İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

ÇOKLU ROBOT SİSTEMLERİNDE DİNAMİK ZAMANSAL PLANLAMA

YÜKSEK LİSANS TEZİ

Celal Uğur USUĞ

Anabilim Dalı : Bilgisayar Mühendisliği

Programı : Bilgisayar Mühendisliği

HAZİRAN 2011

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

ÇOKLU ROBOT SİSTEMLERİNDE DİNAMİK ZAMANSAL PLANLAMA

YÜKSEK LİSANS TEZİ

Celal Uğur USUĞ
(504081508)

Tezin Enstitüye Verildiği Tarih : 06 Mayıs 2011
Tezin Savunulduğu Tarih : 09 Haziran 2011

Tez Danışmanı : Yrd. Doç. Dr. Sanem SARIEL TALAY (İTÜ)
Diğer Jüri Üyeleri : Prof. Dr. Nadia ERDOĞAN (İTÜ)
Yrd. Doç. Dr. Esra ERDEM (SÜ)

HAZİRAN 2011

Anne ve babama,

ÖNSÖZ

Bu çalışmayı oluşturuken bir çok konu ve aşamada destek, fikir ve görüşlerini aldığım danışmanım Sanem SARIEL TALAY'a teşekkürlerimi sunarım.

Haziran 2011 Celal Uğur Usuğ

İÇİNDEKİLER

Sayfa

ÖNSÖZ ..... **v**
İÇİNDEKİLER ..... **vii**
KISALTMALAR ..... **ix**
ÇİZELGE LİSTESİ ..... **xi**
ŞEKİL LİSTESİ ..... **xiii**
ÖZET ..... **xv**
SUMMARY ..... **xvii**
1. GİRİŞ **1**
1.1 Amaç ve Katkılar 2
2. PLANLAMA **5**
2.1 Otonom Planlama 5
2.2 Planlama Problemi Modeli ve Dilleri 5
2.2.1 Planlama problemi 5
2.2.2 Planlama dilleri 6
2.3 Arama Uzayı ile Planlama Yöntemleri 8
2.3.1 İleri durum uzayı arama 8
2.3.2 Geri durum uzayı arama 9
2.4 Kısmi Sıralı Planlama Yöntemi 9
2.5 Planlama Grafı 10
2.6 Ara-hedef Bölümleme ve Artımlı Planlama 11
3. ROBOTLARLA PLANLAMA **13**
3.1 Giriş 13
3.2 Planlamada Zaman ve Kaynak 13
3.2.1 TLPlan planlayıcısı 15
3.3 Planlama Ortam Bilgisi Belirsizliği 15
3.3.1 Koşullu planlama 16
3.3.1.1 Olasılıksal planlama 16
3.3.2 Yürütme gözlemleme ve yeniden planlama 17
3.4 Planlama Temsil Bilgisi Yetersizliği 18
4. PROBLEM TANIMI **21**
4.1 Giriş 21
4.2 Dinamik Zamansal Planlama Problemi 21
4.2.1 Tanımlar 21
4.2.2 Dinamik zamansal planlama problemi 23
4.3 Örnek Bir Problem 23
5. DİNAMİK ZAMANSAL PLANLAMA SİSTEMİ **25**
5.1 Giriş 25
5.2 Sistem Modeli 25
5.3 Sistem Algoritmaları 29
5.4 Yürütme Zamanında Hata Sezme ve Robot Altyordamları 32

6. DİNAMİK ZAMANSAL PLANLAMA SİSTEMİ 33
6.1 Giriş 33
6.2 Dinamik Planlama Problemi İçin Bilinen Çözüm Yöntemleri 33
6.3 Önerilen Yöntemler 33
6.3.1 Sözde operatör yöntemi 34
6.3.2 Hata giderici operatörleri kopyalama yöntemi 37
6.3.3 Hatalı operatör kopyalama yöntemi 39
7. DENEYSEL SONUÇLAR 41
7.1 Giriş 41
7.2 Planlama Deney Kümeleri Üzerinde Başarım Analizi 41
7.2.1 Birinci deney kümesi 41
7.2.1.1 Birinci deney sonuçları 44
7.2.2 İkinci deney kümesi 46
7.3 Benzetim Ortamı Başarım Analizi 50
7.3.1 Benzetim ortamı 50
7.3.2 Deney mimarisi 51
7.3.3 Başarım analizi 52
8. SONUÇ VE İLERİ ÇALIŞMALAR 59
KAYNAKLAR 61

KISALTMALAR

ADL	**:** Action Definition Language
DEP	**:** Decision Epoch Planning
STRIPS	**:** Stanford Research Institute Problem Solver
PDDL	**:** Planning Domain Definition Language
STN	**:** Simple Temporal Network
TGP	**:** Temporal Graph Plan

ÇİZELGE LİSTESİ

Sayfa

Çizelge 2.1 : Kargo problemi operatör temsili örneği. 6
Çizelge 2.2 : Kargo problemi için örnek başlangıç ve hedef durumu ve plan çıktısı.. 6
Çizelge 2.3 : STRIPS, ADL ve PDDL dillerinin karşılaştırılması. 7
Çizelge 5.1 : Yürütme mesajları yapısı. 27
Çizelge 7.1 : Birinci deney ortamı yöntem ve plan stratejileri karşılaştırmaları. 45
Çizelge 7.2 : Benzetim ortamı sistem mesajları yapısı. 51

ŞEKİL LİSTESİ

Sayfa

Şekil 2.1 : İleri durum uzayı arama yöntemi. 8
Şekil 2.2 : Geri durum uzayı arama yöntemi. 9
Şekil 2.3 : Ara-hedef bölümleme. 10
Şekil 3.1 : Plan onarımı. 18
Şekil 5.1 : Dinamik zamansal planlama sistemi. 25
Şekil 5.2 : Dinamik zamansal planlama sistemi bileşenleri. 26
Şekil 5.3 : Dinamik zamansal planlama sistemi durum diyagramı. 27
Şekil 7.1 : Planlama kümeleri başarım analizi için birinci deney ortamı örneği. 42
Şekil 7.2 : Birinci deney kümesi hatasının çözümü için sözde operatör yöntemi tanım kümesi güncellemesi. 43
Şekil 7.3 : Birinci deney kümesi hatasının çözümü için hata giderici operatörleri kopyalama yöntemi tanım kümesi güncellemesi. 43
Şekil 7.4 : Birinci deney kümesi hatasının çözümü için hatalı operatör kopyalama yöntemi tanım kümesi güncellemesi. 44
Şekil 7.5 : Planlama kümeleri başarım analizi için ikinci deney ortamı örneği. 47
Şekil 7.6 : Yöntem 1 ile ikinci deney kümesi için üretilen plan. 48
Şekil 7.7 : Yöntem 2 ile ikinci deney kümesi için üretilen plan. 49
Şekil 7.8 : Ara hedef planlama ile ikinci deney kümesi için üretilen plan. 49
Şekil 7.9 : Yöntem 3 ile ikinci deney kümesi için üretilen plan. 50
Şekil 7.10: Benzetim ortamı başarım analizi deney mimarisi. 52
Şekil 7.11 : Benzetim programında bir hata çözme senaryosu ortamı. 53
Şekil 7.12 : Benzetim ortamı deneylerindeki hatanın çözümü için sözde operatör yöntemi tanım kümesi güncellemesi. 54
Şekil 7.13 : Nesnelerin hedef noktalarına taşınmasını içeren tüm planın işleyişi. 56
Şekil 7.14 : Robotların ilk ve hata sonrası planlama temsili güncellemesi sonrası yeni planları. 57
Şekil 7.15 : Yaratılan ve artımlı olarak güncellenen topolojik haritalar 57

ÇOKLU ROBOT SİSTEMLERİNDE DİNAMİK ZAMANSAL PLANLAMA

ÖZET

Gezgin robotların birbirlerine bağımlı ve karmaşık görevleri etkin şekilde yürütebilmesi için otonom eylem planlama yöntemlerinin kullanılması gerekir. Fakat gerçek dünya kısıtlarını karşılayabilmek için planlama problem tanım kümesini (*domain*) gerçekçi olarak temsil etmek muazzam bir çaba gerektirir. Bu yüzden gezgin robot sistemleri için eylem temsillerini (örneğin etki ve önkoşullarını) tüm durumlarda en mükemmel şekilde temsil edememek bir başka belirsizlik kaynağı teşkil etmektedir. Tanım kümesi temsili tam ve eksiksiz olmadığında, bir planlayıcı dinamik gerçekleşebilecek olaylar karşısında aslında mümkün olmasına rağmen geçerli bir plan üretemeyebilir.

Bu çalışmada gerçek-zaman yürütme hatalarına karşı alternatif planlar üretmek için tanım kümesi güncelleme metodları üzerine odaklanılmıştır. Sınırlı bir arkaplan bilgisinde dahi güncellenmiş tanım kümesi temsiline dayalı yeni bir *yürütülebilir* plan oluşturulması hedeflenmiştir. Bu amaçla önerilen yöntemlerle, plan onarma için gerekli operatörler, ortamda yönlendirici uzmanlar olmadığı durumlar için de, var olan tanım kümesi temsili ile geçerli plan bulmak mümkün olmamasına rağmen alternatif planlar üretilebilir. Hatalar yürütme anında veya yürütme öncesi simülasyon sürecinde tespit edilir. Problemin çoklu robot düzeyine çekilmesi durumundaki çözümün kalitesinin artırımını sağlamak, problemi zamansal açıdan ele almayı gerektirmiştir. Bu da problemi zamansal açıdan çözmeyi sağlayabilen zamansal planlayıcıyı ve buna uygun bir sistemin oluşturulmasını gerektirmiştir.

Önerilen yöntemler biribirleri arasında ve ara hedef planlama ile bir dizi deney kümesi için karşılaştırılmıştır. Temel planlayıcı olarak TLPlan zamansal planlayıcısı kullanıldığından robotların varolan bilgileri ile toplam plan süresi eniyilemesi başarılmıştır. Gerçekçi çoklu robot simülatörü Webots ve TLPlan planlayıcısı ile haberleşen merkezi bir dinamik zamansal planama sistemi yapısı inşa edilmiştir. Gezgin robotları içeren bir hata çözme senaryosu Webots simülatöründe verilmiştir.

DYNAMIC TEMPORAL PLANNING FOR MULTIROBOT SYSTEMS

SUMMARY

The use of automated action planning techniques is essential for efficient mission execution of mobile robots. However, a tremendous effort is needed to represent planning problem domains realistically to meet the constraints of real-world action execution. Therefore, there is another source of uncertainty for mobile robot systems due to the impossibility of perfectly representing action representations (e.g., preconditions and effects) in all circumstances. When the domain representations are not complete, a planner may not be capable of constructing a valid plan for dynamic events even when it is possible.

Robots may take on missions under time constraints. To overcome time constraints temporal planners must be used. Temporal planners, capable of generating efficient sequences of durative actions, are convenient to be used with real robot missions. There have been significant advances in devising efficient temporal planners for continually enhanced benchmark planning domains. However, these algorithms operate at a high level and are not completely capable of dealing with hardware or physical environment limitations. In particular, real world is partially observable and involves different sources of uncertainty due to noisy sensor information, unexpected outcomes of actions and failures.

This research presents a dynamic temporal planning framework for mobile robots to handle action failures due to environmental issues. Different from earlier studies, the presented solution constructs alternative plans even when guidance by experts, relevant new information or repairing operators for replanning is not available. Real outcomes of existing operators may not be completely known in advance (*e.g.*, due to abstraction) or valid plans are not possible with the existing representation of the domain.

The proposed framework includes three domain update procedures that provide flexible reasoning tools to replan accordingly for the resolution of a failure by using the intuitive principle of locality of failure. Failures are detected either during runtime or earlier by a plan simulation process. Background knowledge needed for updating the existing domain operators is almost negligible and a generic approach is applied by searching for effects in existing operators that may resolve the failure. After the required domain updates are performed, a replanning approach is employed as opposed to repairing since makespan (near-) optimal solutions are targeted (Cushing, Benton and Kambhampati 2008). TLPlan (Bacchus and Ady 2001) is used as the forward chaining temporal planner in the system to construct makespan-optimized plans. Low and high-level planning procedures of the robot system are made compatible by using efficient domain (*e.g.*, map) representations.

A centeral dynamic temporal planning system communicating with both the realistic Webots simulator and TLPlan temporal planner is constructed. Experiments are conducted to compare the three presented domain update approaches and the subgaol planning method on specific problem sets with two failure scenorios. A failure resolution scenario is given in the Webots simulator with mobile robots

1. GİRİŞ

Son zamanlardaki teknolojik gelişmelerle robotların algı ve hareket kabiliyetlerindeki artışlar, robotların kendi başına yüksek seviyeli ve karmaşık görevleri yerine getirebilmelerinin önünü açmıştır. Karmaşık ve yüksek seviyeli hedefler bir dizi eylemin sıralı yürütülmesi sonucunda ulaşılabilir. Robotlar birer etmen olarak düşünüldüğünde kendilerine verilebilecek bu karmaşık ve yüksek seviyeli işleri yapabilmek için gereken bu eylemler dizisini eylem planlama ile üretip hedeflerine ulaşabilerler. Robot sistemleri üzerinde eylem planlama konusu, son dönemlerde üzerinde çalışılmaya başlanan ve kendi içerisinde bir çok problemi barındıran güncel bir araştırma konusu haline gelmiştir.

Robotlar gerçek dünyada zaman kısıtlamalı görevler üstlenebilirler. Bu durumda planlama ile birlikte eşzamanlı çizelgeleme ile zamansal maliyetlerinin de göz önünde bulundurulması daha etkin sonuçlar üretmektedir. Bu sayede çizelgelenmiş plan çıktısının zamansal kalitesi daha iyi olmakta ve zaman kısıtlarına uygun plan üretilebilmesi sağlanabilmektedir.

Robotlar planlayıcı tarafından üretilen bir planı gerçek dünyada yürütürlerken bazı gerçek dünya kısıtlarıyla karşı karşıya gelebilirler. Çeşitli algılayıcılar ile kısmi olarak gözlenebilen, belirsizlik ve dinamizm içeren bir dünyada çalışan robotlar söz konusu olduğunda yazılım etmenleri için kabul edilen varsayımlar geçersiz hale gelmektedir. Bu yüzden tasarımda kısıtlar artmakta ve var olan problemin daha karmaşık bir halde ele alınması gerekmektedir. Problemin karmaşıklığı doğal olarak eylemlerin temsilindeki karmaşıklıkları da getirmektedir. Bu durumda, günümüz işlem gücü ile problemleri çözmek zorlaşmakta veya plan üretmek uzun sürmekte ve zamansal maliyet artmaktadır. Bu yüzden planlama tanım kümesi tasarımcıları (*planning domain designers*) gerçek dünya eylemlerinin planlayıcı seviyesindeki eylem temsilleri olan planlama operatörlerini (*planning operators*) karmaşıklığı daha az olacak şekilde tasarşaöa yoluna gidebilmektedir. Karmaşıklığı azaltmak için eylemlerin etkileri ve önkoşulları eksik ve yetersi verilmiş olabilir. Bu durumda ise planlama operatörleri, gerçek dünya eylemlerini temsil edecek çözünürlükte

olmamakta veya robotların içerisinde oldukları bazı durumlar plalama ortamında tanımsız hale gelmektedir.

Bir planın gerçek dünya üzerinde yürütülmesi aşamasında dış kaynaklı beklenmeyen veya öngörülemeyen olaylar planı geçersiz kılabilir. Bu durumlarda gerekli ortam değişikliği algılanıp yeniden plan yapılması veya var olan planın tamir edilmesi gerekmektedir. Fakat planlama tanım kümesi ve eylem temsillerinin yetersiz veya öngörülemez olduğu durumlarda planlayıcı, dinamik olaylar karşısında belirli eylemleri uygulayarak hedef duruma ulaşılması mümkün olmasına rağmen geçerli bir plan üretemeyebilir.

1.1 Amaç ve Katkılar

Bu tezde, robotların karşılaşabilecekleri yürütme hatalarına karşı gerekli planlama tanım kümesi değişikliklerini yaparak alternatif planların bulunabilmesini sağlayan yöntemlerin geliştirilmesi hedeflenmiştir. Tezde önerilen genel amaçlı yöntemler ile, planlama tanım kümesi içinde eylem temsillerinin eksik olduğu durumlarda bile, güncellenmiş planlama tanım kümesi ile yürütülebilir bir planın oluşturulması mümkün olabilmektedir.

Zamansal planlama yapabilen TLPlan planlayıcısı kullanılarak çoklu robotlara ortak çizelgelenmiş bir plan üretilmesini sağlayan, gerçekçi Webots simülatörü ile TCP/IP soket mesajlaşma ile robotlara planın kendilerine ait bölümlerini aktaran, yürütme esnasında simülatörden aldığı bilgilerle robotları ve ortamı gözetleyen ve yürütme de olabilecek hata durumlarında gerekli tanım kümesi güncellemeleri yapabilen bir sistem oluşturulmuştur.

2. bölümde klasik planlama ve klasik planlamada var olan temel kavramlar ve çeşitli planlama yöntemleri analtılmış, 3. bölümde klasik planlamanın gerçek dünyada neden yetersiz kaldığına ilişkin tartışılmı, gerçek dünyada planlamanın getirdiği zorluklardan bahsedilmiş ve bu zorluklarla başedebilmek için literatürde geçen eski ve güncel yaklaşımlara yer verilmiştir. 4. bölümde bu tez kapsamında araştırılan problemin tanımlarla sınırları çizilmiş ve örneklendirilmiş, 5. bölümde ise bu problemi çözebilmek için geliştirilen dinamik zamansal planlama sistemi modeli verilmiştir. 6. bölümde genel geçer planlama temsili güncelleyen yöntemler algoritmalarıyla birilkte sunulmuştur. 7. bölümde, önerilen yöntemlerin deneysel

olarak birbirleriyle karşılaştırıldığı ve 5. bölümde verilen modeli referans alarak geliştirilmiş sistem ile birlikte gerçekci WEBOTS benzetim programı kullanılarak yapılan deneylerin sonuçları sunulmuştur. Son olarak 8. bölümde elde edilen sonuçlar özetlenmekte ve ileriye yönelik çalışmalardan bahsedilmektedir.

2. PLANLAMA

2.1 Otonom Planlama

Etmenler veya robotlar bir ortamda belirli görev yada görevleri yerine getirmek için tasarlanırlar. Bu görevler için uygulanması gereken eylem dizileri bazı uygulama alanlarında daha önceden belirli ve değişmez veya önceden bilinen bir sekansta olabilir. Fakat önceden kestirilemeyen görevlerin veya dinamik ortam etkileşiminin söz konusu olduğu durumlarda hedefe varan eylem dizilerini önceden bilmek olanaksızlaşır. Bu tür durumlarda hedefe ulaşma işi planlama içeriği altında ele alınması gerekir.

En genel tabiriyle planlama, belirli bir durum uzayından yine belirli bir hedef durum uzayına ulaşmak için gereken eylemler dizisini bulma işidir. Bu işi yapan modele ise planlayıcı denilmektedir.

2.2 Planlama Problemi Modeli ve Dilleri

2.2.1 Planlama problemi

Planlayıcılar, algoritmalarının planlama işini yapabilmeleri için söz konusu çözecekleri problemin mantıksal yapıdaki bir modeline ihtiyaç duyarlar. Bu model, ortam temsili ve eylem temsillerinden oluşmuş olup gerçek dünya probleminin soyutlanmış halidir. Ortam temsili gerçek ortamda işlerden etkilenen ve problemle ilgili tüm varlıkları ve onlara ilişkin durumları soyutlarken, eylem (operatör) temsilleri ise bu etkilerin hangi koşullar altında başlatılabileceği ve söz konusu varlık ve durum değişlenleri üzerinde ne sonuçlar doğuracağını modeller.

Planlayıcılar, mantıksal kalıpların (*literal*) birleşiminden oluşan bir ilk ortam durumundan hedef ortam durumuna ulaşmaya çalışır. Ortam modelindeki bu kalıplar için *birinci-dereceden mantık* (firtst-order logic) veya *önermeli mantık* (propositional logic) kullanılabilir. *Kapalı-ortam durumu varsayımı* ile bahsi geçmeyen kalıp ve deyimler yanlış sayılırken *açık-ortam durumu varsayımında* ise bahsi geçmeyen kalıplar bilinmiyor demektir. Gerçek dünyadaki eylemlerin soyutlanmış hali olan

operatörler planlama problemi içinde önkoşul ve etki kümeleri ile temsil edilirler. Bir operatörün bir ortam durumunda uygululanbilir olması için önkoşul kümesindeki ifadelerin sağlanması gerekmektedir. Operatör ilgili ortama uygulandığında ortamı etki kümesindeki değişiklikler ile değiştirir. Çizelge 2.1'de kargo probleminin operatör temsili gösterilmektedir.

Çizelge 2.1 : Kargo problemi operatör temsili örneği.

Operatör	Önkoşul	Ekti
$load(kargo, uçak, havaalanı)$	$at(kargo, havaalanı)$ $at(uçak, havaalanı)$	$in(kargo, uçak)$ ¬at(kargo, $havaalanı$
$unload(kargo, uçak, havaalan$	$in(kargo, uçak)$ $at(uçak, havaalanı)$	$\neg in(kargo, uçak)$ at(kargo, $havaalanı$)
$fly(uçak, from, to)$	$at(uçak\ from)$	$\neg at(uçak\ from)$ $at(uçak\ to)$

Çizelgeden de görüleceği üzere ortamda önkoşul ve etkileriyle 3 türlü operatör bulunmaktadır. Örneğin *load* isimli operatör kargo ve uçağın aynı havalimanında olmasını önkoşul olarak belirtirken etki olarak kargonun uçağın içine alındığını göstermektedir.

Çizelge 2.2 : Kargo problemi için örnek başlangıç ve hedef durumu ve plan çıktısı.

Başlangıç Durumu	Hedef Durumu	Plan Çıktısı
$at(kargo1, ESB)$ $at(kargo2, ANT)$ $at(uçak1, IST)$	$at(kargo2, ESB)$ $at(kargo1, IST\ )$	$fly(uçak1, IST, ANT)$ $load(kargo2, uçak1, ANT)$ $fly(uçak1, ANT, ESB)$ $unload(kargo2, uçak1, ESB)$ $load(kargo1, uçak1, ESB)$ $fly(uçak1, ESB, IST)$ $unload(kargo1, ESB, IST)$

Çizelge 2.2' de kargo problemine ait örnek bir başlanıç ve hedef durumları ve buna ilişkin bir plan çıktısı verilmiştir.

2.2.2 Planlama dilleri

Planlama problemlerinin ifade edilebilmesi için, çeşitli planlama dilleri kullanılmıştır. STRIPS bu diller arasında en eski olan dil olup olumlu önermelerin birleşimi şeklinde oluşturulan önkoşul ve etkilerden oluşan operatörlere ve tipleri olmayan nesneleri içeren bir dildir. Bu dilde var olan kısıtlarından dolayı sınırlı bir problem uzayı temsil edilebilmektedir. ADL dili ise daha genişletilmiş bir dil olup STRIPS diline ek nesne tipleri, negatif önkoşullar, denklik ve eşitsizlik koşulları,

mantıksal veya (∨), her (∀) vardır (∃) ise operatörleri gibi özellikleri barındırmaktadır. PDDL dili planlama dillerinin bir standartta toplamış ve zaman ile alakalı daha sonraki uzantıları ile zamansal problemlerin ifade edilebilirliğine genişlik katan bir dildir. Bu dillere ilişkin bir karşılaştırma Çizelge 2.3'te gösterilmektedir.

Çizelge 2.3 : STRIPS, ADL ve PDDL dillerinin karşılaştırılması (Russell ve Norvig, 2003; Kalisch ve König, 2005).

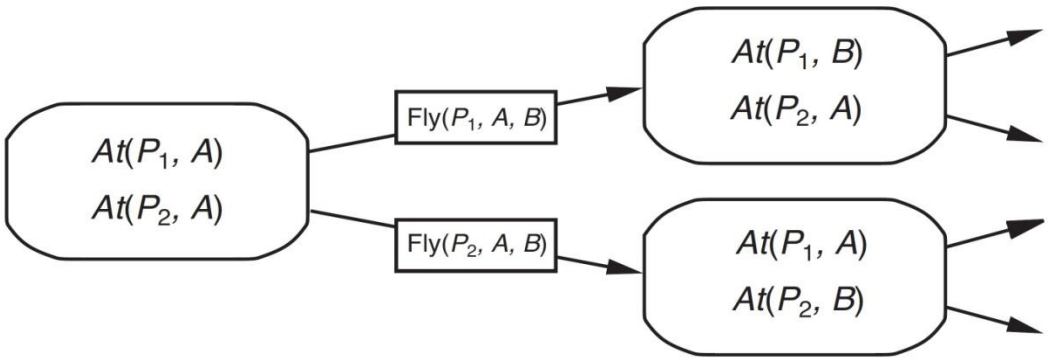
STRIPS	**ADL**	**PDDL**
Kapalı-ortam durumu varsayımı: Bahsedilmeyen önermeler yanlış kabul edilir.	Açık-ortam durumu varsayımı: Bahsedilmeyen önermeler bilinemez kabul edilir.	Kapalı-ortam durumu varsayımı: Bahsedilmeyen önermeler yanlış kabul edilir.
Ortam durumu için sadece olumlu önermeler kullanılır. $P \wedge$ Q	Ortam durumu için olumlu ve olumsuz önermeler birlikte kullanılabilir. $P \wedge \neg$Q	Ortam durumu için olumlu ve olumsuz önermeler birlikte kullanılabilir. $and(P\ not(Q))$
Hedef temsilinde sadece temel önermeler vardır. $Zengin \wedge$ Ünlü	Hedefler temsilinde *her* ve *vardır* operatörleri kullanılabilir. $\exists x\ At(P,x) \wedge At(Q,x)$ ifadesi Q ve P'nin aynı yerde olması hedefidir.	Hedefler temsilinde *her* ve *vardır* operatörleri kullanılabilir. $\exists x\ At(P,x) \wedge At(Q,x)$ ifadesi Q ve P'nin aynı yerde olması hedefidir.
Hedefler mantıksal kesişim şeklindedir. $Zengin \wedge$ Ünlü	Hedefler mantıksal kesişim ve birleşim olarak ifade edilebilirler. $\neg Fakir \wedge$ (Ünlü $\vee\ Zeki$)	Hedefler mantıksal kesişim ve birleşim olarak ifade edilebilirler. $and(not(Fakir)\ or($Ünlü $Zeki))$
Operatör etkileri sadece mantıksal kesişimdir.	Operatörde koşullu etkiler kullanılabilir.	Operatörde koşullu etkiler kullanılabilir.
$P \wedge \neg$Q etkisi P' yi ekle Q'yu çıkar anlamına gelir.	$P \wedge \neg$Q etkisi $\neg$Q ve P' yi ekle $\neg P$ ve Q'yu çıkar anlamına gelir.	$and(P\ not($Q$))$ etkisi $\neg$Q ve P' yi ekle $\neg P$ ve Q'yu çıkar anlamına gelir.
Eşitlik ve eşitsizlik ifadelerini desteklemez.	Eşitlik ve eşitsizlik ifadelerini destekler.	Eşitlik ve eşitsizlik ifadelerini destekler.
Tipleri desteklemez.	Tipleri destekler.	Tipleri destekler.

2.3 Arama Uzayı ile Planlama Yöntemleri

Arama uzayı ile planlama, plan aramada ilk akla gelen yöntemdir. İleri durum uzayı arama ve geri durum uzayı arama olarak iki farklı yaklaşım bulunmaktadır.

2.3.1 İleri durum uzayı arama

İleri durum uzayı arama yöntemi ile planlamada (Progressesion Planning) problemin ilk ortam durumundan başlanarak belirli arama algoritmaları (derinlik önce, en iyi önce, A* vb.) kullanılarak hedef durumu ulaşılmaya çalışılır. Arama ağacındaki her bir düğüm bir ortam durumuna karşılık gelir. Önkoşulları sağlanan operatörlerin bir ortam durumuna uygulanması ile yeni ortam durumları yaratılmış olur. Şekil 3.1'de ileri durum uzayı arama yöntemi gösterilmektedir.

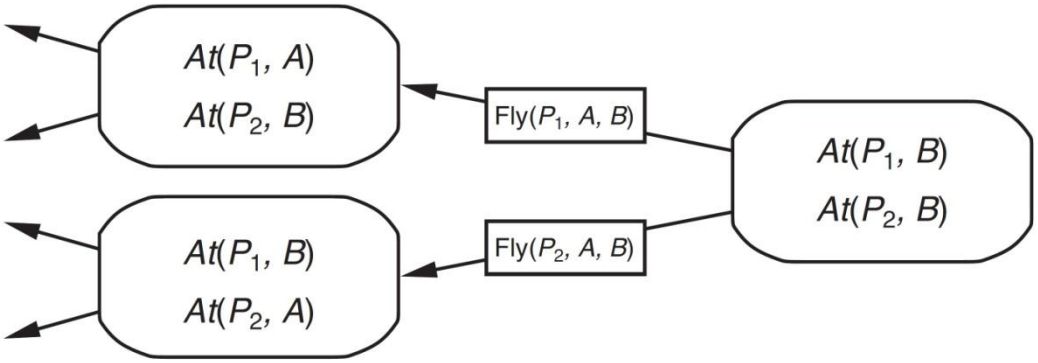


Şekil 2.1 : İleri durum uzayı arama yöntemi (Russell ve Norvig, 2003).

İleri durum uzayı arama yöntemi her ne kadar bir en iyiyi garantileyen (*complete*) bir ağaç arama algoritması kullanıldığında planlama açısından da en iyiyi garantileyen olsa da *ilgisiz eylemler problemi* (*irrelevant action problem*) diye adlandırılan hedefe yönelik olmayan eylemlerin arama ağacında yer alması anlamına gelen problem yüzünden karmaşıklığı çoktur (Russell ve Norvig, 2003). Bu karmaşıklık ortamda bulunan operatörlerin sayısı ve ortam durumunu temsil eden değişkenlerin çokluğu sonucu arama ağacı dallanma faktörünün artmasından ve üretilen düğümlerin benzerliklerin azalmasından kaynaklanmaktadır. Bu durum basit problemlerin çözümlerini dahi zorlaştırmaktadır. Bu yüzden ileri durum uzayı aramasında iyi sezgisel fonksiyonların kullanılmasını gerektirir.

2.3.2 Geri durum uzayı arama

Geri durum uzayı arama yöntemi ile planlamada (Regression Planning), arama hedeften başlanarak ters yönde ilerler. Arama ağcında yeni durumları temsil eden düğümler var olan durumun oluşmasına etki eden eylemlerin ön koşullarının eklenmesiyle birer alt hedef olarak oluşturulur. Arama verilen ilk durumun kapsayack alt hedef kümesi oluşturulana kadar devam eder.



Şekil 2.2 : Geri durum uzayı arama yöntemi (Russell ve Norvig, 2003).

İleri durum uzayı arama yönteminden farklı olarak bu yöntemde seçilen eylemlerin hedefe yönelik olmasından dolayı *alakalı eylemler* ile ağaç düğümleri oluşturulur ve böylece dallanma faktörü azalmış olur. Hedefler açık olarak bir liste olarak değil de bir kısıt kümesi olarak verildiğinde bu yöntemin uygulanması zorlaşır (Russell ve Norvig, 2003).

2.4 Kısmi Sıralı Planlama Yöntemi

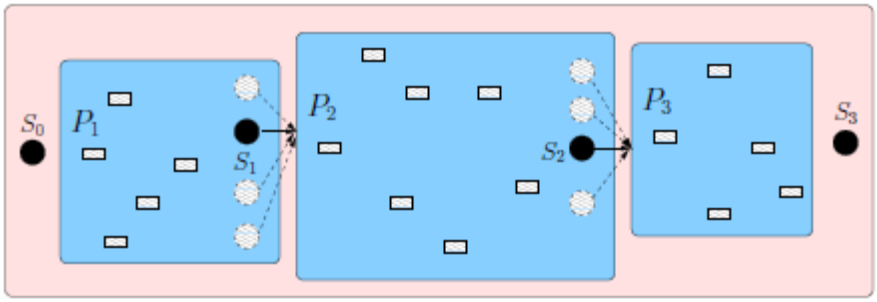
Durum uzayı planlama yöntemleri ile elde edilen plan tüm eylemleri sıralanmış *tam sıralı* plandır. Bundan farklı olarak kısmi sıralı planlamada en az bağımlılık (least commitment) stratejisi denen arama esnasında eylemelerin mümkün olduğu kadar plandaki sırasını belirlemeden (veya erteleyerek) plana eklenmesi stratejisi uygulanarak plan oluşturulur. Arama hedef durumundan başlar ve geri yönde devam eder. Eklenen her eylem bir önceki durum veya durumlar ile nedensel ilişki bağlantıları oluşturur. Bir eylem etkileri yüzünden planda bulunan diğer eylemler ile çatışma oluşturuyorsa plana eylemler arası sırlama kısıtları eklenir. Aramaya açık önkoşul ve çatışma kalmayana kadar devam edilir.

2.5 Planlama Grafı

Planlama grafı veri yapısı (Blum ve Furst, 1995) arama uzayı yönteminde olaşan grafa göre daha küçük bir arama grafı üzerinden yapılan ulaşılabilirlik analizi bilgisi de göz önünde bulundurarak uygunsuz eylemlerin erken elenmesini sağlamaktır. Temelde planlama grafı başlangıç durumuna ilki olan seviyelerden oluşur. Bir zaman dilimine karşılık gelen her seviye o zaman diliminde doğru olan önermeler kümesi ve bu önermeler ile ön koşulları sağlanmış eylemler kümesinden oluşur. Graf aynı zamanda karşılıklı dışlama durumları içerir. Karşılıklı dışlama durumu aynı seviyedeki iki eylemin ön koşullarının veya etkilerinin birbirinin negatifi olan önerme veya önermeler içermesi durumda veya aynı seviyedeki iki önermenin birbirinin zıddı olması durumunda oluşan ve planlama aşamasında kullanılan bilgi kümesidir. Planlama grafı her ne kadar eylemler arası tüm negatif girişimleri tutamadığından bir önermenin ortaya çıkış seviyesinde iyimser olsa da gelişkin sezgisel fonksiyonların bulunması için bilgi sağlayacıdır.

2.6 Ara-hedef Bölümleme ve Artımlı Planlama

Plan probleminde ve temsilindeki karışıklık arttıkça plan arama uzayı genişler. Genişleyen arama uzayında hedefe ulaşmak için gereken zaman ve uzay liner olmayan bir şekilde arttığından karmaşık planların çözümlerinin bulunması zorlaşmaktadır. Bu durumda arama uzayındaki karmaşıklığı azaltmak için eniyileme kıstasını kaybetme maliyetini göz önüne alıp arama uzayında belirli düğümler seçerek bu düğümleri ayrı ayrı hedefler olarak tanımlanmasına ara-hedef bölümleme (*subgoal partitioning*) yöntemi denilir.



Şekil 2.3 : Ara-hedef bölümleme (Chen, Wah ve Hsu, 2006).

Şekil 2.3' ten de görüldüğü üzere S_0 ortam durumundan S_3 ortam durumuna ulaşmak için S_1 ve S_2 olmak üzere iki ara hedef belirlenmiştir. Bu şekilde ardı sıra yapılacak

yerel eniyi (*local optimal*) planlar üreten artımlı planlama (*incremental planning*) ile ana hedefe ulaşılabilir. Chen, Wah ve Hsu (2006) ara hedef bölümleme ilgili yaptıkları çalışmalarında verilen bir problem için ara hedefler bularak ara problemler üreten bir sitem geliştirmişlerdir.

3. ROBOTLARLA PLANLAMA

3.1 Giriş

Gerçek dünyada verilen tüm hedeflerin başarıyla yerine getirilmesi çoklurobot sistemlerinin en öncelikli hedefidir. Bu problem önceleri iş tahsis etme ve iş seçme problemi olarak ele alınmıştır (Gerkey ve Mataric, 2002; Dias ve Stentz, 2002; Sariel, Balch ve Erdogan 2007). Bu yaklaşımın verimini artırmak amacıyla çeşitli eniyileme kriterleri düşünülmesine rağmen otonom eylem planlama teknikleri kaynak kısıtı altında karmaşık eylem dizisi ile ulaşılabilen hedefler için önemli hale gelmektedir.

Klasik planlama ile yapılan hedefe giden eylemlerin sıralanması, gerçek dünya problemlerinin çözümü için yeterli değildir. Eylemler arasında paylaşılan kaynakların olduğu durumlarda bu eylemlerin ne zaman yürütüleceğinin de bilinmesi gerekmektedir. Klasik planlama yöntemleri statik, değişmeyen, tümüyle gözlemlenebilen ve yapılan girdiye göre çıktının her zaman tahmin edilebilceği ortamları varsaymaktadır. Fakat gerçek dünyada dinamik ve değişken bir ortam vardır. Yine klasik planlamada ortamın kusursuz bir şekilde modellenebildiği varsayımı altında çalışılmıştır. Gerçek dünyada robotlar tüm ortam bilgisine ve ortamın doğru modeline sahip değildirler ve üretilen planlar ortam dinamikliği yüzünden anlamsız hale gelebilir. Belirlenimci olamayan ve karmaşıklığı yüksek olan gerçek dünyadaki problemler için klasik planlama yöntemleri tek başlarına yeterli olmamaktadır.

3.2 Planlamada Zaman ve Kaynak

Gerçek dünyada robotlar ortamda paylaştıkları kısıtlı kaynaklar üzerinde eylemlerini yürütürlerken birbirleriyle uyumlu olmaları gerekmektedir. Bu durum ise planlamaya zaman ve kaynak parametrelerinin dahil edilmesini gerektirmektedir.

Planlama eylemlerinin ortamda bulunan paylaşılan kısıtlı kaynaklara, eylemlerin kaynakları kullandıkları süreleri de göz önünde bulundurularak atanmasına çizelgeleme (*scheduling*) adı verilir.

Çizelgeleme, çizelgeleme probleminin kısıt sağmala problemi (CSP) olarak temsil edilerek oluşturulmuş bir planın eylemleri üzerinde planlamadan sonra yapılabileceği gibi planlama ile eşzamanlı olarak da yapılabilmektedir. Çizelgelemenin planlama ile eş zamanlı olarak yapılması çizelgelenmiş planın oluşturulma süresini uzatmasına rağmen çözüm kalitesini artırmaktadır.

Çizelgelemenin planlamayla eş zamanlı yapılmadığı fakat aynı sisteme entegre edildiği sistemler mevcuttur. Halsey, Long ve Fox (2004) çalışmalarında CRIKEY adlı bu sistemi kullanan bir planlayıcıyı sunmuşlardır. PDDL 2.1 dilinde temsil edilen zamansal bir problemi zaman boyutundan arındırıp basit PDDL diline dönüştürerek klasik bir planlayıcıdan *tam sıralı plan* üreten bu sistem, daha sonra bunu *kısmi sıralı bir plana* çevirir. Bu aşamaya kadar sadece çizelgenebilirlik kontrolü yapılmış kısmi sıralı bir plan elde edilmiş olunur. Kısmi sıralı plan daha sonra orjinal problemin zamansal boyunu da kullanılarak zaman kısıtlı ağ (STN) şeklinde ifade edilerek çizelgelenmiş olur. Bu yöntem çizelgeleme işlemleri için planlama safhasında arama uzayını genişletmediğinden performanslı olmasına rağmen zamansal eniyileme gerçeklenmemiş olur. Bir kısmi sıralı planlanlama yapan planlayıcının çizelgeleme de yapabilecek şekilde geliştirildiği IxTET (Ghallab ve Laruelle, 1994) planlayıcısı gibi sistemlerde de benzer olarak planlayıcı arama uzayındaki düğümlerde kısmi durum bilgisi içermesi sonucu bir eylemin çizelgede kendinden önceki yürütülmüş tüm eylemler bilinmemektedir. Bu durum da karmaşık gerçek dünya ortamlarının çözüme ulaştırabilmek için gerekli sezgisel fonksiyonlar tanımlanamayacak olması bu tür yaklaşımların temel eksikliklerindendir.

Çizelgelemenin planlama ile eş zamanlı yapılabilmesini sağlayan sistemlere örnek olarak sınırlı karar adımlı planlama (DEP) yöntemi (Bacchus and Ady 2001) verilebilir. İleri durum arama yönteminde arama ağacındaki düğümlerin içlerinde zaman bilgilerinin tutulmasıyla zamansal planlama yapılır. Bu yöntemde düğümlerdeki zaman bilgisi planlama temsilindeki bir operatörün düğüme uygulanması ile güncellenmez. Düğümlerdeki zaman bilgisinin güncellenmesi süreli bir eylem uygulandıktan sonra elde edilen alt ağaçta uygulanacak o eylemin süresinin

bittiğini belirten planlayıcı tarafında tanımlı özel bir eylem ile olmaktadır. Bu şekilde her eylem uygulandıktan sonraki elde edilen düğümün kök olduğu alt arama ağacında çeşitli düğümlerde özel zaman ilerletici eylem çalışarak zamanın güncellendiği diğer düğümler yaratılmış olur.

Bu yöntem arama ağacı düğümlerinde hem zamansal hem de olgusal tüm bilgileri içerdiğinden karmaşık zamansal senaryolar içeren problemlere çözüm üretebilir. Zaman bilgisinin de düğümlerde tutulmasının bir maliyeti olarak bu yöntemi kullanan bir planlayıcının arama uzayı oldukça geniş olmaktadır. Bu dezavantajı telafi etmek için sezgisel fonksiyonlar kullanılabilir. Planlama ve problem temsiline göre etkili sezgisel fonksiyonların geliştirilmesi sonucu en iyi-öncelikli ve A* arama ile hedef doğrultulu armalarla arama uzayı daraltılarak verimli sonuçlar alınabilmektedir.

3.2.1 TLPlan planlayıcısı

Bu yöntemi kullanan planlayıcılardan biri TLPlan (Bacchus and Ady 2001) planlayıcısıdır. TLPlan, eylemlerin paralel olarak yürütülmesini sağlayabilen, kullanıcıya çeşitli arama algoritmalarıyla etkin sezgisel fonksiyonlar tanımlamasına izin veren ve çizelgeleme ile planlamayı aynı anda yapılmasına olanak tanıyan bir planlayıcıdır. Süreli bir eylem arama ağacındaki önkoşulunu sağlayan bir düğüme uygulandığında oluşan ortam durumu düğümüne sadece ilk etkileri yansır ve bu eylemin bittiğinde oluşması gereken son etkiler yeni düğümün *olay listesine* eklenir ve yeni düğüm yaratılan düğümler listesine eklenir. Yeni bir düğüm bu listeden açılmak için seçildiğinde uygulanabilen eylemler arasında son etkileri yeni yaratacağı düğüme ekleyerek yeni yaratılan düğümün zamanını da güncelleyen *event* adlı özel eylem de vardır. Son etkileri henüz uygulanmamış süreli bir eylem tarafında oluşturulmuş listedeki bir düğüme *event* eylemi dışındaki bir eylem uygulanmasıyla paralellik sağlanmış olunur.

3.3 Planlamada Ortam Bilgisi Belirsizliği

Gerçek dünyada robotlar planlarındaki eylemlerini yürütürlerken algılayıcıları vasıtasıyla çevrelerinden algıladıkları bilgileri kullanmaları bu bilgiler doğrultusunda eğer gerekliyse planlarını değiştirmeleri, yenilemeleri ve beklenmeyen olaylara karşı

koyabilmeleri gereklidir. Gerçek dünyada var olan bu belirsizliğe karşı koşullu planlama, olasılıksal planlama, yürütme gözlemleme ve yeniden planlama başlıklarında toplanabilen çeşitli planlama yöntemleri geliştirilmiştir.

3.3.1 Koşullu planlama

Gerçek dünyada var olan belirsizliğe karşı uygulanan yöntemlerden biri koşullu planlama (*conditional plannning, contingency planning*) diye adlandırılan bu yöntemde planlama tanım kümesinde ortaya çıkacak belirli koşullara karşı planda belirli dallar yaratır. Plan içinde bulunan koşulları simgeleyen bu dallar yürütme esnasında yürütücünün algılayıcıları vasıtasıyla önce algılanır, algı sonucuna göre yürütülmesi gereken eylem seçilir. Plan içine dallar yaratma, operatörlerinin etki kısımlarına koşul ifadeleri eklenerek bir operatörün belirli koşullar altında oluşturacak farklı etki kümlerine sahip olmasıyla gerçekleştirilebilir. Koşullu planlar artımlı bir şekilde oluşturulabilir. Dearden ve diğ., (2003) robotların harcadıkları kaynaklardaki belirsizliğe karşı dayanıklı koşullu planlar üreten kurdukları sistem kurmuşlardır. Bu sistemde üretilen ilk plana artımlı bir şekilde ek dallar eklenmektedir. Koşullu planlar üretmenin bir diğer yolu ise olasılıksal planlamadır.

3.3.1.1 Olasılıksal planlama

Olasılıksal planlamada önceden bilinen veya zamanla elde edilen olasılık bilgilerinin kullanarak ortamda hareket eden veya edecek etmenlere hareket planı oluşturmaya dayanır. Bu yaklaşımdaki temel mantık klasik planlama temeline olasılıksal eklentiler yaparak genişletmektir

Olasılıksal eklentiler yaparak klasik planlamanın yetkinliğini kullanan sistemler bu alanda üzerinde çalışılmış diğer bir yöntemdir. Temelinde gerçek dünyadaki belirsizlikleri, yürütücülerin bilgi dağarcığını durum uzayı ve eylemler üzerinde tanımlı olan olasılıksal bir dağılıma çevirdiği fikri yatar.

Olasılıksal planlama yapabilen bir planlayıcı, koşullu planlamadakine benzer olarak planlama operatörlerinin ayrıştırılmış etki kümelerinde tanımlı belirli olasılık etiketleriyle (*probabilistic outcomes*) kullanarak planlama aşamasında başarı olasılığının en iyilemesini yapar. Bu planlayıcılara örnek olarak operatörlerin olasılıksal etkilerini graf planlama alt yapısına entegre ederek kullanan eniyi koşullu

planlar üreten PGraphplan (Blum ve Langford, 1999), eniyi olmayan fakat hedefe giden en yüksek başarım olasılıklı koşulsuz plan üreten Tgraphplan (Blum ve Langford, 1999) ve geri durum uzayı arama ve karşılıklı dışlama çıkarsamasını kullanarak yürütme eşzamanlılığını da göz önüne alabilen Paragraph (Little ve Thiebaux, 2006) planlayıcıları verilebilir.

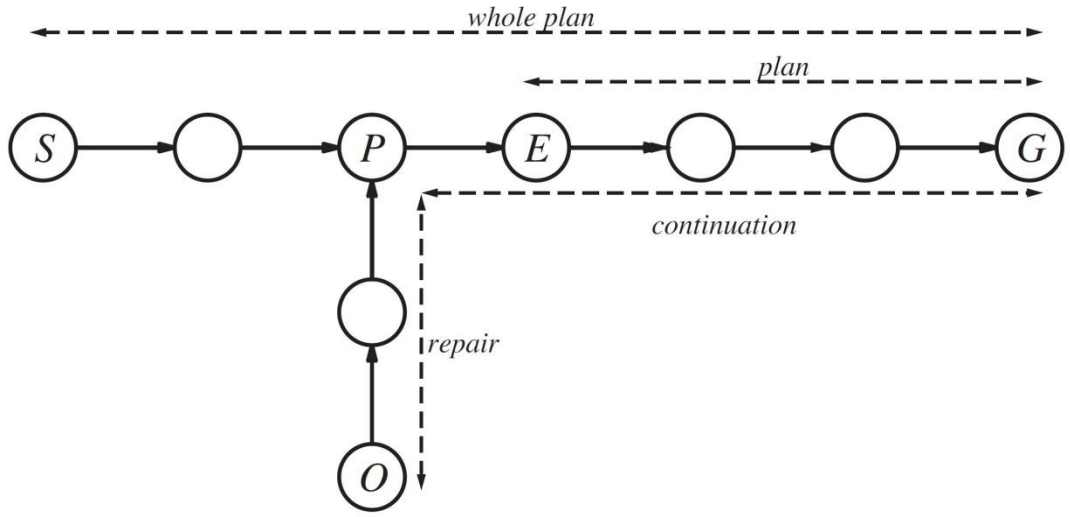
3.3.2 Yürütme gözlemleme ve yeniden planlama

Kuramsal olarak planlanmış bir planın gerçek dünyada yürütülmesi ile eşzamanlı olarak yürütücü veya yürütücüler tarafından doğruluğunun ve geçerliliğinin izlenmesi yürütme gözlemleme (*execution monitoring*) diye adlandırılır. Yürütücüler beklenmeyen bir hata veya olay ile karşılaşıldığında eğer gerekli ise plan güncellemeleri yapılmasıyla dinamik hata ve beklenmeyen olaylara karşı koyabilirler.

Yürütme gözlemleme iki farklı şekilde yapılabilir. Yürütücülerin içinde bulundukları ortam durumunu gözlemleyerek, o ortam durumunun planda sıradaki yürütücekleri eylemlerin önkoşullarını sağlayıp sağlamdıklarını test etmelerine *eylem gözlemleme (action monitoring)*, ortam durumunun sıradaki eylem ile birlikte kalan tüm planı yürütebilmeyi sağlayıp sağlamadığının kontrolüne ise *plan gözlemleme* (*plan monitoring*) adı verilir. Örnek olarak bir planda ardısıra gelen ve tüketilebilir bir kaynağı eşit miktarlarda tüketen iki eylem olduğunu düşünülebilir. Ortamda bulunan o kaynak miktarı iki eylemin kullanacağı toplam kaynak miktarından az fakat tek bir eylemin için gereken kaynak miktarından fazla olsun. Bu durumda plandaki ilk eylem yürütülmeden önce yapılacak eylem gözlemleme sadece ilk eylem için kaynağın yetip yetmemesini göz önünde bulundurarak planın yürütülmesi için onay verirken plan gözlemleme ile birinci eylemin yürütülmesinden sonra kalacak olan kaynak miktarının ikinci eylem için yetmeyeceği anlaşılacağından plan yürütülmesi erken safhada durdurulabilecektir.

Beklenmeyen olay ve hatalar karşısında plan güncellemesi için iki yol kullanılmaktadır. Bunlardan ilki *plan onarma* olarak adlandırılan hata öncesi var olan planı yürütülmeyen kısmını koruyarak ona mümkün olan en uygun (en az maliyetli, en az değişiklik gerektiren, en az eylem içeren, gerçekleşmesi en olası v.b.) noktadan bağlantı yapacak şekilde bir ara plan üretmektir. Şekil 3.1'de de görüldüğü üzere

yürütme öncesi etmen ***S***' den ***G***' ye uzanan tüm planı görülmektedir. Bu plan bir E noktasına kadar yürütülür. Kalan plan yürütlümeden önce eylem gözetleme denen safhada ilgili eylemin önkoşullarını test ederken etmen E noktası yerine bir O noktasında olduğunun farkına varır. Etmen, plan onarma modülü ile onarma (*repair)* bölümünü, planın yürütülmemiş kalan kısmı olan sürdürme (*continuation)* bölümü ile birleştirir.



Şekil 3.1 : Plan onarımı (Russell ve Norvig, 2003).

Plan güncellemesi için kullanılan diğer bir yol ise hata durumundan hedefe giden tüm planı baştan üretmektir. Brenner ve Nebel (2009) ortam bilgisi eksikliği durumlarında her defasında yeniden plan yapan çok etmenli bir sistem geliştirmişlerdir. Bu sistemde diğer etmenlere olan plandaki bağımlılıklar (*commitments*) dikkate alınması gerekir ve yeni üretilen plan ile mevcut eski planın mümkün olduğunca kullanılması sağlanır. Cushing, Benton ve Kambhampati (2008) gerçek dünya söz konusu olduğu zaman, robotlar yeni bir plan üretebilmek için yeterli kaynak ve kapasiteye sahiplerse, planı baştan üretmenin, var olan planı onarmaya göre daha etkin ve daha çok kullanılan bir yöntem olduğunu belirtmişlerdir.

3.4 Planlamada Temsil Bilgisi Yetersizliği

Karmaşık olan gerçek dünyada robotlar ortamın doğru modeline sahip olmayabilirler. Planlama temsili tasarımcıları, gerçek dünyanın planlama temsilindeki modeline soyutlanması aşamasında, kasıtlı veya kasıtsız olarak ortamı

tüm detaylarıyla modellemez veya modelleyemez. Göbelbecker ve diğ. (2010), plan temsilinin eksik veya hatalı olduğu durumlar olabileceğini belirtmişlerdir ve çalışmlarında plan üretilememesi ile sonuçlanan durumların nedenlerini bildirebilecek bir yöntem geliştirmişlerdir.

Ortamın doğrudan modellenemediği veya hatalı modellendiği durumlar için ortam temsili bilgisini öğrenme yoluyla elde eden çalışmalar yapılmıştır. Gill (1994), çalışmasında robotların hatalı veya eksik planlama opertörlerini robotların gerçek dünydaki deneylerinde elde ettikleri bilgilerden çıkarsama yapmasını mümkün kılarak öğrenmelerini sağlamıştır. Wang (1995) ise bu yönteme başka robotları gözlemleyerek öğrenme yaklaşımını eklemiştir. Amir ve Chang (2008) ise diğer çalışmalardaki *tümüyle gözlemlenebilir* ortam varsayımını kaldırarak *kısmi gözlemlenebilir* ortamlarda ortam modeli öğrenilebileceğini göstermişlerdir. Öğrenilerek yapılan bu sistemlerin hepsi belirli önceden belirli öğrenme süreci gerektirmekte dinamik olarak daha önceden karşılaşılmamış olaylara cevap verememektedirler.

Gerçek dünyadaki dinamik olayların planın yürütülmesi sırasında hata oluşturması muhtemeldir. Bu hataların tespitinin yapılabildiği (Witteveen ve diğ., 2005; de Jonge, Roos ve Witteveen 2009) ve tespiti yapılmış hataların çözüldüğü sistemler (van der Krogt ve de Weerdt, 2005 ; Micalizio 2009) olmasına karşın bu sistemler, ortam temsilinin yetersizliği durumunda neler yapılabilceğinin cevabını verememektedirler.

4. PROBLEM TANIMI

4.1 Giriş

Bu bölümde tezin araştıma konusu olan problemin tanımı sunulmaktadır. Diğer bölümlere referans niteliğinde olan bu bölüm ayrıca Bölüm 5 ve Bölüm 6'da bulunan algoritmalarda kullanılacak olan ifadelerin tanımlandığı bölümü oluşturur.

4.2 Dinamik Zamansal Planlama Problemi

4.2.1 Tanımlar

Tanım 1. (Planlama tanım kümesi). Planlama tanım kümesi $\Delta = (\mathcal{T}, \mathcal{C}, \mathcal{P}, \mathcal{O})$, $\mathcal{T}$ tipler kümesi, $\mathcal{C}$ bir değişmezler kümesi, $\mathcal{P}$ yüklemler kümesi, $\mathcal{O}$ ise robotlar tarafından yürütülen gerçek dünya eylemlerini simgeleyen operatörler kümesi olmak üzere bir çokuzludur.

Tanım 2. (Planlama problemi). Planlama problemi P_r, I ve G sırasıyla başlangıç ve hedef ortam durumları olmak üzere $P_r = (\Delta, I, G)$ şeklinde modellenir. Her bir ortam durumu kaynak ve robot ($r_i \in R$) temsillerini de içeren olguların kümesini temsil eder.

Tanım 3. (Operatör). $o \in \mathcal{O}$ operatörü parametre kümesi $param(o)$, ön koşul kümesi $pre(o)$ ve ekleme/çıkarma etkileri kümeleri $add(o)/del(o)$ ile temsil edilir. o operatörü şimdiki ortam durumuna (S) ancak ve ancak $pre(o)$ koşulu sağlanıyorsa uygulanabilirdir.

Tanım 4. (Eylem). Her bir a eylemi gerçek dünyada robotlar tarafından yürütebilen bir eyleme karşılık düşecek şekilde, argüman seti $arg(a)$, yürütmesi için atanmış robot seti ve oluşturulmuş zamansal plandaki $t_s(a)$ başlangıç zamanı ile temsil edilir. Her bir a eylemi, $t_s(a)$ ile son etkilerinin sonuç ortam durumda görüldüğü arasında geçen toplam süreyi veren $dur(a)$ niteliğine sahiptir.

Tanım 5. (Zamansal plan). P_r problemi çözümü olan P zamansal planı, $makespan(P)$ optimizasyona göre sağlayan ve çizelgelenmiş ve paramtreleri

ilklendirilmiş operatörlerden oluşan ve $operator(a_i) = o_i \in \mathcal{O}$ koşulunu sağlayan eylemler dizisidir.

Tanım 6. (Geçerli plan). Bir zamansal planın her aşamasında kalan kısmının yürütülmeden önce geçerlilik kontrolü simülasyonu olan $valid(P, S, G)$ fonksiyonu şimdiki ortam durumu S' ten hedef ortam durumu G' ye ulaşılıp ulaşılamayacağını test eder. Bir planın g*eçerli* olması için $valid(P, S, G)$ fonksiyonu değeri *true* olmalıdır.

Tanım 7. (Yürütülebilir plan). *Yürütülebilir* plan, mevcut durumdaki kaynak dağılımına göre gerçek dünyada yürütülmesi mümkün olan *geçerli* plandır.

Tanım 8. (Hatalı eylem). Bir P planının yürütülmesi esnasında hata oluşup kesilmek zorunda kalan eylem a_{failed} ile gösterilir.

Tanım 9. (Hata ortam durumu). Hata ortam durumu a_{failed} eyleminin yürütülmesinin durdurulması sonucu oluşan ortam durumudur ve S_{failed} olarak ifade edilir.

Tanım 10. (Yürütülemez plan). Bir P planı yürütme esnasında hata sonucu kesildiğinde S_{failed} ortam durumunda *yürütülemez* hale gelir.

Tanım 11. (Hata sebebi). Hatanın sebebi $c \in S_{failed}$ ile temsil edilir ve Δ' da tanımlı $attr(c)$ nitelikler listesine sahiptir. Listedeki her bir $attr_i$ niteliği, S' de $value(attr_i, S)$ değerine sahiptir. Hata durumu sebebi c'nin, robotun alt katman modülleri tarafından tespit edildiği varsayılmaktadır.

Tanım 12. (Hata sonucu güncellenmiş tanım kümesi). İlgili tanım kümesi güncelleme işlemleri c' yi planlama problemi P_r' ye ekleyerek P_r' elde edilmiş olur. Bu güncellemeye rağmen ve aslında *yürütülebilir* bir plan olmasına karşın planlayıcı *yürütülebilir* bir plan bulamayabilir.

Tanım 13. (Gerekli ikincil tanım kümesi güncellemeri sonucu oluşan tanım kümesi). Hata sonucu, planlayıcının P_r' için *yürütülebilir* bir plan bulamaması nedeniyle tanım kümesinde ek güncellemeler yapılmalıdır. Güncellenmiş problem P_r'' den sonra ikincil güncelleme sonucu oluşan güncellenmiş problem P_r''' ile gösterilir.

Tanım 14. (Geçici eylem hatası). *Geçici* eylem hataları yürütme esnasında oluşan ve eylemin geçici olarak yürütülmesini engelleyen hatalardır. Bazı geçici hatalar yeniden planlama ile çözülebilir ve tekrar uygulanabilir hale getirilebilir.

Tanım 15. (Kalıcı eylem hatası). Kalıcı hataları yürütme esnasında oluşan ve oluşan ve eylemin hiçbir şekilde yürütülmesi mümkün olmayan hatalardır. Bir eylem (operatörünün tüm ilklendirilmiş eylemleri olmak zorunda değil) *kalıcı* hataya uğradığında, hiç bir gelecek planda yer alamaz.

4.2.2 Dinamik zamansal planlama problemi

Bu tezin konusu konusu bir planın yürütülmesi esnasında geçici veya kalıcı eylem hataları oluştuğunda gerekli tanım kümesi güncellemesini uygun bir şekilde (örneğin bilgi tabanına hata sebebini aktararak) yapan modelleri ve yöntemleri sunmaktır. Tanım kümesi güncellemesi, hata yapan eylemlerden başka eylemleri de plana katabilecek, otonom bir şekilde *yürütülebilir* bir plan bulmaya dayanır. Önem arzeden bir nokta, *yürütülebilir* yeni plana dahil olacak bu eylemlerin planlayıcı tarafından gerekli tanım kümesi güncellemesi yapılmadan seçilmesinin mümkün olmamasıdır. Sonuç olarak bu yüzden, $P_r{}'$' de *geçerli* bir plan olmadığı veya bir çok *geçerli* fakat *yürütülemez* bir planın bulunabildiği durumlarda (örneğin $pre(a_{failed}) \subseteq S_{failed}$)) bile $P_r{}''$' de *yürütülebilir* (aynı zamanda *geçerli* de olmaktadır) bir planın bulunması amaçlanmıştır. Diğer bir ifade ile hata durumundan sonra *yürütülemez* olan bir planın, aslında c hata olguları planlama probleminde ifade edilse bile planlayıcı açısından hala *geçerli* bir plan olarak görülebilmesidir.

4.3 Örnek Bir Problem

Motive edici bir örnek olarak ve yukarda verilen problemi betimleyebilmek amacıyla örnek bir senaryo verilebilir. Örnek senaryo bir koridorun bir ucunda duran ve koridorun diğer ucuna taşınması gereken bir nesne ve nesneyi taşıyabilecek bir robot içermektedir. Robot kolu ile *kaldır, indir* gibi nesnenin taşınmasında kullanabileceği, *sürükle* gibi kaldırıp indiremeyeceği nesnelerin konumunu değiştirmeye yarayan ve bir konumdan diğer bir konuma gitmesini sağlayan *git* gibi konum değiştirmesini sağlayan eylemlere sahiptir. Fakat robot nesneyi taşırken ortamın haritasını tam bilmediği ve tüm engellerden haberdar olmadığı için yürütme zamanı hataları ile karşılaşabilir. Hedefe giden yolu taşınamayacak kadar büyük bir engel kapattığı için

git eylemi *geçici* hata verir. Bu hata harita ve yol planı güncellendikten sonra robotun yol planlayıcısı modülü tarafından tespit edilir. Bu noktada sadece engelin varlığının ve bilgilerinin bilgi tabanına eklenmesi ile planlayıcı *yürütülebilir* bir plan bulamaz. Aslında robotun engeli sürüklemesi eylemini içeren y*ürütülebilir* bir plan ile robotun hedefine ulaşması mümkündür. Fakat doğal olarak *sürükle* eylemi *hedefe giden yolu temizle* etkisi gibi bir etkiyi içerecek şekilde tasarlanmadığından planlama esnasında göz önüne alınmaz. *Kalıcı* hata için ise nesneyi *sürükle* operatörünün mevcut kaynaklar ile hiç bir şekilde yürütülememesi örnek olarak verilebilir. Böyle bir durumda planlayıcının, o engel için *sürükle* operatörünün sürükleme eylemini hiç bir gelecek plana dahil etmesi mümkün olmaz.

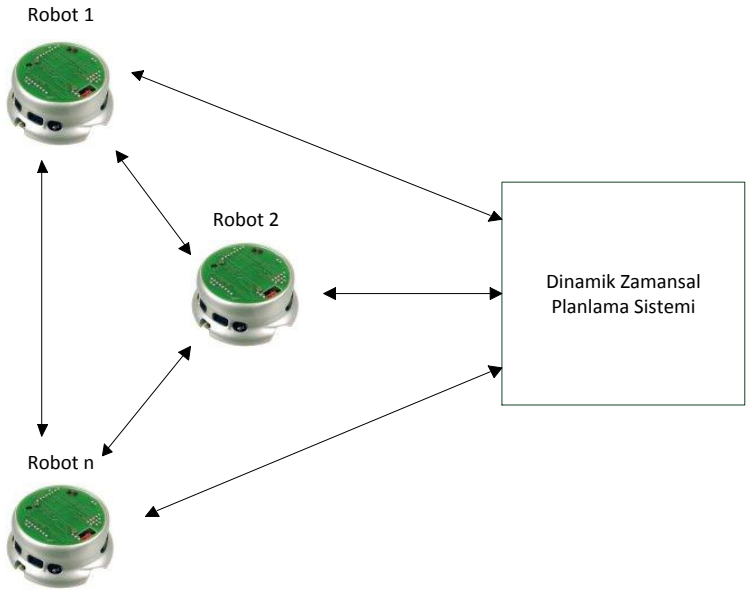
5. DİNAMİK ZAMANSAL PLANLAMA SİSTEMİ

5.1 Giriş

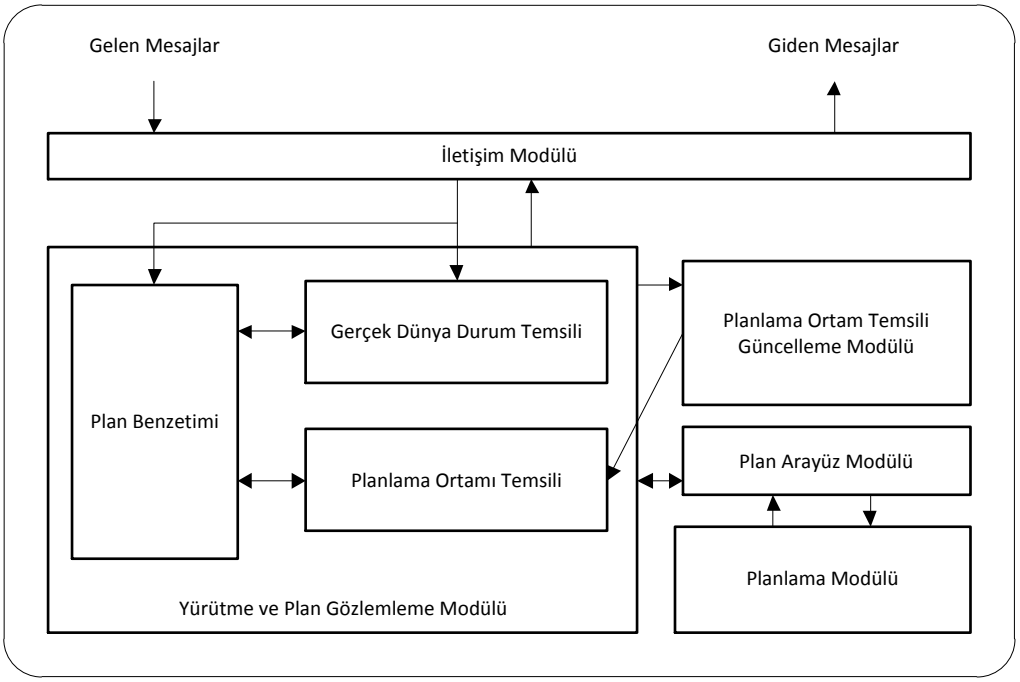
Bu bölümde araştırma konusu olan problemi çözebilmek için model ve yöntemler tanıtılmaktadır. Dinamik zamansal planlama siteminin robotlar ile olan iletişimindeki mesaj yapısından bahsedilecek, genel modeli modüllere ayrılmış haliyle ve durum geçiş diyagramı üzerinde gösterilerek anlatılacak ve son olarak resmi çizilen bu sistem, algoritmik akış açısından ele alınarak detaylandırılacaktır.

5.2 Sistem Modeli

Bu tez kapsamında tanıtılan problemde belirtilen *geçici* eylem hatalarını çözmek için önerilen yöntem, çoklu robot sistemleri içinm bir *dinamik zamansal planlama sistemi* önerilmektedir. Önerilen yöntem, çoklu robot sistemleri için gerekli olan dinamik planlama yordamlarını içermektedir. Sistem ile robotlar arası iletişim Şekil 5.1' de görüldüğü üzere mesajlaşma yoluyla sağlanmaktadır. Bu yapıda tüm robotlar kendi aralarında haberleşebileceği gibi sistemin merkezi şekilde robotlarla iletişim halinde olabildiği varsayılmıştır.



Şekil 5.1 : Dinamik zamansal planlama sistemi.



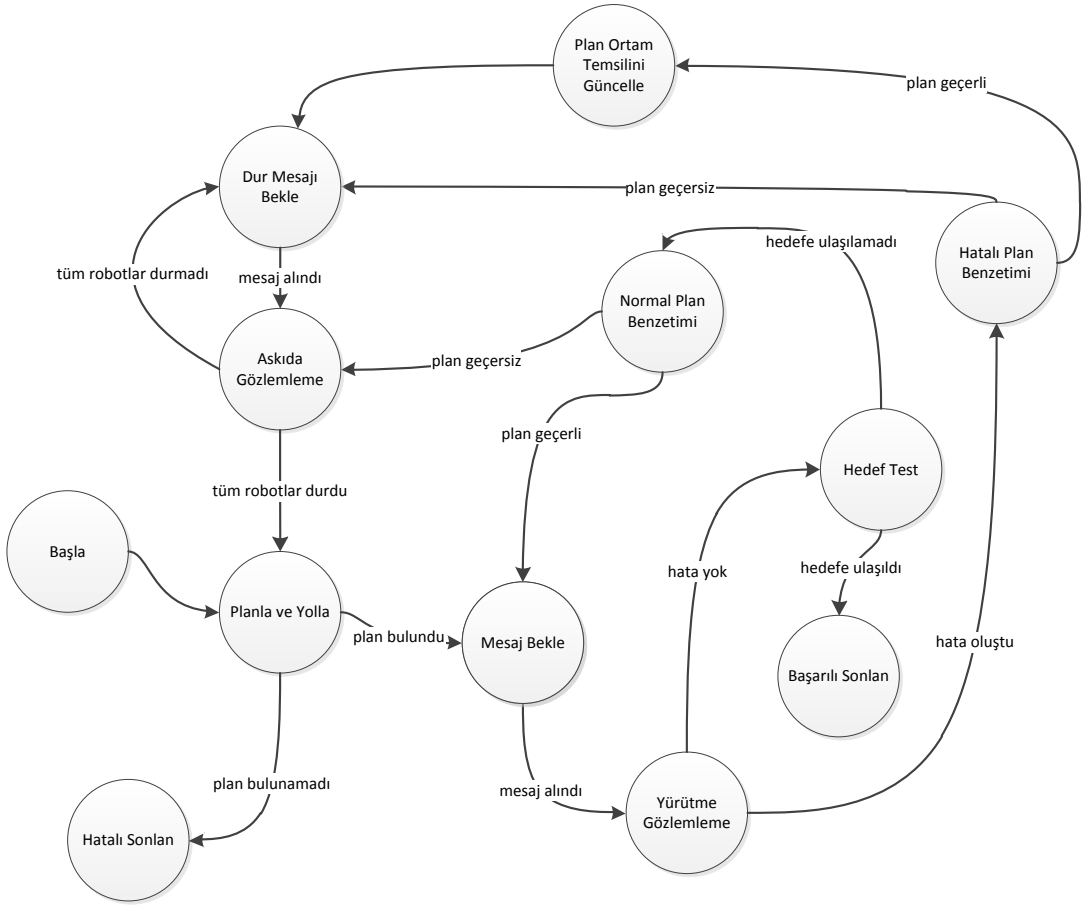
Şekil 5.2 : Dinamik zamansal planlama sistemi bileşenleri.

Şekil 5.2' de de görüdüğü üzere dinamik zamansal planlama sistemi farklı modüllerden oluşmaktadır. Bu modüllerden haberleşme modülü dış çevre ile yürütme ve gözlemleme modülü arasındaki iletişimi gerçekleyen sistem bileşenidir. Yürütme gözlemleme modülü planın gözlendiği, hedefe ulaşılma durumunun ve hataların tespit edildiği ve diğer tüm modüller ile iletişimde olan ana sistem parçasıdır. Planlama modülü bir arayüz üzerinden plan emrini alıp oluşan zamansal plan çıktısını yine bu modüle veren bileşeni oluşturur. Geçici hata durumlarını kotarma amaçlı olarak gerekli olduğu hallerde kullanılan ortam temsili güncelleme modülü yine bu ana modül tarafından tetiklenen ve çıktı olarak güncellenmiş ortam temsili üreten bir sistem bileşenidir.

Robotlar ile sistem arasında mesajların yapısı açıklamaları ile birlikte Çizelge 5.1' de verilmiştir. Robotlar gönderdikleri her mesajda gönderen robotun kimliğini belirten özel bir tanımlayıcı numarası bulunur. Her robot yürüteceği, yürütmekte olduğu veya yürütmüş olduğu eylemi eylem durumu bilgisi ile birlikte gönderir. Mesaj içinde eylem durumu yürütme-sonlandı, yürütmede-hata-oluştu veya yürütme-durduruldu ise robotun dış ortamdan durum sonucu olarak algıladığı algı kümesi de bulunur. Özel olarak eylem durumu yürütmede-hata-oluştu olan mesajlarda hatanın sebebine ilişkin algıların olduğu bir bilgi de bulunmaktadır.

Çizelge 5.1 : Yürütme mesajları yapısı.

Mesaj Bölümü	Mesaj Tipi	Açıklama
Robot No	Normal Mesaj / HataMesajı	Mesajı göndern ortamdaki robotun sisteme kayıtlı numarası
Eylem (*action*)	Normal Mesaj / HataMesajı	Robotun anlık olarak ilgili olduğu planda bulunan eylem numarası
Eylem Durumu	Normal Mesaj / HataMesajı	Robotun anlık olarak ilgilendiği eylemin yürütme durumu: yürütme-başladı (*started*), yürütme-sonlandı (*finished*), yürütmede-hata-oluştu (*failed*) ve yürütme-durduruldu (*stopped*)
Algı Kümesi (*preception*)	Normal Mesaj / HataMesajı	Robotun algılayıcıları ile çevreden edindiği algı kümesi
Hata Sebebi (*cause*)	Hata Mesajı	Robotun eylemi yürütememesine sebep olan ve robot tarafından tespit edilen geçici hata bilgisi



Şekil 5.3 : Dinamik zamansal planlama sistemi durum diyagramı.

Şekil 5.3'te önerilen dinamik zamansal planlama sisteminin durum diyagramı görülmektedir. Verilen başlangıç durumu ve hedef için üretilen plan gerçek dünyadaki robotlara yollanır ve plan yürütülmesi aşamasına geçilir. Yürütme aşamasında robotlardan alınan gerçek dünya algılarını içeren mesajlar işlenir.

Yürütme gözlemleme, robotlardan alınan mesajlar içindeki algı bilgisi kullanılarak yapılır. Planın gerçek dünyada robotlar tarafından yürütülüşü takip edilir ve oluşabilecek hata durumları tespit edilir. Hata durumunun tespiti iki türlü yapılabilir:

(1) Herhangi bir robot yürüttüğü eylem esnasında hatanın oluştuğunu algılayıp bunu açıkça bildirebilir.

(2) Yürütme gözlemleme esnasında robotlardan alınan bilgiler ile planın geçerliliği (*geçerli plan* olup olmadığı) *plan benzetimi* ile sınanır ve anlık ortam durumunda var olan planla hedef duruma ulaşılamıyacağı görülebilir.

Her iki durumda da planın işleyişi askıya alınır ve tüm robotların yürütülmesini durdurması beklenir. (1) durumu için askıya alma, yürüttüğü eylem sırasında hata oluştuğunu tespit eden robotun diğer tüm robotların yürütmeyi durdurması için mesaj yayımlaması ile olurken; (2) durumu için sistem tüm robotlara yürütmelerini durdurmaları için mesaj gönderir. Hatanın tespit tipine göre iki farklı hata kotarma yöntemi bulunmaktadır: (1) durumu söz konusu olduğunda hatayı tespit eden robot hata kaynağını bildirdiğinden problem tanım kümesi bu bilgi ile güncellenir. Bu güncelleme, robotlar tarafından tespit edilen tüm hata bilgisinin bilgi tabanına eklenmesi sonrasında mevcut hatanın mevcut plan tarafından çözülebileceği yanılsaması tespit edildiğinde yapılmakta ve gerçek dünyadaki olumsuzluğun sistem bakış açısından eldeki bilgiler ışığında görülemediği durumda gerçekleşmektedir.

Sunulan problem için tasarlanmış bu çözüm hata sebepleri üzerinde çıkarsama (reasoning) yapmakta ve tanım kümesi temsilinin uygun bir şekilde güncellenmesini içermektedir. Bu çözüm muhakeme için kısıtlı miktarda tanım kümesi bilgisi olması durumunda bile uygulanabilir olmaktadır. Var olan eldeki operatör temsilleri ile *geçerli* bir plan bulunamadığı veya *geçerli* ama *yürütülemez* planların bulunduğu durumlarda bile, uygun tanım kümesi güncellemelerinin ardından yeni yürütülebilir plan oluşturulabilmektedir. Çözüm için gereken muhakeme ve önerilen güncelleme çeşitli alternatif yöntemleri ile birlikte bir sonraki bölüm olan Bölüm 6' da ayrıntılı olarak ele alınmaktadır.

Sahadaki tüm robotlar yürüttükleri eylemleri sonlandırırken veya durdururken kendi gözlemlerini de yollarlar. Bu gözlemler sonucu elde edilen bilgiler kulanılarak yeniden planlama yapılır. Eğer hedefe ulaşabilecek zamansal ve çizelgelenmiş bir plan bulunabilirse plan tekrar tüm robotlara gönderilerek yürütmeye başlanır.

Sistemin genel durum diyagramlarına ek olarak bir sonraki bölümde dinamik zamansal planlama sistemi modeli akışsal olarak ele alınarak detaylandırılmaktadır.

5.3 Sistem Algoritmaları

Dinamik planlama sistemi, planlama, yürütme, izleme ve eğer oluşmuşsa hata kurtarma adlarını verebileceğimiz birbiri içeresine geçmiş dört adımı yürütmektedir. Sunulan problemi çözmek üzere karmaşık görevin başarıyla tamamlanabilmesini sağlayan adımlar sürekli zamansal planlama genel çevrimini oluşturan Algoritma 1ile yapılmaktadır.

Algoritma 1'den de görüldüğü üzere sistemde dört farklı yürütme durumu vardır: *başlangıç (start)*, *yürütülmekte (executing)*, *askıya alındı (suspended)* ve *hata oluştu (failed)*. *Başlangıç* durumu yeni bir planın üretilmesi ve robotların bu planları yürütmeye başlaması arasında geçen süreyi kapsamaktadır. Planlar robotlara aktarıldıktan sonra *yürütülmekte* durumuna geçilir. Bu durumda robotlar kendilerine verilen planları yürütme işiyle meşguldürler. Geçici veya kalıcı eylem hatası tespit edildiğinde *hata oluştu* durumuna geçilir. Hatanın tespiti sonrası yürütülmenin tüm robotlar tarafında durdurulduğu ana kadar olan zaman diliminde *askıya alındı* durumu oluşur.

Algoritma 1, var olan tanım kümesi temsili ile başlangıç durumunda başlar (22. satır). Toplam plan zamanı (*makespan*) eniyilemesi yapabilmek için ileri durum uzayı arama yöntemi kullanan bir zamansal planlayıcı olarak TLPlan (Baccus ve Addy, 2001) kullanılmıştır. TGP tipi ADL operatörler kullanılarak kısıtlı paralellik (concurreny) sağlanmıştır. Plan üretim aşaması sonrası eğer çözüm bulunmuşsa plan robotlara aktarılmakta ve *yürütülmekte* durumuna geçilmektedir.

Algorithm 1 DYNAMICTEMPORALPLANNING (S, G)

Input: current state S, goal state G
Output: returns success or failure: messages to robots are sent

```
1: msg = NULL
2: P = ∅
3: status = start
4: for S ⊉ G do
5:     if status = executing then
6:         if VALID(P, S, G)
7:             msg = WAITMESSAGE()
8:             status = MONITOR(msg, S)
9:         else
10:            SENDALLROBOTSSTOPEXECUTION()
11:            status = suspended
12:    else if status = failed then
13:        if VALID(P, S, G) then
14:            UPDATEDOMAINDESCRIPTION(msg, Δ, S, P)
15:        status = suspended
16:    else if status = suspended then
17:        for all robots stop execution do
18:            msg = WAITMESSAGE()
19:            MONITOR(msg, S)
20:        status = start
21:    else
22:        P = PLANNER(Δ, S, G)
23:        if P = ∅ then
24:            return NOPLAN
25:        status = executing
26:        SENDPLANTOROBOTS(P)
27: return SUCCESS
```

Sistem, *yürütülmekte* durumunda iken WAITMESSAGE yordamında robotlardan yeni bir mesaj alana kadar askıda bekler (7. satır). Her bir mesaj, eylemin yürütülme durumu ile ilgili bilgiyi, algılayıcılardan alınan ortam bilgilerini ve eğer oluşmuşsa hata sebebi *c* hakkında bilgileri içermektedir. Bir mesaj ile uyanan yordam, çalışmasına MONITOR yordamını çağırarak devam eder (Algoritma 2). Algoritma 1, tanım kümesinde bulunan herhangi bir robottan hata mesajı gelene kadar veya yürütülmekte olan planın *geçersiz* olduğunu bildiren VALID(P, S, G) testi olumsuz olarak sonlanana dek kadar *yürütülmekte* durumunda kalmaya devam eder. Hata veya planın *geçersiz* olduğunun tespiti durumlarında tüm robotlara *yürütmeyi-durdur*

mesajı gönderilerek robotların yürüttükleri eylemlerini durudurmaları istenir. *Yürütmeyi-durdur* mesajını alan robot, yürütmekte olduğu eylemi durdurup durum bilgisini bildirir.

hata oluştu durumunda planlayıcı gözünden planın hala *geçerli* fakat artık *yürütülemez* olup olmadığını kontrol etmek için VALID(P, S, G) çağrısı yapılır.Eğer, bir planın robotlar tarafından yürütülemediği fakat planlayıcı açısından *geçerli* olduğunu tespit edilirse, gerekli tanım kümesi güncellemeleri yapılarak hatayı kurtamak için UPDATEDOMAINDESCRIPTION yordamını çağrılır. Bu yordamın alternatif yaklaşımlar ile ayrıntılı olarak Bölüm 6'da verilmiştir. Yapılan gerekli değişikliklerden sonra, tüm robotlardan *yürütme-durduruldu* mesajı gelmesini beklenir. Sonrasında tekrar *başlangıç* durumuna geçilerek yeni bir planın oluşturulması sağlanır. *Plan zamanı genişliği* parametresi açısından eniyi veya eniyiye yakın (makespan near-optimal) çözümler hedeflendiğinden eski planı tamir etmek (repairing) yerine yeniden planlama (replanning) yapılması uygun görülmüştür. Cushing, Benton and Kambhampati (2008)' nin çalışmalarında da bahsedildiği gibi eğer robotlar yeni planı oluşturmak için yeterli kaynağa sahiplerse yeniden planlamanın genel performansı plan tamirine göre daha iyi olmaktadır.

Robotlardan gelen mesajları işleyerek yürütülmekte olan planın durumunu izlemek için Algoritma 2 kullanılır. Var olan planın izlenmesi plan benzetiminin şimdiki ortam durumundan başlanarak uygulanması ile yapılmaktadır. Eylemlerin başlangıç ve sonuç etkilerinin ayrı ayrı uygulanması ile benzetimin zamansal bir bileşeni oluşmaktadır.

Robotlardan *yürütme-sonlandı (finished)*, *yürütmede-hata-oluştu (failed)* veya *yürütme-durduruldu (stopped)* mesajları gelmişse, doğru ortam durum bilgisine ulaşabilmek için mesaj içeriğindeki algı kümesi şimdiki ortam durumu S'e uygulanmalıdır. Diğer mesaj türlerinden farklı olarak *yürütmede-hata-oluştu* mesajı alındığında yürütme durumu olarak *hata oluştu* durumuna geçilir.

Algorithm 2 MONITOR (msg, S)

Input: message msg, currrent state S
Output: status of planning, current state S

1: $status = executing$
2: $a = msg.action$
3: $state_a = msg.state$
4: $percv = msg.perception$

5: **if** $state_a = started$ **then**
6: $S = \text{APPLYSTARTEFFECTS}(a, S)$
7: **else if** $state_a = finished$ **then**
8: $S = \text{APPLYENDEFFECTS}(a, S)$
9: $S = \text{MERGE}(S, percv)$
10: **else if** $state_a = failed$ **then**
11: $S = \text{MERGE}(S, percv)$
12: $status = failed$
13: **else if** $state_a = stopped$ **then**
14: $S = \text{MERGE}(S, percv)$
15: **return** $status$

5.4 Yürütme Zamanında Hata Sezme ve Robot Altyordamları

Verilen sistemde alt katman yol planlaması ile üst katman eylem planlama birlikte genel bir sistem temsili içinde entegre edilmiştir. Bu iki katman arasında sıkı bir bağ olduğundan dolayı topolojik haritalama yaklaşımı uygun bulunmuştur. Bu yaklaşım her iki katman için temsil ve soyutlamada istenilen esnekliğin yakalanmasını sağlamıştır. Topolojik harita, alanda bulunan serbest alanları gösteren graf düğümleri ve aralarında yol olan düğüm çiftleri için ayrıtları içeren graf yapılarıyla temsil edilir. Harita, hedefe olan planın işleyişi esnasında artımlı olarak yaratılır. Bu yaklaşımla robot keskin dönüşler yaptığında veya bir engel tespit ettiğinde (Ranganathan ve Dellaert, 2009) bir önceki düğüme belirli bir uzaklıkta yeni bir düğüm yaratılır (Simmons ve Koenig, 1995).

Robot, kendi yolu üzerinde yeni bir engel keşfettiğinde geçersiz bağlantılar harita temsilinden silinirler. İki düğümün bağlantısı koptuğu zaman yol planlayıcısı haritayı güncellemek için bir bilgi sağlar. Harita düğümleri eylem planlama tanım kümesinde olgular olarak temsil edilebilir. Eğer hata sebebi olarak yolu kapatmış bir engel olarak verilirse bu bilgi hem yol planlayıcısının hem de üst katman eylem planlayıcının kullanabileceği bir şekilde kolayca bilgi tabanına eklenebilir.

6. PLAN TANIM KÜMESİ GÜNCELLEME

6.1 Giriş

Bu bölümde Bölüm 4' te anlatılan probleme ilişkin çözüm yöntemleri tartışılmaktadır. Alternatif yöntem olarak Bölüm 5' te bahsedilen dinamik zamansal planlama siteminin içinde plan tanımı güncelleme modülü için önerilen yöntemler detaylı olarak anlatılacaktır.

6.2 Dinamik Planlama Problemi İçin Bilinen Çözüm Yöntemleri

Yürütme anında hata durumunda sadece başlangıç ve hedef ortam durumlarının değiştirilmesi (Fox ve dig., 2006) ile planlayıcının *geçerli* fakat *yürütülemez* plan bulması durumunu değiştirilemeyebilir. Bu tür planların bulunmasını engellemek adına bir takım ek önlemlerin alınması gerekmektedir (Cushing ve Kambhampati, 2005). Sunulan probleme farklı çeşitli sezgisel çözüm önerileri olabilir : (1) hatalı eylemleri *etkisiz kılmak*, (2) hatalı eylemleri *askıya alamak*, (3) artımlı (*incremental*) veya ara hedef (*subgoal*) planlama. Birinci yöntemde, hatalı eylemleri sadece *etkisiz kılmak*, bu tür eylemlerin hedefe giden eylem sekansı içinde bulunması gerekli olduğu zamanlarda eklenmesini engellenebilir. İkinci yöntemde bu durum, belirli eylemleri belirli bir zaman askıya alarak ve plana bu zaman aralığı içinde dahil etmeden yeni bir planı oluşturmakla engellenebilir. Fakat bu durumda askıya alma süresinin tanım kümesi ve duruma bağımlı olması, önceden kestirilememesini ve bu süreye ait gerçekçi tahminlerin yapılamamasıproblemleri söz konusudur. Artımlı veya ara hedef planlama yötemlerinde eniyi olmayan (*suboptimal*) çözümler üretilebilir veya planlayıcı önerilen ara hedefler için alt planlar bulamayabilir.

6.3 Önerilen Yöntemler

Geçici eylem hatası olduğunda bu hatalı eylemin planlayıcı tarafından yeniden seçilmesini engellemek uygun bir çözüm olarak görülmektedir. Öyleki plan ağacında hata durumu ortadan kalktıktan sonra üretilen dallarda hatalı eylem tekrar geçerli

hale getirilmeli ve eğer hedefe ulaşmak için gerekliyse tekrar kullanılabilmelidir. Bu çalışmada bahsi geçen *eylemi geçersiz hale getirme* işini farklı yaklaşımlarla yapan ama temel mantığı aynı olan çeşitli yöntemler geliştirilmiştir. Sunulan problem için tasarlanmış bu yöntemler hata sebepleri üzerinde çıkarım (reasoning) yapmakta ve tanım kümesi temsilinin uygun bir şekilde güncellenmesini içermektedir. Önerilen yaklaşımlarda ise, çıkarım için kısıtlı miktarda tanım kümesi bilgisi olması durumunda bile uygulanabilir olmaktadır. Çözüm için gereken çıkarım ve önerilen güncelleme için çeşitli alternatif yöntemleri ile birlikte aşağıda alt bölümlerde verilmektedir. Bu yaklaşımların başarım analizleri ve aralarındaki farklar ise bir sonraki bölümde anlatılmaktadır.

6.3.1 Sözde operatör yöntemi

Önerilen sözde operatör yönteminde aslında gerçek dünyada eylem karşılığı olmayan ve sadece planlayıcı tarafında geçerli olan ve planlama aşamasında kullanılacak operatörler oluşturulmakta ve hata durumlarını kotarmak üzere kullanılmaktadır. Bu operatör önkoşul ve etkileriyle hatalı eylemi açmak için kullanılır.

Algoritma 3.1 bu yöntemi kullanarak gerekli tanım kümesi güncellemelerini yapmaktadır. (5.) ve (9.) satırlarda hatalı eylem, ismi eylem adının sonuna "_locked" soneki getirilmiş şekilde, parametreleri ise söz konusu eylemin operatörünün parametleri olan (newPredicate) bir yüklem yaratılıp tanım kümesine eklenmektedir. Eklenen yüklemin parametrelerinin değerleri hatalı eylemin argümanlarından oluşan olgu şimdiki ortam durumuna eklenerek söz konusu eylem *kilitlenir*. (15.) ve (19.) satırlar, eğer daha önceden yaratılmadıysa, hatalı eylem ve hata sebebi çifti (a_{failed}, c) için yeni bir sözde operatör (o_{pseudo}) yaratılır. (21.) satırda yaratılan bu sözde operatörün hatalı eylem ve hata sebebi çifti için yaratıldığı kaydedilir. Yaratılan bu sözde operatör ilk tanım kümesi operatörlerinden hata sebebi ve hatalı eyleme has önkoşulları, etkileri ve parametreleri ile ayrılır. Bu operator hatalı eylemin *kilitli* olduğu süre içinde ve söz konusu hata nedeni c parametre olarak verildiğinde çalışabilmesi ve diğer durumlarda (tekrar) seçilmesini engelleyen özel önkoşullara sahiptir. *Kilitli* olan eylemin *kilitliliğini* kaldırmak için özel bir etkiye (effect) sahiptir. Ek olarak Algoritma 4.1'de verilen MAKEFAILUREPRECONDITIONS yordamı tarafından oluşturulan özel önkoşullar da barındırmaktadır. Bu yordam öncelikle tanım kümesinde c' nin niteliklerini etkileyen

etkilere sahip operatörleri arar. Sonra bulduğu eşleşen niteliklerin şimdiki hatalı ortam durumu S_{failed}'deki değerlere göre eşitsizlik ifadelerinin mantıksal olarak bir kümede toplar. Böylece bu eşitsizlik ifadeleri, planlama aşamasında c'nin seçilen bu özelliklerinden herhangi biri herhangi bir eylem ile değiştiğinde hatalı *kilitli* eylemin plana dahil olabilmesini sağlar. Bu yöntemin temelindeki en önemli düşünce, hatalı olan eylemin plana dahil olmasının, c üzerinde yapılan bir değeşikliğe kadar ertelenmesidir. Eğer sözde operatör önceden var ise (böyle bir hata durumu daha önce oluşmuş ve sözde operatör (a_{failed}, c) çifti için zaten yaratılmışsa) planlayıcı tarafından kilidi açan alternatif eylem hatanın kurtarılması için istenilen etkiyi yapmamış anlamına gelir. Bu durumda Algoritma 3.1'deki gibi (23.) ve (30.) satılar arasında, seçilen bu alternatif eylemin operatörünün etkileri sözde operatörün önkoşullarından silinir. Böylece bu alternatif eylem sonraki planlarda artık hatalı eylem a_{failed}'i uygulanabilir hale getiremeyecektir. Plan öncelik kısıt grafiğinde sözde operatörden bir önceki operatör alternatif operatör ele alınmaktadır. Uygun olmayan alternatif operatör hatanın çözümü için yapılacak planlarda alternatif olarak değerlendirilmeyecektir.

Algorithm 3.1 UPDATEDOMAINDESCRIPTION (msg,Δ,S,P)

Input: message msg, planning domain Δ,
current state S, current failed plan P
Output: updated planning domain Δ

1: $c = msg.cause$
2: $a_{failed} = msg.action$
3: $o_{failed} = operator(a_{failed})$
4: $preList = \emptyset$
5: newPredicate.name = CONCAT($name(o_{failed})$, "_locked")
6: newPredicate.parameters = param((o_{failed}))
7: $pre(o_{failed}) = pre(o_{failed}) \cup \neg newPredicate$
8: newFact = GENERATEFACT (newPredicate, $arg(a_{failed})$)
9: $S = S \cup newFact$
10: $\Delta = \Delta \cup newPredicate$
11: $preList = (newParam = c)$
12: $preList = preList \cup newPredicate$
13: o_{pseudo} = GETPSEUDOOPERATOR(a_{failed}, c)
14: **if** $o_{pseudo} = NIL$ **then**
15: $param(o_{pseudo}) = param(o_{failed}) \cup newParam$
16: $pre(o_{pseudo}) = preList$

17: $pre(o_{pseudo}) = pre(o_{pseudo})$
18: ∪ MAKEFAILUREPRECONDITIONS$(c, \Delta, S, \emptyset)$
19: $del(o_{pseudo}) = del(o_{pseudo}) \cup newFact$
20: $\Delta = \Delta \cup o_{pseudo}$
21: SETPSEUDOOPERATOR$(a_{failed}, c, o_{pseudo})$
22: **else**
23: a_{pre} = GETPREDECESSORACTION(P, o_{pseudo}, c)
24: $o_{pre} = operator(a_{pre})$
25: $executedList$ = GETEXECUTEDOPERATORS(a_{failed}, c)
26: $executedList = executedList \cup o_{pre}$
27: SETEXECUTEDOPERATORS$(a_{failed}, c, executedList)$
28: $pre(o_{pseudo}) = preList$
29: $pre(o_{pseudo}) = pre(o_{pseudo}) \cup$
30: MAKEFAILUREPRECONDITIONS$(c, \Delta, S, executedList)$

Sözde operatör yaratmak için gerekli olacak ön bilgi tümüyle ihmal edilebilir olup genel bir yöntem uygulanabilir. Öyleki robot çok az bilgiye sahip olsa da tanım kümesi tanımının değiştirilmesi oluşan hatanın yerelliğinin sezgisel prensibinden dolayı mümkün olabilmektedir. Anlamsal (semantic) yüklemeler ile hata hakkındaki nedenin veya nedenlerin çıkarsamasının kalitesine bağlı olarak bu yordamın çıktısı çok daha özel olabilir. Örneğin sebep olarak "E nesnesinin engel teşkil etmesi yüzünden" ifadesi girdi olarak verilmişse ve engel teşkil etmenin konum ile bağlantısı çıkarsanabilirse engel teşkil eden nesnenin özelliklerinden sadece konum bilgisi özellikleri yeniden planlamada göz önüne alınacaktır. Eğer böyle bir çıkarsama mümkün değilse nesnenin rengi gibi durum ile ilgisiz nesne özelliklerini içeren önkoşullar da göz önüne alınacaktır.

Algorithm 4.1 MAKEFAILUREPRECONDITIONS (c,Δ, S, l)

Input: failure cause c, planning domain Δ,
current state S, executed operator list l
Output: disjunctive precondition list $prelist$

1: $prelist = \emptyset$
2: **for each** o_i in Δ but not in l **do**
3: **for each** *effect* e_j in $add(o_i) \cup del(o_i)$ **do**
4: **for each** *attribute* $attr_k$ in $attr(c)$ **do**
5: **if** e_j affects $attr_k$ **then**
6: $value = value(attr_k, S)$
7: $prelist = prelist \cup \neg(attr_k = value)$
8: **return** $prelist$

6.3.2 Hata giderici operatörleri kopyalama yöntemi

Hata giderme potansiyeli olan operatörler aslında gerçek dünyada karşılığı olan fakat ortam temsilinde var olan bir hatayı çözmek için doğrudan etkileri olmayan operatörlerdir. Bu yöntemde sözde operatör yöntemindeki sözde operatörün yaptığı işin kopyalanmış operatörlere taşınması yaklaşımı vardır. Bu operatörler ilk hallerinden farklı olarak sadece hata durumunda çalışacak şekilde güncellenip ortam temsiline eklenirler.

Algorithm 3.2 UPDATEDOMAINDESCRIPTION (msg,Δ,S,P)

Input: message msg, planning domain Δ,current state S, current failed plan P
Output: updated planning domain Δ

1: $c = msg.cause$
2: $a_{failed} = msg.action$
3: operatorList $= \emptyset$
4: $o_{failed} = operator(a_{failed})$
5: newPredicate.name $=$ CONCAT($name(o_{failed})$, "_locked")
6: newPredicate.parameters $=$ param((o_{failed}))
7: $pre(o_{failed}) = pre(o_{failed}) \cup \neg\, newPredicate$
8: newFact = GENERATEFACT (newPredicate, $arg(a_{failed})$)
9: $S = S \cup newFact$
10: $\Delta = \Delta \cup newPredicate$
11: $operatorList =$ GETOPERATORLIST(a_{failed}, c)
12: **if** $operatorList = NIL$ **then**
13: $operatorList =$ FINDRELATEDOPERATORS(c, Δ)
14: **for each** o_i *in* operatorList **do**
15: $o_{cloned} =$ CLONE(o_i)
16: $param(o_{cloned}) = param(o_{cloned}) \cup$ param(o_{failed})
17: $preList =$ MAKEFAILUREPRECONDITIONS(c, o_{cloned}, S)
18: $pre(o_{cloned}) = pre(o_{cloned}) \cup preList$
19: $pre(o_{cloned}) = pre(o_{cloned}) \cup newFact$
20: $del(o_{cloned}) = del(o_{cloned}) \cup newFact$
21: ADD(operatorList, o_{cloned})
22: $operatorList = operatorList \cup o_{pre}$
23: SETOPERATORLIST ($a_{failed}, c, operatorList$)
24: $\Delta = \Delta \cup$ operatorList
25: **else**
26: $a_{pre} =$ GETPREDECESSORACTION(P, o_{cloned}, c)
27: $o_{pre} = operator(a_{pre})$
28: $operatorList = operatorList \setminus o_{pre}$
29: SETOPERATORLIST ($a_{failed}, c, operatorList$)
30: $\Delta = \Delta \setminus o_{pre}$

Planlayıcı kopya operatörleri kullanarak plan oluşturduğunda ilk hali ile aynı şekilde gerçek dünyada yürütülebilir birer eylemi temsil ederler. Yürütücü robot açısından orijinal operatörü ile arasında bir fark yoktur. Algoritma 3.2' den bu yöntemin akışı temsilen verilmiştir. İlk yönteme benzer şekilde ilgili olgunun, yüklemin (*predicate*) ve önkoşulun problem temsiline eklenmesiyle hatalı eylem (4.) ve (9.) satırları arası yürütülerek *kilitlenir*. Daha sonra yine benzer olarak hatalı eylem ve sebep çifti için daha önce hata oluşup oluşmadığı kontrol edilir. Daha önce hata oluşmamışsa hata sebebini etkileyen operatörler Algoritma 4.2'de gösterilen FINDRELATEDOPERATORS yordamı ile bulunarak (15.) ve (20.) satırlar arası birer kopyaları yaratılır ve bu kopyalar Algoritma 5.2' de gösterilen MAKEFAILUREPRECONDITIONS yordamı yardımıyla planlayıcı tarafından sadece hatalı eylem kilitli iken çalışıp kilidi açmak için kullanılacak hale getirilirler. Daha önce hata oluşmuşsa hataya uğramış planda kilidi açma işini yapan kopya eylem bulunur ve problem temsilinden silinerek bir sonra oluşacak planda kullanılması önlenmiş olunur.

Algorithm 4.2 FINDRELATEDOPERATORS(c,Δ)

Input: failure cause c, planning domain Δ,
Output: disjunctive operator list $operatorList$ consits of operators affect c

1: $operatorList = \emptyset$
2: **for each** o_i in Δ **do**
3: **for each** *effect* e_j in $add(o_i) \cup del(o_i)$ **do**
4: **for each** *attribute* $attr_k$ in $attr(c)$ **do**
5: **if** e_j affects $attr_k$ **then**
6: $operatorList = operatorList \cup o_i$
7: **return** $operatorList$

Algorithm 5.2 MAKEFAILUREPRECONDITIONS (c,o, S)

Input: failure cause c, operator o, current state S
Output: disjunctive precondition list $prelist$

1: $prelist = \emptyset$
2: **for each** *effect* e_j in $add(o_i) \cup del(o_i)$ **do**
3: **for each** *attribute* $attr_k$ in $attr(c)$ **do**
4: **if** e_j affects $attr_k$ **then**
5: $value = value(attr_k, S)$
6: $prelist = prelist \cup \neg(attr_k = value)$
7: **return** $prelist$

6.3.3 Hatalı operatör kopyalama yöntemi

Son yöntemde hatalı eyleme ilişkin operatörün kopyalanması sağlanmıştır. Diğer yöntemlerden farklı olarak, bu yöntemde hatalı eyleme ilişkin operatör planlayıcı açısından hiç bir zaman seçilmez. İlk halinden farklı olarak kopyalanmış operatör plan arama ağacında sadece hatanın düzelmesi sonrasındaki durumlarda seçilebilecek şekilde önkoşullara sahip olarak orijinal operatörün işini yapar. Yöntemin akışı Algoritma 3.3' te ayrıntılı olarak verilmiştir.

Algorithm 3.3 UPDATEDOMAINDESCRIPTION (msg,Δ,S,P)

Input: message msg, planning domain Δ,
current state S, current failed plan P
Output: updated planning domain Δ

1: $c = msg.cause$
2: $a_{failed} = msg.action$
3: $o_{failed} = operator(a_{failed})$
4: $preList = \emptyset$
5: newPredicate.name = CONCAT($name(o_{failed})$, "_locked")
6: newPredicate.parameters = param((o_{failed}))
7: $pre(o_{failed}) = pre(o_{failed}) \cup \neg\, newPredicate$
8: newFact = GENERATEFACT (newPredicate, $arg(a_{failed})$)
9: $S = S \cup newFact$
10: $\Delta = \Delta \cup newPredicate$
11: $preList = (newParam = c)$
12: $preList = preList \cup newPredicate$
13: o_{cloned} = GETCLONEOPERATOR(a_{failed}, c)
14: **if** $o_{cloned} = NIL$ **then**
15: o_{cloned} = CLONE(o_i)
16: $pre(o_{cloned}) = preList$
17: $pre(o_{cloned}) = pre(o_{cloned})$
18: ∪ MAKEFAILUREPRECONDITIONS($c, \Delta, S, \emptyset$)
19: $\Delta = \Delta \cup o_{cloned}$
20: SETCLONEOPERATOR($a_{failed}, c, o_{cloned}$)
21: **else**
22: a_{pre} = GETPREDECESSORACTION(P, o_{cloned}, c)
23: $o_{pre} = operator(a_{pre})$
24: $executedList$ = GETEXECUTEDOPERATORS(a_{failed}, c)
25: $executedList = executedList \cup o_{pre}$
26: SETEXECUTEDOPERATORS($a_{failed}, c, executedList$)
27: $pre(o_{cloned}) = preList$
28: $pre(o_{cloned}) = pre(o_{cloned}) \cup$
29: MAKEFAILUREPRECONDITIONS($c, \Delta, S, executedList$)

Algoritma 3.3 hatalı eyleme ilişkin operetör kopyalama yöntemini kullanarak gerekli tanım kümesi güncellemelerini yapmaktadır. (5.) ve (9.) satırlar arasında Algoritma 3.1'deki gibi söz konusu hatalı eylem *kilitlenir*. (13.) satırda (a_{failed}, c) ikilisi için daha önce bir hata olup oluşmadığı kontrol edilir. (15.) ve (19.) satırlar arasında eğer daha önceden yaratılmadıysa hatalı eylem ve hata sebebi çifti (a_{failed}, c) için ismi hariç hatalı operatörle tamamen aynı olan yeni bir kopya operatör (o_{cloned}) yaratılır. (20.) satırda yaratılan bu kopya operatör hatalı eylem ve hata sebebi çifti (a_{failed}, c) için kaydedilir. Yaratılan bu sözde operatör asıl orjinalinde hata sebebi ve hatalı eyleme has önkoşulları, etkileri ve parametreleri ile ayrılır. Sadece hatalı eylem *kilitli* olduğu zaman ve söz konusu hata nedeni c parametre olarak verildiğinde çalışabilmesi diğer durumlarda (tekrar) seçilmesini engelleyen özel önkoşullara sahiptir. Ek olarak MAKEFAILUREPRECONDITIONS yordamı tarafından oluşturulan özel önkoşullar da barındırmaktadır. Bu yordam ilk yöntemde verilen Algoritma 4.1 ile aynıdır.

Eğer kopya operatör önceden var ise (böyle bir hata durumu daha önce oluşmuş ve kopya operatör (a_{failed}, c) çifti için zaten yaratılmışsa) planlayıcı tarafından hatayı düzelten eylem yanlış seçilmiştir. Bu durumda (22.) ve (29.) satılar arasında, seçilen bu alternatif eylemin operatörünün etkileri kopya operatörün önkoşullarından silinir. Böylece bu alternatif eylem sonraki planlarda artık hatalı eylem a_{failed}' i uygulanabilir hale getiremeyecektir. Böylece bir sonraki planda uygun olmayan alternatif operatör hatanın çözümü için alternatif olarak değerlendirilmeyecektir.

7. DENEYSEL SONUÇLAR

7.1 Giriş

Bu bölümde önceki bölümlerde tasarımı verilen yaklaşımların deneysel analizi yapılmaktadır. Öncelikle Bölüm 6'da verilen yöntemler ve ara hedef planlama yöntemi çözüm kaliteleri ve çözebildikleri problem kümeleri kriterleri açısından karşılaştırılmaktadır. Daha sonra Bölüm 5' te verilen sistemin dış ortamı Webots benzetim ortamı üzerinde gerçeklenmiş hali ile belirli bir problem üzerinde uygun sonuçlar verip vermediği analiz edilecektir.

7.2 Planlama Deney Kümeleri Üzerinde Başarım Analizi

Bu bölümde Bölüm 6'da sunulan çözüm yöntemlerini karşılaştırmak için iki farklı deney ortamı tasarlanmıştır. Her iki deney ortamı farklı birer planlama temsil kümesini içermektedir. Her ortamdaki deneyler farklı problem kümeleri için tekrarlanmış ve sonuçlar listelenmiştir. Deney sonuçlarında sözde operatör yöntemi *yöntem 1,* hata giderici operatörleri kopyalama yöntemi *yöntem 2* ve hatalı operatör kopyalama yöntemi *yöntem 3* adlarıyla belirtilmektedir.

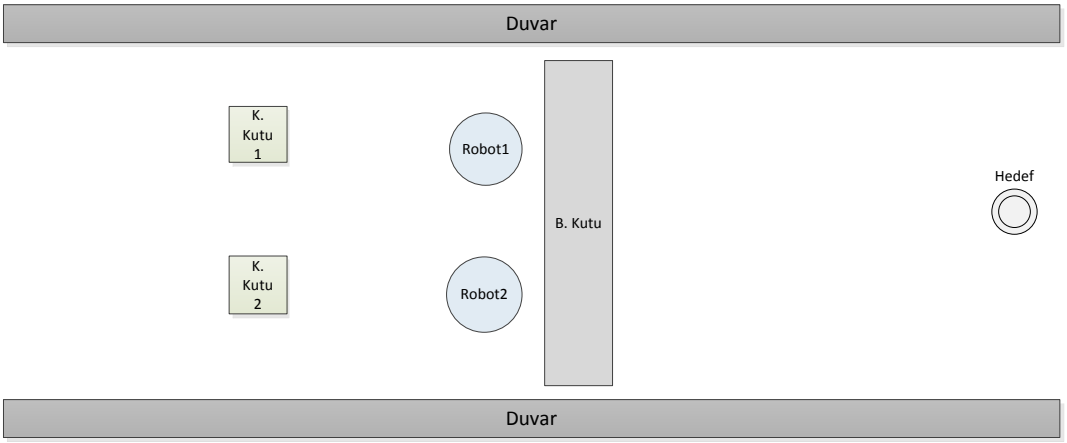
7.2.1 Birinci deney kümesi

Birincil deney ortamı önceki bölümlerde öne sürülen üç yöntemin performanslarını karşılaştırmak için tasarlanmış Şekil 7.1'de temsili olarak resmedilen bir ortamdır.

Ortamın planlama temsil kümesinde altı adet operatör bulunmaktadır:

- *(move-to-loc ?robot ?konum)* operatörü robotun istenen konuma gitmesini,
- *(move-to-obj ?robot ?nesne)* operatörü robotun bir nesnenin yanına üzerinde bir eylem yapmak için konumlanmasını,
- *(pick ?robot ?küçükNesne)* operatörü robotun küçük nesneleri kolu vasıtasıyla sırasıyla yerden kaldırmasını,

- *(drop ?robot ?küçükNesne)* operatörü robotun kolunda bulunan küçük nesneleri yere indirilmesini,
- *(push ?robot ?büyükNesne*) operatörü robotun kaldıramayacağı büyük nesnelerin sürükleme yöntemi ile ittirerek konumunu değiştirmesini sağlar.



Şekil 7.1 : Planlama kümeleri başarım analizi için birinci deney ortamı örneği.

Deney kümesinde kullanılan ortamda robotlar var olan büyük taşınamayan bir kutu (engel) arkasında bulunan taşınabilir küçük kutuları hedef noktaya taşımaları gerekmektedir. Küçük kutular robotlardan geride yerde durmaktadır. Robotlar ilk durum için yolun kapalı olduğu bilgisine sahiptir, gerekli planlama temsili güncellemeleri her 3 yöntem için de yapılmıştır. Bu ilk durum gerçek hata durumu olabilecek, robotların açık olduklarını varsaydıkları bir koridorda kollarındaki küçük kutularla ilerlerken karşılarına çıkan büyük kutudan dolayı ilerleyemedikleri bir senaryoya benzerdir. Tek fark taşınacak kutuların hata durumundaki gibi robot kollarında değil de yerde olmasıdır. Deney kümesinde taşınması gereken küçük kutu sayısı veya kutuları taşıyacak robot sayısının değiştirilerek deney tekrarlanırken değişikliklerin ölçülebilir olması açısından deney ortamı bu şekilde tasarlanmıştır. Bu deney senaryosunda plan oluşturmadaki asıl zorluk büyük kutunun itilmesi durumunda yolun açılacağından planlayıcının çıkarımının olamamasıdır.

Bu deneylerde yöntem 1, yöntem 2 ve yöntem 3 için yapılmış planlama ortam temsili güncellemeleri sonucu değişiklikler sırasıyla Şekil 7.2, Şekil 7.3 ve Şekil 7.4’ te temsilen gösterilmiştir.

```
(def-adl-operator (move-to-loc ?loc)
  (pre
       …
      (not (move-to-loc_locked ?loc))
  )
  …
)
(def-adl-operator (pseudo1 ?obj ?loc)
  (pre
      …
      (and
         (= ?obj obstacle)
         (move-to-loc_locked ?loc)
         (or
           (not (xcoord ?obj obs_x))
           (not (ycoord ?obj obs_y))
         )
      )
  )
  (del (move-to-loc_locked ?loc))
)
```

Şekil 7.2 : Birinci deney kümesi hatasının çözümü için sözde operatör yöntemi operatör yöntemi tanım kümesi güncellemesi.

```
def-adl-operator (move-to-loc ?loc)
  (pre
       …
      (not (move-to-loc_locked ?loc))
  )
  …
)
(def-adl-operator (push-2 ?obj ?loc)
  (pre
      …
      (and
         (= ?obj obstacle)
         (move-to-loc_locked ?loc)
         (or
           (not (xcoord ?obj obs_x))
           (not (ycoord ?obj obs_y))
         )
      )
  )
  (del (move-to-loc_locked ?loc))
  …
)
```

Şekil 7.3 : Birinci deney kümesi hatasının çözümü için hata giderici operatörleri kopyalama yöntemi tanım kümesi güncellemesi.

```
def-adl-operator (move-to-loc ?loc)
  (pre
       …
      (not (move-to-loc_locked ?loc))
  )
  …
)
(def-adl-operator (move-to-loc-2 ?loc ?obj)
  (pre
      …
      (and
         (= ?obj obstacle)
         (move-to-loc_locked ?loc)
         (or
            (not (xcoord ?obj obs_x))
            (not (ycoord ?obj obs_y))
         )
      )
  )
  …
)
```

Şekil 7.4 : Birinci deney kümesi hatasının çözümü için hatalı operatör kopyalama yöntemi tanım kümesi güncellemesi.

7.2.1.1 Birinci deney sonuçları

Ortam temsili güncellemesinin 3 farklı yöntemle yapılması sonucu durumlarda 6 farklı deney kümesi üzerinde ve eniyi-öncelikli arama ve A* arama stratejileri üzerinde planlayıcının çözüm kalitesi analiz edilerek başarım analizi yapılmış ve sonuçlar Çizelge 7.1'de gösterilmiştir. Deneyler Intel Core 2 Duo P8700 2,53 GHz işlemci ve 4GB belleğe sahip bir bilgisayar üzerinde yapılmıştır.

Çizelgede ÜD, plan ağacında üretilen düğüm sayısını, AD ise plan ağacında açılan düğüm sayısını göstermektedir. Planlama süresi, verilen parametre kümesiyle analiz ortamında saniye cinsinden planın bulunma süresini göstermektedir (Bu değer 10 kere koşturma sonucu bulunan ortalama değerdir). Tüm deneylerde planlayıcıya plan bulabilmesi için en fazla 500 saniye zaman kısıtı ve en fazla 20000 düğüm açma bellek sınırı konulmuştur. Çizelgede bulunan *Çözüm Yok* işaretli hücrelere karşı gelen deneylerde bu kısıtlar altında plan bulunamadığı anlamına gelmektedir. Çizelgedeki bir diğer parametre plan süresi ise oluşturulan zamansal planın başlangıcından bitişine kadar tamamlanma süresini (plan süresinin en fazla genişliği) (*makespan*) veren plan kalitesi parametresidir.

Çizelge 7.1 : Birinci deney ortamı yöntem ve plan stratejileri karşılaştırmaları.

Problem	Yöntem	En iyi-öncelikli Arama (Best-first Search)				A* Arama			
		ÜD	AD	Planlama Süresi (sn)	Plan Süresi (*makespan*) (sn)	ÜD	AD	Planlama Süresi (sn)	Plan Süresi (*makespan*) (sn)
1 robot 1 kutu	1	46	23	0,038	46,97	103	54	0,088	46,97
	2	Çözüm Yok				123	68	0,116	46,97
	3	44	22	0,04	46,97	79	45	0,078	46,97
1 robot 2 kutu	1	105	45	0,11	96,77	1148	550	0,78	81,01
	2	Çözüm Yok				1050	558	0,8	81,01
	3	100	44	0,106	96,77	782	421	0,868	81,01
1 robot 3 kutu	1	177	70	0,194	143,38	21586	9720	59,25	118,4
	2	Çözüm Yok				19569	9779	50,55	118,4
	3	175	69	0,194	143,38	14794	7425	25,3	118,4
2 robot 2 kutu	1	128	48	0,136	76,08	35193	12202	170,1	45,13
	2	Çözüm Yok				38128	14376	195,13	45,13
	3	133	50	0,144	76,08	26793	10328	75,58	45,13
2 robot 3 kutu	1	319	101	0,323	151,19	Çözüm Yok			
	2	Çözüm Yok				Çözüm Yok			
	3	296	97	0,31	151,19	Çözüm Yok			
2 robot 4 kutu	1	609	172	0,586	240,43	Çözüm Yok			
	2	Çözüm Yok				Çözüm Yok			
	3	618	174	0,634	240,43	Çözüm Yok			

Çizelge arama stratejileri açısından analiz edildiğinde beklenen bir şekilde A* arama algoritmasının en iyi-öncelikli aramaya göre daha fazla zamanda ve daha fazla düğüm oluşturarak ve açarak sonuç oluşturduğu görülmektedir. Bunun yanında çözüm bulabildiği noktada A* arama stratejisinin doğal olarak daha iyi plan kalitesi elde ettiği görülür. Ortamın karmaşıklığı arttıkça A* arama stratejisi ile yapılan deneylerde her 3 yöntemde de koyulan kısıtlar (zaman ve bellek kısıtları) aşıldığından planlayıcı çözümü bulamamaktadır. Gerçek dünya temsili soyutlamasında artan karmaşıklığa A* arama stratejisi ölçeklenebilir bir cevap verememektedir.

Çizelgede elde edilmiş sonuçlardan arama stratejileri temelli yöntemleri karşılaştırdığımızda en iyi- öncelikli aramada yöntem 2' nin hiç bir problemde çözüm bulamadığını görülmektedir. Bu arama stratejisinde yöntem 1 ve yöntem 3'ün ürettikleri plan kalitesi arasında hiç bir fark olmamasıyla beraber, planlama süreleri ve arama uzayı parametreleri açısından bakıldığında farklılıkların bu problem kümesi için gerçeklemeye bağlı farklılıklar oldukları fakat bu farklılıkların ihmal edilebilir oldukları söylenebilir. A* arama stratejisine bakıldığında ise yöntem 3'ün yöntem 1 ve yöntem 2'ye göre plan kalitesi dışında üretilen ve açılan düğüm sayısı buna paralel olarak planlama süresi açısından daha iyi olduğu görülmektedir. Bunun sebebi yöntem 3'ün kilitli olan eylemi açmak yerine kilitsiz fakat hatanın ortadan kalktığı durumda doğrudan kilitli eylemin kopyasını kullanmasıyla hedefe düğüme ulaşması stratejisindendir.

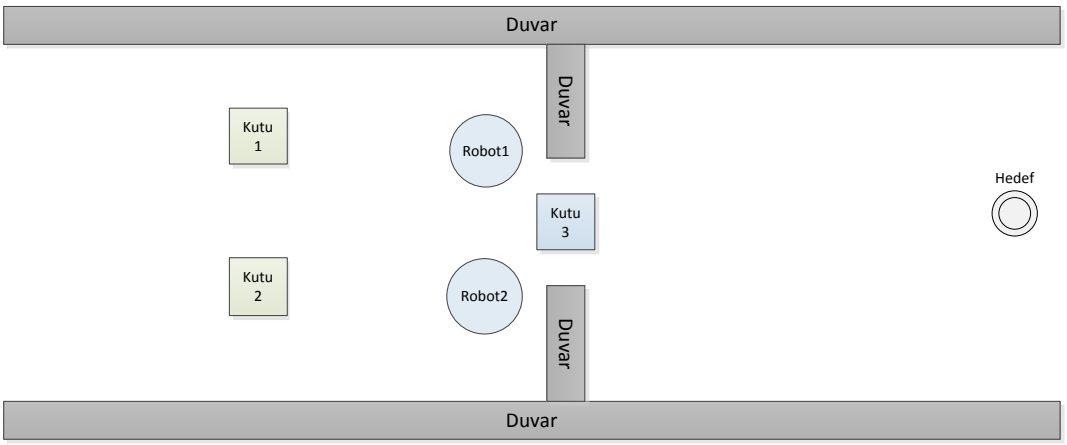
7.2.2 İkinci deney kümesi

İkincil deney ortamı önceki bölümlerde öne sürülen üç yöntemden yöntem 3'ün aslında diğer yöntemlerin veya ara hedef planlama (subgoal planning) yaklaşımı ile çözüm bulunamayan bir uzayı tarif edebilmek amaçlı oluşturulmuş ve Şekil 7.5'te temsili olarak resmedilen bir ortamdır.

Ortamın planlama temsil kümesinde bir önceki deney kümesinden farklı olarak beş adet operatör bulunmaktadır:

- *(move-to-loc ?robot ?konum)* operatörü robotun istenen konuma gitmesini,
- *(move-to-obj ?robot ?nesne)* operatörü robotun bir nesnenin yanına üzerinde bir eylem yapmak için konumlanmasını,

- *(pick ?robot ?küçükNesne)* operatörü robotun küçük nesneleri kolu vasıtasıyla yerden kaldırmasını,
- *(drop ?robot ?küçükNesne)* operatörü robotun kolunda bulunan küçük nesneleri yere indirmesini sağlar.



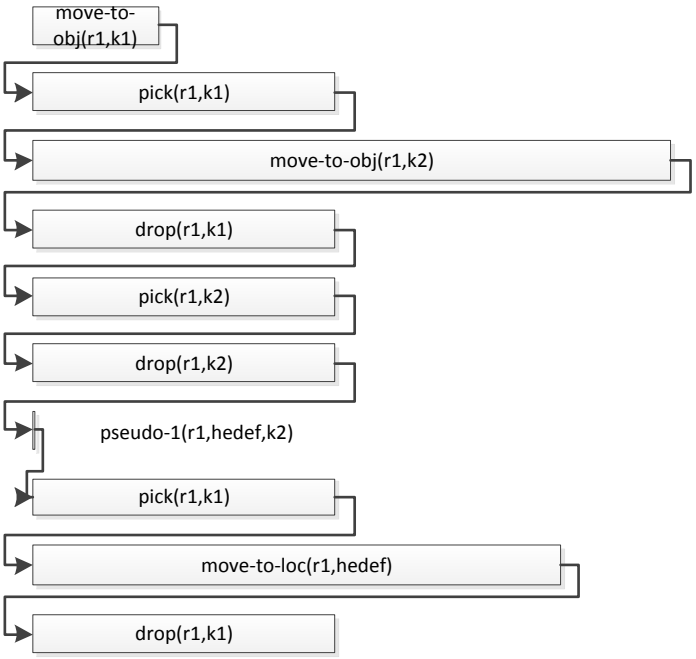
Şekil 7.5 : Planlama kümeleri başarım analizi için ikinci deney ortamı örneği.

Oluşturulan deney kümesindeki ortamda robotlar, ortasında dar fakat kendilerinin geçebileceği büyüklükte bir geçit olan bir koridorun ortasında bulunmaktadırlar. Fakat geçit taşınabilir küçük bir kutu tarafından kapanmıştır. Robotların görevi kendileri ile beraber geçidin arkasında bulunan diğer küçük kutu veya kutuları geçidin diğer tarafında olan hedef noktaya taşımalarını içermektedir. Robotlar ilk durum için yolun kapalı olduğu bilgisine sahiptir ve küçük kutular robotlardan geride yerde durmaktadır.

Bu deney senaryosunda bir önceki bölümde bahsedilen senaryoya ek olarak plan oluşturmadaki probleme özgü en önemli zorluk, plan arama ağacında engel teşkil eden küçük kutunun *pick* eylemi ile havada olduğu koşulu içeren ortam durumları için yanlış çıkarımlar yapılabilmesidir. Bu şekilde koşullanan bir yöntem, aslında hedefe giden yolun açılmış olduğunu olduğunu varsayarak geri kalan planını buna göre oluşturup tekrarlayan yanlış planların veya sonsuz döngülerin oluşmasına yol açabilir.

Daha ayrıntılı bir analiz yapabilmek için, bu tanıtılan ortamda tek robot ve tek kutu için önerilen yöntemlerin ve hedef planlama (*subgoal planning*) yönteminin başarım analizi yapılmıştır.

İlk durum için *move-to-loc* operatörünün *Hedef*'e gidebilecek eyleminin kapalı (*move-to-loc_locked(Hedef)* olgusu bilgi tabanına eklenmiş) ve hata sebebinin *kutu 2* olarak verildiği durum olduğu varsayılmıştır. Tipleri birer nesne olan kutuların planlama ortamında x, y ve z koordinat bilgileri olmak üzere sadece 3 adet özellikle temsil edildiği varsayımı altında bahsi geçen her 3 yöntem kendi planlama ortamı temsili değişikliklerini yapabildiği ve ara hedef planlama yönteminin ise bu koordinatlardan en az birinin değişimini içeren bir hedef tanımlayabildiği varsayılmıştır. Bu koşullar altında TLPlan ile her 3 yöntem ve ara hedef planlama yöntemi ile üretilmiş karşılaştırılmıştır. Yöntem 1, yöntem 2, ara hedef planlama yöntemi ve son olarak başarılı sonuçlar üreten yöntem 3 kullanılarak yapılmış planlama ortamı temsili değişiklikleriyle üretilmiş planlar sırasıyla Şekil 7.6, Şekil 7.7, Şekil 7.8 ve Şekil 7.9'da görülmektedir. Bu şekillerde operatör parametrelerinde bulunan *r1, k1* ve *k2* ifadeleri sırasıyla *robot 1, kutu 1* ve *kutu 2*' yi simgelemektedir. Planlardaki eylemleri temsil eden kutular, eylemlerin plandaki süreleri ile orantılı resmedilmiş ve plandaki sırası yukarıdan aşağı olarak ve öncelik kısıtları oklarla bağlanarak belirtilmiştir.

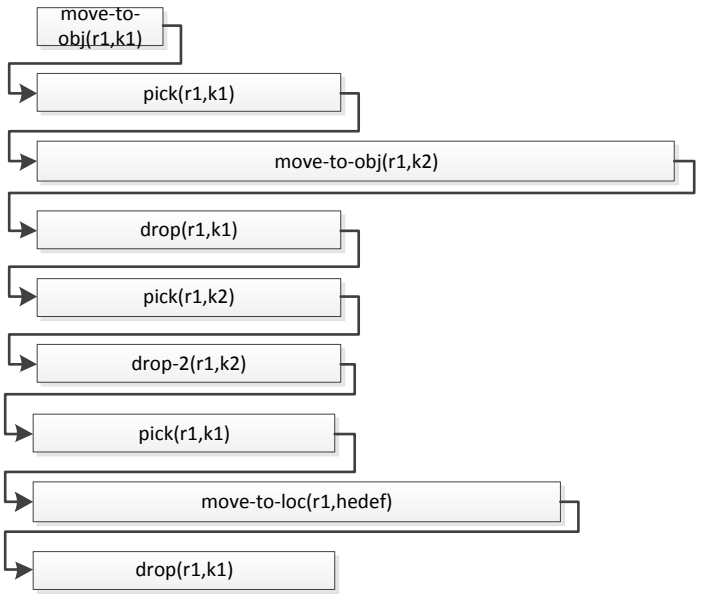


Şekil 7.6 : Yöntem 1 ile ikinci deney kümesi için üretilen plan.

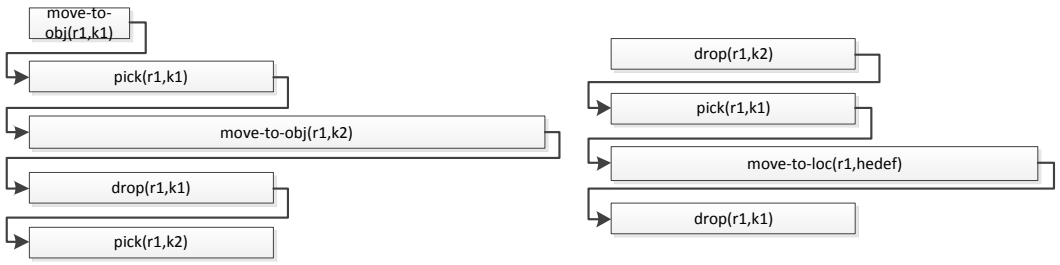
Yöntem 1 kullanılarak yapılmış ortam temsili değişiklikleriyle üretilmiş plan Şekil 7.6'da görülmektedir. Plan aslında hedefe ulaşmasına rağmen mantıksal olarak

hatalıdır. Engel teşkil eden k*utu 2* 'nin kaldırılması anlamına gelen *pick(r1,k2)* eylemi sonrası engelin z ekseni değiştiğinden hedefe giden yol planlama ağacında bu ortam durumu sonrası açılmış olmaktadır. Fakat planın geri kalanında *drop(r1,k2)* eylemi ile engel yerine tekrar konulmuş ve üretilen plan yine yürütülemez hale gelmiştir.

Benzer olarak yöntem 2' de engelin kaldırılıp indirilmesi sonucu geri kalan planın anlamsız hale geldiği görülmektedir.



Şekil 7.7 : Yöntem 2 ile ikinci deney kümesi için üretilen plan.



(a) (b)

Şekil 7.8 : Ara hedef planlama ile ikinci deney kümesi için üretilen plan.

Şekil 7.8' de ara hedef planlama ile yapılan birbiri ardısıra yürütülmesi gereken iki plan görülmektedir. Ara hedef planlamada bu problemdeki hatayı çözmek için önce engelin x, y veya z koordinatlarından birinin değişmesi hedefi için bir plan (a), sonra asıl ulaşılması gereken hedef için ikinci bir hedef için ikinci bir plan (b) üretilir. Fakat şekilden de takip edilebildiği gibi ikinci plan birinci plandan bağımsız olarak

üretildiğinden asıl hedefe ulaşılamamaktadır. Burada planlayıcının hedef olarak engelin z koordinatını değiştirmeyi seçmesinin sebebi zaman maliyetinin düşük olmasındandır.

Diğer yöntemlerden ve ara hedef planlamasından farklı olarak yöntem 3 kilitlenen bir eylemin kilidini hiç kaldırmadığından ve hata sebebi ortadan kalktığında kilitli eylemin bir kopyasını yürütmesi sebebiyle bulunan plan *yürütülebilir* olmaktadır.



Şekil 7.9 : Yöntem 3 ile ikinci deney kümesi için üretilen plan.

Sonuç olarak önerilen 3 yöntem arasından *hatalı operatör kopyalama yöntemi* (*yöntem 3*) olarak adlandırılan yöntemin probleme ilişkin daha etkin çözümler getirdiğinden ve daha geniş bir problem aralığını çözebildiğinden dolayı ortam güncelleme için daha uygundur.

7.3 Benzetim Ortamında Başarım Ananlizi

7.3.1 Benzetim ortamı

Bu bölümde yapılan deneyler profesyonel gezgin robot benzetim yazılımı olan WEBOTS (Michel, 1998) ortamında yapılmıştır. WEBOTS kullanıcıya kütle, sürtünme katsayısı v.b. gibi fiziksel özellikleri ile bir 3B sanal dünyayı hızlı bir şekilde modelleyebilmesine imkan tanır. WEBOTS ayrıca TCP/IP ile üçüncü parti yazılımların kendisi ile haberleşmesine olanak verir (Cyberbotics Ltd, 2010). Böylece üçüncü parti yazılımlar TCP/IP soket bağlantısını kullanarak benzetim ortamına ile bilgi alış verişini sağlayabilir. Deneylerde kullanılan robotlar kol teçhizatına sahip Khepera II robotlarıdır.

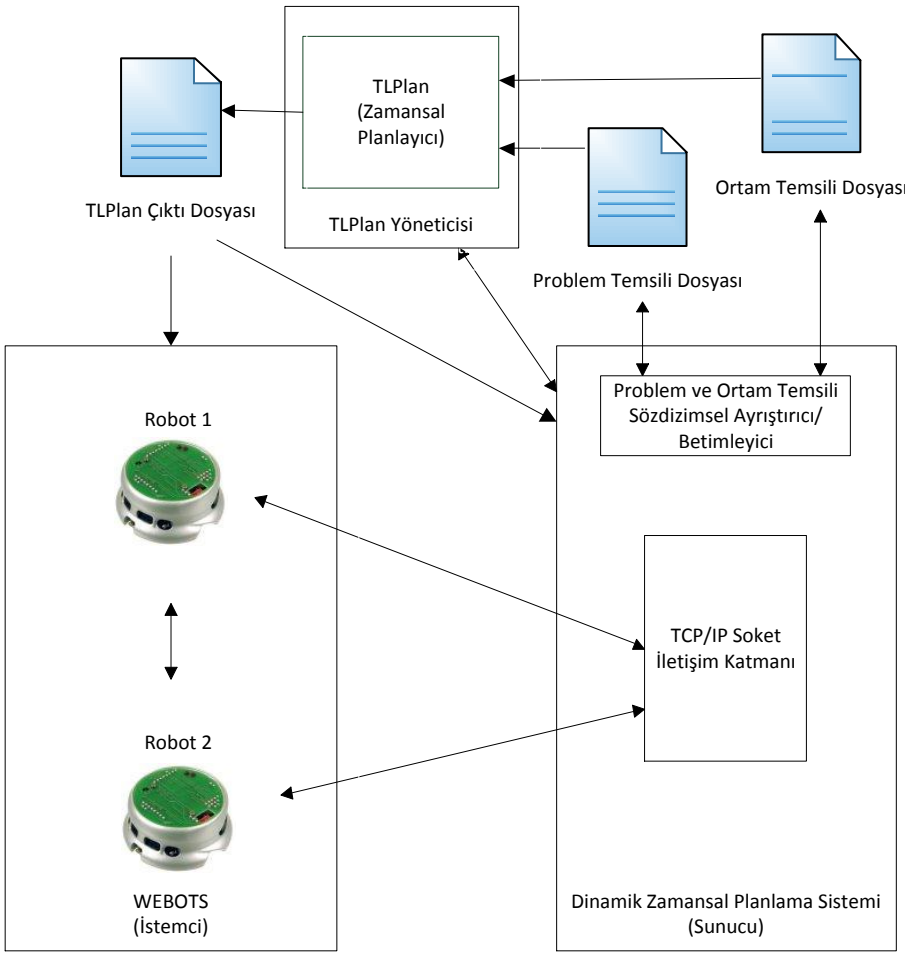
7.3.2 Deney mimarisi

Benzetim ortamı deneylerinde WEBOTS ile dinamik zamansal planlama sistemi arasında TCP/IP soket bağlantısı kullanılmıştır. Haberleşme soket bağlantısı ile istemci/sunucu mimarisine uygun olarak mesajlar üzerinden yapılmaktadır. Mesaj türleri Bölüm 5'te bahsedilen mesajlaşma tasarımı ışığında yapılmıştır. Benzetim ortamından sisteme gelen 5 adet farklı mesaj tipi bulunmaktadır. Bu mesajlar Çizelge 7.2' de açıklamaları ile birlikte listelenmiştir.

Çizelge 7.2 : Benzetim ortamı sistem mesajları yapısı.

Mesaj Tipi	Açıklama
Bağlantı İsteği	Benzetim ortamı tarafından benzetime hazır olduğunda sisteme bağlanmak için gönderilir.
Eylem Başladı	Benzetim ortamındaki bir robotun eylemini başladığını belirtir.
Eylem Sonlandı	Benzetim ortamındaki bir robotun eylemini bitirdiğini belirtir.
Eylemde Hata Oluştu	Benzetim ortamındaki bir robotun eyleminde hata tespit edildiğinde gönderilir.
Eylem Durdu	Benzetim ortamındaki bir robotun eylemini durdurduğunu belirtir.
Plan Hazır	Sistem tarafında bağlantı yapan benzetim ortamına planı ilgili dosyadan okuyabileceğini belirtir.
Sonlan	Sistem tarafından benzetimin sonlanması istenir.

Benzetim ortamı ile yapılan deneyin mimarisi Şekil 7.10' da verilmektedir. Ortam temsili dosyası (*domain file*) ve problem dosyası (*problem file*) TLPlan zamansal planlayıcısının üreteceği plan için girdi dosyaları olup bu dosyalar aynı zamanda sistem açılışında dinamik zamansal planlama sistemi tarafından okunarak problem modeli oluşturulmaktadır. Daha sonra planlayıcı yöneticisinin tetiklenerek planlayıcıdan plan oluşturulması beklenmektedir. Plan arama işlemi sonucu TLPlan yöneticisi tarafından sisteme bildirilir. Eğer plan bulunduysa WEBOTS'un çıktı dosyasını okuyup benzetime başlayabilmesi için sistem WEBOTS'a plan çıktısının okunabileceği anlamına gelen tetiklemeyi gönderir. Plan işleyişi sırasında WEBOTS benzetim ortamından aldığı mesajlar ile sistem güncel durumu gözlenmekte ve tekrar planlama gerektiğinde bu iki dosyayı güncelleyerek planlayıcıyı yeni bir plan üretmesi için tetikleyebilmektedir. Ortam temsili dosyası ve problem dosyasının sistem tarafından güncellenmesi sonrasında planlayıcı yöneticisi tetiklenerek yeni bir plan oluşturulur.

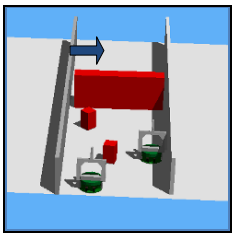


Şekil 7.10 : Benzetim ortamı başarım analizi deney mimarisi.

7.3.3 Başarım analizi

Sunulan problem çözümünü için gerçekçi WEBOTS benzetim ortamında iki gezgin robotlu bir hata çözme senaryosu analiz edilmiştir (Şekil 7.11). Bu senaryo, önerilen çözüm yönteminin testi ve doğruluğunun sınanması için geçici eylem hatasını içerir. Planlama temsil kümesinde altı adet operatör bulunmaktadır: *(move-to-loc ?robot ?loc)* operatörü robotun istenen lokasyona gitmesini, *(move-to-obj ?robot ?obj)* operatörü robotun bir nesnenin yanına üzerinde bir eylem yapmak için konumlanmasını, *(pick ?robot ?smallObj)* ve *(drop ?robot ?smallObj)* operatörleri robotun küçük nesneleri kolu vasıtasıyla sırasıyla yerden kaldırmasını ve yere indirmesini, operator *(push ?robot ?largeObj*) robotun kaldırılamayacağı büyük nesnelerin sürükleme yöntemi ile ittirerek konumunu değiştirmesini *(paint ?robot ?obj ?color)* operatörü ise bir nesneyi boyaması için gerekli olan operatörlerdir.

Planlama probleminin hedefi kırmızı nesnelerin yolu bloke eden büyük nesnenin arkasındaki hedef noktaya taşımaktır. Fakat robotlar koridoru bloke eden engelin varlığının bilgisine başlangıç aşamasında sahip değillerdir



Şekil 7.11 : Benzetim programında bir hata çözme senaryosu ortamı

Planın yürütülme esnasında robotlardan biri *move-to-loc* eylemini yürütürken engeli tespit eder ve yol planlayıcısı şimdiki noktadan hedef noktaya yol bulmanın imkasız hale gelmesinden ötürü hata döndürür. Hataya uğrayan *move-to-loc* eyleminin hata sebebi olarak *engel* adlı nesne harita üzerindeki tahmini hesaplanan yer bilgisi ve var olabilecek diğer özellikleriyle (örnek için renk bilgisi) ile rapor edilir. Bu yeni bilgiler bilgi tabanına girilir ($\boldsymbol{P_r}'$). Eğer bu aşamada (yürütme hatası ile karşılaşılmış ve elde edilen bilgiler bilgi tabanına girilmiş durumda iken) mevcut plan, planlayıcı tarafından hala *geçerli* bulunmuyorsa, tekrar plan üretilerek yürütmeye devam edilir. Bu örnek senaryoda bu hatanın nasıl çözüleceğine dair sistemde bulunan bir bilgi olmadığı varsayılmıştır. Hedefe giden yolu açacak bir eylem bulunmadığından planlayıcı bu bilgi tabanı ile hedefe ulaşamaz. Söz gelimi push operatörü "hedefe giden yolu aç" diye bir etkisi olmadığından planlayıcı tarafından yeni bir plan oluşturulamaz. Yeniden planlamanın yapılamadığı, yapılsa dahi *yürütülemez* planların oluştuğu böyle bir durumda sunulan tanım kümesi güncelleme yöntemi UPDATEDOMAINDESCRIPTION yordamı sayesinde alternatif planların bulunmasına yardımcı olur. Bu yordam tanım kümesine yeni bir sözde operatör etkleyerek planlama problemini $\boldsymbol{P_r}'$' den $\boldsymbol{P_r}''$' ne dönüştürür. MAKEFAILUREPRECONDITIONS yordamı, tanım kümesinde hata sebebinin özelliklerini değiştirebilen operatörleri arar ve engel nesnesinin özelliklerini değiştirebildikleri için *push* ve *paint* operatörlerini hata ile ilgili operatörler olarak bulur. Hem hatalı eyleme (*move-to-loc*) hem de büyük nesnelerin *x* ve *y* kordinatları ve rengini değiştirebilen sırasıyla push ve paint operatörlerinin etkilerine bağlı olarak *pseudo1* adında yeni bir sözde operatör yaratır. Ayrıca hatalı eylemin planlayıcı tarafından tekrar seçilmemesi için kilitleyen olgu

(*move-to-loc_locked*), planlama tanım kümesine eklenirken, bu kilidi açma etkisi de sözde operatörün etkisi olmasıyla tanım kümesi güncellenmiş olur. Hata durumu sonrası tanım kümesi güncellemesi ile eklenen sözde eylemi de içeren değişiklik kümesi Şekil 7.12' de verilmiştir.

```
(def-adl-operator (move-to-loc ?loc)
  (pre
       …
      (not (move-to-loc_locked ?loc))
  )
  …
)
(def-adl-operator (pseudo1 ?obj ?loc)
  (pre
      …
     (and
        (= ?obj obstacle)
        (move-to-loc_locked ?loc)
        (or
          (not (xcoord ?obj obs_x))
          (not (ycoord ?obj obs_y))
          (not (color ?obj red))
        )
     )
  )
  (del (move-to-loc_locked ?loc))
)
```

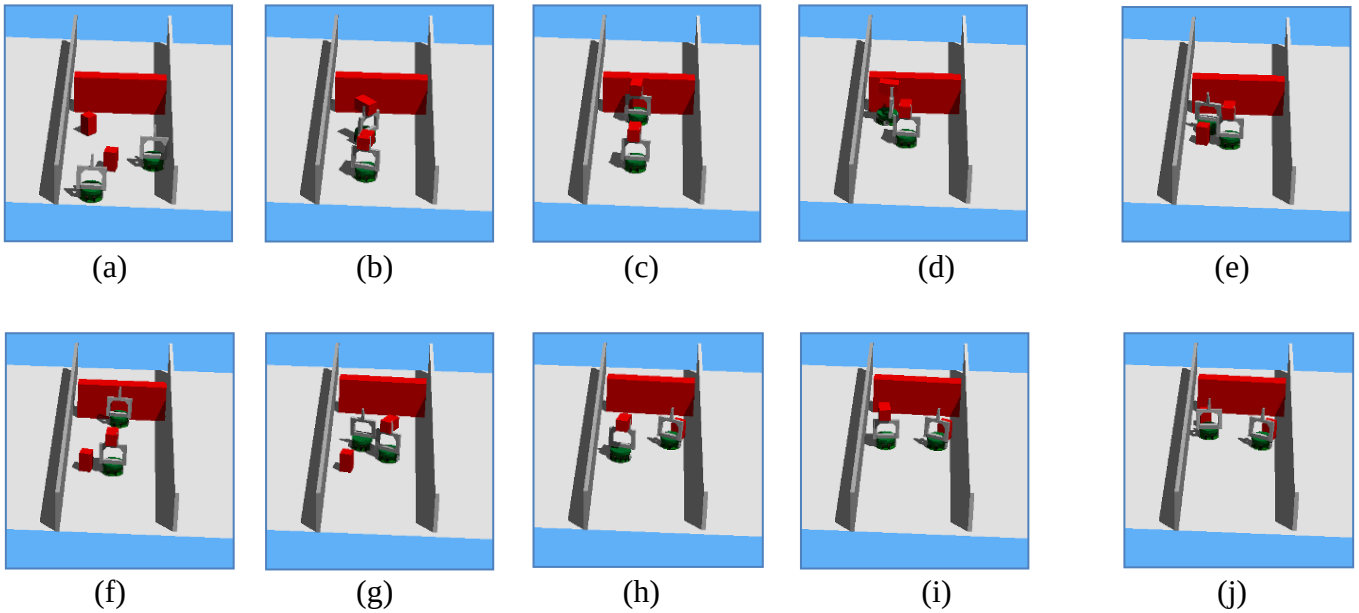
Şekil 7.12 : Benzetim ortamı deneylerindeki hatanın çözümü için sözde operatör yöntemi tanım kümesi güncellemesi.

Üretilen ilk planın ve oluşan yeni planının Webots simülatöründeki işleyişini gösteren kesitler Şekil 7.13' te verilmiştir. (a) ve (c) arası ilk plan yürütülmektedir. (d)' de hata yüzünden tanım kümesi değişikliği oluşmuş ve yeni bir plan üretilmiştir. (e) ve (j) arası kırmızı nesnelerin hedef konumlarına taşınmadan önce engelin uygun bir yere ittirilmesini içeren yeni plan yürütülmektedir. İlk ve hata sonrası tanım kümesi güncellemesi ardından oluşturulan plan Şekil 7.14' te gösterilmektedir. Yeni plan yöntem 1' deki planlama problemi temsili güncellemesine göre oluşturulmuş sözde operatör içermektedir.

Tanım kümesi bilgisi güncellemesi ile yeni bir *yürütülebilir* plan oluşturulmuştur. Yeni plan birinci robot için bir *push* eylemi içerir. İkinci robot ise hedefe doğru olan bir noktaya (konum bilgisi içeren nesneye) yönlendirilir. Sonuç planının maliyetleri açısından *paint* operatörü içeren plan *push* operatörü içerenden daha maliyetli olduğundan dolayı sonuç planı olarak *push* operatörü içeren planı görülmektedir. Fakat eğer maliyetler tam tersi yönde olsaydı *push* operatörü içeren plan üretilcekti.

Hata özelinde bir çıkarsama yapılmadığından bu defa robot ilk olarak engeli boyamaya çalışacaktı. Daha sonra yolun açıldığını varsayan bu plan ile robot bir kere daha hedef noktaya giden eylemi yürüttüğünde hata oluşacaktı. Hataya özel daha çok çıkarsamanın yapılması mümkünse, sözde operatöre alakasız özelliklerin konması engellenerek push operatörünü içeren planın seçilmesi zorlanabilirdi. Gerçeklemede bu değer koridorun boyuna göre hesaplanan bir değerdir. Dikkate alınması gereken bir nokta olarak *push* eyleminin süresinin plan öncesi statik olarak verilmiş olmasıdır.

Webots ortamında oluşturulan benzetim sonucu artımlı olarak oluşturulmuş topolojik harita Şekil 7.15' te verilmiştir. Graftaki düğümler, robot tarafından algılanan veya yeri önceden bilinen taşınacak olan cisimleri, diğer robotu, gidilebilecek alanları ve robotun yön değiştirdiği koordinatları temsil etmektedir. Graftaki ayrıtlar ise robotun aynı hızda izlediği yolu temsil etmektedir. Senaryonun başlangıcında robotlar engel varlığından habersiz oldukarından dolayı tam bağlı bir graf bulunmaktadır (a).Plan koşturulduktan sonra robot cisimleri alıp bu grafa göre yoluna devam ediyorken engele rastladığında başlangıç grafında bulunan ayrıtın kullanılmadığını *bug algoritmasını* kullanarak anlamakta ve grafta gerekli güncellemeyi yaparak (b) graf yeniden oluşturulmaktadır. Bir diğer söylemle robotun izlediği yol planlaması takip edildiğinde hedefe varılamıyorsa yeni bir yol planlamasına ihtiyaç duyulmaktadır. Bu yüzden gerekli güncellemeler yapılmaktadır. Hata durumundan sonra engelli ortama göre yeniden plan üretilir ve yürütmeye başlanır. Yeni plan sonucu hedefe ulaşılabilir olduğu anlaşıldıktan sonra (c)'de görülen graf elde edilir.



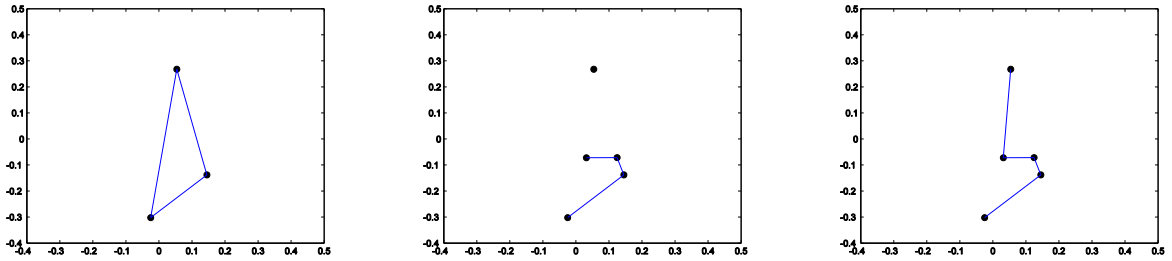
Şekil 7.13 : Nesnelerin hedef noktalarına taşınmasını içeren tüm planın işleyişi.

ilk plan	**robot1**	move-to-obj (robot2, obj2)	pick (robot2, obj2)	move-to-loc(robot2, target)		drop(robot2, obj2)
	robot2	move-to-obj (robot1, obj1)	pick (robot1, obj1)	move-to-loc(robot1, target)	drop(robot1, obj1)	

pseudo1 (obstacle, target)

yeni plan	**robot1**	move-to-obj (robot1, obstacle)	drop (robot1, obj1)	push (robot1, obstacle)	move-to-obj (robot1, obj1)	pick (robot1, obj1)	move-to-loc(robot1, target)	drop (robot1, obj1)
	robot2	move-to-obj (robot2, obstacle)			move-to-loc (robot2, target)	drop (robot2, obj2)		

Şekil 7.14 : Robotların ilk ve hata sonrası planlama temsili güncellemesi sonrası yeni planları.



(a) (b) (c)

Şekil 7.15 : Yaratılan ve artımlı olarak güncellenen topolojik haritalar.

8. SONUÇ VE İLERİ ÇALIŞMALAR

Gezgin robotların birbirlerine bağımlı ve karmaşık görevleri etkin şekilde yürütebilmesi için otonom eylem planlama yöntemlerinin kullanılması şarttır. Fakat gerçek dünya problemleri, planlayıcıların ayrıntılarını tümüyle çözemeyecekleri kadar karışık olabilmektedir. Bu durum, gerçek dünya problemlerinin planlayıcıların çözebileceği şekilde soyutlanarak daha az karmaşık seviyede temsil edilmesini gerektirmektedir.

Gerçek dünya belirsizlik ve dinamizm içermektedir. Robotlar ortam bilgisine tümüyle sahip olmadıklarından beklenmeyen hatalar ile karşılaşabilirler. Bu hatalar planlayıcının kullandığı soyutlanmış modeller içinde çözülemez görülebilir. Bir diğer ifade ile tanım kümesi temsili tam ve eksiksiz olmadığında, dinamik gerçekleşebilecek olaylar karşısında mümkün olduğu halde alternatif planlar üretilemeyebilir.

Bu tezde bu problemi çözmek üzere dinamik gerçekleşebilecek hatalar karşısında alınan hata bilgisi ile birlikte eldeki problem modelini birleştiren ve planlayıcıyı bu doğrultuda yönlendirilerek yürütülebilir planlar bulunmasını sağlayan yöntemler geliştirilmiştir. Bu yöntemler kendi aralarında ve ara hedef planlama yöntemi ile kıyaslanmış ve aralarından önerilen yöntemin ara hedef planlama yöntemine göre daha geniş bir problem aralığını çözebildiği gösterilmiştir.

Çizelgeleme ve planlama eşzamanlı yapılıp temel zamansal planlayıcı olarak TLPlan planlayıcısı kullanıldığından robotların varolan bilgileri ile toplam plan süresi eniyilemesi yapılmıştır. Gerçekçi çoklu robot simülatörü Webots ve TLPlan planlayıcısı ile haberleşen merkezi bir dinamik zamansal planlama sistemi yapısı oluşturulmuştur. Gezgin robotları içeren bir hata çözme senaryosu Webots simülatöründe sınanmış ve oluşturulan sistemin verilen gerçekçi senaryo üzerinde başarımının olumlu olduğu gösterilmiştir.

İleri çalışmalar olarak hata sebebinin doğrudan belirlenemediği durumlarda hata durumunu analiz ederek çıkarsayan bileşenlerin robotlara yüklenmesi mümkün

olabilir. Hareket kabiliyetini artırmak ve bağımlılığı azaltmak amacıyla planlayıcı, merkezi yapıdan robotlar üzerinde yer alan dağıtık bir yapıya geçirilebilir.

KAYNAKLAR

Alami, R., ve Botelho S. C., 2001: Plan-based multi-robot cooperation. In Advances in Plan-Based Control of Robotic Agents.

Amir, E., ve Chang A., 2008: Learning partially observable deterministic action models. In Journal of Artificial Intelligence Research 33(1): 349-402.

Bacchus, F., ve Addy, M., 2001: Planning with resources and concurrency: A forward chaining approach. Internal Joint Conference on Artificial Intelligence, Seatle, USA.

Blum, A. L., ve Furst M. L., 1995: Fast planing through planning graph analysis. Procedings of International Joint Conference on Artificial Intelligence, pp. 1636-1642.

Blum, A. L., ve Langford J., 1999: Probabilistic planning in the Graphplan framework. Procedings of ECP.

Brenner, M., ve Nebel, B., 2009: Continual planning and acting in dynamic multiagent environments. In AAMAS 19(3):297-331.

Cao, H., Lacroix, S., Ingrand, F., ve Alami, R., 2010: Complex Tasks Allocation for Multi Robot Teams under Communication Constraints. 5th National Conference on Control Architectures of Robots, Douai, May 18-19.

Chen, Y., Wah, W., ve Hsu, C., 2006: Temporal planning using subgoal partitioning and resolution in SGPlan. JAIR, 26, 323–369.

Cushing, W., ve Kambhampati, S., 2005: Replanning: A new perspective. Procedings of International Conference on Automated Planning and Scheduling Poster Program.

Cushing, W., Benton J., ve Kambhampati, S., 2008: Replanning as a deliberative re-selection of objectives. Arizona State University Technical Report.

Cyberbotics Ltd, 2010**:** Webots User Guide 6.3.0.

de Jonge, F., Roos, N., ve Witteveen, C., 2009: Primary and secondary diagnosis of multi-agent plan. In AAMAS, 18(2): 267-294.

Dearden, R., Meuleau, N., Ramakrishnan, S., Smith, D. E., ve Washington, R. 2003: Incremental contingency planning. In Proc. Int. Conf. on Applied Planning and Scheduling (ICAPS) Workshop on Planning under Uncertainty.

Dias, M. B., ve Stenz, A., 2002: Opportunistic optimization for market-based multirobot control. In Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS), 2714–2720.

Fox, M., Gerevini, A., Long, D., ve Serina, I., 2006: Plan stability: Replanning versus plan repair. Procedings of International Conference Autonomated Planning and Scheduling, pp. 212-221.

Ghallab, M., ve Laruelle, H., 1994: Representation and control in IxTeT, a temporal planner. Proceedings of 2nd International Conference on AI Planning Systems, AAAI Press, Menlo Park, CA, USA.

Gil, Y., 1994: Learning by experimentation: Incremental refinement of incomplete planning domains. In Proceedings of the Eleventh Intl. Conference on Machine Learning (ICML).

Göbelbecker, M., Keller, T., Eyerich, P., Brenner, M., ve Nebel, B., 2010: Coming up with good excuses: What to do when no plan can be found. In ICAPS, 81-88.

Halsey, K., Long, D., ve Fox, M., 2004: CRIKEY - a temporal planner looking at the integration of scheduling and planning. In Workshop on Integrating Planning into Scheduling ICAPS, pages 46–52, 2004.

Kalisch, M., ve König, S., 2005: Comparison of STRIPS, ADL and PDDL, Introduction to AI planning, Lecture Notes, Bamberg University.

Little, I., ve Thiebaux, S., 2006: Concurrent probabilistic planning in the graphplan framework. Proceedings of International Conference on Automated Planning and Scheduling.

Micalizio, R., 2009: A distributed control loop for autonomous recovery in a multi-Agent plan. In Proc. of the 21th. Int. Joint Conf. on Artificial Intelligence (IJCAI), 1760-1765.

Michel, O., 1998: Webots: Symbiosis Between Virtual and Real Mobile Robots. In Proceedings of the First International Conference on Virtual Worlds.

Ranganathan, A., ve Dellaert, F., 2009: Bayesian surprise and landmark detection. In Proceedings of the IEEE international conference on Robotics and Automation, 1240-1246.

Russell, S., ve Norvig, P., 2003: Artificial Intelligence: A Modern Approach, Prentice Hall.

Sariel, S., Balch, T., ve Erdogan, N., 2007: Incremental Multi-Robot Task Selection for Resource Constrained and Interrelated Tasks. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2314-2319.

Schermerhorn, P., Benton, J., Scheutz, M., Talamadupula, K., ve Kambhampati, S., 2009: Finding and exploiting goal opportunities in real-time during plan execution. In Proc. of IROS, 3912-3917.

Simmons, R., ve Koenig, S., 1995: Probabilistic Robot Navigation in Partially Observable Environments. Int. Joint Conf. on Artificial Intelligence (IJCAI), 1080-1087.

Talamadupula, K., Benton, J., Schermerhorn, P., Kambhampati, S., ve Scheutz, M., 2009: Integrating a closed world planner with an open world robot: A case study. In Proc. of Twenty-Fourth AAAI Conference on Artificial Intelligence: 1561-1566.

van der Krogt, R., ve de Weerdt, M., 2005: Plan repair as an extension of planning. In Proc. of the 15th. Int. Conf. on Automated Planning and Scheduling (ICAPS), 161-170.

Wang, X., 1995: Learning planning operators by observation and practice. In Proceedings of the Second International Conference on AI Planning Systems (AIPS): 335-340.

Witteveen, C., van der Krogt, R., Roos, N., ve de Weerdt, M., 2005: Diagnosis of single and multi-agent plans. In Proc. of 4th Int. Conf. on. Agents and Multiagent Systems (AAMAS), 805-812.

ÖZGEÇMİŞ



Ad Soyad: Celal Uğur Usuğ

Doğum Yeri ve Tarihi: Malatya, 15 Ağustos 1985

Lisans Üniversite: İstanbul Teknik Üniversitesi

Yayın Listesi:

- **Usug, C. U., ve Sariel Talay, S.,** 2011: Dynamic Temporal Planning for Multi Robot Systems. AAAI Workshop on Automated Action Planning for Autonomous Mobile Robots (PAMR), San Francisco, USA.