T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ZEKİ ETMENLERDE ÖĞRENME KABİLİYETİNİN GELİŞTİRİLMESİ VE BİR ATÖLYE TİPİ DİNAMİK ÇİZELGELEME UYGULAMASI

DOKTORA TEZİ

End. Müh. M. Emin AYDIN

Enstitü Anabilim Dalı : **ENDÜSTRİ MÜH.**
Enstitü Bilim Dalı : **ENDÜSTRİ MÜH.**

Bu tez 16.1 10.1 19**97** **tarihinde aşağıdaki jüri tarafından Oybirliği/~~Oyçokluğu~~ ile kabul edilmiştir.**

Prof. Dr. M. Akif Eyler	Prof. Dr. Harun Taşkın	Doç. Dr. Ercan Öztemel
Jüri Başkanı	**Jüri Üyesi**	**Jüri Üyesi**

TEŞEKKÜR

Bu tez, yoğun bir emek ürünüdür. Çalışma boyunca bana bilimsel ciddiyeti kazandırmak için titreyen, çalışmaların seyrini yakından takip ederek bana maddi ve manevi her türlü desteği çekinmeden veren ve en az benim kadar bu tezin ortaya çıkmasında emek sarfeden hocam sayın Doç. Dr. Ercan ÖZTEMEL beye özellikle hürmet ve şükranlarımı sunmayı bir borç bilirim.

Ayrıca akademisyenlik mesleğinin tadına varmamda bana öncülük eden, hayatım boyunca tavsiyelerini unutmayacağım Bölüm Başkanı'mız sayın Prof. Dr. Harun TAŞKIN beye hürmet ve şükranlarımı arz ederim.

Bu tezin ortaya çıkmasında maddi ve manevi destek vermekten kaçınmayan ailem ve arkadaşlarıma da teşekkür ederim. Çalışmalarımın yürümesinde bana bilgisayar bağışlamakla destek sağlayan EDSAN A.Ş.'ye de ayrıca teşekkür ederim.

İÇİNDEKİLER

SİMGELER VE KISALTMALAR VI
ŞEKİLLER LİSTESİ VII
TABLOLAR LİSTESİ IX
ÖZET X
SUMMARY XIV

BÖLÜM 1 GİRİŞ 1
1.1. Tezin amacı ve kapsamı 3
BÖLÜM 2 ZEKİ ETMENLER 5
2.1. Bir Zeki Etmenin Genel Yapısı 7
2.2. Zeki Etmenlerin Özellikleri 8
2.2.1. Yapısal özellikler 8
2.2.1.1. Geleneksel yaklaşım 8
2.2.1.2. Davranış tabanlı yaklaşım 11
2.2.2. Yeteneksel özellikler 14
2.2.2.1. Geleneksel yaklaşım 16
2.2.2.2. Davranış tabanlı yaklaşım 17
2.3. Zeki Etmen Türleri 18
2.3.1. Yapısal olarak zeki etmen türleri 18
2.3.2. İşlevsel olarak zeki etmen türleri 19
2.3.2.1. Robotik etmenler 20
2.3.2.2. Yazılım etmenleri 21

BÖLÜM 3 ZEKİ ETMENLERDE ÖĞRENME 22
3.1. Destekli Öğrenme 23

3.1.1. Genetik algoritmalara dayalı çalışmalar 26
3.1.2. Geçici farklara dayalı algoritmalar 27
3.2. *Q-Öğrenme* Algoritması 29

BÖLÜM 4 DİJİT OYUNU PROBLEMİ 35
4.1 Problemin Tanımı ve Özellikleri 35
4.2. Dijit Oyununu Oynayacak Zeki Etmenin Tasarımı 36
4.2.1. Zeki etmenin mimari yapısı 37
4.2.2. Zeki etmenin *Q-öğrenme* algoritması ile eğitilmesi 42
4.2.2.1. Zeki etmenin eğitimindeki yapısal boyut 42
4.2.2.2.Zeki etmenin eğitilmesi 46

BÖLÜM 5 *Q-I* ÖĞRENME ALGORİTMASI 49
5.1. *Q-I* Öğrenme Algoritması 51
5.2. Tartışma 53
5.3. Sonuç 54

BÖLÜM 6 *Q-II* ÖĞRENME ALGORİTMASI 55
6.1. *Q-II* Öğrenme Algoritması 55
6.2. Dijit Oyununda *Q-II* Algoritması 57
6.3. Sonuç 60

BÖLÜM 7 *Q-I* VE *Q-II* ÖĞRENME ALGORİTMALARINDA YAPISAL KREDİLENDİRME 61
7.1. *Q-I* ve *Q-II* Algoritmalarında Yapısal Kredilendirme 61
7.1.1. Bilgilerin yeniden gösterimi 64
7.2. Öbekleme Analizi 67
7.2.1. Keskin c-Ortalamalar 67
7.2.2. Geliştirilmiş keskin c-ortalama yöntemi 70
7.3. Tartışma 71
7.4. Sonuç 73

BÖLÜM 8 *Q-III* ÖĞRENME ALGORİTMASI 74
8.1. *Q-III* Öğrenme Algoritması 75
8.2. *Q-III* ile Dijit Oynunun Öğretilmesi 78
8.3. Sonuç 80

BÖLÜM 9 İŞ ÇİZELGELEMEDE ZEKİ ETMENLER 81
9.1 Genel Atelye Tipi Çizelgeleme Problemi 81
9.1.1. Performans ölçütleri 83
9.2. Atölyede Dinamik Çizelgeleme 87
9.3. Zeki Etmen Tabanlı Bir Dinamik Çizelgeleme Sistemi 90
9.3.1. İşlerin çizelgeleneceği atölyenin özellikler 91
9.3.2. Dinamik iş çizelgeleme için atölye benzetim ortamı 92
9.3.3. Çizelgeleyici zeki etmen 95
9.3.3.1. Çizelgeleyici zeki etmenin yapısı 95
9.3.3.2. Zeki etmenin öğretilmesi 95
9.3.3.3. Çizelgeleyici zeki etmen ile iş çizelgeleme 99
9.4. Sonuç 100

BÖLÜM 10 SONUÇ 101

KAYNAKLAR 104

ÖZGEÇMİŞ 113

SİMGELER VE KISALTMALAR

x	Durum değişkeni
a	Davranış değişkeni
$Q(x,a)$	a davranışının x durumunda uygulanmasının yararlanma değeri
$e(y)$	y durumunun beklenen en büyük yararlanma değeri
β	Öğrenme katsayısı
γ	Çürütme katsayısı
δ	Ceza fonksiyonu katsayısı
ω	Ceza fonksiyonu başlangıç değeri
μ	Davranışlar arası ilişki katsayısı
$Q_y(x,a)$	$Q(x,a)$ 'nın güncelleşmiş değeri
$Q_e(x,a)$	$Q(x,a)$ 'nın güncelleşme öncesi değeri
$\hat{Q}$	Aktif olmayan davranışların ortalama yararlanma değeri
T_{ij}	i. öbek'e dahil edilen verilerin j. veri de dahil edilince elde edilen toplam değeri
v_{ij}	i. öbek merkezinin j. boyut değeri
d_{ij}	i. verinin j. öbek merkezine olan uzaklığı
TWK	Toplam iş miktarı (Total Work)
k	darlık faktörü
dd	teslim tarihi

ŞEKİLLER LİSTESİ

Şekil-2.1; Bir zeki etmenin çevresi ile etkileşimi 5
Şekil-2.2; Genel olarak bir zeki etmenin yapısal şeması 7
Şekil-2.3; Klasik yaklaşımla geliştirilen bir zeki etmenin genel yapısı 9
Şekil-2.4; Geleneksel yaklaşımla geliştirilen zeki etmenlerin genel yapısı.. 12
Şekil-2.5; Davranış dayalı zeki etmenlerin genel yapısı..... 12
Şekil-2.6; Zeki etmende işlev-davranış-mekanizma ilişkisi 14
Şekil-3.1; Destekli öğrenme stratejisi ile öğrenen bir zeki etmenin çevresi ile etkileşimi 24
Şekil-3.2; Genel olarak bir destekli öğrenme algoritması 25
Şekil-3.3; *Q öğrenme* algoritması..... 30
Şekil-4.1; Bir durum gösteren dijit seti örneği 36
Şekil-4.2 ; Zeki etmenin genel yapısı 38
Şekil-4.3 ; Birinci davranışın örnek bir uygulaması 39
Şekil-4.4; İkinci davranışın örnek bir uygulaması 40
Şekil-4.5; Üçüncü davranışın örnek bir uygulaması 41
Şekil-4.6; Dördüncü davranışın örnek bir uygulaması 41
Şekil-4.7; Zeki Etmenin *Q* Öğrenme ile Eğitilmesi İçin Kurulan Sistem ... 42
Şekil-4.8; *Q* öğrenme sürecinin gösterimi 43
Şekil-4.9; Modüller arası ilişkiler diyagramı 44
Şekil-4.10; Destekleme mekanizmasının puanlamada kullandığı kuralların gösterimi 45
Şekil-4.11; Üç farklı durum için 50 adımlık davranış tercih profili 47
Şekil-5.1; Tipik bir öğrenme süreci 50
Şekil-5.2; *Q-I öğrenme* algoritması 52

Şekil-5.3; Cezalandırma mekanizmasının diğer modüllerle olan etkileşimi 53
Şekil-6.1; *Q-II öğrenme* algoritması 56
Şekil-6.2; Paralel güncelleme kuralının güncelleme modülü ile olan etkileşimi 57
Şekil-6.2; İterasyon sayısı ortalamasının farklı μ değerlerine göre değişimi 59
Şekil-7.1; Yapısal kredilendirmenin öğrenme prosedürüne eklenmesi 62
Şekil-7.2; Yapısal kredilendirme boyunca öğrenme ve veri toplama bloklarının detaylı işleyişi 63
Şekil 7.3; Yeniden gösterim modülünün temsili gösterimi. 66
Şekil-7.4; Öğrenme süreci sonunda zeki etmenin yapısı 71
Şekil-7.5; İki farklı başlangıç durumu ile hedefe ulaşmada seçilen davranışlar profili 73
Şekil-8.1; *Q-III* öğrenme algoritmasının adım adım gösterimi..... 78
Şekil-8.2; Zeki etmenin *Q-III* algoritması ile eğitilmesi 79
Şekil-9.1; Önerilen dinamik çizelgeleme benzetim sisteminin genel akış şeması 94
Şekil-9.2; Çizelgeleyici zeki etmenin genel yapısı..... 95
Şekil-9.3; Destekleme mekanizmasının zeki etmen ve benzetim ortamı ile etkileşimi 96
Şekil-9.4; Zeki etmenin *Q-III* algoritması ile öğrenmesi 97
Şekil-9.5; Bilgilerin yeniden gösterimi prosedürü 98
Şekil-9.6; Puanlandırma prosedürü 98
Şekil-9.7; Öncelik kurallarının ortalama pozitif gecimeye göre performansı 100

TABLOLAR LİSTESİ

Tablo-4.1;	Yerel optimuma örnekler	47
Tablo-5.1;	Bazı başlangıç durumlara *Q* ve *Q-I* algoritmalarının öğrenme iterasyon sayısı	53
Tablo-6.1;	Farklı μ değerleri için ortalama iterasyon sayıları.....	58
Tablo-6.2;	Bazı durumların *Q, Q-I* ve *Q-II* algoritmalarının öğrenme iterasyon sayısının karşılaştırılması	59
Tablo-7.1;	*Q, Q-I, Q-II* ve Öbekleme analizi destekli öğrenme süreçlerinin iterasyon sayısı bakımından karşılaştırılması .	72
Tablo-8.1;	Bazı durumların *Q, Q-I, Q-II* ve *Q-III* algoritmalarının öğrenme iterasyon sayısı	80
Tablo-9.1;	Zeki etmen çizelgeleme performansının diğer öncelik kuralları ile karşılaştırılması	100
Tablo-10.1;	Tez boyunca *Q-öğrenme* algoritmasında gerçekleştirilen gelişim.....	102

ÖZET

Bilim ve teknolojideki hızlı gelişme, problemlerin gittikçe karmaşıklaşması araştırmacıları tüm disiplinlerde kullanılmak üzere yeni yaklaşımlara ve çözüm tekniklerine yöneltmiştir. Bilgisayarların hayata girmesi ile yeni metodlar için yeni imkanlar doğmuştur. Bunların başında kuşkusuz yapay zeka teknolojileri gelmektedir. İnsanın yaşamı boyunca kazandığı tecrübelerden ve bilinçsel eylemlerinden yararlanmak amacı ile gelişim gösteren yapay zekanın bu gün geldiği son noktalardan biri de zeki etmenlerin kullanımıdır.

Zeki etmenler, çevrelerindeki olayları algılayabilen, birbirinden bağımsız hareket edebilen, bulundukları ortamlara uyum sağlayabilen otonom yazılım ve donanım birimleridir. Genel olarak *algılama*, *kavrama* ve *eylem* birimlerinden oluşurlar. Çevredeki olayları alıcılar yardımı ile algılayarak onları belirli amaçlar doğrultusunda yorumlarlar, efektörleri yardımı ile de çeşitli eylemleri gerçekleştirirler. Mesela bir zeki etmen bir robot ise, olayları algılayıp yorumladıktan sonra efektörlerine sağa, sola dönmesini, yürümesini, konuşmasını, bir vanayı açıp kapamasını vb gibi aktiviteleri yaptırabilir. Bunun yanında robotun yürüme eylemini gerçekleştirirken önüne çıkan engelleri farkedebilmesi, ezmemek üzere durması veya alternatif bir eylem gerçekleştirebilmesi de son derece önemlidir. Bu çerçevede zeki etmenlerin bir dış etki veya komut ile değil tamamen öz bilgileri ile her zaman aktif olarak eylemlerini yürütmeleri ve anlamlı bir algılamayı gerçekleştirdikleri an gerekli tepkiyi göstermeleri yazılım sistemlerine yeni imkanlar sağlamaktadır.[Becket ve Bedler, 1993].

Zeki etmenler yazılım sistemlerine bir yapısal yenilik getirmiştir. Bundan yapay zeka ürünleri de payını almaktadır. Geleneksel sistemlerden farklı olarak sistem mimarilerini standardize ederek sanal ortamlarda canlılara benzer şekilde tamamen veya yarı özerk işlev yürütmeyi sağlamaktadır. Sistemlerin bundan böyle algılama-kavrama-eylem ana birimleri ile çatılarak hedeflenen görevleri yapmak üzere çevresi ile etkileşen bir zeki etmen haline dönüştürülmelerini önermektedir. Hedeflenen görevlerin karmaşıklığına göre bir sistemin birden fazla zeki etmen kullanılarak tasarlanmasını, zeki etmenlerin çevre ile ve kendi aralarında etkileşerek işlevleri yürütme imkanını getirmektedir. Bu yapısal yenilik sistemlerin her ortama kolaylıkla adapte edilebilmelerini sağlamaktadır.

Zeki etmenlerin gerek bulundukları ortama rahat uyum sağlayabilmeleri gerekse yeni yetenekler kazanabilmeleri için eğitilebilmeleri oldukça önemli bir konudur. Zeki etmenlerin eğitilmesinde gerçek zamanlı öğrenmede yeterli bilginin her zaman sağlanamaması dolayısıyla doğal ortamlarda canlıların deneme-yanılma şeklinde öğrenmelerine benzer şekilde çoğunlukla *destekli öğrenme stratejisi*nin kullanıldığı görülmektedir. Destekli öğrenme stratejisi ile geliştirilen algoritmalar çalışma şekli birbirlerine benzerler. En önemli benzerlik, hepsinin deneme-yanılma tekniğine göre öğrenme eylemini öngörmeleridir. Burada öğrenme, zeki etmenin çevreden algıladığı aktif duruma karşın bir davranış göstermesi ve bu davranışa çevreden veya bir destekçiden gelen uyarı ile durum-davranış arasında bir eşleştirme kurabilmesi şeklinde gerçekleşir. Bu olay şöyle gerçekleşir: Zeki etmen çevredeki bir durumu algılar, onu bir değerlendirme fonksiyonuna göre değerlendirerek, duruma karşılık yapması gereken davranışı belirler ve o davranışı uygular. Bu davranış çevrede bir durum değişmesine dolayısıyla yeni bir duruma neden olur. Öğrenme boyunca zeki etmenin davranışlarını izleyen bir destekleme mekanizması çevredeki bu değişikliği değerlendirerek etmene bir *yönlendirme uyarısı* gönderir. Bu yapılan davranışın uygun olup olmadığı konusunda zeki etmene bir fikir verir. Oluşan yeni durum ve alınan uyarı ile zeki etmen değerlendirme mekanizmasında kullandığı kriterleri güncelleştirir. Bu deneme-yanılma çevrimi zeki etmen ilgili durum-davranış ilişkisini kavrayıncaya kadar tekrarlanır. Bu işlemler esnasında bazı problemler ortaya çıkmaktadır. Bunlardan birincisi, *geçici kredilendirme problemi* olarak bilinen ve

öğrenme esnasında oluşan hatanın nasıl geriye doğru yayılması gerektiği konusudur. İkincisi ise, kazanılmış tecrübenin yeni durumlar için nasıl kullanılacağı konusudur. Bu da *yapısal kredilendirme* olarak bilinir. Bu her iki problemin çözümü için yapılmış çalışmalar farklı öğrenme algoritmalarının gelişmesini sağlamıştır.

Destekli öğrenme algoritmaları olarak ortaya çıkan çalışamazların en eskilerinden birisi *geçici farklar* üzerine tasarlanmış olan *AHC* algoritmasıdır. Sutton (1988) tarafından geliştirilen bu algoritma bir çok uygulamaya konu olmuş ve yenilerinin geliştirilmesine de kaynak olmuştur. Bunun yanında Watkins (1989) tarafından geliştirilen *Q-öğrenme* algoritması da aynı şekilde yaygın bir uygulama alanı bulmuştur. Benzer şekilde, bir takım istatistiksel değerlemeye dayanan öğrenme algoritmaları ile genetik algoritmalara dayanan bir çok algoritma literatürde yerini almıştır. Bu çalışmada sağlam bir teorik tabanı ve pratik uygulama yelpazesi bulunan *Q-öğrenme* algoritması ve onun yukarıda ifade edilen geçici ve yapısal kredilendirme problemlerine çözüm getirecek iyileştirme önerileri sunulacaktır.

Tez boyunca amaçlar *Q-öğrenme* algoritmasının hedeflenen problemleri irdelenmiştir. Bu irdeleme çerçevesinde üç kademeli bir iyileştirme gerçekleştirilmiştir. Birinci adımda yalnızca yerel optimum problemini çözmeye yönelik olarak *Q-I* yordamı (cezalandırma fonksiyonu), ikinci adımda hem yerel optimuma hem de yavaş yakınsama problemine çözüm olmak üzere paralel güncelleştirme yaklaşımı (*Q-II*) önerilmiştir. Son adımda ise yerel optimum, yavaş yakınsama ve genelleştirme problemlerini çözen *Q-III* algoritması sunulmuştur. Bu gelişmelerden sonra yeni algoritma ile öğretilmek üzere tasarlanan zeki etmenin, dinamik iş çizelgeleme problemine yeni bir çözüm getirip getiremeyeceği araştırılmıştır. Yapılan çalışmalar sonunda zeki etmenlerin öğrenme kabiliyetlerini kullanarak dinamik atölye ortamında durumlara uygun iş çizelgelemeyi başarabildiği görülmüştür.

Tezin ikinci bölümünde zeki etmenler genel olarak tanıtıldıktan sonra üçüncü bölümünde destekli öğrenme algoritmaları hakkında tanıtıcı bir literatür taraması verilmiştir. Bölüm 4'te dijit oyununu oynayacak bir zeki etmen tasarlanmış ve *Q-*

öğrenme algoritması ile eğitilme çalışması sunulmuştur. Bölüm 5 *Q-öğrenme* ile yapılan öğretim sırasında ortaya çıkan yerel optimum problemini aşmak amacı ile bir cezalandırma fonksiyonu içeren yeni bir *Q-öğrenme (Q-I)* tanıtılmıştır. Bölüm 6 ise *Q-öğrenme*'nin yavaşlığı problemi ele alınmış ve paralel güncelleştirme yaklaşımı ile yeni bir *Q-öğrenme (Q-II)* algoritması önerilmiştir. Bölüm 7 sunulan her iki yeni algoritma için bir yapısal kredilendirme sistemini, Bölüm 8 ise önerilen bu yapısal kredilendirme sistemini iyileştirmek amacı ile yeni bir *Q-öğrenme* algoritmasını *(Q-III)* sunmaktadır. Bölüm 9'da Endüstri Mühendisliği'nin önemli problemlerinden olan atölyede iş çizelgeleme ele alınmıştır. Dinamik bir atölye ortamında iş çizelgelemesini tezgahlara iş atayarak yapacak olan bir zeki etmenin tasarlanması ve *Q-III* algoritması ile durumlara göre iş atamasının da hangi öncelik kuralını kullanacağını öğrenmesi ele alınmıştır. Bölüm 10'da da tezin sonuçları verilmiştir.

SUMMARY

Intelligent agents are autonomous systems which perform appropriate behavior using their own knowledge in a dynamic environment. [Steels (1994) , Wooldbridge and Jennings (1995), Hayes-Roth (1990), Brooks (1986), Maes and Brooks (1991), Dorigo and Colombetti (1994)]. They consist of three main parts: perception; to receive messages of environment, cognition; to evaluate these messages in order to make decisions, and action; to perform the decided action. The distinctive characteristic of intelligent agents is their autonomous capability. Details on this topic are presented in Chapter 2.

Learning, which can be defined as improving behavior through the time, is one of the most important topics in research on intelligent agents. In these agents, especially, reinforcement learning techniques are widely employed. In this case, the agent has to take a reinforcement signal which is produced against its actions into account. Among reinforcement learning algorithms, *Q-learning* is reported to be successfully implemented [Mahadevan and Connell (1992), Singh (1992), Lin (1992), and Barto et al (1995)].

Q-learning is realized as an asynchronous dynamic programming method which provides agents with the capability of learning to act optimally in Markovian domains by experiencing the consequences of actions, without requiring them to build map of the respective domain [Watkins and Dayan, (1992)].

The main idea behind the Q learning is the use of a single data structure called *the utility function (Q(x,a))*. That is, the utility of doing action a in state x. During

learning, this algorithm updates the value of $Q(x,a)$ using $<x,a,r,y>$ tuples, where r represents the reinforcement signal (payoff) of the environment and y represents the new state which is to be obtained after the execution of action a in state x. $Q(x,a)$ is computed for each action as:-

$$Q(x,a) = E(r + \gamma e(y) | x,a).$$

where γ is a discounted constant value in $[0,1]$ interval and $e(y)$ is the expected value of y this is also computed as:-

$$e(y) = max[Q(y,a), for \forall a]$$

Q-learning algorithm first initializes the Q value of each action to 0. It then repeats the following procedure. The action with the maximum Q value is selected and activated. Corresponding Q value of that action is then updated using the following equation (updating rule):-

$$Q^n(x,a) \leftarrow Q^o(x,a) + \beta(r + \gamma e(y) - Q^o(x,a))$$

where $Q^o(x,a)$, $Q^n(x,a)$ represent the old and the new Q values of action a in state x, and β is the learning coefficient changing in $[0,1]$ interval. More information *Q-learning* algorithm can be found in details with a literature survey, in Chapter 3.

An illustrative example is presented in Chapter 4 to show the problems which have been emerged during the implementation of Q learning. In this example, the agent deals with a set of vectors each which consists of 8 digits each. The aim of the agent is to transmit the initial state into the goal state through an iterative learning. In each iteration, the agent perceives a vector and tries to transmit it into the goal state using a set of behavioral rules (actions). The following rules are employed.

- To make two digits as the same,
- To make two digits as different,
- To change one of the two digits
- To change the places of the two digits (mutually)
- No change

The agent has some limitations such as selecting and activating only one action at a time and considering at most randomly selected two digits in each action. The ran-

dom selection of the digits makes the problem too difficult to handle. That is due to the fact that this problem has a non-Markovian nature. Since *Q-learning* is developed by using a Markovian decision process, it is not expected to be successful in this case. Therefore the algorithm needs to be modified in order to handle this.

Q-learning algorithm works in such a way that the agent gains experience by trial and error throughout the execution of actions. During this process, the agent figures out how to assign credit or blame to each of its actions in order to improve the behavior. This is called *temporal credit assignment*. Once the agent learns how to behave, it can generalize the knowledge it posses and recall the knowledge when required. The ability of the agent to use its past experience (generalization) is called *structural credit assignment*. However, both, *temporal credit assignment* and *generalization* are not easy to achieve for certain problems such as the one described above.

The problems suffered by traditional *Q-learning* procedure are discussed by Whitehead and Lin (1995), traditional *Q-learning* was developed utilizing a Markovian decision process. Therefore, it may not be implemented as successful as expected for learning in non-Markovian domains. There may be two problems facing local optimum and taking long time to learn. Some attempts have already been made to solve these problems. To overcome the *local optimum* problem, traditional *Q-learning* is modified and a punishment mechanism is employed in order to prevent the activation of the same action over and over again. This algorithm is called *Q-I* and its details are given in Öztemel and Aydın (1997). Although *Q-I* solves the *local optimum* problem in most of the cases, it takes too much time for the agent to reach the conclusion. In Chapter 5 and 6, two versions of *Q-learning*, called as *Q-I* and *Q-II*, are introduced to solve these problems.

The main idea behind *Q-I* is to punish the consecutively performed behaviour which is selected in a wrong way. During the learning, a punishment mechanism observes the learning process to direct it into the right way. To perform the punishment, a function is constructed from Q values of non-selected behavior. Details are explained in Chapter 5.

Q-I managed to solve the local optima problem. However, the learning time remains as a problem. In order to solve both the local optimum and speed problem at the same time, *Q-II* learning algorithm is developed. As stated earlier, the *local optimum* is realized when the same action is activated more than a certain number of iteration. If not a local optimum, the agent usually takes too much time to learn how to select and execute the actions. The reason for this may be the activation of a single action at a time. It is believed that the execution of a single action may easily reveal the effect of that action within the environment [Lin, 1992]. *Q-II* propose to execute a single action, but to update the utilization of all actions in parallel taking the fitness of the action into consideration. This prevents local optimum and enables learning more faster. Since all *Q* values are updated at each iteration, there is always a chance for an activity to be selected and executed. The performances of traditional *Q, Q-I* and *Q-II* are compared with respect to the local optimum and learning time for the problem defined in Chapter 6

Another stage of *Q-learning* studies is to concern with *structural credit assignment* (generalization) problem. *Q-learning* has not a generalization capability in nature. One of the aims of this thesis is to construct a robust structural credit assignment method which is considered in an off-line manner in Chapter 7 and with a new development in *Q-learning* considered in an off-line manner in Chapter 8. The final version of *Q-learning* in this thesis is *Q-III* which composed of *Q-II* and *Hard c-Means* algorithms. To compose both algorithms, the expectation term, *e(y),* is redefined as a distance function instead of a probabilistic maximum function. By these refinement, *Q-learning* composed with the clustering method to create a more definite structural generalization. Details are widely explained in Chapter 8.

In Chapter 9, a dynamic job-shop scheduling system designed by intelligent agent technology is introduced. An intelligent agent is designed to schedule the jobs in the shop using three dispatching rules according to the jobs and shop floor states. Results are so satisfied that development of an intelligent agent based scheduling system in sophisticated manner can solve many problems of this area.

BÖLÜM 1 GİRİŞ

Bilim ve teknolojideki hızlı gelişme, problemlerin gittikçe karmaşıklaşması araştırmacıları tüm disiplinlerde kullanılmak üzere yeni yaklaşımlara ve çözüm tekniklerine yöneltmiştir. Bilgisayarların hayata girmesi ile yeni metotlar için yeni imkanlar doğmuştur. Bunların başında kuşkusuz yapay zeka teknolojileri gelmektedir. İnsanın yaşamı boyunca kazandığı tecrübelerden ve bilinçsel eylemlerinden yararlanmak amacı ile gelişim gösteren yapay zekanın bu gün geldiği son noktalardan biri de zeki etmenlerin kullanımıdır.

Zeki etmenler, çevrelerindeki olayları algılayabilen, birbirinden bağımsız hareket edebilen, bulundukları ortamlara uyum sağlayabilen otonom yazılım ve donanım birimleridir. Genel olarak *algılama*, *kavrama* ve *eylem* birimlerinden oluşurlar. Çevredeki olayları alıcılar yardımı ile algılayarak onları belirli amaçlar doğrultusunda yorumlarlar, efektörleri yardımı ile de çeşitli eylemleri gerçekleştirirler. Mesela bir zeki etmen bir robot ise, olayları algılayıp yorumladıktan sonra efektörlerine sağa, sola dönmesini, yürümesini, konuşmasını, bir vanayı açıp kapamasını vb gibi aktiviteleri yaptırabilir. Bunun yanında robotun yürüme eylemini gerçekleştirirken önüne çıkan engelleri fark edebilmesi, ezmemek üzere durması veya alternatif bir eylem gerçekleştirebilmesi de son derece önemlidir. Bu çerçevede zeki etmenlerin bir dış etki veya komut ile değil tamamen öz bilgileri ile her zaman aktif olarak eylemlerini yürütmeleri ve anlamlı bir algılamayı gerçekleştirdikleri an gerekli tepkiyi göstermeleri yazılım sistemlerine yeni imkanlar sağlamaktadır.[Becket ve Bedler, 1993].

Zeki etmenler yazılım sistemlerine bir yapısal yenilik getirmiştir. Bundan yapay zeka ürünleri de payını almaktadır. Geleneksel sistemlerden farklı olarak sistem mimarilerini standardize ederek sanal ortamlarda canlılara benzer şekilde tamamen veya yarı özerk işlev yürütmeyi sağlamaktadır. Sistemlerin bundan böyle algılama-kavrama-eylem ana birimleri ile çatılarak hedeflenen görevleri yapmak üzere çevresi ile etkileşen bir zeki etmen haline dönüştürülmelerini önermektedir. Hedeflenen görevlerin karmaşıklığına göre bir sistemin birden fazla zeki etmen kullanılarak tasarlanmasını, zeki etmenlerin çevre ile ve kendi aralarında etkileşerek işlevleri yürütme imkanını getirmektedir. Bu yapısal yenilik sistemlerin her ortama kolaylıkla adapte edilebilmelerini sağlamaktadır.

Zeki etmenlerin gerek bulundukları ortama rahat uyum sağlayabilmeleri gerekse yeni yetenekler kazanabilmeleri için eğitilebilmeleri oldukça önemli bir konudur. Zeki etmenlerin eğitilmesinde gerçek zamanlı öğrenmede yeterli bilginin her zaman sağlanamaması dolayısıyla doğal ortamlarda canlıların deneme-yanılma şeklinde öğrenmelerine benzer şekilde çoğunlukla *destekli öğrenme stratejisi*nin kullanıldığı görülmektedir. Destekli öğrenme stratejisi ile geliştirilen algoritmalar çalışma şekli birbirlerine benzerler. En önemli benzerlik, hepsinin deneme-yanılma tekniğine göre öğrenme eylemini öngörmeleridir. Burada öğrenme, zeki etmenin çevreden algıladığı aktif duruma karşın bir davranış göstermesi ve bu davranışa çevreden veya bir destekçiden gelen uyarı ile durum-davranış arasında bir eşleştirme kurabilmesi şeklinde gerçekleşir. Bu olay şöyle gerçekleşir: Zeki etmen çevredeki bir durumu algılar, onu bir değerlendirme fonksiyonuna göre değerlendirerek, duruma karşılık yapması gereken davranışı belirler ve o davranışı uygular. Bu davranış çevrede bir durum değişmesine dolayısıyla yeni bir duruma neden olur. Öğrenme boyunca zeki etmenin davranışlarını izleyen bir destekleme mekanizması çevredeki bu değişikliği değerlendirerek etmene bir *yönlendirme uyarısı* gönderir. Bu yapılan davranışın uygun olup olmadığı konusunda zeki etmene bir fikir verir. Oluşan yeni durum ve alınan uyarı ile zeki etmen değerlendirme mekanizmasında kullandığı kriterleri güncelleştirir. Bu deneme-yanılma çevrimi zeki etmen ilgili durum-davranış ilişkisini kavrayıncaya kadar tekrarlanır. Bu işlemler esnasında bazı problemler ortaya çıkmaktadır. Bunlardan birincisi, *geçici kredilendirme problemi* olarak bilinen ve öğrenme esnasında oluşan hatanın nasıl geriye doğru yayılması gerektiği konusudur.

İkincisi ise, kazanılmış tecrübenin yeni durumlar için nasıl kullanılacağı konusudur. Bu da *yapısal kredilendirme* olarak bilinir. Bu her iki problemin çözümü için yapılmış çalışmalar farklı öğrenme algoritmalarının gelişmesini sağlamıştır.

Destekli öğrenme algoritmaları olarak ortaya çıkan çalışmaların en eskilerinden birisi *geçici farklar* üzerine tasarlanmış olan *AHC* algoritmasıdır. Sutton (1988) tarafından geliştirilen bu algoritma bir çok uygulamaya konu olmuş ve yenilerinin geliştirilmesine de kaynak olmuştur. Bunun yanında Watkins (1989) tarafından geliştirilen *Q-öğrenme* algoritması da aynı şekilde yaygın bir uygulama alanı bulmuştur. Benzer şekilde, bir takım istatistiksel değerlemeye dayanan öğrenme algoritmaları ile genetik algoritmalara dayanan bir çok algoritma literatürde yerini almıştır. Bu çalışmada sağlam bir teorik tabanı ve pratik uygulama yelpazesi bulunan *Q-öğrenme* algoritması ve onun yukarıda ifade edilen geçici ve yapısal kredilendirme problemlerine çözüm getirecek iyileştirme önerileri sunulacaktır.

1.1. Tezin amacı ve kapsamı

Bu tez, yapay zeka teknolojilerinden zeki etmenlerin öğrenme kabiliyeti ile daha iyi desteklenmesi ve öğrenebilen zeki etmenlerin imalat sistemlerinde kullanılmaları konusunda yapılmış bir araştırmayı sunmaktadır. Tezin amaçları;

1. zeki etmenlerde öğrenme kabiliyetinin geliştirilmesi çerçevesinde *Q-öğrenme* algoritmasının **geçici kredilendirme** işlemlerinde meydana gelen **yerel optimum** problemine bir çözüm önermek,

2. *Q-öğrenme* algoritmasının sağladığı geçici karakterdeki deneyimi kalıcı bir deneyim haline dönüştürebilmek yani kalıcı bir **yapısal kredilendirme** yöntemi önermek,

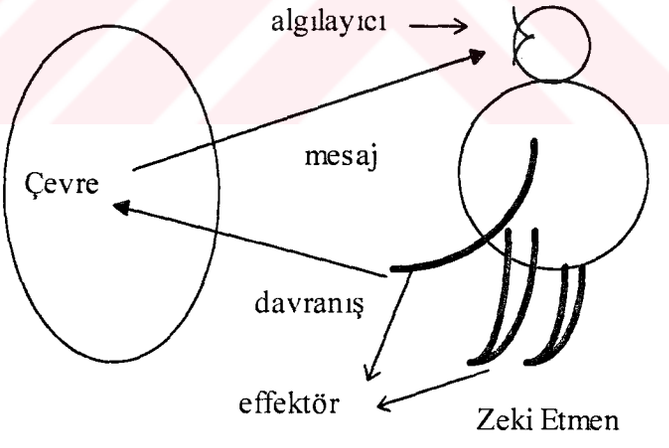
3. bir endüstriyel uygulama olarak, dinamik atölye ortamında alternatif öncelik kurallarını (SPT, COVERT, CR) karışık kullanan bir iş çizelgeleme sistemi prototipini bir öğrenebilen zeki etmen kullanarak geliştirmek

şeklinde sıralanabilir. Tez boyunca amaçlar *Q-öğrenme* algoritmasının hedeflenen problemleri irdelenmiştir. Bu irdeleme çerçevesinde üç kademeli bir iyileştirme gerçekleştirilmiştir. Birinci adımda yalnızca yerel optimum problemini çözmeye yönelik olarak *Q-I* yordamı (cezalandırma fonksiyonu), ikinci adımda hem yerel optimuma hem de yavaş yakınsama problemine çözüm olmak üzere paralel güncelleştirme yaklaşımı (*Q-II*) önerilmiştir. Son adımda ise yerel optimum, yavaş yakınsama ve genelleştirme problemlerini çözen *Q-III* algoritması sunulmuştur. Bu gelişmelerden sonra yeni algoritma ile öğretilmek üzere tasarlanan zeki etmenin, dinamik iş çizelgeleme problemine yeni bir çözüm getirip getiremeyeceği araştırılmıştır. Yapılan çalışmalar sonunda zeki etmenlerin öğrenme kabiliyetlerini kullanarak dinamik atölye ortamında durumlara uygun iş çizelgelemeyi başarabildiği görülmüştür.

Tezin ikinci bölümünde zeki etmenler genel olarak tanıtıldıktan sonra üçüncü bölümünde destekli öğrenme algoritmaları hakkında tanıtıcı bir literatür taraması verilmiştir. Bölüm 4'te dijit oyununu oynayacak bir zeki etmen tasarlanmış ve *Q-öğrenme* algoritması ile eğitilme çalışması sunulmuştur. Bölüm 5 *Q-öğrenme* ile yapılan öğretim sırasında ortaya çıkan yerel optimum problemini aşmak amacı ile bir cezalandırma fonksiyonu içeren yeni bir *Q-öğrenme (Q-I)* tanıtılmıştır. Bölüm 6 ise *Q-öğrenme*'nin yavaşlığı problemi ele alınmış ve paralel güncelleştirme yaklaşımı ile yeni bir *Q-öğrenme (Q-II)* algoritması önerilmiştir. Bölüm 7 sunulan her iki yeni algoritma için bir yapısal kredilendirme sistemini, Bölüm 8 ise önerilen bu yapısal kredilendirme sistemini iyileştirmek amacı ile yeni bir *Q-öğrenme* algoritmasını *(Q-III)* sunmaktadır. Bölüm 9'da Endüstri Mühendisliği'nin önemli problemlerinden olan atölyede iş çizelgeleme ele alınmıştır. Dinamik bir atölye ortamında iş çizelgelemesini tezgahlara iş atayarak yapacak olan bir zeki etmenin tasarlanması ve *Q-III* algoritması ile durumlara göre iş atamasının da hangi öncelik kuralını kullanacağını öğrenmesi ele alınmıştır. Bölüm 10'da da tezin sonuçları verilmiştir.

BÖLÜM 2 ZEKİ ETMENLER

Canlıların doğal ortamdaki yaşamları, duyularla dış dünyayı algılama ve buna karşılık en uygun davranışla tepki gösterme çevrimi olarak özetlenebilir. Dış dünyayı algılama, duyu organları ile tepki gösterme eylemleri de el ve ayaklar gibi bir çok organlarla yürütülür. Canlılardaki bu etkinlikler gibi, bir zeki sistemin gerek sanal ve gerekse gerçek bir ortamda özerk çalışabilmesi bugün yazılım teknolojisi ile başarılabilmektedir. Bu tür özerk çalışabilen yazılımlara Zeki Etmen (ZE) adı verilir. Bir zeki etmenin çevresi ile olan ilişkileri Şekil-2.1'de sunulmuştur.



Şekil-2.1; Bir zeki etmenin çevresi ile etkileşimi

Zeki etmen (intelligent agent) kavramı daha çok 1980'li yılların ikinci yarısından itibaren kullanılmaya başlanmıştır. Başlangıçta farklı anlamlarda kullanılan bu kavramın zamanla hem yapı hem de karakteristik özellikleri üzerinde ortak bir anlayış oluşmuştur [Castillo, (1991), Shoham, (1993)]. Zeki etmenler için yapılan her bir tanımlamada kuşkusuz bir takım özelliklerinden yola çıkılmıştır. İlk

zamanlarda dar kapsamlı tanımların yanında, bir uzman sistem veya bir yapay zeka aracı (tool) da bir zeki etmen olarak algılanmıştır [Hayes-Roth, (1990), Cengeloglu ve arkadaşları, (1994)].

Literatürde yapılan tanımlardan bazıları şunlardır:

- "Zeki etmenler, gerçek zaman şartlarında içinde bulunduğu dinamik ortamla etkileşimli olarak çalışan ve hareket eden yapılardır" [Hayes-Roth, 1990].
- "Zeki etmenler yapacakları işler için muhakeme yapmaya izin veren ve değişen çevre şartlarına davranışlarıyla dinamik adaptasyon sağlayan yapılardır" [Castillo, 1991].
- "Bir zeki etmen; bir şahıs gibi, bir kamyon gibi ya da bir kavramsal varlık gibi sahip olduğu hareketi başlatabilen bir aktif varlıktır" [Becket ve Bedler, 1993].
- "Zeki etmenler, girdi ve çıktı için ihtiyaç duydukları bilgilerle ilgili kararlar veren yapılardır" [Szczerbicki, 1993].
- "Bir zeki etmen; kanaat getirme, kaabiliyetli olma, seçim yapabilme ve taahhüt etme gibi mantıksal birimlerden oluşan ve de mantıksal faaliyet yapan bir varlıktır" [Shoham, 1993].
- "Zeki etmenler, sahip oldukları bilgi ve enformasyon tabanı ile hareket eden nesnelerdir" [Cengeloglu arkadaşları, 1994].
- "Zeki etmenler, içinde bulunduğu çevre ile sürekli etkileşime devam ederken iç ve dış isteklere bağlı olarak sahip oldukları hedefleri başarmak üzere tasarlanmış komple sistemlerdir" [Beer, 1995].

Yapılan tanımların yoğunlaştığı ana karakteristikler;

- sahip olunan *bilgiyi kullanabilme* yetkinliği,
- içerisinde bulunulan çevreye *adaptasyon*,
- yürütülen işlev için gerekirse *muhakeme yapabilme* ve *karar verebilme*

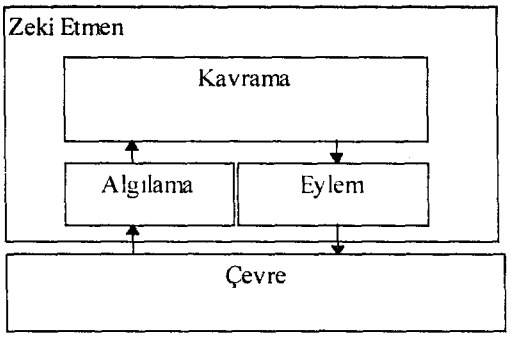
şeklinde ifade edilebilir. Yapısal olarak zeki etmenler canlıların psikolojik ve biyolojik yapılarından esinlenilerek geliştirilmişlerdir [Beer ve Arkadaşları, (1990), Becket ve Bedler, (1993), Steels, (1994)]. Buradan yola çıkılarak bir tanım yapılacak olursa: "Zeki etmenler; canlıların psikolojik ve biyolojik yapılarından esinlenilerek

bir takım görev ve hedefleri başarmak üzere tasarlanan, içinde çalıştığı çevre ile dinamik bir etkileşim halinde olabilen ve yaptığı işlem için gerekli muhakeme ve karar verme yetisi ile özerk davranabilen sistemlere denir".

2.1. Bir Zeki Etmenin Genel Yapısı

Zeki etmenlerin canlıların bir takım psikolojik ve/veya biyolojik yapılarından yola çıkılarak geliştirilen yapılar olması dolayısıyla tasarlanmalarında istenen işlevlerinin yerine getirilebilmesi için canlıların taklit edilen özelliklerine benzerlik göstermesi hedeflenmektedir [Anderson ve Donath, 1990]. Bu hedeflenen benzerliklerin başında zeki etmenlerin özerk olarak eylem gerçekleştirebilmeleri gelir. Özerk bir zeki etmen çevresi ile her hangi bir dış yönlendirmeye gerek duymadan *algılama* ve *eylem gösterme* şeklinde etkileşebilmektedir. Etkileşim yoluyla yapılan eylemler ile çevreye rahatlıkla cevap verilebilmektedir.

Bir zeki etmen Şekil-2.2'de görülebileceği gibi üç temel yapısal alt birimden oluşur. Çevre ile etkileşimi, mesajların algılanabilmesini sağlayan algılama ve mesaja uygun cevabı verebilmeyi sağlayan eylem birimleri sağlar. Alınan mesaja göre en uygun eylemi tespit etmede verilecek kararı da kavrama birimi belirler. Bir zeki etmende bir eylem gerçekleştirme çevrimi şu şekilde gerçekleşir; çevresinden mesajları algılama birimi ile alır, öz bilgisine dayanarak gerekli davranış için karar verir ve bir eylem üretmek üzere içinde bulunduğu çevrede ilgili davranışını uygular. Böylece verilen bir görev veya hedefi başarmak için uygun bir adım atmış olur.



Şekil-2.2; Genel olarak bir zeki etmenin yapısal şeması

Diğer yapay zeka ürünlerinden farklı olarak zeki etmenler, bir dış müdahale veya komuta ile değil tamamen öz bilgisi ve her zaman aktif olması ile işlevlerini yürütürler. Anlamlı bir algılamayı gerçekleştirdikleri an gerekli tepkiyi gösterirler [Becket ve Bedler,1993]. Bu özellik bir zeki etmene diğer yazılım sistemlerinden farklı olarak bir kişilik sağlar.

2.2. Zeki Etmenlerin Özellikleri

Zeki etmenlerin özelliklerini yapısal ve yeteneksel olmak üzere iki kategoride incelemekte yarar vardır. Yapısal özellikler derken sistemin mimari özellikleri kastedilmektedir. Yeteneksel özelliklerden de sistemin edinebildiği yetenekleri anlaşılmaktadır.

2.2.1. Yapısal özellikler

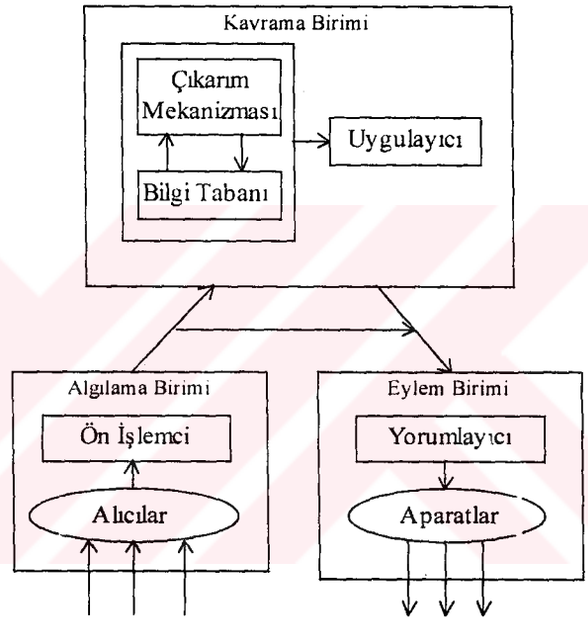
Bir zeki etmenin mimari yapısı, yukarıda da belirtildiği gibi işlevlerin yürütülmesi sırasında hangi davranışın ne zaman ve nasıl gösterileceği, bilgilerin nasıl temsil edileceği, davranışların hangi sıra ve kurala göre yürütüleceği, sistem parçalarının ne şekilde entegre olacağı sorularını yanıtlar.[Kaelbling ve Rosenchein (1990), Maes, (1991)]. Bundan dolayı mimari yapılar bir takım yaklaşım tarzlarına göre yapılandırılırlar. Söz konusu yaklaşımlar temel olarak geleneksel ve davranış eğilimli yaklaşımlardır [Steels, (1994), Simmons, (1994), Wooldridge ve Jennings, (1995)].

2.2.1.1. Geleneksel yaklaşım

Geleneksel yaklaşımda zeki etmenlerin tasarlanması daha çok insan bilgi işleme mekanizması ve algılama psikolojisi baz alınarak gerçekleştirilmiştir. Bu bakımdan söz konusu yaklaşımla tasarlanan sistemlerde bulunan temel birimler *Algılama, Kavrama* ve *Eylem* birimleridir. Bu birimler özellikle ilk dönem zeki etmenlerde belirgin olarak mevcuttur. Allen Newel ve Herbert Simon'ın ilk olarak bu tarzda bir yaklaşım göstermişlerdir. Nitekim ilk yapay zeka çalışmalarını başlatan Amerikalı dört ekolden birinin lideri olan Allen Newel bir psikologdur. Bu ekol daha sonraları

GPS (Genel problem çözücü) ve SOAR sistemlerini geliştirmişlerdir [Brooks,1991a]. Gerek Newel ekolünün geliştirdiği sistemler ve gerekse diğer ilk dönem zeki etmenlerin yapılarında (mesela GUARDIAN) asıl yoğunlaşılan ana birim kavrama birimi olmuştur. Diğer iki birime sistemin performansını etkileyecek çok fazla bir rol verilmemiştir.

Geleneksel yaklaşımla geliştirilen örnek bir zeki etmenin mimari yapısı Şekil-2.3'de verilmiştir. Bu şekil [Hayes-Roth , 1990]'dan uyarlanmıştır.



Şekil-2.3; Klasik yaklaşımla geliştirilen bir zeki etmenin genel yapısı

Algılama birimi, tasarlanan her zeki etmenin çevresinden bilgi almasını sağlar. Bu birimde (sistemler itibariyle değişiklik göstermesine rağmen) değişmeyen alt birimler alıcılar ve ön işlemci birimleridir. Çevreden gelen uyarılar alıcılar aracılığı ile sisteme girerler. Alıcılar sadece fiziki cihazlar olduklarından belirli özellikteki uyarıcılara duyarlı olsalar bile aldıkları her bilgi sistem açısından anlamlı olmayabilir. Uyarıları filtre eden ön işlemci de bilgileri ayrıştırma görevini üstlenir. Şekil-2.3'de verilen mimari yapı Hayes-Roth'un geliştirdiği GUARDIAN adlı zeki etmene aittir. Bu sistemin detayları [Hayes-Roth , 1990]'de bulunabilir.

Kavrama birimi geleneksel yaklaşımda zeki etmenlerin en önemli ana birimidir. Bu birime algılama biriminden gelen bilgiler bilgi tabanı ve çıkarım mekanizması kullanılarak gösterilecek davranışa karar verilir Burada dikkati çeken sembolik bilgi işlemedir. Bu, geleneksel zeki etmenlerin önemli bir özelliği olduğundan bilgi tabanı ve çıkarım mekanizması kavrama biriminin en temel alt elemanları sayılabilirler [Becket ve Bedler, 1993]. Zeki etmenlerin hedef, plan ve davranışları gibi öz bilgiler de bilgi tabanında kural, plan, şema veya kayıtlar halinde gösterilirler. Karar mekanizması işlevi gören bilgi tabanı ve çıkarım mekanizması sistemi, verilen kararla ilgili davranış veya davranış setini belirler ve uygulanmak üzere eylem birimine mesaj gönderir.

Eylem birimi de verilen kararları uygulamak üzere işlev üstlenmektedir. Motor birimi olarak da isimlendirilen bu birimde davranışları harekete geçirecek ve kontrol edecek olan bir uygulayıcı ve ilgili davranışların prosedürleri bulunur [Becket ve Bedler (1993), Rosenbloom ve Arkadaşları (1991)]. Zeki etmen algılama bölümü vasıtasıyla çevreden aldığı bilgileri kavrama birimi vasıtasıyla işler ve çevreye verecek tepkiyi de eylem birimi aracılığı ile gösterir.

Geleneksel yaklaşımla geliştirilmiş mimari yapılardan bazıları şunlardır. STRIPS ilk geliştirilen mimari yapı ile isim yapmış bir sistemdir [Wooldridge ve Jennings, 1995]. Sheykey adlı robotta kullanılan zeki etmenin mimarisi olan bu sistem, bir geleneksel arama tekniği olan Means-Ends analizini kullanır. İkinci bir ünlü mimari de Allen Newel araştırma grubunun geliştirdiği SOAR mimarisidir [Rosenbloom ve Arkadaşları, 1991]. Bu mimari yapının en önemli özelliği yukarıdan aşağıya işlem hiyerarşisi ile problemlere çözüm getirmesidir. SOAR, zeki etmen geliştirmede kullanım açısından yaygın bir ilgi görmüştür. IRMA geliştirilen bir üçüncü klasik mimari yapıdır [Pollack, 1992]. IRMA'da Means-Ends analizi ve klasik mantıksal çıkarsama sistemi kullanılarak sınırlı kaynaklarla daha esnek bir davranış sergilemeye çalışılmıştır. GUARDIAN tıbbi teşhis için tasarlanmış bir zeki etmendir [Hayes-Roth, 1990]. Bunun tasarlanmasında klasik yaklaşıma uygunluğu açık olarak görülmekle birlikte sistemin gerçek zamanlı çalışabilmesi ona hem bilgi tabanlı hem de davranış tabanlı olma özelliği vermektedir.

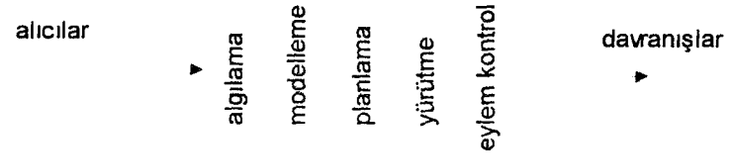
2.2.1.2. Davranış tabanlı yaklaşım

80'lerin ikinci yarısından itibaren ortaya çıkan bu ikinci yaklaşım davranış psikolojisinden esinlenerek ortaya atılmış ve bu anlayışı zeki etmen tasarımına taşımıştır. Bu yüzden *davranış tabanlı yaklaşım* adını da almaktadır. Mümkün olduğunca bilgi-tabanlı sistemlerden faydalanmama düşüncesini taşıyan bu yaklaşımın temsilcileri bilgi-tabanlı sistemlerin bazı dezavantajlarından kurtulmayı amaçlamışlardır. Yaklaşımların farklılığı her iki yaklaşımın zeki etmen tasarımında ön gördüğü ana birimleri de etkilemiştir [Brooks,1986]. Özellikle Brooks (1986) ve Agre ve Chapman (1987), bu anlayışın ilk temsilcileridir. Bu iki çalışmadan sonra özellikle robotik etmenlerde araştırıcıların yoğun dikkatini çeken bu yaklaşım, öğrenme kabiliyeti gibi bir çok yeni yeteneğin zeki etmenlere kazandırılmasını sağlamıştır.

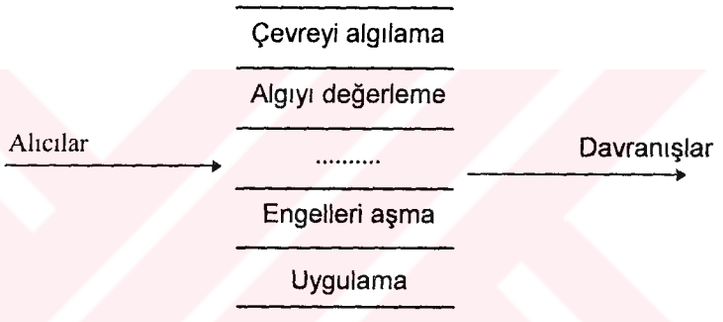
Davranış tabanlı yaklaşım ile bilgi-tabanlı sistemlerin hantallığı, rijitliği ve yavaşlığı gibi dezavantajlarının önlenmesi başarılmıştır. Bu yaklaşımın en önemli gerekçesi geleneksel sistemlerle geliştirilen zeki etmenlerin gerçek dünyadaki uygulanırlıklarının olmayışı veya çok zor oluşudur. Brooks (1991a)'de bu dezavantajı şöyle dile getirmektedir: *"Geleneksel veya bilgi tabanlı sistemlerde zeki etmenler daha çok birtakım teorilerin (problem çözücüler, sembolik konular) test edilmesi amacıyla geliştirilmişlerdir. Bundan dolayı yapılan sembolik muhakemede semboller daha çok sistemi geliştiren kişiyi bilgilendiriyor veya anlamlılığını onun hafızasında buluyordu. Böylece gerçek dünyada uygulanırlıkları olmuyordu. Halbuki bir geliştirilen zeki etmen kendini gerçek dünyada bulmalı ve burada kendi maharetini göstermelidir"*. Bu gerekçe zeki etmenlerin teorik yapıdan kurtulup mutlaka gerçek dünyada uygulanabilmeleri için yapılacak çalışmaların başlangıcı olmuştur.

Davranış-tabanlı yaklaşımla tasarlanan zeki etmenlerin genel yapısı ile geleneksel yaklaşım ürünü zeki etmenlerin genel yapısı arasındaki önemli farkı Brooks (1986) Şekil-2.4 ve Şekil-2.5'deki gibi göstermiştir. Bilgi tabanlı yaklaşımı da uyarılar, çevreden alınınca algılama, modelleme, planlama, işlem yürütme ve eylem

bölümlerinden seri olarak geçtikten sonra üretilirler (bkz Şekil-2.4). Bu zorunlu işlem alınan her anlamlı uyarı sonunda mutlaka işler. Bunun yanında Şekil-2.5'de görülebileceği gibi davranış bazlı yaklaşımda eylemler davranışlar olarak sergilenir ve sistemin yapılanması tamamen paralel bir özellik gösterir.



Şekil-2.4; Geleneksel yaklaşımla geliştirilen zeki etmenlerin genel yapısı



Şekil-2.5; Davranışa dayalı zeki etmenlerin genel yapısı

Algılama eylemi, davranış-tabanlı yaklaşımda bir davranış gibi ele alınmaktadır. Bir zeki etmen geliştirilirken geleneksel anlayışta olduğu gibi (algılama-kavrama-eylem gibi) bağımsız birimlere ayrılmadan algılama işlemi müstakil bir davranış gibi tasarlanır. Sistemin işleyişinde çevreden gelen uyarılar doğrudan ilgili modüllere gitmektedir. Böylece davranışlar uyarılara tepki şeklinde gelişmektedir [Mahadevan ve Connell, (1992)].

Davranış bazlı yaklaşımda zeki etmen daha çok algıladığı bilgi veya uyarıyı hemen eylem birimlerine aktarma ve böylece gerçek dünyaya uygunluk sağlama şeklinde tasarlanır. Bu bakımdan algılama-eylem birimlerinin ilişkileri daha çok önemsenir. Bu çerçevede bazı araştırmacılar zeki etmeni sadece algılama ve faaliyet bölümlerinden oluşturma yoluna gitmişlerdir. Mesela Kaelbling ve Rosenchein

(1990)'nin yaptıkları tasarımda algılama bölümü girdilerin filtre edilmesi işlevini üstlenmiştir.

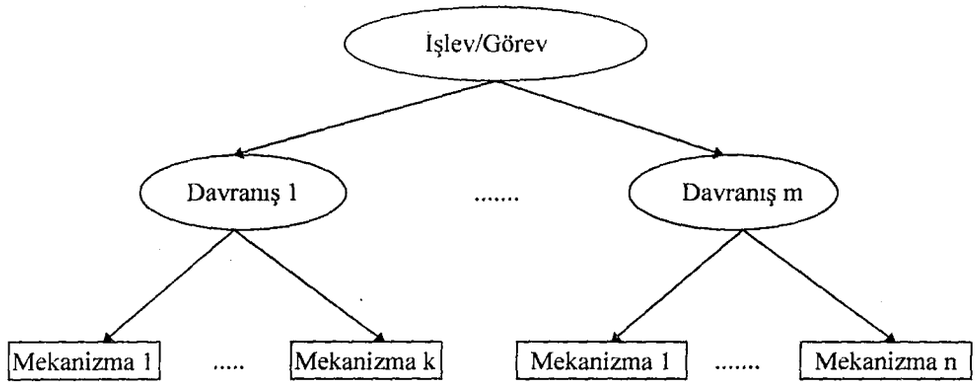
Yukarıda anlatılanlara rağmen davranış tabanlı zeki etmen için esasen çok da oturmuş bir yapının olduğu söylenemez. Her bir araştırmanın kendine özgü bir mimarisi olabilmektedir.

Davranışlar davranış tabanlı yaklaşımın en önemli özelliği ve teorisinin dayanağıdır. Sisteme gelen uyarıların zeki etmen açısından anlamlı olması halinde çevreye bir tepkide bulunulmaktadır. Bir zeki etmen belirlenen işlevi veya görevi yerine getirebilmek için davranışlara ihtiyaç duyar. Davranışlar tasarlanırken birer sistem olarak düşünülürler. Her bir davranış bir modül olarak oluşturulur. Her bir modül bağımsız olduğu gibi bir tek davranışa özgüdür. Modüller arasında bir yatay sıralanma veya hiyerarşik bir yapılanma olmadığından sisteme daima yeni modüller ekleyerek davranış bakımından zenginleştirmek son derece kolaylaşır.

Davranışlar oldukça basit veya oldukça karmaşık olabilirler. *Basit davranışlar* algılama ve eylem birimleri arasındaki reflekslerle gerçekleşirler. *Karmaşık davranışların* gerçekleşmesinde ise muhakeme gibi ek bir takım davranışlara ihtiyaç duyulabilmektedir [Steels, 1994]. Bazı durumlarda karmaşık davranışlarda hiyerarşik bir yapılanma ihtiyacı olabilmektedir [Anderson ve Donath, 1990]. Örneğin bir kutu itekleme görevi için üç davranışa gerek duyulmaktadır: Kutuyu bulma, bulunan kutuyu itekleme ve biraz daha iteklemenin mümkün olup olmadığını kontrol etmek. Yani önce iteklenecek bir kutunun diğer cisimlerden fark edilip bulunması, sonra bu kutuyu itekleme ve sonra da itekleme bitince ilgili kontrolü yapmak söz konusu görevi yerine getirmek için bir gereksinimdir [Mahadevan ve Connell, 1992].

Davranışlar durumlara uygun olarak sergilenerek işlevleri ve hedefleri başarmayı sağlarlar. Yani esasen davranışlar işlevlerin alt birimleridir. Bir davranış da mekanizmalara ihtiyaç duyar. Burada işlev-davranış-mekanizma kavramları arasında bir ilişki söz konusudur. Bu ilişki Şekil-2.6'de sunulmuştur. Ayrıca davranışların

mümkün olduğunca basit ve bir hiyerarşi gerektirmeyecek şekilde tasarlanması büyük önem arz etmektedir. Bu zeki etmenin hızlılığı açısından da önemlidir.



Şekil-2.6; Zeki etmende işlev-davranış-mekanizma ilişkisi

Davranış tabanlı yaklaşımın en önde gelen mimari yapısı Brooks (1986, 1991b, 1991c) tarafından geliştirilen SUBSUMPTION mimarisidir. Bu mimariye göre bir zeki etmenin tüm eylemleri davranışlara indirgenerek ve paralel olarak tasarlanmaktadır. Davranış tabanlı yaklaşımın öncüsü olan bu mimari ile zeki etmen geliştirmek için bir de özel dil (Davranış Dili) geliştirilmiştir [Brooks, 1990b]. Diğer bir mimari yapı ise Maes (1994a) tarafından geliştirilmiş olan ANA mimarisidir. Bu mimarinin en önemli özelliği STRIPS mimarisine benzer nitelik göstermesidir. Ayrıca öğrenmeye izin verebilmektedir. Ferguson (1995) ise hem davranış tabanlı yaklaşıma hem de geleneksel yaklaşıma benzeyen bir mimari ortaya koymuştur. Bu mimarinin özelliği her iki yaklaşımı sentezlemesidir.

2.2.2. Yeteneksel özellikler

Bir zeki etmenin mimari özelliklerinin yanında başarısını doğrudan etkileyen yeteneksel özelliklerinden (karakteristiklerinden) de bahsetmek gerekir. Zeki etmenlerin kişilik bulmasına bu karakteristikler sebep olmaktadır. Verilen bir görevin başarılabilmesi için gerekli olan davranışların seçilmesi ve her bir davranış için ilgili mekanizmaların harekete geçirilebilmesi gerekir. Zeki etmenlerin karakteristiklerinin belirlenmesi de elde etme yolu da mimari yapıların özelliklerine bağlıdır. Zaten

mimari yapılardaki değişikliğin sebebi de bundandır. Yani zeki etmenlerin mimari yapılara göre değişmeyen karakteristikleri ve değişebilen karakteristikleri vardır. Değişmeyenler zeki etmenin tanımı gereği sahip olduğu özelliklerdir. Değişebilenler ise mimari yapılara bağlı karakteristiklerdir. Genel olarak zeki etmenlerde bulunan temel karakteristikleri sıralamak gerekirse:

1. Özerklik (Autonomy)
2. Adaptasyon kaabiliyeti (Adaptivity)
3. İşlev yürütme (Functionality)
4. Tutarlılık (Coherency)

Özerklik karakteristiği, sistemin kendi ortamında tamamen kendi başına hareket edebilmesi özelliğidir. Zeki etmen, içinde bulunduğu çevrede, algıladığı uyarılara göre kendi başına tepki gösterir ve hiç bir yerden destek almadan hareket eder. Çevreye verilen tepkinin zamanını, gerekli enerjiyi (robotik etmenler için söz konusudur) ve yeterli titizliği kendisi belirler [Liljenström, 1995]. Yani zeki etmenin çevreden alınan bir uyarıya en uygun tepkiyi zaman ve enerji açısından en az maliyetle gösterebilme özelliği vardır. Bir mobil robot veya yazılım etmeni bir tepkinin zamanı, enerjisi ve gerekli titizliği konusundaki yetkinliğinin yanında çıkan bir hatanın araştırılması ve giderilmesinde de söz sahibidir [Simmons, 1994].

Özerklik karakteristiğinin bir zeki etmene kazandırılması diğer karakteristikler gibi doğal olarak çeşitli mekanizmalarla gerçekleştirilmektedir. Zeki etmenler özerkleştirirlirken çoğunlukla öğrenme metotları kullanılmıştır. Chi ve Zeigler (1994) *hiyerarşik model tabanlı teşhis metodu* adını verdikleri bir yöntemle, Dorigo ve Colombetti (1994) genetik algoritma tabanlı bir öğrenme sistemi ile, Simmons (1994) da yapılandırılmış kontrol yaklaşımı ile özerklik karakteristiğini geliştirmeye çalışmışlardır. Ayrıca Liljenstörm (1995) karmaşık ve dinamik bir ortamda çalışabilecek zeki etmenlerin yapay sinir ağları yardımıyla özerkleştirilmelerinden bahsederken Arkin (1990) daha özgün bir mimari ile özerklik karakteristiğini tesis etmeye çalışmıştır.

Adaptasyon karakteristiği bir zeki etmenin daha önce karşılaşmadığı yeni durumlarla karşılaşması halinde anlamlı ve mantıklı davranış üretebilmesi anlamına gelir. Canlılar nasıl rahatlıkla çevrelerine uyum sağlayabiliyorlarsa bir zeki etmenin de çevresiyle uyumlu çalışabilmesi gerekir. Bunun için, Beer (1995) yaptığı bir çalışmada çarpıcı karşılaştırmalar yapmakta ve kavramsal analizi genişletmektedir. Daha çok öğrenme teknikleri ile geliştirilen zeki etmenler yeni durumlara rahatlıkla adapte olabilmektedirler. Bu öğrenme teknikleri de daha çok yapay sinir ağları, genetik algoritmalar ve diğer destekli öğrenme teknikleridir. Brooker ve Arkadaşları (1989) özellikle genetik algoritmalara dayanan bilgi tabanlı sınıflandırma sistemleri üzerinde yoğunlaşarak adaptasyonu sağlamaya çalışmışlardır. Bunun yanında Hayes-Roth (1995) daha karmaşık bir yapı kurarak adaptasyonu algılama stratejisi, kontrol stratejisi, muhakeme metodu ve üstlenen görev açısından bir takım analizler yaparak GUARDIAN adlı sistemine adaptasyon karakteristiğini kazandırmıştır.

Her zeki etmenin *işlevsellik* karakteristiğine sahip olması gerekir. Hiç bir zeki etmen amaçsız, görevsiz ve de işlevsiz tasarlanmaz. Zeki etmenler görevleri doğrultusunda işlev yürütürler. Birden fazla görevi olan zeki etmenler, aktif olan görevi ne ise o doğrultuda davranış sergiler. Bu açıdan, Steels (1994) bir zeki etmende her bir görevin bir işlev anlamına geldiğini belirtir.

Tutarlılık karakteristiği, zeki etmenin sergilediği davranışlarında bir uyum içinde bulunması olarak tanımlanabilir. Benzer şartlarda benzer davranışları gösterebilmesi, zeki etmenin tutarlılığını gösterir.

Bir zeki etmen bu sıralanan karakteristiklerin yanında, mimari yapıdan kaynaklanan değişikliklere bağlı olarak da bazı karakteristiklere sahip olabilir.

2.2.2.1. Geleneksel yaklaşım

Geleneksel yaklaşımla geliştirilen zeki etmenlerin en başta gelen özelliği *merkezi yapılanmadır*. Yazılımlar bütünüyle modüler olarak tasarlansalar da sonuçta genel koordinasyon, merkezi bir birim tarafından yapılmaktadır. Bu da merkezi bir görev

dağılımı, ve merkezi bir yürütme anlamına gelir. İkinci önemli karakteristik ise daha önce mimari özellikler açıklanırken belirtilen *sembolik çıkarsama* karakteristiğidir. Bu şartlarda karşılaşılan dış dünya problemleri için bir davranış gösterme ancak bilgi tabanlı bir sistemin bilgiyi sembolik olarak alabilmesi ve bu bilgi ile mantıksal bir çıkarım yapması sonucunda mümkün olabilmektedir [Wooldridge ve Jennings, 1995]. Bu sistemlerin üçüncü önemli bir karakteristiği de *yukarıdan aşağıya* bir işlem seyrine sahip olmalarıdır. Sistemin bilgi tabanında sabit hedefler ve bu hedefleri başarmak için sabit uygulama planları mevcuttur. Sistem çevreden aldığı uyarıları anlamlı bulunca ilgili sabit hedefleri ve yine ilgili planları uygulamaya geçirir. Çevre karşısında bu özelliklerle bir zeki etmenin esneklik göstermesi oldukça zor olur.

2.2.2.2. Davranış tabanlı yaklaşım

Geleneksel anlayışa alternatif olarak geliştirilen bu yaklaşım sistemlerin mimari yapısında olduğu gibi karakteristik tasarımlarında da ön plana çıkmıştır. Davranışların baz alınmasıyla zeki etmenlerin daha nitelikli, daha esnek ve dinamik çevreye duyarlı, adapte olabilir bir yapıya kavuşturulmaları hedeflenmiştir. Ayrdedici özellik olarak söylenebilecek ilk şey, bu yaklaşımın görevleri *dağıtılmış bir yapı* ile yürütmeyi önermesidir. Modüler olarak geliştirilen davranış programları merkezi bir koordinasyona ihtiyaç duymadan mevcut şartlara uygun tepkiyi göstermektedirler. Koordinasyon işlevi öğrenme tekniği veya öncelik kuralları ile gerçekleştirilmektedir [Maes ve Brooks, 1990]. Bununla beraber belirtilmesi gereken bir diğer özellik ise *tepkisellik* özelliğidir. Tepkisellik özelliği ile zeki etmenlerin dinamik bir çevrede gerek gerçek zamanlı olarak meydana gelen değişimi fark edip anında gerekli davranışı gösterebilmeleri ve gerekse daha önce hiç karşılaşmadıkları ortam şartlarına adapte olabilmeleri sağlanır. Bir diğer önemli özellik ise sistemde işlem seyrinin *aşağıdan yukarıya* doğru gerçekleşmesidir. Bu özellik diğerleri ile beraber zeki etmenlere adaptasyon, esneklik ve daha yüksek bir performans sağlamaktadır.

2.3. Zeki Etmen Türleri

Zeki etmenleri yapısal ve işlevsel olmak üzere iki açıdan sınıflandırmak mümkündür. Yapısal olarak zeki etmenlerin tasarlanma amacına göre basitten karmaşığa doğru çeşitli yapılarda geliştirilebildikleri görülmektedir. İşlevsel olarak da zeki etmenlerin yine amaçlarına göre değişik türlerinden bahsetmek mümkündür.

2.3.1. Yapısal olarak zeki etmen türleri

Zeki etmenler işlev, ortam, mimari yapı felsefesi gibi bir çok etkenden dolayı çok değişik yapısal özellikler arz edebildikleri gibi yapısal türlere de ayrılabilirler. Yapısal özellikler, zeki etmenlerin yetenekleri üzerinde belirleyici olduklarından sınıflandırmada da önemli rol oynarlar. Bu çerçevede, zeki etmenlerin temel yeteneklerine ek olarak yeni yetenek ve özellikler kazandırmak yeni sınıfların oluşmasına de neden olacaktır. Yapısal sınıflandırmaya göre zeki etmenler üç sınıfta ele alınabilir:

1. Basit-refleks etmenleri ,
2. Hedef yönlendirmeli etmenler,
3. Tatmin esaslı etmenler.

Basit-refleks etmenleri, canlılardaki ani olaylara verilen reflekslere benzer şekilde algılanan her duruma hemen tepki vererek davranış sergileme esasına dayanırlar. Bu reaktif davranış tipi, zeki etmenlerin daha basit ve bir kaç sabit *algılama-davranma* kuralı ile yapılandırılması ile sağlanabilmektedir [Russell ve Norvig, 1995]. Bu tür zeki etmenler daha çok çevrenin uyarılarına tepki verme tarzında davranmaktadırlar. Zeki etmenlerin cevaplamaya çalıştıkları üç temel soru olan "Ne? Ne zaman? ve Nasıl yapılmalı?" sorularına en basit şekilde cevap vermeye çalışmaktadırlar. Bu açıdan bir *algılama-davranma* kuralı genel olarak şu formda tasarlanmaktadır:

IF *uyarı(j)* **THEN** *davranış(i)*

Burada çevreden gelen uyarılar ile davranışlar arasında doğrudan bir geçiş söz konusudur.

Hedef yönlendirmeli zeki etmenler tamamen hedeflere yönelik tasarlanan daha karmaşık sistemlerdir. Bir zeki etmenin bir veya birden fazla hedefi olabilir. Bu açıdan davranışların hedeflerle ilişkileri, hedeflerin algılama ile olan ilgisi zeki etmenlerin yapılandırılmasında önemlidir. Bu etmenler çoğunlukla sahip oldukları hedefleri başarabilecek şekilde davranışlarının organize edilmesini gerektirirler. Şayet yukarıdaki genel kural ele alınırsa, bunun ne kadar doğrudan ilişkili olduğu anlaşılacaktır. Hedef yönlendirmeli zeki etmenlerde algılama ile davranışlar arasında bu denli doğrudan bir geçiş söz konusu olamaz. Burada hedeflerin başarılması için bir takım mekanizmaları harekete geçirerek son kararı vermek gerekir. Bazan hedefler karmaşık bir davranış zinciri gerektirebileceklerinden hedefleri daha alt hedeflere ayrıştırmak söz konusu olabilir. Bu ayrıştırma sayesinde son hedefi başarmak daha kolaylaşır [Rosenbloom ve Arkadaşları (1991), Mahadevan ve Connell (1992)].

Tatmin esaslı etmenler ise bir takım hedeflere sahip olmanın yanında ek nitelikler taşıyan etmenlerdir. Bir davranış, hedefin başarılması doğrultusunda sergilenmeye karar verilince daha tatmin edici bir etki ortaya çıkarabilmek için hedef yönlendirmeli etmenlerden biraz daha farklı olarak ek birtakım mekanizmaları da harekete geçirmek gerekebilmektedir. Böylece daha zengin ve etkin bir çevre ile etkileşim sağlanır [Russell ve Norvig, 1995] .

2.3.2. İşlevsel olarak zeki etmen türleri

Zeki etmenler üstlendikleri görevlere göre de sınıflandırılmaktadırlar. Bu görevler daha çok uygulama alanlarına göre ön plana çıkan fonksiyonlardır. Bir zeki etmenle, bu açıdan iki ana uygulama alanında görülmektedir. Birincisi robotların kontrol ve kumandasında kullanılmak üzere geliştirilmiş etmenler, ikincisi ise her hangi bir sanal ortamda bir yazılım olarak kullanıldıkları yazılım etmenleridir. Burada zeki etmenler içinde bulunduğu ortama göre kategorize edilirler. Ortam fiziki bir dünya ise robotik etmen, yazılım ortamı ise yazılım etmeni adını alır.

2.3.2.1. Robotik etmenler

Robotik etmenler, robotların kontrol, kumanda ve yönlendirilmesini gerçekleştirmek üzere tasarlanıp geliştirilen etmenlerdir. Klasik yazılım yöntemleri ile robotların kontrol edilmeleri üstlenilen işlevler karmaşıklaştıkça zorlaşmaktadır. Aynı şekilde, bir robotu uzaktan kumanda ile yönlendirmek de beraberinde önemli zorluklar getirmektedir. Özellikle mobil robotların özerkleşme ihtiyaçları ve bununla birlikte ortaya çıkan karar verme zorunluluğu zeki robotik etmenlerin ortaya çıkmasına yol açmıştır. İlk geliştirilen zeki robot **Sheykey** adlı bir mobil robottur. Bu robotun kontrol ve kumandası için geliştirilen sistem STRIPS, aynı zamanda geliştirilen ilk zeki etmendir [Wooldridge ve Jennings, 1995].

Robotik etmenler daha çok MIT robotik laboratuvarındaki AI grubu tarafından tartışılmış ve çeşitli uygulamalarla gelişmeler devam etmiştir. Özellikle Brooks bir dizi mobil robotu, robotik etmenler yardımı ile değişik özellikler için geliştirmeyi başarmıştır [Brooks, 1990a]. Bunların en çarpıcı olanı GENGHIS adlı altı ayaklı böcek tipli robottur. Diğer mobil robotlardan bazıları ise TOM ve JERRY, HERBERT, ALLEN ve SEYMUR olarak sayılabilir. Bütün bu robotlar tasarlanırken Brooks tarafından geliştirilen SUBSUMPTION mimarisi kullanılmıştır. Karar verme sürecinde ise *çoğaltılmış sonlu durum makinaları* adlı sistemden yararlanılmıştır [Brooks, 1990b]. Bunların dışında, Dorigo ve Colombetti (1994) AUTONOMOUSE II ve AUTONOMOUSE IV adlı mobil robotları genetik algoritma tabanlı bir öğrenme algoritmasıyla eğiterek çevre ile etkileşimlerini başarabilmişlerdir. OBELIX, Mahadevan ve Connell (1992) tarafından geliştirilen ve *Q öğrenme* algoritması ile öğretilerek kontrol edilen bir mobil robottur. Ferguson (1995)'de TOURINGMACHINES adlı sanal robotik etmenle seyreden bir araba tasarlamıştır. Benzer şekilde, bir çok sanal ortamda çalışan robotik etmen benzetimleri de literatürde yerlerini almışlardır [Örnekler için bkz. Pigott ve Sattar (1994), Castillo (1991), Agre ve Chapman (1987)]. Robotik etmenlerin sanal ortamlardaki uygulamaları daha çok yeni gelişmekte olan *yapay yaşam* disiplinine konu olmaktadır. Tyrrell (1993) tezinde bu çerçevede davranış seçimi üzerinde yapılan çalışmaları analiz ederek bir yaklaşımda bulunmuştur. Lenger ve arkadaşları

(1994) ise zeki etmen kullanarak insansız seyreden bir arabanın kumandası üzerinde çalışmıştır. Bu çalışmada DAMN adlı bir mimari kullanılmıştır. Zapata ve arkadaşları (1994) da bir mobil robotu hızlı hareket edebilmesi için reaktif davranış gösterebilen bir zeki etmenle kontrol etmişlerdir.

2.3.2.2. Yazılım etmenleri

Zeki etmenler daha çok sanal ortamlarda geliştirildiklerinden uygulamada yazılım etmenlerine daha fazla rastlanılabilmektedir. Yazılım etmenleri, bilgisayar sistemlerinde gerek işletim sisteminin işleyişini rahatlatmak ve gerekse daha özgün bir görevi yürütmek üzere geliştirilen zeki etmenlerdir [Gams ve Hribovsek, 1996]. Daha önce belirtildiği gibi, bir zeki etmen sanal veya benzetim ortamında bir yazılım etmeni olabilmektedir.

Yazılım etmenleri robotik etmenlerde olduğu gibi MIT AI grubu ve sonraları da MIT Media grubu tarafından daha çok gündeme getirilmiştir. Bu gruplar bir çok sistemi geliştirmeyi de başarmışlardır. Söylenebilecek ilk zeki yazılım etmeni Agre ve Chapman (1987) tarafından geliştirilen PENGI adlı oyundur. Bu ürün daha çok davranış bazlı yaklaşımla geliştirilmiştir. Daha sonra Maes'in öncülük ettiği çalışmalar farklı farklı ortamlar için bir çok ürün ortaya koymuştur. INTERNET ortamında kullanıcılara yardımcı olan bir dizi zeki etmen yine aynı grup tarafından geliştirilmiştir. Maxims gelen mesajları silme, ön eleme ve dizme işlemleri yaparken, görüşme çizelgeleme etmeni görüşme saatlerini düzenlemektedir. Bir başka etmen de haber filtre etme işlemi yapmaktadır. Bunun dışında bir dizi animasyon etmenleri de geliştirilmiştir. Bunların en önemlilerinden biri ALIVE'dır. [Lashkari ve arkadaşları (1994), Maes (1994b), (1995), Shardanand ve Maes (1995)]. Gams ve Hribovsek, (1996) VAX sistemini kullanıcı-makina diyalogunu daha hızlı ve az hafıza kullanarak gerçekleştiren bir arayüz etmeni geliştirmişlerdir. Yine Toomey ve Mark (1995) yazılım etmenleri yardımı ile uydudan görüntü nakletmeyi başarmışlardır.

BÖLÜM 3 ZEKİ ETMENLERDE ÖĞRENME

Öğrenme bir sistemin çevresi ile etkileşiminde davranışlarını nasıl geliştireceğini eldeki örneklerden yola çıkarak belirlemesi olarak tanımlanabilir. Bu tanımdan yola çıkarak bir sistemin öğrenme eylemi gerçekleştirebilmesi için temel olarak bir örnekler setine sahip olması gerekir. Sistemlerin öğrenme sürecinde metodolojik olarak bir takım farklılıklar gözlenebilir. Bu farklar öğrenme stratejileri ve metotlarından kaynaklanırlar. Bir çok çeşit öğrenme metodunun yanında öğrenme stratejileri genel olarak *öğretmenli öğrenme, destekli öğrenme ve öğretmensiz öğrenme* gibi üç kategoride incelenmektedir [Bresslof ve Weir, 1991].

Öğretmenli öğrenme stratejisinde, öğretilmek istenen sisteme bir öğretmen tarafından sistem girdileri ve bunlara karşılık yapılması gereken davranış birlikte sunulur. Öğrenecek sistem, girdi ve olması gereken çıktıları birlikte değerlendirerek en uygun davranışı belirlemeye çalışır. Bu sürecin sonucunda etmen, girdilere en uygun çıktıyı verecek hale gelir. Destekli öğrenme stratejisinde ise çevre şartlarına tamamen yabancı olunduğu, bu yüzden deneme-yanılma yöntemi ile girdilere uygun davranış üretmenin söz konusu olduğu görülür. Bunun yanında, öğrenme süreci boyunca sistemin davranışlarını belirlemesini gözleyen ve uygulanan davranışın uygun olup olmadığını söyleyerek sistemi destekleyen bir destekleme mekanizması vardır. Öğrenme sürecinde destekleme mekanizması bu uyarıları ile sistemi, destekleyerek uygun davranış seçebilmesine yardımcı olur. Benzer şekilde, öğretmensiz öğrenme etkileşilen çevre hakkında hiçbir net bilginin elde bulunmadığı ve bir destekleme mekanizması veya öğretmenin de yönlendirmesinin söz konusu olmadığı durumlarda

uygulanan öğrenme türüdür. Burada sistem yalnızca kendi iç dinamiklerini kullanarak öğrenmeye çalışır [Bresslof ve Weir, 1991].
Zeki etmenlerin öğrenmesi konusunda yapılmış bazı çalışmalar şunladır:

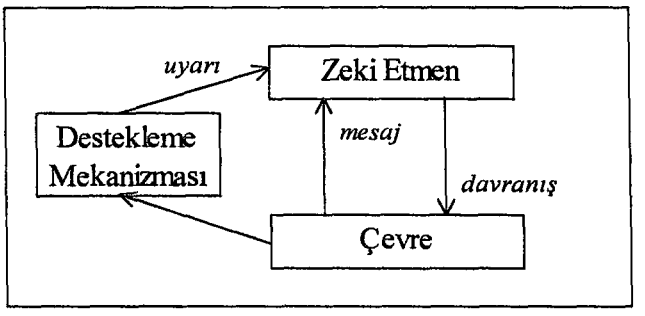
Basye ve arkadaşları (1995) zeki etmenlerin öğretilmesinde dinamik sistemin hızlı değişimlerine karşı algılamadaki değişimleri önemseyecek iki algoritma geliştirmişlerdir. Bu algoritmalardan biri deterministik diğeri de stokastik şartlarda çalışmaktadır.

Maes (1994a) ise kullanıcının davranışlarından öğrenme gerçekleştirerek ortamda bulunan diğer zeki etmenlere bilgi aktaran arayüz etmenlerin bu tarz öğrenmelerini sağlayan bir öğrenme sistemi sunmuştur.

Larid ve arkadaşları (1990) da SOAR mimarisini kullanarak zeki bir etmenin çevresi ile etkileşiminde planlama ve öğrenme kabiliyetlerini birleştiren bir sistem önermişlerdir.

3.1. Destekli Öğrenme

Destekli öğrenme stratejisi ile geliştirilen öğrenme algoritmaları genel olarak gerçek zamanlı öğrenme gerçekleştiren sistemlerde uygulanmaktadırlar. Özellikle *durum-davranış* ilişkisini ortaya koyabilecek kesin *girdi/çıktı* bilgilerine ulaşmanın zor olduğu zamanlarda kullanımı önem kazanmaktadır [Kaelbling, 1996]. Zeki etmenler de gerçek zamanlı olarak çevreleri ile etkileşerek öğrenme gerçekleştirdiklerinden daha çok destekli öğrenme algoritmaları yardımı ile öğrenme yeteneklerini geliştirmektedirler [Brooks, 1991a].



Şekil-3.1; Destekli öğrenme stratejisi ile öğrenen bir zeki etmenin çevresi ile etkileşimi

Destekli öğrenme stratejisinde Şekil-3.1'de görülebileceği gibi, öğrenme sırasında zeki etmen-çevre etkileşimini izleyen bir destekleme mekanizması zeki etmenin ürettiği davranışın çevre şartlarına uygun olup olmadığını değerlendirerek etmene bir uyarı mesajı gönderir. Bu uyarı mesajı ödül veya ceza olabilir. Sutton (1992) destekli öğrenmeyi söz konusu uyarı mesajından gelen ödülü enbüyükleyecek olan durum-davranış eşleştirmesi olarak tarif etmektedir. Zeki etmen ödülünü enbüyükleyerek hatalarını deneme-yanılma yöntemi ile düzeltir ve sonunda algılanan çevre şartlarına en uygun davranışı üretebilir duruma gelir.

Öğrenme süreci boyunca zeki etmen her bir davranışını bir yararlanma fonksiyonu ile değerlendirerek durumlar karşısında alınan toplam puanını nihai olarak maksimum yapmaya çalışmaktadır. Bu çaba zeki etmenin eğilimini oluşturur. Alınan puanların kümülatif olarak toplamını şu formülasyon verir:

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$$

Burada R_t, t anındaki kümülatif puanı, r_t; t anında alınan puanı ve γ; da *0* ile *1* arasında değer alan bir sabit değeri gösterir. Destekleme mekanizmasının ürettiği puan genel olarak -1,1 veya 0 olabilir. Puanlar her adımda γ ile çarpılarak ölçeğe oturtulurlar [Lin, 1992]. Kümülatif puanın enbüyük olabilmesi için yararlanma fonksiyonu değeri en büyük olan davranış seçilir ve uygulanır. Alınan puana göre her iterasyon sonunda yararlanma fonksiyonu güncelleştirilir. Yararlanma fonksiyonunu şu şekilde gösterilir:-

$$U(S, A) = r + \gamma E(S')$$

Burada U(S,A); S durumunda A davranışının yararlanma değerini, r; alınan puanı, E(S'); S durumuna A davranışı uygulandıktan sonra elde edilen yeni durum S' için değerleme değerini ifade eder. Değerleme fonksiyonu olarak isimlendirilen E(S);

$$E(S') = max(U(S', A))$$

şeklinde tanımlanır. Bu ise S' durumunda elde edilmesi beklenen en yüksek yararlanma değeridir. Genel bir destekli öğrenme algoritması Şekil-3.2'de adımlar halinde sunulmuştur. Bu yapı, genel olarak tüm destekli öğrenme algoritmalarında bulunur [Mahadevan ve Connell, 1992].

1. Tüm U(S,A) değerlerine bir başlangıç değer ata.
2. Şu adımları öğrenme gerçekleşinceye dek tekrarla
   - S durumunu gözle,
   - A davranışını seç,
   - A davranışını uygula, S' durumunu ve r puanını elde et,
   - U(S,A) değerini bir kurala göre güncelleştir

$$U_{t+1}(S, A) = U_t(S, A) + (r + E(S'))$$

Şekil-3.2; Genel olarak bir destekli öğrenme algoritması

Destekli öğrenme stratejisi ile bir çok algoritma geliştirilmiştir. Maes ve Brooks (1990) öğrenme süreci boyunca verileri istatistiksel olarak analiz korelasyon katsayısı hesaplayan ve buna göre durumlarla davranışlar arasında anlamlı ilişkinin olup olmadığını belirleyen bir algoritma önermektedirler. Algoritmayı altı ayaklı bir mobil robotun yürümeyi öğrenmesinde kullanmışlardır.

Williams (1992) stokastik yapay sinir ağlarında kullanılmak üzere gradyan metotlarına dayalı bir algoritma (REINFORCE) geliştirmiştir. Bu algoritmanın en önemli özelliği, diğer gradyan metoduna dayalı öğrenme algoritmaları ile birleşik olarak kullanılabilmesidir. En önemli dezavantajı ise, yerel optimuma takılabilme olasılığını potansiyel olarak önleyememesidir.

Heiss (1994) olası bir hafıza problemini gidermek için doğrusal düzgünleştirmeye dayalı bir algoritma önerirken Lin ve Lee (1994) de bir bulanık yapay sinir ağında kullanılmak üzere bir destekli öğrenme prosedürünü geliştirmişlerdir. Benzer şekilde, Levinson (1996) genel olarak oyun oynama eyleminde karşılaşılan zorlukları gidermek için MorphII isimli aramaya dayalı bir destekli öğrenme algoritması sunmaktadır.

Kaelbling (1994a, 1994b) birleşik destekli öğrenme diye tanımladığı *k*-DNF algoritmasını yapay sinir ağlarına ve klasik istatistiksel metotlara alternatif olarak geliştirmiş ve genelleştirme ve öğrenme performansının diğerlerinden daha iyi olduğunu belirtmiştir. Bu algoritmada *k*-DNF fonksiyonunu normal bir algoritmaya ekleyerek birleşik bir yapı elde etmekte ve ilgili algoritmayı ilk halinden daha yüksek performanslı hale dönüştürmektedir.

3.1.1. Genetik algoritmalara dayalı çalışmalar

Yukarıda sıralanan destekli öğrenme algoritmaları arasında yapısal olarak destekli öğrenme stratejisinden başka ortak özellikler bulunmamaktadır. Bunların yanı sıra ortak özelliği olan öğrenme algoritmaları da mevcuttur. Bunlar gerek kullanılan teknikler ve gerekse uygulama yoğunluğu açısından bir kaç kategoride incelenebilirler. Bunlardan genetik algoritmalara dayalı çalışmalar kullanılan teknik açısından bir kategori oluşturabilir. Bunun yanında yoğun ilgi gören *geçici farklar* ve *Q öğrenme* teknikleri de yapılan çalışmalarda iki farklı kategori şeklinde algılanabilirler.

Bu alanda, Dorigo (1993) genetik algoritmalara dayanan bir destekli öğrenme algoritması ile davranış bazlı robotikte bazı uygulamalar sunmuştur. Dorigo (1995)'te ise çalışmalarını geliştirerek ALECSYS adıyla bir öğrenme sistemi tanımlamış ve AutonoMouse isimli mobil robotun çevresi ile etkileşimini öğretmede kullanmıştır. Dorigo ve Colombetti (1994) de ALECSYS sistemini AutonoMouse'un iki yeni tipinde uyguladıklarını rapor etmişlerdir. Bu sistemde, çevreden alınan mesajlara bilinen davranışlarından birisi eşleştirilerek uygulandıktan sonra destekleme

mekanizması ilgili davranışın neticesine göre bir puan üreterek zeki etmeni uyarır. Eğer olumlu bir puan üretildi ise eşleştirme kural haline getirilip kural tabanına atılır. Aksi halde yeni bir sınıflandırıcı sistem üretilerek ilgili çevre mesajı ile eşleştirilir. Sistem tipik bir destekli öğrenme süreci halinde işlemeye devam eder.

Moriarty ve Miikkulainen (1996) genetik algoritmalara dayalı SANE adlı diğer bir destekli öğrenme metodu geliştirmişlerdir. Burada, genetik algoritmalar yardımı ile yapay sinir hücresi popülasyonu oluşturularak sinir ağı tasarlayan metot, AHC ve *Q öğrenme* algoritmalar ile karşılaştırmalı olarak sunulmuştur. Hız açısından önerilen sistemin performansının daha iyi olduğu belirtilmiştir.

Spronck ve Kerckhoffs (1997) genetik algoritmaları kullanarak destekli bir yapay sinir ağı denetleyicisi tasarlamışlardır. Aynı şekilde Uthman (1997) çok etmenli bir benzetim çevresinde zeki etmenleri genetik algoritmaya dayalı, öğretmenli öğrenen bir algoritmadan söz etmektedir. Whitley ve arkadaşları (1993) da genetik algoritmalar yardımıyla geliştirdikleri destekli öğrenme algoritmasını daha sonra üzerinde durulacak olan geçici farklar metodu (AHC algoritması) ile karşılaştırmalı olarak yapay sinir ağlarına dayalı bir pendlum kontrolünde kullanmış ve performans ölçümü yapmışlardır.

3.1.2. Geçici farklara dayalı algoritmalar

Geçici farklar yöntemi ilk defa Sutton (1988) tarafından geliştirilen ve öğretmenli öğrenmeye alternatif olarak sunulan bir algoritmadır. AHC (Adaptive Critic Heuristic) adıyla geliştirilen algoritma daha sonra TD(λ) algoritması haline dönüştürülmüştür. Bu algoritmaların geliştirilmesinin amacı her zaman eşleştirilmiş girdi-çıktı bilgi setleri elde olamadığından hareketle davranışların doğru-yanlış kritiğine göre değerlendirilip hataları azaltmaktır. Buna daha çok gerçek zamanlı çalışan sistemlerde gereksinilmektedir. Bu algoritmaların çoğu yapay sinir ağlarında kullanılmak üzere geliştirilmişlerdir [Tesauro, 1992]. Geçici farklar yöntemi asıl olarak dinamik programlamanın bir tür uygulamasından oluşturulmuştur [Kealbling,

1996]. Bu konuda Russell ve Norvig (1995) ve Lin ve Lee (1996) detaylı bilgiler sunmaktadırlar.

Geçici farklar denince iki algoritma anlaşılmaktadır: AHC veya TD(λ), *Q öğrenme*. *Q öğrenme* algoritması daha sonra ele alınacağından bu bölümde ilk algoritma konusunda yapılan çalışmalar sunulacaktır.

Tesauro (1992), TD(λ)'nın karmaşık gerçek zamanlı problemlerin çözümünde yeterli olup olmadığı üzerinde durmuştur. Özellikle Backgammon adlı oyunun oynanmasında ne derece başarılı olup olmadığını incelemiştir. Sonuçta algoritmanın büyük ölçekli karmaşık sistemlerde yeterli yakınsayamama ve yerel optimum gibi önemli problemlerinin olduğunu belirtmiştir. Yakınsama problemi için Dayan (1992) TD(λ) yaklaşımına *Q öğrenme*deki teoremleri uygulayarak çözüm aramıştır. Bu teorik çalışmanın devamı ise Dayan ve Sejnowski (1994) tarafından yapılmıştır.

Leipins ve arkadaşları (1991), sınıflandırıcı sistemlerde kredilendirme ve keşif yapmak için daha önce kullanılan genetik algoritmalara alternatif olarak AHC metodunu önermişlerdir. Burada keşif yapmak için kural geliştirme sürecinde destekli öğrenmeden yararlanmışlardır.

Bradtke ve Barto (1996), TD(λ) algoritmasının bazı dezavantajlarını gidermek amacıyla doğrusal en küçük kareler metoduna dayanan iki algoritma önermişlerdir. Bu algoritmaların probleme TD(λ)'den daha iyi yakınsadıklarını ifade etmektedirler.

Schapire ve Warmuth (1996) ise TD(λ) algoritmasının en kötü durumları üzerinde durmuş ve gelecekte alınması beklenen puanları değerlendirerek üst ve alt sınırlarını araştırmıştır. Böylece gelecekte alınması beklenen puanlar konusunda daha sağlıklı bir tahmin olanağı yakalanmıştır.

Singh ve Sutton (1996) ise TD(λ) algoritması için strateji belirlemede önemli bir mekanizma olan uygun iz belirlemeye yeni bir metotla yaklaşmışlarıdır. Dağ-araba

problemi açısından değerlendirme yaparak TD(λ)'nin yönlendirilmesini de tartışmışlardır.

3.2. *Q-Öğrenme* Algoritması

Q-öğrenme algoritması geçici farklar yaklaşımı ile geliştirilen ve en çok kullanılan iki algoritmadan birisidir. *Q öğrenme* algoritması dinamik programlama ve Markov karar verme süreçlerine dayalı olarak geliştirilen bir tür destekli öğrenme algoritmasıdır. Watkins (1989) tarafından geliştirilen bu algoritma ile zeki etmenlerin gerçek zamanlı olarak dinamik bir çevrede öğrenme eylemini gerçekleştirebildikleri gösterilmiştir [Watkins ve Dayan, 1992]. Markoviyen olayların en önemli özelliği olayların birbirlerinden bağımsız ve bir sonraki olayın bir önceki olaya dayanıyor olmasıdır. Markoviyen olmayan olaylara da algoritma kendi karakterini yansıtmaktadır.

Bu algoritmanın temeli, basit bir veri yapısına dayanan bir değerlendirme fonksiyonuna ($Q(x,a)$) bağlıdır. Burada, x; etmen tarafından gözlenen durumu, a; ise bu durum karşısında etmenin gerçekleştirmesi gereken davranışı göstermektedir. Zeki etmen, çevre şartlarına (durumlarına) göre uyguladığı davranışlara karşılık, destekleme mekanizmasından gelen uyarı mesajlarını dikkate alarak $Q(x,a)$ fonksiyonunda güncelleştirme yapar. Bu işlemi tekrar tekrar yaparak adım adım durum-davranış eşleştirmesinde iyileştirme amaçlanmaktadır. Arzu edilen öğrenme düzeyine ulaşılınca bu işlem durdurulur. Bu güncelleme işlemi, (x,a,r,y) dörtlüsüne bağlı olarak gerçekleşir. Burada r çevreden gelen uyarı mesajını (ki; buna *puan* denir), y ise x durumuna a davranışı uygulandığında elde edilecek yeni durumu gösterir. $Q(x,a)$ değeri şöyle hesaplanmaktadır.

$$Q(x,a)= E(r + \gamma e(y)|x,a)$$

Burada, $E(.)$ x durumuna a davranışı ile müdahale etmenin beklenen değerini gösterirken, $e(y)$; y durumunun en büyük değerleme fonksiyonu değerini (yani Q değerini), γ ise 0 ile 1 arasında sabit bir değeri gösterir. $e(y)$ şu şekilde hesaplanır.

$$e(y)= max\ Q(y,a),\ \forall a\ için$$

Çevreden gelen uyarıya göre değerlendirme fonksiyonu değeri de şu şekilde güncelleştirilir.

$$Q_y(x,a) \leftarrow Q_e(x,a) + \beta(r + \gamma e(y) - Q_e(x,a))$$

Burada, $Q_y(x,a)$ x durumunda a davranışını gerçekleştirmenin yeni, $Q_e(x,a)$ ise eski Q değerlerini, β da 0 ile 1 arasında bir değer alan öğrenme katsayısını göstermektedir.

1. Her x ve a için $Q_y(x,a)$ değerlerine başlangıç olarak 0 ata.
2. Şu adımları öğrenme süreci bitinceye kadar tekrarla:
   - x durumunu gözle, $x \in S$,
   - a davranışını seç, $a \in A$,
   - a davranışını uygula y yeni durumunu ve r puanını al,
   - $Q(x,a)$ değerini aşağıdaki güncelleme kuralına göre güncelle,

$$Q_y(x,a) \leftarrow Q_e(x,a) + \beta(r + \gamma e(y) - Q_e(x,a))$$

Şekil-3.3; *Q öğrenme* algoritması

Şekil 3.3'te *Q-öğrenme* algoritmasının çalışma prensibi verilmektedir. Bu algoritma kullanılarak bir çok çalışma yapılmıştır. Bu çalışmalar bu bölümde tanıtılacaktır. Özellikle yapay sinir ağlarının elde yeterli bilgi olmaması durumunda ve gerçek zamanlı sistemlerin eğitilmesi söz konusu olduğunda bu algoritma kullanılmıştır.

Q algoritması bazı dezavantajlara da sahiptir. En önemli dezavantajı, yerel optimuma takılmak olarak söylenebilir [Whitehead and Lin,1995]. Yerel optimum problemi, bir davranışa ait Q değerinin bir kaç iterasyondan sonra yüksek bir değere ulaşarak diğer alternatif davranışların seçilmesini önlemesi olarak açıklanabilir. Alternatif davranışların seçilebilmesi için gerekli şansı yakalayabilmeleri ya hiç gerçekleşememekte ya da uzun süre sonra mümkün olabilmektedir. Bu olay, öğrenmenin zorlaşmasına (uzun süre almasına) veya hiç mümkün olmamasına neden olmaktadır.

Genel olarak geçici farklar yönteminin, özel olarak da *Q-öğrenme* algoritmasının yukarıda bahsedilen dezavantajlardan doğan iki temel çözülmesi gereken problemi

vardır. Bunlar *geçici kredilendirme* ve *yapısal kredilendirme* problemleridir [Lin (1992), Mahadevan ve Connell (1992)].

Geçici kredilendirme, zeki etmenin yapmış olduğu bir tecrübe denemesinde o an almış olduğu puanın bir sonraki durum için belirlenecek davranışın seçiminde etkili hale getirilmesi, yani bir önceki denemede almış olduğu kredi ile bir sonraki denemeyi gerçekleştirmesi olarak tanımlanabilir. Bir üçüncü denemede de, ilk denemenin kredisi öğrenme katsayısı dolayısıyla etkisiz hale getirilmektedir. Burada geçici bir tecrübe söz konusudur. Bu geçici deneyim ile öğrenme süreci boyunca zeki etmen durumlar karşısında repertuarındaki davranışların tümünü deneme olanağı bulmaktadır. Geçici farklar yönteminde, özellikle geçici deneyim kazandırma ve etkin bir durum-davranış eşleştirilmesi araması tartışılan bir problemdir. Bu çalışmada geçici kredilendirme konusunda Bölüm 5 ve Bölüm 6'da iki yeni yaklaşım önerilmiştir.

Yapısal kredilendirme problemi ise, bir deneme manevrasında elde edilen tecrübenin daha kalıcı bir deneyim olabilmesini sağlamaktır. Bu konuda yapılan çalışmaların çoğu yapay sinir ağları yardımı ile bir kalıcı tecrübenin sağlanmasına dayanır. Yapısal kredilendirme problemi de *Q-öğrenme* çalışmalarının önemli bir konusudur. Bu çalışmada yapısal kredilendirme konusunda Bölüm 7 ve Bölüm 8'de iki yeni yaklaşım önerilmiştir.

Yukarıdaki problemler ve benzerlerinin çözülmesi için değişik çalışmalar yapılmıştır. Bunlardan bazıları şöyle özetlenebilir.

Lin (1992), *Q-öğrenme* ve AHC algoritmalarını kullanarak öğrenme, öğretme ve deneyim edinerek cevaplama eylemlerini gerçekleştiren farklı etmenler oluşturmuştur. Bu etmenleri daha sonra algoritmaların performansını ölçmek için karşılaştırmıştır. Bu çalışmada *Q öğrenme* kullanımında destekli öğrenmeye yardımcı olacak bir öğretmen mekanizmasından söz edilmektedir. Bu mekanizma *Q-öğrenme*'nin yerel optimum problemine çözüm getirmek üzere geliştirilmiştir.

Mahadevan ve Connell (1992), *Q-öğrenme* algoritmasından yararlanarak bir robotik etmen geliştirmişlerdir. Yaptıkları çalışmada robotik etmene hangi davranışı ne zaman yapması gerektiğinin öğretilmesinde kullandıkları *Q-öğrenme*'nin özellikle yapısal kredilendirme problemi üzerinde durmuşlardır. Yapısal kredilendirme için öbekleme analizine dayalı bir yapı önermişlerdir.

Singh (1992), *Q-öğrenme* algoritmasını ardışık görevlerin zeki etmen tarafından yapılmasını sağlayacak bir uygulama için kullanmıştır. Daha önceki bir çok uygulamada zeki etmenin tek bir görevi üzerinde durularak, davranışların öğretilmesinde *Q-öğrenme* kullanılmışken bu uygulamada daha karmaşık bir görev yapısı ele alınmış ve bunların ardışıklığının sağlanmasına çalışılmıştır. Karar verme sistemi *Q-öğrenme*'nin içinde bulunduğu bir birleşik sistemden oluşmaktadır. Sistemde kullanılan *Q-öğrenme* yapısına birleşimsel *Q-öğrenme* (Compositional *Q* learning) adı verilmiştir.

Koeing ve Simmons (1993) hedefe ulaşmak için en kısa yolu bulma problemini ele almışlardır. Sistemin karmaşıklığı analiz edildikten sonra, en kötü durum karmaşıklığının sınırlarının daraltılmasına yönelik çalışma yapılmıştır. Bunun için *iki yönlü Q-öğrenme* (Bi-directional *Q*) adlı bir öğrenme algoritması geliştirerek gerçek zamanlı ortamda en kısa yol problemine bir çözüm önermişlerdir.

Piggott ve Sattar (1994), daha önce *Lion* algoritması ile yapılan robot algılamasını *Q-öğrenme* algoritması ile yeniden yapmışlardır. Buradaki algılamanın özelliği iteratif bir süreç olmasıdır. İteratif süreçler içinde özellikle *Q-öğrenme*'nin başarılı sonuçlar verdiği belirtilmektedir.

Asada ve arkadaşları (1996) *Q-öğrenme* algoritması ile sanal bir ortamda bir robota futbol oynamasını öğretmişlerdir. Klasikleşmiş parçalara ayırma yöntemiyle yapılandırılan robotlara alternatif olarak *kolay misyondan öğrenme* (Learning Easy Mission LEM) ile robot yarılandırılmıştır. Bu uygulama gerçek robotlarla desteklenmiştir.

McDonald ve Hingston (1997), hedef yönlendirmeli sistemlerin performansına yaklaşım hatasının etkilerini analiz etmişlerdir. Bu tür etkilerin potansiyel olarak bir takım ölçeklendirme zorluklarına sebep olduğunu ortaya koymuşlardır. Sonuçta ölçeklendirmeye gerek duymadan öğrenmeyi gerçekleştirecek *Q-öğrenme*'nin yeni bir versiyonunu geliştirmişlerdir.

Tsitsiklis (1994) *Q-öğrenme*'nin eşzamansız stokastik yakınsamasını incelemiştir. Bu incelemede Watkins ve Dayan (1992) tarafından ortaya konan bazı teorem ve ispatlar daha genişletilerek ele alınmış ve daha geniş bir teorik hareket yelpazesi sağlanmıştır.

Mahadevan (1996) ortalama ödüllü destekli öğrenme kavramını analiz ederek *R-öğrenme* algoritması ile *Q-öğrenme* algoritmasını bu kavram açısından karşılaştırmıştır. Sonuçta *R-öğrenme*'nin *Q-öğrenme*'den daha duyarlı olduğunu belirtmiştir.

Heger (1996) *Q-öğrenme* de değer fonksiyonlarındaki değer kayıplarını minimax tabanlı ve beklenen değer tabanlı Markov karar verme süreçleri için incelemiştir. Özellikle *Q-öğrenme* algoritmasının bu şartlardaki yakınsaması ele alınmıştır. Sonuçta optimal politika belirleme sırasında kayıpları azaltmak için sınırlar belirlenmiştir.

Koeing ve Simmons (1996), hedef yönlendirmeli genişletilmiş görevlere *Q-öğrenme* ve *Q-hat-öğrenme* algoritmalarını uygulayarak on-line öğrenmeyi araştırmışlardır. Ortaya çıkan karmaşıklığı analiz ederek bilgi ve gösterimlerin öğrenme üzerindeki etkisini görmeye çalışmışlardır. Sonuçta; hangi tür destekli öğrenme problemlerinin zor, hangilerinin kolay olduğunu ortaya koymuşladır.

Maclin ve Shavlik (1996) ise Lin (1992)'de olduğu gibi *Q-öğrenme* algoritması ile öğretme yaparken yerel optimallik problemini önleyebilmek için dışarıdan bir sistemle düzeltmeler yaparak sistemi uyaran bir dış öğretmen tasarlamışlardır. Bunun için RATLE adlı bir benzetim ortamı geliştirmişlerdir. Bu sistemde öğrenme

sırasında zeki etmeni uyararak gerekli düzeltmeleri yapan bir öğretmen tasarlama olanağı hazırlanmıştır.

Peng ve Williams (1996) de *Q-öğrenme* ile TD(λ) algoritmalarını her ikisinin avantajlarından yararlanmak amacı ile birleştirmeye çalışarak $Q(\lambda)$ algoritmasını oluşturmuşlardır. Burada özellikle hız problemi ile Markoviyen olmayan problemler için daha iyi sonuçlar elde etmek amaçlanmıştır.

Öztemel ve Aydın (1997a, 1997b) ise *Q-öğrenme* algoritmasının yerel optimum probleminin çözümü için ceza fonksiyonu içeren yeni bir *Q-öğrenme* (*Q-I*) algoritması ve hem yerel optimum hem de hız problemini birlikte ele alan (paralel güncelleme kuralı) *Q-II* algoritmasını önermişlerdir.

BÖLÜM 4 DİJİT OYUNU PROBLEMİ

Dijit oyunu problemi; 8 adet dijitten oluşan bir setin aktif durumunu, hedef olarak belirlenen duruma iteratif olarak dönüştürme problemidir. Bu problem, karşılıklı olarak tuttukları rakamları deneme yanılma yolu ile tahmin edip isabet ettirmeye çalışan iki kişinin oyununa benzemektedir. Bilgisayar 8 adet dijiti rassal olarak belirledikten sonra, oyunu oynamak isteyen kişiye beş adet operasyondan birini yapma olanağı tanımakta ve bir adımda bir operasyonu gerçekleştirme yolu ile hedefe yaklaşmasını istemektedir. Bir adımda hedefin yakalanabileceği durumlar olabileceği gibi, üst üste manevra yaparak hedefi yakalamanın gerektiği durumlar da olabilmektedir. Burada önemli olan aktif durumu iyi analiz ederek en uygun operasyonu yapabilmektir. Bu bölümde adı geçen oyunu bilgisayarda tıpkı bir insan gibi oynamak üzere tasarlanan bir zeki etmenin detayları verilmektedir. Önce problem tanıtılacak, daha sonra bu oyunu oynamak üzere tasarlanan zeki etmenin eğitilmesi açıklanacaktır.

4.1 Problemin Tanımı ve Özellikleri

Dijit oyununda yukarıda anlatıldığı üzere, 8 dijitten oluşan bir bilgi setinin hedeflenen sete iteratif olarak dönüştürülmesi istenmektedir. Dijitlerin her biri *0* veya *1* değerlerinden birini almaktadır. Bir anda 8 dijitin mevcut değerlerinden oluşan sete *bir durum* adı verilmektedir. Her operasyon sonucunda durumlarda *0*'lar *1*'e, *1*'ler de *0*'a dönüştürülerek yeni durumlar oluşturulmaya çalışılmaktadır. Şekil-4.1'de örnek bir durum sunulmuştur. Bir operasyonla aktif durumun tümünü dönüştürmeye izin verilmeyerek iteratif dönüştürme imkanı sağlanmaktadır. Bu şekilde hedefin yakalanması, durumun yakınlığı dolayısıyla bir iterasyonda mümkün olabildiği gibi

onlarca iterasyon da gerektirebilmektedir. İteratif dönüştürme işlemi en fazla iki dijit üzerinde yapılabilmektedir. Söz konusu dijitler rassal olarak her iterasyonun başında belirlenerek oyuncuya bildirilmektedir. Belirlenen dijitlerin biri veya her ikisi değiştirilme olanağına sahip olduğu gibi hiç birinin değiştirilmemesi de söz konusu olabilmektedir. Burada adı geçen alternatif durumlar oyuncunun tercih edeceği bir dönüştürme kuralı sonucu ortaya çıkmaktadır. Kullanılan alternatif dönüştürme kuralları daha sonra tasarlanan zeki etmenin davranışları olarak açıklanacaktır. Verilen iki dijitten birinin değiştirilmesi halinde hangisinin tercih edileceği konusunda bir irade belirlenmesi için üçüncü bir rassal sayıya ihtiyaç vardır. Böylece her defasında toplam üç rassal sayı ve bir dijit seti oyuncuya verilerek en uygun operasyon kuralını seçmesi kendisinden beklenmektedir.

0	*1*	*0*	*0*	*1*	*1*	*0*	*1*

Şekil-4.1; Bir durum gösteren dijit seti örneği

4.2. Dijit Oyununu Oynayacak Zeki Etmenin Tasarımı

Dijit oyunu bilgisayarla etkileşimli olarak oynanabilecek bir oyundur. Normal şartlarda oyuncu, bir insan olup bilgisayarın sunduğu durumları algılayarak uygun davranışı seçmeye çalışacaktır. Bu çalışmada dijit oyununu tıpkı bir insan gibi oynamak üzere bir zeki etmen tasarlanmıştır. Burada bir durum, bir dijit seti ile değiştirilecek dijitleri belirleyen üç rassal sayıdan oluşmaktadır. Mevcut duruma *aktif durum*, hedeflenen duruma da *hedef durum* denilmektedir. Rassal sayıların ilk ikisi, daha önce ifade edildiği gibi dönüştürülmek üzere belirlenen iki dijitin sıra numaralarını, son sayı ise gerektiğinde kullanılmak üzere eşik değerini göstermektedir. Zeki etmen aktif durumu algıladıktan sonra, hedef duruma yaklaştıracak olan en uygun davranışı seçip uygulamaktadır. Bir adımda en çok iki dijiti değiştirmek mümkün olduğundan aktif durumu göz önünde bulundurarak karar vermek gerekir. Yalnız bazı durumlarda hiç değişiklik yapmamak ta söz konusu olabilir. Burada her bir operasyonu gerçekleştirebilmek yani dijitleri değiştirebilmek

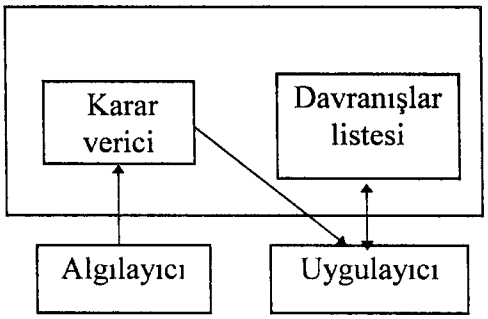
için yapılan her eylem, bir davranış olarak belirtilmiştir. Bu konuda muhtemel davranışlar seti de oluşturulabilir.

4.2.1. Zeki etmenin mimari yapısı

Yukarıda anlatılan oyunu oynayabilecek yeteneklere sahip bir zeki etmeni tasarlamak için öncelikle mimari yapının belirlenmesi ile başlamak gerekir. Mimari yapılar Bölüm 2'de anlatıldığı gibi iki tür yaklaşımla oluşturulurlar. Bilgi tabanlı yaklaşım ile tasarlanacak olan bir zeki etmen, daha çok bilgilenme ve planlama gibi geniş bilgi gerektiren problemler için önerilebilir. Davranış tabanlı yaklaşımda ise, daha çok reaktif davranış gösterme özelliği öne çıkar. Ani davranış gösterme (refleks) ihtiyacını, bu yaklaşım daha iyi karşılar [Ferguson (1995), Wooldridge ve Jennings (1995)]. Üzerinde durulan problemde çok dinamik bir ortam söz konusu olduğundan durumu algılayıp hemen davranış gösterme ihtiyacı davranış tabanlı modelleri kullanmayı uygun yapmaktadır. Bölüm 2'de anlatıldığı üzere davranış tabanlı modellerin daha bir çok avantajları da bu tür modelleri seçmeye sebep olmaktadır.

Davranış tabanlı zeki etmen modelleri ani, (reflektif) davranış gösterme özelliklerinden dolayı çok basit yapılı (algılama-davranma) veya hedef yönelimli olabilmektedir. Burada söz konusu oyunun doğası gereği zeki etmenin hedef yönelimli olmasına ihtiyaç vardır.

Bu çalışmada tasarlanan zeki etmen, genel olarak Şekil-4.2'deki gibi yarılandırılmıştır. Çevre ile iletişimini algılama ve uygulama birimleri ile kuran zeki etmen, algıladığı duruma göre karar verici biriminde uygulanacak davranışı belirler ve kararını uygulayıcıya bildirir. Uygulayıcı da söz konusu davranışı davranışlar listesinden seçerek uygular. Burada davranışlar, birbirine tamamen paralel ve bağımsızdır. Bu bakımdan yeni bir takım davranışların eklenmesi gerektiğinde, rahatlıkla eklenebilmesi kolaylığı getirilmiştir. Bu açıdan zeki etmen davranış bazlı etmenlerin özelliğini göstermektedir. Ancak algılama ve uygulama modüllerinin zeki etmen gövdesine seri bağlanması da geleneksel yaklaşımın özelliğini gösterir.



Şekil-4.2 ; Zeki etmen genel yapısı

Tasarlanan zeki etmen yapısında *algılama birimi* çok basit bir şekilde oluşturulmuştur. Ortam bilgileri sadece bir dijit seti ve üç rassal sayıdan ibaret olduğundan, bu bilgilerin ortamdan algılanması gerekmektedir. Karmaşık bir algılama söz konusu olmadığından algılamada herhangi bir zorluk yoktur.

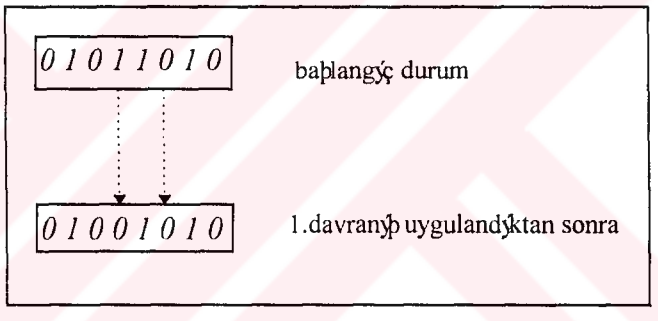
Oluşturulan *karar verici* ile davranışlar listesi birbiri ile doğrudan ilgilidir. Karar verici, öğrenme ile zaman içerisinde olgunlaştırıldığından başlangıçta içi tamamen boştur. Ancak başlangıçta mümkün olan davranışları bilir. Daha sonra öğrenme ile davranışlar ve algılanan mesajlar arasında bir bağlantı kurup durum/davranış eşleştirmesi yapar.

Bu şekilde *davranışlar listesi,* zeki etmenin sahip olabileceği muhtemel bütün davranışları prosedür halinde içerir. Bu bölümde anlatılan problem için tasarlanan zeki etmen, içinde bulunduğu ortamın şartlarına göre dijit oyununu oynayabilmesi için beş adet davranış ile donatılmıştır. Bu davranışlar, genel olarak iki dijit üzerinde yapılan operasyonları içerir. Oyunda kullanılan davranışlar şöyle belirlenmiştir:

1- Rassal olarak belirlenen iki dijiti aynılaştırma,
2- Rassal olarak belirlenen iki dijiti farklılaştırma,
3- Rassal olarak belirlenen dijitten birini değiştirme,
4- Rassal olarak belirlenen dijitlerin yerini değiştirme,
5- Hiç bir değişiklik yapmama.

Birinci davranış, rassal olarak seçilen iki dijit değerlerinin farklı olması halinde aynılaştırılması operasyonudur. Davranış iki dijit değerinin özellikle farklı olması halinde değişiklik yapabilmektedir. Dijit değeri aynı olduğu zaman bu davranış uygulanmaz.

Daha önceki bölümlerde ifade edildiği gibi bir dijit iki değerden birini (*0* veya *1*) alabilir. Dijit değerlerinin aynı olması halleri *{{0,0}* veya *{1,1}}* olabilirken, farklı olma halleri *{{1,0}* veya *{0,1}}* de olabilir. Bu davranışın uygulanabilmesinin ikinci bir ön şartı da, hangi dijitin hangisine eşitleneceği sorusunun cevabının belirlenmiş olmasıdır. Bu belirleme işlemi de çekilen üçüncü rassal sayıyla yapılmaktadır. Algılama birimi tarafından belirlenen dijit seti ve üç rassal sayıyı kullanarak birinci davranışın uygulanması Şekil-4.3'te gösterildiği gibidir.



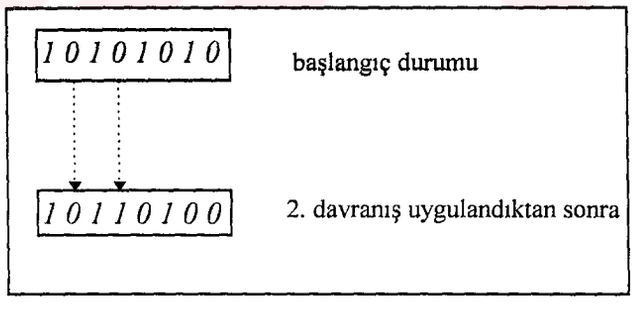
Şekil-4.3 ; Birinci davranışın örnek bir uygulanması.

Örneğin eldeki dijit seti *{0,1,0,1,1,0,1,0}*, rassal sayılarde 3,5,1, olsun. Dijit setindeki dijit numaraları 0'dan başlayıp 7'ye kadar devam ettiğinden, birinci ve ikinci rassal sayılar a=0 , b=7 olan üniform dağılıma uygun bir rassal sayı üretecinden tam sayı olarak elde edilmektedir. Üçüncü rassal sayı ise a=0, b=1 parametrelerine uygun üniform dağılım üretecinden tam sayı olarak elde edilmektedir. Çekilen rassal sayılara göre 3 birinci dijit numarası, 5 ikinci dijit numarası, 1 ise referans dijitin ikinci dijit olacağını göstermektedir. Üçüncü rassal sayı 0 olunca belirlenen dijitlerden ilkini, 1 olunca ikincisini temsil eder. Şekil-4.3'de başlangıç duruma birinci davranış uygulanınca 3 nolu (1.) dijit 5 nolu (2.) dijite eşitlenerek aynılaştırma yapıldığı gösterilmiştir. 3. dijitin 5. dijite eşitlenmesi yani 3. dijitin 1 iken 0 yapılması ve 5. dijitin sabit tutulması, çekilen üçüncü rassal sayının (referans

numarasının) 1 olmasından dolayıdır. Şayet bu değer 0 olsaydı 5. dijit 3. dijite eşitlenecekti.

İkinci davranış, birinci davranışın tersi bir uygulamadır. Yani bu davranış her iki dijitin aynı olmaları halinde farklılaştırma yapar. Birinci davranış gibi bu davranış ta her iki dijitten sadece birini değiştirmeye dayanır. Belirlenen iki dijitten hangisinin değiştirileceğine de benzer şekilde üçüncü rassal sayı kullanılarak karar verilir. Üçüncü rassal sayı 0 iken birinci, 1 iken ikinci dijit değiştirilir.

Örneğin dijit seti *{ 1,0,1,0,1,0,1,0},* rassal sayılar da 3,1,0 olsun. Burada 3. ve 1. dijitler işlem görecek, bunlardan 3. dijit referans dijit olacaktır. Yani 3. dijit 0 iken 1 yapılacak demektir. Bu durum Şekil-4.4'te gösterilmiştir. Birinci davranışın şartlarında rassal sayılar 3 ve 5 idi. Burada ise, 3 birinci sırada ve kendinden daha ön sırada olan 1 ise 2. sırada yer almaktadır. Burada önemli olan dijit numarasının küçüklüğü veya büyüklüğü değil, rassal sayının değeridir. Bu yüzden 3 daha sonraki konumda iken öne geçebilmiştir. Eğer rakamların büyüklüğü önemli olsaydı, 3. dijit yerine 1. dijitin değiştirilmesi gerekirdi. Bu davranış, belirlenen iki dijit değerinin faklı olması halinde durum üzerinde bir operasyon yapmamaktadır.

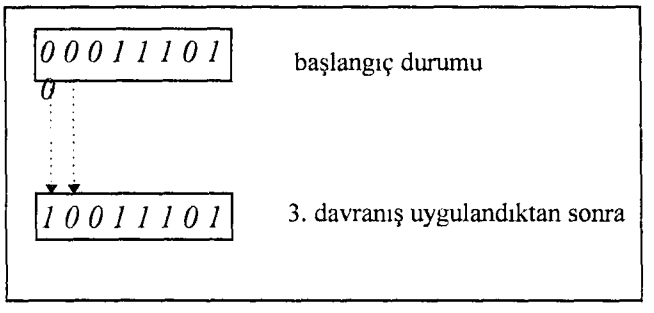


Şekil-4.4; İkinci davranışın örnek bir uygulanması

Üçüncü davranış, tek bir dijitin değiştirilmesi eylemidir. Belirlenen iki dijit numarası ve üçüncü eşik değeri kullanılarak dijit setinin bir dijiti değiştirilir. Belirlenen iki dijitin farklı olup olmamalarına bakılmaksızın uygulama yapılır.

Örnek olarak *{0,0,0,1,1,1,0,1}* dijit seti ve 0,1,0 rassal sayılar belirlenmiş olsun (aktif durum). Belirlenen işlem dijitleri 0. dijit (0) ve 1. dijit (0), adreslenen dijit ise 0

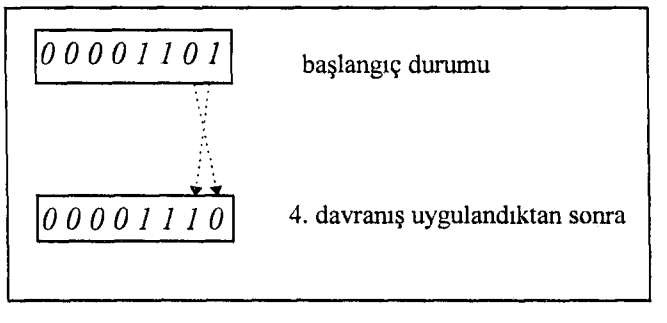
dijitidir. O halde üçüncü davranış ile 0. dijitin değeri 0'dan 1'e dönüşecektir. Bu durum Şekil-4.5'te de sunulmuştur.



Şekil-4.5; Üçüncü davranışın örnek bir uygulanması

Dördüncü davranış da, benzer şekilde belirlenen her iki dijitin yerlerini değiştirme eylemidir. Daha önceki davranışlarda sadece tek bir dijiti değiştiren operasyon yapılırken, bu davranış mümkün olması halinde dijitlerin her ikisinin de değiştirilmesiyle hedefe iki adım birden yaklaşmayı amaçlamaktadır. Burada üçüncü rassal sayıya bir işlev yüklenmez. Çünkü seçilen her iki dijit de operasyona tabi tutulmaktadır.

Örneğin dijit seti *{0,0,0,0,1,1,0,1}*, rassal sayılar da 6,7,1 olsun. 6.dijitin değeri 0, 7.'nin değeri ise 1'dir. Bu davranışın uygulanması ile bu iki dijitin yerleri değiştirilir. Yani 6. dijit 1,7. dijit ise 0 olur. Bu durum Şekil-4.6'da görülebilmektedir. Dördüncü davranış, her iki dijitin değerlerinin hem hedef değerlerden ve hem de birbirinden farklı olması halinde olumlu bir gelişme sağlar. Diğer durumlarda, aktif durum yapısında bir ilerleme sağlanamayacağından hedefe doğru bir ilerleme kaydedilemez.



Şekil-4.6; Dördüncü davranışın örnek bir uygulanması

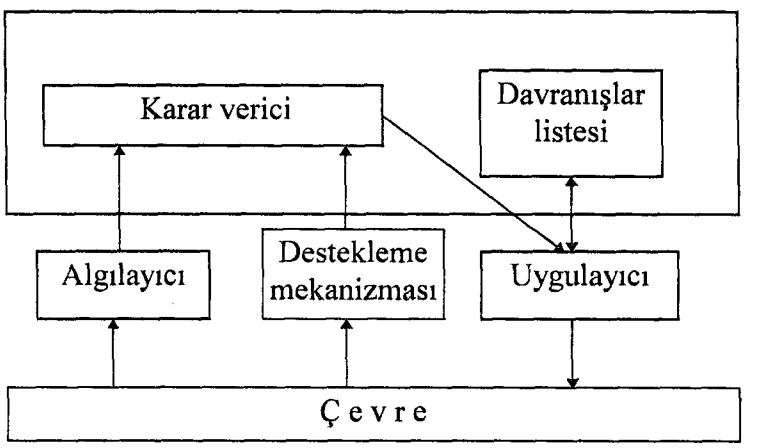
Beşinci ve son davranış ise, zeki etmenin çevre ile etkileşimi sonucunda herhangi bir eylem yapmamasıdır. Bu davranış, özellikle değiştirilmemesi gereken ve/veya mevcut davranışlardan biri ile olumlu bir değişikliğin yapılamayacağı durumların söz konusu olması halinde zeki etmenin durumu geçiştirmesini sağlar. Örneğin hedefe ulaşılmış olma, zeki etmenin aktif duruma hiç bir olumlu operasyon yapamaması hallerinde beşinci davranışa ihtiyaç duyulur.

4.2.2. Zeki etmenin Q-öğrenme algoritması ile eğitilmesi

Q öğrenme algoritması Bölüm 3'te de anlatıldığı gibi deneme yanılmaya dayalı bir destekli öğrenme algoritmasıdır. Bu bölümde bu algoritmanın zeki etmenlerin eğitilmesinde nasıl kullanıldığı ve ne gibi sonuçlara ulaşıldığı ele alınacaktır.

4.2.2.1. Zeki etmenin eğitimindeki yapısal boyut

Üçüncü bölümde de belirtildiği gibi destekli öğrenme algoritmalarının en önemli özelliği, öğrenme sırasında sistemin gösterdiği davranışın olumlu veya olumsuzluğu konusunda dışarıdan bir gözlemci veya bizzat çevre tarafından verilen uyarının öğrenme eyleminde en önemli veri olarak kabul edilmesidir [Dorigo ve Colombetti, 1994]. Bu nedenle daha önce tasarlanan zeki etmen yapısına çevresi ile etkileşimini yorumlayacak destekleme mekanizmasını eklemek gerekecektir. Şekil-4.7'de bu konuda bir öneri sunulmaktadır.



Şekil-4.7; Zeki Etmenin *Q* Öğrenme ile Eğitilmesi İçin Kurulan Sistem

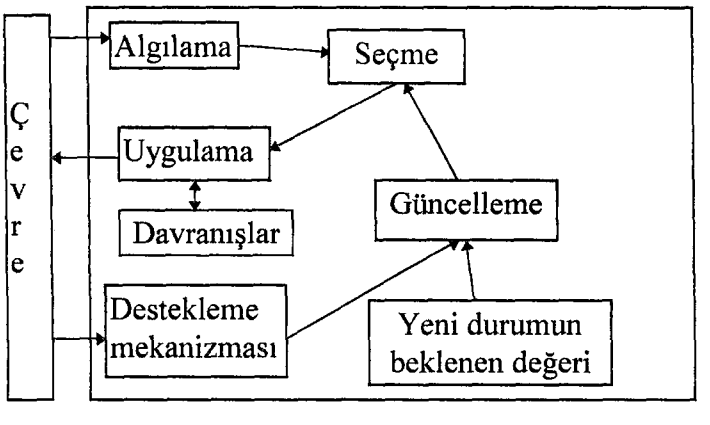
Şekil-4.7'de görülebileceği gibi zeki etmen çevre ile diyaloğundaki yeni işlem akışı şöyle olur: Aktif durumu algılayıcı algılar, ilgili mesaj karar vericiye iletilir, karar verici de uygun davranışı seçer ve uygulayıcıya mesajını iletir. Mesajı alan uygulayıcı, davranışlar listesinden ilgili davranışı uygulamaya koyar. Gerçekleşen operasyon sonunda çevrede meydana gelen değişimi destekleme mekanizması izleyerek sonucu değerlendirir ve karar vericiye bir değerlendirme uyarısı gönderir. Bu uyarı; durumla, seçilen davranış arasındaki ilişki hakkında fikir veren kriteri güncelleştirir. Bu da yeni davranışın seçilmesine yardımcı olur.

Zeki etmen eğitimin gerçekleştirilmesi için Q öğrenme algoritması ile tekrarlanan Şekil-4.8 'de sunulan prosedürü uygular. Gösterilen prosedürün her bir satırı öğrenmede bir modülü göstermektedir. Mesela `Algıla();` algılama modülünden gelen mesajı algoritma ortamına uygun hale getiren küçük bir prosedürüdür. Seçme prosedürü `( seç() );` en büyük Q değerine sahip olan davranışın seçilmesini, `uygula();` zeki etmenin uygulama modülünü çalıştıran ve seçilen davranışı uygulayan prosedürlerdir. `puan();` prosedürü destekleme mekanizması tarafından üretilen puanı, `güncelleştir();` modülü ise uygulanan davranışın Q değerinin yeniden hesaplanması fonksiyonunu yerine getirir. Böylece bir iterasyon tamamlanmış olur. Şekil-4.8'de prosedür olarak verilen modüllerin birbirleriyle olan ilişkileri Şekil-4.9'da akış diyagramı olarak gösterilmiştir

```
main()
        {
        algıla(durum);
        seç(uygun-davranış);
        uygula(uygun- davranış);
        puan(yeni-durum,değerleme_puanı);
        güncelleştir(aktif_davranış_Q_değerini);
        }
```

Şekil-4.8; Q öğrenme sürecinin gösterimi

Şekil-4.9'daki bilgi akışı şu şekilde gerçekleşmektedir: Başlangıçta verilen ilk durum algılama birbirine gelir. Gelen durum bir dijit setinden ibarettir. Söz konusu dijit setinin seçme modülüne anlamlı olarak gönderilmesi için üzerinde operasyona izin verilen dijit numaraları ve referans dijit adresini belirleyecek üç rassel değeri üretmek gerekir. Bunlar algılama biriminde üretilerek,durum oluşturulur. Yeni durum (dijit seti ve rassal sayılar) seçme modülüne iletilirler. Seçme modülü başlangıçta 0'a eşitlenen Q değerini göz önüne alarak rasgele davranışı (tüm davranışların Q değeri eşit olduğundan) seçer. Seçtiği davranışı uygulamak üzere uygulama modülüne ilgili bilgiyi ulaştırır. Uygulama modülü de söz konusu davranışı davranışlar listesinden alarak uygular. Uygulama sonunda elde edilen yeni dijit seti algılama modülüne iletilirken olayları izleyen destekleme mekanizması bir değerlendirme puanı üreterek güncelleme modülüne gönderir. Güncelleme modülü seçilip uygulanan davranışın Q değerini güncelleştirir. Güncelleştirme modülü kendisine iletilen yeni dijit setini ve bir sonraki iterasyon için üretilen rassal sayıları kullanarak yeni durumun beklenen en büyük Q değerini öncelikle hesaplar. Bu hesaplama yeni durum için davranışlar simule edilerek yapılır. Güncelleştirme kurallarının diğer önemli öğesi olan, bir aktif davranışa verilen puan da destekleme mekanizmasından sağlandıktan sonra söz konusu Q değeri güncelleştirilerek seçme modülüne iletilir. Böylece ilk iterasyon bitirilmiş olur. İkinci ve daha sonraki iterasyonlarda seçme modülü, güncelleşen Q değeri ile adım adım rasgelelikten kurtulur ve böylece en büyük Q değerine sahip olan davranışı tercih eder.



Şekil-4.9; Modüller arası ilişkiler diyagramı

Ele alınan zeki etmenin Q öğrenme algoritmasıyla eğitilebilmesi için oluşturlan destekleme mekanizması daha önce ifade edildiği gibi, bir davranışın aktif duruma göre seçilip uygulanması sonucu meydana gelen gelişmeleri izler ve puanlandırır. Puanlandırma eylemi bir takım kurallara göre gerçekleştirilir. Yararlanılan kurallar Şekil-4.10'da gösterilmiştir.

Şekil-4.10'da ifade olunan kural temsili gösterimle sunulmuştur. Bu kurallara göre puanlama sistemi bir davranışın uygulanmasında üç muhtemel sonucun doğabileceğini göstermektedir. Davranış dijit setinde hiçbir değişiklik yapmamak, hedefe yaklaştırmak veya hedeften uzaklaştırmak şeklinde olabilir. Zeki etmenin hiç bir değişiklik yapmama durumu Bölüm 4.2.1'de anlatılan birinci davranışın dijitlerin aynı olması halinde uygulanması, ikinci davranışın dijitlerin farklı olması halinde uygulanması, dördüncü davranışın dijitleri aynı olması halinde uygulanması veya beşinci davranışın uygulanması söz konusu iken ortaya çıkar. Bu durumlarda destekleme mekanizması 0 puanını üreterek zeki etmene gönderir. Diğer durumlarda ise sonuç hedefe yaklaştırıcı ise 1 puanını, uzaklaştırıcı ise -1 puanını üretir. Bu puanlandırma yardımı ile zeki etmen eğitilmeye çalışılır.

```
puanla (eski_durum,yeni_durum,hedef_durum)
      {
         int not =0;
         if (eski_durum=yeni_durum)
             not =0;
         else
             {
                if (yeni_durum hedefe daha yakın)
                   not = 1 ;
                else
                   not = -1
                 }
         return not ;
        }
```

Şekil-4.10; Destekleme mekanizmasının puanlamada kullandığı kuralların gösterimi.

4.2.2.2.Zeki etmenin eğitilmesi

Yapısal olarak yukarıda açıklanan zeki etmene Q öğrenme algoritması kullanılarak bir aktif durum karşısında hangi davranışta bulunması gerektiği öğretilmeye çalışılmaktadır. Herhangi bir başlangıç durumu algılayıp, hedefe doğru adım adım gitmeye çalışan zeki etmene algoritmanın izin verdiği kadar deneyim kazandırılır. Daha önce de belirtildiği gibi her öğrenme iterasyonunda, zeki etmen bir durumu algılar, karar verme mekanizmalarını kullanarak davranış listesinden bir davranış seçer, bu davranışı uygulama birimi ile uygular. Destekleme mekanizması çevrenin durumuna karşı zeki etmenin gösterdiği davranışa bir puan verir. Davranışın uygulaması sonucu ortaya çıkan yeni durum ve destekleme mekanizmasının verdiği puan ile zeki etmen ilgili davranışının Q değerini güncelleştirir. Bu çalışmada β=0.9 ve γ=0.5 olarak alınmıştır. Her bir iterasyonda zeki etmen davranışlarından birini deneyerek durumla ilişkisini tespit etmeye yönelik tecrübe kazanır. Başlangıçta tüm davranışların Q değerleri 0'a eşitlendiğinden algılanan durum için rasgele bir davranış öngörülür. Zamanla seçilen her bir davranışın Q değeri güncelleştirildiğinden her değişik durum için uygun davranış en yüksek Q değerine göre belirlenir. Böylece güncelleşen Q değerleri zeki etmenin deneyimini gösterir.

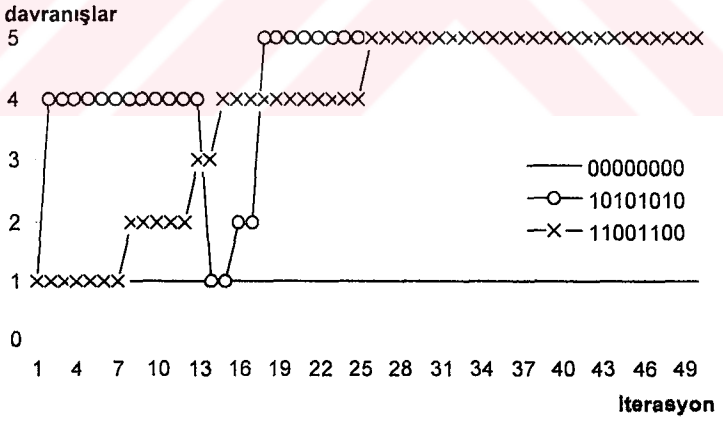
Zeki etmenin Q değerlerinin büyüklüğüne göre seçim yaparak kazandığı deneyim geçicidir. Bölüm 3'te ifade edildiği gibi Q öğrenme algoritması geçici farklar üzerine kurulmuştur. Bu yüzden her bir durum karşısında öngörülen davranış deneme-yanılma ile bir deneyim imkanı sağlamaktadır. Örneğin yapılan davranış olumlu ise ilgili davranışın Q değeri büyüyecek, olumsuz ise küçülecek, kayıtsız ise (0 puan almış ise) değişmeyecektir. Her bir çevrimde Q değerleri güncelleşerek geçmiş uygulamalardan geçici bir tecrübe sağlanacaktır. Bu şekilde tecrübenin genelleştirilmesi zordur. Bu çalışmada tecrübenin genelleştirilmesi konusu da araştırılmış olup daha sonraki bölümlerde ele alınacaktır.

Geliştirilmiş zeki etmen üzerinde yapılan eğitim çalışmalarında yukarıdaki konular gözlenmiş ve eğitim sonucunda yerel optimum problemi ile karşılaşılmıştır. Ortaya

çıkan yerel optimum, zeki etmenin hedefe vardığında sürekli 5. davranışı göstermesi yerine, daha hedefe ulaşmadan sürekli 5. davranışı sergilemesi şeklinde ortaya çıkmıştır. Benzer şekilde dijit setindeki tüm dijitler ya hep 0 veya hep 1'lerden oluştuğunda 1. veya 4. davranışa takılma da bir tür yerel optimum olarak ortaya çıkmıştır. Yapılan bir çok denemede yerel optimum problemi zeki etmenin hedefe varmasını engellemiştir. (Örnekler için bkzTablo-4.1).

Tablo-4.1; Yerel optimuma örnekler

Başlangıç durum	*sonuç*	iterasyon no
0 0 0 0 0 0 0 0	yerel optimum	1
0 0 0 1 0 1 0 1	yerel optimum	10
0 1 0 1 0 1 0 1	yerel optimum	21
1 0 1 0 1 0 1 0	yerel optimum	19
0 0 1 1 0 0 1 1	yerel optimum	14
1 1 0 0 1 1 0 0	yerel optimum	25
1 1 1 1 0 0 0 0	yerel optimum	8
1 1 1 0 0 0 1 0	yerel optimum	7
0 0 0 1 1 1 1 0	yerel optimum	14
1 1 1 1 1 1 1 1	yerel optimum	2



Şekil-4.11; Üç farklı durum için 50 adımlık davranış tercih profili

Şekil-4.11 üç farklı başlangıç durumu ile başlanan oyunlarda yerel optimumla karşılaşıncaya kadar uyguladıkları davranışların profilini göstermektedir. *{0,0,0,0,0,0,0,0}* durumu tamamen aynı dijit değerine sahip olduklarından başından

beri 1. davranışa takılmıştır. Diğer iki örnek ise bir kaç davranışı denedikten sonra 5. davranışa takılmıştır. Bu problemi aşmak üzere *Q* algoritması iyileştirilerek daha sonraki bölümlerde anlatılan algoritmalardan ilk olarak *Q-I*, sonra *Q-II* ve son olarak ta *Q-III* geliştirilmiştir. İlk iki algoritma yalnızca geçici deneyim kazanmak üzere geliştirilmişken, üçüncüsü zeki etmene hem geçici ve hem de kalıcı deneyim kazandırmak için geliştirilmiştir.

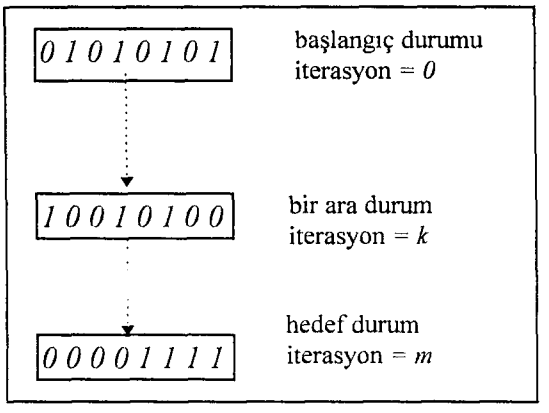
BÖLÜM 5 *Q-I* ÖĞRENME ALGORİTMASI

Q-öğrenme algoritmasının uygulanmasında Bölüm 4'te de anlatıldığı gibi zeki etmenlerin eğitilmesinde yerel optimum problemi ile karşılaşılmaktadır. Tezin bu bölümünde bir önceki bölümde ele alınan dijit oyununu oynamak üzere geliştirilen zeki etmenin eğitilebilmesi için, *Q* öğrenme algoritmasının geliştirilmesi için yapılan çalışmalar anlatılmaktadır.

Dijit oyununun bir özetinin burada sunulması problemin özelliğinin belirlenebilmesi için kolaylaştırıcı olacaktır. Oyun, verilen bir başlangıç durumu önceden belirlenen hedef duruma dönüştürme eylemini içerir. Her durum 8 dijitlik bir binary seriden oluşmaktadır. Zeki etmenin kendisine verilen başlangıç durumunu istenen hedef duruma getirebilmek için alternatif davranışlar sergilemesi gerekmektedir. Daha önce belirtildiği gibi bu davranışlar şöyle sıralanabilir.

- seçilen iki dijit değerini aynılaştırma.
- seçilen iki dijit değerini farklılaştırma.
- seçilen iki dijitten birini rassal eşik değerine göre değiştirme.
- seçilen iki dijitin yine rassal eşik değerine göre yerlerini değiştirme.
- duruma kayıtsız kalma.

Etmen her iterasyonda sadece bir davranış gerçekleştirebilecek ve her bir davranışta da sadece rasgele seçilen iki dijit üzerinde işlem yaparak yeni bir durum oluşturacaktır. Aynı işlemleri tekrar tekrar yaparak hedef duruma ulaşmaya çalışacaktır (bkz. Şekil-5.1).



Şekil-5.1; Tipik bir öğrenme süreci

Burada rassal değişkenlerin etkin olarak kullanılması probleme reaktif bir özellik kazandırmaktadır. Her iterasyonda hem dijit setinin değişmesi ve hem de işlenmek üzere sunulan dijit numaralarının rassal olarak belirlenmesi durum uzayında çok hızlı bir değişimi meydana getirmektedir. Bu hızlı değişim, zeki etmenin davranış sergilemede reaktif bir karakter kazandırmaktadır. Dijit oyunu probleminin bir başka özelliği de gündeme gelmiş bir durumun sonraki iterasyonlarda da gündeme gelebilmesidir. Şöyleki; onuncu iterasyonda ikinci dijit *1* iken üçüncü davranışla *0* yapılarak hedefe yaklaştırdığı varsayılsın. Onbirinci iterasyonda tamamen ilgisiz bir durum karşısında zeki etmen onuncu iterasyonda edindiği Q değeri ile karar vermek durumundadır. Yani daha önce atılmış bir adım sonraki iterasyonlarda tekrar geri çekilebilir. Durumlar arası geçişlerde rassal değişkenler ardışık durumların birbirlerini etkilememesini sağlayarak problemi Markoviyen olmayan bir yapıya sokmaktadır. Q öğrenme algoritmasında Markoviyen olmayan olayları da öğretebilmek amacıyla durum değişkenleri, Q değerleri ile ikililer oluşturarak Markoviyen ortam oluşturulmaktadır. Bu problem Bölüm 4'te anlatıldığı gibi Q algoritması ile çözülmeye çalışılmış fakat problemin özelliğinden dolayı yerel optimum oluşmuş ve öğrenme gerçekleştirilememiştir. Yerel optimumdan kurtulabilmek için *Q-1* adı verilen aşağıdaki yöntem önerilmektedir.

Bilindiği gibi Q öğrenme algoritması her zaman global optimumu garantilememektedir. Bu özellikle Markoviyen olmayan olayların öğrenilmesinde

kendini göstermektedir. Bu problemi çözmek için bazı çalışmalar yapılmıştır. Whitehead ve Lin (1995), Markoviyen olmayan olayların Q öğrenme algoritması ile öğretilmesinin bir hayli zorluklar getirdiğini ifade etmişlerdir. Bunun için bir çok algoritmanın birleştirilerek kullanılmasını sağlayan bir temsil metodunu önermişlerdir. Lin (1992) ise bunun için öğretmenli öğrenmeye benzer bir sistemle zeki etmene öğrenme sürecinde destek sağlamayı deneyerek öğretmeyi başarmıştır. Yoksa tek başına Q öğrenme algoritması bazan yerel optimumdan kurtulamayabilmektedir. *Q-I* öğrenme algoritması buna benzer şekilde bir destekleme prosedüründen ibarettir.

5.1. *Q-I* Öğrenme Algoritması

Q-I, bilinen Q öğrenme algoritmasının iyileştirilmesi yolu ile yerel optimum probleminden kurtulmak için geliştirilmiştir. En önemli özelliği, yerel optimumu aşmak için öğrenme sürecine bir cezalandırma mekanizması ile destek vermesidir. Önerilen cezalandırma mekanizması, sürekli olarak peş peşe seçilen davranışa belirli bir zaman dilimi sonucunda ceza vererek başka davranışların seçilmesine olanak tanımaktadır. Sürekli seçilen davranışın Q değerini bir süre için seçilmeyi engelleyecek oranda azaltarak başka bir davranışın seçilmesini sağlayan cezalandırma mekanizması şu şekilde çalışır:

Q-I öğrenme algoritması Şekil-5.2'de sunulduğu üzere adım adım uygulanır. Her iterasyonda seçilen davranışın bir önceki iterasyonda da seçilip seçilmediği kontrol edilir. Bir davranış ardarda *10* iterasyon seçildi ve aldığı puan da *0* ise cezalandırılır. Bu durumda Q değerinin güncelleştirilmesi aşağıda verildiği gibi normal güncelleme kuralı yerine cezalı davranış için belirlenen kural yani;

$$Q_{y}(x,a) \leftarrow \delta Q_{e}(x,a) - \hat{Q} \qquad \text{kullanılarak yapılır.}$$

Burada δ katsayısı Q değerlerindeki artışları yavaşlatmayı amaçlayan *[0,1]* aralığında bir sayıyı göstermektedir. $\hat{Q}$ ise aktif davranış dışındaki davranışların Q değerlerinin ortalaması olup şöyle hesaplanır:-

$$\hat{Q} = \frac{\sum_{i=1}^{n} Q_i + \omega}{n} \quad \text{eğer i aktif davranış değilse}$$

ω tüm değerler sıfır (*0*) iken cezalandırmanın optimum bir değer olması amacıyla kullanılan *[0, 1]* aralığında çok küçük bir başlangıç değerdir. Her iki güncelleştirme kuralı sentezlenerek şu kural elde edilir:

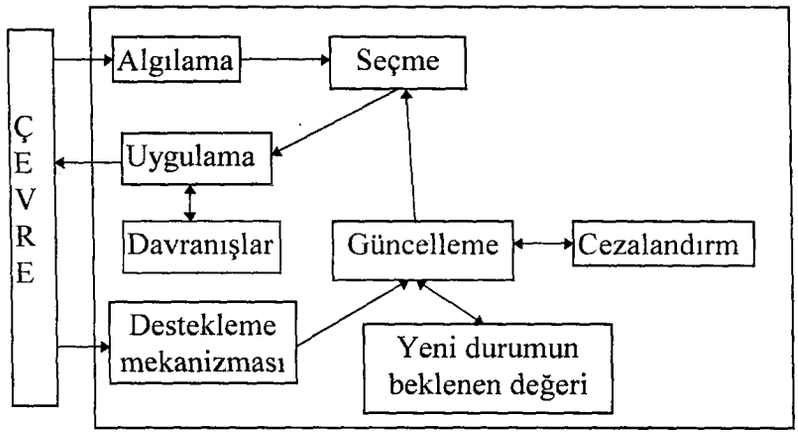
$$Q_1(x,a) \leftarrow \begin{cases} \delta Q_c(x,a) - \hat{Q} & \text{cezalı davranış için} \\ Q_c(x,a) + \beta(r + \gamma e(y) - Q_c(x,a) & \text{diğerleri için} \end{cases}$$

1. Her x ve a için $Q_1(x,a)$ değerlerine başlangıç olarak *0* ata.
2. Şu adımları tekrarla:
   - x durumunu gözle, $x \in S$,
   - a davranışını seç, $a \in A$,
   - a davranışını uygula y yeni durumunu ve r puanını al,
   - $Q(x,a)$ değerini aşağıdaki güncelleme kuralına göre güncelle,

$$Q_1(x,a) \leftarrow \begin{cases} \delta Q_c(x,a) - \hat{Q} & \text{cezalı davranış için} \\ Q_c(x,a) + \beta(r + \gamma e(y) - Q_c(x,a) & \text{diğerleri için} \end{cases}$$

Şekil-5.2: *Q-I öğrenme* algoritması

Q-I algoritmasının blok diyagramlar halindeki gösterimi Şekil-5.3'te verilmiştir. Bölüm 4'te zeki etmenin öğrenme esnasında hangi birimlerinin birbirleri ile nasıl etkileştikleri Şekil-4.9'da verilmişti. Burada güncelleme modülünün diğer modüllerle daha çok diyalog halinde olduğu görülmektedir. Süreçte yerel optimumun baş gösterip göstermediğini kontrol etmesi için her iterasyonda cezalandırma mekanizması ile bilgi alış-verişi yapılmaktadır.



Şekil-5.3; Cezalandırma mekanizmasının diğer modüllerle olan etkileşimi

5.2. Tartışma

Bölüm 4'te belirtildiği gibi dijit oyunu problemi, bilinen Q algoritması kullanılarak çözülmeye çalışılmış, ancak zeki etmenin bir kaç iterasyondan sonra yerel optimuma yakalandığı görülmüştür. Bu problem bu bölümde anlatılan cezalandırma fonksiyonu kullanılarak çözülmüştür. Değişik başlangıç durumlarına ait Q ve Q-I algoritmalarının karşılaştırılması Tablo-5.1'de verilmiştir. Tablodan da görülebileceği gibi Q algoritması ile yapılan çözüm araştırmalarının tümünde yerel optimum problemi baş göstermiş, ancak Q-I algoritması sayıları verilen iterasyonlar sonunda çözüm vermiştir. Bu problemlerin çözümünde deneme-yanılma yoluyla δ ve ω parametre değerleri sırasıyla 0.5 ve 0.01 olarak tespit edilmiştir.

Tablo-5.1; Bazı başlangıç durumlarına göre Q ve Q-I algoritmalarının öğrenme iterasyon sayısı

Başlangıç durum	*Q*	*Q-I*
0 0 0 0 0 0 0 0	yerel optimum	35624
0 0 0 1 0 1 0 1	yerel optimum	13029
0 1 0 1 0 1 0 1	yerel optimum	4353
1 0 1 0 1 0 1 0	yerel optimum	27652
0 0 1 1 0 0 1 1	yerel optimum	2282
1 1 0 0 1 1 0 0	yerel optimum	2232
1 1 1 1 0 0 0 0	yerel optimum	16124
1 1 1 0 0 0 1 0	yerel optimum	7836
0 0 0 1 1 1 1 0	yerel optimum	25
1 1 1 1 1 1 1 1	yerel optimum	9535

5.3. Sonuç

Bu bölümde *Q-öğrenme* algoritmasının oldukça dinamik bir probleme uygulanması ve oluşan problemlerin çözümüne yönelik bir iyileştirme çalışması anlatılmıştır. İyileştirme sunucunda, cezalandırma mekanizması ile yerel optimuma takılan zeki etmen desteklenmiş ve hedefe ulaşması (öğrenmesi) sağlanmıştır. Tablo-5.1'de de görülebileceği gibi yerel optimum aşıldığı halde hedefe ulaşma uzun iterasyonlar sonunda ulaşılabilmektedir. Bu da algoritmanın daha da iyileştirilmesine gerek duyulduğunu göstermektedir. Çünkü öğrenme zamanının uzun olması istenen bir durum değildir. Öğrenme süresini kısaltmaya yönelik gerçekleştirilen çalışma bir sonraki bölümde tartışılmaktadır.

BÖLÜM 6 *Q-II* ÖĞRENME ALGORİTMASI

Bölüm 5'te anlatılan *Q-I* algoritması ile her ne kadar yerel optimum problemi çözüldü ise de *Q-I* algoritması ile zeki etmenin hedefi yakalaması oldukça uzun zaman almaktadır. Bir öğrenme algoritmasının performansının öğrenme süresi ve olayı kavramadaki başarısı açısından ölçüldüğü bilinen bir gerçektir. O nedenle öğrenme süresinin azaltılmasını sağlayacak bir çalışma gerçekleştirilmiş ve *Q-II* adı verilen algoritma oluşturulmuştur. Tezin bu bölümünde *Q-II* algoritması ve bu algoritma kullanılarak dijit oyununun öğretilmesi konusundaki çalışmalar anlatılacaktır.

6.1. *Q-II* Öğrenme Algoritması

Q-II algoritması, hem *Q* öğrenme algoritmasının yerel optimum problemi hemde *Q-I* algoritmasının zaman problemlerini çözmek üzere geliştirilmiştir. Algoritmanın en önemli özelliği, gerek yerel optimumu aşmak ve gerekse zaman problemini elimine etmek için öğrenme sürecindeki güncelleştirme kuralına yeni bir yaklaşım getirmesidir. Bu yaklaşımda, zeki etmenin bir iterasyon boyunca aktifleştirdiği bir davranışından kazandığı tecrübeyi genelleştirerek diğer davranışlarına da aralarındaki ilişkiler oranında yansıtabilmesi sağlanmıştır.

Daha önce belirtildiği gibi *Q* öğrenme algoritması, her bir algılanan duruma karşı bir davranışı seçip uygulamayı ve sadece bu davranışın değerleme fonksiyonu (*Q*) değerini güncelleştirmeyi öngörür. Burada dayanılan gerekçe, canlıların normal yaşamlarında yaptıkları davranış ne ise onunla ilgili deneyim kazanabilecekleri ve bir uygulamada ancak bir davranış için deneyim edinilebileceği varsayımıdır [Lin,

1992]. Halbuki bir çocuğun örneğin elini yakıcı sıcaklıktaki bir cisime değdirmesi ona aynı zamanda sıcak bir cisime yaklaşılmaması gerektiği tecrübesini de kazandırmaktadır. Eğer bu örnekte sadece sıcak cisme el değdirme davranışına özel bir güncelleştirme söz konusu olsaydı çocuğun sıcak cisimlerden kaçma veya onlara yaklaşmaktan kaçınma davranışlarını sergilememesi gerekirdi. Bu felsefeden hareketle bir davranıştan edilen deneyimin zeki etmen repertuarındaki diğer davranışlar için de bir belirleyiciliğinin olduğu söylenebilir. Bu belirleyicilik ancak davranışların birbirlerini etkileme miktarlarınca olabilir. Dolayısıyla bu tür bir genelleştirme yapabilmek için davranışlar arası etkilenme oranını bilmek gerekir. Bir zeki etmen tasarlanırken onun söz konusu davranışları da beraberinde tasarlandığından aralarındaki etkilenme ilişkisi de belirlenebilir. Söz konusu etkileşim oranları ile etkileşim matrisi adı verilen bir matris oluşturulup güncelleme kuralında değişiklik yapılmıştır. Etkileşim matrisi μ ve bu matrisinin her bir elemanı da μ_i olarak tanımlanmıştır. Bu elemanlar kullanılarak davranışların hepsinin Q değerleri güncelleştirilmektedir. Buna paralel güncelleme denmektedir. Bu güncelleme kuralı şu şekilde gösterilebilir:

$$Q_y(x,a_i) \leftarrow \mu_i(Q_e(x,a_i) + \beta(r + \gamma e(y) - Q_e(x,a_i)))$$

Bu kural gereğince aktive edilen bir davranışın sağladığı deneyim, diğerlerine de aradaki ilişki oranında yansıtılmaktadır. Bu işlem normal şartlarda bir davranışın Q değeri aktive edilmiş gibi gücelleştirildikten sonra etkileşim katsayısı ile çarpılarak yapılır. Bu çarpım sonucunda aktive edilmediği halde bir diğer davranışın aktive edilmesinden elde edilen deneyimden yeni bir Q değeri bulunmuş olur. Bu işlemler mevcut tüm davranışlar için yapılarak paralel bir günceleştirme sağlanmış olur.

1. Her x ve a için $Q_y(x,a)$ değerlerine başlangıç olarak 0 ata.
2. Şu adımları hedefi yakalamadıkça tekrarla:
   - x durumunu gözle, $x \in S$,
   - a davranışını seç, $a \in A$,
   - a davranışını uygula y yeni durumunu ve r puanını al,
   - $Q(x,a)$ değerini aşağıdaki güncelleme kuralına göre güncelle,

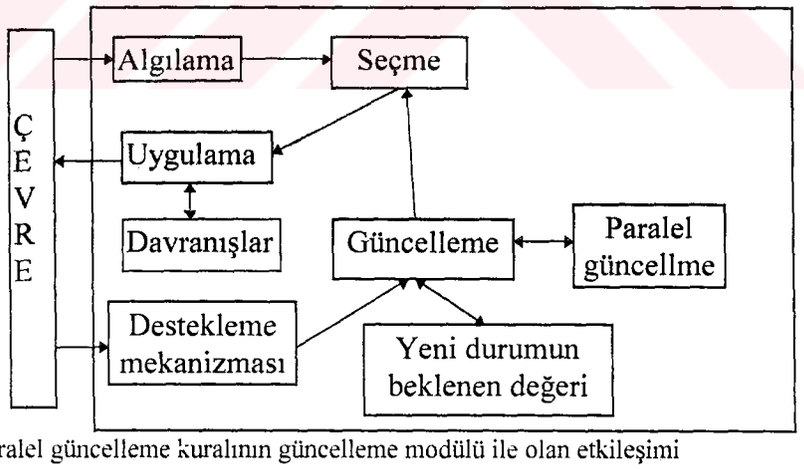
$$Q_y(x,a_i) \leftarrow \mu_i(Q_e(x,a_i) + \beta(r + \gamma e(y) - Q_e(x,a_i)))$$

Burada μ_i aktif davranışın i. davranışla olan ilişkisini, $e(y)$ y durumunun en yüksek Q değerini ifade eder.

Şekil-6.1: *Q-II öğrenme* algoritması

Literatürde paralel güncelleştirme yaklaşımına benzer iki önemli uygulamadan söz edilebilir. Singh (1992) ardışık görevlerden oluşan kompozit görevlerin başarılması için geliştirdiği sistemde her bir alt görevi bir Q modülü ile ifade etmiştir. Bir karar verme işleminde her bir Q modülünü çalıştırarak her alt görev için bir Q değeri hesapladıktan sonra geliştirdiği stokastik anahtarlama mekanizması ve yönlendirme modülü ile en son kararı etkileyecek genel bir Q değerine ulaşmaktadır. Bu değer de en son kararın verilmesinde kullanılmaktadır. Bu çalışmanın paralel güncelleştirme kuralına benzeyen yönü bir iterasyonda her ikisinin de tüm alternatif seçenekler için bir Q değerini hesaplamalarıdır. Ancak seçeneklerin seçimi sırasında Q değerleri hesaplanıyor iken, paralel güncelleştirmede bir davranış uygulandıktan sonra Q değerleri hesaplanmaktadır. Barto ve arkadaşları (1995) ise gerçek zamanlı dinamik programlamada her alternatif için bir değer hesaplayarak bir karara varmaktadırlar. Hesaplamalar dinamik programlamanın bir versiyonu şeklinde genişlemektedir.

Q-II algoritmasının uygulanma adımları Şekil-6.1'de sunulmuştur. Sistemin blok diyagramlar halinde ise Şekil-6.2'de verilmiştir. Bu şekilde paralel güncelleme sisteminin diğer modülleri ile etkileşimleri gösterilmektedir.



Şekil-6.2; Paralel güncelleme kuralının güncelleme modülü ile olan etkileşimi

6.2. Dijit Oyununda *Q-II* Algoritması

Bölüm 4'te de belirtildiği gibi dijit oyunu problemi, bilinen Q algoritması kullanılarak çözülmeye çalışılmış, ancak zeki etmenin bir kaç iterasyondan sonra yerel optimuma yakalandığı görülmüş ve bu problemin çözümü için Bölüm 5'te

anlatılan *Q-I* algoritması önerilmişti. *Q-I* algoritması cezalandırma mekanizması yardımı ile yerel optimumu aşmış ancak bu başarı uzun iterasyonlar sonucunda yakalanmıştı. Bunun yeniden ele alınıp hem yerel optimumdan kurtulma ve hem de iterasyon sayısından doğan zaman problemini elimine etmek için yapılan çalışma sonucunda paralel güncelleştirme kuralını içeren *Q-II* öğrenme algoritması geliştirilerek dijit oyununa uygulandı. Zeki etmenin kullandığı davranışlar için oluşturulan etkileşim matrisi (μ) davranışların problematik gereği birbirleri ile etkileşebilecekleri varsayılarak belirlendi.

$$\mu = \begin{pmatrix} 1 & 0.6 & 0.6 & 0.6 & 0.6 \\ 0.6 & 1 & 0.6 & 0.6 & 0.6 \\ 0.6 & 0.6 & 1 & 0.6 & 0.6 \\ 0.6 & 0.6 & 0.6 & 1 & 0.6 \\ 0.6 & 0.6 & 0.6 & 0.6 & 1 \end{pmatrix}$$

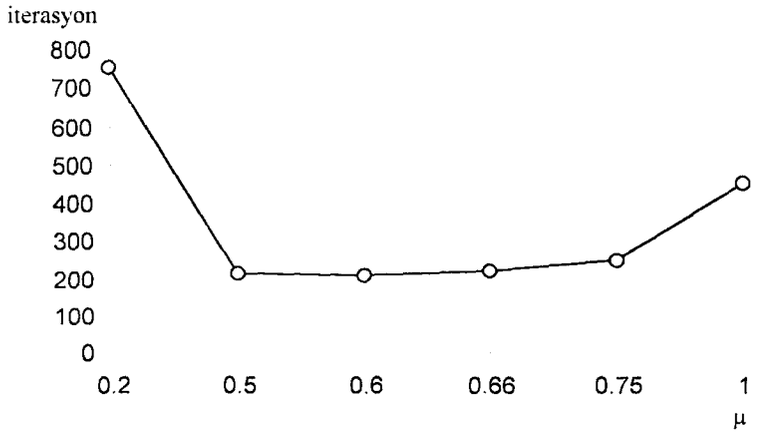
μ matrisi incelenince her bir davranışın bir diğeri ile olan etkileşim katsayısı 0.6 olarak belirlendiği görülecektir. O halde bu matris şu şekilde de yazılabilir:

$$\mu_i = \begin{cases} 1 & a_i \text{ aktif davranış ise} \\ 0.6 & \text{diğer durumlarda} \end{cases}$$

μ matrisi oluşturulurken yapılan eğitme çalışmalarında *0.6* yerine farklı değerler de denenmiştir. Bu denemeler sonucunda *0.6*'nın daha iyi olduğu anlaşıldı. Bu konuda yapılan denemelerin sonuçları Tablo-6.1'de sunulmuştur. Şekil-6.2'de ise μ değerinin değişim grafiği verilmiştir.

Tablo-6.1; Farklı μ değerleri için ortalama iterasyon sayıları

μ	Ortalama iterasyon sayısı
0.20	754
0.50	216
0.60	210
0.66	218
0.75	247
1.00	454



Şekil-6.2; İterasyon sayısı ortalamasının farklı μ değerlerine göre değişimi

Değişik başlangıç durumlar için yapılan uygulama örnekleri *Q-I* algoritması sonuçları ile karşılaştırmalı olarak Tablo-6.2'de verilmiştir. Burada *Q-II* algoritmasının başarısı belirgin bir şekilde gözükmektedir. Bu uygulamada deneme-yanılma yoluyla β ve δ parametre değerleri sırasıyla *0.5* ve *0.9* olarak tespit edilmiştir.

Tablo-6.2'de açık olarak belirtildiği gibi yapılan on uygulamada geleneksel *Q* algoritması tümüyle yerel optimuma takılmışken diğer iki algoritma belirtilen iterasyon sonunda hedefi yakalayabilmişlerdir. *Q-I*'in yalnızca *{0,0,0,1,1,1,1,0}* durumunda *Q-II* önüne geçtiği ve diğer durumlarda geri kaldığı görülmektedir.

Tablo-6.2; Bazı durumların *Q*, *Q-I* ve *Q-II* algoritmalarının öğrenme iterasyon sayısının karşılaştırılması

Başlangıç durum	*Q*	*Q-I*	*Q-II*
0 0 0 0 0 0 0 0	yerel optimum	35624	160
0 0 0 1 0 1 0 1	yerel optimum	13029	239
0 1 0 1 0 1 0 1	yerel optimum	4353	160
1 0 1 0 1 0 1 0	yerel optimum	27652	160
0 0 1 1 0 0 1 1	yerel optimum	2282	160
1 1 0 0 1 1 0 0	yerel optimum	2232	1477
1 1 1 1 0 0 0 0	yerel optimum	16124	307
1 1 1 0 0 0 1 0	yerel optimum	7836	307
0 0 0 1 1 1 1 0	yerel optimum	25	239
1 1 1 1 1 1 1 1	yerel optimum	9535	1477

6.3. Sonuç

Bu bölümde *Q-öğrenme* algoritmasının yerel optimuma takılması probleminin çözümü ve *Q-I* algoritmasının uzun öğrenme zamanı probleminin çözümü için geliştirilen *Q-II* algoritması anlatılmıştır. *Q-II* algoritmanın dijit oyununa uygulanması gerçekleştirilmiş ve *Q-I* ile karşılaştırılması yapılarak daha iyi sonuç verdiği gösterilmiştir. *Q-I*'de olduğu gibi *Q-II*'de de zeki etmenin genelleştirebilme yeteneğinin geliştirilmesi yönünde bir çalışma yapılmamıştır. Bu konu bir sonraki bölümde ele alınacaktır.

BÖLÜM 7 *Q-I* VE *Q-II* ÖĞRENME ALGORİTMALARINDA YAPISAL KREDİLENDİRME

Q öğrenme algoritmaları ailesinde iki türlü deneyim (geçici, kalıcı) kazanma söz konusudur. Bölüm 5 ve Bölüm 6'da *Q-I* ve *Q-II* algoritmaları yalnızca geçici deneyim (geçici kredilendirilme) kazanma açısından incelenmiştir. İlgili bölümlerde belirtildiği gibi geçici kredilendirme, Markoviyen özellikten dolayı sadece bir iterasyondan diğerine geçişte geçerli olabilen bir tecrübedir. Bundan dolayı zeki etmen bir örnek problemin çözümü boyunca geçici kredilendirme ile edinmiş olduğu deneyimini bir diğer örnek için kullanamaz. Bir zeki etmenin bir problem uzayının değişik örneklerinde edindiği deneyimine dayanarak yeni örneklere çözüm üretmesi geçici kredilendirme ile kazanılan deneyimin genelleştirilmesi ile mümkün olacaktır. Söz konusu genelleştirme daha önce belirtildiği gibi yapısal kredilendirme adı verilir. Her iki kredilendirme türü, Bölüm 3'de detaylı olarak açıklanmıştır.

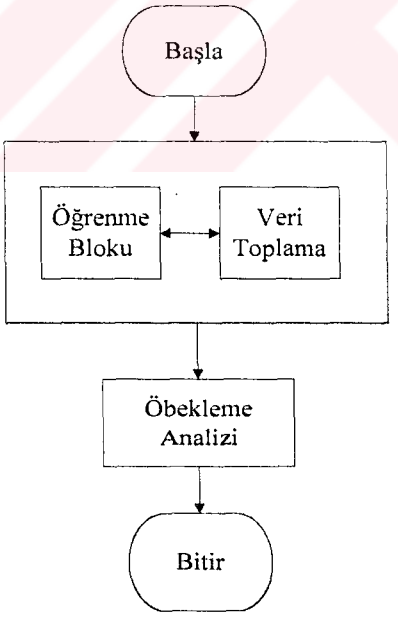
Bu bölümde yapısal kredilendirme problemini çözmek için geliştirilen *öbekleme analizine* dayalı bir yaklaşım anlatılmaktadır.

7.1. *Q-I* ve *Q-II* Algoritmalarında Yapısal Kredilendirme

Daha önce de belirtildiği gibi *Q* öğrenme ailesine mensup algoritmalar zeki etmene deneme-yanılma yöntemiyle deneyim kazandırırlar. Bu deneyimin kalıcı bir deneyim olabilmesi için öğrenme esnasında ek bazı mekanizmalar ile sistemi desteklemek gerekir. *Q* öğrenmenin doğasında kalıcı bir tecrübe kazandırmak olmadığından değişik çalışmalarda değişik metotlara başvurulmuştur. Anderson (1987) geçici farklar yöntemini, Lin (1992) de *Q* öğrenme ve AHC algoritmalarındaki faydalanma

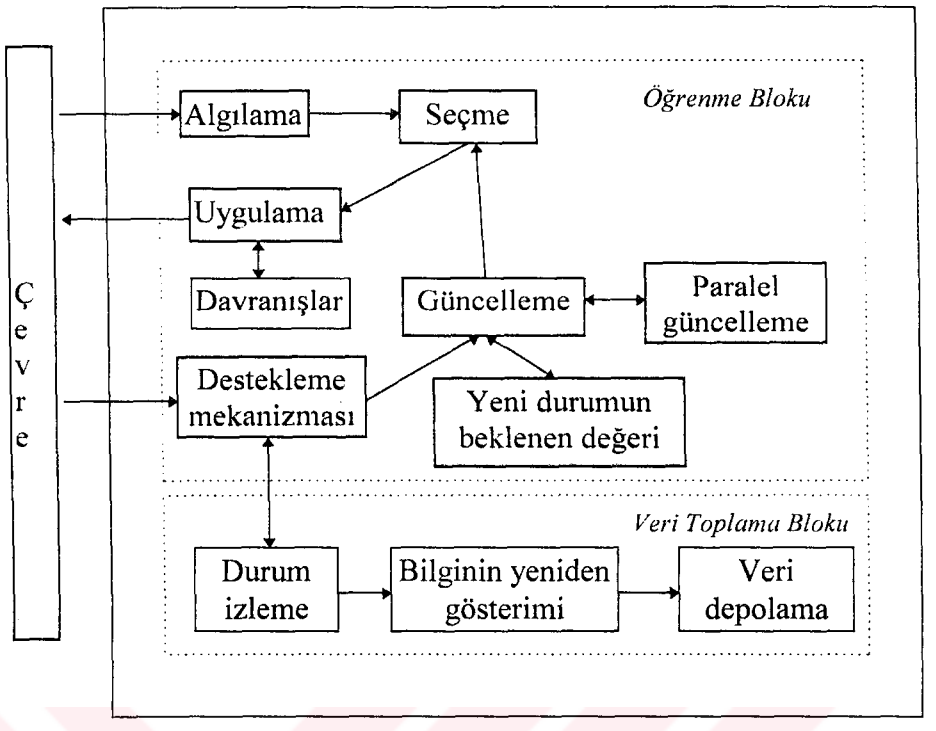
fonksiyonunu çok katmanlı algılayıcı yapay sinir ağına öğreterek yapısal kredilendirmeyi başarmışlardır. Bunun yanında Watkins (1989) CMAC adlı bir algoritmayı, Grefenstette ve arkadaşları (1990) genetik algoritma yöntemini, Chapman ve Kaelbling (1991) de karar ağacı adlı bir algoritmayı bu amaçla kullanmışlardır. Mahadevan ve Connell (1992) ise öbekleme analizine baş vurmuştur.

Bu çalışmada Mahadevan ve Connell (1992)'ın çalışmasına benzer şekilde öbekleme analizine dayalı bir sistem ile zeki etmene kalıcı deneyim kazandırıcı bir yöntem geliştirilmiştir. Bu yöntemin işleme yordamı, Şekil-7.1'deki akış şemasında verilmiştir. Öğrenim boyunca bir veri toplama modülü yardımıyla başarılı manevralar tespit edilerek depolanır. Daha sonra veriler öbekleme analizi ile sınıflandırılarak her bir davranışın duyarlı olacağı durumlar ayrı ayrı merkezlerde öbeklendirilir. Bu işlemden sonra yeni bir durumla karşılaşılınca ilgili merkezlere uzaklıklar hesaplanarak hangi davranışın aktive olacağına karar verilir.



Sekil-7.1; Yapısal kredilendirmenin öğrenme prosedürüne eklenmesi

Şekil-7.1'deki akış şemasında gösterilen öğrenme ve veri toplama bloklarının ilişkilerini gösteren detaylar Şekil-7.2'de verilmiştir.



Şekil-7.2; Yapısal kredilendirme boyunca öğrenme ve veri toplama bloklarının detaylı işleyişi

Veri toplama prosedürü, öğrenme boyunca durum izleme modülü yardımıyla olumlu bir tercihin gerçekleşip gerçekleşmediğini belirler. Olumlu bir durum-davranış eşleştirmesi gerçekleşince ilgili bilgileri yeniden gösterilmek üzere *bilginin yeniden gösterimi* modülüne gönderir. Burada verilerin öbekleme analizine uygun hale getirilmesi için yeniden bir gösterim gerçekleştirilir. Bu yeni bilgi, durum uzayından bir örnek durumu ve eşleştirildiği davranıştan oluşur. Bilgiler yeni haliyle dosyaya kaydedilir. Bilginin yeniden gösterimi Bölüm 7.1.1'de detaylı olarak anlatılmaktadır. Böylece zeki etmen hedef durumu yakalayıncaya kadar devam edecek olan öğrenme sürecinden verileri toplar. Hedef durum yakalanınca öğrenme süreci sona erer ve toplanan verilere sınıflandırılma amacıyla öbekleme analizi uygulanır.

Öbekleme analizi yukarıda da anlatıldığı gibi bir öğretmensiz öğrenme yöntemidir. Bu yöntemde veriler seçilen karakteristiklere göre sınıflandırılırlar. Karakteristiğin değeri çerçevesinde oluşacak veri merkezleri ve her bir verinin bu merkezlere olan uzaklıkları üyelik durumunu verir. Bir verinin öklidyen olarak en yakın olduğu merkez, ait olduğu merkezi yani eşleştiği davranışı verir. Bundan dolayı söz konusu karakteristik başarılı bir sınıflandırma yapabilmek için öklid uzayında ayrdedici bir

özelliği temsil etmelidir. Veri toplama sürecinde bilginin yeniden gösterimine olan gereksinim, durumların eşleştikleri davranışları sembolize eden merkezlerle olan ilişkisini başarılı olarak gerçekleştirmesinden kaynaklanmaktadır.

7.1.1. Bilgilerin yeniden gösterimi

Sistemde işlenen verilerin gösterimi, zeki etmenin durumlar ve davranışlar arasındaki ilişkiyi daha net ortaya koymak için yapılmaktadır. Problem son derece dinamik bir yapıda olduğundan ve değişik durumları iç içe barındırabildiğinden benzer durumları yakınlaştıran ve farklı durumları da ayrıştıran bir indeks geliştirmek ve yeni durum gösterimini bununla ifade etmek gerekir. Tasarlanan gösterim yaklaşımında geliştirilen indekste kullanılacak bilgiler, dijit setinden işlenmek üzere seçilen iki dijit değeri ve hedefe göre uygun olup olmamalarıdır. Bunun için dijit değerleri yeniden değerlemeye sokularak değiştirilmektedir. Böylece *0* ve *1*'lere aşağıda sunulan yeni değerler atanmaktadır.

$$0 \leftarrow \begin{cases} 1 & \text{konumu amaca uygunsa} \\ 3 & \text{değilse} \end{cases}$$

$$1 \leftarrow \begin{cases} 2 & \text{konumu amaca uygunsa} \\ 4 & \text{değilse} \end{cases}$$

Yeniden bilgi gösterimi için geliştirilen indeks x ve y gibi iki değerden oluşur. Bu indeksde iki değerin kullanılması, öbekleme analizinde öklidyen uzaklık hesaplamak için kaynaklar tarafından ideal olarak önerilmektedir [Ross, 1995]. Yukarıda anlatılan yeniden gösterim değerleri hem x ve hem de y için öngörülmektedir.

Yeni indeks hesaplamasında daha önce çekilen rassal sayılardan ilk ikisi öncelikle bu değerleri alırlar. Bunun yanında üçüncü rassal sayı işlevi gereği çekilen iki dijitin durumlarının amaca uygunlukları ve değişmesi teklif edilen dijitin adresini belirler. Eğer seçilen iki dijitin üçüncü rassal sayıya göre değiştirilmesi hedefe uygunsa yeni bilgi şöyle hesaplanır:

$$x = rnd_3 + 1$$
$$y = rnd_3 + 3.$$

Bu formülasyonda $rnd_3 = 0$ ise $x=1$, $y=3$ olurken $rnd_3 = 1$ durumunda $x=2$, $y=4$ değerlerini alırlar. Üçüncü rassal sayının değişmesini önerdiği dijitin değişmesi halinde hedeften uzaklaşılacak ise gösterim formülasyonu yeni ifadelerle dönüşümü gerçekleştirir. Bu şartlarda x değerinin hesaplanması şu formülasyona göre yapılır:

$$x = 0.4\,p_1 + 0.6\,p_2$$

Burada p_1 iki dijitten ilkinin p_2 de ikincisinin aldığı değeri ($\{1, 3\}$ veya $\{2, 4\}$) ifade eder. Bu bağıntının kullanılmasından amaç iki dijitin değerlerinden bir indeks üretmektir. Katsayılar ise iki değerin indeks üzerindeki ağırlıklarını göstermektedir. Bu katsayılar gösterimde daha dengeli ve tutarlı sonuçlar elde etmek üzere deneme-yanılma ile belirlenmiştir. Bununla beraber seçilen iki dijitin aynı veya farklı değerli olmaları durumuna göre de gösterim indeksinin hesaplanma formülasyonu;

$$x = \begin{cases} (0.4p_1 + 0.6p_2) + 1 & \text{farklı} \\ (0.4p_1 + 0.6p_2) - 3 & \text{aynı} \end{cases}$$

$$y = \begin{cases} rnd_3 - 3 & \text{farklı} \\ rnd_3 + 1 & \text{aynı} \end{cases}$$

olur.

Yeniden bilgi gösterimi prosedürü Şekil-7.3'te sanki program kodu halinde verilmiştir.

```
struct cluster indeks(struct cluster tmp,int rnd[])
{
  float p[2];
  register int i;

  if ( rnd[3]'ün adreslediği dijit hedef uygunsa )
     {
       tmp.x = rnd[2]+1.0f;
       tmp.y = rnd[2]+3.0f;
      }
  else
     {
     for(i=0;i<2;i++)
        {
          if (rnd[i])==0)
             {
              if (rnd[i] hedefe uygunsa )
                 p[i] = 1.0f;
              else
                 p[i] = 3.0f;
              }
          else
             {
              if (rnd[i] hedefe uygunsa )
                 p[i] = 2.0f;
              else
                 p[i] = 4.0f;
             }
         }

    if ( rnd[1]==rnd[2] )
       {
         {
          if (rnd[1] ve rnd[2] hedefe uygun değilse )
             tmp.y = rnd[2]-3.0f;
          else
             tmp.y = rnd[2]+1.0f;
          }
        tmp.x = 0.4f*p[0]+0.6f*p[1]+1.0f;
       }
    else
      {
        tmp.x = 0.4f*p[0]+0.6f*p[1]-3.0f;
        tmp.y = rnd[2]+1.0f;
       }
   }
  return tmp;
}
```

Şekil 7.3; Yeniden gösterim modülünün temsili gösterimi.

Örneğin seçilen iki dijitin değerleri *(0,0)* ve $rnd_3 = 1$ iken hedefe uygun olmasın. Bu durumda;

$x = rnd_3 + 1$ 'den $1+1=2$ $\quad y = rnd_3 + 3$'den $1+3=4$ olur.

Dolayısıyla $(x,y)=(2,4)$ elde edilir. Eğer hedefe uygun olsaydı;

$x = 0.4*1+0.6*3-3=-0.8$

$y = 1+1=2$ buradan $(x,y)=(-0.8,2)$ olur.

7.2. Öbekleme Analizi

Klasik öbekleme analizi, verilerin öklid uzayında sınıflandırılmasını sağlayan bir istatistiksel yöntemdir. Bir veri setinde her bir elemanın diğer elemanlarla paylaştığı ortak özellikleri ve ayrıldığı farklı özellikleri vardır. Bu yöntemle verilere ait benzerlik ve farkları ortaya koyarak veri seti alt setlere ayırabilmektedir. Bezdek bu klasik yöntemi, verilerin alt sınıflara olan üyeliklerine göre yeniden ele alarak *c-Ortalamalar öbekleme* ismiyle geliştirmiştir. Ortak özellikleri ile verileri yaklaştıran ve farklılıklarından dolayı da uzaklaştıran bu yeni yöntem, iki alt yönteme ayrılmaktadır: *Keskin c-Ortalamalar* ve *Bulanık c-Ortalamalar* [Ross, 1995]. Özellikle *Bulanık c-Ortalamalar* yöntemi için bir çok çalışma yapılmıştır. [Pal ve Bezdek, (1995), Emami ve arkadaşları, (1996)]. Bu çalışmada *Keskin c-Ortalamalar* yöntemi değiştirilerek kullanılmıştır.

7.2.1. Keskin c-Ortalamalar

Bu yöntem, verileri keskin bir biçimde sınıflandırdığından bu adı almıştır. Bir veri noktası bir sınıfa ya üyedir veya değildir şeklinde bir değerleme yapar. Sınıflandırma yapabilmek için önce elde sonlu sayıda veriden oluşan bir veri setine gerek duyulur. Bu veri seti n adet veriden oluşan bir X seti olsun. X evrensel setinin elemanları şu şekilde ifade edilir;

$$X = \{x_1, x_2, \ldots, x_n\}$$

Tüm verilerin ortak olarak değerlendirmeye alınması gereken bir de m adet özelliği olmalıdır. Bu özellikler yardımı ile veriler sınıflandırılır. x_i bir veri noktası ise sahip olduğu özellikler ;

$$x_i = \{x_{i1}, x_{i2}, \ldots, x_{im}\}$$

olur. Veri setinin eleman sayısı ve sınıflandırmada göz önüne alınacak özellik sayısı belirlendikten sonra verilerin kaç farklı alt sınıfa ayrılması gerektiğini belirlemek gerekir. Alt sınıf sayısı da c olsun. c'nin anlamlı olabilmesi için 2'den küçük ve n'den de büyük olmaması gerekir. Yani $2 \leq c \leq n$ olmalı. Bir veri setini, o setteki veri (eleman) sayısından çok veya ikiden az alt sınıfa ayırmaya kalkmak anlamsızdır. X evrensel kümesinin alt sınıflarından oluşan setler A_i ise X'in alt sınıfları $\{A_i, i=1,2,3,....,c\}$ olur. Alt sınıfların özellikleri ise tüm alt sınıfları toplamının evrensel sete eşit olması, her hangi iki alt sınıfın kesişiminin olmaması ve bir alt sınıfın boş setten ve evrensel setten farklı olmasıdır. Bunların ifadeleri şu şekilde verilebilir;

$$\bigcup_{i=1}^{c} A_i = X$$

$$A_i \cap A_j = \varnothing \quad her\ i \neq j$$

$$\varnothing \subset A_i \subset X \quad her\ i$$

Her bir veri noktasının yukarıdaki özellikleri sağlayan c sayıdaki alt sınıfa olan üyeliği ise $\chi_{Ai}(x_k)$ ile ifade edilsin. Alt sınıfların bu üç özelliğinin üyelik değerinde de olması gerekir. Buna göre;

$$\bigvee_{i=1}^{c} \chi_{A_i}(x_k) = 1 \quad \text{her k için}$$

$$\chi_{A_i}(x_k) \wedge \chi_{A_j}(x_k) = 0 \quad \text{her k için}$$

$$0 < \sum_{k=1}^{n} \chi_{A_i}(x_k) < n \quad \text{her i için}$$

ifadeleri $\chi_{Ai}(x_k)$'nin özelliklerini verir. Üyelik değeri ise yöntemin en önemli özelliği olan keskin sınıflandırmadan dolayı 0 veya 1 olabilmektedir. Yani;

$$\chi_{A_i}(x_k) = \begin{cases} 1 & x_k \in A_i \\ 0 & x_k \notin A_i \end{cases}$$

Yani eğer k noktasındaki veri (x_k) A_i sınıfından ise üyelik ($\chi_{Ai}(x_k)$) değeri 1 aksi durumda 0 olur. Kolaylık için $\chi_{ij} \equiv \chi_{Ai}(x_j)$ kabul edilirse χ_{ij} $(i=1,2,...,c; j=1,2,...,n)$ j. veri noktasının i. alt sınıfa olan üyeliğini ifade eder. χ_{ij}'lerden c satırlı ve n sütunlu bir U matrisi elde edilir. Buradan öbekleme eylemi boyunca U matrisinin oluşturacağı bir M_c uzayı elde edilir.

$$M_c = \left\{ U \middle| \chi_{ij} \in \{0,1\}, \sum_{k=1}^{c} \chi_{ik} = 1, 0 < \sum_{k=1}^{c} \chi_{ik} < n \right\}$$

M_c uzayında U matrisinin alacağı toplam değer sayısı da şöyle bulunur:

$$\eta_{M_c} = \left(\frac{1}{c!}\right)\left[\sum_{i=1}^{c}\binom{c}{i}(-1)^{c-i}.i^n\right]$$

Bu yöntem, verilen bu tanımlarla sınıflandırma sürecini işletir. Yöntem, iteratif olarak her alt sınıfa ait olan verileri belirlemek için öncelikle veri uzayında bulunan öbek (alt sınıf) merkezlerini hesaplar. v_i bir merkez ise m tane özelliğe göre ayrı ayrı alacağı değerlerin bir vektörü olarak ifade edilmelidir.

$$v_i = \{v_{i1}, v_{i2}, \ldots, v_{im}\} \quad i = 1, \ldots, c$$

Bu vektörün öbek (alt sınıf) sayısına göre açılımı yapılırsa v_{ij}'lerden oluşan $m \times c$ boyutlu bir v matrisine dönüşür. v_{ij} değerleri de şu formülasyona göre bulunur:

$$v_{ij} = \frac{\sum_{k=1}^{n} \chi_{ik} . x_{kj}}{\sum_{k=1}^{n} \chi_{ik}}$$

Merkezlerin koordinatları bulunduktan sonra veri noktalarının bu merkezlere olan öklidyen uzaklıkları da şu formülasyonla hesaplanır:

$$d_{ik} = d(x_k - v_i) = \|x_k - v_i\| = \left[\sum_{j=1}^{m}(x_{kj} - v_{ij})^2\right]^{1/2}$$

Bu hesaplamalardan sonra U matrisi güncelleştirilir. Güncelleştirme en yakın merkeze üyeliği ayarlama şeklinde gerçekleşir. Bu yöntem boyunca aslında bir iteratif optimizasyon yapılmaktadır. Optimize edilen fonksiyon ise bütün sistemin amaç fonksiyonu olan *J(U,v)*'dir. Bu fonksiyonun ifadeside şu şekildedir:

$$J(U,v) = \sum_{k=1}^{n}\sum_{i=1}^{c}\chi_{ik}(d_{ik})^2$$

Sistemin amacı *J(U,v)* fonksiyonunu enazlamaktır. Amaç fonksiyonunun enazlanması de yapılan iteratif U ve v değerlerinin güncellenmesi ile yapılmaktadır. İteratif sürecin ne zaman sona ereceği konusu, bu yöntemin temel problemlerindendir. Bu genellikle baştan iterasyon sayısı belirlenerek yapılmaktadır. *Keskin c-Ortalamalar* yöntemini şu adımlarla algoritmik olarak ifade etmek mümkündür:

Adım 1: c ($2 \le c \le n$)'yi belirle, ve U matrisinin ilk değerlerini ata,

$$U^{(0)} \in M_c$$

Sonra $r=0,1,2,..$ tekrarla

Adım 2: c sayıdaki öbeğin merkez koordinatları hesapla,

$$\left\{v_i^{(r)} \text{ ile } U^{(r)}\right\}$$

Adım 3: $U^{(r)}$ matrisini üyelik değerlerine göre güncelle,

$$\chi_{ik}^{(r+1)} = \begin{cases} 1 & d_{ik}^{(r)} = min\left\{d_{jk}^{(r)}\right\} \text{ her } j \in c \\ 0 & \text{diğer durumlar} \end{cases}$$

Adım 4: Eğer

$$\left\|U^{(r+1)} - U^{(r)}\right\| \leq \varepsilon \text{ (tolerans değeri) ise dur}$$

değilse

$$r \leftarrow r+1 \text{ yap ve Adım 2'ye git.}$$

Gerek öbekleme analizinin gerekse *Keskin c-Ortalamalar* yöntemin diğer detayları Ross (1995)'de bulunabilir.

Yukarıda anlatılan öbekleme analizinde tamamen verilerin özelliklerine dayalı olarak bir kendi kendine sınıflandırma söz konusudur. Her adımda verilerden yola çıkılarak önce merkez koordinatları hesaplanır, sonra da verilerin bu merkezlere olan uzaklıkları bulunur. Bu uzaklıklara dayanılarak verilerin başlangıçta kabul edilen merkezlere olan üyelikleri yeniden belirlenir. Yeniden belirlemede bir önceki iterasyonda farklı bir merkeze üyeliği kabul edilen bir veri en yakın olduğu merkeze doğru üyeliği değiştirilir. Bu yeni belirlenen üyelik bir sonraki iterasyon için belirleyicidir. Veriler başlangıçta sadece bir set halinde verilerek analiz edilmesi istenir. Sonuçta da sınıflandırılmış olarak kullanıcıya sunulur.

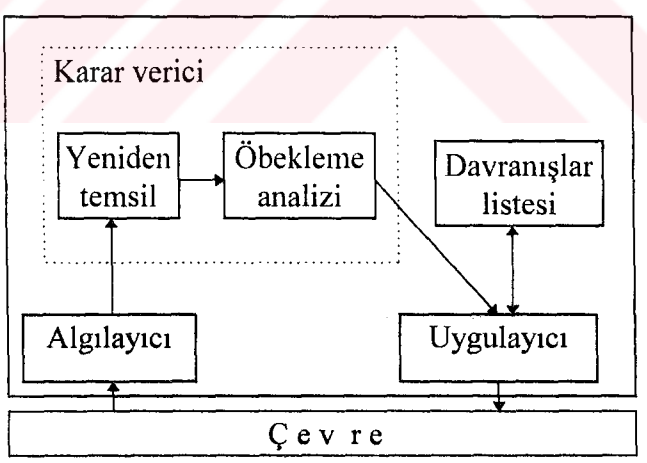
7.2.2. Geliştirilmiş keskin c-ortalama yöntemi

Bu çalışmada *Keskin c-Ortalamalar* yöntemi öğretmensiz öğrenme stratejisi yerine öğretmenli öğrenme sayılabilecek bir strateji ile uygulanmıştır. Yukarıda anlatılan teorik yapı yerine; geliştirilen sistemde veriler zaten öğrenim boyunca olumlu eşleştirmenin gerçekleşmesi halinde toplandıklarından üyelikleri sabit kabul edilerek bir merkez koordinatları hesabı ve uzaklık belirlemesi yapılmıştır. Çünkü U matrisi

optimal hali ile sisteme sunulmaktadır. Amaç fonksiyonunun diğer değişkeni olan v ise *Keskin c-Ortalamalar* algoritmasından yalnız bir iterasyon yararlanılarak optimize edilmiş olur.

7.3. Tartışma

Q-I ve *Q-II* algoritmaları için bu bölümde önerilen yapısal kredilendirme sistemi çevrim dışı olarak çalışır. Zeki etmen, öncelikle geçici kredilendirme ile hedefine ulaşmaya çalışır. Bu esnada veri toplama modülü de olumlu bir durum-davranış eşleştirmesi gerçekleştikçe verileri indeks halinde depolar. Hedef başarıldıktan sonra öğrenme ve veri toplama işlemleri son bulur. Bu aşamadan sonra verilere öbekleme analizi uygulanarak verilerin yoğunlaştıkları merkez koordinatları bulunur. Söz konusu her bir merkez bir davranışı gösterir. Sisteme indeks halinde girilen bir verinin bu merkezlere olan öklidyen uzaklıkları hesaplanır. Zeki etmen, en yakın olan merkezin temsil ettiği davranışa karar verir. Bu süreci Şekil-7.4 detaylı olarak vermektedir.



Şekil-7.4: Öğrenme süreci sonunda zeki etmenin yapısı

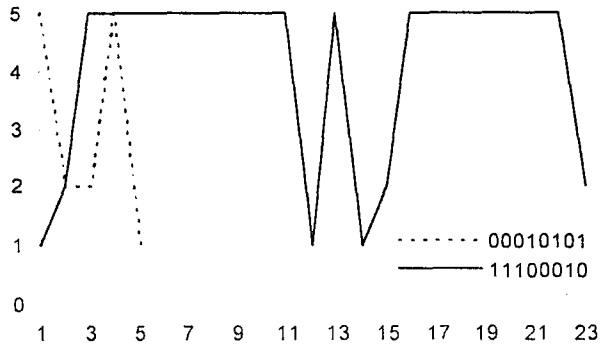
Zeki etmen çevrim dışı öğrenme sürecinden sonra çevresini tanıyarak bilinçli etkileşime hazır duruma gelir. Bu aşamada karar verici, algılayıcıdan aldığı bilgiyi öbekleme analizinde işlenebilecek bir gösterim ile yeniden düzenledikten sonra çevreden gelen bu yeni verinin hangi davranışla daha iyi eşleşebileceğine karar verir. Bu karar, verinin en yakın olduğu öbeği sembolize eden davranış için verilir.

Zeki etmen öğrenme süreci boyunca edindiği deneyimlerini bu aşamadan sonra kullanınca çok daha kısa sürede hedefe ulaşabilmektedir. Bu kapsamda Tablo-7-1 daha önceki bölümlerde anlatılan yaklaşımların karşılaştırmasını göstermektedir. Bu tabloda görüldüğü gibi öbeklemenin kullanılması öğrenme iterasyon sayısını oldukça azaltmaktadır.

Tablo-7.1: *Q, Q-I, Q-II* ve Öbekleme analizi destekli öğrenme süreçlerinin iterasyon sayısı bakımından karşılaştırılması

		İterasyon Sayısı		
Başlangıç durum	*Q*	*Q-I*	*Q-II*	*Öbekleme*
0 0 0 0 0 0 0 0	yerel optimum	35624	160	15
0 0 0 1 0 1 0 1	yerel optimum	13029	239	5
0 1 0 1 0 1 0 1	yerel optimum	4353	160	5
1 0 1 0 1 0 1 0	yerel optimum	27652	160	15
0 0 1 1 0 0 1 1	yerel optimum	2282	160	15
1 1 0 0 1 1 0 0	yerel optimum	2232	1477	23
1 1 1 1 0 0 0 0	yerel optimum	16124	307	23
1 1 1 0 0 0 1 0	yerel optimum	7836	307	23
0 0 0 1 1 1 1 0	yerel optimum	25	239	14
1 1 1 1 1 1 1 1	yerel optimum	9535	1477	23

Şekil-7.5'de Tablo-7.1'de sunulan bazı durumların oyun boyunca hangi davranışların seçilmesiyle hedefe vardıklarını grafik olarak sunmaktadır. En kısa sürede hedefe ulaştırılan durum daha 5. adımda hedefe ulaşmışken en uzun süreli hedefe varma örnekleri ise 23. adımda ulaşmışladır. Şekil-7.5'de 5. davranışın çok sık seçilmesi dikkat çekicidir. Bu, zeki etmenin durum karşısında beklemekten başka yapabileceği bir şeyinin olmadığını göstermektedir.



Şekil-7.5: Iki farklı başlangıç durumu ile hedefe ulaşmada seçilen davranışlar profili

7.4. Sonuç

Yapısal kredilendirme çalışması zeki etmenlerin *Q* öğrenme ailesindeki algoritmalarla eğitilmelerinin önemli bir aşamasıdır. Bu bölümde önerilen *Q-I* ve *Q-II* algoritmaları için geliştirilen yapısal kredilendirme sistemi anlatıldı. Önerilen yaklaşım bir çok problemi çözebilmesine rağmen deneyimi çevrim dışı olarak zeki etmene sunması bir dezavantajdır. Bir zeki etmenin etkin bir öğrenme sistemine sahip olabilmesi mümkün olduğunca kısa sürede deneyimlerini kullanabilir olmasına bağlıdır. Bu açıdan önerilen yapısal kredilendirme sistemlerinin çoğu dezavantajlı durumdadır. Bu probleme çözüm bulmak için bu tezde yeni bir yaklaşım önerilmiştir. *Q-III* algoritması olarak adlandırılan bu yaklaşım, bir sonraki bölümde anlatılmıştır.

BÖLÜM 8 *Q-III* ÖĞRENME ALGORİTMASI

Önceki bölümlerde önerilen *Q-I* ve *Q-II* algoritmaları hem geçici ve hem de yapısal kredilendirme problemleri açısından iyi sonuçlar vermelerine rağmen bazı dezavantajları da sahiptirler. Örneğin yapısal kredilendirme sistemi ile kazanılan kalıcı deneyimden *Q-I* ve *Q-II* algoritmaları öğrenme anında yararlanamamaktadırlar. *Q-öğrenme* ailesinin bu durumu yapısal bir problemden kaynaklanmaktadır.

Yapısal problem, geçici ve yapısal kredilendirme terimleri arasında bir bağıntının olmaması ve böylece karşılaşılan her yeni probleme rahatlıkla uygulanabilme zorluğu taşıması şeklinde tanımlanabilir. Daha önce belirtildiği gibi *Q öğrenme* ailesine mensup algoritmalarda geçici kredilendirme ile yapısal kredilendirme kavramlarının birbirleri ile bir bağları yoktur. Söz konusu ilişkiyi oluşturmak algoritmanın kullanımını kolaylaştıracağı gibi öğrenme sürecini de kısaltacaktır. Böylece algoritmanın yapısal probleminden kasıt her iki kredilendirme türünün bir bağıntı ile ilişkilendirilmesi ve birlikte algılanması gerekliliğidir. Bu bağlamda Bölüm 7'de çevrim dışı olarak çözülen yapısal kredilendirme probleminin çevrim içi olarak çözülmesi algoritmanın yapısal problemini çözmeye yönelik bir adım olacaktır.

Bu bölümde geçici ve yapısal kredilendirme çalışmalarını birleştirerek zeki etmene davranışların seçiminde daha sağlam bir deneme-yanılma eylemi gerçekleştirecek olan *Q-III* algoritması tanıtılacaktır.

8.1. *Q-III* Öğrenme Algoritması

Bölüm 7'de sunulan yapısal kredilendirme yöntemi *Q* öğrenme algoritmaları için çevrim dışı olarak gerçekleştirilebilir. Kazanılan kalıcı deneyimin anında öğrenme sürecine yansıtılması bu yüzden mümkün olmamaktadır. Bu dezavantaj çoğunlukla öğrenme sürecinde yavaşlamaya neden olmaktadır. Çevrim içi bir öğrenme sistemi öneren *Q-III* algoritması esasen *Q-II* algoritması ile öbekleme analizi yöntemlerinden *Keskin c-Ortalamalar* yordamının birleştirilmesinden ortaya çıkmıştır.

Q öğrenme ailesinde, seçilen davranışın faydalanma değeri güncelleştirilirken elde edilen yeni durumun beklenen faydalanma değerine ve çevrenin ürettiği uyarıcı puana dayalı hesaplama yapılır. Bu her iki değer bir davranışın bir sonraki iterasyonda seçilmesini doğrudan etkilemektedir. Güncelleme kuralını hatırlama açısından yazmak gerekirse:-

$$Q_y(x,a_i) \leftarrow \mu_i(Q_e(x,a_i) + \beta(r + \gamma e(y) - Q_e(x,a_i)))$$

Bu ifadede görülen *r* herhangi bir bağımlılığı olmayan bir değer iken *e(y)*, *y* olarak ifadelendirilen yeni durumun beklenen en yüksek faydalanma değeridir. *e(y)* bağıntısı beklenen değerden kaynaklanan probablistik karakterinin tam belli olmaması dolayısıyla problemden probleme değişebilmektedir. Bu değişkenlik faydalanma fonksiyonunda bir yapısal problem ortaya çıkarmaktadır. Burada *e(y)* ifadesini problemden bağımsız bir yapıya kavuşturmak bir gerekliliktir. Çünkü geliştirilen yapısal kredilendirme sistemi zeki etmene doğrudan kalıcı bir deneyim sağlayamamaktadır. O nedenle iki algoritmayı (*Q-II* ve *Keskin c-Ortalamalar*) birleştirme çalışmasının anahtar terimi *e(y)*'dir. *e(y)* terimi, stokastik karakterinden dolayı ya olasılıklı bir istatistiksel formülasyon veya simulasyon aracılığı ile hesaplanabilmektedir. Bu çalışmada *e(y)* değerinin saptanması için simülasyona yaklaşımı kullanılmıştır. Gerek istatistiksel yöntem ve gerekse simulasyon kullanılarak yapılan hesaplamalarda detaylar açısından kesin bir bağıntıdan bahsetmek zordur. Ancak genel olarak şu bağıntı verilmektedir:

$$e(y) = max\ Q(y,a),\ \forall a\ için$$

Bununla birlikte *e(y)* değerine deneyimlerin genelleştirilmesi yani her iki deneyim türünün birleştirilmesini sağlayabilecek yeni bir tanım getirmek gerekir.

Keskin c-Ortalamalar algoritması kullanılarak verilerin öğrenme esnasında öbeklenerek öklidyen uzaklıklarının saptanmasını sağlamak için *e(y)* değerinin ifadesini bir uzaklık fonksiyonu olarak yeniden belirlemek Q ailesinin deneyim açısından yapısal problemini çözecektir. O halde yeni *e(y)* ifadesi;

$$e(y) = d_{ik} = d(x_k - v_i) = \|x_k - v_i\| = \left[\sum_{j=1}^{m} (x_{kj} - v_{ij})^2\right]^{1/2}$$

şeklinde yeniden tanımlanabilir. Bu tanımlama aynı zamanda her davranışa ait faydalanma (Q) değerinin tanımını da yeniden yapmayı gerektirmektedir. Yeni Q tanımı ise ;

$$Q(x, a_i) = E(r + \gamma e(y) \mid \|x - v_i\|)$$

ifadesine dönüşür. Bu tanıma göre *i.* davranışın *x* durumundaki Q değeri durumun *i.* merkezden olan öklidyen uzaklığı ve *r* değerinin toplamının beklenen değeridir. Bu tanımlama öncelikle davranışları temsil eden öbek merkezlerinin koordinatlarının hesaplanmış olmasını gerektirmektedir. Adı geçen hesaplamalar için Bölüm 7'de anlatılan *Keskin c-Ortalamalar* algoritmasındaki öbek merkezleri saptama metodu kullanılabilir. Ancak bu bölümde farklı olarak öbek merkezleri hesabı daha önce toplanan veriler yerine içinde bulunulan iterasyonda elde edilen veriler kullanılarak yapılır. Öbek merkezlerinin saptanması *Q-III* algoritmasında işlem sırasının değişmesine neden olmaktadır. İşlem sıralarının değişmesi özellikle güncelleştirme kuralının yeri için söz konusu olmuştur. Bu takdirde yeni işlem sırası ;

1. Başlangıç değerlerinin atanması,
2. Durumun algılanması,
3. Faydalanma değerinin saptanması,
4. Uygun davranışın seçilmesi,
5. Seçilen davranışın uygulanması,
6. Destekleme mekanizmasının olguya ait puanı açıklaması,
7. Durum-davranış eşleştirmesi olumlu ise ilgili merkeze yeni bir verinin sunulması,

8. Öbek merkezleri koordinatlarının güncelleştirilmesi,
9. Hedefe varıldı ise işlemin durdurulması, aksi halde 2. adıma geri dönülmesi.

İşlem adımlarına dikkat edilince *Q* değerlerinin güncellenmesinin yanında öbek merkezleri koordinatlarının da güncelleştirildiği görülmektedir.

Q-III algoritmasındaki öğrenme sürecinde öncelikle bir davranışa ait *Q* değerlerine ve öbek merkezleri koordinatlarına başlangıç değeri olarak *0* atanır. Daha sonra yeni durum algılanarak her bir davranışın *Q* değerleri güncelleştirilir. En küçük *Q* değerine sahip davranış seçildikten sonra uygulanır. Uygulama sonucunda elde edilen yeni duruma göre destekleme mekanizması bir puan üretir. Puan olumlu olunca *-1* değerini, olumsuz olunca *1* değerini alır. Burada olumlu olan durumunda *-1* üretilmesinin hedefi minimum *Q* değerine sahip olan davranış seçildiğinden merkeze olan uzaklığı cebirsel olarak biraz daha kısaltmaktır. Bu puan olumlu olduğu takdirde sistem, durum-davranış eşleştirmesini yeni bir veri olarak saklar ve bu veriye dayanılarak ilgili öbek merkezi koordinatları güncelleştirilir. Olumlu olmaması durumunda da bir değişiklik yapılmadan bir sonraki iterasyona geçilir. Bu süreç Şekil-8.1'de adımlar halinde sunulmuştur. Burada *c* öbek sayısını, *m* sınıflandırmada dikkate alınan özellik sayısını, T_i, *i.* öbek etrafında toplanan verilerin toplamını, *n* bir öbek etrafında toplanan veri sayısını, v_{ij} ise *i.* merkezin *j.* boyutunu göstermektedir.

Q-III algoritmasında yer alan *Q* değerlerinin güncelleme kuralı *Q-II*'den farklı olarak davranışlar arası ilişki katsayısının (μ_{ij}) paralel güncelleme yaparken bağıntının sağ tarafının tümü ile çarpılması yerine yalnızca *r* puanı ile çarpılması önerilmektedir. Burada önemli bir avantaj elde edilmektedir. Bu davranış sayısında yapılabilecek yeni eklemelerin kolaylaşmasıdır. *Q-II* 'de paralel güncelleme yapılırken *Q* değerlerinde meydana gelebilecek olası artış ve azalışlar için simulasyon yapılmaktadır. Simulasyon, özellikle davranış sayısının çok olması durumlarında sisteme yavaşlatıcı ve hantallaştırıcı bir etki yapmaktadır. Oysa *Q-III* algoritmasında böyle bir kısıt ortaya çıkmamaktadır. Zeki etmenin davranışları arasında var olan

etkileşimler söz konusu davranışlar arası ilişki matrisinde gösterildikten sonra paralel güncelleştirmede bir problem ortaya çıkmamaktadır.

1. c ve m değerlerini belirle, Q_i, v_{ij} , T_{ij} değerlerine *0* ata,
2. Aşağıdaki adımları uygula
   - Yeni durumu algıla,
   - Q değerlerini aşağıdaki kurala göre güncelleştir,

$$Q_y(x, a_i) \leftarrow Q_e(x, a_i) + \beta(r\mu_i + \gamma e(y) - Q_e(x, a_i))$$

burada

$$e(y) = d_{ik} = \|x_k - v_i\| = \left[\sum_{j=1}^{m} (x_{kj} - v_{ij})^2\right]^{1/2}$$

   - En küçük Q değerli davranışı seç,
   - Seçilen davranışı uygula,
   - Yeni durumu (y) algıla ve r değerini elde et,
   - r olumlu ise

$$T_{ij} \leftarrow T_{ij} + x$$

   - seçilen davranış için öbek merkezi koordinatlarını şu kurala göre güncelle,

$$v_{ij} = \frac{T_{ij}}{n}$$

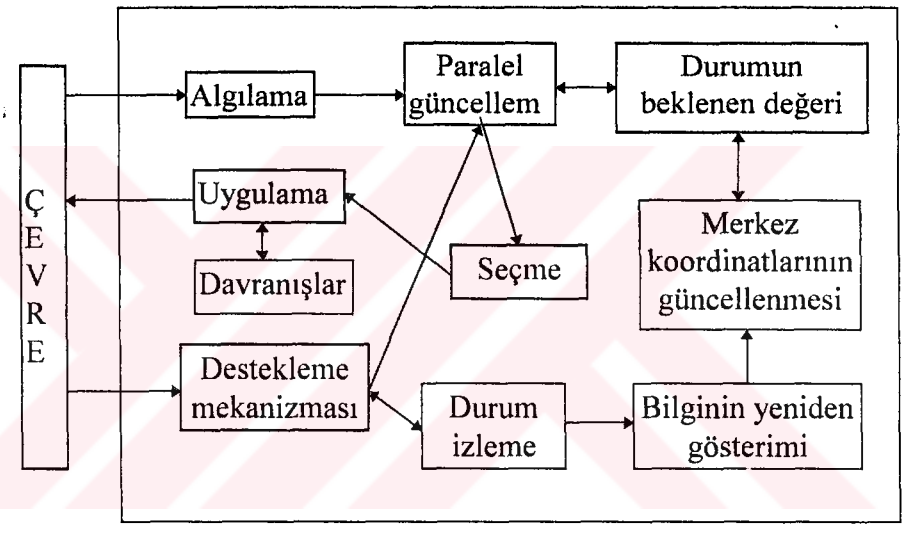
3. Hedefe ulaşınca dur, aksi takdirde 2. adıma git.

Şekil-8.1; *Q-III* öğrenme algoritmasının adım adım gösterimi.

8.2. *Q-III* ile Dijit Oynunun Öğretilmesi

Q-III algoritması kullanılarak dijit oynunu oynayabilmesi için zeki etmen eğitilmiştir. Eğitim süreci Şekil-8.2'de diyagram halinde sunulmuştur. Zeki etmen, yapılan yeni tasarıma göre hem geçici hemde yapısal kredilendirme eylemlerini paralel olarak yürütür. Bölüm 7'de önerilen yapıdan farklı olarak kredilendirme eylemlerini ayrı bloklar halinde değil de tek blok halinde beraberce gerçekleştirir. Daha önce belirtildiği gibi zeki etmen önce çevreden aktif durumu algılar. Sonra Q değerlerini güncelleştirir ve algılanan duruma en uygun olan davranışı uygulanmak üzere seçer. Seçilen davranışı uyguladıktan sonra *destekleme mekanizması*, bir puan üretir ve değerlendirilmek üzere *durum izleme modülü* ve *paralel güncelleme modüllerine* iletir. *Paralel güncelleme modülü* bir sonraki iterasyonda Q değerlerinin saptanmasında (geçici kredilendirme eyleminde) bu puanı kullanır. *Durum izleme modülü* ise yapısal kredilendirme eylemlerinin harekete geçmesi için bu puanı

değerlendirir. Bu aşamadan sonra puanın olumlu olması durumunda Bölüm 7'de anlatılan bilginin yeniden gösterimi modülü çalışır ve öbekleme eylemlerinde kullanılmak üzere bilginin yeni şeklini merkez koordinatlarının güncelleştirilmesi modülüne gönderir. Burada merkez koordinatları güncellenir. *Durumun beklenen değeri* bir sonraki iterasyon için yeni merkez koordinatları kullanılarak *Q* değerlerinin tespit eder. Burada öbek merkezlerinin güncelleştirilmeleri ve durumların bu merkezlere olan öklidyen uzaklıkları zeki etmene geçici ve kalıcı deneyim terimleri arasında bir etkileşim sağlamaktadır. Söz konusu etkileşim öğrenme sürecini hızlandırmaktadır. Bu sürecin detayı Şekil-8.2'de sunulmuştur.



Şekil-8.2; Zeki etmenin *Q-III* algoritması ile eğitilmesi

Başlangıç davranışlar hiç bir duruma özel bir duyarlılık göstermezler. Yani zeki etmenin başlangıçta davranışlar arası tercihini etkileyen bir deneyimi yoktur. Zamanla deneme-yanılma yöntemi ile davranışlara ait durumların öbeklendikleri merkezler saptanarak deneyim elde edilir. Bu deneyim her geçen adımda daha da artmaktadır. Hedef durum yakalandıktan sonra öğrenme eylemi tamamlanmış sayılır. Değişik başlangıç durumları ile yapılan öğrenme eylemi *Q*, *Q-I*, *Q-II* ve *Q-III* ile karşılaştırmalı olarak Tablo-8.1'de verilmiştir. Buradan da görülebileceği gibi *Q-III* algoritması bir örnek dışındaki diğer tüm örneklerde çok daha kısa sürede öğrenmeyi başarmaktadır.

Tablo-8.1: Bazı durumların *Q, Q-I, Q-II* ve *Q-III* algoritmalarının öğrenme iterasyon sayısı

Başlangıç durum	*Q*	*Q-I*	*Q-II*	*Q-III*
0 0 0 0 0 0 0 0	yerel optimum	35624	160	78
0 0 0 1 0 1 0 1	yerel optimum	13029	239	103
0 1 0 1 0 1 0 1	yerel optimum	4353	160	62
1 0 1 0 1 0 1 0	yerel optimum	27652	160	27
0 0 1 1 0 0 1 1	yerel optimum	2282	160	604
1 1 0 0 1 1 0 0	yerel optimum	2232	1477	285
1 1 1 1 0 0 0 0	yerel optimum	16124	307	109
1 1 1 0 0 0 1 0	yerel optimum	7836	307	103
0 0 0 1 1 1 1 0	yerel optimum	25	239	82
1 1 1 1 1 1 1 1	yerel optimum	9535	1477	82

Bu yapısal özelliklere sahip zeki etmen öğrenme eylemini gerçekleştirdikten sonra dijit oyununun herhangi bir başlangıç durumuyla oynamayı başarmıştır. Her bir duruma en uygun davranışı reaktif olarak belirleyerek uygulamış ve zeki etmen bilgilerini sürekli güncelleme yaparak aktif tutabilmiştir.

8.3. Sonuç

Bu bölümde *Q öğrenme* algoritmasında yapılan geliştirme çalışmalarının son adımı sunulmaktadır. Önerilen yeni algoritma gerek geçici ve gerekse kalıcı deneyim kazanma konusunda yapısal bir yaklaşımdır. Burada geçici ve kalıcı deneyim terimlerinin matematiksel bir bağıntı ile birbirini etkilemeleri sağlamıştır. Bu sayede zeki etmen öğrenme eylemini daha hızlı gerçekleştirmektedir. Bununla beraber, zeki etmen öğrenmeyi aktif bir eylem haline getirip çevre karşısında bir sürekli bilgilenme aracı olarak kullanma olanağı bulmaktadır. *Q-III,* dijit oyunu gibi çok dinamik bir probleme başarı ile uygulanmıştır. Bundan sonraki bölümlerde *Q-III*'ün Endüstri Mühendisliği'ne ait önemli problemlerden atölye tipi iş çizelgeleme problemine nasıl uygulandığı anlatılmaktadır.

BÖLÜM 9 İŞ ÇİZELGELEMEDE ZEKİ ETMENLER

İş çizelgeleme problemi hangi işin hangi makinada ve hangi sırayla işleme konulması gerektiğine karar verme olayı olarak tanımlanabilir. İş çizelgeleme problemine çeşitli yaklaşımlarda bulunulmuştur. Bu yaklaşımlar iki başlık altında sınıflandırılabilir. *Atölye tipi çizelgeleme, Akış tipi çizelgeleme*. Bu bölümde ilk olarak genel atelye tipi çizelgeleme problemi ve özellikleri tanıtılacak daha sonra da çalışmanın amacı zeki etmenlerin dinamik iş çizelgeleme probleminde kullanılabileceğini göstermek olduğundan dinamik atelye tipi çizelgeleme üzerinde durulacak ve geliştirilen sistem tanıtılacaktır.

9.1 Genel Atelye Tipi Çizelgeleme Problemi

Atelye-tipi üretim çizelgeleme probleminde; m tane $\{M_1, M_2, ... M_m\}$ makinada işlenmek üzere n tane $\{J_1, J_2, ... J_n\}$ iş mevcuttur. Her bir işin, her bir makinada sadece ve sadece bir kez işlem gördüğü varsayılır. Makinada işin işlenmesine operasyon denir ve i. işin j. makinadaki operasyonu O_{ij} olarak gösterilir. İşler, makinalarda belli bir sıra dahilinde işlenir ve bu sıra, *teknolojik kısıt, iş seyri* veya *rota* olarak adlandırılır. Genel atelye tipi üretim için teknolojik kısıtların oluşumuna dair hiçbir sınırlama yoktur. Her bir iş kendi işlem sırasına sahiptir ve diğer işlerin işlem sıralarından bağımsızdır.

Bununla birlikte bütün işler, aynı işlem sırasına sahip olduğunda özel bir durum ortaya çıkar. Böyle durumlarda problem akış-tipi çizelgeleme problemi olarak adlandırılır. Akış-tipi ve atelye-tipi çizelgeleme arasındaki fark daha sonra ayrıntılı olarak incelenecektir.

Her bir operasyon (O_{ij}), belli bir zaman içerisinde gerçekleştirilir. Bu zaman uzunluğu, *işlem zamanı* olarak adlandırılır ve p_{ij} olarak gösterilir. Basitleştirmek amacıyla, işi yürütmek üzere gereken makinayı ayarlama veya hazırlama için gerekli olan zamanın, yani, hazırlık zamanının ve işi makinaya taşımak amacıyla geçen zamanın da p_{ij} içinde bulunduğu varsayılır. Ayrıca, p_{ij} 'nin sabit ve önceden bilindiği de varsayılanlar arasındadır [French, 1982].

Dolayısıyla bu tezde operasyon zamanlarının, teslim tarihlerinin deterministik ve çizelgeleyici tarafından önceden bilindiği varsayılmıştır.

Yukarıdaki varsayımlara ek olarak, makinaların her zaman iş işlemeye müsait olduğu varsayılır. Fakat bu varsayım işler için geçerli değildir. Bazı işler, çizelgeleme başladıktan sonra bile işlenmek için elverişli durumda olmayabilir. İşlenmek üzere i. işin atelyede hazır olduğu zamana i. işin hazır zamanı denir ve r_i ile gösterilir.

Genel olarak problem, işlerin makinalardan geçtiği bir sıra bulmaktır. Bu sıra;

- Teknolojik kısıtlarla bağdaşır olmalı, yani olurlu bir çizelge olmalı ve
- Bazı performans kriterlerine göre optimal veya optimale yakın olmalıdır.

Yukarıda bazı varsayımlar problemin tanımı içinde verilmiştir. Bunlara ek olarak atelye tipi çizelgeleme problemini genelleştirmek için bazı tanımların yapılmasına gerek vardır. Bunlar şöyle sıralanır [French, 1982].

1. **Herbir iş bir bütündür** : İş farklı operasyonlardan oluşmasına rağmen, aynı işin iki operasyonu hiçbir şekilde aynı anda işlenemez. Böylece bazı pratik problemler tartışmanın dışına çıkartılabilir. Yani son ürün için montaj edilmeden önce alt parçaların aynı anda imal edildiği durumlar gibi.
2. **İş Bölme Yoktur** : Herbir operasyon, başladığı zaman, diğer operasyon o makinada başlatılmadan önce tamamlanmalıdır.
3. **Bir iş, her bir makinada bir tane olmak üzere, m tane farklı operasyona sahiptir.** İşin aynı makinada iki defa işlem görmesi olasılığı hesaba katılmaz.

4. **İş iptali söz konusu değildir** : Bütün işler tamamlanıncaya kadar işlenmelidir.
5. **İşlem zamanları çizelgeden bağımsızdır** : Burada iki şey varsayılmaktadır :
   - Hazırlık zamanı iş sırasından bağımsızdır. Yani, işe ait makinayı ayarlamak için gereken zaman en son işlem gören işten bağımsızdır.
   - Makinalar arasında işleri taşımak için gereken zaman ihmal edilmektedir.
6. **Ara stoğa izin verilir** : İşler bir sonraki makinanın boşalması için bekleyebilir. Bu önemli bir varsayımdır. Örneğin, çelik mil üretilirken demirin sıcak oluşundan dolayı iş bir sonraki operasyon için beklemek zorunda kalabilir.
7. **Makinanın her bir tipinden sadece bir tane vardır** : İşlerin işlenmesi esnasında aynı işi yapan birden fazla makinanın olmadığı varsayılır. Bu varsayım, "beklemek"ten kaçınmak için belli makinaların çoğaltılması durumunu ortadan kaldırır.
8. **Makinalar boş kalabilir.**
9. **Hiçbir makina, aynı anda birden fazla operasyonu işleyemez.**
10. **Makinalar asla bozulmaz ve çizelgeleme periyodu boyunca kullanıma hazırdır.**
11. **Teknolojik kısıtlar önceden bilinir ve sabittir.**
12. **Rassallık söz konusu değildir. Özellikle** ;
    - İşlerin sayısı bilinir ve sabittir,
    - Makinaların sayısı bilinir ve sabittir,
    - İşlem zamanları bilinir ve sabittir,
    - Hazırlık zamanları bilinir ve sabittir
    - Belli bir problemi tanımlamak için gereken her türlü nicel değerler bilinir ve sabittir.

9.1.1. Performans ölçütleri

Çizelgelemede amaçları ifade etmek her zaman kolay değildir. Amaçlar, çok karmaşıktır ve genellikle birbirleriyle bağdaşmazlar. Ancak, çizelgelemede ne derece

başarılı olunduğuna karar vermek için birtakım kriterleri tanımlamak gereklidir. Aksi takdirde çizelge hakkında yargıya varmak zorlaşır.

Bu durumlarda kararlaştırılmış teslim tarihlerine uymak tercih edilir. Aksi takdirde güvenilirlik kaybına uğranılacak ve de finansal ceza (maliyet) söz konusu olabilecektir. Bazen ise teslim tarihi çok önemli olmayabilir ve çizelgeleme periyodunun uzunluğu enazlanmak istenilebilir çünkü bütün işler tamamlandıktan sonra makinalar başka işler için kullanılabilir. Böylece makinaların aylak (boş) kalma zamanları enazlanmış olacaktır. Aylak makina, aylak sermaye yatırımı demektir. Bunlara ek olarak, stok maliyetleri enazlanmak istenilebilir. Bununla sadece son ürünlerin stoklanması maliyeti kastedilmemekte, aynı zamanda makinalar arasında işlenmek üzere bekleyen yarı işlenmiş parçaların stoklanma maliyetleri (ara stok maliyetleri) de kastedilmektedir.

Kısaca, amaçlar çok farklı, çeşitli ve birbirleriyle çelişir olabilir. Ama çizelgeleme probleminin etkinliğinin ölçülmesi için amaçların tanımlanması gereklidir. Performans ölçütlerini kesin matematik terimlerle tanımlamadan önce bazı tanımlar ve notasyonların verilmesi gerekir. Bunlar:-

r_i : i işinin hazır zamanı

p_{ij} : i işinin j.makinada işlem zamanı

a_i : J_i işinin teslim tarihi

q_i : J_i işine ait tahsisat. (Teslim tarihi ve hazır zaman arasında işlenme için hesaba katılan zaman periyodu.) $q_i = a_i - r_i$

W_{ik} : J_i işinin k. operasyonu için bekleme zamanı. $W_{i(k)}$, J_i işinin $M_{j(k-1)}$ makinasında tamamlanması ile M_k makinasında işleme başlaması arasında boşta kalan zamandır.

W_i : J_i'nin toplam bekleme zamanı. $W_i = \sum_{k=1}^{m} W_{ik}$

C_i : J_i nin tamamlanma zamanı (yani J_i işinin işlenmesinin bittiği zamandır.)

$$C_i = r_i + \sum_{k=1}^{m} (W_{ik} + P_{ij(k)})$$

F_i : J_i işinin akış zamanı (yani J_i işinin atelyede harcadığı zaman)

$$F_i = C_i - r_i$$

L_i : J_i nin gecikmesi (yani işin tamamlanma zamanı ile teslim tarihi arasındaki farktır.) $L_i = C_i - d_i$.

Eğer iş erken bitmiş ise, yani teslim tarihinden önce tamamlandıysa L_i negatiftir. Tam tersine eğer L_i pozitif ise iş geç kalmış demektir ve buna işin pozitif gecikmesi denir. Böylece yeni iki kriter tanımlanmaktadır.

T_i : J_i işinin pozitif gecikmesi. $T_i = \max\{L_i, 0\}$

E_i : J_i işinin erkenliği. $E_i = \max\{-L_i, 0\}$.

Yukarıdaki terminolojide bazı şaşırtıcı ifadeler olabilir. Zaman kavramı iki farklı manaya sahiptir. Zaman, hem bir zaman aralığını, hem de belli bir zaman noktasını ifade eder. Dolayısıyla hazır zaman, tamamlanma zamanı, zamandaki bir noktayı ifade ederken işlem zamanı, bekleme zamanı ve akış zamanı bir zaman aralığını gösterir.

Genellikle bu miktarların ortalamaları ve maksimum değerleri performans kriteri olarak alınır. Örneğin, $\overline{F}$ ortalama akış zamanı, C_{max} ise maksimum tamamlanma zamanını gösterir.

Diğer bir kriter olan M_j makinasındaki aylak zaman ise;

$$I_j = C_{max} - \sum_{i=1}^{n} P_{ij}$$

şeklinde ifade edilir.

Son olarak belli bir zamanda çeşitli durumlarda işlerin sayısını hesaplamak için bazı değişkenleri tanımlamak gerekirse;

$N_{W}(t)$: t zamanında işlenmeye hazır olmayan veya makinalar arasında bekleyen işlerin sayısı,

$N_p(t)$: t zamanında gerçekten işlenmekte olan işlerin sayısı,

$N_c(t)$: t zamanında tamamlanmış olan işlerin sayısı,

$N_u(t)$: t zamanıyla tamamlanacak olan işlerin sayısı.

Bu değişkenler arası ilişkiler şöyledir.

$$N_{W}(t) + N_p(t) + N_c(t) = n \qquad (\text{bütün } t\text{'ler için})$$

$$N_{W}(t) + N_p(t) = N_u(t) \qquad (\text{bütün } t\text{'ler için})$$

$$N_u(0) = n,$$

$$N_u(C_{max}) = 0.$$

$$\overline{N_u} = \frac{1}{C_{ma}} \int_0^{C_{max}} N_u(t)dt$$

Bu notasyonlara bağlı olarak çizelgeleme kriterleri üç sınıfta toplanabilirler:

1. **Tamamlanma zamanına dayanan kriterler**; *maksimum akış zamanı, maksimum tamamlanma zamanı, ortalama akış zamanı ve ortalama tamamlanma zamanı* ($F_{max}, C_{max}, \overline{F}$ ve $\overline{C}$) dir. Bu kriterlerin enazlanması istenir.

2. **Stok ve yararlanma maliyetlerine dayanan kriterler** ise; *işleme hazır olmayan veya makinalar arasında bekleyen işlerin ortalama sayısı* ($\overline{N_W}$), *tamamlanmamış olan işlerin sayısı* ($\overline{N_u}$) ve *tamamlanmış olan işlerin sayısı* ($\overline{N_c}$) enküçüklenmeye *herhangi bir zamanda halihazırda işlenmekte olan işlerin ortalama sayısını* ($\overline{N_p}$) ise enbüyüklenmeye çalışılır. Bunların yanında *ortalama* ($\overline{I}$) veya *maksimum makina aylak zamanı* (I_{max}), enküçüklenir.

3. **Teslim tarihine dayanan kriterler** ise şöyle özetlenebilir: Çizelge maliyeti genellikle hedef teslim tarihlerine ne kadar yaklaşıldığı ile ilgili olduğu durumda uygun performans kriterleri sırasıyla *ortalama gecikme* ($\overline{L}$), *maksimum gecikme*

(L_{max}), *ortalama pozitif gecikme* ($\overline{T}$), *maksimum pozitif gecikme* (T_{max}) olacaktır. Bütün bu kriterler dikkate alınınca enküçüklenmesi istenir.

9.2 Atölyede Dinamik Çizelgeleme

Bir atölyede çizelgeleme eylemi statik ve dinamik olmak üzere iki farklı alternatif yöntemle gerçekleştirilebilir. Statik çizelgeleme, atölyede işlenecek işlerin tümü başlangıçta bütün bilgileriyle beraber elde bulunuyorsa yapılan çizelgelemedir. Bu tarz çizelgeleme eyleminde işler seçilen performans ölçütüne göre işlem zamanları göz önüne alınarak sıralanırlar. İşlerin sıralaması her makina için teknolojik kısıtlar altında ayrı ayrı yapıldıktan sonra atölyede işler işlenmeye başlanır[Wan,1995].

Dinamik çizelgelemede ise yukarıda anlatılanların aksine işler atölyeye başlangıçta değil, imalat süreci boyunca gelmektedir. Bu yüzden işlerin detay bilgileri başlangıçta bilinmemektedir. İmalat süreci boyunca gelen işler önce ilgili bilgilerin tespitinden geçtikten sonra atölyeye alınırlar. Bir işin detay bilgileri daha önceki kısımlarda anlatıldığı gibi işleme, hazır olma, hazırlık vb. zamanları, teslim tarihi gibi bilgilerdir. Bilgiler belirlendikten sonra teknolojik sıraya göre iş ilgili makinaya gider. Burada kuyruk varsa kuyruğa eklenir, yoksa doğrudan ilk operasyon için makinada işlem görmeye başlar. Bu süreç boyunca daha önce ifade edilen ön kabullerden bazıları serbest bırakılabilir veya aynen kabul edilebilir. Bu durum sistemin hedef ve özelliklerine göre karara bağlanır. Baker (1974) tarafından gerektiğinde serbest bırakılabileceği belirtilen ön kabuller şunlardır:

1. Her iş kesin bir operasyon sırasına sahiptir,
2. Bir operasyon yalnızca bir makinada yapılabilir,
3. Atölyede her bir tip makinadan yalnızca bir adet bulunur,
4. İşlem zamanı ve teslim tarihleri peşin olarak bilinirler,
5. Hazırlık zamanları sıradan bağımsızdır.

Bir dinamik çizelgeleme sisteminde bu beş kabulden bir veya bir kaçı geçersiz yapılabilirler. Örneğin bir çok dinamik çizelgeleme sisteminde 4. kabul geçerli

değildir. Paralel makinaların olduğu atölyelerde 3. kabulün geçerli olması anlamsızdır. Bir çok çizelgeleme sisteminde makinaların hiç bozulmadığı varsayılır. Oysa normal şartlarda bu kabulün sürekliliği imkansızdır. Bunun gibi diğer kabuller de atölye özelliğine göre değerlendirilirler. Ancak burada dinamik çizelgelemenin ana karakterini belirleyen özellik işlerin imalat süreci boyunca atölyeye gelmesi ve çizelgelemenin de imalata paralel giden bir imalat benzetim sistemi olmasıdır. Benzetim, dinamik çizelgeleme boyunca atölyede meydana gelen olayları simule ederek ortamda işlerin atölye içi maceralarını ve makinaların işlerle olan etkileşimlerini canlandırır. Benzetim boyunca işler belli bir olasılık dağılımına göre atölyeye gelirler ve yine bir olasılık dağılımına göre işlem zamanı ve teslim tarihleri belirlenir. Dinamik çizelgeleme konusunda Evans (1993) bir çizelgeleme sistemi simulasyonunu sunmaktadır. Montezari ve Van Wassenhove (1990) bir esnek imalat sistemi için öncelik kurallarını karşılaştırmıştır. Wan (1995) çevrim dışı ve gerçek zamanlı çizelgeleme tekniklerini karşılaştırmaktadır. Kullanım kolaylığı açısından çevrim dışı çizelgelemeyi önermektedir.

Dinamik çizelgeleme çalışmalarını iki kategoride incelemek mümkündür. Birincisi sezgisel yöntemler olarak isimlendirilebilir. Bu yöntemler daha çok sezgisel algoritmalar halinde çözümler önerirler. Bunlardan Sun ve Lin (1994) dinamik çizelgeleme yapan sezgisel bir sistem çatısı ile çizelgeleme problemini bir kesikli olay optimal kontrol problemi olarak ele almış ve geriye doğru çizelgeleme isimli bir algoritmik yöntemle çözüm önermişlerdir. Özellikle dinamik çizelgelemenin içinde bulunduğu ortamla entegre olması hedeflemişlerdir.

Yih ve Thesen (1991), gerçek zamanlı çizelgelemeyi bir yarı markov karar model olarak ele alınmıştır. Bir esnek imalat sisteminin durum uzayını küçültmek ve daha rahat çözüme ulaşmak için stokastik modelini oluşturduktan sonra optimize ederek bir çözüm önermektedirler.

Chang ve Sullivan (1990), dinamik atölye ortamları için Giffler ve Thampson algoritmasını düzelterek sunmuşlardır. Böylece işleri azaltarak çizelgeleme imkanı bulmuşlardır.

Matsuura ve arkadaşları (1993), dinamik imalat ortamlarında sıralama ve iş atama teknikleri ile yapılan çizelgeleme eylemlerini karşılaştırdıktan sonra anahtarlı sıralama adlı bir yaklaşımla işleri çizelgelemeyi önermektedir. Bu yöntemle makina bozulmaları gibi ani gelişme ve değişmeleri değerlendirme olanağını yakalayabildiklerini belirtmektedirler.

Ishii ve Talavage (1991), esnek imalat sistemlerinde gerçek zamanlı çizelgeleme yapmak için kısa dönem çizelgeleri üreten bir algoritma sunmuşlardır.

Arzi (1995), çok hücreli bir esnek imalat sistemi için dinamik çizelgeleme yapacak iki aşamalı bir yöntem önermektedir. Dağıtılmış bir mimariye sahip bu sistem üretim kontrolünü her hücre için ayrı ayrı yapmaktadır. Bir kaç kriter açısından sistem performansı ölçülmüştür.

Sabuncuoğlu ve Hommertzheim (1992), bir esnek imalat sisteminde makinalar ve AGV için bir dinamik atama algoritması önermektedirler. Algoritma çeşitli öncelik kuralları, ilgili sistem yükü ve iş statülerini kullanarak bilgiyi hiyerarşik bir yapıya sokmaktadır. Bu bilgi seviyelerinin kullanarak atamaları makinalara veya AGV'ye yapmaktadır. Sonuçta bir çok kural ile yaptıkları karşılaştırmada ortalama akış zamanı ve ortalama pozitif gecikme kriterleri açısından iyi bir performans yakalamışlardır.

Donath ve Graves (1989), esnek montaj sistemleri adlı bir sistem önermektedir. Bu sistem çalışırken çizelgelemeyi göz önüne almaktadırlar. Montaj sistemi çoklu ürün üretiminde montaj hattı dengeleme yapmaktadır. Ele aldıkları çizelgeleme sistemi de gerçek zamanlıya yakın bir çizelgeleme yapmakta ve rota tayin etmektedirler.

Çalışmaların ikinci kategori ise yapay zeka tekniklerine dayanan çalışmalardır. Bu grubun bünyesinde çok sayıda yapay zeka tekniği kullanan çalışmadan bahsetmek mümkündür. Bu çalışmaların hemen hemen tümü çevrim dışı olarak tasarlanmış oluşturulmuş ve test edilmiştir. Özellikle Sim ve arkadaşları (1994), dinamik

çizelgelemeyi uzman sistem ve yapay sinir ağlarını birleşik kullanan bir sistemle gerçekleştirmeyi başarmışlardır. Atölyede dokuz makina bulunmakta, işler en az 3 en çok 6 operasyon gerektirmektedir. Sistem veri toplama, yapay sinir ağını eğitme ve benzetimle çizelgeleme olmak üzere üç aşamalı olarak çalışmaktadır.

Shaw ve arkadaşları (1992), bir esnek imalat sisteminde dinamik çizelgeleme yapacak bir sistem geliştirmişlerdir. Öncelik kurallarını uzman sistem yardımıyla seçen sistem, çizelgeleme boyunca kullandığı kuralları tümevarımsal öğrenme tekniğine göre üretmektedir.

Nakasuka ve Yoshida (1992), uzman sistem yardımıyla yaptıkları dinamik çizelgeleme sisteminde bilgi edinimini bir makina öğrenmesi algoritması ile gerçekleştirmektedirler. Yapılan çalışma "hangi öncelik kuralı ne zaman kullanılmalı?" sorusuna cevap aramaktadır. Çalışma bir esnek imalat sistemi ortamı içi geliştirilmiştir.

Taşgetiren (1996), çalışmasında dinamik çizelgelemede atölye ve iş durumuna göre öncelik kuralı belirlemede karar verici olarak uzman-yapay sinir ağı birleşimi bir mekanizma önermiştir. Bu yaklaşım Chang (1985)'ın yaklaşımına sistematik olarak benzemektedir. Taşgetiren, Chang'dan farklı olarak kural tabanlı uzman sistemle çizelgeleme eylemini atölye benzetimi ortamında yaptıktan sonra elde edilen verilerle yapay sinir ağını eğitmekte ve uzman sistemin çıkarım mekanizmasını YSA modeline öğretmektedir. Daha sonra uzman sistem YSA modeli ile paralel olarak çalışmakta ve birleşik olarak karar vermektedir.

9.3 Zeki Etmen Tabanlı Bir Dinamik Çizelgeleme Sistemi

Bu çalışmada zeki etmene dayalı bir sistem önerilmektedir. Bu çizelgeleme sisteminde dinamik bir atölye ortamında çizelgeleyici bir zeki etmen işleri çizelgelemektedir. İşlerin atölyeye gelişi ve atölyenin işleyişi zeki etmenden tamamen ayrıktır ve onun dışında gelişmektedir. Ortamı kontrol ederek gelen işleri seçilen

performans kriterini eniyileyecek uygun bir kurala göre makinalara atarak sonuçta bir iş çizelgesi elde etmek amaçlanmıştır.

Bu bölümde öncelikle işlerin geldiği atölye ve yapılacak çizelgelemenin ön kabul ve özellikleri belirlendikten sonra atölye ortamının benzetimi sunulacaktır. Daha sonra çizelgeleyici zeki etmen tanıtılarak *Q-III* öğrenme prosedürüne göre eğitilmesi ve sistemin performansı ile ilgili bilgiler sunulacaktır.

9.3.1 İşlerin çizelgeleneceği atölyenin özellikler

Dinamik çizelgeleme sisteminin geliştirilebilmesi için her şeyden önce içinde imalat yapılması tasarlanan atölyenin özellikleri ve kabullerinin bilinmesi gerekmektedir. Atölyenin özellikleri şu ön kabullerle belirlenmiştir:

1. Atölyede 9 farklı makina bulunmaktadır. Aynı tip makina yoktur.
2. Atölyede 5 tip operasyon sırasına sahip olan iş yapılabilmektedir. Bu işlerin operasyon süreleri standart değildir.
3. Bir iş tüm makinaları ziyaret eder.
4. İşlerin gelişler arası süreleri üstel dağılıma uyar. Hazır olma zamanları buna göre belirlenir.
5. İşlerin işlem zamanları üniform dağılıma göre belirlenir. Bu süreye transfer ve hazırlık zamanları dahildir.
6. İşlerin teslim tarihleri TWK yöntemine göre belirlenmektedir.
7. Bir makinada bir iş bir defa işlenir.
8. Operasyonlar sıraya göre yapılmaktadır. Önceki operasyon bitmeden sonrakine geçilemez.
9. Makinalar bir anda bir operasyonu yapabilirler.
10. Makinalar benzetim boyunca bozulmaz.
11. Makinalar daima üretime hazır haldedirler.
12. Makinalar boş kalabilirler.
13. İşler kuyruklar oluşturabilirler.
14. İş iptali söz konusu değildir.
15. Performans kriteri olarak *ortalama pozitif gecikme* ($\overline{T}$) alınmıştır.

9.3.2 Dinamik iş çizelgeleme için atölye benzetim ortamı

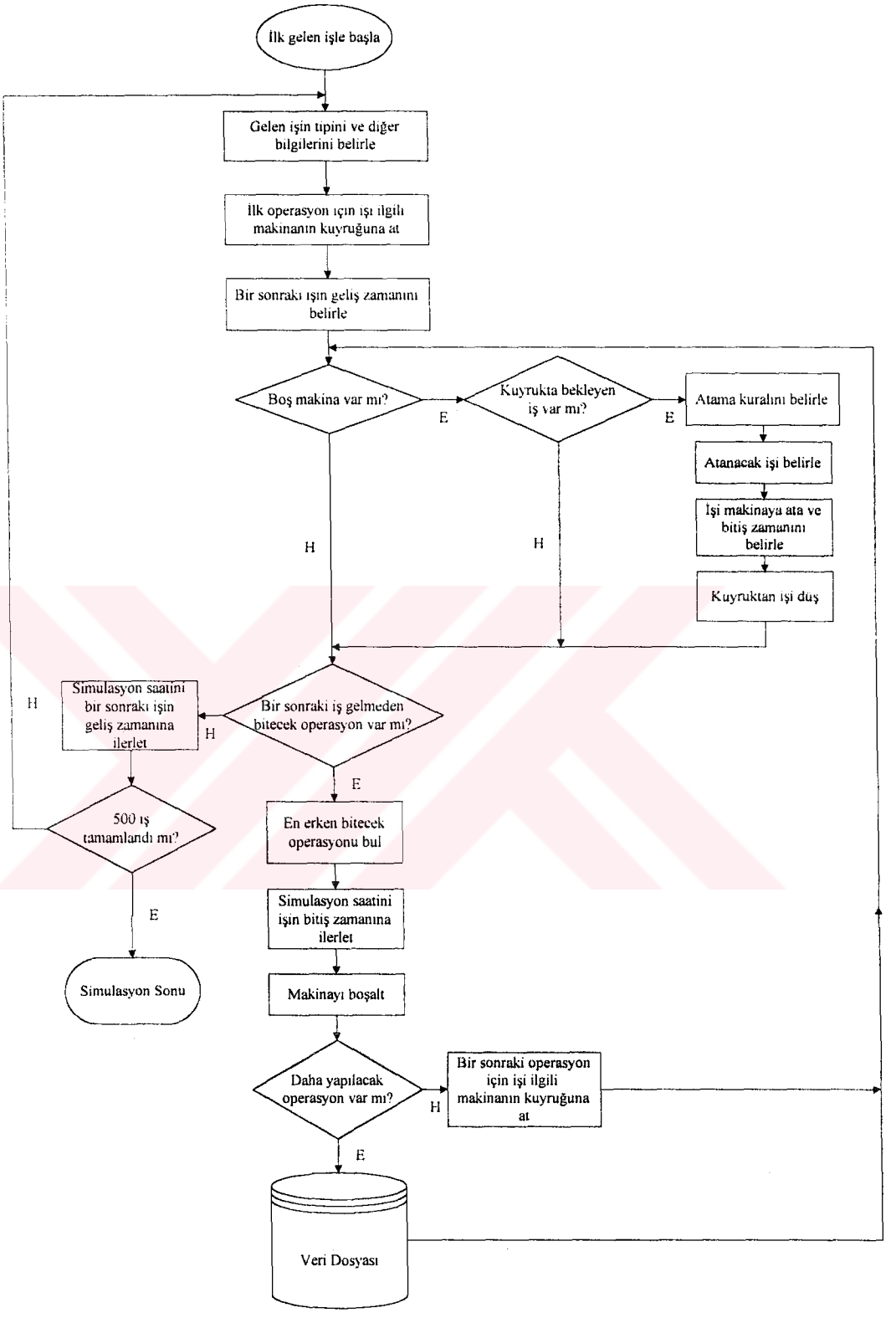
İş çizelgelemesi için değişik benzetim tekniklerinden yararlanılmaktadır. Atölyede meydana gelen faaliyetler, olaylar geliştikçe yeniden değerlendirildiklerinden ve olayların oluşumu bir süreklilikten çok bir kesikli karakter taşıdıklarından *kesikli olay artımlı benzetim* tekniği kullanılarak benzetim sistemi geliştirilmiştir. Sistemin *kesikli olay artımlı benzetim* tekniği ile tasarlanmasında detay bilgiler için Banks ve arkadaşları (1996)' dan yararlanılmıştır.

Atölye tipi dinamik iş çizelgeleme sistemi için geliştirilen benzetim modelinin genel akış şeması Şekil-9.1'de sunulmuştur. Benzetim atölyeye bir işin gelmesi ile başlamaktadır. Gelen işin hazır olma zamanı ile benzetim saati işlemeye başlamaktadır. İşin teknolojik bilgileri (hazır olma zamanı, işlem zamanı, operasyon sırası) belirlendikten sonra ilk operasyon için iş ilgili makinaya gönderilir. Bu aşamadan sonra atölyeye gelecek olan bir sonraki işin ne kadar süre sonra geleceği belirlenir. Bu süreye gelişler arası süre denir. Gelişler arası süre üstel dağılıma uygun olarak belirlenir. Bu şu eşitlikle belirlenir:-

$$t = 2 \times exp(4.5)$$

Burada t gelişler arası süre, ortalaması 4.5 olan üstel dağılımın 2 katı olarak üretilir. Gelişler arası süre tespit edildikten sonra makinalar kontrol edilir. Boş olan makinanın başında bekleyen iş veya iş kuyruğu varsa atanacak işin belirlenmesinde uygulanacak olan kuyruk disiplini (atama kuralı) tespit edilir. Sistemde seçilebilecek alternatif kurallar SPT (en kısa işlem süreli iş), COVERT [Baker (1974), Taşgetiren (1996)] ve CR [Singh, 1996] olmak üzere üç adettir. Alternatif kurallardan duruma en uygun olanı seçildikten sonra çalıştırılır ve kuyruktan makinaya atanacak bir iş belirlenir. İşin makinaya atanmasından sonra operasyon başlatılır, operasyonun bitiş zamanı belirlenir, ve kuyruktan bir iş düşülür. Şayet boş makina yoksa, boş makinaların başında bekleyen iş yoksa veya boş olan makinaların tümüne iş yüklendikten sonra atölyeye bir sonraki iş gelmeden bitecek işlerin olup olmadığı kontrol edilir. Var olan işler arasından ilk bitecek iş bulunarak benzetim saati söz konusu işin bitiş zamanına ilerletilir. Makina boşaltıldıktan sonra ilgili işin bu

makinadaki operasyonun son operasyon olup olmadığı kontrol edilir. Son operasyon olması durumunda iş bilgileri dosyaya kaydedilir ve iş atölyeden çıkarılır. Yapılan işlem son operasyon değilse bir sonraki operasyon için iş ilgili makinanın kuyruğuna gönderilir. Bu aşamadan sonra bir sonraki iş gelmeden bitecek işin olup olmadığı tekrar kontrol edilir. Var olması halinde biraz önce izlenen süreç tekrarlanır, yoksa benzetim saati bir sonraki işin atölyeye giriş zamanına ilerletilir ve en baştan beri anlatılan prosedür tekrarlanır. Benzetim 500 iş bitirilince durdurulur. Benzetim sonunda işlerin çizelgesi ve çizelgenin performans değeri elde edilir.



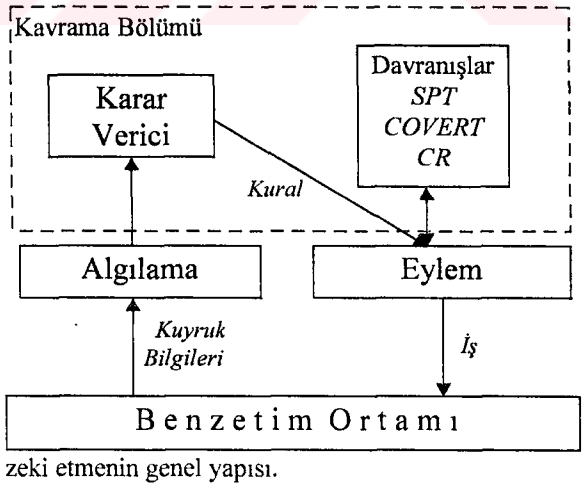
Şekil-9.1; Önerilen dinamik çizelgeleme benzetim sisteminin genel akış şeması.

9.3.3 Çizelgeleyici zeki etmen

Çizelgeleme işlevini yürütecek olan zeki etmen, dinamik atölye ortamı ile etkileşerek yapılacak işlemi ve işlemin özelliklerini tanıyacak ve çizelgelemeyi duruma en uygun kuralı kullanarak gerçekleştirecektir. Bu bölümde çizelgeleme yapacak olan zeki etmen önce yapısal olarak, sonra öğrenme eylemi açısından ele alınmıştır.

9.3.3.1 Çizelgeleyici zeki etmenin yapısı

Zeki etmen genel olarak davranışlar etrafında yapılaşmaktadır. Ana modülleri algılama, kavrama ve eylem birimleridir. Algılama bölümü yalnızca sisteme benzetim ortamından bilgi aktaran basit bir mekanizmadır. Kavrama bölümünde algılanan bilgileri değerlendiren bir karar mekanizması bulunmaktadır. Zeki etmen gelen bilgiler ışığında değerlendirme yaparak en uygun kuralı seçer. Eylem bölümünde ise sadece zeki etmenin davranışları sayılacak olan öncelik kuralları bulunmaktadır. Seçilen uygun kuralın çalıştırılması sonucunda benzetim ortamına atanacak iş bildirilir. Geliştirilen zeki etmen Şekil-9.2'de gösterilmektedir.

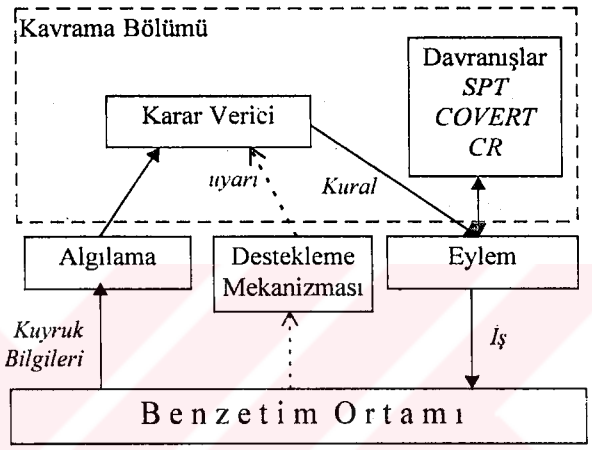


Şekil-9.2; Çizelgeleyici zeki etmenin genel yapısı.

9.3.3.2 Zeki etmenin öğretilmesi

Zeki etmen normal şartlarda öğrenmesini bitirmiş olarak çizelgeleme yapınca Şekil-9.2'deki gibi bir yapısal özellik gösterir. Aynı şekilde zeki etmen benzetim ortamında

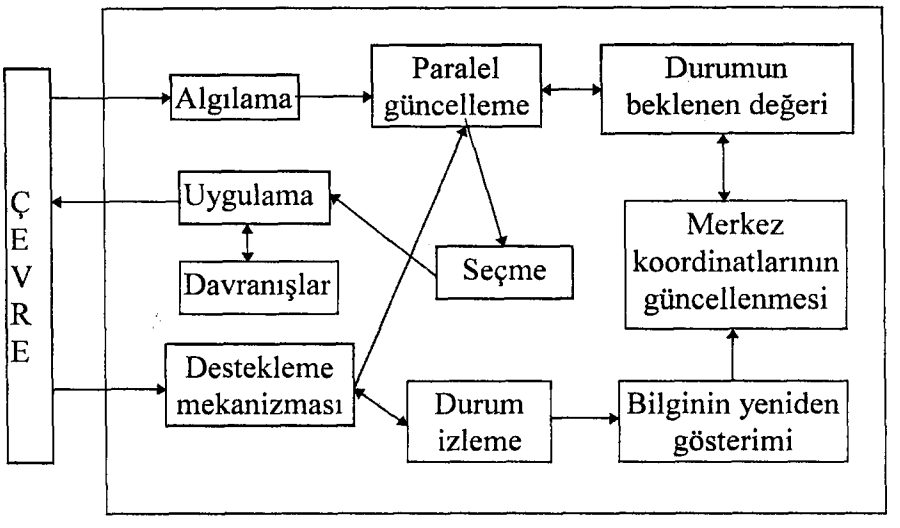
daima öğrenmeye hazır durumda olup yeni durumlar hakkında bilgiler edinebilmektedir. Zeki etmen sistemdeki ilişkileri öğrenmek üzere *Q-III* algoritması ile eğitilmiştir. *Q-III* algoritması ile zeki etmenin öğrenebilmesi için tasarlanan destekleme mekanizması, daha önceki bölümlerde olduğu gibi sistem dışında bulunarak zeki etmenin gerçekleştirdiği eylemleri değerlendirme ve öğrenmeyi yönlendirme görevini üstlenir. Destekleme mekanizması ve zeki etmen arasındaki etkileşim Şekil-9.3'te verilmiştir.



Şekil-9.3; Destekleme mekanizmasının zeki etmen ve benzetim ortamı ile etkileşimi

Zeki etmenin öğrenebilmesi için Şekil-9.3'de sunulan *zeki etmen-benzetim ortamı-destekleme mekanizması* üçlüsünün etkileşebilmesi gerekir. Bu etkileşime göre benzetim ortamından ilgili makinanın başında bekleyen işlerin oluşturduğu kuyruk bilgilerini zeki etmen algıladıktan sonra, duruma uygun öncelik kuralının seçilmesi için karar verici davranışlardan (alternatif öncelik kurallarından) birini seçer. Bununla ilgili eylem bölümünü ilgili kuralı çalıştırmak için uyarır. Eylem bölümü de ilgili kuralı aktive ettikten sonra seçilen işi benzetim ortamına iletir. Durumu ve yapılan işi izleyen destekleme mekanizması bir uyarı ile karar vericiyi uyarır. Karar verici de uyarıya göre bilgilerini güncelleştirir. Buradaki öğrenme süreci anlatılan bu işlem çevrimini tekrarlayarak benzetim süreci boyunca öğrenmesini gerçekleştirmektedir. Bu bölümde anlatılan *Q-III* algoritması süreci Bölüm 8'de anlatılan sistematiğin aynısıdır. Şekil-9.3'de verilen üçlü etkileşim şemasında öğrenmenin detayları sunulmamıştır. Öğrenme sürecinin geniş bir açılımı Şekil-9.4'te

verilmiştir. Burada *Q-III* algoritması işlerken modüller arasında nasıl bir ilişki ağı oluşturduğunu görmek daha kolaydır.



Şekil-9.4; Zeki etmenin *Q-III* algoritması ile öğrenmesi

Burada Bölüm 8'den farklı olarak üzerinde durulması gereken kısım, bu uygulamada bilginin yeniden gösterimi ve destekleme mekanizmasının notlandırmasıdır. Bu iki mekanizma birbirlerine paralel olarak bilgiyi değerlendirirler.

Q-III algoritmasının çalışmasında genelleştirmenin yapılabilmesi için Şekil-9.4'de de görülebileceği gibi bilgilerin yeni bir gösterimle ifade edilmesi gerekmektedir. Bu öbekleme analizinin yapılmasının getirdiği bir ihtiyaçtır. Dinamik çizelgelemenin yapılabilmesi için zeki etmenin gereksindiği bilgiler; kuyruk uzunluğu ve kuyruktaki işlerin ortalama gevşek zamanıdır. Daha önceki bölümde anlatıldığı gibi bu bilgilerin öğrenme sürecinde doğrudan kullanılması bu haliyle öğretici olmamaktadır. Dolayısıyla yeni bir uzaya taşımak gerekir. Kullanılan dönüşüm mekanizması Şekil-9.5'te verilmiştir. Şekilde verilen yaklaşımda bilgiler bir indekse dönüştürülmektedir. Bu indeks yardımıyla yeniden ifade edilen bilgiler bir başka uzaya taşınarak kritik sınır değerlerin öğrenmede karışıklığa meydan vermemesi sağlanmaktadır. `tmp` olarak ifade edilen indeks `x` ve `y` gibi iki boyuttan oluşmaktadır.

```
struct clust index(int toolk,float mslkt)
{
    register int i;
    float bias=0.0f;
    struct clust tmp;

    if ( toolk<5 && ((mslkt<40 && mslkt>0) || mslkt<-250))
       bias=-2.0f;
    else if (mslkt>80)  bias=5.5f;

     tmp.x=sin(mslkt)+bias;
     tmp.y=cos(mslkt)+bias;

     for(i=0;i<C;i++)
       tmp.dis[i]=0.0f;
   return tmp;
 }
```

Şekil-9.5; Bilgilerin yeniden gösterimi prosedürü

Destekleme mekanizması, öğrenme sürecinde zeki etmenin tercihlerini değerlendirerek puanlar vermektedir. Mekanizma *Q-III* algoritmasının öngördüğü şekilde -1 veya 1 şeklinde puanlandırma yapmaktadır. Puanlandırma prosedürü Şekil-9.6'da verilmiştir. Buna göre durum bilgileri ile davranışlar olumlu olarak eşleştirilmişse puan -1, aksi takdirde 1 olur.

```
puan(int toolk,float mslkt,int num)
 {
  int not=0;
  if ( toolk<5 && ((mslkt<40 && mslkt>0) || mslkt<-250))
     {
       if ( num==0 )
         not=-1;
       else
         not=1;
      }
    else
      {
          if ( mslkt>80)
            {
             if ( num==1 )
                not=-1;
             else
                not=1;
            }
           else
            {
            if ( num==2 )
               not=-1;
            else
               not=1;
             }
       }
     return(not);
  }
```

Şekil-9.6; Puanlandırma prosedürü

9.3.3.3 Çizelgeleyici zeki etmen ile iş çizelgeleme

Çizelgeleyici zeki etmen, her bir atama olayında karar verici olarak çizelgeleme benzetimine katılır. Benzetimi boyunca gelen kuyruk bilgilerine göre başlangıçta deneme-yanılma yoluyla kuralları uygular. Her uygulamada destekleme mekanizması durum bilgileri ile seçilen öncelik kuralını değerlendirir ve ilgili görüşünü bir puanla zeki etmene iletir. Bu puanın olumlu olması halinde zeki etmen durum ve kural eşleştirmesi hakkında hem geçici ve hem de kalıcı bir kanaat edinir. Olumsuz puan alması halinde ise yalnızca geçici kanaat edinir. Öğrenme süreci böylece devam ederek zeki etmen atölye ortamını tanımaya çalışır. Bu aşamadan sonra zeki etmen atölyede çizelgeleri tek başına herhangi bir müdahale olmadan yapabilecek duruma gelir.

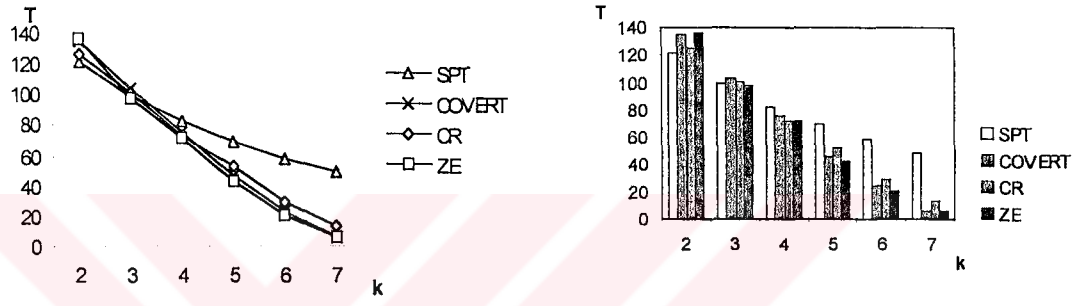
Yapılan çalışmada zeki etmen, bir çok defa denenmiş ve farklı parametre değerleri için eğitilmiştir. Daha sonra yapılan çizelgeleme işleminde ortalama pozitif gecikme değerini diğer öncelik kurallarından (SPT, COVERT ve CR) daha fazla minimize edebilmiştir. Bu öncelik kuralları benzetim boyunca tek başlarına kullanılma yerine kompozit olarak kullanılmışlardır. Yani durumlar değerlendirilerek bu üçünden en uygun olan seçilerek değişimli kullanılmışlardır. Değişimli kullanımı da zeki etmen öğrenmiş ve uygulamıştır. Denemelerde çizelgeleme periyodu boyunca öncelik kuralları farklı teslim tarihi darlık faktörü (k) değerleri için önce tek tek uygulanmış sonra da zeki etmenin kontrolünde değişimli uygulanmışlardır. Yukarıda belirtildiği gibi işlerin teslim tarihleri (dd) TWK (Total Work) yöntemi ile belirlenmektedir. Bu şu şekildedir:-

$$dd = r_i + k \times \sum_{i=1}^{n} p_i \quad k = 2,3,4,5,6,7$$

Burada k darlık faktörü, r_i hazır olma zamanı p_i ise operasyon zamanıdır. k'nın 2,3,4,5,6,7 değerleri için denemeler yapılmıştır. Bütün denemelerde zeki etmen daha minimum bir pozitif gecikme değeri yakalamıştır. Yapılan denemelerin sonuçları Tablo-9.1 ve Şekil-9.7'de sunulmaktadır. Tablo-9.1'de altı çizili değerler minimum değerlerdir.

Tablo-9.1; Zeki etmen çizelgeleme performansının diğer öncelik kuralları ile karşılaştırılması

Kural \k	*2*	*3*	*4*	*5*	*6*	*7*
SPT	**121.66**	99.00	82.48	69.40	58.63	49.61
COVERT	135.40	103.59	75.84	47.23	23.96	6.21
CR	125.76	101.15	72.93	52.73	29.76	14.04
ZE	136.44	**97.71**	**71.97**	**43.40**	**21.34**	**5.73**



Şekil-9.7; Öncelik kurallarının ortalama pozitif gecikmeye göre performansı

9.4 Sonuç

Bu bölümde *Q-III* algoritması ile öğretilen zeki etmenler yardımı ile dinamik atölye ortamında iş çizelgelemesi yapan bir sistem tanıtılmıştır. Sistemin asıl amacı *Q-III* öğrenme algoritması ile eğitilen zeki etmenlerin endüstriyel problemlere de uygulanıp yüksek performanslı sistemler tasarlanabileceğini göstermektir. Çizelgeleyici zeki etmen dinamik bir atölye ortamında rasgele gelen işleri rotalarına göre atölye ve iş durumlarını göz önünde bulundurarak başarılı bir çizelgeleme eylemi gerçekleştirmektedir. Sistem performansını daha da yükseltmek ve bu konuda yapısal bir öneri getirmek daima mümkündür. Bu bakımdan öğrenebilen zeki etmenlerin endüstriyel problemlere çok daha başarılı çözümler önerebileceği görülmektedir. Bu sistemin bir ileri adımı bilginin yeniden gösterimini bulanık mantık teknolojisinden yararlanarak daha sağlıklı gerçekleştirmek olabilir.

BÖLÜM 10 SONUÇ

Bu tez çalışmasında zeki etmenlerin öğrenme kabiliyetlerinin geliştirilmesi ele alınmıştır. Önce zeki etmen kavramı üzerinde durulmuş ve bir genel literatür taraması yapılmıştır. Daha sonra öğrenme konusunda ve özellikle destekli öğrenme sistemleri üzerinde durularak yapılan çalışmalar hakkında bir literatür taramasına verilmiştir. Bunların devamında dijit oyunu oynamak üzere bir zeki etmen tasarlanmış, çok üzerinde durulan öğrenme algoritmalarından *Q-öğrenme* algoritması ile eğitilmeye çalışılmıştır. Bu çalışmalar sırasında söz konusu algoritmanın bazı problemleri gözlenmiş ve bunların önlenmesi için çalışmalar yapılmış olup üç kademeli bir iyileştirme önerilmiştir. İyileştirilen algoritma ile imalat sistemlerinin önemli problemlerinden biri olan dinamik atölyede iş çizelgeleme yapacak bir zeki etmen eğitilmiştir. Bu zeki etmenin performansını diğer klasik kurallardan daha iyi olduğu gözlenmiştir.

Bu tezin bilim ve teknolojiye katkılarını şöyle sıralamak mümkündür.

1. *Q-öğrenme* algoritmasının yerel optimum probleminin cezalandırma fonksiyonu yardımı ile çözülmesi,
2. *Q-öğrenme* algoritmasının yavaş yakınsama probleminin paralel güncelleştirme kuralı yardımı ile çözülmesi,
3. *Q-öğrenme* algoritmasının yapısal kredilendirme problemine yeni yaklaşımlar getirilmesi;

- Çevrim dışı genelleştirme yaklaşımı,
- Çevrim içi genelleştirme yaklaşımı.

4. Dinamik atölye ortamında iş çizelgelemenin çevrim içi öğrenebilen zeki etmenler yardımı ile yeniden ele alınması ve bu yöntemle çok daha iyi performanslı çizelgeleme sistemlerinin geliştirilebileceğinin gösterilmesi.

Q-öğrenme algoritması ile ilgili iyileştirme çalışmaları Tablo-10-1'de özet olarak sunulmuştur. Bu tabloda gelişim adımları boyunca hangi problemlerin çözüldüğünü hangilerinin yeniden baş gösterdiğini de içerilmektedir.

Tablo-10.1; Tez boyunca *Q-öğrenme* algoritmasında gerçekleştirilen gelişim

	Q-I	*Q-II*	*Q-III*
Beklenen problem	• yerel optimum	• yerel optimum • yavaş yakınsama	• genelleştirme
Öneri	• cezalandırma fonksiyonu	• paralel güncelleştirme	• paralel güncelleştirme • genelleştirme (öbekleme)

Bu çalışmanın yapılan katkıları yanında kapısını araladığı yeni araştırma imkanlarından da bahsetmek mümkündür:

1. *Q-öğrenme* için önerilen yapısal kredilendirme sisteminin daha etkin olabilmesinin araştırılması,
2. *Q-III* algoritmasının gelecekte üzerinde durulabilecek tarafı, kazanılan puanın olumlu olması durumunda kalıcı deneyim kabul etmesinin yanında olumsuz puanların söz konusu olduğu durumlarda da kalıcı bir deneyim kazanabilmesinin başarılması,

3. İş çizelgelemede performansın sürekli arttırılmasını esas alan bir çizelgeleyici zeki etmen tasarlanmasının mümkün olup olmadığı,

4. Zeki etmene dayalı dinamik çizelgeleme sistemi tasarlanırken öğrenme aşamasında ihtiyaç duyulan *bilginin yeniden gösterimi* ve *destekleme mekanizması* prosedürlerinin geliştirilmesinde bulanık mantık kullanarak öğrenme etkinliğinin ve veriminin arttırılmasının sağlanıp sağlanamayacağının araştırılması.

KAYNAKLAR

1. **Agre, P., Chapman. D.,** (1987) "PENGI: An implementation of a theory of activity", In: *Proceedings of the Sixth National Conferance on Artificial Intelligence (AAAI-87),* pp:268-272, Seattle, WA.
2. **Anderson, C.W.**, (1987), "Strategy learning with multilayer connectionist representations", In: *Proceedings of International Workshop on Machine Learning*, pp:103-114.
3. **Anderson, T.L., Donath, M.,** (1990), "Animal behavior as a paradigm for developing robot autonomy", *Robotics and Autonomous Systems*, 6, pp:145-168.
4. **Arkin, R.C.,** (1990), "Integrating behavioral, perceptual, and world knowledge in reactive navigation", *Robotics and Autonomous Systems*, 6, pp:105-122.
5. **Arzi, Y.,** (1995), "On-line scheduling in a multi-cell flexible manufacturind systems", *Int. Jour. of Production Research*, 33(12), pp:3283-3300.
6. **Asada, M., Noda, S., Tawaratsumida, S., Hosoda, K.,** (1996) "Purpusive behavior acquisition for a real robot by vision-based reinforcement learning", *Machine Learning*, 23, pp:279-303.
7. **Basye, K., Dean, T., Kaelbling, L.P.,** (1995), "Learning dynamics: system identification for perceptionally challenged agents", *Artificial Intelligence*, 72, pp:139-171.
8. **Becket, W., Bedler, N.I.,** (1993), "Integrated behavioral agent architecture", In: *Proceedings of the Third Conferance on Comp. Gener. Forces and Behavioral Representation, 17-19 March 1993,* pp:57-68, *Orlando, Florida, USA.*
9. **Baker, R.K.,** (1974), *Introduction to sequencing and scheduling*, John Wiley&Sons Inc., Toronto.

10. **Banks, J., Carson, J.S.II, Nelson, B.L.**, (1996), *Discrete-event system simulation*, Prentice-Hall, NewJersey.

11. **Barto, A.G., Bradtke, S.J., Singh, S.P.,** (1995) "Learning to act using real-time dynamic programming", *Artificial Intelligence*, 72, pp: 81-138.

12. **Beer, R.D., Chiel, H.J., Sterling, L.S.,** (1990), "A biological perspective on autonomous agent design", *Robotics and Autonomous Systems*, 6, pp:169-186.

13. **Beer, R.D.,** (1995), "A dynamical systems perspective on agent-environment interaction", *Artificial Intelligence*, 72, pp: 173-215.

14. **Bradtke, S.J., Barto, A.G.,** (1996) "Linear least-squares algorithms for temporal differance learning",*Machine Learning*, 22, pp: 33-57.

15. **Bresslof, P.C., Weir, D.J.,** (1991), "Neural Networks",*GEC Jour. of Research*, 8(3), pp:151-169.

16. **Brooker, L.B., Goldberg, D.E., Holland, J.H.,** (1989), "Classifier systems and genetic algorithms", *Artificial Intelligence*, 40, pp: 235-282.

17. **Brooks, R.,** (1986), "A robust layered control system for a mobile robot", *IEEE Journal of Robotics and Automation*, RA-2(1), pp:14-23.

18. **Brooks, R.,** (1990a), "Elephant don't play chess", *Robotics and Autonomous Systems*, 6, pp:3-15.

19. **Brooks, R.,** (1990b), "The behavior language; user's guide", *A.I. Memo 1227*, April-1990.

20. **Brooks, R.,** (1991a), "Intelligence without reason", In *Proceedings of the 1991 International Joint Conference on Artificial Intelligence*, pp:569-595.

21. **Brooks, R.,** (1991b), "Intelligence without representation", *Artificial Intelligence*, 47, pp:139-159.

22. **Brooks, R.,** (1991c), "New approaches to robotics", *Scieincce*, 253, pp:1227-1232.

23. **Castillo, D.,** (1991), *Toward a paradigm for simulating intelligent agents*, PhD Thesis, University of Central Florida, USA.

24. **Cengeloglu, Y., Khajenoori, S., Linton, D.,** (1994), "DYNACLIPS (DYNAmic CLIPS): A dynamic knowledge exchange tool for intelligen agents", In: *Proc. of The 3rd CLIPS Conference*, September-1994, NASA, USA.

25. **Chang, F.-C.,** (1985), *A knowledge-based real-time desicion support system for job shop scheduling at the shop floor level*, PhD Thesis, The Ohio State University, Ohio.

26. **Chang, Y.L., Sullivan, R.S.,** (1990), "Schedule generation in a dynamic job shop", *Int. Jour. of Production Research*, 28(1), pp:65-74.

27. **Chapman, D., Kaelbling, L.P.,** (1991), "Input generalization in delayed reinforcement learning", In: *Proceedings of IJCAI-91.*

28. **Chi, S.-D., Zeigler, B.P.,** (1994), "Hierarchical model-based diagnosis for high autonomy systems", *Jour. of Intelligent and Robotic Systems*, 9, pp:193-207.

29. **Dayan, P.,** (1992) "The convergence of TD(λ) for general λ", *Machine Learning*, 8 pp:341-362.

30. **Dayan, P., Sejnowski, T.J.,** (1994) "TD(λ) converges with probablity 1", *Machine Learning*, 14, pp:295-301.

31. **Donath, M., Graves, R.J.,** (1989), "Flexible assembly systems: test results for an approach for near real-time scheduling and routing of multiple products", *Int. Jour. of Production Research*, 27(2), pp:215-227.

32. **Dorigo, M., Colombetti, M.,** (1994) "Robot shaping:Developing autonomous agents through learning", *Artificial Intelligence*, 71(2), pp:321-370.

33. **Dorigo, M.,** (1995), "ALECSYS and AutonoMouse: learning to control a real robot by distributed classifier systems", *Machine Learning*, 19, pp:209-240.

34. **Dorigo, M.,** (1993), "Genetics-based machine learning and behavior-based robotics: a new synthesis", *IEEE Trans. On Systems, Man, and Cybernetcis*, 23(1), pp:141-153.

35. **Düğenci, M.,** (1996), *Genetik algoritmalarla permütasyon tipi iş sıralama*, Yüksek Lisans Tezi, Sakarya Üniversitesi Fen Bilimleri Enstitüsü, Adapazarı.

36. **Emami, M.R., Turksen, I.B., Goldenberg, A.A.,** (1996), "An improved fuzzy modeling algorithm, Part II: System identification", In: *Proceedings of NAFSIP '96*, 20-21 June 1996, U.C. Berkley, USA.

37. **Evans, J.R.,** (1993), *Applied production and operation managment*, West , USA.

38. **Ferguson, I.A.,** (1995), "On the role of BDI modeling for integrated control and coordinated behavior in autonomous agents", *Applied Artificial Intelligence*, 9, pp: 421-447.

39. **French, S.,** (1982), *Sequencing and scheduling: An introduction to the mathematics of the job-shob,* John Wily & Sons, NewYork.

40. **Gams, M., Hribovsek, B.,** (1996), "Intelligent-personel-agent interface for operating systems", *Applied Artificial Intelligence*, 10, pp: 353-383.

41. **Grenfenstette, J.J., Ramsey, C.L., Schultz, A.C.,** (1990), "Learning sequential decision rules using simulation models and competition", *Machine Learning*, 5, pp:355-382.

42. **Hayes-Roth, B.,** (1995), "An architecture for adaptive intelligent systems", *Artificial Intelligence*, 72, pp:329-365.

43. **Hayes-Roth, B.,** (1990), "Architectural foundations for real-time performance in intelligent agents", *The Real-Time Systems*, 2, pp:99-125.

44. **Heiss, M.,** (1994), "Reinforcement learning or tracking of input-output maps", *Applied Artificial Intelligence*, 8, pp: 483-496.

45. **Heger, M.,** (1996) "The loss from imperfect value functions in expectation-based and minimax-based tasks", *Machine Learning*, 22, pp:197-225.

46. **Ishii, N., Talavage, J.J.,** (1991), "A transient-based real-time scheduling algorithm in FMS", *Int. Jour. of Production Research*, 29(12), pp:2501-2520.

47. **Kaelbling, L.P.**, (1996) "Introduction", *Machine Learning*, 22, pp:7-9.

48. **Kaelbling, L.P.**, (1994a) "Assocative reinforcement learning: functions in *k*-DNF", *Machine Learning*, 15, pp:279-298.

49. **Kaelbling, L.P.**, (1994b) "Assocative reinforcement learning: agenerate and test algorithm", *Machine Learning*, 15, pp:299-319.

50. **Kaelbling, L.P., Rosenchein, S.J.,** (1990), "Action and planning in embedded agents", *Robotics and Autonomous Systems*, 6, pp:35-48.

51. **Koeing, S., Simmson, R.G.,** (1996) "The effect of representation and knowledge on goal-directed exploration with reinforcement learning", *Machine Learning*, 22, pp:227-250.

52. **Koeing, S., Simmson, R.G.,** (1993) "Complexity analysis of real-time reinforcement learning",In: *Proc. of the 11th national Conference on Artificial Intelligence*, 11-15 July 1993, Washington, USA.

53. **Larid, J.E., Yger, E.S., Hucka, M.,** (1990), "Robo-Soar:An integration of external interaction, planning, and learning using Soar", In: *Machine Learning for Autonomous Agents, Ed:W.V.Velde.*

54. **Lashkari, Y., Metrel, M., Maes, P.,** (1994), "Collobrative ınterface agents", In: *Proceedings of The 12th National Conference on Artificial Intelligence*, 1, pp:444-449, AAAI Press, August-1994, USA.

55. **Lenger, D., Rosenblatt, J., Hebert, M.,** (1994), "A behavior-based system for off-road nevigation", *IEEE Trans. On Systems, Man, and Cybernetcis*, 10(6), pp:776-783.

56. **Levinson, R.,** (1996), "General game-playing and reinfprcement learning", *Computational Intelligence*, 12(1), pp:155-176.

57. **Liepins, G.E., Hilliard, M.R., Palmer, M., Rangarajan, G,** (1991) "Credit assignment and discovery in classifier systems", *Int. Jour. of Intelligent Systems*, 6(1), pp:55-69.

58. **Liljenström, H.,** (1995) "Autonomous learning with complex dynamics", *Int. Jour. of Intelligent Systems,* 10, pp:119-153.

59. **Lin, C.-T., Lee, C.S.G.,** (1996), *Neural and fuzzy systems: A neuro-fuzzy synergism to intelligent systems*, Prentice-Hall, New Jersery.

60. **Lin, C.T., Lee, C.S.G.,** (1994), "Reinforcement structure/parameter learning for neural-network-based fuzzy logic control systems", *IEEE Trans. on Fuzzy Systems,* 2(1), pp: 46-63.

61. **Lin,L.J.,** (1992) "Self-improving reactive agents based on reinforcement learning, planning and teaching", *Machine Learning*, 8, pp:293-321.

62. **McDonald, M.A.F., Hingston, P.,** (1997), "Discounted reinforcement learning does not scale", *Computational Intelligence*, 13(1), pp: 126-143.

63. **Maclin, R., Shavlik, J.W.,** (1996) "Creating advice-takimg reinforcement learning", *Machine Learning*, 22, pp:251-281.

64. **Maes,P., Brooks, R.,** (1990), "Learning to coordiate behaviors", *Proceedings of the 8th AAAI*, pp:796-802, Morgan Koffmen.

65. **Maes, P.,** (1991) "The agent network architecture (ANA)", *SIGART Bulletin,* 2(4), pp:115-120.

66. **Maes, P.,** (1994a) "Social interface agents: acquiring competence by learning from user and other agents", In: *Proceedings of The 1994 AAAI Spring Symposium on Software Agents.*

67. **Maes, P.,** (1994b) "Agents that reduce work and information overload", *Communications of The ACM,* 37(7), pp:31-40.

68. **Maes, P.,** (1995) "Artificial life meets entertainment: Lifelike autonomous agent", *Communications of The ACM,* 38(11), pp:31-40..

69. **Mahadevan, S., Connell, J.,** (1992), "Automatic programming of behavior-based robots using reinforcement learning", *Artificial Intelligence*, 55, pp:311-365.

70. **Mahadevan, S.,** (1996) "Average reward reinforcement learning: foundations, algorithms, and empirical results", *Machine Learning*, 22, pp:159-195.

71. **Matsuura, H., Tsubone, H., Kanezashi, M.,** (1993), "Sequencing, dispatching and switching in a dynamic manufacturing environment", *Int.Jour. of Production Research*, 31(7), pp:1671-1688.

72. **Montazeri, M., van Wassenhove, L.N.,** (1990), "Analysis of scheduling rules for an FMS", *Int. Jour. of Production Research*, 28(4), pp:785-802.

73. **Moriarty, D.E., Miikkulainen, R.,** (1996), "Efficient reinforcement learning through symbiotic evolution", *Machine Learning*, 22, pp:11-32.

74. **Nakasuka, S., Yoshida, T.,** (1992) , "Dynamic sheduling system utilizing machine learning as aknowledge aquisition tool", *Int. Jour. of Production Research*, 30(2), pp:411-431.

75. **Öztemel, E., Aydın, M.E.,** (1997a), "Zeki etmenlerde öğrenme: *Q-I* algoritması", In: *New Trends in Artificial Intelligence and Neural Networks, ed: T. Çiftçibaşı, M. Karaman, V. Atalay*, pp:200-206, EMO, Ankara.

76. **Öztemel, E., Aydın, M.E.,** (1997b), "*Q-II*: An improved learning algorithm for intelligent agents", In: *Proceedings of 11th European Simulation Multiconference, ed: A.R. Kaylan & A. Lehmann*, 1-4 June 1997, pp:275-279, Istanbul.

77. **Pal, N.R., Bezdek, J.C.,** (1995), "On cluster validity for the Fuzzy c-Means model", *IEEE Trans. On Fuzzy Systems,*, 3(3), pp:371-379.

78. **Peng , J., Williams, R.J.,** (1996) "Incremental multi-step *Q*-learning", *Machine Learning*, 22, pp:283-290.

79. **Piggott, P., Sattar, A.,** (1994) "Reinforcement learning of iterative behaviour with multiple sensors", *Journal of Applied Intelligence*, 4, pp:351-365.

80. **Pollack, M.E.**, (1992), "The uses of plans", *Artificial Intelligence*, 57, pp:43-68.

81. **Rosenbloom, P.S., Larid, J.E., Newell, A., McCarl, R.,** (1991), "Apreliminary analysis of the Soar architecture as a basis for general intelligence", *Artificial Intelligence*, 47, pp:289-325.

82. **Ross, T.J.,** (1995), *Fuzzy logic with engineering applications*, McGraw-Hill, New York.

83. **Russell, S.J., ve Norvig, P.,** (1995), Artificial Intelligence: A modern approach, Prentice-Hall International Inc.,New Jersey,USA.

84. **Sabuncuoğlu, I., Hommertzhem, D.L.,** (1992), "Dynamic dispatching algorithm for scheduling machines and automated guided vehicles in a flexible manufacturing system", *Int. Jour. of Production Research*, 30(5), pp:1059-1079.

85. **Schapire, R.E., Warmuth, M.K.,** (1996) "On the worst case analysis of temporal difference learning algorithms", *Machine Learning*, 22, pp:95-121.

86. **Shardanand, U., Maes, P.,** (1995), "Social information filtering: Algorithms for automating 'world of mouth'", In: *Proceedings of Chi-95 Conference*, May-1995, Denver, USA.

87. **Shaw, M.J., Park, S., Raman, N.,** (1992), "Intelligent scheduling with machine learning capabilities: The induction of scheduling knowledge", *IIE Transection* 24(2), pp: 156-168.

88. **Shoham, Y.,** (1993), "Agent-oriented programming", *Artificial Intelligence*, 60, pp:51-92.

89. **Sim, S.K., Yeo, K.T., Lee, W.H.,** (1994), "An expert neural network system for dynamic job shop scheduling", *Int.Jour. of Production Research*, 32(8), pp:1759-1773.

90. **Simmson, R.G.,** (1994), "Structured control for autonomous robots", *IEEE Trans. On Robotics and Automations,*, 10(1), pp:34-43.

91. **Singh, S.P.,** (1992) "Transfer of learning by composing solution of elemantal sequential tasks", *Machine Learning*, 8, pp:323-339.

92. **Singh, S.P., Sutton, R.S.,** (1996) "Reinforcement learning with replacing eligibility traces", *Machine Learning*, 22, pp:123-158.

93. **Singh, N.,** (1996), *Systems aprooach to computer integrated design and Manufacturing,* John Wily & Sons,Canada.

94. **Spronck, P.H.M., Kerckhoffs, E.J.H.,** (1997), "Using genetic algorithms to design neural reinforcement controller for simulated plants", In: *Proceedings of 11th European Simulation Multiconference, ed: A.R. Kaylan & A. Lehmann*, 1-4 June 1997, pp:292-299, Istanbul.

95. **Steels, L.,** (1994) "The artificial life roots of artificial intelligence", *Artificial Life Journal,* 1(1), MIT Press, Cambridge.

96. **Sun, D., Lin, L.,** (1994), "A dynamic job shop scheduling framework: A backward approach", *Int. Jour. of Production Research*, 32(4), pp:967-985.

97. **Sutton, R.S.,** (1992) "Introduction: The challenge of reinforcement learning", *Machine Learning*, 8 pp:225-227.

98. **Sutton, R.S.,** (1988) "Learning to predict by the methods of temporal differances", *Machine Learning*, 3, pp:9-44.

99. **Szczerbick, E.,** (1994), "Model-based generation of knowledge for autonomous systems", *Int. Jour. of Systems Science*, 25(3), pp:453-472.

100. **Taşgetiren, M.F.,** (1996), *Atölye tipi çizelgeleme problemi için bir uzman-yapay sinir ağı modeli*, Doktora Tezi, İstanbul Üniversitesi Sosyal Bilimler Enstitüsü Üretim Anabilimdalı, İstanbul.

101. **Tesauro, G.,** (1992) "Practical issues in temporal differance learning", *Machine Learning*, 8, pp:257-277.

102. **Tyrrell, T.,** (1993) Computational mechanism for action selection, PhD Thesis, University of Edinburg, UK.

103. **Tsitsiklis, J.N.,** (1994 "Asynchronous stochastic approximation and Q-learning", *Machine Learning*, 16, pp:185-202.

104. **Toomey, C., Mark, W.,** (1995), "Satellite image dissemination via software agents", IEEE Expert, October 1995, pp:44-51.

105. **Uthmann, T.,** (1997), "Modeling different aspects of intelligent behaviour in a multi-agent environment", In: *Proceedings of 11th European Simulation Multiconference, ed: A.R. Kaylan & A. Lehmann*, 1-4 June 1997, pp:267-274, Istanbul.

106. **Wan, Y.W**, (1995), "Which is better, off-line or real-time scheduling?", *Int. Jour. of Production Research*, 33(7), pp:2053-2059.

107. **Watkins, C.J.C.H.**, (1989) "Learning from delayed rewards", PhD Thesis, Cambridge Univ. UK.

108. **Watkins,C.J.C.H., Dayan, P.,** (1992) "Technical note: *Q-learning*" *Machine Learning*, 8, pp:279-292.

109. **Whitehead, S.D., Lin, L.J.,** (1995) "Reinforcement learning of non Markov desicion processes", , 73, pp:271-306.

110. **Whitley, D., Dominic, S., Das, R., Anderson, C.W.,** (1993), "Genetic reinforcement learning for neurocontrol problems", *Machine Learning*, 13, pp:259-284.

111. **Williams, R.J.,** (1992) "Simple statisitcal gradient-following algoritms for connectionist reinforcement learning", *Machine Learning*, 8 pp:229-256.

112. **Wooldridge, M., Jennings, N.R.,** (1995) " Intelligent agent: theory and practice", *The Knowledge Engineering Review*, 10(2), pp:115-152.

113. **Yih, Y., Thesen, A.,** (1991), "Semi-Markov desicion models for real-time scheduling", *Int. Jour. of Production Research*, 29(11), pp:2331-2346.

114. **Zapata, R., Lepinay, P., Thompson, P.,** (1994), "Reactive behaviors of fast mobil robots", *Jour. of Robotic Systems*, 11(1), pp:13-20.

ÖZGEÇMİŞ

M. Emin AYDIN, 1968 yılında Kars-Kağızman'da doğdu. İlk ve orta öğrenimini memleketi olan Kars'ta tamamladı. 1987 yılında girdiği İ.T.Ü. Sakarya Mühendislik Fakültesi Endüstri Mühendisliği Bölümü'nü 1991 yılında bitirdi. Şubat 1992'de İ.Ü. Sosyal Bilimler Enstitüsü Üretim Anabilim Dalında başladığı yüksek lisans eğitimini, Mayıs 1994'te bitirdi. Eylül 1994'ten beri Sakarya Üniversitesi Mühendislik Fakültesi Endüstri Mühendisliği Anabilim Dalı'nda doktora öğrencisidir. Ayrıca Temmuz 1992'den beri Sakarya Üniversitesi Mühendislik Fakültesi Endüstri Mühendisliği Bölümü'nde araştırma görevlisi olarak çalışmaktadır. Evli ve dört çocuk babasıdır.