T.C
İSTANBUL AYDIN ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

SUDOKU PROBLEMİNİN ALGORİTMA TASARIMINA YAPAY ZEKA DESTEKLİ YENİ BİR YAKLAŞIM MODELİ VE UYGULAMASI

YÜKSEK LİSANS TEZİ

Hazırlayan

Tuğba KARAOĞLU

Tez Danışmanı

Doç. Dr. Ahmet BABANLI

İSTANBUL-2012

T.C
İSTANBUL AYDIN ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

SUDOKU PROBLEMİNİN ALGORİTMA TASARIMINA YAPAY ZEKA DESTEKLİ YENİ BİR YAKLAŞIM MODELİ VE UYGULAMASI

YÜKSEK LİSANS TEZİ

Tuğba KARAOĞLU
Y1013.010008

Tez Danışmanı
Doç. Dr. Ahmet BABANLI

İSTANBUL-2012

KABUL VE ONAY BELGESİ

Doç. Dr. Ahmet BABANLI'nın danışmanlığında **Tuğba KARAOĞLU** tarafından hazırlanan **"Sudoku Probleminin Algoritma Tasarımına Yapay Zeka Destekli Yeni Bir Yaklaşım Modeli ve Uygulaması"** adlı bu çalışma, jürimiz tarafından İstanbul Aydın Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalında Yüksek Lisans Tezi olarak kabul edilmiştir.

…../…../2012

JURİ

Danışman: Doç. Dr. Ahmet BABANLI …….

Üye : Prof. Dr. Ali GÜNEŞ …….

Üye : Yard. Doç. Metin ZONTUL …….

Tezin Savunulduğu Tarih: 9/10/2012

Bu tez çalışması İstanbul Aydın Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun ..... tarih ve ..... sayılı kararı ile onaylanmıştır.

Fen Bilimleri Enstitüsü Müdürü

ÖNSÖZ

Makinelerin karşılaştıkları ve daha önceden görmemiş oldukları sorunları çözmesini amaçlayan Yapay Zeka, ilk kez 1956 yılının yaz aylarında Darmouth Üniversitesi'nde yapılan bir konferansta ortaya çıkmıştır. İlk yıllarında çok hızlı bir ivme yakalayan ve pek çok insanı hayrete düşüren yapay zeka programları, ilerleyen yıllarda aynı ivmeyi ne yazık ki koruyamamıştır. Bunun nedeni, insanların kolaylıkla çözdüğü sorunları bir makineye anlatmanın zorluğu ve makinelerin o zamanki işlem gücü kısıtlarıydı. Ne var ki, son yirmi yıldır çok yüksek bir hızla ilerleyen teknoloji, karmaşık yapay zekaya olanak sağlayacak donanımların üretilmesini sağlamış ve modern yapay zekanın ilerlemesinde büyük katkıda bulunmuştur. Günümüzde, en kolay ve en etkin olarak programlanabilen aygıtlar bilgisayarlar olduğu için yapay zeka bir bilgisayar bilimi olmuştur ve halen pek çok uygulamada etkin olarak kullanılmaktadır.

TEŞEKKÜR

Çalışmalarımı yönlendiren, araştırmalarımın her aşamasında bilgi, öneri ve yardımlarını esirgemeyerek akademik ortamda olduğu kadar beşeri ilişkilerde de engin fikirleriyle yetişme ve gelişmeme katkıda bulunan danışman hocam sayın Doç. Dr. Ahmet BABANLI'ya teşekkür ederim.

Bu çalışmam süresince bana her zaman destek olarak, tecrübeleriyle bana yol gösteren değerli hocalarım Prof. Dr. Ali GÜNEŞ'e ve Yard. Dç. Metin ZONTUL'a teşekkür ederim.

Çalışmalarım süresince birçok fedakârlıklar göstererek beni destekleyen biricik eşim Şükrü KARAOĞLU ve kızım Ayşe Eylül KARAOĞLU'na en derin duygularla şükranlarımı sunarım.

İÇİNDEKİLER

KISALTMALAR

DFS: Derinlik Öncelikli Arama (Depth First Search)

YZ: Yapay Zeka

ŞEKİLLER LİSTESİ

GİRİŞ

Bilgisayar ve Video Oyunları pazarı son yıllarda giderek ivmelenen bir gelişme göstermeye başlamıştır. Bunun temel nedenlerinden birisi gelişmiş oyunların artık sadece belirli oyun salonlarından çıkıp masaüstü sistemlerin ve kullanılan özelleşmiş grafik hızlandırıcıları ile hemen her alanda karşımıza çıkmaya başlamasıdır. Hemen her yıl ikiye katlanan işlemci ve grafik işleyici yongaların hızı oyun konusundaki rekabeti körüklemiştir. Özellikle son yıllarda yeni pazarlar arayışına giren dev şirketlerin oyun programlama ve özelleşmiş oyun konsolu tasarımına ağırlık vermeleri de pazardaki rekabetin ve potansiyelin açık bir göstergesidir. [1]

Yıllar geçtikçe işlem gücünün artması sadece oyunların daha hızlı çalışmasını değil oyunların biçimlerini de değiştirmeye başlamıştır. Önceleri sadece hamle tabanlı (satranç, dama ya da basit zeka oyunları) ve düşük kaliteli iki boyutlu karakterlerin tekdüze bir mantığa göre hareketlerine dayanan oyunlar mevcutken günümüz bilgisayarlarında gerçek zamanlı ve neredeyse gerçeğe yakın görünümde ortamlarda onlarca karakterin birbiri ile insan davranışına yakın hareketlerle etkileşimini sergilemektedir. Oyunların ve oyun şirketlerinin her geçen gün artması da kullanıcıların daha gerçekçi grafik ve daha akıllı oyun karakterlerine olan isteğini arttırmıştır. Bu noktada ise oyun programanın en zor ve en etkileyici yönlerinden yapay zeka devreye girer. [1]

Oyun programlamada kullanılan yapay zekanın amacı oyuncu ile etkileşimde bulunan karakterlerin ya da oyunun geçtiği ortamın mümkün olduğunca gerçek insan, topluluk ve dünya (ya da kimyasal- fiziksel - biyolojik olarak anlamlı ortam) yaşam ortamına benzetilmeye çalışılmasıdır. Başka bir anlamda da bilgisayarın yetenekli bir oyuncuya aynı derecede yetenekli karşılık vermesi, oyuncunun kurduğu planlara benzeyen planlar ya da tuzaklar kurması, gerektiğinde oyuncuyu zor duruma düşürüp onu yenebilmesi oyun programındaki yapay zeka kalitesini belirler. [1]

Bu tez çalışmasında karara dayalı ve tam bilgili oyun grubuna giren Sudoku Oyunu ele alınmıştır.

Bölüm I'de doğal zeka ve yapay zeka tanımı, yapay zekanın amaçları, kısaca tarihçesi, yapay zekanın araştırma ve uygulama alanlarına yer verilmiştir.

Bölüm II'de Yapay Zekanın araştırma ve uygulama alanlarından biri olan Oyun kavramı üzerinde durulmuş, Oyun teorisinden bahsedilmiştir. Ayrıca bilgisayar oyunları niteliklerine göre gruplara ayrılmış ve oyunlar, stratejileri gereği algoritma karmaşıklığına göre sınıflandırılmıştır. Böylece oyunların birbirleriyle gerek durum uzayı karmaşıklığı gerekse oyun ağacı karmaşıklığı açısından kıyaslanmasına imkan verilmiştir.

Bölüm III'de bilgisayar oyunlarını konu edinen benzer tez çalışmalarına yer verilmiş ve gerekli görülen yerler Sudoku oyunu ile kıyaslanmıştır. Sodoku oyunu anlatılmış, oyun kurallarına ve arayüzüne yer verilmiştir. Bölümde yapay zekada kullanılan arama yöntemleri sınıflandırılarak, bu tez sudoku probleminde genellikle kullanılan derinlik öncelikli arama metodu anlatılmış, bu metodun 8-puzzle probleminde ve mevcut Sudoku problemlerinde nasıl kullanıldığı, problemlerin durum uzayı oluşturulup, buradan hareketle arama ağaçları çizilerek örneklendirmeye çalışılmıştır. Son olarak Sudoku problemi yeni bir algoritma modeli ile çözümlenmeye çalışılmış, modele ait fonksiyonlara ve genel program akışına yer verilmiştir.

Bölüm IV'te ise yapılan bu çalışmanın Sonuç kısmına yer verilmiştir.

BÖLÜM I

1.1 YAPAY ZEKA NEDİR?

Yapay zeka konusuna girmeden önce zekanın kısa bir tanımını yapmak yerinde olacaktır. Zeka, insanın düşünme, akıl yürütme, nesnel gerçekleri algılama, kavrama, yargılama, sonuç çıkarma, soyutlama, öğrenme yeteneklerinin tümüdür. Ayrıca Soyutlama, öğrenme ve yeni durumlara uyma gibi yetenekler de zeka kapsamı içindedir. (Nabiyev, 2003).

Zeka, bilgi alma ve onu gerektiğinde kullanabilme, çeşitli bilgi parçaları ile ilişki kurabilme ve tüm bu parçaları birleştirerek sonuca ulaşabilme kabiliyetidir. Bir insan parçaları birleştirme işinde ne kadar başarılı olursa o kadar zeki olarak nitelendirilmektedir.

Yapay zeka ise, bu özelliklere sahip organik olmayan sistemlerdeki zekadır. Yapay zeka kabaca; bir bilgisayarın ya da bilgisayar denetimli bir makinenin, genellikle insana özgü nitelikler olduğu varsayılan akıl yürütme, anlam çıkartma, genelleme ve geçmiş deneyimlerden öğrenme gibi yüksek zihinsel süreçlere ilişkin görevleri yerine getirme yeteneği olarak tanımlanmaktadır. (Nabiyev, 2003).

Yapay Zeka hakkında kesin hatlarla belirlenmiş bir tanım yapmak mümkün olmamakla birlikte kısaca genel birtakım tanımlarda bulunabiliriz. Genel tanıma göre yapay zeka, us olarak adlandırılan insan beyni fonksiyonlarının yapay simülasyonlarla bilgisayarlarda gerçekleştirilmesidir (Üstkan, 2007).

Yapay zeka, us olarak adlandırdığımız insan beyni fonksiyonunun yapay simülasyonunun bilgisayarlarla gerçekleştirilmesidir. Daha geniş anlamda, genel olarak akıllı davranış olarak ifade edilebilecek hesaplayıcı anlamayı (computational understanding) ve buna bağlı mekanizmalar (robotlar, konuşan bilgisayarlar gibi) üretmek için bilim ve mühendisliğin bir alanını ifade etmek üzere kullanılan terimdir (Adlassnig, 2002: 1-2).

Yapay zeka araştırmalarının tanınmış isimlerinden D. Lenat ve E. Feigenbaum (1987) zekayı problem çözme açısından ‘arama alanı’ kavramı üzerinden şu şekilde tarif etmektedirler: “Zeka, karmaşık bir problemi, çözüm arama alanını daraltarak

kısa yoldan çözebilme kabiliyetidir." Feigenbaum (1989) daha sonra zekayı 'bilgi kullanımı' kavramına bağlı olarak şöyle tarif etmiştir: "Zeka, karmaşık bir problemi çözmek için gerekli bilgileri toplayıp birleştirebilme kabiliyetidir."

Bunların dışında zekayı sistem tanıma açısından şöyle tarif edebiliriz: Zeka, daha önce düzensiz sanılan bir sistemdeki düzenliliği ve düzenli olduğu kabul edilen bir sistemdeki düzensizlikleri fark edebilme kabiliyetidir. (Kocabaş, 2006)

Yapay zeka, insanlarda zeka ile ilgili zihinsel fonksiyonları bilgisayar modelleri yardımıyla inceleyip bunları formel hale getirdikten sonra yapay sistemlere uygulamayı amaçlayan bir araştırma alanıdır. "Yapay zeka" terimi ilk olarak önemli yapay zeka programlama dillerinden biri olan LISP'i geliştiren ve yapay zeka alanındaki öncülerden biri olan John McCarthy tarafından 1956 yılında ortaya atıldı (Russell & Norvig, 1995, s. 17-18).

Yapay Zeka, insanlar tarafından yapıldığında zeka gerektiren şeyleri gerçekleştiren makineler yapma bilimidir. (Minsky, 1968)

Yapay Zeka, zeki davranışları taklit eden bilgisayar programlarının yapımı boyunca insan zekasının doğasını anlamayı amaç edinen disiplindir. (Bonnet, 1885)

"İnsan beynine benzeyen bir makinenin yapılabilmesi için 300 trilyon dolardan fazla para gerekmektedir. Böyle bir makinenin çalışabilmesi için ise 1 trilyon watt'lık elektrik enerjisine ihtiyaç vardır." (Dr.V.Grey Walter)

Yapay zeka dört kategoriye ayrılabilmektedir (Russel, 2003):

- İnsan gibi düşünen sistemler
- İnsan gibi davranan sistemler.
- Mantıklı düşünen sistemler.
- Mantıklı davranan sistemler.

İnsan gibi davranan sistemler

Yapay zeka araştırmacılarının baştan beri ulaşmak istediği ideal, insan gibi davranan sistemler üretmektir. Turing zeki davranışı, bir sorgulayıcı kandıracak kadar bütün bilişsel görevlerde insan düzeyinde başarım göstermek olarak tanımlamıştır[Turing 50]. Bunu ölçmek için de Turing Testi olarak bilinen bir test önermiştir. Turing testinde denek sorgulayıcıyla bir terminal aracılığıyla haberleşir.

Eğer sorgulayıcı deneğin insan mı yoksa bir bilgisayar mı olduğunu anlayamazsa denek Turing testini geçmiş sayılır. Turing, testini tanımlarken zeka için bir insanın fiziksel benzetiminin gereksiz olduğunu düşündüğü için sorgulayıcıyla bilgisayar arasında doğrudan fiziksel temasdan söz etmekten kaçınmıştır. Burada vurgulanması gereken nokta, 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 biliminin araştırma alanına girmektedir. Bu çalışmalarda asıl amaç genellikle insanın düşünme süreçlerini çözümlemede bilgisayar modellerini bir araç olarak kullanmaktır.

Rasyonel düşünen sistemler

Bu sistemlerin temelinde mantık yer alır. Burada amaç çözülmesi istenen sorunu mantıksal bir gösterimle betimledikten sonra çıkarım kurallarını kullanarak çözümünü bulmaktır. YZ'de çok önemli bir yer tutan mantıkçı gelenek zeki sistemler üretmek için bu çeşit programlar üretmeyi amaçlamaktadır. Bu yaklaşımı kulllanarak gerçek sorunları çözmeye çalışınca iki önemli engel karşımıza çıkmaktadır. Mantık formel bir dil kullanır. 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 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.

Yapay zekanın temelleri bir çok farklı alanlardan beslenmektedir. Felsefe, Matematik, Algoritma, Ekonomi, Psikoloji, Bilgisayar Mühendisliği, Sinir bilimleri, Kontrol teorisi ve Sibernetik ve Dilbilim başlıcaları olarak sayılabilmektedir. (Russel, 2003).

1.2 YAPAY ZEKANIN AMAÇLARI

Yapay zeka alanında yapılan çalışmalarda amaçları şöyle sıralayabiliriz:

1) İnsan beyninin fonksiyonlarını bilgisayar modelleri yardımıyla anlamaya çalışmak.

2) İnsanların sahip olduğu zihinsel yetenekleri, bilgi kazanma, öğrenme ve buluş yapmada uyguladıkları strateji, metot ve teknikleri araştırmak.

3) Bu öğrenme metotlarını formel hale getirmek ve bilgisayarlarda bilgi sistemleri halinde uygulamak.

4) İnsanlarını bilgisayar kullanımını kolaylaştıracak insan/bilgisayar ara birimleri geliştirmek.

5) Belli bir uzmanlık alanı içindeki bilgileri bir ‘bilgi sistemi’ (veya ‘uzman sistem’) halinde toplamak.

6) Geleceğin bilgi toplumunun kurulmasında önemli rol oynayacak ‘genel bilgi sistemleri’ geliştirmek.

7) Yapay zeka iş yardımcıları ve ‘zeki robot timleri’ geliştirmek.

8) Bilimsel araştırma ve buluşlarda faydalanmak üzere, ‘araştırma yardımcıları’ geliştirmek.

1.3 YAPAY ZEKANIN KISA KRONOLOJİK TARİHÇESİ

1943 – McCulloch&Pitts: Beynin Boolean devre modeli

1950 – Turing'in "Bilgi işleyen makineler ve zeka"

1956- Dartmaouth Görüşmesi: "Yapay Zeka" ismi ortaya atıldı.

1952-1969 – IBM satranç oynayabilen ilk programı yazdı. YZ konusundaki ilk uluslar arası konferans düzenlendi.

1950'ler – İlk YZ programları, Samuel'in kontrol edici programı, Netwell ve Simon'ın mantık teorisi, Gelernter'ın geometrik motoru.

1965 – Robinson'un mantıklı düşünme için geliştirdiği tam bir algoritma

1966-73 – YZ hesapsal karmaşayla karşılaşır. Sinir ağları araştırmaları hemen hemen kaybolur.

1969-79 – Bilgiye dayalı sistemlerin ilk gelişme adımları

1980 – YZ Endüstri haline gelir.

1986 – Yapay sinir ağları tekrar popüler oldu.

1987 – YZ bilim haline geldi.

1995 – Zeki Ajanlar ortaya çıkar.

1997 – Deep Blue Kasparov'u yendi.

1998 – İnternetin yaygınlaşması ile YZ tabanlı birçok program geniş kitlelere ulaştı.

2000-- Robot oyuncaklar piyasaya sürüldü. Halen birçok elektronik cihazda YZ uygulamaları kullanılmaktadır.

1.4 BİLGİSAYARLAR DÜŞÜNEBİLİR Mİ?

Yapay Zeka kavramının geçmişi modern bilgisayar bilimi kadar eskidir. Fikir babası, "Makineler düşünebilir mi? " sorusunu ortaya atarak Makine Zekasını tartışmaya açan Alan Mathison Turing'dir. 1943 yılında İkinci dünya savaşı sırasında Kripto Analizi gereksinimleri ile üretilen Elektro-Mekanik cihazlar sayesinde Bilgisayar Bilimi ve Yapay Zeka kavramları doğmuştur.

İnsanlar, kendi varlığının bilincindedir. Çevrelerini kendi benlerinin deneyimi ile algılamaktadırlar. Kendi beninin bilincine sahip insan, ayrıca evrende kendi benleri olmayan nesneler ve kendi benleri olan başkaları ile birlikte yaşadıklarını da bilirler. Bütün bunlar arasındaki farkları algılarlar. Nesneleri ele alıp, boyutları, ağırlıkları ile tanımlayabilir ve incelerler. Diğer insanlara ise nesneler olarak yaklaşmazlar. Etkinliklerini onların da ayrı birer benlerinin oldu unu bilerek ayarlar; böylece başkalarıyla birlikte yaşadıkları bir dünyanın farkına varırlar. Bütün bunlar bedenimizin algılayıcılar (sensors) aracı ile yapar ve dil aracı ile içselleştirirler. Dolayısıyla, us bedenden bağımsız ve soyut bir varlık olmak yerine, bedenin doğrudan bir fonksiyonu olmalıdır. Bu durumda yapay zekaya sahip bir sistem geliştirildiğinde bunun, insan benzeri bir bedene sahip yapının içine gömülmesi (embodying) gerekmektedir (Mingers, 2001: 108-109).

Pekiyi, ister sadece çok gelişmiş algoritmalara dayalı bir bilgisayar programının, isterse de kendi beninin bilincine sahip robot, android veya benzeri bir yap da çevresini algılayabilen bir makinenin geliştirildiği varsayılsın; bu sistemin gerçekten zeki olduğunu nasıl anlayacağız? (Baştan, 2003: 192-193).

1.4.1 TURİNG TESTİ

1950 yılında yayınladığı Computing Machinery and Intelligence isimli ünlü makalesinde Alan Turing, tasarımlanmış bir sistemin gerçekten zeki olup olmadığını sorgulayabilecek bir test önermektedir. Testini öyle betimlemektedir: Bir sorgulayıcı bir monitör ve klavye aracı ile düşünebildğii ileri sürülen bir bilgisayara ve gönüllü bir insana sorular yöneltecektir. Hem insan, hem de bilgisayar fiziksel bir engelin arkasında olacaktır. İetişim sesli olarak kurulmayacaktır. Karşı tarafların cevaplar yine monitör aracı ile sorgulayana gönderilecektir. Sorgulayıcı, karşı taraftan gelen cevaplar değerlendirerek, normal insan reaksiyonlarından farklı, yapay bir karşılık yakalamaya çalışacaktır. Sorgulayıcı, hangi cevapların gönüllüye, hangilerinin de makineye ait olduğunu ayırt edemezse, makine zeki olarak kabul edilecektir. Bu teste daha sonralar Turing Testi adı verilmiştir (Nolfi, Floreano, 2001: 500).

Tabii kar tarafa çok miktarda, sürekli ve gerçek anlama yetene i gerektiren özgün sorular yöneltmek gerekmektedir. Böyle bir dizi seri sorgulamada sistemin gerçek anlama yeteneğinin olmadığı sonucunu yakalamak mümkündür. Sorgulayıcının becerisi, kısmen böyle özgün soru biçimleri bulabilmesine bağlıdır. Kısmen de derinlemesine bir analiz yapabilecek ve gerçek anlayışın varlığını ortaya çıkarabilecek şekilde birbiriyle bağlantılı sorular tasarlayabilmesinde gizlidir.

Örneğin;

Sorgulayıcı : Duydu uma göre, bu sabah bir gergedan pembe bir balonla Mississipi boyunca uçmuş. Buna ne dersin?

Bilgisayar: Oldukça gülünç geldi bana.

Bilgisayarın cevabı oldukça ihtiyatlıdır ve karşılık tatminkardır.

Sorgulayıcı: Sahi mi? Bir zamanlar amcam da aynı şeyi yapmıştı gitmiş ve dönmüştü. Yalnız onunki kirli-beyaz renkte ve çizgiliydi. Bunda gülünç olan ne var? (Baştan, 2003: 193)

Anlama yeteneği yoksa bu tip sorular karşısında bilgisayar eninde sonunda tuzağa düşecektir. Hafıza bankasında gergedanların kanatlarının olmadığına dair bir bilgi varsa, ilk soruyu Gergedanlar uçamaz diyerek, ikinci soruyu da Gergedanlar çizgili de ildir eklinde yanıtlayıp kendini ele verebilecektir (Penrose, 1998: 4-7).

Turing'in "Computing Machinery and Intelligence" makalesi şu cümleyle başlar: "Makineler düşünebilir mi?" Turing'in "Taklit oyunu" olarak adlandırdığı aşağıdaki yöntem bugün "Turing testi" olarak bilinmektedir.

Bir odada bulunan bir kadın ve erkek, kendilerinin kadın olduğuna bir sorgulayıcıyı ikna etmeye çalışırlar. Kadın sorulara doğru cevap verir, erkek ise "Onu dinleme, gerçek kadın benim" gibi cevaplar yazar. Turing, eğer erkeğin oynadığı rolü bir makine başarıyla gerçekleştirseydi, buna ne derdik, diye sorar. Eğer taklit oyununda bu rolü başarıyla oynayabilecek makineler yapabilseydik, bunlar düşünen makineler olmaz mıydı? Test sunulup tartışıldıktan sonra Turing şu yorumu yapar: İlk baştaki "Makineler düşünebilir mi?" sorusunun tartışmaya değmeyecek kadar anlamsız olduğuna inanıyorum. Bununla birlikte, inanıyorum ki, yüzyılın sonunda, sözcüklerin kullanımı ve genel eğitimli görüşleri, çelişkili olduğu sanısına kapılmadan düşünen makinelerden söz etmeyi mümkün kılacak denli değişmiş olacak. (Gültekin, 2006)

1.4.2 ÇİN ODASI TESTİ

California üniversitesinden John SEARLE bilgisayarların düşünemediğini göstermek için bir düşünce deneyi tasarlamıştır. Bir odada kilitli olduğunuzu düşünün ve odada da üzerlerinde çince tabelalar bulunan sepetler olsun. Fakat siz Çince bilmiyorsunuz. Ama elinizde Çince tabelaları İngilizce olarak açıklayan bir kural kitabı var. Kurallar Çinceyi tamamen biçimsel olarak, yani söz dizemlerine uygun olarak açıklamaktadır. Daha sonra odaya başka Çince simgelerin getirildiğini ve size Çince simgeleri odanın dışına götürmek için, başka kurallar da verildiğini varsayın. Odaya getirilen ve sizin tarafınızdan bilinmeyen simgelerin oda dışındakilerce 'soru` diye, sizin oda dışına götürmeniz istenen simgelerin ise 'soruların yanıtları` diye adlandırıldığını düşünün. Siz kilitli odanın içinde kendi simgelerinizi karıştırıyorsunuz ve gelen Çince simgelere yanıt olarak en uygun Çince simgeleri dışarı veriyorsunuz. Dışta bulunan bir gözlemcinin bakış açısından sanki Çince anlayan bir insan gibisiniz. Çince anlamanız için en uygun bir program bile Çince anlamanızı sağlamıyorsa, o zaman herhangi bir sayısal bilgisayarın da Çince anlaması olanaklı değildir. Bilgisayarda da sizde olduğu gibi, açıklanmamış Çince simgeleri işleten bir biçimsel program vardır ve bir dili anlamak demek, bir takım

biçimsel simgeleri bilmek demek değil, akıl durumlarına sahip olmak demektir. (Pirim, 2005)

Bu düşünce deneyi açıkça Turing testini hatırlatacak şekilde oluşturulmuştur. Searle'ye göre bir bilgisayar ucu açık sorulan insandan ayırt edilemeyecek şekilde yanıtlasa bile onun sorulan anlayıp anlamadığı ayrı bir tartışma konusu olacaktır. Bu düşünce deneyinde Searle CPU'yu, talimatlar kitabı ise programın kendisini temsil etmektedir. Böylece bilgisayarların gerçek zekaya sahip olmadan, *görünürde* zeki davranış sergileyebilecekleri ileri sürülmüştür. (Gültekin, 2006)

Yapay zeka alanında çalışanların çoğu, bu düşünce deneyine çeşitli yanıtlar verdiler. Bu deneye verilen bir yanıt ise, Çin odasının gerçek dünyada gerçek bir robota dönüştürülmesi olmuştur. Çin odası deneyine karşı güçlü bir yanıt: Anlamayı, anlama dizgesinin hiçbir parçasında bulamayacak oluşumuzdur. Çince anlayan bir dizgenin parçalarına baktığımızda anlama faaliyetini yürüten hiçbir parça göremeyiz. Ne de olsa ana dili Çince olan birinin beynini çıkarıp sökersek, anlamayla ilgili tek bir parça bile yoktur. Aynı şekilde anlama işlevini odanın hangi parçasının yürüttüğünü sormak anlamsız olmalıdır. Odaya istediğiniz Çince soruyu sorabilir ve bu sorunuza cevap alabilirsiniz. Anlama işlevini odanın hangi parçasının gerçekleştirdiğini bulup bulmamamız ne kadar önemlidir? Yapay zeka taraftarlarının Çin odası deneyine verdiği en güçlü yanıtlardan birisi budur. Yapay zeka taraftarları, anlama kavramını insan merkezli yaklaşımdan kurtarmak ve anlama sözcüğünün kullanımını biraz değiştirmek gerektiğini savunmaktalar. (Gültekin, 2006)

1.5 YAPAY ZEKANIN ARAŞTIRMA VE UYGULAMA ALANLARI

Yapay zeka araştırma alanlarını şöyle sıralayabiliriz: **Oyunlar,** Otomatik Teorem İspatlama, Doğal Dil Anlama ve Çeviri, Şekil Tanıma, Robotik, Bilgi Tabanlı Sistemler, Makina Öğrenmesi, Makina Buluşları, Bilimsel Buluşların Modellendirilmesi

1.6.1 *Oyunların Modellenmesi*

Satranç, dama, tavla gibi oyunlar araştırmacılar için yapay zekanın ilk zamanlarından bu yana tercih edilen bir alan olmuştur. İlk sistemler kısıtlı bir zamanda çok sayıdaki çözüm yolunu göz önünde bulundurma becerisi üzerine kurulmuştur. Bu çalışmalarda artık tecrübeye dayalı bilgi stratejileri kullanılarak genel çözüm arama kavramlarına dönülmektedir. Bu alanda yapılan çalışmalar daha sonra Artificial Life (AL -Yapay Hayat) adında yeni bir araştırma alanının ortaya çıkmasına yol açmıştır. (Kocabaş,2006)

1.6.2 *Otomatik Teorem İspatlama*

Bu alandaki çalışmalar, yapay zekanın erken döneminde, 1950'lerde başlamıştır. Özellikle sembolik mantıkta ispatlanan teoremlerin daha basit ispat yollarının bulunmasında kayda değer sonuçlar elde etmiştir. Bu çalışmalardan, sembolik mantığa dayanan güçlü bir yapay zeka dili olan Prolog programlama dili ortaya çıkmıştır. Günümüzde paralel Prolog derleyicileri geliştirme çalışmaları devam etmektedir. (Kocabaş, 2006)

1.6.3 *Bilgi Tabanlı Sistemler*

Bu alandaki çalışmaları üç alt başlık altında değerlendirebiliriz: Bilgi Gösterimi, Uzman Sistemler, Bilgi Tabanlı Simülasyon, ve Genel Bilgi Sistemleri.

Bilgi Gösterimi: Bilgi sistemlerinde bilgi çok farklı şekillerde gösterilebilmektedir. Yapay zekada bilgi gösterim metotlarını üç seviyede sınıflandırıyoruz: Bilgi Düzeyi, Sembol Düzeyi, Aygıt Düzeyi. Bilgi düzeyi metotlarda bilgi, kurallar, mantık yapıları, çerçeveler, senaryolar ve vak'a kayıtları şeklinde gösterilmektedir. Sembol düzeyi metotlarda ise bilgi, vektörler ve matris yapıları içinde gösterilmektedir. Aygıt düzeyinde bilgi, bir ağ yapısı içinde gösterilir.

Uzman Sistemler: Tasarım, planlama, teşhis, özetleme, kontrol ve tavsiyede bulunma gibi konularda insan uzmanların yaptıkları tür faaliyetleri otomatik olarak uygulamak üzere geliştirilen bilgisayar programlarıdır.

Bilgi Tabanlı Simülasyon: Bu alanda kullanılmak üzere geliştirilen sistemler afet yönetimi, kriz yönetimi, stratejik planlama ve bazı askeri alanlarda uygulanmaktadır. (Kocabaş, Öztemel, Uludağ & Koç, 1996).

Genel Bilgi Sistemleri: Uzman sistemlerin en zayıf tarafı insan uzmanların sahip olduğu sağduyu bilgisine ve genel bilgilere sahip olmamasıdır. Bundan dolayı uzman sistemlere, insan uzmanların aksine kendi uzmanlık alanlarının biraz dışındaki problemler verildiği zaman ya hiçbir çözüm veremezler, yahut anlamsız bir çözüm verirler. Uzman sistemlerin bu eksikliğinin giderilmesi için insanların sahip oldukları genel bilgileri ve sağduyu bilgilerini de taşıyan genel bilgi sistemleri geliştirilmeye başlanmıştır.

1.6.4 ***Doğal Dil Anlama ve Çeviri:***

Bu alandaki çalışmalar, otomatik tercüme, doğal dilde yazılmış metinlerin açıklanması ve üretilmesi ve konuşmaların otomatik işlenmesi gibi faaliyetleri kapsar. Belli alanlardaki metinlerin tercümesi için geliştirilmiş sistemlerin tercümelerin doğruluk oranı yüksek (% 90'ın üzerinde) olabilmektedir. Fakat genel maksatlı otomatik tercüme sistemlerinde doğruluk oranı % 25'lere kadar düşebilmektedir (Kocabaş, 1996).

1.6.5 ***Örüntü Tanıma***

Görme bir makinenin çevresini fark etmeye yönelik başka bir özelliktedir. İşitme yoluyla algılama durumunda olduğu gibi görme probleminin basitleştirilmesi, basit şekillerin algoritmik şekil tanıma metotları yardımıyla tanımlanmasından ibarettir. Bunlar bir metin içindeki harfler, bir sahne resmi içindeki eşyalar, mikroskop altında bir resmin üzerindeki hücreler ya da kromozomlar, bir uydu resmi üzerindeki özel bölgeler olabilir. Bununla birlikte, bir sahnenin veya görüntünün gerçekten anlaşılması bu metotların ötesine gitmeyi gerektirir: Tıbbi teşhis amacı ile radyolojik görüntülerin açıklanması, basılı ya da elyazısı bir metnin anlaşılması, üretim zinciri üzerindeki bir nesnenin kontrolünde olduğu gibi. Bu son etkinlikler

özellikle robotbilime aittir. Görüntülerin işlenmesi yapay zekanın endüstriyel alandaki ilk uygulamalarından biridir. (Kocabaş, 2007)

1.6.6 ***Robotik***

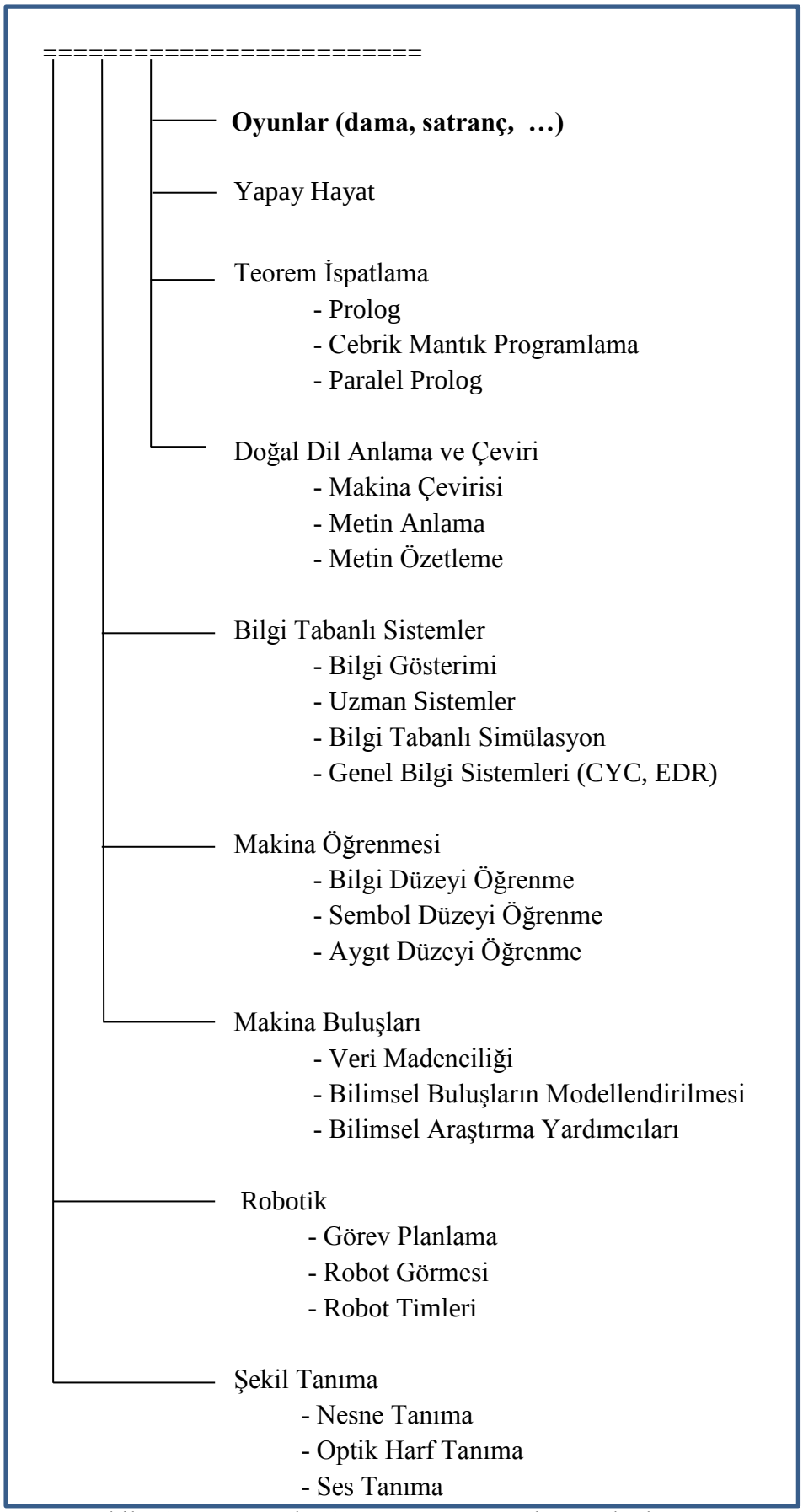
Robotbilim geniş bir alana yayılmış durumdadır. Özellikle sanayide iş otomasyonu etkinliklerini kapsar (fabrikasyon, yönetim, tamir). Varolan robotların büyük bölümü işleri sıra ile durmaksızın tekrarlar ve zeki davranış göstermezler. Buna karşın yeni kuşak robotlar, giderek çevrelerini algılamaya ve hareketlerini planlamaya yönelik zeka yeteneği ile donatılmaktadır. Robotik alanındaki çalışmalar, robot görmesi, görev planlama, robot timleri, mikro robotlar ve robot kolonileri üzerinde yoğunlaşmaktadır.

1.6.7 ***Makina Öğrenmesi***

Öğrenme üzerine yapılan çalışmalar yapay zeka çalışmalarının daha ilk dönemlerinde başlatılmıştır. İlk çalışmalar "algılayıcılar" (perceptrons) adı verilen, aygıt düzeyinde basit sistemler üzerinde başlamıştır. Daha sonra sembol düzeyi öğrenme metotları geliştirilmeye başlanmıştır. 1970'lerin sonlarına doğru bilgi tabanlı sistemlerin ortaya çıkmasıyla bilgi düzeyi öğrenme metotları geliştirilmeye başlanmıştır. (Kocabaş, 2006)

1.6.8 ***Makina Buluşları ve Veri Madenciliği***

Makina öğrenmesi üzerine yapılan çalışmalar daha sonra makina buluşları alanına doğru gelişmeye başlamıştır. Büyük veri tabanlarından indüktif, dedüktif ve analojik öğrenme metotlarıyla yeni bilgiler ortaya çıkarılmıştır. İlk olarak tıp veri tabanlarındaki ilaç uygulamaları ve bunların etkileri üzerinden yeni tedavi yolları ortaya çıkarılmıştır. Bu çalışmalar daha sonra yeni bilimsel araştırmalardan elde edilen veriler üzerinde de uygulanmaya başlamıştır. (Kocabaş, 2007)



Şekil I-1: Yapay Zeka Araştırma ve Uygulama Alanları

BÖLÜM II

2.1 OYUNLAR

İlk bilgisayarlı oyun programı 1961 yılında *Nolan Bushnell* (MIT - Massachusetts Institute of Technology) tarafından yazılan ve çok basit olan *Uzay Savaşı* (Space War)'dır. İlk bilgisayarlı oyun olan *Computer Space* ise Bushnell'in oluşturduğu ATARI şirketi tarafından, birkaç başarısızlıklardan sonra yalnız 1972 yılında piyasaya sürülebilmiştir. (Nabiyev, 2010)

Bilgisayarlı oyunlarda oyunun kendisinin oluşturulması en ilginç aşamalardandır. Borland'ın programcılık yarışması 1.sinin söylediği gibi *"oyunları oluşturdukça 2 kez mutluluk duymaktayım; programı yazan sırada ve oyuna oynamağa başladığımda...* Her halde oyun programı yazmayan bir programcıya rastlamak imkansızdır. Her ay yüzlerce oyun piyasaya sürülmektedir. Fakat bu oyunlardan yalnız bir kaçı tutunabilir. Mantığa dayalı stratejili oyunlar, günümüzde en güncel oyunlar olmamasına rağmen, kendisine her zaman ihtiyaç duyan sabit bir piyasa bulmaktadır ve tasarımcılarına dikkate değer kar saklamaktadır. Örneğin, "*Software Toolworks",* şirketinin ATARI, AMIGA ve MAC bilgisayarlarında ve WINDOWS ortamında çalışan ünlü "*Chess Master"* programı, henüz 1993 yılında, yani sonradan geliştirilmiş olan 'ChessMaster 4000 Turbo" versiyonu çıkmadan önce, her ay 4000 adet satmakta idi. Günümüzde de internet üzerinden oynanan MVC Chess programı veya cep telefonları için mantık oyunları çok ilgi çekmektedir. (Nabiyev, 2010)

Çeşitli oyunların en iyi bilgisayarlı stratejilerinin bulunması için farklı kuruluşlar tarafından ödüller belirlenmektedir. Örneğin Japonya, *Go* oyununu *Birinci Dan* seviyesinde oynayabilen program için 1.000.000 $ ödül belirlemiştir. Kurallarının basitliğine bakmayarak, uzmanlar durum uzayı çok büyük olduğu için en güçlü *Go* programının yalnız 2050'de tasarlanabileceğini düşünmektedirler. Bu tahminler, örneğin satranç için 2010 yılıdır. Yıllar arasındaki farklılıkların nedeni durum uzayının büyüklüğü ile ilgilidir. Eğer satrançta 5-6 derinlikte uygun algoritmalarla en iyi hamle seçilebilirse, Go oyununda bu derinlik yeterli olmamaktadır. Çünkü oyunun başlangıcında 19*19 = 361 hamle olmasına rağmen, oyunun ortalarında yaklaşık 200 durum söz konusudur (Satrançta ortalama 50). Go,

konuma bağlı bir oyundur. Günümüzde ise pozisyonlu oyunların bilgisayara eğitilmesi açık olan bir problemdir. İyi programların oluşturulması ise her pozisyona göre değişik değer fonksiyonu belirlemekle mümkün olabilmektedir. Örneğin, satrançta rakipler için piyonun değerini kaleye, vezirinkini ata eşitleyerek farklı durumlar elde edilebilir. Öte yandan pozisyonlu oyunlarda bu değerleri daha esnek tutarak en optimum değerler belirlenebilir. (Nabiyev, 2010)

Stratejili oyunlarda sezgisel fonksiyon değerlerinin belirlenmesi büyük önem taşımaktadır. Ancak daha sağlam değerlendirme için durum uzayının büyük kısmının optimum biçimde araştırılması gerekmektedir. Bu ise yalnız iyi programcı olmakla mümkün değildir. Bunun için matematik ve oyun teorisinin bilinmesi gerekir. (Nabiyev, 2010)

2.2 OYUN TEORİSİ

Oyun teorisinin temelleri 1928 yılında Macar asıllı matemetikçi John von Neumann'ın ispatladığı *min-max teoremi* ile atılmıştır. Atom bombasından bilgisayara kadar birçok icatlar ve araştırmalarda önemli katkısı olan Neumann, oyun teorisinin ekonomide uygulamalarını 1943 yılında iktisatçı Oscar Morgensten ile beraber yazdıkları " The Theory of Games and Ekonomic Behaviour" adlı kitapla gündeme getirmişti. Ayrıca genç bir matematikçi olan John Forbes Nash, Neumann'ın yaklaşımını genelleştirerek 1950 yılında çok sayılı oyuncunun olduğu oyunlarda *denge teoremini* ispatlamış ve 1994 yılında bu buluşu için Nobel Ödülü almıştı.

Bugün bildiğimiz anlamda oyun teorisi, yukarıda bahsettiğimiz bu iki teoreme dayanır.

2.2.1 Min-max Yöntemi

Min - Max algoritması esasında çok bilinen bir oyun stratejisi algoritmasıdır. Max ve Min iki kişilik oyunda taraflar olarak adlandırılır. Max ilk harekete başlayan taraftır, en sonunda oyun bittiğinde kazanan ödülü alırken kaybeden cezalandırılır. Herhangi bir oyunu bir çeşit arama problemi olarak tanımlarsak aşağıdaki maddelerle ifade edilebilir.

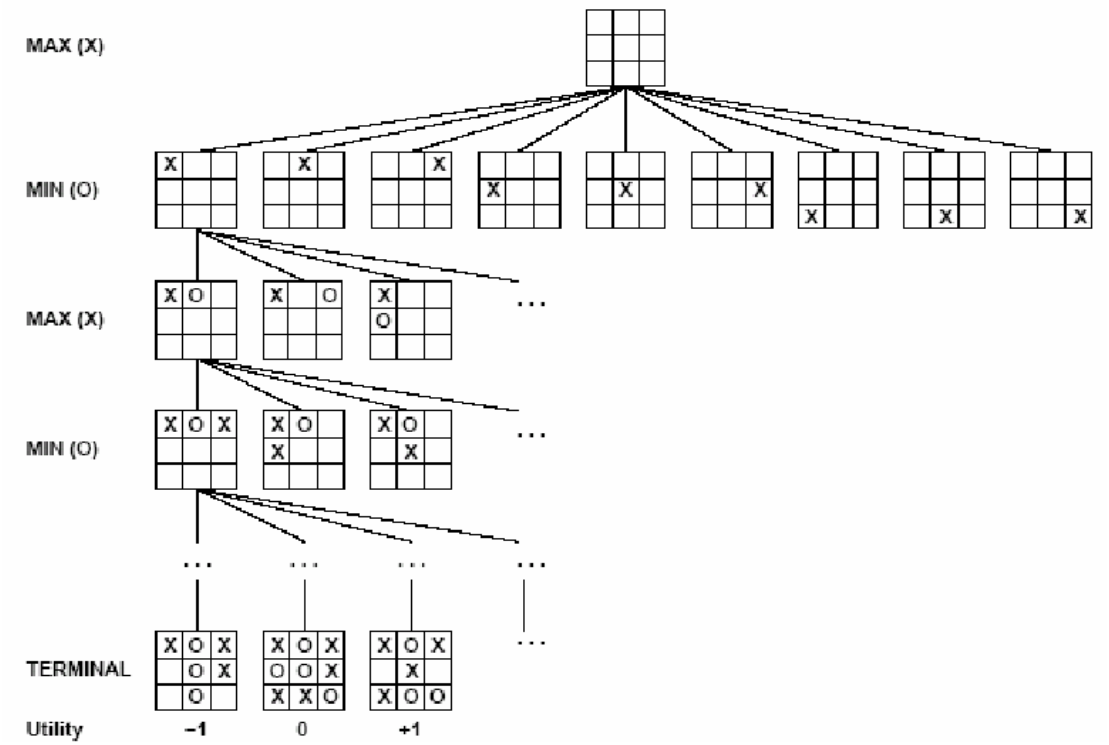
- Başlangıç Durumu: Boş Oyun Tahtası
- Başarı Fonksiyonu: Legal Hareketlerin Listesi
- Terminal Testi: Oyun Bitti mi?
- Başarı Fonksiyonu (Utility Function): Her bir terminale değer verilir galibiyet (+1) , mağlubiyet = (-1) , beraberlik (0)

Algoritmanın uygulandığı problemleri bir arama ağacı gibi düşünürsek, Her düğüme, MAX için bir kazanma ya da MIN için bir kazanma olup olmama durumuna, göre 1 ya da 0 verilecektir. Minmax işlemi ile bu değerler grafiğin yukarısına doğru ardışık ebeveyn düğümleri boyunca şöyle yayılacaktır. Ebeveyn

durum MAX düğümünde ise, buna çocukları arasından maksimum değer verilir Ebeveyn durum MIN düğümünde ise, çocuklarının minimum değeri verilir. Her bir duruma iliştirilmiş bir değer, bu oyuncunun başarmak için ümit ettiği en iyi durumun değeridir. Bu türetilmiş bir değerler bu olası hareketler içerisinden seçilir. Düğüm değeri yüksek olduğu sürece kazanma olasılığı artar. Değer küçüldüğü takdirde ise rakibin kazanma şansı artmış olur. Oyunculardan birisi sürekli olarak büyük değerleri diğeri ise küçük değerleri takip edecektir.

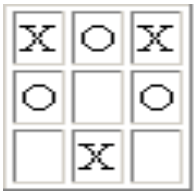
Başlangıç durumu ve legal hareketler her bir kenar için oyun ağacını tanımlar, Şekil II-2'de Üç Taş oyunu için oluşturulan oyun ağacını gösteriliyor. Başlangıç durumundan itibaren Max'in dokuz hamle şansı vardır. Max'in hareketini X ile Min'in hareketini O ile gösterirsek oyun her iki taraftan biri yaprak elemanlardan birine ulaşıp, kendilerine kazandıracak, yani kendi taşlarının üç tanesinin yatay dikey veya çapraz olarak arka arkaya gelmesiyle oluşacak terminal durumların oluşması ile oyun sona erer.

Max 'ın görüş açısına göre bakarsak, Yapraktaki yüksek değerler Max için atanırken düşük değerler Min için atanır. Max açısından baktığımız için, Max arama ağacını kontrol eder ve kendi için en uygun hamleyi seçip kazanımı en yükseğe çıkarırken Min 'in kaybını da en yükseğe çıkaracak hamleyi yapar. Normal bir arama probleminde hedef duruma ulaşmak işlemi sabit sıralı hareketlerden oluşur. Buna karşın bir oyunda rakibinizde hareket yapacağı için, Rakibin her hareketi için en uygun en hatasız hareketin yapılaması oyun algoritmalarının temel unsurudur. Üç Taş oyunu çok karmaşık bir oyun olmadığı için, Uygun arama ağacının oluşturtup uygun değerin bulunması kolaydır. Optimum bir strateji için ağaçta ki her bir elemana Min-Max diye adlandırılan değerler atanır. Bu değerler Max'm kazancını artırmaya yöneliktir. Max arama ağacını kontrol edip bir sonraki hamlesini belirlerken Yüksek Min-Max değerine sahip ağaç elemanını tercih eder buna karşın Min 'de tam tersini tercih edip Max'in kaybetmesi için uğraş verir (Russell and Norving ,2001). Max Hamlesine karar verirken oluşturulan karar ağacında tekrarlı olarak rakibin hamlesine göre karşılık gelen bütün hamlelerin sonuçlarına bakar ve kendine en yüksek kazancı sağlayacak hamleyi seçer.



Şekil II-1: Üç Taş Oyunu İçin Arama Ağacı

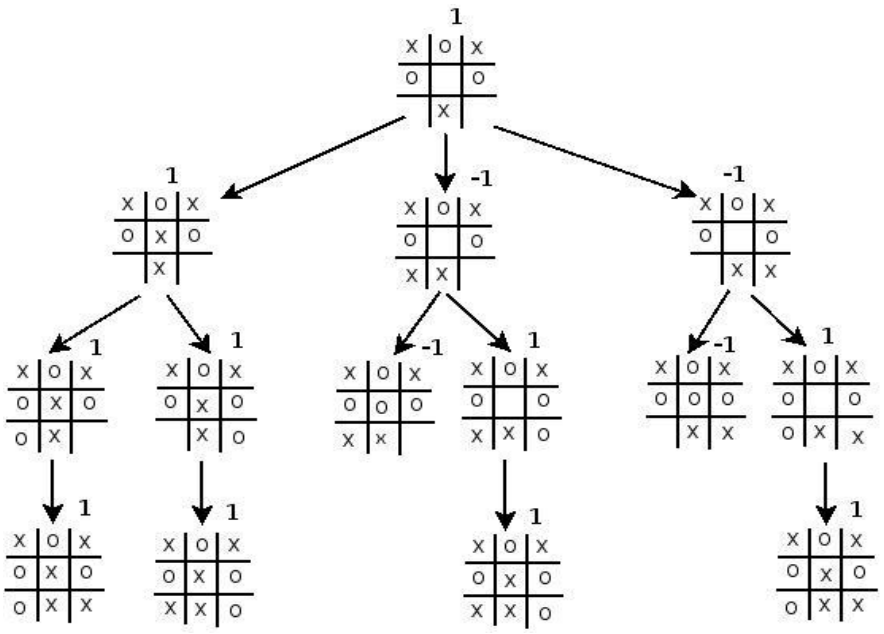
Algoritmanın çalışma prensibinin anlaşılması için Üçtaş oyunu için oluşan örnek bir pozisyon problem olarak ele alınmıştır Şekil II-2 ’de görülmektedir. Problemin çözüm için oluşturulan arama ağacı Şekil II-1 ’de görülmektedir.



Şekil II-2: Üç Taş Oyunu İçin Örnek Bir Pozisyon

Bu örnek için Başarı Fonksiyonu (Utility Function): X kazanırsa 1; O kazanırsa -1 ve Beraberlik için 0 olarak her bir duruma atanıyor. Öncelikle yapılması gereken oyunun güncel anından itibaren bundan sonraki bütün pozisyon ihtimallerini kapsayan bir arama ağacı oluşturmak ve bu arama ağacının her bir Düğümüne (Node) yaprağa kadar bir başarı fonksiyonu değeri atamaktır. Başarı fonksiyonu

atama işlemi ağaç ortaya çıkarıldıktan sonra sondan başa doğru yani yapraklardan köke doğru verilir. Şekil 6.18 ‘de yukarıda bahsedilen problem için bir sonraki hamlenin X ‘in yapacağı düşünülerek arama ağacı oluşturulmuş ve her bir Düğüme ilgili Başarı fonksiyonu atanmıştır. Arama ağacı oluşturduktan sonra sıra karar aşamasına gelir. Sıra X oyuncusunda olduğu için ve X’in amacı bir sonraki hamlenin en iyisi olmasıdır, Bu da X oyuncusunu ‘1’ Başarı Fonksiyonuna sahip dala yönelmesidir. X oyuncunsun altında üç dal vardır üçünün içinde Başarı fonksiyonu en büyük olan sol daldır “1” . Bu nedenle X sol taraftaki yolu seçer. Bu aşamada sıra O’ya gelir O’nun amacı Rakibin bir sonraki hareketini en aza indirecek şekilde hareket emektir yani “-1” Başarı fonksiyonuna sahip Düğüme ulaşmaktır. Ama O oyuncusu hangi hamleyi yaparsa yapsın altında ki dallarda ki Başarı fonksiyonu 1 ’dir Bu da X oyuncunun kazanma durumunu temsil eder. O oyuncusu iki daldan birini seçer, X oyuncusuna tekrar sıra geldiğinde altında ki dallar artık yapraktır ve daha derine inilmez her iki yapraktaki başarı fonksiyonu 1 olduğu için X oyuncusu hangi yolu takip ederse etsin oyunu kazanır. Arama işleminin algoritması Şekil II-1 ’de görülmektedir



Şekil II-3:Üç Taş Oyunu İçin Örnek Arama Ağacı

2.2.2 $\alpha - \beta$ (Alfa- Beta) Budama Yöntemi

Minimax algoritmasının açıklamasından da anlaşılacağı gibi, yöntemde çözüm ağacının oluşturulması herhangi bir durum değerlendirmesinden yoksundur. Yalnız ağacın tamamı oluşturulduktan sonra terminal düğümlerin değerlendirilmesi gerçekleştirilir. Buradan dallar sayısı arttıkça durumların değerlendirilmesi ve en iyi gidişin yapılması için gerekli hesaplama zamanı büyümektedir. Örneğin, satranç için ortalama dallanma faktörünün 35 olduğu düşünülürse, herhangi bir satranç pozisyonunu minimax yöntemiyle 6 derinliğe kadar araştırmak için ortalama olarak $35^0 + 35^1 + 35^2 + 35^3 + 35^4 + 35^5 + 35^6 = 1.892.332.261$ adet durumun değerlendirilmesi gerekecektir. Bu ise, hem zaman hem de bellek gereksinimi açısından kabul edilemeyecek bir sayıdır. Öte yandan arama ağacında birçok pozisyonun değerlendirilmeden göz ardı edilmesi mümkündür. Ağacın oluşturulması sırasında durum değerlendirmelerinin de yapılması, yani belirli sınırlamalar getirilmesi, minimax algoritmasını daha etkin kılmaktadır. Bu yaklaşımın temelinde J.McCarty'nin önerdiği α ve β gibi iki değişkenin kullanımı yatmaktadır. (Nabiyev, 2010)

Yöntemde iki değişkenin kullanılması, oyuncular için (MAX ve MIN) durumlarını ifade eden sezgisel fonksiyonun değerlendirilmesi ile ilişkilidir. *α – β yönteminin* ana amacı, mutlak biçimde değeri iyi olanı değil, kötü olmayan gidişin bulunmasıdır. Burada *α,* MAX oyuncusu için garantilenmiş en küçük değerlendirmedir. *β ise*, MAX'ın alabileceği fonksiyon değerlerinin en büyüğüdür. MIN açısından ele alındığında β, onun için garantilenmiş değerler içerisinden en kötüsüdür. Böylece aranan fonksiyonun değeri *(α,β)* aralığındadır. Eğer herhangi bir durumun değeri (α,β) aralığı dışında olursa bu, araştırılan durumun önem taşımadığını ifade eder. Buradan ise benzeri durumların kesin olarak değerlendirilmesinin gereksiz olduğu sonucuna varılır ve ağacın belirli kısmı incelenmez. Çözüm ağacının belirlenen aralığa göre bazı dallarının değerlendirmeye alınmaması, yöntemde budama olarak isimlendirilmektedir

A ve B düğümleri MIN sırasında olduğundan A'nın değeri olan 3'ten kucuk bir sayı B'nin değeri olarak seçilecek olursa B'nin bir sonraki adımda A'yı geçemeyeceği kesindir ve B düğümüyle daha fazla zaman kaybetmeye gerek yoktur.

B'nin ilk çocuğunun değeri 2 olduğu bilindiği anda diğer çocukların değerleri 2'den büyük olması koşulunda en küçük değer kalacak olan 2'nin B'nin değeri olacağı, diğer çocukların değerlerinin 2'den küçük olması koşulunda ise B'nin değerinin 2'den küçük (ve dolayısıyla MAX sırasında A'nın değeri olan 3'u geçemeyecek) bir değer alacağından B ile ilgilenmeye gerek yoktur.

Ya da kısaca;

alfa = bilinen en iyi MAX değeri ve beta = bilinen en iyi MIN değeri olmak uzere,

1. MAX düğümlerinde, herhangi bir yolu izlemeye başlamadan önce, bir önceki yolun değerini beta değeri ile karşılaştır. Eğer değer beta'dan büyükse bu düğümü atla

2. MIN düğümlerinde, herhangi bir yolu izlemeye başlamadan önce, bir önceki yolun değerini alfa değeri ile karşılaştır. Eğer değer alfa'dan küçükse bu düğümü atla

Alfa-Beta kesintileri, MiniMax algoritmasında yapay zeka kalitesini düşürmeden önemli hız kazancı doğurabilir. Ancak bu hızlanmanın ölçeği arama ağacının yapısına bağlıdır. MAX düğümlerinin değerleri küçükten büyüğe doğru sıralı ise, ya da MIN düğümlerinin değerleri büyükten küçüğe sıralı gelmişse alfa-beta kesintilerinin performansa bir katkısı bulunmaz.

2.3 OYUN TÜRLERİ

	Karara Dayalı (Deterministic)	**Şansa Dayalı (Chance)**
Tam Bilgili (Perfect information)	• Satranç (chess) • Dama (checkers) • Go • Othello • Sudoku (diamond)	• Tavla (backgommon) • Monopoly
Eksik Bilgili (Imperfect information)	• Savaş Gemisi (Battleships) • Üç Taş Oyunu (Tic-tac-toe) • Mayın Tarama(minesweeper)	• Briç (bridge) • Poker • Kelime Bulmaca (scrabble) • Nuclear war

Şekil II-4: Bilgisayar Oyunlarının Niteliklerine Göre Sınıflandırılması

Tümel bilgili (complete information) bir oyunda oyuncular kendi kişisel stratejilerini ve netice fonksiyonlarını diğer oyuncular gibi bilmektedir. Buna ek olarak, her oyuncu diğer oyuncuların tüm bilgiye sahip olduğunu bilmektedir. Tikel bilgili (incomplete information) oyunlarda oyuncular oyunun kurallarını ve kendi kişisel tercihlerini bilmekte, ancak diğer oyuncuların netice fonksiyonlarını bilmemektedir. Tam bilgili (complete information) bir oyun, oyuncuların ardışık olarak stratejileri seçtiği ve diğer oyuncuların seçiminin ne olduğunun farkında olduğu bir oyun türüdür. Satranç bu oyuna örnek verilebilir. (Aktan, C. C., & Bahçe, A. B., 2007)

Eksik bilgili (incomplete information) oyun, oyuncuların yalnızca diğer oyuncularının ne yapacağını tahmin ederek, bir başka oyuncunun hamlesini bilmeden hareket etmesidir. (Kelly, A., 2003) , (Brams, S. J. 2005).

Sudoku oyunu, tek kişilik oynanan bir oyun olup, oyuncu kendi kişisel stratejilerini ve netice fonksiyonlarını bildiğinden tam bilgili (complete information) oyun grubuna girmektedir. Ayrıca Sudoku oyununda kesinlikle şans faktörü yer almadığından, oyun tamamen bulunulan pozisyona göre karar vermeye odaklıdır. Dolayısıyla Sudoku Oyunu aynı zamanda karara dayalı oyun grubunda yer alır.

BÖLÜM III

3.1 BENZER ÇALIŞMALAR

Kocaeli Üniversitesi Fen Bilimleri Enstitüsüne ait Yapay Zeka Yöntemleriyle Oyun Geliştirme (Gürbüzer, 2008) adlı yüksek lisans tez çalışmasında 3x3'lük oyun tahtasında 2 kişiyle oynanan ve oyun türleri başlığı altında açıkladığımız tam bilgili (perfect information) ve karara dayalı (deterministic) oyunlar grubuna giren Tic-Tac-Toe Oyunu (Üç Taş Oyunu) programlanmış, insan hamlesine karşılık verilebilecek bilgisayar hamlesi oluşturulmaya çalışılmıştır. Bu tez çalışmasında da Oyunlar başlığı altında açıkladığımız 2 kişilik oyunların programlanmasında sıkça kullanılan Minimax yönteminden yararlanılmıştır. Oyun oynama aşamasında daha ilk insan hamlesine karşın, bilgisayarın hamle yapmak için oldukça bariz bir bekleme süresine ihtiyaç duyduğu gözlemlenmiş ve insanlar için çok basit bir oyun olan Tic-Tac-Toe Oyununun (Üç Taş Oyunu) bilgisayarların oynaması için oldukça işlem gücü gerektirdiği belirtilmiştir. Sudoku oyununda problem çözme sürecinde işlem süresi 2-3 sn. olmakla birlikte, başlangıçtan hedef duruma gidilebilecek (Sudoku Probleminin Durum Uzayı başlığı altında hesaplandığı üzere) yaklaşık 10^8 durum varken, Tic-Tac-Toe Oyununda (Üç Taş Oyunu) 3^9 durum (yaklaşık 10^5 durum) mevcuttur. Dolayısıyla Sudoku oyununun oyun ağacı karmaşıklığı (durum uzayı) Tic-Tac-Toe oyunundan büyük olmasına rağmen çok daha kısa sürede sonuca ulaşılabilmektedir., Buradan hareketle, oyun programlamada kullanılan algoritmaların, oyunların niteliklerine göre iyi seçilmesi ve hatta gerekli görüldüğünde var olan algoritmalardan bağımsız özgün bir algoritma oluşturulmalıdır denilebilir.

Marmara Üniversitesi Fen Bilimleri Enstitüsüne ait Esnek Programlama Yaklaşımları İle Oyun Geliştirme (Erkalkan, 2010) adlı yüksek lisans tez çalışmasında İstanbul Fighter isimli bir oyun programlanmış ve 2D programlama açıklanmıştır.

Ankara Üniversitesi Fen Bilimleri Enstitüsüne ait Dağıtık Yapay Zeka Destekli 3 Boyutlu Domino Oyunu (Nooraden, 2011) adlı yüksek lisans tez çalışmasında 4 farklı oyuncunun 4 farklı bilgisayarda TCP/IP protokolü yardımıyla ağ üzerinden oynayabilecekleri üç boyutlu bir domino oyunu programlanmıştır. Oyunun sonucu, taşların başlangıçta rastgele dağılımına bağlıdır. Sudoku Oyunu Probleminin Çözümlenmesi başlığı altında belirtildiği üzere, "Başlangıç", "Orta" ve "Zor" olarak oyuncunun seçeceği duruma bağlı olarak rastgele atanan sayılar doğrultusunda hedef durum şekillenmiştir. Bu bağlamda Domino Oyunu üzerine yapılan bu çalışma, durum uzayında oluşacak başlangıç durum ve buna bağlı oluşacak hedef durum açısından Sudoku Oyunu ile paralellik göstermektedir.

Gazi Üniversitesi Bilişim Bilimleri Enstitüsüne ait Yapay Zeka Uygulamalarında Kullanılan Arama Algoritmalarının Kıyaslanması (Benzer, 2007) adlı yüksek lisans tez çalışmasında arama algoritmalarının çalışmalarını incelemek için birbirleriyle kıyaslama işlemi yapılmış, bunun için de 8-puzzle problemi kullanılmıştır. Kıyaslama işlemi için 1000`er adet 8-puzzle başlangıç durumu örneği oluşturulmuştur. Bu oluşturulan örnekler 5 dakikalık süre içerisinde BFS, DFS ve A* algoritmaları tarafından, hedef duruma ulaşmak üzere çözümlendirilmeye çalışılmış ve çözüm için açtıkları durum sayıları bir metin dosyasına kayıt edilmiştir. Elde edilen veriler analiz edildiğinde derinliğine aramada kökten yaprağa kadar yalnız bir yol depolandığı için bellek gereksinimi azdır. Dallanma faktörü b ve maksimum derinliği m olan durum uzayında derinliğine aramada yalnız bm düğümün saklanması gerekir. Genişliğine aramada ise bu b^d idi. Burada, d en sıg çözümün bulunduğu derinliktir. Genişliğine aramada 111 terabyte bellek gerekirken derinliğine aramada 12 kilobyte ın yeterli olduğu bahsi geçen tez çalışmasında belirtilmiştir. Dolayısıyla, bu tez çalışmasında daha az bellek gerektiren DFS (Depth First Search) arama tekniğine göre Sudoku oyunu arama ağacı oluşturulmuştur.

3.2 SUDOKU OYUNU NEDİR?

Sudoku, Su Doku Japon tipi bir bulmacadır. Japonca "Sayılar tek olmalı" anlamına gelen "Suuji wa dokishin ni kagiru" kelimelerinin kısaltması olan Sudoku, günümüzde Asya'dan, Avrupa ve Kuzey Amerika'ya da yayılan oldukça popüler bir oyundur. [2]

Japonya'da 1980'lerden beri popüler olan Sudoku'nun kökeni 18. yüzyılda İsviçreli matematikçi Leonhard Euler tarafından tasarlanan Latin Kareleri'ne dayanır. Oyundaki amaç, 9 kareden oluşan her sütunu, satırı ve bloğu 1'den 9'a kadar olan rakamları içerecek şekilde, hiçbir rakamı tekrarlamadan veya atlamadan doldurmaktır. [7]

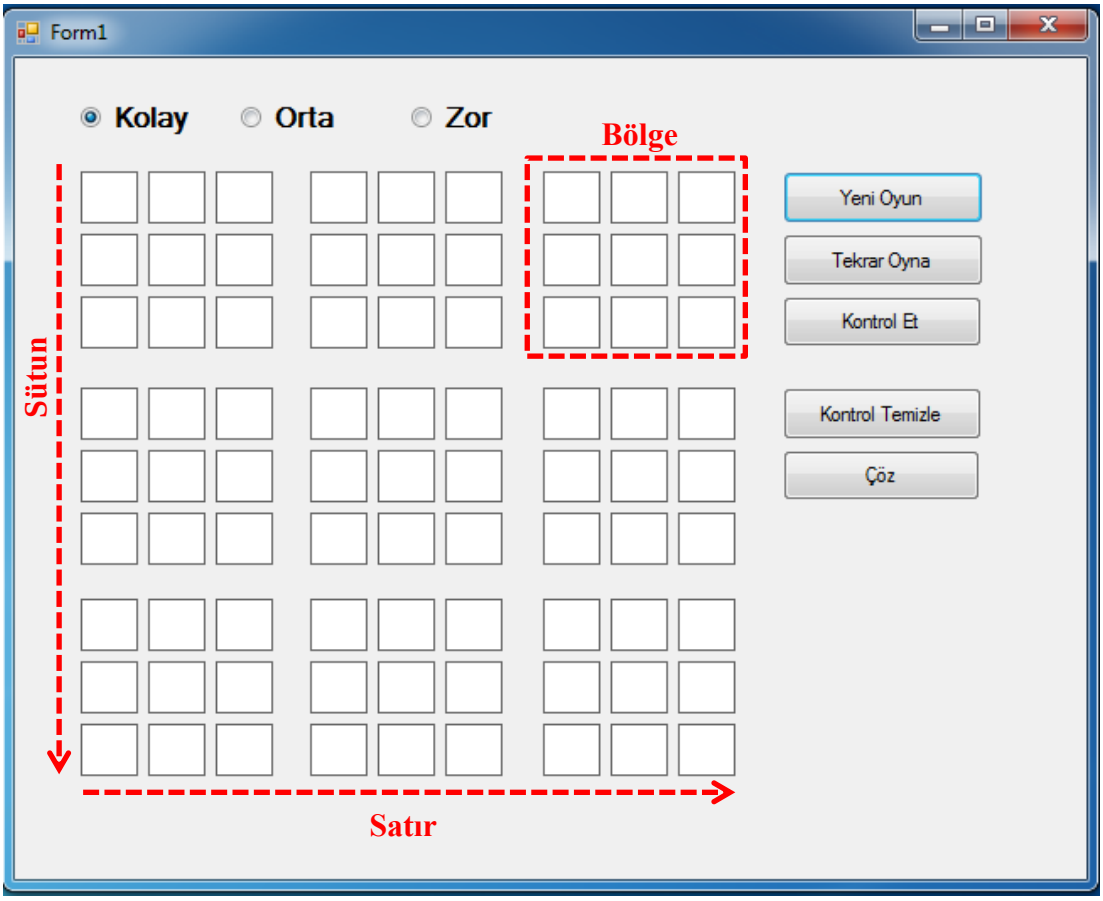
ABD'de Number Place (rakam yerleştirme) olarak bilinen oyun, Türkiye'de 1994 yılından bu yana Diamond adıyla piyasadadır. [3]

Dünya'nın en çok ilgi gören oyunlarının başında gelen bu bulmaca, rakamlarla oynanmasına karşın matematikle ilgisi olmayan bir oyun olup, toplama veya çarpma gibi matematiksel işlem bilgisi gerektirmez. [4]

Sudoku oyunu günümüzün düşünme ve mantık yürütme yeteneğini geliştirmeye en fazla fayda sağlayan zeka oyunu olarak bilinmektedir. Bu yüzden uzak doğuda okullarda çocuklara sudoku oynama imkanları sunulur. Sudoku oyununun mantığı 3 boyutlu olarak düşünebilme yeteneğimizi geliştirmektir. Her sudoku oyununun tek bir çözümü olduğu için tahmin edilerek çözmek neredeyse imkansızdır. Bu yüzden verilen sayılardan yola çıkarak her bir hücreye hangi sayının geleceğini bulmak gerekir. Bazen Sudoku oyununun birden fazla çözümü olabileceği iddia edilir fakat birden fazla çözümü olan sudoku oyunu sadece sayı karmaşası oluşturur. Gerçek Sudoku oyununda sadece tek bir çözüm olması gerekir. [10]

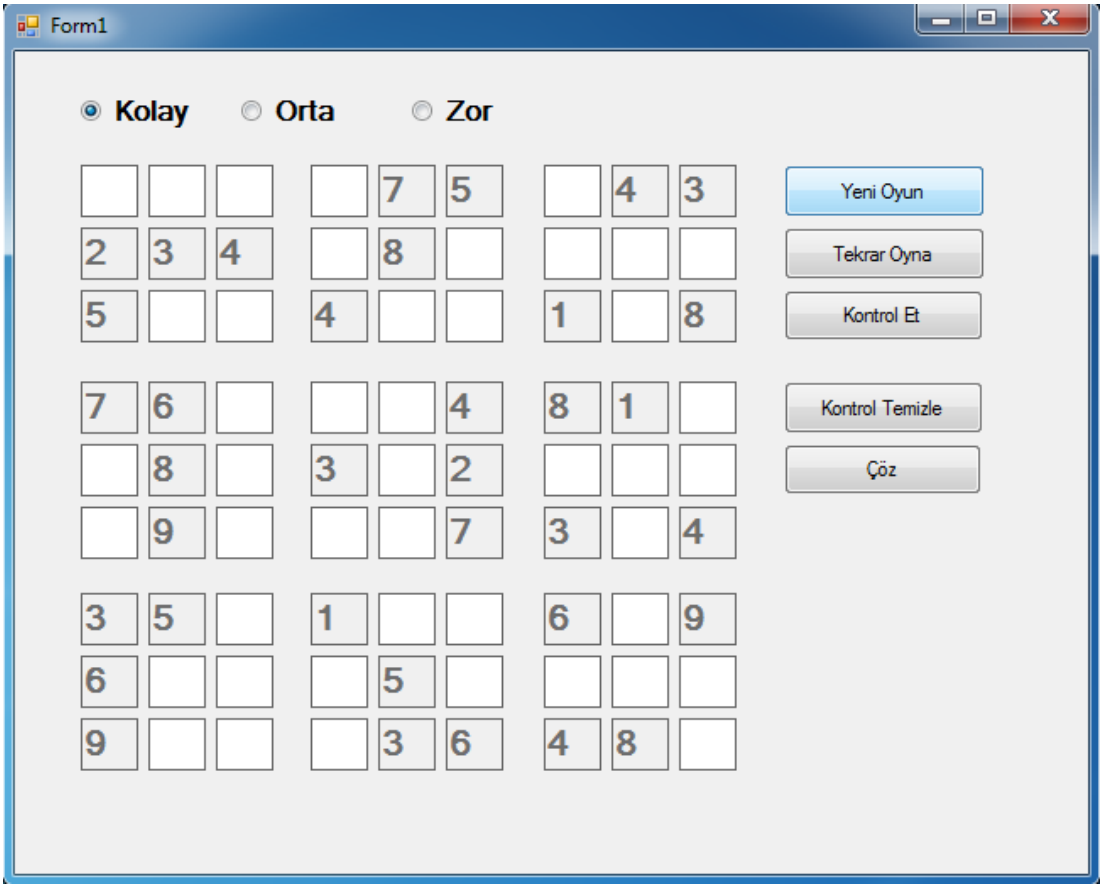
Sudoku oyununda temel olarak iki çözüm stratejisi bulunur. Sudoku çözüm yollarından birincisi hangi hücrede hangi sayıların olabileceği, ikincisi ise hangi kutucukta hangi sayların olamayacağıdır. Bu şekilde sudoku oyununda verilen sayılara dayanarak hangi hücrelerde hangi sayıların olabileceği ve hangi sayıların olamayacağı mantığına dayanarak çözülebilmektedir. Sudoku çözmek için sıralı gitmek ise her zaman sudoku çözümünde kolaylık sağlamaktadır. Bu yüzden sayıları karmaşık olarak kullanmak yerine sıralı olarak kullanmak çözümü kolaylaştırır. [10]

3.2.1. SUDOKU OYUNUNUN ARAYÜZÜ



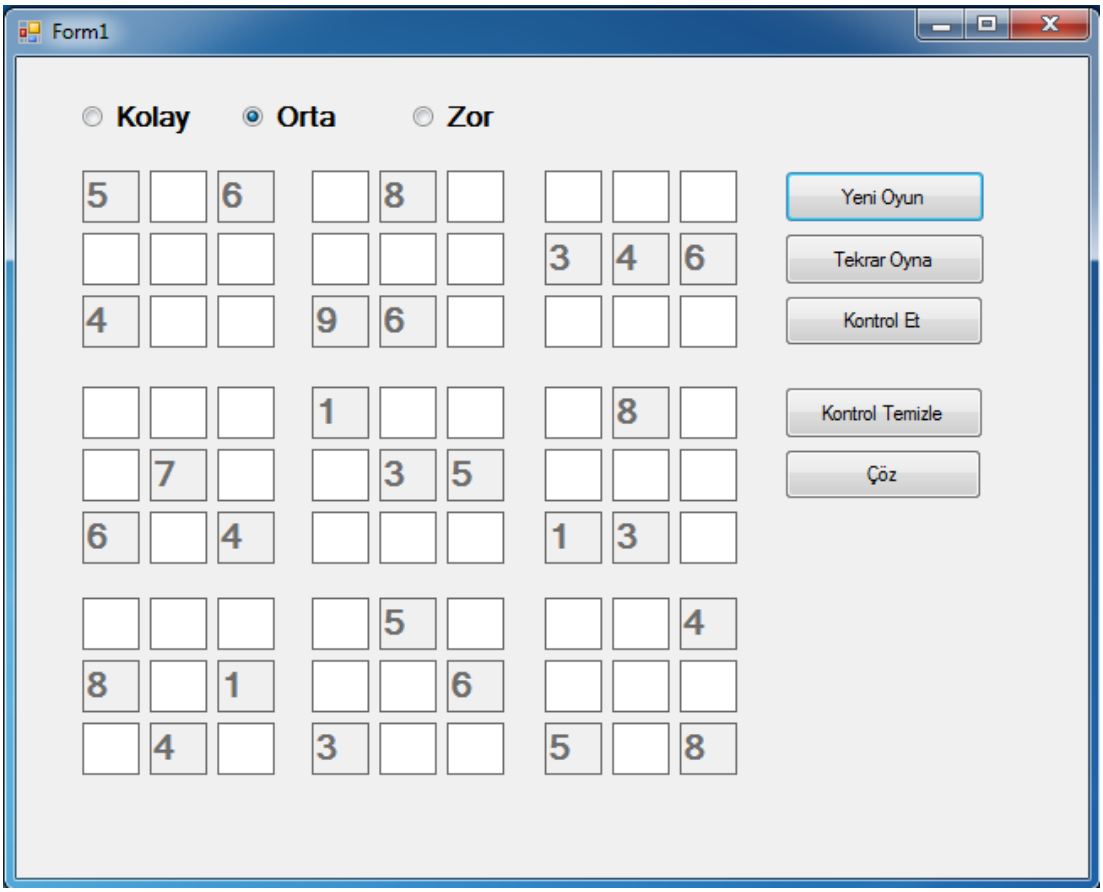
Şekil III-1 : Sudoku Oyununun Arayüzü

Şekil II-6'da da görüldüğü gibi sudoku oyunu 9x9' luk bir ızgarada oynanır. Bu ızgara "bölge" isimli 3x3 alt ızgaralara bölünmüştür. 9x9' luk ızgaradaki her 9x1'lik dikey kesite "sütun", 1x9'luk her yatay kesite ise "satır" denir.



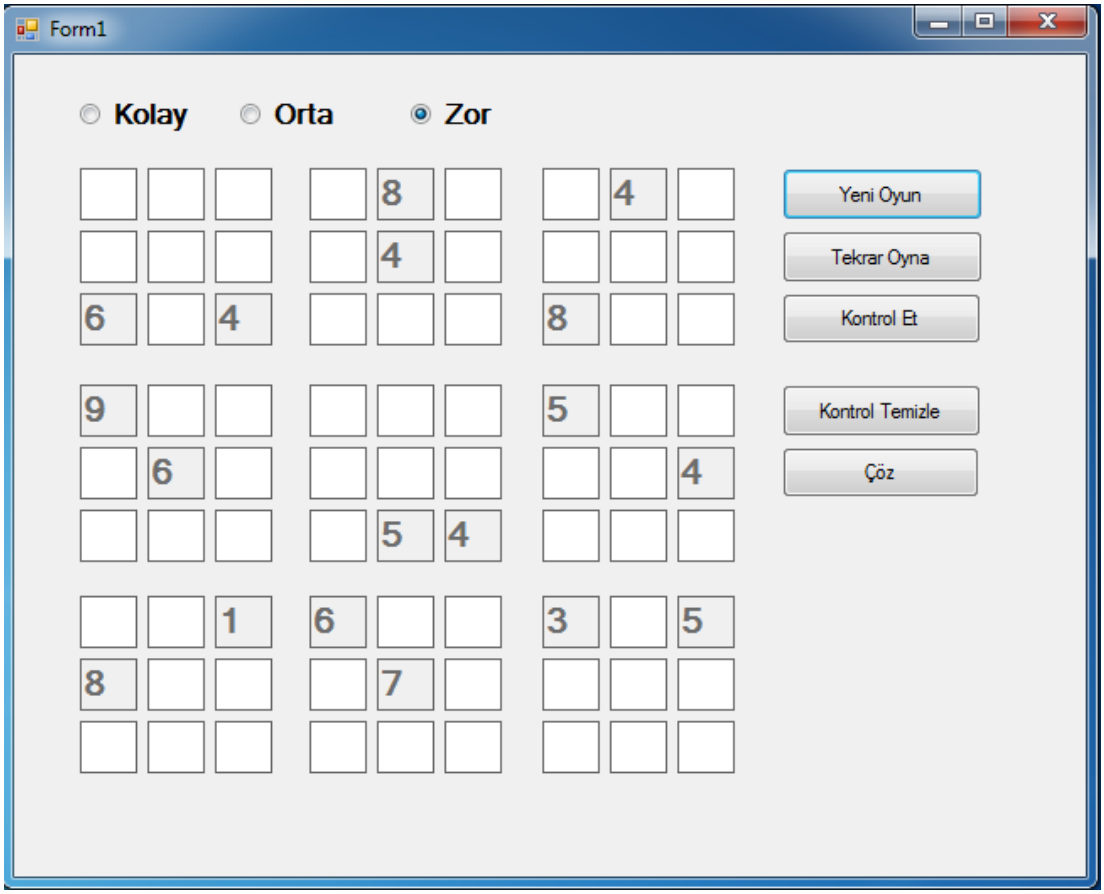
Şekil III-2: "Kolay" Düzeyde Oynanan Sudoku Oyunu

Şekil II-7'de de görüldüğü üzere Sudoku, sayılarla doldurulmuş bazı ızgara hücreleri ile başlar. Eğer oyun "Kolay" derecede oynanıyorsa, her bölgedeki açılmış kutucuk sayısı 4 olup, 9x9'luk ızgarada toplam açılan kutucuk sayısı 36 tanedir. Izgarada toplam 81 kutucuk olduğundan, oyuncudan doğru bulması gereken kutucuk miktarı 45 'tir.



Şekil III-3: “Orta ” Düzeyde Oynanan Sudoku Oyunu Arayüzü

Şekil II-8’de de görüldüğü üzere Sudoku Oyunu “Orta” derecede oynanıyorsa, her bölgedeki açılmış kutucuk sayısı 3 olup, 9x9’luk ızgarada toplam açılan kutucuk sayısı 27 tanedir. Izgarada toplam 81 kutucuk olduğundan, oyuncudan doğru bulması gereken kutucuk miktarı 54 ‘tür.

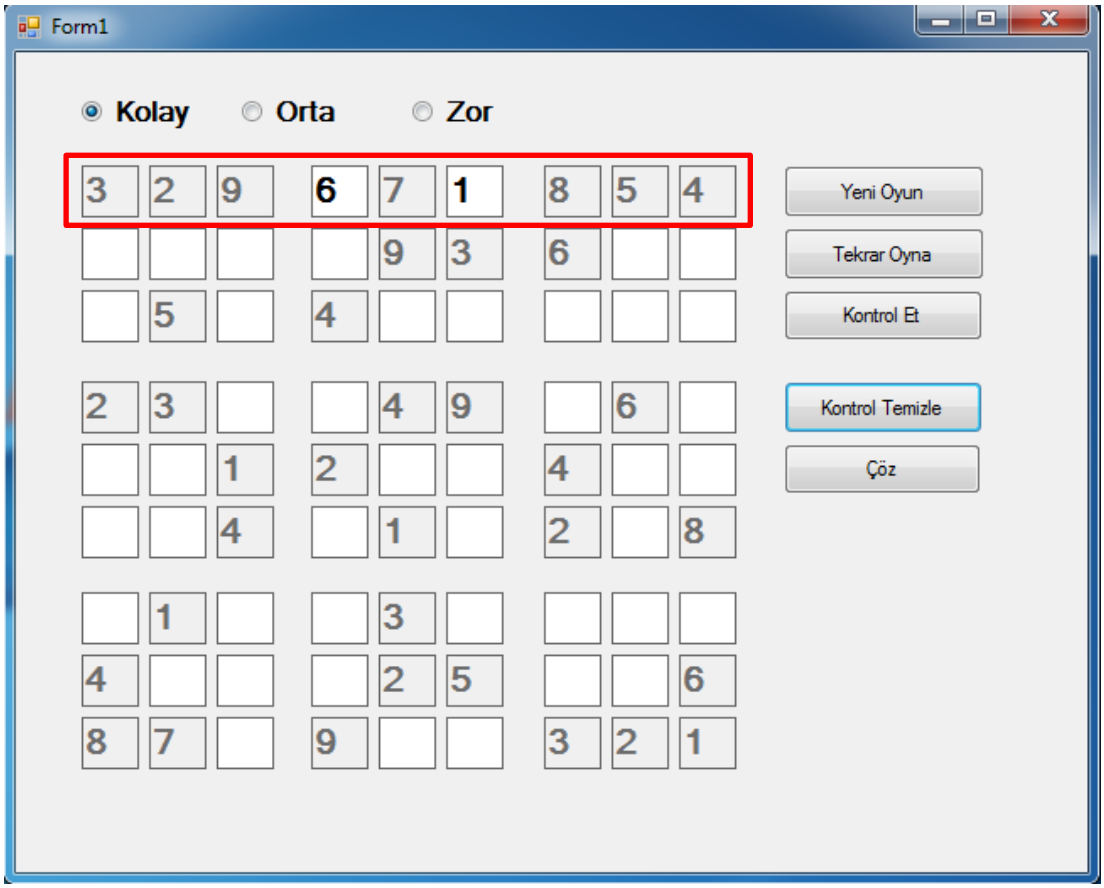


Şekil III-4: "Zor " Düzeyde Oynanan Sudoku Oyunu Arayüzü

Şekil II-9'da görüldüğü üzere Sudoku Oyunu "Zor" derecede oynanıyorsa, her bölgedeki açılmış kutucuk sayısı 2 olup, 9x9'luk ızgarada toplam açılan kutucuk sayısı 18 tanedir. Izgarada toplam 81 kutucuk olduğundan, oyuncudan doğru bulması gereken kutucuk miktarı 63 'tür.

3.2.2. OYUNUN KURALLARI

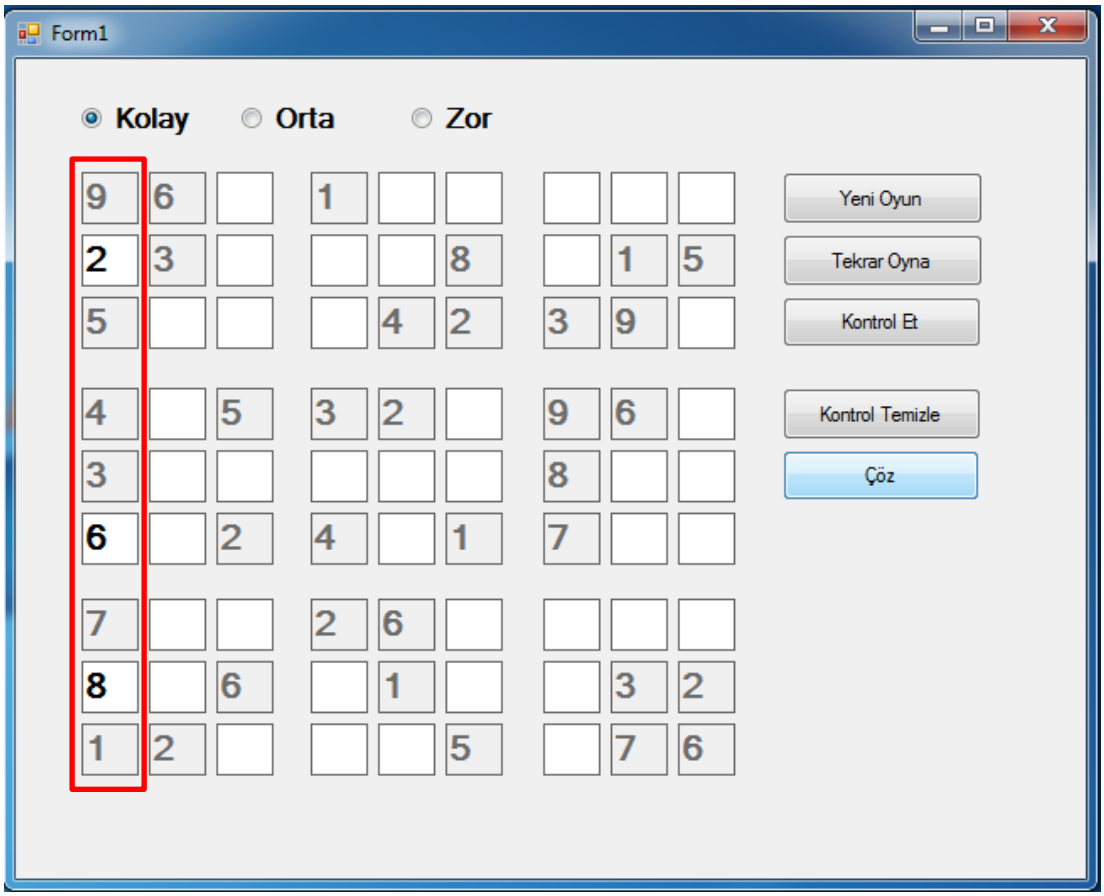
KURAL-1: Her rakam her sütun üzerinde yalnızca bir kez kullanılmalıdır.



Şekil III-5: Sudoku Oyununda Yatay Izgarada Sayı Düzeni

Şekil II-10'da ilk satırda oyuncudan bulması gereken rakam 2 adettir. Daha önce 3,2,9,7,8,5 ve 4 rakamları önceden rastgele atanmış olup geriye 6 ve 1 rakamları kalmış olduğundan ilk satırda açık hücreler için sadece bu sayılar kullanılabilir durumdadır.

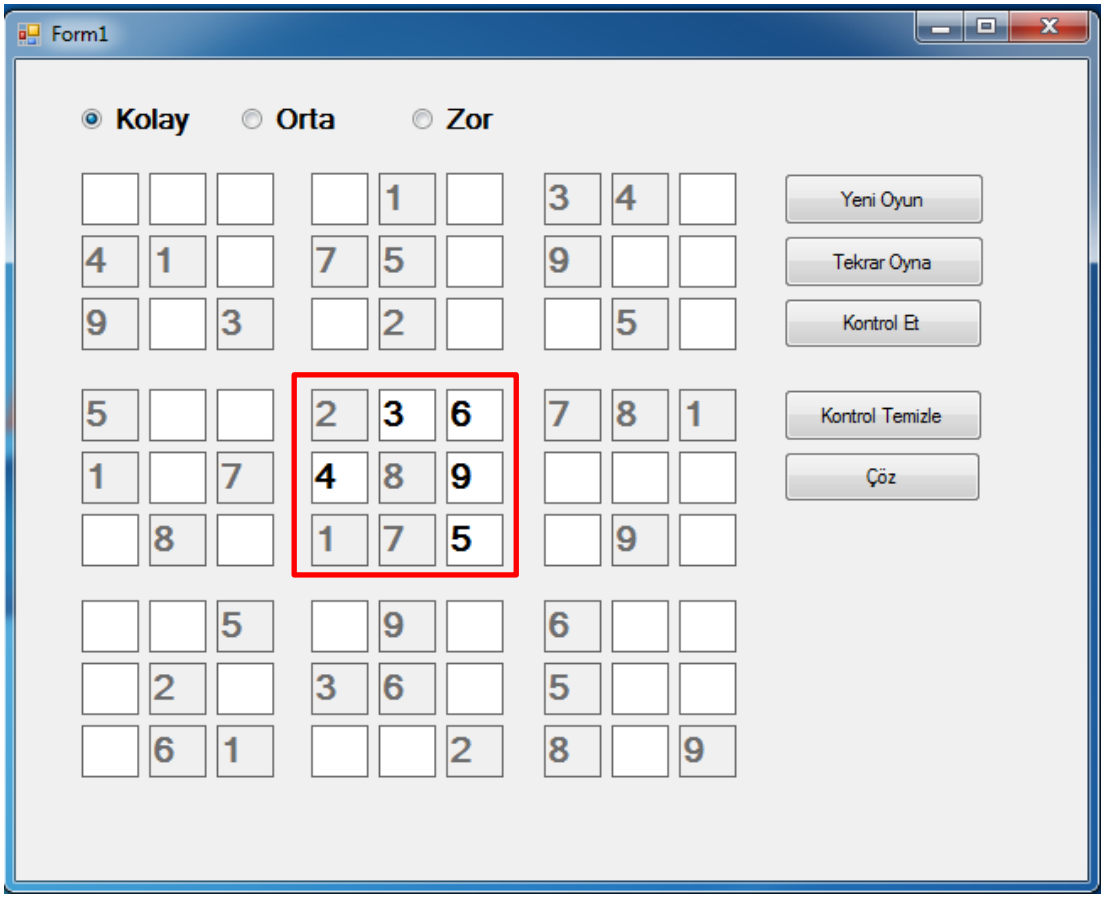
KURAL-2: Her rakam her sütun üzerinde yalnızca bir kez kullanılmalıdır.



Şekil III-6: Sudoku Oyununda Düşey Izgarada Sayı Düzeni

Şekil II-12'de ilk sütunda oyuncudan bulması gereken rakam 3 adettir. Daha önce 9,5,4,3,7 ve 1 rakamları önceden rastgele atanmış olup geriye 2,6 ve 8 rakamları kalmış olduğundan ilk sütunda açık hücreler için sadece bu sayılar kullanılabilir durumdadır.

KURAL-3: Her rakam her bölge üzerinde yalnızca bir kez kullanılmalıdır.



Şekil III-7: Sudoku Oyununda 3x3'lük Bölge Izgarada Sayı Düzeni

Şekil II-13'de 4. Bölgede oyuncudan bulması gereken rakam 5 adettir. Daha önce 2,8,1 ve 7 rakamları önceden rastgele atanmış olup geriye 3,6,4,9 ve 5 rakamları kalmış olduğundan bu bölgede açık hücreler için sadece bu sayılar kullanılabilir durumdadır.

Bu kurallar, her sayının her bir satır, sütun ve bölgede yalnızca bir kez kullanılabileceği şeklinde özetlenebilir.

4.1 ARAMA YÖNTEMLERİ

Arama, yapay zeka alanında problem çözmede yaygın olarak kullanılmaktadır. Bundan dolayı arama çoğu zaman "arayarak problem çözme" konusu olarak da literatürde geçmektedir. Problem çözme yapay zekanın önemli bir kısmıdır (Coppin, 2004)

Aramanın amacı, en uygun çözümü veya yakın çözümü çabucak bulmaktır. (Weixiong, 1999).

Arama algoritmaları iki ana başlıkta toplanabilir: (Benzer, 2007)

- Uninformed Search (Blind-Kör)-Bilgiye Dayanmayan Arama
  - Breadth-first Search (Önce Genişliğine Arama)
  - Uniform Cost First
  - **Depth-First Search (Derinlik Öncelikli Arama)**
  - Depth Limited Search (Derinlik Sınırlı Arama)
  - Iterative Deepening Search (Yineli Derinleştirmeli Arama)
  - Bidirectional Search (Çift Yönlü Arama)

- Informed Search (Heuristic-Sezgizel) - Bilgiye Dayalı Arama
  - Best –first /Greedy Search (En İyi Arama)
  - A* Search (Toplam Yol Maliyetinin Azaltılması)
  - Bellek Sınırlı Arama (Memory-Bounded
  - Sezgisel Bulma Fonksiyonları (Heuristic Functions)

Bu tez çalışmasında Bilgiye Dayanmayan Arama (Uninformed Search) yöntemlerinden Derinlik Öncelikli Arama (Depth First Search) kullanıldığından, sadece bu arama yöntemi incelenmiştir.

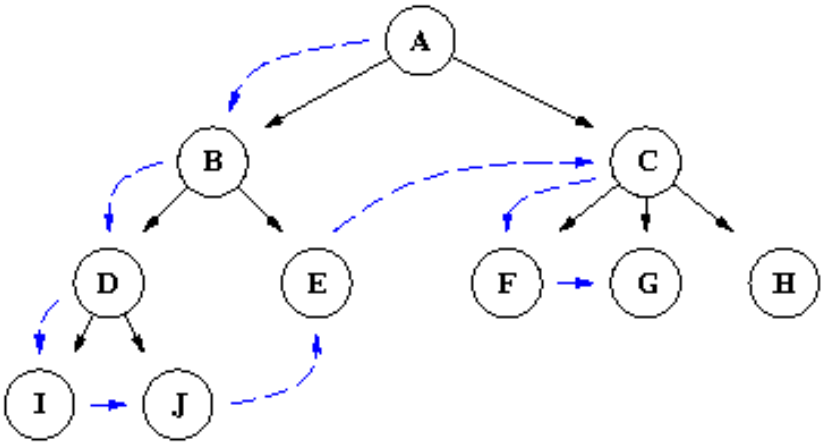
4.1.1 DERİNİNE ARAMA (DEPTH-FIRST SEARCH)

Derinine arama algoritması bir graf üzerindeki dolaşma yöntemlerinden biridir. (Çölkesen, 2000; Chen ve Shin, 1990; Ibrahim, vd., 2001).

Derinine arama (DFS) algoritması, bulmaca ve oyunlar gibi yapay zeka problemlerine uygulanabilmektedir (Nabiyev, 2010)

Derinlik öncelikli aramada daima ağacın en derin düğümlerinden biri açılır. Potansiyel çözümünün çok derinlerde olmadığı durumlarda yaygın olarak kullanılmaktadır. (Tucker, 1996)

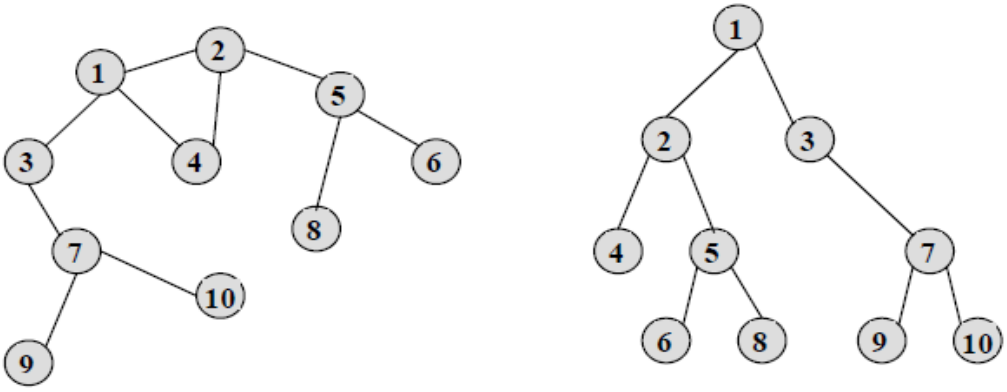
DFS algoritması "son giren ilk çıkar" (Last-in-first-out) mantığına dayanmaktadır. (Charniak, 1985)



Şekil III-8: Arama Ağacı Üzerinden Derinlik Öncelikli Aramanın Çalışması [8].

Algoritmanın uygulamasında bir başlangıç düğümü seçilir ve başlangıç düğümünden itibaren gidilebilecek en alt seviyedeki düğüme kadar gidilir. Ulaşılamayan bir düğüm olduğu zaman geriye dönüş yapılarak (*backtrack*) başka bir düğümün ayrıtından itibaren devam edilir. (Sakalli vd.,2006; Stojmenovic vd., 2000).

Derinlik öncelikli aramada dallanma sayısı ***b*** ve gidilebilecek maximum derinlik seviyesi ***m*** olmak üzere, bir problemde durum uzayı ***bm+1*** adet düğüm içerir. (Russell, S. & Norvig, P., 2003, p.75). DFS algoritmasının adımları, yukarıda verilen Şekil III-8 üzerinde açıklanmaktadır.



a) Uygulama Öncesi Graf b) Uygulama Sonrası Graf

.

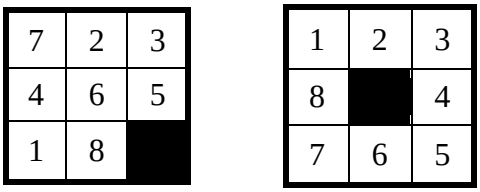
Şekil III-9: DFS Uygulaması İçin Bir Örnek Graf (Grimaldi, 2003)

DFS algoritmasının verilen bir G = (V , E) grafı için uygulaması gerçekleştirilmeden önce noktaların bir düzende oluşturulması gerekmektedir. Bu şekilde, eğer bir v noktasına komşu olan iki veya daha fazla nokta varsa ve bu noktaların hiçbiri ziyaret edilmemişse ilk hangi noktaya gidileceğine kesin olarak karar verilebilir (Grimaldi, 2003).

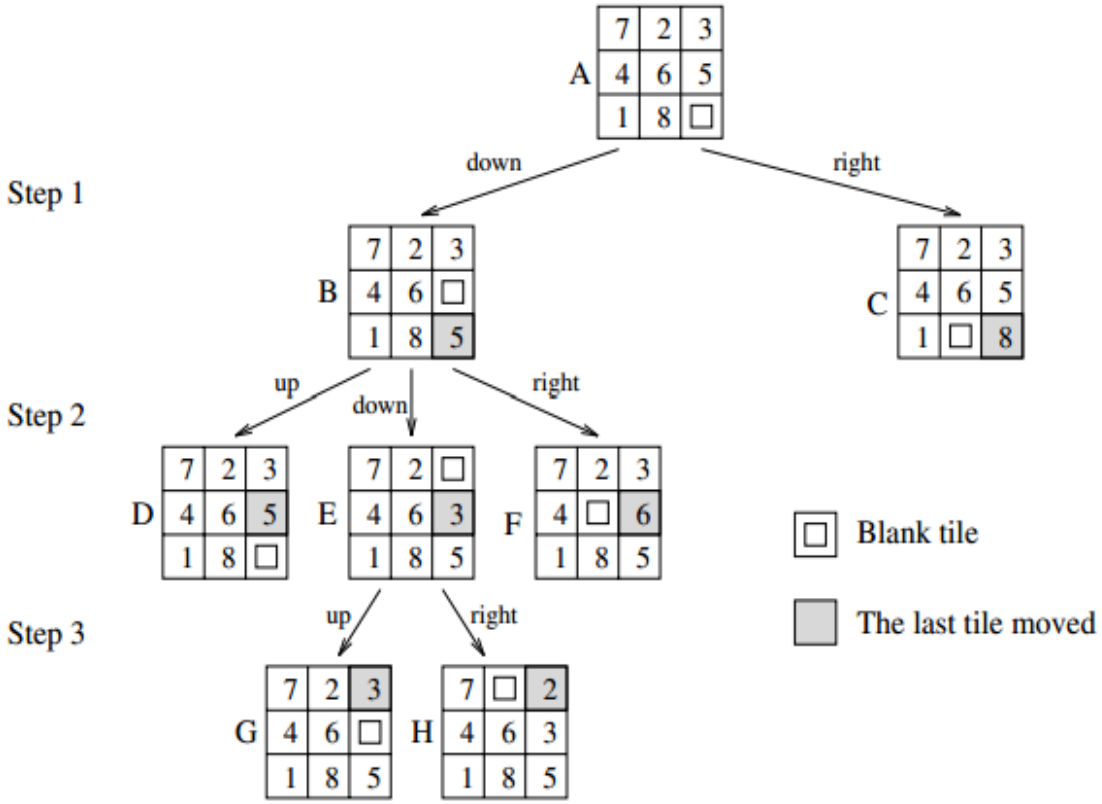
Şekil III-9'a bakıldığı zaman uygulama öncesi graf görülmektedir. Düğümlerin düzenleri ise 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 gibidir. Uygulamanın başında 1 numaralı düğüm kök düğüm olarak belirlenir ve *T* ağacı oluşturulur. Daha sonra ilk gelen düğüm 2 numaralı düğümdür ve daha önceden uğranılmamıştır. Bu düğüm ile bağlantılı olan kenar T ağacına eklenir ve 2 numaralı düğümden itibaren devam edilir. 2 numaralı düğümden sonra 4 numaralı veya 5 numaralı düğüme gidilir. Burada ilk önce 4 numaralı düğüm olduğu için ona gidilir. T ağacına 4 numaralı düğüm eklenir. 4 numaralı düğümden ileriye doğru gidiş kapalı olduğundan dolayı 2 numaralı ata düğüme geri dönüş yapar. 2 numaralı düğümden bu sefer 5 numaralı düğüme gidilir. Buradan ise sırasıyla 6 ve 8 numaralı düğümlere ulaşılır ve sonunda en yukarıda olan 1 numaralı kök düğüme geri dönülür. Bu noktadan sonra 3 numaralı düğüme gidilir. 3 numaralı düğümden sırasıyla 7, 9 ve 10 numaralı düğümler dolaşılarak grafın tümü üzerindeki dolaşım tamamlanmış olur. Şekil II-11.b' de uygulama tamamlandıktan sonraki yeni durum görülmektedir. (Esin, 2006)

4.2 8-PUZZLE PROBLEMİNDE DERİNİNE ARAMA YÖNTEMİ

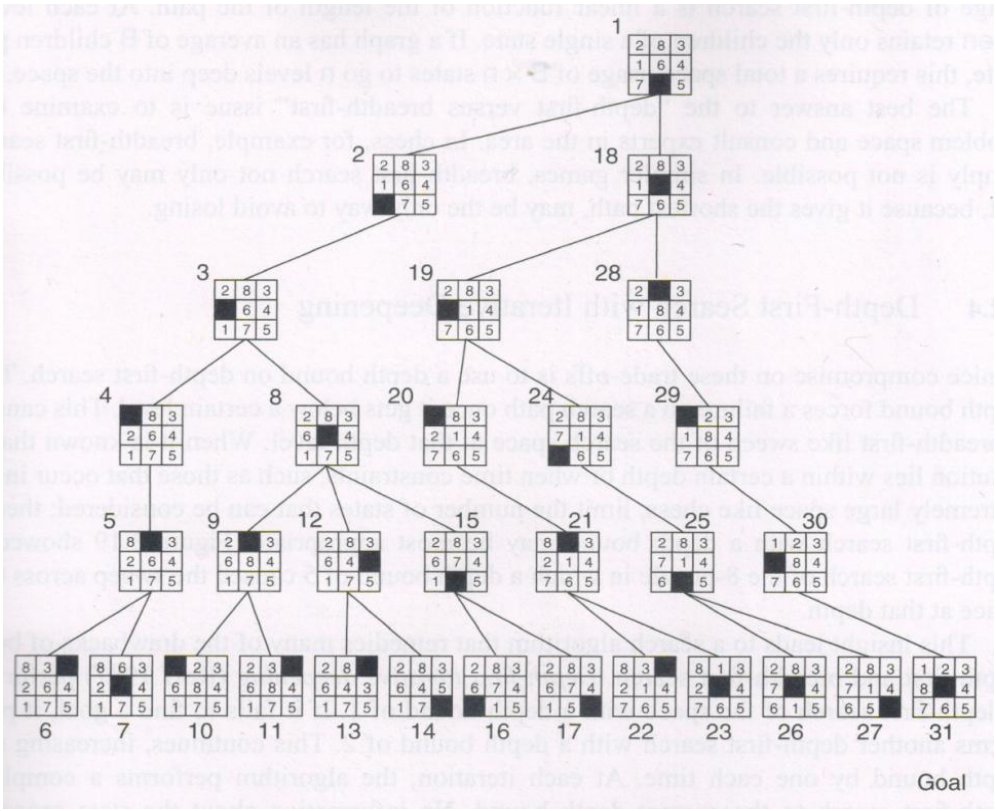
8-Puzzle problemi 1'den 8'e kadar sayılarla doldurulmuş ve bir karesi boş olan 3×3 boyutunda bir matrisin istenilen bir hedef duruma getirilmesinin amaçlandığı bir oyun olarak tanımlanabilir. Puzzle üzerindeki elemanların yalnızca boşluk ile yer değiştirdiği (kaydırıldığı) hareketler geçerli kabul edilmektedir.



Şekil III-10: 8-Puzzle Probleminde Başlangıç ve Hedef Durumlar



Şekil III-11:8-Puzzle Probleminin Derinine Aranmasından Bir Kesit [9]



Şekil III-12: 8-Puzzle Probleminin Derinine Arama Ağacı

Şekil III-12'den anlaşılacağı üzere 1 numaralı başlangıç durumdan 31 numaralı hedef duruma erişebilmek için 1.seviyede açılan kollardan itibaren, gidilebilecek en derin düğüme kadar gidilir, varılan düğüm hedef düğüm ile kıyaslanır, hedef düğüme gelinmemişse aynı dalın başka koluna geçilir. Varılan yerde hedef düğüm yoksa aynı yoldan geri dönülerek hedef düğüm bulununcaya kadar devam edilir.

8-puzzle örneğinde 9!/2 = 181440 ulaşılabilir durum mevcuttur. Yani 8-puzzle probleminin derinine arama ağacındaki düğüm sayısı 181440 dır. (Reinefeld, A., 1993).

15-puzzle örneği ise (4 x 4 lük bir oyun tahtası) yaklaşık 1,3 trilyon olasılık mevcuttur ve rastgele seçilen örnekler en iyi arama algoritmaları tarafından birkaç mili saniyede uygun bir şekilde çözülebilmektedir. 24 - puzzle (5 x 5 lik tahta) yaklaşık olarak 1025 olasılık mevcuttur ve böyle bir rastgele örneği mevcut makineler ve algoritmalarla uygun bir şekilde çözülmesi hala oldukça zordur. (Benzer, A. İ., 2007)

4.3 SUDOKU PROBLEMİNİN DERİNİNE ARAMA YÖNTEMİ İLE ÇÖZÜMLENMESİ

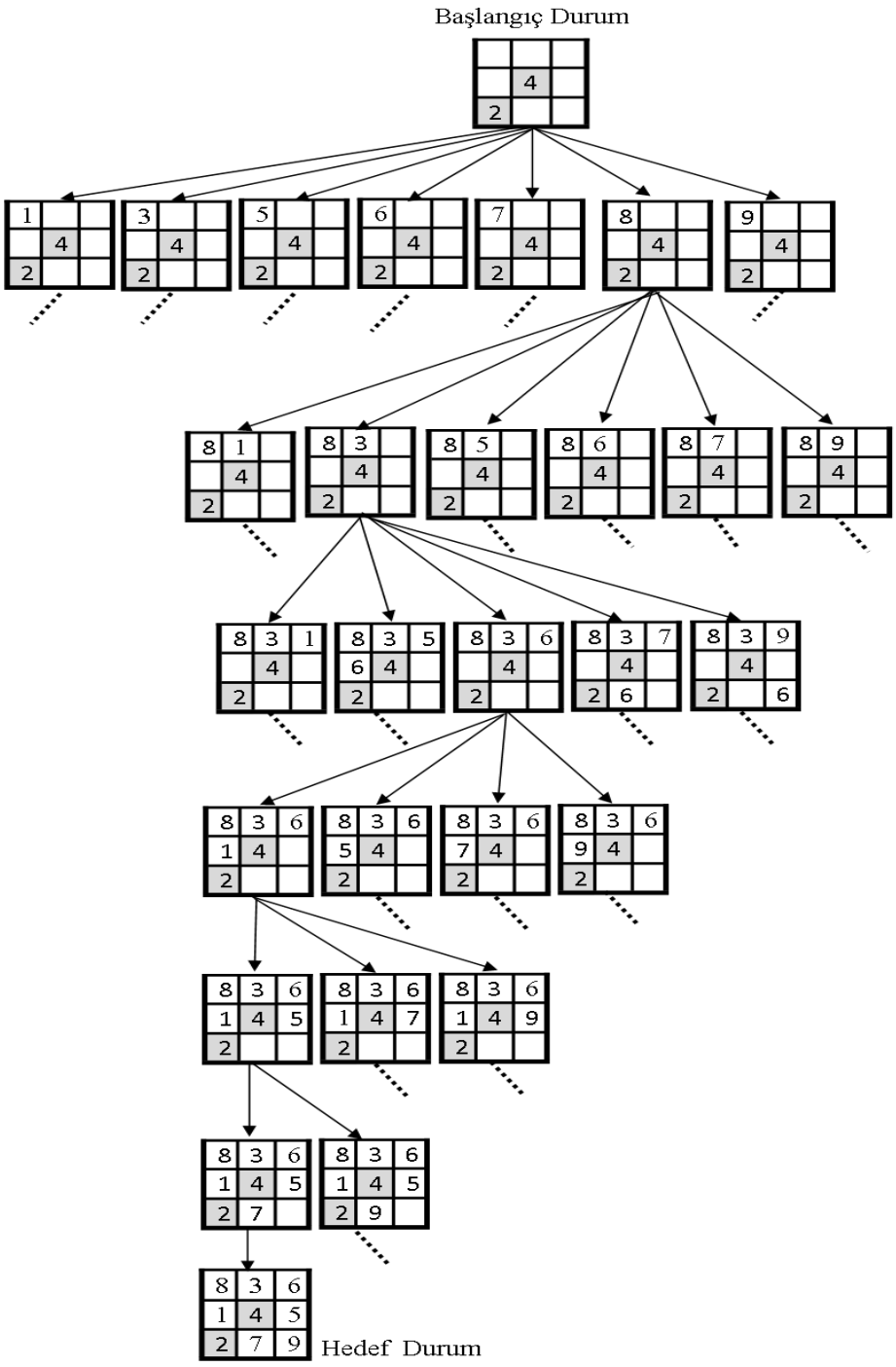
Standart Sudoku problemleri için derinine arama yöntemi en etkili metot olmasının yanında, Donald Knuth'un tam matris kapsamlarının çözümü için Bağlantılarda Gezme algoritması da popülerliğe sahiptir.

Gidilebilecek maximum derinlik seviyesi bilindiğinde, derinlik öncelikli arama (DFS) metodu kullanmak, diğer yapay zeka algoritmalarına göre, problemin sonucuna daha hızlı erişmeye imkan sağlamaktadır. Sudoku problemi algoritmasında derinlik seviyesi, seçilen oyun seviyesiyle doğru orantılıdır. Oyun zorluk düzeyi arttıkça gidilebilecek maximum derinlik seviyesi de artacaktır. Örneğin, oyunun düzeyi "zor" olarak seçildiyse, her bölgede 2 adet kutucuk rastgele belirlenmiş olacağından geriye kalan 7 adet kutucuk, derinine aramanın 7 seviyeden oluşacağını göstermektedir.

4.3.1 SUDOKU PROBLEMİNİN DURUM UZAYI

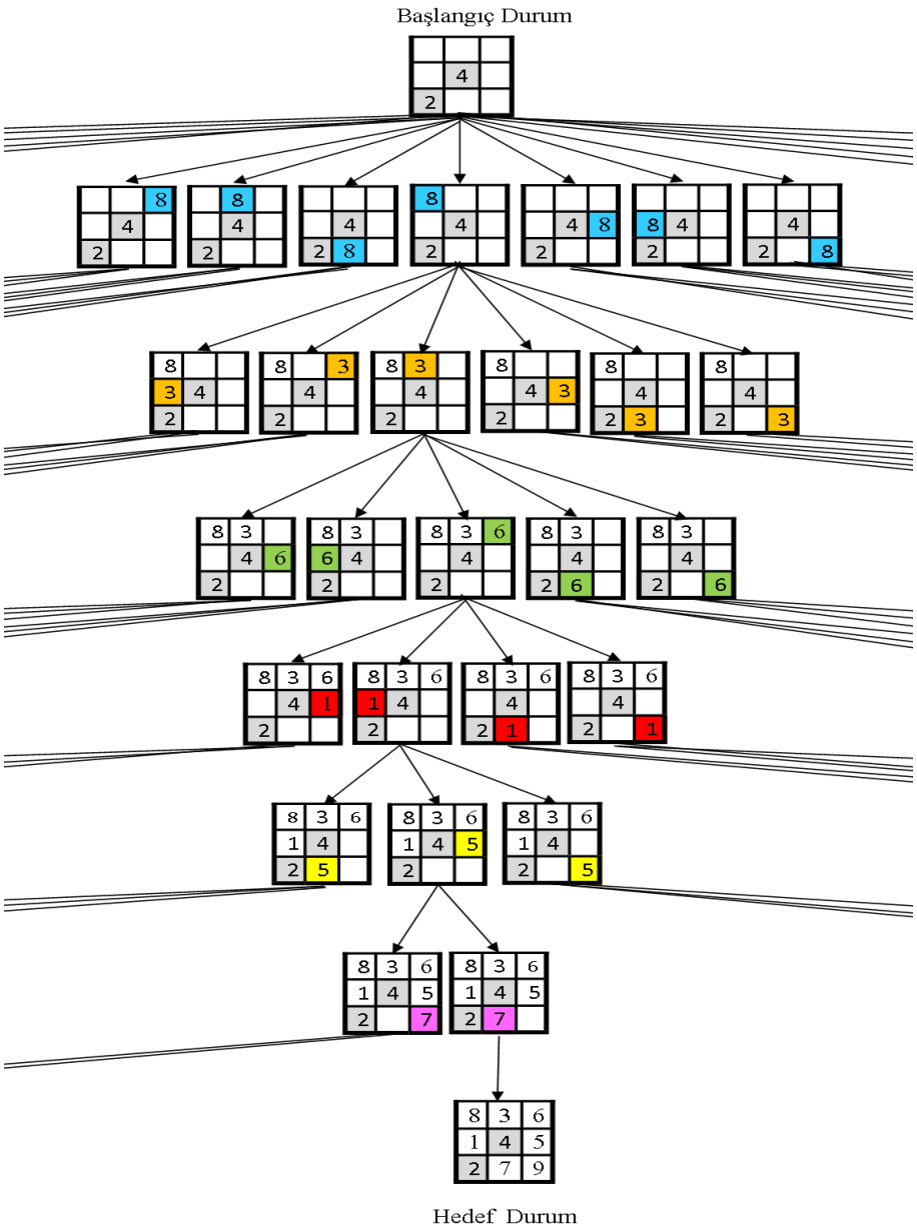
Eğer bir problemi çözüyorsak, genellikle çözümler arasındaki en iyi olanı arıyoruz demektir. Mümkün tüm çözümlerin uzayına (istenen çözümün aralarından bulunduğu çözümler kümesi) arama uzayı (durum uzayı) adı verilir. Arama uzayındaki her nokta bir olası çözümü temsil eder. (Kalaycı, 2006, s:69)

Şekil III-13'de Sudoku probleminin durum uzayı gösterilmiştir. Buradan anlaşılacağı üzere, 9x9 luk matrisin 3x3 lük 0. Bölgeye ait olası başlangıç durumu belirtilmiş, oyuncunun klavyeden girmesi muhtemel rakamların varyasyonu başlangıç durumun(kök) dallarını (çocuk düğüm) oluşturmuş ve hedef durum gösterilmiştir.



Şekil III-13 : Sudoku Probleminin Durum Uzayı

4.3.2 SUDOKU PROBLEMİNİN ARAMA AĞACI



Şekil III-14 : Sudoku Probleminin Arama Ağacı

4.3.3 SUDOKU PROBLEMİNİN DERİNİNE ARAMA YÖNTEMİNE GÖRE ÇALIŞMA ŞEKLİ

Şekil III-14' de görüldüğü üzere başlangıç durumdan hedef duruma ulaşmak için her dalda arama yapılmaktadır. Derinine Arama Yönteminde gidilebilecek en uç düğüme kadar gitmek mantığı esastır. Sudoku probleminin sadece bir bölgesinin sonuca ulaşması genel problemin çözümlenmesi demektir. Derinine arama başladığı andan itibaren başlangıç noktasından hareketle, buna bağlı dallardan arama yapmaya başlanır. Arama her dalın en uç kısmındaki yaprak düğüme kadar devam eder. Gelinen nokta, hedef durumla kıyaslanır. Eğer sonuca ulaşılamadıysa, bulunulan noktaya nereden gelindiyse geliş yolu takip edilerek bulunulan daldaki diğer açılmamış düğüme geçilir. Eğer tüm durumlar taranmış ve sonuç bulunamamışsa arama başarısız olmuştur demektir.

4.3.4 SUDOKU PROBLEMİNİN DURUM UZAYINDA OLUŞAN DÜĞÜM SAYISININ HESAPLANMASI

Derinlik	**Dallanma Sayısı**
Derinlik 1	49
Derinlik 2	49.36 = 1764
Derinlik 3	49.36.25 = 44100
Derinlik 4	49.36.25.16 = 705600
Derinlik 5	49.36.25.16.9 = 6350400
Derinlik 6	49.36.25.16.9.4 = 25401600
Derinlik 7	49.36.25.16.9.4.1 = 25401600
TOPLAM	**57.905.064**

Şekil III-16: 9x9 luk Sudoku Probleminin 3x3 lük Matrisinin Durum Uzayında Oluşan Dallanma Sayısı

Derinine Arama yönteminde ***b*** dallanma sayısı ve ***m*** derinlik seviyesi olmak üzere, ***durum uzayında oluşan düğüm sayısı= bm + 1*** idi.

Şekil II-21 ve Şekil II-22'de görüldüğü üzere Sudoku probleminin durum uzayında oluşturulan derinine arama ağacında derinlik seviyesi 7 ve dallanma sayısı Şekil II-20'de gösterildiği üzere 57.905.064 olduğundan;

toplam düğüm sayısı 7 x 57.905.064 + 1 = 405.335.449 $\approx$ $\mathbf{10^8}$ dir.

4.4 DURUM UZAYI KARMAŞIKLIĞINA GÖRE OYUNLARIN SINIFLANDIRILMASI

Durum uzayı karmaşıklığı, verilmiş başlangıç durumdan erişilmesi mümkün olan durumlar sayısı şeklinde ifade edilmektedir. Oyun ağacı karmaşıklığı ise verilmiş durumdan, çözüm aşamasında oluşan uç düğümler sayısı şeklinde tanımlanmaktadır. Bu oyunların yapay zeka yöntemlerini kullanmadan çözülemeyeceği aşikardır. Bilgisayarlar, en azından oyunlarda insanlarla kıyaslanabilecek bir rekabete girebilmektedir. (Nabiyev, 2010)

Çeşitli oyunlar için daha önceden hesaplanmış olan bazı oyunların *durum uzayı* ve *oyun ağacı* karmaşıklığı Şekil III-17'de gösterilmiştir.

Bu tez çalışmasında Sudoku probleminin durum uzayı ve oyun ağacı karmaşıklıkları hesaplanarak Şekil III-17'deki tabloya eklenmiştir.

Oyunlar	**Durum Uzayı karmaşıklığı**	**Oyun Ağacı Karmaşıklığı**
Awari	10^{12}	10^{32}
Dama (Checkers)	10^{22}	10^{31}
Satranç (Chess)	10^{46}	10^{123}
Çin Daması (Chinese Chess)	10^{48}	10^{150}
Dörtlü bağlama (Connect-	10^{14}	10^{21}
Dakon-6	10^{15}	10^{33}
Zorbacı (Domineering) (8x8)	10^{15}	10^{27}
Dama (Draughts)	10^{30}	10^{54}
Go (19x19)	10^{172}	10^{360}
Moku (15x15)	10^{105}	10^{70}
Hex (11x11)	10^{57}	10^{98}
Kalah (6,4)	10^{13}	10^{18}
Othello	10^{28}	10^{58}
Pentominoes	10^{12}	10^{18}
4*4*4 tic-tac-toe, Cubic	10^{30}	10^{34}
Renju (15x15)	10^{105}	$10^{7\circ}$
Shogi	10^{71}	10^{226}
Sudoku (Diamond) (9x9)	**10^{8}**	**10^{72}**

Şekil III-17: Çeşitli Oyunlar İçin Durum Uzayı ve Oyun Ağacı Karmaşıklığı (Nabiyev, 2010)

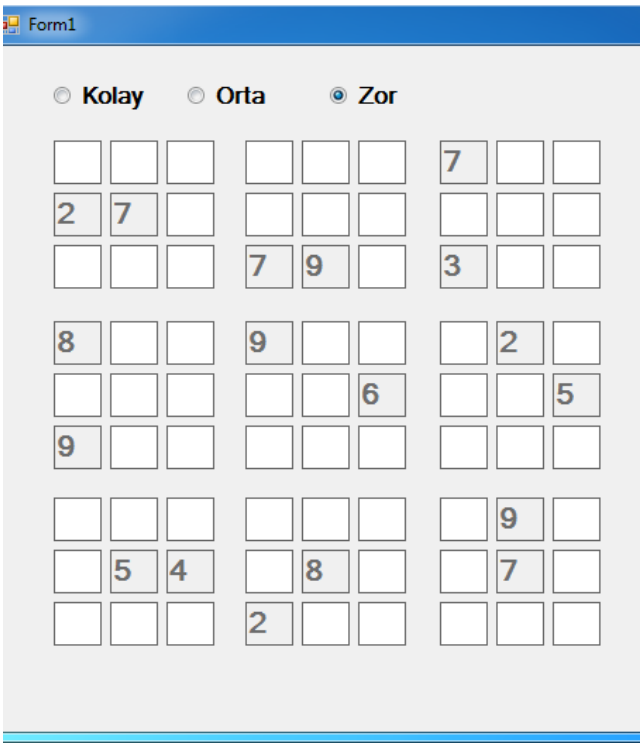
Oyunların gruplandırılması, oyun sırasında durum uzayının boyutlarının değişmesine göre belirlenmektedir. *Yakınsamalı oyunlarda* oyun, alandaki birçok örnek veya taşla başlar ve oyun ilerledikçe bu taşlar tahta üzerinden silinir. *Uzaksamalı oyunlarda* ise bunun tersi, boş veya boşa yakın bir durumla oyuna başlanır ve oyun süresince taşlar eklenir. (Nabiyev, 2010) Örneğin satranç yakınsamalı, sudoku uzaksamalı oyunlardandır.

4.5 SUDOKU OYUNU PROBLEMİNİN ÇÖZÜMLENMESİNDE YENİ BİR YAKLAŞIM MODELİ

Sudokunun ciddi problem çözümlerini oluşturmak için hızlı hesaplamalar yaparak olabildiğince etkili çözümler bulmaya dayalı derinine arama algoritması kullanılmaktadır. Önceden hesaplandığı üzere, DFS yöntemine göre Sudoku probleminin durum uzayında oluşan düğüm sayısı yaklaşık 10^8 'dir, yani 100.000.000 durum denenmektedir.

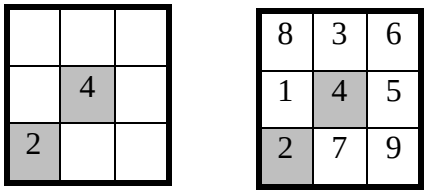
Bu tez çalışmasında oluşturulan model algoritmada sudoku probleminin zorluk derecesini belirleyip olabildiğince insanların çözüm metoduna benzetmeye çalışılmıştır. Bu yöntem önceden açılmamış hücrelere 1'den 9'a kadar sayılar rastgele atanarak kontrol yaptırılmıştır. "Zor" seviyede oynanan bir oyunda, ızgarada önceden açılan hücre sayısı 18 olup, geriye kalan 63 hücre için teker teker denenen durum sayısı 63x9=567'dir.

Şekil III-18)Sudoku Probleminin Başlangıç Durumlarından Bir Örnek

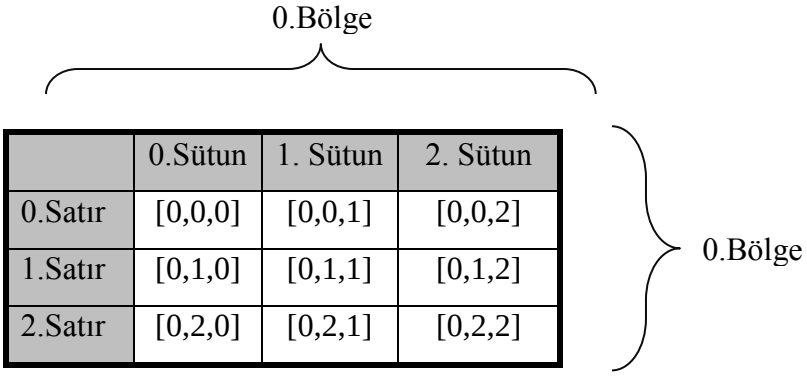


Şekil III-19) Sudoku Probleminde Başlangıç Duruma Göre Belirlenen Hedef Durumlardan Bir Örnek

Yukarıdaki şekillerden de anlaşılacağı üzere, Sudoku oyununda sadece 1 tane başlangıç durum ve hedef durum yoktur. Oyuna başlanırken, rastgele atanmış kilit rakamlar (arka plan gri olan) oyunun başlangıç durumunu belirlemekte, hedef durum ise başlangıç durumuna göre şekillenmektedir. Dolayısıyla Sudoku Oyunu için bir durum uzayı oluşturabilmek, oyuna başlanmadan önce rastgele oluşturulan rakamlara bağlıdır. Şekil III-20'de 9x9 luk matrisin 3x3 lük ilk bölgesinde, başlangıç duruma göre oluşturulan hedef durum gösterilmiştir.



Şekil III-20: 9x9 'luk Başlangıç Matrisinden Alınan 3x3 'lük 0. Bölge Matrisinin Hedef Durumu



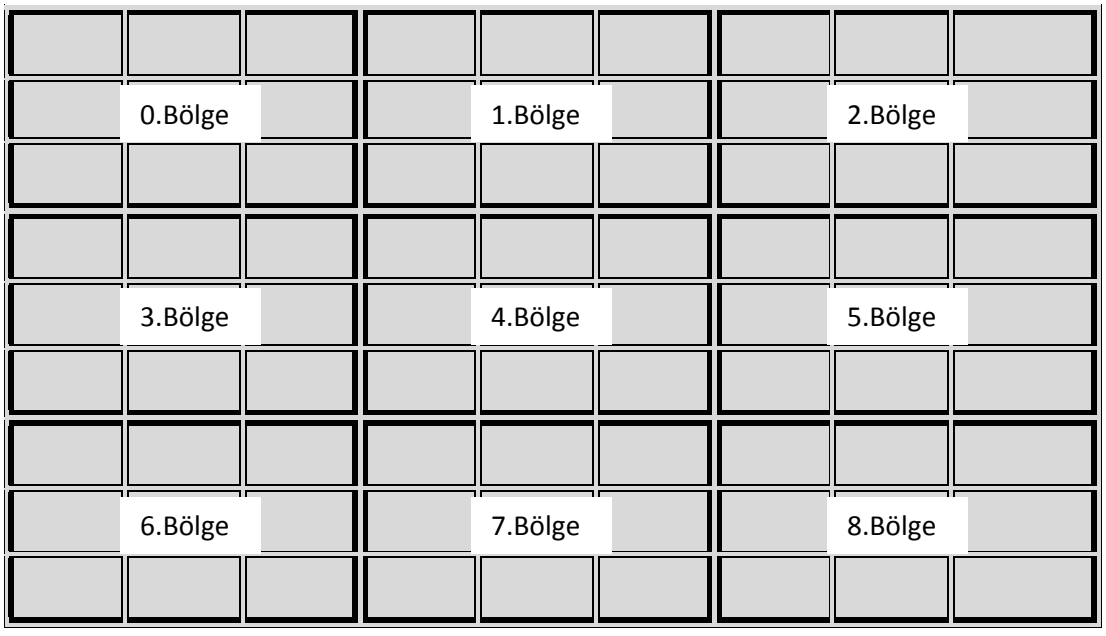
Şekil III-21: 9x9 'luk Başlangıç Matrisinden Alınan 3x3 'lük 0. Bölge Matris Koordinatları

Yukarıdaki şekle göre dizi [0,0,0] ifadesinde ilk indis, bölge numarası, ikinci indis satır numarası, üçüncü indis ise sütun numarasını ifade etmektedir. Bu ifadeyi aşağıdaki şekilde yazabiliriz:

dizi [**0,0,0**] = dizi [**bölge** numarası, **satır** numarası, **sütun** numarası]

[0,0,0]	[0,0,1]	[0,0,2]	[1,0,0]	[1,0,1]	[1,0,2]	[2,0,0]	[2,0,1]	[2,0,2]
[0,1,0]	[0,1,1]	[0,1,2]	[1,1,0]	[1,1,1]	[1,1,2]	[2,1,0]	[2,1,1]	[2,1,2]
[0,2,0]	[0,2,1]	[0,2,2]	[1,2,0]	[0,2,1]	[0,2,2]	[2,2,0]	[2,2,1]	[2,2,2]
[3,0,0]	[3,0,1]	[3,0,2]	[4,0,0]	[4,0,1]	[4,0,2]	[5,0,0]	[5,0,1]	[5,0,2]
[3,1,0]	[3,1,1]	[3,1,2]	[4,1,0]	[4,1,1]	[4,1,2]	[5,1,0]	[5,1,1]	[5,1,2]
[3,2,0]	[3,2,1]	[3,2,2]	[4,2,0]	[4,2,1]	[4,2,2]	[5,2,0]	[5,2,1]	[5,2,2]
[6,0,0]	[6,0,1]	[6,0,2]	[7,0,0]	[7,0,1]	[7,0,2]	[8,0,0]	[8,0,1]	[8,0,2]
[6,1,0]	[6,1,1]	[6,1,2]	[7,1,0]	[7,1,1]	[7,1,2]	[8,1,0]	[8,1,1]	[8,1,2]
[6,2,0]	[6,2,1]	[6,2,2]	[7,2,0]	[7,2,1]	[7,2,2]	[8,2,0]	[8,2,1]	[8,2,2]

Şekil III-22: 9x9'luk Izgarada Kullanılan Matris Koordinatları



Şeki III-23: 9x9' luk Izgarada Bölgelerin Numaralandırılması

4.5.1 MODEL ALGORİTMANIN İŞLETİLMESİ

Sudoku yazılımı, Windows ortamında Microsoft'un .NET Framework 2.0 kitaplıkları kullanılarak C# dili ile yazılmıstır. Yazılımın çalıştırılabilmesi için Microsoft .NET Framework 2.0'ın bilgisayara kurulması gerekmektedir.

Progamda herhangi bir sınıf yapısı kullanılmamış, global ve local değişkenler oluşturularak parametrik değerler döndürülmüştür. Oyun Kuralları başlığı altında belirtilen tüm kurallar, fonksiyon oluşturularak kontrol edilmiş olup, her satırda rakamların tek olması kuralı yataykontrol(), her sütunda rakamların tek olması kuralı düseykontrol(), her bölgede rakamların tek olması kuralı ise kupicikontrol() fonksiyonu ile sağlanmıştır.

KURAL-1: Her rakam, her satır üzerinde yalnızca bir kez olmalıdır.

```
        private void yataykontrol(int ii, int jj, int
deger)
        {
            satir= ii -(ii % 3);

            for (int i = satir; i < satir + 3; i++)
            {
                for (int k = 0; k < 3; k++)
                {
                    if (dizi[i, jj, k,0] == deger)
                    {
                        kontrol = false;
                        break;
                    }
                }

            }
```

Her satırda aynı rakamın bir kez kullanılabileceği şartını sağlayabilmek için yukarda kaynak kodları gösterilen yataykontrol() fonksiyonu oluşturulmuş, parametre olarak **bölge kodu ii** ve bölge içindeki **satır sırası jj** alınmıştır. Alınan bölge kodu ii -(ii % 3) işlemi ile bölgenin genel sırasındaki ilk yeri belirlenmiş, daha sonra elde edilen bölge sırasına göre her o bölgede bulunan bölgelerin belirtilen satırında üretilen sayının daha önce kullanılıp kullanılmadığı tespit edilmiştir.

KURAL-2: Her rakam, her sütun üzerinde yalnızca bir kez olmalıdır.

```
private void dikeykontrol(int ii, int kk, int deger)
        {
            sutun = ii % 3;
            for (int i = sutun; i < sutun + 7; i = i + 3)
            {
                for (int j = 0; j < 3; j++)
                {
                    if (dizi[i, j, kk,0] == deger)
                    {
                        kontrol = false;
                        break;
                    }
                }
            }
        }
```

Her sütunda aynı rakamın bir kez kullanılabileceği şartını sağlayabilmek için yukarIda kaynak kodları gösterilen dikeykontrol() fonksiyonu oluşturulmuş, parametre olarak **bölge kodu ii** ve bölge içindeki **sütun sırası kk** alınmış, alınan bölge kodu ii % 3 işlemi ile bölgenin genel sırasındaki ilk bölge belirlenmiştir. Daha sonra elde edilen bölge sırasına göre her o bölgede belirten sütunlarında üretilen sayının daha önde kullanılıp kullanılmadığı tespit edilmiştir.

KURAL-3: Her rakam, her bölge üzerinde yalnızca bir kez olmalıdır.

```
private void kupicikontrol(int ii, int deger)
        {
            for (int j = 0; j < 3; j++)
            {
                for (int k = 0; k < 3; k++)
                {
                    if (dizi[ii, j, k,0] == deger)
                    {
                        kontrol= false;
                        break;
                    }
                }
            }
       }
```

Her bölgede aynı rakamın bir kez kullanılabileceği şartını sağlayabilmek için yukarda kaynak kodları gösterilen kupicikontrol() fonksiyonu oluşturulmuş, parametre olarak alınan bölge numarası için satır ve sütün kontrolleri yapılarak üretilen sayının daha önde kullanılıp kullanılmadığı tespit edilmiştir

```
private void button1_Click(object sender, EventArgs e)
        {
            kontrol = false;
            dizibosalt();

            sayac = 0;
            while (sayac < 81)
            {
                x = sayac % 9;
                i = (sayac - x) / 9; // bölge sayısı
                k = x % 3; // sutun
                j = (x - k) / 3; //satır

                kontrol = false;
                while (!kontrol)
                {
                    kontrol = true;
                    devam = true;
                    sayi = rasgele.Next(1, 10);

                    dizi[i, j, k, sayi] = 1;

                    if (dizikontrol(i, j, k))
                    {
                        dizisifirla(i, j, k);
                        sayac = sayac - 2;
                        devam = false;
                    }
                    else
                    {
                        kupcicikontrol(i, sayi);
                        yataykontrol(i, j, sayi);
                        dikeykontrol(i, k, sayi);
                    }
                }
                if (devam)
                    dizi[i, j, k, 0] = sayi;
                sayac++;
            }
            seviyeBelirle();
            doldur();
        }
```

Yukarıda Yeni Oyun butonuna tıklayınca çalışan program kodları gösterilmiştir. Burada kullanılan dizi değişkenleri aşağıdaki gibidir.

dizi [i, j, k, sayi] = 1 veya 0;

dizi [bölge sayısı, satır sayısı, sütun sayısı, üretilen sayı] = 1 veya 0

Diziye 1 atanmışsa, üretilen sayı daha önce kullanılmış; 0 atanmışsa önceden kullanılmamış anlamındadır.

Örneğin;

dizi[5,2,0,7] = 1 ifadesi, 5.bölgenin 2. Satırının ilk hanesine 7 değeri daha önce kullanılmış olduğundan (dizi değeri 1 olduğu için) bu hanede kullanılamaz anlamını taşımaktadır.

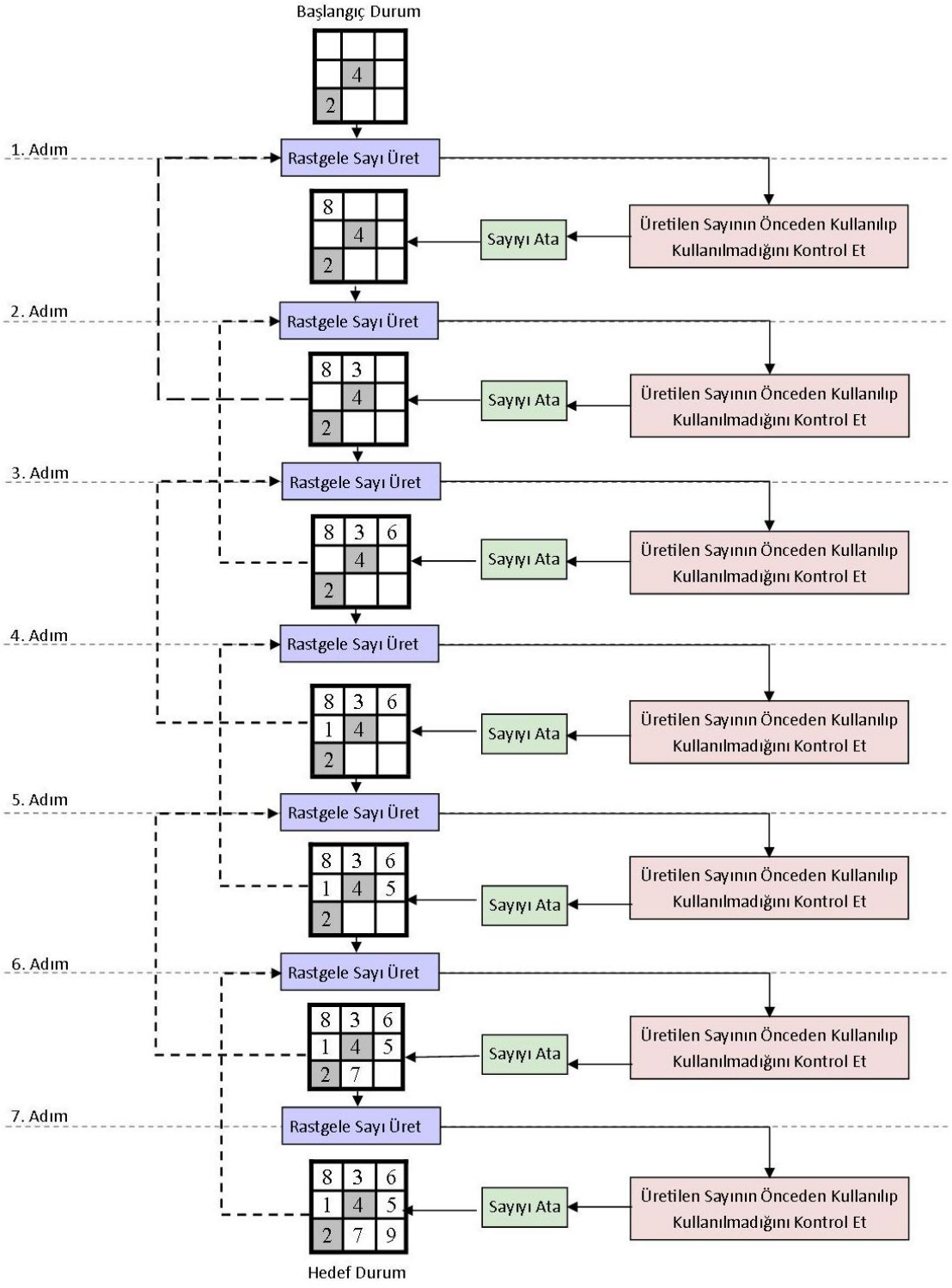
Yukarıdaki program kodlarında Sudoku Oyunu'nunun temel çalışma mantığı gösterilmiş olup, sudoku oyunu 9x9 olmak üzere 81 adet hücreden oluştuğundan bir **sayac** değişkeniyle döngü kurulmuş, bu döngünün içinde sayacın 0'dan 80'e kadar alacağı değerler i, j, k yani bölge, satır ve sütun değerlerine atanmıştır. Kontrol değişkenimize false değeri atıyoruz. Kontrol değeri true değerini alana kadar şu işlemler yapılmaktadır:

1. Rastgele bir sayı üretiliyor.
2. Üretilen sayının kullanıldığına dair bilgi 1 olarak diziye atılıyor.
3. Dizikontrol fonksiyonu ile dizi için üretilen tüm değerlerin kullanılıp kullanılmadığı kontrol ediliyor. Eğer kullanıldıysa bulunan hücre için kullanılan değerler 0 lanıyor. Sayac değeri 2 azaltılıp, devam değişkenine false değeri atanıyor. Dizikontrol fonksiyonundan dönen değer false ise bölge, satır ve sütun kontrolleri yapılıyor.

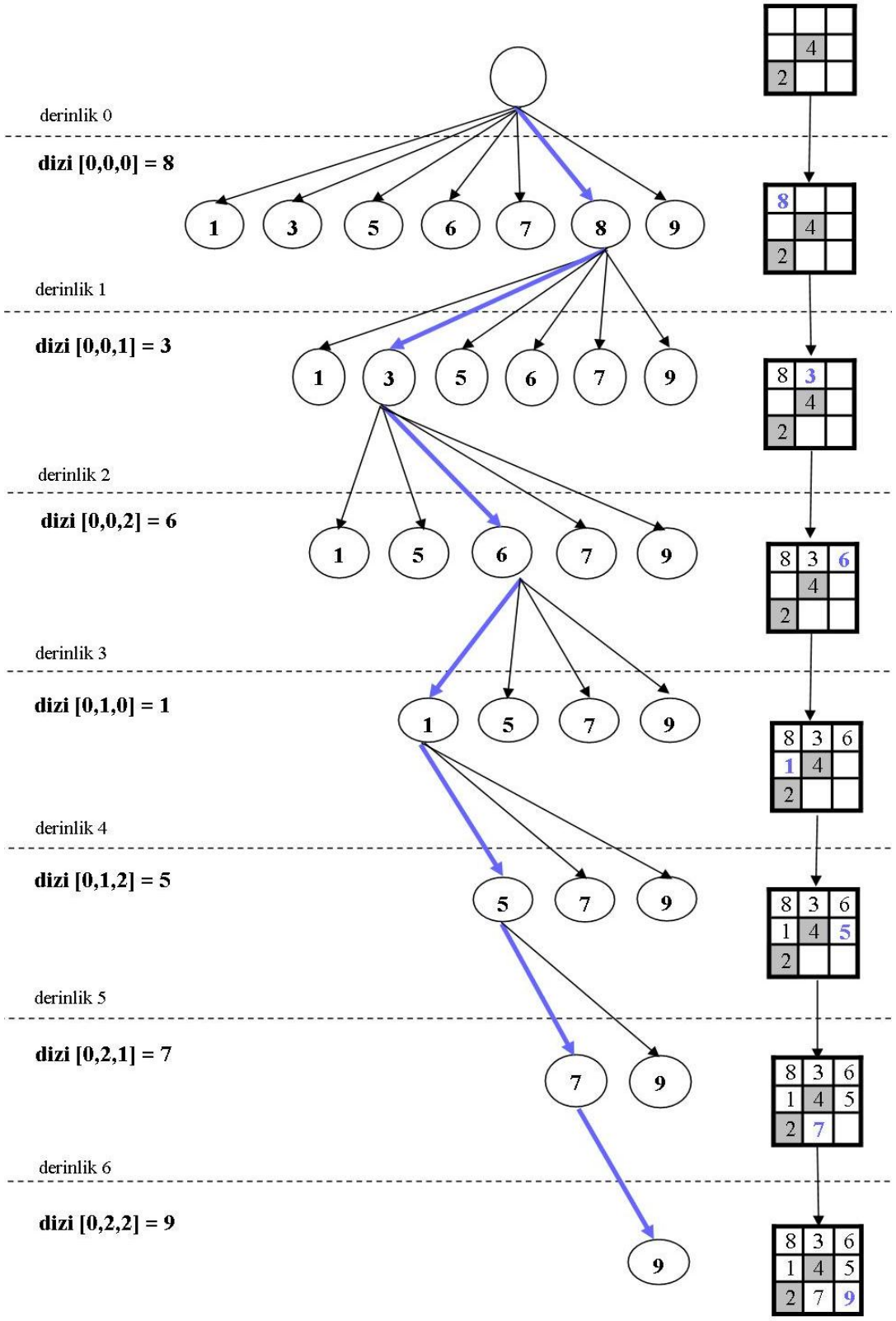
4. Devam değeri true ise dizinin 4. hanesinin 0. ıncı indexine üretilen sayı atanıyor.

5. Kolay, zor ve orta seviye belirleniyor ve doldur fonksiyonu ile dolduruluyor.

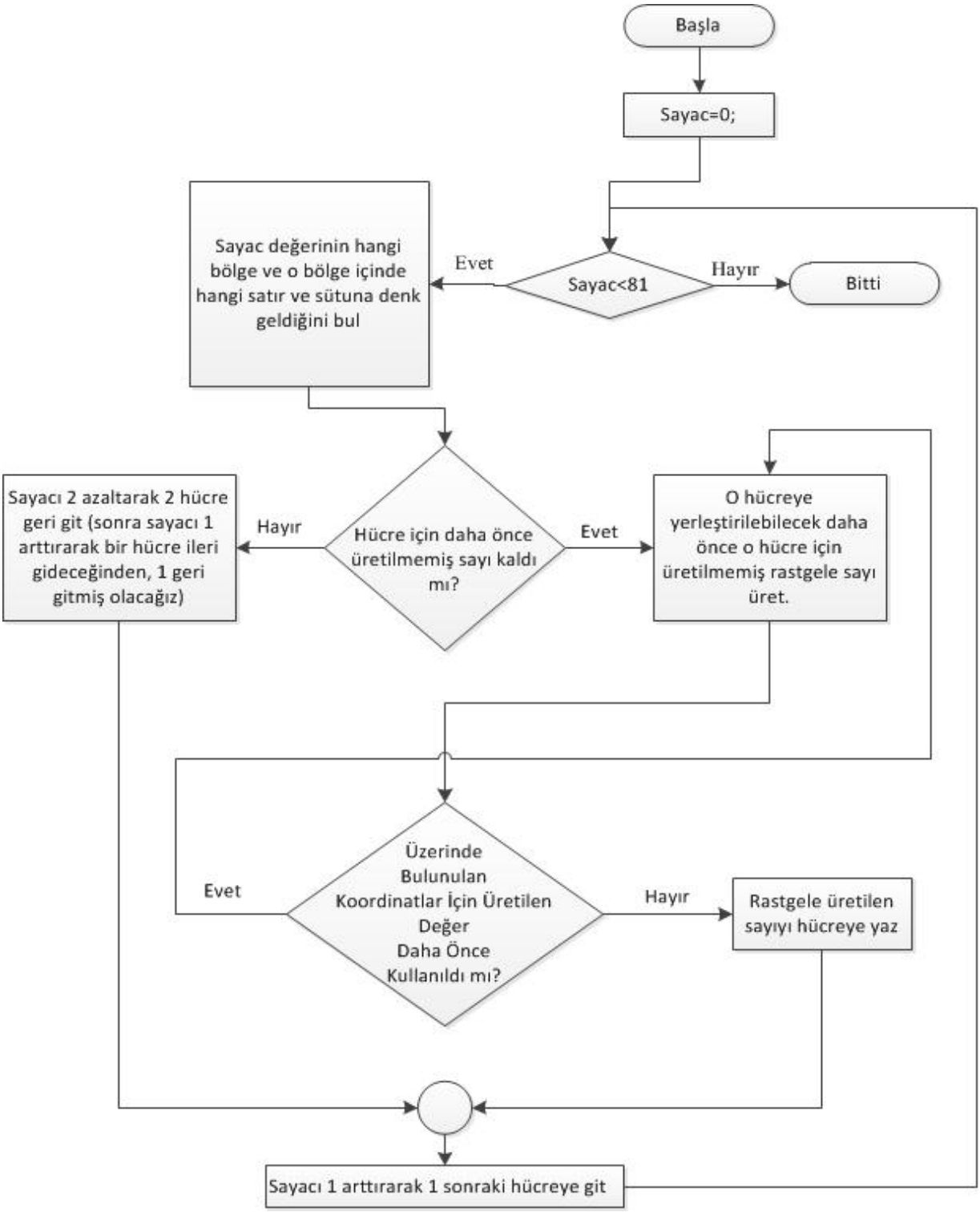
Bir kutu için tüm değerlerin denenip denenmediği kontrol ediliyor. True dönerse dizi için bütün değerler denenmiş kabul edilmektedir. Oyuncunun seçtiği kolay, orta ve zor seçeneklerine göre her bölgede sabit sayıda hangi hücrelerin gösterileceği rastgele belirlenir. ilk olarak diziSeviye adlı dizinin içi sıfırlanmaktadır. Daha sonra derece sayısınca bir döngü içinde rastgele sayılar üretilerek hangi hücrelerin açılacağı diziSeviye adındaki diziye atılmıştır. Kolay seviyesinde her bölgede 4 er tane olmak üzere toplam 36 hücre rastgele belirlenir. Orta seviyesinde her bölgede 3 er tane olmak üzere toplam 27 hücre, Zor seviyesinde her bölgede 2 şer tane olmak üzere toplam 18 hücre rastgele belirlenmiştir.



Şekil III-24: Model Algoritma İle Sudoku Probleminin Çalışma Şekli



Şekil III-25: Model Algoritmada Dizi Kullanımı



Şekil III-26: Model Algoritmanın Akış Diyagramı

BÖLÜM IV

SONUÇ VE ÖNERİLER

Sudoku oyunu, tek kişilik oynanan bir oyun olup, oyuncu kendi kişisel stratejilerini ve netice fonksiyonlarını bildiğinden tam bilgili (complete information) oyun grubuna girmektedir. Ayrıca Sudoku oyununda kesinlikle şans faktörü yer almadığından, oyun tamamen bulunulan pozisyona göre karar vermeye odaklıdır. Dolayısıyla Sudoku Oyunu aynı zamanda karara dayalı oyun grubunda yer alır.

Oyunların algoritma yapıları, oyun sırasında durum uzayının boyutlarının değişmesine göre belirlenmektedir. *Yakınsamalı oyunlarda* oyun, alandaki birçok örnek veya taşla başlar ve oyun ilerledikçe bu taşlar tahta üzerinden silinir. *Uzaksamalı oyunlarda* ise boş veya boşa yakın bir durumla oyuna başlanır ve oyun süresince taşlar eklenir. Dolayısıyla Sudoku Oyunu uzaksamalı oyunlar grubuna girmektedir.

Standart sudoku problemlerinin en etkili çözüm yolu derinine arama yöntemi olarak bilinmektedir. Bu arama metodu doğru sonucu bulana kadar her olasılığı deneme fikrine dayanır. Yani tüm çözümler kümesinde derinlemesine bir arama yapmak anlamına gelir. Arama esnasında eğer denenen yol tıkandıysa bu yolun giriş noktasına, gelinen yoldan geri dönülür. Yani diğer bir alternatif yola, girilecek yere gelinir ve o denenir. Eğer tüm durumlar taranmış ve sonuç bulunamamışsa arama başarısız olmuştur demektir.

Bu tez çalışmasında sudoku probleminin çözümüne olanak sağlayan model algoritma, kısmen derinine arama yapmaktadır. Ancak burada kullanılan model algoritmada arama yapılan kolda, her aşamada yinelemeli bir yapı kullanılmıştır. Eğer model algoritmada o yoldaki tüm alternatifler taranarak sonuç bulunamadıysa, bir önceki kontrol noktasına dönerek diğer daha kapsamlı alternatif yol denenmiştir. Bu işlem sürekli kendini çağıran yani yinelemeli bir yapı kullanılarak oluşturulmuştur. Bu işlemde her duruma fazladan bir değişken atanır ve bu durumun

ulaştığı durumların değerleri bu durumun bu değişkenine atanır. Bu değer tutularak bir sonraki yinelemeli çağırılır. Aramada hızlandırma yapmak için bir değer seçildiğinde yinelemeli çağırma yapılmadan algoritma hem ilerideki kontrol noktalarından değerleri uyuşmazlık göstereni siler hem de sınırlamalara bakarak hangi alternatifin taranması gerektiğini belirler.

Derinine arama algoritması, mevcut durum uzayında (10^8 durum = 100 milyon durum) hedef duruma ulaşabilmek için tüm olası durumları denemektedir. Bu tez çalışmasında, önceden açılmamış hücrelere 1'den 9'a kadar sayılar rastgele atanarak kontrol yaptırılmıştır. "Zor" seviyede oynanan bir oyunda, ızgarada önceden açılan hücre sayısı 18 olup, geriye kalan 63 hücre için teker teker denenen durum sayısı 567'dir.

Sudoku oyununda oyuncunun hücrelere girdiği sayıların, arka planda oyun kurallarına uygun sayı olup olmadığı kontrol edilmektedir. Girilen sayılar koordinatlarına uygun olmadığı takdirde, sürekli geri adım atılarak geriye doğru arama yaptırılmış, random atanan sayıların doğrusu bulununcaya kadar bu işlemler devam etmiş ve sonuçta 81 adet hücreye oyun kurallarına uygun sayılar yerleştirilmiştir.

Sudoku oyunu, insan tarafından oynaması basit bir oyun olsa da, aslında geri planda oğun ağacı karmaşıklığının diğer bilgisayar oyunlarıyla kıyaslandığında göz ardı edilemez sayıda düğüme sahip olduğu aşikardır.

Buradan hareketle, oyun programlamada kullanılan algoritmaların, oyunların niteliklerine göre iyi seçilmesi ve hatta gerekli görüldüğünde var olan algoritmaların yardımıyla özgün bir algoritma oluşturulmalıdır denilebilir.

KAYNAKÇA

Adlassng, Klaus-Peter, Artificial-Intelligence-Augmented Systems , Artificial Intelligence In Medicine, 1st. Internetional Workshop, 24, 1-4., 2012.

Amir, A., Problem Solving By Artificial Intelligence, 2007.

Aktan, C. C., & Bahçe, A. B., Kamu Tercihi Perspektifinden Oyun Teorisi, Ankara, 2007

Baştan, S., Yapay Zeka, Yeni İletişim Teknolojileri ve Örgütsel Değişim: Akıllı Örgüte Doğru, Yönetim ve Ekonomi, Cilt:10 Sayı:1, 2003

Benzer, A. İ., Yapay Zeka Uygulamalarında Kullanılan Arama Algoritmalarının Kıyaslanması, Ankara Üniversitesi Fen Bilimleri Enstitüsü Yüksek Lisans Tezi, 2007

Brams, S. J., *Game Theory*, New York University, December, International Encyclopedia of The Social Sciences, 2005

Charniak, E. and McDermott, D., "Introduction to Artificial Intelligence", Addison-Wesley Publishing Company, USA, 260-267 ,1985.

Chen, Ming-Syan, Shin, Kang G., "Depth-First Search Approach for Fault Tolerant Routing in Hypercube Multicomputers", IEEE Transactions on Parallel and Distributed Systems, Vol.1, No:2, ss. 152-159, 1990

Coppin, B., "Artificial intelligence illuminated", Jones and Bartlett Publishers, USA, 70,75-81,90-91,103-110, 2004.

Çölkesen, R., "Veri Yapıları ve Algoritmalar", Papatya Yayıncılık, s.s 346, 2000.

E., Murat Esin, Erdoğan, S. Z., "Self Cloning Ant Colony Approach and Optimal Path Finding", Proceedings of The Euro American Conference on Telematics and Information Systems, EATIS 2006, Kolombiya, ISBN:958-8166-38-1, 2006

Erkalkan, E., Esnek Programlama Yaklaşımları ile Oyun Geliştirme, Marmara Üniversitesi Fen Bilimleri Enstitüsü Yüksek Lisans Tezi, 2010.

Feigenbaum, E.A. An Interview. Expert Systems. 6 (2) s. 112-115, 1989

Fogel, D.B., An Evolutionary Approach to the Traveling Salesman Problem, Biological Cybernetics, 60:139-144, 1988

Gürbüzer, G., Yapay Zeka Yöntemleriyle Oyun Geliştirme, Kocaeli Üniversitesi Fen Bilimleri Enstitüsü Yüksek Lisans Tezi, 2008

Grimaldi, Ralph P., Discrete and Combinatorial Mathematics, Addison Wesley, 5. Baskı., 2003

Gültekin, Ö., Satranç ve Satrancın Yapay Zeka Tartışmalarındaki Yeri, Journal of İstanbul Kültür University, s. 119-128, 2006

Güzel, M. S., Altı Eksenli Robot Kolun Hareketsel Karakteristiğinin Görsel Programlanması ve Gerçek Zamanlı Uygulama, s. 72-76., Ankara Üniversitesi Fen Bilimleri Enstitüsü Yüksek Lisans Tezi, 2008

Görz, G., Bernhard, N., Artificial Intelligence, Frankfurt, Amazon Press, 2005

Halaç, A., Erbil, T., Falay, T., Yapay Zeka , PC NET Dergisi, Sayı: 59., 2002

Holland, J., Adaption in Natural and Artificial Systems, University of Michigan Press, 1975

Ibrahim, Muhammed A.M., Xinda, Lu, Rwakarambi, J.M., "Parallel Execution of An Irregular Algorithm Depth First Search (DFS) on Heterogeneous Clusters of Workstation", The Proceedings of International Conferences on Info-tech and Infonet ICII 2001 – Beijing, Çin, 2001.

Kalaycı, E.T.,Yapay Zeka Teknikleri Kullanan Üç Boyutlu Grafik Yazılımları İçin "Extensible 3d" (X3d) İle Bir Altyapı Oluşturulması Ve Gerçekleştirimi, Ege Üniversitesi Fen Bilimleri Enstitüsü Yüksek Lisans Tezi s.s 70, 2006.

Kocabaş, Ş., Yapay Zeka Araştırma ve Uygulama Alanları, 2006

Kocabaş, Ş., Öztemel, E., Uludağ, M.& Koç, N., Design of a DIS Agent: The AISim System. In Proceedings of the Sixth Computer Generated Forces and Behavioral Representation, s. 119-124., 1996.

Kelly, A., Decision Making Using Game Theory, Cambridge: Cambridge University Press, s. 1-6., 2003.

Lenat, D.B. & Feigenbaum, E.A., On the thresholds of knowledge. In: Proceedings of the Tenth International Joint Conference on Artificial Intelligence, s. 1173-1182, 1987

Mingers, J., Embodying Information Systems: The Contribution Phenomenology ,Information And Organization, Vol.11., 2001

Nabiyev, V.V., Algoritmalar- Teoriden Uygulamalara, Seçkin Yayıncılık, 2009

Nabiyev, V.V., "Yapay Zeka", Seçkin Yayıncılık, Ankara, s.s 97-98,105 106,113,127-140, 2003.

Nolfi , S., Floreano, D., Evolving Artificial Intelligence , Forum: Trends In Cognitive Science, Vol.5, No.11., 2001

Nooraden, O., A., Dağıtık Yapay Zeka Destekli 3 Boyutlu Domino Oyunu, Ankara Üniversitesi Fen Bilimleri Enstitüsü Yüksek Lisans Tezi, 2011.

Penrose, R., Bilgisayar ve Zeka: Kralın Yeni Usu I, Çeviren: Tekin Dereli., 1998.

Pirim, H., Journal Of Yaşar University, Yapay Zeka, 81-93, 2005.

Reinefeld, A., "Complete Solution of the Eight-Puzzle and the Benefit of Node Ordering in IDA*", IJCAI-93. Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence, USA, 248-253., 1993

Russell, S. & Norvig, P., Artificial Intelligence: A Modern Approach. New Jersey: Prentice Hall., 2003

Sakallı, M, Pearlman, W. A., 2006, "SPIHT Algorithms Using Depth First Search with Minimum Memory Usage", Conference on Information Sciences and Systems, CIPR Technical Report TR-2006-10., 2006.

Stojmenovic, I., Russell, M., Vukojevic, B., "Depth First Search and Location Based Localized Routing and QoS Routing in Wireless Networks", ICPP '00: Proceedings of the Proceedings of the 2000 International Conference on Parallel Processing, IEEE Computer Society, University of Ottawa, Canada, 2000.

Tarım, V., Graf Teorisine Dayalı Web Arayüzlü Yol Problemi Uygulaması, Beykent Üniversitesi Fen Bilimleri Enstitüsü Yüksek Lisans Tezi, 2007.

Tucker, A., "The Computer Science and Engineering Handbook", CRC Press, Florida, 676 - 696 , 1996.

Üstkan, S., Uzman Sistemler-Genel, Sakarya Üniversitesi Adapazarı Meslek Yüksekokulu Çalışması, 2007.

Weixiong, Z., ”State-Space Search”, Springer Publisher, USA, 1-3 ,1999.

E-KAYNAKÇA

[1] http://www.elektrik.gen.tr/icerik/oyun-programlama-ve-yapay-zeka

[2] http://tr.wikipedia.org/wiki/Sudoku

[3] http://www.sudokuoyna.net/nedir.html

[4] http://www.turk3.org/index.php?do=search&story=carpma&subaction=search

[5] http://www.go.metu.edu.tr/nedir1.html

[6] http://www.kutuoyunlari.com/klasik-oyunlar/tavla-nasil-oynanir

[7] http://www.bilkurdu.com/SudokuYeni/Sag.htm

[8] http://www.cogsci.ucsd.edu/%20~batali/%20108b/lectures/search.html%20(2000)

[9] http://www-users.cs.umn.edu/~karypis/parbook/Figures/chap11.pdf

[10] http://sudoku.yazarokur.com/

EK-1: C# Kodları

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;


namespace sudoku
{
    public partial class Form1 : Form
    {
        bool kontrol,devam,diziKont;

        int satir, sutun, sayi, sayac,derece;
        int i, j, k, x;

        int[, , ,] dizi = new int[9, 3, 3,10];
        int[] diziSeviye = new int[81];

        Random rasgele = new Random();

        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            kontrol = true;
        }


-------------------------------------------------------
private void kupcicikontrol(int ii, int deger)
        {
            for (int j = 0; j < 3; j++)
            {
                for (int k = 0; k < 3; k++)
                {
                    if (dizi[ii, j, k,0] == deger)
                    {
                        kontrol= false;
                        break;
                    }
                }

            }

       }
-------------------------------------------------------
```

```
-----------------------------------------------------------
private void yataykontrol(int ii, int jj, int deger)
        {
            satir= ii -(ii % 3);

            for (int i = satir; i < satir + 3; i++)
            {
                for (int k = 0; k < 3; k++)
                {
                    if (dizi[i, jj, k,0] == deger)
                    {
                        kontrol = false;
                        break;
                    }
                }

            }

        }

-----------------------------------------------------------
private void dikeykontrol(int ii, int kk, int deger)
        {
            sutun = ii % 3;
            for (int i = sutun; i < sutun + 7; i = i + 3)
            {
                for (int j = 0; j < 3; j++)
                {
                    if (dizi[i, j, kk,0] == deger)
                    {
                        kontrol = false;
                        break;
                    }
                }
            }
        }
-----------------------------------------------------------
```

```
-----------------------------------------------------------
private void button1_Click(object sender, EventArgs e)
        {
            kontrol = false;
            dizibosalt();

            sayac = 0;
            while (sayac < 81)
            {
                x = sayac % 9;
                i = (sayac - x) / 9; // küp sayısı
                k = x % 3; // sutun
                j = (x - k) / 3; //satır

                kontrol = false;

                while (!kontrol)
                {
                    kontrol = true;
                    devam = true;
                    sayi = rasgele.Next(1, 10);

                    dizi[i, j, k, sayi] = 1;

                    if (dizikontrol(i, j, k))
                    {
                        dizisifirla(i, j, k);
                        sayac = sayac - 2;
                        devam = false;
                    }
                    else
                    {
                        kupcicikontrol(i, sayi);
                        yataykontrol(i, j, sayi);
                        dikeykontrol(i, k, sayi);
                    }
                }
                if (devam)
                    dizi[i, j, k, 0] = sayi;
                sayac++;
            }
            seviyeBelirle();
            doldur();
        }

-----------------------------------------------------------
```

```
-----------------------------------------------------------
private bool dizikontrol(int ii, int jj, int kk)
        {
            bool kont;
            kont = true;

            for (int l = 1; l < 10; l++)
            {
                if (dizi[ii, jj, kk, l] == 0)
                {
                    kont = false;
                }
            }
            return kont;
        }

-----------------------------------------------------------
 private void dizisifirla(int ii, int jj, int kk)
        {
            for (int l = 0; l < 10; l++)
                dizi[ii, jj, kk, l] = 0;
        }
        private void dizibosalt()
        {
            for (int i = 0; i < 9; i++)
                for (int j = 0; j < 3; j++)
                    for (int k = 0; k < 3; k++)
                        for (int l = 0; l < 10; l++)
                            dizi[i, j, k, l] = 0;
        }
```

```
private void seviyeBelirle()
        {

            if (radioButtonKolay.Checked == true)
                derece = 36;
            else
                if (radioButtonOrta.Checked == true)
                    derece = 27;
                else
                    derece = 18;

            for (int i = 0; i < 81; i++)
                diziSeviye[i] = 0;

            for (int j = 0; j < 9; j++)
            {
                for (int i = 0; i < (derece /  9); i++)
                {

                    sayi = rasgele.Next(0, 9);
                    if (diziSeviye[sayi + ( j * 9 )] == 0)
                        diziSeviye[sayi + ( j * 9 )] = 1;
                    else
                        i = i - 1;
                }
            }
        }
```

```
------------------------------------------------------------
private void doldur()
        {
            label1.Visible = false;
            foreach (Control c in this.Controls)
                if (c is TextBox)
                {
                    ((TextBox)c).Text = "";
                    ((TextBox)c).Enabled = true;
                    ((TextBox)c).BackColor = Color.Empty;
                }

            foreach (Control c in this.Controls)
            {

                if (c is TextBox)
                {
                    sayac = Convert.ToInt32(((TextBox)c).Tag) -
1; ;
                    x = sayac % 9;
                    i = (sayac - x) / 9;
                    k = x % 3;
                    j = (x - k) / 3;

                    if (diziSeviye[sayac] == 1)
                    {
                        ((TextBox)c).Text = dizi[i, j, k,
0].ToString();
                        ((TextBox)c).Enabled = false;
                    }
                }
            }

        }

------------------------------------------------------------
private void button2_Click(object sender, EventArgs e)
        {
            doldur();
        }
```

```
private void kontDizi()
        {
            diziKont = true;
            foreach (Control c in this.Controls)
            {
                if (c is TextBox)
                {
                    sayac = Convert.ToInt32(((TextBox)c).Tag) -
1; ;
                    x = sayac % 9;
                    i = (sayac - x) / 9;
                    k = x % 3;
                    j = (x - k) / 3;


                    if (((TextBox)c).Text != dizi[i, j, k,
0].ToString())
                    {
                        diziKont = false;
                        ((TextBox)c).BackColor = Color.Red;
                    }
                    else
                        ((TextBox)c).BackColor = Color.Empty;
                }
            }
            if (diziKont)
                label1.Visible = true;

        }

------------------------------------------------------------
private void button3_Click(object sender, EventArgs e)
        {
            kontDizi();
        }

------------------------------------------------------------
private void button4_Click(object sender, EventArgs e)
        {
            foreach (Control c in this.Controls)
                if (c is TextBox)
                    ((TextBox)c).BackColor = Color.Empty;

        }
```

```
private void button5_Click(object sender, EventArgs e)
        {
             foreach (Control c in this.Controls)
                 if (c is TextBox)
                 {
                     sayac = Convert.ToInt32(((TextBox)c).Tag)
- 1; ;
                     x = sayac % 9;
                     i = (sayac - x) / 9;
                     k = x % 3;
                     j = (x - k) / 3;
                     ((TextBox)c).Text = dizi[i, j, k,
0].ToString();

                 }

        }



    }
}
```

ÖZET

Yüksek Lisans Tezi

SUDOKU PROBLEMİNİN ALGORİTMA TASARIMINA YAPAY ZEKA DESTEKLİ YENİ BİR YAKLAŞIM MODELİ VE UYGULAMASI

Tuğba KARAOĞLU

İstanbul Aydın Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Doç. Dr. Ahmet BABANLI
Ekim, 2012

Yapay Zeka, günümüzde bilgisayar bilimlerinin en gözde dallarından biridir ve yapay zeka bilim dalı, makinelerin zeki davranmalarını sağlamaya çalışarak onların daha çok ve daha çeşitli sorunlarla tek başlarına başa çıkmalarını sağlar. Şüphesiz ki günümüzde yapay zekanın gelişmesindeki payı en yüksek olan sektörlerden biri ise bilgisayar oyunları sektörüdür. Günümüzdeki bilgisayar oyunları dünya satranç şampiyonlarını ve dama ustalarını bile yenebilmektedir. Bu motivasyon ile yola çıkılan bu çalışmada öncelikle yapay zekanın tanımı irdelenmiş, oyunlarda kullanılan yapay zekaya yardımcı arama metodları ortaya konmuş ve sudoku probleminin algoritma tasarımına farklı bir yaklaşım modeli işlenmiş ve bunun uygulaması yapılmıştır.

Anahtar Sözcükler: Yapay Zeka, Bilgisayar Oyunları, Sodoku Oyunu, Derinlik Öncelikli Arama

ABSTRACT

Master Thesis

A NEW APPROACH AND APPLICATION TO THE ALGORITHM OF SUDOKU PROBLEM WHICH IS BEING SUPPORTED ARTIFICIAL INTELLIGENCE

Tuğba KARAOĞLU

İstanbul Aydın University
Graduate School of Natural and Applied Sciences
Department of Computer Engineering

Supervisor: Asst. Prof. Dr. Ahmet BABANLI
December, 2012

Artificial Intelligence is one of the most popular branches of the computer science and it aims to make machines act intelligently, rendering them able to cope with more in number and more complex problems by themselves. Without a doubt, one of the most active sectors which aid artificial intelligence development today is the video game sector, which created programs that can beat world chess champions and checkers masters. This work is a quest motivated by these causes, which first identifies artificial intelligence, then explain the computation methods that aid artificial intelligence science and sudoku game has been programmed using a different algorithm structure.

Keywords: Artificial Intelligence, Computer Games, Sudoku Problem, Depth-First Search