YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YAPAY ZEKÂ İLE BÜTÜNLEŞİK SİMÜLASYON ORTAMLARI ve BİR KAVŞAKDAKİ TRAFİK IŞIKLARININ ANALİZİ İÇİN ZEKİ BİR SİMÜLASYON ORTAMININ TASARLANMASI

Endüstri Yük.Müh. Tufan DEMİREL

F.B.E. Endüstri Mühendisliği Anabilim Dalı Endüstri Mühendisliği Programında Hazırlanan

DOKTORA TEZİ

Tez Savunma Tarihi	**: 22 Şubat 1999**
Tez Danışmanı	**: Prof. Yaşar Baki CENGİZ (YTÜ)**
Jüri Üyeleri	**: Prof.Dr. Güneş GENÇYILMAZ (İÜ)**
	: Doç.Dr. Mehmet TANYAŞ (İTÜ)

Prof. Yaşar B. Cengiz 12.3.1999

İSTANBUL, 1999

İÇİNDEKİLER

SİMGE LİSTESİ

YS_i	Kavşaktaki i. yol için yeşil yanma süresi
KS_i	Kavşaktaki i. yol için kırmızı yanma süresi
GAS_i	Kavşaktaki i. yola bir saatlik peryot boyunca gelen araç sayısı
m	Kavşaktaki yol sayısı
$ÇS$	Çevrim süresi
X	Rassal değişkenler
D	Karar değişkenleri
Y	Çıktı değişkenleri

KISALTMA LİSTESİ

S	Simülasyon
US	Uzman Sistem
YZ	Yapay Zekâ
NYP	Nesneye Yönelik Programlama
YA	Yöneylem Araştırması
EİS	Esnek İmalat Sistemleri

ŞEKİL LİSTESİ

ÇİZELGE LİSTESİ

ÖNSÖZ

Tez çalışmam esnasında, bana özgür bir çalışma ortamı hazırlayan ve yardımlarını hiçbir zaman esirgemeyen Sayın Hocam Prof.Yaşar Baki CENGİZ Bey'e çok teşekkür ederim.

Tez çalışmam sırasında ve her zaman büyük bir manevi destek gördüğüm ve tez çalışmamın tüm külfetini çeken, benimle birlikte yorulan ve uykusuz kalan anneme, babama ve kardeşime sonsuz teşekkürlerimi ve sevgilerimi sunarım. Ayrıca çalışmalarım sırasında, çok büyük bir destek ve yardım gördüğüm sevgili arkadaşım Arş.Gör.Nihan ÇETİN'e de içtenlikle teşekkür ederim.

Tufan Demirel
Eylül, 1998

ÖZET

Simülasyon ve yapay zekâ teknikleri arasında kuvvetli bir metodolojik benzerlik vardır. Sonuçta her iki yöntemde de yapılan çalışma, karar vermeye yardım eden bilgisayar tabanlı bir model oluşturmaktır. Simülasyon tekniği birçok problemin çözümünde ve analizinde oldukça başarılı bir şekilde kullanılmaktadır. Bir simülasyon etüdünün yapılması için gerekli olan tüm faaliyetlerde yapay zekâ ve uzman sistem tekniklerinden faydalanılabilinir. Bu sayede bir simülasyon çalışmasının çok daha kısa sürede yapılması sağlanmış olur. Ayrıca, simülasyon tekniğini bilmeyen kullanıcıların da bir simülasyon çalışması yapmasına olanak sağlanır. Bu tez çalışması ile öncelikle yapay zekâ ve simülasyon tekniklerinin birleştirilmesine yönelik olarak yapılmış olan temel taşı niteliğindeki çalışmalar derlenmiş ve daha sonra, açıklanan teknikler kullanılarak farklı yapılara sahip kavşaklardaki trafik ışıklarının simülasyon ile optimizasyonu için yapay zekâ ile bütünleşik bir simülasyon sistemi önerilmiş ve bir prototip sistem geliştirilmiştir. Programlama dili olarak Borland Pascal 7.0 kullanılmıştır. Geliştirilen sistem bir diyalog arayüzü ile kullanıcıdan gerekli olan tüm verileri eksiksiz bir şekilde elde ederek, kullanıcının problemine çözüm üretir. Sistem, otomatik olarak çeşitli senaryolar oluşturarak bu senaryoların analizi ve mukayesesi ile optimum çözüme ulaşmaya çalışır.

ABSTRACT

There is a strong methodological similarity between simulation and artificial intelligence techniques. In fact, the study done in both techniques is to build a computer based model that is aimed to make a decision. Simulation technique is sucessfully utilized in solving a numerous number of problems. Techniques of artificial intelligence and expert systems are also used in all activities that are required for making a simulation study. Therefore, a simulation study is performed in a shorter time. Meanwhile, it is allowed to make a simulation study for the users having no simulation techniques backgrounds. In this thesis, the works performed to connect artificial intelligence and simulation techniques are examined and a simulation system integrated with artificial intelligence is proposed, using techniques mentioned above to optimize the simulation of traffic lights on the crossroads having different structures. Then a prototype model is built. PASCAL 7.0 is used as a programming language. The system developed will acquire all data from the user via a dialog interface and then will give a solution to the user's problem. The system will automatically generate various scenarios and it will attempt to get the optimum solution by analyzing and comparing these scenarios.

1. GİRİŞ

Mühendislikte amaç, insan yaşamını kolaylaştıracak sistemler yapmaktır. Bu sistemlerin çoğu, insanın yeteneklerini geliştirmeye yöneliktir. Örneğin otomobil ile, bir yere yürüyerek gitmekten çok daha hızlı gideriz; bilgisayar ile daha hızlı hesap yapabiliriz vb. Bazı sistemler ise insanın yapamadıklarını yapmak içindir; uçmak veya saniyede on milyon toplama işlemini yapmak gibi. Bilgisayarların bilgi saklayabilme kapasiteleri ve hesaplama hızları insanınkine göre çok daha fazladır. Bu, pek şaşırtıcı olmamalı, çünkü bilgisayarların ortaya çıkış nedeni zaten budur (Alpaydın, 1995).

Herhangi bir sorunun bilgisayar yardımıyla çözümünde temel yaklaşım, sorunu bilgisayarın anlayabileceği bir hale getirmek veya başka bir deyişle, bilinen bir dilde bir bilgisayar programı oluşturmaktır. Ancak bu şekilde sorun bilgisayara aktarılabilinir. Bu durumda bilgisayarlardan ancak programlama bilgisine sahip olanların faydalanabilmesi, bir dezavantaj olarak ortaya çıkmaktadır. Bu olumsuzluğa bir anlamda çare bulmak amacıyla, bilgisayarların ilk kullanılmaya başlanıldığı zamanlardan itibaren hazır program paketleri üzerinde çalışılmıştır. Böylece, kullanıcının sadece veri girişi ile problemini bilgisayara aktarıp çözebilmesi sağlanmıştır. Birçok matematiksel işlem, veri tabanı uygulamaları ve kelime işlem paketleri bu tür hazır paket programlara örnek sayılabilir (Ayağ vd., 1991).

Doukidis ve Paul (1990) Yöneylem Araştırması Derneği üyeleri arasında bir anket düzenleyerek, yapay zekâ ve uzman sistem tekniklerinin kullanımının ne durumda olduğunu ortaya çıkarmaya çalışmışlardır. Bazı anket sonuçlarına bakıldığında, çizelge 1.1'de YA modelleme alanlarının kullanım sıklığı görülmektedir. Değerlendirme ölçeği 1 ile 5 arasındadır ve "hiç kullanmadım" cevabı 1, ve "sıkça kullanıyorum" cevabı da 5 ile değerlendirilmiştir. Gene aynı çalışmada ankete katılanların YZ ile ilgileri çizelge 1.2'de gösterilmiştir.

Çizelge 1.1 Modelleme alanlarının kullanım sıklığı

Modelleme Alanı	Ortalama Kullanım
İstatistik ve Kestirim	3.35
Finansal veya Yatırım Modelleme	2.60
Simülasyon	2.54
Sezgisel Metotlar	2.34
Envanter Kontrolü	2.10
Matematik Programlama	2.08
Şebeke Analizi	2.02
Karar Teorisi	1.97
Üretim Çizelgeleme	1.74
Kuyruk Teorisi	1.58
Diğer	1.51

Çizelge 1.2 Ankete katılanların YZ ile ilgileri

Yapay Zekâ ile İlgi	%	Ankete katılan
Yapay zekâ ile ilgilenmemiş	78.5	208
Yapay zekâ ile ilgileniyor	21.5	57
YZ kullanıcısı	8.7	23
YZ geliştiricisi ve kullanıcısı	7.5	20
YZ geliştiricisi	5.3	14

Hazır paket programlar ile yapay zekâ uygulamalarının da başladığı söylenebilir. Bilgisayarların, hız ve kapasitelerindeki gelişmelere paralel olarak, kullanımdaki ekonomiklik sonucu, her alanda yaygınlaşması, yapay zekâ paketlerinin de gelişmesine hız kazandırmıştır. Hazır paket programların mevcut olan özelliklerine problem algılama, irdeleme, çözme, sonuçlara yorum getirme ve ileride benzer problemlerde kullanılmak üzere elde edilen sonuçlardan öğrenme özellikleri ilave edildiğinde yapay zekânın bugün için tanımlanan yapısı ortaya çıkmaktadır.

Bir simülasyon çalışması ile yapay zekâ ve uzman sistem çalışmasındaki adımlar oldukça benzerdir ve sonuçta her ikisinde de amaç, karar vermeye yardım eden ve destek sağlayan bilgisayar tabanlı bir model oluşturmaktır. Genellikle her iki yöntem de, göz önüne alınan sistemdeki belirsizliği modellemektedir.

Bir simülasyon çalışması göz önüne alındığında, simülasyoncu öncelikle problemini tanımlamalı ve formüle etmeli, daha sonra da onu bir simülasyon modeli haline getirmelidir. Bu arada veriler de analiz edilmelidir. Klasik olarak bir simülasyoncu, genel amaçlı bir programlama dilini (BASIC, FORTRAN, PASCAL vb.) veya özel amaçlı bir simülasyon dilini (SIMAN, GPSS, SLAM, vb.) kullanarak gerçek sistemin bir modelini oluşturur. Daha sonra bu model geçerli hale getirilir ve en sonunda da deneysel tasarımlar ve denemeler yapılarak sonuçlar bulunur ve analiz edilerek bulunan sonuçlar simülasyoncu tarafından yorumlanır. Bu son kısımda özellikle uygulama sahası ve incelenen duruma bağlı olarak, bir denemeden çok daha fazla deneysel tasarım ve senaryolar oluşturmak gerekmektedir. Bir karar vericinin sistem durumu hakkında bir karar verebilmesi için, çok sayıda senaryo geliştirip, geliştirilen bu senaryoların sonuçlarını ve analizlerini de birbirleriyle mukayese etmesi gerekmektedir. Ancak bunlar yapıldıktan sonra simülasyon sonuçları ve elde edilen bilgiler karar destek amacıyla kullanılabilir. Yapay zekâ ile bütünleşik simülasyon sistemleri veya simülasyon ortamları, bir kullanıcıya tüm çevrim boyunca destek sağlarlar. Böylece simülasyon tekniğini hiç bilmeyen veya az bilen kullancıların da bir simülasyon çalışması yapmasına ve tekniği bilen kullanıcıların ise çok daha kısa sürede sonuç elde etmesine olanak sağlanır.

1.1. Tezin Amacı

Bu tez çalışmasında ilk olarak yapılan, simülasyon ile yapay zekâ arasındaki benzerliklerden yola çıkarak bir simülasyon etüdünde yapay zekâdan ne şekilde faydalanılabileceğinin ortaya konulması ve bugüne kadar bu konu üzerinde yapılmış olan temel taşı niteliğindeki çalışmaların sonuçlarının incelenmesidir. Bu yapıldıktan sonra da, trafik kavşaklarında ışıkların simülasyon ile optimizasyonuna yönelik olarak zeki bir simülasyon ortamının tasarımı üzerinde durularak, bir sistem geliştirilmiştir.

1.2. Tezin Organizasyonu

Bu tez çalışması, *"Giriş"*, *"Simülasyon ve Modelleme"*, *"Yapay Zekâ ve Uzman Sistemler"*, *"Yapay Zekâ ile Bütünleşik Simülasyon Ortamları"*, *"Bir Kavşaktaki Trafik Işıklarının Optimizasyonu İçin Zeki Bir Simülasyon Ortamının Tasarlanması"* ve *"Sonuçlar ve Değerlendirmeler"* olmak üzere altı ana bölümde toplanmıştır.

Birinci bölüm giriş bölümü olup, bu bölümde tez konusu kısaca açıklanmış, *simülasyon* ile *yapay zekânın* benzeşimi üzerinde durulup, *yöneylem araştırması* konularının uygulanma sıklıkları ve yapay zekâ/uzman sistem tekniklerinin kullanımlarıyla ilgili olarak çeşitli bilgiler verilmiştir.

İkinci bölümde ise *simülasyon* ve *modelleme* konusu üzerinde durulmuş ve simülasyon kullanımının amaçları, avantaj ve dezavantajları ve uygulama alanları açıklandıktan sonra model ve modelleme ile ilgili bilgiler verilerek bir simülasyon etüdünde yapılması gereken işlemler hakkında bilgi verilmiştir.

Üçüncü bölümde de, *yapay zekâ* ve *uzman sistem* konusu açıklanmıştır. Bu bölümde bu konu etraflıca incelenmiş, yapay zekânın gelişimi ve yapay zekâdaki yaklaşımlar belirtilerek, bir uzman sistemin genel yapısı ve bir uzman sistemde bilginin organizasyonu konuları açıklanmış ve bir uzman sistemin nasıl geliştirileceği üzerinde durulmuştur.

Dördüncü bölüm *yapay zekâ* ve *simülasyon* tekniğinin birleştirilme yaklaşımları üzerinde yoğunlaştırılmış ve yapay zekâ/uzman sistem felsefesi ile simülasyonun benzerlikleri literatürle desteklenerek ortaya konulmuştur. Gene bu bölümde otomatik programlama yaklaşımları, bir sorunun algılanması için zeki kullanıcı arayüzleri, simülasyon programı üreteçlerinden söz edilerek bilgisayar programlama dillerinin gelişiminin, simülasyon ve yapay zekâ üzerindeki etkileri ortaya konulmuştur. Ayrıca nesneye yönelik programlamaya da değinilerek, yapay zekâ ile tamamen bütünleşik bir simülasyon yapısı üzerinde durulmuştur.

Beşinci bölümde, *bir kavşaktaki trafik ışıklarının optimizasyonu için zeki bir simülasyon ortamının tasarımı* üzerinde durulmuştur. Öncelikle trafik simülasyonu konusuna değinilerek, trafik ışıklarının simülasyonu için bir model önerilmiştir. Daha sonra geliştirilen bu modeli de kapsayan zeki bir ortam tasarlanmış ve geliştirilen sistem açıklanmıştır. En sonunda da geliştirilen bu sistem, farklı kavşak yapılarında çalıştırılarak elde edilen sonuçlar ortaya konulmuştur.

Altıncı ve son bölümde ise, bu tez çalışması ile elde edilen sonuçlar belirtilerek, gelecekte bu konular üzerine neler yapılabileceği üzerinde durulmuş ve genel bir değerlendirme yapılmıştır.

2. SİMÜLASYON ve MODELLEME

Simülasyon, gerçek bir sistemin modelini tasarlama süreci ve sistemin davranışını anlamak veya değişik stratejileri değerlendirmek amacı ile, geliştirilen bu model üzerinde denemeler yapmaktır (Halaç, 1982).

Bir başka tanıma göre simülasyon, gerçek bir prosesin veya sistemin zamana bağlı olarak modelini tanımlayan matematiksel bir modeldir. Simülasyon ister elle, isterse bilgisayar ile yapılsın, bir sistemin yapay kayıtlarının oluşturulması ve gerçek sistemin işletim karakteristikleriyle ilgili sonuçlarının elde edilmesinde bu yapay kayıdın incelenmesini kapsamaktadır (Banks ve Carson, 1984).

Zamana göre bir sistemin davranışı, bir simülasyon modeli geliştirilerek incelenir. Bu model genelde, sistemin işletimiyle ilgili bir kabuller kümesi şeklindedir. Bu kabuller, sistemin ilgili elemanları veya birimleri arasındaki matematiksel, lojik ve sembolik ilişkilere göre ifade edilir. Bir model geliştirildikten ve kabul edildikten sonra, gerçek olaylarla ilgili çok çeşitli soruları araştırmak üzere kullanılabilir. Sistemin performansı üzerindeki etkileri tahmin etmek için, öncelikle sistemdeki potansiyel değişiklikler simüle edilmelidir. Simülasyon, böyle sistemler geliştirilmeden önce tasarım safhasındaki sistemleri incelemek için de kullanılabilir. Bu nedenle simülasyon modellemesi, mevcut sistemdeki değişikliklerin etkisini tahmin edici bir analiz metodu ve çeşitli durumlarda yeni sistemlerin performansını tahmin edici bir tasarım metodu olarak kullanılabilir.

Bir sistemin bileşenleri arasındaki etkileşimleri matematiksel bağıntılar olarak ifade etmekle, gerekli bilgiyi, sanki gerçek sistemi gözlemliyormuş gibi toplayabiliriz. Bu sebepten simülasyonun doğası, normal olarak analitik modellerle analiz edilmesi zor olan karmaşık sistemlerin temsil edilmesinde daha fazla esnekliğe veya serbestliğe izin verir. Bununla beraber, her nekadar simülasyon esnek bir yöntem ise de, özellikle simüle edilen sistem optimize edilmeye uğraşılınca, ilgili simülasyon modelinin geliştirilmesi hem zaman alıcı ve hem de pahalı olur (Cengiz, 1989).

Bazı durumlarda, matematiksel metotlarla bile "çözülebilecek" kadar basit bir model

geliştirilebilir. Bu şekildeki çözümler, diferansiyel denklemler, ihtimal teorisi veya diğer matematiksel teknikleri kullanarak bulunabilir. Çözüm genelde, sistem performansının ölçüleri olarak adlandırılan bir veya daha fazla nümerik parametreden oluşabilir. Ancak, çok sayıdaki gerçek sistemler oldukça karmaşıktır ve bu nedenle bu sistemlerin modellerini matematiksel olarak çözmek hemen hemen imkânsızdır. Bu durumlarda, zamana göre sistemin davranışını simüle etmek için nümerik, bilgisayar tabanlı simülasyon kullanılabilir. Simülasyon sonucu, sanki gerçek sistem gözleniyormuş gibi veriler elde edilir. Simülasyon sonucu elde edilen bu veriler, sistemin performans ölçülerini tahmin etmek için kullanılır.

2.1. Simülasyonun Kullanım Amaçları

Özel amaçlı simülasyon dilleri, düşük operasyon maliyetleri için yüksek hesaplama kabiliyetleri ve simülasyon metodolojisindeki gelişmeler, simülasyonu yöneylem araştırmasında ve sistem analizinde en çok kullanılan ve kabul edilen bir metot yapmıştır. Simülasyonun hangi şartlar altında kullanılması gerektiği birçok yazar tarafından incelenmiştir. Bunları genel olarak sınıflandırırsak, simülasyon aşağıdaki amaçlar için kullanılabilir (Banks ve Carson, 1984):

(1) Simülasyon, karmaşık bir sistemin iç yapısını veya karmaşık bir sistemdeki alt sistemi incelemek için kullanılabilir,

(2) Bilgi, organizasyonel ve çevresel değişiklikler simüle edilebilir ve modelin davranışı üzerinde bu değişikliklerin etkileri incelenebilir,

(3) Bir simülasyon modelinin tasarımından elde edilen bilgiler, incelenen sistemin geliştirilmesine büyük ölçüde katkıda bulunmaktadır,

(4) Simülasyon girdilerini değiştirerek ve sonuçları inceleyerek, hangi değişkenlerin daha önemli olduğu ve değişkenlerin birbirlerini nasıl etkiledikleri hakkında bilgi edilinir,

(5) Simülasyon, analitik çözüm metodolojisini destekleyen bir bilgi verici araç olarak kullanılabilir,

(6) Simülasyon, uygulamadan önce yeni tasarımlar ve politikalar deneyerek durumun ne olacağını görmek için kullanılabilir,

(7) Simülasyon, analitik sonuçları test etmek için kullanılabilir.

2.2. Simülasyonun Avantajları ve Dezavantajları

Simülasyon birçok durumda uygun bir metot olmasına rağmen sistem analisti, belirli bir durumdaki metodolojiyi izlemeden önce, simülasyonun avantajlarını ve dezavantajlarını gözönüne almalıdır. Simülasyonun temel avantajlarını şu şekilde sıralayabiliriz (Banks ve Carson, 1984):

(1) Bir model kurulduktan sonra bu model, teklif edilen tasarımları veya politikaları analiz etmek için tekrar tekrar kullanılabilir.

(2) Girdiler belirsiz olsa da, simülasyon metotları teklif edilen sistemi analiz etmek için kullanılabilir,

(3) Gerçek sistemden elde edilen verilere göre, simülasyon verilerini elde etme maliyeti oldukça düşüktür,

(4) Uygulama açısından, simülasyon metotları analitik metotlara nazaran daha basittir. Bu nedenle, simülasyon metotları kullanan potansiyel kullanıcıların sayısı analitik metotları kullananlardan daha fazladır (Potansiyel kullanıcılar; simülasyon metotlarını kullananlar ve gelecekte de kullanacak olanları ifade etmektedir.),

(5) Analitik modeller matematiksel olarak kontrolu sağlamak için, çok basit kabuller kullanırlar; oysa simülasyon modellerinde kullanılan kabuller çok daha fazla ve detaylıdır.

(6) Bazı durumlarda simülasyon, bir problemin çözümünü veren tek araçtır.

Simülasyonu kullanmadan önce, gözönüne alınması gereken dezavantajlar ise şunlardır:

(1) Dijital bilgisayarlara ait simülasyon modellerinin kurulması ve işletmeye alınması aşırı zaman gerektirdiğinden oldukça masraflı olabilir,

(2) Genelde, simülasyon modelinin çok fazla sayıda çalıştırılması gerekir ve bu yüzden yüksek bilgisayar maliyetleri ortaya çıkar,

(3) Simülasyon, analitik teknikler yeterli olduğu anlarda, nadiren kullanılır. Bu durum, kullanıcıların simülasyon metotlarını iyi bilmemeleri ve matematiksel temelleri unutmaları sonucu ortaya çıkar.

Schmidt ve Taylor (1970) tarafından belirtilen yukarıdaki ilk iki dezavantaj, özel amaçlı simülasyon dillerini (SIMAN, SLAM, SIMCRIPT, GPSS vb.) ve daha büyük bilgisayarları kullanarak ortadan kaldırılabilir.

2.3. Simülasyonun Uygulama Alanları

Simülasyon, çok çeşitli alanlarda uygulama alanına sahiptir. Hillier ve Lieberman (1980), bu tekniğin geniş uygulama alanlarını belirtmek için aşağıdaki örnekleri vermişlerdir:

(1) İşletme politikaları ve uygulamalarındaki (bakım kapasitesi, tesislerin, yedek uçakların vb.) değişiklikleri test etmek için bir havayolu şirketi tarafından büyük bir havaalanındaki operasyonların simülasyonu,

(2) En iyi trafik akışını belirlemek için, trafik ışıklarının simülasyonu,

(3) Optimal tamir personeli sayısını belirlemek için, bakım operasyonu simülasyonu,

(4) Bir radyasyon kalkanına yansıyan radyasyonun yoğunluğunu belirlemek için, bu kalkandaki yüksüz parçacıkların akış simülasyonu,

(5) Uygulama, kapasite ve tesislerin şekillerindeki değişiklikleri değerlendirmek için, çelik üretim operasyonunun simülasyonu,

(6) Ekonomik politika kararlarının etkilerini tahmin etmek için ekonomi simülasyonu,

(7) Savunma ve saldırı silah sistemlerini değerlendirmek için büyük çaplı askeri savaşların simülasyonu,

(8) Büyük çaplı dağıtım ve envanter kontrol sistemlerinin tasarımını geliştirmek için bu sistemlerin simülasyonu,

(9) Firmanın politikaları ve operasyonlarındaki değişiklikleri değerlendirmek için tüm firmanın genel operasyonlarının simülasyonu,

(10) En ekonomik düzeyde, tatmin edici servis sağlamak için, gerekli parça kapasitesini belirlemek maksadıyla bir telefon iletişim sisteminin simülasyonu,

(11) En ideal baraj, elektrik santralı ve sulama işlerinin şeklini belirlemek için, ırmak havza operasyonlarının simülasyonu.

2.4. Sistem ve Sistemin Çevresi

Bir sistemi modellemek için, sistem kavramını ve sistem sınırını anlamak gerekir. Sistem, bir amaca ulaşmak için, aralarında düzenli ilişki olan veya birbirlerini etkileyen elemanlar grubu olarak tanımlanabilir. Sisteme bir örnek olarak, otomobil üretimi yapan bir üretim sistemini gösterebiliriz. Makinalar, parçalar ve işçiler bir montaj boyunca beraber çalışarak, yüksek kaliteli araçlar üretirler.

Sistem, bazen kendi dışında oluşan değişikliklerden de etkilenebilir. Bu şekildeki değişiklikler, sistemin çevresinde oluşur.

Sistemlerin modellenmesinde, sistemin sınırı ve çevresi hakkında karar vermek gerekir. Bu karar, çalışmanın amacına bağlı olabilir.

2.4.1. Bir sistemin elemanları

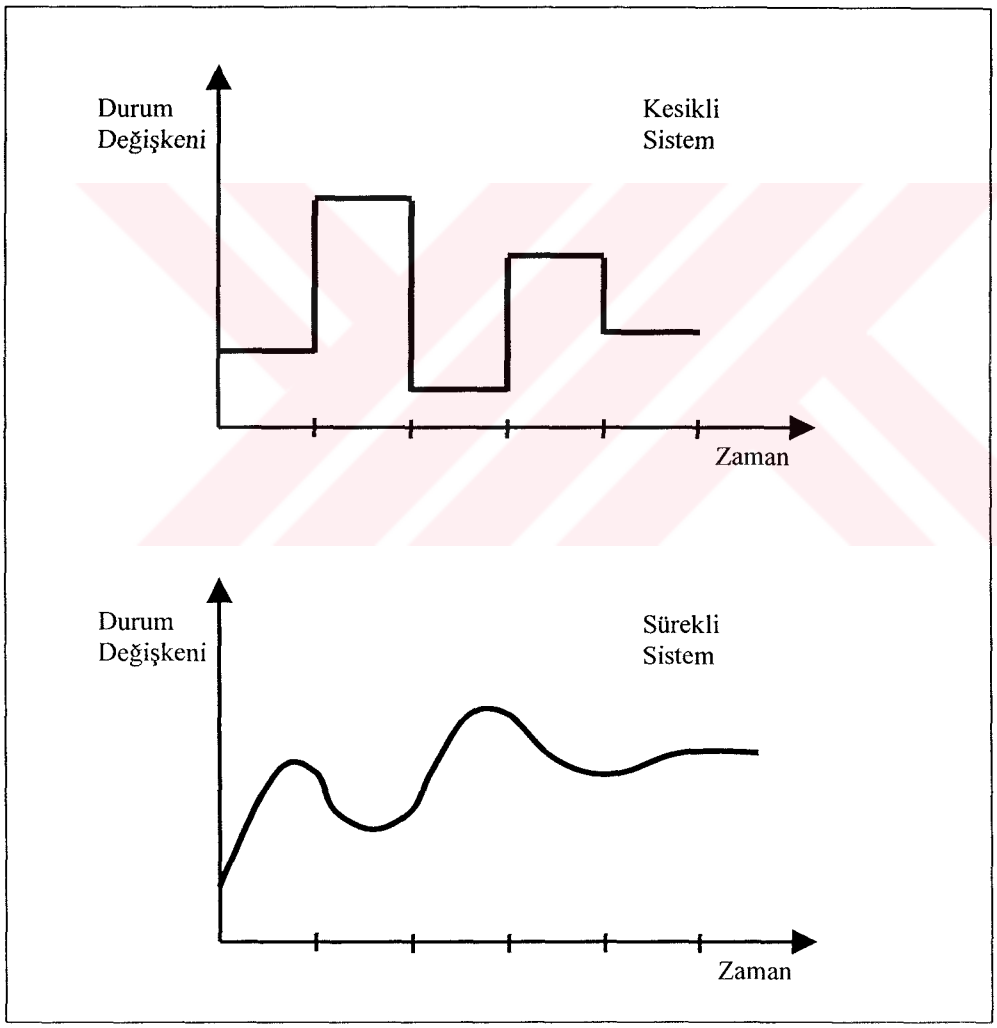
Bir sistemi anlamak ve analiz etmek için, çok sayıda terim tanımlanmıştır. Bu terimleri eleman, özellik, faaliyet, durum, olay, sistem içi, sistem dışı olarak sınıflandırabiliriz. Eleman, sistemdeki ilgili unsurdur. Özellik, bir elemanın niteliğidir. Faaliyet, belirli uzunluktaki bir zaman peryodunu temsil eder. Bir banka örneği gözönüne alındığında, müşteriler elemanlardan biri, hesap bakiyeleri bir özellik ve para yatırma bir faaliyet olabilir.

Bir çalışmadaki sistemi oluşturan elemanların kümesi, başka bir çalışmadaki tüm sistemin bir alt kümesi olabilir. Örnek olarak, para çekme ve yatırma için gerekli veznedar sayısının belirlenmesinde yukarıdaki banka incelendiğinde sistem, veznedarlar ve hatta bekleyen müşterilerin bulunduğu kısım olarak tanımlanabilir. Eğer çalışma özel veznedar (seyahat çeki veren veznedar, ticari çek veren veznedar gibi) sayısını belirlemek üzere genişletilirse, sistemin tanımı genişletilebilir.

Sistem durumu, çalışmanın amacına bağlı olarak, herhangi bir anda sistemi tanımlamak için gerekli değişkenler kümesi şeklinde tanımlanabilir. Banka örneğindeki durum değişkenleri, meşgul veznedarların sayısı, hatta bekleyen veya servis gören müşterilerin sayısı ve bir sonraki müşterinin geliş zamanıdır. Olay ise, sistemin durumunu değiştirebilen ani oluşumdur. Sistem içi terimi, bir sistem içinde ortaya çıkan faaliyetleri ve olayları tanımlamak için; sistem dışı terimi ise, sistemi etkileyen çevredeki olayları ve faaliyetleri tanımlamak için kullanılmaktadır. Banka örneğinde, müşterinin gelişi bir sistem dışı olay ve müşteri servisinin tamamlanması da bir sistem içi olaydır.

2.4.2. Kesikli ve sürekli sistemler

Sistemler, kesikli veya sürekli olarak sınıflandırılabilir. Pratikte çok az sayıda sistem tamamen kesikli veya süreklidir. Çünkü, tek bir değişim tipi sistemde etkili olduğundan, bir sistemi kesikli veya sürekli olarak sınıflandırmak mümkündür. Kesikli sistem, durum değişkeninin (değişkenlerinin) zamana göre sadece kesikli nokta kümesinde değiştiği bir sistemdir. Banka, kesikli sisteme bir örnek olarak gösterilebilir, çünkü durum değişkeni (bankadaki müşteri sayısı), bir müşteri geldiğinde veya müşteriye verilen servis tamamlandığında değişmektedir.



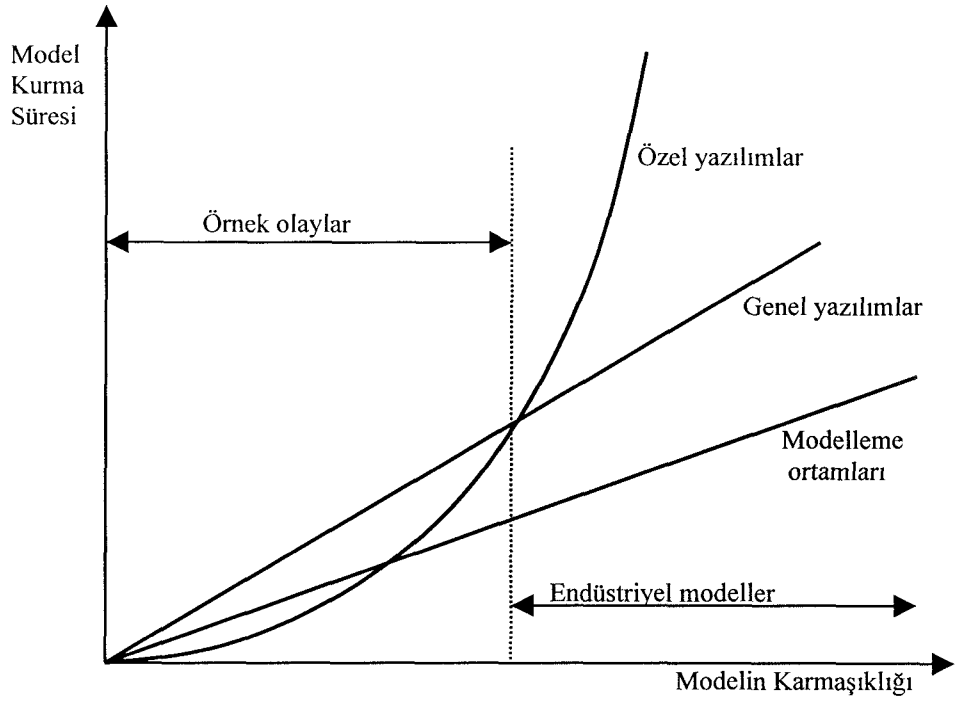
Şekil 2.1 Durum değişkeninin sürekli ve kesikli sistemlerdeki değişim formu.

Sürekli sistem ise, durum değişkeninin (değişkenlerinin) zamana göre sürekli olarak değiştiği bir sistemdir. Sürekli sisteme örnek olarak, bir barajdaki su yüksekliği gösterilebilir. Yoğun yağmur yağışından sonra sular, barajın arkasındaki göle akmaktadır. Su baskınını kontrol etmek ve elektrik üretmek için sular barajdan çekilmelidir. Burada, buharlaşma da su düzeyini azaltmaktadır. Böylece, durum değişkeni olan barajdaki su yüksekliğinde, bu etkiler altında sürekli bir değişim olmaktadır. Şekil 2.1'de sürekli ve kesikli durum değişkenlerinin bir zaman periyodunda nasıl değiştikleri görülmektedir.

2.5. Karmaşık Sistemler ve Modelleme

Karmaşık sistemler, artan bir şekilde birçok büyük alanlar için kullanılmaktadır. Bu alanlar arasında yazılım sistemleri, üretim sistemleri, bilgisayar sistemleri, ulaştırma sistemleri, ekolojik sistemler vb. sayılabilir. Bu sistemlerin karmaşıklığı, onların boyutlarının (sistem içindeki elemanların adedi) ve operasyonlarının (sistemin elemanları arasındaki etkileşimlerin tipi ve sayısı) birleştirilmiş olmasından kaynaklanmaktadır. Karmaşık sistemlerde ele alınan ana problemler, bunların boyutları, operasyonların anlaşılabilmesi, sistemin verimliliğinin arttırılması ve sistemin performansının hesaplanması olarak karşımıza çıkmaktadır (Hill, 1996). Hill (1996) tarafından Kellert'in bir çalışmasında yer alan modellenecek sistemin karmaşıklığına ve seçilen yazılımın tipine bağlı olarak model geliştirmenin beklenen süresinin bir diyagramı Şekil 2.2'de görülmektedir.

Sistemin elemanları arasındaki ilişkileri anlamak veya yeni bir politika altında sistemin nasıl davranacağını tahmin etmek için bir sistemi incelemek gereklidir. Bir sistemi incelemek için, sistemin kendisi ile deney yapmak mümkündür. Ancak, bu durum her zaman geçerli değildir. Yeni bir sistem oluşmadan önce sistem, teorik formda veya tasarım safhası şeklindedir. Mevcut sistem ile deney yapmak pratik bir yöntem değildir. Örnek olarak, enflasyon altında çalışanların etkilerini belirlemek için işsiz kişilerin oranını iki katına çıkarmak mümkün değildir. Banka örneğinde, bekleme hattı uzunluğunun etkisini incelemek için veznedar sayısını azaltmak, müşterileri memnun etmez ve bu nedenle müşteriler, hesaplarını rakip bankaya yatırırlar. Sonuç olarak, sistemlerin incelenmesi sistemin bir modeli ile yapılmaktadır.



Şekil 2.2 Modelin karmaşıklığı ile modelleme süresinin karşılaştırılması

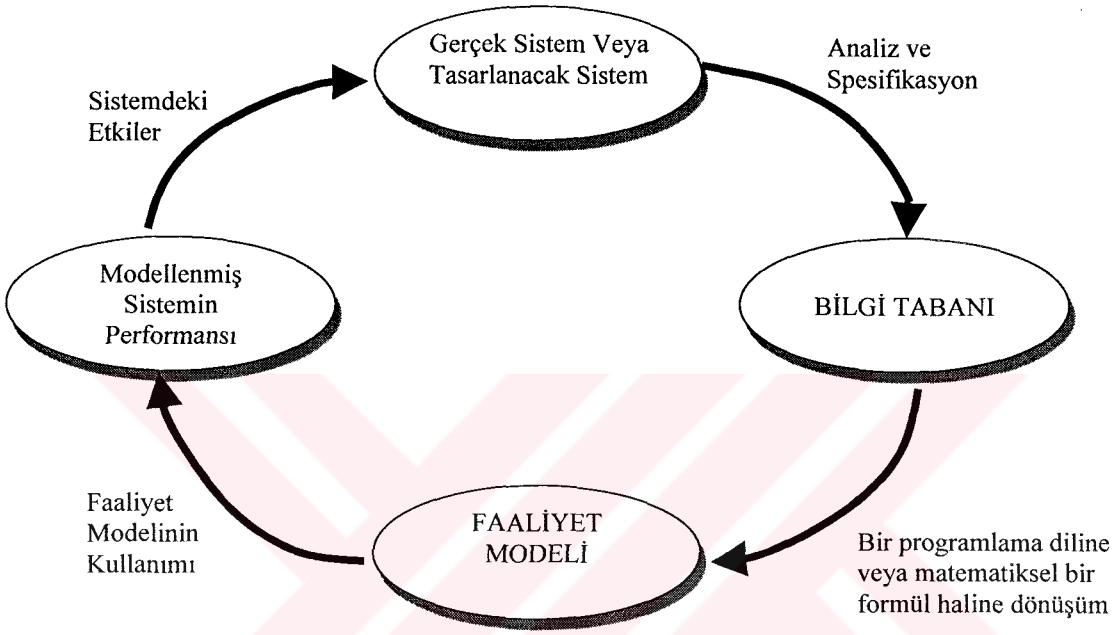
Bir sistemin performansıyla ilgili bir çalışmada, öncelikle bu sistemin bir modelinin oluşturulması gereklidir. Hill (1996) tarafından belirtildiğine göre, Poper bir model fikrinde birbiriyle bağlantılı üç kavram olduğunu ileri sürmüştür. Bu üç kavram ise şunlardır:

- Bir model gerçek bir sistem ile benzeşmelidir,
- Bir model gerçek bir sistemin basitleştirilmiş bir teşkili olmalıdır ,
- Bir model gerçek bir sistemin ideal bir hali olmalıdır.

Pratikte bir model, modellenecek olan gerçek bir sistemin gözlemleri baz alınarak oluşturulur. Bir bilgi modeli, faaliyet modeli ve modelleme işlemi arasındaki ilişkiler Şekil 2.3'de görülmektedir (Hill, 1996). Şekilde görülen iki kavramdan biri olan bir sistemin bilgi modeli, sistemin yapısının ve fonksiyonlarının doğal veya grafik bir dille formüle edilmesidir. Eğer bir sistem mevcutsa, bir bilgi modeli gözlem safhası esnasında elde edilmiş olan bilgilerin tümünü içermektedir. Eğer bir sistem mevcut değilse, o zaman bir bilgi modeli, topoloji ve tasarımcıların işlem spesifikasyonlarını kapsar. Faaliyet modeli

ise, bilgi modelinin bir matematiksel formülasyona veya bir programlama diline dönüştürülmesidir.

Model, sistemi incelemek üzere sistemin örneği olarak tanımlanır. Birçok çalışma için, bir sistemin tüm detaylarını gözönüne almak gereksizdir; bu nedenle model, sadece sistemin bir yardımcısı değil, aynı zamanda sistemin basit bir şeklidir. Diğer yandan model, gerçek



Şekil 2.3 Modelleme işlemi

sistemden alınan sonuçlara izin verecek şekilde detaylı olmalıdır. Değişik araştırmalar için, aynı sistemin farklı modelleri de kurulabilir.

2.6. Model Tipleri

Modeller, matematiksel veya fiziksel modeller olarak sınıflandırılabilir. Bir matematiksel model, sistemi temsil etmek için sembolik notasyon ve matematiksel denklemleri kullanır. Simülasyon modeli, belirli tipte bir matematiksel sistem modelidir.

Simülasyon modelleri, statik veya dinamik, deterministik veya stokastik ve kesikli veya sürekli olarak sınıflandırılabilirler. Monte Carlo Simülasyonu olarak bilinen statik

simülasyon modeli, zamanın belirli bir anındaki sistemi temsil etmektedir. Dinamik simülasyon modelleri ise, zamana göre değişen sistemleri temsil etmektedir. Saat 8.00 ile 17.00 arasında çalışan bir bankanın simülasyonu, dinamik simülasyona bir örnek olarak verilebilir.

Rassal değişken içermeyen modeller, deterministik modeller olarak sınıflandırılırlar. Deterministik modeller, tek bir çıktı kümesi veren girdi kümesine sahiptir. Deterministik modellere örnek olarak, tüm hastaların randevu saatlerine göre geldikleri bir dişçi muayenehanesini gösterebiliriz. Stokastik simülasyon modeli ise, girdi olarak bir veya daha fazla rassal değişkeni gözönüne almaktadır. Rassal girdiler, rassal çıktılar oluştururlar. Çıktılar rassal olduğu için, bu çıktılar modelin gerçek karakteristiklerinin tahminleri olarak gözönüne alınabilir. Bir bankanın simülasyonu, genellikle rassal gelişler arası süreleri ve rassal servis sürelerini kapsamaktadır. Bu nedenle, stokastik bir simülasyonda, çıktı ölçütleri (bekleyen ortalama müşteri sayısı, bir müşterinin ortalama bekleme zamanı) sistemin gerçek karakteristiklerinin istatistiksel tahminleri olarak ele alınırlar.

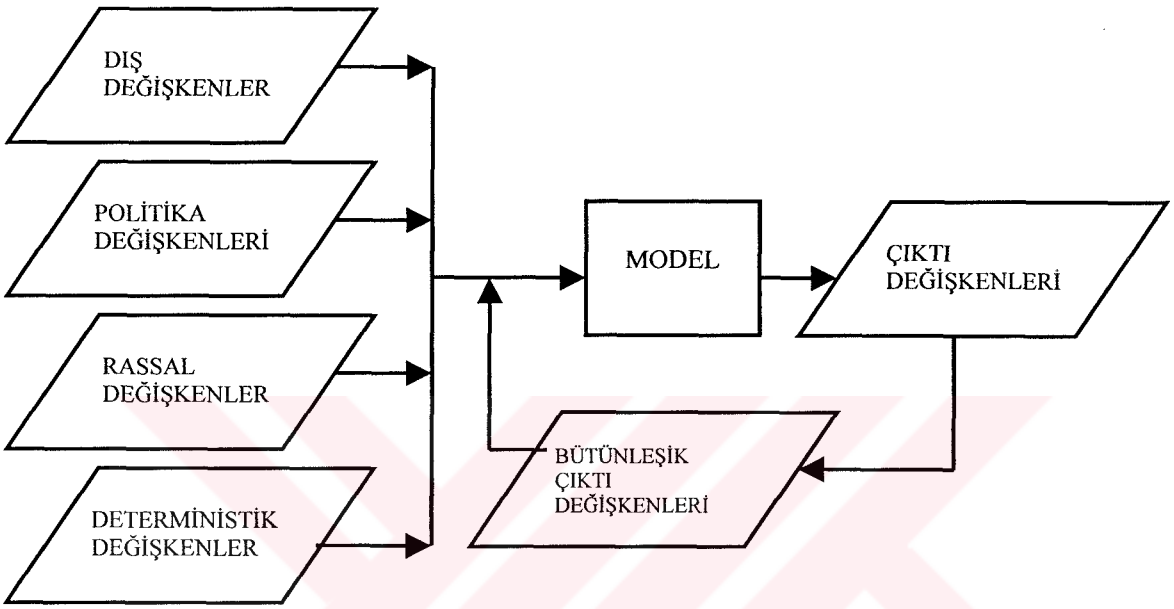
Kesikli ve sürekli modeller, analog bir şekilde tanımlanmıştır. Ancak, kesikli bir simülasyon modeli kesikli bir sistemi modellemek için; sürekli bir simülasyon modeli de sürekli bir sistemi modellemek için her zaman kullanılmaz. Ayrıca simülasyon modelleri, kesikli ve sürekli şekilde karma modeller olabilirler. Kesikli veya sürekli (veya hem kesikli hem sürekli) simülasyon modelini kullanma seçimi, sistem karakteristiklerinin ve çalışma amacının bir fonksiyonudur. Bu nedenle, her mesajın karakteristiğinin ve hareketinin çok önemli olduğu bir iletişim kanalı kesikli olarak modellenebilir. Aynı şekilde, kanaldaki mesajların akışı önemli olduğunda, sürekli simülasyon kullanarak sistem modelleme daha uygun olmaktadır. Burada ele alınan modeller kesikli, dinamik ve stokastik yapıdadır. (Banks ve Carson, 1984).

2.7. Bir Simülasyon Modelindeki Değişkenler

Naylor ve Gatts'e (1976) göre simülasyonda kullanılan değişkenler çıktı, bütünleşik çıktı, dış, politika değişkenleri veya rassal değişkenler olarak sınıflandırılabilirler. Birçok

simülasyon modelleri probabilistik olmadığından, burada deterministik değişkenleri de göz önüne almak gerekmektedir. Şekil 2.4, simülasyon modellerindeki bu değişkenlerin rolünü göstermektedir (Watson ve Blackstone,1990).

Buradaki değişkenleri sırayla inceleyelim:



Şekil 2.4 Simülasyon modellerinde kullanılan değişkenler

Çıktı Değişkenleri: Çıktı değişkenleri,analiste ve/veya yönetime gerekli ara sonuçları veya son bilgileri sağlamaktadır. Bir üretim modelinde çıktı değişkenine örnek olarak, satılan mamullerin maliyeti ve envanterdeki yarı mamulle ilgili bilgiler gösterilebilir.

Bütünleşik Çıktı Değişkenleri: Daha önce de belirtildiği gibi, simülasyon modelleri dinamiktir ve zamana göre sistemin davranışını tanımlamaktadır. Sonuçta, zaman sonunda sistemin durumu, bir sonraki zaman periyodu boyunca sistemin analizinde gerekli bir girdi olmaktadır. Bütünleşik çıktı değişkenleri, bir zaman periyodundan diğerine sistemin durumu hakkında bilgi vermektedir.

Bütünleşik çıktı değişkenlerine (seri ilişkili değişkenler) ait örnekler üretim, finans ve pazarlama modellerinde ortaya çıkmaktadır. Periyot sonu bitmiş mamul envanteri, gelecek periyodun gelecek mamul envanterinin belirlenmesinde gerekli bir girdidir.

Dış Değişkenler: Sistemin davranışını etkileyen bu değişkenler, dış değişkenler olarak belirtilmektedir. Dış değişkenlerin yapısına örnek olarak şu önermeyi kullanabiliriz: " X, Y'den etkilenir; ancak Y, X'den etkilenmez". Dış değişkenler çoğunlukla sistem dışı, çevresel veya kontrol edilemeyen değişkenler olarak adlandırılırlar. Bir üretim modelinde, grevler ve hammadde eksikliği, dış değişkenler olarak gözönüne alınabilir.

Politika Değişkenleri: Birçok sistem, yönetimin incelediği kontrol faktörlerini içermektedir. Örnek olarak bir organizasyonda yönetim, hangi şartlar altında hangi makinanın alınacağına karar verebilir. Bu şekildeki faktörler, simülasyon modellerindeki politika değişkenleri olarak gözönüne alınırlar. Bu faktörler, yönetimin kararlarına ve kontroluna göre değiştiği için, politika değişkenleri bazen karar veya kontrol edilebilir değişkenler olarak da adlandırılırlar. Birçok simülasyon modelinde politika değişkenlerinin hangi sistem politikasında daha etkin olduğu araştırılır.

Rassal Değişkenler: Bazı sistemlerin davranışını belirlemek için, bunların rassal veya probabilistik yapısını anlamak gerekir. Rassal değişkenler, bu role hizmet etmektedir.

Deterministik Değişkenler: Birçok sistem probabilistik olmasına rağmen analist, deterministik bir modelin kullanıcının amaçları için yeterli bilgileri sağlayacağını bilir. Bu durumda, sadece tek değerli tahminler gerektiren deterministik değişkenler kullanılır. Sistemdeki varyasyon miktarının çok az olduğu probabilistik modellerde, bazen deterministik değişkenler de kullanılmaktadır.

2.8. Kesikli Olay Sistem Simülasyonu

Burada yapılan çalışmalar, kesikli olay sistem simülasyonuna dayanmaktadır. Kesikli olay sistem simülasyonu, durum değişkenlerinin zamana göre sadece kesikli noktalarda değiştiği sistemlerin modellenmesidir. Simülasyon modelleri, analitik metotlar yerine

nümerik metotlarla analiz edilirler. Analitik metotlar, modeli "çözmek" için indirgenmiş matematiksel analizi kullanmaktadır. Örneğin, bazı envanter modellerindeki minimum maliyet politikasını belirlemede diferansiyel denklemler kullanılmaktadır. Nümerik metotlar ise, matematiksel modelleri "çözmek" için hesaplama prosedürleri kullanırlar. Nümerik metotların kullanıldığı simülasyon modellerinde, modellerin çözümü yerine işletimi sağlanır. Yani, model kabullerine bağlı olarak sistemin yapay bir kayıdı oluşturulur ve gerçek sistem performans ölçülerini tahmin ve analiz etmek için gözlemler elde edilir. Gerçek simülasyon modelleri oldukça büyük ve saklanan veya işletilen bilgilerin sayıları da çok fazla olduğundan, modellerin işletimi bilgisayarlar tarafından yapılmaktadır. Ancak, küçük modellerin elle simülasyonu daha kolay anlaşılmaktadır.

Çizelge 2.1 Kesikli olay modellemedeki bazı kavramlar

Kavram	Açıklama
Sistem	Bir zaman dilimi boyunca bir veya birden çok amacı başarmak için birbiriyle etkileşim halindaki varlıkların (gezen birimler, yani insan, makinalar vb.) tümü.
Model	Sistemin özet bir temsilidir ve çoğunlukla sistem durumu, gezen birimler ve onların nitelikleri, kümeler, olaylar, faaliyetler ve gecikmeler terimleriyle tanımlanan bir sistemdeki mantıksal ve/veya matematiksel ilişkileri kapsamaktadır.
Sistem Durumu	Herhangi bir zamanda bir sistemi tanımlamayı gerektiren tüm bilgileri kapsayan değişkenlerin tümü.
Gezen Birim (Varlık)	Bir model ile temsil edilen bir sisteme ihtiyaç gösteren sistem içindeki herhangi bir nesne veya bileşen (bir servisçi, bir müşteri, bir makina).
Nitelikler	Verilen bir gezen birimin özelliği.
Küme	Bazı mantıksal davranışlar ile düzenlenmiş birbiriyle benzeşen gezen birimlerin tümü.
Olay	Bir sistemin durumunda aniden meydana gelen değişim (sisteme yeni bir müşterinin gelmesi gibi).
Faaliyet	Belirli bir uzunluktaki bir zaman süresi.
Gecikme	Belirsiz bir uzunluktaki bir zaman süresi.

Bir sistemin kesikli olay modelini geliştirmek için bazı kavramlar kullanılmaktadır. Bu ana kavramlar Çizelge 2.1'de kısaca tanımlanmıştır (Banks ve Carson, 1984):

Kümeler bazı durumlarda listeler, kuyruklar veya zincirlerdir. Bir faaliyet deterministik veya olasılığa dayalı olabilir veya herhangi bir matematiksel fonksiyondur. Bir müşterinin kuyrukta beklerken harcadığı zaman, gecikmeye tipik bir örnektir.

Sistemler zaman boyunca değiştiğinden dinamik bir yapıdadır. Bu yüzden zaman boyunca sürekli olarak değişen sistem durumu, gezen birimlerin nitelikleri ve aktif gezen birimlerin sayısı, kümelerin içerikleri, faaliyet ve içeriklerin tümü zamanın bir fonksiyonudur. Bütün bu zamanların hepsi, bir saat değişkeniyle temsil edilmektedir.

2.9 Simülasyon Çalışmasındaki Adımlar

Bir simülasyon çalışmasının yapılmasındaki çevrim Şekil 2.5'te gösterildiği gibi 10 adımdan meydana gelmektedir (Balcı, 1987; Nance, 1981). Şekilde görülen kesikli oklar bulunulan safha esnasında yapılmakta olan işlemi tanımlamaktadır. Düz çizgili oklar ise, bir sonraki safhaya ulaşma esnasında bir önceki safhaya danışıldığını yani bir önceki safhadan faydalanıldığını göstermektedir (Balcı, 1992). Simülasyon çevrimindeki safhaların ve işlemlerin açıklaması aşağıda verilmiştir:

Problemin formülasyonu: Her çalışma, problemin tanımlanmasıyla başlar. Eğer problem, yöneticiler veya diğer kişiler tarafından ortaya çıkarılmışsa analist, tanımlanan problemi açıkça anlamalıdır. Problemin tanımı analist tarafından yapılırsa, yöneticiler veya problem sahipleri formülasyonu anlamalı ve aynı noktada karar kılmalıdırlar. Fakat bazı durumlarda çalışma ilerlerken problemin yeniden formüle edilme durumu ortaya çıkmaktadır.

Çözüm tekniklerinin araştırılması: Formüle edilmiş olan bir problemin çözülmesinde kullanılabilecek tüm alternatif teknikler belirlenmelidir. Çünkü herhangi bir çözüm tekniği çok fazla maliyetli olabilir, yeterli miktarda faydalı olmayabilir veya çalışmanın amaçlarına tam olarak cevap vermeyebilir. Etüdün amaçlarına tam olarak cevap verebilen ve kabul edilebilir bir kâr/maliyet oranına göre bir çözüm tekniği belirlenerek seçilmelidir.

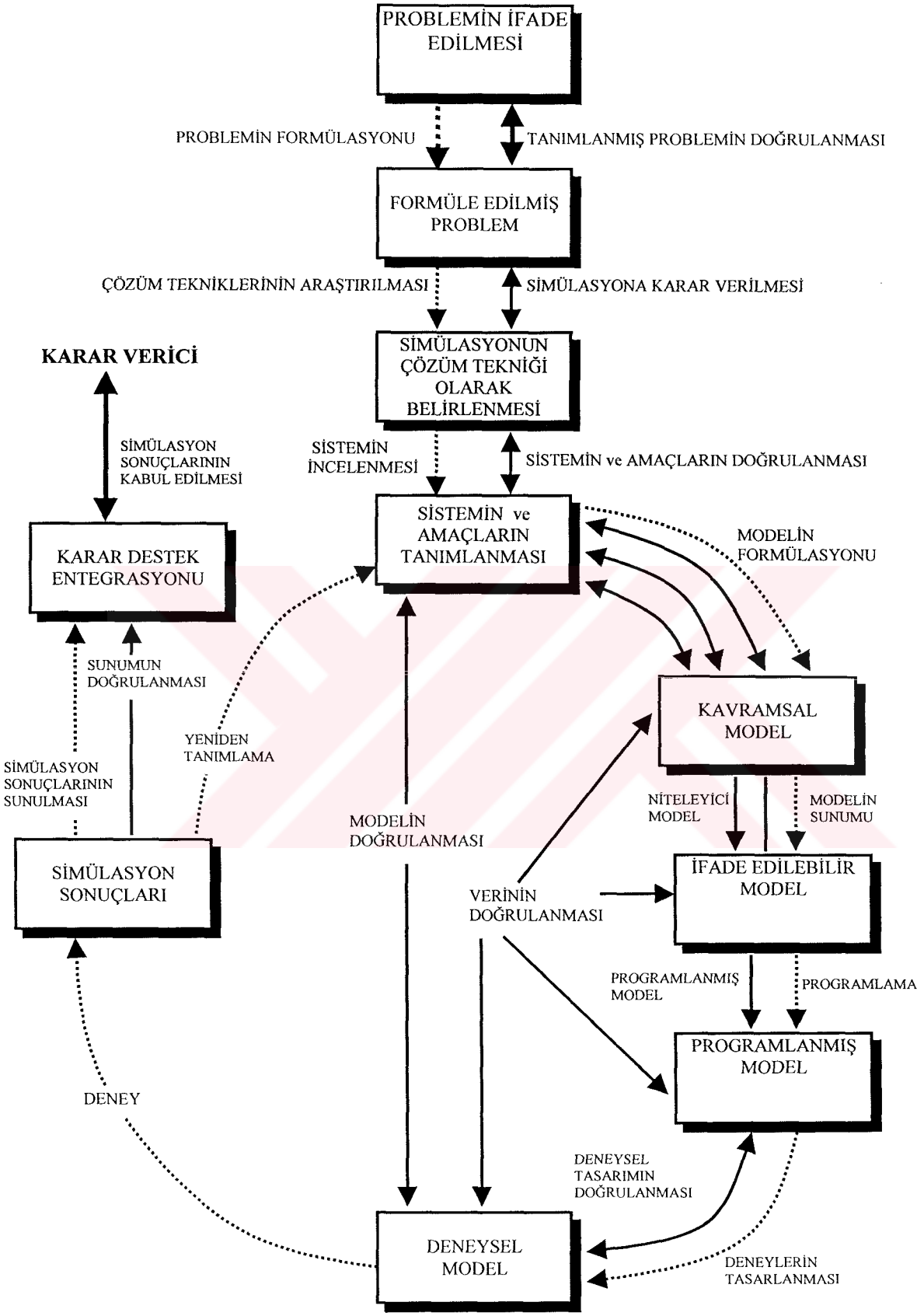
Sistemin incelenmesi: Formüle edilmiş bir problemden oluşan bir sistemin karakteristikleri, sistemin tanımlanması ve modellenmesini dikkate almak için gözden geçirilmelidir. Shannon (1975), altı tane ana sistem karakteristiği olduğunu belirtmiştir. Bu altı ana karakteristik değişim, ortam, davranış, düşük performansa eğilim, karşılıklı dayanışma ve organizasyondur.

Model kurma ve modelin formülasyonu: Bir sistemin modelini kurma, bilimin sanat yönünü oluşturmaktadır. Her örnekte, başarılı ve uygun modelleri kurmak için gerekli kuralların bulunması zor olmasına rağmen, izlenecek bazı genel kurallar bulunmaktadır. Modelleme sanatı, problemin temel özelliklerini belirlemek, sistemi karakterize eden temel kabulleri seçmek, değiştirmek ve daha sonra kullanışlı uygun verileri elde edene kadar modeli genişletmek için bir özellik ile tamamlanır. Böylece, basit bir modelle işe başlamak ve modeli daha sonra modelin temel amaçlarına aykırı olmamak şartıyla geliştirmek en uygun adım olmaktadır. Bu prensibin bozulması, model kurma ve bilgisayar masraflarının artmasına neden olacaktır. Model ve gerçek sistem arasında birebir karşılaştırma yapmak gereksizdir. Sadece, gerçek sistemin niteliği önemlidir(Banks ve Carson, 1984).

Model kurmada, model kullanıcısını da gözönüne almak gerekmektedir. Model kullanıcısını gözönüne alarak hem oluşan modelin kalitesi, hem de modelin uygulanmasında modeli kullanan kişinin güveni artar.

Modelin formülasyonu ise, tahayyül edilmiş kavramsal bir modeli, çalışan bir sistemi temsil edebilecek hale getirme işlemidir. Kavramsal model de, gerçek sistemin modelinin bir modelleme uzmanının zihninde formüle edilmiş halidir (Nance, 1981).

Amaçların ve tüm proje planının belirlenmesi: Amaçlar, simülasyon tarafından cevaplandırılacak soruları belirtmektedir. Bu noktada simülasyonun, belirtilen amaçlar ve formüle edilen problem için uygun bir metodoloji olup olmadığına karar verilmelidir. Simülasyonun uygun bir metot olduğuna karar verdikten sonra tüm proje planı, gözönüne alınacak alternatif sistemlerin tanımını ve bu alternatiflerin etkinliklerini değerlendirecek bir metodu kapsamalıdır. Bu plan ayrıca çalışan kişilerin sayısı, çalışmanın maliyeti ve her



Şekil 2.5 Bir simülasyon çalışmasının çevrimi

kademenin sonundaki tahmini sonuçlarla, çalışmanın her safhasını gerçekleştirmek için gerekli gün sayısıyla ilgili çalışma planlarını da kapsamaktadır.

Verilerin toplanması: Modelin kurulması ve gerekli verilerin toplanması arasında sabit bir ilişki bulunmaktadır. Modelin zorluk derecesi değişirken, gerekli veri elemanları da değişmektedir. Bir simülasyonun gerçekleştirilmesinde veri toplama, gerekli toplam zamanın büyük bir kısmını oluşturduğundan, veri toplamaya genellikle model kurmanın ilk safhalarında mümkün olduğu kadar erken başlamak gerekmektedir. Bu çalışmanın amaçları, toplanacak veri türü üzerinde önemle durmaktır. Banka örneğinde amaç, değişik veznedar sayısına göre bekleme hattının uzunluğunu öğrenmek ise, gerekli veri türleri, gelişler arası zamanların (günün farklı zamanlarında) dağılımı, veznedarlara ait servis zamanı dağılımı ve değişen şartlar altında bekleme hatlarının uzunluklarına bağlı önemli dağılımlar olabilir. Bu son veriler, simülasyon modelinin geçerliliğini kabul etmek için kullanılmaktadır.

Program Yazma: Gerçek sistemler, çok sayıda bilgi depolaması ve hesabı gerektiren modeller şeklinde ortaya çıktıklarından model, dijital bir bilgisayarda programlanmalıdır. Modelleyici modeli FORTRAN, PASCAL gibi genel amaçlı bir dilde mi, yoksa GPSS, SIMSCRIPT veya SLAM gibi özel amaçlı bir simülasyon dilinde mi programlayacağına karar vermelidir. Genel amaçlı diller, daha uzun bir yazılım süresi gerektirirler. Ancak, özel amaçlı dillere göre daha hızlı çalışırlar. Bununla beraber, özel amaçlı dillerde yazılım ve hata bulma çok çabuk olduğundan, model kurucuların çoğunluğu bu dilleri kullanmaktadır.

Programın onaylanması: Onaylama, simülasyon modeli için hazırlanan bilgisayar programıyla ilgilidir. Bilgisayar programı doğru çalışıyor mu? Çok karmaşık modellerde, iyi bir hata ayırıcı olmadan tüm modeli başarılı bir şekilde programlamak oldukça zordur. Girdi parametreleri ve modelin mantıksal yapısı, programda doğru bir şekilde yazılmış ise, program doğru olarak kabul edilir.

Modelin Onaylanması: Burada yapılan çalışma, modelin gerçek sistemi doğru bir şekilde temsil ettiğini belirlemektir. Uygunluk, modelin ayarlanması, modelin ve gerçek sistemin

davranışlarının karşılaştırılması ve bu ikisi arasındaki farklılıkları kullanarak elde edilir. Bu proses, model doğru olarak kabul edilene kadar tekrar edilir. Yukarıda bahsedilen banka örneğinde, mevcut durumlarda bekleme hatlarının uzunluklarıyla ilgili veriler toplanmıştır. Simülasyon modeli, bu sistem ölçüsüne cevap verir mi? Bu, uygunluk sorularından birisidir.

Deneysel Tasarım: Burada, simüle edilecek alternatifler belirlenmelidir. Çoğunlukla, hangi alternatifin simüle edileceği kararı, tamamlanmış ve analiz edilmiş çıktıların bir fonksiyonu olabilir. Simüle edilecek her sistem tasarımı için, ilk periyodun uzunluğu, simülasyon çıktılarının uzunluğu ve her çıktıda verilecek cevapların sayılarıyla ilgili kararlar verilmelidir. Burada yapılan çalışma, özel bir amaç için simülasyon modeli ile deney yapma işlemidir. Shannon (1975) tarafından belirtilen deneme amaçları, farklı işletim politikalarını mukayese etmek, sistemin davranışını değerlendirmek, ve optimizasyonu ile fonksiyonel ilişkileri belirlemek, olarak karşımıza çıkmaktadır.

Yeniden tanımlama: Üretim çıktıları ve onların analizi, simüle edilen sistem tasarımlarının performans ölçülerini tahmin etmek için kullanılmaktadır. Tamamlanan çıktı analizine bağlı olarak analist, ilave çıktılara ihtiyaç olup olmadığına ve bu ilave denemelerin hangi tasarımda yapılacağına karar vermelidir.

Dökümantasyon ve simülasyon sonuçlarının sunulması: Bir modelin dökümantasyonunun hazırlanmasının nedeni, model kullanıcılarının performans girdi ve çıktı parametreleri arasındaki ilişkileri ve bazı performans çıktı ölçülerini "optimize" eden girdi parametrelerini belirlemek için model parametrelerini değiştirmek istemeleridir.

Tüm analizin sonuçları, açık ve kısa olarak rapor edilmelidir. Bu, model kullanıcılarına (karar vericiler) son formülasyonu, belirtilen alternatif sistemleri, alternatiflerin karşılaştırıldığı kriterleri, denemelerin sonuçlarını ve problemin uygun çözümünü gözden geçirme imkânı vermektedir. Ayrıca kararlar daha yüksek bir düzeyde onaylanırsa rapor, model kullanıcı veya karar verici için bir onay aracı sağlar ve modelin güvenilirliğine ve model kurma prosesine katkıda bulunur.

Uygulama: Uygulama safhasının başarısı, önceki adımların ne kadar iyi bir şekilde uygulandığına bağlıdır. Ayrıca bu başarı, analistin tüm simülasyon prosesi boyunca son model kullanıcısı olarak ne kadar kullanıldığına bağlıdır. Eğer model kullanıcı, model kurma prosesi boyunca çalışmışsa ve bu kullanıcı modelin yapısını ve çıktılarını anlarsa, başarılı bir uygulama ihtimali de artar. Aksine, model ve modelin önemli kabulleri doğru olarak açıklanmamışsa, simülasyon modelinin uygunluğuna rağmen, uygulama başarısız olacaktır.

Bir simülasyon etüdünde problemin formülasyonu, amacın ve tüm projenin belirlenmesi adımlarından oluşan kısım, buluş ve adaptasyon periyodudur. Problemin ilk tanımı genellikle belirsizdir ve başlangıç amaçları yeniden belirlenmeli ayrıca orijinal proje planı iyi bir şekilde oluşturulmalıdır (Banks ve Carson, 1984). Model kurma, veri toplama, program yazma, programı onaylama, model ve deneysel tasarımı onaylama adımlarında sürekli bir ilişki gereklidir.

Modelin işletilmesi ile ilgili safhalarda simülasyon modeli ile deney yapılması için tamamen uygun bir plana sahip olunmalıdır. Kesikli olay stokastik simülasyonu aslında, istatistiksel bir deneydir. Çıktı değişkenleri, rassal hatalar içeren tahminlerdir ve bu yüzden doğru bir istatistiksel analiz gerektirirler.

2.10 Simülasyonda Zaman İlerletme Teknikleri

Bir simülasyon modeli geliştirilirken, bu simülasyon modeli için bir zaman akış metodu da düşünülmelidir. Zaten simülasyon çalışmalarının birçok, olayları zaman sıralı olan sistemlerle ilgilidir.

Dijital simülasyondaki zamanlama probleminin nedeni ise, gerçek bir sistemin bileşenlerinin eş zamanlı olarak işlemelerine karşın, simüle edilen sistemin bileşenlerinin sıralı olarak işlemeleridir. Çünkü Bilgisayarların çoğunluğu bir zaman anında sistem bileşenlerinden yalnız bir tanesini gözönüne alabilir. Bu ifade paralel mantıkla çalışan çok

işlemcili bilgisayarlar için geçerli değildir. Ancak günümüzde paralel mantıkla çalışan çok işlemcili bilgisayarlar yaygın olarak kullanılmamaktadırlar.

Gerçek sistemlerde olaylar sistemin farklı parçalarında aynı anda oluşabildiklerinden ve sistemin değişik parçalarında bir bağımlılık olduğundan dolayı simülasyon modellerinde zaman akış mekanizmasının kurulması oldukça önemlidir. Ancak bu şekilde, sistem bileşenlerinin simüle edilen performansları zaman içinde senkronize edilebilirler.

Simülasyon modellerinin çoğu, belli bir zaman süresi boyunca sistem performansının değerlendirilmesi ile ilgili olduğundan, model tasarımlanırken ve simülasyon dili seçilirken en önemli noktalardan birisi de zaman tutma yöntemi olmaktadır. Simülasyonda zaman tutmanın iki yönü vardır (Erkut, 1991):

- Zamanı ilerletmek veya sistem zaman durumunu güncelleştirmek
- Değişik elemanların ve olay oluşumlarının senkronizasyonu

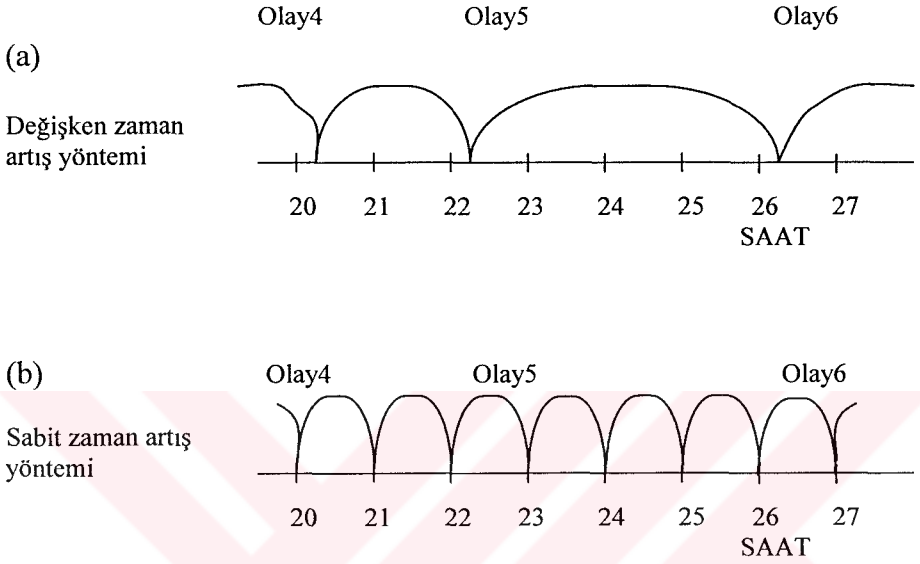
Herbir elemanın faaliyetleri, diğer elemanların durumuna ve faaliyetlerine bağlı olduğundan, bunlar zaman içinde koordine edilmeli veya senkronize edilmelidir. Buna göre model, simülasyon zamanı boyunca ilerleyecek biçimde tasarımlanmalıdır ki, ancak bu şekilde olaylar belirli bir sırada ve belirli zaman aralıklarında meydana gelebilsin.

Simülasyon modellerinde zaman akışını sağlamak için iki temel yöntem vardır. Bunlar düzgün zaman artışlı yöntem ve değişken zaman artışlı yöntemdir.

2.10.1. Sabit zaman artışlı yöntem

Bu yöntemde simülasyon modelinin zaman adımları eşit aralıklardan oluşmaktadır. Sürekli sistemlere yapılan süreksiz yaklaşımların kabul edilebilmesi için, zaman artışlarının boyutu yeteri kadar küçük seçilmelidir. Zaman akışının denetlenmesi için ana zaman (simülasyon zamanı) model içine dahil edilmiştir. Bu yaklaşımda simülasyon genellikle sıfır anından başlar ve belirlenmiş olan sabit bir zaman aralığı geçtikten hemen sonra, herhangi bir olayın meydana gelip gelmediğine bakılır. Eğer herhangi bir olay meydana gelmiş ise, tüm

etkileşimler kayıt edilir ve değişkenlerin yeni değerleri hesap edilir. Daha sonra ana zaman artış miktarı ilerletilir ve proses tekrar edilir. Şekil 2.6b'de sabit zaman artışlı yöntem görülmektedir (Watson ve Blackstone, 1990). Burada sistem zamanı, daha önceden belirlenmiş sabit uzunluklu zaman aralıklarında güncelleştirilir ve simülasyon zaman içinde olaylara da basarak ilerler.



Şekil 2.6 Değişken ve sabit zaman artış yöntemi

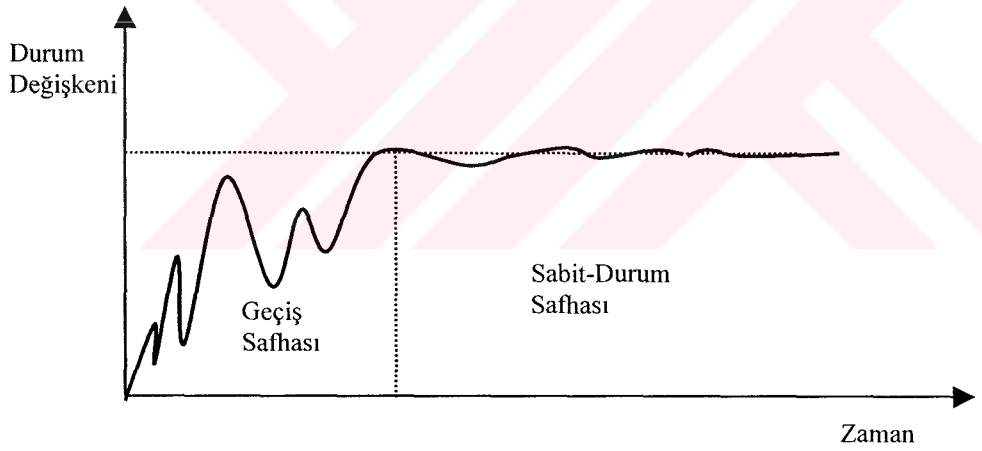
2.10.2.Değişken zaman artışlı yöntem

Bu yöntemde simülasyon zamanı bir olaydan bir sonraki olaya ilerleyerek artış sağlar. Bu durumda çoğunlukla artış adımları düzgün olmamaktadır. Şekil 2.6a'da değişken zaman artışlı yöntem görülmektedir. Herbir zaman adımının büyüklüğü, olayların meydana gelişleri arasındaki zaman farkına eşittir. Simülasyon sıfır anında başlar. Sistem bileşenlerinin simüle edilen performanslarının sonucu olan olayların oluşma zamanları belirlenir. Ana zaman, en erken olayın (bir sonraki ilk olay) oluşum zamanına getirilir. Gerekli hesaplamalar ve değişken değerlerinin güncelleştirilmesi yapılır. Daha sonra ana zaman, yani simülasyon saati bir sonraki en erken olayın oluşum zamanına getirilir ve bu proses, simülasyon süresince tekrarlanır.

2.11. Simülasyonda Başlangıç ve Denge Koşulları

Bir simülasyon modelini çalıştırmaya başlamak için seçilen en genel yol, sıfır başlangıç zamanında ve sistemin boş olduğunu kabul etmektir. Gerçek dünya modellerinin çoğunda bu koşul, sadece sistemin ilk kez çalıştırılması halinde geçerlidir. Örneğin bir hastahane gerçekte hiçbir zaman boş olmayacaktadır. Bir havaalanında yerde ve havada daima uçaklar olacaktır. Dolayısıyla, bir simülasyon modeli çalıştıktan bir süre sonra ancak, gerçek sistemi daha iyi bir şekilde temsil etmeye başlayacaktır.

Bir simülasyon etüdünde, bir durum değişkeninin simülasyon süresi boyunca aldığı değerler gözlendiğinde karşımıza iki safha çıkmaktadır. Bu iki safha, geçis safhası ve sabit-durum safhasıdır. Geçiş safhası, sistem ilk olarak çalıştığında ortaya çıkan sistemin geçici bir davranışıdır. Sistemin gerçek davranışı, ancak sabit-durum safhasında ortaya çıkmaktadır. Simülasyonun işletimi sırasında durum değişkeninin geçiş ve sabit-durum safhasındaki değişimi Şekil 2.7'de görülmektedir.



Şekil 2.7 Geçiş ve sabit-durum safhasının bir görünümü

Verilerin sahip olduğu eğilimin etkisini minimize etmek için, geçiş safhasının etkisini azaltmak üzere en azından üç yol vardır (Halaç, 1982):

(1) Mümkün olduğu kadar fazla ve yeterli miktarda bilgisayar koşumu kullanmak. Bu ise, geçiş safhasından elde edilen verilerin (ilgili değişkenin değerleri) ortalamaya olan etkilerini azaltarak, denge koşulu verisinden elde edilen ortalamayı etkilemeyecektir.

(2) Başlangıç safhasının uygun bir parçasını atmak.

(3) Tipik denge koşulu olan ilk başlangıç şartlarını belirleyerek bunları sisteme dahil etmek ve bu şekilde de geçiş safhasının etkisini azaltmak.

Geçiş safhasını elde edilen sonuçlardan çıkarmak için ve bu tez çalışmasında da kullanılan ve Welch(1981; 1983) tarafından geliştirilen, geçiş şartlarından sabit durum şartlarına geçiş noktasının hesaplanma yöntemi dört adımdan oluşmaktadır. Bu aşamalar aşağıda verilmiştir.

Adım 1. Birbirinden bağımsız ve her birinin uzunluğu m olan, n tane ($n \geq 5$ olmak şartıyla) simülasyon çıktısı oluşturulur. Simülasyon uzunluğu olan m, başlangıçta simülasyonu yapan analistin kararına bağlı olmaktadır. Daha sonra, $(Y_1, Y_2, \ldots, Y_m)$ uzunluğu, belirlenmemiş simülasyon uygulamalarının, ilk koşumundaki stokastik işlem çıktısı olarak kabul edilir ve $j = 1, 2, \ldots, n$; $i = 1, 2, \ldots, m$ olmak üzere, Y_{ji} değerleri hesaplanır. Bu değerler Şekil 2.8'de gösterildiği gibi bir matris formu haline getirilir (Law ve Kelton, 1991).

Simülasyon Tekrarı	
1	$Y_{11}, Y_{12}, Y_{13}, Y_{14}, \ldots, Y_{1,m-2}, Y_{1,m-1}, Y_m$
2	$Y_{21}, Y_{22}, Y_{23}, Y_{24}, \ldots, Y_{2,m-2}, Y_{2,m-1}, Y_{2,m}$
:	: : :
:	: : :
N	$Y_{n1}, Y_{n2}, Y_{n3}, Y_{n4}, \ldots, Y_{n,m-2}, Y_{1n,m-1}, Y_{n,m}$
Ortalama Değerler	$\bar{Y}_1, \bar{Y}_2, \bar{Y}_3, \bar{Y}_4, \ldots, \bar{Y}_{m-2}, \bar{Y}_{m-1}, \bar{Y}_m$
Hareketli Ortalamalar ($w=1$)	$\bar{Y}_1(1), \bar{Y}_2(1), \bar{Y}_3(1), \ldots, \bar{Y}_{m-1}(1)$

Şekil 2.8 Simülasyon çıktılarının ortalama değerleri ve hareketli ortalamaları

Adım 2. İkinci aşamada, hesaplanan Y_{ji} değerlerinin ortalamaları alınır (bağıntı 2.1).

$$\overline{Y}i = \sum_{j}^{n} \frac{y_{ji}}{n} \quad , \qquad i = 1,2,3,..., \; m \tag{2.1}$$

Hesaplanan bu ortalama değerlerin ($\overline{Y}_1$, $\overline{Y}_2$, $\overline{Y}_3$, $\overline{Y}_4$, . . . , $\overline{Y}_{m-2}$, $\overline{Y}_{m-1}$, $\overline{Y}_m$) ortalamalarının [$E(\overline{Y}i) = E(Yi)$] ve varyanslarının [$Var(\overline{Y}i) = Var(Yi)/n$] olduğu kabul edilir. Daha sonra bu ortalama değerlerin, yatay eksende simülasyon uzunluğu m, düşey eksende ise $\overline{Y}_1$, $\overline{Y}_2$, $\overline{Y}_3$, $\overline{Y}_4$, . . . , $\overline{Y}_{m-2}$, $\overline{Y}_{m-1}$, $\overline{Y}_m$ değerleri alınarak grafiği çizilir.

Adım 3. Bu kademede ise hareketli ortalama olarak ifade ettiğimiz $\overline{Y}i(w)$ değerleri hesaplanır. Burada w eğri düzeltme faktörüdür ve $w \leq (m/2)$ olacak şekilde $\overline{Y}_i(w)$ değerleri aşağıdaki çevrim şarta bağlı olarak bağıntı 2.2 veya bağıntı 2.3 yardımıyla hesaplanır.

$$Yi(w) = \begin{cases} \dfrac{\sum_{s=-w}^{w} \overline{Y}_{i+s}}{2w+1} & \text{eğer } i = w+1, \ldots, m-w \text{ ise} \qquad (2.2) \\ \dfrac{\sum_{s=-(i-1)}^{i-1} \overline{Y}_{i+s}}{2w+1} & \text{eğer } i = 1,2, \ldots, w \text{ ise} \qquad (2.3) \end{cases}$$

Adım 4. Son aşamada ise $\underline{Y}(w)$, i = 1, 2, 3, … , m-w değerlerine ait grafikler çizilir ve $\overline{Y}_1(w)$, $\overline{Y}_2(w)$, ..., $\overline{Y}_{m-w}(m-w)$ değerlerinin birbirlerine, ya da eğrinin doğrusal hale yaklaştığı nokta geçiş şartlarından sabit durum şartlarına geçilen nokta olduğu kabul edilir. Simülasyoncunun isteğine bağlı olarak, bu iki bölgeden herhangi

birisi baz alınarak simülasyon uzunluğuna karar verilir. Simülasyon işletimi ise, baz alınan bölgedeki değerlerle yapılır.

Bu şekilde simülasyon çıktılarının güvenilirliği arttırılmış ve daha sağlıklı simülasyon çıktıları elde edilmiş olur.

3. YAPAY ZEKÂ ve UZMAN SİSTEMLER

Son yirmi, otuz yıl içerisinde bilgisayar teknolojisi dev adımlarla gelişmiştir. Önümüzdeki yirmi, otuz yıl içerisinde de hız, kapasite ve mantık tasarımında büyük ilerlemeler kaydedileceği konusunda da pek az kuşku vardır. Birkaç sene önce kullanılan bilgisayarlar şu anda gözümüze son derece hantal ve yavaş görünmektedir.

Daha önceleri sadece insanın düşünce sisteminin alanında gerçekleştirilebilen sayısız işlemler, bugün bir insanın asla erişemeyeceği hız ve doğrulukta, bilgisayarlar tarafından gerçekleştirilmektedir (Penrose, 1989).

Etkileşimli insan-makina arayüzlerinin ve ilgili veri tabanlarının ortaya çıkması ile, bilgisayarlar ham veri işleme makinalarından, gerçek karar destek sistemlerine dönüşmüştür. Etkileşim, bilgisayar ile daima doğal bir ilişkiyi desteklemektedir. İlişkili veri tabanı sistemleri ise, depolanan farklı tipteki veriler arasındaki ilişkilerin bulunması ve kullanılmasını sağlar. Bu kabiliyetler, direkt olarak analize ve karar verme prosesine yardım eden bilgisayar destekli sistemlerin gelişmesini mümkün kılmıştır.

3.1.Yapay Zekâ ve Gelişimi

Yapay zekâ, belirli sınırlı alanlar için insan düşüncesinin karakteristiklerinden olan, öğrenme kabiliyeti, sonuçlandırma, problemleri çözme ve konuşma dilini anlamayı örnekleyebilen bilgisayar sistemlerinin tasarımı ile ilgilidir.

Yapay zekâ ile ilgili çeşitli kaynaklardaki tanımlardan bazılarını şu şekilde sıralayabiliriz:

"Yapay zekâ, zeki davranışın incelenmesidir" (Genescreth, 1987).

Feigenbaum ve McCorduck (1983) tarafından yapılan bir tanıma göre yapay zekâ, "insanların birbirlerinde zekice olarak kabul ettikleri davranışlara sahip bilgisayarların yapılmasıyla ilgili bir bilgisayar bilimleri alt alanıdır".

Bu tanımlarda geçen "zeki davranış" kelimesi sadece problem çözme, santranç oynama ve teorem ispatları gibi yüksek oranda zekâ gerektiren davranışlarla kısıtlı olmayıp, cisimleri tanıma ve basit bazı yazıları anlayabilme gibi, normalde yüksek zekâ gerektirmeyecek davranışları da içermektedir. Hatta cisimlerin ve seslerin algılanması gibi problem çözme ile direkt ilgili olmayan bazı faaliyetlerin de yapay zekânın ilgi alanına dahil edilebileceği Garnlam (1988) tarafından belirtilmiştir.

İnsan, yapay zekâ için en iyi modeldir. Çünkü insan, mükemmel bir biyolojik öğrenme makinasıdır. İnsanın hataları, kapasite limitleri ve unutma mekanizmaları dünyanın en geniş, en esnek ve en verimli veri tabanını ve bilgi alma sistemini oluşturmaktadır (James, 1984).

Bir başka tanıma göre yapay zekâ, bilgisayarları çekici kılmak, insan zekâsına sahip bilgisayarlar geliştirmek, insanın zeki davranışlarını taklit eden makinalar (robotik) yapmaktır. Yapay zekâ ile uğraşan bilim adamları daima bilgisayar programları ile uğraşırlar ve programlar geliştirirler. Bu programlar bir anlamda, insanın bir problem karşısındaki düşünmesini gerçekleştirirler ve problemi çözerler, ve bu programlar belirli bir ölçüde insanın muhakeme yeteneğine de sahiptirler ve problemi çözerken bu yeteneği de kullanırlar.

Yapay zekâ adı ilk defa 1956 yılında Jonh McCarty tarafından Amerika'da "Machine Intelligence" kongresinde ortaya konmuş bir kavramdır. Tam doğru bir kavram olmamasına rağmen bütün dünyada yerleşmiş ve kabul görmüştür (Bayazıt, 1992a; 1992b).

1960'larda yapay zekâ ile uğraşan bilim adamları, çok geniş problem gruplarını çözmek için genel metotlar bularak, karmaşık düşünme işlemini simüle etmeye çalışmışlar ve bu metotları genel amaçlı programlarda kullanmışlardır. Bu genel amaçlı yapay zekâ programları da, daha geniş kapasiteli bilgisayarları gerektirmiştir. Çünkü bu şekilde programlar ya belleği dolduruyor, ya da çok yavaş çalışıyorlardı. 1970'lerde ise problemlerin formüle edilişinin tanımlanması gibi teknikler kullanılmaya başlanmıştır. Fakat, bilgisayar zamanını çok almayacak ya da bilgisayar belleğinde çok yer kaplamayacak çözümlerin başarılı olarak nasıl bulunacağı konusunu çözmek, o yıllar için

erken sayılabilirse de, bu stratejilerde bazı başarılara ulaşılmış, fakat genelde elde edilen sonuçlar uygulamaya geçememiştir. Ancak 1970'li yılların sonlarına doğru, yapay zekâ ile uğraşan bilim adamları bazı önemli sorunlar olduğunu fark etmişlerdir. Bunlar da, bir programı güncel yapabilmek için, problem alanı kapsamında, birçok yüksek düzeyli bilgi yüklemesinin gereğidir. Bunun gerçekleştirilmesi ile birlikte özel amaçlı, çok dar bir problem alanında uzman olan bilgisayar programlarının ve bilgisayar sistemlerinin geliştirilmesi sağlanmıştır. Sınırlı bir problem alanında, yüksek düzeyli bilgi ile desteklenerek oluşturulmuş zeki bilgisayar programlarıyla da, yapay zekâ'nın bir alt alanı olan uzman sistemler ortaya çıkmıştır.

İnsanlar bilgiyi nasıl elde etmekte, nasıl organize etmekte ve nasıl kullanmaktadır? 1950'lerdeki başlangıcından bu yana yapay zekâ, bu proseslerin anlaşılmasıyla ilgilenmiştir. Araştırma ve uygulamalar iki genel sınıfa ayrılmaktadır (Shannon, vd. 1985):

(1) İnsanların yaptıkları şekilde, doğal insan kabiliyetlerini (görme, konuşma vb.) yansıtan veya taklit eden araştırma ve uygulamalar,
(2) Gerçek uzman tarafından kullanılan veya tecrübelerin sonuçlarını taklit eden araştırma ve uygulamalar.

İkinci uygulama sınıfı, genelde "uzman sistemler" olarak adlandırılan uygulamaları kapsamaktadır. Bu uygulamalar, özel olarak eğitilmiş veya yetenekli insanlar tarafından normal olarak yapılan işlerin otomasyonuyla ilgilidir.

3.2.Yapay Zekâda Yaklaşımlar

Yapay Zekâ ile uğraşan araştırmacıların çalışmaları dört grupta toplanabilir (Russel ve Norvig, 1995). Düşünce süreçleri ve akıl yürütme ile davranış, bu yaklaşımların iki boyutudur. Yapay zekâda dört grupta toplanan çalışmalar ise şunlardır:

- İnsanlar gibi düşünen sistemler
- İnsanlar gibi davranan sistemler
- Rasyonel düşünen sistemler

- Rasyonel davranan sistemler

Bu yaklaşımları kısaca açıklayalım.

İnsan gibi davranan sistemler: Yapay zekâ araştırmacılarının baştan beri ulaşmak istediği ideal, insan gibi davranan sistemler üretmektir. Turing (1950), zeki davranışı, bir sorgulayıcıyı kandıracak kadar bütün bilişsel görevlerde insan düzeyinde başarım göstermek olarak tanımlamıştır. Bunu ölçmek için de, "*Turing Testi*" denilen bir test önermiştir. Turing testinde denek, sorgulayıcısıyla bir terminal aracılığıyla haberleşir. Eğer sorgulayıcı, deneğin insan mı bilgisayar mı olduğunu anlayamazsa, denek testi geçmiş sayılır.

Turing, testini tanımlarken zekâ için bir insanın fiziksel benzetiminin gereksiz olduğunu düşündüğü için sorgulayıcıyla bilgisayar arasında doğrudan fiziksel temastan söz etmekten kaçınmıştır.

Burada önemli olan husus, bilgisayarda zeki davranışı üreten sürecin insan beynindeki süreçlerin modellenmesiyle elde edilebileceği gibi, tamamen başka prensiplerden de hareket edilerek üretilmesinin olası olmasıdır.

İnsan gibi düşünen sistemler: İnsan gibi düşünen bir program üretmek için insanların nasıl düşündüğünü saptamak gerekir. Bu da psikolojik deneylerle yapılabilir. Yeterli sayıda deney yapıldıktan sonra, elde edilen bilgilerle bir kuram oluşturulabilir. Daha sonra bu kurama dayanarak bilgisayar programı üretilebilir. Eğer programın giriş/çıkış ve zamanlama davranışı insanlarınkine eşse, programın düzeneklerinden bazılarının insan beyninde de mevcut olabileceği söylenebilir.

İnsan gibi düşünen sistemler üretmek, bilişsel bilimin araştırma alanına girmektedir. Bu çalışmalarda asıl amaç, genellikle insanın düşünme süreçlerini çözümlemede bilgisayar modellerini bir araç olarak kullanmaktır (Akın, 1995).

Rasyonel düşünen sistemler: Bu sistemlerin temelinde mantık yer alır. Burada amaç, çözülmesi istenen sorunu mantıksal bir gösterimle betimledikten sonra, çıkarım kurallarını kullanarak çözümünü bulmaktır.Yapay zekâda çok önemli bir yeri olan mantıkçı, gelenek zeki sistemler üretmek için bu çeşit programlar üretmeyi amaçlamaktadır.

Bu yaklaşımı kullanarak gerçek sorunları çözmek istenildiğinde, iki önemli engelle karşılaşılır. Mantık, formel bir dil kullanr. Gündelik yaşamdan kaynaklanan, çoğu kez de belirsizlik içeren bilgileri mantığın işleyebileceği bu dille göstermek hiç de kolay değildir. Bir başka güçlük de, en ufak sorunların dışındaki sorunları çözerken kullanılması gerekecek bilgisayar kaynaklarının üstel olarak artmasıdır.

Rasyonel davranan sistemler: Amaçlara ulaşmak için inançlarına uygun davranan sistemlere rasyonel sistemler denir. Bir ajan, algılayan ve harekette bulunan bir şeydir. Bu yaklaşımda YZ, rasyonel ajanların incelenmesi ve oluşturulması olarak tanımlanmaktadır.

Rasyonel bir ajan olmak için gerekli koşullardan biri de doğru çıkarımlar yapabilmek ve bu çıkarımların sonuçlarına göre harekete geçmektir. Ancak, yalnızca doğru çıkarım yapabilmek yeterli değildir. Çünkü bazı durumlarda doğruluğu ispatlanmış bir çözüm olmadığı halde, gene de bir şey yapmak gerekebilir. Bunun yanında çıkarımdan kaynaklanmayan bazı rasyonel davranışlar da vardır. Örneğin, sıcak bir şeye değince, insanın elini çekmesi bir refleks harekettir ve uzun düşünce süreçlerine girmeden yapılır.

Bu yüzden yapay zekâyı rasyonel ajan tasarımı olarak gören araştırmacılar, iki avantaj öne sürerler. Birincisi bunun, "düşünce yasaları" yaklaşımından daha genel olması, ikincisi ise bilimsel geliştirme yöntemlerinin uygulanmasına daha uygun olmasıdır (Akın, 1995).

3.3 Yapay Zekânın İlgilendiği ve Kullanıldığı Alanlar

Bir işin yapılması ya da problemin çözülmesi için bilgisayar kullanılmaya karar verildiğinde eğer problem, matematiksel fonksiyonlar olarak tariflenemiyor ya da çalışma alanının kısıtları tanımlanamıyorsa, yapay zekâ çalışma alanı içerisindeyiz demektir. Bu durumda bilgisayarı, uzman insanın problemlere yaklaşımlarını taklit edecek şekilde

programlamamız gerekir. Eğer insanın bilgi işleme düzeni tam olarak bilinemiyorsa, bu düzeni yaklaşım olarak taklit etme yoluna gidilir.

Yapay zekânın ilgilendiği alanlar, genel olarak aşağıda görüldüğü gibi sıralanabilir:

- Bilgisayarın görmesi: Robot çalışmalarında kaydedilen gelişmeler sonucu ilgi, bu makinaların karşılarına çıkan veya bulundukları ortamda olan cisimleri farkedip onları tanımlayabilmelerinin sağlanabilmesi üzerine yoğunlaşmıştır,
- Konuşma sesinin üretilmesi,
- Ses algılama:Makinaların harici sesleri (insan sesi ve herhangi başka bir makinanın sesi) tanımlayabilmesi,
- İnsan dilinden anlama ve tercüme yapabilme,
- Düşünme, sonuç çıkarma ve problem çözme: Bu konu yapay zekânın en geniş alanıdır. Bu alana giren bazı konuları şöyle sıralayabiliriz: Matematiksel teorem ispatlama, zekâ oyunları, öğrenen makinalar, uzman sistemler.

Bu alanlara ek olarak bazı yapısal elemanlar da araştırma konusudur. Bunlar içerisinde arama, doğal dili işleme, desen karşılaştırma ve algılama, mantık, belirsizlik ve bulanık mantık sayılabilir (Agrawala, 1977).

Aşağıda ise yapay zekânın bazı uygulama sahaları kısaca tanıtılmıştır:

Arama: Arama terimi, yapay zekâ uygulamalarında bir probleme çözümler bulma olarak kullanılır. Buradan çıkarılacak anlam bir veri tabanından özel bir bilgi parçacığının bulunması değildir. İki şehir arasındaki en kısa yolun bulunması ya da matematiksel bir teoremin doğruluğunu ispatlama gibi problemlerin çözümü, bu çalışma alanında yapılan çalışmalara örnek olabilir. Bu alanda yapılan çalışmalar sonucu birçok arama yöntemi geliştirilmiştir (Schildt, 1987).

Doğal dili işleme: Yapay zekânın en çok araştırma yapılan alanıdır. Bu konu üzerinde yapılan çalışmaların oldukça fazla olmasının nedeni, doğal dil işleme çalışmalarının sonuçlarının, insanların bilgisayarlarla doğrudan iletişim yeteneğini bilgisayara

kazandırmak olmasıdır. Bu amaca ulaşmadaki zorluk, insanların kullandıkları dilin çok karmaşık ve çok boyutlu olmasındandır. Bu problemin zorluk gösteren diğer bir yönü de, bilgisayarlarla insanların kuracağı iletişimin çok benzer yapıları içermesidir. Aynı cümlenin farklı durumlarda farklı anlamlar taşıması bu zorluklara örnek verilebilir (Schildt, 1987).

Desen gösterimleri: Desen işleme ve şekillerle çalışma, robotlar ve şekilsel süreçlerin analizlerini içeren uygulamalar için önemlidir. Örneğin, "Bir televizyon ekranı üzerindeki nesnelerin başlangıç ve bitiş yerlerini veya hangi nesnelerin hangi nesneler üzerine konulabileceğini bilgisayara nasıl tanıttırabiliriz?" sorusu gündeme geldiğinde bu tür çalışmalara olan ihtiyaç kendini gösterir. Gölgelerin nesnelerden ayırdedilebilmesi, bu çalışmaların odak noktasıdır (Turban, 1990).

Robotlar: Robot uygulamaları, yapay zekâda hareketlerin nasıl kontrol edilebileceği konusunda yapılan çalışmalara verilen addır. Montaj hatlarında kullanılan endüstriyel robotlar bu çalışmaların ürünüdür. Bu alanda yapılan çalışmaların en önemli uğraşımı, kesikli hareketlerden, doğal hareketler kümesi üretebilme olmuştur. Genel amaçlı robotlar üretme çalışmaları içerisinde bulunan araştırmaların en önemli sorununu, doğal dilin robotlar tarafından anlaşılır hale getirilmesi, değişen çevre koşulları ve beklenmedik olaylar karşısında robotların davranışlarının tarif edilememesi oluşturur (Turban, 1990).

Öğrenme: Yapay zekânın ilginç çalışma alanlarından birisi de, öğrenme yeteneğini bilgisayara öğretme yönünde yapılan çalışmalardır. Bu alanda yapılan çalışmalar, yapılması istenen şeylerden, gözlemlerden ve oluşan hatalardan yeni şeyler öğrenen bilgisayar yazılımları oluşturma çalışmaları olarak tanımlanabilir. Öğrenmenin diğer bir anlamı da, tecrübelerden yararlanabilir bilgisayarlar geliştirmektir (Schildt, 1987).

Mantık: Diğer bir yapay zekâ çalışma alanı, mantığın standart kurallarını kullanarak bir önermenin mantıksal olarak doğruluğunu gösteren programlar üretmektir. Bu alanda yapılan çalışmalar, matematiksel teoremlerin ispatı, sembolik mantık gibi alanlarda yoğunlaşır (Turban, 1990).

Bulanık kümeler: Karar verici bazen vereceği kararları, elinde net bilgiler olmadan da vermek durumundadır. Bu tür bilgiler, açık olmayan ve yapıya bağlı olarak bir olasılık da içeren durumlar oluşturmaktadır. Söz konusu bu bilgileri kullanarak istenen sonuçlara ulaşabilen makinalar ve programlar geliştirme çalışmaları, bu alan içerisinde düşünülebilir (Turban, 1990).

Uzman sistemler: Bu alan yapay zekânın en önemli uygulama alanlarından biridir. Bu konuyla ilgili geniş bilgiler uzman sistem bölümünde açıklanmıştır.

3.4.Uzman Sistem Kavramı

Uzman sistemler, yapay zekâ bilim dalının bir alt bilim dalı olarak ortaya çıkmış, oldukça ilgi görmüş ve bu konudaki çalışmalar da önemli bir yoğunluk kazanmıştır.

Uzman sistem uygulamaları ilk yapay zekâ araştırmalarından farklı olabilir. Çünkü bu şekildeki uygulamaların temel amacı, verilen bir sonuca ulaşmak için uzman bir insan tarafından kullanılan temel bir mekanizmayı anlamak değil, uzman bir insanın çıkardığı sonuçları tutarlı bir şekilde benzetmektir. "Uzman" veya "Bilgi Tabanlı Sistemler", belirli bir alanda çok sayıdaki uzmanın tecrübelerini bir kurallar serisi şeklinde derlemek üzere tasarlanırlar. Buradaki kurallar, belirli bir problemle ilgili bir hareket tarzı olarak kullanıcı için sonuçlar ve kabuller çıkartmak (veya otomatik olarak elde etmek) üzere kullanılırlar.

Özellikle karmaşık problemleri sezgisel algoritmalarla çözüp, "optimal" çözüm yerine "en iyiye yakın" çözümleri tercih ederek kolay işletilir ve anlaşılır sistemlerle çalışmak, kişileri uzman sistem yaklaşımına yöneltmiştir.

Uzman sistemler hakkında değişik kaynaklarda farklı tanımlara rastlamak mümkündür. Bunlardan bazılarına aşağıda yer verilmiştir:

"Uzman sistem, özel bir takım problemlerin çözümünde, uzmanların bilgisini ve düşünme işlemini taklit etmeyi amaçlayan, danışman bilgisayar programlarıdır" (Turban, 1990).

"Uzman sistem, uzman bir insanın davranışlarını taklit ederek belirli bir alanda uzmanlık gerektiren karmaşık problemlerin çözümü için bir veya birden fazla yaklaşım öneren akıllı bilgisayar programlarıdır" (Schild, 1987).

Bir başka tanıma göre de, sınırlı bir problem alanında yüksek seviyede bir performans elde etmek için, uzman bilgisini kullanan bilgisayar programlarına "uzman sistem" denilir. Uzman sistemler, sadece akademik bir nosyon değildir ve nitekim bunların gerçek dünyada da birçok alanda uygulamaları gerçekleştirilmiştir. Günümüze kadar geliştirilen uzman sistemlerin uygulama alanları olarak aşağıdakiler sayılabilir:

- Ticaret,
- Vergilerin hazırlanması ve planlanması,
- Borç verme ve kredi limitlerinin belirlenmesi,
- Çeşitli hastalıkların teşhisi ve tedavisi,
- Karmaşık makinaların bakımı,
- Hava yollarında uçakların atanması ve uçuşların çizelgelenmesi,
- Gemilere ve uçaklara kargo yükleme,
- Bilgisayarların tasarlanması ve konfigürasyonu,
- Baskılı devrelerin tasarımı ve yerleştirilmesi,
- Tesis tasarımı,
- Kimyasal proseslerin kontrolu ve çizelgelenmesi,
- Madenlerin içerisindeki artıkların analizi ve izlenmesi,
- Hızlı yemek servislerinin geliştirilmesi.

Günümüze kadar geliştirilmiş olan birçok uzman sistemden bazıları uygulama alanlarıyla birlikte Çizelge 3.1'de gösterilmektedir (Watson ve Blackstone, 1989).

3.5. Bir Uzman Sistemin Genel Yapısı

Bir uzman sistemin geleneksel programlardan en temel farkı, geleneksel programlar veri işlerken, uzman sistemlerin bilgi işlemesidir. Bu bölümde bir uzman sistemin genel yapısı ve bileşenleri anlatılmaktadır. Şekil 3.1'de görüldüğü gibi tipik bir uzman sistemin en

temel bileşenleri olarak, kullanıcıyla etkileşimi sağlayan bir giriş-çıkış arayüzü, çıkarım mekanizması, bilgi tabanı, öğrenme modülü gibi elemanlar sayılabilir (Patterson, 1990).

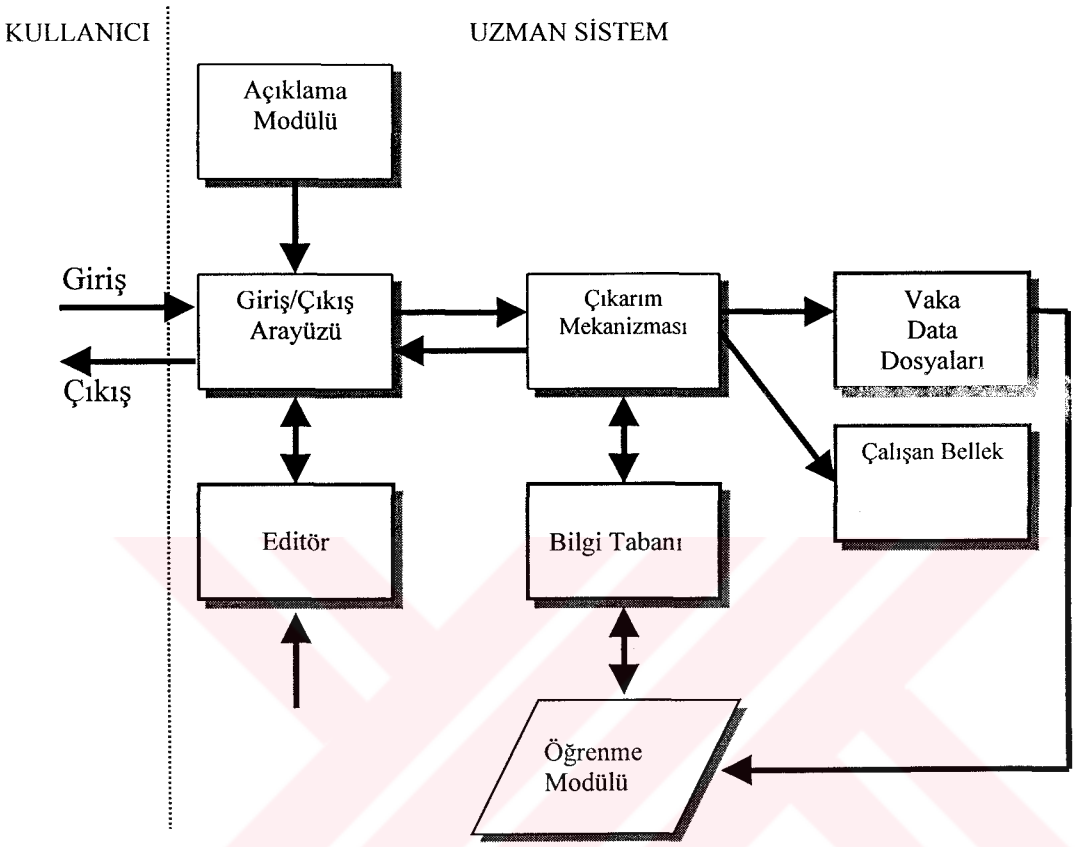
Çizelge 3.1.Geliştirilmiş olan bazı uzman sistemler

Uzman Sistem	Uygulama Alanı
ACE	Telefon kablosu problemlerinin belirlenmesi
DELTA	Lokomotif tamir problemlerinin belirlenmesi
DENDRAL	Kütle-spektroskopik analizin yapılması
ExperTAX	Vergilerin hazırlanması ve planlanması
INTERNEST	Dahili hastalıkların teşhisi
MYCIN	Bakteriyel hastalıkların teşhisi
Mentor	Havalandırma sistemleri problemlerinin belirlenmesi
PlanPower	Personel finansal planlamanın yapılması
PUFF	Akciğer hastalıklarının teşhisi
XCON	Bilgisayar konfigürasyonlarının geliştirilmesi

Bir uzman sistem, karar vermeye yardımcı bir kimse gibi kullanılabilmektedir. Bu da ancak çok iyi bir şekilde tasarlanmış bir bilgi tabanı ile gerçekleştirilir ve uzman sistem bir bilgi tabanı ile sürekli etkileşim halinde çalışır. Bundan dolayı da bir bilgi tabanı bütün uzman sistemlerin can damarı, yani en önemli bir parçasıdır. Bir bilgi tabanı, spesifik bir bilgi alanı ile ilgili olgular (gerçekler) ve kurallar olmak üzere iki tür bilgi içermektedir. Bir bilgi tabanında yer alan olgular spesifik bir alandaki farklı durumları göstermektedir. Bilgi tabanı içerisindeki kurallar ise önceden belirlenmiş olan insan sezgilerinin bilgisayara aktarılmış bir halidir (Ignizio, 1991).

Çıkarım mekanizması bir uzman sistemin, kullanıcı tarafından bir giriş/çıkış arayüzü ile etkileşimli olarak soru cevap formunda sorgulama ile girilmiş olan dinamik bilgi ile birlikte, bilgi tabanına depolanmış olan statik bilgiyi (olgular ve kurallar) kullanan bir bileşenidir. Çıkarım mekanizmasının, alan bilgisinin nasıl uygulanacağına karar veren parçasına yorumlayıcı ve çıkarım mekanizmasının, alan bilgisinin farklı parçalarının ne

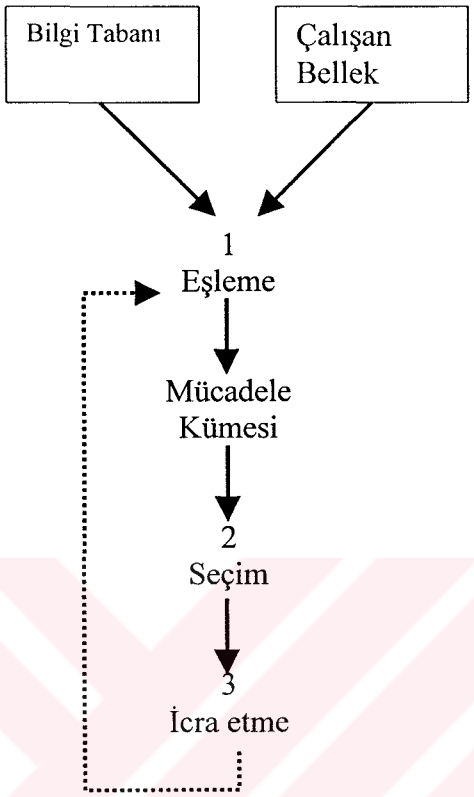
zaman ve hangi düzende uygulanacağına karar veren parçasına da "gösterge" adı verilmektedir.



Şekil 3.1 Tipik bir uzman sistemin görünümü ve bileşenleri

Çıkarım işlemi sürekli tekrar eden üç safha içinde oluşmaktadır. Bu üç safha eşleme, seçim ve icra etmedir. Eşleme safhası esnasında, çalışan bellek içindekiler, bilgi tabanında yer alan gerçekler ve kurallar ile karşılaştırılırlar. Birbiriyle uygun ve aralarında bir mutabakat olan denklikler bulunduğu zaman, karşılaştırılmış uygun kurallar mücadele seti içerisine yerleştirilir. Atanan ve aralarında bir mutabakat olan eşlemeyi bulmak, bir başkasını da bulmayı gerektirebilir. Bir çevrim esnasında birbiriyle mukayese edilecek tüm kurallar mücadele seti içerisine bir kez ilave edildikten sonra, kuralların biri icra edilmek üzere seçilir. Seçilmiş olan kural icra edildikten sonra, kuralın hareket (faaliyet) parçası kuraldan dışarıya taşınır. Bir çıkarım işlemi esnasındaki bu çevrim Şekil 3.2'de görülmektedir

(Patterson, 1990). Bu şekildeki bir çevrim ile karşılaştırmalar yapılarak probleme çözüm bulunmaya çalışılır.



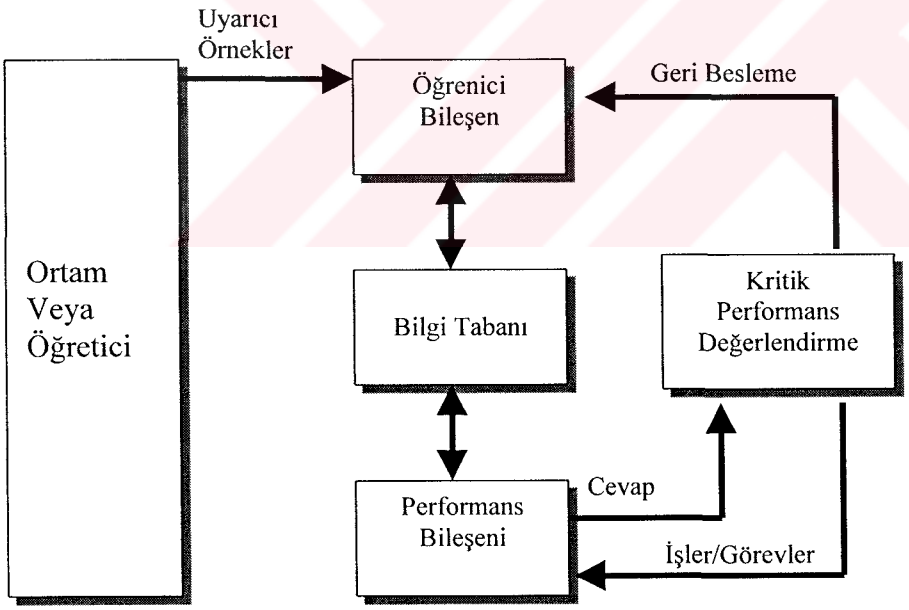
Şekil 3.2 Çıkarım çevrimi

Bir uzman sistemdeki açıklama modülü, istendiği zaman sonuçlandırma işleminin bir açıklamasını kullanıcıya sağlar. Bu modül kullanıcıya, sorgulamadaki nasıl ve niçin cevaplarıyla ilgili bilgileri sunar.

Giriş/çıkış arayüzü, kullanıcıya bir uzman sistem ile iletişim kurmasını sağlayan bir modüldür. Kullanıcı iletişimini, seçimli menüler, doğal dile yakın bir anlatım, grafikler ve etkileşimli diyaloglar aracılığı ile kurar. Burada en önemli amaç, kullanıcıya problemini büyük bir kolaylıkla uzman sisteme aktarmasını temin etmektir. Dolayısıyla giriş/çıkış arayüzlerinin kullanıcıya kullanım kolaylığı sağlaması gerekmektedir.

Bir editör, bir bilgi tabanına ilave edilmek üzere yeni bir kuralın oluşturulmasını, güncelleğini kaybeden bir kuralın silinerek bilgi tabanından çıkarılmasını veya mevcut kuralların güncelleştirilmesini sağlamak amacıyla kullanılır.

Öğrenme modülü ve vaka data dosyaları her uzman sistemde bulunan yaygın birer bileşen değildir. Bu modüller bir bilgi tabanına, oluşturulması veya tasfiye edilmesi hususunda destek olarak kullanılmaktadır. Öğrenme, giriş uyarıcılarının bazı formlarından oluşturulacak yeni bilgi yapılarını gerektirir. Yeni bilgi, bir bilgi tabanının içerisine yerleştirilmeden önce onun kullanımı test edilmiş olmalıdır. Genel bir öğrenme modülü Şekil 3.3'de tanımlanmıştır (Patterson, 1990). Burada ortam, bir öğrenici sistemin tümünün bir parçasını içermektedir. Bir ortam, rassal olarak bir uyarıcı etki sağlayan doğal bir yapı ya da bir öğretici gibi daha organize bir eğitim kaynağı olarak, öğrenici bileşene dikkatlice seçilmiş eğitim örnekleri sağlar. Bazı sistemler için ortam, klavyede çalışan bir kullanıcı olabilir.



Şekil 3.3 Genel bir öğrenme modeli

Bilgi, bilgi tabanında bilgi yapılarını güncelleştirmede ve oluşturmada kullanılan bir öğrenici ortama yüklenir. Bu aynı bilgi, bir performans bileşeni tarafından bir problemi çözmek, bir oyunu oynamak için bilgi tabanının dışına taşınır.

Bir görev veya iş verildiği zaman bir performans bileşeni, işin hazırlanmasında tanımlanmış hareketlere bir cevap üretir. Kritik modül ise, bu cevaplara bağlantılı bir optimal cevabı değerlendirir. Geri besleme, bir performans kabul edilebilir değilse, bununla ilgili durumu öğrenici bileşene bildirir. Eğer sonuçlar uygun ise öğrenme başarılmıştır ve bilgi tabanında bir değişimin yapılmasıyla sistemin performansı sağlanmış olur.

3.6.Uzman Sistemlerin Avantajları

Uzman sistemlerin, gerek problemlere bir insan uzmanın tarzında yaklaşabilmeleri ve gerekse de bilgisayarın gücünden yararlanabilmeleri sayesinde geleneksel karar destek sistemlerinin sınırlarının aşılması mümkün olur. Sargeant (1990), yaptığı bir çalışmasında uzman sistemlerin avantajlarını aşağıdaki gibi sıralamıştır:

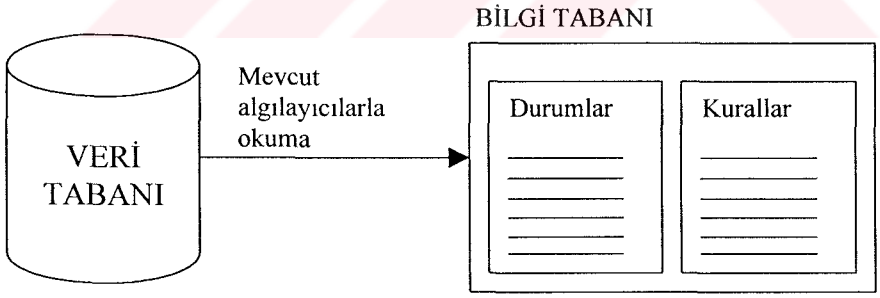
- İşlemlerin etkinliğini ve güvenilirliğini arttırmak,
- İşlemlerin merkezlere yeniden yerleştirilmesine daha fazla bir serbestlik sağlamak,
- Gözden kaçan değerlendirmelerden ve beceri kayıplarından doğan kayıpları azaltmak,
- Bilgi tabanlı şirket oluşumuna zemin hazırlamak,
- Finansal bilgi tabanlarına eğilimi arttırmak,
- Çalışma alanı içindeki birimlere bilgi ulaşımını arttırmak,
- İyi karar verebilme olasılığını ve sıklığını arttırmak ve de kararlılığı yükseltmek,
- Uzmanlığın yayılmasını sağlamak,
- Uzman olmayan kişilerin de uzman sistemler yardımıyla anında, ucuza ve uzman düzeyinde kararlar alabilmesini sağlamak,
- Eldeki verilerden yararlanmayı arttırmak,
- Kişilerin önyargılarına ve o andaki ruh hallerine bağlı kalmadan, mevcut kanıtlarla objektif olarak karar vermek.

Bunların dışında uzman sistemler, ayrıca modüllerden oluştukları için dinamiktirler ve değişikliklere daha rahat uyum sağlayabilirler.

3.7.Uzman Sistemlerde Dış Ortam Arayüzü

Çoğu zaman bir uzman sistem, dış ortamlarla bütünleşik ve etkileşim halinde olmalıdır. Bir uzman sistem bazı durumlarda, algılayıcılardan ve diğer aygıtlardan veya dış veri tabanlarından, dış verilere gereksinim duymaktadır. Bazı diğer durumlarda ise bir uzman sistem, dış aygıtların faaliyetlerini kontrol etmek durumundadır. Bütün bunların nedeni de, bir uzman sistemin ortam hakkında daha fazla bilgi elde etmek veya ortamdaki değişimleri öğrenme ihtiyacından ortaya çıkmaktadır. Ancak bu şekilde bir uzman sistem kendisinin önerdiği bir sonucu kontrol edebilmektedir.

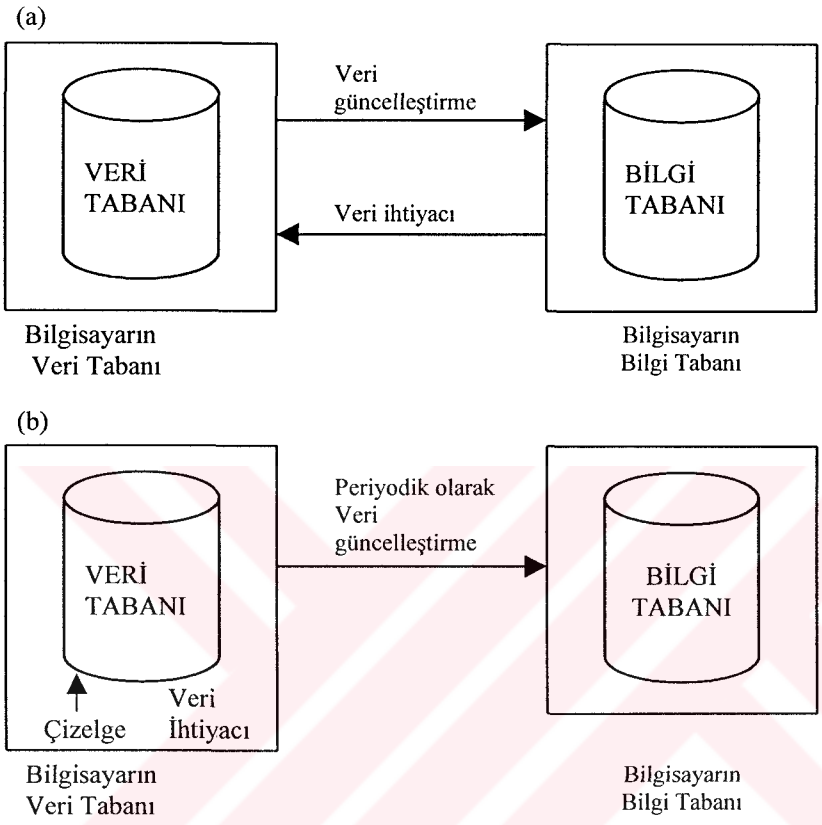
Bir uzman sistemde veri tabanı ile bilgi tabanının bütünleştirilmesi için iki yaklaşım mevcuttur. Eğer mevcut bir veri tabanı varsa, bu durumda genellikle veri tabanından bilgi okuyan bir uzman sistem konfigürasyonu düşünülebilinir. Uzman sistem, veri tabanından okuduğu veriyi daha sonra kendi bilgi tabanı formatı içerisine dönüştürmekte ve ancak bundan sonra bu bilgi üstünde muhakeme yapabilmektedir. Bu durumdaki bir yaklaşım Şekil 3.4'de görülmektedir (Payne ve McArthur, 1990). Bununla birlikte, bir veri tabanının bir bilgi tabanı ile bütünleşik olarak düşünüldüğü bir yaklaşımda veri tabanı uzman sistemin kendi bünyesi içinde olacak şekilde tasarlanmaktadır.



Şekil 3.4 Bir veri tabanından bir bilgi tabanına veri aktarımı

Bir veri tabanı ile bir bilgi tabanı arasındaki bir ana entegrasyonda göz önünde tutulan ilişkilerin birbirlerini nasıl etkilediğinin belirlenmesi gerekmektedir. Şekil 3.5a'da görüldüğü gibi bir bilgi tabanı, bir veri tabanından verilere gereksinim duymakta veya Şekil 3.5b'de görüldüğü gibi, bir veri tabanı periyodik olarak bir bilgi tabanına veri

göndermektedir (Payne ve McArthur, 1990). Eğer bir bilgi tabanı, verilere ihtiyaç duyuyorsa, bilgi tabanının verilere ne zaman ihtiyaç duyacağının belirlenmesi ve eğer veri tabanı bir bilgi tabanına veri gönderecekse, bu işlemin formatının belirlenmesi gerekmektedir.



Şekil 3.5 Bir veri tabanı ile bilgi tabanının etkileşimi için alternatifler

3.8. Bir Uzman Sistemde Bilgilerin Organizasyonu ve Yönetimi

Uzman sistemler, bazı özel problem alanlarında bir uzman insan ile karşılaştırma yaparak, bir performans düzeyine erişmeyi sağlayacak bilgisayar tabanlı problem çözme sistemleridir. Uzman sistemler, oluşturuldukları yapı ve gelişme prosesleri bakımından klasik programlardan farklılık göstermektedir. Klasik programlarda bilgisayara ait karar komutları üzerinde durulmuştur. Uzman sistemlerin geliştirilmesindeki temel problemlerden bir tanesi de, uzmanların sahip olduğu ve kullandığı bilginin nasıl temsil

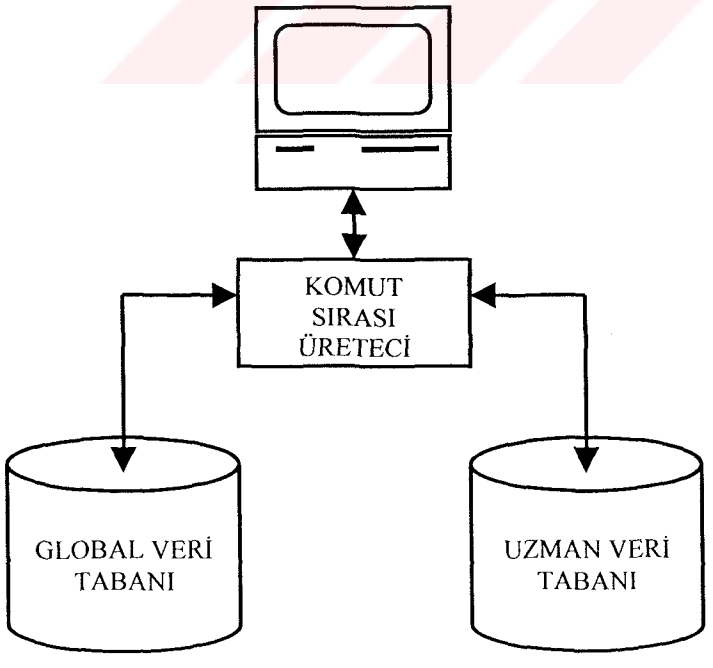
edilip kullanılacağıdır. Tipik bir uzman sistemde bilgiler, aşağıda gösterildiği gibi üç kategoride organize edilebilir :

(1) Belirli bir problemi tanımlayarak, girdi verilerini ve mevcut çözüm statülerini veya durumlarını izleyen global bir veri tabanı.

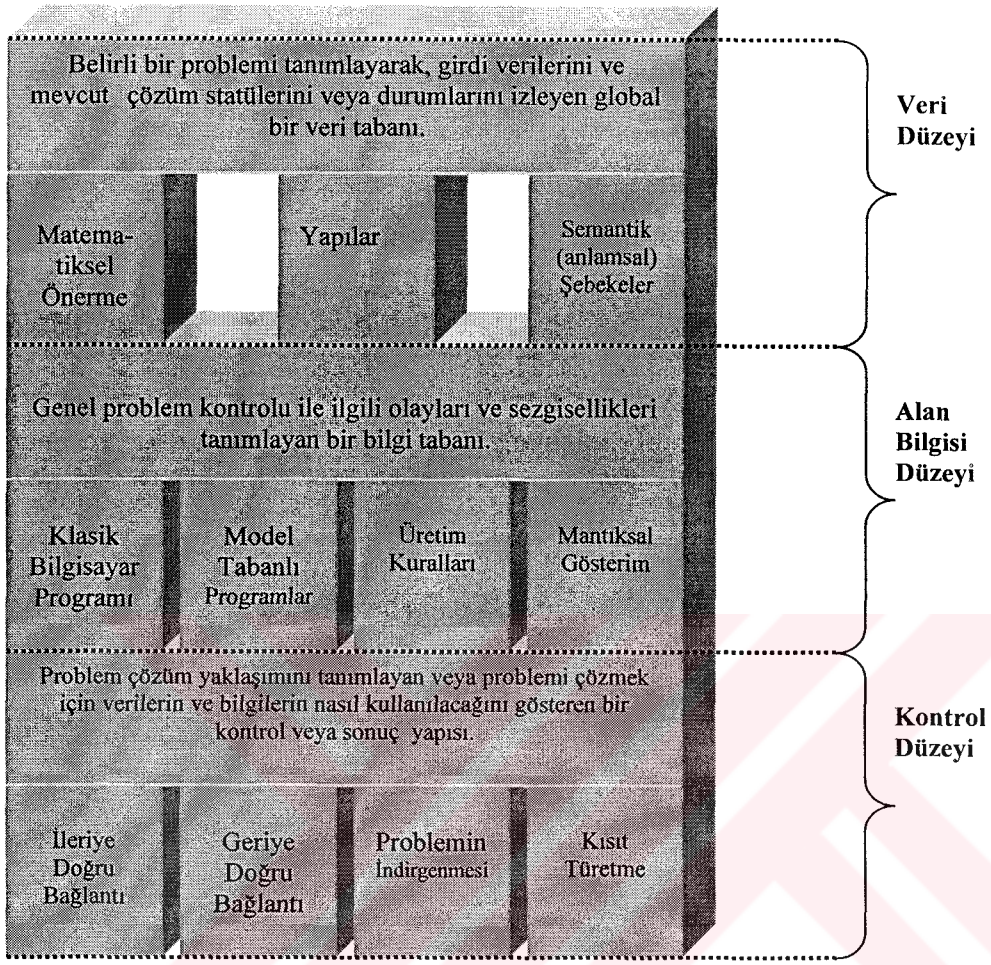
(2) Genel problem kontrolu ile ilgili olayları ve sezgisellikleri tanımlayan bir bilgi tabanı. Bu kontrol bilgisi, kurallar formu şeklinde olduğundan, bazen prosedüre dayalı veya ilgili bilgi olarak tanımlanır.

(3) Problem çözüm yaklaşımını tanımlayan veya problemi çözmek için verilerin ve bilgilerin nasıl kullanılacağını gösteren bir kontrol veya sonuç yapısı.

Bu nedenle uzman sistemler tipik olarak, veri ve dil yapılarının oluşturulmasını sağlamaktadır. Bu yapılar, açıklama bilgisinin (problem verisi) kontrol edilebilir spesifik problem çözme bilgisinin (ilişkiler) ve Şekil 3.6'da gösterilen kullanışlı arabirimlerle bu bilginin işletilmesi için kontrol yapılarının (sonuç veya prosedüre yönelik) saklanması ve



Şekil 3.6 Uzman sistemlerde kullanışlı arabirimler (Shannon vd., 1985)



Şekil 3.7 Uzman sistemlerde bilginin organizasyonu ve bilgi düzeyleri

temsil edilmesi için kullanılmaktadır. Uzman sistemlerin birçok Şekil 3.7'de görüldüğü gibi, bilgiyi üç düzeyde organize etmektedir. Bunlar veri tabanı, bilgi tabanı ve kontrol yapısıdır. Bir uzman sistemin oluşturulmasındaki temel konulardan birisi de, bu verinin ifade edilmesi ve saklanması problemidir. Aşağıdaki incelemelerde, tanımlanan metodolojilerin birçoknun tüm bu üç alanda uygulanabileceği ve uygulandığı unutulmamalıdır.

3.8.1. Veri düzeyi (Global Veri Tabanı)

Global veri tabanındaki girdi, çözülmekte olan belirli bir problemle ve mevcut olay durumlarıyla ilgili veri veya olaylar formundaki açıklayıcı bir bilgidir. Açıklayıcı bilgiyi sunmak için birçok alternatif yol vardır. Burada uygun sunuş formunun seçilmesi, uygulamaya bağlı olmaktadır. Temel ilgi odakları etkin saklama, yeniden kullanma ve modifikasyonun kolaylığı konuları üzerinedir. Halen kullanılan metotlar aşağıda açıklanmıştır:

1. Matematik Önerme (Mantık) : Önerme matematiği, çok çeşitli deyimlerin açıklandığı geçerli bir dildir.Burada basit açıklamalar, " uygun formlu formüller " olarak adlandırılmıştır.

Örneğin " Ml makinası boştur "

MCH-IDLE (M1) [MAKİNA-BOŞ (M1)]
veya
STATUS (Ml, IDLE) [DURUM (M1, BOŞ)]

olarak ifade edilebilir. Matematik önerme matematikten ziyade, geçerli bir mantıkla ilgilidir. Bu matematikle problemler sembolik olarak ifade edilebildiğinden bu yapay zekâda oldukça kullanışlıdır. Önerme matematiğinin gerçek gücü şöyle açıklanabilir: Normal arzulanan lisandaki (türkçe,ingilizce,vb.) cümleler, bir bilgisayar ile işlenerek ve diğer bilgilerle karşılaştırılarak geçerli bir forma dönüştürülür.

2. Yapılar : Yapılar, belirli bir eleman veya olay hakkındaki tüm bilgilerin, beraberce saklandığı veri yapılarıdır. Bir yapı, bir nesne veya bir kavram hakkındaki olaylar ve verilerin toplamıdır. Bu yapılar, bazen verilen bir elemana bağlı olan ispatlar grubu veya kümesidir. Birçok yapı tabanlı bilgi gösterim planları, farklı eleman tipleri için, farklı yapı tiplerine sahip olma düşüncesini kapsamaktadır. Bu yapıdaki alanlar veya kanallar, yapı tipleriyle ilgili bilgileri içermektedir. Örneğin üretilecek bir parçanın yapısı, parçanın

numarası, kullanılacak malzeme, parça şekline göre sınıflandırma, tezgâh rotası, değişik tezgâhlardaki işlem zamanları gibi kanallardan oluşmuş bir veri yapısı olabilir.

3. Semantik (Anlamsal) Şebekeler : Açıklayıcı bilgiler grafiksel olarak ifade edilebilirler. Anlamsal ağlar (şebekeler) yapılara benzemektedirler, ancak bilgi, tanımlanan eleman etrafında toplanmıştır ve buradaki elemanlar bir grafikteki düğümler ile ifade edilirler. Aynı bilgi, önerme matematiği şeklinde ifade edilebilse de, bazı tip ilişkiler en iyi bir şekilde grafiksel olarak açıklanabilirler. Örneğin bilgisayar programlarının, kod satırları olarak yazıldıklarında anlaşılmaları yüksek seviyeli bir dilde bile çok zor veya olanaksızdır; bunlar ancak bir akış diyagramı şeklinde ifade edildiklerinde kolayca anlaşılabilirler. Bunun gibi aynı ilişkileri açıklamak için önerme matematiği kullanılsa bile, uzun bir mantıksal deyim dizileri birbirlerini takip ettikleri için farklı elemanlar arasındaki ilişkiler ve etkileşimler pek açık değildir.

3.8.2. Alan bilgisi düzeyi

Bu bilgi düzeyi, belirli tipteki bir problemin, sorun veya alan bilgisi (yani imalat sistemi, dağıtım sistemi, bilgisayar şebekeleri, vb.) özelliğini tanımlamaktadır. Böylece sistem çözüme hazırlanır. Bu bilgi genelde prosedüre yöneliktir ve bir problemin verilerinin, problemin çözümü için nasıl kullanılacağını açıklar. Bir uzman sistemin gücü, problem alanındaki spesifik bilgiye dayanmaktadır. Birçok uzman sistem, en iyi uzmanların sahip olduğu bilginin araştırılması ve daha sonra da bu bilginin kural ağı oluşturulmak üzere birleştirilmesi ile elde edilen yüzlerce kuralı içermektedir. Veri düzeyi bölümünde belirlenen tekniklerin tümü, alan bilgisinin ifade edilmesinde kullanılabilmektedir. Ancak bu bilgilerin ifade edilmesinde birçok yöntem vardır. Bunlar ;

1. Klasik Bilgisayar Programı : Çözüm prosedürü iyi anlaşılırsa ve bir algoritma olarak programlanırsa, çözüme ulaşmak için verileri işletmek üzere klasik bir bilgisayar programı yazılır. Bilindiği gibi birçok simülasyon modeli, bu şekildeki problem veya alan spesifik bilgisini temsil etmektedir.

2. Model Tabanlı Programlar : Burada alana bağımlı bilgi, matematiksel semboller ile veya model tabanlı programlar şeklinde yazılır. Bu programlar, belirli şartlar verilerde bulunduğunda, kontrol yapısı tarafından aktif hale getirilirler. Bunların bazıları, durum araklı araştırma programları şeklindedir. Bu programlar mevcut durumları diğer durumlara dönüştürmek üzere matematiksel işlemler uygulayarak, bazı son durumların veya amaç durumlarının, ilk durumdaki şeklini bulmayı hedeflemektedirler.

3. Üretim Kuralları : Belirli ilgi alanlarındaki model tabanlı programların bir tipi de üretim kurallarıdır. Bunlar,

IF < KOŞUL > **THEN** < İLK HAREKET >

formundaki programlardır. Şart genellikle mevcut durum hakkındaki özellikleri test eden önermelerin bir bileşimidir ve ilk hareket mevcut durumu değiştirmektedir.

4. Mantıksal Gösterim : Prosedüre dayalı bilgi ilk sıradaki önerme mantığı şeklinde ifade edilebilir. Örneğin, A : B1 , B2 şeklindeki bir önerme şöyle ifade edilebilir. B_1 ve B_2 doğruysa, A doğrudur veya B1 ve B2 nin işletilmesiyle A'yı sağlayan bir durumun üretilmesi için, bir prosedür söz konusudur. Bazen bu bilgi AND / OR (VE / VEYA) graf formunda saklanabilir. Daha sonra, bir ağaç araştırması yaparak, çözüm veren önermeler kümesi bulunduğunda bir çözüm elde edilir. Önerme mantığının en önemli özelliklerinden birisi de, bunun geniş bir alan üzerindeki kuralların niteliklerine olanak tanımasıdır (Shannon vd., 1985)

3.8.3. Kontrol düzeyi

Kontrol düzeyinde bilgisayar programı, cevaplandırılacak sorunun ne olacağı hakkında ve daha sonra mevcut problemi çözmek için veri tabanı ve bilgi tabanının ne şekilde kullanılacağına dair kontrol stratejisi hakkında karar verir. Kontrol stratejileri sabit veya deneysel olarak sınıflandırılabilir. Sabit bir kontrol stratejisinde, bir uygulama kuralı seçilir ve daha sonraki kabuller için herhangi bir hazırlık yapmadan uygulanır. Deneysel kontrol stratejilerinde ise bir kural seçilir ve bu kural uygulanır. Ancak tatmin edici bir çözüm

bulunmadığı taktirde (çözüm yoksa), farklı bir kuralın uygulanması için, bu hesaplama noktasına daha sonra dönmek üzere bir çalışma yapılır.

Bir kontrol komut üreteci, problemin çözülmesinde göz önüne alınan adımları organize ve kontrol eder. Bu üreteç, gerekli bilgi ve yazılım modüllerini alarak uygun bir sıra altında yazılım modüllerini işletir, çıktıyı analiz eder ve uygun bir formda cevaplar verir. Komut üreteci bazen kullanıcıya, nedenleri satır açıklamaları şeklinde ve sonuçların elde edilmesinde kullanılan verileri geri besler. Buradaki adımlar veya kurallar, sistem durumları IF - THEN (EĞER - İSE) kurallarının veya probabilistik dalların birbirlerine bağlanması gibi modellerin kullanılması ile gerçekleştirilir. Kontrol stratejisi, çözülecek problemin bir stratejisidir ve bu strateji için birçok yaklaşım kullanılmaktadır. Bu yaklaşımlar şu şekilde açıklanabilirler :

l. İleriye Doğru Bağlantı : Eğer çözüm ilk veri ve şartlar (mevcut global veri tabanı) kümesinden başlatılırsa ve bazı amaç veya sonuçlara doğru hareket ederse bu çözüm, model, veri, olay veya öncelikli olarak adlandırılır. Eğer model zamana bağlı değil ise, dal ve sınır, tepeye tırmanma veya ağaç araştırma teknikleri gibi durum aralıklı araştırma metotları bu durumda kullanılabilir.

Bu teknik, bir kurala ait koşul cümlesiyle başlayan ve faaliyet kurallarını harekete geçiren ve ileri doğru kurallar zinciriyle çalışan işlerde kullanılır. İleriye doğru zincirleme sırasında çıkarım mekanizması, IF (EĞER) koşullarının bilgi tabanındaki diğer kurallarla uyumunu araştırır. Bu işlemlerle yeni değerler oluşturulduktan sonra, eylem kuralları uygulanır. İleriye doğru bağlantı tekniği, başlangıç bilgilerinden, ileriye doğru araştırma yaparak ilerler. Kullanıcılar, bu teknikte istedikleri düzende istedikleri kadar veri sağlarlar. Ancak, kurallar yeterli veri olduğu zaman çalışmaya başlar. Şöyle de açıklamak olanağı vardır. Yeni gelen verilerle kuralların bütün koşulları ya da diğer kuralların sonuçları eşlendiği zaman, sistem işlemeye başlar. Bu durumda, gerçekte bir problemin sonucunu içeren bir kuralın bulunması ilkesi kullanılmaktadır.

2. Geriye Doğru Bağlantı : Arzulanan sonuç ve amaç durumu önceden biliniyorsa, ancak sonucun yöntemi bilinmiyorsa, bu durumda geriye doğru çalışma arzulanır ve model,

amaca veya beklentiye yönelik olarak adlandırılır. Hesaplama yapısı nümerikse bu durumda, dinamik programlama metodu kullanılabilir.

Geriye doğru bağlantı sistem neyi ispatlamak istiyorsa, onunla başlayan bir çıkarım metodudur. Geriye doğru bağlantı, bir kuralı eşlendiği cümleleriyle bulmak için, kuralların geriye doğru arandığı işlerde kullanılır. Kurallar, ortaya konan IF (EĞER) koşullarını deneyerek ihtiyaç duyulan yeni değerleri bulmak için bir kuraldan diğerine doğru ilerler.

Geriye doğru bağlantı tekniğinde, bir problemin özel bir kısmı üzerinde çalışılır.Bu, bir probleme ya da alt probleme özgü bir çözümdür. Kuralların sonuçları, yalnız göz önüne alınan problemlere özgü çözüm, ya da kısmi çözüm olduğunda çalışır. Bir kural kullanılırsa, onun koşulları üzerinde odaklanılan sisteme özgü yeni bir çözüm olur.

3. Problemin İndirgenmesi : Bu teknikte çözümlenecek problem, birbirinden ayrı olarak çözülecek şekilde alt problemlere bölünür veya ayrılır.Alt problemlerin çözümlerinin birleştirilmesi bazı durumda genel problemin çözümünü sağlar. Daha küçük bir araştırma alanıyla ilgili problemleri açıklayan bu yaklaşım, bir amacın elde edilmesi için yapılacak çok sayıdaki ilişkisiz işlerdeki problemlere uygulanabilir. DENDRAL, problem indirgeme tekniğini kullanan ileriye doğru bağlantı modeline örnektir. MYCIN ise geriye doğru bağlantı ve problem indirgeme tekniklerini kullanmaktadır.

4. Kısıt Türetme : Bu problem çözüm tekniğinde mümkün çözüm kümesi, çözümün küçük elemanları, birbirine benzemesi gerektiğini gösteren "lokal (bölgesel) kısıtlar" oluşturan kurallar veya matematiksel işlemlerle, daha çok kısıtlanır. Kısıtlar, tatmin edilmesi gerekli ilişkiler (veya alt amaçlar) olarak gözlenilebilirler.

3.9. Uzman Sistemlerin Geliştirilmesi

Bir uzman sistemin geliştirilmesi, diğer tip bilgisayar uygulamalarından farklıdır. İlk fark olarak, özel bir donanım ve yazılımın kullanımının mümkün olması gösterilebilir. Diğer bir fark uzmanın önemidir. Aktif katılım ve istek duymayan bir uzman, geliştirme çabalarında hatalara neden olabilir. Bir diğer fark ise, prototip geliştirmenin önemidir. Başlangıçtaki

sisteme, genellikle küçük bir geliştirme uygulanır ve bir önceki versiyon tasfiye edilir ve genişletilir.

Bir uzman sistem geliştirilirken ilk olarak, eldeki problem tanımlanmalıdır. Örneğin, problem nedir? O nerede mevcuttur? İlk olarak bu başarılmış olmalıdır. Daha sonraki adım, problemin durumunun detaylandırılarak belgelenmesidir. Bu ise, hedefleri, amaçları, kısıtları ve problemle ilgili değişkenleri içermektedir. Eğer mümkün ise, problemin durumu matematiksel model formunun içerisine yerleştirilmiş olmalıdır. Diğer taraftan, problem elemanları şematik olarak kurulmalıdır.

Bir uzman sistemin geliştirilmesi için yapılan çalışmalar temel olarak 6 adımda sıralanabilinir. Bu 6 adım ve her adımdaki faaliyetler şu şekildedir (Watson ve Blackstone, 1989):

(1) Uygun bir problemin belirlenmesi,
(2) Bir prototipin geliştirilmesi,
(3) Bütün bir sistemin geliştirilmesi,
(4) Sistemin değerlendirilmesi,
(5) Sistemin bütünleştirilmesi,
(6) Sistemin güncelleştirilmesi.

Bu adımlar bu şekilde belirlendikten sonra, her adımda yapılması gereken faaliyetler de şöyle sıralabilir.

1. Uygun bir problemin belirlenmesi : Bir uzman sistemin geliştirilmesindeki en önemli adım, bir problemin seçilmesidir. Uygun olmayan uygulama sahaları için bir uzman sistem geliştirmek ya mümkün değildir veya olağan dışıdır. Bu adımdaki faaliyetler şunları içermektedir :

- Bir problem alanı ve spesifik bir işin belirlenmesi,
- Geliştirme işlemine katkıda bulunacak uzman veya uzmanların belirlenmesi,
- Probleme bir deneme yaklaşımının belirlenmesi,

- Çabaların maliyetinin ve kazandıracaklarının analiz edilmesi,
- Spesifik bir geliştirme planının hazırlanması.

2. Bir prototip sistemin geliştirilmesi : Bilgi mühendisi, bir uzman ile bir uygulama geliştirmeye çalışır. Tipik olarak bir uzman sistem, birçok versiyonu geliştirildikten sonra eksiksiz olarak tamamlanır. Bir prototip geliştirmek için yaygın olarak aşağıdaki aktiviteler yerine getirilir :

- Uygulama alanı hakkında bilgilenme,
- Performans kriterlerinin ayrı ayrı belirlenmesi,
- Uzman sistem kurma araçlarının seçilmesi,
- Bir başlangıç versiyonunun geliştirilmesi,
- Bir olayın etüt edilmesiyle, bu versiyonun test edilmesi,
- Uzman sistemin yeniden gözden geçirilmesi ve yeniden test edilmesi,
- Tamamlanmış bir uzman sistem için detaylandırılmış bir tasarımın geliştirilmesi.

3. Bütün bir sistemin geliştirilmesi : Bir bilgi mühendisi veya uzman, bir prototip sistem geliştirdikten sonra, her yönüyle tatmin edilmiş, tamamlanmış bir sistemi geliştirmeye hazırdır. Burada gerekli olan aktiviteler şunlardır :

- Bütün bir sistemin temel yapısının oluşturulması,
- Bilgi tabanının genişletilmesi,
- Bir kullanıcı arayüzünün uyarlanması,
- Sistem performansının izlenmesi.

4. Sistemin değerlendirilmesi : Bir uzman sistemin kullanıma konmadan önce, dikkatli bir şekilde değerlendirilmesi gerekmektedir. Bu adımdaki faaliyetler şunları içermektedir :

- Uzman sistemin performans kriterlerine bağlı olarak karar verme ve yeteneklerinin test edilmesi,
- Arabirimlerinin değerlendirilmesi.

5. Sistemin bütünleştirilmesi : Bir uzman sistemin, bir çalışma ortamına entegrasyonu gerekmektedir. Bunun için de şu faaliyetlerin yerine getirilmesi gerekmektedir :

- Bunun teknoloji transferi için düzenlenmesi,
- Sistemin diğer veri tabanları, aletler veya diğer donanımlarla iletişim kurabilmesi,
- Sistemin kabiliyetlerinin kolay ve hızlı bir şekilde arttırılması.

6. Sistemin güncelleştirilmesi : Uzman sistemlerin çoğunun zaman boyunca değiştirilmesi gerekmektedir. Bunun nedeni ise, şartların değişmesidir. Durumların, diğer farklı veya yeni durumları ve sezgisel değişiklikler ile şartları dikkate alması gerekmektedir. Uzman sistemlerin kullanımına destek için önemli şart, sistemin güncelleştirilmesinin sağlanmasıdır.

4.YAPAY ZEKÂ ile BÜTÜNLEŞİK SİMÜLASYON ORTAMLARI

Simülasyon tekniği birçok problemin çözümünde ve analizinde oldukça başarılı bir şekilde kullanılmaktadır. Simülasyon ve yapay zekâ teniklerinin esas amacı gerçek dünya olaylarının modellerini geliştirmektir.

Yapay zekâ ve simülasyon tekniklerinin temeldeki benzerlikleri ve iki tekniğin de birbirlerine olan gereksinimleri, bu iki yöntemi bütünleşik bir hale getirmiş ve getirmeye devam etmektedir. YZ ve simülasyon tekniklerinin birleştirilme fikri 1970'lerin sonlarına doğru ortaya atılmıştır (Ören, 1977). Bu konuda zamanımıza kadar oldukça ilginç çalışmalar yapılmıştır. Aşağıda bu konuda yapılmış olan ve literatürde önemli bir yer tutan çalışmalar kısaca özetlenmiştir.

Heidorn (1974) tarafından geliştirilen NLPQ (Natural Language Programming for Queueing Simulations) paketi, sistemin İngilizce dilinde yazılarak tanımlanmasını isteyerek, GPSS dilinde simülasyon programı üretir.

Shannon, Mayer ve Adelsberger'in (1985), uzman sistemler ve simülasyon ilişkisini ayrıntılı olarak inceledikleri çalışmalarında, bu tür uzman sistem paketlerinin oluşturulmasında etkin olarak kullanılabilecek PROLOG ve LISP gibi dillerin analizi yapılmaktadır.

O'Keefe ve Roach (1987) tarafından yapılan çalışmada, PROSS (The Prolog Simulation System) adı verilen paket ile geleneksel kesikli simülasyon ve yapay zekâ entegrasyonu sağlanmıştır.

Mellichamp ve Wahab (1987) tarafından geliştirilen paket, EİS'lerinin planlanmasında LISP ile SIMAN simülasyon programını üretmektedir. Bu paket aynı zamanda sistemin teknik ve mali analizlerini de yapmaktadır.

Paul ve Chew (1987) tarafından geliştirilen simülasyon programı üreteci, PASCAL dilinde yazılmış olup simülasyon programını da PASCAL dilinde üretmektedir. Üretilen her bir kod, LIBSIM adı verilen program kütüphanesindeki rutinleri kullanmaktadır.

Brazier ve Shannon (1987) tarafından geliştirilen bir paket, TURBO PROLOG ile hazırlanmıştır ve fabrika ortamındaki taşıma araçlarından olan, otomatik kontrollü araçların çalışma sistemlerini incelemek üzere, SIMAN dilinde simülasyon programı üretmektedir.

Mellichamp ve Wahap (1987) tarafından geliştirilen PPS (Process Planning Simulations) paketi, EİS'nin simülasyon modelleri hakkında detaylı bilgi gerektirmeden, SIMAN dilinde program üretmektedir; FORTRAN dilinde yazılmıştır ve üç ayrı fonksiyondan oluşmaktadır:1)Etkileşimli sistem tanımı, 2)SIMAN, 3)Etkileşimli finansal planlama.

Etkileşimli diyalog paketlerine verilecek bir örnek, Haddock (1987) tarafından BASIC dilinde yazılmış pakettir. SIMAN dilinde simülasyon programı üretmekte olan bu paket, EİS'nin modelenmesine yöneliktir. Daha sonra, Haddock ve O'Keefe bu program üretecindeki kısıtlamaları ortadan kaldırabilmek ve bazı uzmanlık isteyen analizleri yapabilmek için, çıkarım mekanizması geliştirmişlerdir. Çıkarım mekanizması, simülasyon sonuçlarının istatistiksel olarak güvenilir olduğu koşulları belirlemek için gerekli prosedürleri içeren bir kural tabanını kullanmaktadır.

Kim, Funk ve Fichter (1988) çalışmalarında, EİS'de çizelgeleme konusunda bir uzman sistem geliştirmişlerdir. Sistem, IBM PC/AT'de SMALLTALK/V'de geliştirilmiş kural tabanlı bir programdır.

Kusiak ve Heragu (1988) tarafından yapılan çalışmada, KBML adı verilen bir uzman sistem geliştirilmiştir. Sistem, makina yerleşim problemlerini çözmeye yöneliktir. Etkileşimli diyalog yardımıyla kullanıcıdan gerekli bilgileri almaktadır. LISP'te hazırlanmıştır; bilgi tabanı kurallardan oluşmaktadır.

Roger ve Willams (1988) çalışmalarında, otomatik işleyen üretim hücrelerinin kontrol simülasyonu için bilgi tabanlı sistemlerin kullanımından bahsetmektedirler. Bu sistem, gerçek zamanlı kontrol için, simülasyon modellerinin yeniden kullanımına izin verecek MUSE yapay zekâ araçlarına ilave olarak geliştirilmiştir. Bu sisteme KBSC (Knowledge-based simulation and control) aracı adı verilmiştir.

Schroer ve Tseng (1989) tarafından geliştirilen AMPS (Automated Manufacturing Programming System) paketi etkileşimli diyalogla üretim sistemlerinin modellenmesine yardımcı olmaktadır. Bu paket LISP'de yazılmıştır ve simülasyon kodlarını GPSS/PC dilinde otomatik olarak oluşturmaktadır.

Mellichamp ve Park (1989) tarafından veri toplamadan çıktı analizine kadar her aşamada istatistiksel analzileri yapabilecek SESSA isimli bir uzman sistem geliştirilmiştir. Bu, istatistiksel analiz yapan ilk uzman sistemdir. OPS83'de yazılmıştır.

Shearn (1990), çalışmasında PASSIM (Pascal Discrete Event Simulation Program Generator) adı verilen genel amaçlı bir simülasyon üretecinden bahsetmektedir. Üreteç, koşulları öğrenme ve pratik olarak problem çözmede oldukça etkindir.

Manivannan ve Pegden (1990) tarafından geliştirilen JITSAI isimli paket, tam zamanında üretim sistemlerine ilişkin kural tabanlı bir simülasyon paketidir. LISP dilinde oluşturulmuştur ve parça akış problemleri, makina hazırlama ve bozulma durumları, ara stok seviyelerinin belirlenmesi, kalite kontrol problemleri gibi üretimle ilgili bilgileri modelleme yeteneğine sahiptir.

Raczynski (1990) tarafından yazılan QMG (Queueing Model Generator) paketi, PASCAL dilinde programlanmış olup grafiksel etkileşim özelliğine sahiptir. Paket, GPSS veya XCELL dilinde bekleme hatlarının simülasyon programını üretir.

Ülgen ve Thomasma (1990), nesneye yönelik program geliştirme amacı olarak SMALLTALK-80'i bir simülasyon üreteci olarak kullanmışlar ve buna SMARTSIM adını vermişlerdir. Kullanıcı, grafik etkileşimli olarak, paketin sunduğu standart şekillerden

yararlanarak üretim sistemini oluşturmakta, simülasyon kodlarını üretmekte ve animasyon yapabilmektedir.

Raczynski ve Stanislaw (1990) tarafından geliştirilen QMG (The Queueing Model Generator) üreteci, model yapısını tanımlamak için kullanıcıyla etkileşim halindeki bir blok diyagram editörünü kullanır. Üretici programı, PASION dilinde kodlanmakta ve PASCAL dilinde program üretmektedir. Bu paket özellikle EİS'lerin tasarımına yönelik olarak geliştirilmiştir.

Mellichamp ve Venkatachalam (1990) tarafından geliştirilen uzman sistem, simülasyon analistine, simülasyon sırasındaki hataları ortaya çıkarmada yardımcı olmaktadır. INDEX (Interactive Debugging Expert) adı verilen bu uzman sistem, çeşitli simülasyon dilleri ile yazılan programlarda sıklıkla yapılan mantıksal hataların nedenlerini ortaya koymaya yöneliktir.

Lu ve Tcheng (1991) tarafından geliştirilen ve AIMS (Adaptive and Interactive Modelling System) adı verilen paket simülasyona, makina öğrenmesine, çok amaçlı model formülasyonuna ve model kullanımına olanak sağlamaktadır. Bu paket değişik safhalardaki mühendislik tasarımlarını desteklemek için simülasyonun ve optimizasyonun nasıl entegre edileceğini göstermektedir.

Sztrimbely ve Weymounth (1991) çalışmalarında kural tabanlı bir yapıya sahip, bilgi mühendisliği uzman sistem metodolojisine dayanan, ADSI adlı çizelgeleme programını geliştirmişlerdir. Bu uzman sistem, özellikle büyük metal endüstrileri içindir. Kullanıcı, mouse yardımıyla ekran üzerinde ekipmanların yerleşimini yaparak sistemin simülasyonunu animasyonlu olarak gözleyebilmektedir. Simülasyon programı kısmı, nesneye yönelik programlama tekniklerinden faydalanmak için C++ dilinde yazılmıştır.

Ayağ ve diğerleri (1991) tarafından gerçekleştirilen bir çalışmada, üretim sistemlerinin simülasyonu ve yapay zekâ/uzman sistemlerin genel yapıları incelenerek, üretim sistemlerinde simülasyon tekniği uygulamaları açısından yapay zekâya düşecek görevler ortaya konulmuştur.

Akhun ve diğerleri (1992) tarafından yapılan bir çalışmada, grup teknolojisi imalat sistemlerinin simülasyonu için BASIC dilinde bir simülasyon üreteci geliştirilmiştir. Üreteç SIMAN dilinde simülasyon programı üretmektedir.

Dinçmen ve Araz (1992) çalışmalarında, atölye tipi üretim için geliştirdikleri bir simülasyon üretecini açıklamaktadırlar. PROLOG dilinde yazılan ve etkileşimli bir diyalog arayüzü içeren paket, modeli kurulacak atölye ortamının tanımlanmasında kullanıcıya destek sağlamaktadır.

Bolte ve diğerleri (1993) yaptıkları çalışmalarında, yüksek seviyeli nesneye yönelik bir simülasyon ortamı geliştirmişlerdir. Nesneye yönelik programlama yaklaşımıyla, yapay zekâ kabiliyetleri ve simülasyon tekniğini birlikte kullanarak, zeki simülasyon nesneleri oluşturmuşlardır.

Zuylen (1994) çalışmasında, istatistiksel analiz, simülasyon ve fiziksel sistemlerin analizi vb. sayısal programların (yazılımların), geliştirilmeleriyle ilgisi olmayan kullanıcılar için kullanım zorluğuna değinmiştir. Zuylen, bir kullanıcı arayüzü de içeren bu programların kullanımı için bilgi gerektiğini belirtip bu tür yazılımların yeni kullanıcılar tarafından anlama ve kullanım zorluğunun, kullanıcıya destek sağlayan bir bilgi tabanıyla çözülebileceğine değinmiştir.

Javor (1995) tarafından yapılan bir çalışmada, simülasyon ve modellemenin etkinliğini arttırmak için Petri net ve yapay zekâ tekniklerinden oluşan bir yaklaşım önerilmiştir.

Kaiser ve Beaumariage (1997) çalışmalarında, simülatörler ve simülasyon dilleri gibi klasik simülasyon sistemlerinde, zeki simülasyon çıktı analizi desteği ve zeki modelleme özelliklerinin ya çok az veya hiç olmadığından söz etmektedirler. Ayrıca yapay zekâ, nesneye yönelik programlama ve simülasyon tekniklerinin entegre edildiği kavramsal bir modeli açıklamaktadırlar.

Bu bölümde simülasyon ve yapay zekâ/uzman sistem tekniklerinin birleştirilme yaklaşımları üzerinde durulmaktadır.

4.1. Bilgisayarla Simülasyonda Klasik Yaklaşım

Simülasyon modellemesindeki klasik bir yaklaşım Şekil 4.1' de gösterilmektedir (Schroer ve Tseng, 1989). Klasik yaklaşıma göre, simülasyon yapan kişinin modellenecek sisteme veya probleme ait genel bir bilgisi olması gerekmektedir. Bu bilgiye göre simülasyonu yapan kişi, daha sonra sistemin modelini tanımlamalıdır. Sonuç ise, detaylı bir problem spesifikasyonudur.

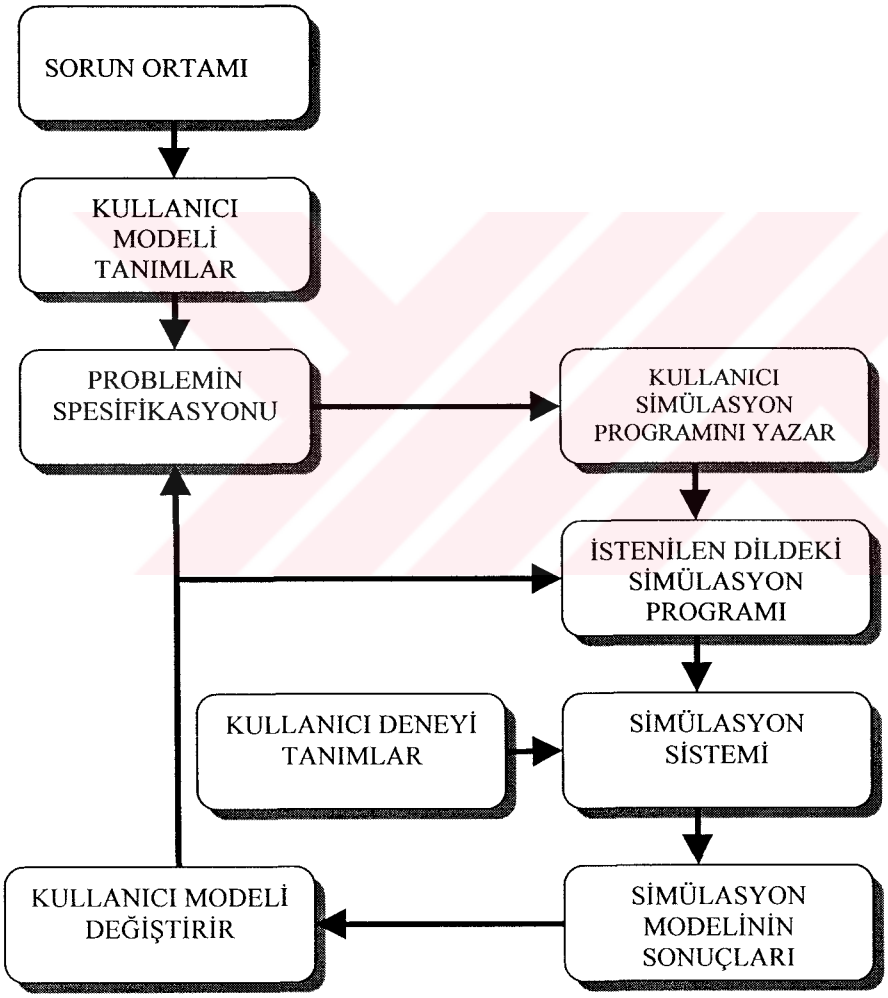
Simülasyon için bir sonraki adım, problem spesifikasyonuna göre, genel amaçlı bir programlama dili veya özel amaçlı bir simülasyon dili kullanılarak, modelin bir bilgisayar programının oluşturulmasıdır. Daha sonra, geliştirilen bu simülasyon programı, tüm yazılım hatalarının düzeltilmesi için incelenir. Hatalar düzeltildikten sonra, model doğru olarak kabul edilir ve onaylanır. Modelin doğruluğu, simülasyon modelinin bir simülasyon dili tarafından doğru olarak yazılması ile oluşur. Başka bir deyişle modelin doğruluğu, programın düzgün bir şekilde çalıştığını gösteren kontrol prosesidir. Modelin onaylanması ise, modelin davranışı ile gerçek sistemin davranışının karşılaştırılmasıyla ve gerekirse modelin, gerçek sistemi temsil etmek üzere, kabiliyetlerinin arttırılması için değiştirilmesi ve bunun sonucu olarak da, modelin gerçek sistemi yansıtmasıyla olur.

Deneyin yapılamasında simülasyoncu başlangıç şartlarını, duruş şartlarını, sabit durum (denge) ve örnek boyutunu belirlemelidir. Simülasyon genelde, sistemin boş ve serbest olduğu kabul edilerek başlatılır. Bu yüzden, sistem dengede olmadan önce, bir zaman periyodu boyunca çalıştırılmalıdır.

Buradaki problem, denge halindeki bir sistemi göz önüne alarak yapılan, modelcinin ihmal etmek istediği hatanın üstündeki noktayı bulmaktır. Conway Tekniği, genelde dengeyi belirlemek için kullanılır. Bu teknik, periyodik olarak ölçümleri elde ederek simülasyonu oluşturur. Her tekrardan sonra, elde edilen istatistiklerin herbiri, sistemin davranışını göstermek için bir zaman fonksiyonu olarak belirtilir. Conway Tekniği, bir ölçüm ihmal

edilen kümenin minimum veya maksimum değeri olana kadar tüm ölçümleri ihmal eder. Bu ihmal edilen ölçüm kümesi, toplanan verilerden çıkartılan standart ölçüm kümesi olarak kullanılır.

Bir simülasyon çalışmasını durdurmak için gerekli şartlar, başlangıç şartlarına benzerdir. Örneğin, üretim simülasyonlarının çoğunda, istenilen sayıda parça elde edildiğinde simülasyon durdurulur. Bu yaklaşımı kullanarak, simülasyon modeline giren gezer birimlerin sayısı sınırlandırılmaz. Bu yüzden çalışmanın sonunda gezer birimler hâlâ sistemde kalırlar ve kullanılan kaynaklar olurlar.



Şekil 4.1 Bilgisayarla simülasyonda klasik bir yaklaşım

4.2. Yapaz Zekâ ile Bütünleşik Simülasyon Sistemlerinin Tasarlanması

Simülasyon modeli kurmada klasik bir yaklaşım sistem hakkında detaylı bilgileri elde eden, simüle edilecek olan sistemi geliştiren ve elle bir program üreten bir simülasyoncuyu içermektedir. Buradaki yazılım, genel amaçlı bir programlama dili veya özel amaçlı bir simülasyon diliyle, sistemin bilgisayarda icra edilebilen bir modelidir. Bilgisayarla simülasyona klasik yaklaşımda, bir simülasyoncu tarafından yerine getirilen, elde etme ve programlama işlerinin her ikisi de otomatik olabilmektedir.

Klasik simülasyon yaklaşımı ile yapay zekâ tekniklerini birleştiren mevcut araştırmalar, iki spesifik alanda toplanmışlardır. İlk alan, problem spesifikasyon prosesinin otomatikleştirilmesine yönelik olarak yapay zekânın kullanımıdır. İkinci ve daha zor olan saha ise, arzulanan simülasyon dilinde otomatik olarak programın oluşturulması için yapay zekânın kullanılmasıdır.

4.2.1. Problem spesifikasyonunun otomatik olarak elde edilmesi

Simülasyon modeli kurmanın, model ile ilgili spesifikasyonları elde etme kısmını otomatikleştirme yaklaşımları için üç alternatif yapı, Şekil 4.2'de şematik olarak görülmektedir (Schroer ve Tseng, 1989). Otomatik problem spesifikasyonu, kullanıcıya bir yapay zekâ desteği olarak göz önüne alınabilir.

Tam bir modelin kurulması için, örneğin bir kuyruk sistemini göz önüne aldığımızda, kuyruk modelinin yapısı, kuyruk disiplinleri, gelişlerin dağılımı ve her bir servis süresinin dağılımları ve parametreleri gibi ilave detaylara gereksinim duyulmaktadır. Bu bilgiler de, dış ortamdan bir kullanıcıdan otomatik olarak elde edilebilmektedir.

Burada, iki tür bilgi içeren bilgi tabanları, otomatik problem spesifikasyonunu desteklemek için kullanılmaktadır. Bu bilgi tabanındaki bilgiler, problem alanına yönelik bilgi ve simülasyon modelleme bilgileridir. Sonuçta her problem alanı, kendi bilgi tabanını gerektirir. Simülasyon modelleme bilgisi de, arzulanan dilde son simülasyon programının yazılması için gerekli bilgidir.

Bu bilgi tabanları daha sonra, sistem ile kullanıcı arasındaki karşılıklı diyaloğu yöneten bir kurallar kümesini tanımlamak için kullanılır. Sistem, kullanıcının spesifik sorulara verdiği cevaba bağlı olarak, değişik kuralları kullanarak farklı bir mantık işletebilmektedir. Karşılıklı diyaloğun tamamlanmasında sistem, problem veya modelin bir iç spesifikasyonunu tanımlayacaktır.

Simülasyon modelinin veya problem spesifikasyonunun detaylı bir yapıda tanımlanmasında kullanıcıya yardım etmek üzere üç yaklaşım kullanılabilmektedir. Şekil 4.2'de görülmekte olan bu üç yaklaşım şunlardır :

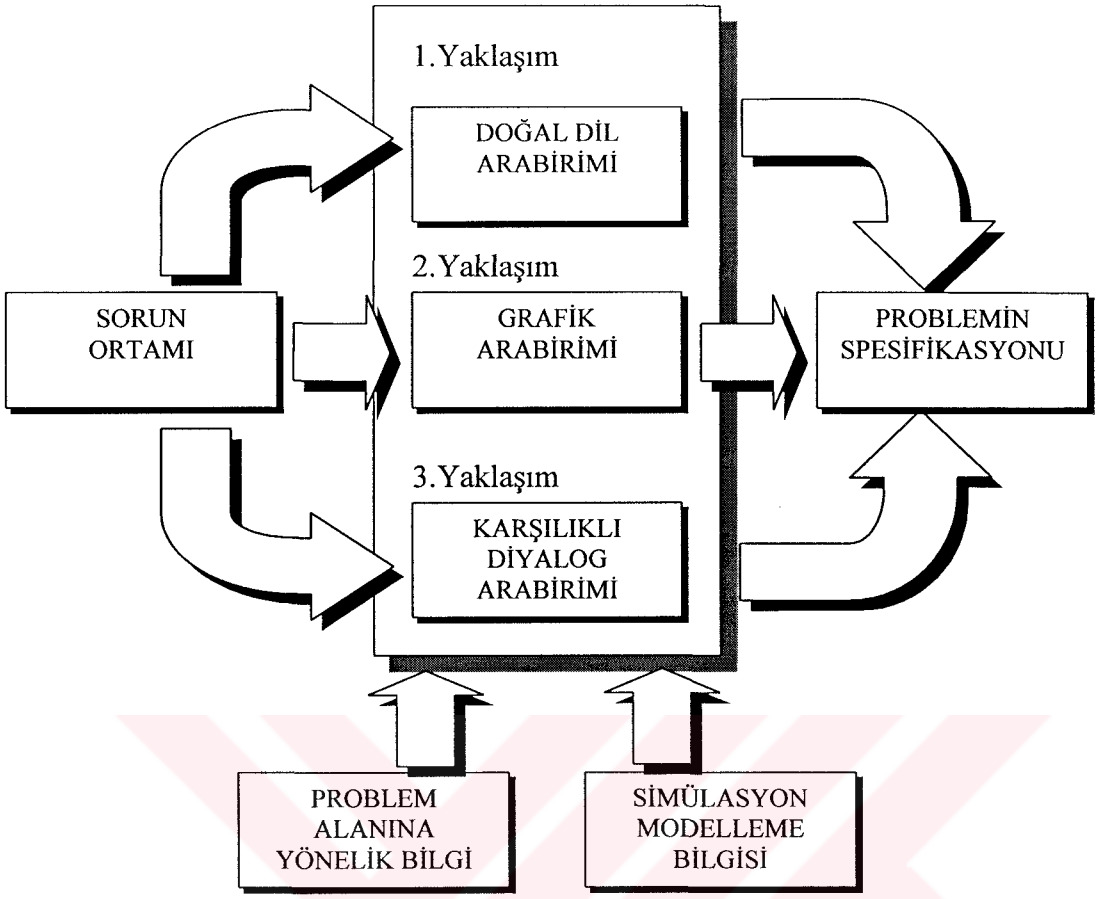
(1) Doğal dil arayüzü
(2) Etkileşimli grafik arayüzü
(3) Etkileşimli diyalog arayüzü

4.2.1.1. Doğal dil arayüzü

Bir doğal dil arayüzü, sorunun veya problemin bir kullanıcı tarafından, bilgisayara bir klavye yardımıyla serbest metin formatında tanımlanmasını sağlar. Girilen bu metin, bir doğal dil arayüzü tarafından önce dilbilgisi, daha sonra da içerik yönünden incelenir ve yorumlanır. Bu sistemlerin birçok, eksik bilgi ve olası tutarsızlıkları belirlemek için, kullanıcıyla etkileşimli olarak çalışırlar.

Doğal dil etkileşimli paketlere örnek olarak, Heidorn (1974) tarafından kuyruk modellerinin simülasyonu için geliştirilen doğal dil programlama sistemi NLPQ (Natural Language Programming for Queueing Simulations 1974) verilebilir. Bu sistemde sorun, İngilizce dilinde bilgisayara girilir ve daha sonra NLPQ, GPSS dilinde simülasyon programı oluşturur.

Diğer bir örnek ise, elektronik üretim simülasyon sistemi EMSS (Electronic Manufacturing Simulation System)dir. Ford ve Schroer (1987) tarafından geliştirilen bu sistemde, sorun yine İngilizce dilinde bilgisayara girilir ve EMSS de, SIMAN dilinde simülasyon programları oluşturur.



Şekil 4.2 Otomatik problem spesifikasyonuna ait üç yaklaşım

4.2.1.2. Etkileşimli grafik arayüzü

Problemin tanımlanmasında kullanıcıya yardımcı olan ikinci bir yaklaşım da etkileşimli grafik arayüzüdür. Grafiksel etkileşimde, simülasyonu yapılacak sistemin oluşturulması için, kullanıcı tarafından klavye veya bir mouse ile seçilebilen, şekillerden oluşan bir menü mevcuttur. Sistem oluşturulduktan sonra kullanıcı, klavye yardımı ile şekillere karşılık gelen özellikleri girer.

Grafiksel etkileşimli paketlere bir örnek olarak, Khoshnevi ile Chen (1986) tarafından LISP dilinde yazılmış olan paket gösterilebilir. Burada kullanıcı için modelin oluşturulmasında kullanılmak üzere bir şekil kütüphanesi mevcuttur. Bu şekiller, " CREATE ", " SEIZE ", " SERVICE ", " TRANSFER " ve " GATE " bloklarını içermektedir. Kullanıcı, bu şekillerden seçim ve ekranda uygun yerleştirme ile üretim

sistemini ve iş akışını tanımlar. Sistem, eksik veya fazla bilgileri kontrol ederek kullanıcıyı harekete geçirir. Modelin grafik şekli tamamlandıktan ve gerekli özellikler girildikten sonra, sistem otomatik olarak uygun SLAM simülasyon programını oluşturur.

4.2.1.3. Etkileşimli diyalog arayüzü

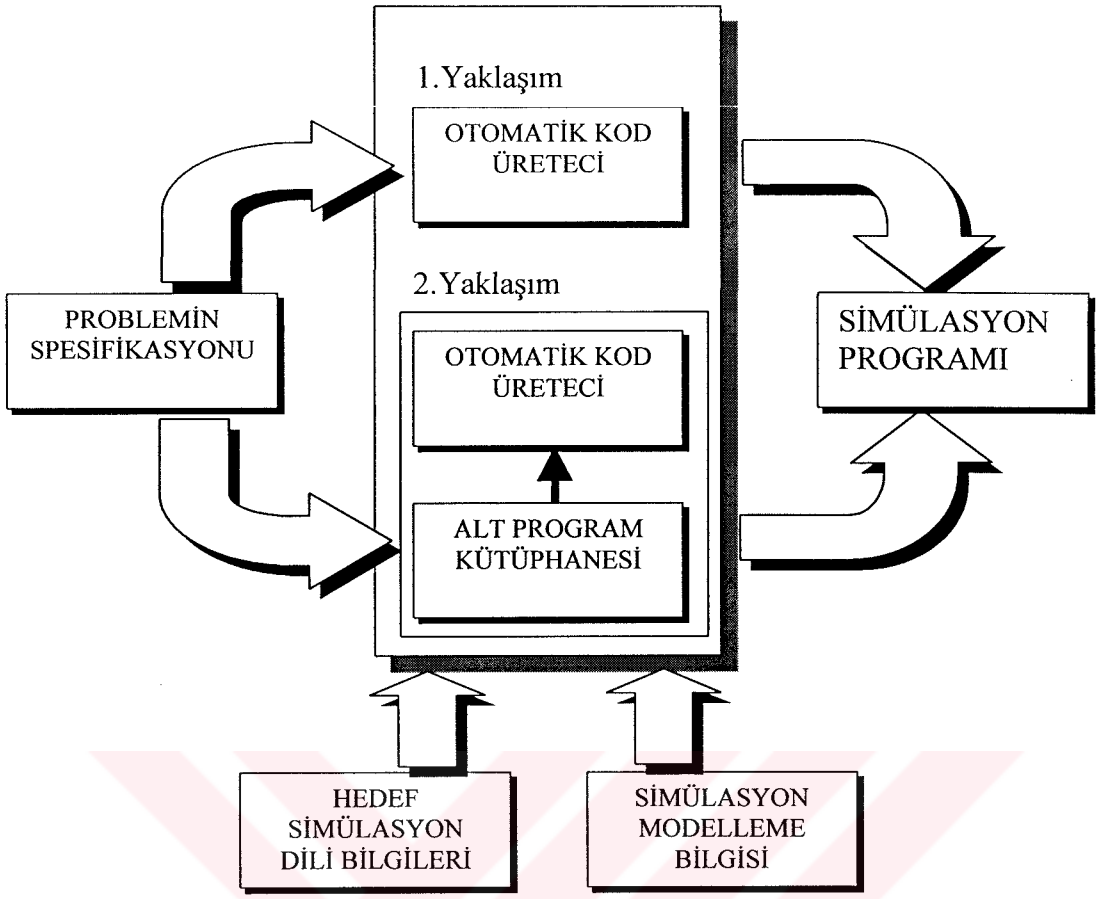
Problemin spesifikasyonunun tanımlanmasında kullanıcıya yardım eden üçüncü bir yaklaşım olarak da karşılıklı diyalog arayüzü mevcuttur. Burada sistem, kullanıcıya sorulan bir dizi sorulardan oluşmaktadır. Soruların sırası, kullanıcının cevaplarına bağlı olarak değişebilmektedir. Bir problem spesifikasyonunun tanımlanması için kullanılan bu üç yaklaşımdan geliştiriciler tarafından en çok kullanılanı, karşılıklı diyalog arayüzüdür.

Karşılıklı diyalog arayüzünü kullanan çok sayıda sistem geliştirilmiştir. Haddock ve Davis (1985), esnek üretim sistemi için bir simülasyon üreteci geliştirmişlerdir. Sistem burada, esnek imalat sisteminin karakteristiklerinin bir simülasyon modeline çevrilmesinde kullanıcıya yardım eden menülü bir ekran kümesinden oluşmaktadır. Bu giriş karakteristikleri, esnek imalat sistemindeki, istasyon sayısı ve tiplerini, envanter transfer araçlarının geliş hızlarını, hazırlık zamanlarını ve işleme zamanlarını içermektedir. Sistem, IBM PC'de BASIC ile yazılıp, SIMAN dilinde simülasyon programını oluşturmaktadır.

Brazier ve Shannon (1987), otomatik klavuzlu araç sistemlerinin modellenmesi için otomatik bir programlama sistemi geliştirmiştir. Sistem, kullanıcıya otomatik klavuzlu araçların tanımlanmasında yardımcı olmak üzere karşılıklı bir diyalog kullanır. Problem spesifikasyonu elde edildikten sonra sistem, uygun SIMAN programını yazar. Bu paket IBM PC'de Turbo PROLOG'da yazılmıştır.

4.2.2. Otomatik olarak simülasyon programını oluşturma yaklaşımları

Simülasyon ile yapay zekânın birleştirilmesine ait otomatik problem yaklaşımının şematik bir görünümü şekil 4.3'de verilmektedir (Schroer ve Tseng, 1989).



Şekil 4.3 Otomatik programlama sistemlerindeki yaklaşımlar

Temelde, iç problem spesifikasyonlarının alınması ve daha sonra arzulanan simülasyon dilinde programın yazılması için iki farklı yaklaşım uygulanmaktadır. İlk yaklaşım, problem spesifikasyonuna ait iç yapıya göre, direkt olarak bir otomatik kod üreteciyle simülasyon programını yazmaktır.

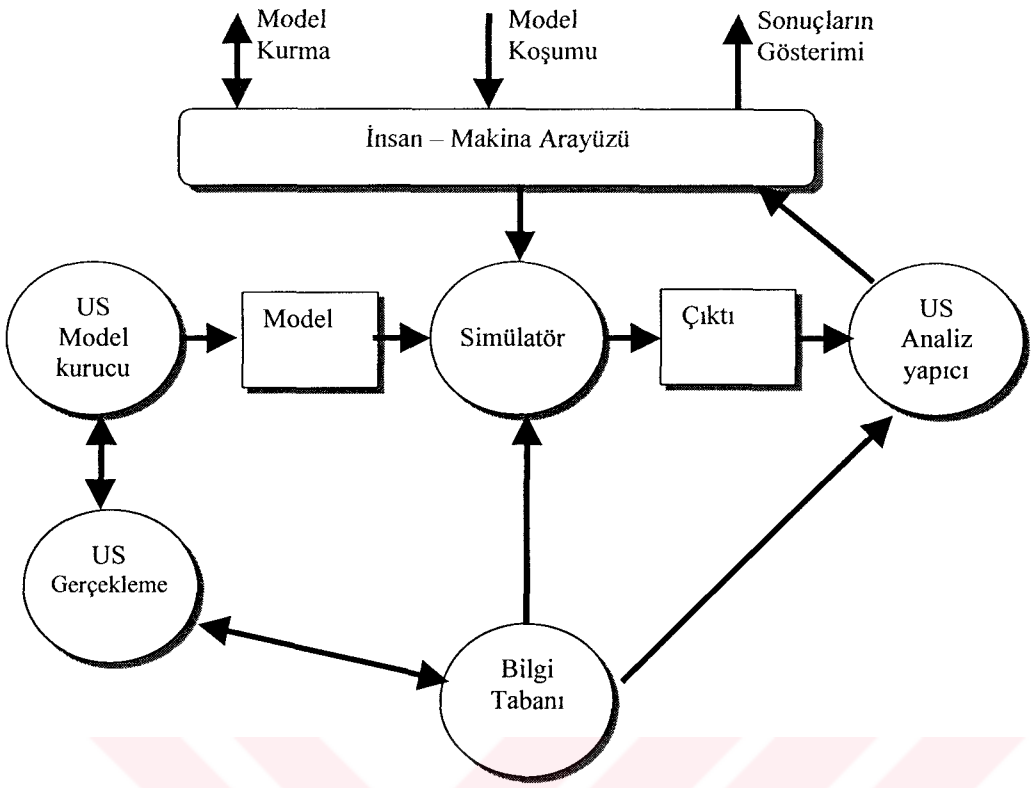
İkinci yaklaşım ise, simülasyon programının otomatik olarak yazılmasına yardımcı olan önceden tanımlanmış alt programlardan (makrolar) oluşan bir kütüphaneyi kullanarak, bu makroların uygun bir kombinasyonu ile simülasyon programını oluşturur. Bu şekildeki bir yaklaşımın avantajı, literatürde daha önceden incelenen problemler yerine, daha modernize problemleri çözebilmesidir. Dezavantajı ise, birçok alt programın belirli bir alana yönelik olmaları ve bu makroların bilgisayar belleğinde oldukça fazla yer kaplamalarıdır.

Otomatik simülasyon yazılımını desteklemek için iki bilgi türü kullanılmaktadır. Bu bilgi türleri, simülasyon modelleme bilgisi ve arzulanan simülasyon dili bilgisidir. Simülasyon modelleme bilgisi, Şekil 3.9'daki otomatik problem spesifikasyonu için kullanılan bilgi tabanıyla aynıdır. Arzulanan simülasyon dili bilgisi ise, simülasyon diline yönelik bir bilgidir. Bu bilgi tabanları daha sonra, iç problem spesifikasyonlarına göre, uygun simülasyon programını oluşturmak için bir kural kümesinin tanımlanmasında kullanılır.

4.2.3. Simülasyon programı üreteçlerinin yapısı

Simülasyon programı üreteçleri, kullanıcıdan alınan veriler ile, bilinen bir simülasyon dilinde simülasyon programını otomatik olarak oluşturan paket programlardır. Genelde iki aşamalı olan üreteçlerin birinci aşamasında, diğer bütün yapay zekâ paketlerinde de bulunan bir sorun algılama mekanizması vardır. Yani daha önce de incelediğimiz gibi, problem spesikasyonunun otomatik olarak elde edilmesi modülü vardır. İkinci aşamada ise bir simülasyon programı oluşturma modülü yer alır (Ayağ vd., 1991). Sonuç olarak, bu iki modülün birleştirilmesi ile ortaya çıkan yapı bir simülasyon üretecidir. Bir simülasyon üreteci, kullanıcıdan alınan veriler ile önce problemin spesifikasyonunu elde eder ve daha sonra da bu problem spesifikasyonunu kullanarak, bilinen bir simülasyon dilinde otomatik olarak bir simülasyon programını oluşturur. Sistem, simülasyon programının oluşturulacağı alana yönelik bilgileri, seçilen simülasyon dili bilgilerini ve genel simülasyon modelleme bilgilerini de içermektedir.

Zeki simülasyon programı üreteçleri, uzman sistem ve simülasyon program üreteçlerinin birlikte kullanılmasıyla ortaya çıkan melez bir yapıdır (Szymankiewicz vd., 1988). Zeki simülasyon program üreteçleri, bir insan uzmanın yapacağı bir işi çok daha kısa sürede gerçekleştirir. Sistem, kullanıcı ile etkileşimi çok daha kolaylaştıracak gerçek zamanlı etkileşimli bir arayüzü kullanmaktadır. Zeki simülasyon program üreteçlerinin genel yapısı Şekil 4.4'de görülmektedir (Szymankiewicz vd., 1988).



Şekil 4.4 Zeki simülasyon üretecinin genel yapısı

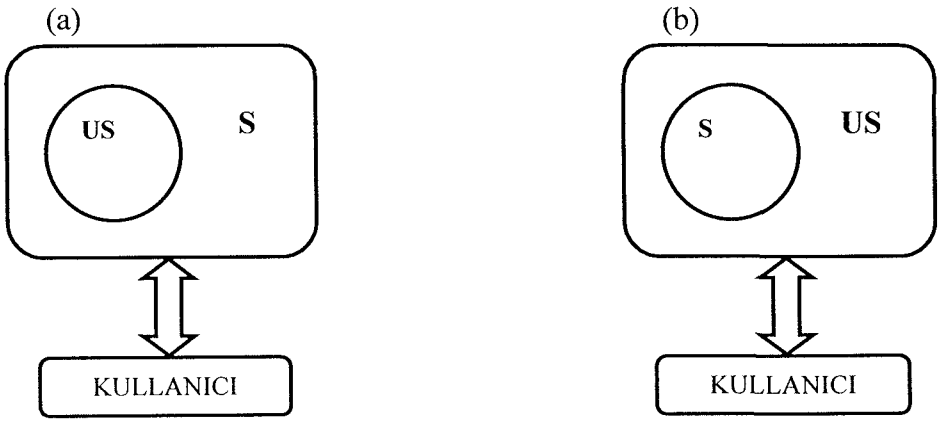
4.3. Simülasyon ve Uzman Sistemlerin Birleştirilmesi için Yaklaşımlar

Simülasyon ve uzman sistemler arasında kuvvetli bir metodolojik benzerlik vardır. Bu benzerlik kullanılabilir; ancak iki alan arasındaki karşılıklı verim artışı sadece bu benzerliğe bağlı değildir.

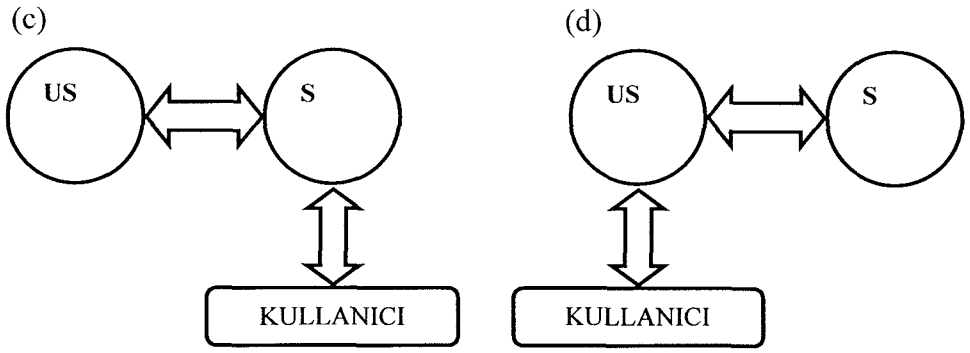
Şekil 4.5 ile 4.8 arasında, simülasyon ve uzman sistemlerin birleştirilmesine ait bir sınıflandırma gösterilmektedir (O'Keefe, 1986). Bu iki elemanın birleştirilmesindeki en uygun yol, Şekil 4.5a'da veya ters olarak Şekil 4.5b'de gösterildiği gibi bir simülasyon modeli içinde bir uzman sistemin yerleştirilmesi ile olmaktadır. Burada birçok simülasyon modeli, veri yerine bilgiyi kullanmaktadır. Örneğin kuyruk öncelik kuralı, bir bilgidir. Bu şekildeki bir kuralı program olarak yazmak yerine, bilgi tabanında saklamak daha uygundur. Bir uzman sistemin içinde simülasyon kullanılması için iki neden vardır. Bunlardan birincisi uzmanın , sistem kullanıcısı için bazı sonuçların elde edilmesinde,

simülasyonu işletmesidir.Buna örnek olarak, zeki bir öğretim sistemi olan, SOPHIE gösterilebilir. İkincisi ve daha önemlisi ise, uzman sistem tarafından bir veya birden fazla zamana bağlı değişkenlerin kullanılması ve böylece bunların değerlerinin güncelleştirilmesi için simülasyona ihtiyaç duyulmasıdır.

Ayrı bir yazılım olarak tasarlanan, geliştirilen ve uygulanan simülasyon ve uzman sistemler, paralel olarak birbirlerini etkilerler. Bir simülasyon modeli, bir uzman sisteme sorular sormaktadır. Bu yapı Şekil 4.6c'de görülmektedir. Bu, simülasyonun karmaşık bir sistem için geliştirildiği ve bir uzman sistemin, bu sistem içinde karar vermenin bir parçası şeklinde ortaya çıktığı durumlarda faydalı olmaktadır. Bu durumda simülasyon, karar kurallarını simüle etmek veya program halinde yazmaktan ziyade, bunları doğrudan elde edebilir. Simülasyonu işleten veya sonuçlarını kullanan uzman sistemler, Şekil 4.6d''de görülmektedir. Bu şekildeki uzman sistemler, bilgi mühendislerinin artan bir ilgisini çekmektedir. Kullanıcı üzerinden veya gerçek bir ortamdan olmak üzere, bir uzman sistemin test edilmesinden ziyade uzman sistem, simülasyon ile test edilebilir. Burada sadece geliştirme zamanı azaltılmamakta aynı zamanda, test daha anlaşılır olmaktadır. Ayrıca, eğer simülasyon geçerli bir model ise, bu durumda simülasyonu kontrol eden veya simülasyona cevap veren bir uzman sistemin kendisi de geçerlidir. Bu yaklaşım, gerçek zamanlı proses kontrolunda faydalı olmaktadır. Bir uzman sistemin son kontrolu yaptığı sahalardaki proses, bu prosesin sürekli bir simülasyonu ile test edilebilir. Bu, daha fazla araştırma gerektiren ilginç bir sahadır (O'Keefe, 1986).

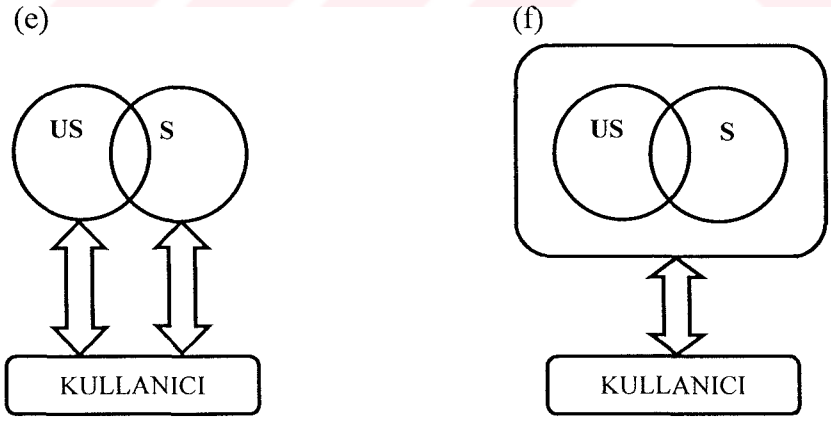


Şekil 4.5 Simülasyon (S) ve uzman sistemlerin (US) içiçe yerleştirilerek birleştirilmesi



Şekil 4.6 Simülasyon (S) ve uzman sistemlerin (US) paralel bir şekilde birleştirilmesi

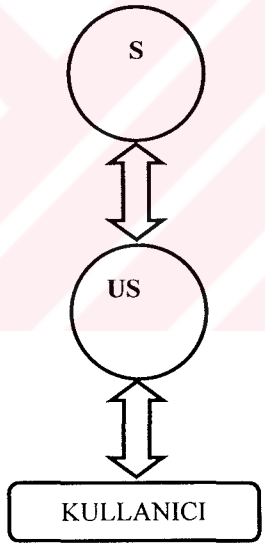
Şekil 4.5a'dan 4.6d'ye kadar olan temel yöntemlerin kullanıcıları diğerlerine direkt olarak geçiş yapamamakta olsa bile, birçok örnekte hem uzman sistem, hem de simülasyon bazı işleri yapmak için beraberce kullanılmaktadır. Şekil 4.7e'de etkin olarak, her ikisi de (uzman sistemlerin ve simülasyonun) bazı verileri ortak olarak kullandıklarından, simülasyon ve uzman sistem bir işte ortak olarak çalışmaktadır. Bu şekildeki bir ortak yapının veya beraber olmanın uygulaması, uzman sistemin tamamlanmış bir simülasyon modelinde veya modelin geliştirilmesinde, kullanıcıya yardım eden bir yöntem olduğunda ortaya çıkmaktadır.



Şekil 4.7 Simülasyon (S) ve uzman sistemlerin (US) bir işbirliği yapacak şekilde birleştirilmesi

Deneye ve analize yardım eden uzman sistemler, çalışma için ilginç bir alan oluşturmaktadır. Günümüzde simülasyon modellerinin tecrübesiz simülasyon kullanıcılarına yani, müşterilere sunulmasında, artan bir eğilim olmakta ve bu şekildeki modellerin yanlış kullanılmalarını engelleme ve doğru kullanımlarını destekleme çalışmalarında büyük bir artış gözlenmektedir.

Ortak bir simülasyon ve uzman sistem, Şekil 4.7f'de olduğu gibi, yazılımın büyük bir kısmı tarafından kapsanmaktadır. Buradaki herbir eleman, direkt olarak karar verici tarafından kullanılan büyük bir karar destek sisteminin bir parçası veya bir simülasyon ortamının bir elemanı olabilir. Hem simülasyona, hem de bilgi tabanlı metotlara bağlı yeni araçlar bu kategoriye düşmektedir. Bunlara örnek olarak, Carnegie-Mellon Üniversitesi'nde geliştirilen " Bilgi Tabanlı Simülasyon Sistemi " ve Rand Şirketi'nde geliştirilen " Kural Esaslı Simülasyon Sistemi " verilebilir.



Şekil 4.8 Zeki ön elemanlar

Bilgi tabanlı metotlara ait en önemli uygulamalardan birisi de, "Zeki Ön Elemanlar" olmaktadır. Bu formdaki bir yapı, şekil 4.8'de görüldügü gibidir. Bu, bir uzman sistemdir ve bir simülasyon paket programı ile kullanıcı arasında olup, kullanıcı ile bir diyalog kurarak, paket program kullanmak için gerekli komutları veya programı oluşturur ve daha sonra programdaki sonuçları yorumlar ve açıklar. Her ne kadar simülasyonu işletmese ve

sonuçları yorumlamasa da, simülasyon program üreteçleri zeki ön elemanların fonksiyonlarından bazılarını gerçekleştirir. Burada kullanılan bilgiler şunları içermektedir (Haddock, 1987):

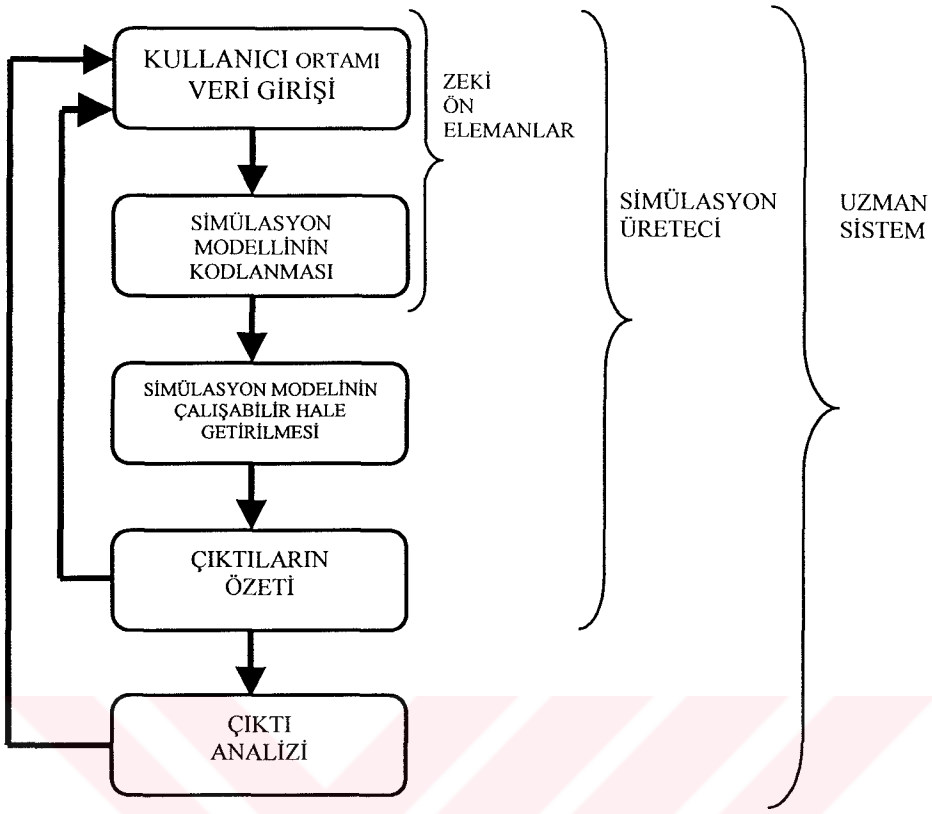
(1) Diyalog iletimi (Doğal dil arayüzü veya en azından kullanıcıya yönelik serbest formatlı giriş).
(2) Bazı kullanıcı modeli, (Buradaki sistem, kullanıcı tecrübeli veya tecrübesiz olsun, kullanıcıya bağlı ihtiyaçları ayarlar).
(3) Hedef dilin modeli (Burada bazı kararlar kullanıcıya danışılmadan zeki ön eleman tarafından verilir).

Bazı simülasyon üreteçleri 3 numaralı maddenin elemanlarını içermektedirler. Çok az sayıdaki üreteçler ise, 1 ve 2 numaralı maddeleri kapsamaktadır. Bunlardan bir istisna olarak FORTRAN ile yazılmış, ekolojik sistemler ile ilgili dinamik modeller üreten " ECO " verilebilir. Burada, Doukids ve Paul, 1 nolu maddeyi sağlayacak şekilde faaliyet-çevrim diyagramlarına bağlı model spesifikasyonlarının çıkarılması için doğal dil tabanlı bir sistem geliştirmişlerdir.

Simülasyon ve uzman sistemlerin birleştirilmesindeki kabulde, yukarıda açıklanan sınıflandırmalar kullanışlı olmamaktadır. Bu şekildeki bir birleşim ciddi sonuçlar ortaya çıkarabilir. Uzman sistemlere ilgi duyulmasına karşın, bunların tam olarak uygulanması çok az olmaktadır. Yukarıdaki analiz de, simülasyonda uzman sistemlerin uygulanmasındaki üç noktayı ortaya çıkarmaktadır. Bu üç nokta şu şekilde özetlenebilir:

(1) Mevcut simülasyon ve bilgi tabanlı metotların birleştirilmesiyle oluşturulan yeni simülasyon araçları.
(2) Özellikle deney ve analiz safhasında tecrübesiz simülasyoncular için fikir veren sistemler.
(3) Mevcut simülasyon paket programları için geliştirilen zeki ön elemanlar

Zeki ön elemanlar, simülasyon üreteçleri ve uzman sistemlerin birbirleriyle olan bağlantıları Şekil 4.9'da gösterilmektedir (Haddock, 1987).



Şekil 4.9 Uzman sistemler, simülasyon üreteçleri ve zeki ön elemanlar

4.4. Bir Simülasyon Modeli İleYapay Zekâ Arasındaki Farklar

Simülasyon dilleri veya simülasyon modelleri yapay zekâda kullanılan çok sayıdaki fikirleri içermektedir. Örnek olarak, yapılacak işlemlerin veya gezer birimlerin dinamik olarak değişen kabiliyetlerini, karakteristiklerini tanımlayan özellikler (yapı), sistemdeki gezen birimlerin akışını dinamik olarak değiştirmek (üretim kuralları veya şartlı olasılıklar), durum değişkenlerine bağlı olarak sistemi değiştirmek (model tabanlı programlar) ve şebeke formundaki sistem bilgilerinin sunulması gösterilebilir. Simülasyon kavramları ile yapay zekâda kavramlarının bir karşılaştırılması Çizelge 4.1'de verilmiştir (Reddy, 1987). Doğal olarak, " bugün bildiğimiz doğru bir şekilde yazılmış simülasyon modeli ile yapay zekâ tabanlı simülasyon sistemi arasındaki fark nedir? " şeklinde bir soru sorulabilir.

Çizelge 4.1 YZ kavramları ile simülasyon kavramlarının karşılaştırılması

Yapay Zekâ	Simülasyon
Nesne	Gezen birim / kaynak / kısıt / işlem / faaliyet / olay bilgisi / saat
Özellik	Olay tanımlama / veri depoları
Metot	Olay davranışı
Kontrol mekanizması	Şarta bağlı olay
Mesaj	Olay icrası
Nesne ağı	Model / durum
Dünya/İçerik	Senaryo / kontrol noktası
Kurallar	Kısıtlar / olay davranışları / deney üretme teknikleri
Mantık	Kısıtlar
Planlama	Dneylerin tasarlanması
Teşhis	Sonuçların analizi
Öğrenme	Model içindeki nedensel bağlantıların ortaya çıkarılması

Temel fark, modelin kurulması ve işletilmesinde ortaya çıkabilir. Bugün simülasyon, modeli tanımlayan bir senaryoya (Girdiler) karar veren, deneyi işleten, sonuçları analiz eden, başka bir senaryoya karar veren, deneyi çalıştıran ve benzeri durumlardaki iteratif bir prosestir. Mevcut simülasyon sistemleri, uygun bir modele karar vermede veya problemimizin çözümünü bulmamızda ve modelin incelenmesinde bize yardımcı olmamaktadır. Modelleyiciler, sistemin mantığını ve davranışını zorunlu deyimler kümesine dönüştürmelidirler. Bu deyimler daha sonra, kontrol deyimi tarafından durdurulana kadar, yukarıdan aşağıya doğru işletilmelidirler. Yapay zekâ tabanlı bir uzman sistem, modelleyicinin sistem hakkındaki bilgileri açıkladığı (özellikle elemanların tanımı), amacı tanımladığı ve çözümün bulunması için bilgisayarın çalıştırıldığı çok farklı bir metodolojiyi izlemektedir. Sistem hakkındaki bir bilginin açıklanmasında modelleyici, tüm mümkün ve/veya kabul edilebilir modellerin tümünü tanımlamaktadır. Uzman simülasyon sisteminin sorumluluğu, arzulanan spesifikasyonları sağlayan modeli otomatik olarak bulmak ve istenilen çözümün elde edilmesinde uygun bir araştırma yöntemini işletmektir.

İkinci fark ise, yapay zekâ tabanlı bir sistemde veri tabanı, bilgi tabanı ve kontrol yapısı birbirinden ayrıdır ve her biri, diğerini etkilemeden kolayca değiştirilebilir. Bugün birçok simülasyon modeli, her ne kadar bazı simülasyon dilleri iki veya üç kısma ayrılsa da (model, deney, çıktı analizi gibi), bunlarda bilgi ve kontrolun entegrasyonu sağlanmıştır. Bir yapı içinde, sistem modeli ve deneysel tasarım arasındaki temel farklılık Zeingler ve Ören (1979) tarafından geliştirilen sistem teorisine dayanmaktadır.

Üçüncü fark, veri tabanının yapısındadır. Klasik veri tabanı sistemleri, statik (verilerin işlenmesi esnansında) veri tabanlarında gerçekleştirilen, geçerli olarak tanımlanmış veri yapılarını ve operasyonlarını gerektirmektedirler. Bu, ihtiyaçların değişmesine bağlı olarak, kolayca değişmeyen çok kararlı bir işlemdir. Aksine yapay zekâ, veri tabanı sisteminin yapısını, toplanacak ve kullanılacak ilave bir veri tipini sağlamaktadır. Bu sembolik veriler, dar bir problem alanı hakkındaki olaylara ait bilgileri, yargıları, kuralları, sezgiyi ve deneyimi (tecrübeye dayalı bilgi) temsil etmektedir. Sezgisel veri, veri tabanındaki klasik veri elemanları arasındaki ilişki kurallarını vermektedir ve burada kolayca eklemeler yapılabilir ve de değiştirilebilir.

Klasik veri tabanı ile sezgisel verinin birleştirilmesi, " Bilgi Tabanı " olarak adlandırılır. Buna göre yapay zekâ sistemleri, klasik veri tabanında gerekli olan veri tabanındaki veri elemanlarının direkt olarak ilişkilerinin gösterilmesine bağlı değildir. Bu durum, bilgi tabanının değerlendirilmesi (sonuç mekanizması olarak bilinen), bilgi tabanındaki verinin yorumlanması, lojik sonuçların oluşturulması ve konu ile ilgili bilginin çıkartılmasında, bilgi tabanının değiştirilmesinde sistemin farklı prosesleri kullanmasına izin vermektedir. Bu ilave esneklik ile yapay zekâ sistemleri, klasik veri tabanı bilgi sistemleri ile çözülemeyen bazı problemlerin çözümünü verebilmektedir.

Mevcut simülasyon modelleri ve bir uzman simülasyon sistemi arasındaki dördüncü fark, belirli karakteristikler şeklinde olabilir. Aşağıda, klasik simülasyonun karakteristiklerinden bazıları verilmektedir. Bunlar ;

- Temel olarak nümerik,
- Algoritmik (Çözüm adımları kesin olarak belirli),

- Entegre bilgi ve kontrol,
- Ayrı olarak veya modelin dışında yapılan, modelleme işlem adımları (Örneğin, uygunluk için girdi verilerinin test edilmesi, örnek boyutunun belirlenmesi, işletilecek deneyin tasarlanması vb.),
- Kullanıcı tarafından planlanmış model (Yani kullanıcı, programın operasyonlarını açıklamalıdır).

Diğer yandan yapay zekâ tabanlı simülasyon sistemi şu özelliklere sahiptir:

- Birçok sembolik proseslere sahiptir,
- Modele yönelik araştırma kullanmaktadır (Çözüm adımları belirli değildir),
- Bilgi alanından ayrı bir komut yapısına sahiptir,
- Kullanıcı tarafından verilen kararların minimize edilmesi için mümkün olan uzman bilgiler modelin içinde bulunmalıdır,
- Kendi tecrübesinden öğrenebilecek ve gerektiğinde kendini değiştirebilecek bir modele sahip olması gerekir. Genelde yapay zekâ sistemleri teorik olarak mümkün olmasına rağmen bunu henüz tam olarak başaramamaktadır.

Farklılıkla ilgili olarak beşinci metot ise, yapay zekâ tabanlı simülasyon sisteminin kullanıldığı yerdir. Michie'ye (1980) göre bir uzman sistem için üç farklı kullanıcı modu vardır. Bunlar ;

(1) Sistem bilgisini arttırmak veya azaltmak, öğretici olarak kullanıcı (kullanıcı öğretici rolünde)
(2) Problemlere cevap verme (kullanıcı müşteri rolünde)
(3) İnsanların kullanımı amacıyla, bilgi tabanının toplanması (kullanıcı öğrenci rolünde)

"Kullanıcı öğretici rolünde" modunda, simülasyon sistemi güçlü model geliştirme kabiliyetleri sağlamaktadır. Bu kabiliyetler kullanıcılara, yeni bir modelleme dilinin öğrenilmesi için 100 veya daha fazla saat harcamadan, gerçek veya teklif edilen sistemin sunulmasını kolayca geliştirmelerine ve değiştirmelerine, olanak vermektedir. Modelin

formu veya veriler için gerekli formatı bilmeden, sistem veya gezen birimler hakkındaki tanımlayıcı bilginin kullanıcı tarafından geliştirilmesine olanak veren ayrı bir veri girişi arayüzü olmalıdır.

"Kullanıcı müşteri rolünde" modunda simülasyon, kullanıcıya kendi ana dilinde soru sormasını sağlar ve sistemin hangi tür deneyi çalıştırması gerektirdiğine, gerekli örnek boyutuna vb. karar verir. Simülasyon sistemi, kullanıcı tarafından yönetilmeden sistemin mevcut durumuna bağlı olarak (rotalar, işlem zamanları vb.) soruları cevaplamak üzere ihtiyaç duyduğu bilgileri ortak veri tabanından alır ve bu sistem öğrenme yeteneğine sahip olup, böylece sistem çalışırken kendi kendini modifiye edebilmektedir. Bu durum, işletim boyunca geriye doğru izleme veya zaman sapmaları özelliğine ve ileriye veya geriye doğru bağlanma özelliğine sahip olabilir. Örnek olarak sistem, optimal planının veya programın bulunmasına imkan vermekte ya da, belirli bir amaca nasıl erişileceğini bulabilmektedir. Ancak bugünkü modellerin birçok, verilen bir senaryo (girdiler) için olası sonuçlar vermektedir. Son olarak kullanıcı, istenilen çıktı tipini veya formunu seçebilmelidir.

"Kullanıcı öğrenci rolünde" modunda ise, sistem uzmanlar tarafından verilen sayısal kararların (planlama) sonuçlarını alma özelliğine sahiptir ve kullanılan kuralların veya sezgisel metotların özetini çıkartır. Böylece o, diğer kişilere de öğretilebilir. Sistem ayrıca model çözümlerini açıklama özelliğine de sahip olmalıdır.

4.5. Uzman Sistemlerin Simülasyon Metodolojisi Üzerindeki Etkileri

Yapay zekâ ve özellikle uzman sistemler sahalarından elde edilen teknikler ve teorik sonuçlar, simülasyon uygulamaları için yeni ve ilginç bir teknoloji ortaya çıkarmaktadır. Bu teknoloji de başlangıç aşamasından zamanımıza kadar epeyce yol katetmiş ve katetmektedir. Burada şöyle bir soru sorulabilir: "Bu teknik simülasyon metodolojisi üzerinde önemli bir etkiye sahip olacak mıdır? ". Bu soruyu cevaplama yaklaşımındaki bir yol, uzman sistem tekniğinin uygulanması için uygun bir sahanın olup olmadığını görmek üzere simülasyonu incelemektir. Burada hatırlanması gereken bir konu da, uzman sistemin çözümü için belirli bir problemin uygunluğunu belirlemek üzere göz önüne alınan

kriterlerin, bu yeni tekniğin başarıları ile beraber birçok başarısızlıklarının sonucu olduğudur. Aşağıda bu karakteristiklerin bazıları incelenmiştir (Lapin, 1991).

(1) *"Bir veya daha fazla uzman kişilerin varlığı önemlidir".* Uzman sistem ile ilgili genel bir yanlış anlama, uzman sistem metodunun hiçbir şey olamadan bilgiyi oluşturacağı ve uzman olacağıdır. Metodoloji ile ilgili teoriler ve kavramlarda ortaya çıkan avantajlar, simülasyonun sanat kısmının, ispat edilen metotlar ile yer değiştirme arzusuna sevk etmektedir.

(2) *" Bilgi tabanı yeterli derecede sınırlı olmalıdır ve tercihen spesifik alanda olmalıdır".* Bu karakteristikten elde edilen üretim veya bilgisayar şebekeleri gibi belirli alanlarda sınırlı olan uzman sistemler, simülasyon sistemleridir. Genel amaçlı bir genel uzman sistem simülasyon sistemi, çoğunlukla geçerli değildir ve yeni sistemler spesifik alanlarda olmalıdır.

(3) *"Problem ile ilgili bir uzmanın performansı, yeni başlayan birinin veya stajyerin performansından daha iyidir".* Uzman tarafından problemin başarılı çözümleri, bir amatör tarafından oluşturulan çözümden daha iyi ve daha hızlıdır ve daha büyük başarı, tecrübeye göre elde edilen bilgiye veya mental kabiliyete bağlıdır. Bu kriter, uzman kişilerin kabiliyetlerinin amatörlere aktarılması için kullanılabilir.

(4) *"Problem kararları, çok değişik alternatiflerin ve belirsizliklerin de göz önüne alınmasını gerektirir".* Uzmanın karar alternatiflerini sadece ortaya çıkarması yetmez bu alternatiflerin aynı zamanda yeterli sayıda ve sayısal olması gerekmektedir. Ayrıca, bir alternatifin uygunluğunu belirleyen faktörler belirsizlik içerebilir. Uzman (pratik tabanlı olarak çalışmıyorsa), mümkün sonuçlar ve bunların delilleri arasındaki destek derecelerini atayabilmelidir. Simülasyon çalışmasının bu karakteristikler altında uygun olduğunu görmek amacıyla, sadece başlangıç koşulları, otokorelasyon, örnek boyutunun belirlenmesi gibi konular göz önüne alınmalıdır.

(5) *"Gezen birimlerin sayısı, ilişkili özellikleri ve alternatif veya karar faktörlerini tanımlayan kullanışlı ilişkiler sınırlıdır".* Temelde, bilgileri temsil etmek ve kullanmak için uzman sistemin işleteceği (elemanlar, olaylar, karakteristikler vb.) elemanlar sayıca sınırlıdır. Buna göre mevcut uzman (bazı özel eğitimlerle) hangi elemanın, hangi karakteristiklerinin ve bunlar arasındaki hangi ilişkinin problemle ilgili olduğunu tanımlayabilir. Bu durumun karakterize edilmesindeki diğer bir yol ise, işin uzmanın gerçek düşüncesine göre bazı sınırlı zaman periyodu gerektiren bağımsız işlere bölünüp bölünemeyeceğini belirlemektir.

Pratik bir görüş açısından ise, simülasyon sistemleri için uzman sistem metodunun kullanılmasının pratikliği göz önüne alındığında, hesaba katılması gereken birçok özellikler vardır. Gelişmenin maliyeti ile kâr arasındaki değişim yüksektir. Ancak bu şekildeki sistemlerin geliştirilmesinin potansiyel kârı, gelişimi kaçınılmaz yapmaktadır.

Burada diğer bir konu daha ele alınmalıdır. Simülasyon sadece uzman sistem metoduna ihtiyaç duymamakta, aynı zamanda uzman sistemlerin de simülasyona ihtiyaçları olmaktadır. Eğer yapay zekâ ve uzman sistemlerin mevcut kullanılmaları incelenirse, bu kullanımlardan bazılarının zaman alanlı olduğu (yani, karar için bir zaman boyutu yoktur) görülür. Çünkü yapay zekâ uzmanları, zamanın nasıl kullanılacağını henüz bilmemektedirler. Simülasyonun gerçek gücünden birisi de, zamana göre olayları ve etkilerini tahmin etme ve projelendirme özelliğidir. Uzman sistemin kullanımı, simülasyon metodu birleştirilene ve kullanılana kadar sınırlı kalmaktadır. Bu yüzden, her iki alan da birbirlerinden yararlandığından dolayı birleşme kaçınılmazdır.

4.6. Programlama Dillerindeki Gelişmelerin Simülasyon ve Uzman Sistemlere Etkileri

Japon " Beşinci Kuşak Bilgisayar Projesi "nde ve Amerika'da Austin Teksas'taki Mikro Elektronik ve Bilgisayar Şirketi'nde simülasyonla ilgili geniş çalışmalar yapılmıştır. Bu çabaların her ikisi de, uygulamaya yönelik olarak, donanım ve yazılımdaki üstünlükleri geliştirmeyi amaçlamaktadır. Yazılım terimlerinde, bilgisayarla iletişim ve bağlantı yolu ile ilgilidir. Burada " Beşinci Kuşak "ın nereden geldiği ve ne anlam verdiği kesin olarak

belli değildir. Çizelge 4.2'de beş kuşak boyunca yazılım ve donanımdaki gelişmeler gösterilmektedir. İlk kuşak makina dili, ikincisi ASSEMBLER dilidir. Üçüncü kuşak dilleri ise, FORTRAN, BASIC, COBOL, C, PASCAL ve ADA gibi genel amaçlı yüksek düzeyli dilleri içermektedir. Bunlar prosedüre yönelik dillerdir (Shannon vd., 1985).

Çizelge 4.2 Beş kuşak boyunca donanım ve yazılımdaki gelişmeler.

Kuşak	Donanım	Yazılım
1	Termiyonik (iyon) supablı bilgisayarın yapılması	Makina kodu
2	Bilgisayarda kesikli yarı iletken transistör kullanımı	Assembler (düşük seviyeli sembolik dil)
3	Bilgisayarlarda entegre devrelerin kullanımı (çok ince silikon bir yüzey içinde fabrikasyon olarak transistör, direnç, kapasitör ve diyotlar)	Yüksek seviyeli diller (COBOL, FORTRAN, BASİC, PASCAL, C vb.)
4	Bilgisayarlarda çok büyük ölçekli entegrasyon (VLSI) teknolojisinin geliştirilmiş devrelere uygulanması	Program üreteç uygulamaları, prosedürel olmayan diller (kelime işlemciler, tablolama paketleri, grafik paketleri vb.)
5	Bilgisayarlarda insan düşünce işlemini taklit eden paralel işlem yapmanın uygulanması	Yapay zekâ, çoklu değer veya bulanık mantığı kullanan diller

Dördüncü kuşak dilleri, birçok yazılım kategorisini içeren bir şemsiye terimdir. Burada en azından üç temel alan vardır. Bunlar sunuş dilleri (geçerli soru, doğal soru, raporlama, grafik, vb.), özel fonksiyonlarda odaklaşmış özel diller (tablolama, modelleme, analiz, simülasyon vb.) ve tüm uygulamaları oluşturmak için (veri tabanı yöneticileri) verilerin elde edilmesi, modifikasyon ve tanımlama ile ilgili uygulama üreticileridir. Bu dillerin, son kullanıcı araçları olarak piyasada bulunmalarına rağmen, bunların öğrenilmesi gerektiği bir gerçektir. Bunlardan sadece bir kaçı programcı gerektirmeyen (tablolama, LOTUS'taki gibi) araçlardır. Bunların birçok kullanıcı eğitimi gerektirmektedir. Gerçekte 4.kuşak dilleri, programcılar için mükemmel bir prodüktivite aracıdır.

Beşinci kuşak dilleri ise, kendilerini 4.kuşaktan ayıran birçok özelliklere sahiptirler. Bunlar doğal dil komutlarının kullanılmasıyla öğrenmenin, kullanmanın ve desteğin kolaylaştırılması; dökümantasyonun basılması; güncelleştirmenin veya değişikliklerin kolaylaştırılması; bilgisayardan bilgisayara aktarımın kolaylaştırılması ve tasarıma bağlı geçici hafızanın kullanılmasıdır. Ayrıca, programcı olmayanlar kendi uygulamalarını geliştirdiklerinde beşinci kuşak dilleri, programlama işini tamamen otomatik olarak yapabilmektedirler. Doldurulacak formları menüler, ikonlar veya karşılıklı sorular şeklinde sunan sistemler gerekmektedir. Arzulanan uygulamanın, kullanıcıya daha sonra ihtiyaç duymadan, kullanıcının cevaplarını verebilecek nitelikte olabilmesidir. Başka bir deyişle, 5.kuşak simülasyon sistemleri, 4.kuşakta geliştirilen araçları entegre etmektedir ve simülasyon modelleme uzmanı ile beraber, uzman programcının bilgisini elde etmektedir.

Beşinci kuşak dilleri mantıksal problemlere izin veren, geçerli çıkarımlar yapmak için bilgi tabanıyla birlikte bir yapıyı da bünyesinde bulundurabilen, böylece rasyonel bir insanın davranışlarını benzetebilen bir yapıdadır. Pratikte bir insan, kendi deneyimlerinden öğrenir. Benzer şekilde, bir bilgisayarın da bunu gerçekleştirebilmesine yönelik yaklaşım ve modellerin de entegrasyonu sağlanmaktadır. Beşinci kuşak ile tanımlanmış olan genel elemanlar ise şunlardır (Szymankiewicz, 1988):

- İnsan-makina arayüzü: Kullanıcının problemini doğal bir dil ile menüler, sesler veya resimleri kullanarak, kolaylıkla etkileşimli bir şekilde tanımlamasını sağlayan bir birimdir.
- Uzman sistemler: Bir bilgi tabanı ile uzman bilgisini kullanarak çıkarımlar yapan bir yapay zekâ tekniğidir.
- Çok büyük ölçekli entegrasyon: Daha çok küçültülmüş fiziksel birimler içerisinde yoğunlaştırılmış ucuz işlem gücü.

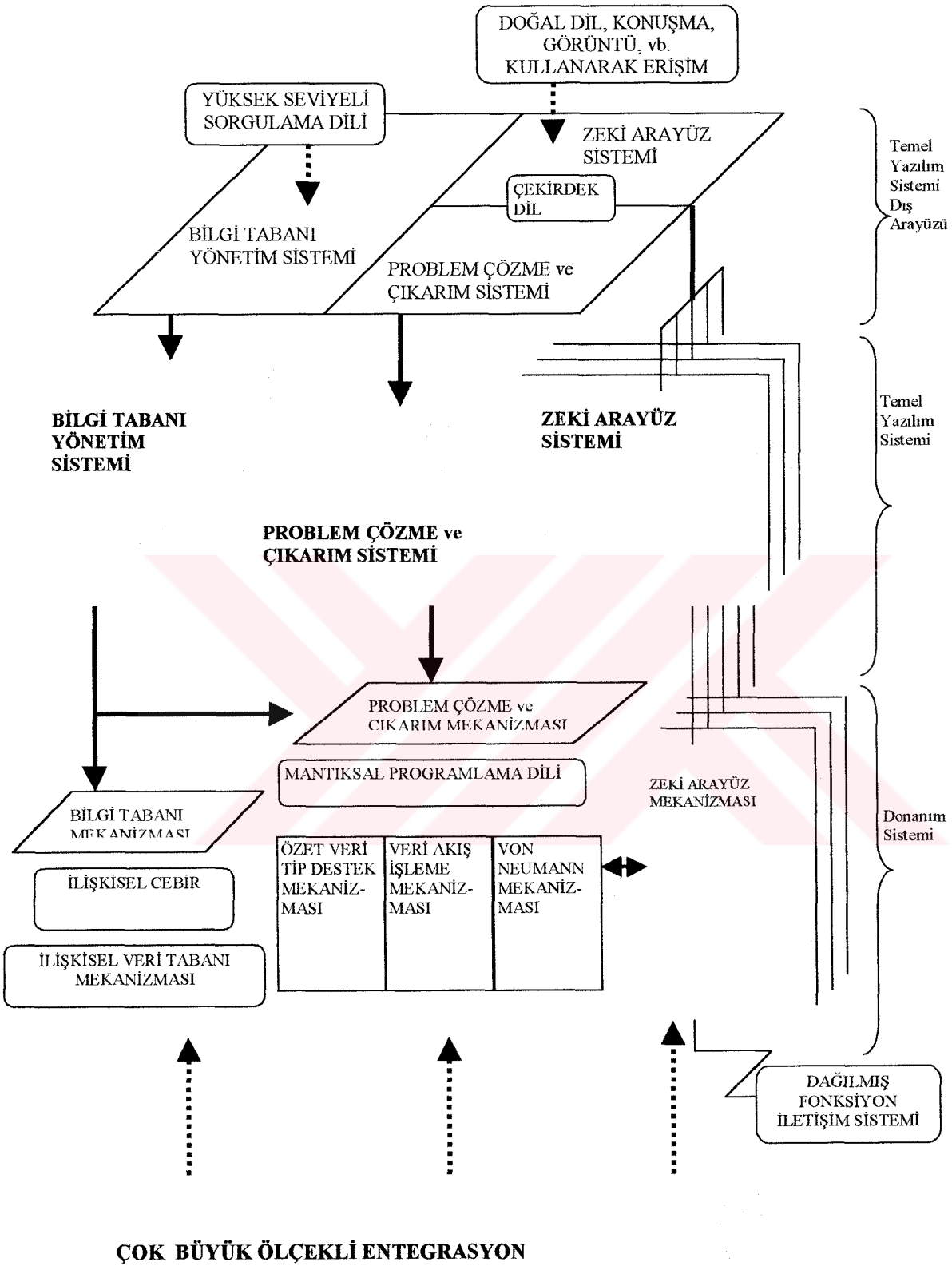
Beşinci kuşak makinalar, çok büyük ölçekli bilgi tabanlarıyla desteklenmek ihtiyacındadırlar ve mantıksal çıkarım opsiyonu ile çok hızlı çalışabilirler; yazılım ve donanımda paralel işleme yeteneğini kullanırlar ve kullanıcı-makina arayüzü ile yüksek kullanım olanağı sağlamaktadırlar. Bunlarla ilgili çalışmalar halen devam etmektedir.

Beşinci kuşak donanım ve yazılımların genel yapısı Şekil 4.10'da görülmektedir (Szymankiewicz, 1988).

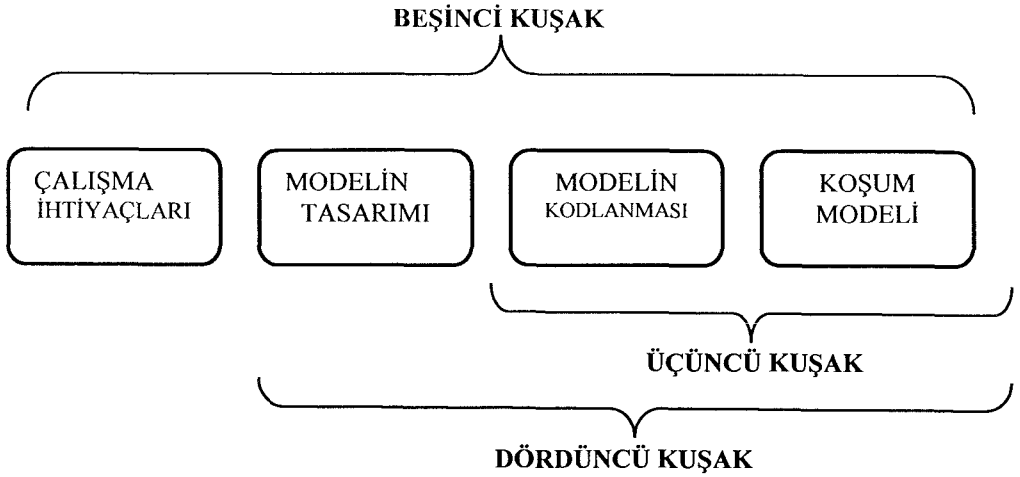
Beşinci kuşak yazılımın etkisini incelemede diğer bir yol ise, Şekil 4.11'in göz önüne alınmasıdır. 3.kuşak dilleriyle, ASSEMBLER veya makina diline nazaran, simülasyon modellerinin yazılması ve işletilmesi daha kolaydır. Ancak, modelleme tecrübesi ile beraber programlama uzmanının servisini elde etmek zordur. Günümüzde 4.kuşak simülasyon dillerinin gelişmesiyle, simülasyonun kullanımı ve kabul edilebilirliği, ileriye doğru büyük bir adım atmıştır. 4.kuşak dilleri, sadece program yazmayı kolaylaştırmayıp, aynı zamanda model tasarım safhalarında, bir dünya görüşü ve kavramsal rehberlik sağlamaktadır. Ne yazık ki bugünkü simülasyon, kullanıcının amaçlarını, hedeflerini ve ihtiyaçlarını uygun veri kullanarak, doğru deney tasarımı altında işleterek ve uygun şekilde analiz ederek, uygun bir modele dönüştürmek için büyük çapta uzmanlık gerektirmektedir. Beşinci kuşak uzman simülasyon sisteminin amacı, kullanıcının ilgili sistemi, amaçları, hedefleri ve cevap formlarını doğal anlamlı bir dilde tanımlamasını sağlamak ve gerisini uzman sisteme bırakmaktır.

Dördüncü ve beşinci kuşak simülasyon sistemlerinden, birbirlerine geçiş başlamıştır. Etkileşimli grafik model kurulmasının ve veri girdilerinin artan kullanımı grafiksel ve animasyonlu çıktı analizi, modelleme, deneysel ve çıktı analizi yapılarının ayrılması, dilin içinde çok sayıda istatistiksel analizin kullanılması gibi tüm adımlar, gerekli ilk adımlar olmaktadır ve ardışık olarak devam etmektedir. Fakat çok yakın bir gelecekte bazı yeni ve radikal olarak farklı simülasyon sistemlerinin gelişimini görmek mümkün olabilir. Yapısal sınırlamalardan dolayı, üçüncü kuşak dillerinin geliştirilmesi mümkün değildir.

Mevcut olarak kullanılan simülasyon dilleri, prosedüre yönelik veya komuta yöneliktir. Yani bir program, çözüme erişmek için gerekli adımları açık olarak belirtmelidir. Uzman sistem geliştirilmesinde genelde kullanılan diller, LISP ve PROLOG, nitelik olarak birbirlerinden farklıdır. Orijinal LISP (List Processing) fonksiyonel bir dildir. PROLOG (Programming in Logic) ise, açıklayıcı veya tanımlayıcı bir dildir. PROLOG dilinde, sorular halinde, elemanlar ile ilgili deyimler ve kurallarla, sadece problemin tanımlanması



Şekil 4.10 Beşinci kuşak donanımların ve yazılımların genel yapısı



Şekil 4.11 Programlama dillerinin rolü

gerekir. Problemin tanımlanması yeteri kadar hassas ise, problem kolaylıkla çözülebilir. Tekrarlı fonksiyonları destekleme kabiliyeti de önemlidir. Programlar ve veriler, sembolik işletimi kolaylaştırabilecek güçlü ve üniform bir mekanizma yardımıyla her iki dilde de oluşturulabilir. Bunların dilbilgisi ve yorumlama yapısı, programcıya diğer programlarla beraber çok çeşitli şekilde verilerin (sayılar, kelime parçaları, karakter dizileri ve listeler) işletilmesini sağlayan programları ortaya çıkarmasına ve programın davranışını değiştirmesine imkân vermektedir.

Simülasyon için PROLOG ve/veya LISP benzeri dillerin kullanılmasından elde edilen avantajlar aşağıda verilmiştir :

- Genel prosedüre yönelik anlamsal ağlara (semantiklere) ilave olarak, mantığa bağlı açıklayıcı anlamsal ağlar sağlanmaktadır.
- Programlar ve veriler, form olarak bellidir ve bu yüzden kolaylıkla işletilebilirler.
- Prosedürlerin elemanları, diğer programlama dillerinde olduğu gibi, girdi ve çıktı parametreleri olarak sabit değildirler ve prosedürler çok sayıda girdi ve çıktıya sahip olabilirler.

- Geriye izleme, verilen bir problemin çözümüne ait bir kümenin bulunmasında güçlü bir araçtır. Bu, aynı zamanda yüksek düzeyli bir iterasyon formu şeklinde ortaya çıkmaktadır.
- Temel elemanlar (alanlar, değişkenler ve terimler) klasik programlama dillerinde kullanılan dizi ve kayıtlardan daha üstün olan genel ve esnek bir veri yapısını sağlamaktadır.

4.7. Nesneye Yönelik Programlama, Simülasyon ve Yapay Zekânın Birleşik Kullanımı

Nesneye yönelik programlama, yapısal programlamaya bir alternatif olarak ortaya çıkmış ve çıktığından zamanımıza kadar da, yoğun bir ilgi ile kullanılmaya başlanmıştır. Bunun nedeni olarak, bu tür programlamanın daha biçimsel oluşu, yazılım geliştirme ve grafik etkileşime geçiş bakımından programcıya kolaylıklar sağlaması gösterilebilir. NYP yaklaşımı ilk kez 1966 yılında SIMULA simülasyon dilinde kullanılmıştır (Ayağ vd., 1991).

NYP'nin ana fikri, bir nesne (veri yapısı) hakkındaki tüm bilgiyi organize etmek ve tek bir yerde depolamaktır. Bu ise, bilginin kapsülleştirilmesi anlamına gelmektedir. Nesneye yönelik programlamada her nesne, içinde onu işleyecek olan metotları (algoritmaları) taşır. Bu sayede her nesne kendisiyle ilgili tüm işlemleri kontrol eder ve böylece de programlar, diğer programlardan bağımsız olarak kendi verisi ve işleme metotları ile bir bütün olarak tasarlanabilir. Belirli bir problemin çözümü için nesneye yönelik yaklaşım uygulamasında, problemin tanımlanması, alan nesnelerinin tanımlanması, nesnelere gönderilecek mesajların tanımlanması ve bir problemin çözümündeki mesaj sıralarının tanımlanması olmak üzere dört adımlı bir prosedür izlenebilir (Pinson ve Wrener, 1988).

Nesneye yönelik programlama, simülasyon modelini oluşturma işlemlerine de kolaylıkla uygulanabilir. NYP açısından yapılması gerekenler şu şekilde özetlenebilir (Ayağ vd., 1991):

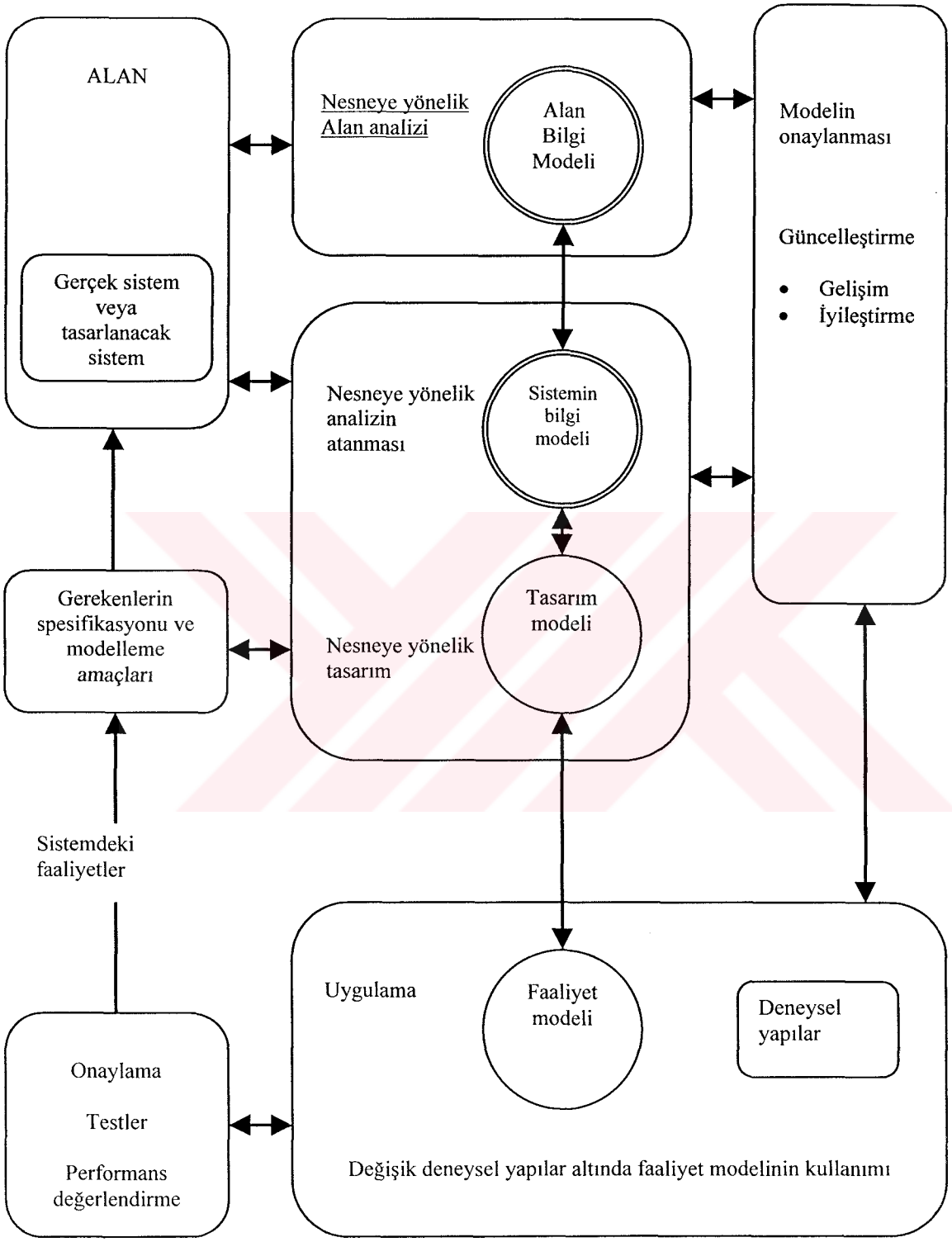
- Kullanıcı ilk olarak gerçek hayat nesneleri (örneğin bir imalat sistemi için iş parçası, tezgah, işçi, vb.) ile ilgili model nesnelerini oluşturur ve tanımlar.
- Model nesnelerinin davranışlarını, gerçek hayat nesnelerinin davranışı olarak tanımlar ve bunların çeşitli girdilere tepkisini gösterir.
- Nesneler, fonksiyonel ve ilişkisel faaaliyetleri tanımlayan mesajların geçmesi suretiyle etkileşirler. Nesneler arasındaki tüm haberleşme, mesajlar aracılığıyla olur.

Nesne modelleme çevriminin genel bir görünümü, alan analizinin, nesneye yönelik analizin, nesneye yönelik tasarımın ve sonuç olarak uygulamanın birbiri ardı sıra belirlenmesi şeklinde karşımıza çıkmaktadır. Şekil 4.12'de nesneye yönelik modelleme işleminin bir diyagramı görülmektedir (Hill, 1996).

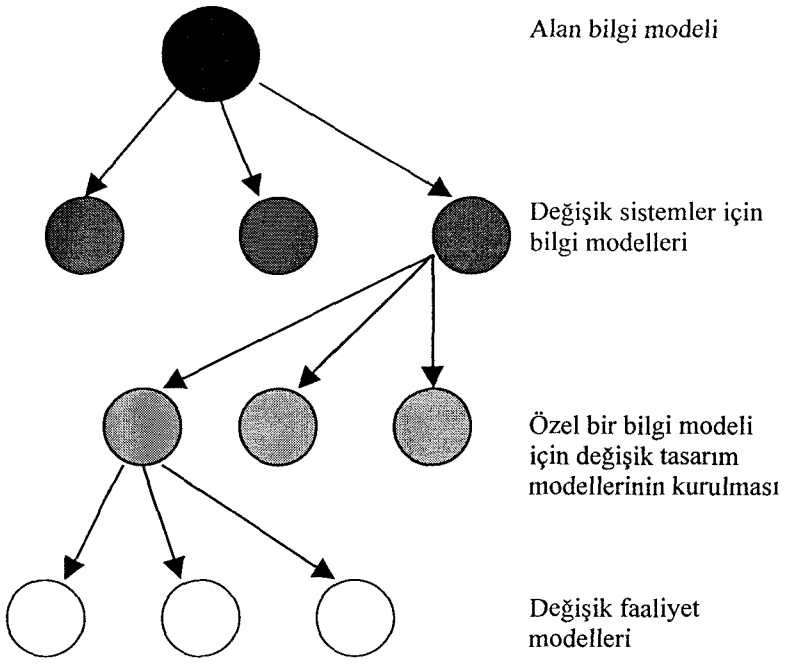
Nesneye yönelik yaklaşım, gerçek hayat nesnelerini tanımlayan doğal mekanizmayı sağlar. Dahası, benzer nesneler sınıflar halinde gruplanabilir ve bu sınıflar, sınıf-alt sınıf hiyerarşisi içinde organize edilebilirler. Şekil 4.13'de değişik modeller arasındaki ilişkilerin bir hiyerarşisi görülmektedir.

Kaiser ve Beaumariage (1997) tarafından yapılan bir çalışmada simülasyon, NYP ve YZ teknikleri bir arada kullanılarak, kavramsal bir model oluşturulmuştur. Geliştirilen kavramsal tasarım Şekil 4.14'de görüldüğü gibidir. Burada görüldüğü gibi, sistem dışı ve sitem içi olmak üzere iki zeki yapı mevcuttur ve zeki ajan nesnelerinin nasıl kullanıldığı da açıkça görülmektedir. Her bir simülasyon nesnesi, karar vermek için gerekli olan bilgiyi gönderecek ve dolaysız olarak birleştirilmiş bir zeki ajan nesnesine sahiptir.

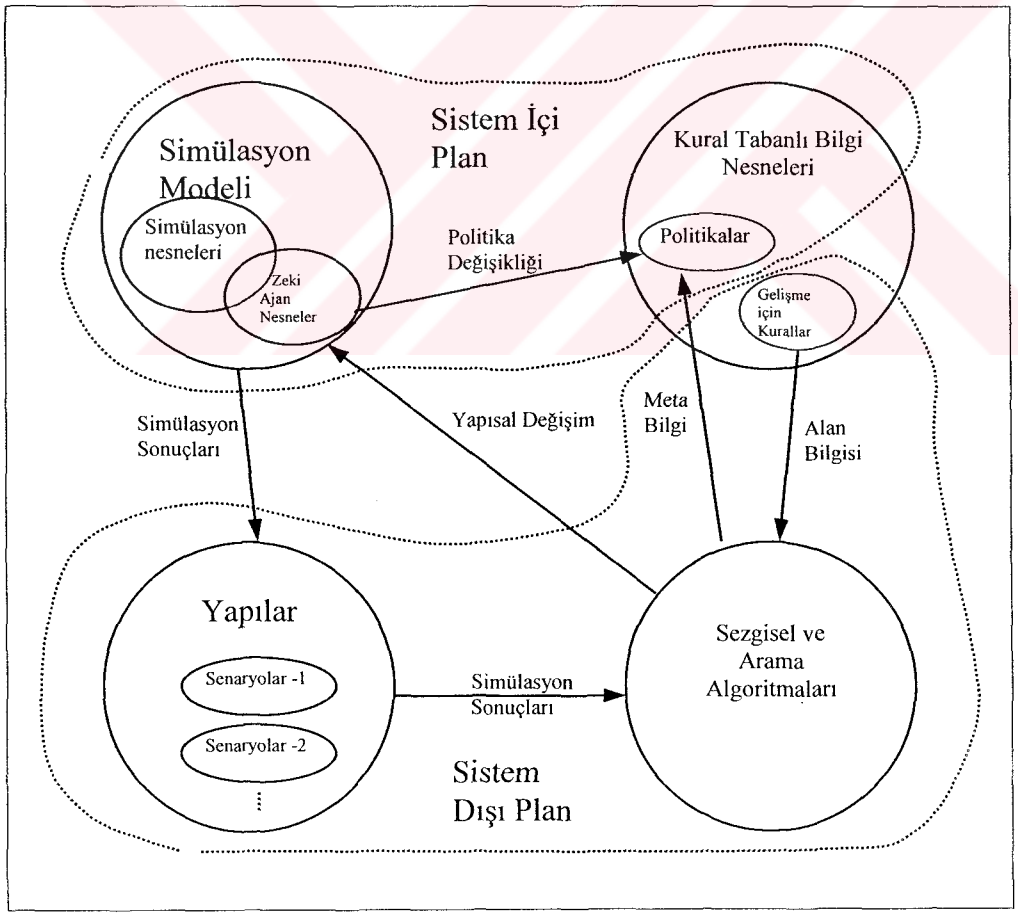
Simülasyoncular uzun zamandır simülasyon dilerinin içerisine yapay zekâ kabiliyetlerini ilave etmek istemişlerdir. Maalesef dillerin çoğu için uygulaması yararlı YZ teknikleri, geliştirilen simülasyon dilleri içerisinde yerinde değildirler. Nesneye yönelik programlama dilleri, YZ ve simülasyonun her ikisi için birbiriyle uyumlu bir temsilini sağlayacak yeterli bir güce sahiptirler. Zekâ ilave edilmiş bir simülasyon modeli, yapay zekâ kabiliyetlerini kendi yapısına almış olarak geliştirilmiş zeki simülasyon nesnelerinden



Şekil 4.12 Nesneye yönelik modelleme işlemi



Şekil 4.13 Değişik modeller arasındaki hiyerarşik ilişkiler.



Şekil 4.14 Simülasyon, yapay zekâ ve NYP'nin entegrasyonuyla oluşturulmuş kavramsal bir model.

oluşmaktadır. Burada zeki simülasyon nesneleri (ajanlar) diğer simülasyon nesneleri için soruları cevaplar ve problemleri çözerler; bir simülasyon modelinin işletimi esnasında diğer simülasyon nesnelerine yardım ederler (Bolte vd., 1993).

4.8. Yapay Zekâ ile Tamamen Bütünleşik Zeki Bir Simülasyon Ortamı

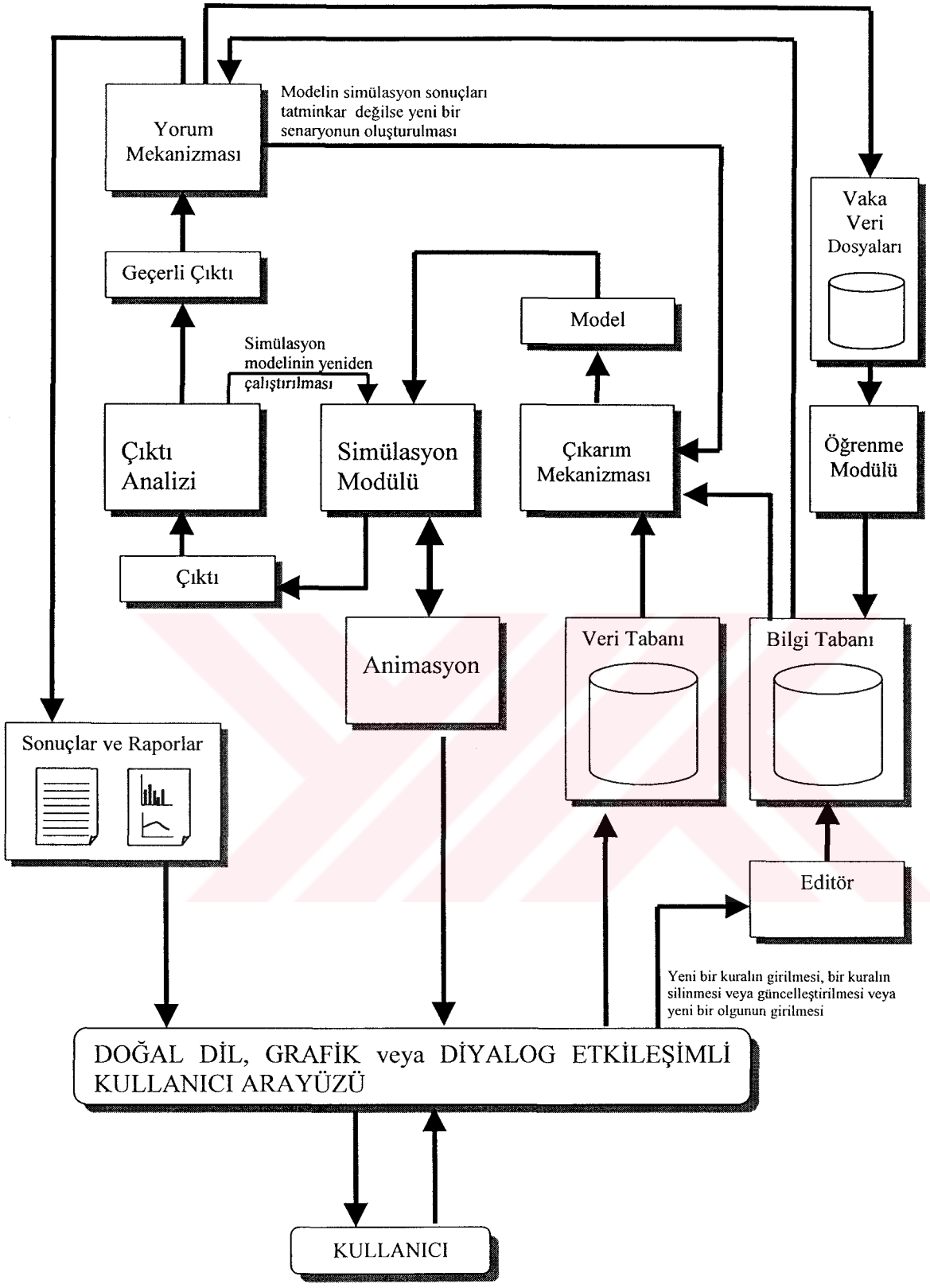
Bir simülasyon çevriminin bütün kısımları ideal bir yaklaşımda mevcut yapay zekâ ve uzman sistem teknikleriyle gerçekleştirebilir. Bu durumdaki bir yapay zekâ ile tamamen bütünleşik bir simülasyon ortamı yapısında olması gerekenler şu şekilde özetlenebilir:

1. Kullanıcının problemini en kolay şekilde aktarmasını sağlayan, kullanıcı ile etkileşimli bir arayüz. Bu arayüz kullanıcıdan verileri bir diyalog, doğal dil, menüler, ikonlar veya grafik etkileşimler ile kolaylıkla elde edebilecek bir yapıda olmalıdır. Ayrıca bu arayüz kullanıcının yeni bir kural veya bilgiyi bilgi tabanına ekleyecek, bilgi tabanında mevcut bir bilgiyi veya kuralı güncelleştirebilecek ve de bilgi tabanından güncelliğini kaybetmiş bir kuralı silebilecek bir editör ile etkileşimini sağlayacak nitelikte olmalıdır.
2. Kullanıcı tarafından bir arayüz ile girilen veriler, bir veri tabanına aktarılmalı ve daha sonra bir çıkarım mekanizması, kullanıcıdan elde edilen verileri ve sistemin bilgi tabanını kullanarak gerekli hesaplamalar ve karşılaştırmalar sonucu bir simülasyon modelini oluşturabilmelidir.
3. Düşünülen bu sistem, çıkarım mekanizması ile oluşturulan bu modeli çalıştırabilecek bir simülatör modülüne de sahip olmalıdır. Simülatör modülü bu simülasyon çıktısını elde eder.
4. Elde edilen simülasyon çıktısı ise uzman bir çıktı analiz modülü sayesinde analiz edilerek, eğer koşum uzunluğu ve elde edilen sonuçlar belirli şartları sağlıyorsa, bunu geçerli çıktı olarak sunar veya aynı modeli gerekli değişikliklerden sonra tekrar çalıştırılması için simülatör modülüne aktarır. Buradaki döngü geçerli bir çıktı elde edilene kadar sürer.
5. Bütünleşik sistemin animasyon modülü ise model simülatör tarafından çalıştırılırken aynı zamanda bu model görsel olarak da kullanıcıya sunulur. Böylece, simülasyon modeli çalışırken sistemdeki tüm oluşumlar görsel olarak da ortaya konulur.

6. Çıktı analiz modülü ile elde edilen geçerli çıktı, bir yorum mekanizması tarafından yorumlanır. Yorum mekanizması, sistemin bilgi tabanı ve veri tabanı ile etkileşim halinde çalışır. Yorum mekanizması ayrıca çıkarım mekanizması ile de etkileşim halindedir ve sonuçların durumuna göre otomatik olarak alternatif model ile ilgili verileri tekrar veri tabanına aktararak, bu yeni duruma göre sistemin tekrar işlemesini temin eder.
7. Yorum mekanizmasının işlemesi sonucunda elde edilen çıktı, raporlar, tablolar ve grafikler halinde kullanıcıya sunulur. Bütün bu bilgiler kullanıcıya bir arayüz desteği ile ulaştırılır.

Yukarıda tarafımdan önerilen ve açıklanan sistemin şematik bir görünümü Şekil 4.15'de görülmektedir. Bütün bunların dışında bir uzman veri analiz modülü de sisteme dahil edilebilir ve böylece kullanıcı, girdi verilerini de aynı ortam içinde analiz edebilir. Bu sağlandığında da yapay zekâ, bir simülasyon etüdünün tüm çevriminde kullanılmış olur. Ayrıca sisteme bir de öğrenme modülü ilave edildiğinde sistem tam anlamıyla yapay zekâ ile bütünleşik hale getirilmiş olur.

Açıklanan bu türden bir yapay zekâ ile tamamen bütünleşik bir simülasyon ortamına literatürde pek fazla rastlanmamakla birlikte açıklanan yapıların bir veya birkaçını üzerinde barındırabilen sistemler geliştirilmiş durumdadır ve birçok yerde bu veya buna benzer sistemlerin geliştirilmesi sürmektedir. Bunlarla ilgili çalışmalar 3. bölümün başında verilmiştir.

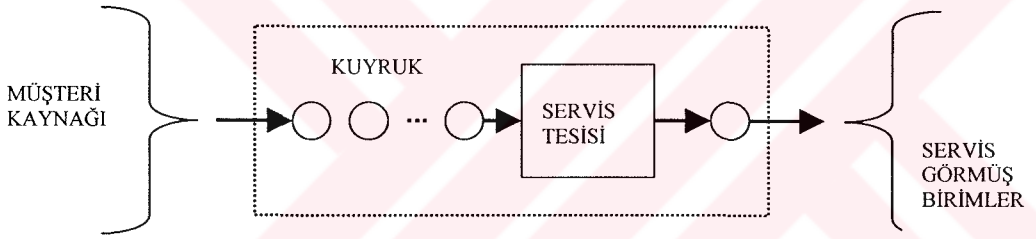


Şekil 4.15 Yapay zekâ ile tamamen bütünleşik bir simülasyon ortamı

5. BİR KAVŞAKTAKİ TRAFİK IŞIKLARININ OPTİMİZASYONU İÇİN ZEKİ BİR SİMÜLASYON ORTAMININ TASARLANMASI

Bir simülasyon çevriminin çeşitli kısımlarının, mevcut yapay zekâ ve uzman sistem teknikleri kullanılarak oluşturulabileceği daha önceki bölümlerde açıklanmıştır. Aynı teknikler kullanılarak, bir kavşaktaki trafik ışıklarının simülasyonu için yapay zekâ destekli bir simülasyon ortamı da oluşturulabilir.

Bir kavşaktaki trafik ışıklarının analizi, karşımıza yöneylem araştırması konularından biri olan bir "kuyruk problemi" olarak çıkmaktadır. Kuyruk sistemi ise bir bekleme hattını ve bir servis ortamını içeren bir sistemdir ve kuyruk teorisi, analizcilerin kuyruk sistemlerinin davranışlarını tanımlamalarına dayanan bir yönetim bilim dalıdır. Banks ve Carson (1984)'un açıkladığı ve Şekil 5.1'de görüldüğü gibi, bir kuyruk sistemi dört kısımdan oluşmaktadır. Bunlar potansiyel müşteri kaynağı (talep birimleri), kuyruk, servisçi ve servis görmüş talep birimleridir.



Şekil 5.1. Bir kuyruk sisteminin genel yapısı

Kuyruk problemlerine olan yaklaşımlar üç sınıfta toplanabilmektedir. İlk yaklaşımda sistem tasarımı, deneme-yanılma esasına göre olmaktadır. Bu yaklaşım basit sistemler için maliyeti etkileyici niteliktedir. Birkaç alternatif incelendikten sonra, optimum bir tasarım belirlenir. Ancak, incelenen sistem daha karmaşık olduğunda, deneme-yanılma prosedürüyle optimal bir tasarımın bulunması oldukça zorlaşmaktadır.

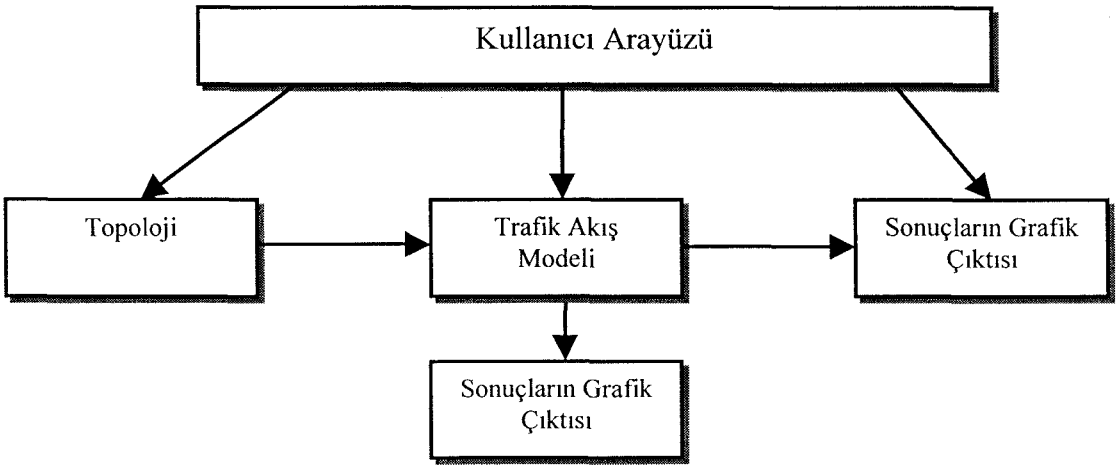
Diğer bir yaklaşım ise, analiz için analitik metotların kullanılmasını kapsamaktadır. Standart matematiksel ve istatistiksel metotlarla sistemin davranışını tanımlayan ifadeler geliştirilmiştir. Bu ifadeler, maliyet faktörleriyle birleştirildiğinde analist, sistemin tasarımı hakkında bir fikir sahibi olabilir. Bu yaklaşım mümkün olan her yerde kullanılabilir.

Ancak birçok kuyruk sisteminin analitik metotlarla analiz edilmesi oldukça zordur. Çünkü kuyruk sistemleri karmaşık bir yapı aldıkça bu yapılara uygun analitik modelleri oluşturmak güçleşmektedir. Bu durumda analist, nümerik bir analiz metodu olan simülasyon yaklaşımını kullanmalıdır. Bu bölümde açıklanan ise, bir kavşaktaki trafik ışıklarının optimizasyonu için zeki bir simülasyon ortamının ne şekilde tasarlanacağıdır.

Büyük şehirlerin en büyük problemlerinden biri olan trafik sorununa yönelik olarak yapılan, trafik simülasyonu ve trafiği modellemeyle ilgili çalışmalarında Hilliges ve diğerleri (1993) tarafından çok kaynaklı şebekelerde zamana bağlı trafik akışının dinamik bir modeli oluşturulmuştur. Modelin esas amacı, trafik şebekeleri için bir planlama elemanı elde edilmesi ve daha geniş olarak da ulaştırma modellerinin gerçek zamanlı simülasyonudur.

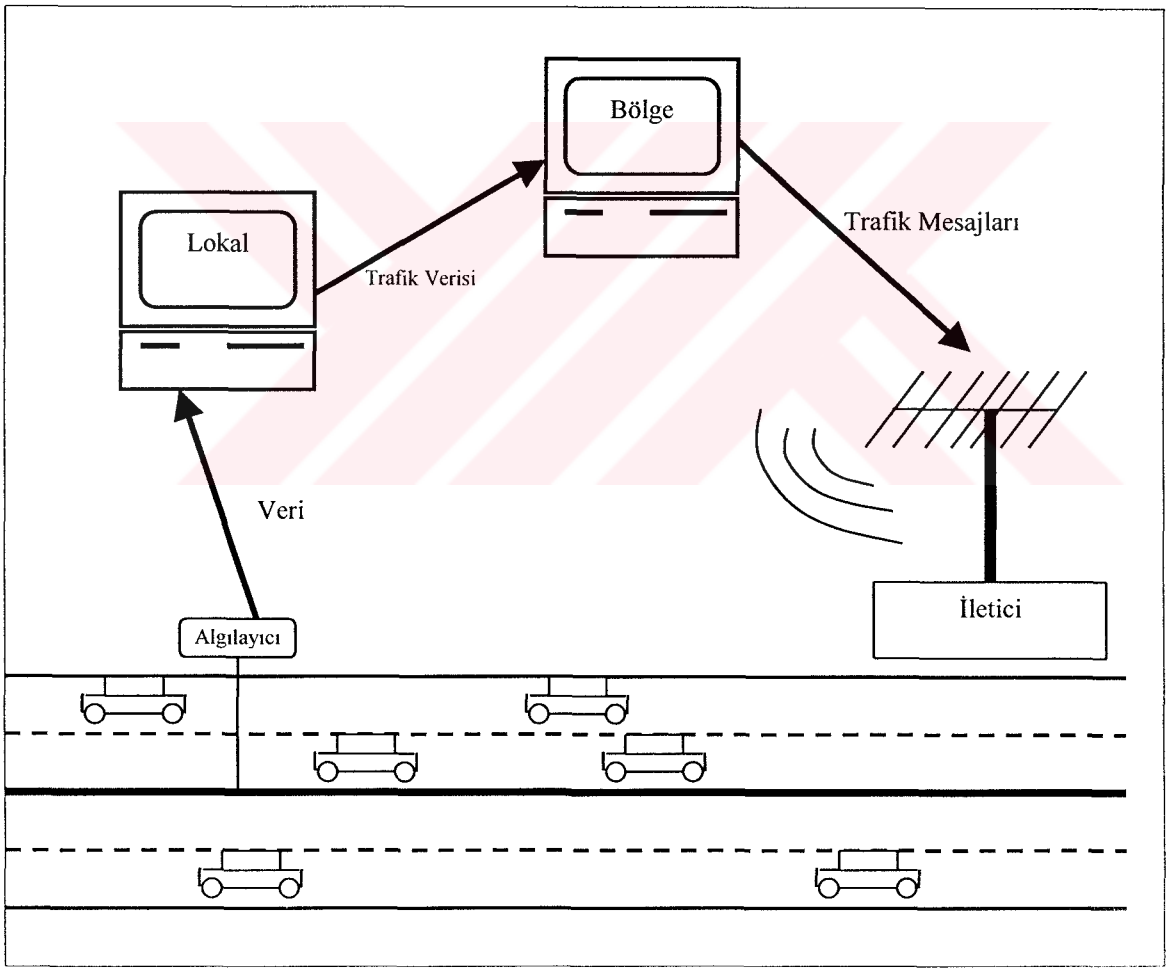
Gantz ve Mekenson (1990), yaptıkları çalışmalarında federal otoyol yönetiminin trafik simülasyon programı ile global, trafik sinyal optimizasyon programıyla trafik akışı modellerinin bir karşılaştırmasını ortaya koymuşlardır.

Cremer ve Merbrer (1993), büyük otoyol şebekelerindeki trafik akışı için, esnekliği oldukça yüksek bir simülasyon paketi geliştirmişlerdir. Bu programın esası, serbest akış şartlarından sıkışık durumlara kadar trafik akış dinamikleri için global bir model oluşturmaktır. Geliştirdikleri simülasyon programının genel yapısı, Şekil 5.2'de görüldüğü gibidir.

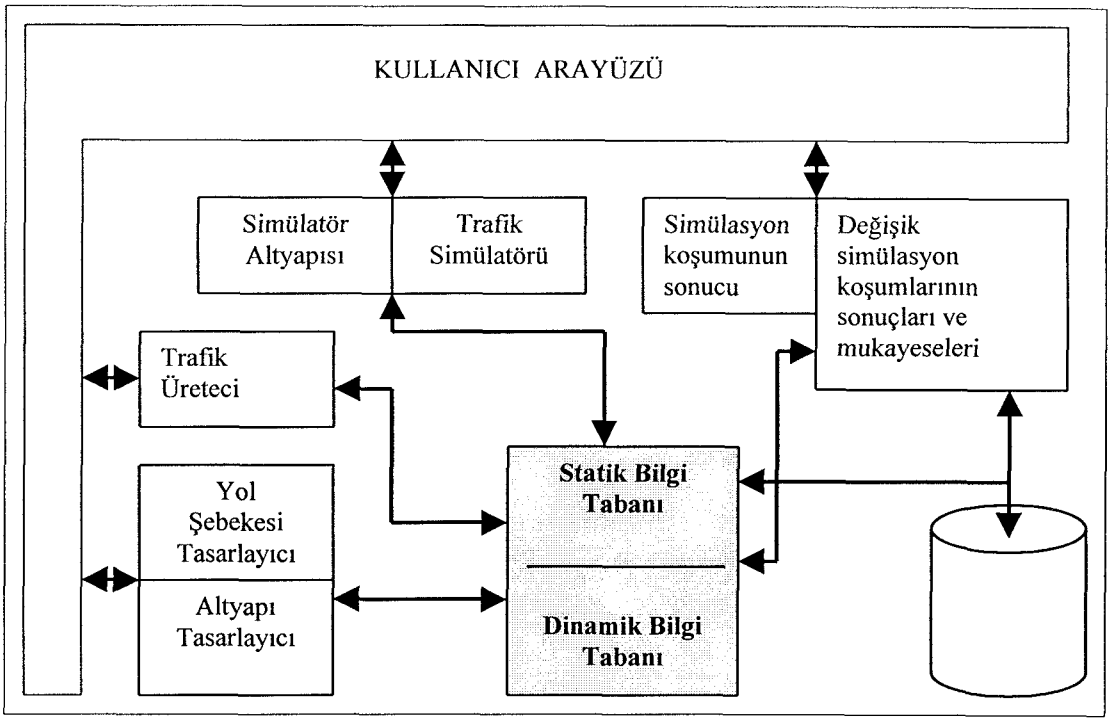


Şekil 5.2 Cremer ve Merbrer'in geliştirdikleri simülasyon paketinin blok yapısı

König ve Langbein (1992), PROROAD adı verilen bir proje çerçevesinde, bir bütünleşik trafik yönetim sistemine yönelik bilgi yapılarının simülasyonu için bir yapı geliştirmişlerdir. Onlar projenin ana hedefinin, farklı trafik yönetim stratejilerini ve farklı bilgi sistemi yapılarını değerlendirmek olduğunu belirtmişlerdir. Geliştirilen prototip sistemin yapay zekâ tekniklerinden de faydalandığı ve bir grafik etkileşimli kullanıcı arayüzüne sahip olduğu, yazarlar tarafından belirtilmiştir. PROROAD sisteminin iletişim yapısı Şekil 5.3'te görülmektedir. Bu sistemde yol şebekeleri trafik simülasyonuna baz teşkil etmektedir. Prototip sistemin yapısı da Şekil 5.4 görüldüğü gibidir. Burada herbir bileşen kendi başına bir modül gibi düşünülerek oluşturulmuştur (König ve Langbein, 1992).



Şekil 5.3 PROROAD sisteminin iletişim yapısı



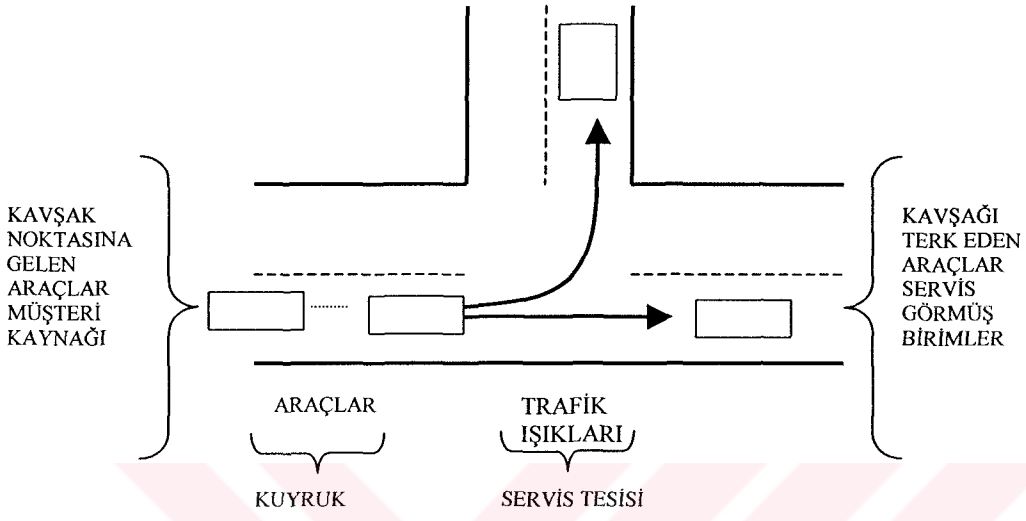
Şekil 5.4 PROROAD paketinin genel yapısı.

Liu (1997) tarafından yapılan bir çalışmada, Java programlama dili kullanılarak trafik akışı ve sinyalizasyonu simüle edilmiştir. Geliştirilen program demo niteliğinde olup, internet üzerinden çalışmaktadır. Bu çalışmada araçların, akış hacimlerinin ve trafik ışıklarının simülasyonu görsel olarak da kullanıcıya sunulmaktadır.

5.1. Bir Kuyruk Problemi Olarak Kavşak Sinyalizasyon Sistemi

Bir kavşaktaki sinyalizasyon sisteminin analizinin genel yapısı, bir kuyruk problemi olarak karşımıza çıkmaktadır. Burada sistem yol veya kavşaktır; talep birimleri olarak araçlar mevcuttur ve servisçiler olarak da trafik ışıkları sayılabilir. Bir kuyruk sisteminin elemanları olarak bir kavşağa baktığımızda, potansiyel müşteri kaynağı olarak kavşak noktasına gelen araçları görebiliriz, yani bekleyen araçlar kuyruk elemanını oluşturmaktadır; trafik ışıkları servis tesisi olmaktadır ve trafik ışıklarını geçip kavşağı terk eden araçlar da servis görmüş birimleri oluşturmaktadır. Bu açıklananlar Şekil 5.5'de şematik olarak görülmektedir.

Kavşak sistemindeki potansiyel müşteri kaynağı olarak gelen araçların kuyruk uzunluğu ise sonsuz olarak kabul edilebilmektedir. Araçlar bu kavşaktan gelecekte de geçmeye devam edeceklerdir. Zaten potansiyel müşteri kaynağının sonsuz sayılabildiği kuyruk sistemleri genelde analitik modellemeye daha yatkındır.

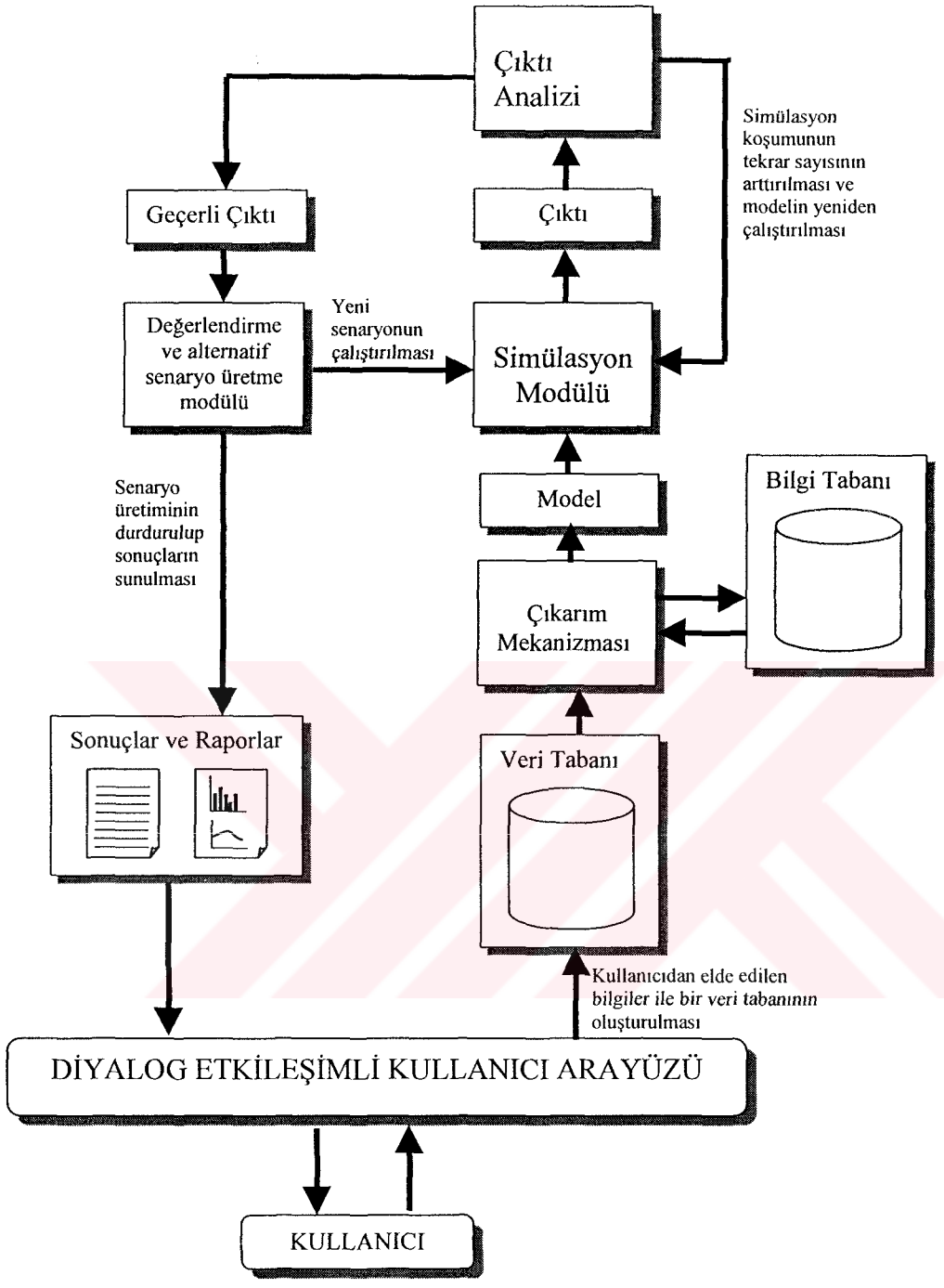


Şekil 5.5 Bir kuyruk modeli olarak kavşak yapısı

5.2.Trafik Simülasyonu İçin Yapay Zekâ Destekli Bir Ortamın Tasarlanması

Burada geliştirilen sistem bir diyalog arayüzü ile kullanıcıdan elde edilen verileri kullanarak bir problemin simülasyon ile çözümünde kullanıcıya tam destek sağlamaktadır. Kullanıcı bir kez verilerini sisteme tanımladıktan sonra, gerisini yapay zekâ destekli simülasyon ortamına bırakmaktadır. Geliştirilen ortam, problemleri girilen veri şartlarında çözerek sonuçları kullanıcıya sunmaktadır. Tasarlanan yapı içerisinde yer alan bileşenler bir kullanıcı arayüzü, çıkarım mekanizması, veri tabanı, bilgi tabanı, simülasyon modülü, çıktı analizi modülü ve senaryo geliştirme modülü olarak sayılabilirler. Sonuçlar kullanıcıya grafikler ve tablolar halinde sunulmaktadır. Tasarlanan sistemin genel yapısı Şekil 5.6'da görülmektedir.

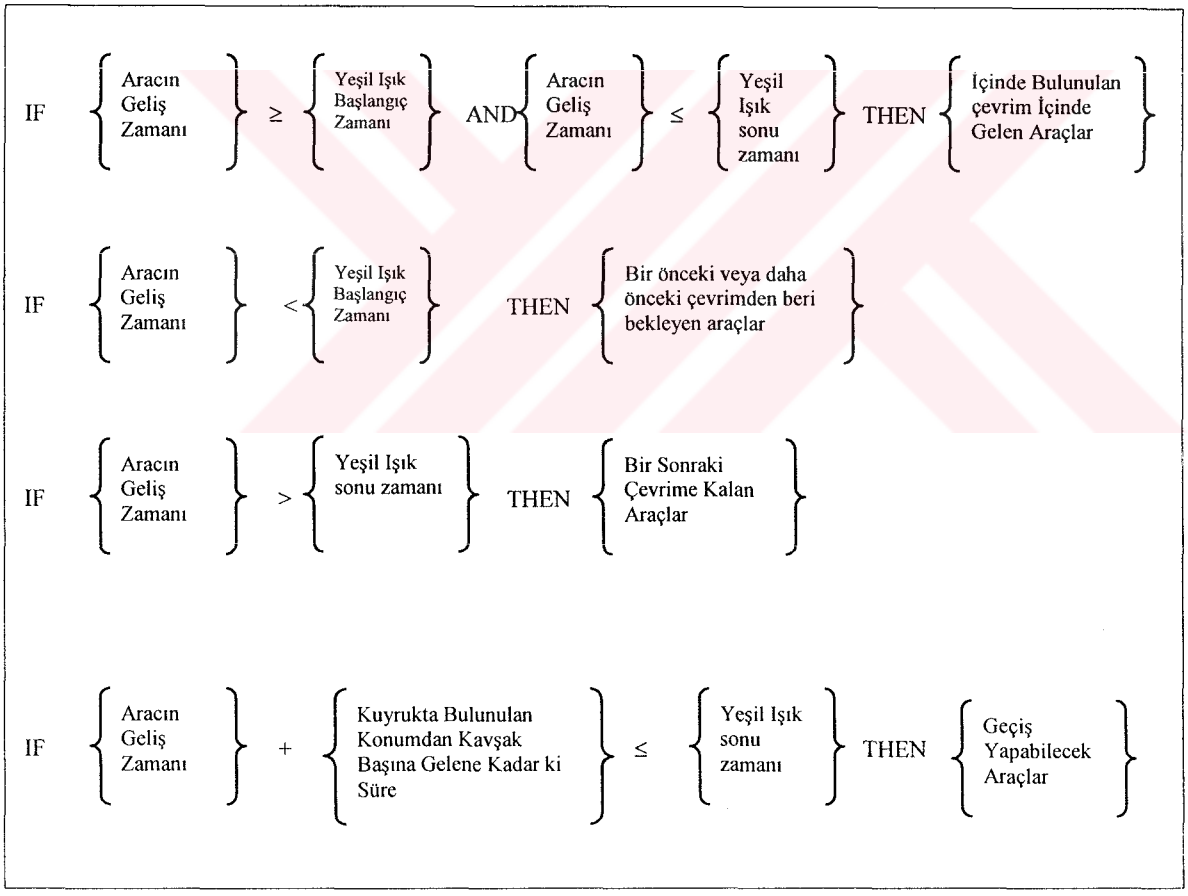
Burada diyalog arayüzü, kullanıcıdan verilerin elde edilmesine ve sonuçların kullanıcıya sunulmasına olanak verir. Diyalog arayüzü, bölüm 5.2.1'de ayrıntılı olarak açıklanmıştır.



Şekil 5.6 Tasarlanan sistemin genel yapısı

Çıkarım mekanizması ise, kullanıcıdan elde edilen verilerden oluşan veri tabanını ve sistemin bünyesinde yer alan bilgi yapılarını kullanarak, girilen şartlara uygun model yapısını sistemin kendi bünyesi içerisinde oluşturur. Girilmiş olan parametreler ile farklı

yapılardaki kavşak tipleri oluşturulur. Geliştirilen sistem bir makro model içermektedir. Bu makro model ise kullanıcı tarafından girilen verilere bağlı olarak bir çıkarım mekanizması tarafından istenilen yapıdaki kavşak modeli haline dönüştürülerek model oluşturulmaktadır. Böylece kavşak yapısını belirleyen unsurlara değerler atanarak kavşak yapısı şekillendirilir ve girilen kavşağın bir modeli kurulur. Oluşturulan kavşak modelinde kaç geliş yönü olduğu, kavşağın toplam kaç yoldan oluştuğu, her bir yolun şerit sayısı, hangi geliş yönlerinden hangi yönlere dönüleceği belirlenmiş olmaktadır. Bu şekilde de geliştirilen sistem tarafından kullanıcının istediği kavşak yapısı, simülasyonu yapılacak şekilde kurulmuş ve bir modeli ortaya çıkarılmış olur. Programın içerisine yerleştirilmiş olan if/then yapısı ile kavşak şartları elde edilir ve program bu koşullara göre işler. Şekil 5.7'de program yapısı içerisinde yer alan bazı koşul yapıları görülmektedir.



Şekil 5.7 Bazı kural yapıları

Çıktı analizi modülü ise, Welch'in (1983) geliştirdiği ve Bölüm 2.11'de açıklanan algoritmaya göre çalışmaktadır. Her simülasyon etüdü 5 kere tekrar edilmektedir. Analizde ortalama sistem süresi baz olarak alınmakta ve bu beş tekrarın ortalaması bulunarak ortaya çıkan eğri algoritma ile düzeltilip, sabit durum safhasına geçilen nokta belirlenip, ortalamalar baş taraf atılarak (geçiş safhası) hesap edilmekte ve böylece geçiş safhasının ortalamalar üzerindeki etkisi ortadan kaldırılmış olmaktadır. Bu ise simülasyon sonuçlarının daha sağlıklı ve güvenilir olmasını sağlar.

5.2.1. Bir diyalog arayüzünün tasarlanması

Bir model, ancak eksiksiz bir şekilde tanımlanmış veriler yardımıyla kurulabilir. Dolayısıyla sistem, kullanıcıdan gerekli olan tüm verileri elde etmelidir. Bir modelin kurulması için gerekli olan problem spesifikasyonunun, yani bir problemin eksiksiz ve ayrıntılı tanımından oluşan verilerin elde edilmesi amacıyla menülerden ve sorulardan oluşan bir diyaloğun geliştirilmesi gerekmektedir. Burada geliştirilen diyaloğun, simülasyon modeli için bir kavşak sisteminin tüm karakteristiklerini kullanıcıdan elde edebilecek nitelikte olması zorunludur. Gerekli olan tüm veriler kullanıcıdan elde edildikten sonra, ancak bir simülasyon modeli oluşturulup, çeşitli deneysel tasarımlar altında çalıştırılabilir.

Çeşitli kavşak yapılarının oluşturulup,trafik ışıklarının simülasyonu için gerekli tüm verilerin elde edilmesi amacıyla tasarlanan diyalog, kullanıcıdan öğrenilecek veriler açısından aşağıda görüldüğü gibidir:

- Simülasyonu yapılacak olan kavşağın adı (bilgi amaçlı)
- Kavşaktaki yol sayısı
- Kavşaktaki her yolun şerit sayısının ayrı ayrı kullanıcıya sorulması
- Kavşaktaki her yol için araçlarının geliş dağılımlarının kullanıcıdan elde edilmesi. Bunun için de bir menü halinde aşağıdaki soruların kullanıcıya sorulması gerekir:

Kavşaktaki I. yol için gelişlerin yapısı

[1] Poisson Dağılımı

[2] Üssel Dağılım

[3] Normal Dağılım

[4] Düzgün Dağılım

[5] Sabit Değer

I.yol için gelişlerin yapısını giriniz :

Eğer girilen değer 1 ise,

Poisson Dağılımına ait 1 dakikada gelen araç sayısı nedir?

Eğer girilen değer 2 ise,

Üssel dağılıma ait gelişler arası süre nedir?

Eğer girilen değer 3 ise,

Normal dağılımın ortalama değeri nedir?

Normal dağılımın standart sapması nedir?

Eğer girilen değer 4 ise,

Düzgün dağılımın minimum değeri nedir?

Düzgün dağılımın maksimum değeri nedir?

Eğer girilen değer 5 ise,sabit değer nedir?

- Araçların bulundukları yoldan, hangi yola gidebilecekleri
- Kavşaktaki yolların herbirinden, araçların hangi olasılıkla hangi yola saptıkları
- Bir aracın sapacağı yola bağlı olarak kavşağı boşaltma süresi (Bu, sabit bir değer kabul edilebileceği gibi, program yapısında yapılabilecek bir modifikasyon ile çeşitli dağılımlara uygun olarak da girilebilir hale gelebilir. Bu durumda araç gelişleri için kullanılan menü ve sorular burada da kullanılır)
- Bir aracın bulunduğu konuma bağlı olarak, kavşak girişine kadar geçecek olan süreyi belirlemek üzere, bir aracın bir araçlık mesafeyi katetme süresi

- Bekleyen araçların yakıt harcamalarını belirlemek üzere, bir araç beklerken onun saniyedeki yakıt harcaması
- Simülasyon koşum süresi

Bazı kavşaklarda trafik ışıklarının simülasyonu için geliştirilen bu diyalog ile, kullanıcıdan simülasyon için gerekli olan tüm veriler otomatik olarak elde edilerek problemin bir spesifikasyonunun oluşturulması sağlanır.

5.2.2. Bir kavşağın simülasyonu için model tasarımı

Bir kavşak yapısı incelendiğinde, farklı şerit sayısına sahip yollar, trafik ışıkları, kavşak noktasına gelen araçlar, araçların bir kavşak noktasındaki farklı davranışları ve trafik ışıkları önünde oluşan araç kuyrukları göze çarpmaktadır. Gerçek sistemi temsil etmek üzere tasarlanacak modelin, tüm bu sayılan unsurları içermesi gerekmektedir.

Öncelikle kavşağın yapısının bir modeli oluşturulmalıdır. Yani kavşağın kaç geliş yönüne sahip olduğu, kavşakta birleşen toplam yol sayısı ve herbir yolun içerdiği şerit sayısı tanımlanmalıdır.

Bir trafik sisteminde ışıkların durumuna bakıldığında, bir yola verilen geçiş hakkı bitip bu yol için kırmızı ışık yandığında, daha önceden kırmızı ışıkları yanan yollardaki ışıkların anında yeşile dönmediği görülür. Bunun nedeni ise, geçiş hakkına sahip bir yola kırmızı yandığında daha önce yeşil yanarken kavşağa girmiş ve henüz kavşağı terk etmemiş araçlar olabilmektedir. İşte böylece bütün yönlere kırmızı ışık yanarken, yeşil ışıkta kavşağa girmiş olan araçların bu esnada kavşağı boşaltmaları temin edilmiş ve trafik tehlikeye sokulmamış olur. Şekil 5.8 ve Şekil 5.9'da görülen üçlü kavşak incelendiğinde, üçü geliş yönü olmak üzere ve geliş ve gidiş yönleri ayrı sayılmak koşulu ile toplam beş yol ve de beş şerit vardır. Işıkların durumuna bakıldığında 1. ve 2. yola yeşil ışık yanmakta, 1. yönden gelen araçlar 5 istikametine ve 2. yönden gelen araçlar ise 4 istikametine gitmektedirler. Şekil 5.8'de durum 2'ye bakıldığında ise, artık 1. ve 2. yön için kırmızı ışık yanmıştır. Buna rağmen kırmızı ışıkta bekleyen 3 yönüne hemen yeşil ışık yanmamaktadır.

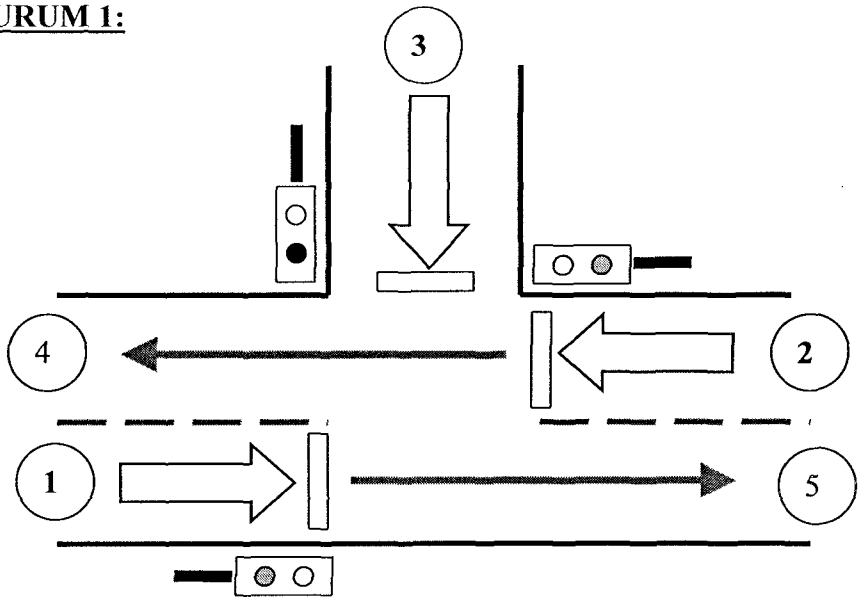
1 ve 2 yönünden gelen ve kavşakta kalan araçların bu esnada kavşak geçişlerini tamamlamaları sağlanır. Burada emniyet süresi olarak 1 ve 2 yönünden gelen araçların kavşak boşaltma süresi (kavşak geçiş süresi) olan bir değer kadar tüm yönlere kırmızı ışık yanar. Aynı durum Şekil 5.9' da da geçerlidir. Görüldüğü gibi bu yapıdaki bir kavşakta trafik ışıkları ve araç geçişleri için dört durum söz konusudur.

Trafik lambaları kırmızı, sarı ve yeşil ışık olmak üzere üç farklı renk ışık içermesine rağmen ışıklarla ilgili olarak yapılan çalışmalarda sadece kırmızı ve yeşil ışık dikkate alınmaktadır. Bunun nedeni ise, sarı ışık bir uyarı niteliğinde olup yeşil ışıktan sonra ve kısa süreli olarak yanmakta ve de kısa bir zaman sonra kırmızı ışığın yanacağını sürücüye bildirmektedir. Sarı ışıkta da yeşil ışıkta olduğu gibi geçiş yapılabilmektedir. Dolayısıyla sarı ışık süresi yeşil ışık süresi içinden, yeşil ışık süresinin belirli bir miktarı olarak planlanabilmektedir. Ayrıca birçok ülkede sarı ışık kullanılmamaktadır. Bu durumda yeşil ışığın süresinin sona ermek üzere olduğu sürücüye yeşil ışığın sonlara doğru yanıp sönmesiyle bildirilir ve böylece sürücüye kısa bir zaman sonra kırmızının yanacağı bildirilmiş olur. Bütün bu sebeplerden dolayı da yapılan çalışmalarda sadece kırmızı ve yeşil ışık sürelerini belirlemek yeterli olmaktadır.

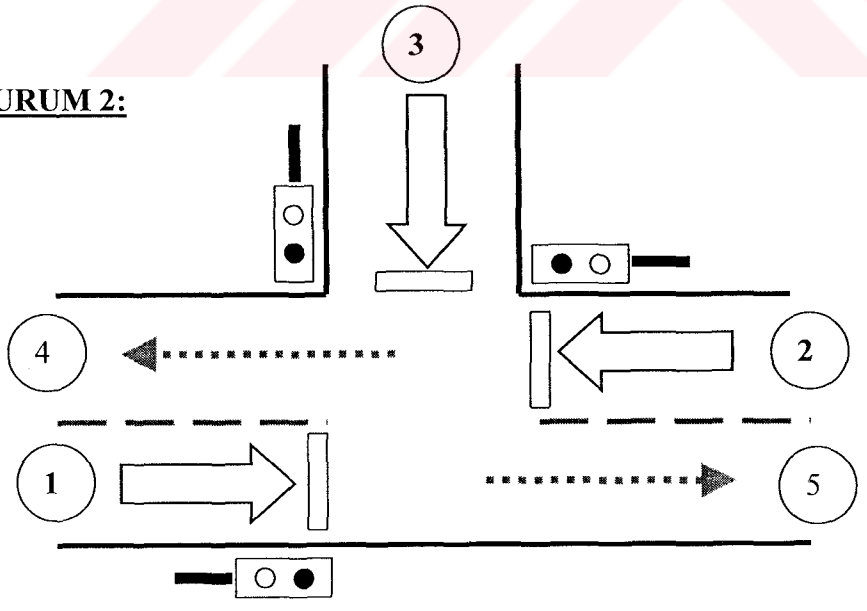
Araçların kırmızı ışıkta beklemeleri bir çevrim süresi boyuncadır. Çevrim süresi, geçiş hakkı verilmiş olan bir yöne tekrar bir geçiş hakkı verilene kadar geçen süredir. Bu süre esnasında kavşaktaki tüm geliş yönlerine bir kere geçiş hakkı verilir.

Kavşaklardaki yaya geçişleri kavşağın genel performansını azaltan bir unsurdur. Yaya geçişleri bazı durumlarda sistemi hiç etkileyemeyecek şekilde veya en az etkileyecek şekilde planlanmalıdır. Bunun her ikisi de mümkün değilse bu durumda tüm çevrim süresi içinden belirli bir süre de yaya geçişi için planlanmakta ve bu belirlenen süre boyunca araç geçişi durmaktadır. Bu şekildeki bir planlama kavşağın genel performansını oldukça etkilemektedir. Yaya geçiş etkisini azaltmak için ise kademeli geçişler planlanmakta ve araç geliş yönleri için kırmızı ışık yandığında yayalar yolun yarısını geçmekte ve daha sonra da diğer yarısını geçmektedir. Böylece yaya geçiş etkisi azaltılmakta ve bazı durumlarda ise tamamen ortadan kaldırılmaktadır.

DURUM 1:

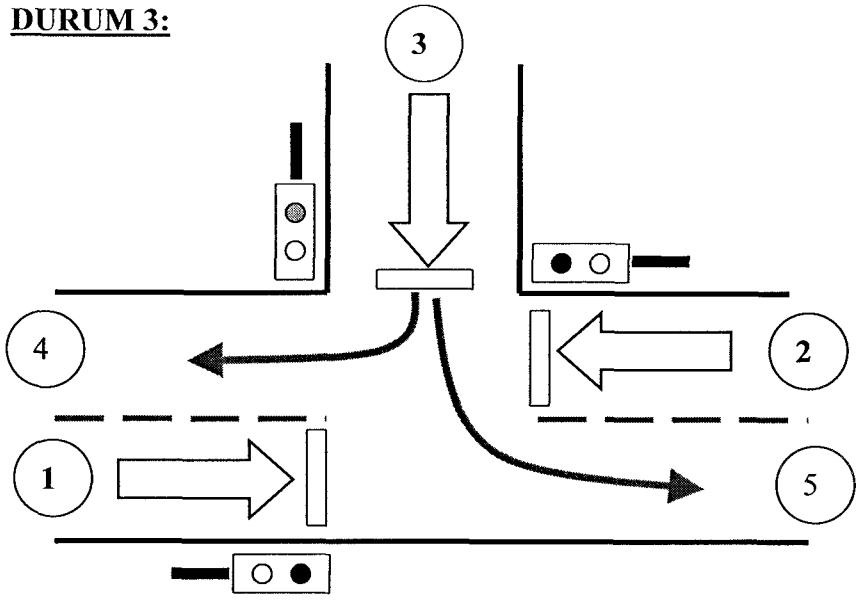


DURUM 2:

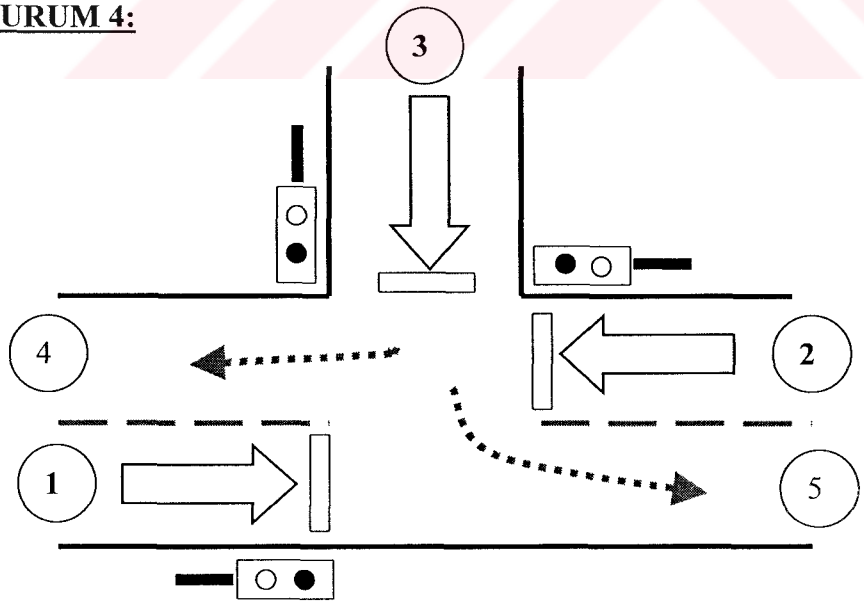


Şekil 5.8 Bir ve iki nolu geliş yönü geçiş hakkına sahip iken kavşağın durumu

DURUM 3:



DURUM 4:



Şekil 5.9 Üç nolu geliş yönü geçiş hakkına sahip iken kavşağın durumu

Bir gün boyunca trafik akışı ve yoğunluğu incelendiğinde, günün farklı saatlerinde trafik akışının ve yoğunluğunun oldukça değişiklik gösterdiği görülür. Özellikle iş başlangıç saatlerinde ve iş çıkışlarında trafik oldukça yoğundur. Planlamalarda bunların dikkate alınması gerekmektedir. Belirli zaman periyotlarındaki trafik yoğunluk ölçümlerinin sonuçlarına göre trafik ışıklarının da bu periyotlarda değişecek şekilde çalıştırılması trafik akışını daha düzenli bir hale getirebilir. Bu ise hem yazılım ve hem de donanımın geliştirilmesi ile gerçekleştirilebilir.

Ayrıca video kamera ile kavşakların kontrolü gerçekleştirilmektedir. Elde edilen görüntüler analiz edilerek kırımızı ışıkta bekleyen araçların adedi bu görüntülerin analizi ile otomatik olarak elde edilebilmekte ve bu güncel verilere göre de ışık ayarlamaları yapılabilmektedir. Bu da ancak geliştirilen özel yazılım ve donanımlarla gerçekleştirilmektedir. Bölüm 5'de kısaca açıklanan PROROAD sistemi de güncel verilerle çalışacak bir şekilde geliştirilmiştir (König ve Langbein, 1992).

Tasarlanan simülasyon modelinin amaçlarını kısaca şu şekilde özetleyebiliriz:

- Belirli bir zaman biriminde kavşaktan geçen araç sayısının maksimizasyonu,
- Kuyrukta bekleyen araç sayısının minimizasyonu
- Araçların ortalama bekleme süresinin minimizasyonu,
- Bekleme süresi boyunca harcanan yakıtın minimizasyonu.

Modelin amaçlarına bakıldığında, problemin çok amaçlı görülmesine karşın birinci sıradaki amacın, yani belirli bir zaman biriminde kavşaktan geçiş yapabilen araç sayısının maksimizasyonu ile diğer tüm amaçların da optimize edilmesi temin edilmiş olur. Trafik mühendisliğinde yapılan çalışmalarda da bir kavşağın başarısı, belirli bir zamanda geçen araç sayısına bağlıdır.

Yukarıda sayılan tüm amaçlar aslında birbirleriyle doğrusal olarak ilişkilidirler ve birinin sağlanması diğer tüm amaçları da sağlar. Bir kavşaktan belirli bir zaman biriminde geçen araç sayısının maksimize edilmesiyle kuyrukta bekleyen araç sayısının en aza inmesi,

bekleme süresinin kısalması ve dolayısıyla da yakıt harcamasının minimizasyonu da temin edilmiş olur.

Simülasyon modelinin, kavşak yapısına bağlı olan ve simülasyon çalışması esnasında değiştirilemeyecek olan bazı kısıtları vardır. Modelin sahip olduğu bu kısıtlar şunlardır:

- Kavşaktaki yol sayısı
- Kavşaktaki herbir yoldaki şerit sayısı
- Çevrim süresi
- Yeşil ışığın yanma süresi (çevrim süresi ve belirli bir zaman periyodunda kavşağa gelen araç sayısına bağlı olarak)
- Kırmızı ışığın yanma süresi
- Kavşak boşaltma süresi
- Bir araçlık mesafeyi katetme süresi (Bu parametre hız formülü ile tanımlandığında değişken bir yapı kazanır).

Simülasyon modelinin çalıştırılması sonucu elde edilen değerler kullanılarak aşağıda özetlenmiş olan bilgiler hesaplanabilir:

- Belirli bir zaman diliminde kavşaktan geçen araç sayısı,
- Kuyrukta bekleyen ortalama araç sayısı,
- Bekleyen araçlar için ortalama yakıt sarfiyatı,
- Ortalama bekleme süresi,
- Kavşağa gelen bir aracın beklemek zorunda kalma ihtimali,
- Bir araç için ortalama sistem süresi,
- Kavşak verimi.

Bu bilgileri 5.1'den - 5.6'ya kadar olan formüller yardımıyla hesaplayabiliriz.

$$\text{Kuyrukta Bekleyen Ortalama Araç Sayısı} = \frac{\text{Kuyrukta Bekleyen Araç Sayısı}}{\text{Çevrim Adedi}} \qquad (5.1)$$

$$\text{Araçların Sistemde Harcadığı Ortalama Süre} = \frac{\text{Araçların Sistemde Harcadığı Toplam Süre}}{\text{Toplam Geçen Araç Sayısı}} \quad (5.2)$$

$$\text{Gelen Bir Aracın Beklemek Zorunda Kalma Olasılığı} = \frac{\text{Toplam Bekleyen Araç Sayısı}}{\text{Toplam Geçen Araç Sayısı}} \quad (5.3)$$

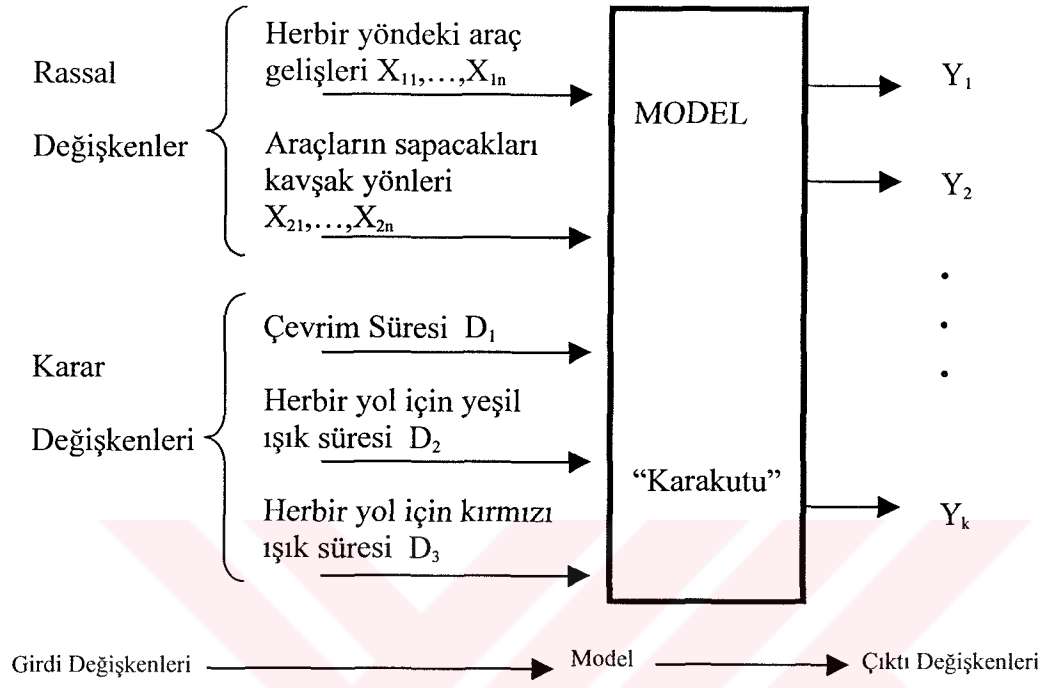
$$\text{Kavşak Verimi} = \frac{\text{Kavşaktan Geçiş Yapan Toplam Araç Sayısı}}{\text{Kavşağa Gelen Toplam Araç Sayısı}} \quad (5.4)$$

$$\text{Bekleyen Araçlar İçin Ortalama Yakıt Sarfiyatı} = \frac{\text{Beklerken Harcanan Yakıt Sarfiyatı}}{\text{Toplam Bekleyen Araç Sayısı}} \quad (5.5)$$

$$\text{Ortalama Bekleme Süresi} = \frac{\text{Toplam Bekleme Süresi}}{\text{Kavşaktan Geçen Toplam Araç Sayısı}} \quad (5.6)$$

Yukarıda hesaplanabilecek olan değerler, herbir senaryo için farklı olmaktadır. Simülasyon ile bizim asıl amacımız en iyi çevrim süresini, her yol için yeşil ışık ve kırmızı ışık sürelerini hesaplamaktır. Bu amaçla, çok sayıda senaryo geliştirilip farklı çevrim ve ışık süreleri altında kavşağın performansına bakılarak en iyi çevrim ve ışık sürelerine karar verilir. Bu ise oldukça zaman alıcı bir işlemdir. Çünkü her simülasyon etüdü sonunda, yeni bir alternatif senaryo oluşturularak analiz edilir ve bunun sonunda optimum sonucu veren senaryodaki çevrim ve ışık süreleri problemin çözümü olarak kabul edilir. İşte böylece, simülasyonun bünyesine yapay zekâ ve uzman sistem teknikleri yerleştirilerek, bu kısmın otomatik olarak bir uzman sistem tarafından yönetilmesi ve sonuçta en iyi senaryonun bulunarak kullanıcıya sunulması sağlanabilir.

Geliştirilen simülasyon modelinde girdi değişkenlerinin, bir dönüştürücü olarak "karakutu" şeklinde ifade edebileceğimiz model tarafından çıktı değişkenleri haline dönüşümü Şekil 5.10'da görülmektedir.



Şekil 5.10 Model girdi-çıktı dönüşümü

Burada, kontrol edilemeyen girdi değişkenleri Xij, karar değişkenleri Di ve çıktı değişkenleri de Yi'dir. Bu dönüşüm aynı zamanda bağıntı 5.7'de görüldüğü gibi bir fonksiyon olarak da ifade edilebilir:

$$f(X,D) = Y \tag{5.7}$$

Modelde rassal değişkenler olan Xij'lere herhangi bir müdahalede bulunulmadığına göre, bu fonksiyonun optimizasyonu ancak karar değişkeni olan Di'lerin farklı değerleri için modeli icra ederek sağlanabilir. Bu durumda farklı Di değerleriyle farklı senaryolar oluşturulabilir ve daha önce de belirtildiği gibi, çıktı değişkeni olan Yi'ler içinden

simülasyon süresi boyunca kavşaktan geçen araç sayısının en yüksek olduğu durumdaki senaryo, problemin en iyi çözümü olarak kabul edilebilir. Bu durumu sembolik olarak, bir döngü içinde yer alan 5.8'deki fonksiyon ile de ifade edebiliriz. Bu senaryolar içinden maksimizasyon veya minimizasyon yapılmasına bağlı olarak optimal senaryonun seçimi, 5.9'daki ifade ile sembolik olarak gösterilmiştir.

$$for \quad i := 1 \quad to \quad n \quad do \qquad (5.8)$$

$$f(X,D_i)=Y_i;$$

$$optimal \quad senaryo = \max/\min\{Y_1, Y_2, ..., Y_n\} \qquad (5.9)$$

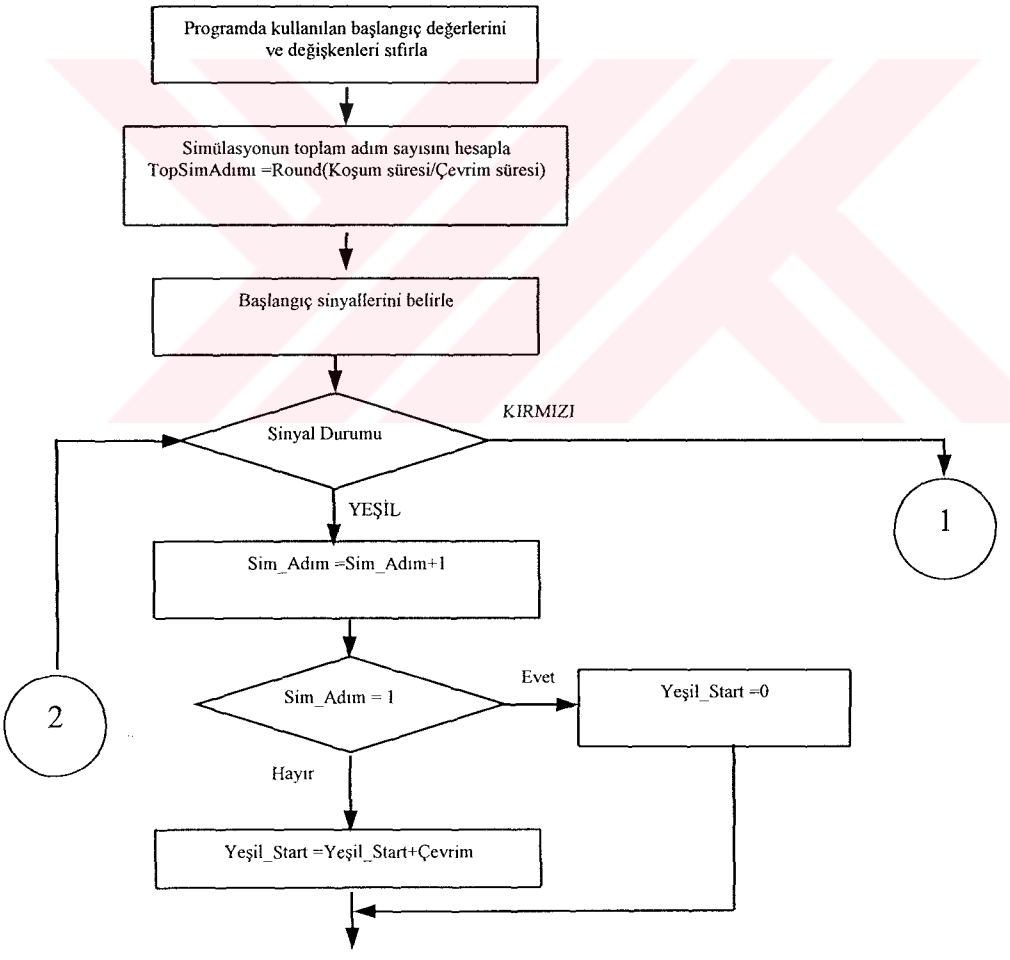
Senaryolar arasındaki fark ε gibi son derece küçük bir değer olduğunda, tam olarak optimali yakalama olasılığı çok yüksek olur. Bu durumda, oluşturulacak senaryo sayısının ise çok fazla olması gerekir. Fakat trafik kavşağı probleminde senaryo sayısı sonsuz değildir. Bir kavşak yapısında çevrim süresinin ve herbir yola verilen yeşil ve kırmızı ışık sürelerinin bir sınırı vardır. Bu sınır da, optimizasyon için bir üst sınır oluşturarak senaryo sayısının sonlu olmasına neden olur.

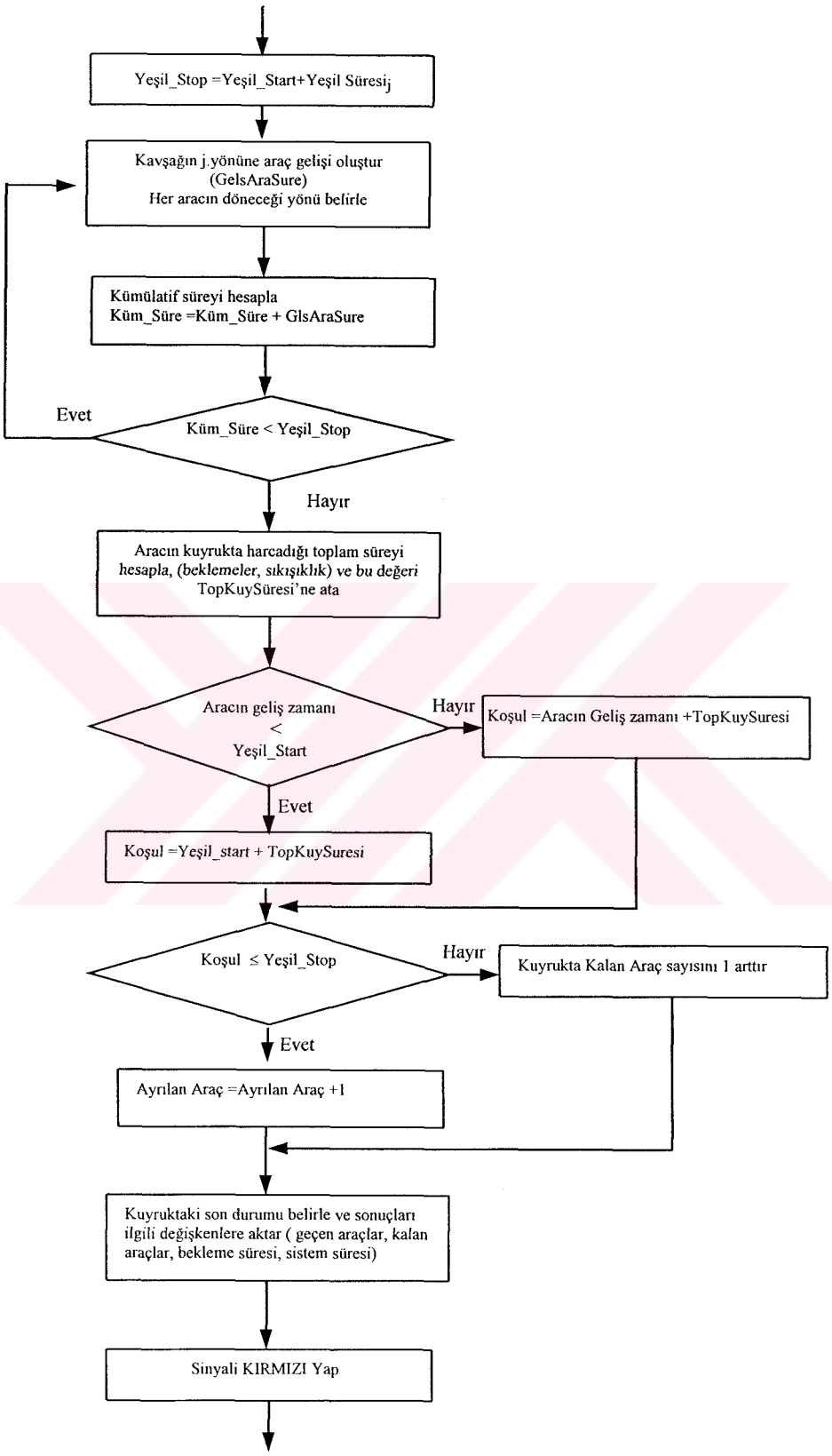
Kavşak simülasyonu için geliştirilen algoritmanın genel yapısı Şekil 5.11'de görüldüğü gibidir. Düşünülen yaklaşımda, her yeşil ışık sonunda, kuyrukta bekleyen ve içinde bulunulan çevrimde gelen araçların kavşaktan geçiş işlemleri yapılır. Diğer zamanlarda da her yön için kavşağa gelişler oluşturulur. Simülasyon sırasında simülasyon adımı olarak çevrim süresi göz önüne alınır; ayrıca gelişleri oluşturmak için her geliş yönü için gelişler

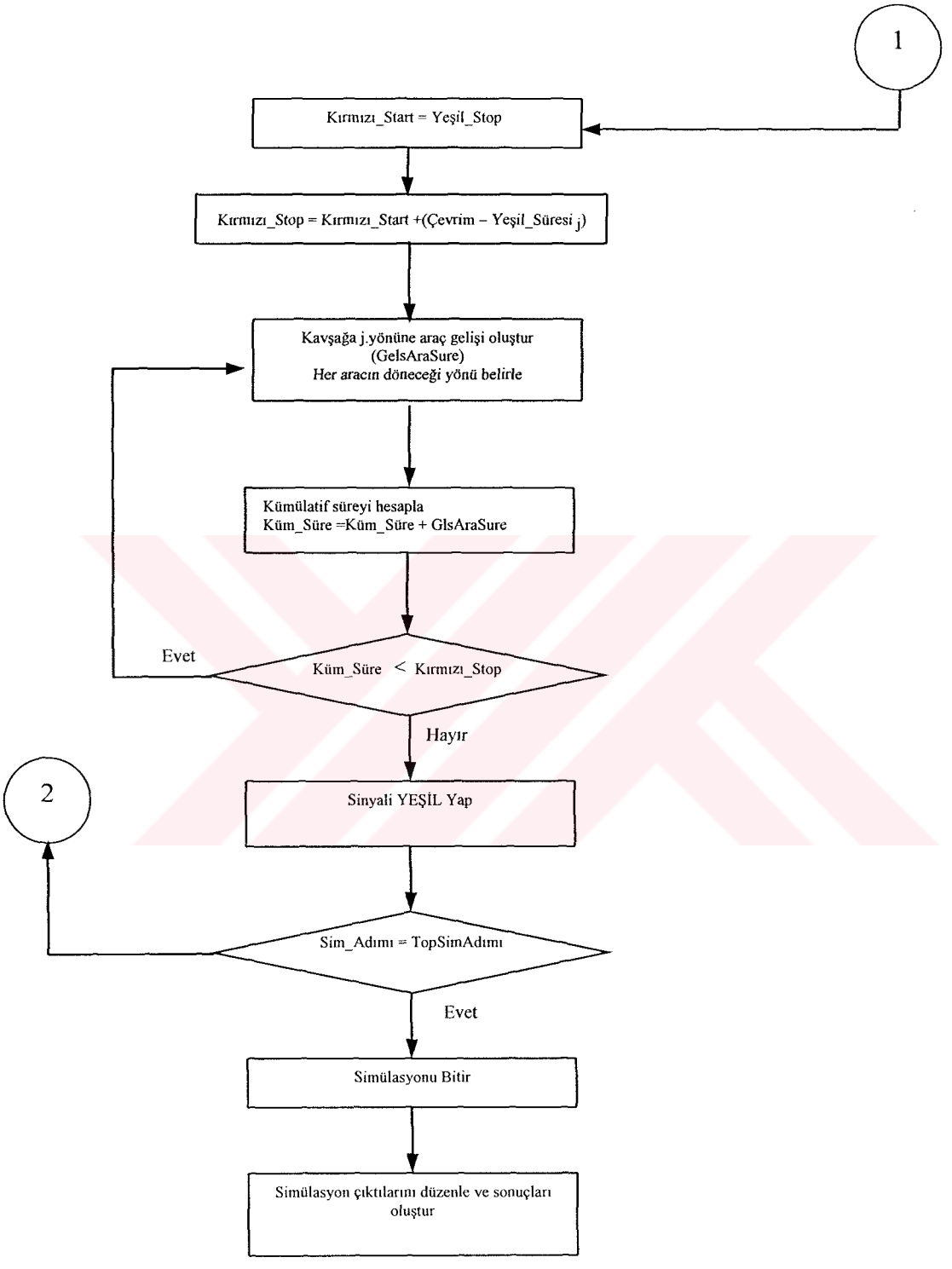
arası süre ve bu sürelerin kümülatifi belirlenerek, zaman ayarlaması sağlanır. Simülasyonun bir kere koşumundaki adım sayısı ise bağıntı 5.10 ile belirlenmektedir.

$$\text{Toplam Simülasyon Adımı} = \text{Koşum Süresi} / \text{Çevrim Süresi} \qquad (5.10)$$

Bu şekilde, simülasyon modeli gerçek bir kavşak yapısına uygun olarak ve aşağıda bir geliş yönü için verilen genel bir algoritma ile çalışır ama tüm bunlar aynı anda bütün geliş yönlerine ve o yön için geçerli olan sinyale ve duruma uygun olarak senkronize olarak işler.




Yeşil_Stop =Yeşil_Start+Yeşil Süresi$_j$
Kavşağın j.yönüne araç gelişi oluştur (GelsAraSure)
Her aracın döneceği yönü belirle
Kümülatif süreyi hesapla
Küm_Süre =Küm_Süre + GlsAraSure
Evet
Küm_Süre < Yeşil_Stop
Hayır
Aracın kuyrukta harcadığı toplam süreyi hesapla, (beklemeler, sıkışıklık) ve bu değeri TopKuySüresi'ne ata
Aracın geliş zamanı < Yeşil_Start
Hayır
Koşul =Aracın Geliş zamanı +TopKuySuresi
Evet
Koşul =Yeşil_start + TopKuySuresi
Koşul ≤ Yeşil_Stop
Hayır
Kuyrukta Kalan Araç sayısını 1 arttır
Evet
Ayrılan Araç =Ayrılan Araç +1
Kuyruktaki son durumu belirle ve sonuçları ilgili değişkenlere aktar (geçen araçlar, kalan araçlar, bekleme süresi, sistem süresi)
Sinyali KIRMIZI Yap



Şekil 5.11 Bir kavşağın simülasyonu için geliştirilen algoritmanın genel yapısı

5.2.3. Simülasyon modellerindeki varsayımlar

Bir simülasyon modeli, çeşitli varsayımları içermektedir. Bu varsayımlar, bir olayın daha kolay modellenebilmesini sağlamaktadırlar. Trafik ışıklarının simülasyon modellerinde kabul edilen varsayımlar aşağıda verildiği gibidir:

- Kavşak noktasına gelen araçlar için gelişler arası süre Poisson , üssel, normal ve düzgün dağılımlardan herhangi biri tarafından tanımlanabilmektedir veya sabit bir değer olmaktadır.

- Herbir yoldaki trafik ışıklarının önündeki araç kuyrukları için sonsuz yeterlilikte bir alan olduğu kabul edilmektedir. Yani araç kuyruk uzunlukları sınırsızdır.

- Her yolda araç geçiş kuralı olarak ilk gelen ilk servis görür (FIFO) kuralı uygulanmaktadır. Bu kural, gerçekte kavşak yapısının doğası gereği araçlar yol boyunca arka arkaya sıralı durduğundan zorunlu olarak ortaya çıkmaktadır.

- Araçların kavşağı boşaltma süresi, aracın sapacağı yola bağlı olarak sabit bir değer olmaktadır.

- Araçların bir araçlık mesafeyi katetme süreleri, sabit bir değer olarak kabul edilmektedir.

- Araçların beklerken harcadıkları yakıt miktarı, araçların farklılıklarına bakılmaksızın ortalama bir değer ile sabit olarak kabul edilmektedir.

- Sürücülerin ışığı algılama süresi dikkate alınmamaktadır.

- Araçlarda ve trafik ışıklarında herhangi bir arıza meydana gelmediği kabul edilmektedir.

Geliştirilen sistem bu varsayımları göz önüne alarak bir simülasyon modeli oluşturmaktadır.

5.2.4. Başlangıç senaryosu ve alternatif senaryolar oluşturabilmek için sezgisel bir yaklaşım

Geliştirilen modelde daha önce de belirtildiği gibi karar değişkenleri, çevrim süresi, herbir yol için yeşil ışık ve kırmızı ışık süreleri olarak belirlenmiştir. Bu karar değişkenleri altında optimize edilmeye çalışılan çıktı değişkeni de belirli bir zaman diliminde (simülasyon süresinde) kavşaktan geçiş yapabilen araç sayısıdır. Çevrim süreleri ve ışık sürelerinde değişiklikler yapılarak farklı senaryolar oluşturulmaktadır.

Öncelikle bir başlangıç senaryosu oluşturulmalıdır ve bununla ilgili olarak bir başlangıç çevrim süresi ve buna bağlı olarak da yeşil ışık ve kırmızı ışık süreleri belirlenmelidir. Başlangıç çevrim süresi için kullanıcıdan herhangi bir değer alınabilir veya programın bünyesi içine bir başlangıç çevrim süresi yerleştirilebilir ve program otomatik olarak bu değer ile başlar. Geliştirilen sistemde yapılan ise sabit ve ufak bir zaman dilimi belirlemek ve bunu toplam geliş yönü ile çarparak bir başlangıç çevrim süresi belirlemektir. Programda bu amaçla kullanılan formül 5.11'de görüldüğü gibidir.

Çevrim Süresi = Geliş Yönü Adedi x 5 (saniye) (5.11)

Her yön için yeşil ışık ve kırmızı ışık sürelerini ise şu şekilde belirlemekteyiz. Öncelikle kavşağın her yolunun kesintiye hiç uğramadan çalıştığı düşünülmektedir ve bir saat boyunca kavşağa gelen araçlar, gözlem yapılarak veya bir ön simülasyon (her yön için önceden belirlenmiş olan gelişler arası süreler ve dağılımlar kullanılarak bir saatlik bir simülasyon etüdü) çalışmasıyla belirlenmektedir. Daha sonra ise, kavşağa gelen tüm araçların toplamı ile her yola gelen araç sayıları ayrı olarak oranlanıp herbir geliş yönü için bir yüzde (%) değer bulunmaktadır. Sonuç olarak herbir yolun çevrim süresinden kendi yüzdesi kadar bir pay sahibi olabileceği düşünülmektedir. Bu şekilde de bağıntı 5.12 kullanılarak her geliş yönüne ait yeşil ışık süreleri belirlenmekte ve bağıntı 5.13 ile de kırmızı ışık süreleri elde edilmektedir. Burada yapılan her geliş yönünün yoğunluğunu

bulmak ve bir çevrim süresi boyunca herbir geliş yönüne kendi yoğunluğu kadar bir oranda geçiş hakkı vermektir.

$$YS_i = ÇS * \frac{GAS_i}{\sum_{j=1}^{m} GAS_j} \qquad (i = 1,2,....,m) \tag{5.12}$$

$$KS_i = ÇS - YS_i \tag{5.13}$$

YS_i = Kavşaktaki i. yol için yeşil yanma süresi

KS_i = Kavşaktaki i. yol için kırmızı yanma süresi

GAS_i = Kavşaktaki i. yola bir saatlik peryot boyunca gelen araç sayısı

m = Kavşaktaki yol sayısı

$ÇS$ = Çevrim süresi (Kavşaktaki bir yola tekrar geçiş hakkı verilene kadar geçen süre)

Burada açıklanan yaklaşımın Pascal dilindeki kodlaması aşağıda görüldüğü gibidir. Algoritma yapısı içerisinde ayrıca her yol için belirlenen yeşil ışık süreleri, o yöne ait maksimum kavşak geçiş süresi veya süreleri ile karşılaştırılıp, yeşil ışık süresinin en az maksimum kavşak geçiş süresi kadar olması sağlanır.

```
Procedure SINYAL_SU;
Var K2,K3    :byte;
   DURUM   :boolean;
   KAT_S    :real;
Begin;
TOP_ARAC:=0;
CEVRIM:=GELIS_ADEDI*5;{baslangic deger}
For K2:=1 to GELIS_ADEDI Do TOP_ARAC:=TOP_ARAC+GAS[K2];
For K2:=1 to GELIS_ADEDI Do
        YESIL_SU[K2]:=Round(CEVRIM*(GAS[K2]/TOP_ARAC));
DURUM:=False;
```

```
Repeat
For K2:=1 to GELIS_ADEDI DO
        Begin;
        if YESIL_SU[K2]<MAX_KGS[K2]
          then Begin;
             KAT_S:=MAX_KGS[K2]/YESIL_SU[K2];
             For K3:=1 to GELIS_ADEDI Do
                    YESIL_SU[K3]:=Round(YESIL_SU[K3]*KAT_S);
             End
end;
For K2:=1 to GELIS_ADEDI Do
          CEVRIM:=CEVRIM+YESIL_SU[K2];
End;(*Sinyal Suresi*)
```

Bu yaklaşım, başlangıç ışık sürelerini belirlemek için ve otomatik olarak yeni bir senaryo oluşturmak için de kullanılmaktadır. Her bir yeni senaryo, çevrim süresine bir miktar süre ilavesi ve ilave edilen bu sürenin yukarıdaki algoritma ile herbir geliş yönüne ait yeşil ışık sürelerine eklenmesiyle oluşturulur. Yeni bir senaryo oluşturulurken yukarıdaki algoritma yanında ayrıca, yol verimi (geçen araç/gelen araç) belirli bir değerin üzerinde olan yönlere ilave edilecek yeşil ışık zamanını, verimi düşük olan yollara aktarmak şeklinde bir yaklaşımda göz önünde bulundurulmaktadır. Böylece sadece bir yöne çok fazla ağırlık verilmesi önlenmiş olur ve de araç hacmi düşük olan yöne gereğinden fazla yeşil ışık süresi verilmemiş olur. Burada açıklanan sezgisel yapının PASCAL dili ile ifadesi aşağıda görüldüğü gibidir.

```
Procedure SenUret;
var S2    :byte;
   sayac1:byte;
   Art   :integer;
   Begin
   cevrim:=cevrim+(gelis_adedi*2);
   art:=gelis_adedi*2;
   if cevrim<120 then
```

```
begin
Top_Verim:=0;
Sayac1:=0;
for s2:=1 to gelis_adedi do
begin
DuzVerim[s2]:=1-(Departure[s2]/GelArac[S2]);
Top_Verim:=Top_verim+DuzVerim[s2];
if (Departure[s2]/GelArac[S2])>0.80 then Sayac1:=Sayac1+1;
end;
for s2:=1 to gelis_adedi do
begin
DuzVerim[s2]:=DuzVerim[s2]/Top_Verim;
if (Sayac1=0) or (sayac1=Gelis_Adedi)
then
Yesil_Su[s2]:=Yesil_Su[s2]+Round(Art*(GAS[s2]/TOP_ARAC))
else
Yesil_SU[s2]:=Yesil_SU[s2]+Round(DuzVerim[s2]*art);
end;
end
else Sen_Son:=True;
end;
```

Geliştirilen sistem, bu algoritmaları kullanarak otomatik olarak senaryolar oluşturur. Sistem maksimum çevrim süresi olarak seçilmiş olan 120 saniyenin üstüne çıkmayacak şekilde otomatik olarak senaryolar üretir ve bu süre aşıldığı an, senaryo üretimi durdurulup üretilmiş senaryolar içinden en iyi sonuçlara sahip olan senaryo, problemin çözümü olarak kabul edilir ve sonuçlar kullanıcıya tablo ve grafik formlarda sunulur. Otomatik olarak üretilen bu senaryoların bir noktada durdurulması gerekmektedir. Senaryo üretimi için çevrim süresi belirlenen bir minimumdan başlatılmakta ve gene belirlenen bir maksimuma kadar arttırılmaktadır. Dolayısıyla senaryolar bu belirlenmiş olan minimum ve maksimum değerler arasında üretilmektedir. Bu çalışmada maksimum çevrim süresi sezgisel olarak 120 saniye olarak belirlenmiştir. Bu süre istenildiğinde arttırılabilir veya azaltılabilir

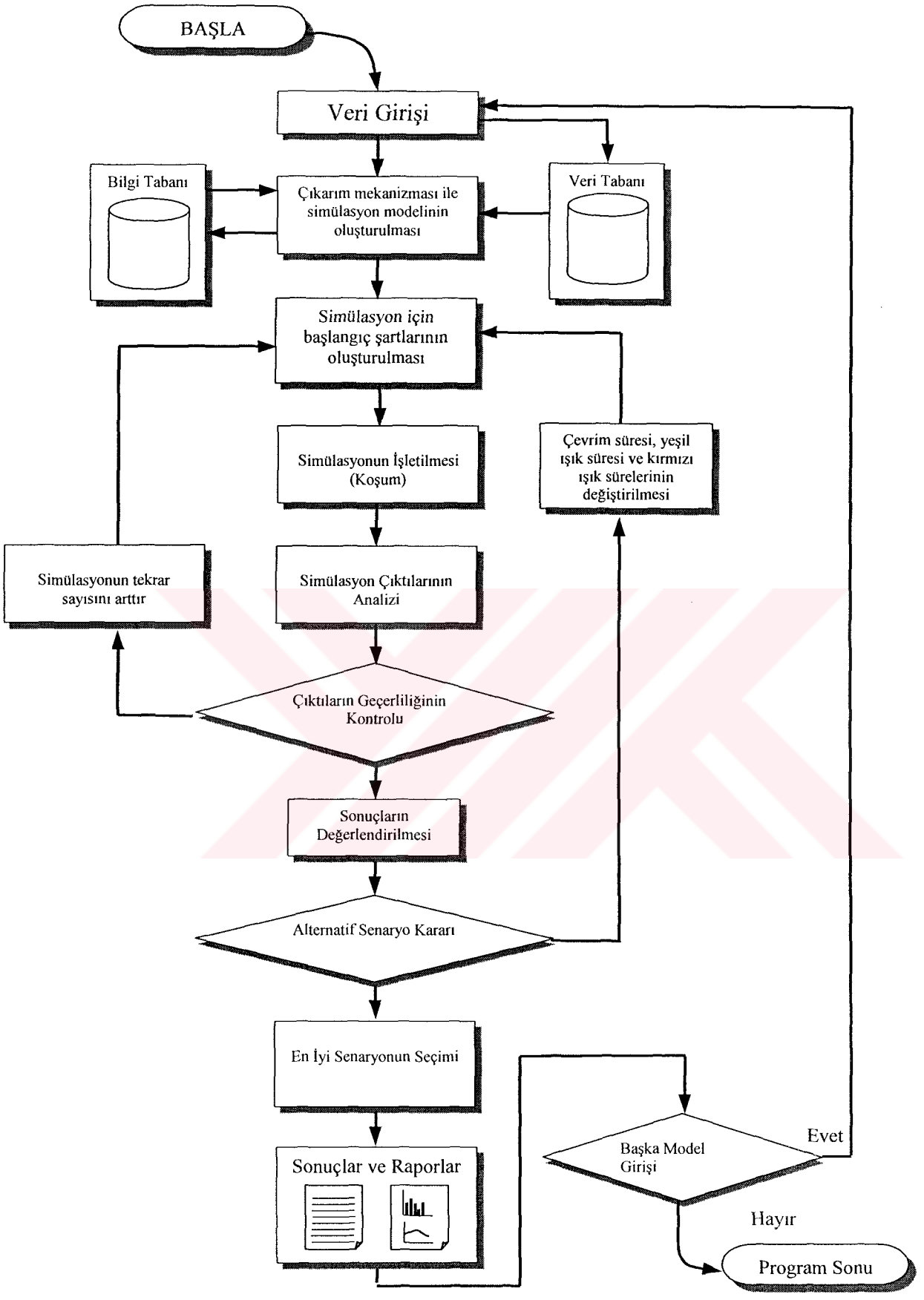
araştırılan aralık büyütüldükçe oluşturulan senaryoların sayısı artmaktadır. Otomatik olarak senaryo üretimini sağlayan prosedüre, gerektiği takdirde ortaya çıkarılan yeni yaklaşımların da ilave edilmesiyle sistem çok daha fazla stratejileri deneyebilecek bir yapıya kavuşturulabilir. Yapılması gereken, sadece bu prosedüre geliştirilen yeni yaklaşımın ilave edilmesiyle gerçekleştirilir. Bu yapılırken programın diğer prosedür ve fonksiyonları olduğu gibi kalır.

5.2.5. Geliştirilen bilgisayar programının genel yapısı

Bir kavşaktaki trafik ışıklarının simülasyonu için geliştirilen ve yapay zekâ ile bütünleşik bir yapıda olan bilgisayar programı Borland Pascal 7.0 ile yazılmıştır. Pascal'ın özelliğinden faydalanılarak, program modüller halinde çeşitli prosedür ve fonksiyonlardan oluşturulmuştur. Bu da programa, güncelleştirme ve geliştirme açısından büyük bir esneklik sağlamaktadır. Geliştirilen yazılımın genel bir akış diyagramı Şekil 5.12'de verilmiştir.

Geliştirilen program daha önceden analiz edilmiş veriler ile çalışmaktadır. Yani kullanıcı daha önceden simülasyon verileri ile ilgili dağılımları, dağılım parametrelerini ve sabit değerleri belirlemelidir.

Kullanıcı öncelikle veri girişini yapar. Sistem, kullanıcıdan gerekli tüm verileri alabilecek bir diyalog yapısına sahiptir. Kullanıcıdan elde edilen veriler kullanılarak bir çıkarım mekanizması ile verilere uygun bir kavşak yapısı ve simülasyon modeli oluşturulur. Daha sonra simülasyon için başlangıç şartları belirlenir ve sonra simülasyon işletilmeye başlanır. Geliştirilen sistemde çıktı analizi Welch (1981 ve 1983)'in algoritmasına göre gerçekleştirilmektedir. Bu algoritmanın yapısı Bölüm 2.11'de açıklanmıştır. Bu algoritma gereği en az 5 kez simülasyon koşumu yapılır ve bu 5 koşumun ortalaması alınır. Seçilen bir değişkenin bu ortalama değerleri bir grafik üzerinde gösterilerek ortaya çıkan eğri geliştirilen sistem tarafından otomatik olarak düzeltilmeye çalışılır. Eğer algoritma koşullarını sağlayan düzeltilmiş bir eğri bulunamazsa bir simülasyon koşumu daha yapılarak düzeltilmiş bir eğri bulunana kadar işleme devam edilir. Uygun bir budama noktası bulunamaz ise simülasyon tekrar sayısı arttırılarak simülasyon baştan çalıştırılır.



Şekil 5.12 Geliştirilen bilgisayar programının genel yapısı

Uygun bir budama noktası bulunduğunda ise, bu noktadan önceki değerler hesaplama dışında bırakılır. Daha sonra sonuçlar değerlendirilir ve alternatif bir senaryo üretilerek bu alternatif senaryo analiz edilir. Alternatif senaryo üretimine son verilince de senaryolar içinden en iyi olan senaryo problemin çözümü olarak seçilerek, tüm sonuçlar ve raporlar kullanıcıya sunulur. Programın veri girişi, gelişler arası sürenin hesaplanması, dağılımlara uygun rassal veri üretimi vb. prosedürler ve fonksiyonlarının bir kısmı Ek1'de verilmektedir.

Bütün bunların sonucunda kullanıcı, yeni bir problem girişi yapabilir veya programı sonlandırabilir.

Geliştirilen sistemin işleyişinin ve çeşitli prosedürlerin birbirleriyle olan ilişkilerin bir bütün olarak ortaya konulması için sistem kademe kademe açıklanmıştır. Bu sayede, sisteme bir bütün olarak baktığımızda işleyiş tarzı ortaya konulmuş olmaktadır. Sistem çalışırken ortaya çıkan yapı ve sonuca ulaşmak için gerçekleştirilmesi gerekli tüm fonksiyonlar adımlar olarak verilmiştir.

Girilen bir problemin sonuçlandırılması için programın çalıştırılması sırasında yapılan tüm işlemler ana hatlarıyla adımlar olarak ortaya konulmuştur. Böylece her işlemin sırası, kurallara bağlı olarak nelerin yapılması gerektiği, hangi verilerin elde edilmesi gerektiği, kavşak modelinin oluşturulması, otomatik olarak bir başlangıç senaryosunun oluşturulması, simülasyon için başlangıç şartlarının oluşturulması, simülasyonun çalıştırılması, otomatik olarak çıktı analizinin yapılması, otomatik olarak senaryonun oluşturulması, sonuçların sunulması ve adımların birbirleriyle olan ilişkileri bir bütün olarak belirlenmiş olur.

Geliştirilen sistemin işleyişinin ana adımları aşağıda görüldüğü gibidir. Bu, sisteme bir bütün olarak baktığımızda karşımıza çıkan bir yapıdır.

Adım 1 Kullanıcıdan, etkileşimli bir diyalog ile gerekli olan tüm verilerin elde edilmesi. Bu veriler ise, simülasyonu yapılacak kavşağın adı, kavşaktaki yol sayısı, her yolun şerit sayısı, her geliş yönü için dağılım tipi ve

dağılım parametreleri, araçların sapacakları yönler, sapma aralıkları, bir araçlık mesafeyi kat etme süresi, simülasyon koşum süresi.

Adım 2 Program yapısında yer alan makro kavşak modelinin girilen değerlere uygun kavşak yapısı şeklinde modellenmesi ve verileri girilen kavşak yapısının oluşturulması.

Adım 3 Kavşaktaki her geliş yönünün, aynı anda ışık olmadan geçiş yapabileceği kabulü ile bir ön simülasyonun yapılması. Bu sayede her geliş yönü için bir saatte kavşağa gelen araç sayılarının tespit edilmesi.

Adım 4 Kavşak için bir başlangıç senaryosunun oluşturulması. Başlangıç senaryosu için geliştirilen sezgisel yapı bölüm 5.2.4'de açıklanmıştır. Kavşak için çevrim süresi, yeşil ışık ve kırmızı ışık sürelerinin belirlenmesi.

Adım 5 Simülasyon için başlangıç şartlarının oluşturulması ve programda kullanılan tüm değişkenlerin sıfırlanması ve sabitlere değerlerinin atanması.

Adım 6 Simülasyonun işletilmesine başlanması.

Adım 7 Simülasyonun toplam adım sayısının hesaplanması.
TopSimAdımı = Round (Koşum Süresi/Çevrim Süresi)

Adım 8 Başlangıç sinyallerinin atanması.
Bir numaralı geliş yönüne yeşil diğer yönlere kırmızı atayarak başlangıç sinyallerinin oluşturulması.

Adım 9 Geliş yönü i için.
Eğer sinyal durumu = YESIL ise
Adım 10'a git

Eğer sinyal durumu = KIRMIZI ise

Adım 28'e git

Adım 10 Simülasyon adımını 1 arttır.

Simülasyon Adımı = Simülasyon Adımı + 1

Simülasyon adımı hangi çevrim içinde bulunulduğunu göstermektedir.

Adım 11 Eğer Simülasyon Adımı = 1

İse Yesil Başlangıç = 0

Eğer Simülasyon Adımı > 1 ise

Yesil Başlangıç = Yeşil Başlangıç + Çevrim Süresi

Adım 12 Yesil Sonu = Yesil Başlangıç + Yeşil $Süresi_i$ (i. Geliş yönüne ait yeşil süresi)

Adım 13 Kavşağın i. Geliş yönüne araç gelişini oluştur. Gelen araç için gelişler arası süreyi belirle (Geliş Ara Süre j)

Adım 14 Kavşağın i. Geliş yönüne gelen araç için döneceği yönü belirle.

Adım 15 Kümülatif Süreyi hesapla

Kümülatif Süre = Kümülatif Süre + Geliş Ara Süre

Adım 16 Eğer Kümülatif Süre < Yeşil Stop

İse Adım 13'e dön

Aksi takdirde Adım 17 git

Adım 17 Eğer (Aracın geliş zamanı ≥ Yeşil $Başlangıç_i$)

ve (Aracın Geliş Zamanı ≤ Yeşil $Sonu_i$)

İse Gelen aracı içinde bulunulan çevrim içinde gelen araç olarak belirle.

Adım 18 Eğer (Aracın geliş zamanı) < Yeşil $Başlangıç_i$

İse

Aracı daha önceki çevrimlerden bekleyen araç olarak belirle.

Adım 19 Eğer (Aracın geliş zamanı) > Yeşil sonu$_i$

İse

Aracı bir sonraki çevrime kalan araç olarak belirle.

Adım 20 Her aracın harcadığı toplam süreyi belirle. Kuyrukta bekleme süresi ve aracın kuyrukta bulunduğu konumdan kavşağın başına gelene kadar geçen sürenin toplamı.

Adım 21 Aracın Geliş zamanı < Yeşil Başlangıcı

İse

Koşul = Yeşil Başlangıcı + TopKuySüresi

Adım 23'e git

Adım 22 Aracın Geliş zamanı ≥ Yeşil Başlangıcı

ise

Koşul = Aracın geliş zamanı + TopKuySüresi

Olarak belirle ve Adım 23'e git

Adım 23 Eğer Kosul ≤ Yeşil Sonu$_i$

ise

Aracı geçen araç olarak belirle

Ayrılan araç = Ayrılan araç + 1 olarak belirle ve Adım 25'e git.

Adım 24 Eğer Koşul > Yeşil Sonu$_i$ ise

Kuyrukta kalan araç sayısını 1 arttır.

Adım 25 Geliş yönü i İçin Yeşil sonu

zamanına kadar gelen tüm araçların incelenip incelenmediğini belirle

Eğer Çevrim süresince gelen tüm araçlar incelendi

ise Adım 26'ya git
Aksi takdirde Adım 21'e git.

Adım 26 Geliş yönlerindeki kuyruklardaki son durumları belirle ve sonuçları ilgili değişkenlere aktar. Geçen araçlar, kalan araçlar, bekleme süreleri ve sistem sürelerini belirle ve Yeşil Durumunu sonlandır.

Adım 27 Sinyali kırmızı yap.

Adım 28 Kırmızı Başlangıcı = Yeşil Sonu olarak belirle.

Adım 29 Kırmızı Sonu=Kırmızı Başlangıcı + (Çevrim süresi – Yeşil $Süresi_i$)

Adım 30 Kavşağın i. Geliş yönüne araç gelişi oluştur ve araç için gelişler arası süreyi hesapla (GelişAraSüre)

Adım 31 Gelen araç için döneceği yönü belirle

Adım 32 Kümülatif süreyi belirle
Kümülatif süre = Kümülatif süre + Geliş AraSüre

Adım 33 Eğer Kümülatif süre < Kırmızı sonu
İse Adım 30'a git

Adım 34 Eğer Kümülatif süre ≥ Kırmızı sonu
ise Adım 35'e git

Adım 35 Kırmızı ışık sonuna ulaşıldı
Sinyali yeşil yap

Adım 36 Eğer Simülasyon Adımı = Toplam Simülasyon
Adımı ise simülasyonun t. tekrarınını sonlardır ve Adım 37'e git

Aksi takdirde Adım 9'a dön

Adım 37 t. tekrarın sonuçlarını belirle ve sakla.

Adım 38 Simülasyon tekrarını 1 arttır
t=t+1

Adım 39 Eğer t<tekrar sayısı
ise Adım 4'e git
aksi takdirde Adım 40'a git

Adım 40 Ön simülasyon sonuçlarına göre geliş yoğunluğu en fazla olan geliş yönünü belirle

Adım 41 En yoğun geliş yönü için otomatik çıktı analizine başla. Çıktı analizi için en uygun geliş yönüne ait sistem süresini baz alarak al.

Adım 42 Her simülasyon tekrarındaki en yoğun yöne ait sistem süresi çıktılarının ortalamasını alarak ortalama sistem süresi çıktısını oluştur.

Adım 43 Elde edilen ortalama sistem süresi çıktı değerlerinin bir grafik üzerinde oluşturularak kullanıcıya sunulması

Adım 44 Eğri düzeltme değerini 10 olarak belirle
EgriDuzDeg = 10

Adım 45 Eğri düzeltme değerinin "EgriDuzDeg" olarak kabul edildiği ve hareketli ortalamanın hesaplanması ve oluşturulan yeni grafiğin çizilerek kullanıcıya sunulması

Adım 46 Oluşturulan grafiğin analiz edilerek, eğride sabit duruma geçilip geçilmediğinin belirlenmesi

Adım 47 Eğer sabit duruma geçildi
ise
Adım 48'e git
Aksi takdirde
EgriDuzDeg= EgriDuzDeg+10
olarak belirle ve Adım 45'e git

Adım 48 Eğer EgriDuzDeg ≤ (Gözlem Sayısı/2)
ise Adım 45 e dön
aksi takdirde (Tekrar sayısı=Tekrar sayısı +1)
olarak belirle ve adım 4'e dön

Adım 49 Sabit duruma geçilen noktanın (budama noktası) belirlenmesi.

Adım 50 Elde edilen simülasyon çıktılarından sabit durum önceki çıktı değerlerini çıkararak tüm çıktı değişkenleri için yeni ortalamaların hesaplanması.

Adım 51 Senaryo = Senaryo +1 olarak belirlenmesi ve elde edilen simülasyon çıktılarının bir tablo formunda kullanıcıya sunulması. Bu tablo da, Çevrim Süresi, Herbir geliş yönü için, Yeşil ışık süresi, kırmızı ışık süresi, geçen araç sayısı, gelen araç sayısı, ortalama kuyruk uzunluğu, verim ve ortalama sistem süresi bilgileri yer almakta ayrıca kavşağa toplam gelen araç sayısı, kavşaktan toplam geçen araç sayısı ve ortalama kavşak verimi bilgileri de bulunmaktadır.

Adım 52 Eğer Senaryo ilk Senaryo ise
Adım 54'e git
Aksi takdirde Adım 53'e git

Adım 53 Senaryonun bir önceki senaryo ile karşılaştırılarak en iyi senaryonun seçilmesi. Senaryo seçiminde kavşaktan geçiş yapan araç sayısı ve

kavşak verimi karşılaştırılarak karar verilmektedir. Senaryo seçimi yapılırken ayrıca geçen araç sayısı farklı fakat kavşak verimi aynı olan senaryo alternatif senaryo olarak belirlenmekte ve alternatif senaryo olarak kullanıcıya sunulmaktadır.

Adım 54 Alternatif bir senaryonun belirlenmesi için öncelikle yeni bir çevrim değerinin belirlenmesi

Cevrim Suresi=Cevrim Suresi + (Gelis Adedi x 2)

Arttırım süresi ise

art = geliş adedi x 2

Adım 55 Eğer Cevrim Suresi < 120

ise Adım 56'ya git

aksi takdirde Adım 62'ye git

Adım 56 Toplam verim = 0 olarak belirle

Adım 57 Tüm geliş yönleri için düzeltilmiş verimlerin hesaplanması

$DuzVerim_i$=1- (Ayrılan $araç_i$/gelen $araç_i$)

ToplamVerim = Toplam Verim + $DuzVerim_i$

Adım 58 Tüm geliş yönleri için

Eğer (Ayrılan $araç_i$/gelen $araç_i$) > 80

ise Say=Say + 1

Adım 59 Tüm geliş yönleri için

($DuzVerim_i$=$DuzVerim_i$/Toplam Verim) belirle

Adım 60 Eğer (Say=0) veya (Say=geliş Adedi)

ise

$YesilSuresi_i$=$YesilSuresi_i$+Round(Artx(Gelen Arac $Sayısı_i$/Toplam Arac Sayısı))

Aksi takdirde

$YesilSuresi_i = YesilSuresi_i + Round\ (DuzVerimxart)$

olarak tüm geliş yönleri için hesapla

Adım 61 Oluşturulan yeni senaryonun çalıştırılması için Adım 5'e git.

Adım 62 Senaryo üretiminin durudurulması ve elde edilen optimal senaryonun ve eğer mevcut ise alternatif senaryo veya senaryoların kullanıcıya sunulması.

Adım 63 Her senaryoda kavşaktan geçiş yapan araç sayıları histogramının kullanıcıya sunulması

Adım 64 Bilgilerin saklanması ve kullanıcıya yeni bir model girişi yapılıp yapılmayacağının sorulması.

Adım 65 Eğer Yeni Model girişi yapılacak

ise Adım 1'e dön

Aksi takdirde Adım 66'ya git.

Adım 66 Programdan çıkılması.

5.3. Geliştirilen Prototip Model ile Yapılan Bir Uygulama ve Sonuçları

Geliştirilen zeki simülasyon ortamı, farklı kavşak yapılarına cevap verebilecek niteliktedir. Fakat gene de sistemin bazı sınırlamaları mevcuttur. Tasarlanan sistemin cevap verebileceği kavşak sınırlamaları ise şunlardır:

- Maksimum geliş yönü =7 tane,
- Toplam şerit sayısı =12 tane,
- Her yol için maksimum kuyruk uzunluğu = 400 araç,
- Aynı anda yeşil yanabilecek yol sayısı =2 tanedir. (aynı anda geçiş yapabilecek yol sayısı)

Bu değerler ise, bilgisayar hafızasının maksimumda kullanıldığı değerler değildir ve gerekirse bütün değerlerde arttırıma gidilebilir. Sistem gerçek kavşak yapıları altında gerçek ve teorik verilerle çalıştırılmıştır. Örnek kavşak yapısı olarak, dört yol kavşak tipi üzerinde sistem çalıştırılmış ve sonuçlar alınmıştır.

Dört yol kavşak modeli: Sistem, daha önceden analizi edilmiş verilerle çalışmaktadır. Bu kavşak yapısı ile ilgili veriler Çizelge 5.1 ve 5.2'de görüldüğü gibidir.

Çizelge 5.1 Dört yol kavşağının geliş yönü, dağılım tipi ve dağılım parametreleri ile ilgili veriler.

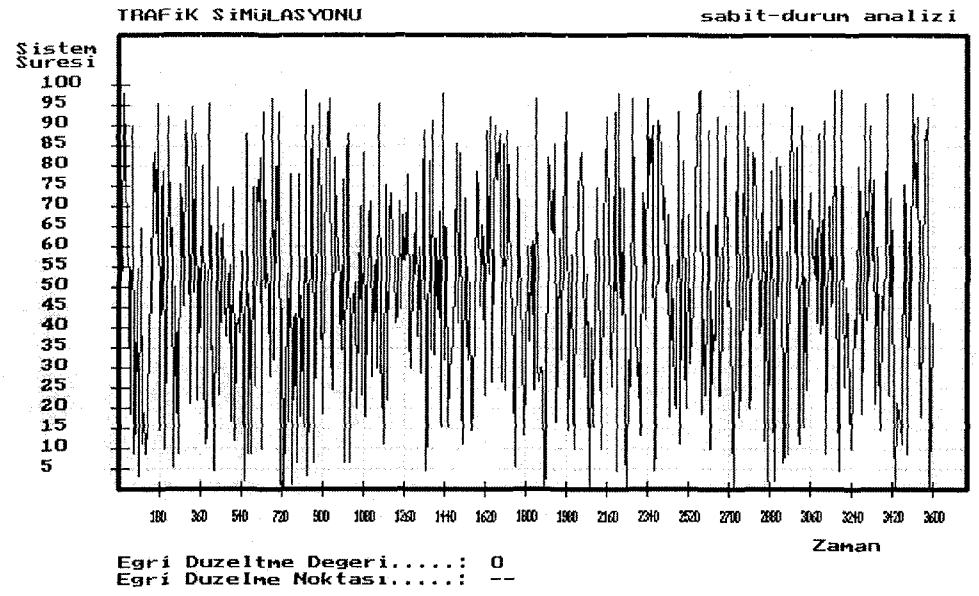
Kavşak Tipi : Dört Yol		
Geliş Yönü	Dağılım Tipi	Dağılım Değeri (Gelişler arası süre (Saniye))
1	Üssel	10
2	Üssel	8
3	Normal	Ort.:9 ve Std.Sap.:3.5
4	Üssel	5

Çizelge 5.2 Dört yol kavşağında yol, şerit yapısı ve kavşağa gelen araçların dönüşleriyle ilgili veriler

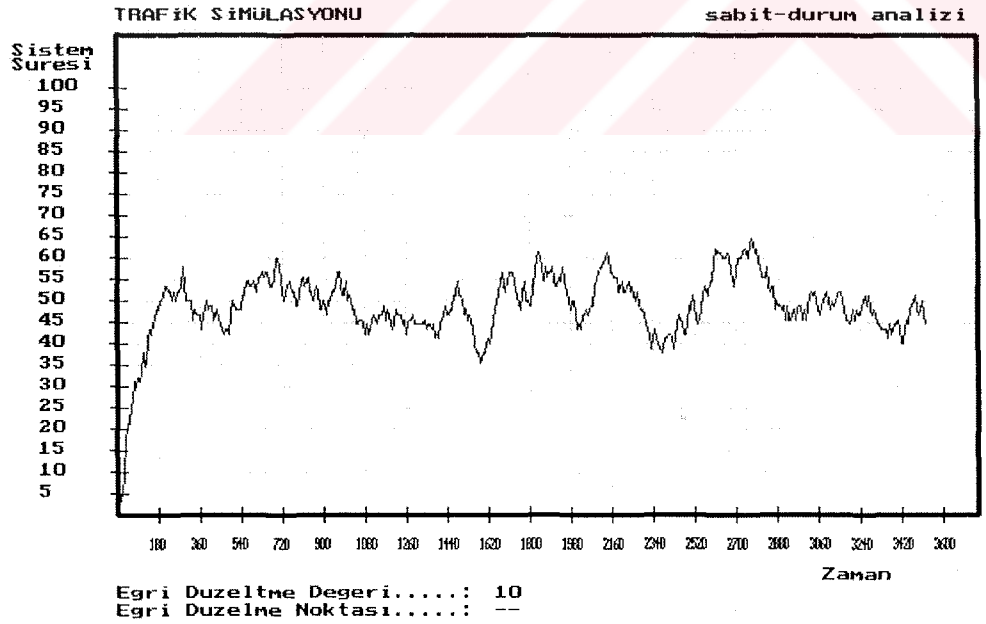
YOL NO	Her yolun şerit sayısı	Dönüş yapılabilecek yol sayısı	Dönüş yapılabilecek yönler	Dönüş Süreleri (Saniye)	Dönüş olasılıkları
1	1	3	6 7 8	2 3 4	0.3 0.3 0.4
2	2	3	7 8 5	2 3 3	0.3 0.2 0.5
3	1	3	8 5 6	2 2 3	0.7 0.2 0.1
4	2	3	5 6 7	2 2 3	0.4 0.2 0.4
5	1	-	-	-	-
6	2	-	-	-	-
7	1	-	-	-	-
8	2	-	-	-	-

Programın çalıştırılması sonucu elde edilen çıktılar

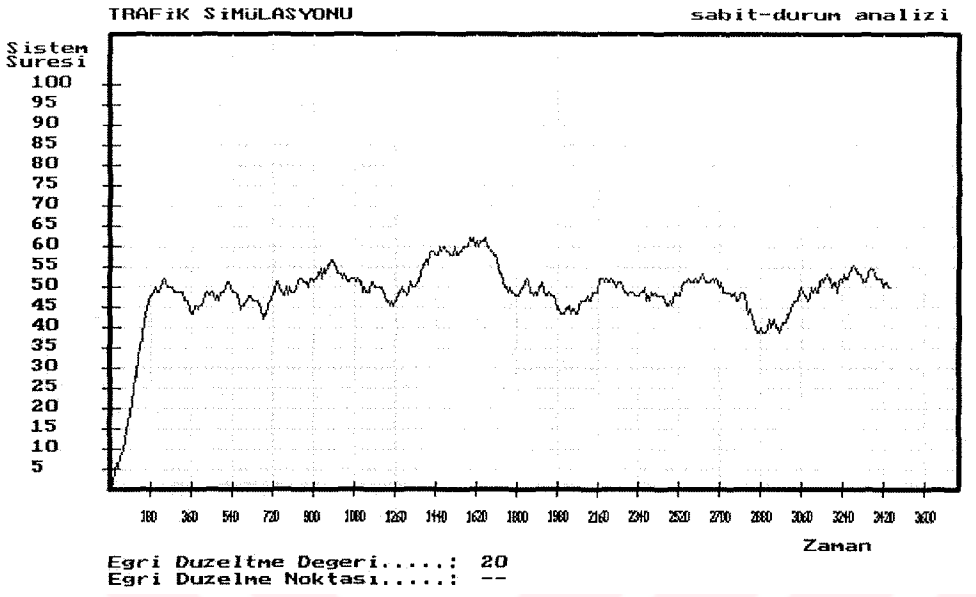
Program çalıştırıldığı zaman, ilk olarak kullanıcıdan verilerini girmesini istemektedir. Yukarıdaki veriler bir kullanıcı tarafından diyalog etkileşimli bir arayüz vasıtasıyla sisteme girilmektedir. Veri girişleri Ek2'de verilmiştir. Geliştirilen yazılım, kullanıcıdan verileri elde ettikten sonra, girilen verilere bağlı olarak problemin çözümünü, alternatif senaryolar üreterek bulmaya ve problemi sonuçlandırmaya çalışır. Programın çalıştırılmasıyla elde edilen sabit durum analizi ile ilgili çıktılar, en iyi senaryonun çıktısı ve her senaryoda kavşaktan geçen araçların sayıları ile ilgili histogram çıktısı, Şekil 5.13 ile Şekil 5.18 arasındaki şekillerde verilmiştir. Program tarafından oluşturulan alternatif senaryoların bazıları da Ek 3'de verilmiştir.



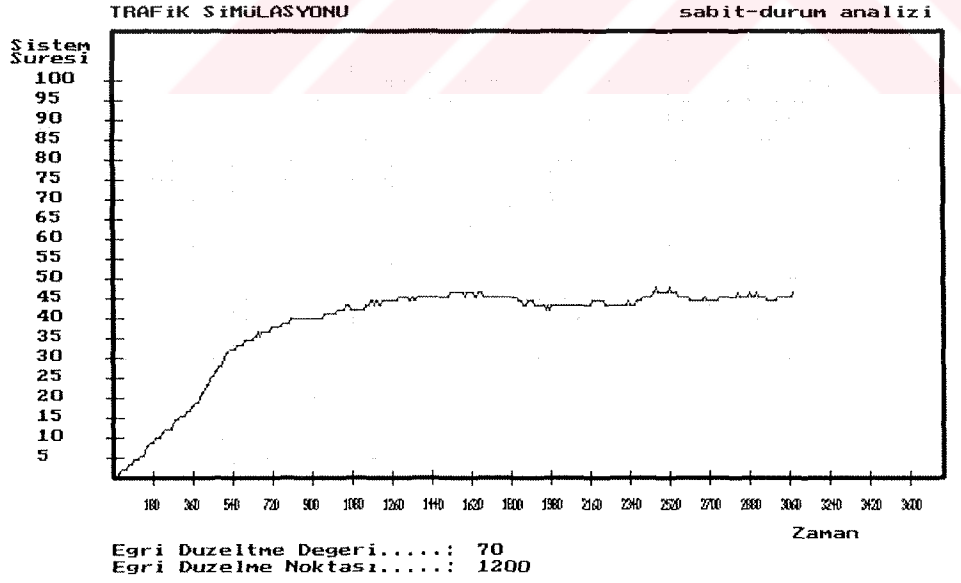
Şekil 5.13 Dört yol kavşağının sabit durum analiziyle ilgili olarak ilk çıktı



Şekil 5.14 Eğrinin düzeltilmesiyle ilgili çıktılar -1



Şekil 5.15 Eğrinin düzeltilmesiyle ilgili çıktılar –2



Şekil 5.16 Düzeltilmiş eğri ve bir budama noktası bulunmuş çıktı

En iyi çözüm

Simülasyon Sonuçları

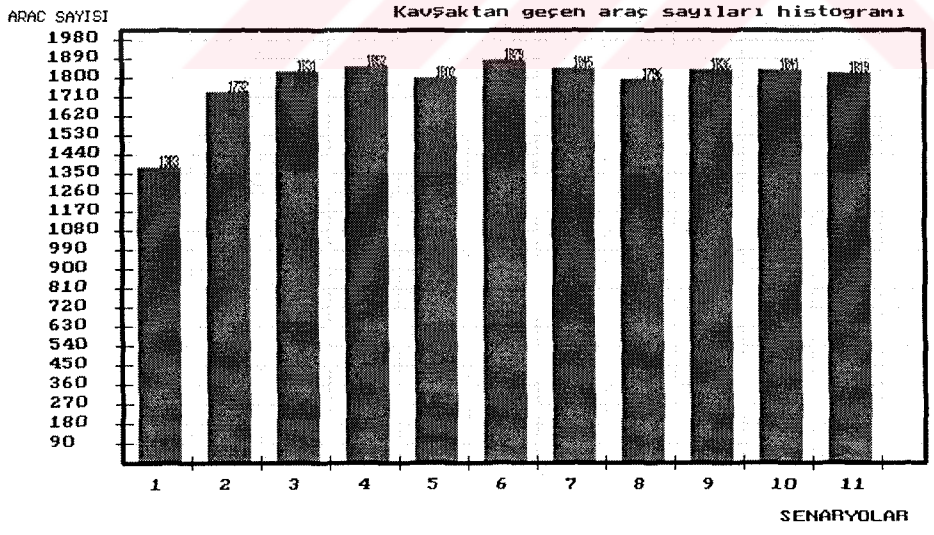
Kavsak Adi : DORTYOL-1 SENARYO 6

Cevrim Suresi : 77 saniye

		Gelisler		
	1.Yon	2.Yon	3.Yon	4.Yon
	------	------	------	------
Yesil Isık Suresi	: 15	15	18	20
Kırmz Isık Suresi	: 62	62	59	57
Gecen arac Sayısı	: 313	461	381	724
Gelen arac Sayısı	: 358	476	389	735
Ort.Kuy.Bek.Sure.	: 173.4	37.2	37.6	35.6
Ort.Kuy.Uzunlugu	: 23	11	9	16
Verim	: 0.87	0.97	0.98	0.99
Ortalm Sis. Suresi	: 184.01	44.74	49.15	46.16

Kavsaktan gec Arac	:	1879
Kavsaga gelen Arac	:	1958

Şekil 5.17 Otomatik olarak oluşturulmuş senaryolar içinden seçilmiş olan en iyi senaryo



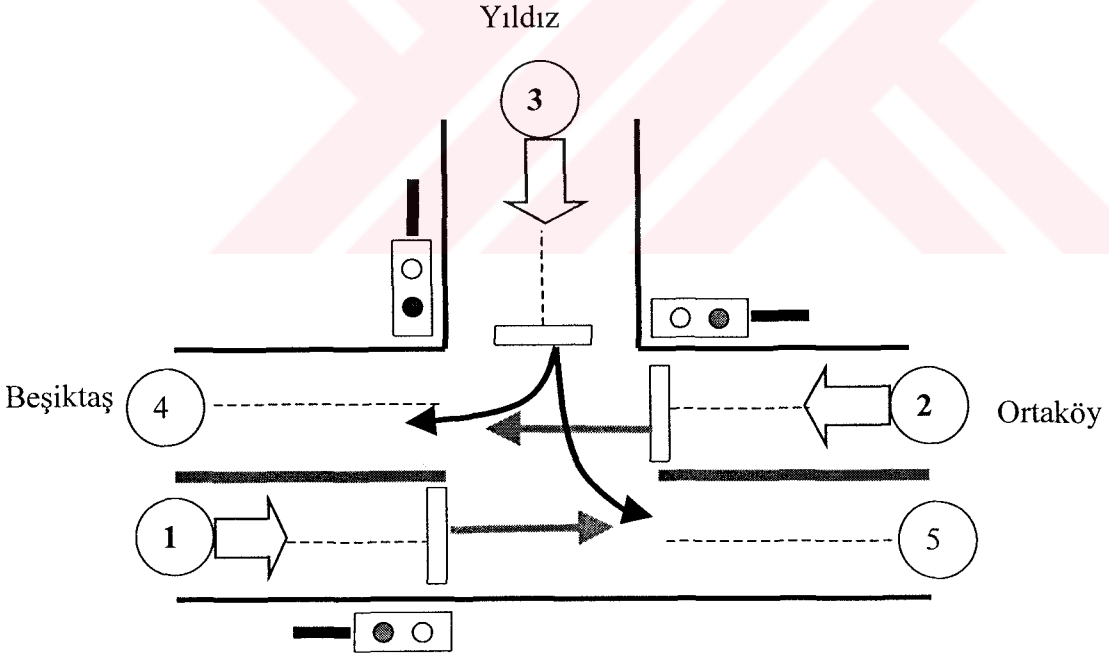
Şekil 5.18 Kavşaktan geçen araç sayıları ile ilgili histogram

5.4. Geliştirilen Prototip Model ile Yapılan Gerçek Şartlardaki Bir Uygulama ve Sonuçları

Gerçek uygulama için göz önüne alınan kavşak üç yolun birleşmesinden oluşmaktadır. Kavşak yapısının genel özellikleri şunlardır:

- Kavşağın geliş yönü = 3
- Kavşaktaki toplam yol sayısı = 5
- Toplam şerit sayısı = 10
- Aynı anda yeşil yanan yön sayısı =2

Kavşağın genel bir görünümü aşağıdaki şekil 5.19'da görüldüğü gibidir. Şekilden de görüldüğü gibi 1. Yönden gelen araçlar 4. ve 5. yöne sapmaktadırlar. 2. Yönden gelen araçlar 4. yöne ve 3. yönden gelen araçlar ise 5. yöne sapmaktadırlar. 2. ve 3. yöne aynı anda yeşil ışık yanmaktadır.



Şekil 5.19. Kavşağın yapısı.

Ele alınan kavşak üzerindeki ölçümler saat 12:45 ile 13:05 arasında 20 dakika boyunca yapılmıştır. Elde edilen bu ölçüm değerleri ile BESTFIT programı kullanılarak uygun bir dağılım belirlenmiştir.

Çizelge 5.3 Birinci geliş yönüne ait ölçüm değerleri

GELİŞ YÖNÜ 1	
Gelişler Arası Süre (saniye)	Gözlenen Araç Sayısı
2	1
3	1
4	1
5	3
6	3
7	6
8	4
9	7
10	9
11	9
12	10
13	11
14	9
15	7
16	7
17	2
18	1
Toplam Gözlenen Araç Sayısı : 92	

Çizelge 5.4 İkinci geliş yönüne ait ölçüm değerleri

GELİŞ YÖNÜ 2	
Gelişler Arası Süre (saniye)	Gözlenen Araç Sayısı
1	101
2	87
3	71
4	51
5	34
6	19
7	22
8	8
Toplam Gözlenen Araç Sayısı : 393	

Çizelge 5.5 Üçüncü geliş yönüne ait ölçüm değerleri

GELİŞ YÖNÜ 3	
Gelişler Arası Süre (saniye)	Gözlenen Araç Sayısı
1	87
2	67
3	53
4	40
5	42
6	30
7	18
8	7
9	9
10	3
Toplam Gözlenen Araç Sayısı : 350	

Tüm geliş yönleri için BESTFIT yazılımından elde edilen sonuçlar EK4'te verilmiştir. Geliş yönü 1'den gelen araçlar 4. veya 5. yöne dönebilmektedirler. 20 dakikalık gözlem boyunca 1.yönden toplam 92 araç gelmiştir. Bunların 23'ü 4.yöne ve 69'u da 5.yöne sapmaktadır. Sonuçta bu bir olasılık olarak belirtilirse 1.yönden gelen araçlar 0.25 olasılıkla 4.yöne ve 0.75 olasılıkla da 5.yöne sapmaktadır.

Aşağıdaki çizelgelerde kavşak ile ilgili bilgiler yani kavşağın her geliş yönüne ait dağılımlar ve dağılım parametreleri, yol sayısı ve her bir yolun şerit sayısı, her geliş yönünden dönüş yapılabilen yönler, dönüş süreleri ve dönüş olasılıkları özetlenmiş olarak verilmektedir.

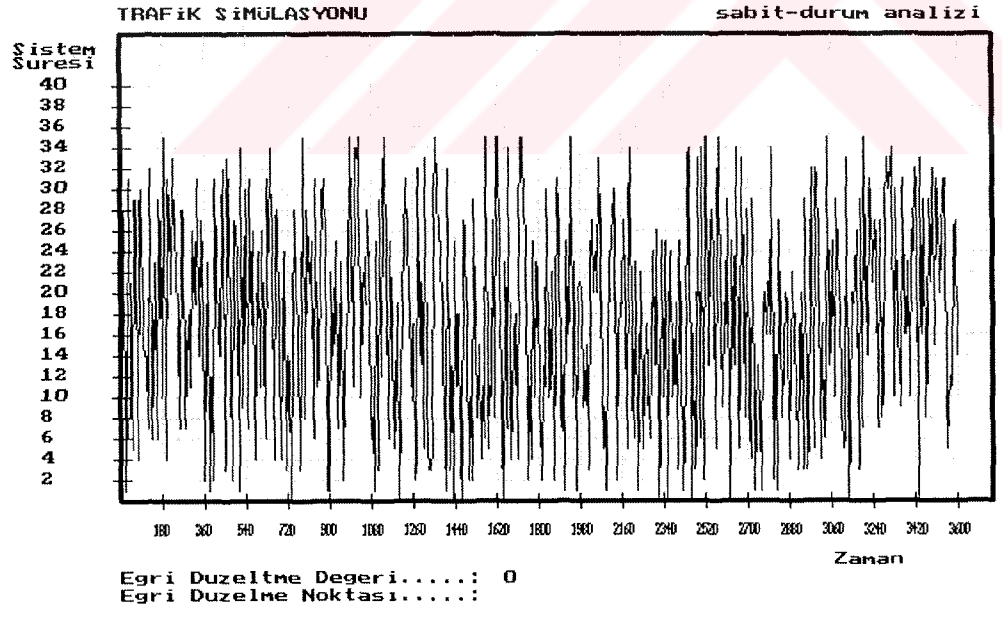
Çizelge 5.6 Üç yol kavşağının geliş yönü, dağılım tipi ve dağılım parametreleri ile ilgili veriler.

Kavşak Tipi : Üç Yol		
Geliş Yönü	Dağılım Tipi	Dağılım Değeri (Gelişler arası süre (Saniye))
1	Normal	Ort.:11.27 ve Std.Sap.:3.45
2	Üssel	3.04
3	Ussel	3.44

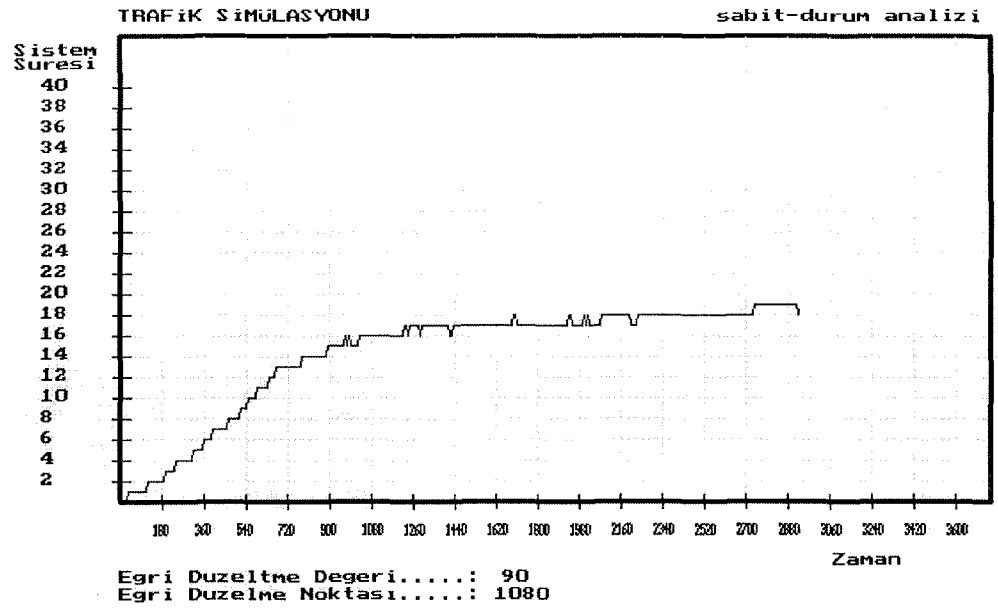
Çizelge 5.7 Üç yol kavşağının yol, şerit yapısı ve kavşağa gelen araçların dönüşleriyle ilgili veriler

YOL NO	Her yolun şerit sayısı	Dönüş yapılabilecek yol sayısı	Dönüş yapılabilecek yönler	Dönüş Süreleri (Saniye)	Dönüş olasılıkları
1	2	2	4 5	2 4	0.25 0.75
2	2	1	5	3	1.0
3	2	1	4	3	1.0
4	2	-	-	-	-
5	2	-	-	-	-

Elde edilen bu bilgiler sisteme girilerek çalıştırılmış ve bu kavşak için aşağıdaki sonuçlar elde edilmiştir. Şekil 5.20 ile Şekil 5.21'de otomatik çıktı analizinin sonuçları görülmektedir. Şekil 5.22 ile Şekil 5.23'te ise oluşturulan senaryolar içinden seçilen en iyi senaryo ve alternatif olarak belirlenen senaryo görülmektedir. Şekil 5.24'te de otomatik olarak oluşturulmuş olan tüm senaryolar için kavşaktan geçiş yapan araç sayıları histogramı görülmektedir.



Şekil 5.20 Kavşağın sabit durum analiziyle ilgili olarak ilk çıktı



5.21 Düzeltilmiş eğri ve bir budama noktası bulunmuş çıktı

En iyi çözüm

Simülasyon Sonuçları

Kavsak Adi : UCYOL SENARYO 4

Cevrim Suresi : 41 saniye

		Gelisler	
	1.Yon	2.Yon	3.Yon
Yesil Isık Suresi	: 13	24	24
Kırmz Isık Suresi	: 28	17	17
Gecen arac Sayısı	: 316	1192	1057
Gelen arac Sayısı	: 320	1206	1070
Ort.Kuy.Bek.Sure.	: 14.5	19.7	16.1
Ort.Kuy.Uzunlugu	: 4	14	12
Verim	: 0.99	0.99	0.99
Ortalm Sis. Suresi	: 18.57	31.23	26.61

Kavsaktan gec Arac	:	2565
Kavsaga gelen Arac	:	2596
Verim	:	0.99

Şekil 5.22 Otomatik olarak oluşturulmuş senaryolar içinden seçilmiş olan en iyi senaryo

Alternatif çözüm (verimler eşit)

Simülasyon Sonuçları

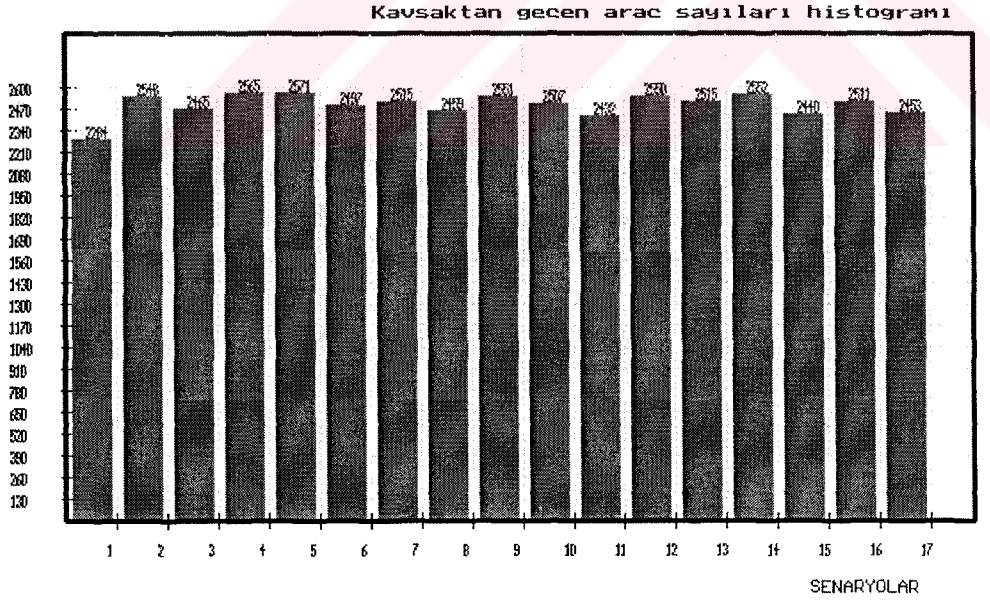
Kavsak Adi : UCYOL SENARYO 5

Cevrim Suresi : 47 saniye

		Gelisler	
	1.Yon	2.Yon	3.Yon
Yesil Isık Suresi	: 16	27	27
Kırmz Isık Suresi	: 31	20	20
Gecen arac Sayısı	: 319	1198	1054
Gelen arac Sayısı	: 323	1211	1070
Ort.Kuy.Bek.Sure.	: 15.9	21.0	18.3
Ort.Kuy.Uzunlugu	: 4	16	13
Verim	: 0.99	0.99	0.99
Ortalm Sis. Suresi	: 20.35	33.85	29.97

Kavsaktan gec Arac	:	2571
Kavsaga gelen Arac	:	2604
Verim	:	0.99

Şekil 5.23 Otomatik olarak seçilmiş olan alternatif senaryo



Şekil 5.24 Kavşaktan geçen araç sayıları histogramı

6. SONUÇLAR ve ÖNERİLER

Yapay zekâ ve özellikle uzman sistemlerin simülasyon ile birleştirilmelerinin sonucu kazanılan bilgiler, oluşturulan teknikler ve elde edilen teorik ve pratik sonuçlar, simülasyonun değişik alanlardaki uygulamalarını son derece ilginç bir hale getirmiştir. Bir simülasyon çevriminin bütün kısımları, mevcut yapay zekâ ve uzman sistem teknikleri kullanılarak otomatik olarak yapılabilir. Böylece, çok daha kısa sürede bir simülasyon etüdünün yapılmasına olanak sağlanır.

Yapay zekâ tabanlı simülasyon yapıları ile klasik simülasyon sistemleri arasındaki farklar ve yapay zekâ tabanlı simülasyon sistemlerinin sahip olduğu bir takım avantajları aşağıdaki gibi sıralayabiliriz:

- Yapay zekâ ile bütünleşik bir simülasyon sistemi ile klasik simülasyon sistemi arasındaki temel fark, modelin kurulması ve işletilmesinde ortaya çıkmaktadır. Yapay zekâ ve uzman sistem tekniklerinden faydalanan bir simülasyon ortamı, otomatik problem spesifikasyonuna sahiptir ve kullanıcıdan gerekli tüm veriyi diyalog, grafik veya doğal dil etkileşimli bir arayüz ile elde ederek problemi belirler.

- Yapay zekâ tabanlı bir sistemde veri tabanı, bilgi tabanı ve kontrol yapısı birbirinden ayrı olarak bağımsız durumdadır ve herbir yapı, birbirini etkilemeden kolayca değiştirilebilir. Günümüzdeki simülasyon yazılımlarında da model ve deney birbirinden ayrılmış durumdadır.

- İdeal bir yapay zekâ tabanlı simülasyon ortamı, kendi bünyesinde yer alan bir öğrenme modülü ile sistem kendi bilgilerinden ve tecrübelerinden yeni şeyler öğrenebilmeli ve gerektiğinde kendi bilgi tabanını değiştirebilmelidir. Bu, teorik olarak mümkün olmasına rağmen, hâlâ tam olarak oturtulmuş bir yapı değildir.

- Zeki bir simülasyon ortamı, kullanıcıdan elde edilen verilerden oluşan veri tabanını ve sistemin can damarı olan bilgi tabanını kullanarak çıkarımlar yapan ve model kurabilen bir çıkarım mekanizmasına sahiptir.

Bu tez çalışmasında yapay zekâ ile simülasyon tekniklerinin benzerlikleri ve her iki tekniğinde birleştirilme yaklaşımları incelenmiştir. Bunun sonucunda her iki tekniğin en ileri düzeyde ideal olarak ne şekilde bütünleştirilebileceğini gösteren kavramsal olarak soyut bir model önerilmiştir. Böylece yapay zekâ ile tamamen bütünleşik ideal bir simülasyon ortamı ve bu ortamda en ileri düzeyde neler olması gerektiği ortaya konulmuştur. Daha sonra ise ideal bir sistem olarak önerilen bu yapının bazı kısımları somut olarak gerçekleştirilerek ortaya konulmuştur. Tasarlanan bu yapı bir kavşaktaki trafik ışıklarının analizine uygulanmıştır. Böylece geliştirilen sistem ile, bir kullanıcıdan veriler elde edildikten sonra herhangi bir karar vericiye danışılmadan kararları sistem kendi alarak otomatik olarak senaryolar üretmekte ve üretilen senaryolar birbirleriyle mukayese edilerek yorumlanmakta ve en iyi senaryo belirlenerek sistem tarafından kullanıcıya sunulmaktadır.

Bir kavşak sisteminin optimizasyonu için gösterge olan kriter olarak belirli bir zaman diliminde kavşaktan geçiş yapan araç sayısı seçilmiştir. Karar değişkeni olarak da, çevrim süresi, yeşil ışık süresi ve kırmızı ışık süresi belirlenerek, alternatif senaryoların oluşturulmasında belirlenen bu karar değişkenleri kullanılmıştır. Çözüm yaklaşımı olarak da, seçilen bu karar değişkenlerinin farklı değerleri için çok sayıda senaryo otomatik olarak üretilip bunların analizi yoluna gidilmiştir.

Yapay zekâ yaklaşımı problemin kullanıcıdan elde edilmesi, modelin oluşturulması, çıktılar üzerinde sabit durum analizi yapılması, alternatif senaryoların üretilmesi ve sonuçların değerlendirilmesi konuları üzerinde yoğunlaştırılmıştır. Sistem, kullanıcının daha önce analiz etmiş olduğu veriler ile çalışmaktadır. Yani girdi verilerinin analizi, sistem dışında bırakılmıştır.

6.1. Tezin Bilime Katkısı

Bu tez çalışması ile öncelikle yapay zekâ/uzman sistem teknikleri ve simülasyonun birleştirilmesine yönelik olarak daha önce yapılmış önemli ve temel taşı niteliğindeki çalışmalar derlenerek, yapay zekâ ile bütünleşik bir simülasyon ortamının nasıl tasarlanması gerektiği açıklanmıştır.

Ortaya çıkarılan yaklaşımlardan yola çıkılarak kavşaklardaki trafik ışıklarının simülasyon ile optimizasyonu için yapay zekâ ile bütünleşik bir simülasyon ortamı önerilmiş ve böyle bir sistemin geliştirme aşamaları ortaya konularak bir prototip sistem elde edilmiştir.

6.2. Gelecekteki Araştırmalar İçin Öneriler

Yapay zekâ ile simülasyonun birleştirilmesi fikri ortaya atıldığından beri bu alanda çok sayıda çalışma yapılmış ve yeni fikirler ortaya atılmıştır. Fakat yine de yapay zekâ ile tamamen bütünleşik ve öğrenme yapılarını da içeren simülasyon sistemleri, ticari paketlerde çok fazla ortaya çıkmamıştır. Bu konu ile ilgili çalışmalar halen sürmektedir.

Yapay zekâ alanında ortaya çıkan yapay sinir ağları ve bulanık mantık teknikleriyle yapısal programlamaya bir alternatif olarak ortaya çıkan nesneye yönelik programlama yaklaşımlarının da simülasyon yöntemiyle entegrasyonu oldukça ilginç çalışmalar ortaya çıkaracaktır. Nesneye yönelik programlama, yapay zekâ ve simülasyonun birleştirilmesiyle ilgili çalışmalar yapılmaktadır.

Ayrıca üzerinde çalışmaların sürdüğü diğer bir konu da, paralel simülasyondur. Paralel mantıkla çalışan çok işlemcili bilgisayarların son derece yüksek çalışma hızları sayesinde, oldukça karmaşık ve çok büyük sistemlerin simülasyonu da çok hızlı bir şekilde yapılabilmektedir. Buradaki yaklaşım, bir işi küçük parçalara bölmek ve her işin her parçasını ayrı bir işlemciye yüklemek ve bunlar arasındaki iletişimi sağlamaktır. Paralel sistemlerin de diğer konular ve tekniklerle entegrasyonu çok hızlı simülasyon çözümleri sağlayacaktır.

Ayrıca geliştirilen prototip model de gelişmeye açıktır. Araç hareketlerinde hız formülü kullanılabilir ve geliştirilen yaklaşım, ardışık kavşaklardaki trafik ışıklarının senkronizasyonu için de kullanıldığında ve bu konu da modele dahil edildiğinde sistemin çözüm yapabileceği saha oldukça genişlemiş olur. Bu ve yukarıda sayılan konular, teknikler ve bunların entegrasyonları üzerinde ilginç çalışmalar yapılabilir.

KAYNAKLAR

Agrawala, A.K., (1977), Machine Recognition of Patterns, IEEE Press, New York.

Akhun,M., Ayağ,Z., ve Ertay,T., (1992), "Üretim Sistemlerinde Benzetim ve Uzman Sistemler", Benzetimde İlerlemeler'92, 6-7 Temmuz, Boğaziçi Üniversitesi, İstanbul.

Akın,H.L., (1995), "Yapay Zekâda Vücut ve Beyin Problemi", Bilgi İşleyen Makina Olan Beyin, 16-17 Ekim, Boğaziçi Üniversitesi, İstanbul.

Alpaydın,E., (1995), "Yapay Zekâ/Doğal Bilgisayarlar", Bilgi İşleyen Makina Olan Beyin, Bozağiçi Üniversitesi, 16-17 Kasım, İstanbul.

Ayağ, Z., Dinçmen, M., Durmuşoğlu, M.B. ve Ertay,T., (1991), "Üretim Sistemlerinde Benzetim Amaçlı Uzman Sistemler", Sanayide Bilgisayar Kullanımı ve Otomasyonu 90-91, ITÜ-KOSGEB.

Balcı,O., (1987), "Guidelines for Successful Simulation Studies", Technical Report TR-85-2, department of Computer Science, VPT&SU, Blacksburg, VA.

Balcı,O., (1992), "Guidelines for Successful Simulation Studies", Proceedings of the Advances in Simulation'92 Symposium, Boğaziçi University, July 6-7, İstanbul.

Banks,J. ve Carson, J.S., (1984), Discrete-Event System Simulation, Prentice-Hall, Inc., New York.

Bayazıt, E.N. ve Kavaklı, M., (1992a), "Uzman Sistem Metotlar-1" Bilgisayar Magazin,14: 60-66.

Bayazıt, E.N. ve Kavaklı, M.,(1992b), "Uzman Sistem Metotlar-2", Bilgisayar Magazin, 15: 57-74.

Bolte,J.P., Fisher,J.A. ve Ernst,D.H., (1993), "An Object-Oriented, Message-Based Environment for Integrating Continous, Event-Driven and Knowledge-Based Simulation", Proceedings: Application of Advanced Information Technologies: Effective Management of Natural Resources, ASAE, June 18-19, Spokane, WA.

Brazier, M.K. ve Shannon, R.E., (1987), "Automatic Programming of AGVS Simulation Models",. In Proceedings of the 1987 Winter Simulation Conference, Atlanta, Georgia:703-708.

Brazier,M.K. ve Shannon,R.E., (1987), "Automatic Programming of AGVs Simulation Models", Proceedings of the 1987 Winter Simulation Conference, Atlanta, Georgia:703-708.

Cengiz,Y.B., (1989), Yöneylem Araştırması, Basılmamış Eser, Y.T.Ü., İstanbul.

Conway, R.W., (1962), Some Tactical Problems in Simulation Methods, Memo RM-3244-PR, Rand Corporation, October 1962.

Cook, T.M. ve Russel, R.A., (1981), Introduction To Management Science, Prentice Hall Inc., New York.

Cremer,M. ve Meibner,F., (1993), "Traffic Prediction and Optimization Using an Efficient Macroscopic Simulation Tool", Proceeding of the 1993 Europan Simulation Multi Conference.

Delmar, D., (1982), Operational and Industrial Management, McGraw Hill Co., New York.

Dinçmen,M. ve Araz,T., (1992), "Atölye Tipi Üretimin Uzman Sistemlerle Benzetimi", Benzetimde İlerlemeler'92, 6-7 Temmuz, Boğaziçi Üniversitesi, İstanbul.

Doukidis, C.I. ve Paul, R.J., (1985), " Research into Expert Systems to Aid Simulation Model Formulation ", J. Opr. Res. Soc.,36 (4) :319-325.

Doukidis,G.I. ve Paul,R.J., (1990), "A Survey of the Application of Artifical Intelligence Techniques within the OR Society", Artificial Intelligence in Operational Research, ed. Georgios I. Doukidis and Ray J. Paul, 9-21, 1992, Macmillan, London.

Erkut, H., (1991), Yönetimde Simülasyon Yaklaşımı, İrfan Yayıncılık, İstanbul.

Feigenbaum, E.A. ve McCorduct, P., (1983), The Fifth Generation, Reading, Addision Wesley, Massachusetts.

Ford,D.R., Schroer,B.J., (1987), "An Expert Manufacturing Simulation System", Simulation, 48(5): 193-200.

Gantz,D.T. ve Mekenson,J.R., (1990), "Flow Profile Companson of a Microscopic Car-Following Model and A Macroscopic Platoon Dispension Model for Traffic Simulation", Proceeding of the 1990 Winter Simulation Conference.

Garnham, A., 1988, Artificial Intelligence, Routledge&Kagan Paul Ltd., London.

Genesereth, M. R., (1987), Logical Foundations of Artificial Intelligence, Los Altos, CA: Morgan Kaufmann Publishers.

Haddock, J., (1988), "A Simulation Generator for Flexible Manufacturing Systems Design and Control", IIE Transactions, 20(1): 22-31.

Haddock, J. ve Davis, R.P., (1985), "Building a Simulation Generator for Manufacturing Cell Design and Control ", In Annual International Industrial Engineering Spring Conference Proceeding, Los Angales, California:237-244.

Haddock, J., (1987), " An Expert System Framework Based On a Simulation Generator", Simulation, 48(2): 45-53.

Halaç,O., (1982), İşletmelerde Simülasyon Teknikleri, İstanbul Üniversitesi, İstanbul.

Heidorn, G.E., (1974), "English as a Very High Level Language for Simulation Programming", SIGPLAN Notice, 9: 91-100.

Heidorn,G.E., (1974), "English as a Very High Level Language for Simulation Programming", SIGPLAN Notices, 9:91-100.

Hill, D.R.C., (1996), Object-Oriented Analysis and Simulation, Addison-Wesley, Harlow, England.

Hillier, F.S. ve Lieberman, G.J., (1980), Introduction to Operations Research, 3rd.Ed., Holden-Day, New York.

Hilliges,M., Reiner,R. ve Weidlinch,W., (1993), "A Simulation Model of Dynamics Traffic Flow in Networks", Proceedings of the 1993 Euorapen Simulation Multi Conference.

Ignizio, J.P., (1991), Introduction to Expert Systems, McGraw Hill, New York.

James, M., 1984, Artificial Intelligence in BASIC, Butterworth&Co Ltd., Kent, England.

Javor, A., (1995), "Petri Nets and AI in Modelling and Simulation", Matematics and Computers in Simulation, 39:477-484.

Kaiser,R. ve Beamariage,T.G., (1997), "Conceptual Design of an Artifical Intelligence Architecture for Decision Making in Manufacturing Simulation", Proceedings of Object-Oriented Simulation Conference'97, January 12-15, Phoenix, Arizona.

Khoshnevis, B. ve Chen, A.P., (1986), "An Expert Simulation Model Builder ", Intelligent Simulation Environment, 17: 129-132.

Kim,J., Funk,K.H. ve Fichter,E.F., (1988), "Towards an Expert System for FMS Scheduling: A Knowledge Acquisition Environment", Expert System and Intelligent Manufacturing, Micheal D.Oliff, Editor:215-234.

König R. ve Langbein, R., (1992), "PROROAD Simulation of the Consequences of Traffic Management Strategies on the Road Traffic", Proceedings of the 1992 Europan Simulation System.

Kusiak,A. ve Heragu,S.S., (1988), "Knowledge Based System Guides Machine Layout in Flexible Manufacturing System", Industrial Engineering:48-53.

Lapin, L.L., (1991), Quantitative Methods For Business Decision, Harcourt Brace Jovanovich Inc.

Law,A.M. ve Kelton,W.D., (1991), Simulation Modelling&Analysis, McGraw-Hill, NY.

Liu, K., (1997), "Trafic Simulation", http://www.wh2.tru-dredem.de/~robert/beleg-e.htm.

Lu,S.C.Y. ve Tcheng;D.K., (1991), "Integration of Simulation Learning and Optimization to Support Engineering Design", Annuals of the CIRP, 40: 143-146.

Manvannan,S. ve Pegden,J.D., (1990), "A Rule Based Simulator for Modelling Just-In-Time Manufacturing Systems (JITSAI)", Simulation:109-117.

Mellicamph,J.M. ve Wahap,A.E.A., (1987), "Process Planning Simulation on FMS Modelling Tool for Engineers", Simulation, 49(5): 201-208.

Mellichamp, J.M. ve Park,Y.W., (1989), "A Statistical Expert System for Simulation Analysis", Simulation,52(4): 134-139.

Mellichamp,J.M., Venkatachalam,A.R., (1990), "An Interactive Debugging Expert System for GPSS/H Simulation Models", Simulation, 51:175-178.

Miche,D., (1980), " Knowlegde-Based Systems ", University of Illinois at Urbana-Champaign, Report 80-1001.

Murray, K.J. ve Sheppard, S.V., (1988), "Knowledge-Based Simulation Model Specification", Simulations, 50(3): ll2-119.

Nance, R.E., (1981), "Model Representation in Discrete Event Simulation: The Conical Methodogy", Technical Report CS81003-R, Department of Computer Science, VPI&SU, Blacksburg, VA.

Naylor,T.H. ve Gattis, D., (1976) "Corporate Plannig Models", California Management Review, :69-78.

O'Keefe,R.M., Roach,J.W., (1987), "Artificial Intelligence Approaches to Simulation", Journal of Operational Research Society, 48 (8): 713-722.

O'Keefe, R., (1986), "Simulation and Expert Systems - A Taxonomy and Some Examples", Simulation, 46 (1):l0-16.

Ören,T.I., (1997), "Simulation-as It has been and Should be", Simulation, 29(5):182-183.

Patterson,D.W., (1990), Introduction to Artificial Intelligence&Expert Systems, Prentice Hall, Englewood Cliffs, New Jersey.

Paul,R.J. ve Chew,S.T., (1987), "Simulation Modelling Using an Interactive Simulation Program Generator", Journal of Production Research Society, 38(8): 735-752.

Payne, E.C. ve McArthur, R.C., (1990), Developling Expert Systems, John Wiley & Sons, New York.

Penrose,R., (1997), Bilgisayar ve Zekâ, Çev:T. Dereli, Tubitak, Popüler Bilim Kitapları Nurol Matbaacılık, Ankara.

Pinson,L.J. ve Wrener,R.S., (1988), An Introduction to Object Oriented Programming and Smalltalk, Addison-Wesley Publishing Company, Reading, MA.

Raczynski,S., (1990), "Graphical Description and a Program Generator for Queueing Models", Simulation, 48(5): 147-151.

Reddy, R. (1987), "Epistemology of Knowledge Based Simulation", Simulation, 48(4):162-166.

Rogers,P. ve Williams,D.J., (1988), "On-Line Scheduling of Machining Cells Using Knowledge-Based Simulation", Proceedings of the 4th International Conference on Simulation in Manufacturing, London.

Russel,S. ve Norving,P., (1995), Artificial Intelligence A Modern Approach, Prentice-Hall, Englewood Cliffs, NJ.

Sargeant R., 1990, Expert Systems are Modifying 1992 Manufacturing Sector Strategies, Proc. 1st Conf. AI and Expert Systems in Manufacturing: 25-37.

Schildt, H., (1987), Advanced Turbo PROLOG, McGraw-Hill, Berkeley, New York.

Schmidt,J,W, ve Taylor, R.E., (1970), Simulation and Analysis of Industrial Systems, Irwin, Homewood.

Schroer, B.J. ve Tseng, F.T., (1989), "An Intelligent Assistant for Manufacturing System Simulation", International Journal of Production Research, 27 (10): 1665-1683.

Shannon, R.E., Mayer, R. ve Adelsberger, H.H., (1985), "Expert Systems and Simulation ", Simulations, 44 (6): 215-284.

Shannon, R.E., Mayer,R. ve Adelsberger,H.H., (1985), "Expert System and Simulation", Simulation, 47:275-284.

Shannon,R.E.,(1975), Systems Simulation: The Art and Science, Prentice-Hall, Englewood Cliffs, NJ.

Shearn,D.C.S., (1990), PASSIM-A Pascal Discrete Event Simulation Program Generator", Simulation, 51:

Shivnan,J. ve Browne,J., (1986), "Process Planning Simulation on FMS Modelling Tool for Engineers", Simulation, 48(5): 201-208.

Sztrimbely,M.W. ve Weymounth,P.J., (1991), "Dynamic Process Plant Simulation and Scheduling:An Expert System Approach", Simulation, 52:175-178.

Szymankiewicz,J., Donald,J. ve Turner,K., (1988), Solving Business Problems By Simulation, McGraw Hill, New York.

Turban, E., (1990), Decision Support and Expert Systems, MacMillan Publishing Company, New York.

Turing, A.M., (1950), "Computing Machinery and Intelligence", Mind, 59:433-460.

Ülgen,O.M., Thomasma,T., (1990), "SMARTSIM:An Object Oriented Simulation Program Generator for Manufacturing Systems", International Journal of Production Research, 28(9): 1713-1730.

Watson, H.J. ve Blackstone Jr, J.H., (1989), Computer Simulation, John Wiley & Sons, New York.

Welch,P.D., (1981), "On the Problem of the Initial Transient in the Steady-State Simulation", IBM Watson Research Center, Yorktown Heights, N.Y.

Welch,P.D., (1983), The Statistical Analysis of Simulation Results in the Computer Performance Modelling Handbook, S.S.Lavenberg, ed. 268-328, Academic Press, N.Y.

Zeingler,B.P. ve Ören,T.I., (1979), "Concept for Advanced Simulation Methodologies", Simulation, 32 (3):9-11.

Zhang, S.X., Schroer, B.J. ve Tseng, F.T., (1989), "Automatic Programming Approach to Simulating Prelaunch Activities", Simulations, 59(2) : 23-29.

Zuylen,V.H.J., (1994), "Knowledge Based Support of Users of Numerical Programs", Mathematics and Computers in Simulation, 36:327-336.

EKLER

EK 1

Geliştirilen yazılımın, veri girişi, dağılımlara uygun rassal veri üretimi, vb. prosedür ve fonksiyonlarından örnekler

```
function Duzgun(A,B:real):real;
        Var     XX:real;
Begin
   Rassal:=Random;
   XX:=A+Rassal*(B-A);
   Duzgun:=XX;
End;

function NORMAL(N_ORT,StdSAPM:real):real;
Var     Sum,X,Z      :Real;
       Sayac        :Integer;
Begin
   Sum:=0;
   for Sayac:=1 to 12 Do
      Begin
        Rassal:=RANDOM;
        Sum:=Sum+Rassal;
      End;
   Z:=Sum-6.0;
   X:=N_ORT+Z*StdSAPM;
   NORMAL:=X;
End;

function Poisson(P_ORT:real):Real;
Var     Total,Int,JJ:Real;
       J            :Integer;
Begin
   J:=0;
   total:=0;
   While (Total<=1) Do
   Begin
     Rassal:=RANDOM;
     Int:=-Ln(1.0-Rassal)/P_ORT;
     Total:=Total+Int;
     J:=J+1;
   End;
   JJ:=(60/J);
   Poisson:=JJ;
End;

function Ussel(U_ORT:real):real;
```

```
Var     TT:real;
 Begin;
    RASSAL:=Random;
    TT:=(-1)*(U_ORT)*Ln(Rassal);
    Ussel:=TT;
 End;


Procedure VeriGir;
Var tip,K1,K2,C1:integer;

Function DagTipi(Yon:integer):integer;
Var S:integer;
Begin
ClrScr;
(*Cerceve;*)
Repeat
TextColor(LightMagenta);
GotoXY(5,5);Write(Yon,'.');
TextColor(LightCyan);
Write(' YONDEKI GELIS TIPI');
TextColor(LightBlue);
GotoXY(5,7);Write('[1] POISSON Dagilimi');
GotoXY(5,8);Write('[2] USSEL Dagilim');
GotoXY(5,9);Write('[3] DUZGUN Dagilim');
GotoXY(5,10);Write('[4] NORMAL Dagilim');
GotoXY(5,11);Write('[5] SABIT Deger');
TextColor(LightCyan);
GotoXY(5,13);Write('Dagilim Tipini Giriniz....:');
Read(s);
Until s in [1,2,3,4,5];
DagTipi:=s;
End;


Begin
ClrScr;
(*Cerceve;*)
GotoXY(5,3);
Write('Sim□lasyonu yapilacak kavsagin adini giriniz.....:');
Read(KAV_ADI);
GotoXY(5,5);
Write(KAV_ADI,' kavsagina gelis sayisini giriniz...:');
Read(GELIS_ADEDI);
For K1:=1 to GELIS_ADEDI Do
        Begin
        Tip:=Dagtipi(K1);
        DAG_TIPI[K1]:=Tip;
```

```
        Case Tip of
        1:Begin;
          GotoXY(5,15);
          Write('Poisson Dagilimina ait ortalamayi giriniz..:');
          Read(DAG_PRMT[K1,1]);
          End;
        2:Begin;
          GotoXY(5,15);
          Write('Ussel dagilima ait ortalamayi giriniz....:');
          Read(DAG_PRMT[K1,1]);
          End;
        3:Begin;
          GotoXY(5,15);
          Write('Duzgun dagilimin minimum degerini giriniz...:');
          Read(DAG_PRMT[K1,1]);
          GotoXY(5,16);
          Write('Duzgun dagilimin maksimum degerini giriniz...:');
          Read(DAG_PRMT[K1,2]);
          End;
        4:Begin;
          GotoXY(5,15);
          Write('Normal Dagilimin ortalamasini giriniz...:');
          Read(DAG_PRMT[K1,1]);
          GotoXY(5,16);
          Write('Normal Dagilimin standart sapmasini giriniz...:');
          Read(DAG_PRMT[K1,2]);
          End;
        5:Begin;
          GotoXY(5,15);
          Write('Sabit degeri giriniz.....:');
          Read(DAG_PRMT[K1,1]);
          End;
        End;(*case*)
      End;

ClrScr;
(*Cerceve;*)
GotoXY(5,3);
Write(KAV_ADI,' kavsagindaki toplam yol sayisini giriniz....:');
Read(YOL_ADEDI);
For K1:=1 to YOL_ADEDI do
        Begin;
        GotoXY(5,4+K1);
        Write(K1,'. yolun serit sayisini giriniz....:');
        Read(SERIT[K1]);
        End;
ClrScr;
Cerceve;
```

```
C1:=0;
for K1:=1 to GELIS_ADEDI do
 begin;
 GotoXY(5,1+K1+C1);
 Write(K1,' .yonden gelen araclar kac farkl yola donebiliyorlar.:');
 Read(DONUS_SA[K1]);
 for K2:=1 to DONUS_SA[K1] do
 begin;
 C1:=C1+1;
 GotoXY(5,1+K1+C1);
 Write(K1,' .yoldan araclar n sapt g ',K2,' .yolun numaras n giriniz..:');
 Read(YOL_ROTA[K1,K2]);
 end;
 end;

C1:=0;
ClrScr;
(*Cerceve;*)
for K1:=1 to GELIS_ADEDI do
for K2:=1 to DONUS_SA[K1] do
 begin;
 C1:=C1+1;
 GotoXY(5,C1+1);
 Write(K1,' .yonden gelen araclar n ',YOL_ROTA[K1,K2],' .yola sapma olas l g n
giriniz..:');
 Read(YOL_OLS[K1,K2]);
 end;

C1:=0;
ClrScr;
(*Cerceve;*)
for K1:=1 to GELIS_ADEDI do
for K2:=1 to DONUS_SA[K1] do
 begin;
C1:=C1+1;
GotoXY(5,C1+1);
Write(K1,' .yonden gelen araclar n ',YOL_ROTA[K1,K2],' .yola gecis suresi...:');
Read(KGS[K1,K2]);
if K2=1 then MAX_KGS[K1]:=KGS[K1,K2]
      else if KGS[K1,K2]>MAX_KGS[K1] then MAX_KGS[K1]:=KGS[K1,K2];
 end;

End;(*VeriGir*)
```

EK 2
Dört yol kavşağının veri girişi

Simülasyonu yapilacak kavsagin adini giriniz.....:DORTYOL1

DORTYOL1 kavsagina gelis sayisini giriniz...:4

1. YONDEKI GELIS TIPI

[1] POISSON Dagilimi
[2] USSEL Dagilim
[3] DUZGUN Dagilim
[4] NORMAL Dagilim
[5] SABIT Deger

Dagilim Tipini Giriniz....:2

Ussel dagilima ait ortalamayi giriniz....:10

2. YONDEKI GELIS TIPI

[1] POISSON Dagilimi
[2] USSEL Dagilim
[3] DUZGUN Dagilim
[4] NORMAL Dagilim
[5] SABIT Deger

Dagilim Tipini Giriniz....:2

Ussel dagilima ait ortalamayi giriniz....:8

3. YONDEKI GELIS TIPI

[1] POISSON Dagilimi
[2] USSEL Dagilim
[3] DUZGUN Dagilim
[4] NORMAL Dagilim
[5] SABIT Deger

Dagilim Tipini Giriniz....:4

Normal Dagilimin ortalamasini giriniz...:9
Normal Dagilimin standart sapmasini giriniz...:3.5

4. YONDEKI GELIS TIPI

[1] POISSON Dagilimi
[2] USSEL Dagilim
[3] DUZGUN Dagilim
[4] NORMAL Dagilim
[5] SABIT Deger

Dagilim Tipini Giriniz....:2

Ussel dagilima ait ortalamayi giriniz....:5

```
DORTYOL1 kavsagindaki toplam yol sayisini giriniz....:8

1. yolun serit sayisini giriniz....:1
2. yolun serit sayisini giriniz....:2
3. yolun serit sayisini giriniz....:1
4. yolun serit sayisini giriniz....:2
5. yolun serit sayisini giriniz....:1
6. yolun serit sayisini giriniz....:2
7. yolun serit sayisini giriniz....:1
8. yolun serit sayisini giriniz....:2
```

```
1 .yonden gelen araclar kac farklı yola donebiliyorlar.:3
1 .yoldan aracların saptıgı 1 .yolun numarasını giriniz..:6
1 .yoldan aracların saptıgı 2 .yolun numarasını giriniz..:7
1 .yoldan aracların saptıgı 3 .yolun numarasını giriniz..:8
2 .yonden gelen araclar kac farklı yola donebiliyorlar.:3
2 .yoldan aracların saptıgı 1 .yolun numarasını giriniz..:7
2 .yoldan aracların saptıgı 2 .yolun numarasını giriniz..:8
2 .yoldan aracların saptıgı 3 .yolun numarasını giriniz..:5
3 .yonden gelen araclar kac farklı yola donebiliyorlar.:3
3 .yoldan aracların saptıgı 1 .yolun numarasını giriniz..:8
3 .yoldan aracların saptıgı 2 .yolun numarasını giriniz..:5
3 .yoldan aracların saptıgı 3 .yolun numarasını giriniz..:6
4 .yonden gelen araclar kac farklı yola donebiliyorlar.:3
4 .yoldan aracların saptıgı 1 .yolun numarasını giriniz..:5
4 .yoldan aracların saptıgı 2 .yolun numarasını giriniz..:6
4 .yoldan aracların saptıgı 3 .yolun numarasını giriniz..:7
```

1 .yonden gelen aracların 6 .yola sapma olasılıgını giriniz..:.3
1 .yonden gelen aracların 7 .yola sapma olasılıgını giriniz..:.3
1 .yonden gelen aracların 8 .yola sapma olasılıgını giriniz..:.4
2 .yonden gelen aracların 7 .yola sapma olasılıgını giriniz..:.3
2 .yonden gelen aracların 8 .yola sapma olasılıgını giriniz..:.2
2 .yonden gelen aracların 5 .yola sapma olasılıgını giriniz..:.5
3 .yonden gelen aracların 8 .yola sapma olasılıgını giriniz..:.7
3 .yonden gelen aracların 5 .yola sapma olasılıgını giriniz..:.2
3 .yonden gelen aracların 6 .yola sapma olasılıgını giriniz..:.1
4 .yonden gelen aracların 5 .yola sapma olasılıgını giriniz..:.4
4 .yonden gelen aracların 6 .yola sapma olasılıgını giriniz..:.2
4 .yonden gelen aracların 7 .yola sapma olasılıgını giriniz..:.4

1 .yonden gelen aracların 6 .yola gecis suresi...:2
1 .yonden gelen aracların 7 .yola gecis suresi...:3
1 .yonden gelen aracların 8 .yola gecis suresi...:4
2 .yonden gelen aracların 7 .yola gecis suresi...:2
2 .yonden gelen aracların 8 .yola gecis suresi...:3
2 .yonden gelen aracların 5 .yola gecis suresi...:3
3 .yonden gelen aracların 8 .yola gecis suresi...:2
3 .yonden gelen aracların 5 .yola gecis suresi...:2
3 .yonden gelen aracların 6 .yola gecis suresi...:3
4 .yonden gelen aracların 5 .yola gecis suresi...:2
4 .yonden gelen aracların 6 .yola gecis suresi...:2
4 .yonden gelen aracların 7 .yola gecis suresi...:3

Simulasyon kosum suresini giriniz...:3600

EK 3
Geliştirilen program tarafından oluşturulan senaryolardan örnekler

Simülasyon Sonuçları

Kavsak Adi : DORTYOL-1 SENARYO 1

Cevrim Suresi : 37 saniye

	Gelisler 1.Yon	2.Yon	3.Yon	4.Yon
Yesil Isık Suresi	: 5	7	6	10
Kırmz Isık Suresi	: 32	30	31	27
Gecen arac Sayısı	: 96	407	192	688
Gelen arac Sayısı	: 331	420	402	693
Ort.Kuy.Bek.Sure.	: 1347.6	29.9	911.4	19.2
Ort.Kuy.Uzunlugu	: 124	7	106	8
Verim	: 0.29	0.97	0.48	0.99
Ortalm Sis. Suresi	: 1352.97	34.40	917.29	25.86

Kavsaktan gec Arac	:	1383
Kavsaga gelen Arac	:	1846

Simülasyon Sonuçları

Kavsak Adi : DORTYOL-1 SENARYO 2

Cevrim Suresi : 45 saniye

	Gelisler 1.Yon	2.Yon	3.Yon	4.Yon
Yesil Isık Suresi	: 9	7	9	10
Kırmz Isık Suresi	: 36	38	36	35
Gecen arac Sayısı	: 316	391	316	709
Gelen arac Sayısı	: 384	435	401	738
Ort.Kuy.Bek.Sure.	: 321.4	233.0	322.7	94.2
Ort.Kuy.Uzunlugu	: 37	31	40	25
Verim	: 0.82	0.90	0.79	0.96
Ortalm Sis. Suresi	: 329.36	237.77	330.36	101.43

Kavsaktan gec Arac	:	1732
Kavsaga gelen Arac	:	1958

Simülasyon Sonuçları

Kavsak Adi : DORTYOL-1 SENARYO 3

Cevrim Suresi : 53 saniye

	1.Yon	2.Yon	3.Yon	4.Yon
Yesil Isık Suresi	: 12	9	12	11
Kırmz Isık Suresi	: 41	44	41	42
Gecen arac Sayısı	: 326	450	335	720
Gelen arac Sayısı	: 371	458	393	730
Ort.Kuy.Bek.Sure.	: 182.9	29.4	224.5	59.2
Ort.Kuy.Uzunlugu	: 23	8	29	18
Verim	: 0.88	0.98	0.85	0.99
Ortalm Sis. Suresi	: 191.87	35.36	233.05	67.37

Kavsaktan gec Arac: 1831
Kavsaga gelen Arac: 1952

Simülasyon Sonuçları

Kavsak Adi : DORTYOL-1 SENARYO 8

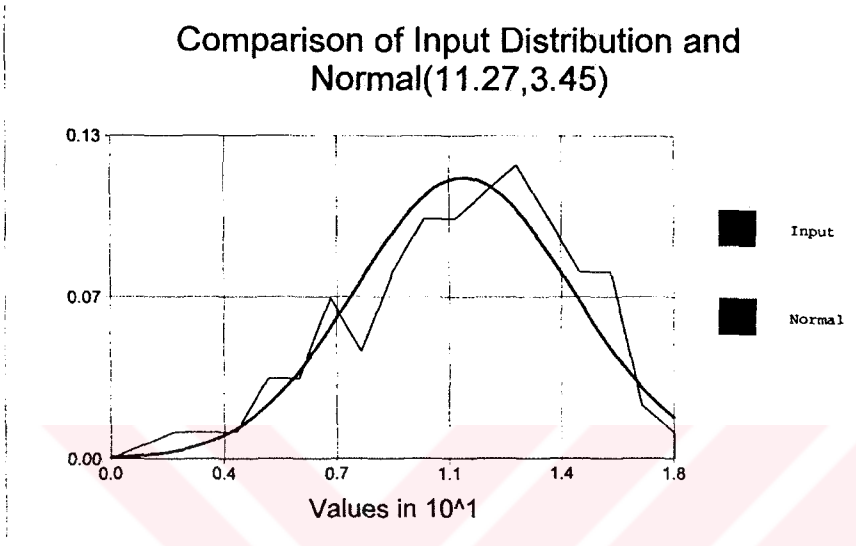
Cevrim Suresi : 93 saniye

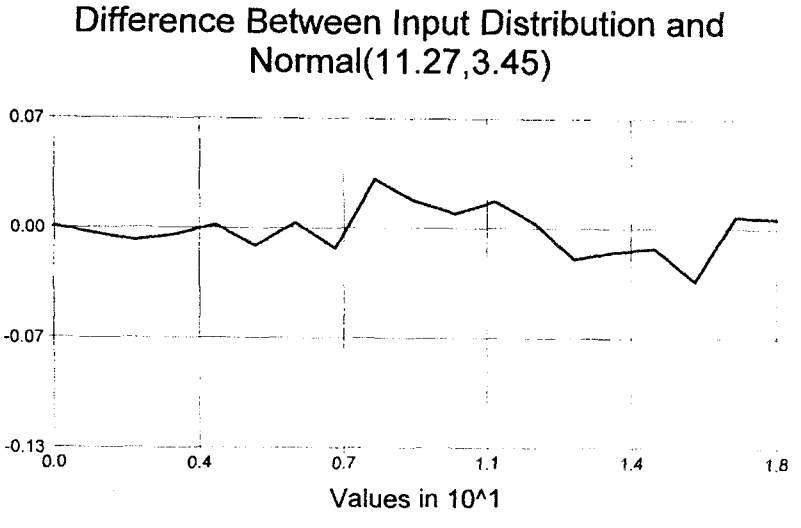
	1.Yon	2.Yon	3.Yon	4.Yon
Yesil Isık Suresi	: 17	19	22	26
Kırmz Isık Suresi	: 76	74	71	67
Gecen arac Sayısı	: 304	415	396	681
Gelen arac Sayısı	: 356	422	407	694
Ort.Kuy.Bek.Sure.	: 313.1	38.1	48.2	37.0
Ort.Kuy.Uzunlugu	: 37	11	12	17
Verim	: 0.85	0.98	0.97	0.98
Ortalm Sis. Suresi	: 324.74	45.99	61.52	48.56

Kavsaktan gec Arac : 1796
Kavsaga gelen Arac : 1879

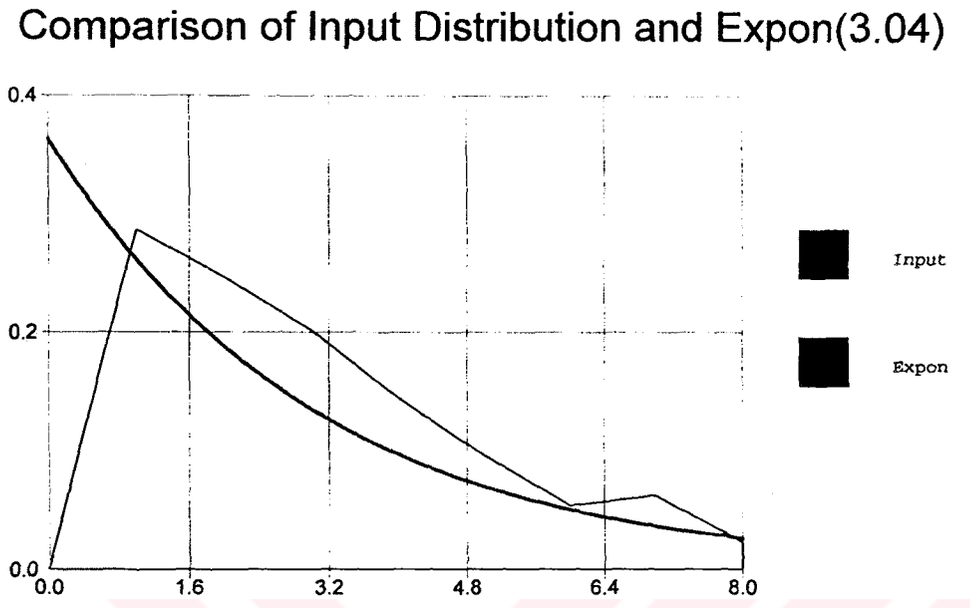
EK 4
Geliş yönü 1,2 ve 3 için BESTFIT programından elde edilen sonuçlar

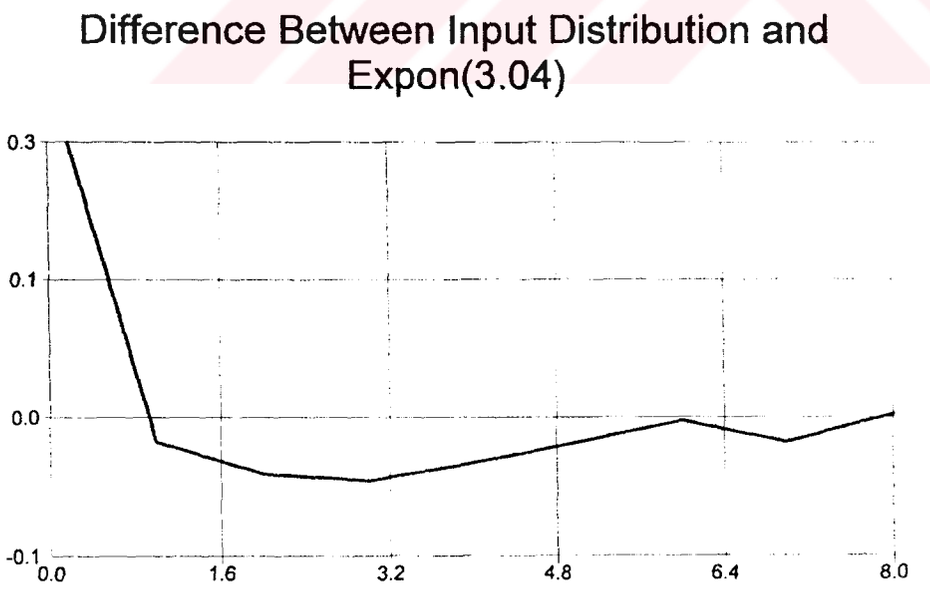
1.Geliş Yönü için sonuçlar



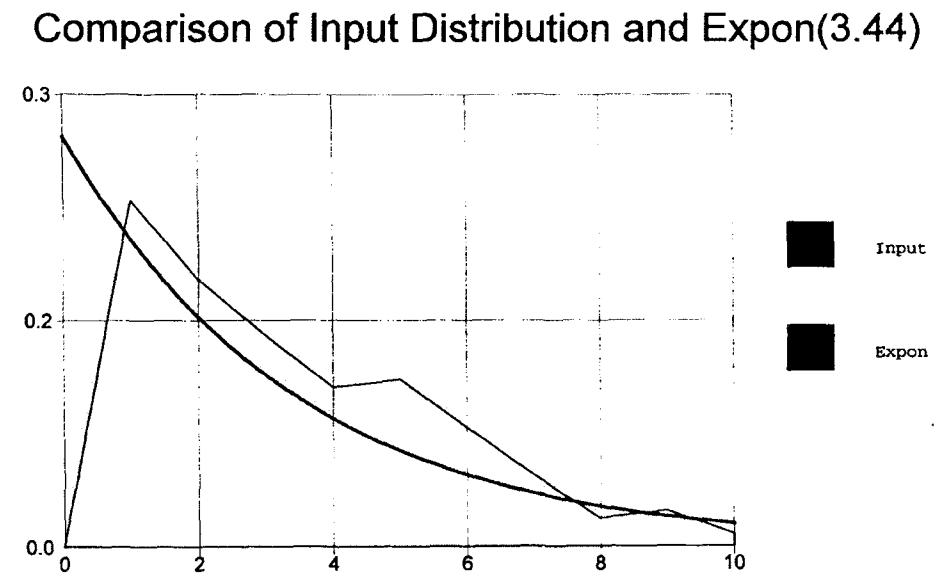


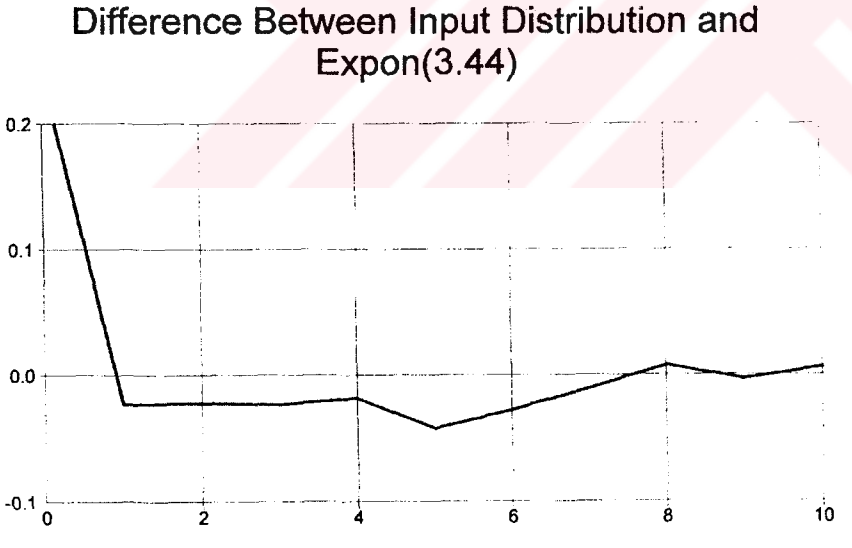
2.Geliş Yönü için sonuçlar





3.Geliş Yönü için sonuçlar





EK 5

Gerçek şartlardaki uygulamanın veri girişi

Simülasyonu yapilacak kavsagin adini giriniz.....:UCYOL

UCYOL kavsagina gelis sayisini giriniz...:3

1. YONDEKI GELIS TIPI

[1] POISSON Dagilimi
[2] USSEL Dagilim
[3] DUZGUN Dagilim
[4] NORMAL Dagilim
[5] SABIT Deger

Dagilim Tipini Giriniz.....:4

Normal Dagilimin ortalamasini giriniz...:11.27
Normal Dagilimin standart sapmasini giriniz....:3.45

2. YONDEKI GELIS TIPI

[1] POISSON Dagilimi
[2] USSEL Dagilim
[3] DUZGUN Dagilim
[4] NORMAL Dagilim
[5] SABIT Deger

Dagilim Tipini Giriniz....:2

Ussel dagilima ait ortalamayi giriniz....:3.04

3. YONDEKI GELIS TIPI

[1] POISSON Dagilimi
[2] USSEL Dagilim
[3] DUZGUN Dagilim
[4] NORMAL Dagilim
[5] SABIT Deger

Dagilim Tipini Giriniz....:2

Ussel dagilima ait ortalamayi giriniz....:3.44

```
UCYOL kavsagindaki toplam yol sayisini giriniz....:5

1. yolun serit sayisini giriniz....:2
2. yolun serit sayisini giriniz....:2
3. yolun serit sayisini giriniz....:2
4. yolun serit sayisini giriniz....:2
5. yolun serit sayisini giriniz....:2
```

```
1 .yonden gelen araclar kac farklı yola donebiliyorlar.:2
1 .yoldan aracların saptıgı 1 .yolun numarasını giriniz..:4
1 .yoldan aracların saptıgı 2 .yolun numarasını giriniz..:5
2 .yonden gelen araclar kac farklı yola donebiliyorlar.:1
2 .yoldan aracların saptıgı 1 .yolun numarasını giriniz..:5
3 .yonden gelen araclar kac farklı yola donebiliyorlar.:1
3 .yoldan aracların saptıgı 1 .yolun numarasını giriniz..:4
```

1 .yonden gelen aracların 4 .yola sapma olasılığını giriniz..:0.25
1 .yonden gelen aracların 5 .yola sapma olasılığını giriniz..:0.75
2 .yonden gelen aracların 5 .yola sapma olasılığını giriniz..:1.0
3 .yonden gelen aracların 4 .yola sapma olasılığını giriniz..:1.0

1 .yonden gelen aracların 4 .yola gecis suresi...:3
1 .yonden gelen aracların 5 .yola gecis suresi...:4
2 .yonden gelen aracların 5 .yola gecis suresi...:2
3 .yonden gelen aracların 4 .yola gecis suresi...:2

ÖZGEÇMİŞ

Doğum tarihi	06.02.1966	
Doğum yeri	İstanbul	
Lise	1980-1983	Kabataş Erkek Lisesi
Lisans	1984-1989	Yıldız Üniversitesi Mühendislik Fak. Endüstri Müh. Böl.
Yüksek Lisans	1990-1993	Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Endüstri Müh. Anabilim Dalı, Endüstri Müh. Programı
Doktora	1993-1998	Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Endüstri Müh. Anabilim Dalı, Endüstri Müh. Programı
Çalıştığı kurum	1990-Devam ediyor	Yıldız Teknik Üniversitesi Makina Fak. Endüstri Mühendisliği Bölümü Araştırma Görevlisi