uygulanabilmesi için bir ön koşulun sağlanmış olması gerekir ve herhangi bir operatörün ortam üzerinde bir etkisi vardır. Boyama örneği için bazı operatörler, bu operatörlerin ön koşulları ve ortam üzerindeki etkileri Çizelge 2.1'de verilmiştir.

Çizelge 2.1: Boyama problemi için bazı operatör tanımları.

Operatör	Ön Koşul	Etki
fırca-al	el-bos	¬(el-bos)
		fırca-elde
fırca-boya	fırca-elde	fırca-boyali
tavan-boya	fırca-elde	¬kuru(tavan)
	kuru(merdiven)	boyali(tavan)
	fırca-boyali	¬(fırca-boyali)
merdiven-boya	fırca-elde	¬kuru(merdiven)
	fırca-boyali	boyali(merdiven)
		¬(fırca-boyali)
fırca-birak	fırca-elde	el-bos

Operatörler için ön koşul ve etki dışında maliyetten de bahsedilebilir. Bir operatörü gerçekleştirmenin maliyeti vardır ve bu maliyet her operatör için farklı olabilir. Örneğin fırçayı boyaya batırmak gibi kolay bir işin maliyeti 1 birim iken tavanı boyamak gibi daha zor bir işin maliyeti 5 birim olabilir.

2.1.2 Planlama probleminin temsili

Planlama probleminin temsilinde iki temel nokta bulunmaktadır. Bunlardan birincisi başlangıç durumu, ikincisi ise hedef durumudur. Bir önceki bölümde verilen boyama problemi göz önünde bulundurulursa başlangıç ve hedef durumlarını şu şekilde tanımlanabilir:

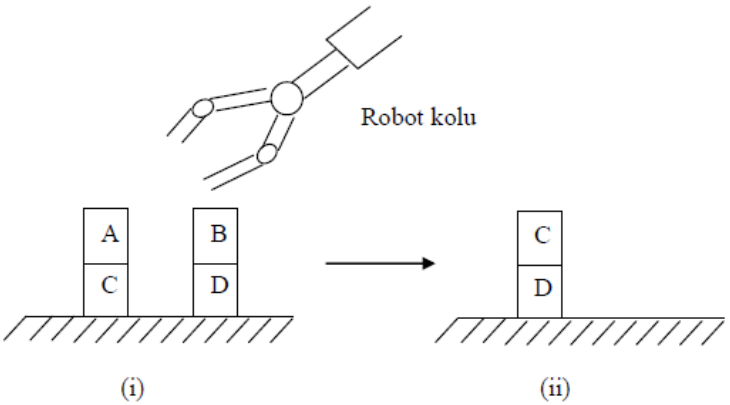
Başlangıç durumu: {kuru(kapi), kuru(merdiven), kuru(tavan),¬boyali(kapi), ¬boyali(merdiven), ¬boyali(tavan), el-bos}

Hedef durumu: {boyali(kapi), boyali(merdiven), boyali(tavan}

Burada amaç daha önceki bölümde bir kısmı verilen operatörler kullanılarak başlangıç durumunda bulunan ortamı hedef durumu haline getirmektir.

2.1.3 Örnek durum - blok inşası (blocks world)

Bu bölümde, blok inşası probleminin temsili ele alınacaktır.



Şekil 2.1: Blok dünyası problem temsili.

Şekil 2.1(i) blok inşası probleminin başlangıç durumunu, (ii) ise problemin hedef durumunu göstermektedir. Bu örnekte problem, hedef durumunu gerçekleştirmek

için robot kolunun yapması gereken eylemlerin sırasına karar vermektir. Problemin başlangıç ve hedef durumlarının tanımları şu şekildedir:

Başlangıç durumu:

(on A C); A, C'nin üzerinde

(on B D)

(ontable C); C masanın üzerinde

(ontable D)

(clear A); A'nın üzeri boş

(clear B)

(empty); Robot kolunun durumu

Hedef durumu:

(on C D)

(clear C)

(ontable D)

Problemde kullanılabilen eylemlerin tanımları ise Çizelge 2.2'de verilmiştir.

2.1.4 Planlama dilleri

Genellikle kullanılan 3 farklı planlama dilinden bahsedilebilir. Bunlar:

1. STRIPS (Stanford Research Institute Problem Solver)
2. ADL (Action Description Language)
3. PDDL (Planning Domain Definition Language)

Bu kısımda öncelikle bu diller tek tek incelenecek daha sonra farkları verilecektir.

STRIPS: Bu dilde 3 farklı alt ifade bulunur: durumların temsili, hedeflerin temsili ve eylemlerin temsili.

Çizelge 2.2: Blok inşası operatör tanımları.

Operatör	Ön Koşul	Etki
(pickup x)	(ontable x)	¬(ontable x)
	(clear x)	¬(clear x)
	(empty)	¬(empty)
		(hold x)
(putdown x)	(hold x)	¬(hold x)
		(ontable x)
		(clear x)
		(empty)
(takeoff x y)	(on x y)	¬(on x y)
	(clear x)	¬(clear x)
	(empty)	¬(empty)
		(clear y)
		(hold x)
(puton x y)	(hold x)	¬(hold x)
	(clear y)	¬(clear y)
		(on x y)
		(clear x)
		(empty)

(i) Durumların temsili: Planlayıcılar ortamı mantıksal koşullara ayrıştırır ve bir durumu olumlu önermelerin birleşimi olarak temsil eder (Russell ve diğ., 1996). Örneğin daha önce bahsedilen boyama probleminde bir durum STRIPS dilinde kuru(kapi) Λ kuru(merdiven) şeklinde ifade edilebilir.

(ii) Hedeflerin temsili: Hedefler de durumlar gibi temsil edilir. Örneğin boyali(kapi) Λ boyali(merdiven) gibi.

(iii) Eylemlerin temsili: Genel olarak bir eylem 3 parçadan oluşur (Russell ve diğ., 1996):

- Eylem ismi ve parametre listesi. Örneğin blok inşası probleminde kullanılan *(takeoff x y)* gibi.
- Ön koşul. Olumlu önermelerin birleşiminden oluşur. Eylemin yürütülebilmesi için bu koşulun sağlanmış olması gerekir.
- Etki. Eylemin yürütüldüğünde ortamın nasıl etkileneceğini belirtir.

Action (takeoff (x, y))

Precond: on (x, y) Λ clear (x) Λ empty

Effect: ¬on (x, y) Λ ¬clear (x) Λ ¬empty Λ clear (y) Λ hold (x)

Yukarıda bir eylemin bütün olarak STRIPS dilinde temsili yer almaktadır.

ADL: Bu dil STRIPS dilinin geliştirilmiş hali olarak düşünülebilir. Strips dilinde olduğu gibi başlangıç durumu, amaç ve eylemlerden oluşur. Her eylem STRIPS'de taşıdığı 3 özelliği burada da taşır.

Action (takeoff (x, y))

Precond: on (x, y) Λ clear (x) Λ empty Λ Block (x) Λ Block (y) Λ (x $\neq$ y)

Effect: ¬on (x, y) Λ ¬clear (x) Λ ¬empty Λ clear (y) Λ hold (x)

Yukarıdaki örnek, takeoff eyleminin ADL dili ile temsilidir.

PDDL: Planlama problemleri için standart bir dil yaratma çabası sonucu ortaya çıkmıştır (Russell ve diğ., 1996).

(: action takeoff

(: parameters ?x ?y)

(: preconditions (and (on ?x ?y) (clear ?x) (empty)))

(: effect (and (not (on ?x ?y)) (not(clear ?x))(not(empty))(clear ?y)(hold ?x))

Yukarıdaki örnek, takeoff eyleminin PDDL dili ile temsilidir.

STRIPS, ADL ve PDDL planlama dillerinin genel farkları Çizelge 2.3'de yer almaktadır.

2.2 Planlama Yöntemleri

Bu bölümde planlama amacıyla kullanılan değişik yöntemler tanıtılacaktır. İncelenecek yöntemler sırasıyla:

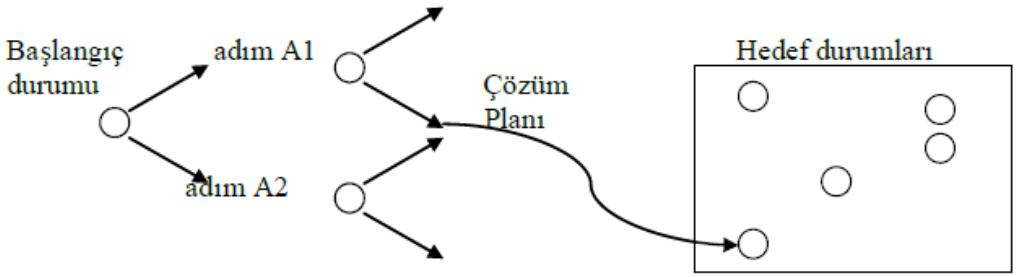
1. İleri durum uzayı arama algoritması
2. Geri durum uzayı arama algoritması
3. Kısmi sıralı arama

4. Hiyerarşik görev ağı planlama

5. Grafik planlama

2.2.1 İleri durum uzayı arama algoritması

Bu yöntemde planlayıcı işe başlangıç koşullarını içeren ortamdan başlar. Planlayıcı, o andaki durumda ön koşulları sağlanan bir operatör seçer, operatörün etkilerini ortama ekleyerek ve negatif etkileri ortamdan çıkararak yeni durumu oluşturur. Böylece bütün hedefler başarılana kadar arama devam eder. İleri durum uzayı arama yönteminde problem, genellikle herhangi bir durumda uygulanabilecek olan eylem sayısının çok fazla olmasıdır. Bu durum, arama uzayının boyutunun sonuca ulaşmayı engelleyecek kadar büyük olmasına neden olur (Smith ve diğ., 2000). İleri durum uzayı arama algoritması Algoritma 1'de verilmiştir. Algoritmanın işleyiş biçimi ise Şekil 2.2'de yer almaktadır.



Şekil 2.2: İleri durum uzayı arama yöntemi.

Algoritma 1 IDU

Input: *simdiki-durum*
Output: *plan*
if hedefler = *simdiki-durum* **then**
  return *plan*
end if
simdiki-durum'da ön koşulu sağlanan bir eylem *A* seç
if seçilebilecek eylem yok **then**
  return *fail*
end if
A eyleminin etkileri ile uyuşmayan *simdiki-durum*'un önermelerini sil
A eyleminin etkilerini *simdiki-durum*'a ekle
IDU(*s', plan* | *A*)

Çizelge 2.3: STRIPS, ADL ve PDDL karşılaştırması.

STRIPS	**ADL**	**PDDL**
Kapalı ortam varsayımı: Bahsedilmeyen önermeler yanlış kabul edilir.	Açık ortam varsayımı: Bahsedilmeyen önermeler bilinemez kabul edilir.	Kapalı ortam varsayımı: Bahsedilmeyen önermeler yanlış kabul edilir.
Sadece olumlu önermeler:	Olumlu ve olumsuz önermeler:	Olumlu ve olumsuz önermeler:
Fakir Λ Yaşlı	Fakir Λ ¬Yaşlı	and (Fakir) (not (Yaşlı))
A Λ ¬B:	A Λ ¬B:	and (A) (not (B)):
A'yı ekle, B'yi çıkar.	A ve ¬B'yi ekle, ¬A ve B'yi çıkar.	A ve ¬B'yi ekle, ¬A ve B'yi çıkar.
Amaç cümlesi ∀ ve ∃ içermez.	Amaç cümlesi ∀ ve ∃ içerir.	Amaç cümlesi ∀ ve ∃ içerir.
Amaç cümlesi ∧'lerden oluşur. ∨ içermez.	Amaç cümlesi ∧ ve ∨ içerir.	Amaç cümlesi ∧ ve ∨ içerir.
Değişkenlerin tipleri yoktur.	Değişkenlerin tipleri vardır.	Değişkenlerin tipleri vardır.
= işlemi yoktur.	= işlemi vardır.	= işlemi vardır.

2.2.2 Geri durum uzayı arama algoritması

Bu algoritmada hedeflerden birisini gerçekleyen bir eylem seçilir ve plana eklenir. Gerçeklenen hedef, hedef kümesinden silinir ve seçilen eylemin ön koşulu alt hedef olarak kümeye eklenir. Bu işleme kümedeki alt hedeflerin hepsi başlangıç koşulunun alt kümesi olana kadar devam edilir (Smith ve diğ., 2000). Algoritma 2'de genel yapı verilmiştir.

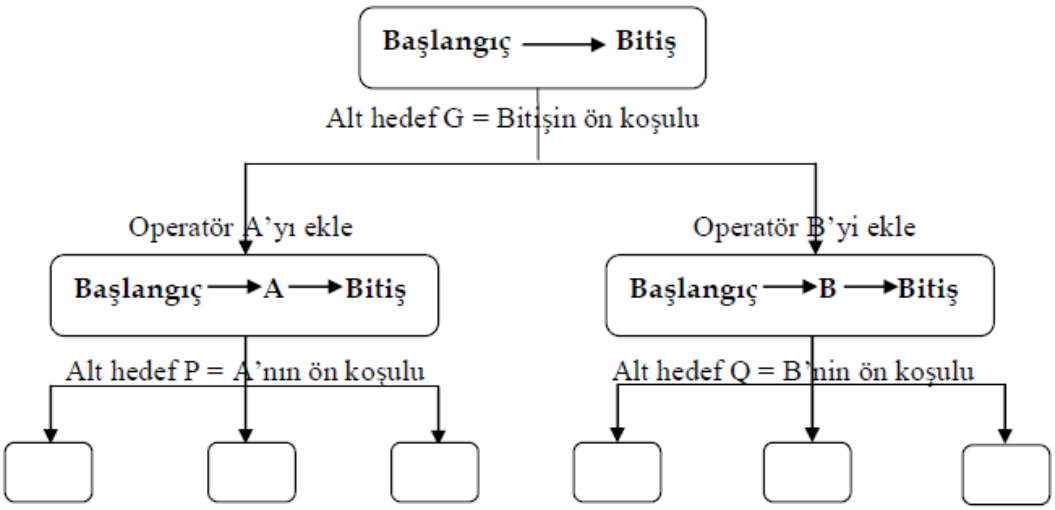
Algoritma 2 AltHedef

Input: *hedefler, kisitlar*
Output: *plan*

if *kisitlar* uyumsuz **then**
  return *fail*
end if
if *hedefler* = baslangic−kosulari **then**
  return *plan*
end if
Bir gol g seç, $g \in$ *hedefler*
g'yi saglayan bir eylem A sec
AltHedef($hedefler - g + onkosul\ (A), yeni-kisitlar, plan + A$)
if A zaten plana dahil **then**
  AltHedef ($hedefler - g, yeni-kisitlar, plan$)
end if

2.2.3 Kısmi sıralı arama

Planlama problemi için plan üretmenin bir başka yolu, bütün plan bileşenlerini artımlı bir şekilde ekleyerek kısmi sıralı plan oluşturmaktır. Kısmi sıralı planın temel uygulaması olan geri zincirleme algoritmasında, plan adımlarının ön koşulları birer birer sağlanır. Her iterasyonda, sağlanacak olan ön koşul seçilir ve bu ön koşulu sağlayabilecek operatörler ve plan adımları belirlenir. Daha sonra plana yeni bir bağlantı eklenerek ön koşul gerçeklenmiş olur. Eklenen bağlantıya tehdit oluşturan bağlantılar silinir. Bu işlem, planda yer alan her adımın her ön koşulunu sağlayan bir bağlantı oluşturulana ve bütün negatif tehditler kaldırılana kadar devam eder (Yang, 1997).

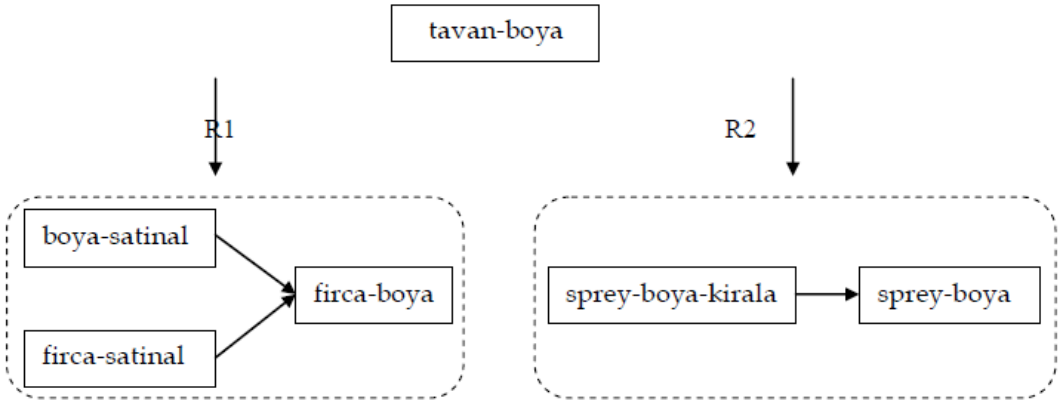


Şekil 2.3: Geri zincirleme.

2.2.4 Hiyerarşik görev ağı planlama

Bu yöntemde, verilen yüksek seviyeli görevler daha küçük alt görevlere indirgenir. Daha sonra alt görevler arasındaki uyuşmazlıklar çözülür. Bu işlem plandaki bütün görevler primitif hale gelene kadar devam edilir (Yang, 1997).

Boyama probleminde yer alan tavan-boya operatörünün hiyerarşik görev ağı planlamasında nasıl indirgendiği Şekil 2.4'te verilmiştir. Algoritma 4'de ise yöntemin sözde kodu verilmiştir.



Şekil 2.4: Operatör indirgeme örneği.

2.2.5 Grafik planlama

Grafik planlamanın temelinde yatan fikir ulaşılabilirlik analizi kullanılarak birbiriyle uyumsuz olan eylem kombinasyonlarını ortadan kaldırmaktır. Başlangıç durumundan başlanarak her adım sonrası oluşabilecek olası önerme kümeleri hesaplanır. Örneğin

Algoritma 3 KSPlan

```
Input: Operatör kümesi O, ilk plan φ_init
Output: plan
AcikListe := { φ_init }
while AcikListe bos do
  φ := AcikListe'de yer alan en düşük maliyetli plan
  φ AcikListe'den silinir
  if Dogru(φ) = Doğru then
    return φ
  else
    if TehditVar(φ) then
      SUCC := TehditCoz(t,φ)
    else
      SUCC := YeniOnKosul(φ)
    end if
    Üretilen bütün SUCC elemanlarını AcikListe'ye ekle
  end if
end while
return fail
```

ilk adım için, bu küme başlangıç koşullarından bir adım sonra ulaşılabilecek önermelerin birleşiminden oluşur (Smith ve diğ., 2000).

Ancak elde edilen önermelerin hepsi birbirleriyle uyumlu olmayabilirler. Grafik planlamada uyumsuz eylem ve önermelerin arasındaki ilişki için karşılıklı dışlama (mutex) kullanılır. Karşılıklı dışlama kuralları şu şekildedir (Smith ve diğ., 2000):

- İki eylem verilen bir adımda şu koşullardan biri sağlanırsa birbirini karşılıklı dışlar:
  - Zıt etkileri varsa
  - Birinin etkisi diğerinin zıt ön koşulu ise
  - Verilen adımda eylemlerin ön koşulları birbirlerini karşılıklı dışlıyorsa
- İki önerme verilen bir adımda şu koşullardan biri sağlanırsa birbirini karşılıklı dışlar:
  - Zıt önermelerse
  - Önermeleri ortaya çıkartan tüm eylem çiftleri bir önceki adımda birbirlerini karşılıklı dışlıyorlarsa

Algoritma 4 HTNPlan

```
Input: N
Output: plan

if N uyuşmazlık içeriyor then
  if uyuşmazlık çözülemiyor then
    return fail
  else
    uyuşmazlığı çöz
  end if
end if
if N sadece primitif görevler içeriyor then
  return N
end if
N içinden primitif olmayan bir görev ( t ) seç
t → E
N' ← N'de t ile E'yi değiştir
HTNPlan ( N' )
```

Grafik planlama için elde edilen çizge daha sonra kısıt sağlama problemine (Constraint satisfaction problem - CSP) dönüştürülerek standart CSP arama teknikleri ile çözülebilir (Do ve Kambhampati, 2000).

2.3 Son Dönemde Planlama ile ilgili Yapılan Çalışmalar

Cubek (2011) çalışmalarında sembolik planlamada Metric-FF sayısal planlayıcısını, hareket planlamada OpenRave planlama ve simülasyon sistemini kullanmışlardır. OpenRave IKFast robot kinematik derleyicisi ile kinematik denklemleri çözerek hareket planlaması yapar. Bouguerra ve diğ. (2007) çalışmalarında görev ve bilgi planlamada, TLPlan zamansal planlayıcısının olasılıksal ve koşullu bir uzantısı olan PTLPlan planlayıcısını kullanmışlardır. Bilgi planlama, gerekli bilgileri toplamak için hareket ve gözlem eylemlerini içeren bir plan oluşturur. PTLPlan inanç durumları (belief states) alanında arama yapar. Bir diğer çalışmada (Talamadupula ve diğ., 2011) Sapa metrik-zamansal planlayıcının bir uzantısı olan SapaReplan planlayıcısı kullanılmıştır. SapaReplan PSP ile yeniden planlamayı ele alır. Nguyen ve diğ. (2011) çalışmalarında kullanıcı tercihlerinin dahil olduğu planlamalarda, tek bir plan bulmak yerine bir plan kümesi bulmanın daha etkin bir çözüm olduğunu savunmuşlardır. Bir plan kümesinden plan seçimi kullanıcı tarafından zor bir işlem

olacağından, önerilen sistem temsili bir alt küme belirlemektir. Araştırmacılar Metric-LPG planlayıcısını ve zamansal planlamayı her bölgede daha iyi bir çözüm olacak şekilde bölmek için entegre dışbükey tercih (Integrated Convex Preference) modelini kullanmışlardır. Aker ve diğ. (2011) çalışmalarında sembolik görev planlama, hareket planlama ve nedensel çıkarsama mekanizmalarını birbirleri ile sıkı iletişim içinde kullanan bir sistem sunulmaktadır. Bu çalışmada sembolik planlama ve çıkarsama için CCALC'tan faydalanılmakta ve hareket planlama algoritması olarak Hızlı-tarayıcı Rasgele Ağaçlar (Rapidly-exploring Random Trees - RRTs) kullanılmaktadır. Burbridge ve Dearden (2012)'in çalışmasında da klasik bir sembolik planlayıcı ile birlikte RRT'den faydalanılmaktadır. Bir diğer çalışmada da sembolik planlama (HATP- İnsan-Farkında Görev Planlayıcı - Human-Aware Task Planner) ve geometrik planlama bir robot asistan sistemi üzerinde tümleştirilmektedir. Hiyerarşik regresyon-tabanlı planlama (Hierarchical regression-based planning) algoritmasına dayalı bir çalışma da bulunmaktadır. Farklı bir çalışma olarak Ugur ve diğ. (2011) çalışmasında ise ileri yönde çıkarsamaya dayalı planlamada önkoşulsuz ve klasik durum-geçiş kuralları olmayan eylemler üzerinden öngörüsel olarak durum güncellemesi yapan bir planlama yaklaşımı sunulmaktadır. Hanheide ve diğ. (2010) çalışmasında robotlar belirlenen kısıtlara göre kendilerine hedefler üretmekte ve bu hedefler arasından hedef durumun bilgi kazanımı değerinin o hedefin maliyetine oranına bakılarak kendilerine uygun hedefleri seçmektedirler.

3. ÖĞRENME

Öğrenme, robotun çevresiyle etkileşimi sonucu çeşitli gözlemler elde etmesi ile başlar. Sadece bu gözlemlerin kaydedilmesi ile öğrenme olabileceği gibi karmaşık hipotezlerin üretimi şeklinde de olabilir. Bu bölümde önce tümevarımsal öğrenme (inductive learning) ile ilgili kuramsal bilgiler verilecek ve daha sonra öğrenme konusunda literatürde yer alan çalışmalardan bahsedilecektir.

3.1 Tümevarımsal Öğrenme

Deterministik denetimli öğrenmede, bir algoritmaya giriş olarak bilinmeyen bir fonksiyonun belli giriş değerleri için ürettiği sonuçlar verilir ve bu bilinmeyen fonksiyon ya da ona yakın bir fonksiyon bulunmaya çalışılır. Daha biçimsel olarak söylemek gerekirse, elimizde $(x, f(x))$ ikilisi yer almaktadır. Burada x giriş vektörünü, $f(x)$ ise x'e uygulanan fonksiyonun çıkış vektörünü temsil etmektedir. Tümevarımsal öğrenmenin görevi, f fonksiyonun örnek kümesi verildiğinde f fonksiyonuna yakınsayan bir h fonksiyonu elde etmektir. h fonksiyonu *hipotez* olarak adlandırılır. Burada zor olan kısım h fonksiyonunun f fonksiyonuna yakınsayıp yakınsamadığını söyleyebilmektir. İyi bir hipotez verilen örnekleri iyi bir şekilde tahmin edebilmelidir (Russell ve diğ., 1996).

3.1.1 Öznitelik tabanlı öğrenme

Pratik uygulamalar açısından; nitelik tabanlı öğrenme, gözetimli öğrenme algoritmalarının temelini oluşturmaktadır (Bratko ve diğ., 1996). Nitelik tabanlı sınıflandırıcılar için en temel yaklaşımlardan biri olan *Karar Ağaçları* ve *Bayes Network* sınıflandırıcıları, bu tez çalışmasında başarım ölçümü için kullanılmıştır.

Karar Ağaçları, en basit ama aynı zamanda en başarılı öğrenme algoritmalarından biridir. Bir karar ağacı giriş olarak öznitelikler kümesi tarafından tanımlanan bir nesne veya durum bilgisini alır. Sonuç olarak giriş için tahmin edilen bir çıkış değerini üretir.

Karar ağacının çıktısı, bir dizi test gerçekleştirilerek üretilir. Ağacın her ara düğümü bir niteliği test eder. Düğümden çıkan dallar ise testin olası değerleri olarak etiketlenir. Ağaçtaki her yaprak düğümü, o düğüme ulaşılırsa döndürülecek değeri tutar (Russell ve diğ., 1996).

Bayes Network, rastgele değişkenler kümesi ile bu değişkenlerin koşullu bağımlılıklarını gösteren yönlü bir graftır. Değişkenler ağın düğümlerini oluşturur ve her düğüm annesi ile ilişkili koşullu olasılık dağılımına sahiptir (Russell ve diğ., 1996).

Öznitelik tabanlı öğrenmenin pratik uygulamalarda başarılı olmasını sağlayan avantajları şu şekilde sıralanabilir (Bratko ve diğ., 1996):

- Göreli basitlik
- Hesaplama verimliliği
- Kullanıcılar tarafından kolay anlaşılırlığı ve uygulanabilirliği
- Gürültülü ve eksik verilerle başa çıkılabilmesi

Nitelik tabanlı öğrenmenin avantajlarının yanında ciddi sınırlamaları da vardır:

- Önceki bilgi birikimi oldukça sınırlı bir şekilde ifade edilebilir.
- İlişkisel açıklamalardaki eksikliği yüzünden bazı sistemlerde kullanıma uygun değildir.

3.1.2 Tümevarımsal mantıksal programlama

Tümevarımsal Mantıksal Programlama (Inductive logic programming - ILP) oluşturulan hipotezleri lojik olarak ifade ederek, tümevarımsal yöntemler ile birinci dereceden mantık (first-order logic) temsilini birleştirir. ILP'nin 3 önemli avantajı vardır (Schalkoff, 1990):

1. Bilgi tabanlı tümevarımsal öğrenme problemlerine titiz bir yaklaşım sunar.
2. Örneklerden birinci dereceden teoriler üretmek için algoritmalar sunar. Böylelikle öznitelik tabanlı algoritmaların uygulanmasının zor olduğu sistemlerde kullanılabilir.

3. ILP ile insanlar tarafından kolaylıkla okunabilen hipotezler üretilir. Böylece deney yapma, hipotez üretme ve tartışmanın bilimsel çevrimlerinde kullanılabilir. Bu tür bir kullanım sinir ağları gibi *kara kutu* sınıflandırıcılarda mümkün değildir.

ILP'de problem çözümü için genel olarak *yukarıdan aşağıya yaklaşım (top-down approach)* ve *tersten tümdengelim (inverse deduction)* olmak üzere 2 temel yaklaşım vardır.

3.1.2.1 Yukarıdan aşağıya yaklaşım

Bu yaklaşım genel bir kural üretmekle başlayıp, eldeki verilerle uyumlu olacak şekilde bu kuralın özelleştirilmesiyle devam eder. Algoritmanın sözde kodu Algoritma 5, 6 ve 7'de yer almaktadır.

Algoritma 5 FOIL($ornekler, hedef$) **çıktı:** Horn ifadeleri

Giriş: örnek kümesi, $ornekler$ ve hedef yüklemi, $hedef$
yerel değişkenler: cümle kümesi, $ifadeler$
while $ornekler$ pozitif örnek içeriyor **do**
  $ifade$ = YENI-CUMLE($ornekler, hedef$)
  $ifade$ tarafından kapsanan örnekleri $ornekler$ kümesinden sil
  $ifade$ değişkenini $ifadeler$ kümesine ekle
end while
return $ifadeler$ kümesi

Genel algoritma olumlu örneklerin belli bir kümesi kapsanana ve olumsuz örneklerin hepsi dışlanana kadar cümle üretmeye devam eder. Cümle tarafından kapsanan olumlu örnekler eğitim kümesinden çıkarılır. Bu süreç kümede olumlu bir örnek kalmayıncaya kadar devam eder.

3.1.2.2 Tersten tümdengelim

ILP'de kullanılan ikinci önemli yöntem tümdengelim kullanılarak yapılan kanıtlama işleminin terse çevrilmesi ile oluşturulmuştur. *Tersten tümdengelim*'in dayandığı temel yaklaşım, eğer örnek bir sınıflandırma *Geçmiş* Λ *Hipotez* Λ *Açıklamalar* sırasını izliyor ise, çözümleme kullanılarak kanıtlama yapılabiliyor olmasıdır. Eğer kanıtlama ters yönde çalıştırılabilirse, bir hipotez elde edilebilir. Bu durumda asıl yapılması gereken çözüm işlemini ters yöne çevirebilecek bir yol bulmaktır (Russell ve diğ., 1996).

Algoritma 6 YENI-CUMLE($ornekler, hedef$) **çıktı:** Horn ifade

yerel değişkenler: $ifade$, l, cümleye eklenecek deyim
$genisletilmisOrnekler$, genişletilmiş örnek kümesi
$genisletilmisOrnekler = ornekler$
while $genisletilmisOrnekler$ negatif örnek içeriyor **do**
l = DEYIM-SEC(YENI-DEYIMLER($ifade$),$genisletilmisOrnekler$)
l deyimini $ifade$ cümlesine ekle
$genisletilmisOrnekler$ = ORNEK-GENISLET ile üretilen örnek kümesi
end while
return $ifadeler$ kümesi

Algoritma 7 ORNEK-GENISLET($ornek, deyim$)

if $ornek$, $deyim$ i doğrularsa **then**
return $deyim$ de yer alan her değişkenin her bir olası sabit değeri ile genişletilmiş $ornek$ kümesi
else
return $\varnothing$
end if

Yukarıdan aşağıya yaklaşım kullanılırken verilen gözlemler için daha kısa ve genel hipotezler üretilirken, tersten tümdengelimde daha özel ve uzun hipotezler üretilir. Yukarıdan aşağı yaklaşımın dezavantajı, pozitif bir örneği kapsamayan cümleler üretilirken zaman kaybı yaşanabilmesidir. Tersten tümdengelim yönteminin dezavantajı ise bilgi birikiminin bu yöntemde verimsiz kullanımı nedeniyle bu bilginin tekrar düzenlenmesinin gerekliliğidir. Bu düzenleme işlemi de problem uzayının genişlemesine yol açar (Zelle ve diğ., 1994).

3.2 Son Dönemde Robot Öğrenmesi ile ilgili Yapılan Çalışmalar

Literatürde robot sistemlerinde öğrenme konusunda çeşitli yöntemler kullanılmaktadır. Kober ve diğ. (2012) önerdiği sistemde ödül ağırlığına bağlanımla (reward weighted regression) hızlandırılmış bir takviyeli öğrenme (reinforcement learning) algoritması yardımıyla farklı durumlar robotun temel motor ilkelleriyle ilişkilendirilmektedir ve parametrik fonksiyon yaklaşıklamasının kısıtlarına karşı maliyet düzenlemesine dayalı çekirdek bağlanımından (Cost-regularized Kernel Regression - CrKR) yararlanılmaktadır. Gerçek dünya verilerinin dinamik Bayes ağlarıyla modellendiği bir diğer sistemde ise takviyeli öğrenme parçacık filtresi (particle filter) ile beraber kullanılmaktadır. J. Elfring ve Bruyninckx (2011) çalışmasında ortamdaki

nesnelerin beklenen konum ve hızlarını belirlemek üzere çoklu hipotez filtrelerinden (multiple hypothesis filters - MHF) yararlanılmaktadır. Robotun eylemleriyle nesneler arasındaki ilişkilerin (örn., uygun kavrama pozisyonlarının belirlenmesi) öğrenilmesi konusunda da çeşitli çalışmalar bulunmaktadır. Kroemer ve diğ. (2012) çalışmasında çekirdeğin lojistik bağlanımından (kernel logistic regression - KLR) faydalanılarak motor ilkelleri ile nesnelerin bölümleri arasındaki ilişkiler belirlenmektedir. Ugur ve diğ. (2011) önerdiği sistemde aynı amaçla k-means demetleme algoritmasıyla beraber destek vektör makinelerinden (SVM) faydalanılmaktadır. Aynı algoritmalardan yararlanılan Ridge ve diğ. (2009) çalışmasındaysa k-means demetleme öncesinde kendini düzenleyen harita (self-organizing map – SOM) eğitimi yapılmaktadır ve destek vektör makinesiyle sınıflandırma öncesinde k en yakın komşu yöntemi ile demetleme yapılmaktadır. Burbridge ve Dearden (2012)'in önerdiği sistemde gözetimli öğrenme yöntemleri uygulanarak sembolik ve geometrik durumlar arasındaki ilişkiler öğrenilmektedir. Cantrell ve diğ. (2012) hata durumları ile karşılaşıldığında, ortamdaki bir uzman (insan), robota bu hatayı nasıl kotarabileceği bilgisini doğal dil komutları ile söylemektedir. Robot bu komutları bilgi tabanına kaydedip, ileride tekrar aynı durumlar ile karşılaşıldığında bu bilgilerini kullanarak hata durumlarını kotarmaktadır. Bazı durumlarda robotlar ortam hakkında tam bilgiye sahip iken bazı durumlarda bilgileri tam değildir. Bu tür durumlarda eksik olan bilgiler robot tarafından öğrenilmeli ve kullanılmalıdır. Bir çalışmada verilen görevlerin robotlar tarafından yerine getirilmesi için CRAM adında bir yazılım aracı önerilmiştir (Beetz ve diğ., 2010). Bu araç yardımı ile "masayı yemek için hazırla" gibi karmaşık robot kontrol programları anlaşılabilmektedir. Sistem CRAM adı verilen plan dilini kullanmaktadır. Ayrıca sistem KnowRob aracını da kullanarak öğrenme ve ortama adaptasyon sağlayabilmektedir. Chitta ve diğ. (2012) çalışmalarında bir nesnenin tutma noktalarını belirlemeye çalışmışlardır. Bilinen nesnelerin tutma noktaları, önceden oluşturulmuş ve nesnelerin 3 boyutlu modellerinden oluşan bir veritabanı yardımıyla belirlenirken; bilinmeyen nesnelerin tutma noktaları nokta bulutu verisindeki derinlik bilgisi kullanılarak öğrenilir.

Robotların insanlar tarafından gerçekleştirilen eylemleri öğrenmesi konusunda da bazı çalışmalar yapılmıştır. Bu konudaki bir çalışmada Cubek (2011) gösteriden

öğrenmenin (learning from demonstration - LfD) yüksek seviyeli yaklaşımı olan sembolik kodlama yöntemi kullanılmıştır. Yüksek seviyeli öğrenme daha soyut yeteneklerin ve hedef odaklı görevlerin tanımlanmasına olanak sağlar ve kavramsal alanlar arasında kümeleme yöntemi ile gösteriler arası benzerlikler öğrenilir.

Bir diğer çalışmada Mösenlechner ve Beetz (2011), masa üzerinde bulunan nesneleri tutmaya çalışırken tutma eyleminin parametreleri olarak tanımlanan robotun nerede durması gerektiği ve nesnelerin tutma noktaları gerçek zamanlı öğrenilir. Öğrenilen parametreler için durağanlık (stability), görünürlük ve ulaşılabilirlik önemlidir. Durağanlık, sahnenin belli bir süre Bullet simülasyon motorunda çalıştırılması ile doğrulanır. Görünürlük ve ulaşılabilirlik, masa üzerindeki bir nesnenin diğer bir nesneyi kapatması sonucu robotun nesneyi belirleyip tutmasına engel teşkil edeceği için önemlidir.

Zettlemoyer ve diğ. (2005) tarafından yapılan çalışmada, planlamada kullanılan eylemler olasılıksal STRIPS kuralları olarak tanımlanmıştır ve çalışmanın amacı bir eylem yürütüldükten sonra ortamda oluşan değişiklikleri içeren eğitim örneklerinden bu kuralların öğrenilmesidir. Öğrenme işlemi bir kural setinin arka plan bilgisinden (background knowledge) olasılıksal öğrenilmesi ve yeni bilginin türetilmesini kapsar.

4. BİLİŞSEL ROBOTLARDA GÜRBÜZ GÖREV YÜRÜTME PROBLEMİ

Bir bilişsel robotun, genellikle planlama görevi olarak verilen karmaşık ve bilişsel bir problemi çözmesi beklenir. Robot, planlama problemi ve tanım kümesi bilgisini alıp bir planlayıcı kullanarak problemi çözmeye ve hedeflerine ulaşmaya çalışır. Planlayıcı, planlama sonucu olarak bir dizi eylem üretir. Robot planlanan eylemleri yürütürken çeşitli hatalarla karşılaşabilir. Hata, yeni bir plan üretilerek kotarılabilir. Ancak bazı durumlarda yeniden planlama, problemi çözmek için yeterli olmaz. Robot bu durumlar hakkında çıkarımlar yapmalı ve deneyimlerinden yararlanmalıdır. Bu bölümde bilişsel robotlarda gürbüz görev yürütme için planlama, yürütme, gözlemleme ve öğrenme problemlerinin temsili yer almaktadır.

4.1 Planlama

Bilişsel robotlar, hedeflerine ulaşabilmek amacıyla yürütmeleri gereken eylem dizilerini oluşturabilmek için bir eylem planlama sistemine ihtiyaç duyarlar. Planlama görevi $\Pi = (\Delta, s_0, s_G)$ şeklindeki bir çokuzlu tarafından temsil edilir. Bu çokuzluda Δ planlama tanım kümesini, s_0 başlangıç durumunu, s_G ise hedef durumunu temsil eder. Planlama tanım kümesi ise $\Delta = (T, C, S, O)$ şeklindeki bir çokuzlu ile temsil edilir. T tip kümesini, C tanım kümesi sabitlerinin kümesini, S durum kümesini ve O planlama operatörlerini belirtir. Bir planlama operatörü $o \subseteq O$, ön koşulları ve etkileri ile ($o = \{pre(o), add(o), del(o)\}$) temsil edilir. Başlangıç durumu s_o'dan hedef durumu s_G'ye ulaşmak için seçilen operatörlerin sırasıyla yürütülmesi ile planlama görevi tamamlanır. Herhangi bir $s \subseteq S$ durumunda operatör o'nun seçilebilmesi için önkoşulların sağlanması ($pre(o) \subseteq s$) gerekir. o operatörünün s durumunda yürütülmesinden sonra $s' = add(o) \cup (s \setminus del(o))$ yeni durumuna geçilir.

Gerçek dünya sistemlerinin doğası gereği robotun yürütebileceği eylemler çoğunlukla süreli eylem (*TGP* tipi) şeklindedir (Mausam ve Weld, 2008; Kecici ve Sariel-Talay, 2010). Süreli bir eylem ya belirli ya da rastgele bir süreye sahiptir. Bu yüzden

etkileri ($add(o), del(o)$) eylem yürütülürken bilinmeyen bir noktada ortaya çıkar. Eylemin önkoşulları ($pre(o)$) eylem yürütülmeden önce sağlanmalıdır ve eylem yürütülürken eylemin kendi sonucu olarak değişmediği sürece değiştirilmemelidir. *Anlık* (*Instantaneous*) eylemler ise sadece bir zaman adımında yürütülürler. Zamansal planlayıcılar, verimli bir şekilde *TGP* eylem dizisi oluşturabilirler, bu yüzden gerçek robot görevlerinde kullanmak için daha uygunlardır.

4.2 Yürütme ve Gözlemleme

Geçerli bir plan P oluşturulduktan sonra, robot her a_t eylemini ($a_t \in A \rightarrow o_t$) fiziksel dünyada sırasıyla yürütebilir. Robot, yürütme sırasında herhangi bir problem ile karşılaşmazsa s_G durumuna ulaşmış olur. Ancak belirlenimci olmayan eylemler ve fiziksel ortamlardaki belirsizlikler yüzünden, eylemler yürütülürken çeşitli hatalar ortaya çıkabilir (Karapinar ve diğ., 2012). Bu çalışmada eylem yürütme hataları ele alınmaktadır. Bu hatalardan bazıları robotun ortam hakkındaki yanlış ya da eksik bilgilerinden kaynaklanabilir. Örneğin robot bir nesnenin fiziksel/görsel özellikleri hakkında eksik bilgiye sahip ise, bu nesne üzerinde yürüteceği eylemlerde de hata ile karşılaşabilir (Bouguerra ve diğ., 2007).

Eylemleri başarılı bir şekilde yürütme sürecinde hataları algılama en önemli sorunlardan birisidir. Robot, eylemlerini yürütürken aynı zamanda ortamın durumunu gözlemlemeli ve yürütme zamanı hatalarından çıkarım yapmalıdır (Pettersson, 2005). Robot, donanımında oluşan hataları algılamak için gerekli olan sensörlere sahip olsa da ortamdaki aykırı durumları algılamak için dışalgı (exteroception) gereklidir. Bu durumda bir ya da daha fazla sensörden gelen verileri yorumlamak, hatalardan çıkarım yapabilmek için oldukça önemlidir. Bu nedenle, robotun bir hatayı tespit edebilmesi ve doğru kararlar verebilmesi için, gördüğü sahneyi yorumlaması ve çıkarım araçlarını bu doğrultuda kullanması gerekir.

TGP şeklindeki eylemler yürütülme zamanları boyunca gözlenmelidir. Hem *TGP* hem de *ani* eylemler için eylemlerin beklenen etkilerinin ortamda oluşup oluşmadığını kontrol etmek için eylem sonunda gözlem yapılması gereklidir. Ortamda oluşan anormallik ile yürütme hataları arasında fark vardır. Yürütme hatası doğrudan eylem yürütülürken ya da yürütüldükten sonra gözlenebilir ve doğrudan eylem ve eylemin hedefleri ile

ilgilidir. Anormallik ise sahnede beklenmeyen bir durum ile ilgili olabilir. Ayrıca eylemlerin yan etkileri ya da düzgün olarak yürütülmemeleri sonucunda da ortamda anormallikler oluşabilir.

4.3 Hata Kotarma

Hata tespit edildikten sonra robotun hatayı kotarması gerekir. Hatalardan sonra dört farklı durum oluşabilir:

1. a_t eylemi başarısız olurken mevcut durum s_t'de değişiklik olmayabilir. Bu durumda planın geri kalanı ($rest_t(P) = o_{t:G}$) hala geçerlidir.

2. Başarısız olan eylem robotu geçerli olan durumdan plan içinde yer alan başka bir duruma ($s_{t\pm k}$) taşıyabilir. Planın geri kalanı ($rest_{t\pm k}(P) = o_{t\pm k:G}$) bu durumda da geçerlidir.

3. Başarısız olan eylemden sonra oluşan durum planda yer almayan bir duruma karşılık düşebilir. Artık üretilen plan geçersizdir. Tekrar plan yapılması gerekir.

4. Bir önceki durum sonucu plan üretildiğinde geçerli bir plan oluşmayabilir.

İlk durumda, robot eylemi yanlış parametre ile yürüttüğü için başarısız olabilir. Böyle bir durumda robot aynı eylemi bütün mantıklı seçenekler değerlendirilene kadar farklı şekillerde deneyebilir. Eğer yine başarısız olursa başarısız olan eylem ile aynı sonuçları üretebilecek ve planın geri kalanıyla çelişmeyecek alternatif eylem ya da eylemler kümesine yönelmelidir. Örneğin bir nesne üzerinde *kaldırma* eylemi yerine *itme* eyleminin seçilmesi gibi. Bu önlem de başarısız olursa robot, başarısızlığa neden olan eylem seçeneklerini içermeyen alternatif bir plan oluşturmaya çalışmalıdır.

4.4 Öğrenme

Robot ortamını modelleyebilmek için bir bilgi tabanına (knowledge base) sahip olmalıdır. Bilgi tabanı $KB = (F, H)$ şeklinde ifade edilir. F olgular kümesini ve H hipotezler kümesini içerir. Robot, ihtiyaç duyduğunda gözlemlerine dayanarak $KB \models H$ olacak şekilde yeni hipotezler (H) üretir ve KB bu hipotezlere göre

güncellenir. Hipotezler, robotun gelecek görevlerindeki başarımını arttırmak için kullanılabilir. Güdümlü öğrenmede (supervised learning), bilinmeyen bir fonksiyonun $(x, f(x))$ giriş ve çıkış değerleri verilir ve bu fonksiyonu modelleyen bir hipotez (h) öğrenilir. Öğrenilen bu hipotez henüz karşılaşılmamış örnekleri doğru bir şekilde tahmin edebilirse, doğru bir hipotez olarak sınıflandırılabilir.

Bu çalışmada ele alınan problem, eylem yürütme durumları (eylemin parametreleri, sonucu vs.) ile başarısızlık durumları arasında ilişki kurmak için bir öğrenme yönteminin geliştirilmesini gerektirmektedir. Bu ilişkinin kurulması hataların kotarılması veya önlenmesi için gereklidir. Robot yaşamı boyunca eylem yürütme sonuçlarını ve ortamı gözleyebildiği için artımlı ve devamlı bir öğrenme yöntemine ihtiyaç duyulmaktadır. Ayrıca robotun hedeflerine ulaşabilmesi amacıyla yapılması gereken çıkarım ve planlama işleri için bilgi tabanı, sembolik olarak ifade edileceğinden, kullanılacak olan öğrenme algoritması hipotezleri mantıksal cümleler şeklinde ifade edebilmelidir. Tümevarımsal Mantıksal Programlama (Inductive Logic Programming) yaklaşımı bu ihtiyaçların tümünü karşılamaktadır. Bu yüzden bu çalışmada bu yaklaşımdan faydalanılmıştır.

4.5 Örnek Problemler

Öğrenme problemini göstermek için motive edici bir örnek olarak, bir kol ve iki parmak tutucu ile donatılmış bir bilişsel robotu düşünelim. Robotun uzun dönemli çalışması sırasında aynı eylemi farklı bağlamlarda birkaç kez gerçekleştirmesi gerekebilir (şekil 4.1'de olduğu gibi). Bir satranç oynama senaryosu göz önüne alındığında, robot; *kaldırma*, *indirme*, *itme* ve *taşıma* eylemlerini kullanarak satranç taşları ile etkileşime geçebilir. Bu örnek senaryoda robotun, bir satranç atını kaldırmaya çalıştığını ve başarısız olduğunu farzedelim. Hatanın; taşın şekli ya da ağırlığı, görüntü işleme problemi ya da yanlış kavrama pozisyonu gibi birçok sebebi olabilir. Bu hata ile başa çıkabilmek için robotun, alternatif kavrama pozisyonları bulması gerekir. Eğer başka bir kavrama şekli başarılı olursa *kaldırma* eylemi için elde edilen yeni parametreler ve satranç atı ile ilişkilendirilerek bilgi tabanında tutulması gerekir. Eğer hiçbir parametre *kaldırma* eylemi için başarılı olamazsa, hamle yapmanın tamamlanabilmesi için alternatif bir eylemin bulunması gerekir. *İtme* eylemi *kaldırma*

eyleminin alternatifi olabilir. Bu yüzden robotun eylem seçme mekanizması, yürütülen eylemler sonucu elde edilen deneyimleri gözönüne almalıdır. Yürütme durumları ile başarısız yürütme sonuçlarının ilişkilendirilmesi ve *KB*'nin güncellenmesi için etkin bir öğrenme mekanizması gereklidir. Böylelikle hata kotarma önlemleri plana dahil edilebilir.

Blok inşası ortamında, aynı robot satranç taşları yerine bloklar ile etkileşimde bulunmaktadır. Robot bazı blokların yerlerinin görüntü işleme metodları ile tam olarak belirlenemediğini farkedebilir. Böyle bir durumda bu blokları hedeflemek yerine, konumları doğru olarak belirlenebilen bloklar ile görevlerini yerine getirebilir. Başka bir durumda ise, blokların yüksekliği arttıkça *yerleştirme* eylemi başarısız olmaya başlayabilir. Bu örnekte ise robotun blok yığını ile başarısızlık arasında bir ilişki kurması gerekir.

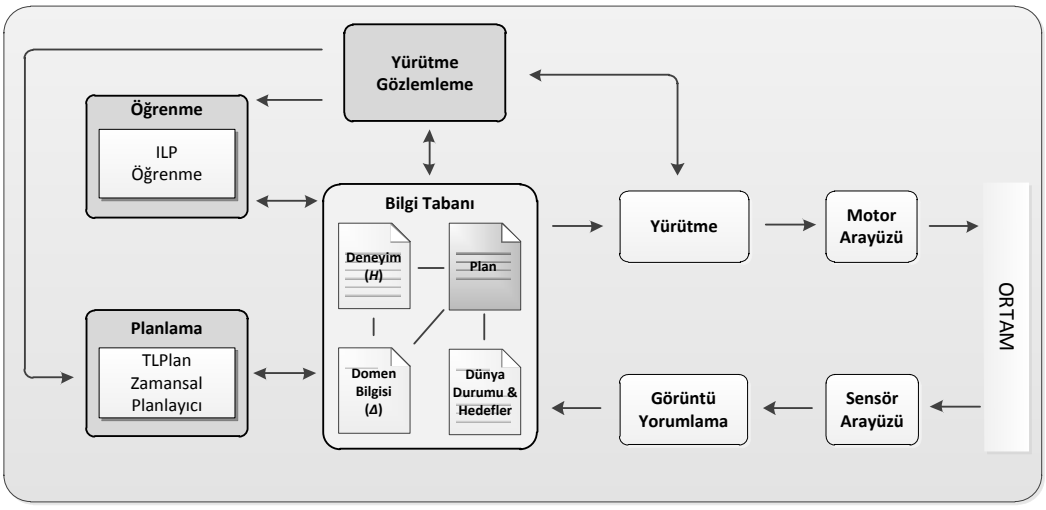
Şekil 4.1: Farklı durumlarda *kaldırma* eyleminin yürütülmesi (blok dünyası ve satranç ortamı).

5. BİLİŞSEL ROBOTLARDA PLANLAMA, YÜRÜTME VE ÖĞRENME SİSTEMİ

Bu çalışmada bilişsel robotlar için planlama, yürütme, gözlem ve öğrenme birimlerini içeren bir sistem önerilmiştir. Bu bölümde sistem bileşenleri ayrıntılarıyla incelenecek ve önerilen yöntemler tanıtılacaktır.

5.1 Önerilen Sistem

Bu çalışmada önerilen sistem beş temel süreçten oluşmaktadır: *Planlama*, *Görüntü Yorumlama*, *Yürütme*, *Yürütme Gözlemleme* ve *Öğrenme*. Bütün bu süreçler robot belleğinde tutulan bir bilgi tabanını sorgulama ve güncelleme yetkisine sahiptir. Bu bilgi tabanı, planlama tanım kümesi bilgilerini, ortamın güncel durumunu, planlama problemi için tanımlanmış hedefleri, plan ve robotun yürütme esnasında elde ettiği deneyimleri içerir. Bu süreçleri içeren sistemin temel bileşenleri ve aralarındaki bilgi akışı Şekil 5.1'de gösterilmiştir. Bu süreçler robotun duyarga ve motor arabirimlerine bağlanarak robotun bilişsel bileşenlerini oluştururlar. Şeklin basitleştirilmesi açısından haritalama ve yol planlama birimleri sistem diyagramında gösterilmemiştir.



Şekil 5.1: Bilişsel robotlar için planlama, yürütme ve öğrenme sistemi.

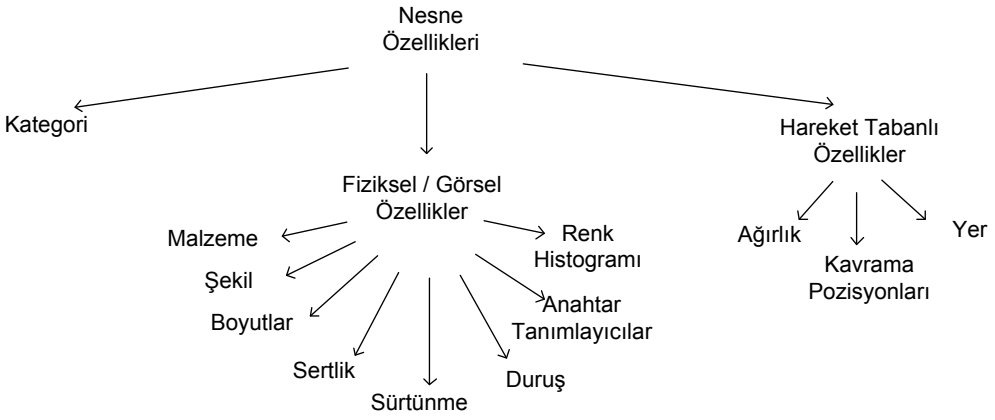
Sistemde ilk olarak, ileri zincir çıkarsamalı zamansal bir planlayıcı olan TLPlan (Bacchus ve Winter, 2001) kullanılarak bir plan üretilir. TLPlan, tekli ve çoklu robot sistemlerinde *süreli* ve *anlık* eylemler için plan üretebilmektedir. TLPlan giriş olarak tanım kümesi bilgisini (operatorler, olgular vs.), mevcut ortam durumunu, verilen hedefleri ve gürbüz görev yürütme için gereken geçerli planları üretmek amacıyla öğrenilen hipotezleri alır. Çıktı olarak ise eğer varsa geçerli bir plan üretir. Verilen durum için geçerli bir plan üretildiğinde, *Yürütme* birimi tarafından yürütülebilmesi için KB'de saklanır.

Yürütme gözlemleme sürecinde güncellenen durumların kontrol edilmesi ve olası hataların sezilebilmesi için, planın yürütülmesi esnasında bir yandan da sürekli olarak gözlenmesi sağlanır. *Görüntü Yorumlama* birimi sensörler ile ortamdan elde edilen bilgilere dayanarak planlama ve gözlenen durum bilgisini günceller. Bütün sensör verileri toplandıktan sonra ortamdaki nesneler ve ilişkileri yorumlanıp önerme haline getirilerek *KB*'de kodlanır. Sistemde gerçek sensör girişleri ile beklenen değerler karşılaştırılarak hata tespiti yapılır. Hatanın sezilmesiyle birlikte güncellenen ortam durumuna göre ya yeniden planlama yapılır ya da öğrenme süreci kullanılarak yeni hipotezler oluşturulur.

İnsanlarda hata tespitine benzer bir şekilde (Wolpert ve Ghahramani, 2000), önerilen sistemde de gerçek sensör girişleri ile beklenen değerler karşılaştırılarak hata tespiti yapılır. Örneğin robot çeşitli sebeplerle, *kaldırma* eylemini yürütürken başarısız olabilir. *Ortam Görüntü Analiz Birimi* sensörler ile ortamdan elde edilen bilgilere dayanarak tanım kümesi bilgisini ve durumu güncellemelidir. Bütün sensör verileri toplandıktan sonra ortamdaki nesneler ve ilişkileri yorumlanıp önerme haline getirilerek *KB*'ye kodlanır. Görüntüdeki nesneleri algılayıp tanımak için (Sjoo ve diğ., 2011; Cubek, 2011; Hinterstoisser ve diğ., 2012) ve ortamla ilişkilerini (Ersen ve Sariel-Talay, 2012) belirlemek için gelişmiş 3D görüntü işleme tekniklerine ihtiyaç vardır. Robot, bilinen nesnelerle etkileşime girdiğinde, nesnelerin önceden tanımlanmış modellerini (örneğin şablonlar) ve farklı görsel/fiziksel özelliklerini kullanabilir. Eğer nesnenin modeli önceden bilinmiyorsa; robotun, nesnelerin bazı gözlenebilir özelliklerini (SIFT (Lowe, 2004), VPFH (Rusu ve diğ., 2010), boyut, renk vs.) belirlediği ve saklayabildiği kabul edilmiştir.

Öğrenme sürecinde robotun eylemler, eylemlerin parametreleri ve ortamda bu eylemler tarafından etkilenen varlıklar hakkında kazandığı tecrübelere dayanarak öğrenebilmesi sağlanmaktadır. Bu birim hata sezilmesi durumunda yeniden planlama yapılmasının mümkün olmadığı durumlarda da alternatif planlar üretilebilmesi için destek verir.

Bu tez çalışmasında eylem yürütme hataları gözlemlenerek hipotezler oluşturan bir deneyimsel öğrenme yöntemi önerilmiştir. Yürütmenin gözlemlenmesi sonucu elde edilen iki tip hipotez vardır. Birinci tip hipotezler farklı bağlamlarda güvenli eylem yürütme için uygun parametrelere ilişkin olarak üretilmektedir. İkinci tip hipotezler ise tümüyle başarısız olan eylemler ile ilgilidir. Bu hipotezler gelecekteki planlama süreçlerini yönlendirmek amacıyla planlama tanım kümesini (Δ) güncellemek için kullanılır. Bu yüzden robot bir eylemi yürüttüğünde başarılı olsa da olmasa da tecrübe kazanmış olur. Hipotezler ilgili nesnelerin gözlenen özellikleri ve ilgili dünya durumu göz önüne alınarak üretilir. Hipotez üretimi için kullanılan özellikler Şekil 5.2'de yer almaktadır.



Şekil 5.2: Nesnelerin gözlenebilir özellikleri. Robot etkileşimde bulunduğu nesnelerin önceden tanımlanmış modellerini biliyorsa bu özelliklerin bazı değerleri doğrudan belirlenmiş olur, aksi takdirde, bu özellikler gözlemler sonucu elde edilebilir.

5.2 Planlama, Yürütme ve Gözlemleme

Planlama ve yürütme işlemleri için kullanılan temel süreç (Algoritma 8) hedef durumuna ulaşılıncaya ya da geçerli bir plan üretilememe durumuna kadar çalışmaya

devam eder. Geçerli bir plan bulunduğunda, plandaki bütün eylemler yürütülürken bir yandan da ortam hata durumlarına karşı gözlemlenir.

Algoritma 8 Gürbüz Planlama, Yürütme ve Gözlemleme (Δ, s_0, s_G)

Giriş: Planlama tanım kümesi Δ, başlangıç durumu s_0, hedef durumu s_G
Çıktı: *başarılı* ya da *hata*

$s_t = s_0$
while $s_t \not\subseteq s_G$ **do**
  P = Planlama(Δ, s_t, s_G)
  if $P = \varnothing$ **then**
    return *hata*
  end if
  s_t = Yürütme ve Gözlemleme(P, s_t)
end while

return *başarılı*

Yürütme ve gözlemleme süreci (Algoritma 9) sürekli bir şekilde eylemlerin başarılı olarak yürütülüp yürütülmediğini kontrol eder. Hem başarılı hem de başarısız yürütme durumlarında KB güncellenir (yeni hipotezler üretilir ya da var olanlar güncellenir). Eylem yürütülürken verilen bağlamda en başarılı eylem yürütme modeli saklanır ve saklanan model gelecekte tekrar kullanılır. Bir eylem a_t yürütüldükten sonra dört farklı durum oluşabilir:

1. a_t başarılı bir şekilde yürütülmüştür, a_{t+1} yürütülmeye başlanabilir.

2. Hata oluşmasına rağmen (belli bir zaman aralığında oluşması beklenen eylem etkilerinin oluşmaması vb.) mevcut durum değişmez. a_t eyleminin parametreleri güncellenmelidir. Bu süreçte robot başarısız olan eylemi; başarılı olmadığı, önceden belirlenmiş bütün modeller denenmediği ya da tehlikeli bir durum oluşmadığı sürece yürütme modellerini değiştirerek gerçekleştirmeye devam eder.

3. Birkaç denemeden sonra a_t eylemi için bütün yürütme modelleri denenmiş olabilir. Robot başarısız olan eylemi yürütmekten vazgeçer. Bu durumda verilen bağlam ve eylem ikilisi için yeni bir hipotez üretilir. Bu arada planlama tanım kümesi güncellenerek alternatif eylem seçimi için yönlendirme yapılır. Daha sonra yeni bir plan üretmek için planlayıcı çağırılır.

Algoritma 9 Yürütme ve Gözlemleme(P, s_t)

Giriş: Mevcut durum s_t, plan P
Çıktı: $s_{t\pm k}$ ya da beklenmeyen bir durum

$P = P_{t:G}$
while $P! = \varnothing$ **do**
  $eylem$ = POP(P)
  $parametreler$ = KB.YurutmeTipiAlma($eylem$,$nesne$)
  while $true$ **do**
    $yurutmeGozlem$ = Yürütme($eylem[parametreler], s_t$)
    s' = GörüntüYorumlama(t)
    TemsilGüncelleme($eylem$,$nesne$,$parametreler$,$yurutmeGozlem$)
    if $yurutmeGozlem$ = $başarılı$ **then**
      $s_t \leftarrow s'$
      break
    else
      if $s' = s_t$ **then**
        $parametreler$ = KB.YeniYurutmeTipi($eylem$,$nesne$)
      else
        return s'
      end if
    end if
  end while
end while
return s'

---

functionTemsilGüncelleme($eylem$,$nesne$,$parametreler$,$yurutmeGozlem$)

KB.Ekle([$eylem$, $nesne$, $parametreler$, GörüntüYorumlama(t), $yurutmeGozlem$])
if $yurutmeGozlem$ = $başarılı$ **then**
  h = ILP-Öğrenme($eylem$, KB)
  KB.YurutmeTipiAtama([h, $eylem$])
else
  h = ILP-Öğrenme($eylem$, KB)
end if

KB.Güncelleme([h,$eylem$])
KB.Güncelleme(Δ)

---

4. Robot istenmeyen bir durumda olduğunu keşfedebilir. Gerekli tanım güncellemeleri ile yeni bir plan üretmek için planlayıcı çağırılır.

Ulaşılan bu sonuçlar *planlama problemi*'nin bir parçası olarak *KB*'ye de eklenir. Ayrıca ortamın durumu da sürekli olarak *KB*'de saklanır. Güncellenen bütün eylemler

ve bu eylemler ile ilgili bilgiler saklanarak öğrenme biriminin yeni bir hipotez üretmesi ya da mevcut hipotezleri güncellemesi sağlanır.

5.3 Yaşam Boyu Deneyimsel Öğrenme

Önerilen öğrenme yöntemi, *Tümevarımsal Mantıksal Öğrenme* (ILP) süreçlerini içermektedir (Quinlan, 1990). ILP'de asıl amaç gözlemler ile uyumlu bir hipotezler kümesi üretmektir. Üretilen hipotezler mantıksal cümleler kümesi ile ifade edilir. ILP, yeni gözlemler ortaya çıktıkça hipotez üretmeyi, hipotezi güncellemeyi ya da hipotezleri iptal etmeyi sağlar. Yürütmenin gözlemlenmesi sonucu elde edilen iki tip hipotez vardır. Birinci tip hipotez farklı bağlamlarda güvenli eylem yürütme için uygun parametrelere ilişkin olarak üretilen hipotezlere karşılık düşer. İkinci tip hipotez ise yürütme hataları ile eylemlerin içerikleri arasındaki ilişkiyi ifade eder. İki durumda da bilinen ya da gözlenen nesnenin özellikleri, ilişkileri ve ortamın gözlenebilen özellikleri göz önünde bulundurularak ilgili olan özellikler hipoteze dahil edilir. Bu hipotezler gelecekteki plan süreçlerini yönlendirmek amacıyla planlama tanım kümesini (Δ) güncellemek için kullanılır. Bu yüzden robot bir eylemi yürüttüğünde başarılı olsa da olmasa da tecrübe kazanmış olur.

Bu çalışmada iki farklı ILP uygulaması (*FOIL* ve *PROGOL*) kullanılmış ve her iki yöntemin başarımları karşılaştırılmıştır.

5.3.1 FOIL algoritması

FOIL algoritması yukarıdan aşağıya yaklaşım yöntemi kullanılarak oluşturulmuştur ve iki parçadan oluşur (Algoritma 10 ve 11). Algoritma 10, *KB* ve eylem tipini giriş olarak alır. Gözlem kümesi *KB*'den elde edilir ve hipotez üretmek için kullanılır. Bu algoritmada iki boyutlu *A* nitelik dizisi üretilir. Dizinin sütunları gözlemde yer alan nesnelerin niteliklerine karşılık düşer. Her bir nitelik için gözlemlere göre belli bir kazanç (gain) değeri atanır ve bu kazanç nitelik dizisinin ilk satırını oluşturur. Dizinin ikinci satırı, niteliğin mevcut hipotezde kullanılıp kullanılmayacağını gösterir. Pozitif ve negatif örnekler, elde edilen gözlemlere ve eylem tipine göre oluşturulur. Algoritmanın çıktısı ise bir hipotez uzayıdır (*H*). Hipotez, ILPGeneration (Algoritma 11) alt yordamı çağırılarak oluşturulur. Algoritma 10; gözlem kümesindeki bütün

pozitif örnekler, üretilen hipotez uzayı tarafından kapsanana kadar çalışmaya devam eder. Örneğin, *kaldırma* eylemi ile ilgili hipotezler üretilmek isteniyorsa elde edilen hipotezler, başarısız olan bütün *kaldırma* eylemi gözlemlerini açıklamalıdır.

Algoritma 11 özyineli (recursive) şekilde çalışır. Her iterasyonda henüz hipoteze eklenmemiş nitelikler için kazanç değerleri hesaplanır. En yüksek kazanca sahip nitelikler hipoteze eklenir. Eklenen nitelikler sonucu kapsanan pozitif ve negatif gözlemler gözlem kümesinden silinir. Ayrıca nitelik dizisi güncellenerek hangi niteliklerin hipoteze dahil edildiği işaretlenir. Bu çalışmada FOIL algoritması, niteliklerin ağırlıkları göz önüne alınacak şekilde iyileştirilmiştir. Yüksek öncelikli bir nitelik bir hipotezde yer almıyor ancak hipoteze dahil bir nitelik ile birlikte tüm olumlu örneklerde yer alıyorsa, bu özellik de hipoteze eklenir. Örneğin satranç senaryosunda taşın kategorisi diğer niteliklere göre daha yüksek önceliğe sahiptir. Bu yüzden eğer mümkünse hipoteze eklenmelidir. Algoritma 11 aşağıda yer alan koşullardan herhangi biri sağlanana kadar çalışmaya devam eder:

1. Hipoteze yeni bir nitelik eklemenin olumlu bir yönde etkisi yoksa (MIN_BEST_GAIN).

2. Negatif örnek dizisi boşsa ($N' = \varnothing$).

Algoritma 10 ILP-Learning(*eylem*, *KB*)

Giriş: eylem, O gözlem kümesini elde etmek için KB
Çıktı: hipotez uzayı, H
Global değişkenler: 2-D nitelik dizileri A ve A', pozitif örnek dizileri P ve P', negatif örnek dizileri N ve N', öncül liste $antecedent$

O=KB.getObservationHistory()
$A = \varnothing$
$H = \varnothing$
for each o in O **do**
  $eylem$ e göre P ve N dizilerini oluştur
  while P != $\varnothing$ **do**
    $N' \leftarrow N$, $P' \leftarrow P, A' \leftarrow A$
    ILPGeneration$(antecedent, o, H)$
  end while
end for
$return\ H$

Algoritma 11 ILPGeneration$(antecedent, o, H)$

Giriş: $antecedent$, gözlem o, şu ana kadar üretilen hipotez uzayı H
Global değişkenler: nitelik dizisi A_h

for each $a \notin antecedent$ & $a \in A$ **do**
  kazanç değerlerini hesapla ve A dizisine kaydet
end for
$i = A$ dizisinde en iyi kazanca sahip niteliğin indeksi
if $A[i][0] \leq$ MIN_BEST_GAIN **then**
  $H = H \cup \{antecedent \rightarrow consequent(o)\}$
  P & N dizilerini düzenle
  $return$
end if
for each en iyi kazançtan daha düşük kazanca sahip $a_j \in A_h$ **do**
  if a_j, a_i tarafından kapsanan bütün pozitif örnekleri kapsıyor **then**
    $antecedent = antecedent \cup a_j$
    $A'[j][1] = 1$
    P' & N' dizilerini düzenle
  end if
end for
for each en iyi kazanca sahip $a_j \in A$ **do**
  $antecedent = antecedent \cup a_j$
  $A'[j][1] = 1$
  P' & N' dizilerini düzenle
end for
if $(N' = \varnothing)$ **then**
  $H = H \cup \{antecedent \rightarrow consequent(o)\}$
  $antecedent$ içeren kayıtları P den çıkar
  $return$
else
  ILPGeneration$(antecedent, o, H)$
end if

5.3.2 PROGOL algoritması

PROGOL algoritması tersten tümdengelim yöntemine dayanmaktadır (Muggleton, 1995). Genel algoritma (Algoritma 12) 4 adımdan oluşmaktadır:

1. *Örnek seçimi*: Genelleştirme yapmak için örnek seçimi için kullanılır. Eğer örnek yoksa algoritma sona erer, varsa sonraki adıma geçilir.

2. *En özgül cümlenin üretilmesi*: Seçilen örneği sağlayan ve dil sınırlamalarına uyan en özgül cümle seçilir. Seçilen cümle bir deyimler kümesidir.

3. *Arama*: Seçilen özgül cümleden daha genel bir cümle elde edilir. Bu cümle, en özgül cümlenin alt kümesi olan (en özgül cümlenin deyimlerinin bir alt kümesini içerir), en iyi skora sahip cümlenin aranması ile oluşturulur.

4. *Gereksiz örneklerin silinmesi*: En iyi skora sahip olan cümle güncel hipoteze eklenir. Bu cümle tarafından kapsanan örnekler, örnek kümesinden silinir. İlk adım olan örnek seçimine geri dönülür.

Algoritma 12 ILP-Learning(*eylem*, *KB*)

Giriş: eylem, O gözlem kümesini elde etmek için KB
Çıktı: hipotez uzayı, H

$H = \varnothing$
O=KB.getObservationHistory()
for each o in O **do**
  C = GetMostSpecificClause()
  G = SearchForGeneralClause(C)
  $H = H + G$
  O' = GetCoveredExamples(G,O)
  $O = O$ - O'
end for
return H

5.3.3 Hata durumları için üretilen hipotezler

ILP, robotların çalışma süreleri boyunca elde ettiği yürütme sonuçlarını kullanarak hipotezler üretmelerine yardımcı olur. Bu bölümde hata durumları için hipotezlerin nasıl üretildiği anlatılmıştır. Robotun, aşağıda yer alan örnek gözlem kümesini gözlediğini varsayalım:

Kategori(satranc_At) $\wedge$ *Boy(küçük)* $\wedge$ *Renk(siyah)* $\wedge$ *Malzeme(ahşap)* $\wedge$ *BaşarısızKaldırma*

Kategori(satranc_At) $\wedge$ *Boy(küçük)* $\wedge$ *Renk(beyaz)* $\wedge$ *Malzeme(ahşap)* $\wedge$ *BaşarısızKaldırma*

Kategori(satranc_At) $\wedge$ *Boy(büyük)* $\wedge$ *Renk(siyah)* $\wedge$ *BaşarılıKaldırma*

Kategori(satranc_At) $\wedge$ *Boy(büyük)* $\wedge$ *Renk(beyaz)* $\wedge$ *BaşarılıKaldırma*

Her gözlem *kaldırma* eyleminin bir kez yürütülmesine karşılık düşer. Eylem ile ilgili olgular (bu örnekte nesnenin özellikleri) ve gözlenen eylemin sonucu (başarılı ya da başarısız olması) hesaba katılır. ILP yöntemi yukarıda verilen gözlem kümesine uygulandığında şu hipotezler üretilir:

Kategori(satranc_At) $\wedge$ *Boy(küçük)* $\wedge$ *Malzeme(ahşap)* $\Rightarrow$ *BaşarısızKaldırma*

Kategori(satranc_At) $\wedge$ *Boy(büyük)* $\Rightarrow$ *BaşarılıKaldırma*

ILP'nin en güçlü yanlarından biri de, hipotezlerin yüklem mantığı gösterim dili ile temsil edilmeleri sonucu, kolayca doğal dile çevirilebilmeleridir. Yukarıda elde edilen hipotezler şu şekilde yorumlanabilir: eğer robot *kaldırma* eylemini küçük ve ahşaptan yapılmış satranç atları üzerinde gerçekleştiriyorsa yürütme başarısız olacaktır. Uzun satranç atları söz konusu ise *kaldırma* eylemi başarılı olacaktır.

Bu hipotezler oluşturulduktan sonra robotun şu gözlemi yaptığını varsayalım:

Kategori(satranc_At) $\wedge$ *Boy(küçük)* $\wedge$ *Malzeme(plastik)* $\wedge$ *Renk(siyah)* $\wedge$ *BaşarısızKaldırma*

Bu gözlemden sonra *KB*'de yer alan ilk hipotez şu şekilde genelleştirilecektir:

Kategori(satranc_At) $\wedge$ *Boy(küçük)* $\Rightarrow$ *BaşarısızKaldırma*

5.3.4 Eylem yürütme modelleri için üretilen hipotezler

Robot bir eylemi yürütürken hata ile karşılaştıktan sonra mevcut durum değişmemişse, önce aynı eylemin yürütme modelini değiştirmeyi düşünecektir. *KB*, her eylem-nesne çifti için önceden belirlenmiş yürütme modelleri kümesini içermektedir. Satranç atını kaldırmak için *KB*'de α, β ve γ olacak şekilde farklı nesne kavrama modelleri olduğunu varsayalım. Robot ilk kaldırma denemesinde başarısız olursa yeni yürütme modeli *KB*'den alınacaktır. Böylece robot başlangıç kavrama modelini $KavramaTipi(\alpha)$ dan $KavramaTipi(\beta)$ ya çevirecektir. *Kaldırma* eylemi bu sefer

$KavramaTipi(\beta)$ ile denenecektir. Eğer bu deneme de başarısız olursa, kavrama poziyonu bu kez $KavramaTipi(\gamma)$ olarak belirlenecektir. Üçüncü denemenin sonunda son deneme başarılı ise, *KB*'de şu bilgiler yer alacaktır:

$$Kategori(satranc\_At) \wedge Boy(küçük) \wedge Renk(siyah) \wedge KavramaTipi(\alpha) \wedge BaşarısızKaldırma$$

$$Kategori(satranc\_At) \wedge Boy(küçük) \wedge Renk(siyah) \wedge KavramaTipi(\beta) \wedge BaşarısızKaldırma$$

$$Kategori(satranc\_At) \wedge Boy(küçük) \wedge Renk(siyah) \wedge KavramaTipi(\gamma) \wedge BaşarılıKaldırma$$

Bütün bu gözlemler düşünüldüğünde satranç atı üzerindeki *kaldırma* eylemi için en iyi yürütme modeli şu hipotez ile kayıt edilir:

$$Kategori(satranc\_At) \wedge Boy(küçük) \wedge Renk(siyah) \Rightarrow KavramaTipi(\gamma)$$

Robot bundan sonra siyah ve küçük satranç atları için *kaldırma* eylemini yürütürken önce $KavramaTipi(\gamma)$ modelini deneyecektir.

Bir başka örnek olarak, robotun beyaz bir satranç atını kaldırmak istediğini varsayalım. Robot önce $KavramaTipi(\alpha)$'yı, eğer başarısız olursa $KavramaTipi(\beta)$'yı deneyecektir. Burada dikkat edilmesi gereken nokta, verilen bu özellikleri içeren bir hipotezin daha önce üretilmemiş olmasıdır. İki deneme de başarısız olursa $KavramaTipi(\gamma)$ denenecektir. Eğer aşağıdaki durum gözlenirse:

$$Kategori(satranc\_At) \wedge Boy(küçük) \wedge Renk(beyaz) \wedge KavramaTipi(\gamma) \wedge BaşarılıKaldırma$$

daha önceki hipotez şu şekilde güncellenecektir:

$$Kategori(satranc\_At) \wedge Boy(küçük) \Rightarrow KavramaTipi(\gamma)$$

Verilen hipotez şöyle yorumlanır: gelecekte satranç atları üzerinde *kaldırma* eylemleri yürütülürken; robot, küçük olanlar için renkten bağımsız olarak $KavramaTipi(\gamma)$ yürütme modelinin başarılı olacağına inanacaktır.

Eğer daha önceden tanımlı yürütme modellerinden hiçbiri başarılı olmazsa, satranç atları üzerindeki *kaldırma* eylemleri için farklı kavrama tiplerinin denenmesi durdurulacaktır. Bir önceki bölümde açıklandığı üzere bu hata durumu için bağlam-hata çiftine karşılık yeni bir hipotez oluşturacak ve bu hipotezler sonraki planlama süreçlerini yönlendirmek için kullanılacaktır.

5.3.5 Bilinmeyen nesneler için üretilen hipotezler

Robot bilinmeyen bir nesne ile etkileşimde olduğu zaman, nesnenin önceden belirlenmiş modelini elde etmek mümkün değildir. Bu durumdaki strateji, yeni hipotezler üretmek için nesnenin temel görsel özelliklerini (SIFT, VPFH vs.) kullanmaktır. Nesnenin gözlenebilen bütün özellikleri hipotezde yer alabilir. Örnek bir hipotez aşağıda verilmiştir:

$$Kategori(belirsiz) \wedge SIFT(\alpha) \wedge Boy(küçük) \Rightarrow BaşarısızKaldırma$$

Eğer bilinmeyen başka bir nesnenin görsel özellikleri hipotez ile eşleşirse, hipotez bilinmeyen yeni nesne için kullanılabilir. Aynı yöntem eylem yürütme modelleri için üretilen hipotezlerde de kullanılabilir.

5.4 Hipotezlerden Planlamaya Geçiş

Yaşam boyu öğrenme süreci sürekli olarak yeni hipotezler üretir. Bu hipotezler robotun gelecekteki görevlerindeki başarımını iyileştirmek için kullanılır. Hipotezler, planlamayı çeşitli alternatif şekillerde yönlendirebilirler:

- başarısız olan eylemin planlama tanım kümesindeki ön koşullarını değiştirmek,
- başarısız olan eylemin planlama tanım kümesindeki etkilerini değiştirmek,
- arama ağacını budamak için yeni kontrol kuralları üretmek,
- başarısız olan eylemin maliyetini değiştirmek.

İlk yaklaşımda; başarısız olan eylemin, hipotezler tarafından tanımlanan belirli bağlamlarda seçiminin önlenmesi sağlanır (Usug ve Sariel-Talay, 2011). İkinci

yaklaşımda ise hipotezler ile başarısız olan eylemlerin etkileri ilişkilendirilir ancak bu durum temsilleri karmaşıklaştırabilir. Üçüncü yaklaşım arama ağacını budayarak eylemin seçimini engeller. Bu tez çalışmasında, eğer alternatif bir yol yoksa başarısız olan eylemin seçilmesini garanti etmek için dinamik maliyet hesaplama yöntemi önerilmektedir. Eğer mevcut durum hipotezler tarafından ifade edilen durumlardan birine karşılık geliyorsa eylemin maliyeti arttırılır. Böylelikle hipotezlerin planlayıcıyı yönledirmesi sağlanmış olur. Robot bu yaklaşım ile gelecek görevlerinde oluşabilecek hata durumlarından kaçabilir. Hatanın nedenini ayırt etmek mümkün olmasa bile hipotezler yürütme görevlerine bilgi birikimi olarak hizmet edebilirler. Böylelikle, robot sürekli başarısız olan *kaldırma* eylemini seçmek yerine alternatif *itme* eylemini seçebilir. Bu yaklaşımda planlayıcı, eğer hedefi gerçekleştirmek için alternatif bir yol yoksa başarısız olan eylemi seçmeye devam edebilir.

6. DENEYLER

Tez çalışmasında önerilen sistemin ve öğrenme sürecinin başarım oranını ölçmek amacıyla üç farklı deney ortamı tasarlanmıştır. İlk deney ortamında simülasyon üzerinde öğrenme algoritmasının başarımı nitelik tabanlı öğrenme algoritmaları *ID3* ve *Bayes Ağları (BA)* ile karşılaştırmalı olarak incelenmiştir. İkinci deney ortamı ise gerçek robot ve nesneler kullanılarak oluşturulmuştur. Daha sonra *ILP*'nin en önemli avantajlarından biri olan bilgi birikimi kullanımı analiz edilmiştir.

6.1 Benzetim Ortamı Deneyleri

Birinci deney setinde, önceden belirlenmiş hipotezlere göre *kaldırma* eylemi için rastgele olarak üretilmiş gözlem setleri kullanılarak başarım ölçülmüştür. 6 farklı hipotez belirlenip, her hipotez için her biri 30 farklı gözlemden oluşan eğitim seti ve 10 farklı gözlemden oluşan test seti oluşturulmuştur. Her bir eğitim seti için *ILP (FOIL ve PROGOL)*, *ID3* ve *Bayes Ağları* algoritmalarının başarılı ve başarısız eylemler için hipotezler üretmeleri sağlanıp, test setleri üzerinde bu hipotezlerin başarımı ölçülmüştür. Bunun için 3 farklı ölçüm kullanılmıştır: kesinlik (precision), anma (recall) ve F-ölçütü. 3 algoritmanın sonuçları Çizelge 6.1'de yer almaktadır.

Çizelgeden görülebileceği üzere *ID3*, *Bayes Ağları* ve *FOIL* birbirlerine yakın ve *PROGOL*'den daha düşük F-Ölçütü değerlerine sahiptirler. *Bayes Ağları*, test seti örneklerini genellikle *başarısız* olarak sınıflandırdığından *başarısız* sınıfı için çok yüksek doğru artı oranına ve *başarılı* sınıfı için de çok düşük doğru artı oranına sahiptir.

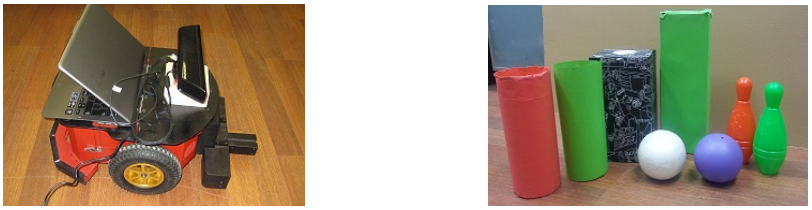
Çizelge 6.1: ID3, Bayes Ağları ve ILP sonuçları

Algoritma	Kesinlik	Anma	F-Ölçütü
ID3	0,648	0,643	0,645
BA	0,634	0,753	0,651
FOIL	0,652	0,672	0,662
PROGOL	0,803	0,756	0,769

PROGOL'de ise iki sınıf için doğru artı oranları ortalama olarak *ID3*, *Bayes Ağları* ve *FOIL*'e göre daha iyi çıkmıştır. Ortalama sonuçlara bakılırsa, *PROGOL* algoritmasının daha iyi sonuçlar verdiği görülmüştür.

6.2 Robot Deneyleri

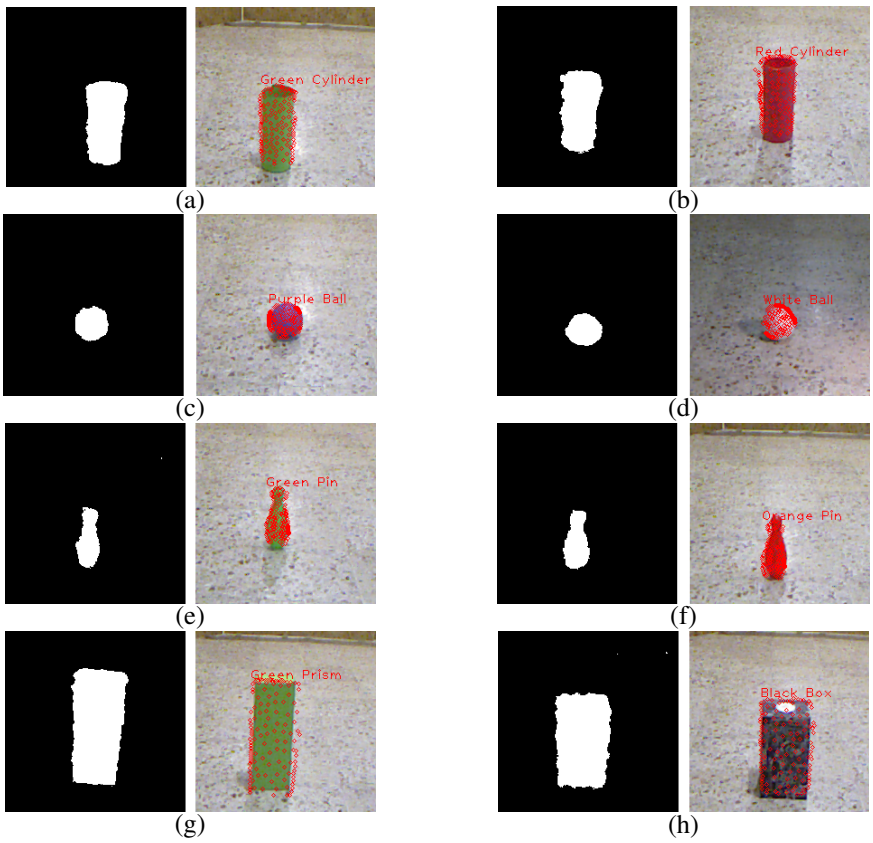
Gerçek robot deneyleri standart bir tutucu, sonar sensör halkası ve üzerinde Kinect sensör bulunan Pioneer 3DX robot ile gerçekleştirilmiştir. Deneyde kullanılan robot ve nesneler Şekil 6.1'de gösterilmiştir. Seçilen nesneler 4 farklı kategoriden oluşmaktadır: *dikdörtgen prizma*, *silindirik kutu*, *plastik lobut* ve *top*. Ortamdaki nesnelerin tanınması için çok-kipli şablon eşleştirmesine dayanan LINE-MOD (Hinterstoisser ve diğ., 2012) yaklaşımı kullanılmıştır. LINE-MOD yönteminde, nesnelerin şekillerinden ayırt edilebilmesi için yüzey normallerinden, farklı renklerin ayırt edilebilmesi için ise renk gradyanlarından yararlanılmaktadır. Kullanılan nesneler için LINE-MOD'un ürettiği çıktı Şekil 6.2'de yer almaktadır. Şekilden de görülebileceği üzere bütün nesneler ortamda belli bir düzlemde bulundukları sürece tanınabilmektedir.



Şekil 6.1: (a) Pioneer 3DX robot, Robee. (b) Gerçek robot deneylerinde kullanılan nesneler.

Birinci deneyde *kaldırma* eyleminin bütün nesneler üzerindeki başarısı test edilmiştir. Robotun nesneye olan uzaklığı, nesnenin ağırlık merkezi referans alınarak hesaplanmıştır. Kinect yaklaşık olarak 60 cm.'den daha kısa mesafelerde uzaklık ölçümü için çok güvenilir değildir. Bu durumda sonar sensörler kullanılarak daha doğru sonuçlar elde edilebilir. Bu yüzden hataların tespiti için, nesneye olan uzaklık belli bir değerin üstünde olduğunda Kinect, altında olduğunda ise sonar sensörler kullanılmıştır.

Çizelge 6.2 birinci deneyin sonuçlarını göstermektedir. Her bir nesne için beş farklı *kaldırma* eyleminin başarım oranı çizelgede verilmiştir. Robot, kutu ve lobutlardan



Şekil 6.2: Deneylerde kullanılan nesnelerin şablonları ve tanınabilen noktaları. (a) yeşil silindirik kutu, (b) kırmızı silindirik kutu, (c) mor plastik bowling topu, (d) beyaz köpükten top, (e) yeşil plastik lobut, (f) turuncu plastik lobut, (g) yeşil prizmatik kutu, (h) siyah prizmatik kutu.

birer tanesi için eylemi yürütmede başarısız olmaktadır. Kutuyu almaya çalışırken oluşan hata, nesnenin fiziksel deformasyonundan kaynaklanmaktadır. Deformasyon yüzünden sonar sensörler nesnenin, robotun önünde olup olmadığına karar verirken hata oluşturmaktadır. Ancak bu sebep robot tarafından nesnenin bir niteliği olarak gözlenememektedir. Bunun yerine hipotez oluşturulurken gözlenebilen fiziksel nitelikler göz önüne alınmıştır. Lobutu almaya çalışırken oluşan hata ise robotun görüntü işlemesi ile ilgili bir problemden kaynaklanmaktadır. Robot bazen turuncu lobut ile yeşil lobutun şablonlarını birbirine karıştırmaktadır (Şekil 6.3), bu da uzaklık ölçülürken hataya yol açmaktadır. Bu yüzden nesneye yaklaşırken çoğunlukla hata oluşmaktadır. Lobutun üst kısmı dar olduğu için nesnenin konumunu sonar sensörlerle de net olarak algılamak mümkün olamamaktadır.

Farklı nesneler için yürütülen *kaldırma* eylemlerinde başarısız durumlar için hipotez üretmek amacıyla bütün gözlemler ILP algoritmasına giriş olarak verilir. PROGOL ile

Çizelge 6.2: Belirlenen nesnelere göre eylem başarım oranı

Kategori	Şekil	Renk	Malzeme	Boyut	Başarı (%)
kutu	*prizma*	*yeşil*	*kağıt*	*küçük*	60
kutu	*prizma*	*siyah*	*kağıt*	*büyük*	60
kutu	*silindir*	*yeşil*	*plastik*	*büyük*	60
kutu	*silindir*	*kırmızı*	*plastik*	*büyük*	40
lobut		*turuncu*	*plastik*	*küçük*	20
lobut		*yeşil*	*plastik*	*küçük*	60
top	*küre*	*mor*	*plastik*	*küçük*	100
top	*küre*	*beyaz*	*köpük*	*küçük*	60

Şekil 6.3: Bazı durumlarda yeşil ve turuncu lobut birbirine karıştırılmaktadır.

orijinal ve geliştirilmiş FOIL algoritmalarının sonuçları sırasıyla Çizelge 6.3, 6.4 ve 6.5'de verilmiştir.

Sonuçlardan görüldüğü üzere, robot her yeni gözlem sonucunda hipotezini gözden geçirir. PROGOL ve orijinal FOIL uygulamasında daha genel hipotezler elde edilirken, geliştirilmiş FOIL bizim senaryomuz için daha anlamlı hipotezler üretmektedir.

FOIL'in nihai hipotez uzayı (Çizelge 6.5, son satırı) *kaldırma* eyleminin aşağıdaki durumlar için eylem maliyetinin arttırılmasını önermektedir:

Kategori(kutu) $\wedge$ *Renk(kırmızı)*

Kategori(lobut) $\wedge$ *Renk(turuncu)*

Üretilen hipotezler, robot için bir öncelik modeli oluşturulmasını sağlar. Hipotezlere göre *kaldırma* eyleminin maliyeti değiştirilerek, robotun alternatif bir eylem (örneğin *itme*) ile nesneleri taşıması ya da tutabileceğine inandığı alternatif nesneleri seçmesi sağlanabilir.

Çizelge 6.3: PROGOL uygulaması ile üretilen hipotezler. Herbir satır gözlem geçmişi ile ilişkilendirilen hipotezi göstermektedir.

Gözlem Geçmişi	Hipotez uzayı
o_1	$\Rightarrow$ *BaşarılıKaldırma*
$o_{1:2}$	$\Rightarrow$ *BaşarılıKaldırma*
$o_{1:3}$	$\Rightarrow$ *BaşarılıKaldırma*
$o_{1:4}$	*Renk(siyah)* $\Rightarrow$ *BaşarılıKaldırma*
	Renk(yeşil) $\Rightarrow$ *BaşarılıKaldırma*
	Renk(kırmızı) $\Rightarrow$ *BaşarısızKaldırma*
$o_{1:5}$	*Renk(siyah)* $\Rightarrow$ *BaşarılıKaldırma*
	Renk(yeşil) $\Rightarrow$ *BaşarılıKaldırma*
	Kategori(lobut) $\Rightarrow$ *BaşarısızKaldırma*
	Renk(kırmızı) $\Rightarrow$ *BaşarısızKaldırma*
$o_{1:6}$	*Renk(siyah)* $\Rightarrow$ *BaşarılıKaldırma*
	Renk(yeşil) $\Rightarrow$ *BaşarılıKaldırma*
	Renk(kırmızı) $\Rightarrow$ *BaşarısızKaldırma*
	Renk(turuncu) $\Rightarrow$ *BaşarısızKaldırma*
$o_{1:7}$	*Renk(siyah)* $\Rightarrow$ *BaşarılıKaldırma*
	Renk(yesil) $\Rightarrow$ *BaşarılıKaldırma*
	Kategori(top) $\Rightarrow$ *BaşarılıKaldırma*
	Renk(kırmızı) $\Rightarrow$ *BaşarısızKaldırma*
	Renk(turuncu) $\Rightarrow$ *BaşarısızKaldırma*
$o_{1:8}$	*Renk(siyah)* $\Rightarrow$ *BaşarılıKaldırma*
	Renk(yeşil) $\Rightarrow$ *BaşarılıKaldırma*
	Kategori(top) $\Rightarrow$ *BaşarılıKaldırma*
	Renk(kırmızı) $\Rightarrow$ *BaşarısızKaldırma*
	Renk(turuncu) $\Rightarrow$ *BaşarısızKaldırma*

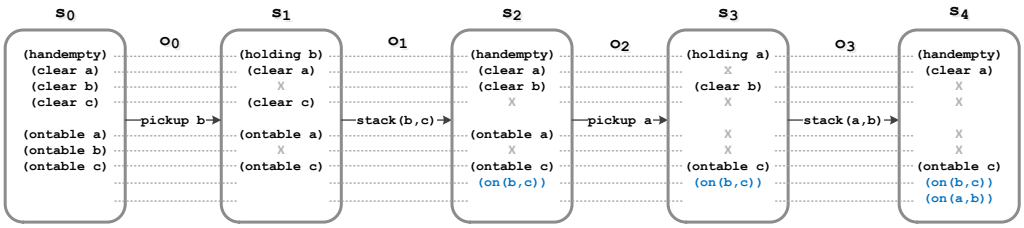
6.3 Bilgi Birikimi Kullanımı

ILP'nin, geleneksel nitelik tabanlı sınıflandırıcılar karşısındaki temel üstünlüğü, bilgi tabanlı bir temsil kullanmasıdır. Hipotezler yüklem mantığı gösterim dili ile ifade edilebilir. Ayrıca hipotezlerin genelleştirilmesi için robotun önceki bilgi birikimi de kullanılabilmektedir. Bu durumu örneklendirmek için blok inşası planlama problemini ele alalım. Robot bu ortamda *pick up*(*kaldırma*) ve *stack*(*yerleştirme*) eylemlerini yürütebilmektedir. Şekil 6.4, 3 bloktan oluşan bir ortamda yürütülen eylemler ile sonuçlarını her bir ara durum için göstermektedir.

Bu örnekte değişik durumlarda *yerleştirme* eyleminin başarısız olması durumunda üretilen hipotezler incelenmiştir. Hata farkedildikten sonra, eylem ile ilgili bütün olgular gözleme eklenir. Buradaki ekleme işlemi Algoritma 9'da yer alan güncelleme

Çizelge 6.4: FOIL algoritmasının orijinal versiyonu ile üretilen hipotezler. Herbir satır gözlem geçmişi ile ilişkilendirilen hipotezi göstermektedir.

Gözlem Geçmişi	Hipotez uzayı
o_1	*Kategori(kutu)* $\Rightarrow$ *BaşarılıKaldırma*
$o_{1:2}$	*Kategori(kutu)* $\Rightarrow$ *BaşarılıKaldırma*
$o_{1:3}$	*Kategori(kutu)* $\Rightarrow$ *BaşarılıKaldırma*
$o_{1:4}$	*Şekil(prizma)* $\Rightarrow$ *BaşarılıKaldırma*
	Renk(yeşil) $\Rightarrow$ *BaşarılıKaldırma*
	Renk(kırmızı) $\Rightarrow$ *BaşarısızKaldırma*
$o_{1:5}$	*Şekil(prizma)* $\Rightarrow$ *BaşarılıKaldırma*
	Renk(yeşil) $\Rightarrow$ *BaşarılıKaldırma*
	Kategori(lobut) $\wedge$ *Malzeme(plastik)* $\Rightarrow$ *BaşarısızKaldırma*
	Renk(kırmızı) $\Rightarrow$ *BaşarısızKaldırma*
$o_{1:6}$	*Renk(yeşil)* $\Rightarrow$ *BaşarılıKaldırma*
	Şekil(prizma) $\Rightarrow$ *BaşarılıKaldırma*
	Renk(kırmızı) $\Rightarrow$ *BaşarısızKaldırma*
	Renk(turuncu) $\Rightarrow$ *BaşarısızKaldırma*
$o_{1:7}$	*Renk(yeşil)* $\Rightarrow$ *BaşarılıKaldırma*
	Şekil(prizma) $\Rightarrow$ *BaşarılıKaldırma*
	Kategori(top) $\Rightarrow$ *BaşarılıKaldırma*
	Renk(kırmızı) $\Rightarrow$ *BaşarısızKaldırma*
	Renk(turuncu) $\Rightarrow$ *BaşarısızKaldırma*
$o_{1:8}$	*Renk(yeşil)* $\Rightarrow$ *BaşarılıKaldırma*
	Kategori(top) $\Rightarrow$ *BaşarılıKaldırma*
	Şekil(prizma) $\Rightarrow$ *BaşarılıKaldırma*
	Renk(kırmızı) $\Rightarrow$ *BaşarısızKaldırma*
	Renk(turuncu) $\Rightarrow$ *BaşarısızKaldırma*



Şekil 6.4: 3 bloktan oluşan bir ortamda plan ve oluşan durumlar.

fonksiyonundaki $KB.GetRelevantFacts()$ işlemine karşılık düşmektedir. İlgili olguları almak için nesnelerin uzamsal yakınlığı sezgisel yaklaşımı kullanılmıştır. Böylelikle *StackFailure* (*BaşarısızYerleştirme*) durumunda, öğrenme aşamasında sadece robotun elinde bulunan nesne ile yığında yer alan nesneler değerlendirilmiştir.

Çizelge 6.5: FOIL algoritmasının geliştirilmiş versiyonu ile üretilen hipotezler. Herbir satır gözlem geçmişi ile ilişkilendirilen hipotezi göstermektedir.

Gözlem Geçmişi	Hipotez uzayı
o_1	*Kategori(kutu) ∧ Şekil(prizma) ∧ Renk(yeşil) ∧ Malzeme(kağıt) ∧ Boyut(küçük) ⇒ BaşarılıKaldırma*
$o_{1:2}$	*Kategori(kutu) ∧ Şekil(prizma) ∧ Malzeme(kağıt) ⇒ BaşarılıKaldırma*
$o_{1:3}$	*Kategori(kutu) ⇒ BaşarılıKaldırma*
$o_{1:4}$	*Kategori(kutu) ∧ Şekil(prizma) ∧ Malzeme(kağıt) ⇒ BaşarılıKaldırma* *Kategori(kutu) ∧ Renk(yeşil) ⇒ BaşarılıKaldırma* *Kategori(kutu) ∧ Renk(kırmızı) ⇒ BaşarısızKaldırma*
$o_{1:5}$	*Kategori(kutu) ∧ Şekil(prizma) ∧ Malzeme(kağıt) ⇒ BaşarılıKaldırma* *Kategori(kutu) ∧ Renk(yeşil) ⇒ BaşarılıKaldırma* *Kategori(lobut) ∧ Renk(turuncu) ∧ Malzeme(plastik) ∧ Boyut(küçük) ⇒ BaşarısızKaldırma* *Kategori(kutu) ∧ Renk(kırmızı) ⇒ BaşarısızKaldırma*
$o_{1:6}$	*Renk(yeşil) ⇒ BaşarılıKaldırma* *Kategori(kutu) ∧ Şekil(prizma) ∧ Renk(siyah) ∧ Malzeme(kağıt) ⇒ BaşarılıKaldırma* *Kategori(kutu) ∧ Renk(kırmızı) ⇒ BaşarısızKaldırma* *Kategori(pin) ∧ Renk(turuncu) ⇒ BaşarısızKaldırma*
$o_{1:7}$	*Renk(yeşil) ⇒ BaşarılıKaldırma* *Kategori(top) ∧ Şekil(küre) ∧ Renk(mor) ⇒ BaşarılıKaldırma* *Kategori(kutu) ∧ Şekil(prizma) ∧ Renk(siyah) ∧ Malzeme(kağıt) ⇒ BaşarılıKaldırma* *Kategori(kutu) ∧ Renk(kırmızı) ⇒ BaşarısızKaldırma* *Kategori(lobut) ∧ Renk(turuncu) ⇒ BaşarısızKaldırma*
$o_{1:8}$	*Renk(yeşil) ⇒ BaşarılıKaldırma* *Kategori(top) ∧ Şekil(küre) ⇒ BaşarılıKaldırma* *Kategori(kutu) ∧ Şekil(prizma) ∧ Renk(siyah) ∧ Malzeme(kağıt) ⇒ BaşarılıKaldırma* *Kategori(kutu) ∧ Renk(kırmızı) ⇒ BaşarısızKaldırma* *Kategori(lobut) ∧ Renk(turuncu) ⇒ BaşarısızKaldırma*

Şimdi robotun amacının 3 bloğu üst üste yerleştirmek olduğunu düşünelim. İlk yerleştirme eylemi ($stack(b,c)$) başarılı, ikinci yerleştirme eylemi ($stack(a,b)$) başarısız olduğunda şu pozitif ve negatif gözlemler elde edilir:

$$holding(b) \wedge clear(c) \wedge ontable(c) \Rightarrow \neg StackFailure(b,c)$$

$$holding(a) \wedge clear(b) \wedge ontable(c) \wedge on(b,c) \Rightarrow StackFailure(a,b)$$

Sonuç olarak şu hipotez üretilir:

$$holding(A) \wedge clear(B) \wedge ontable(C) \wedge on(B,C) \Rightarrow StackFailure(A,B)$$

Başarısız olan eylem için sadece bir gözlem olduğu için hipotez gözlenen bütün bilgileri içerir. ILP'nin güçlü yönlerinden biri de, eğer ortamın nedensel modeli varsa hipotezlerin genelleştirilebilmesidir. Örneğin bilgi tabanında şu kural mevcut ise:

$$ontable(X) \wedge clear(X) \Rightarrow Tower(1,X)$$

$$on(Y,X) \wedge Tower(N-1,X) \wedge (N > 1) \Rightarrow Tower(N,Y)$$

bu kural kullanılarak hipotez şu şekilde genelleştirilebilir:

$$Tower(2,B) \Rightarrow StackFailure(A,B)$$

4 bloktan oluşan bir senaryoda, eğer robot üçüncü *yerleştirme* eyleminden sonra hata oluştuğunu farkederse, pozitif ve negatif gözlemler şu şekilde oluşur:

$$holding(c) \wedge clear(d) \wedge ontable(d) \Rightarrow \neg StackFailure(c,d)$$

$$holding(b) \wedge clear(c) \wedge ontable(d) \wedge on(c,d) \Rightarrow \neg StackFailure(b,c)$$

$$holding(a) \wedge clear(b) \wedge ontable(d) \wedge on(c,d) \wedge on(b,c) \Rightarrow StackFailure(a,b)$$

Bu durumda elde edilen hipotez:

$$on(B,C) \wedge on(C,D) \Rightarrow StackFailure(A,B)$$

$$Tower(3,B) \Rightarrow StackFailure(A,B)$$

Blokların görsel özellikleri (boyut, renk vs.) gözlenebiliyor ise bu özellikler de temsilde yer alabilir. Beklenildiği üzere nitelik tabanlı sınıflandırıcılar yukarıda belirtildiği şekildeki ilişkileri kolaylıkla modelleyemezler. Bunun için bütün ilişkilerin nitelik olarak kodlanması gerekmektedir.

Yukarıda verilen örnek için, oluşan hatanın sebepleri; görüntü işleme problemleri, belli bir yükseklikten sonra robot kolunda oluşan ya da düzgün yerleştirilemeyen bloğun neden olduğu dengesizlikler olabilir. Eğer bu problemler için herhangi bir nicel veya nitel ölçüm yolu yoksa, hatanın tam olarak nedenini bulmak mümkün olamamaktadır. Ancak tezde önerilen sistem ile hatanın altında yatan neden tam olarak bulunamasa bile, örneğin robot, 2'den fazla bloğu üst üste koyamayacağına karar verip gelecekteki görevleri için planlarını buna göre belirleyebilir.

Hatalardan sonra oluşan durumların ortamdaki etkisine göre hata durumları ile ilişkilendirilen maliyetler belirlenebilir. Hipotezlerden maliyet belirlemeye geçerken, riskten kaçınılan ya da risk alınan stratejiler kullanılabilir. Eğer oluşabilecek hata herhangi bir zarara yol açmayacak ise robot risk alarak eylemlerini yürütebilir. Diğer taraftan, hata sonucu zararlı durumların oluşma ihtimali var ise güvenliği garanti altına almak için eylemler riskten kaçınılarak yürütülmeye çalışılabilir.

6.4 Alternatif Hipotez Temsili

Bu bölümde tez çalışması boyunca kullanılan hipotez yapısının alternatif şekillerde nasıl ifade edilebileceği incelenmektedir.

Robotun başarısız eylemleri için hipotezlerini oluştururken kullandığı yapı şu şekilde belirlenmiştir:

$$\textit{Özellik1(x)} \wedge \textit{Özellik2(y)} \wedge ... \Rightarrow \textit{EylemİsmiI\_Başarısız}$$

$$\textit{Özellik2(y)} \wedge \textit{Özellik3(a)} \wedge ... \Rightarrow \textit{EylemİsmiI2\_Başarısız}$$

...

Bu şekildeki bir temsilin dezavantajı, her bir eylem için ayrı gözlem kümelerinin belirlenmesinin gerekliliğidir. Her bir başarısız eylem ayrı bir sınıfı temsil eder. Bu sorunu çözmek için temsil şu hale dönüştürülebilir:

$$EylemİsmiN\_Başarısız \wedge Özellik1(x) \wedge Özellik2(y) \wedge ... \Rightarrow Başarısız$$

$$EylemİsmiM\_Başarısız \wedge Özellik2(y) \wedge Özellik3(a) \wedge ... \Rightarrow Başarısız$$

...

Böylece başarısız olan eylemin kendisi hipotezin sol tarafında saklanarak sınıf değerleri sadece başarılı ve başarısız eylem olarak belirlenmiş olur. Bütün veri kümesi tek bir yerde toplanıp, hipotezler daha genel sınıfları temsil edecek hale getirilmiş olur. Bu temsilde karşılaşılacak olan bir sorun ise veri kümesinin boyutunun artmasıyla öğrenme algoritmasının karmaşıklığının da artacak olmasıdır.

7. SONUÇLAR

Bu tez çalışmasında, bilişsel robotlarda görevlerin başarıyla yürütülebilmesi için bir sistem ve bu sistem içinde deneyimsel bir öğrenme yöntemi sunulmuştur. Önerilen yaklaşım, eylem yürütme hatalarından öğrenmek için deneyim tabanlı öğrenme yöntemi olan ILP metodunu içermektedir. Bu öğrenme süreci; planlama, yürütme, gözlemleme ve öğrenme adımlarını içeren bir yapının parçası olarak sunulmuştur. Öğrenme süreci hem daha verimli yürütme modelleri elde etmek hem de hata durumları ile değişik şartların ilişkilendirilmesi için hipotezler üretmek amacıyla kullanılmıştır.

Tezde önerilen yaklaşımla, robot elde ettiği her yeni gözlemde hipotez uzayını yeniden düzenler. Üretilen hipotezler yüklem mantığı gösterim dili ile kurallar bütünü olarak ifade edilerek hem çıkarım hem de planlama için kullanılabilir. Bu kurallarda; gözlenebilen nesne özellikleri, nesneler arası ilişkiler ve ortamla ilgili olgular kullanılabilir. Kısmi olarak belirtilen ortam durumları da bu kurallar ile kolaylıkla temsil edilebilir. Ayrıca ILP, robotun önceki bilgi birikiminin kullanımına da olanak sağlamaktadır. Bu bilgiler ile hipotezlerin genelleştirilmesi sağlanabilir. ILP kısmi ortam bilgileri ile de çalışabildiği için tümüyle gözlenemeyen ortamlarda da kullanılabilir. Bütün bu avantajlar, deneyimsel öğrenme süreci için ILP yöntemini daha verimli kılmaktadır.

Öğrenme sürecinin sonuçları robotun ileri planlama görevlerindeki kararları yönlendirmek için kullanılabilir. Hata durumlarının oluşma şartları hipotezler ile ifade edilir ve daha sonra bu hipotezler eylemlerin maliyet hesaplarında kullanılabilir. Böylelikle başarısız olan eylemler için yapılan maliyet hesapları ile gelecekte bu eylemlerin seçimi engellenerek genel sistem daha gürbüz bir hale getirilmiş olur.

Önerilen yaklaşımda kullanılan öğrenme yönteminin başarım ölçümü için çeşitli deneyler tasarlanmıştır. İlk deneyler, benzetim ile elde edilen veri kümeleri üzerinde gerçekleştirilmiştir. Bu deneyde iki farklı ILP algoritmasının (PROGOL

ve FOIL) başarımı ile nitelik tabanlı öğrenme algoritmaları olan ID3 ve Bayes Network algoritmalarının başarımı karşılaştırılmış ve PROGOL algoritmasının diğer algoritmalardan daha iyi sonuçlar verdiği görülmüştür. İkinci deney gerçek robot ve nesnelerden oluşan bir ortamda gerçekleştirilmiştir. Bu ortamda, robotun farklı nesneler üzerinde yürüttüğü *kaldırma* eylemlerinden elde ettiği deneyimi kullanarak ürettiği hipotezler incelenmiş ve robotun üretilen hipotezleri gelecekteki eylemlerini çok daha verimli bir şekilde yürütmek için kullanabileceği görülmüştür.

ILP'nin nitelik tabanlı sınıflandırıcılar karşısındaki önemli özelliklerinden biri de hipotezlerini öğrenirken bilgi birikimini kullanabilmesidir. Örnek bir senaryo üzerinde bu özellik incelenerek, bilgi birikiminin hipotezlerin genelleştirilmesine olanak sağladığı gözlenmiştir.

KAYNAKLAR

[1] **Aker, E., Erdogan, A., Erdem, ve Patoglu, V.,** 2011. Housekeeping with multiple autonomous robots: Representation, reasoning and execution. In *AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning*.

[2] **Bacchus, F. ve Winter, M.,** 2001. Planning with resources and concurrency a forward chaining approach.

[3] **Beetz, M., Jain, D., Mösenlechner, L. ve Tenorth, M.,** 2010. Towards performing everyday manipulation activities. *Robotics and Autonomous Systems*, 58(9):1085 – 1095.

[4] **Beetz, M., Mösenlechner, L. ve Tenorth, M.,** 2010. Cram - a cognitive robot abstract machine for everyday manipulation in human environments. In *IROS*, pages 1012–1017. ISBN 978-1-4244-6674-0.

[5] **Bicchi, A.,** 2000. Hands for dexterous manipulation and robust grasping: a difficult road toward simplicity. *IEEE Transactions on Robotics and Automation*, 16(6):652–662.

[6] **Bouguerra, A., Karlsson, L. ve Saffiotti, A.,** 2007. Active execution monitoring using planning and semantic knowledge. In *Proc. of the ICAPS Workshop on Planning and Plan Execution for Real-World Systems, Providence, RI, 2007*, pages 9–15.

[7] **Bratko, I., Cestnik, B. ve Kononenko, I.,** 1996. Attribute-based learning. *AI Commun.*, 9(1):27–32.

[8] **Burbridge, C. ve Dearden, R.,** 2012. Learning the geometric meaning of symbolic abstractions for manipulation planning. In Guido Herrmann, Matthew Studley, Martin J. Pearson, Andrew T. Conn, Chris Melhuish, Mark Witkowski, Jong-Hwan Kim, and Prahlad Vadakkepat, editors, *TAROS*, volume 7429 of *Lecture Notes in Computer Science*, pages 220–231. Springer. ISBN 978-3-642-32526-7.

[9] **Cantrell, R., Talamadupula, K., Schermerhorn, P., Benton, J., Kambhampati, S. ve Scheutz, M.,** 2012. Tell me when and why to do it!: run-time planner model updates via natural language instruction. ISBN 978-1-4503-1063-5. doi: 10.1145/2157689.2157840.

[10] **Chitta, S., Jones, E., Ciocarlie, M. ve Hsiao, K.,** 2012. Perception, planning, and execution for mobile manipulation in unstructured environments. *IEEE Robotics and Automation Magazine, Special Issue on Mobile Manipulation*, 19.

[11] **Cubek, E.,** 2011. Learning and application of high-level concepts with conceptula spaces and pddl. *ICAPS-11 Workshop on Planning and Learning (PAL)*.

[12] **Cubek, R. ve Ertel, W.,** 2011. Learning and execution of high-level concepts with conceptual spaces and pddl. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS) Workshop on Learning and Planning*.

[13] **Do, M. ve Kambhampati, S.,** 2000. Solving planning-graph by compiling it into csp.

[14] **Ersen, M. ve Sariel-Talay, S.,** 2012. Learning interactions among objects through spatio-temporal reasoning. In *Proceedings of the AAAI-12 Workshop on Problem Solving Using Classical Planners at the Twenty-Sixth AAAI Conference*.

[15] **Hanheide, M., Hawes, N., Wyatt, J., Göbelbecker, M., Brenner, M., Sjöö, K., Aydemir, A., Jensfelt, P., Zender, H. ve Kruijff, G.,** 2010. A framework for goal generation and management. In *Proceedings of the AAAI Workshop on Goal-Directed Autonomy*, Atlanta, GA, USA. AAAI.

[16] **Hinterstoisser, S., Cagniart, C., Ilic, S., Sturm, P., Navab, N., Fua, P. ve Lepetit, V.,** 2012. Gradient response maps for real-time detection of textureless objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(5):876–888. ISSN 0162-8828.

[17] **Elfring, M.J.G., Dries, S. ve Bruyninckx, H.,** 2011. In *IROS Workshop on Knowledge Representation for Autonomous Robots*.

[18] **Karapinar, S., Altan, D. ve Sariel-Talay, S.,** 2012. A robust planning framework for cognitive robots. In *Proceedings of the AAAI-12 Workshop on Cognitive Robotics (CogRob)*.

[19] **Kecici, S. ve Sariel-Talay, S.,** 2010. Tlplan-c: An extended temporal planner for modeling continuous change. In *Proceedings of the 20th International Conference on Automated Planning and Scheduling (ICAPS), Combining Action and Motion Planning Workshop*.

[20] **Kober, J., Wilhelm, A., Oztop, E. ve Peters, J.,** 2012. Reinforcement learning to adjust parametrized motor primitives to new situations. *Auton. Robots*, 33(4):361–379.

[21] **Kroemer, O., Ugur, E., Oztop, E. ve Peters, J.,** 2012. A kernel-based approach to direct action perception. In *ICRA*, pages 2605–2610. IEEE. ISBN 978-1-4673-1403-9.

[22] **Lowe, D.G.,** 2004. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110. ISSN 0920-5691.

[23] **Mausam, W. ve Weld, D.S.,** 2008. Planning with durative actions in stochastic domains. *Journal of Artificial Intelligence Research*, 31(1):33–82. ISSN 1076-9757.

[24] **Mösenlechner, L. ve Beetz, M.,** 2011. Parameterizing Actions to have the Appropriate Effects. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

[25] **Muggleton, S.,** 1995 Inverse Entailment and Progol. *New Generation Computing, Special issue on Inductive Logic Programming*, 13(3-4): 245–286, 1995.

[26] **Nguyen, T., Do, M., Gerevini, A., Serina, I., Srivastava, B. ve Kambhampati, S.,** 2011. Planning with partial preference models. *CoRR*, abs/1101.2279.

[27] **Pettersson, O.,** 2005. Execution monitoring in robotics: A survey. *Robotics and Autonomous Systems*, 53:73–88.

[28] **Quinlan, J.,** 1990. Learning logical definitions from relations. *Machine Learning*, 5:239–266.

[29] **Ridge, B., Skocaj, D. ve Leonardis, A.,** 2009. Unsupervised Learning of Basic Object Affordances from Object Properties. In *Proceedings of the Fourteenth Computer Vision Winter Workshop (CVWW)*, pages 21–28, Eibiswald, Austria.

[30] **Russell, S.J., Norvig, P., Candy, J.F., Malik, J.M. ve Edwards, D.D.,** 1996. *Artificial intelligence: a modern approach*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA. ISBN 0-13-103805-2.

[31] **Rusu, R.B., Bradski, G., Thibaux, R. ve Hsu, J.,** 2010. Fast 3d recognition and pose using the viewpoint feature histogram. In *Proceedings of the 23rd IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

[32] **Schalkoff, R.J.,** 1990. *Artificial Intelligence: An Engineering Approach*. Mcgraw-Hill College.

[33] **Sjoo, K., Pronobis, A. ve Jensfelt, P.,** 2011. Functional topological relations for qualitative spatial representation. In *Advanced Robotics (ICAR), 2011 15th International Conference on*.

[34] **Smith, D.E., Frank, J. ve Jónsson, A.K.,** 2000. Bridging the gap between planning and scheduling. *Knowledge Engineering Review*, 15:2000.

[35] **Talamadupula, K., Schermerhorn, P., Benton, J., Kambhampati, S. ve Scheutz, M.,** 2011. Planning for agents with changing goals. In *ICAPS 2011 System Demonstration*, Freiburg, Germany, June.

[36] **Ugur, E., Sahin, E. ve Oztop, E.,** 2011. Unsupervised learning of object affordances for planning in a mobile manipulation platform. In *ICRA*, pages 4312–4317. IEEE.

[37] **Usug, U.C. ve Sariel-Talay, S.,** 2011. Dynamic temporal planning for multirobot systems. In *Proceedings of the AAAI-11 Workshop on Automated Action Planning for Autonomous Mobile Robots (PAMR)*.

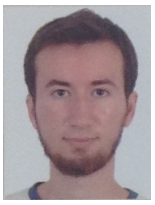
[38] **Wolpert, D.M. ve Ghahramani, Z.,** 2000. Computational principles of movement neuroscience. *Nature Neuroscience*, 3 Suppl:1212–1217, November. ISSN 1097-6256. doi: 10.1038/81497.

[39] **Yang, Q.,** 1997. *Intelligent planning: a decomposition and abstraction based approach*. Springer-Verlag, London, UK, UK. ISBN 3-540-61901-1.

[40] **Zelle, J.M., Mooney, R.J. ve Konvisser, J.B.,** 1994. Combining top-down and bottom-up techniques in inductive logic programming. In *Proceedings of the Eleventh International Workshop on Machine Learning (ML-94)*, pages 343–351, Rutgers, NJ, July.

[41] **Zettlemoyer, L., Pasula, H. ve Kaelbling, L.,** 2005. Learning planning rules in stochastic worlds. In *AAAI*.

ÖZGEÇMİŞ



Ad Soyad: Sertaç KARAPINAR

Doğum Yeri ve Tarihi: Aydın 7 Şubat 1988

Adres: Türkiye - İstanbul(Avr.) - Bakırköy - Ataköy

E-Posta: karapinarse@itu.edu.tr

Lisans: İTÜ Bilgisayar Mühendisliği

Y. Lisans: İTÜ Bilgisayar Mühendisliği

TEZDEN TÜRETİLEN YAYINLAR/SUNUMLAR

- **Karapinar, S., Altan, D. ve Sariel-Talay, S.**, 2012. A Robust Planning Framework for Cognitive Robots. *The AAAI-12 Workshop on Cognitive Robots (CogRob).*

- **Karapinar, S. ve Sariel-Talay, S.**, 2012. Failure Handling In a Planning Framework. *The AAAI-12 Student Workshop.*