1. GİRİŞ

Robotik dünyası gerçek dünyada otonom olarak çalışan ve insan müdahalesinin en az olduğu robot sistemleri geliştirmek için çalışmaktadır. Otonom kelimesi etkinlik ve güvenirlik özelliklerini de içinde barındırmaktadır. Bu sebeple, robot dünyası çalışanları belirsizlik ve beklenmedik olay durumlarında çalışabilen sistemler geliştirmek üzere çalışmakta, bu amaçla mimariler oluşturmaktadırlar. Bu amaçla oluşturulan mimariler genellikle seviyeli bir yapı taşımaktadır. Her mimarinin seviye sayısı farklılık göstermesine karşılık genelde ortak olan alt seviyelerin daha detay işleri, üst seviyelerin ise "üst seviyede soyutlama – high level of abstraction" olarak adlandırılan daha genel işleri ele alıyor olmalarıdır.

Genel amaçlı bir robot sisteminden beklenenler dikkate alındığında ancak 1998'li yıllardan sonra gelişmelerin kaydedilmeye başlandığı bu seviyeler iş seviyesinde "bir araya getirme" (assembly) planlaması ve hareket seviyesinde "engellerden sakınma" (obstacle avoidance) problemlerine sınırlı şartlar altında çözüm bulmuşlardır.

Kontrol seviyesinde robot kolları için çok detaylı kontrol stratejileri geliştirilmiş, robot kolu dinamik modeli tüm detayları ile çalışılarak robotun çok hızlı ve hassas çalışabilmesi için teorik alt yapı olgunlaştırılmıştır.

Donanım seviyesi ise arzu edilen düzeyde hızlı gelişememiştir. Özellikle hassas hareket yeteneğine sahip elektrik motoru tahrikli robotlarda robotun taşıyabileceği yüke karşılık kendi ağırlığı oranı düşük düzeyde kalmış, almaç (sensör) teknolojilerindeki gelişmeler, özellikle

kuvvet-buru (tork) ve robot görüntü (robot vision) almaçlarının kısıtları ile sınırlı kalmıştır.

Seviyeler arası arayüzlerde gelişmeler de 1998'li yıllardan sonra gelişmeye başlamıştır ve basit veri dosyası transferi ötesinde grafik tasarımlar ve gerçek zamanlı veri aktarımları gerçekleştirilmiştir. Kapalı çevrim modeli uzun bir süre en alt iki seviye arasında kullanılmış ve 1996'dan sonra da donanım ve hareket planlama seviyeleri arasında kullanılmaya başlanmıştır (Xi, Tarn, and Bejczy, 1996).

Daha sonraki çalışmalarda aktif robot görüntü stratejileri, geri besleme bilgisini görüntü desteği ile örnekleme frekansında yapmayı önermektedirler. Ancak çok yoğun hesaplamaların yer aldığı bu tür stratejilerin örnekleme frekansında kontrol terimi üretmesi için oldukça önemli sadeleştirmelere gidilmekte kullanılan modellerin basite indirgenmesi gibi çözümler üzerinde durulmaktadır. Bu tür sadeleştirmeler iş seviyesinin gereksinimleri dikkate alınmadan yapıldığı için sistemin iş seviyesinde sağlıklı alternatif planlar üretmesi kısıtlanmaktadır. Mevcut mimariler beklenmedik olay durumlarını mümkün olduğunca alt seviyelerde çözmeye çalışmakta, yüksek seviyelerde çalışmanın sağlayacağı fırsatlardan yararlanmamaktadırlar. Örneğin Subsumption Mimarisi (Arkin, 1999) olarak adlandırılan mimari her seviye için davranışlar (behaviours) tanımlamakta, beklenmedik olayları çözmek için alt seviyelerdeki davranışları kullanmaktadır. Eldeki alt seviyeli davranışların problemleri çözememesi durumunda ise alt seviye davranışları çeşitlendirilmektedir.

Robotların pratikte yaygın olarak kullanılabilmesi için iş seviyesinde alacakları komutları belirli bir esneklik ile gerçekleştirebilme yeteneklerinin olması kaçınılmazdır. Üst seviyelerin kapalı çevrim model

içerisinde yer alması robot yeteneklerini son derece arttıracaktır. Örneğin bir cismin bir yerden alınıp başka bir yere konması bir iş olarak tanımlandığında, cisim herhangi bir nedenle yarı yolda robot tarafından düşürüldüğü durumda, klasik denetleyici politikalar robotun sanki cismi hala taşıyormuş gibi hareketini tamamlayıp ondan sonra sistemin durumu incelemesi stratejisini izlemektedirler. Böyle bir düşme olayı meydana geldiğinde iş seviyesine derhal müdahale etme gereği açıkça görülebilir.

Bu çalışmada sabit bir kamera (robot kolu üzerine monte edilmemiş) ve eklem üzerindeki optik kodlayıcı bilgileri kullanılarak iş seviyesinde alınan komutları kapalı çevrim model ile gerçekleştirebilen bir akıllı robot planlayıcısı geliştirilmiştir. Bu planlayıcının temel yapısı Şekil 3.2'de verilmiştir.

Önerilen iş planlayıcısı ürettiği iş seviyesi komutların geçerliliğini görüntü almacından alınan bilgiler ışığında kontrol edebilecek yeteneğe sahiptir. Görüntüden gelen bilgileri bir geri besleme bloğundan geçirerek kendisi için gereken yeterli detay bilgiye sahip olabilmektedir. Görüntü bilgilerinin iş planlayıcısının içinde çalıştığı dünya modeli üzerine uygulanabilmesi için zaman değişkenine ek olarak yeni değişkenlerin yaratılması kaçınılmazdır. Örneğin tanımlanan işi en az enerji ile gerçekleştirme veya en az alt iş dizini ile gerçekleştirme kriterleri zaman kriterine alternatif olabilmektedir.

Gerçek hayata uyarlanabilen robot görüntü problemlerinin çözümleri genellikle duruma bağlı (case dependent) varsayımlardan yola çıkarak gerçekleştirilmektedir. Diğer bir deyişle, önerilen sistemin tanımlanabilecek tüm ortamlarda aynı başarım ile çalışması mümkün görünmemektedir. Örneğin fabrika ortamında taşıyıcı bant üzerinde giden farklı makine parçaları basit bir dünya modeli oluştururken,

birbirini engelleyen şekilde yığılmış cisimler (occluded objects) tarafından oluşturulan model, görüntü tanıma konusunda karmaşık bir çözüm gerektirebilir.

Bu sistemde kullanılan dünya modeli çok karmaşık olmamasına rağmen iş planlayıcı bloğunda çalışan yapay us algoritmaları ve geri besleme bloğunun sisteme entegrasyonu ciddi problemler olarak ortaya çıkmaktadırlar.

Yukarıda anlatılanlar ışığında bu çalışmada gerçekleştirilmiş işler şu şekilde sıralanmaktadır:

Bir robot çalışma uzayında genel amaçlı robot iş planlama yapabilen ve planların işletimi sırasında dış aracıların sebep olacağı beklenmedik olayları ele alan bir mimari geliştirilmiş ve bu mimarinin blok dünyasında tanımlanabilecek robot işleri için uygulaması yapılmıştır.

Beklenmedik olayları bir görme sisteminden gelen geri besleme ile sembolik üst seviyede (planlama seviyesi) yakalayan ve ele alan görme geri beslemeli bir yeniden planlama mekanizması geliştirilmiştir.

Görme sisteminden gelen bilgiler üst seviyede doğrudan kullanılamıyacağından, görüntü bilgilerini üst seviyenin anlayabileceği şekle dönüştüren bir geri besleme mekanizması oluşturulmuştur.

Görme geri beslemeli yeniden planlama mekanizmasına, öğrenen ve zaman içerisinde elde edilen tecrübeleri kullanarak beklenmedik olaylar durumunda zaman geçtikçe daha iyi kararlar veren bir nitelik kazandırılmıştır.

Görme geri beslemeli yeniden planlama kavramı bu çalışmanın başlaması ile ortaya atılmış bir kavramdır. Daha sonra 1998'de (Haigh, 1998) çalışmasında yeniden planlamanın görme geri beslemeli yapılabileceğinden bahsedilmiş ancak sistematik bir mekanizma ortaya koyulmamıştır. Bu çalışmada ise görme sisteminden elde edilen bilgiler bir geri besleme bloğundan geçirilip görme seviyesine iletilmektedir. Bu şekilde görme bilgisinden faydalanılarak çalışma uzayı içerisinde bir beklenmedik olay olup olmadığı anlaşılmakta ve eğer olmuş ise karar mekanizması ile ele alınmaktadır.

Yeniden planlama mekanizması içerisinde yer alan öğrenme metodu bu çalışma kapsamında geliştirilmiş ve orijinaldir. Bu metod alternatif hareket dizinleri arasından optimal enerjiyi gerektiren dizini seçen alternatif tabanlı bir mekanizmadır.

Önerilen mimari kullandığı seviye sayısı itibariyle 3-Tier olarak adlandırılan mimariyi taban alan, planlama, yeniden planlama (beklenmedik olayları görüntüden yakalayıp iş seviyesinde bu bilgiyi kullanarak mevcut planları kısmen ya da tamamen değiştirme) yapabilen ve önceki tecrübelerden yararlanarak öğrenebilen bir mimaridir. Robot işleri söz konusu olduğunda, üst seviyede beklenmedik olaylarla başa çıkma özelliğine sahiptir.

Bu çalışmada beklenmedik olaylarla planlama seviyesinde baş etmenin etkinliği arttıracağı düşüncesi ile yola çıkılmış ve robotların yine üst seviyede öğrenerek otonom ve etkin olarak beklenmedik olaylarla başa çıkabileceği bir sistem gerçekleştirilmiştir. Bu çalışmalar sırasında beklenmedik olayların yan etkileri yanında olumlu etkilerinin olabileceği de göz önünde bulundurulmuştur.Tezin izleyen bölümleri şu şekilde organize edilmiştir: Bölüm 2'de planlamanın tarihçesi ve öğrenme

metodları anlatılmıştır. Bölüm 3'de alan bağımsız (domain independent) planlayıcı seviyesi ve Bölüm 4'de ise alan bağımlı seviyeden bahsedilmiştir. Bölüm 5'te görme sistemi tanıtılmıştır. Bölüm 6'da ise sonuç ve tartışma yer almaktadır. Oneriler de bu bölümde verilmektedir.

2. PLANLAMANIN TARİHÇESİ

2.1 Literatürdeki Planlayıcılar ve Kullandıkları Problem Gösterimi ve Çözümü Metodları

Yapay us planlamanın amacı, bir aracının hedeflerini gerçekleştirebilmesi için bir eylem dizisi sentezlemesini sağlayacak kontrol algoritmalarının oluşturulmasıdır. Robot planlama araştırmacıları yapay us planlama metodlarını robot planlama metodları olarak kullanma eğilimi göstermektedirler. Şu an için tamamen yapay us planlaması çalışmaları yapan araştırmacılar olduğu gibi tamamen robot planlaması yapan araştırmacılar da vardır.

Planlama problemlerinin çözümde en yeni, hızlı, sistematik ve stokastik SAT (SATisfiability) algoritmalarını kullanabilmek için problemlerin "propositional formulae" gösterimine derlenmesi oldukça ilgi çekmiştir (Weld, 1999). Bu algoritmalar kısıt sağlama ve arama teknolojisindeki son gelişmelerden etkilenmişlerdir ve oldukça iyi performans seviyeleri vardır. Örneğin, BLACKBOX planlayıcısı (Kautz and Selman, 1998) 10^{16} muhtemel durumdan oluşan 105-eylemlik bir lojistik planı sadece 6 dakikada oluşturmaktadır.

Buna karşılık klasik olmasına rağmen yeni birçok planlayıcıda kullanılan STRIPS (Fikes and Nilsson, 1971) gösteriminde, her eylem bir dünyadan diğer bir dünyaya uzanan bir geçiş fonksiyonunu tanımlayan bir birleşik (conjuctive) önkoşul ve birleşik etki kümesi olarak belirtilir. Bir dünyada bir eylemin işletilmesi, dünyanın durum tanımını almak ve yol üstündeki çelişkili hazır bilgileri (literals) yok ederek her hazır

bilgiyi, eylemin etki bileşenlerinden (conjuction) alarak mevcut duruma eklemek olarak tanımlanır.

Planlama probleminin basit bir formülasyonu üç girdi içerir:

1. Dünyaya ait başlangıç durumun önermeler hesabı (predicate calculus) türünde formal bir dilde tanımlanması.

2. Aracı hedeflerinin, örneğin aracının planlayıcıdan belli bir davranışı gerçekleştirmesini beklemesi gibi, formal bir dilde tanımlanması.

3. Önkoşulları eylemlere taşıyacak işleçler.

Yukarıda belirtildiği şekilde bir formülasyon yaptıktan sonra, bir planlama, tanımlanmış bir işi gerçekleştirecek dizini bulana kadar bir muhtemel eylemler uzayında arama yapmak olarak tanımlanabilir (Luger and Stubblefield, 1993).

Bazı yakın tarihli çalışmalar planlama problemlerinin çözümlerinde "dinamik CSP" (Constaint Satisfaction Problems) kullanır. Bunun sebebi CSP ile planlama problemleri arasında gözlenen benzerliklerdir (Falkenhainer and Forbus, 1998). Dinamik CSP bir kısıt sağlama problemidir ve değişkenler kümesi ve ilgili kısıtlar daha önceki değişkenlere ait değerlerin seçimine bağlı olarak değişirler. Bu aynı zamanda literatürdeki bazı problemlerdeki "hesaplama takip edilemezliği"nin de sebebidir. Sonuçta tüm değişkenlere değer atanır ancak etkinlik açısından hangi değişkene önce değer atandığının çok büyük önemi vardır. Genelde, en az olası ve çelişmeyen değeri olan değişkeni seçmek iyi bir sezinlemedir. Bu çalışmada çalışılan karışmış

taşlar probleminde, değişkenler birbiri ile çelişmemektedir ancak üzerinde durulan konu optimal maliyeti verecek olan bir eylemler dizisi bulmaktır.

Bir planlayıcı bir problemi, problemin optimizasyon isteklerinden dolayı çok özel bir algoritma ile çözebileceği gibi çok sayıda çalışma uzayına ait problemleri tek bir çıkarsama mekanizması ile de çözebilir. İki çözüm alternatifinin kombinasyonundan kısıtlar ve kontrol kuralları ile tanımlanabilen ve hesaplamada takip edilebilirliği olan problem çözümleri ortaya çıkmaktadır.

Bu çalışmada kullanılan problemler için kısıtlar ve kontrol kuralları cinsinden tanımlanabilen optimal ya da alt optimal çözümler bulunmamaktadır. Optimal çözüme ulaşmak üzere literatürdeki metodların bir katkısı olmadığı gözlenmiştir.

STRIPS tabanlı gösterimlerde, bir planlayıcı, işleçlerden ve arama kontrol kurallarından oluşan bir iş modelinin oluşturulması olarak da tanımlanabilir.

Bu çalışmada kullanılan planlayıcı STRIPS tabanlı gösterimi kullanmakta olup bir işleçler ve arama kontrol kuralları kümesine sahiptir. Kullanıcı arayüzü, kullanıcıdan planlanması istenen probleme ait başlangıç ve hedef durumların alınmasını sağlar. Sonraki işlem ise bir plan üretip işletmektir.

Tarih sırasına göre literatürdeki planlama sistemlerine bakılacak olursa Shakey (Nilsson, 1984), bir robot üzerinde planlama sistemi kullanan ilk sistemdir. Bu proje gerçek dünya belirsizliğini göz ardı eden bir klasik planlayıcı üzerine oluşturulmuştur (Fikes, Hart and Nilsson,

1972) ve işletilebilir bir planı oluşturmak için kararlı bir model kullanmıştır. İşletim hataları oluştuğunda ise yeniden planlama modülü uyandırılmaktadır.

Bu öncü yaklaşım kısmen başarılı olarak kabul edilse de, refleks hareketler (reaktivity) içermiyor olması açısından eleştirilmiş ve gerçek dünyanın belirsizliğini ele alan planlama sistemleri konusunda yapılacak çok sayıda araştırmaya neden olmuştur.

Araştırmacılar, 1990'dan sonraki yıllarda klasik planlama metodlarına sadık kalarak mükemmel uzay bilgisi varsayımlarını esnekleştirme yönünde çalışmalar yapmaya başlamışlardır (Peot and Smith, 1992; Etzioni, Hanks, Weld, Draper, Lesh and Williamson, 1992; Pryor and Collins, 1993; Draper, Hanks and Weld, 1994).

Bu çalışmalar arasında olan şartlı planlama yöntemi çalışma uzayı içerisindeki bütün olası beklenmedik olayları dikkate alan bir yaklaşımdır ve plan işletilmeden önce akla gelebilecek her beklenmedik olay için bir planlama yapar (Atkins, Durfee and Shin, 1996; Mansell, 1993; Pryor, 1994; Schoppers, 1989). Şartlı planlama, (Pryor and Collins, 1996)'da belirsiz hata durumları ile başa çıkma yöntemlerinden bir tanesi olarak sunulmaktadır. Bu planlama türünde bütün şartlarda, her şart için çalışan bireysel planların birleşiminden oluşan tek bir planın başarılı olması amacı taşınmaktadır. Bu amacı taşımayan fakat belirsizlik olay durumlarına çözüm olmaya çalışan başka yaklaşımlar da vardır.

Çok karmaşık uzaylarda, beklenmedik olay sayısının çok fazla olması tamamen şartlı planlamanın uygulanabilirliğini düşürmektedir. Ancak bu planların tehlikeli uzaylarda kullanılması uygun olmaktadır.

Cassandra (Pryor and Collins, 1993) aşağıdaki özelliklere sahip bir şartlı planlayıcıdır:

1. Planlar hangi eylemler dizinini tercih etme kararını vermek üzere karar adım noktaları ile desteklenmektedirler.

2. Bilgi toplama adımları karar adımlarından farklıdır.

3. Bir eylemi gerçekleştirmenin mümkün olduğu durumlar o eylemin mutlaka gerçekleştirilmesi gerektiği durumlardan farklıdır.

Olasılık tabanlı planlayıcılar başarı olasılığının yüksek olduğu planlar oluşturma hedefini gütmektedir. Olasılığa dayalı planlamada gerçekleşmesi daha muhtemel olaylar için şartlı planlar üretilir (Blythe, 1994; Dean and Boddy, 1988; Gervasio and DeJong, 1991; Kushmerick, Hanks and Weld, 1993). Tahminlenemeyen ya da az rastlanan olaylar durumunda ise yeniden planlama yapılmaktadır. Bu yaklaşım, beklenmedik olayların çoğuna hızlı cevaplar üretiyor olmasına rağmen, dünyada meydana gelen değişikliklerinden kaynaklanan fırsatları kaçırabilmektedir. Örneğin, bir mobil robota bir bina içerisindeki farklı ofislerde çalışan bazı kişilere mektuplarını iletmesi görevi tanımlanmışsa ve bu kişilerden biri koridorda robot ile karşılaşıp robottan mektubu kendi alırsa, bu durum robotun kontrolü dışında bir gelişme olmasına rağmen robot için bir fırsattır. Özellikle söz konusu kişinin ofisi robota göre en uzaktaki ofis ise ve robot bu fırsatı fark edebilecek olursa o ofise kadar gitmeyecek ve kendisine tanımlanacak diğer işler için daha kısa sürede hazır duruma gelecektir. Mobil robotun bu fırsatı fark edebilmesi, eğer olasılık tabanlı bir planlama yapıyorsa, bu olasılığı daha önceden dikkate almış ve bu olasılık için planlama yapmış olması ile mümkündür.

Şu ana kadar açıklanan sistemlerin hiç birinin gerçek bir robot üzerinde uygulanmadığına dikkati çekmek gerekmektedir.

Planlama ve işletimin kesilebildiği (Ambros-Ingerson and Steel, 1988; Dean, Basye, Chekaluk, Hyun, Lejter and Randazza, 1990; Georgeff and Ingrand, 1989; Nourbakhsh, 1997) ve işletimden önce tüm planı oluşturmanın hedeflenmediği sistemler de bulunmaktadır. Hem olasılık tabanlı sistemlerde, hem de planlama ve işletimin kesintiye uğratılabildiği sistemlerde, her şartı dikkate almak gerekmediğinden, üzerinde düşünülmesi gereken planların hangi şartlar için oluşturulacağının belirlenmesidir.

Başka bir planlama yaklaşımı da paralel planlama ve işletim yaklaşımıdır. Bu yaklaşımda planlayıcı ve plan işletici birbirine sıkı sıkıya bağlıdır (Drummond, Swanson, Bresina and Levinson, 1993; Lyons and Hendriks, 1992; McDermott, 1992; Pell, Bernard, Chien, Gat, Muscettola, Nayak, Wagner and Williams, 1997). İşletici bir plan olmasa da çevreye tepkide bulunabilir. Planlayıcı hedef sağlama olasılığını arttırmak için sürekli olarak işleticinin davranışında değişiklikler yapar. Bu yaklaşım hızlı reaksiyonlar sağlar, ancak bir plan mutlaka önceden hazırlanmış olmalıdır. Reaksiyon verebilme özelliğinden dolayı bu yöntem bazı durumlarda hedeften uzaklaşmaya sebep olabilmektedir. Bundan daha da önemlisi, planlayıcı, planlama esnasında işletim esnasında değişmesi muhtemel varsayımlara dayanarak planlarını üretmektedir.

Planlama ve işletimin kesintiye uğratılabildiği yaklaşımda planlama için harcanan emek azaltılmaktadır çünkü alternatif muhtemel sonuçlar anında göz ardı edilebilmektedir. Aynı zamanda çevredeki değişikliklere anında cevap vermek mümkün olabilmektedir. Örneğin, sistemin, batarya

gücü gibi sınırlı sistem kaynaklarının azaldığını ya da kapıların açılması kapanması gibi dış olayları fark etmesi mümkündür. Bu şekilde, planlama ve işletimin kesilebilir olması planlama çabalarını azaltırken sistem içinde fırsatlar yaratılmasına da sebep olmaktadır.

Bu yaklaşımdaki önemli noktalardan bir tanesi, planlamanın ne zaman bitirileceği ve işletimin ne zaman başlatılacağıdır. Dean et. Al, (Dean, Basye, Chekaluk, Hyun, Lejter and Randazza, 1990) en fazla bilgi sağlayan eylemi seçerek bu problemi çözer. Nourbakhsh (Nourbakhsh, 1997), dünya hakkında basitleştirme varsayımlarını yaptıktan sonra bir şartlı plan dalına uzanan tüm hareketleri işleterek bu problemi çözer. Varsayımlar her zaman için hedefe ulaşabilirliği engellemeyecek şekilde yapılmaktadır. (Gervasio and DeJong, 1991) önceden genel ve tam planlar üretir ve işletim zamanında hangi eylemleri işleteceğini almaçlardan sağladığı bilgilere dayanarak yapar.

Planlama ve işletimin kesintiye uğratılabildiği sistemlere örnek olarak Rogue (Haigh, 1998) verilebilir. Rogue, Prodigy 4.0 planlayıcısı üzerine kuruludur. Prodigy 4.0 sebepler-sonuçlar yöntemi kullanarak planlarını geri zincirleme metodu ile oluşturur. Mevcut durumda bir eylemin tüm önkoşullarının doğruluğu bilindiğinde, o eylem işletilir.

Rogue sisteminin ne zaman bir eylem işleteceğini seçerken kullandığı üç metodu vardır. İlk metod, hedefe gittiği bir eylemler dizisi içinden ilkini seçmektir. İkinci metod ise bir seçim esnasında birden fazla eylem bulunması durumudur. Rogue bu eylemler arasından toplam beklenen işletim etkinliğini maksimum yapan eylemi seçer. Üçüncü metodda ise Rogue öğrenme yolu ile hangi eylemi işleteceğini seçer. Bu durumda istatistiksel veriler toplar ve bu verilerden faydalanarak bir eylemi ne zaman işletmenin uygun olacağını tahminler.

Bir eylem işletilmek üzere seçildiğinde, o eylemin ön koşullarının doğruluğu test edilir. Test sonucu olumlu olursa, eylem işletilir ve eylemin oluşturduğu etkilerin doğruluğu test edilir. Bu işlemlerin gerçekleştirilmesine paralel olarak sistemin hedeflerini etkileyebilecek olaylar izlenmektedir. Örneğin, bir izleme fonksiyonu batarya seviyesini test ederken diğeri de belirli nesneler için çalışma uzayını tarar.

Bir diğer planlama yaklaşımı da reaktif planlamadır. Bu yöntemde ise bir davranış, bir küme alt seviyeli reaksiyon kuralı ile kontrol edilir. Aslında reaktif yaklaşımda kullanılan alt seviyeli reaksiyonlar planlama kavramına karşılık gelmemelerine rağmen bu yaklaşımın planlama taraftarlarının çok fazla tepkisini almaması için reaktif planlama olarak adlandırılmıştır.

1980'li yıllardan beri sağlam robot sistemleri geliştirmede planlamanın mı yoksa reaktivitenin mi daha etkin olduğu robot dünyasının ana tartışma konularından birisidir. Hangisinin daha etkin olduğu, üzerinde çalışılan problem uzayına bağlı olarak değişir.

Gerçek bir dünyada çalışan bir sistemin asenkron olan hedefleri ele alabilme yeteneği olmalıdır. Asenkron hedeflere, ilk gelene ilk servis mantığı ile yaklaşan sistemler etkin değildir. Rogue sistemi ise mevcut işlere ait içeriği kaybetmeden yeni hedefleri dikkate alır ve yeni hedefler ile ortaya çıkan fırsatlardan faydalanır. Birbiri ile uyumlu olan işleri akıllı bir şekilde birleştirerek kullanıcı isteklerine hızlı ve etkin cevaplar verir.

Kesintiye uğratılabilir planlayıcılar arasında, sadece PRS (Georgeff and Ingrand, 1989) asenkron hedefleri ele alabilmektedir. Rogue, PRS'in açıkça akıl yürüttüğü düşük seviyedeki detaylar için soyutlama yapar. Bu durumda Rogue'un daha güvenilir ve etkin olduğu söylenebilir çünkü

sistem fonksiyonları uygun bir şekilde alt parçalara ayrılmaktadır (Pell, Bernard, Chien, Gat, Muscettola, Nayak, Wagner and Williams, 1997; Simmons, Goodwin, Haigh, Koenig and O'Sullivan, 1997). NMRA (Pell, Bernard, Chien, Gat, Muscettola, Nayak, Wagner and Williams, 1997) ve 3*T* (Bonasso and Kortenkamp, 1996) planlayıcıları çok sayıda asenkron hedeften oluşan uzaylarda faaliyet gösterirler ve her ikisi de yeni hedeflere ve ciddi eylem başarısızlıklarına mevcut planlamayı terk edip planlayıcıyı tekrar başlatarak cevap verirler.

RAPS ise, (Firby, 1989; Firby, 1994), soyut bir sistem tarafından kontrol edilen bir davranışlar ve reaksiyonlar kütüphanesini aktif hale getirilen bir mimaridir. RAPs gibi Carnegie Mellon Üniversitesi'nde geliştirilen TCA sistemi de çok çeşitli robotlar üzerinde alt seviyede yer alan kontrol mekanizması olarak kullanılmıştır. Bu robotlar arasında bina içi mobil robotlar, ayaklı robotlar, gezegen tarayıcıları ve uzay gemileri vardır. TCA işleri zamanlama ve senkronize etme, kaynak elde etme, çevre izleme ve hata ele alma konularında kolaylıklar sağlar. RAPs ise eylemler için metod ve içerikleri belirlemeye olanak tanır ve TCA gibi aynı kolaylıkları gerçekleştirmek üzere yapılandırılmıştır.

Bazı önceki çalışmalar beklenmedik olayları en yüksek soyutlama seviyesinde ele alır (Laird, Congdon, and Coulter, 1998; Veloso, Carbonell, P'erez, Borrajo, Fink, and Blythe, 1995). Ancak, bu çalışmalar mevcut plan üzerinde değişiklikler yaparak eldeki planı, robot çalışma uzayında beklenmedik olayların meydana getirdiği etkiler durumunda da kullanışlı hale getirmeye çalışmaktadırlar.

Eldeki planı beklenmedik olay sonrasında yeni duruma uygun olacak şekilde değiştirmek bir durum için atanmış işleç değerleri beklenen etkileri gerçekleştirmediğinde, seçilen işlecin değişkenlerine

alternatif değerler atayarak ya da o durumda uygulanabilecek başka bir işleç seçerek (Laird, Congdon, and Coulter, 1998; Veloso, Carbonell, P'erez, Borrajo, Fink, and Blythe, 1995) sağlanır. Dolayısıyla, yeniden planlama, beklenmedik olayın gerçekleşmesinden sonra ortaya çıkan durumda işletildiğinde istenen etkileri ortaya çıkaracak olan "değer(ler), işleç" çiftini bulmak olarak tanımlanabilir.

Oysaki her zaman için eldeki planı yeni duruma uygun olacak şekilde değiştirip kullanmak yerine kimi zaman yeni durumun etkilerini ortadan kaldırıp eski planı kullanmak kimi zaman da yeni durumdan hedef duruma yeni bir plan yapmak (örn. yeni durumdan sonra mevcut planı değiştirmek) daha etkin olacaktır. Kimi durumlarda ise yeni gelinen durumdan hedef duruma daha önceden işletilmiş ve başarımı kanıtlanmış planların kullanılması söz konusu olabilmelidir.

Beklenmedik olaylar sonucunda eldeki planının geçerliliğini yitirmesi durumunda yöntem, planların üretildiği işlerin doğasına bağlı olarak değişir. Örneğin, bir mobil robotun yolu üstündeki ve sapması gereken bir koridoru dikkatinden kaçırması durumunda tekrar koridoru bulup koridorda ilerlemesi sadece koridoru atladığını fark etmesi, koridorun başlangıcına gitmesi ve koridorda ilerleme hareketine başlaması ile mümkündür (Koening and Simmons, 1998). Bu örnekte robotun yaptığı hatayı fark edip düzeltmesi için sadece tek bir yol olmasına rağmen, beklemedik bir olay durumunda verilen bir hedefi gerçekleştirecek alternatif eylem dizinleri arasında seçim yapmaya olanak tanıyan problem çeşitleri de vardır. Bu çalışmada, problem uzayı bu problem çeşitlerini de kapsayacak şekilde genişletilmiştir ve bunlardan ikisi üzerinde yüksek soyutlama seviyesinde yeniden planlama uygulaması yapılmıştır. Bu amaçla bir alternatif seçme mekanizmasına

ihtiyaç vardır. Bu alternatif mekanizmasının işleyişi Bölüm 4'de açıklanacaktır.

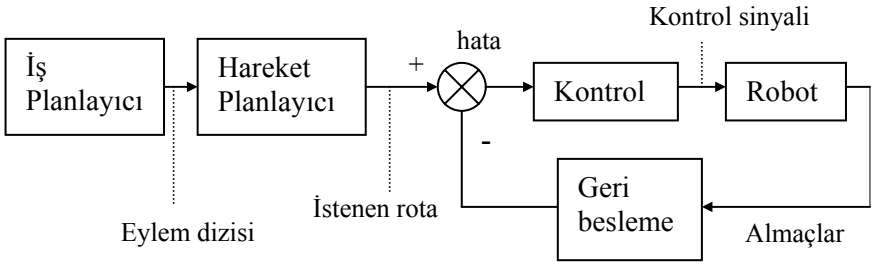
Öte yandan, bazı planlayıcılar, eylem aksamalarının, yararlı/zararlı yan etkilerin ve dış kaynaklı olayların nadiren olacağını varsayarak her eylem işletiminden sonra çalışma uzayı izlemesi yapmadan robota birden fazla eylem için komut dizinleri ya da bir planın tamamını gönderirler. Bazı planlayıcılar sıkça karşılaşılan ve bilinen hataları basit ve doğrudan uygulanabilen davranışlarla ele alan iç toparlama fonksiyonları (internal recovery procedures) kullanırlar.

Şu ana kadar anlatılan planlama sistemleri ve beklenmedik olayları ele almak üzere geliştirilen sistemler robot mimarilerinin bileşenleri ya da bizzat mimarinin uygulanması şeklinde ortaya çıkarlar. Sonraki paragraflarda bu mimarilerin tarihsel gelişimi üzerinde durulacaktır.

1980'lerde yapay us toplumunda yerleşmeye başlayan görüntü bir otonom (akıl yürütme yapabilen) robotun üç adet fonksiyonel bölüme ayrılması gerektiği şeklindeydi. Bu fonksiyonlar bir duyumsama sistemi, bir planlama sistemi ve bir de işletim sistemidir (Nilsson, 1980). Duyumsama sisteminin görevi almaçlardan gelen (genellikle sonar ve görme) verilerden bir dünya modeli oluşturmaktır. Planlayıcın görevi ise bir başlangıç ve hedef modeli girdi olarak alıp hedefi gerçekleştirecek planı üretmektir. İşleticinin görevi ise plan içerisinde yer alan eylemleri işletmektir.

SPA yaklaşımının (Duyumsa- Planla- Harekete geç) iki tane önemli mimari özelliği vardır. İlk özellik, sistemin açık döngü kontrol akışı prensibi ile çalışmasıdır. Açık döngü kontrol akışında (Şekil 2.1) kontrol, almaçlardan dünya modeline oradan da eyleyicilere iletilir ve hiçbir

zaman ters yönde gerçekleşmez. Mimarinin diğer özelliği ise, sistemin akıllı bölümünün işletici değil planlayıcı olmasıdır. Ortaya atıldığı yıllarda SPA yaklaşımın bir takım eksiklerinin olduğu ortaya çıktı. Örneğin çalışma uzayı içerisinde beklenmedik olaylar meydana geldiğinde ve belirsizlikler ortaya çıktığında açık-döngü plan işletimi yetersiz kalmaktaydı. Ayrıca planlama ve dünya modelleme işlemleri son derece zor problemlerdi.

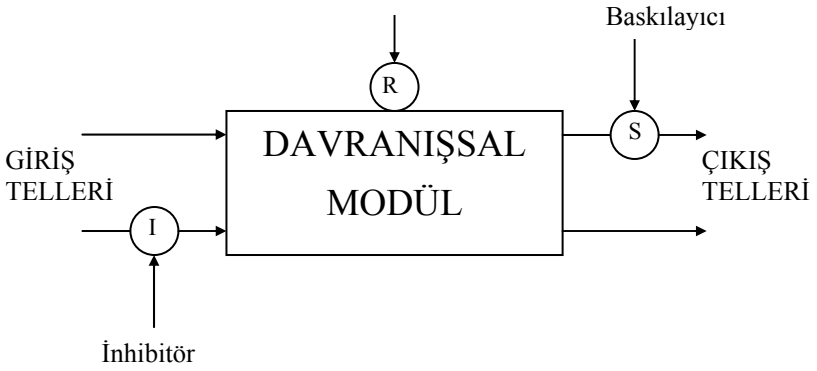


Şekil 2.1 Bir Açık Döngü Robot Kontrol Sistemi

SPA metodunun eksikliklerine karşılık, "Subsumption" mimarisi geliştirildi (Brooks, 1986). Bu mimaride de bilgi akışı yönü SPA'de olduğu gibi almaçlardan eyleyicilere doğrudur. "Subsumption", SPA yaklaşımından plan oluşturmayı ve sembolik gösterimi reddetmesi ile ayrılır. "Subsumption"daki seviyeler geleneksel şekilde oluşturulmamışlardır. Bunun yerine, her seviye, çıkış portlarının tellerle giriş portlarına bağlandığı her biri davranış olarak adlandırılan küçük sonlu durum makinalarından oluşan bir ağdır (Şekil 2.2). "Subsumption" karmaşık kontrol mekanizmaları oluşturmak üzere bu temel yapı taşlarını bir mekanizma kullanarak oluşturur. Bu mekanizma bir telin içeriğini başka bir telden gelen bir içerik ile değiştirme işlemini gerçekleştirir. Bu işlem, bir giriş ya da çıkış portunda gerçekleşiyor olmasına göre "bastırılma" ya da "inhibition" olarak adlandırılır. Dolayısıyla

“inhibition” ile bilgi iletimi kesilirken, bastırma ile bir mesajın yerine başka bir mesajı baskın kılınır. “Subsumption” ayrıca daha karmaşık ve işe özel kontrol programlarını (davranışlar) birbirinin üstüne seviyelendiren bir tasarım metodu kullanır. Ancak, bu yöntemi zorunlu kılan (hatta destekleyen) herhangi bir mimari mekanizması sağlamamaktadır. Davranışlar arasındaki koordinasyon önceliğe sahip davranışların “baskılama” yetilerine sahip olması ile sağlanır. Özellikle çarpmalardan-arınmış robot dolaşması (navigasyon) alanında başarı kazanmıştır.

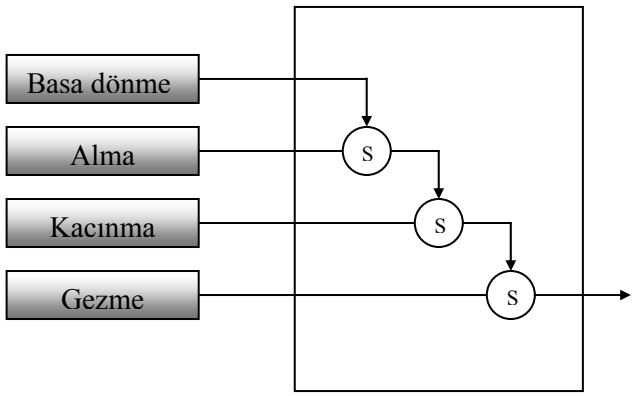
1989 yılında gerçekleştirilen Genghis adındaki robot için tanımlanan davranışlar ayakta durma, basit yürüme, kuvvet dengeleme, ayak kaldırma, çevreyi algılama, eğim stabilizasyonu ve insana doğru yürüme davranışlarıdır (Brooks, 1989).



Şekil 2.2 “Subsumption” Mimarisinde Kullanılan bir Sonlu Durum Makinası

Daha sonra Maja Mataric (Mataric, 1993) tarafından gerçekleştirilen ve robotun hareket ettiği uzay içerisinde bulunan

nesneleri toplamak için gerçekleştirdiği hareketler temelde gezme, kaçınma, alma ve geri dönme davranışları ile tanımlanmışlardır. Gezme hareketi rasgele bir yönde bir süre gitmeyi hedefler. Kaçınma hareketi solda bir engel var ise sağa dönme ve yola devam etme, sağda bir engel varsa sola dönme ve devam etme, eğer üç denemede de engel ile karşılaşılıyorsa robotun başlangıç noktasına geri dönmesi ile eğer hem sağda hem de solda engel var ise robotun geri dönüp ters yönde hareket etmesi davranışlardan oluşmuştur. Alma hareketi ise duyumsanan nesneye dönme, nesneye doğru ilerleme ve nesneye ulaşıldığında da robot elinin nesneyi tutmak üzere kapatılması davranışını içermektedir. Başlangıç noktasına dönme hareketi ise bulunduğu pozisyondan başlangıç noktasına dönme ve o yönde ilerleme davranışı olarak tanımlanmıştır. Nesne toplama senaryosu için koordinasyon Şekil 2.3'de olduğu gibidir.

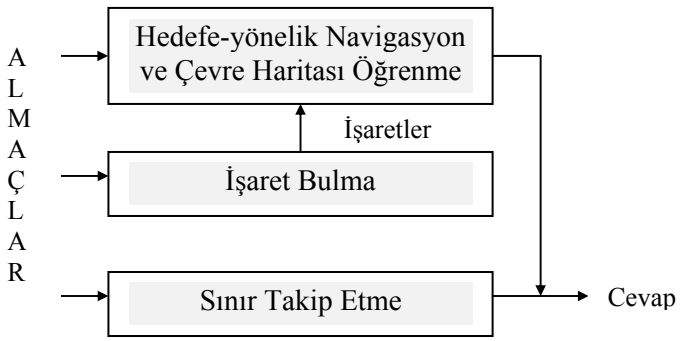


Şekil 2.3 "Subsumption" Tabanlı Nesne Arayan Robot için Uyarım-Cevap Şeması

Şekil 2.3'de görüleceği gibi robot herhangi bir zamanda bir tane davranış işletmektedir. Robot nesnenin farkına vardığında gezme

davranışı baskılanır ve nesne robot eli tarafından tutulduğunda da, başlangıç noktasına dönme davranışı alma davranışını baskılar. Dolayısıyla robotun geri dönme esnasında farkına varıp almak isteyeceği başka nesneler bulunmasına karşın alma hareketi baskılandığından robot bu nesneleri alma hareketini gerçekleştirilmeyecektir.

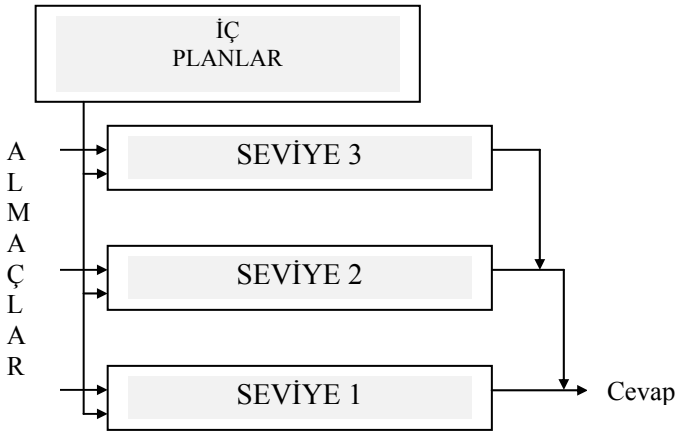
Saf "Subsumption" mimarisi sadece doğrudan almaçlardan gelen bilgiden faydalanıp herhangi bir çevre bilgisi saklama ihtiyacı duymamasına karşılık, sonraki çalışmalarda çevre haritalarını tutan ve kullanan "Subsumption" mimarileri ortaya çıkmıştır. Şekil 2.4'de çevre haritasının oluşturulması şeklinde bir gösterimin "Subsumption" mimarisine dahil edilmiş hali görülmektedir.



Şekil 2.4 "Subsumption" Türünde bir Mimariye Gösterimin Dahil Edilmesi

Daha sonra ise sadece çevre haritasını tutmayan aynı zamanda geçilecek ara pozisyonları saklayan iç planlar oluşturulmaya başlandı (Payton, 1991). İç planlar alt davranışların üstüne oturan başka bir davranış olarak ifade edildi (Şekil 2.5). Böylece, gerekli olduğunda,

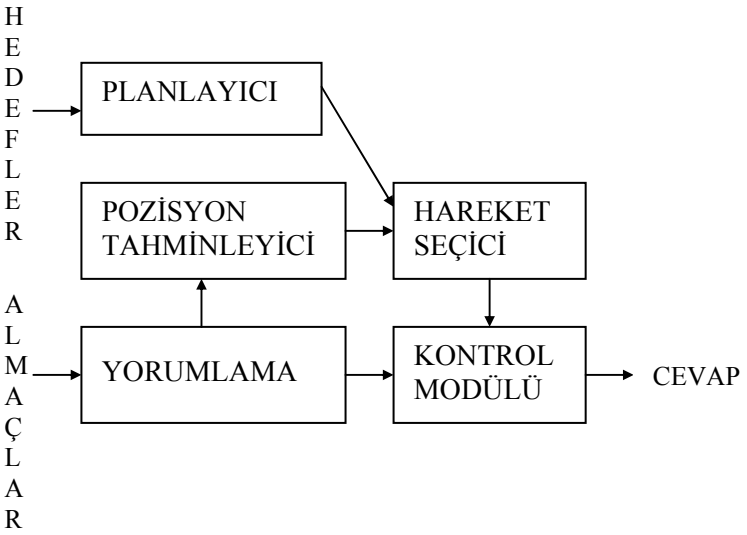
düşük seviyeli davranışlar devreye giriyorlar ancak her şey yolunda gittiğinde tekrar üst seviyedeki davranış şekline dönüyorlardı.



Şekil 2.5 İç Planlar Kullanarak Davranışsal Kontrol

(Simmons and Keoning, 1995)'de yapılan çalışmada ise dünyayı simgeleyen bir çizge oluşturulur. Bu çizgenin oluşturulmasında çalışma alanının örneğin bir ofisin yapısı (koridorlar düz), yaklaşık değer veren ölçümler (kapı aralıklarının genişliği), karar noktaları (örneğin sağa ya da sola dönüş yapılan noktalar) arasındaki mesafe ve çevre içerisindeki nesnelerin modellenmesinden faydalanılır. Özelleştirilmiş olasılık modelleri olan Markov modelleri dünya modeli içerisindeki farklı pozisyonlarda bir robotun işletebileceği hareketleri kodlamakta kullanılır. Seçilen hareketler örneğin sağa ya da sola 90 derece dön ya da 1 metre boyunca ilerle şeklinde basit hareketlerdir. Planlanmış eylemler engelden kaçınma hareketi için destekleyici hareket durumundadırlar. Planlayıcı dünya modeli içerisindeki bir noktada uygulayacağı hareketlere A^* algoritması kullanarak karar verir. Markov tabanlı modellenmiş

planlayıcı robota sağa sola dönme, ileri hareket et ve dur şeklinde komutlar verir. Navigasyon mimarisi bir eylem seçme mekanizması kullanır (Şekil 2.6).



Şekil 2.6 Hareket Seçme Kontrollü Mimari

Almaçlar aracılığıyla elde edilen konum bilgisine dayanarak bir hareketin olasılık bilgileri oluşturulur. Bu sistem XAVIER (RWI B24 tabanlı bir donanım) adı verilen bir mobil robot üzerinde çalıştırılmış (Koening and Simmons, 1998) ve hareketlerinin %88'ini başarı ile tamamladığı bildirilmiştir (Şekil 2.7).

Şekil 2.7 Xavier Mobil Robotu

1986'da "Subsumption"ın takdiminden sonraki yılları değişik robot kontrol mimarileri takip etmiştir. Bunlar birbirinden bağımsız olarak ortaya çıkmışlardır (Kaelbling, 1988; Soldo, 1990; Arkin, 1990; Georgeff and Lanskey, 1987; Simmons, R. G., 1990). Bu mimarilerden bir tanesi "Subsumption"a benzer bir tasarıma sahip olan T/R-3 mimarisidir (Agre and Chapman, 1990). Ancak, "Subsumption"ın aksine, T/R-3 "soyutlama" kavramını reddetmemektedir. "Subsumption"da yüksek seviyeler alt seviyelerle kendi sonuçlarını zorla kabul ettirme şeklinde arayüz oluştururlar. Ancak, T/R-3'te, üst seviyelerin alt seviyeler ile arayüz oluşturması üst seviyelerin alt seviyelere girdi veya öneri sağlaması şeklindedir (Payton, Rosenblatt and Keirsey, 1990; Agre and Chapman, 1990). Öte yandan T/R-3 iş yönelimli (taskable) değildir. Yani kontrol programını yeniden yazmadan başka bir iş tanımlamak mümkün değildir. Bu eksikliğin karşısında, üç ana bileşenden oluşan yeni kontrol mimarileri geliştirilmiştir. Bu mimari 3*T* olarak adlandırılan mimaridir ve aşağıdaki bileşenlere sahiptir.

1) Reaktif bir geri besleme mekanizması.

2) Yavaş bir planlayıcı.

3) İlk iki bileşeni birbirine bağlayan bir sıralama mekanizması.

Reaktif geri besleme mekanizmasına kontrol modülü adı da verilmektedir. Bu modül almaçları eyleyicilere bağlayan bir ya da daha çok sayıda geri besleme döngüsünden oluşur. Kontrol modülü tarafından hesaplanan transfer fonksiyonları işletim sırasında değiştirilebilir. Bir transfer fonksiyonu davranış ya da yetenek olarak ta adlandırılabilir. İşletim esnasında hangi transfer fonksiyonunun aktif edileceği bir dış girdi tarafından belirlenir.

Eyleyicinin fiziksel olarak gösterdiği davranışı bir transfer fonksiyonundan ayırmak üzere fiziksel davranışlar için kullanılan adların ilk harfleri büyük yapılır. Davranışlara örnek olarak duvar kenarı izleme, engellerden kaçarak ilerleme ve kapı aralarından geçme hareketleri verilebilir.

Öte yandan sıralayıcının görevi herhangi bir zamanda hangi davranışın aktif edileceğinin belirlenmesi ve o davranışa ait parametrelerin oluşturulmasıdır. Ayrıca, stratejik durumlarda, davranışların işletimi kesilerek başka davranış(lar) işletime konabilirler. Dolayısıyla belli bir davranış dizinini işletmek yerine duruma uygun bir davranışın işletim için seçilmesi özellikle dinamik ortamlarda istenen bir özelliktir. Sırayıcılara örnek olarak RAPs (Firby, 1989), PRS (Georgeff, and Lanskey, 1987), REX/GAPPS (Kaelbling, 1987) ve ESL (Gat, 1997) verilebilir.

En üst seviyedeki planlama mekanizması ise yoğun ve zaman gerektiren hesaplamaların yapıldığı bileşendir. Bunun anlamı bu seviyede planlama ve arama-tabanlı algoritmaların çalışıyor olmasıdır. Planlama seviyesinde bir ya da daha fazla işlem aynı anda yürütülüyor olabilir. Örneğin bir yanda plan üretiliyorken diğer yanda bir görme sistemi izleme yapıyor olabilir. Bu seviyede çalışan algoritmalar herhangi bir mimari ile kısıtlandırılmamış olmalarına rağmen alt seviyelere bilgi gönderme zamanlarının makul süreler içerisinde olması beklenmektedir.

Planlayıcı seviyesi alt seviyelere iki şekilde arayüzlenir. Bunlar planlayıcının işletilmek üzere bir planı sıralayıcıya üretip göndermesi ya da planlayıcının sıralayıcıdan gelen sorgulara cevap vermesi şeklindedir.

Robot planlamada gelinen son seviyede araştırmacılar robot mimarileri 3*T* olarak adlandırılan mimariyi ve onun değişik versiyonlarını kullanma eğilimi göstermektedirler. Beklenmedik olaylarla başa çıkma durumu söz konusu olduğunda ise bu mimarilerin en alt iki seviyesi olan kontrol ve sıralama seviyelerini kullanma üzerine yoğunlaşmaktadırlar. Planlama seviyesinde beklenmedik olay ve belirsizliklerle başa çıkmaya çalışmanın etkinliği ve güvenliği arttıracağı yönünde bir kanı oluşmuş olmasına rağmen henüz bu düşünceyi kanıtlayıcı çalışmalara yeterli miktarda eğilme olmamıştır.

Bir yandan robot mimarilerin geliştirilirken bir yandan da 90'lı yılların ortalarından sonra "makine öğrenme" kavramı ortaya çıkmıştır. Bu kavram makinaların (bilgisayarların, robotların vb.) otonom hale dönüştürülmesinde büyük önem taşımaktadır çünkü öğrenme kavramı eldeki tecrübelerle önceden yapılan hataları tekrarlamama kavramını da beraberinde getirmektedir. Bu şekilde yazılım ya da donanım olan aracıların sağlamlığı ve güvenilirliği artmaktadır.

2.2 Makine Öğrenme: Tarihçesi ve Tanımı

Bilgisayarların önceki tecrübelerden öğrenmeleri sağlanabilir. Bu amaçla, bilgisayarlar rutin işlerini gerçekleştirirken bazı öğrenme modellerinin toplanması gerekir. Örneğin, bilgisayarlar yeni hastalıklar için en etkin olan tedavi yöntemlerini tıbbi kayıtlardan, bilgisayar tabanlı konutlar, konutta yaşayanların kullanım modellerine (patternlarına) dayanarak enerji maliyeti optimizasyonunu tecrübeden öğrenirken, şahsi yazılım asistanları elektronik sabah gazetelerindeki ilginç konuları yakalamak için kullanıcıların gelişen ilgi alanlarını öğrenmelidirler (Mitchell, 1997). Bilgisayarlar için öğrenmeyi sağlamakla bilgisayarların kullanım alanları ve yetenekleri artacaktır. Ayrıca bilgi işleme algoritmalarının detaylı bir şekilde anlaşılması insanoğlunun öğrenme becerilerinin ya da aksinin daha iyi anlaşılmasını sağlayacaktır.

İnsanların öğrendiği gibi bilgisayarların öğrenmesinin nasıl sağlanacağı henüz bilinmemektedir. Ancak, belli başlı işler için etkin öğrenme algoritmaları geliştirilmiştir. Diğer yandan öğrenmede teorik bir anlayış ortaya çıkmaya başlamıştır. Faydalı öğrenme tiplerini sergilemek üzere çok sayıda pratik bilgisayar programları ve algoritmaları ortaya çıkmıştır. Önem taşıyan ticari uygulamalar da oluşturulmaktadır. Bu uygulamalar, ses tanıma, “data mining” vb. uygulamalarıdır. Ekipman bakım kayıtları içeren büyük veritabanlarından, borsa uygulamalarından, mali işlemlerden, tıbbi kayıtlardan ve benzerlerinden değerli bilgileri alabilmek için makine öğrenme algoritmaları kullanılmaktadır. Makine öğrenmenin bilgisayar bilimleri ve bilgisayar teknolojisinde gittikçe artan bir önemde rol oynayacağı gözlenmektedir.

Bir çalışmada (Schultz, Grefenstette and Adams, 1996), bir robot, karmaşık bir robot davranışını uyarı-cevap kuralları kümesi olarak öğrenmektedir. Başlangıçta öğrenme simülasyon olarak gerçekleşmektedir. Pek çok robot çalışmasında öğrenmenin gerçek bir robot üzerinde yapılmasının maliyeti büyüktür.

Gittikçe artan bir önem kazanan öğrenme şöyle tanımlanabilir:

Bir bilgisayar programının bir T işlerini kapsayan sınıf ve P performans ölçütü için eğer T işlerindeki performansı P ölçümü itibari ile artıyorsa, programın E tecrübesinden öğrendiği söylenir.

Örneğin, bir bilgisayar programının, satranç oyununu içeren iş sınıfları için kazanma yeteneği ölçüldüğünde performansı arttırabilir. Bunu sağlamanın bir yolu programın kendine karşı oynanan oyunlarla elde ettiği tecrübeden faydalanmasıdır.

Mevcut çalışmalar, eldeki bilgiden faydalanan değişik yollar üzerine yoğunlaşır. Örneğin genetik algoritmaların doğasında başlangıç bilgisinin yeterli olması zorunluluğu yoktur. Başlangıçtaki heterojen popülasyon değişik kaynaklardan gelen değişik kural kümelerinden oluşur. Bu kümeler elle üretilen kurallar ve bunların otomatik üretilen kombinasyonlarını içerir. Bir örnek verilecek olursa, 50 kuraldan oluşan bir popülasyona ait her kural kümesi 20 deneme ile başarım oranı açısından değerlendirilir. Başarım oranı, 20 deneme sonucunda ölçülen performansların ortalamasıdır. Sistem 50 jenerasyon için çalıştırılmıştır. Kural kümeleri evrim (evolution) süresince insan etkileşimi ile değişikliğe uğratılır.

Bu çalışmaların bir ucunda geleneksel robotik yaklaşımı bütün robot davranışlarının programlandığı bir mühendislik yaklaşımı kullanır (Grefenstette and Schultz, 1994). Bu yaklaşım robotun ve robot çalışma uzayının çok miktarda insan analizi gerektiriyor olması açısından kullanışlı değildir ve tamamen kontrol edilebilen ve statik çalışma uzaylarında etkindir. Öte uçta, araştırmacılar, insan tasarımcılar tarafından sağlanan modellere en az bağımlılık gösteren akıllı robotlar geliştirmeye çalışmaktadırlar. Bu çalışmalar insan bilgi mühendisliği maliyetini düşürmektedir. Bu düşüncelerle geliştirilen sistemlerde robotların kendi davranışlarını öğrenmesi sağlanabilir.

2.2.1 Eğitim tecrübesinin seçilmesi

Karşılaşılan ilk tasarım seçeneği bir sistemin öğreneceği eğitim tecrübesinin seçilmesidir (Weld, 1999). Sahip olunan eğitim tecrübesinin öğrenicinin başarısı ya da başarısızlığı üzerinde çok önemli etkisi vardır. Önemli bir anahtar özellik, performans sistemi tarafından yapılan seçimleri ilgilendiren doğrudan ya da dolaylı geri beslemenin eğitim tarafından sağlanıp sağlanmadığıdır. Örneğin, satranç oynamayı öğrenmede, sistemin bireysel satranç tahtası durumlarından ve her biri için doğru adımdan oluşan doğrudan eğitim örneklerinden faydalanması mümkündür. Alternatif olarak, sadece hareket dizinleri ve oynanmış değişik oyunların sonuçları gibi dolaylı bilgiye sahip olunabilir. Bu ikinci durumda, oyunun başındaki bazı özel hareketlerin doğruluğu sadece sonuçta oyunun kazanılıp kaybedildiği bilgisinden elde edilmektedir. Burada ayrıca, öğrenci "kredi atanması" olarak adlandırılan ve oyun sonucuna katkısı itibari ile hareket dizinindeki hangi hareketin ne derecede ödül veya ceza hak ettiğinin belirlemesi problemi ile karşı karşıyadır. Kredi atanması problemi zor olabilir çünkü oyun kaybedilmiş

olsa bile oyunun başlangıcındaki hareketler optimal olmasına rağmen sonraki hareketler zayıf olduğu için oyun kaybedilebilir. Böylece, doğrudan eğitim verilerinden öğrenmek doğrudan olmayan verilerden öğrenmekten zordur.

Eğitim verilerinin başka bir özelliği de eğitim örneklerinin öğrenici tarafından ne derece kontrol edilebildiğidir. Öğrenci bilgi veren tahta durumlarını seçmede ve her biri için doğru hareketi belirlemede öğretmene bağlı olabilir. Alternatif olarak, öğrencinin kendisi karmaşık bulduğu tahta durumlarını önerebilir ve doğru hareketi öğretmene sorabilir. Öğrencinin tahta durumları ve (dolaylı) eğitim sınıflandırmaları üzerinde sanki öğretmen olmadan kendi kendine oynuyormuşcasına tam bir kontrolü vardır. Eğitim tecrübesinin üçüncü bir önemli özelliği de üzerinde tüm sistem performansı P'nin ölçüldüğü örnekleri ne kadar iyi temsil ettiğidir. Örneğin aşağıda verilecek satranç öğrenme örneğinde performans ölçütü P dünya turnuvasını kazanan oyunların yüzdesidir. Eğer eğitim tecrübesi E sadece programın kendisine karşı oynanan oyunlardan oluşuyorsa, bu durumun daha sonra üzerinde test yapılacak durumların dağılımını tamamen temsil etmemesi tehlikesi vardır.

Satranç öğrenme problemi:

İş T: satranç oynama.

Performans ölçütü P: dünya kupasında kazanılan oyunların yüzdesi.

Eğitim tecrübesi E: kendine karşı oynanan oyunlar.

Öğrenme sisteminin tasarımını tamamlamak için, şimdi aşağıdakilerin seçilmesi gerekmektedir:

Öğrenilmesi gereken bilginin tam tipi.

Bu hedef bilginin gösterimi.

Bir öğrenme mekanizması.

Bir öğrenme mekanizması farklı öğrenme metodları için farklı şekillerde tanımlanabilir. Kimi öğrenme metodları öğrenme mekanizmasını bir "fonksiyon" olarak adlandırırken kimileri de bu mekanizmayı bir "karar verici" olarak adlandırır. Daha sonraki bölümde takviyeli öğrenme metoduna ait karar vericiler açıklanacaktır. Satranç oyunu için ise öğrenme yöntemi olarak bir fonksiyon kullanılacaktır.

Öğrenilmesi gereken bilgi tipi için en belirgin seçim herhangi bir tahta durumu için en iyi hareketi seçen bir program ya da fonksiyondur. Bu fonksiyonu *ChooseMove* fonksiyonu olarak adlandıralım ve $ChooseMove : B \rightarrow M$ notasyonunun da yasal tahta durumları kümesi B'den herhangi bir durumu girdi olarak aldığını ve yasal hareketler kümesi *M*'den herhangi bir hareket dizinini çıktı olarak gösterdiğini varsayalım.

Böylece T işi için P performansının iyileştirilmesi problemi *ChooseMove* gibi herhangi bir hedef fonksiyonun öğrenilmesine indirgenmiştir. *ChooseMove* fonksiyonu verilen örnekte çok belirgin bir seçim olmasına rağmen sisteme verilen bilginin dolaylı eğitim tecrübesi olması durumunda seçilmesi ve öğrenilmesi çok zor bir fonksiyon da olabilir. Alternatif ve öğrenilmesi daha kolay olan başka bir hedef fonksiyonu, herhangi bir tahta durumuna sayısal bir değer atayan fonksiyondur. Bu hedef fonksiyonu V olarak adlandırılırsa, $V : B \rightarrow \Re$ notasyonu V'nin, B kümesine ait herhangi bir yasal tahta durumunu

gerçel bir değere eşleştirmek için kullandığını gösterir. ($\Re$ gerçel sayılar kümesini göstermektedir.) Bu fonksiyon daha iyi tahta durumlarına daha yüksek puan atamak için kullanılır. Eğer sistem, böyle bir V hedef fonksiyonunu öğrenebiliyorsa, bu fonksiyonu herhangi bir tahta durumundan en iyi hareketi seçmek için de kullanabilir. Böylece her yasal hareketten sonra bir sonraki tahta durumu oluşturulur ve V fonksiyonu en iyi sonraki durumu ve böylece de en iyi hareketi seçmek için kullanılır.

Herhangi bir tahta durumunda hedef fonksiyon V'nin değeri ne olmalıdır? Daha iyi durumlara daha yüksek değer atayan herhangi bir değerlendirme fonksiyonu iş görecektir. B içerisindeki herhangi bir b tahta durumu için bir V(*b*) hedef değerini şöyle tanımlanabilir.

1. Eğer b kazanılma sağlayan final bir tahta durumu ise, V(*b*) = 100'dür.

2. Eğer b kaybetmeye sebep olan final bir tahta durumu ise, V(*b*) = -100'dür.

3. Eğer b eşitliğe sebep olan final bir tahta durumu ise V(*b*) = 0'dır.

4. Eğer b oyun içinde son bir durum değilse, V(*b*) = V(*b'*)'dür ve b', b durumundan başlandığında ulaşılan sonuç durumu gösterir. Oyunun sonuna kadar optimal maliyet korunmuştur.

Bu özyineli (recursive) tanım her b tahta durumu için bir V(*b*) değeri belirtmesine rağmen, bu tanım satranç oyuncusu tarafından etkin olarak hesaplanamadığı için kullanılamamaktadır. Yukarıda belirtilen ve oyunun bitmiş olduğu durumlar için (durum 1-3) ve herhangi bir tahta

durumu için (durum 4), V(*b*) değerinin hesaplanması oyun sonuna kadar optimal bir dizin aramayı gerektirir. Bu durumda öğrenmenin amacı V'nin işleyen bir tanımını bulmaktır; yani satranç oynama programı ile durumları değerlendiren ve gerçek zaman limitleri içerisinde hareketler seçen bir tanım bulunmalıdır.

Böylece, öğrenme işlemi bir ideal hedef fonksiyonu V için işleyen bir tanım bulmaya indirgenmiştir. Genelde tam olarak işleyen bir V fonksiyonu bulmak çok zor olabilir. Aslında öğrenme algoritmaları hedef fonksiyona yakın bir fonksiyon elde ederler ve bu sebeple bir hedef fonksiyonu öğrenmek "fonksiyon tahminleme" olarak adlandırılır. Bu adımdan itibaren **V** sembolü program tarafından öğrenilen fonksiyonu belirtirken, V sembolü ideal hedef fonksiyonunu belirtir.

2.2.2 Hedef fonksiyon için bir gösterim seçilmesi

Öğrenme programının öğreneceği ideal hedef fonksiyonu V'nin öğrenilmesi için bir gösterim seçilmesi gerekmektedir. Bu amaçla çok sayıda seçenek vardır. Örneğin bir **V**'yi temsilen her bir elemanın farklı bir tahta durumu için bir değer belirttiği büyük bir tablo kullanılabilir ya da tahta durumunun özelliklerine karşılık gelen bir kurallar kümesi kullanarak **V**'nin gösterimi sağlanabilir. Ayrıca **V**'yi simgelemek üzere tahta durumlarını tanımlayan bir ikinci derece polinom fonksiyonu ya da bir yapay sinir ağı kullanılabilir. Başka çalışmalarda daha farklı fonksiyonlar da tanımlanmıştır. Bu çalışmalar süresince dikkat edilmesi gereken nokta, ideal hedef fonksiyonunu simgeleyecek ve mümkün olabilecek en yakın ve açık gösterimin bulunmaya çalışıldığıdır. Öte yandan, oldukça açık bir gösterim bulmak için programın simgelediği alternatif hipotezler arasından seçim yapmak üzere daha fazla eğitim

verisine ihtiyaç duyulur. Örneğin, basit bir gösterim ele alınacak olursa, verilen bir tahta durumu için aşağıdaki tahta özelliklerinin doğrusal bir kombinasyonu olacak şekilde bir **V** fonksiyonu hesaplanacaktır.

x1: tahta üzerindeki siyah taşların sayısı.

x2: tahta üzerindeki kırmızı taşların sayısı.

x3: tahta üzerindeki siyah şahların sayısı.

x4: tahta üzerindeki kırmızı şahların sayısı.

x5: kırmızı tarafından tehdit altında olan siyah taşların sayısı (bu durum kırmızı taşlara sıra geldiğinde yakalanabilir).

x6: siyah tarafından tehdit altında olan kırmızı taşların sayısı.

Böylece öğrenme programı,**V**(*b*)'yi aşağıdaki gibi bir doğrusal fonksiyon olarak ifade edecektir.

$$\mathbf{V}(b) = w_0 + w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + w_5x_5 + w_6x_6$$

Bu adımdan sonra öğrenme mekanizması, ağırlık hesaplamaları yapacaktır ve farklı ağırlıklar için çok sayıda hipotez ortaya çıkacaktır. Bu aşamada önemli olan bu ağırlıkların belirlenmesi ve en iyi hipoteze karar verilmesidir. Dikkat edilecek olursa bir satranç oyunu için strateji belirleme ya da öğrenme, hedef fonksiyon gösteriminde yer alan ve w_0 dan w_6 ya kadar olan katsayı değerlerini öğrenme problemine dönüşmüştür.

Hedef fonksiyonu **V**'yi öğrenmek için her biri bir b tahta durumunu tanımlayan ve b için eğitici değeri $V_{train}(b)$ olan bir küme eğitici örneğe ihtiyaç vardır. Başka bir deyişle, her eğitici örnek, formatı (b, $V_{train}(b)$) olan sıralı bir çift şeklindedir. Örneğin, siyahların oyunu kazandığı bir eğitici örneğinde $V_{train}(b)$ hedef fonksiyonun değerinin +100 olduğu bir b tahta durumunu tanımlar.

2.3 Takviyeli Öğrenme (Reinforcement Learning – RL)

(Kaelbling, Litmann and Moore, 1996) ve daha yeni olarak ta (Sutton and Barto, 1998) takviyeli öğrenme (Reinforcement Learning - RL) alanında araştırma bilgileri sağlamaktadır. Bu bilgiler takviyeli öğrenme için iki metod kullanımını önermektedir: değer fonksiyonu uzayını arayan metodlar ve karar verici uzayını arayan metodlar. İlki geçici fark (Temporal Difference - TD) metodu ve diğeri de evrimsel algoritmalar (Evolutionary Algorithms – EA) metodu olarak adlandırılır.

Takviyeli öğrenme metodu, gerçek dünyada kullanıldığında çok büyük arama uzaylarında arama yapma, kısmen gözlenebilir durumlar, nadiren oluşan durumlar ve statik olmayan çevreler ile ilgili problemlerden kaynaklanan zorluklar ile karşı karşıyadır. Ancak değer fonksiyonu uzayında arama, karar verici uzayında aramadan daha başarılıdır.

Bütün takviyeli öğrenme metodları, sıralı karar işlerini çevre ile deneme yanılma etkileşimleri ile çözme hedefini paylaşırlar. Bir sıralı karar işinde, aracı, bir ödül fonksiyonunu optimize etmek üzere durum geçişlerini etkileyen hareketleri seçerek dinamik bir çevre ile iletişime

geçer. Daha formal bir şekilde ifade edilecek olursa, aracı, herhangi bir verilen zamana ait bir s_t durumunda, durumunu algılar ve bir a_t hareketini seçer. Sistem aracıya bir sayısal r(s_t, a_t) yani bir ödül vererek ve $s_{t+1} = \delta(s_t , a_t)$ durumuna geçerek cevap verir. Durum geçiş fonksiyonu şu anki durum ve aracının hareketi ile veya stokastik süreç içererek kararlaştırılabilir.

Aracının hedefi durumları harekete çeviren bir $\pi : S \rightarrow A$ karar vericisi öğrenmektir. ***Optimal karar verici,*** π^*, pek çok şekilde tanımlanabilir fakat genellikle bütün s durumları üzerinde en büyük toplanmış (cumulative) ödülü üreten bir karar verici olarak tanımlanır.

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, V^{\pi}(s), (\forall s)$$

$V^{\pi}(s)$, s durumundan π karar vericisi kullanarak elde edilen toplanmış ödüldür. Bu yaklaşım, ödülleri zaman içerisinde azaltmak için bir γ indirim oranı kullanır. Toplam ise sonsuz bir zaman ekseninde aşağıdaki şekilde hesaplanır.

$$V^{\pi}(s_t) = \sum_{i=0}^{\infty} \gamma^i r_{t+i}$$

r_t, t zamanında elde edilen ödüldür. Alternatif olarak $V^{\pi}(s)$ sonlu bir h zaman diliminde ödüllerin toplamı olarak hesaplanabilir:

$$V^{\pi}(s_t) = \sum_{i=0}^{h} r_{t+i}$$

Çoğunlukla almaçlar, aracıya tam bir durum bilgisi veremezler ve durum kısmen gözlenebilir. Akıllı aracılar takviyeli öğrenmenin yanında, planlama ve denetimli öğrenme (supervised learning) metodu kullanılarak da tasarlanabilirler.

Planlama ile denetimli öğrenme arasındaki farklar genellikle planlama metodlarının durum geçiş fonksiyonunun açık bir modelini gerektirmesi dolayısıyla oluşur. Bir planlama algoritması, aracıyı bir başlangıç durumdan bir hedef duruma ilerletmek üzere muhtemel hareket seçenekleri arasından bir hareket dizisi oluşturur. Planlama algoritmaları bir çevre modelini kullandığından, istenmeyen durum geçişlerini geri alabilirler. Bunun aksi olarak, takviyeli öğrenme, yeterli bir şekilde izlenebilecek bir hareket modeli olmayan durumlarda kullanılır. Sonuç olarak, RL metodunda bir aracı, hareketlerinin etkilerini göstermek üzere aktif bir şekilde çevreyi araştırmalıdır. Planlamanın aksine, RL aracıları durum geçişlerini geri alamazlar. RL araştırmaları bir aracının, çevresi hakkında yeterli bilgisi olmadığı için planlamayı gerçekleştirememesi durumunda aracının davranışları üzerinde odaklanırlar.

Ayrıca aracılar denetimli öğrenme ile eğitilebilirler. Denetimli öğrenmede, aracıya hareketin doğru olup olmadığını belirtecek şekilde durum-hareket çifti örnekleri sunulur. Denetimli öğrenmedeki amaç eğitici örneklerden genel bir karar verici çıkarsamaktır. Böylece, denetimli öğrenme doğru etiketlenmiş örnekler sağlayan bir sisteme ihtiyaç duyar. Bunun aksine RL doğru ya da yanlış kararlara ait ön bir bilgi ihtiyacı duymaz.

Özet olarak, RL, planlamanın ve denetimli öğrenmenin pratik olmadığı durumlarda akıllı aracılar tasarlamak için esnek bir yaklaşım sunar. RL, önemli uzay bilgisinin olmadığı ve elde edilmesinin yüksek

maliyetli olduğu durumlara uygulanabilir. Sık olarak kullanılan bir RL işine bir robot kontrol işi örnek olarak verilebilir. Otonom robot tasarımcıları bir robot kontrol karar vericisi tasarlamak için kullanılması düşünülen planlama veya denetimli öğrenme metodları için gerekli olan ve çalışılacak çevreye ait yeterli bilgiye sahip değildirler. Bu durumda, RL'nin hedefi, robotu, çevresini araştırırken etkin karar vericileri üretmek üzere etkinleştirmektir.

2.3.1 Karar verici uzayına karşılık değer fonksiyonu uzayı

Tanımlandığı şekilde bir takviyeli öğrenme problemi verildiğinde, nasıl bir optimal π^* karar verici kullanılacağına karar verilmelidir. Bu amaçla kullanılabilecek iki çeşit yaklaşım vardır. Bunlardan ilki karar verici uzayında arama yaklaşımı, diğeri ise değer fonksiyonu uzayında arama yaklaşımıdır.

Karar verici uzayı metodları karar vericilerin açık bir gösterimini tutarlar ve bunları çok sayıda arama işlemi aracılığıyla değiştirirler. Dinamik programlama, değer iterasyonu, "simulated-annealing" ve evrimsel algoritmaları (evolutionary algorithms) da kapsayan çok sayıda arama metodları vardır.

Buna karşılık, değer fonksiyonu metodları bir karar vericinin açık bir gösterimini tutmazlar. Bunun yerine, herhangi bir duruma ait bir optimal karar verici için bir beklenen toplanmış ödül fonksiyonu döndüren bir V^{π^*} değer fonksiyonunu öğrenmeye çalışırlar. RL için kullanılan değer fonksiyonu yaklaşımlarının odak noktası bu değer fonksiyonlarının tecrübe ile öğrenilmesidir. Değer fonksiyonlarını

öğrenmek için kullanılan en yaygın yaklaşım geçici fark (Temporal Difference – TD) metodudur.

2.3.2 Takviyeli öğrenme için geçici fark algoritması

Geçici fark (TD) algoritması, değer tahminlerini günlemek için ardışık durumlardan toplanan tahminleme farklarına ait gözlemleri kullanır. Örneğin ardaşık iki durum olan i ve j durumları için sırasıyla 5 ve 2 tahminleme değerleri döndürülüyorsa, aradaki fark, durum i'nin tahminlerin üstünde olduğunu ve j durumuna ait tahminler ile uyuşacak şekilde düşürülebileceğini gösterir. Değerler fonksiyonu V'ye yapılan günlemeler aşağıdaki kural ile gerçekleştirilir:

$$V(s_t) = V(s_t) + \alpha(V(s_{t+1}) - V(s_t) + r_t)$$

α öğrenme oranını ve $+ r_t$ herhangi bir ara ödülü gösterir. Böylece ardaşık durumlara ait tahminlemeler arasındaki fark olan $(V(s_{t+1}) - V(s_t))$ tahminleme hatasının bir ölçütü olarak kullanılır. Son tahmin $V(s_n)$'nin sadece sıfır olmayan ödüller içerdiği ve ardaşık durum geçişlerine ait $V(s_0)\ldots V(s_n)$ tahminler zincirini ele alalım. Bu dizinin pek çok iterasyonu sonucunda, her durumun değeri kendinden sonra gelenler ve $V(s_n)$ ile sonuçta anlaşmaya varacak şekilde günlenir. Başka bir deyişle, tek ödül değer tahminleri zincirinde geriye doğru beslenir. Net sonuç ise sistemin herhangi bir durumda beklenen ödülü tahminleyebileceği doğru bir değer fonksiyonudur.

Daha önceden de bahsedildiği gibi, TD metodlarının hedefi optimal karar verici V^{π^*} için değer fonksiyonunun öğrenilmesidir. V^{π^*},

verildiğinde, optimal $\pi(s)$ hareketi aşağıdaki denklem kullanılarak hesaplanabilir:

$$\pi(s) = \underset{a}{\operatorname{argmax}}\, V^{\pi^*}(\delta(s, a))$$

Daha önceden RL'de durum geçiş fonksiyonu $\delta(s, a)$'nın aracı tarafından bilinmediği belirtilmişti. Herhangi bir bilgiye sahip olmadan yukarıdaki fonksiyonu değerlendirmek mümkün değildir. Hesaplanabilecek alternatif bir fonksiyon da $Q(s, a)$ şeklinde ifade edilebilecek bir değer fonksiyonudur. Q fonksiyonu s durumunda a hareketini işletmenin ve ardından optimal hareketler yapmanın beklenen değerini ifade eden bir değer fonksiyonudur:

$$Q(s, a) = r(s) + V^{\pi}(\delta(s, a))$$

$r(s)$, s durumunda alınan herhangi bir ara ödülü simgeler. Q fonksiyonu verildiğinde aşağıdaki denklem kullanılarak optimal karar vericiğe ait hareketler doğrudan hesaplanabilir.

$$\pi^*(s) = \underset{a}{\operatorname{argmax}}\, Q(s, a)$$

Q fonksiyonu aşağıdaki TD günleme denklemi kullanılarak öğrenilir:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha(\max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) + r(s_t))$$

2.3.3 Takviyeli öğrenme için evrimsel algoritmalar

Karar verici uzayı yaklaşımı uygun bir hedef fonksiyonu optimal yapmak için karar vericiler üzerinde arama yapar. Evrimsel algoritmalar (EA) karar verici uzayını aramak için kullanılabilecek algoritmalardan biridir. EA algoritmaları Darwin'in doğal ayıklama ile evrim teorisinden türetilmiş olan global arama teknikleridir. Bir evrimsel algoritma kromozom adı verilen yapılarla kodlanmış bir potansiyel çözümler popülasyonunu tekrarlamalı olarak günler. Jenerasyon adı verilen her tekrarda, EA çözümleri değerlendirir ve her çözümün çevreye uygunluğuna bağlı olarak bir evlat (offspring) oluşturur. Çözümlerin alt yapıları olan genler mutasyon ve yeniden birleştirme olarak adlandırılan genetik işleçler aracılığı ile değişikliğe uğratılırlar. Bunun altında yatan fikir, iyi çözümlere ait yapıların gelecekteki jenerasyonlarda daha iyi çözümler oluşturmak üzere mutasyona uğratılması veya birleştirilmesidir. Genetik algoritmaları, evrimsel programlamayı, genetik programlamayı ve evrimsel stratejileri kapsayan çok çeşitli evrimsel algoritmalar geliştirilmiştir. EA'ların başarılı bir şekilde yaygınlaşmasının sebebi az sayıda gerekliliklerinin olmasıdır. Bu gereklilikler şöyle sıralanabilir:

1. Arama uzayı ile kromozomlar arasında uygun bir eşleme.

2. Yeterli bir uygunluk fonksiyonu.

Örnek olarak, parametrik optimizasyonda, parametre listesini gerçel sayılardan oluşan bir vektör ya da parametreleri kodlayan bir bit dizini olarak ifade etmek mümkündür. Her iki gösterimde de mutasyon ve çaprazlama işleçleri kolaylıkla uygulanabilir.

Kullanıcı EA için kontrol parametrelerine karar vermek durumundadır. Bu parametreler popülasyon büyüklüğü, mutasyon oranı, yeniden birleştirme oranı ve ebeveyn seçim kurallarıdır. EA'ların değişik kontrol parametleri değerleri için sağlam çalıştığını ifade eden çok sayıda çalışma vardır.

Buraya kadar anlatılan öğrenme metodları öğrenmenin tanımında da belirtildiği gibi bir P performans ölçütü için bir E tecrübesinden faydalanarak bir T sınıfına ait işlerin gerçekleştirilmesinde P performans ölçütünün arttırılmasını sağlar. Bu çalışma kapsamında kullanılan ve geliştirilen öğrenme metodunda bir T sınıfına ait işlerin (bloklar dünyası işleri) P performansı (robot kolunun beklenmedik olaylar sırasında doğru karar verme yüzdesi) bir eğitim tecrübesi E'den (geçmiş işletimlere ait maliyetler) faydalanarak iyileştirilmektedir. Literatürde tanımlanan yöntemler öğrenmelerini ya bir öğrenme fonksiyonuna ait ağırlık katsayılarını ya da kural tabanlı bir sisteme ait yeni kuralları öğrenerek gerçekleştirirken, bu çalışmadaki öğrenme metodu daha sonra ne amaçla kullanıldığı açıklanacak olan bir eşik değerini öğrenerek gerçekleştirmektedir.

Yukarıda anlatılan öğrenme metodlarının her birinde optimal yapılmaya çalışılan bir fonksiyon ya da karar verici vardır. Bu çalışmada planlama yapmak için kullanılan yöntemde bir fonksiyon belirlenip bu fonksiyonda yer alan ağırlıklar hesaplanmamaktadır. Öte yandan takviyeli öğrenmede bahsedilen geçici fark yönteminde adı geçen ödül mekanizması çerçevesinde bir plan içerisinde yer alacak her bir hareket sonucunda kat edilen yol, toplam kat edilen yola eklenen bir ödül değeri olarak düşünülebilir. Geçici fark yönteminde sisteme ödüllerin sağlanması almaçlar aracılığı ile olmasına rağmen bu çalışmada bir hareketin sağlayacağı ödül önceden bellidir. Toplam yolun hesaplanması

toplam ödülün oluşturulması anlamına gelmektedir. Bir durumda iken o durumda uygulanabilecek alternatif hareketler vardır. Bu sebeple optimal hareketler dizisini oluşturmak için bir "karar verici" olması gerekmektedir. Bu çalışmada kullanılan "karar verici" sezgisel bir yöntem kullanır. Bu sezgi "en yakındakine hareket et" sezgisidir. Her bir hareketin sağlayacağı ödül belli olmasına rağmen sonraki hareket seçimleri önceki hareket seçimlerine bağımlı olduğundan optimal toplam ödülü hesaplamak polinom zamanda çözülemeyecek bir problem olarak karşımıza çıkmaktadır. Toplam ödülün ya da yolun hesaplanması ile ilgili olarak daha ayrıntılı açıklamalar sonraki bölümlerde verilecektir.

Bu çalışmada yeniden planlama için de bir öğrenme mekanizması kullanılmaktadır. Bu mekanizma, sistemin önceki işletimleri sonucunda elde edilen maliyetlerin saklanması ve sistemin yeniden planlama mekanizmasında "karar verici" olarak kullanılan bir "eşik" değerinin saklanan maliyetlerden faydalanılarak zaman içerisinde iyileştirilmesi şeklindedir. Mekanizma bu çalışma çerçevesinde oluşturulmuş maliyet tabanlı bir mekanizmadır. Maliyet tabanlı öğrenme yapan başka çalışmalar (Tan, 1993) olmasına rağmen bu çalışmadaki mekanizma orijinaldir.

3. ALAN BAĞIMSIZ (DOMAIN INDEPENDENT) PLANLAYICI SEVİYESİ

3.1 Görme Destekli Planlama (VGP) Sisteminin Genel Yapısı

Bu çalışma kapsamında bir otonom aracı (agent) oluşturulmaktadır. Bu aracı, bloklar dünyası olarak adlandırılan çalışma uzayları için tanımlanan işleri gerçekleştirebilmekte ve bu esnada beklenmedik olaylar ortaya çıkacak olursa bu olayları ele alabilmektedir. Oluşturulan aracı tanımlanan işleri gerçekleştirirken çalışma uzayını kuş bakışı olarak bir kamera ile izlemekte ve beklenmedik olayları bu izleme sonucunda elde ettiği bilgileri kullanarak yakalamaktadır.

Çalışma uzayı iki ya da üç boyutlu olmasına bağlı olarak $n \times n$ ya da $n \times n \times h$ ($n > 0$ ve $h > 0$) sayıda hücreden oluşacak şekilde tanımlanmıştır. Robot kolunun hareketleri robot kolunun bir hücreden diğer bir hücreye hareketleri olarak tanımlamaktadır.

Bu çalışmada ele alınan problemler kaldır/koy problemleridir ve çalışma uzayında hareket ettirilecek nesneler robot elinin kavrayabileceği büyüklükte bloklar veya bir robot koludur. Kaldır/koy işlemleri, özellikle, elektronik kartlarda devre bileşenlerinin yerleştirilmesi, fabrikalarda kutuların taşınması, limanlarda yük indirme/bindirme işlerinde önem kazanmaktadır. Bu işlerin gerçekleştirilmesi esnasında görme sistemlerinin kullanılması oluşabilecek hata durumlarının yakalanması ve güvenliğin sağlanması için önemlidir.

Herhangi bir uzaydaki nesnelere (blok ya da robot kolu) ait özellikler nesne (object) adı verilen bir veri yapısı içinde tutulurlar (Şekil

3.1) ve nesnenin tutulan özellikleri arasında nesnenin tipi, adı, boyutu, o anki x, y, z koordinatları ile hedef x, y, z koordinatları bulunmaktadır. Uzaydaki blokları birbirinden ayırmak için her bloğun üst yüzeyine bir harf ya da işaret içeren bir etiket yapıştırılmaktadır.

```
Object {

    char type[ ];  //Nesne tipi robot kolu ya da blok olabilir.

    char name;   // Blok adları P(iyon), V(ezir), K(ale), S(ah), A(t) ,F(il) olabilir.

    char destinationOccupancy[ ];  //Nesnenin hedef pozisyonu dolu ya da boş
                                   //olabilir.

    char distanceToDestination[ ] ; // Seçilen hedef en yakındır ya da değildir.

    int destx, desty, curlocx, curlocy; //nesnenin şu anki //koordinatlari ve
        hedef koordinatlarını tutan değişkenler.

    int processed;        //Nesnenin işletilip işletilmediğini tutan değişken.
} ;
```

Şekil 3.1 Nesne Veri Yapısı

Kaldır/koy işlemleri için 5 serbestlik dereceli Rhino robot kolu kullanılmaktadır. Robot kolunu belirtilen koordinatlara götürmek için bilgisayardan üst seviyeli komut (motor adı, hareket yönü ve kodlayıcı adım sayısı) alan ve bu komutları alt seviyeli kontrol sinyallerine çevirip robot motorlarına gönderen bir kontrol ünitesi kullanılır. Robot kolunun uç kısmının gideceği koordinatlar ile robot koluna ait her iki eklem arasındaki açılar arasındaki ilişki Denavit-Hartenberg parametleri kullanılarak oluşturulan kinematik denklemleri ile çözümlenmiştir (Bkz. Ek 1).

Üst seviyeli komutların oluşturulması, gerçekleştirilmesi istenen işe ait başlangıç ve bitiş durumlarının parametre olarak verildiği bir planlayıcı ile sağlanır. Windows tabanlı bir PC üzerinde çalışan ve iş planlayıcı olarak adlandırılan bu planlayıcının ürettiği planlar sembolik ifadelerden oluşmaktadır. Bu sembolik ifadeleri içeren her plan adımı yine üst seviyeli olan ve belirtilen koordinatlara ulaşmayı sağlayacak motor adı, hareket yönü ve kodlayıcı adım sayısı bilgilerini içeren komutlara derlenir. Bir plan adımını gerçekleştirecek bir ya da daha fazla derlenmiş komut olabilir.

Her plan adımının derlenmesinden sonra ortaya çıkan komutlar kontrol ünitesine gönderilir. Bir plan adımının işletilmesinin ardından iki türlü hareket gerçekleştirilmiş olabilir. Bunlardan ilki robot kolunun elinde bir blok olmaksızın sadece pozisyonunu değiştirmek amacı ile bir pozisyondan başka bir pozisyona gitme hareketidir. Örneğin robot kolunun başlangıç pozisyonundan çalışma uzayındaki bir hücreye o hücredeki bloğu kaldırmak için gitmesi böyle bir harekettir (move_arm). Diğeri ise robot kolunun elinde bir blok olduğu halde bir pozisyondan başka bir pozisyona gitmesi hareketidir (move_block). Örneğin robot eli bir bloğu bulunduğu bir hücreden kaldırıp başka bir hücreye koymak için o hücreye doğru ilerlemesi hareketi move_block hareketidir.

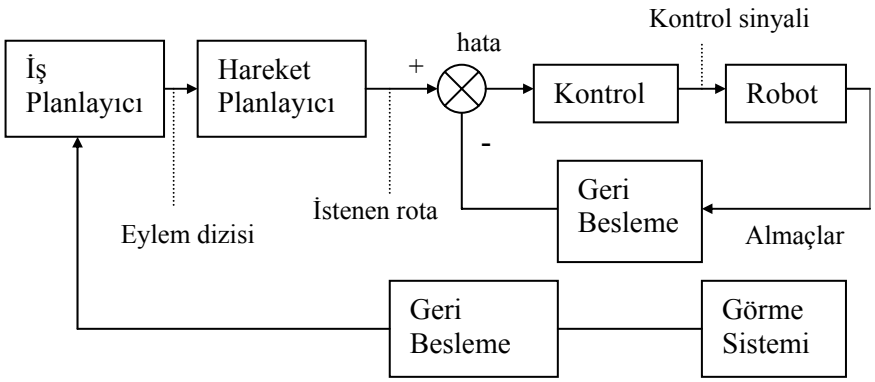
Üst seviyeli komutların oluşturulması bazı kısıtlar altında gerçekleştirilir. Tanımlanabilecek kısıtlar arasında blokların boyutu, eğer uzay üst üste blok yerleştirmeye izin veriyorsa birbiri üstüne yerleştirilebilecek maksimum blok sayısı ve bir bloğun kendinden daha küçük bir blok üzerine yerleştirilemeyeceği fakat kendinden küçük yan yana iki blok üzerine yerleştirilebileceği kısıtları vardır. Planlama sırasında bazı kısıtların göz ardı edilmesi ya da bir robot kolunun

mekaniğinde oluşmuş hataların zamanla birikmesi sonucunda bir bloğun yanlış bir pozisyona yerleştirilmesi söz konusu olabilir.

Eğer sonuçta bir blok taşıma eylemi gerçekleşmiş ise robot kolu bilgisayardan iletilen komutlarla başlangıç pozisyonuna alınır ve çalışma uzayı bir kamera ile beklenmedik bir olay olup olmadığını ya da son plan adımının doğru işletilip işletilmediğini tespit etmek için görüntülenir. Görüntünün değerlendirilmesi sonucunda yolunda gitmeyen bir durum olduğu ortaya çıkarsa işletilmekte olan plan kısmen ya da tamamen geçerliliğini yitirir. Bu durumda yeniden planlama yapma yaklaşımı yani yeni bir plan üretme ya da eldeki planın değişikliklerle tekrar kullanılabilir hale getirilmesi gibi yaklaşımlar kullanılır.

Bu çalışma kapsamında hedeflenen nokta literatürdeki planlayıcılardan daha yüksek performansa sahip bir planlayıcı elde etmek olmayıp kaldır/koy türünde planlama problemlerini çözen bir sistem mimarisi gerçekleştirmek, beklenmedik olayları yüksek soyutlama seviyesinde görme tabanlı yeniden planlama yaparak ele almak ve yeniden planlama kavramını beklenmedik olaylar sonucunda oluşan durumda alternatif bir işleç seçme durumu olmaktan çıkarıp maliyet optimizasyonunu da dikkate alan farklı alternatif eylem dizinleri arasından optimal ya da alt optimal olanı seçebilen bir kavram olarak genişletmektir.

Yüksek soyutlama seviyesinde (iş planlayıcı seviyesine görme sisteminden gelen geri besleme ile) yeniden planlama yapma düşüncesi Şekil 3.2'de gösterilen kontrol ve hareket planlama seviyelerine yapılan geri beslemelerin beklenmedik olayları ele almada yetersiz kaldığının gözlenmesi sonucu ortaya çıkmıştır.



Şekil 3.2 Geri Besleme Bilgisinin İş Seviyesine İletildiği Bir Kapalı Döngü Sistemi

Üst seviyeli iş planlayıcıya bir görme sisteminden gelen geri besleme ulaşırken kontrol seviyesini tasarlamakta kullanılan reaktif modelleme seviyesine de almaçlardan gelen geri besleme iletilmektedir. Örneğin çalışma uzayı içerisinde robot kolunun bir nesneyi elinden düşürmesi ya da bir nesneyi yanlışlıkla başka bir nesne üzerine koymaya çalışması gibi hata durumları reaktif modellemenin yapıldığı kontrol seviyesinde çözülebilecek problemler olarak ele alınırken çalışma uzayı içerisindeki nesnelerin yerlerinin başka bir dış aracı tarafından değiştirilmesi problemi üst seviyede ve yeniden planlama modülüne görme sisteminden gelen geri beslemenin değerlendirilmesi sonucunda alınacak kararlar ile çözülmektedir.

Bazı planlayıcılar bir geri besleme almaksızın birden fazla eylemi arka arkaya işletmektedirler. Bu sistemlerin güvenirliği yoktur. Bazı planlayıcılar da beklenmedik olayları ele almak için karmaşık toparlama programları (recovery routines) kullanmaktadırlar. Bu çalışmada planlayıcının akıl yürütme (reasoning) ile beklenmedik olaylara çözüm bulmasını sağlanmaktadır. Dış aracıların beklenmedik olaylara sebep

olması dışında işletilen eylemlerin çalışma uzayında beklenmedik yan etkileri de olabilmektedir. Böyle bir durumda yan etkileri ortadan kaldırıp yan etkilere sebep olan eyleme alternatif bir eylem kullanarak sorunu çözmek mümkündür. Ancak bu çalışmada içerisinde eylemlerin yan etkileri değil herhangi bir eylemin işletimi esnasında ya da sonrasında dış aracıların çalışma uzayında oluşturması muhtemel beklenmedik olaylar ele alınmaktadır.

Önerilen planlama/yeniden planlama mimarisi kaldır/koy problemleri için optimal mesafe kat eden eylem dizinleri (planlar) oluşturmayı hedeflemektedir. Ancak, bu mimari gerekli uzay tanımlamaları sisteme verildiğinde kaldır/koy problemleri çerçevesinden çıkmamak kaydı ile optimal mesafe bulma problemleri dışındaki problemleri de çözecek şekilde tasarlanmıştır.

3.2 Planlama Metodu ve Algoritması

Planlama işleçler ve arama kontrol kurallarından oluşan bir iş modeli (task model) belirlenmesi işlemidir.

İlerdeki paragraflarda, işleçler tanımlanacak, planlama algoritması verilecek ve son olarak da planlama algoritmasını gerçekleştirmek için kullanılan arama kontrol kuralları tanıtılıp bu kuralların planlayıcının planlama kararlarını nasıl yönlendirdiği açıklanacaktır.

Genel amaçlı bir planlayıcı oluşturabilmek için farklı çalışma uzayları için planlama yapılabiliyor olması gerekmektedir. Bu çalışmada oluşturulan planlayıcı STRIPS tabanlı VGP (Vision Guided Intelligent Planner) planlayıcısıdır. STRIPS tabanlı yaklaşım eylemlerin ve hedef ve

başlangıç durumların gösterimi için "predicate calculus" yöntemi kullanır. Planlayıcının genel amaçlı olması özelliği, farklı planlama uzaylarının (uzay işleçlerinin ve kuralların) gösterimine izin veren ve Sebepler-Sonuçlar analizi (Bigus and Bigus, 1997) olarak adlandırılan planlama mekanizması ile sağlanır. Sebepler-Sonuçlar analizi, bir problem uzayı içerisinde gerçekleştirilen aramayı yönlendirmek için alternatif olan durum transformasyon işleçleri arasından sezgisel bir şekilde seçim yapılması yöntemidir (Luger and Stubblefield, 1993). Bu yöntem, bir o anki durum ile bir hedef durum arasındaki söz dizimsel farkları inceler ve bu farkları azaltacak bir transformasyon ya da başka bir deyişle bir işleç (eylem) seçer. Örnek olarak iki mantıksal ifadenin eşitliğinin ispat edilmeye çalışıldığını varsayalım. O anki durumun bir Λ işleci içerdiğini ve hedef durumun bu işleci içermediğini varsayalım. Bu durumda sebepler-sonuçlar yöntemi Λ işlecini ifadelerden kaldırabilecek "de Morgan" kuralı gibi bir transformasyon seçecektir.

Bir durumu başka bir duruma taşıyan transformasyon ya da işlemlere örnek olarak bir robot kolunun bir bloğu başka bir blok üzerine koyması işlemi verilebilir. Bu işlemi gerçekleştirecek olan işleç yerleştir(x, y) olarak tanımlanırsa bu işlecin ön koşulları ve etkileri aşağıdaki şekilde ifade edilebilir:

Ön koşullar: üstü-boş(y) ve robot-elinde(x).

Etkiler: üstünde(x,y) ve robot-elinde() ve üstü-boş(x).

"üstü-boş", "robot-elinde" ve "üstünde" ifadelerinin her biri bir hazır bilgi (literal) olarak adlandırılır ve yerleştir(x, y) işleci işletildikten sonra bir y bloğunun üstünün boş olması ve robot kolunun x bloğunu

taşıyor olması durumu x bloğunun y bloğu üstünde olması, robot elinin boş olması ve x bloğunun üstünün boş olması durumuna değiştirilir.

Sebepler-Sonuçlar analizi yöntemi kullanarak planlama yaparken bir durum uzayı aramasını ileriye ya da geriye zincirleme yöntemi ile yapmak mümkündür (Bigus and Bigus, 1997). İleri zincirleme yönteminde bir başlangıç durumunu bir hedef duruma taşıyacak olan işleç dizini başlangıç durumdan başlayarak hedef duruma ulaşılacak şekilde oluşturulur. Bir işleç seçildikten sonra o işlece ait ön koşullar o durum için doğrulanmış demektir ve o işlecin etkilerinin mevcut duruma eklenmesi ile bir dünya modeli oluşturulur. Hedef duruma ulaşıldığında tüm işleçler belirlenmiş olur ve plan olarak adlandırılan bir işleçler (eylemler) dizini ortaya çıkar. Bu durumda planlayıcının tuttuğu dünya modeli ile hedef durum aynı olmalıdır.

Geriye zincirleme yönteminde ise hedef duruma gelinmesini sağlayan işleç dizini hedef durumdan başlangıç duruma ulaşıncaya kadar geri giderek oluşturulur. Bu yöntemde ilk seçilen işleç plan içerisinde son işletilecek işleçtir. Etkilerinin hedef durumda var olduğu tespit edilen işleçlerden birisi eylemler dizisini oluşturmak üzere seçilir ve o işlecin ön koşullarının hedef duruma eklenmesi ile beklenen dünya modeli oluşturulur. Bu işlem en son işlecin ön koşulları mevcut duruma eklenip başlangıç durum elde edilinceye kadar devam eder. Sonuçta tüm işleçler belirlenmiş olur ve plan olarak adlandırılan bir işleçler dizini ortaya çıkar

Her iki yöntem sonucunda elde edilen planlar robot tarafından işletilir. Bu çalışmada kullanılan yöntem ileriye zincirleme olmasına rağmen (Veloso, Carbonell, P'erez, Borrajo, Fink, and Blythe, 1995), geriye zincirleme yapan çalışmalara örnek olarak verilebilir. Öte yandan bu çalışmada geriye zincirleme kullanılmasına engel olan herhangi bir

kısıt olmamasına rağmen, ileriye zincirleme metodu insan akıl yürütme modeline daha yakın olması sebebi ile tercih edilmiştir.

Bazı planlama yöntemleri işletime başlamadan önce tüm planın oluşturulmuş olmasını hedeflemezler. Kimi planlayıcılar kısmen oluşturdukları planları işletime koyar ve her şeyin beklendiği gibi gittiğini tespit ettikten sonra planların geri kalan kısmını oluştururlar. Bu planlayıcılardan Bölüm 2'de planlama ve işletimin kesilebildiği planlama yöntemleri olarak bahsedilmişti.

Herhangi bir problem uzayı içerisinde bir eylem dizini oluşturmak üzere arama yaparken bir karar noktasında birden fazla işleç arasından seçim yapmak gerekebileceği gibi seçilen bir işlece birden fazla değerin atanabilir olması durumu da söz konusu olabilir. Bu çalışma kapsamında işleçler arasından seçim yapma durumu yerine bir işlece değer ataması yaparken olası değerler arasından minimum seyahat mesafesini sağlayacak ya da sağlaması muhtemel atamayı seçerek akıl yürütülmektedir.

Üzerinde çalışılan problemler sonuçta optimal bir seyahat mesafesini elde etme amacı güdüyorsa bir işlecin değişkenlerinin aldığı değerlerin seçilmesinde kullanılan algoritma ya da yöntem optimal seyahat mesafesini elde etmede büyük önem taşımaktadır. Bu çalışmada üzerinde durulan problemler optimal çözüm isteyen problemler grubuna girmelerine rağmen "NP-complete" olmalarından dolayı optimal çözümleri yoktur. Buna rağmen değer atamaları optimal seyahat mesafesine yaklaştığı varsayılan sezgisel metodlarla gerçekleştirilmektedir.

Planlama ya da başlangıç durumu hedef duruma taşıyacak bir eylem dizisi bulma işlemi, tüm hedefler gerçekleştirilinceye kadar devam eder. Bu çalışmada, hedefe optimal şekilde ulaştıracağı düşünülen işleçlere ait değişken değerlerinin seçiminde sezgisel yöntemlerin kullanılması işlemi arama kontrol bilgisi kullanılarak gerçekleştirilir. Sezgisel tabanlı arama kontrol bilgisi planlayıcıya kontrol kuralları şeklinde verilmektedir. Kontrol kuralları bir problem uzayı içerisinde verilen bir başlangıç durumundan hedef duruma gitmek için yönlendirme yapan kurallar olarak tanımlanırlar (Haigh, 1998). Planlama problemleri için çözüm olan algoritmaların detayları kontrol kuralları içerisinde gerçekleştirilmektedir.

Kontrol kuralları her karar noktasındaki seçim sayısını mevcut bir arama ağacını küçülterek (pruning) yani plan oluşturulması esnasında birden fazla işleç (eylem) arasından bir tanesini önererek azaltmayı hedefler. Kontrol kuralları mevcut duruma ve meta-bilgiye bağlı olarak hangi seçimlerin yapılması (veya göz ardı edilmesi) gerektiğini belirten *eğer-ise* kurallarıdır. Özellikle, herhangi bir karar noktasında önceden belirtilmiş bazı planlama tercihlerini seçer, tercih eder veya dışlar (Veloso, Carbonell, P'erez, Borrajo, Fink, and Blythe, 1995; Laird, Congdon, and Coulter, 1998). Kontrol kuralları planlamayı özel hedeflere ve arzu edilen planlara doğru odaklar. İki ayrı kontrol kuralı *eğer-ise-değilse* yapısı kullanarak tek bir kontrol kuralı olarak da oluşturulabilir.

Bir kontrol kuralının yapısı şu şekildedir:

Eğer A önkoşulları sağlanıyorsa X eylemlerini aksi halde Y eylemlerini işlet.

Bir iş planlayıcı eğer kural-tabanlı çalışıyorsa öğrenme yetenekleri ile duruma dayalı kurallar üretebilir. Aşağıda verilmiş kurallar örnek bir planlayıcı tarafından duruma dayalı olarak üretilmiş kurallardır.

Eğer ((şu-anki-saat < 11) veya (şu anki saat > 8)) A ofisine ait işleri dikkate alma.

Yukarıdaki kontrol kuralı bir çalışma uzayında yapılan gözlemler sonucu elde edilen tecrübelerden faydalanarak üretilmiştir. Bu kontrol kuralının çıkarsanmasının sebebi sabah 8-11 saatleri arasında A ofisinde bir çalışan olmaması dolayısıyla ofisteki kişilerle iletişime geçilerek yapılması beklenen işlerin gerçekleştirilemeyeceğini sisteme ifade edebilmektir. Bu çalışmada kullanılan kontrol kuralları Bölüm 4'de verilmektedir.

Yukarıdaki açıklamalar doğrultusunda belirtilen uzaylar için bir plan üretmesi düşünülen planlayıcı algoritması Şekil 3.3'de verilmiştir.

```
planner(Başlangıç-durum, Hedef-durum){

while(gerçekleştirilmemiş bir hedef var ) { // her
        //gerçekleştirilmemiş hedef için kontrol kuralını işlet.
          Çalışma-uzayına-özel-kontrol-kuralını-işlet;
}}
```

Şekil 3.3 Planlayıcı Algoritması

Planlayıcıyı, işletim esnasında, dış aracıların etkileri sonucunda çalışma uzayında meydana gelebilecek değişiklikleri yakalamak için

çalışma uzayını izleyen bir görme sistemi desteklemektedir. Bu görme sisteminin özellikleri Bölüm 5'de açıklanacaktır.

3.3 İşletim Mimarisi ve Planlayıcının Mimarideki Yeri

İş planlayıcı, tanımlanan bir iş (bir başlangıç ve hedef durumu çifti) için sembolik seviyede adımlar dizini üretir. Üretilen adımlar planlama uzayına ait işleçler içerir. Literatürde kullanılan izlenceleyici (scheduler) kelimesi planlama kelimesi ile aynı anlamda kullanılıyor olmasına karşın aralarında farklar vardır.

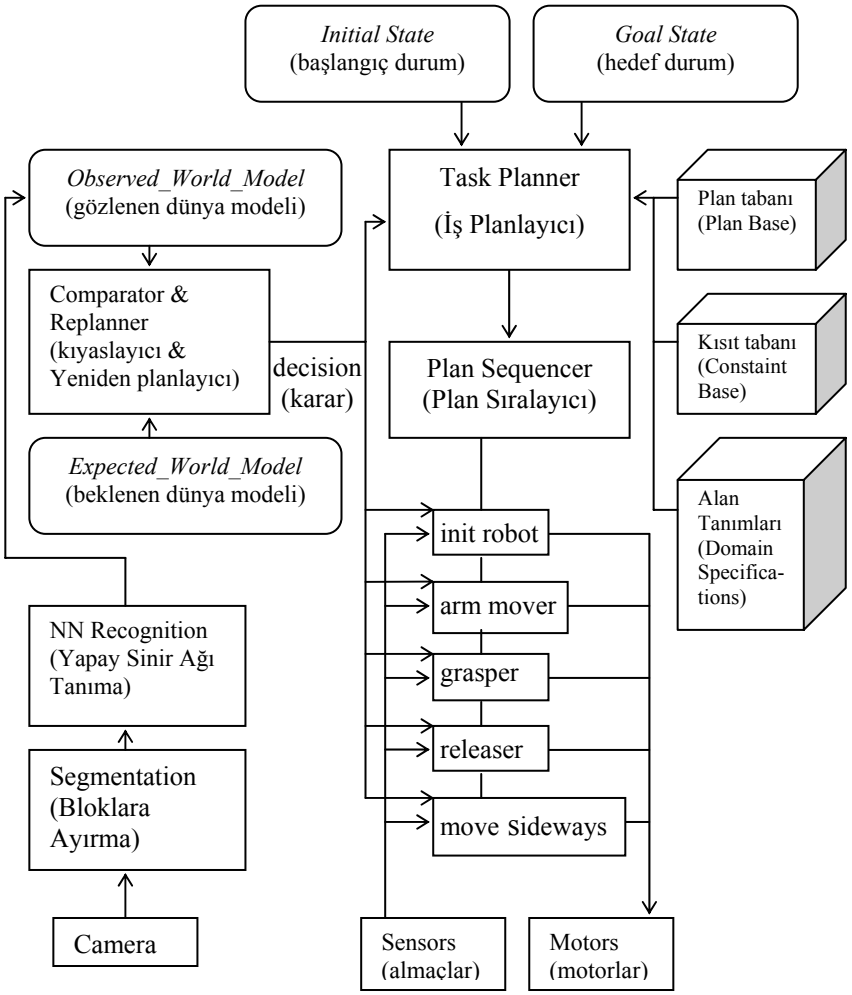
İzlenceleyici değişik iş veya aktivitelerin zaman önceliği itibari ile sıraya sokulması anlamını taşımaktadır. Öte yandan bir planlayıcı verilen bir işi ya da aktiviteyi önce alt işlere ayırır ve alt işleri gerçekleştirmesi muhtemel eylemler için zaman tabanlı sıralama yapar.

Bu çalışmada kullanılan planlama ve işletim mimarisi Şekil 3.4'de verilmiştir. VGP planlayıcısı Sebep-Sonuç analizi yöntemini kullanarak durum uzayı araması yapar. Verilen mimari incelenecek olursa, Yapar Sinir Ağı eğitme, başlangıç ve hedef durumların alınması, planlama, görüntü yakalama ve işleme, işlenmiş görüntülerden dünya modellerinin oluşturulması, yeniden planlayıcının devreye sokulması ve planların işletim için alt seviyede davranışlara çevrilmesi gibi işlerin olduğu görülür. Bu işleri en üst seviyede bir izlenceleyicinin (scheduler) zamanlaması gerekmektedir (Şekil 3.5).

İş izlenceleyici çalışma uzayından alınan örnek görüntülerle Yapay Sinir Ağı eğitimi yapar ve başlangıç ve bitiş durumlarını almak için ya kullanıcı girdisi bekler ya da kendisi bu durumları rastgele üretir.

Ardından eldeki başlangıç ve hedef durumları için planlayıcıyı çağırır. Planlayıcı Sebepler-Sonuçlar analiz yöntemini kullanarak bir plan üretir.

Üretilen plan bir plan veri tabanında saklanır. Çalışma uzayı tabanı, yeni uzayların eklenmesi ile genişletilebilecek şekilde tasarlanmıştır. Uzay eklemede dikkat edilecek nokta, eklenmek istenen uzayın bu çalışmada önemli bir yeri olan ve planlama/yeniden planlama esnasında mümkün olabildiğince minimum (optimal) seyahat mesafesi sağlayacak eylemler seçme esasına dayanan metodlar ile uzlaşıyor olması gerektiğidir. Başka bir deyişle, bir uzayı, uzay veritabanına eklerken o uzay için toplam seyahat mesafesinin minimum yapılmasının kullanıcı için önem taşıyor olmasına dikkat edilmelidir. Aksi halde bu çalışmada yer alan minimum seyahat mesafesini sağlayan eylemler dizinini seçme metodlarından faydalanılmamış olacaktır. Ayrıca, eklenmesi düşünülen uzayın gerektirdiği görme yetenekleri (örneğin üç boyutlu görme yeteneği gerektirip gerektirmemesi) uzay tabanını genişletmede önem taşımaktadır. Bu hususu bir kısıt olarak değerlendirmek mümkündür.



Şekil 3.4 Önerilen Mimari

```
main(){
NN_training (); // Görüntüyü segmentlere ayırmak ve nesneleri tanımak için
                //yapay sinir ağları eğitilmektedir.

domain_input(); // Hangi alanda (domain) çalışılacağı kullanıcı tarafından
                //girilmektedir.

suser_input () ;  //Başlangıç ve bitiş durumları ya kullanıcı tarafından
                //girilmekte ya da program içerisinde oluşturulmaktadır.

Expected_World_Model = Initial_State; // Beklenen dünya modeli
                                      //Expected_World_Model
                                      //matrisinde tutulur ve başlangıç
                                      //değeri olarak başlangıç durumu //
                                      // (Initial_State) atanır.

plan = planner(Initial_State, Goal_State);  //bir başlangıç ve hedef durum
                                            //için plan üretilir.
execute(plan);  //Üretilen plan işletilir.
}
```

Şekil 3.5 İş İzlenceleyici Algoritması

İzlenceleyici oluşturulan planı alt komutlara derlenmek ve robot tarafından işletilmek üzere işletim (execute) fonksiyonuna gönderir. Algoritması Şekil 3.6'da verilen işletim fonksiyonu parametre olarak aldığı plandaki her adımı plan sıralayıcısı (PlanSequencer) modülüne göndererek her adımın önce davranışlara (behaviours) derlenmesini sağlar. Bir plan adımı Şekil 3.4'de mimaride belirtilen "init robot, arm mover, grasper, releaser, move sideways" davranışlarından bir ya da birkaçına derlenir. Her davranış bir ya da daha fazla motor adı ve kodlayıcı adım sayısından oluşan alt komutlar içermektedir. Bir davranış bu şekilde bir plan adımı kapsamında aktif edilip sonlandırılabileceği gibi yeniden planlama modülünün verdiği bir karar sonucunda ya da almaçlardan gelen bilgiler sonucunda da aktif edilip sonlandırılabilir.

Örneğin bir plan adımı sonucu işletilmekte olan “arm mover” (kolu hareket ettir) davranışı almaçlardan gelen bilgiler sonucunda kolun bir nesneye çarpmak üzere olduğunun anlaşılması üzerine aniden sonlandırılabilir. Kolun bir nesneye çarpacak olması durumunda kolun çarpmasını önleyecek yeni bir yol planlamak için yeterli zaman olmadığından, sistem, almaçlardan gelen bilgilere dayanarak reaktif bir tepki gösterir. Reaktif tepkileri sağlayabilmek için davranışların her biri almaçlardan doğrudan bilgi alabilecek şekilde tasarlanmıştır ve almaçlardan gelebilecek sıra dışı her bilginin değerlendirilmesi planlayıcıya ait komutların işletilmesinden önceliklidir.

```
execute(plan){        //plan, içerisinde bir plan içeren dosyadır.

  for each plan step {

      plan_step = get_a_plan_step_from_plan(); // sırası ile her eylem işletilir.

      PlanSequencer(plan_step);   // İşletici seviyesi

      Update Expected_World_Model;       // Beklenen dünya modelini günle.

      Grab an image of the world;              //Çalışma uzayının bir görüntüsünü al.

      Process grabbed image to form Observed_World_Model; //Gözlenen dünya
                                          //modelini oluşturmak üzere alınan görüntüyü işle.

      dm=comparator(Expected_World_Model,Observed_World_Model);
      //Beklenen dünya modeli ile gözlenen dünya modeli arasındaki farkı bul.

      if (dm > 0) replanner();  // Eğer iki model arasında fark var ise yeniden
                                          //planlama yap.
       } //for
}
```

Şekil 3.6 Plan İşletim Algoritması

Bir plan adımının robot kolu tarafından işletilmesinin ardından iki işlem gerçekleştirilmektedir. Bunlardan ilki işletim esnasında tutulan bir iç dünya modelinin gerçekleştirilen bir plan adımının etkileri ile günlenmesidir (Update *Expected_World_Model*), Şekil 3.6. Eğer plan adımı sadece robot kolunun bir başlangıç noktasından ya da çalışma uzayı içerisindeki bir hücreden başka bir hücreye bir blok taşımaksızın hareketini içeriyorsa çalışma uzayında bir değişiklik olmayacaktır. Dolayısıyla bu hareketin iç dünya modeli üzerinde bir etkisi olmayacaktır. Ancak robot kolu plan adımını gerçekleştirdikten sonra bir bloğu bir hücreden başka bir hücreye taşımışsa, sisteme ait iç dünya

modelinde ilk hücrenin içeriği boşaltılmalı ikinci hücrenin içeriği taşınan bloğun tipi ile günlenmelidir. Sistemin iç dünya modeli beklenen dünya modeli (*Expected_World_Model*) olarak adlandırılmıştır.

Gerçekleştirilecek diğer işlem ise çalışma uzayının bir görüntüsünü almak (Grab an image of the world), Şekil 3.6 ve bir dış dünya modeli oluşturmaktır (Process grabbed image to form *Observed_World_Model*), Şekil 3.6. Bir görüntü alındıktan sonra görüntüde yer alan her hücre (dolayısıyla eğer varsa hücre içerisindeki blok) görüntüden ayrılır (segmentation), Şekil 3.4. Eğer ayrılan hücre bir blok içeriyorsa o bloğa ait etiketin üstündeki harf ya da şekil bir tanıma işleminden geçirilir (NN Recognition), Şekil 3.4. Bu şekilde çalışma uzayındaki her hücrenin bir blok içerip içermediği, içeriyorsa hangi tipte bir blok içerdiği belirlenecek ve iki ya da üç boyutlu bir matris veri yapısı kullanarak bir dış dünya modeli oluşturulacaktır. Matrisin her elemanı bir hücreye karşılık gelecektir. Dış dünya modeli bir kamera ile görüntü alma ve bu görüntüyü değerlendirme sonucunda oluşturulduğundan dolayı gözlenen dünya modeli (*Observed_World_Model*) olarak adlandırılır.

Eğer bir plan adımı başarı ile işletildiyse veya çalışma uzayı bir dış aracı tarafından birtakım değişikliklere uğratılmadıysa iç dünya modeli ile dış dünya modelinin aynı olması gerekmektedir. Ancak, bu modeller kısmen veya tamamen farklılık gösteriyorsa ya plan adımı başarı ile gerçekleştirilmemiştir ya da çalışma uzayı içerisinde beklenmedik bir olay meydana gelmiştir.

3.4 Yeniden Planlama

Beklenmedik olayların sebepleri dış aracılardır. Beklenmedik bir olay durumunda eldeki plan kısmen ya da tamamen geçerliliğini yitirmiş olacağından, işletim fonksiyonu içerisinden çağrılan yeniden planlama (replanner) modülü devreye girecektir.

Planlamada olduğu gibi yeniden planlama esnasında da maliyet optimizasyonu söz konusudur. Yeniden planlama modülü yeni bir plan üretmek için planlama modülünü çağırdığından planlayıcının performansındaki bir iyileşme yeniden planlayıcının performansındaki bir iyileşmeye karşılık gelmektedir.

Şekil 3.7'deki yeniden planlayıcı, beklenmedik bir olay sonrasında ne yapılacağına karar veren bir karar mekanizması konumundadır. Karar mekanizması aşağıda listelenen üç alternatif arasından seçim yaparak bir karara varır.

0) Bir optimal ya da alt-optimal çözümü olduğu garanti olan bir ara-durum planı işletmek.

1) Beklenmedik olay sonrası durumu beklenmedik olay öncesi duruma getirmek ve beklenmedik olay öncesi duruma kadar işletilmiş planı kaldığı yerden işletmeye devam etmek.

2) Beklenmedik olay öncesi duruma kadar işletilmiş planı iptal etmek ve beklenmedik olay sonrası durumdan hedef duruma yeni bir plan üretmek.

Bu üç alternatifin açıklamaları şu şekildedir.

0) Bu alternatif her zaman için diğer iki alternatiften önceliklidir. Bunun sebebi ara-durum planının optimal ya da alt-optimal bir çözüm olduğunun bilinmesidir. Şekil 3.7'deki algoritmada, *get_intermediate_state()* fonksiyonu bir problem (başlangıç ve bitiş çifti) için "doğru" değerini döndürürse, benzer bir problem (aynı başlangıç ve bitiş durum çifti) için daha önce plan tabanına kayıtlanmış optimal ya da alt optimal bir plan var demektir. Bu plan başlangıç durumdan hedef duruma giden bir plan olabileceği gibi başlangıç durumdan hedef duruma giden optimal ya da alt optimal bir yol üzerinde ulaşılmış bir ara durumdan hedefe giden bir plan da olabilir. *get_intermediate_state()* fonksiyonu "yanlış" değerini döndürürse yeniden planlayıcı diğer iki alternatif üzerinde durur.

1) Eğer beklenmedik olay sonrası durum ile beklenmedik olay öncesi durum arasındaki fark küçük ise beklenmedik olay sonrası durumu beklenmedik olay öncesi duruma getirmek ve beklenmedik olay öncesi duruma kadar işletilmiş planı kaldığı yerden işletmeye devam etmek kararı uygulanır. Bunun sebebi aradaki farkın küçük olması durumunda beklenmedik olay sonrası durumu beklenmedik olay öncesi duruma getirmenin maliyetinin düşük olacağı düşüncesidir. Beklenmedik olay öncesi durum ile sonrası durum arasındaki farkı tutan bir "dm" değişkeni vardır. Bu çalışma için "dm" değişkeni iki durum arasında kaç tane bloğun yerinin farklı olduğunu tutmaktadır. Bu farkın küçük ya da büyük bir fark olup olmadığı bir eşik değeri ile kıyaslanarak ortaya çıkar. Eğer "dm" değişkeni değeri, daha önce atanmış bir "thr" eşik değerinden daha küçük bir değere sahip ise beklenmedik olay olmadan önceki duruma gidilir ve orijinal planın geri kalan kısmı işletilir. Bu durumda, Şekil 3.7'de oluşturulan *new_plan* planı beklenmedik olaydan sonra ortaya çıkan durumu beklenmedik olayın etkileri olmadan önceki duruma getiren bir eylemler dizinidir. Bu esnada, *alter1_cost()* fonksiyonu,

geriye dönme için gereken planlama zamanını ve hedef duruma erişmek için kat edilmesi gereken toplam yolu hesaba katarak yeniden planlama için alternatif 1'i kullanmanın maliyetini hesaplar.

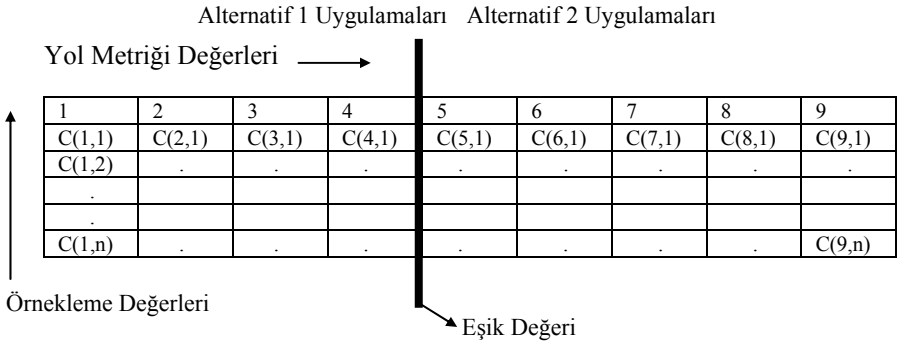
2) Eğer "dm" değişkeni değeri "thr" eşik değerinden büyük ise çalışma uzayı beklenmedik olay sonrası çok farklılaşmış demektir. Bu durumda uzayı beklenmedik olay öncesi duruma getirmektense beklenmedik olay sonrası durumdan hedef duruma yeni bir plan yapmak daha az maliyetli olacaktır. Bu durumda *alter2_cost()* hesabı ile alternatif 2'yi seçmenin maliyeti hesaplanır. Bu durumda alternatif 2'nin maliyetinin, aynı durumda alternatif 1'in uygulanmasının maliyetinden daha düşük olacağı düşünülmektedir.

```
replanner() {

(bool) yes = get_intermediate_state();
                        // plan veritabanından verilen  başlangıç ve hedef
                        //durumu için önceden saklanmış bir plan olup
                        //olmamasına göre boolean yes değişkenine değer
                        //sdöndürülüyor.

if(yes){ //eğer önceden saklanmış bir plan var ise if işletiliyor.

        new_plan = retrieve_plan_for_intermediate_state(); //veritabanından
                                                //ilgili plan alınıyor ve işletiliyor.
        execute(new_plan);}
else
  if(dm != 0 && dm < thr) {
                        //eğer önceden saklanmış bir plan yok ise
                        //ve dm değeri eşik değerinin altında ise gözlenen
                        //dünya modeli ile beklenen dünya modeli arasında yeni
                        // bir plan üretiliyor.
     new_plan=planner(Observed_World_Model, Expected_World_Model);
     execute(new_plan);
     execute(plan);
  }  //rest of plan will be executed.
  else {
                        //eğer dm değeri eşik değerinin üstünde ise gözlenen
                        //dünya modeli ile hedef durum arasında tamamen yeni
                        //bir plan üretiliyor ve eski plan göz //ardı ediliyor.
     alter2_cost();
     new_plan = planner(Observed_World_Model, Goal_State);
     execute(new_plan);
  }
}
```

Şekil 3.7 Yeniden Planlama Algoritması

Sistemin çalışmasının başlangıcında, eşik değeri gelişi güzel atandığı için yeniden planlayıcı doğru kararlar veremeyebilir. Doğru verilen karar sayısını arttırmak için zaman içerisinde, beklenmedik olaylar ortaya çıktıkça eşik değeri düzenlenir. Eşik değerini düzenlemek

için geçmişte ortaya çıkmış beklenmedik olaylara ait maliyet bilgilerinden faydalanılır. İkinci ve üçüncü alternatifler tarafından aktif olarak kullanılan eşik değerinin düzenlenmesi için kullanılacak geçmişte ortaya çıkmış beklenmedik olaylara ait maliyet bilgileri bir maliyet tablosunda tutulurlar. Bir maliyet tablosu P (> 0) ve Q (> 0) boyutlarına sahip iki boyutlu bir dizindir (Şekil 3.8). Bu tabloyu iki bölüme ayıran eşik değerinin solundaki maliyet değerleri alternatif 1'e sağındaki maliyet değerleri ise alternatif 2'ye aittir.



1	2	3	4	5	6	7	8	9
C(1,1)	C(2,1)	C(3,1)	C(4,1)	C(5,1)	C(6,1)	C(7,1)	C(8,1)	C(9,1)
C(1,2)	.	.	.	.	.	.	.	.
.								
.								
C(1,n)	.	.	.	.	.	.	.	C(9,n)

Şekil 3.8 Maliyet Tablosunun Görünümü

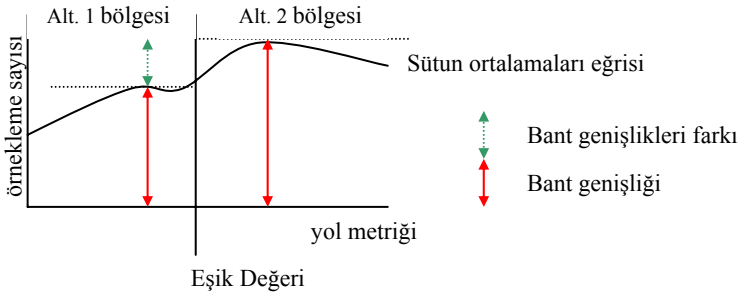
Maliyet tablosu üzerinde gösterilen yol metriği değerleri beklenmedik olay öncesi ve sonrasına ait durumların arasındaki farkı belirtmektedir. Örneğin, en üst satırda yer alan 1, 2 vb. değerleri beklenmedik olay öncesi ve sonrası farkın sırasıyla 1, 2 vb. olduğunu göstermektedir. Bu fark ya eşik değerinden büyük ya da küçük olacaktır. Daha önce de bahsedildiği gibi bu farkın eşik değerinden küçük olması durumlarında beklenmedik olay sonrası durumu beklenmedik olay öncesi duruma döndürme (alternatif 1) tercih edilmelidir. Aksi halde alternatif 2

seçilerek beklenmedik olay sonrası durumdan hedef duruma tamamen yeni bir plan üretilmelidir.

Eğer dm değeri bir plan adımı işletildikten sonra 0'dan farklı bir değere sahip ise yani beklenmedik olay öncesi ve sonrası durum arasında bir fark var ise tabloda o farka ait sütuna sıradaki boş hücreye olacak şekilde bir maliyet değeri yerleştirilir. Örneğin, C(1,1) tablo elemanı yol metriği (dm) değerinin 1 olduğu durumda maliyet tablosunun (1,1) koordinatlarındaki hücreye formül 1 kullanarak bir maliyet değeri hesaplanıp yerleştirildiğini belirtmektedir. Eğer yerleştirme işlemi sırasında (1,1) koordinatlarındaki hücre dolu olsaydı yine aynı sütün içerisinde yer alan ve boş olan sıradaki hücreye yerleştirme yapılmaya çalışılacaktı. Böyle bir durumda (1,1)'in dolu ve (1,2)'nine boş olduğunu varsayılırsa (1,2) koordinatlarındaki hücreye yerleştirme yapılacak ve maliyet gösterimi de C(1,2) olacaktı. Bu çalışmada beklenmedik olay öncesi ve sonrası durum arasındaki fark iki durum arasında farklı yerlerde olan blokların sayısı olarak hesaplanırken, başka çalışmalar için başka metrikler kullanılabilir. Eşik değeri 0 - P arasında (0 < eşik değeri < P) değer alır ve eğer gerekirse bir düzenleme algoritması kullanılarak düzenlenir.

Yeniden planlayıcı maliyet tablosunun tamamı dolmadan eşik değerini düzenlemez. Tablo tamamen dolduğunda ise tablodaki her sütunun ortalaması alınır. Eşik değerinin bağlı olduğu sütun ile bir önceki sütun arasındaki anormal bir fark eşik değerinin bir düzenleme gerektirdiğini belirtmektedir. Bunun temelinde yatan fikir eşik değerinin solunda ve sağında kalan bölgelerin ayrı ayrı ortalaması alınırsa ve sistem kararlı hale gelmemişse bu ortalamaların oldukça farklı olacağı fikridir. Bu ortalamalar arasındaki farkın eşik değeri öncesindeki ve sonrasındaki kolonlar arasındaki farka da yansıyacağı düşünülmektedir.

Bu varsayımın sebebi de maliyet tablosundaki her sütünün ayrı ayrı ortalaması alındığında eşik değerine kadar olan eğrinin bant genişliğinin eşik değerinden sonraki eğrinin bant genişliğinden farklı olacağı düşüncesidir. Şekil 3.9 bir maliyet tablosuna ait sütunlarından oluşturulan eğride eşik değerinin kararlı olmaması dolayısıyla oluşan bant genişlikleri farkını göstermektedir.



Şekil 3.9 Sütun Ortalamaları Eğrisi

Eşik değerini düzenleme eşik değerini önceden tespit edilmiş bir miktarda arttırma ya da azaltma ile sağlanır. Düzenlemenin bir arttırma ya da azaltma ile sağlanacağı kararına sezgisel bir metod ile varılır. Bu sezgi önceki kolon ile eşik kolunu arasında bir artış var ise alternatif 1'e ait bölgedeki maliyetlerin ortalamasının alternatif 2'ye ait bölgedeki maliyetlerin ortalamasından az olduğu sezgisidir. Bu sebeple bölge 1'deki ortalamayı arttırıp bölge 2'deki ortalamayı azaltmak üzere bölge 2'deki ilk sütun bölge 1'e dahil edilir. Bu da eşik değerinin arttırılması yani maliyet tablosu üzerinde sağa kaydırılması ile sağlanır. Eğer eşik değeri öncesi kolondan eşik kolonuna bir azalma var ise eşik değeri azaltılır yani maliyet tablosu üzerinde sola kaydırılır.

Dolayısıyla Şekil 3.7'de verilen yeniden planlama algoritmasında, eşik değerini (thr), yol metriği (dm) değeri ile kıyaslayan "if-else" yapısı içerisinde maliyet tablosunu günleyen, eşik değerini düzenleyen ve her düzenlemeden sonra tabloyu boşaltan komutlar olmalıdır. Eşik değerini düzenleyen algoritma Şekil 3.10'da verilmiştir.

Eşik Düzenleme Algoritması:

1. Maliyet tablosunun her sütunu için bir ortalama maliyet hesapla ve bu maliyetleri bir ortalama maliyet dizininde sakla.

2. Bir önceki sütun ile eşik sütunu maliyet ortalaması arasında anormal bir artma ya da azalma var mı diye kontrol et.

3. Eğer anormal bir fark gözleniyorsa
   Bu fark bir artış ise, eşik değeri sağa doğru adım-büyüklüğü kadar kaydırılır.
   Bu fark bir azalma ise, eşik değeri sola doğru adım-büyüklüğü kadar kaydırılır.
   Maliyet tablosu yeni maliyet değerleri kaydetmek üzere bosaltılır.

4. Eğer anormal bir fark gözlenmiyorsa
   Bir şey yapma.

Şekil 3.10 Eşik Düzenleme Algoritması

Bir maliyet fonksiyonun tanıma şöyledir:

$$C_p = c_1 T_p + c_2 E_p \qquad (1)$$

fonksiyonunda c_1 ve c_2 sabit katsayılardır.

T_P: Verilen bir p yolu için yeni bir iş planı üretme zamanı.

E_p: Verilen bir p yolu için robot kolunun üretilen iş planını gerçekleştirmesi için harcayacağı enerji.

Bu durumda, Alternatif 1 ve Alternatif 2 için maliyetler şu şekilde tanımlanır:

$$C_1 = c_1 T_1 + c_2 E_1$$

$$C_2 = c_1 T_2 + c_2 E_2$$

T_1, T_2 sırasıyla Alternatif 1 ve Alternatif 2 için yeni bir iş planı üretmek için harcanan zamanlardır.

E_1, E_2 sırasıyla Alternatif 1 ve Alternatif 2 için üretilen iş planlarını işletmek için robot kolu tarafından harcanan enerjilerdir.

Bu bölümde açıklanan eşik değerinin düzenlenmesi yaklaşımı aslında bir "öğrenme" niteliği taşımaktadır ve beklenmedik olaylar geliştikçe işletim hakkında daha çok veri toplanarak sistemin daha sağlıklı karar alması sağlanmaktadır. Kullanılan öğrenme yaklaşımı geçmiş işletimlere ait maliyetleri saklayarak daha sonra bu maliyetleri eşik değerini düzenlemek için kullanmaktadır. Buradaki çalışmada kullanılan öğrenme metodu bu çalışma kapsamında geliştirilmiştir. Bu metod da Bölüm 2'de verilen literatürdeki öğrenme metodları gibi bir T sınıfı (bloklar dünyası) işleri için sistemin P performansını (beklenmedik olay durumlarında doğru karar verme olasılığını) E eğitim tecrübesinden (zaman içerisinde toplanan maliyet değerlerinden) faydalanarak arttırmaktadır. Bölüm 2'de verilen öğrenme metodları bir öğrenme fonksiyonunun zaman içerisinde iyileştirilmesini "öğrenme" metodu olarak seçerken bu çalışmada eşik değerinin zaman içerisinde iyileştirilmesi "öğrenme" metodu olarak seçilmiştir. Bölüm 2'de verilen

öğrenme literatürüne ek olarak öğrenme işlemini maliyet tabanlı gerçekleştiren başka çalışmalar da vardır. Bu çalışmadaki maliyet öğrenme kavramı Öğrenme metodu hangisi olursa olsun sonuçta sistem performansında bir iyileşmenin gözlenmesini sağlamalıdır.

3.5 Alan Bağımsız Planlama Seviyesi ile Alan Bağımlı Seviye Arasındaki Arayüz

Alan bağımsız seviye ortak bir planlama mimarisini yani çalışma alanı ne olursa olsun uygulanması gereken sistemin akış adımlarını ifade etmektedir. Ancak bu adımlar uygulanırken bu adımların alana özel (domain specific) fonksiyonlar çağırması ve bu fonksiyonlara alana özel parametreler geçirilmesi ile genel ve ortak olan akış alana özel yapılar ile arayüzlenir.

Şekil 3.5'de belirtildiği gibi *domain_input()* fonksiyonu kullanıcıdan hangi alanda çalışılacağı bilgisini alır. Bu çalışma için alan "karışmış taşlar" ya da "konteyner" uzayı olabilmektedir ve bu alanlar Bölüm 4'te tanıtılacaklardır. *domain_input()* fonksiyonunun işletiminden sonra gelen her adım her alan için standarttır. Ancak bu adımlarda kullanılan veri yapıları örneğin başlangıç, bitiş durumları, beklenen dünya modeli (*Expected_World_Model*) seçilen alana göre farklılık gösterir. Ortak veri yapıları kullanılamamasının sebebi her alan için durum ve dünya bilgisinin farklılık göstermesidir. Örneğin bir problem için dünya modeli iki boyutlu bir matriste tutulabilirken diğer bir probleminde bu modelin üç boyutlu olmalısı gerekebilir.

Sonuçta üretilen plan farklı uzaylar için farklılık gösterecektir. Öte yandan üretilen planın Şekil 3.6'da verilen plan işletim algoritması

kullanılarak işletilmesinde kullanılan adımlar her alan için aynı olmasına rağmen her plan adımının işletilmesinde sonra kameradan alınan görüntü işlenerek oluşturulan gözlenen dünya modeli (*Observed_World_Model*) ve beklenen dünya modeli (*Expected_World_Model*) için alana özel veri yapıları kullanmak gerekecektir. Şekil 3.1'de gösterilen nesne veri yapısı ise çalışma alanları içinde yer alan nesneleri tutmaktadır ve bütün alanlar için kullanılmaktadır. Bunun sebebi çalışma alanlarının boyutları açısından farklılık göstermelerine rağmen içlerinde bulundurdukları nesneler açısından bir farklı göstermemeleridir. Nitekim çalışma alanlarındaki nesneler ya bloklar ya da bir robot koludur. Dolayısıyla örneğin bir gözlenen dünya modeli bir alan için

"object Observed_World_Model_Domain1[Row][Col]"

olarak tanımlanırken başka bir alan için

"object Observed_World_Model_Domain2[Row][Col][Height]"

olarak tanımlanır. "Row" bir matrisin satır, "Col" sütun ve "Height" yüksekliğini tutan değişkenlerdir. Bu modeller boyutları açısından farklı tanımlanmış olmalarına rağmen tür bakımından her ikisi de Şekil 3.1'de verilen nesne (object) veri yapısı şeklinde tanımlanmışlardır. Dolayısıyla her nesne, tip (type), ad (name), taşınmak istediği pozisyonun başka bir nesne tarafından işgal edilip edilmediği (destinationOccupancy), bulunduğu pozisyondan hedef pozisyona olan uzaklığı (distanceToDestination), mevcut koordinatları (curlocx, curlocy, curlocz), hedef koordinatları (destx, desty, destz) ve bir plan oluşturulurken bir plan adımı ile yerine yerleştirilip yerleştirilmediği (processed) bilgileri ile ifade edilmektedir.

Yine tüm alanlar için ortak olan dolayısıyla alan bağımsız olan yapılar arasında move_block (Bkz. Ek 2) ve move_arm (Bkz. Ek 3)

işleçleri bulunmaktadır. Bu işleçleri tanımlayan fonksiyonlar incelenecek olursa bu fonksiyonların (object *obj) parametrelerini aldıkları görülecektir. Her alan "object" veri yapısı türünden tanımlanabildiği için ortak işleçlere "object" türünde bir parametrenin geçirilmesi uygundur.

Öte yandan fonksiyonlar içinde alan türüne göre ve iki tane alan üzerinde çalışıldığı varsayılırsa *Initial_Domain1* ya da *Initial_Domain2*, *Goal_Domain1* ya da *Goal_Domain2*, *World_Model_Domain1* ya da *World_Model_Domain2* veri yapıları (matris türünde) kullanılmaktadır. *World_Model_Domain1* ya da *World_Model_Domain2* planlama esnasında dünya modelini tutan ve tüm planlayıcıların tutmakta uzlaştıkları dünya modelidir. Reaktif sistemler bu modeli tutmayı reddetmekte ve almaç bilgilerine dayanarak anlık bilgilere göre refleks hareketlerle işleri gerçekleştirmeye çalışmaktadırlar.

4. ALAN BAĞIMLI SEVİYE

4.1 VGP Mimarisinin İşletilebildiği Örnek Çalışma Uzayları (Alanlar)

Görme Destekli Planlayıcı (VGP), iki tane bloklar dünyası uzayı için uygulanmıştır. Bu uzayların ilki karışmış taşlar uzayı, diğeri ise bir limanda konteyner yükleme/boşaltmanın yapıldığı uzaydır. Bu uzaylar planlama ve yeniden planlama konularındaki fikirlerin gerçekleştirilmesine izin vermektedir

Her uzayın kendine ait arama kontrol kuralları vardır ve planlayıcı bir uzay için planlama yaparken o uzaya ait kontrol kurallarını etkinleştirir. Her iki problemde de robot kolunun kat edeceği yolun maliyetinin optimal olması dikkate alınmaktadır.

Problemler "karışmış taşlar problemi" (Şekil 4.1) ve konteyner yerleştirme/kaldırma problemidir. İlk problemin detayları (Yıldırım and Tunalı, 1999)'da verilmektedir. Problemin kısa bir tanımı ise aşağıdaki gibidir:

İki boyutlu bir uzayda bir başlangıç ve bir hedef durum alınır. Bloklar çalışma uzayına ait hücrelere her bir bloğun ağırlık merkezinden geçen eksen ile her hücrenin ağırlık merkezinden geçen eksen çakışacak şekilde yerleştirilmektedirler.

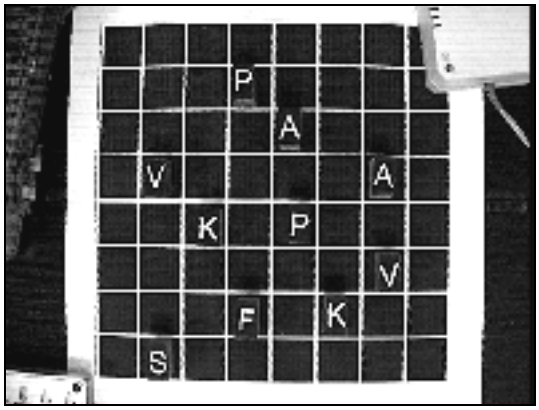
Çözülmek istenen problem, başlangıç durumdaki blokları başlangıç pozisyonlarından alıp hedef durumdaki pozisyonlarına minimum maliyet ile taşıyacak eylemler sırasını bulmaktır. Bu

problemdeki anahtar nokta aynı tür bloktan birden fazla olabilmesi ve bu bloklardan her birinin hedef pozisyonda birbiri yerine yerleştirilebilmesidir. Problemdeki diğer önemli noktalar ise şunlardır:

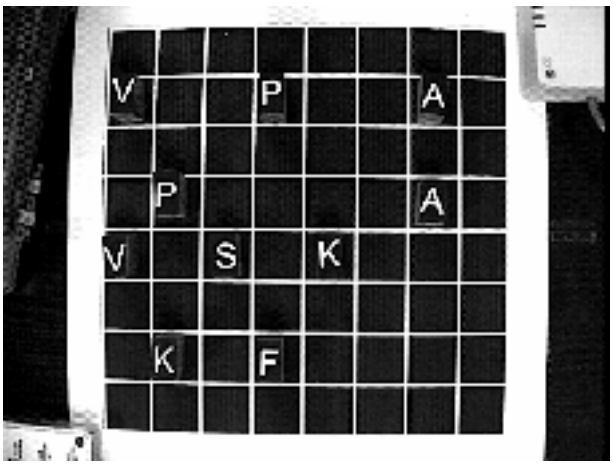
Bir bloğun hedef pozisyonu dolu olabilir.

Blokların başlangıç pozisyonlarından hedef pozisyonlarına taşınmaları esnasında blokların, çalışma uzayında yer alması muhtemel başka bir aracı tarafından karıştırılması ile beklenmedik olaylar ortaya çıkabilir ve eski eylemler sırası artık geçersiz olacağından yeni bir eylemler sırası bulmak (yeniden planlama yapmak) gerekebilir.

Eğer bu problemdeki bloklar bir satranç tahtası üzerine yerleştirilen satranç taşları ve çalışma uzayı da satranç tahtası olarak alınırsa bir bloktan birden fazla olabileceği rahatça görülebilir. Örneğin tahtanın başlangıç durumu için Şekil 4.1'de görüldüğü gibi 2 tane A etiketli, 2 tane K etiketli vb. blok (satranç taşı) olabilir. Bu çalışma kapsamında karışmış taşlar problemi için satranç oyununda olduğu gibi 6 farklı tipte taş olduğu varsayılmıştır. Tahta üzerinde toplam kaç adet taş olduğu satrançta olduğu gibi tahtanın boyutları ile ilişkilendirilmiştir. $n \times n$ boyutlarında bir tahta üzerinde maksimum $n \times n / 2$ taş olabileceği varsayılmıştır ($n > 0$). Bir karışmış taş planlama problemi için mevcut taş tiplerinden (A, V, Ş, K, P, F) kaç tanesinin tahta üzerinde yer alacağı ve her tipten kaçar adet olacağı (örneğin A'dan 2 adet, P'den 6 adet) program tarafından rast gele üretilmektedir. Tahta boyutlarını kullanıcı program girdisi olarak vermektedir.



Şekil 4.1. Karışmış Taşlar İçin Örnek bir Başlangıç Pozisyonu



Şekil 4.2 Karışmış Taşlar İçin Örnek bir Hedef Pozisyonu

Örneğin V etiketli taş (4,2) koordinatlarından alınıp hedef durumdaki (1,1) ya da (4,1) koordinatlarına yerleştirilebilir. Aynı durum (6,7) başlangıç koordinatlarındaki V etiketli taş için de geçerlidir. Hangi V etiketli taşın önce hedef durumdaki pozisyona yerleştirileceği ve hangi

taşın hangi pozisyona yerleştirileceği sonuçta toplam kat edilen yolun minimum yapılması ile doğrudan ilgilidir.

Önerilen çözümde, robot kolu sistem çalışmaya başlamadan önce bir başlangıç konumuna alınır. Şekil 4.3'de verilen algoritma gereği ulaşabileceği ve yerine konmamış bloklar arasından kendine en yakın olanını seçer. Bu bloğu koyabileceği alternatif pozisyonlar arasından, yerine konacak bloğa en yakın pozisyonda olana koymaya karar verir. Eğer seçilen en yakın pozisyon dolu ise o pozisyonu oradaki bloğu geçici olarak boş bir hücreye koyarak boşaltır ve yerine konmak istenen bloğu boşalan yerine yerleştirir. Daha sonra robot kolu, bulunduğu noktadan ulaşabileceği ve yerine yerleştirilecek bloklar arasından kendisine en yakın olanını seçerek yerine yerleştirmeye çalışır. Bu işlem yerine yerleştirilmemiş blok kalmayıncaya kadar devam eder. Planlama algoritmasının kural tabanlı bir şekilde modellenmesi işlemi, Şekil 3.1'de verilen planlama döngüsü içerisinde tüm bloklar yerine yerleştirilinceye kadar her defasında karışmış taşlar kontrol kuralının çağrılması ile gerçekleştirilir.

"karışmış taşlar" problemi için planlama algoritması Şekil 4.3'de verilmektedir.

1. Bütün nesneleri artan sırada ve currposrob (mevcut robot pozisyonu) değişkeni içindeki koordinatlara olan mesafeleri itibari ile sırala.

2. Her nesne için yol maliyetini hesapla.

Yol maliyeti = curposrob'dan nesneye olan uzaklık + nesneden minimal uzaklıktaki hedef pozisyona olan uzaklık.

3. En küçük seyahat maliyeti olan nesneyi seç.

4. Eğer trcost < infinity (kat edilen yol sonsuzdan küçük) koşulu sağlanmıyorsa, her nesne için occtrcost (hedefin dolu olması durumu maliyeti) hesapla.
occtrcost = currposrob'dan occ. (hedef pozisyonu dolduran) nesneye mesafe + occ. nesneyi en yakın boş pozisyona yerleştirme mesafesi + hareket ettirilecek nesneye geri gitmek için kat edilen mesafe + hareket ettirilecek nesnenin boşalan hedef koordinatlarına getirilmesi için kat edilen mesafe.

5.En küçük mesafeyi seç.

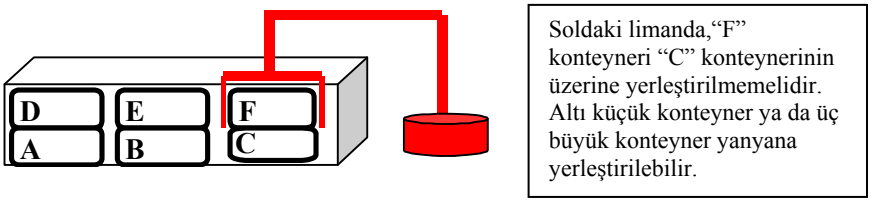
6. Yerine yerleştirilmemiş bir taş var ise adım 1'e git.

Şekil 4.3 En Yakın Blok Sezgisi Kullanan İş Planlama Algoritması

Karışmış taşlar problemi için oluşturulan bir plan işletilirken beklenmedik olayların olması söz konusu olabilir. Her beklenmedik olay sonrasında bir tahtanın ne kadar karışmış olduğunu gösteren yol metriği değeri hesaplanır. Bu değer bu çalışma için tahtanın beklenmedik olay olduktan sonra kaç taşının yerinin değiştiğini gösterecek şekilde seçilmiştir. Bu değer sistemin başlangıcında rast gele atanmış bir eşik değeri ile kıyaslanarak Bölüm 3.4'de bahsedilen alternatiflerden ilki çözüm getiriyorsa ilkinin değilse diğer ikisinden birisinin uygulanmasına karar verilir. Verilen karara uygun bir şekilde maliyet hesabı yapılır ve bu maliyet eşik seviyesinin düzenlenmesinde kullanılmak üzere maliyet tablosuna yerleştirilir (Yıldırım and Tunalı, 1999). Karışmış taşlar

problemi için alternatif seçme kararını vermede önem taşıyan eşik değerinin nasıl düzenlendiği Bölüm 3.4'de açıklanmış ve örnek bir sistem çalışması Bölüm 4.5'de verilmiştir.

Diğer problem ise Şekil 4.4'de bir görünümü verilen konteyner uzayına ait "konteyner yerleştirme" problemidir. Bu problemde, bir miktar konteyner gemiden alınıp limanda kendileri için ayrılan sahalara yerleştirileceklerdir. Ayrıca bir konteyner limandan alınıp gemiye yüklenebilir.



Şekil 4.4 Örnek Bir Konteyner Uzayı

Konteyner uzayında ise bir iş tanımında gerçekleştirilmemiş hedef sayısı bir ya da daha fazla olabilir.

İlk problem itibari ile bu çalışmada gerçekleştirilmek istenen satranç oynayan bir robot değil, satranç taşlarını kullanarak kaldır/koy işlemleri için planlama yapan ve ortaya çıkan hata durumlarında yeniden planlama yapabilen bir robot kolu mimarisi ve yazılımı geliştirmektir. Planlama için katledilen yolun minimum yapılmasına çalışılırken yeniden planlamada bir maliyet fonksiyonunun minimum yapılmasına çalışılmaktadır.

Söz konusu çalışma uzayı dinamiktir ve beklenmedik olaylar söz konusu olabilmektedir. Örneğin, tahta üzerindeki taşların yerleri, planlar işletilirken, çalışma uzayı içerisindeki başka bir aracı tarafından değiştirilebilmektedir.

4.2 Alana Özel Arama Kontrol Kuralları (Search Control Rules)

VGP planlayıcısı ya kolun hareketini "en yakındaki" hatalı yerdeki taşa hareket ettiren ya da hatalı yerdeki taşı "en yakındaki" hedefe yerleştiren sezgisel tabanlı bir kontrol kuralı kullanır. Aslında sezgisel yaklaşımı "en yakın taş" olan ve Şekil 4.3'te verilen ağaç arama algoritması kontrol kuralları kullanılarak gerçekleştirilmiştir.

Şekil 4.5'de karışmış taşlar problemi için kullanılan kontrol kuralı verilmiştir. Bir bloğun hedef pozisyonuna taşınması kararı verildiğinde bu kontrol kuralı içerisindeki A ön koşullarının mevcut durumda geçerli olup olmadığı kontrol edilir. A ön koşulları taşınacak nesnenin bir blok olup olmadığını, nesneye en yakın hedef pozisyona karar verilip verilmediğini ve hedef pozisyonun dolu olup olmadığını kontrol eder. Eğer tüm bu ön koşullar sağlanıyorsa, B eylemleri işletilir. B eylemleri bloğu boş olan hedef pozisyona taşır. Eğer hedef pozisyon boş değil ise, bu durum *empty_destination(&obj)* fonksiyonunda yakalanır ve dolu olan hedef pozisyonu, bu pozisyonu işgal eden bloğun en yakın boş sahaya taşınması ile boşaltılır (C eylemleri). Eğer hareket ettirilecek nesne blok değil de robot kolu ise bu durumda kol kendine en yakın olan ve hedef pozisyonuna götürülecek bloğu bulmaya çalışır. Böylece, X ön koşulu hareket ettirilecek nesnenin robot kolu olup olmadığını kontrol eder. Eğer öyle ise kolun hareket edeceği hedef pozisyonun dolu olup olmadığını (o

pozisyonda bir blok olması beklendiği için dolu olmalıdır) ve kolun kendine en yakın bloğu seçip seçmediğini kontrol eden eylemler ile robot kolunu en yakın bloğa hareket ettiren "move-arm" eyleminin de içinde bulunduğu Y eylemleri işletilir. Sistemin statüsü, planlayıcının mevcut durumuna bağlı olarak "arm-is-moving" veya "block-is-moving" olarak günlenir.

Konteyner uzayına ait planlama yapmak için genel planlayıcı algoritması içerisinden konteyner uzayına ait kontrol kuralları çağrılır. Hangi kontrol kuralının çağrılacağı kullanıcı tarafından yerleştirme ya da kaldırma işleminin seçilmiş olmasına göre belirlenir. Şekil 4.6 'da bu uzaya ait yerleştirme kontrol kuralı verilmektedir.

Konteyner uzayına ait yerleştirme kontrol kuralı içerisinde bulunan *find_closest_empty_destination_from_arm(&obj,&x,&y,&z)* önkoşulu basit bir kontrol yapan bir ön koşul olmayıp bir algoritma içermektedir. Bu ön koşula ait algoritma Şekil 4.8'de verilmiştir. Bir limanın Şekil 4.4'de gösterildiği gibi üç-boyutlu ve hücrelerden oluşan bir yapısının olduğunu varsayılacak olursa bu yapı içerisinde küçük boyutlu bir konteyner tek bir hücreye sığarken büyük bir konteyner iki hücrelik yer kaplamaktadır. Bu nokta göz önünde bulundurulduğunda Şekil 4.8'de verilen algoritmadaki ilk "if", limandaki (i, j, k) hücresinin dolu olup olmadığını, konteyner boyutunun küçük olup olmadığını (boyut 0) ve (i, j-1, k) hücresinin yani (i, j, k)'nın altındaki hücrenin "dolu" ya da "yer" olup olmadığını kontrol eder. Eğer alttaki hücre dolu ya da yer ise, yukarıdaki fonksiyon limana yükleme için (i, j, k) döndürür ancak değilse, fonksiyon başka boş bir hücre bulmak üzere aramaya devam eder.

İkinci “if” ise, büyük bir konteyner yerleştirilmesine çalışılır. Büyük bir konteyner yan yana iki tane hücre kaplar, örn. (i,j,k) ve (i,j+1, k). İkinci if ise konteyner boyutunun büyük olup olmadığını, (i,j,k) ve (i, j, k+1)’in boş olup olmadığını, k+1’in hücresel yapı içinde olup olmadığını ve (i, j-1, k) ve (i, j-1, k+1)’in dolu veya yer olup olmadığını kontrol eder. Eğer öyle ise, fonksiyon yerleştirme için (i, j, k) döndürür ve değilse başka bir boş hücre bulmak için aramaya devam eder.

Şekil 4.7’deki fonksiyonda, *get_topmost_object(obj)* fonksiyonu yerinden kaldırılacak konteyner sütununda en üstteki konteyner’ın y ekseni koordinatını alır. *get_temp_destination(obj, &x, &y, &z)* fonksiyonu en üstteki konteyner’ı geçici olarak koymak üzere geçici bir hücrenin koordinatlarını döndürür. *move_to_temp_destination(obj, x, y, z, toplevel)* fonksiyonu en üstteki konteyner’ı (x, y, z) koordinatlarındaki boş hücreye yerleştirir. Böylece, en üstteki konteyner yerinden oynatıldığı için boşalan yer fonksiyondaki bir sonraki deyim ile “boş” olarak atanır. Şekil 4.7’deki *make_container_approachable(object obj)* fonksiyonu bir konteyner’i başka bir konteyner’in altına yerleştirmek gerektiğinde ya da üstünde başka konteyner(lar) olan bir konteyner’ı bulunduğu yerden çıkarmak için üstündeki konteyner(lar)ı geçici bir yere yerleştirmek için kullanılabilir.

```
Mixed_Pieces_controlRule(){
Object obj;

A  if(obj.type == "BLOCK"){    // nesne tipi blok mu?
        find_closest_destination_from_block(obj);
                              // bloğa en yakın hedefi seç.

        if(empty_destination(obj)){ //en yakın hedef boş mu?

            move_block(obj);   // hedef boş ve nesneyi hedefe taşı.

            strcpy(obj.type, "ARM"); //sıradaki eylem blok taşımasa da
            // robot  kolunu hareket ettireceği için nesne tipini robot kolu
            // olarak ata.
        }                                                              B
        else {
            find_closest_empty_cell_from_destination(obj); //hedefe
                 // en yakın geçici bir hücre bul. Dolu hedefteki nesneyi
                // boş hücreye koy.

            move_block(obj); //hedefi boşalan bloğu hedefine koy.      C

            strcpy(obj.type, "ARM"); //bir blok yerine taşındığı için bir
                        //sonraki eylem robot kolunu bir blok taşımaksızın
                        // hareket ettirecektir.
    }
X  else if(obj.type == "ARM") { //nesne tipi robot kolu mu?

        find_closest_destination_from_arm(obj); //robot koluna en yakın
                                                //blok hücresini seç.
  Y
        move_arm(obj); //bloğun bulunduğu koordinatlara hareket et
        strcpy(obj.type, "BLOCK"); //bir sonra hareket ettirilecek nesne
      } //if.                        //tipini blok olarak ata.

    unprocessed = unprocessed – 1; //İşlenmemiş blok sayısını bir azalt.
    }
```

Şekil 4.5 Karışmış Taşlar Uzayı Kontrol Kuralı

```
void controlRule_container_insertion(){
int x,y,z;

if(!strcmp(obj.type,"container") &&
  !find_closest_empty_destination_from_arm(&obj,&x,&y,&z)) {

        insert_container(&obj,x,y,z);

        strcpy(status,"insertion-is-completed");
}

else {
        strcpy(status,"no-place-for-insertion");

       unprocessed --;
}
 }
```

Şekil 4.6 Konteyner Yerleştirme Uzayı Kontrol Kuralı

```
void make_container_approachable(object obj){
int x,y,z,j,toplevel = 0;

toplevel = get_topmost_object(obj);

for(j=toplevel; j > 0; j--) {

    get_temp_destination(obj, &x, &y, &z); // return temporary destination
                                           //coordinates in x,y,z.
    move_to_temp_destination(obj,x,y,z,toplevel);

   port[obj.curlocz][j][obj.curlocx].name=' '; //make topmost cell empty.
}}
```

Şekil 4.7 Üstü Dolu Bloğun (konteyner) Üstündeki Blokları Kaldırma Algoritması

```
int find_closest_empty_destination_from_arm(object *obj, int *x, int *y, int
*z){

int i,j,k;
for(i = 1; i <= depth; i++)   // konteyner uzayı üç boyutlu bir uzaydır.
  for(j = 1; j <= height; j++)
    for(k = 1; k <= width; k++){

        if (obj->size == 0 && port[i][j][k].name==' '
            && port[i][j-1][k].name != ' ')
                              //küçük bir blok olduğu, nesnenin altının yer ya da
                              //dolu olduğu kontrol edilmektedir.
            {
            *x=i;*y=j;*z=k; return 0;
            }
         if (obj->size == 1 && (k+1)<=width &&
            port[i][j][k].name==' ' &&
            port[i][j][k+1].name==' ' &&
            port[i][j-1][k].name != ' ' &&
            port[i][j-1][k+1].name != ' ')
                  //Eğer blok yerleştirildiğinde liman
                  //sınırları dışına çıkmıyorsa ve blok için yanyana boş iki hücre
                  //varsa ve hücrelerin altı dolu ise ilk hücrenin koordinatları
                  //bloğun yerleştirileceği koordinatlar olarak döndürülür.

           {
           *x=i;*y=j;*z=k; return 0;
           }
  } //for
      return 1; }
```

Şekil 4.8 Konteyner Uzayı İçin En Yakın Boş Hücre Bulma Algoritması

Sonuç olarak kontrol kuralında adı geçen işleçlerden oluşan bir plan oluşturulur. Karışmış taşlar problemine ait bir plan kütüğünün içeriği Şekil 4.9'da verilmektedir.

```
arm   0 0 : 4 1
K 4 1 : 3 1
arm   3 1 : 1 1
V 1 1 : 4 2
arm   4 2 : 4 4
P 4 4  : 1 2
```

Şekil 4.9 Örnek bir Plan Kütüğü

Tanımlanan iki problem incelendiğinde, üretilen eylemlerin sırası harcanan enerjiyi (kat edilen toplam yol ile ilişkili olarak) optimal yapmada önem taşımaktadır. Her iki problemde de bir başlangıç ve hedef durumu tanımlanmakta ve bir iş tamamlandığında, robot kolunun kat ettiği yolu minimum yapan bir plan dizini üretilmektedir. Optimal maliyetin sağlanması, robot kolunun çalışma uzayını değiştiren beklenmedik olayların ele alınmasında dikkate alınmalıdır. Böylece, kat edilen toplam yolu ve de planlama zamanını minimum yapan eylemlerin seçimi yeniden planlama için geçerlidir.

4.3 Plan Üretilmesi ve İşletilmesi Örneği

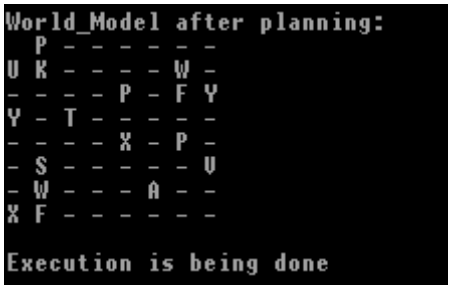
Başlangıç ve hedef durumları ya kullanıcıdan program girdisi olarak alınır ya da program tarafından üretilir. Eğer kullanıcıdan girdi

olarak alınmayıp program içerisinde üretiliyorsa koordinatlar rast gele üretildiğinden daha önceden üretilmiş koordinatların bir daha üretilmediğinin kontrolünü yapmak gerekir.

Şekil 4.10'daki iki matris örnek bir karışmış taşlar problemi için başlangıç ve bitiş durumlarını göstermektedir. Burada farklı planlama algoritmaları kullanılabilir. Önemli olan robot planlama ve yeniden planlama mimarisinin farklı planlama algoritmalarını kullanabilecek esneklikte olmasıdır.

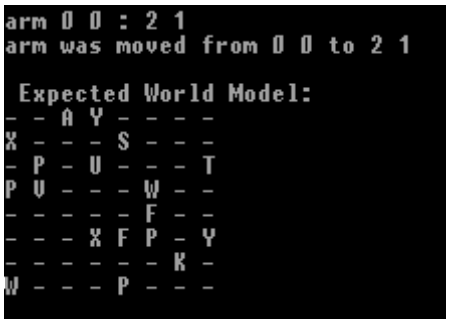
Şekil 4.10 Rasgele Üretilmiş Başlangıç ve Bitiş Durumları

Planlama süresince, *World_Model* matrisi günlenmektedir. Planlamanın sonunda, *World_Model*, *Goal* matrisine eşit olmalıdır. Eşitlik, Şekil 4.10 ve Şekil 4.11'deki *Goal* ve *World_Model* matrislerinin kıyaslanması ile gözlenebilir.



Şekil 4.11 Planlamadan Sonra Dünya Modeli

Şekil 4.11'de gösterildiği gibi, planlama yapıldıktan sonra işletim başlar. İlk komut robot kolunun başlangıç (0, 0) pozisyonundan en yakın taşa hareketidir. *Expected_World_Model*'da taşların yeni pozisyonları günlenir. İlk kol hareketi *Expected_World_Model*'da bir değişikliğe yol açmaz çünkü herhangi bir taş hareket ettirilmemiştir (Şekil 4.12).



Şekil 4.12 Kol Hareket Ettikten Sonra Beklenen Dünya Modeli (*Expected_World_Model*)

İlk hareketten sonra, bir kamera görüntüsü alınır ve bir *Observed_World_Model* oluşturulur. *Expected_World_Model* ile *Observed_World_Model* kıyaslanır. Eğer arada bir fark yok ise bu her

şeyin beklendiği gibi ilerlediği anlamını taşır. Örneğin Şekil 4.13'de X bloğu (2, 1)'den (5, 5)'e hareket ettirilir. *Expected_World_Model* günlenir ve ekranda görüntülenir.

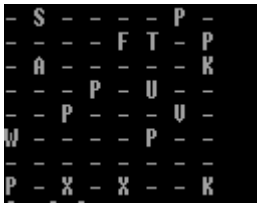
Şekil 4.13 Plan Kütüğündeki Sonraki Adım İşletildikten Sonra Beklenen Dünya Modeli (*Expected_World_Model*)

4.4 Beklenmedik Olay Örneği

VGP planlayıcısı için aşağıdaki örnek verilebilir. Yazılım başlangıç ve de bitiş durumlarını rast gele olarak üretir. Karışmış taşlar problemi için örnek bir planlamaya ait başlangıç ve hedef durumları Şekil 4.14'de verildiği gibidir.

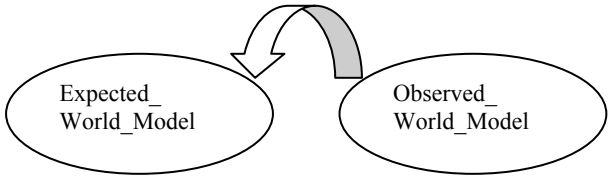
Şekil 4.14 Plan Tarafından Üretilmiş Başlangıç ve Bitiş Durumları

Yazılım daha önce açıklanan şekilde planlama yapar. Bu örneğe ait planlama sonucunda oluşturulan ilk plan adımı " ARM 0 0 1 2" eylemidir ve robot kolunun (0, 0) başlangıç koordinatlarından (1, 2) koordinatlarına hareket ettirileceğini göstermektedir. (1, 2) koordinatlarında hedef pozisyonuna götürülecek olan ve robot kolunun başlangıç pozisyonuna en yakın olan taş vardır. Bu hareket tamamlandıktan sonra, bir dış aracı tarafından beklenmedik bir olay gerçekleşir ve tahta üzerindeki taşların bazılarının yerleri değiştirilir. Bir dış aracı, robot kolu dışında hareket yeteneği olan ve çalışma uzayı içerisindeki blokların yerini değiştirerek beklenmedik olaylar yaratabilen bir aracıdır. Karıştırma işleminden sonra tahtanın görünümü Şekil 4.15'de olduğu şekildedir.



Şekil 4.15 Karıştırmadan Sonra Tahtanın Durumu (Gözlenen Dünya Modeli - *Observed_World_Model*)

Tahta üzerinde bir karışıklık olduğundan ve bu karışıklığın farkına varıldığından dolayı, bir planlama kararı verilmesi gerekmektedir. Bir ara-durum planı saklı değilse tahta durumu hatanın olduğu durumdan önceki duruma çekilmeli ve orijinal plana kaldığı yerden devam mı edilmeli yoksa gözlenen durumdan hedef duruma yeni bir plan mı üretmelidir? İşletimin başlangıcında, yol metriği, yani yeri değiştirilmiş olan taşların sayısı daha önceden değeri atanmış olan bir eşik değeri ile kıyaslanmaktadır. Görüldüğü üzere, tahtanın beklenen durumu ile gözlenen durumu arasında 15 taş değişikliği farkı vardır. Yani, plan kütüğündeki ilk hareket işletildikten sonra tahtanın başlangıç durumunda bir değişiklik olmayacaktır. Dolayısıyla gözlenen durum ile tahtanın başlangıç durumu aynı olacaktır. Tahtanın bir durumu üzerine beklentiler *Expected_World_Model* olarak adlandırılır. Tahtanın gözlenen durumu ise *Observed_World_Model* olarak adlandırılır. Eğer her şey beklendiği gibi yürürse aradaki fark sıfır olmalıdır. Bu durumda, iki model arasındaki fark 15'tir. Yani 15 tane taş beklenen pozisyonlarında değildir. Aradaki fark yine 15 olan eşik değeri ile karşılaştırıldığına, beklenmedik olay olmadan önceki duruma dönme kararı verilir ve *Observed_World_Model*, *Expected_World_Model*'a dönüştürülür. Bu durumda yeni başlangıç ve bitiş durumları Şekil 4.16'da gösterilen *Observed_World_Model* ve *Expected_World_Model* çiftine karşılık gelir.



Şekil 4.16 Yeni Başlangıç ve Hedef Durumu

Şekil 4.16'da görüldüğü gibi, tahtanın durumu tekrar orijinal başlangıç durumundadır. Bu durumda orijinal plan tekrar işletilebilir.

Ancak, beklenmedik olay plana ait birkaç adım işletildikten sonra ortaya çıkabilir ve böylece *Expected_World_Model* orijinal başlangıç durumundan farklı olacaktır. Bu durumda bazı adımlar zaten işletilmiş olduğu için orijinal planın geri kalan kısmı işletilecektir. Sonraki plan adımına ait olan işaretçi (pointer), yazılım içinde bir değişkende tutulmaktadır.

4.5 Eşik Düzenleme Çalışması Örneği

Verilecek örnek için 16 × 16 boyutlarından bir tahta için çalışılmıştır. Tahta üzerinde 128 adet taş bulunmaktadır. İki renkte taş olduğu varsayılmış ve her renk için 64 taş olduğu kabul edilmiştir. Tüm taş tiplerin tahta üzerinde olduğu varsayılmış ve her tipten kaçar adet

olduğu Şekil 4.17'de belirtilmiştir. Taşların iki renk olması bir sorun yaratmamakta olup farklı renkteki bir taş tipi farklı bir tip olarak düşünülebilir. Nitekim siyah bir Vezir hiçbir zaman hedef pozisyonda beyaz bir vezirin yerine yerleştirilmeyecektir. Sonuç olarak bir renkteki taşların hedef pozisyonu için de yine o renkteki taşlar seçilirse problem istenen biçimde şekillendirilmiş olacaktır.

Oluşturulan maliyet tablosunun boyutları 80 X 5 olarak alınmıştır. Bu problem için kullanılan yol metriği beklenmedik olay olduktan sonra beklenmedik olay öncesi ve sonrası durum arasındaki farkı yani kaç tane taşın yerinin değiştiğini belirtmektedir. Bu sayıda yerine beklenmedik olay öncesi yerleştirilmiş ancak beklenmedik olay sonrası yerlerinden oynatılmış taşlar da dahildir. Yol metriğinin (dm) aldığı değerler $0 < dm < 80$ şeklinde ifade edilebilir.

Beyaz Grup		**Siyah Grup**	
Taş türü	*Taş adedi*	Taş türü	*Taş adedi*
V	4	V	4
S	4	S	4
A	8	A	8
K	16	K	16
F	16	F	16
P	16	P	16

Şekil 4.17 Her Çeşit Taştan Kaçar Adet Olduğunu Gösteren Tablo

Maliyet tablosunda her sütün için 5 adet beklenmedik olayın maliyet değeri kaydedilir. Bu örnekteki maliyet tablosunu doldurmak için 400 adet maliyet hesabı yapılmış olmalı yani 400 kere beklenmedik olay gerçekleşmiş olmalıdır.

Bu çalışmaya özel olmak üzere eşik değerlerinin 1'in üstüne çıkması ve –1'in altına inmesi sırasıyla anormal bir artış ve azalışı ifade etmektedir. Başlangıçta eşik değeri 5 olarak atanmıştır ve eşik kolonu öncesindeki kolandan eşik kolonuna olan bir artış için eşik değeri 10 arttırılırken önceki kolon ile eşik kolonu arasındaki azalış için eşik değeri 10 azaltılacaktır. Şekil 4.18'de eşik değerinin verilen değerleri için eşik kolonunun solundaki kolondan, eşik kolonuna olan değişimler verilmektedir. Şekilde görüleceği gibi bir artış kaydedildikçe eşik değeri 5'ten itibaren sırasıyla 15, 25, 35 ve 45 değerlerine çıkarılmıştır. Eşik değeri 45'e çıkarıldığında eşik değeri öncesi kolon ile eşik değeri kolonu arasındaki fark –3.08'e düştüğü için eşik değeri 35 değerine geri çekilmiştir ve eşik değeri 35 olarak stabilize edilmiştir. Bu durumu Şekil 4.18'de gözlemlemek mümkündür.

Eşik Değeri	Değişim Değeri
5	+3.16
15	+7.68
25	+7.38
35	+5.37
45	-3.08

Şekil 4.18 Eşik Kolonu Öncesi Kolondan Eşik Kolonuna Olan Değişimler

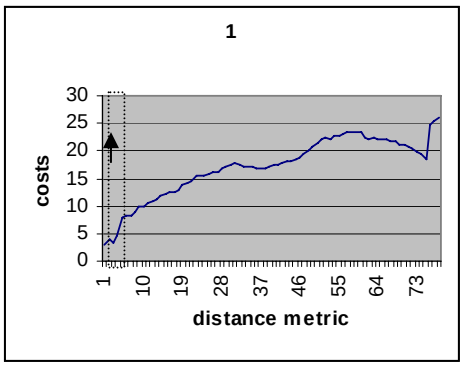
Şekil 4.19, 4.20, 4.21, 4.22, 4.23 ve 4.24 yol metriği (distance metric) değerlerine karşılık maliyet değerlerinin (costs) verildiği tablolardır ve eşik seviyesinin solunda kalan değerler alternatif 1 için ortalama maliyetleri ve eşik seviyesinin sağında kalan değerler ise alternatif 2 için ortalama maliyetleri gösterirler. Düşük yol metriği değerlerinde alternatif 1'in tercih edilmesi ve yüksek yol metriği değerlerinde alternatif 2'nin tercih edilmesinin sebebi yol metriği düşük

değerlere sahip iken bulunulan beklenmedik olay sonrası durumdan beklenmedik olay öncesi duruma dönmenin ve orijinal planı kalınan yerden işletmenin (alternatif 1) daha az maliyetli bir çözüm olmasıdır. Beklenmedik olay sonrası durum beklenmedik olay öncesi durumdan oldukça farklı ise yani yol metriği yüksek bir değere sahip ise beklenmedik olay sonrası durumu beklenmedik olay öncesi duruma taşımak daha çok maliyetli olacaktır. Bunun yerine alternatif 2'yi işletmek yani bulunulan durumdan hedef duruma yeni bir plan üretmek daha az maliyetli olacaktır.

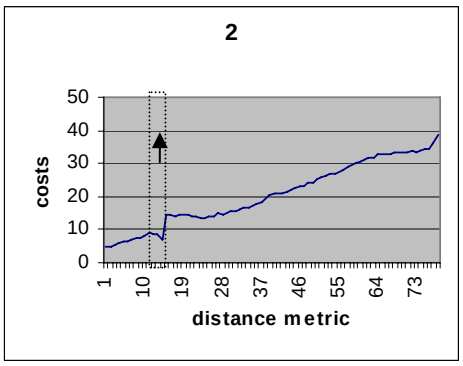
35 değerindeki stabilizasyonun aşağıdaki grafik alanlarının kıyaslanması cinsinden incelenecek olursak işe Şekil 4.23 ve Şekil 4.24'de gösterilen grafiklerin alanlarını kıyaslamakla başlamak gerekir. Şekil 4.24, Şekil 4.23 için yapılan örneklemelerde tüm yol metriği değerleri için alternatif 2'nin seçilmesi sonucu alınan ortalamaları göstermektedir. Şekil 4.23 için hesaplanan grafik altı alan 1281 ve Şekil 4.24 için hesaplanan alan ise 1336'dır. Aradaki fark 55'dir. Öte yandan Şekil 4.22 için hesaplanan alan 1522'dir. Eğer Şekil 4.22 için tamamen alternatif 2 değerleri kullanılacak olsa oluşturulan grafikte hesaplanan alan 1602 olurken iki grafik arasındaki fark 80 olmaktadır. Bu fark önce elde edilmiş olan 55 değerinden büyüktür. Bunun sebebi Şekil 4.22'nin eşik seviyesinin stabilize olduğu değere sahip olmasıdır.

Eğer mevcut bir durumda ön koşulları sağlanan çok sayıda karar adımları var ise ve sadece bunlardan bir tanesinin işletilmesi söz konusu ise, bu adımlardan toplam maliyeti en az kılan seçilir. Buna ek olarak, beklenmedik olay durumlarında, sadece beklenmedik olayın etkilerini ortadan kaldırmak için geriye gitmek (backtrack) yerine geriye gitme ile doğrudan hedef duruma gitme arasında bir karar verme mekanizmasının çalıştırılması maliyet optimizasyonu açısından etkinliği arttırır. Bu

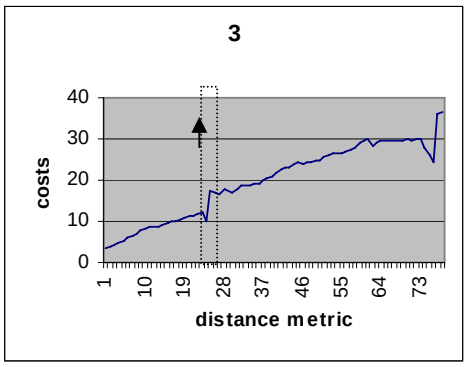
mekanizma daha önceden kullanılmış planların indekslenerek saklanması ile zenginleştirilir. Daha önceden tasarlanmış bazı ara durumlar için bulundukları durumdan hedef duruma gittiği bilinen eylem dizinleri (yollar) saklanmıştır. Eğer bu durumlardan herhangi biri ile beklenmedik olaydan sonraki durum arasındaki fark belli bir eşik değerinin altında ise doğrudan, bulunulan durumdan hedef duruma olan eylemler dizini işletilir.



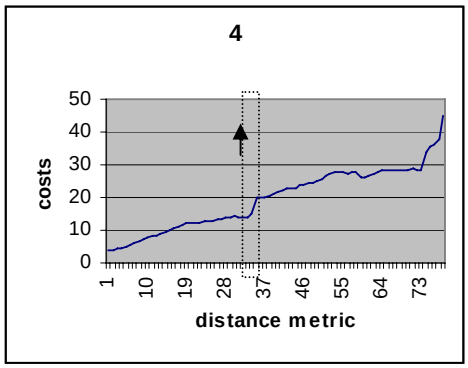
Şekil 4.19 Eşik 5 için maliyet tablosu ortalamaları.



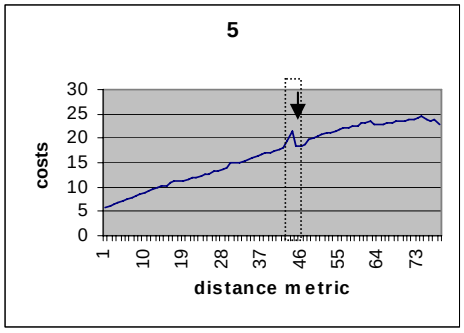
Şekil 4.20 Eşik 15 için maliyet tablosu ortalamaları.



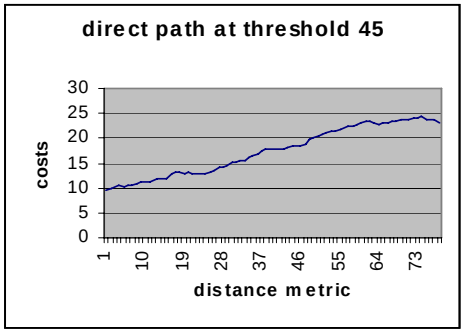
Şekil 4.21 Eşik 25 için maliyet tablosu ortalamaları.



Şekil 4.22 Eşik 35 için maliyet tablosu ortalamaları.



Şekil 4.23 Eşik 45 için maliyet tablosu ortalamaları.



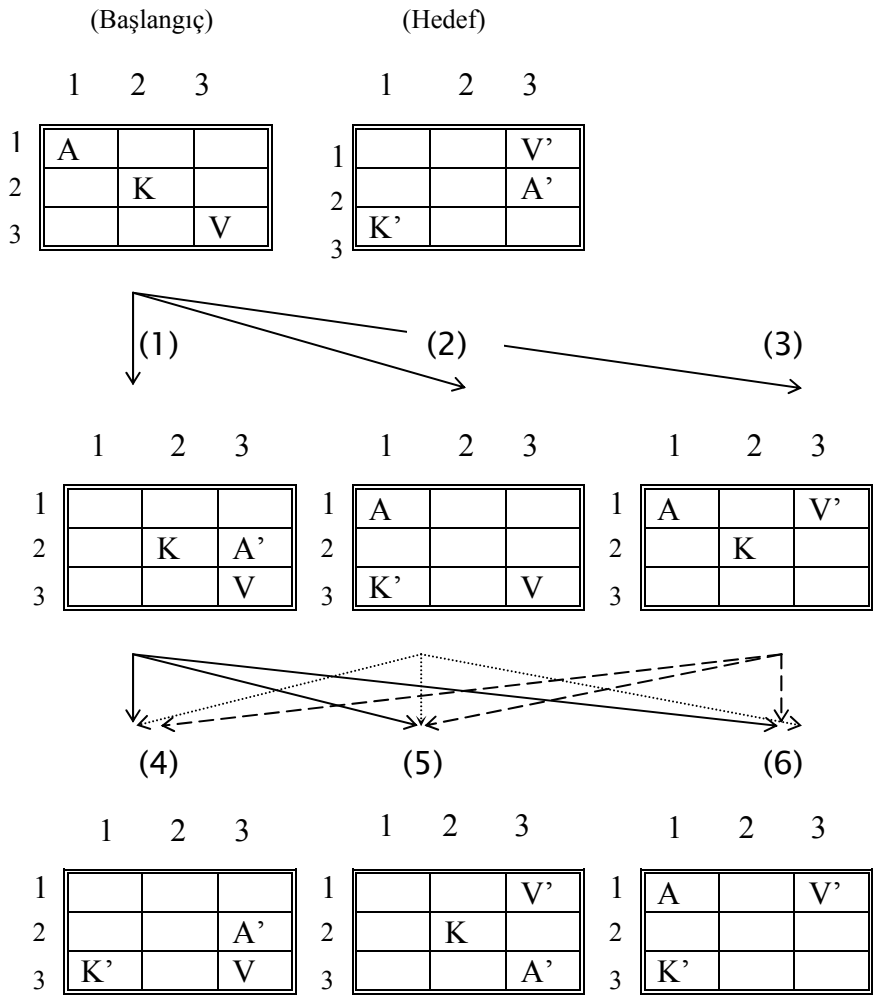
Şekil 4.24 Şekil 4.23'de tüm eşik değerleri için alternatif 2 kullanılması sonucu elde edilen maliyet tablosu sütun ortalamaları.

4.6 Karışmış Taşlar Problemi İçin "En Yakın Taş" Sezgisine Alternatif Çözüm: Dijkstra'nın En Kısa Yol Çözümünün Karışmış Taşlar Problemine Uyarlanması

Dijkstra'nın önerdiği en kısa yol problem çözümünde, çalışma uzayındaki her nesne için başlangıç ve hedef durumları kesin olarak bilinmektedir. Karışmış taşlar probleminde ise, bir taş tipinden birden fazla olabileceği ve her taş kendi tipinde bir taşın yerine konabileceği için her nesne için hedef durumlar (pozisyonlar) kesin olarak bilinmemektedir. Eğer bu problemdeki her taş için belli bir başlangıç ve bitiş pozisyonu tanımlanırsa en kısa yol bulma algoritması bu problemin çözümü için de kullanılabilir.

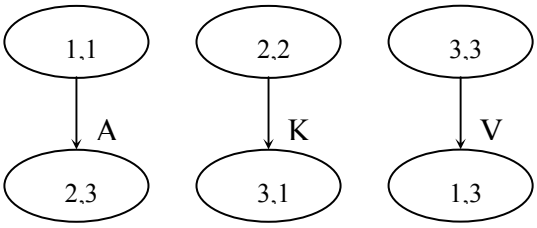
Dijkstra'nın en kısa yol çözümü, bir başlangıç ve hedef çifti olarak tanımlanmış bir problemin çözümü için bir arama ağacı üzerinde bir yol aramayı gerektirir. Böyle bir arama yapmanın sebebi problemin çözümü olabilecek çok sayıda yolun olması ve bu yollar arasından bir seçim yapılması gerekliliğidir. Bu problem yaklaşımı, örnek bir ağaç üzerinde çözüm detayları açıklanarak anlatılabilir. Şekil 4.25 bir başlangıç ve hedef durum arasında gidilebilecek yolların tümünü gösteren örnek bir arama ağacıdır. (1), (2), (3) , (4), (5) ve (6) bu ağaç üzerindeki ara durumlardır. (1), (2) ve (3) durumları sadece bir taşı hedef koordinatlarına yerleştirdikten sonra elde edilirken (4), (5) ve (6) durumları iki taşı hedef koordinatlarına yerleştirdikten sonra elde edilmiştir.

Örneğin:



Şekil 4.25 Örnek bir Başlangıç ve Hedef Durum Arasında Oluşması Muhtemel Yolların Tümü (Örnek bir Arama Ağacı)

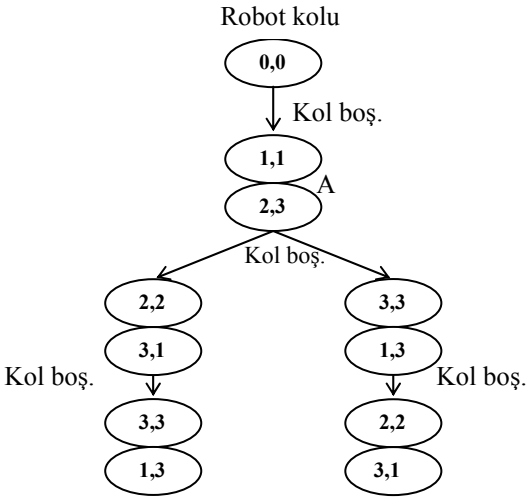
Öncelikle, robot kolunun taşları hedef koordinatlarına yerleştirirken kat edeceği mesafenin sabit bir toplam olduğunu varsayalım ve yukarıdaki örnek için çalışalım. Bu durumun gösterimi Şekil 4.26'daki gibidir.



Şekil 4.26 Hedef Koordinatlarına Yerleştirilebilecek Taşlar ve Başlangıç ve Hedef Koordinatları

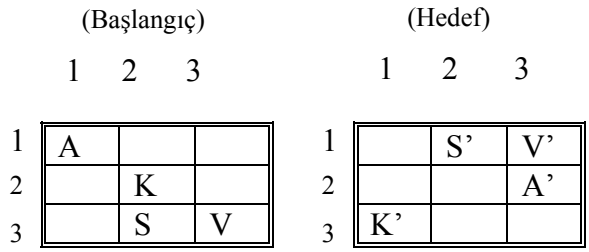
Ancak, planlamanın başlangıcında robot kolu önceden tanımlı bir pozisyondadır. İlk olarak bu pozisyondan (1, 1), (2, 2) veya (3, 3) koordinatlarından birine gitmesi gerekmektedir. Eğer örneğin (1, 1)'e gitmeyi seçerse, sonraki adımda mutlaka (2, 3)'e gidecektir. Daha sonra ise hangi yolun toplam seyahat mesafesini minimum yapacağına bağlı olarak ya (2, 2)'ye ya da (3, 3)'e gidecektir.

Öncelikle robot kolunun başlangıç pozisyonuna en yakın koordinatlardaki taşa gittiğini varsayalım. Eğer başlangıç koordinatları (0, 0) ise ilk hareket (1, 1)'e olacaktır. (1, 1) den (2, 3)'e hareket *sabit harekettir* (Şekil 4.27). Yani bu hareket bir alternatif içermemektedir. Sabit bir hareket Dijkstra'nın en kısa yol algoritmasındaki tek bir ara düğüm olarak düşünülebilir. Yolların sayısı (n-1)! ile bulunurken, n, taş sayısına eşit olan sabit hareketlerin sayısıdır.



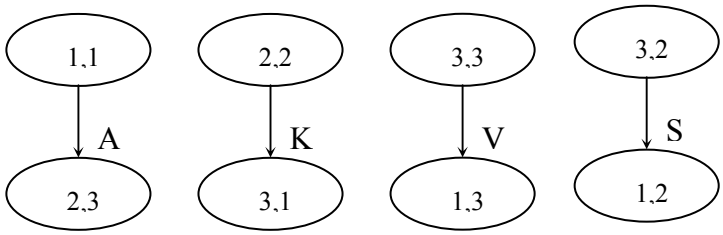
Şekil 4.27 Sabit Hareketler

(Sabit hareketlerin sayısı –1) hedef duruma olan yolların sayısına eşittir. Eğer taşların sayısı Şekil 4.28'de olduğu gibi 3'ten 4'e çıkarılırsa ağaç üzerindeki dalların sayısı daha da çoğalacaktır.



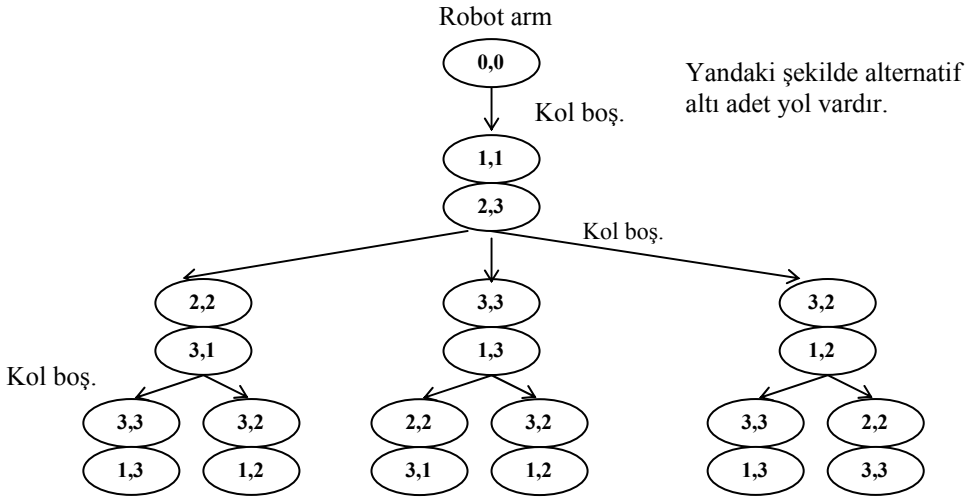
Şekil 4.28 Örnek bir Başlangıç ve Hedef Durumu

Şekil 4.29 örneğine ait sabit hareketler Şekil 4.28'dedir.



Şekil 4.29 Hedef Koordinatlarına Yerleştirilebilecek Taşlar ve Başlangıç - Hedef Koordinatları

Yol ağacının görünümü Şekil 4.30'dadır.



Şekil 4.30 Arama Ağacı

Tahta üzerindeki taşların sayısını daha fazla olması da söz konusu olabilir. Sayının daha fazla olduğu durumlarda ağaç üzerindeki yolların sayısı da artış gösterecektir. Örneğin eğer taşların sayısı 20 olarak alınırsa mevcut yolların sayısı, (20-1)! = 19×18..... ×1 > 335,221,286,400'dir. Eğer taşların sayısı 15 ise, mevcut yolların sayısı, 14! = 87,178,291,200'dir. Bunlara ek olarak sabit hareketlerin alternatiflerinin olması durumunda oluşacak yolların sayısının daha da artacağı gözlenebilir. Örneğin, Şekil 4.28'de tahta üzerinde kale yerine bir şah daha olduğu varsayılacak olursa, sıra bir şahın hareket ettirilmesine geldiğinde şahın hareket edebileceği iki alternatif hedef pozisyonun da hesaba katılması gerekmektedir. Böyle bir durumda alternatif yol sayısının en az her taştan Şekil 4.28'de olduğu gibi birer adet var olma durumundaki kadar olacağı ve bazı durumlarda daha da arttıracağı açıkça görülmektedir.

Yol sayısındaki bu artışların sebebi karışmış taşlar probleminin "NP-complete" özelliğinden kaynaklanmaktadır. "NP-complete" bir problemin bir çözümü olmasına karşılık bu çözüm polinom zamanda bulunamamaktadır.

Bu problemin NP-complete oluşunun ispatı şu şekildedir:

Literatürde geçen "Travelling Salesman" probleminde bir satıcının n adet (n > 0) şehri her şehri bir kere ziyaret etmek şartıyla, en kısa yolu kat ederek dolaşması için şehirleri hangi sırada ziyaret etmesinin gerektiği bulunmaya çalışılmaktadır. Bu problemin "NP-complete" olduğu ispatlanmıştır. Karışmış taşlar probleminin "Travelling Salesman" problemi ile benzer yönleri vardır. Örneğin karışmış taşlar probleminde robot kolunun bir taşın başlangıç pozisyonuna gitmesi, taşı başlangıç pozisyonundan alıp hedef pozisyonuna koyması ve ardından

başka bir taşın başlangıç pozisyonuna gitmesi durumu söz konusudur. Bu sıra içerisinde robot kolunun bir taşı başlangıç yerinden alıp hedef yerine koyma sırasında kat ettiği mesafe göz ardı edilecek olursa problem "Travelling Salesman" problemine indirgenmiş olacaktır. Dolayısıyla karışmış taşlar problemi bir robot kolunun bir taşın başlangıç pozisyonundan başka bir taşın başlangıç pozisyonuna hareketi olarak düşünülebilir. Bu hareket sırası "Travelling Salesman" probleminde satıcının bir şehirden başka bir şehre ziyareti sırasına denk gelmektedir. Bu yüzden karışmış taşlar problemi de NP-complete bir problem olmaktadır. Mevcut harekete, bir taşın başlangıç pozisyonundan alınıp hedef pozisyonuna konması hareketi de eklenecek olursa problemin karmaşıklığı artacak ve NP-complete olma özelliği korunacaktır.

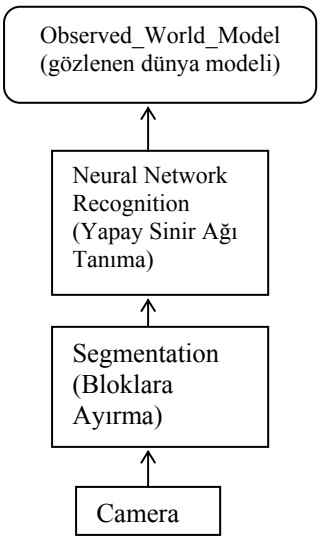
En kısa yol çözümüne ait program bilgileri Ek 4'de verilmektedir.

5. GÖRME SİSTEMİ

5.1 Görme Sistemi Olarak Kullanılan Metodlar

Bu çalışmada kullanılan görme sistemi, iş seviyesinde beklenmedik olayların yakalanabilmesi için ihtiyaç duyulan geri beslemenin sağlanabilmesi amacı ile kullanılmaktadır. Görme sistemi çalışma uzayına ait iki boyutlu görüntülerin alınmasına yardımcı olmasına rağmen bu görüntüler iş seviyesi tarafından beklenmedik olayların yakalanabilmesi için doğrudan kullanılamamaktadırlar. Bir geri besleme modülünün alınan bir görüntüyü iş seviyesinin anlayacağı şekle dönüştürmesi gerekmektedir. Bu amaçla iki boyutlu görüntülerin her birinden iş seviyesinin doğrudan kullanabildiği bir dünya modeli (gözlenen dünya modeli) oluşturulur ve iş seviyesi oluşturulan dünya modelini beklenen dünya modeli (*Expected_World_Mode*l) ile kıyaslayarak beklenmedik bir olay olup olmadığını tespit eder.

Kamera ile alınan görüntüden bir gözlenen dünya modeli oluşturulabilmesi için robot çalışma uzayını temsil eden görüntünün çalışma uzayında yer alan hücrelere ayrılması (segmentation) ve ardından da her bir hücrenin bir yapar sinir ağı tanıma algoritmasından geçirilerek her hücre içerisinde eğer var ise yer alan nesnenin belirlenmesi gerekir. Bu çalışmada kullanılan dünya modelleri dikkate alındığında her hücrenin içerebileceği nesne bir blok olabileceğinden her hücrenin içerisinde bir blok olup olmadığının anlaşılması ve eğer var ise bloğun tipinin belirlenmesi gerekmektedir. Çalışma uzayı içerisindeki her hücrenin içeriğinin belirlenmesi sonucunda çalışma uzayına ait bir gözlenen dünya modeli (*Observed_World_Model*) oluşturulur (Şekil 5.1).



Şekil 5.1 Görme Sistemi Blok Diyagramı

Çalışma kapsamında çalışma uzayından uzaklığı 56.5 cm olan sabit bir kamera ve boyutları 32 × 32 cm olan bir nesne (satranç tahtasının) kullanılmıştır.

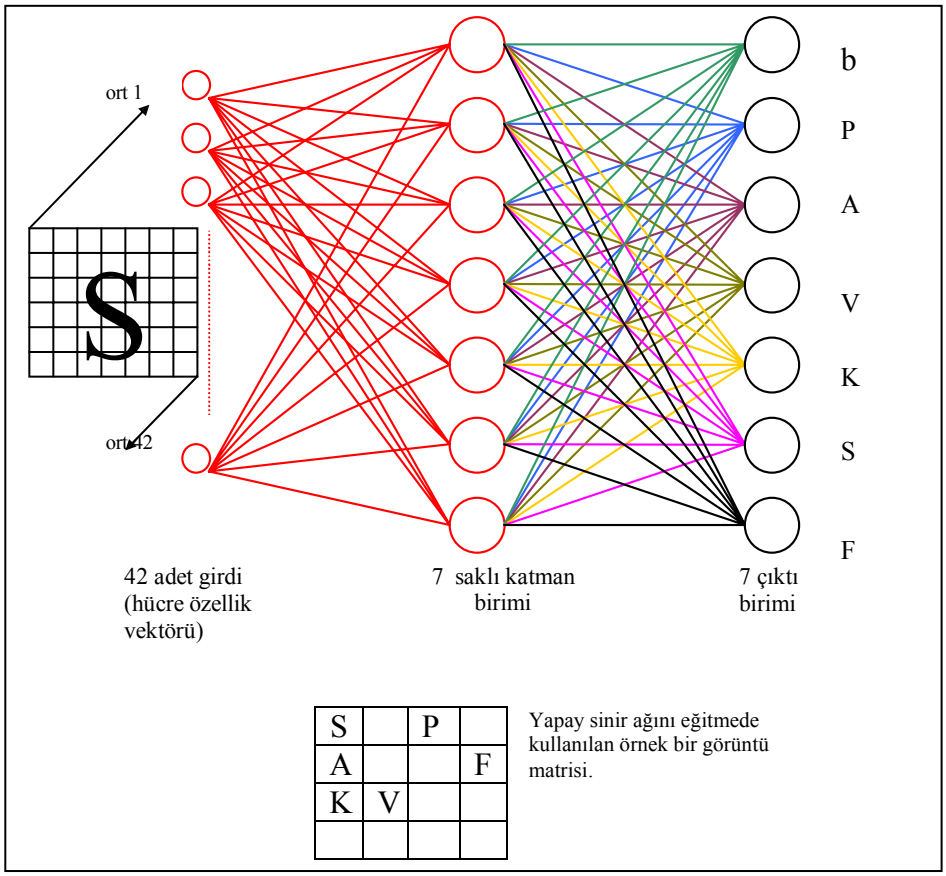
Kamera, lens seçimi ve sistemin nesneleri ile birlikte oluşturulmasından sonra kamera kalibrasyonunun doğru yapılması gerekmektedir (Bkz. Ek 5). Kamera ile alınan görüntüler iki boyutlu uzaya ait olmasına rağmen robotun çalışma uzayı üç boyutludur. İki boyutlu uzaydan elde edilen bilgilerin üç boyutlu uzaya eşleştirilmesi gerekmektedir. Gözlenen dünya modeli çalışma uzayındaki nesnelerin görüntü üzerinde hangi hücrelerde bulunduğunu belirtmesine rağmen bu nesnelerin bir referans noktasına göre gerçek uzaklıklarını vermemektedir. Yazılım olarak her hücrenin orta noktasının koordinatları robot üzerinde belirlenen bir referans noktasına göre belirlenmektedir ve bir tabloda tutulmaktadır. Dolayısıyla robot kolu herhangi bir hücreye

gitmek durumunda iken bu dönüştürme tablosundan o hücreye üç boyutlu dünyada hangi (x,y,z) ekseni hareketleri ile gidilebileceği alınır ve kinematik çözümler ile bu eksen hareketleri robot eklem açılarına dönüştürülür. Bu aşamadan sonra açıların kodlayıcı adımlarına dönüştürülmesi gerekmektedir çünkü robot kontrol ünitesinin herhangi iki eklem arasındaki açıyı istenen açı yapabilmesi için o açı değerini kodlayıcı adım sayısı cinsinden alması gerekmektedir.

Mevcut görüntü yakalama kartı ile bitmap ve jpeg formatında görüntüler kaydedilebilmektedir. Bu çalışmada görüntüler 640 × 480 çözünürlüğünde 256 gri seviyesinde bitmap olarak kaydedilmiştir (Bkz. Ek 6). Bitmap görüntüler robot kolunun çalışma uzayını simgelediğinden nesne ayırma (segmentation) ve nesne tanıma (recognition) işlemlerine tabi tutulmaktadırlar. Elde edilen görüntü ilk olarak bir nesne ayırma işlemi için bir eşik değeri kullanarak siyah-beyaz görüntü haline getirilir. Nesne ayırma bir görüntü içerisindeki her hücrenin tek tek ana görüntüden ayrılması ve o hücre üzerinde uygulanacak işlemler tamamlanıncaya kadar geçici olarak bir matris içerisinde tutulması anlamına gelmektedir. Bir hücre ile ilgili işlemler tamamlandıktan sonra görüntü içerisindeki bir sonraki hücre ana görüntüden ayrılır ve üzerinde işlemler yapılır. Ayırma işlemi görüntü içindeki her hücre için yapılır. Ayırma işlemine görüntüdeki ilk hücre ile başlanır ve soldan sağa ilerleyerek sırasıyla her satır için gerçekleştirilir. Öte yandan her hücredeki blok tipinin belirlenmesi için gerekli olan tanıma işlemi için ise yapay sinir ağı tanıma yöntemi kullanılmıştır. Yapay sinir ağı ile tanıma yapabilmek için öncelikle yapay sinir ağını eğitmek gerekmektedir. Bu çalışmada kullanılan yapay sinir ağı mimarisi, yapay sinir ağını eğitme ve yapay sinir ağı ile tanıma konuları sonraki alt bölümlerde ele alınmaktadır.

5.2 Kullanılan Yapar Sinir Ağı Mimarisi

Bu çalışmada bir "multi layer perceptron" yapay sinir ağı kullanılmıştır. Yapay sinir ağının eğitilmesi "backpropogation" yöntemi ile yapılmaktadır. Kullanılan mimari Şekil 5.2'de verilmektedir.



Şekil 5.2 Kullanılan Yapay Sinir Ağı Mimarisi ve Soldaki Görüntü ile Beslenmesi

Şekil 5.2'de verilen mimari 42 elamanlı bir girdi katmanı, 7 elemanlı bir saklı katmanı ve 7 elemanlı bir çıktı katmanından oluşan bir

yapay sinir ağı mimarisidir. Yapay sinir ağının girdisini oluşturmak için alınan bir görüntüdeki bir hücre kullanılır. Örneğin şekilde verilen görüntüye ait 16 adet hücre vardır. Bir gözlenen dünya modeli oluşturulabilmesi için bu hücrelerin her birinin yapay sinir ağından tek tek geçirilmesi gerekmektedir.

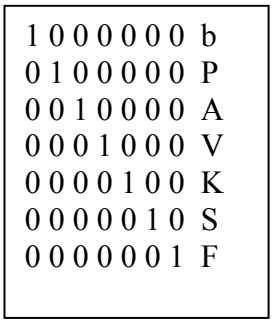
Bir hücrenin yapay sinir ağından geçirilmesi için bu hücreye ait 6 × 7 boyutlarında bir matris oluşturulması gerekmektedir. Şekil 5.2'de verilen örnek görüntüye ait ilk hücre için bu matrisin nasıl oluşturulduğu gösterilmektedir. İlk hücre enine 6 satıra ve boyuna 7 sütuna ayrılacak şekilde parçalara ayrılır. Soldan sağa olmak üzere sırasıyla her parçanın ortalaması alınır. İlk parçanın ortalaması *ort1* ve son parçanın ortalaması da *ort42* olarak ifade edilmiştir. Bu ortalamalar soldan sağa olmak üzere özellik matrisine yerleştirilir. Bir satır dolunca bir sonraki satıra geçilir. Bu ortalamalar yapay sinir ağına şekilde gösterildiği şekilde girdi olarak verilmektedirler. Bir hücreye ait özellik matrisi oluşturulup yapay sinir ağına verildikten sonra bir sonraki hücre için aynı işlemler tekrarlanır.

Hücre özellik bilgileri hem yapay sinir ağı eğitmek hem de yapay sinir ağı ile tanımak için kullanılırlar.

5.2.1 Yapay sinir ağı eğitme

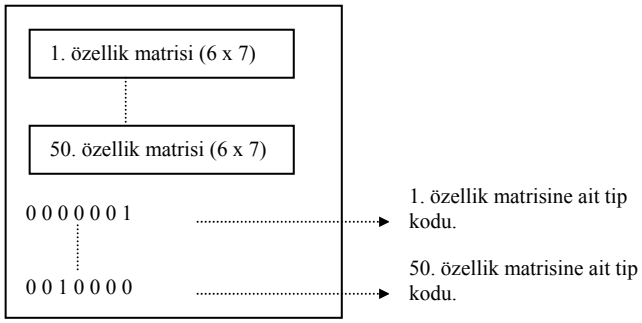
Yapay sinir ağı eğitmek için adapte olabilen bir ağ yapısı kullanılmıştır. Adapte olabilen bir ağ yapısında bir eğitici yapay sinir ağına her bir nesneyi tanıtır. Her bir nesnenin tanıtılması o nesneye ait özelliklerin ve nesne tipinin yapay sinir ağına verilmesi şeklinde gerçekleştirilir. Örneğin bu çalışmada bir çalışma uzayına ait her bir hücreye ait 42 adet özellik matrisi oluşturulur ve yapay sinir ağına

eğitmek üzere girdi olarak verilir. Aynı zamanda o hücrenin tipi (S, V, K, F, A, P) yapay sinir ağına belirtilir. Bu çalışmada yapay sinir ağı Şekil 5.1'de verilen çıktı tiplerinden hangisini tanımışsa ona ait çıktı değerini 1 diğerlerini ise 0 yapay. Dolayısıyla her tip için Şekil 5.3'deki kodlar kullanılmaktadır.



Şekil 5.3 Hücre Tip Kodları

Yapay sinir ağını eğitirken de bir hücre için hem bir özellik matrisi hem de bir kod belirtilir. Bu çalışmada yapay sinir ağı 50 adet görüntü ile eğitilmektedir. Dolayısıyla her tipin hangi özellik matrisine ve çıktı vektörüne karşılık geldiği bir eğitim kütüğünde tutulmaktadır. İlk hücresinin A ve son hücresinin F olduğu bir eğitim kütüğünün içeriği Şekil 5.4'de verildiği gibidir. 50 görüntü ile eğitim yapıldığından dolayı n x n çalışma uzayının boyutları olmak üzere bir eğitim kütüğünde 50 × n × n adet özellik matrisi ve bunlara karşılık gelen 50 × n × n adet tip kodu bulunmalıdır .



Şekil 5.4 Bir Eğitim Kütüğünün İçeriği

Yapay sinir ağını "supervised learning" metodu ile eğitmenin amacı yapay sinir ağının her çıktı birimi için doğru çıktılar üretmesinin sağlanmasıdır. Her çıktı birimi j'nin Oj çıktısı aşağıdaki fonksiyon ile tanımlanmış olsun.

$$O_j=f(net_j),\ net_j= \Sigma_i\, w_{ji}o_i+\theta_j \quad \text{(i bir önceki katmandır).} \qquad (2)$$

Bir çıktı katmanı aktivite seviyesini iki aşama ile hesaplar. Bu aşamalardan ilki formül (2) kullanarak toplam girdi ağırlığının hesaplanmasıdır. w_{ji}, i ve j birimleri arasındaki bağlantının ağırlığıdır. θ_j ise bir çıktı üretilirken karşılaştırma yapılan bir eşik değeridir. o_i bir önceki katmandan gelen çıktı değeridir. Formüldeki *f* fonksiyonu için ise sigmoid fonksiyonu kullanılır: $O_j = 1/(1+e^{-x})$. Diğer aşama ise her çıktı birimi için değerler hesaplandıktan sonra,

$$E_p= \tfrac{1}{2} \left[\Sigma_i \left(t_{pj} - o_{pj}\right)^2\right] \qquad (3)$$

ifadesi ile tanımlanan E_p hata miktarının hesaplanmasıdır. $_P$, tanıması yapılan bir örnektir (bu çalışmadaki her bir hücre). Verilen

ifadede o_{pj} çıktı katmanındaki j'inci birimin değerini ve t_{pj} , j'inci birimin istenen değerini belirtmektedir.

Toplam hata E'nin hesaplanması ise her birim için hesaplanan hataların toplanması ile elde edilmektedir, $E = \Sigma_p E_p$. Amaçlanan ise doğru w_{ji} ve θ_j değerlerinin seçilmesi ile hatanın yeterince küçültülmesidir.

5.2.2 Yapay sinir ağı ile tanıma

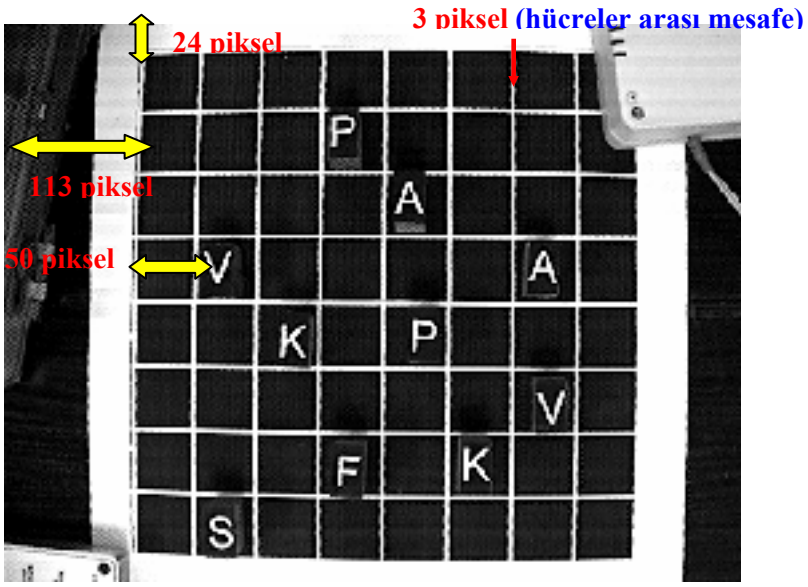
Yapay sinir ağı tanımada izlenen yol ise görüntü içerisindeki her hücrenin yapay sinir ağından geçirilerek her çıktı birimi için üretilen çıktıların alınması ve yapay sinir ağının o hücre için ürettiği çıktıların hangi tipe karşılık geldiğinin bulunmasıdır (Bkz. Ek 7).

Yapay sinir ağında bir eğitim kütüğü kullanılmamasına rağmen her hücre için bir özellik matrisi oluşturulması ve her tanıma için bu matrisin bir tip kodu olmadan yapay sinir ağına verilmesidir. Yapay sinir ağı daha önce eğitim esnasında hesapladığı uygun w_{ji} ve θ_j değerlerini kullanarak bu tip kodlarını çıktı birimleri şeklinde oluşturacaktır. Örneğin verilen bir özellik matrisi için yapay sinir ağı "0 0 1 0 0 0 0 " çıktılarını yani tip kodunu oluşturmuşsa, bu sonuç yapay sinir ağının verilen hücreyi "A" tipi olarak tanıdığını ifade eder.

Bu çalışmada yapılan eğitimler sonucunda yapay sinir ağının tanıma yüzdesi % 70'dir. Farklı ışık şartlarında yapay sinir ağının tekrar eğitilmesi ihtiyacı ortaya çıkmaktadır.

5.3 Nesne Ayırma (Object Segmentation)

Şekil 5.5'de örnek bir kamera görüntüsü verilmektedir. Bu görüntü alınıp kaydedildikten sonra yapılacak işlemler görüntüden hücreleri dolayısıyla görüntü içerisindeki blokları ayırma ve bu hücreleri yanı blokları tanıma işleminden geçirmektir. Tanıma işlemi sırasıyla çalışma uzayındaki her hücre için yapılmaktadır. Hücre ayırma ve ardından da tanıma işlemlerinin sonucunda bilgisayar bir görüntüde yer alan her hücredeki bloğun hangi harf ya da işaret ile etiketlenmiş olduğunu öğrenecek ve bu bilgiyi kullanarak bir dünya modeli (*Observed_World_Model*) oluşturacaktır. *Observed_World_Model* iki boyutlu bir matris veri yapısıdır. Bu matrisin her bir elemanı daha önce tanımı verilen ve "object" olarak adlandırılan veri yapısı şeklinde tanımlanmıştır. Karışmış taşlar uzayı iki boyutlu olmasına rağmen konteyner uzayı üç boyutludur. Konteyner uzayı için gözlenen dünya modeli de iki boyutludur. Bunun sebebi sabit bir kameranın kuş bakışı olarak iki boyutlu bir uzay görüntüsü almasıdır.



Şekil 5.5 Karışmış Taşlar Çalışma Uzayına Ait Bir Kamera Görüntüsü.

Çalışma uzayında yer alan nesneler Şekil 5.5'de gösterildiği gibi harflerle etiketlenebileceği gibi basit şekiller ile de etiketlenebilir. Örneğin Şekil 5.6'daki görüntü her biri satranç tahtasının bir taşını temsil eden yıldız, kare, üçgen, içi boş yuvarlak, içi dolu yuvarlak ve artı şekilleri ile etiketlenmişlerdir. Eğer bu şekilde bir etiketleme söz konusu ise daha sonra bu şekillerin temsil ettikleri harflere dönüştürülmeleri gerekmektedir (Şekil 5.8).

Kaydedilen bir görüntü içerisindeki hücrelerin ayrılması işlemi görüntü içerisindeki her hücrenin ilk pikseline ulaşmayı ve o hücreyi görüntünün geri kalan kısmından ayırıp $n \times n$ boyutlarında ayırma (segment) matrisine yerleştirmesini gerektirir. Daha sonra bu matris daha önce açıklandığı gibi 6 satır ve 7 sütuna ayrılır. Böylece bir hücre 42 adet

parçaya ayrılmış olur. Her parçanın içerdiği piksellerin renk değerleri ortalaması alınır. Böylece sonuçta bir hücre için 42 adet ortalama yani özellik (feature) elde edilir ve bu özellikler bir özellik matrisinde saklanır. Bu matris, tanıma işleminde yapay sinir ağının girdi birimleri olarak kullanılacaktır.

Şekil 5.5'deki görüntü incelenecek olursa görüntünün sadece hücrelerden oluşmadığı ve çalışma uzayı dışında kalan kısımların da görüntü içerisinde yer aldığı görülecektir. Bu kısımlar Şekil 5.5'de gösterildiği gibi görüntüsü alınan satranç tahtasının (çalışma uzayının) solunda kalan (113 piksellik) ve üstünde kalan (24 piksellik) kısımlardır. Bir görüntü içerisindeki bir hücrenin ilk pikseline ulaşmak için bu kısımların da hesaba katılması gerekir. Dolayısıyla ya bu kısımları görüntüden ayırmak ya da her hücrenin ilk pikseline ulaşırken bu mesafeleri de dikkate almak gerekir. Bu çalışmada ikinci yöntem uygulanmıştır. Fazlalık kısımların hesaplara dahil edilmesi *top_displacement* (üst fazlalık) ve *left_displacement* (sol fazlalık) değişkenleri ile sağlanır. Bir hücrenin ilk pikselinin, görüntünün ilk pikseli referans alınarak, x (*start_Row_Pixel*) ve y (*start_Col_Pixel*) koordinatlarının hesaplanması için aşağıdaki formüller kullanılır.

```
start_Row_Pixel = top_displacement +
                  (Number_of_Cells * SegmentSize) +
                  total _distance_between_segments);
```

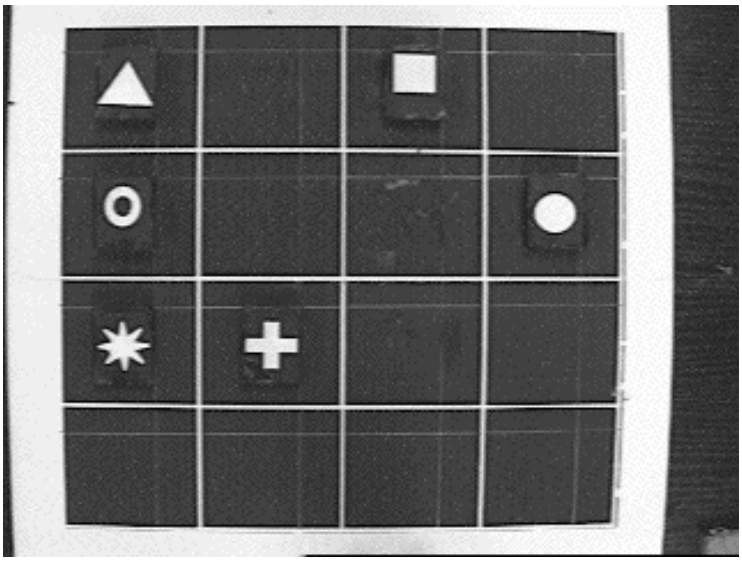
Number_of_Cells, ayrılacak hücreye ulaşıncaya kadar görüntünün en üstünden hücreye kadar olan hücre sayısını vermektedir. *SegmentSize* hücrenin kaç pikselden oluştuğunu vermektedir. *total _distance_between_segments* her iki ardaşık hücre arasındaki beyaz piksel sayısıdır.

Örneğin Şekil 5.5'deki V içerikli hücrenin ilk pikseline ulaşmak için resmin sol üst köşesindeki ilk pikselden y yönünde 24 + 3 × 50 + 3 × 3 = 183 piksel ilerlemek gerekmektedir eğer her hücrenin 50 × 50 boyutlarında olduğu ve hücreler arası beyaz piksel sayısının 3 olduğu varsayılırsa.

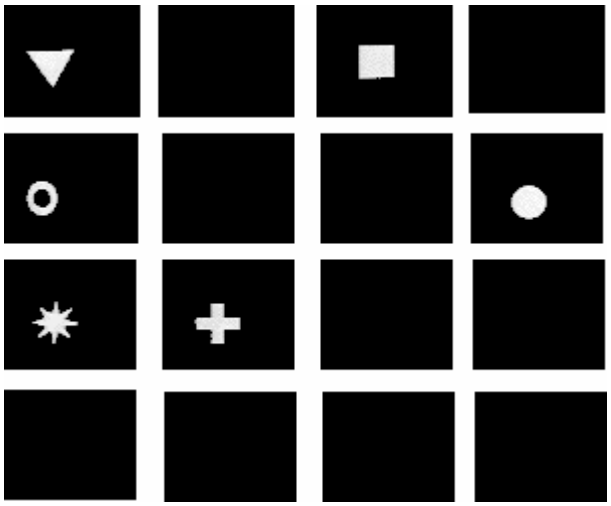
x değeri hesabi için aşağıdaki formül kullanılır:

start_Col_Pixel = left displacement +
*(Number_of_Cells * SegmentSize) +*
total _distance_between_segments);

Örneğin Şekil 5.5'deki V içerikli hücrenin ilk pikseline ulaşmak için resmin sol üst köşesindeki ilk pikselden x yönünde 113 + 1 × 50 + 1 × 3 = 166 piksel ilerlemek gerekmektedir.



Şekil 5.6 Kamera ile Alınmış 4 × 4 Boyutlarındaki Bir Tahtanın Görüntüsü



Şekil 5.7 Önceki Resime ait Görüntünün Nesnelerine Ayrılmış Hali

Şekil 5.6'daki gösterilen nesnelerin her birisi satranç taşına ait bir harfi temsil edecek şekilde program tarafından kodlanmıştır. İçi boş yuvarlak S, kare P, yuvarlak F, yıldız K ve artı V harflerini temsil etmektedirler. Görüntünün nesnelerine ayrıldıktan sonraki görünümü Şekil 5.7'de verilmiştir. Dikkat edilecek olursa görüntü nesnelerine ayrılmadan önce siyah beyaz görüntü haline dönüştürülmüştür.

S		P	
A			F
K	V		

Şekil 5.8 Gözlenen Dünya Modeli (*Observed_World_Model*)

6. SONUÇ VE TARTIŞMA

Bu çalışma kapsamında, bir robot kolu için iş planlayan ve beklenmedik olaylar durumunda iş planlarını değiştirerek yeniden planlama yapan bir mimari geliştirilmiş ve bu mimarinin uygulaması gerçekleştirilmiştir. Bu mimarinin mevcut mimarilerden en belirgin farkı yeniden planlamayı görme sisteminden gelen bir geri besleme ile yapmasıdır. Robot mimarilerinde beklenmedik olayların sembolik planlama seviyesinde ele alınması kullanılan bir yöntemdir. Ancak alt seviyelerin beklenmedik olayları ele almada yetersiz kalması nedeni ile araştırmacılar, mevcut mimarilerini yeniden planlamayı dahil edecek şekilde geliştirmektedirler.

Yeniden planlama seviyesine bir geri beslemenin verilmesi yeniden planlama sürecinin gerçek dünya hakkında bilgi edinilerek yapılmasını sağlamaktadır. Geri besleme daha sağlam planlar üretilmesine ve bu yüzden de robotların sağlam ve güvenilir olmasına sağlamaktadır. Yeniden planlama seviyesinin bir görme sisteminden gelen geri besleme ile desteklenmesi kavramı bu çalışma ile ortaya atılmış ve daha sonra başta Carnegie Mellon Üniversite'sinden (CMU) "Prodigy" araştırma grubu olmak üzere diğer gruplardan ilgi görmüştür. Bu çalışma ve CMU üniversitesi bu kavramın uygulamasını farklı çalışma uzayları için gerçekleştirmiştir.

Bu çalışmanın ortaya attığı başka bir kavram da yeniden planlamanın bir karar mekanizması içermesi ve bu mekanizmanın alternatif robot hareket dizinleri arasından en kısa yol kat edecek olanını seçecek şekilde oluşturulmasıdır. Literatürdeki çalışmalar yeniden planlama sırasında robotun bir hareket dizinini işleterek kat edeceği

toplam yolu hesaba katmamakta sadece son işletilen hareketin istenen durumu gerçekleştirmemesi durumunda başka bir hareketi işletime koymaktadırlar.

Çalışmanın diğer bir özelliği de yeniden planlamaya ait karar mekanizmasının bir öğrenme metodu içermesidir. Bu metod geçmişte gerçekleştirilmiş işletimlere ait bilgileri saklayarak bu bilgileri daha sonraki kararlar vermek için kullanmaktadır. Bu sebeple sistem başlangıçta yeterince iyi olmayan kararlar verse de zaman içerisinde kararlarını iyileştirmektedir.

Bu çalışma için geleceğe ait öneriler şu şekilde sıralanabilir:

Sistem dış kaynaklı farklı türde beklenmedik olaylar için denenebilir.

Zamanın kritik olduğu durumlarda çalışma uzayını tamamen değerlendirip bir gözlenen dünya modeli oluşturmak yerine dünya modelinin gereken kadarından kısmi bir dünya modeli oluşturulması sağlanabilir.

Sistem bir robot kolu uygulaması olmasına rağmen hareket yeteneğine sahip mobil robot çalışmaları için de uyarlanabilir.

Çift kamera ile üç boyutlu görüntülerden gözlenen dünya modeli oluşturulabilir. Bu şekilde çalışma uzayı içerisindeki nesnelerin birbirini örtmesi (occlusion) durumlarında da gözlenen dünya modeli oluşturulmak mümkün olacaktır.

Öğrenme ile ilgili bölümde bahsedilen “evrimsel algoritmaların” üretilen planların iyileştirilmesinde bir fayda sağlayıp sağlamayacağı test edilebilir.

KAYNAKLAR DİZİNİ

Agre, P.E. and Chapman, D., 1990, What are plans for?, Robotics and Autonomous Systems, 6, 17-34pp.

Ambros-Ingerson, J.A., and Steel, S., 1988, Integrating planning, execution and monitoring, In Proceedings of the National Conference on Artificial Intelligence (AAAI-88), St. Paul, MN. (Menlo Park, CA: AAAI Press), 83-88pp.

Arkin, R. C., 1990, Integrating behavioral, perceptual, and world knowledge in reactive navigation, Robotics and Autonomous Systems, 6, 105-122pp.

Arkin, R.C., 1999, Behaviour-Based Robotics, The MIT Press, Cambridge, Massachusetts,

Atkins, E.M., Durfee, E.H. and Shin, K.G., 1996, Detecting and reacting to unplanned-for world states, In Papers from the 1996 AAAI Fall Symposium Plan Execution: Problems and Issues, Boston, MA. (Menlo Park, CA: AAAI Press), 1-7pp.

Bigus, J.P. and Bigus, J., 1997, **Constructing Intelligent Agents With Java : A Programmer's Guide to Smarter Applications**, John Wiley and Sons.

Blythe, J., 1994, Planning with external events, In Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence, Seattle, WA. (San Mateo, CA: Morgan Kaufmann), 94-101pp.

Bonasso, R.P. and Kortenkamp, D., 1996, Using a layered control architecture to alleviate planning with incomplete information, In Proceedings of the AAAI Spring Symposium Planning with Incomplete Information for Robot Problems, Stanford, CA. (Menlo Park, CA: AAAI Press), 1-4pp.

KAYNAKLAR DİZİNİ (devam)

Brooks, R.A., 1986, A robust layered control system for a mobile robot, In the Proceedings of the IEEE International Conference on Robotics and Automation, Los Alamitos, Calif.: IEEE Computer Society Pres, 824-829pp.

Brooks, R., 1989, A robot that walks: emergent behaviors from a carefully evolved network, Proceedings of the IEEE International Conference on Robotics and Automation, 692-94pp.

Carbonell, J.G. and the Prodigy Research Group, 1992, Prodigy4.0: the manual and tutorial, Technical Report CMU-CS-92-150, School of Computer Science, Carnegie Mellon University.

Dean, T.L., and Boddy, M., 1988, An analysis of time-dependent planning, In Proceedings of the National Conference on Artificial Intelligence (AAAI-88), St. Paul, MN. (Menlo Park, CA: AAAI Press), 49-54pp.

Dean, T., Basye, K., Chekaluk, R., Hyun, S., Lejter, M. and Randazza, M., 1990, Coping with uncertainty in a control system for navigation and exploration, In Proceedings of the Eigth National Conference on Artificial Intelligence (AAAI-90), Boston, MA. (Cambridge, MA: MIT Press), 1010-1015pp.

Draper, D., Hanks, S. and Weld, D., 1994, A probabilistic model of action for least_commitment planning with information gathering, In Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence, Seattle, WA. Morgan Kaufmann, 178-186pp.

Drummond, M., Swanson, K., Bresina, J., and Levinson, R., 1993, Reaction-First search, In Proceedings of the Thirteenth International Joint BIBLIOGRAPHY 167 Conference on Artificial Intelligence (IJCAI-93), Chambery, France. (San Mateo, CA: Morgan Kaufmann), 1408-1414pp.

KAYNAKLAR DİZİNİ (devam)

Etzioni, O., Hanks, S., Weld, D., Draper, D., Lesh, N. and Williamson, M., 1992, An approach to planning with incomplete information, In Proceedings of the Third International Conference on Knowledge Representation and Reasoning, Boston, MA, Morgan Kaufmann, 115-125pp.

Falkenhainer, B. and Forbus, K., 1998, Setting up large scale qualitative models, In Proc. 7th National Conf. AI, 301-306pp.

Fikes, R.E. and Nilsson, N.J., 1971, STRIPS: a new approach to the application of theorem proving to problem solving, *Artificial Intelligence*, 2, 3-4, 189-208pp.

Fikes, R.E., Hart, P.E. and Nilsson, N.J., 1972, Learning and executing generalized robot plans, *Artificial Intelligence*, 3(4), 231-249pp.

Firby, R.J., 1989, Adaptive Execution in Complex Dynamic Worlds, PhD thesis, Yale University, New Haven, CT.

Firby, R.J., 1994, Task networks for controlling continuous processes, In Hammond, K. (Eds.), Artificial Intelligence Planning Systems: Proceedings of the Second International Conference (AIPS-94), Chicago, IL. (Menlo Park, CA: AAAI Press), 49-54pp.

Gat. E., 1997, ESL: a language for supporting robust plan execution in embedded autonomous agents, In Proceedings of the IEEE Aerospace Conference, Los Alamitos, Calif. IEEE Computer Society Press.

Georgeff, M.P. and Lanskey, A., 1987, Reactive reasoning and planning, In Proceedings of the Sixth National Conference of Artificial Intelligence, Menlo Park, California, AAAI Pres, 677p.

KAYNAKLAR DİZİNİ (devam)

Georgeff, M. and Ingrand, F.F., 1989, Decision-making in an embedded reasoning system, In Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89), Detroit, MI (San Mateo, CA: Morgan Kaufmann), 972-978pp.

Gervasio, M.T. and DeJong, G.F., 1991, Learning Probably Completable Plans, Technical Report UIUCDCS-R-91-1686, University of Illinois at Urbana-Champaign, IL, Urbana, IL.

Grefenstette, J. and Schultz A., 1994, An evolutionary approach to learning in robots", Machine Learning Workshop on Robot Learning, New Brunswick, NJ, July 13, (NCARAI Report: AIC-94-014).

Haigh, K.Z., 1998, Situation-Dependent Learning for Interleaved Planning and Robot Execution, Ph.D thesis, CMU.

Kaelbling, L.P., 1987, REX: a symbolic language for the design and parallel implementation of embedded systems.", the AIAA Conference on Computers in Aerospace, Wakefield, Mass.

Kaelbling, L.P., 1988, Goals as parallel program specifications, In Proceedings of the Sixth National Conference on Artificial Intelligence, Menlo Park, California, AAAI Press. I.

Kaelbling, L.P., Litmann, M.L., and Moore, A.W., 1996, Reinforcement Learning: A survey, *Journal of Artificial Intelligence Research*, 4, 237-285pp.

Kautz, H. and Selman, B., 1998, Blackbox: A new approach to the application of theorem proving to problem solving, In AIPS98 Workshop on Planning as Combinatorial Search, 58-60pp.

KAYNAKLAR DİZİNİ (devam)

Koening, S. and Simmons, R.G., 1998, Xavier: a robot navigation architecture based on partially observable markov decision process models, *Artificial Intelligence and Mobile Robots*, D. Kortenkamp, R.P. and Bonasso, R.M. (Eds.), MIT Press.

Kushmerick, N., Hanks, S., and Weld, D, 1993, An algorithm for probabilistic planning, Technical Report 93-06-03, Department of Computer Science and Engineering, University of Washington, Seattle, WA.

Laird, E.L., Congdon, C. B. and Coulter, K.J., 1998, The Soar User's Manual Version 8.2, University of Michigan.

Luger, G.F. and Stubblefield, W.A., 1993, *Artificial Intelligence, Structures and Strategies for Complex Problem Solving*, The Benjamin/Cummings Inc.

Lyons, D.M. and Hendriks, A.J., 1992, A practical approach to integrating reaction and deliberation, In Hendler, J. (Eds.), Artificial Intelligence Planning Systems, Proceedings of the First International Conference (AIPS-92), (San Mateo, CA: Morgan Kaufmann), 153-162pp.

Mansell, T.M., 1993, A method for planning given uncertain and incomplete information, In Proceedings of the Ninth Conference on Uncertainty in Artificial Intelligence, Washington, DC. (San Mateo, CA: Morgan Kaufmann), 250-358pp.

Mataric, M., 1993, Synthesizing group behaviours, Proceedings of the Workshop on Dynamically Interacting Robots, International Joint Conference on Artificial Intelligence (IJCAI-93), Chambery, France, August, 1-10pp.

KAYNAKLAR DİZİNİ (devam)

McDermott, D., 1992, Transformational Planning of Reactive Behavior, Technical Report YALE/CSD/RR#941, Computer Science Department, Yale University, New Haven, CT.

Mitchell, T.M., 1997, MachineLearning, McGRAW-HILL.

Nilsson, N.J., 1980, Principles of Artificial Intelligence, Palo Alto: Tioga Press.

Nilsson, N.J., 1984, Shakey the Robot, Technical Report 323, AI Center, SRI International, Menlo Park, CA.

Nourbakhsh, I., 1997, Interleaving Planning and Execution for Autonomous Robots, PhD thesis, (Dordrecht, Netherlands: Kluwer Academic).

Payton, D., Rosenblatt, J.K. and Keirsey, D., 1990, Plan-Guided reaction, IEEE Transactions on Systems, Man and Cybernetics, 20 (6), 1370-1382pp.

Payton, D., 1991, Internalized Plans: A representation for Action Resources, in *Designing Autonomous Agents*, Maes, (P. Eds.), MIT Press, Cambridge, M.A., 89-103pp.

Pell, B., Bernard, D.E., Chien, S.A., Gat, E., Muscettola, N., Nayak, P.P., Wagner, M.D. and Williams, B.C., 1997, An autonomous spacecraft agent prototype, In proceedings of the First International Conference on Autonomous Agents, Marina del Rey, CA, 253-261pp.

Peot, M.A., Smith, D.E., 1992, Conditional nonlinear planning, In Proceedings of the First International Conference on Artificial Intelligence Planning Systems, College Park, Maryland, Morgan Kaufmann, 189-197pp.

KAYNAKLAR DİZİNİ (devam)

Pryor, L. and Collins, G., 1993, "Cassandra: Planning with contingencies", Technical report 41, Institute for the Learning Sciences, Northwestern University.

Pryor, L.M., 1994, Opportunities and Planning in an Unpredictable World, PhD thesis, Northwestern University, Evanston, Illinois, Available as Technical Report number 53.

Pryor, L. and Collins, G., 1996, Planning for Contingencies: A Decision_based Approach, *Journal of Artificial Intelligence Research*, 4, 287-339pp.

Schoppers, M.J., 1989, Representation and Automatic Synthesis of Reaction Plans, PhD thesis, Department of Computer Science, University of Illinois, Urbana-Champaign, IL, Available as Technical Report UIUCDCS-R-89-1546.

Schultz, A.C., Grefenstette, J.J. and Adams, W., 1996, Robo-Shepherd: learning complex robotic behaviours, Naval Research Laboratory, Wahington, DC, USA.

Simmons, R.G., 1990, An architecture for coordinating planning, sensing and action, DARPA Workshop on Innovative Approaches to Planning, Scheduling, and Control, San Diego, California.

Simmons, R. and Keoning, S., 1995, Probabilistic Robot Navigation in Partially Observable Environments, Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-95), Montreal, CS, August, 1080-87pp.

KAYNAKLAR DİZİNİ (devam)

Simmons, R., Goodwin, R., Haigh, K. Z., Koenig, S., O'Sullivan, J., 1997, "A layered architecture for office delivery robots.", In Johnson, W. L. (Eds.), Proceedings of the First International Conference on Autonomous Agents, Marina del Rey, CA. (New York, NY: ACM Press), 245-252pp.

Soldo, M., 1990, Reactive and preplanned control in a mobile robot, In proceedings of the IEEE International Conference on Robotics and Automation, Los Alamitos, Calif., IEEE Computer Society Press.

Sutton, R.S. and Barto, A., 1998, Reinforcement Learning: An Introduction, MIT Press, Cambridge , M.A.

Tan, M., 1993, Cost-sensitive learning of classification knowledge and its applications in robotics, Machine Learning, 13(1), 1-33pp.

Veloso, M.M., Carbonell, J., P'erez, M.A., Borrajo, D., Fink, E. and Blythe, J., 1995, Integrating planning and learning: The Prodigy architecture, *Journal of Experimental and Theoretical Artificial Intelligence*, 7(1), 81-120pp.

Weld, D. S., 1999, Recent Advances in AI planning, AI Magazine.

Wolovich, W.A, 1987, Robotics: Basic Analysis and Design, CBS College Publishing.

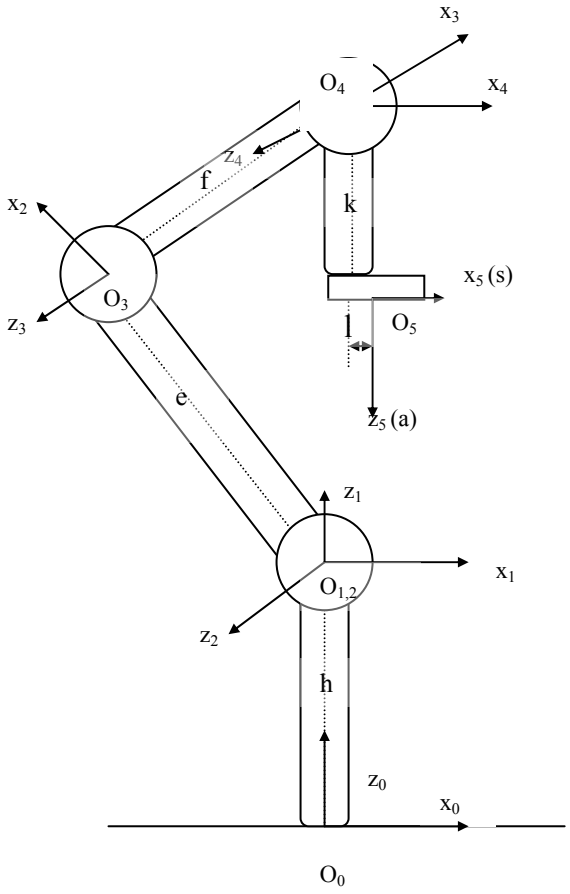
Xi, N., Tarn, T.J. and A.K. Bejczy, 1996, Intelligent planning and control for multirobot coordination: An event-based approach, IEEE Transactions on Robotics and Automation, V:12, no:3, 439-452pp.

Yıldırım, Ş. and Tunalı, T., 1999, A new methodology for dealing with uncertainty in robotic tasks, XIV. Int. Symp. on Computer and Information Sciences, Kuşadası, TURKİYE.

EKLER

Ek 1 Rhino Robot Ters Kinematiği

Aşağıda 5 serbestlik dereceli Rhino robota atanan eksenler gösterilmektedir. Bu değerler kullanılarak Denavit-Hartenberg parametreleri hesaplanmaktadır. h, e, f, k ve l eklem uzunluklarıdır.



Hesaplanan Denavit-Hartenberg (D-H) parametreleri (Wolovich, 1987) tablosu aşağıdaki gibidir:

Eklem → / D-H ↓	1	2	3	4	5
α	0	90	0	0	90
b	0	0	e	f	l
θ	θ_1	θ_2	θ_3	θ_4	θ_5
d	h	0	0	0	k

Aşağıdaki matrisler Denavit-Hartenberg parametreleri kullanılarak her eklem için elde edilmiş transformasyon matrisleridir.

T^1_0 tranformasyon matrisi:

$$\begin{bmatrix} c1 & -s1 & 0 & 0 \\ s1 & c1 & 0 & 0 \\ 0 & 0 & 1 & h \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

T^2_1 tranformasyon matrisi:

$$\begin{bmatrix} c3 & -s3 & 0 & e \\ s3 & c3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

T^3_2 tranformasyon matrisi:

$$\begin{bmatrix} c3 & -s3 & 0 & e \\ s3 & c3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

T^4_3 tranformasyon matrisi:

$$\begin{bmatrix} c4 & -s4 & 0 & e \\ s4 & c4 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

T^5_4 tranformasyon matrisi:

$$\begin{bmatrix} c5 & -s5 & 0 & l \\ s3 & c3 & -l & -k \\ s5 & c5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Oluşturulan matrislerin ${}^{1}_{0}T$ ${}^{2}_{1}T$ ${}^{3}_{2}T$ ${}^{4}_{3}T$ ${}^{5}_{4}T$ şeklinde çarpılması ile ${}^{5}_{0}T$ matrisi oluşturulur.

${}^{5}_{0}T$ matrisindeki elemanlar aşağıdaki gibi adlandırılırlar.

$$\begin{bmatrix} nx & sx & ax & px \\ ny & sy & ay & py \\ nz & sz & az & pz \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Matris çarpımları sonucunda her eleman için elde edilen değerler aşağıdaki gibidir. Aşağıdaki gösterimde "s" sinus fonksiyonunu ve "c" cosinus fonksiyonunu simgelemektedir. 1, 2, 3, 4 ve 5 sayıları sırasıyla θ_1, θ_2, θ_3, θ_4, ve θ_5 açılarını simgelemektedir. l, k, f , e ve h ise robot eklemlerinin uzunluklarını ifade etmektedir. A ve B sıfırdan farklı gerçek sayı tutan değişkenlerdir.

$$nx = c1\ c234\ c5 + s1\ s5 \quad sx = -c1\ c234\ s5 + s1\ c5 \quad ax = c1\ s234$$
$$ny = s1\ c234\ c5 - c1 s5 \quad sy = -s1\ c234\ s5 - c1\ c5 \quad ay = s1\ s234$$
$$nz = s234\ c5 \quad sz = -s234\ s5 \quad az = -c234$$

$$px = l\ c1\ c234 + k\ c1\ s234 + c1\ (f\ c23 + e\ c2)$$
$$py = l\ s1\ c234 + k\ s1\ s234 + s1\ (f\ c23 + e\ c2)$$
$$pz = l\ s234 - k\ c234 + f\ s23 + e\ s2 + h$$

θ_1 değeri hesaplanması:

$$\left.\begin{array}{l} c1\ s234 = ax \\ s1\ s234 = ay \end{array}\right\} \quad \frac{c1\ s234}{s1\ s234} = ax / ay$$

θ_1 = Atan2(ay / ax) yada θ_1 = Atan2(-ay / -ax)'dir. Dolayısıyla θ_1 ve

$\theta_1 \pm 180$ çözümleri vardır.

θ_3 değeri hesaplanması:

$-c234 = az, \quad c1\ s234 = ax, \quad s1\ s234 = ay$
$px = l\ c1az + k\ ax + c1(f\ c23 + e\ c2)$

$pz = l\ (ay / s1) + k\ az + f\ s23 + e\ s2 + h$

$(\ c1f)\ c23 + (c1\ e)\ c2) = px + az\ l\ c1 - k\ ax$

$f\ s23 + e\ s2 = pz - l\ (ay / s1) - k\ az - h$

$f\ c23 + e\ c2 = (px + az\ l\ c1 - k\ ax) / c1 = A$

$f\ s23 + e\ s2 = pz - l\ (ay / s1) - k\ az - h = B$

$f^2\ c^2 23 + e^2\ c^2 2 + 2fe\ c23c2 = A^2$

$f^2\ s^2 23 + e^2\ s^2 2 + 2fe\ s23s2 = B^2$

$f^{2(}\ c^2 23 + s^2 23) + e^{2(}\ c^2 2 + s^2 2) + 2fe\ (c23c2 + s23s2) = A^2 + B^2$

$f^2 + e^2 + 2fec(\theta_1 + \theta_2 + \theta_3) = A^2 + B^2$

$2f\ e\ c_{\theta 3} = A^2 + B^2 - f^2 + e^2$

$c_{\theta 3} = (\ A^2 + B^2 - f^2 + e^2) / 2\ fe$

$$\theta_3 = Atan2\left[\pm\sqrt{1-((A^2+B^2-f^2-e^2)/2fe)^2}\ \ /(A^2+B^2-f^2-e^2)/2fe\right]$$

Bu şekilde θ_3, θ_1 'e bağımlıdır ve θ_3 'ün 4 çözümü vardır:

Eğer θ_1 ise θ_3 *ve* $-\theta_3$.
Eğer $\theta_1 \pm 180$ ise θ_3 ve $-\theta_3$.

θ_2 değeri hesaplanması:

$fc23 + e\ c2 = A$
$fs23 + e\ s2 = B$

$fc2\ c3 - fs2\ s3 + ec2 = A$
$fs2c3 + fs3\ c2 + es2 = B$

$(fc3 + e)\ c2 - (fs3)\ s2 = A$
$(fc3 + e)\ s2 + (fs3\ )c2 = B$

$$\theta_{21} = Atan2\left[\frac{(fc3+e)B-(fs3)A}{(fc3+e)A+(fs3)B}\right]$$ eğer ($\theta_1\ \theta_3$) ve ($\theta_1 \pm 180\ \theta_3$) ise.

$$\theta_{22} = Atan2\left[\frac{(fc3+e)B-(fs3)A}{(fc3+e)A-(fs3)B}\right]$$ eğer (θ_1 -θ_3) ve ($\theta_1 \pm 180$ -θ_3) ise.

Bu durumda ($\theta_1\ \theta_3$) değerlerine bağlı olarak θ_2'nin 4 adet çözümü vardır.

θ_4 değeri hesaplanması:

c1 s234 = ax
-c234 = az

c1(s23c4 + c23s4) = ax

-(s23c4 – s23s4) = az

(c23)c4 – (s23) s4 = az

(s23)s4 + (s23) c4 = ax / c1

$$\theta_4 = Atan2\left[\frac{(c23)(ax/c1)-(s23)(-az)}{(c23)(-az)+(s23)(ax/c1)}\right]$$

Bu durumda θ_4'ün, $(\theta_1 \ \theta_2 \ \theta_3)$ bağımlı olan 4 adet çözümü vardır. Bu çözümler şöyledir:

$(\theta_1 \ \theta_{21} \ \theta_3)$ $(\theta_1 \ \theta_{22} \ -\theta_3)$
$(\theta_1 \pm 180 \ \theta_{21} \ \theta_3)$ $(\theta_1 \pm 180 \ \theta_{22} \ \theta_3)$

θ_5 değeri hesaplanması:

nx = c1c234c5+s1s5 *sx =-c1 c234 s5 + s1c5* *-c234 = az*

(s1)c5 – (-az c1)s5 = sx
(s1)s5 + (-az c1)c5 = nx

$$\theta_5 = Atan2\left[\frac{(s1)(nx)-(-azc1)(sx)}{(s1)(sx)+(-azc1)(nx)}\right]$$

θ_5, θ_1 'e bağımlıdır ve θ_1'in iki değeri vardır. Dolayısıyla θ_5'in de iki değeri vardır.

Dolayısıyla robotun dört adet ters kinematik çözümü şöyledir:

$(\theta_1$	θ_{21}	θ_{31}	θ_{41}	$\theta_{51})$
$(\theta_1$	θ_{22}	$-\theta_{31}$	θ_{42}	$\theta_{52})$
$(\theta_1$	θ_{23}	θ_{32}	θ_{43}	$\theta_{53})$
$(\theta_1 \pm 180$	θ_{24}	$-\theta_{32}$	θ_{44}	$\theta_{54})$

Ek 2 move_block işleci

```
void move_block(object *obj){

 if(!strcmp(status,"block-is-moving") &&
   !strcmp(obj→destinationOccupancy,"EMPTY")){

    strcpy(obj→type,"ARM");

    if(domain == 1) {         //karışmış  taşlar alanında çalışılıyorsa

      World_Model_Domain1 [obj→destx][obj→desty] = *obj;
      World_Model_Domain1 [obj→curlocx][obj→curlocy].name = ' ';
      Goal_Domain1 [obj→destx][obj→desty].processed = 1;
      Initial_Domain1 [obj→curlocx][obj→curlocy].processed = 1;

    } else { ///konteyner alanında çalışılıyorsa

      World_Model_Domain2[obj→destx][obj→desty] [obj→destz]  = *obj;
      World_Model_ Domain2 [obj→curlocx][obj→curlocy].[obj→curlocz] .name = ' ';
      Goal_ Domain2 [obj→destx][obj→desty] [obj→destz] .processed = 1;
      Initial_ Domain2 [obj→curlocx][obj→curlocy]. [obj→curlocz].processed = 1;
    }
      obj→name = ' ';
      obj→curlocx = obj→destx;
      obj→curlocy = obj→desty;
      obj→curlocz = obj→destz;
      obj→destx = -1;
      obj→desty = -1;
      obj→destz = -1;
      strcpy(obj→destinationOccupancy, "FULL");
      obj→distance = -1;
}
```

Blok Taşıma Eylemi

Ek 3 move_arm işleci

```
void move_arm(object *obj){

 if(!strcmp(status,"arm-is-moving") &&
       !strcmp(obj→destinationOccupancy,"FULL"));

    strcpy(obj→type,"BLOCK");

    if(domain == 1)            //karışmış  taşlar alanında çalışılıyorsa

       obj→name = World_Model_Domain1[obj→destx][obj→desty].name;
    else
       obj→name = World_Model_Domain2[obj→destx][obj→desty] [obj→destz].name;

     obj→curlocx = obj→destx;

     obj→curlocy = obj→desty;

     obj→curlocz = obj→destz;

     obj→destx = -1;

     obj→desty = -1;

     obj→destz = -1;

     obj→processed = 0;

     strcpy(obj→destinationOccupancy, "UNKNOWN");

     obj→distance = -1;
}
```

Robot Kolunu Hareket Ettirme Eylemi

Ek 4 En Kısa Yol Problemine Ait Kodun İrdelenmesi (Fonksiyonlar)

1. [void robotarm_to_closest_misplaced(int xcor,int ycor, int XCOR, int YCOR, char InputM1[D][D]);]

Bu fonksiyon robot kolunun (0, 0) başlangıç pozisyonuna en yakın hatalı yerdeki taşın koordinatlarına karar verir. xcor ve ycor değişkenleri robot kolunun başlangıç pozisyonunun koordinatlarını tutarken XCOR ve YCOR hatalı yerdeki taşın koordinatlarını tutar. char InputM1[D][D] başlangıç durum matrisidir.

2. [void misplaced_to_fixed_goal(int xcor, int ycor, int *XCOR, int *YCOR);]

xcor ve ycor şu anki hatalı yerdeki taşın koordinatlarıdır. Sabit hedef koordinatları için bir düğüm açılacaktır, bu düğümün hatalı taştan olan uzaklığı hesaplanacaktır ve hatalı yerdeki taş bu düğüme yerleştirilecektir. XCOR ve YCOR, hatalı taşın x ve y koordinatlarının değerlerini çağıran fonksiyona döndürür.

3. [void robotarm_to_all_misplaceds(int robhandx, int robhandy);]

Bu fonksiyon işletildiğinde, robot kolu hatalı yerdeki bir taşı sabit koordinatlarına yerleştirmiştir ve yeni pozisyonuna ait koordinatlardan ulaşılabilen sonraki bütün hatalı yerdeki taş koordinatlarının belirlenmesi için beklemektedir. Bu fonksiyon özyinelidir (recursive). Ulaşılabilen her düğüm için sabit bir hedef pozisyon koordinatları hesaplanır. Sabit hedef pozisyon koordinatlarının her birisi için, bütün erişilebilir düğümler hesaplanır. Bu durum, sabit bir hedeften ulaşılabilecek bir pozisyon kalmayıncaya kadar devam eder.

4. [NODEPTR setnext(NODEPTR p, int ind, int x, int y, int distance);]

Bu fonksiyon p tarafından işaret edilen düğümün ind'inci işaretini (pointer) bu fonksiyonda yeri ayrılan (allocate) düğüm olarak atar. Ayrıca **int x, int y, int distance** sahaları yeri ayrılan düğümün içeriğidir ve fonksiyon yeri ayrılan düğümün adresini döndürür.

En Kısa Yol Algoritması

1. Robot koluna ait başlangıç (0,0) koordinatları için bir düğüm aç ve mesafesini 0 olarak günle.

2. Robot kolunun başlangıcına en yakın hatalı yerdeki taşın koordinatlarını bul, bu taş için bir düğüm sahası ayır ve robot kolunun kat edeceği toplam mesafeyi hesapla.

Toplam mesafe = robot kolu ile en yakın hatalı yerdeki taş arasındaki mesafe.

3. Robot koluna en yakın hatalı yerdeki taşın sabit koordinatlarını elde et, onun için bir düğüm sahası ayır ve düğüm içerisinde koordinatlar ile kat edilecek toplam mesafeyi günle.

Toplam mesafe = robot kolu ile en yakın hatalı yerdeki taşın koordinatları + hatalı yerdeki taş ile hatalı yerdeki taşın sabit koordinatları arasındaki mesafe.

4. Robot kolunun bulunduğu yerden ulaşılabilecek "**işlenmemiş**" ve hatalı yerde olan bir taş bulduktan sonra adım 4.1'e git. Eğer robot

kolundan ulaşılabilecek hatalı yerde olan bir "işlenmemiş" taş bulunamıyorsa, robot kolunun ***önceki şu an pozisyonu*** için özyineli bir dönüş yap.

4.1 İşlenmemiş hatalı yerdeki taş için, bir düğüm sahası ayır ve düğümün içeriğini taşın koordinatları ve taşın kat edeceği toplam yol ile güncelleştir.

> Toplam mesafe = Toplam mesafe + robot kolunun bulunduğu pozisyondan hatalı yerde olan taşa kadar olan mesafe.

"işlenmemiş" olan hata yerdeki taşı, **"işlenmiş"** hatalı yerdeki taş olarak güncelleştir.

Kolun o anki pozisyonunu sabit koordinatlar olarak güncelleştir.

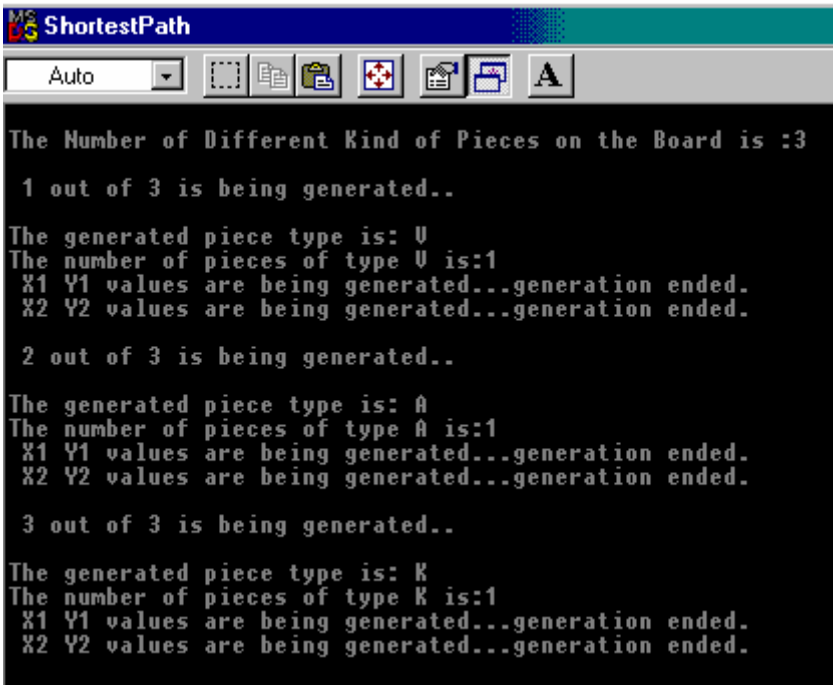
Adım 4'e git.

"İşlenmiş" olan hatalı yerdeki taşı, **"işlenmemiş"** hatalı yerdeki taş yap.

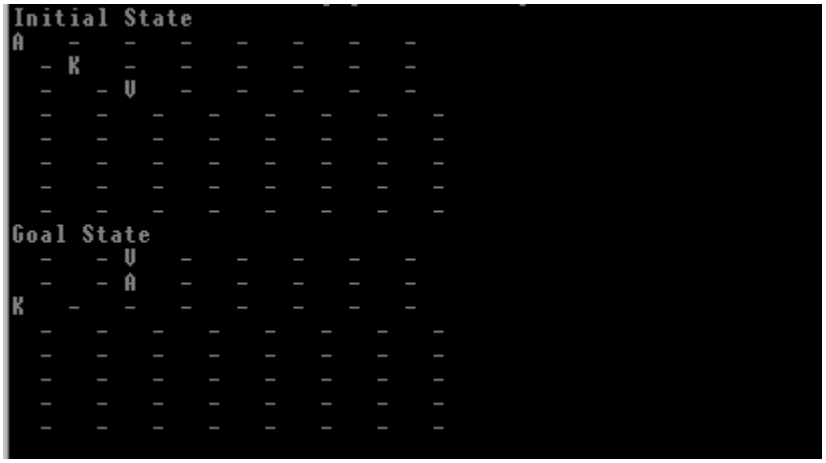
Robot kolunun o anki pozisyonunu önceki koordinatlar olarak güncelleştirerek diğer yolların da genişletilebilmesini sağla.

En Kısa Yol Programının Girdileri ve Çıktıları

Program çalıştırıldığında, ilk ekran tahta üstünde kaç farklı taş olduğu ekranda görüntülenir ve her farklı tipten kaç adet taş olacağı her taşın koordinatları ile üretilir..



Girdiler Üretiliyor



Taşların Başlangıç ve Bitiş Durumları

Üretilen Yol

Ek 5 Kullanılan Kamera, Görüntü Yakalama Kartı Ve Lensin Teknik Özellikleri

Bu projede kullanılan kamera Hitachi KPD-40 renkli kamera ve görüntü yakalama kartı Winnow Videum AV PCI kartıdır. Görüntü yakalama kartının 704 × 576 çözünürlüklere kadar olan durağan görüntü yakalama özelliği vardır. Video çözünürlüğü 640 × 480 olabilmekte ve saniyedeki görüntü sayısı PAL için 25 ve NTSC için 30 olabilmektedir. Renk sayısı maksimum 16.8 milyondur. Ses giriş ve çıkış özelliği vardır. Ses ve görüntü senkron olarak alınabilmektedir. Windows NT 4.0 ve Windows 2000 işletim sistemleri üzerinde çalıştırmak mümkündür.

Kameranın özellikleri ise söyle sıralanabilir:

Kamera 1/3 inçlik bir CCD görüntü cihazı ve bir sayısal sinyal işleme devresi kullanır.

CCD üzerindeki mikro lensler 2 lükslük aydınlatmayı yakalayabilecek özelliktedir. CCD'nin NTSC için 380,000 ve PAL için 440,000 etkin pikseli vardır. Bu şekilde NTSC için 470 ve PAL için 460 TV satırı oluşturulabilmektedir.

C veya CS yerleştirmeli lens kullanabilmektedir.

Balans kontrolü ile otomatik izleme yapabilmektedir.

Manuel iris ayarı vardır.

40° 'nin altındaki sıcaklıklarda tutulmalıdır.

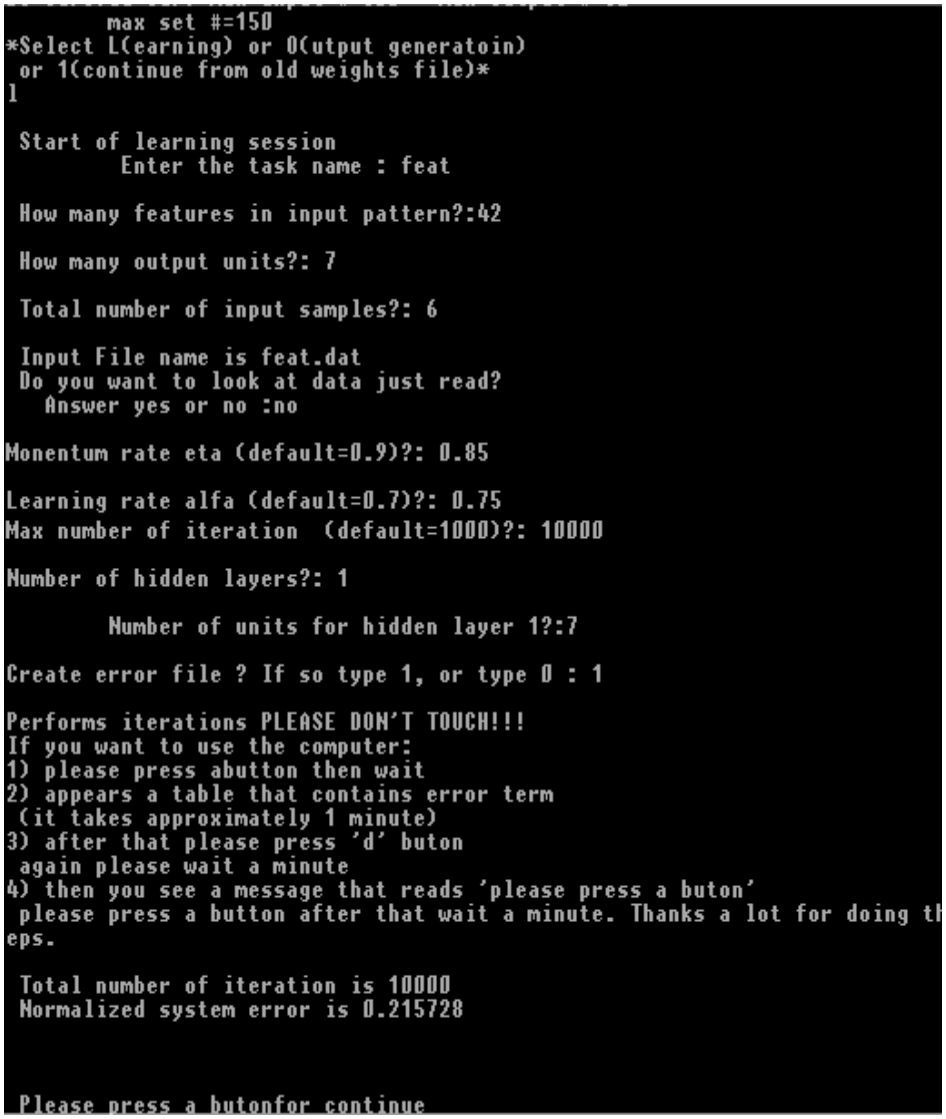
Kullanılan Cosmicar C70802 lensin özellikleri 1/3 inch'lik CCD'ye takılabiliyor olması, 8 mm odak uzaklığına sahip olması, F1.2, manuel iris ve sabit odaklama özelliklerinde olması ve CS yerleştirmeli olmasıdır.

Ek 6 Kullanılan Bitmap (.bmp) Görüntü Dosyalarının Özellikleri

Bir bitmap görüntünün başlık (header) içeriği 640 × 480 çözünürlükteki ve 256 gri seviyeli bir görüntü için aşağıdaki şekildedir. 1078'inci byte'tan itibaren de görüntünün byte'ları yer almaktadır.

fileheader.bfType	19778 //bitmap görüntü formatı
fileheader.bfSize	308280 // toplam byte sayısı
fileheader.bfReserved1	0
fileheader.bfReserved2	0
fileheader.bfOffBits	1078 //header uzunluğu
infoheader.biSize	40 // palet byte sayısı
infoheader.biWidth	640 // boyuna piksel sayısı
infoheader.biHeight	480 // enine piksel sayısı
infoheader.biPlanes	1
infoheader.biBitCount	8
infoheader.biCompression	0
infoheader.biSizeImage	0
infoheader.biXPelsPerMeter	2834
infoheader.biYPelsPerMeter	2834
infoheader.biClrUsed	0
infoheader.biClrImportant	0

Ek 7 Bir Yapay Sinir Ağı Eğitme Ekranı



Bir Yapay Sinir Ağı Eğitme Ekranı

ÖZGEÇMİŞ

1971 Isparta doğumlu ve T.C. vatandaşı olan yazar Ege Üniversitesi Bilgisayar Mühendisliği Bölümü'nde sırasıyla 1993 ve 1995'de lisans ve yüksek lisans eğitimini tamamladıktan sonra 1996'da Uluslararası Bilgisayar Enstitüsünde doktora eğitimine başlamıştır. 1997 yılında ders döneminin tamamlanmasının ardından, tez çalışmasına başlayan yazar 1999 yılında Milli Eğitim Bakanlığı Norveç Araştırma Konseyi burslusu olarak 4 ay süreyle Norveç, Trondheim, NTNU (Norwegian University of Science and Technology) üniversitesinde hibrid robot mimarileri üzerine çalışma yapmış ve bu çalışmada bahsedilen akıllı robot mimarisinin temeli atılmıştır. 2001 yılında bir yıl süre ile Tübitak NATO A2 bursu ile Carnegie Mellon Üniversitesi (CMU), Pittsburgh, USA, Robotik Enstitüsünde "akıllı işletim izleme" (intelligent execution monitoring) konusunda dağıtık robot mimarileri ortamında çalışmalarda bulunmuş, alanında önde gelen bu üniversitede değişik mimarileri inceleme ve tartışma olanağı bulmuştur. Aynı üniversitede ikinci dönem açılan dağıtık sistemler dersi asistanlığını yapmıştır. 1996 yılında araştırma görevlisi olarak başladığı Uluslararası Bilgisayar Enstitüsünde araştırma görevliliği halen devam etmektedir. Bu görev süresince yürüttüğü uygulamalı tez çalışmaları yanında "çoklu ortam projelerinde" görev yapmış, "çoklu ortam" laboratuarı sorumluluğu görevini yürütmüştür.