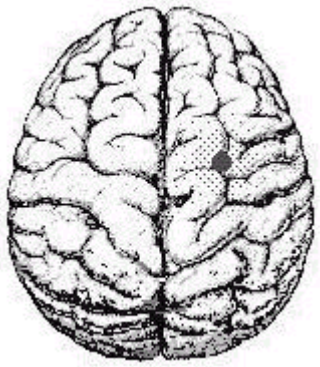
BÖLÜM 1
GİRİŞ

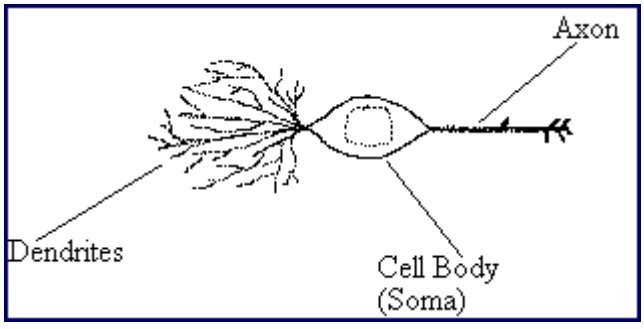
1.1 İNSAN BEYNİ

İnsanoğlu icat ettiği pek çok şeyi doğadaki benzerlerinden ilham alarak geliştirmiştir, bizim konumuz da yapay zeka olduğuna göre, öncelikle zeka yetisine sahip tek organ olan beyini inceleyerek işe başlamalıyız...



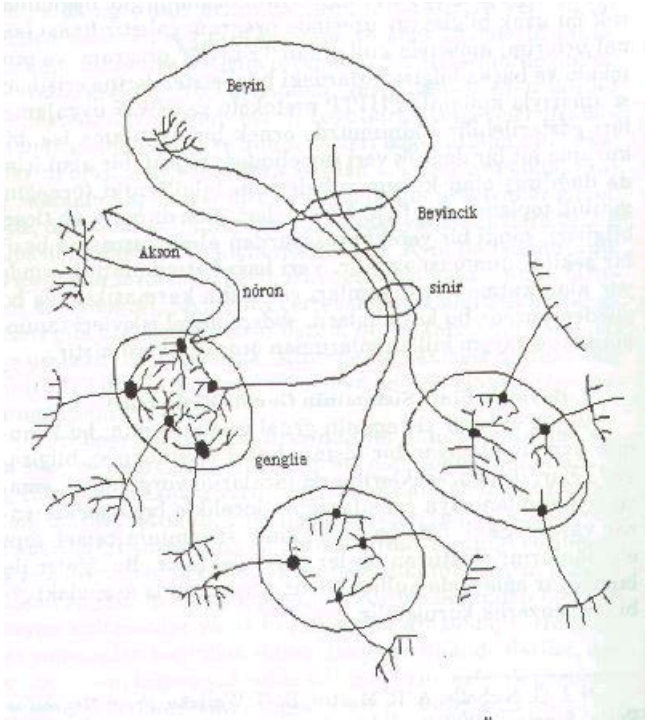
İnsan beyni, birbiri ile karmaşık ilişkiler içinde bulunan 3 paund'luk bir nöron hücreleri kitlesidir. Tüm aktivitelerimizi kontrol eder, yaradılışın en görkemli ve gizemli harikalarından biridir. İnsan zekasını, duyuların yorumunu, hareketlerin denetimini oluşturur. Bu inanılmaz organ bilim adamlarını olduğu kadar, bilim dışında olanları da şaşırtmaktadır. Beyin üzerine duyulan büyük ilgi ve konu üzerinde yapılan çalışmalar, yeni başlamış değildir. İnsanda ve diğer canlılarda yaşamsal faaliyetlerin yerine getirilmesinde merkez konumunda bulunan beyin üzerindeki çalışmalar yüzyıllardır yapılmakta ve bugün de tam olarak anlaşılamadığı için içinde bir çok disiplin içeren nörolojik bilimler alanında çalışmalar hızla devam etmektedir.

İnsan beynindeki bir nöron:



Beynimizin sadece 1 cm3'ünde, bir trilyon bağlantıya sahip, 100 milyar sinir hücresi (nöron) bulunmakta ve bu nöronlar arasında her bir saniyede 10 milyon x milyar kere uyarı gerçekleşmektedir. Bütün bunlar beraberce yaklaşık 1300 gram ağırlığında, sınırsız kompleks bir kimyasal fabrikada gerçekleşmektedir. Bu fabrika içerisinde hücreler arası bağlantılar ve etkileşimler ve bu etkileşimi sağlayan elektriksel etkiler ve kimyasal maddeler hafıza sistemimizin temelini teşkil etmektedir. Bu sinir hücreleri bir bilgisayarın işlemcisine göre kat kat yavaş çalışmaktadırlar, ancak insan beyninin gücü bu milyarlarca hücrenin aynı anda ve beraberce (paralel olarak) çalışabilmesinden kaynaklanmaktadır.

Beynimize gelen bir sinyalin sinirler tarafından tüm vücudumuza iletilmesinin, saniyenin 50 de 1'i gibi kısa bir sürede gerçekleştiğini biliyor muydunuz ?



İnsan beyni hiç bir bilgisayarla karşılaştırılmayacak kadar karmaşık ve üstün bir sisteme sahiptir. Beynin içine derinlemesine girildikçe, bizim kavrayabilme sınırlarımızı zorlayan detaylarla karşılaşırız, orda henüz kavramayı tam olarak beceremediğimiz bambaşka bir dünya vardır.

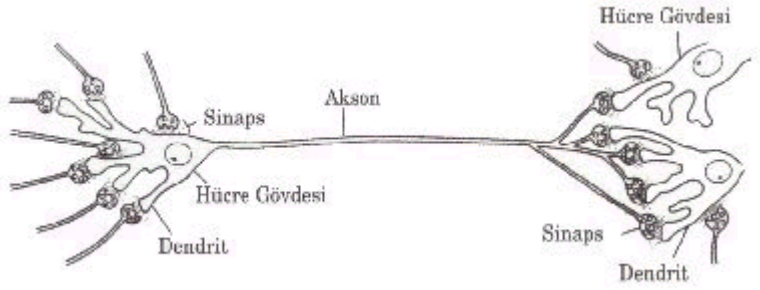
Bizim yerimize düşündüğünü zannettiğimiz beyin aslında karar verme yeteneğine sahip olmayan basit hücrelerden oluşur. Dişideki yumurta hücresinin, erkekten gelen sperm hücresiyle birleşmesi sonucu meydana gelen hücre, tekrar-tekrar bölünerek binlerce, milyonlarca hücre oluşturur. Vücutta bulunan tüm hücrelerin ortak özellikleri vardır. Çekirdek, mitekondri, sitoplazma vb... Fakat her hücre farklı bir dokuyu oluşturur. Beyin ve sinir sistemini oluşturan hücrelere nöron denir. Nöronların ise diğer hücrelerden belirgin olarak görülen farklılıkları akson ve dendrit adı verilen iki uzantılarının olmasıdır. Hücre çoğalmasının 18. gününde sinir sisteminin ilk farklılaşmaları oluşmaya başlar. Embriyonun sinir sistemi oluşmaya başlarken başkalaşan sinir hücrelerinin akson ve dendritleri hücre gövdesinden uzar. Her nöronun sahip olduğu akson ve dendritlerin uzunlukları birbirinden farklıdır ve hepsi sahip oldukları uzunluklara göre bir görev üstlenmişlerdir. Mesela, omurilikten ayağa mesaj iletecek akson 1 m. uzunluğundayken, gözümüzden beynimize uzanan diğer bir akson sadece 5 cm. uzunluğundadır.

Vücuttaki milyarlarca akson ve dendrit, görevlerini gerçekleştirmek için sadece kendilerine gerekli olacak uzunluğa kadar gelişir ve ardından büyümeleri durur. Vücuttaki tüm nöronların sahip olduğu bu uzantılar sayesinde tüm bilgiler gereken yerlere iletilir. Nöronların bu şekilde olması, vücudun her kösesine yayılarak sinir sistemimizi oluşturmalarını ve vücudumuzdaki haberleşmeyi çok hızlı bir şekilde gerçekleştirmelerini sağlar. Böylece beyin vücuttaki her noktadan eksiksiz bilgi alır. İletişimin en önemli elemanları ise elbette ki nöronlardaki akson ve dendritlerdir. Her ikisi arasında çok uyumlu bir iş bölümü vardır. Dendritler gelen mesajı hücre gövdesine iletmekle, aksonlar ise hücre gövdesinde değerlendirilen bu mesajı başka bir nörona iletmekle görevlidirler.

Bir nöronun birden çok dendrite sahip olması onun vücudunun değişik yerlerindeki nöronlarla birebir iletişim halinde olmasını sağlar. İnsan bedenindeki 100 milyar nöron göz

önüne alındığında ve bunların her birinin birden fazla dendrite sahip olduğu düşünüldüğünde, sinir sisteminin, ne kadar karmaşık olduğu daha iyi anlaşılacaktır. Tipik bir nöron 1.000 ile 10.000 farklı bağlantıya sahip olabilir ve 1.000 farklı nörondan bilgi alabilir. Bu da, 'bir nöronun dendritleri, vücuttaki diğer 10.000 farklı nöronla dolaylı bağlantıya geçerek, onlardan da kendisine bilgi akışı olmasını sağlayabilir' demektir –ki bu böylece uzayıp gider-. Bu rakamlar üzerinde düşündüğümüzde de, sinir sistemindeki sonsuz bağlantıların oluşturduğu karmaşık ağ daha iyi hayal edilebilir.

Beyin ve sinir sisteminde fiziksel katmana bakıldığında, işlemci, sinyal iletim ortamı ve yol verici olarak, sinir sisteminin temel öğesi olan nöron, ya da sinir hücresi görülmektedir. Sinir hücresini oluşturan dendrit, hücre gövdesi, akson ve akson uçları (sinaps) şekilde gösterilmiştir.



Dendritler sinaptik sinyalleri girdi olarak almakta, hücre gövdesi bu sinyalleri -bilindiği kadarıyla- analog bir yöntemle işlemekte ve üretilen denetim sinyali ya da sinyalleri aksonlar aracılığı ile denetlenecek hedef hücrelere iletilmektedir.

Tipik bir nöron, hücre gövdesi ve dendritleri üzerine dış kaynaklardan gelen elektrik darbelerinden üç şekilde etkilenir. Gelen darbelerden bazısı nöronu uyarır, bazısı bastırır, geri kalanı da davranışında değişikliğe yol açar. Nöron yeterince uyarıldığında çıkış kablosundan (aksonundan) aşağı bir elektriksel işaret göndererek tepkisini gösterir. Genellikle bu tek akson üzerinde çok sayıda dallar olur. Aksondan inmekte olan elektrik işareti dallara ve alt dallara ve sonunda başka nöronlara ulaşarak onların davranışını etkiler. Nöron, çok sayıda başka nöronlardan genellikle elektrik darbesi biçiminde gelen verileri alır. Yaptığı iş bu girdilerin karmaşık ve dinamik bir toplamını yapmak ve bu bilgiyi aksonundan aşağı göndererek bir dizi elektrik darbesi biçiminde çok sayıda başka nörona iletmektir. Bu çalışma mantığı örnek alınarak yapay sinir ağları geliştirilmiştir. Nöron, bu etkinlikleri sürdürmek ve molekül sentezlemek için de enerji kullanır fakat başlıca işlevi işaret alıp işaret göndermektir, yani bilgi alışverişidir.

Ortalama bir beyinde milyarca sinir hücresi vardır. Dolayısıyla sayıları arttıkça beyin işlevlerinin de artacağı açıktır. Nöron sayısı kadar önemli olan bir diğer özellik; nöronların uzantıları aracılığı ile diğer nöronlarla oluşturdukları ilişkilerdir. Bilgi alışverişinin yapıldığı bu ilişki noktaları (sinapslar) nöron başına 1000 ile 10.000 arasında değişir. Sinapslar, etkiye akım

var / akım yok şeklinde tepki gösterir. Demek ki, bir nöron 10^3 hatta 10^4 tepki verebilir. 10^{10} nöron olduğuna göre, sinir sisteminde tepki sayısı ya da bilgisayar deyimiyle söylersek bit sayısı, 10 trilyon ile 100 trilyon arasında değişecektir. Bu bit sayısı 500 sayfalık, bir milyon kitabı dolduracak büyüklüktedir. (Yaklaşık 116.416 Gb.)

1.2 ÖĞRENME

Beynin en önemli işlevlerinden birisi de insanın çevresinde olanları öğrenmesi ve edindiği bilgileri daha sonra kullanmak üzere depolamasıdır. Çevreden gelen uyarıların değerlendirilmesi ve uygun davranışların geliştirilmesi öğrenme yoluyla olmaktadır. Öğrenilen bilginin saklanmasını ise bellek sağlar. Öğrenme çok geniş bir kavram olup görme, işitme, dokunma, tat ve doku duyguları ile algılanan uyarıların beyinde ilişkilendirilme, tekrarlama gibi birden çok beyin işlemi sonucu gerçekleşir. Öğrenmenin doğrudan bir ölçümü yapılamayıp ancak ortaya çıkan davranış değişiklikleri ile değerlendirilebilmektedir.

Öğrenme biçimleri uyarı yanıt ilişkisine göre ilişkilendirilmiş (asosiye) ve ilişkilendirilmemiş (asosiye) olmayan üzere iki ana gruba ayrılmaktadır. Çevreden gelen tekrarlayan uyarıya karşı oluşan belirli bir yanıtın, zaman içinde meydana gelen değişme, ilişkilendirilmemiş öğrenme biçimini oluşturur. Bu öğrenme biçiminde tekbir yanıt ve ona karşı oluşmuş başka bir uyarı ile ilişkilendirilmemiş belirli bir yanıt söz konusudur. Bir alt biçimi olan alışma uyaranın etkinliğinin zaman içinde sönmesi ve ilk ortaya çıkan yanıtın şiddetinin azalmasıdır. Bulunduğumuz odada saatin tik taklarını bir süre sonra duymamamız bu öğrenme biçimi için bir örnektir. Bunun tam tersi olan duyarlılaşmada ise yanıtın şiddeti tekrarlayan uyarı ile artar. Ocak üzerinde çok sıcak olan bir kabı ilk elledğimizde elimizi hızla geri çekeriz. Daha sonra kap ılıklaşsa bile biz kaba değdiğimizde kabın sıcaklığı ile uyumlu olmayacak şekilde elimizi hızla çekeriz. Bu iki tip öğrenme biçimi, en basit organizmalardan en karmaşık organizmalara kadar tüm canlılarda kullanılır.

İlişkilendirilmiş öğrenme biçimlerinden birisi, klasik şartlanmadır ve Pavlov'un köpeklerle yaptığı sindirim sistemi çalışmaları en bilinen örneği oluşturur. -Daha önce tükürük salgılanmasına neden olmayan bir uyarı (zil sesi), belli bir süre ve aşamadan sonra salgılamaya neden olur. Zil sesini duyduktan sonra yemek verilen köpek, bir süre sonra bunun tekrarlanması sonucunda yemek verilmeden zil sesini duyduğunda tükürük salgısında artış olur. Zil sesi şartlı uyaran, yemek şartsız uyaran, zil karşısında oluşan tükürük salgısı şartlı reflekstir.- Şartlı refleksin oluşması için şartlı ve şartsız uyaranların belli sayıda tekrar etmesi gerekir. Pavlov'a göre hayvanlar ve insanlarda öğrenme düşüncelerin ilişkilendirilmesi değil, uyaranların ilişkilendirilmesidir. Rescola ve Wagner bu model üzerindeki çalışmalarında klasik şartlanmanın tek başına şartlı ve şartsız uyaranın birlikteliği ve tekrarlanması sonucu oluşmayacağını ileri sürmüşlerdir. Rastgele bir araya gelen uyarılar bir anlamlılık oluşturmuyorsa ne kadar sık

tekrarlasa da öğrenme biçimine dönüşmez. Canlılar tüm olasılık ve bağlantıları değerlendirip birbiriyle ilişkisi olan şartlı ve şartsız uyaranları bir araya getirerek öğrenmeyi gerçekleştirir. Bir başka deyişle beyin, çevredeki birbiriyle bağlantılı ya da ilişkili olayları seçer ve saptar.

Diğer bir önemli ilişkilendirilmiş (asosiyatif) öğrenme örneği ise operan şartlı öğrenmedir. Bu öğrenme biçimine deneme yanılma yöntemi de denmektedir. Klasik şartlanma iki uyarı arasındaki bağlantıyı içerirken, operan şartlanma bir uyarı ile canlının bu uyarıya karşı oluşturduğu davranışı içerir. Skinner'in incelediği operan şartlanma modelinde bir kafes içine konan sıçan, bir ışık karşısında bir düğmeye basarak yiyeceğe ulaşacağını öğrenir. Başlangıçta yiyeceğe nasıl ulaşacağını bilemeyen sıçan, birbirinden farklı davranışlar sergiler ve önünde duran düğmeye rastgele basarken yemeğe ulaşır. Bu davranışını birkaç kez tekrarlayıp aynı sonuca ulaşan sıçan, ışık yandığında düğmeye basar ve yiyeceğini alır.

Farklı gibi görünen klasik ve operan şartlanmada temel kurallar aynıdır. Ödüllendirme ve kaçınma mekanizmaları gelişen davranışı belirlemektedir ve her iki şartlanma biçiminde de aynı sinir sistemi mekanizmaları yer alır. Tüm canlılar çevrede olanları ve rastlantıları ilişkilendirilmiş öğrenme ile fark eder ve öğrenir. Ancak gerçekte şartlı ve şartsız uyaranlar, öğrenme modellerinde olduğu gibi tek başlarına ve düzenli aralıklarla tekrar etmezler. Canlılar karşı karşıya kaldıkları pek çok uyaran arasında aralarında yaşamını devam ettirmede önemli olan biyolojik olarak anlamlı bir ilişkinin olduğu uyaranlar arasında bağlantı kurar. Bu ilişkilendirilmiş öğrenme biçimleriyle canlılar birbiriyle ilişkili ve ilişkisiz olayları birbirinden ayırt ediyor ve çevrede olanların nedensel bağlantılarını saptıyor. Hangi uyarıların önemli olduğu, dikkate alınması gerektiği için ya daha önceden sinir sisteminde programlanmış doğru bilgi ya da sonradan öğrenme gerekmektedir. Genetik ve gelişimsel programlama, değişik aşamalarda en basit canlılardan en karmaşık canlı olan insana kadar tüm canlılarda bulunmaktadır. İnsanın yaşamını devam ettirmesi, çevreye uyum sağlaması ve bulunduğu noktadan daha ileriye gitmesi öğrenme, esnek karar verebilme ve farklı uyaranlar arasında yeni bağlantıları farkedebilmesi ile gerçekleşebiliyor.

Edinilen bilginin saklanması ve geri çağrılmasına göre öğrenme ve bellek, iki ana guruba ayrılır. Çevremizde olanlar, evren, insanlar ve yerler ile olan bilgileri, sözcüklerle ifade edilen, tanımlayıcı bellek ya da deklaratif bellek biçiminde saklarız. Algı ve motor yeteneği gerektiren bazı işleri nasıl yapılacağı konusunda sözcüklerle ifade edemediğimiz, tanımlama biçimine getirilmemiş olan tepkisel (refleksif) bellek biçimini kullanırız. Tanımlayıcı belleğin oluşması bilinçli bir düşünme sürecini gerektirir. Bu süreç içinde değerlendirme, karşılaştırma ve bir araya getirme gibi bilişsel işlemleri kullanır. Tanımlayıcı bellekten bilgilerin çağrılma işlemi yaratıcı bir süreç olup, yeniden sıralama, yeniden yapılandırma ve orijinal olanı yoğunlaştırma

işlemlerini içerir. Bilginin tanımlayıcı olarak depolanması, bizim kişisel algı yapımıza göre ve daha önce edinilmiş bilgilere göre kişiden kişiye farklılık göstererek oluşmaktadır.

Tepkisel (refleksif) bellek ise bir işlemin fark edilmeden çok sayıda tekrarı sonucu zaman içinde birikerek oluşur. Tepkisel bellek bilinçli düşünme ya da karşılaştırma, değerlendirme gibi bilişsel (kognitif / cognitive) işlemler gerekmeden oluşur ve genellikle kelimelerle ifade edilmez. Bazı algı ve motor yeteneklerin kazanılması, gramer gibi bazı kuralların öğrenilmesi tepkisel bellek ile olmaktadır.

Pek çok durumda her iki bellek ve öğrenme biçimi de yer alır. Örneğin araba kullanmak başlangıçta tanımlayıcı (deklaratif) bellek ile gerçekleşirken bir zaman sonra tepkisel (refleksif) belleğe geçer ve artık araba kullanma kuralları her kullanışta sözcüklerle ifade edilmez, kısaca otomatikleşir.

1.3 YAPAY ZEKAYA GİRİŞ

Yapay zeka, insanın düşünme yapısını anlamak ve bunun benzerini ortaya çıkaracak bilgisayar işlemlerini geliştirmeye çalışmak olarak tanımlanır. Yani programlanmış bir bilgisayarın düşünme girişimidir. Daha geniş bir tanıma göre ise, yapay zeka, bilgi edinme, algılama, görme, düşünme ve karar verme gibi insan zekasına özgü kapasitelerle donatılmış bilgisayarlardır.

Bu konudaki ilk çalışma McCulloch ve Pitts tarafından yapılmıştır. Bu araştırmacıların önerdiği, yapay sinir hücrelerini kullanan hesaplama modeli, önermeler mantığı, fizyoloji ve Turing'in hesaplama kuramına dayanıyordu. Her hangi bir hesaplanabilir fonksiyonun sinir hücrelerinden oluşan ağlarla hesaplanabileceğini ve mantıksal "ve" ve "veya" işlemlerinin gerçekleştirilebileceğini gösterdiler. Bu ağ yapılarının uygun şekilde tanımlanmaları halinde öğrenme becerisi kazanabileceğini de ileri sürdüler. Hebb, sinir hücreleri arasındaki bağlantıların şiddetlerini değiştirmek için basit bir kural önerince, öğrenebilen yapay sinir ağlarını gerçekleştirmek de olası hale gelmiştir.

1950'lerde Shannon ve Turing bilgisayarlar için satranç programları yazıyorlardı. İlk yapay sinir ağı temelli bilgisayar SNARC, MIT'de Minsky ve Edmonds tarafından 1951'de yapıldı. Çalışmalarını Princeton Üniversitesi'nde sürdüren Mc Carthy, Minsky, Shannon ve Rochester'le birlikte 1956 yılında Dartmouth'da iki aylık bir çalışma toplantısı düzenledi. Bu toplantıda bir çok çalışmanın temelleri atılmakla birlikte, toplantının en önemli özelliği Mc Carthy tarafından önerilen Yapay zeka adının konmasıdır. İlk kuram ispatlayan programlardan Logic Theorist (Mantık kuramcısı) burada Newell ve Simon tarafından tanıtılmıştır.

Daha sonra Newell ve Simon, “insan gibi düşünme” yaklaşımına göre üretilmiş ilk program olan General Problem Solver (Genel sorun çözücü) ‘ı geliştirmişlerdir. Simon, daha sonra fiziksel simge varsayımını ortaya atmış ve bu kuram, insandan bağımsız zeki sistemler yapma çalışmalarıyla uğraşanların hareket noktasını oluşturmuştur.

Bundan sonraki yıllarda mantık temelli çalışmalar egemen olmuş ve programların başarımlarını göstermek için bir takım yapay sorunlar ve dünyalar kullanılmıştır. Daha sonraları bu sorunlar gerçek yaşamı hiçbir şekilde temsil etmeyen oyuncak dünyalar olmakla suçlanmış ve yapay zekanın yalnızca bu alanlarda başarılı olabileceği ve gerçek yaşamdaki sorunların çözümüne ölçeklenemeyeceği ileri sürülmüştür.

Geliştirilen programların gerçek sorunlarla karşılaşıldığında çok kötü bir başarım göstermesinin ardındaki temel neden, bu programların yalnızca sentaktik bir şekilde çalışıp konu ile ilgili bilgileri kullanmamasıydı. Bu dönemin en ünlü programlarından Weizenbaum tarafından geliştirilen Eliza, karşısındaki ile sohbet edebiliyor gibi görünmesine karşın, yalnızca karşısındaki insanın cümleleri üzerinde bazı işlemler yapıyordu. İlk makine çevirisi çalışmaları sırasında benzeri yaklaşımlar kullanılıp çok gülünç çevirilerle karşılaşılınca bu çalışmaların desteklenmesi durdurulmuştur.

Zeki davranışı üretmek için bu çalışmalarda kullanılan temel yapılardaki bazı önemli yetersizliklerin de ortaya konmasıyla bir çok araştırmacılar çalışmalarını durdurdular. Buna en temel örnek, sinir ağları konusundaki çalışmaların Minsky ve Papert’in 1969’da yayınlanan Perceptrons adlı kitaplarında tek katmanlı algaçların bazı basit problemleri çözemeyeceğini gösterip aynı kısırlığın çok katmanlı algaçlarda da beklenilmesi gerektiğini söylemeleri ile bıçakla kesilmiş gibi durmasıdır.

Her sorunu çözecek genel amaçlı program yerine belirli bir uzmanlık alanındaki bilgiyle donatılmış programlar kullanma fikri yapay zeka alanında yeniden bir canlanmaya yol açtı. Kısa sürede uzman sistemler adı verilen bir metodoloji gelişti. Fakat burada çok sık rastlanan tipik bir durum, bir otomobilin tamiri için önerilerde bulunan uzman sistem programının otomobilin ne işe yaradığından haberi olmamasıydı.

İnsanların iletişimde kullandıkları Türkçe, İngilizce gibi doğal dilleri anlayan bilgisayarlar konusundaki çalışmalar bu sıralarda hızlanmaya başladı. Doğal dil anlayan programların dünya hakkında genel bilgiye sahip olması ve bu bilgiyi kullanabilmek için genel bir metodolojisi olması gerektiği belirtilmekteydi.

Uzman dizgelerin başarıları beraberinde ilk ticari uygulamaları da getirdi. Yapay zeka yavaş yavaş bir endüstri haline geliyordu. DEC tarafından kullanılan ve müşteri siparişlerine göre donanım seçimi yapan R1 adlı uzman sistem şirkete bir yılda 40 milyon dolarlık tasarruf

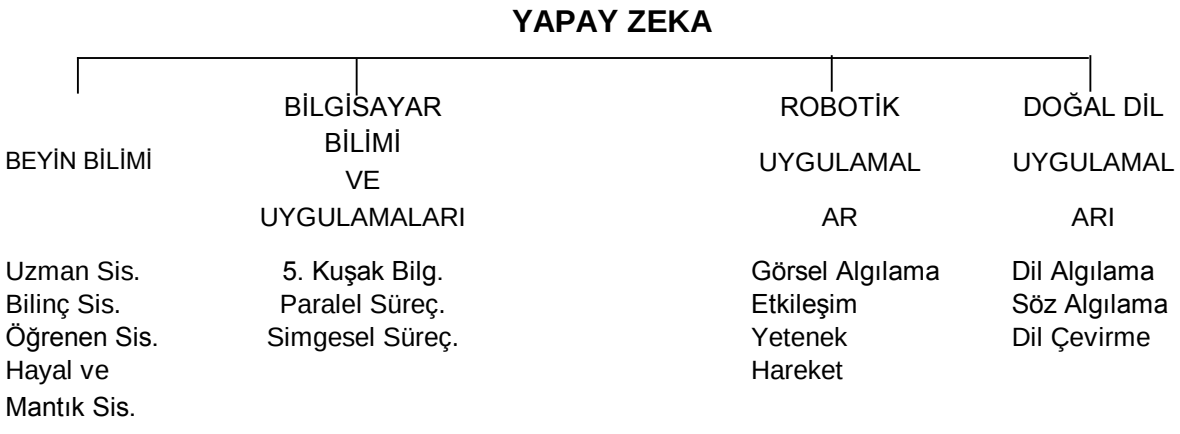
sağlamıştı. Birden diğer ülkelerde yapay zekayı yeniden keşfettiler ve araştırmalara büyük kaynaklar ayrılmaya başlandı. 1988'de yapay zeka endüstrisinin cirosu 2 milyar dolara ulaşmıştı. Bütün bu çalışmaların sonunda yapay zeka araştırmacıları iki guruba ayrıldılar. Bir gurup insan gibi düşünen sistemler yapmak için çalışırken, diğer gurup ise rasyonel karar verebilen sistemler üretmeyi amaçlamaktaydı

1.4 YAPAY ZEKA

Günlük yaşantıda zihinsel kapasite çok önemli olduğu için insanlar bilimsel olarak akıllı insan (Homo sapien) olarak isimlendirilmiştir. Yapay zeka, *YZ,* (AI-Artificial Intelligence) alanı zeki varlıkları anlamaya çalışır. Bu nedenle YZ çalışmalarının bir amacı kendimiz hakkında daha fazla şey öğrenmektir. Felsefe ve psikolojiden farklı olarak YZ yalnız zekayı anlamaya değil aynı zamanda zeki varlıklar yapmaya çabalar. YZ çalışmalarının bir diğer nedeni de başlangıç aşamasında bile ilginç ve yararlı ürünler yapılmış olmasıdır. Şeklen 1956'da başlayan YZ en yeni disiplinlerden biridir. Fizik gibi disiplinler de çalışanlar bütün iyi fikirlerin Galileo, Newton, Einstein ve diğer bilim adamları tarafından üretildiğini ve yeni bir fikrin ortaya çıkması için yıllar boyu süren çalışmaların gerektiğini düşünebilir. Ama YZ çok yeni bir disiplin olduğu için üretilecek çok fikir vardır.

YZ; algılama, mantıksal muhakeme gibi genel amaçlı alanlar ve satranç oynama, şiir yazma, matematik teoremlerinin ispatlanması ve hastalıkların teşhisi gibi özel amaçlı alt alanları içerir. Bilgisayar,metamatik,mühendislik,biyoloji,dilbilim ve psikoloji bilimlerinin ortak etkileşiminden oluşan yani bir bilim dalının yaratmayı tasarladığı yapay beyine, yapay zeka adı verilmektedir.

Yapay zekayı oluşturan bileşenleri ortak bir şemada toplayabiliriz:



YZ nedir ?

Değişik şekillerde yapılan YZ tanımları aşağıdaki 4 gruptan birine düşer

- İnsan gibi düşünen sistemler.
- İnsan gibi davranan sistemler.
- Rasyonel düşünen sistemler.
- Rasyonel davranan sistemler.

Yukarıda görüldüğü tanımlar insana veya rasyonelliğe göre yapılmıştır. Bu da insanların her zaman rasyonel davranmadıklarını belirtmektedir. İnsan merkezli yaklaşım hipotez ve deneysel doğrulamayı içerirken, rasyonalist yaklaşım matematik ve mühendisliği içermektedir. Aşağıda her tanım ayrıntılı olarak incelenmiştir.

İnsan gibi Davranmak: Turing Test yaklaşımı

Turing Test, zekanın işlemsel tanımını sağlamak için Alan Turing (1950) tarafından tasarlanmıştır. Turing zeki davranışı, soru soran kimseyi tüm anlamaya ait işlemlerde şaşırtacak şekilde insan seviyesinde davranma olarak tanımlamıştır. Sorular bir hat üzerinden uzaktan sorulmakta ve hattın ucunda insan mı yoksa bilgisayar mı olduğu anlaşılmaya çalışılmaktadır. Bu testi geçmek için bilgisayarın insan seviyesinde performans göstermesi gerekir. Şimdilik bu testi geçmek çok zor görünmektedir. Bu test için bilgisayarın aşağıdaki yeteneklere sahip olması gerekmektedir:

- **Doğal dil işleme:** Türkçe veya İngilizce (insan dilleri) haberleşebilmeyi sağlamak için.
- **Bilgi Gösterimi :** Soruşturma sırasında veya öncesinde bilgi depolamak için.
- **Otomatik Muhakeme :** Depolanan bilgiyi kullanarak cevap verebilme veya yeni sonuçlar çıkarma.
- **Öğrenme :** Yeni koşullara adapte olma ve kalıpları saptama.

Turing test'te soruşturmayı yapan ve bilgisayar birbirini kesinlikle görmez. Buna karşın *Toplam Turing Test*'te vidyo sinyali de kullanılır. Böylece soruşturmayı yapan kişi fiziksel nesneleri algılama yeteneğini de ölçebilir. Bu test için bilgisayarın aşağıdaki yeteneklere sahip olması gerekir:

- **Görme :** Nesneleri algılamak için.
- **Robotik :** Nesneleri hareket ettirmek için.

YZ'de Turing Testi geçmek için yapılan büyük bir çalışma yoktur. İnsan gibi davranma meselesi insanlarla etkileşen YZ programları için geçerlidir. Örneğin kullanıcı ile iletişimde bulunan bir dil işleme sistemi insan gibi konuşmalıdır.

İnsan Gibi Düşünmek

Verilen bir programın insan gibi düşündüğünü söyleyebilmek için insanların nasıl düşündüğünü saptamanın bir yolunu bulmamız gerekir. İnsanın nasıl düşündüğünü iki şekilde

öğrenebiliriz. Birincisi kendi düşüncelerimizi gözleme, ikincisi ise psikolojik deneyler yapmaktır. Düşünme ile ilgili yeterince kesin teorilere erişilirse bu teoriler bilgisayar programı olarak ifade edilebilir.

Bazı araştırmacılar programları herhangi bir yöntemi kullanarak doğru sonucu bulacak şekilde yazarken bazı araştırmacılar da sonucun doğru ya da yanlış olmasına bakmaksızın insanların düşündüğü şekilde çözümü arayan programlar yazmaktadır.

Rasyonel Düşünme: Düşünme Kanunları

Doğru düşünmeyi kodlamaya ilk çalışanlardan biri Aristotle'dır. Aristottle mantığına göre doğru önermeler verildiğinde daima doğru sonuç elde edilir. Aşağıda buna ait bir örnek verilmiştir:

Socrates insandır. İnsanlar ölümlüdür. → Öyleyse Socrates ölümlüdür.

Bu düşünme kanunlarının düşünme işlemini yönettiği kabul edildi ve Mantık alanını başlattı. YZ'de mantık taraftarları problemleri mantıksal notasyonla tanımlayarak akıllı sistemler oluşturmaya çalışmışlar. Bu konuda karşılaşılan iki büyük engel:

- Konuşma diline özgü bilgilerin %100 kesinlik ifade etmediği durumda mantık formunda belirtme güçlüğü
- Problemleri prensipte çözmekle, pratikte çözmek arasındaki fark. Birkaç düzine önerme bile bilgisayarın sınırını aşabilir. Bu da prensip olarak çözülebilecek problemlerin pratikte çözümünü güçleştirmektedir.

Rasyonel Davranma: Rasyonel Ajan Yaklaşımı

Rasyonel davranma, inanışları baz alarak amaca erişecek şekilde davranmaktır. Farklı YZ çalışmalarına benzer şekilde yaklaşabilmek için ajan (agent) sözcüğü kullanılmaktadır. Buradaki kullanım normal anlamının dışındadır. Ajan algılayan ve davranan herhangi bir şeydir. Ajan yaklaşımında YZ çalışmaları; rasyonel ajan çalışmaları ve rasyonel ajan oluşturmadır.

Düşünme kanunlarında vurgulanan doğru çıkartımdır. Doğru çıkartımda bulunmak bazen rasyonel ajanın (RA) bir parçası olabilir. Çünkü rasyonel davranmanın bir yolu, mantıksal olarak bir davranışın istenen amacı sağlayacağı sonucu çıkartılırsa bu sonuç üzerine belirlenen şekilde davranılır. Diğer yandan doğru çıkartım tam rasyonellik değildir. Bazen yapılacak şeyin doğru olacağı ispat edilmeden yapılması gerekebilir. Örneğin kızgın bir sobadan elin çekilmesi reflekstir ve kızgınlığı hissettikten sonra hangi hareketin yapılacağını düşünerek yapılan yavaş haraketten daha etkilidir. Sobadan elin çekilmesi bir doğru çıkartım sonucu olmamakta ama

rasyonel bir davranıştır. Mükemmel rasyonellik daima doğru şeyi yapmaktır ve karmaşık bir çevrede daima doğru şeyi yapmak mümkün değildir.

RA yaklaşımının iki üstünlüğü vardır:

- Düşünme kanunları yaklaşımından daha geneldir. Çünkü rasyonel davranmak için doğru çıkartım faydalı bir mekanizma olmasına karşın gerekli değildir.
- Rasyonelliğin tanımı açıkça yapıldığı için bilimsel gelişmeye insan davranışından veya insan düşüncesinden daha yatkındır.

1.5 YAPAY ZEKAYA YAKLAŞIMLAR

Yapay zeka gerçekten de bilgisayar bilimleri içinde en belirsiz olanlardan birisidir. Bu nedenle ne olduğu ne olmadığı (hatta olup olmadığı bile) hala tartışılmaktadır. Dolayısıyla benzer durumlarda da olduğu gibi bazı yaklaşımlar mevcuttur.

Matematiksel Yaklaşım

Kaos teorisinin beynin üst düzey fonksiyonlarının modellenmesinde önemli bir rol oynayacağı düşünülmektedir. İnsan beyni gibi bir fonksiyon üstlenmesine çalışılan bir sistemin tasarlanmasındaki çabalar için, kuşkusuz kaos teorisi çok önemli bir yer tutmaktadır. Çünkü tasarılar ortaya konulacak modelleri temel almaktadır.

Kaos teorisi, sayısal bilgisayarların ve onların çıktılarını çok kolay görülebilir hale getiren ekranların ortaya çıkmasıyla gelişti ve son on yıl içinde popülerlik kazandı. Ancak kaotik davranış gösteren sistemlerde kestirim yapmanın imkansızlığı bu popüler görüntüyle birleşince, bilim adamları konuya oldukça kuşkucu bir gözle bakmaya başladılar. Fakat son yıllarda kaos teorisinin ve onun bir uzantısı olan fraktal geometrinin, borsadan meteorolojiye, iletişimden tıbba, kimyadan mekaniğe kadar uzanan çok farklı dallarda önemli kullanım alanları bulması ile bu kuşkular giderek yok olmaktadır.

Teoriye temel oluşturan matematiksel ve temel bilimsel bulgular, 18.yüzyıla, hatta bazı gözlemler antik çağlara kadar geri gidiyor. Yunan ve Çin mitolojilerinde yaradılış efsanelerinde başlangıçta bir kaosun olması rastlantı değil. Özellikle Çin mitolojisindeki kaosun, bugün bilimsel dilde tanımladığımız olgularla hayret verici bir benzerliği olduğunu görüyoruz. Batı'da da daha sonraki dönemlerde bilim adamları tarafından karmaşık olgulara dair gözlemler yapılmıştır. Poincare, Weierstraas, von Koch, Cantor, Peano, Hausdorff, Besikoviç gibi çok üst düzey matematikçiler tarafından bu teorinin temel kavramları oluşturulmuştur.

Karmaşık sistem teorisinin ardında yatan yaklaşımı felsefe, özellikle de bilim felsefesi açısından inceleyecek olursak, ortaya ilginç bir olgu çıkıyor. Aslında bugün pozitif bilim olarak

nitelendirdiğimiz şey, batı uygarlığının ve düşünüş biçiminin bir ürünüdür. Bu yaklaşımın en belirgin özelliği, analitik oluşu yani parçadan tüme yönelmesi (tümevarım).

Genelde karmaşık problemleri çözmede kullanılan ve bazen çok iyi sonuçlar veren bu yöntem gereğince, önce problem parçalanıyor ve ortaya çıkan daha basit alt problemler inceleniyor. Sonra, bu alt problemlerin çözümleri birleştirilerek, tüm problemin çözümü oluşturuluyor. Ancak bu yaklaşım görmezden gelerek ihmal ettiği parçalar arasındaki ilişkilerdir. Böyle bir sistem parçalandığında, bu ilişkiler yok oluyor ve parçaların tek tek çözümlerinin toplamı, asıl sistemin davranışını vermekten çok uzak olabiliyor.

Tümevarım yaklaşımının tam tersi ise tümevarım, yani bütüne bakarak daha alt olgular hakkında çıkarsamalar yapmak. Genel anlamda tümevarımı Batı düşüncesinin, tümdengelimi Doğu düşüncesinin ürünü olarak nitelendirmek mümkündür. Kaos yada karmaşıklık teorisi ise, bu anlamda bir doğu-Batı sentezi olarak görülebilir. Çok yakın zamana kadar pozitif bilimlerin ilgilendiği alanlar doğrusallığın geçerli olduğu, daha doğrusu çok büyük hatalara yol açmadan varsayılabildiği alanlardır. Doğrusal bir sistemin girdisini x, çıktısını da y kabul edersek, x ile y arasında doğrusal sistemlere özgü şu ilişkiler olacaktır:

Eğer x_1'e karşılık y_1, x_2'ye karşılık y_2 elde ediyorsak, girdi olarak x_1+x_2 verdiğimizde, çıktı olarak y_1+y_2 elde ederiz.

Bu özellikleri sağlayan sistemlere verilen karmaşık bir girdiyi parçalara ayırıp her birine karşılık gelen çıktıyı bulabilir, sonra bu çıktıların hepsini toplayarak karmaşık girdinin yanıtını elde edebiliriz. Ayrıca, doğrusal bir sistemin girdisini ölçerken yapacağımız ufak bir hata, çıktının hesabında da başlangıçtaki ölçüm hatasına orantılı bir hata verecektir. Halbuki doğrusal olmayan bir sistemde y'yi kestirmeye çalıştığımızda ortaya çıkacak hata, x'in ölçümündeki ufak hata ile orantılı olmayacak, çok daha ciddi sapma ve yanılmalara yol açacaktır. İşte bu özelliklerinden dolayı doğrusal olmayan sistemler kaotik davranma potansiyelini içlerinde taşırlar.

Kaos görüşünün getirdiği en önemli değişikliklerden biri ise, kestirilemez determinizmdir. Sistemin yapısını ne kadar iyi modellersek modelleyelim, bir hata bile (Heisenberg belirsizlik kuralına göre çok ufak da olsa, mutlaka bir hata olacaktır), yapacağımız kestirmede tamamen yanlış sonuçlara yol açacaktır. Buna başlangıç koşullarına duyarlılık adı verilir ve bu özellikten dolayı sistem tamamen nedensel olarak çalıştığı halde uzun vadeli doğru bir kestirim mümkün olmaz. Bugünkü değerleri ne kadar iyi ölçersek ölçelim, 30 gün sonra saat 12'de hava sıcaklığının ne olacağını kestiremeyiz.

Kaos konusunda bu uzun girişten sonra konunun beyinle ilişkisine gelelim. Beynin fizik yapısı ve görünüşü fraktaldır. Bu yapı, beynin gerek evrimsel, gerekse canlının yaşamı

sürecindeki gelişimin ürünüdür ki, bu gelişimin deterministik (genlerle belirli), ancak çevre ve başlangıç koşullarına son derece duyarlı, yani kaotik olduğu açıktır. Beynin yalnızca oluşumu değil, çalışma biçimi de kaotiktir. Beyni oluşturan inanılmaz boyuttaki nöron ağının içinde bilgi akışı kaotik bir şekilde gerçekleşir. Kaotik davranışın tarama özelliği ve bunun getirdiği uyarlanırlık (adaptivite) sayesinde, beyin çok farklı durumlara uyum sağlar, çok farklı problemlere çözüm getirebilir, çok farklı fonksiyonları gerçekleştirir.

EEG sinyalleri üzerine yapılan araştırmalar göstermiştir ki, sağlıklı bir insanın sinyalleri kaotik bir davranış gösterirken, epilepsi krizine girmiş bir hastanın sinyalleri çok daha düzenli, periyodik bir davranış sergilemektedir. Yani epilepsi krizindeki hastanın beyni, kendini tekrarlayan bir davranışa takılmış ve kaotik (yani sağlıklı) durumda sahip olduğu adaptivite özelliğini yitirmiştir. Bunun sonucu hasta, kriz sırasında en basit fonksiyonlarını bile yerine getiremez olur.

Kaos bilimini ortaya çıkaran, karmaşık olguları basit parçalara ayırmak yerine onları bir bütün olarak görme eğilimi, beyni inceleyen bilim adamlarının da yaklaşımını belirlemiştir. Eskiden beyin farklı fonksiyonlardan sorumlu merkezler şeklinde modellenirken, artık holistik (bütünsel) beyin modeli geçerlilik kazanmıştır. Bu modele göre herhangi bir işlev gerçekleştirilirken, beynin tümü bu olguya katılmaktadır.

Önümüzdeki yıllarda beynin yalnız alt düzey fizyolojik işleyişinin değil, öğrenme, hatırlama, fikir yürütme gibi üst düzey işlevlerinin de modellenmesinde kaosun çok önemli bir rol oynayacağı görülmektedir.

Fiziksel Yaklaşım

Tüm vücut fonksiyonları en temelde fiziğe dayanır. Fakat burada fiziğin oynadığı rol nedir? Bu, "taşı bıraktım yere düştü" tarzında bir fizik değildir. Böyle olsaydı beyin bugüne kadar çok kolay çözülürdü, hatta Descartes bile belki çözmüş olurdu. Söz konusu olan, son yetmiş yıl içinde fizikçilerin kullanmakta olduğu ve doğayı matematiksel bir yapı çerçevesinde anlayıp anlatabilme yöntemi olan kuantum mekaniğinin özellikleri ile durumu bağdaştırabilmektir. Bir masa üzerinde duran nesneyi yerçekimi çeker ama masa buna karşı gelir. Dolayısıyla nesne üzerine uygulanan toplam kuvvet sıfırdır. Üzerindeki koşullar böyle devam ettiği sürece, istediği gibi hareket edebilir. Yani biraz dokunulsa ve sürtünme olmasa nesne teorik olarak sonsuza kadar hareket edecek. Oysa kuantum mekaniğine göre serbest parçacık olarak algıladığımız bir nesne, yani üzerinde hiçbir dış etki olmayan nesne, her yerde olabilir. Ama doğanın bunun üzerinde etkili olan sayısal özellikleri, ancak; atomlar ve atomaltı nesneler düzeyinde kendini gösterebiliyor. Cisimlerin boyutları büyüdükçe bu etkiler bazı

karmaşıklıkların arasında yok oluyor, o zaman bu nesnelerde koyduğumuz yerde duruyorlar. Fakat bir elektronu siz şuraya koydum diyemiyorsunuz; üzerinde hiçbir kuvvet olmayan bir elektron, evrende herhangi bir yerde bulunabiliyor. Bunu gördüm, buldum dediğiniz anda, o herhangi yerlerden bir tanesi gerçekleşmiş oluyor. Tüm diğer yerlerin serbest bir elektronun yeri olarak ortaya çıkma olasılığı aynı, eşit. Bir elektronun bir atom içinde sahip olabileceği fiziksel durumlar enerji, momentum, açısal momentum gibi fiziksel parametrelerle belirleniyor. Kuantum mekaniği bu değerlerin belli nitelikler taşımasını gerektiriyor. Sistemin bu değerlerle belirlenen fiziksel durumların hangisinde bulunduğunu, ölçme yapmadan bilemiyoruz. Elektronun nerede olduğunu ya da ölçtüğümüzde, ölçmeden önce –diyelim ki milyardabir saniye önce- orada olduğundan bile emin değiliz. Kuantum mekaniğinin hesaplayabilirliği bu kadar.

Evet, kuantum mekaniğinde bir hesaplanamazlık var. Zihin fonksiyonlarında da bir hesaplanamazlık var. Beyin demiyoruz, çünkü bunun fonksiyonlarının bir kısmı, organları denetleyen istemsiz kısmı belki daha kolay anlaşılıyor. Ama burada söz konusu olan, kollara ve bacaklara emir verme, karar verme mekanizması. Bu nasıl fizikle açıklanabilecek? İşte zorluk burada ve kuantum mekaniği burada devreye giriyor. Zihin bir çok şeyi algılıyor, bunları bir şekilde biriktirip, belleğe yerleştiriyor. Fakat önemli olan karar verme aşamasında birikmiş verilerin tümünden daha fazla bir toplam olup olmadığı sorusudur.

Zihin konuşmamıza komutları nasıl veriyor? Herkesin beyninde her an kafasından geçen düşüncelerle bir çok belki milyonlarca karar veriliyor, bu nasıl oluyor? İşte tüm bu verilerin, beyne girmiş olan bilgi kırıntılarının oluşturduğu fiziksel durumlar ve bunların sayıyla ifade etmekte zorlanacağımız kombinezonlarından her biri bir kuantum mekaniksel durumun bir bileşeni gibi görülebilir. Kuantum mekaniksel durum bileşenleri demekle, serbest bir elektronun uzayın herhangi bir noktasında bulunmasını kastediyoruz. Bu bulunuş bir fiziksel durumdur. Hepsi varit bu elektron için, fakat biz elektronu yakaladığımız yani ölçtüğümüz anda diyoruz ki elektron burada; bu durumlardan bir tanesi ortaya çıktı. Bunu dışarıdan müdahale ederek yapıyoruz. Beyin ise zihin fonksiyonları sırasında bu müdahaleyi nasıl yapıyor? Penrose, zihnin çalışma mekanizması ile bir kuantum mekaniksel sistemin özellikleri arasında analoji kurma imkanı olduğunu söylemektedir.

Burada hesaplanamazlık, yani bir algoritmaya indirgenemezlik konusu en temel bir hususu oluşturuyor. Bu iki sistemden bir tanesinde hesaplanamazlık olmadığı gösterilebilirse bütün bu söylenenler ortadan kalkmış olacak. Aslında hesaplanamazlık, bir algoritmaya indirgenemezlik matematikte bilinmeyen bir şey değildir. Mesela bir yüzeyi çinilerle kaplayacaksınız, biçimleri ne olsun ki yüzey arada hiçbir boşluk kalmadan kaplanabilsin.

Matematikçiler, bir yüzeyin hangi şekilde çinilerle periyodik olarak kaplanabileceğinin bir algoritmaya bağlanamayacağını kanıtlamışlardır.

1980'lerde anesteziyologlar tarafından beyin hücrelerindeki mikrotübüller keşfedilmiştir. Bunlar, hücrelerin içinde gayet ince bir iskelet gibi yapı oluşturuyorlar ve mitoz bölünme sırasında ortaya gelerek sınır oluşturup bölünmeyi denetliyorlar. İçlerinde bulunan çok ince lifleri oluşturan protein moleküllerinin ilginç bir özelliği var. Bunların içindeki bir elektron iki değişik durumda bulunabiliyor. Elektronun bu iki durumunu 0 ve 1 durumları gibi alabilirsiniz. Belli bir takım anestetikler verildiğinde bu elektronun yer değiştiremez hale geldiği, yani uyuşturmanın verdiği bilinç kapatılması sırasında bu elektronun donduğu görülüyor. O zaman zihin fonksiyonlarında bu elektronun yer değiştirmesi bir takım kuantum mekaniksel durumlar oluşturmaya yol açabilir. Çünkü elektronun bulunduğu yer için matematiksel olarak bir kuantum mekaniksel durum yazabiliyorsunuz. Bunun gibi bir hücrede milyonlarca var, nöron şebekeleri içinde kaç tane olduğunu ve bunların yaratabileceği değişik sonuç durumlarını düşünün. İşte Penrose'nin, acaba olsa olsa nerede olabilir sorusuna bulamadığı cevap bu. Bunun uygun bir aday olabileceğini 1992 yılında bir anesteziyologun ona söylemesi üzerine öğrenmiştir. Ama gene de bizi şu soruyla karşı karşıya bırakmaktan da kendini alamıyor: " Acaba parça bütünü anlayabilecek mi? Parça bütünü içine alabilecek mi? Yani, biz acaba bunu anlama yeteneğine sahip miyiz?" (Gödel teoremi, Russel paradoksu, veya çok eskilerin dediği irade-i külliye/ irade-i cüzziye sorunu gibi bir şey). Aynı soru kuantum mekaniği için de soruluyor: Acaba daha temel düzeyde bilgi (i) Doğada mı yok? (ii) Var da doğa bize yasaklamış mı? (iii) Yoksa bizim yeteneklerimiz mi elvermiyor? Şimdilik genel inanç (i) doğrultusunda.

Psikolojik Yaklaşım

Beynin nöroanatomik, biyokimyasal ve fizyolojik açıdan incelenmesi yoğun biçimde sürmektedir. Fakat beyni bir canlının içinde işlev gören bir uzuv olduğunu görerek değerlendirirsek, ister istemez davranış bilimleri de işin içine girmektedir. Çünkü özellikle gelişmiş beyinli memeli hayvanların önemli özelliklerinden biri de çevreleri ile etkileşime girmeleri ve bu sayede yeni şeyler öğrenerek bunları daha sonra hatırlayabilmeleridir. Bu davranışlar açısından da beyin bilgisayar etkileşimi ve benzerliklerine bakılması gereklidir.

Bilgisayarlar ile insanlar arasında ilk bakışta öğrenme ve bellek konusunda çok önemli işlevsel benzerliklerin bulunduğu biliniyor. Öğrenme ve bellek mekanizmaları bize bilgi edinme ve deneyimlerden yararlanma olanağı sağlamaktadır. Bilgisayarlar da genelde öğrenme ve belleklerinde bilgi tutabilme özelliklerine sahipler. Bu açıdan bakıldığında ortaya felsefi sorunlar çıkmaktadır. Bunlardan biri Turing'in öngördüğü öğrenme makinesidir. Bu makinenin insan gibi

öğrenebildiğinin testi de turing testi olarak bilinmektedir. Bu konu hakkında felsefi yaklaşım başlığı altında bilgi verildiğinden burada girilmeyecektir.

Böyle bir öğrenme makinesinin temelinde yatan aksiyomatik sistemdeki belirsizliğin Gödel tarafından kanıtlanmış olması, zaten bilginin niteliği ve bilgi edinme yöntemlerinin yeniden gözden geçirilmesine yol açtığı gibi insan bilgisayar karşılaştırmasının temelindeki varsayımların sorgulanmasını da gündeme getirmiştir. Bilgisayarların öğrenmelerine ilişkin şemalarda genellikle bir girdi kanalı, bir işlemciye denk gelen bir kutu ve bilgisayarın ürününü gösteren bir çıktı kanalı gösterilir. Bu girdi ve çıktı kanallarına ve kapağını açarak işlemci kutusunun içine bakıldığında, görülen olgular bilgisayar ile beyin arasında önemli farkların olduğunu ortaya koymaktadır. Burada olayın psikolojik yönüyle ilgili olarak Freudcu bir yaklaşımla nerede bunun libidosu veya Neyzen TEVFİK'i anımsayarak fikri varsa efkarı nerede bunun diye sorular sorulabilir. Tüm bu soruların dışında basit bir örnekle konuya yaklaşalım: bir bilgisayarınız var, fakat her yerde iyi çalışan bilgisayarınız bazı yerlerde doğru çalışmıyor, üstelik sabahları daha iyi öğleden sonra ise kötü çalışıyor yani tekliyor. Ne düşünürsünüz? Bilgisayarınızın bozulduğunu düşünerek tamire götürürsünüz. Ve belki de tamire götürürken bilgisayarınızın insanlaşmaya başladığını düşünebilirsiniz. Burada belirtilmek istenen aslında bilgisayarlardan hiç beklenmeyen bu davranışın bizim hem psikolojimizde hem de fizyolojimizde yerleşik bir olgu olduğudur. Çünkü bilgisayarlardan çok farklı olarak bizim için olayların zamanla ve mekanla kayıtlı bir yanı vardır. Olayların zaman içindeki dizilimi ve mekan içindeki dağılımı bizi temelden etkilemekte ve daha duyu ve algılama gibi temel süreçlerden başlayarak bizi tamamıyla biçimlendirmektedir.

Bilgisayarlarda girişleri iyi bir şekilde düzenlediğiniz takdirde işlem kutusunun niteliğini incelemeden ne olursa olsun çıktının ne olacağını biliyoruz. Buna paralel olarak psikolojideki davranışçı ekole göre, siz kişinin girdilerini gerektiği biçimde düzenleyebildiğiniz sürece kutu, yani a, b, veya c kişileri avukat, doktor veya mühendis olabiliyor. Bu tür radikal davranışçı yaklaşımı bugünkü bilgisayar teknolojileriyle birleştirdiğinizde bilgisayarla beyin arasında çok fazla bir benzemezlik olmadığı görülebilir. Ancak bu tür yaklaşımın geçerli olmadığı, girdilerle çıktılar arasındaki kutunun içeriği ve özelliklerinin araştırılmaya başlanmasıyla gündeme gelmiştir. Özellikle Gestalt psikolojisinin vurguladığı görüş, algılamada uyaranları teker teker inceleyip sonuçları sentezlemenin mümkün olamayacağı tezidir. Yani algılamada bütün, parçalarının toplamından farklıdır. Gestalt psikolojisine göre, bir olayı anlamak için tümünü bir arada ve bir anda algılamak gerekli, çünkü olayın tümünün dinamiği, parçaların teker teker incelenmesi ile ortaya çıkan tablodan farklıdır. Bir karenin uçlarına yerleştirdiğimiz ışıkları yakıp söndürmeyi frekansı arttırarak sürdürdüğümüzde önce kare görünen şeklin frekans arttıkça daire

veya çember şeklinde algılandığını görürüz. Bu örnek bize çoğu kez bir olayı parçalarına bölüp parçalarının her birinin beynimizi nasıl etkilediğine bakarak bir bütün yaratmamızın mümkün olmadığını göstermektedir. Uyaranların yada üzerimizde psikolojik etki yaratan durumların teker teker incelenmesinin, bu uyaran yada durumların toplamının yarattığı tabloyu tümüyle anlamamıza yeterli olmayacağı gerçeğidir. Bu bakımdan beynimizi etkileyen uyaran yada durumları birer bağımsız girdi olarak değerlendirmemiz mümkün değildir. Uyaranların üzerimizde yaptıkları etki, zaman ve mekan içindeki dizilimlerine ve birbirleriyle etkileşimlerine bağlıdır.

Sonuç olarak, beynimiz ve beynin bağlı olduğu canlı organizma, zaman ve mekan içinde davranışlarını değiştiren, zamandan ve mekandan etkilenen bir yapıya sahiptir. Bunlar şu aşamada bilgisayarda mevcut değildir. Bilgi edinmede, felsefenin ortaya çıkardığı sınırların yanısıra, bugünkü koşullarda bile beyin ile bilgisayar arasında bir koşutluğun ancak basit bir ilk yaklaşım için geçerli olduğu görülmektedir.

Felsefi Yaklaşım

Yapay zeka felsefesi en geniş anlamıyla yapay zekanın gerçekten mümkün olup olmadığını soruşturan bir felsefe koludur. Bilgisayarlar düşünebilir mi? Sorusu yapay zeka felsefesinin en temel sorunudur. Bilgisayarların icadından buyana, bu soru bir çok felsefeci, bilim adamı veya yapay zeka araştırmacısı tarafından tartışılmıştır. Bu güne kadar bir problem olarak kalmasının nedeni bu sorunun cevabı hakkında ortak bir uzlaşma sağlanamamasındandır. Hatta, bunun felsefi bir problem mi? Yoksa empirik bir problem mi? Olduğunda dahi mutabık kalınamamıştır.

Turing makinesi ve turing testi

Yapay zeka felsefesini ilk ortaya çıkaran kişi ünlü ingiliz mantık ve matematikçisi Alan Turing'dir. Dartmouth konferansından altı yıl önce, yani 1950 yılında Turing, Mind adlı felsefe dergisinin Ağustos sayısında *Computing Machinery and Intelligence* adlı bir makale yayınlamıştır. Bu makalede Turing "Makineler düşünebilir mi?" sorusunu dikkatli bir felsefi tartışmaya açmış ve makineler düşünebilir iddiasına karşı olan itirazları reddetmiştir.

1936 yılında Turing bilgisayar tasarımının mantıki temelleri üzerine bir makale yazmıştır. Bu makalenin konusu matematiksel mantığın soyut bir problemi ile ilgilidir ve bu problemi çözerken Turing bugün Turing makinesi diye adlandırılan, program depo eden genel amaçlı bilgisayarı kuramsal olarak icat etmeyi başarmıştır. Turing makinesi kuramsal bir hesap makinesi

olup hesaplarını karelere bölünmüş ve her karede yalnızca bir sembol bulunabilen bir bant aracı ile yapar. Sadece sonlu sayıda içsel durumları vardır. Bir karedeki sembolü okuduğu zaman halihazırdaki durumuna ve sembolün ne olduğuna göre durumu değişebilir.

Alan Turing ayrıca Turing testi olarak adlandırılan ve bir bilgisayarın veya başka bir sistemin insanlarla aynı zihinsel yetiye sahip olup olmadığını ölçen bir test geliştirmiştir. Genel anlamda bu test bir uzmanın, makinenin performansı ile bir insanınkini ayırt edip edemeyeceğini ölçer. Eğer ayırt edemezse, makine insanlar kadar zihinsel yetiye sahip demektir. Bu testte bir insan ve bir bilgisayar, deneyi yapan kişiden gizlenir. Deneyi yapan hangisiyle haberleştiğini bilmeden bunların ikisiyle de haberleşir. Deneyi yapan kişinin sorduğu sorular ve deneklerin verdiği cevaplar bir ekranda yazılı olarak verilir. Amaç, deneyi yapanın uygun sorgulama ile deneklerden hangisinin insan, hangisinin bilgisayar olduğunu bulmasıdır. Eğer deneyi yapan kişi güvenilir bir şekilde bunu söyleyemez ise, o zaman bilgisayar Turing testini geçer ve insanlar kadar kavrama yeteneğinin olduğu varsayılır.

Çin odası deneyi

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 ingilizce olarak açıklayan bir kural kitabı bulunsun. Kurallar çinceyi tamamen biçimsel olarak, yani söz dizimlerine uygun olarak açıklamaktadır. Daha sonra odaya başka çince simgelerin getirildiğini ve size çince simgeleri odanın dışına götürmek için, başka kurallarda 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.

Bilgi, bilinç ve yapay zeka

Beyin etten yapılmış bir bilgisayar mıdır? Bir bilgisayar üretildiği fiziksel malzemeler dolayısıyla zamana tabi olarak çalışır ve devrelerinin bağlantılarına ve yazılıma göre ulaşılan

sonuçlar neden-sonuç ilişkisi bakımından sıkı bir gerekirciliği (determinizmi) ortaya koyar. Bu bakımdan, insan bilinci de, insanın tüm bedensel işlevlerinin yönetim merkezi olan beynin, elektriksel ve kimyasal süreçlere bağlı olarak, fiziksel varolanın (uzay ve zamanda varolanın) tabi olduğu neden-sonuç ilişkisine, nedenselliğe bağlı olan bir süreçten başkası değil midir? Yani bilinç ve akıl tümüyle fiziksel süreçlere indirgenebilir mi?

Bu sorular düşünce tarihi içinde derin kökleri olan önemli sorulardan bir kaçıdır. Eğer biz tüm insani özelliklerin fiziğe tabi olan bedensel işlevlere indirgenebileceğini savunuyorsak, bu yaklaşımla beynin etten yapılmış bir bilgisayar olduğunu, yani yapay zekanın henüz yeterince gelişmemiş bir insan prototipi olduğunu kabul ediyoruz demektir. Buna karşılık, insanın yalnızca fiziksel süreçlere tabi olan bir makineye indirgenemeyeceğini savunuyorsak, bunun gerekçelerinin ortaya konması gerekir.

Şimdi, eğer tüm bilgimizin deneyle başladığını kabul ediyorsak, bilginin ortaya çıkması için gerekli iki koşulu şöyle ifade edebiliriz: Deneyimden gelen malzeme ya da veriler ve bu verilerin, aklın kendi sahip olduğu formlar aracılığı ile kalıba dökülmesi ve sonuçta bilginin üretilmesi.

Verilerin kalıba dökülmesi, önerme formuna sokulması bir fiildir ve bu fiilin yapılması için bilincin ortaya çıkması gerekir. Yani her bilgi fiili bir bilinç fiilidir. Şimdi soru şudur: Bilinç bir beyin süreci midir? Yoksa beyin süreçlerinin arkasında duran ve bu süreçlerin sonucunda, bir şeye (nesneye) yönelmek suretiyle ortaya bir bilgi konulmasını sağlayan etkin neden, bilinç fiilinin kendisi midir?

Bilgi bir bilinç durumudur, düzensiz bir veriler topluluğunun algılanması değildir. Şeylerin bir bilgi nesnesi yada onların bağlantılarının bilgisi olarak ortaya çıkması, o nesneye bir birlik verilmesi ile olanaklıdır, bu ise bu birliği veren öznenin, "ben"in kendi birliğinin bilincinde olmasıyla olanaklıdır. Yani her bilgiye birliğini veren ben bilinci her bilgiden önce gelmektedir. Eğer beyin süreçleri ile "ben" bilinci aynı şeyse, zamana ve nedenselliğe tabi olan beyin süreçlerinin nasıl olup da farklı ben bilinçlerinin ortaya çıkmasını sağladığı ise karanlıkta olan bir sorudur.

Aklın deneyden gelen uyarılara dayalı bilgi üretmesinin yanında, kendisi deneyden gelmeyen, ama deneyle gelen malzemeyle doğa bilimlerinin yapılabilmesinin koşulunu oluşturan matematik ve matematiksel nesnelerle ilgili değerlendirmeler, bilincin beyin süreçlerine indirgenemeyeceği yönünde bir destek sağlamaktadır.

Matematiğin ve matematiksel nesnelerin (sayı, üçgen gibi) ne olduğu sorusunun yanıtı kolaylıkla verilemez, ama ne olmadığının yanıtı üzerine şunlar söylenebilir. Matematiğin

nesneleri ve onların bağıntıları zamana ve neden-sonuç ilişkisine bağımlı değildir. Bu tür nesnelerin bağıntılarını özsel olarak farklı ilkeler yönetmektedir (çelişmezlik ilkesi gibi).

Eğer matematiksel nesnelerin zamana ve neden sonuç ilişkisine tabi olmadıklarını görüyorsak, bundan, bu nesnelerin fiziksel süreçlerin dışında kalan bir dayanağa sahip oldukları sonucu çıkar. Bu nedenle matematiksel nesneler, fiziksel süreçlere tabi olarak ortaya çıkan şeyler değildir; ama fiziksel olanın, malzemenin, düzene ve sıraya sokulmasının dayanağını oluşturması nedeniyle, fiziksel süreçlerin insan için anlaşılabilir ve bilgisine ulaşılabilir bir şey olmasını sağlarlar.

Bu bakımdan insan beynini yalnızca fiziksel süreçlere tabi olan bir bilgi işleme merkezi olarak görmek, matematiksel nesneleri de zamana ve neden-sonuç ilişkisine bağlı olarak görmek sonucunu getirir ki, o zaman sayı, üçgen gibi fiziksel nesnelerin bağıntılarının kesinlik ve zorunluluğunun hesabını vermek olanaksız olacaktır. Yani fiziksel süreçler, bu süreçlerin dışında kalan ilkelerle işleyen soyut nesnelerin dayanağı olamazlar. O halde, eğer matematiksel nesneler ve matematik, zamana ve neden-sonuç ilişkisine tabi değillerse ve bunların dayanağını fiziksel süreçler oluşturmuyorsa, bu dayanağın fiziksel süreçlere tabi olmayan bir şey olduğunu, yani aklın kendi unsurlarının bu süreçlerin dışında olduklarını düşünmek durumundayız. Bunun ise anlamı şudur: İnsan bilinci ve aklı, yalnızca fiziksel süreçlere tabi olan ve nöron ağlarından oluşmuş beyin organının üstünde bir "yer"e, "iç"e sahiptir. Bu yer (iç), aklın yanı sıra, "özgür irade"nin de dayanağını oluşturur.

Eğer insan varlığı, yalnızca fiziksel bir nesne olarak görülürse, yani "empirik ben"den ibaretse, burada özgür iradeye yer yoktur. Çünkü fiziksel bir nesne olarak zamana ve nedenselliğe tabi olan insan varlığı sıkı bir gerekircilik içinde belirlenmiştir. Öte yandan, insanın tüm empirik belirlenimlerinin arkasında duran, onun zeminini oluşturan, ama zaman ve nedensellikle belirlenmemiş bir "aşkınsal (transandantal)" yanı vardır ki, bilincin ortaya çıkmasının arkasında duran ve özgür iradenin dayanağı olan, onun bu aşkınsal yanıdır.

Günümüzün ünlü fransız filozofu Georges Canguilhem araçsallıkçılığın (instrumentalisme) her türüne karşı çıkarak, teknik sapmanın her köşe bucağa yayılmasını eleştirmektedir. "Beyin ve düşünce" adlı yazısında *elektronik hırdavatçılığın* her kesimi etkisi altına aldığını vurgulayan filozof, yapay zekadan enformasyon modellerine değin her türlü teknolojik başarının getirdikleri kadar götürdükleri de olduğunu savunmuştur. İnsan zihninin bir bilgisayara sığdırılamayacağını, ve bilgisayarında sonuç olarak insan zihninin tüm yetilerinin üstesinden gelemeyeceğini dile getiren filozof, bu anlayışın eskilerin frenoloji görüşüne benzediğini söyler. Oynanılan, ayarlanmaya çalışılan, belirsiz amaçlara yönlendirilen bir insan dünyasına karşı; düşüncenin kaçınılmaz ve normal durumuna denkmiş gibi yutturulan bir teknik evren aracılığı ile ortaya çıkan açmazı açmanın tek yolunun felsefeye düştüğünü söyleyen Canguilhem'e göre, " şu andaki

egemenliğinin başkasına devredilemez hakkı olarak *ben*'in savunulması felsefenin biricik görevidir."

Sonuç olarak yapay zeka çalışmalarının ve nörolojinin yönünün ve olanaklarının belirlenebilmesi için, insanın ve insan aklının ne olduğunun soruşturulması, ama bu soruşturmanın yalnızca bilişim bilimleri ve deneysel psikoloji alanında değil, metafiziksel olarak felsefe içinde de soruşturulması gerekmektedir

BÖLÜM 2
ZEKİ AJANLAR

2.1 GİRİŞ

Ajan, sensorlarıyla çevresini algılayan ve efektörleriyle bu çevre üzerinde davranan herhangi bir şeydir. İnsan sensor olarak göz kulak ve diğer organlara, efektör olarak da el, kol, ağız ve diğer vücut kısımlarına sahiptir. Robotik ajan, kamera ve infrared bulucuları sensor olarak değişik motorları da efektör olarak kullanır.

2.2 AJANLAR NASIL DAVRANMALI

Rasyonel ajan doğru şeyi yapan ajandır. Bu yanlış bir şey yapmaktan daha iyidir ama doğru şeyi yapmak ne demektir? İlk yaklaşım, doğru hareket ajanı en başarılı yapacak harekettir. Bu durumda ajanın başarısına ne zaman ve nasıl karar verileceği sorusunu ortaya çıkarır.

Performans Ölçüsü

Ajanın ne kadar başarılı olduğunu saptayan kritere performans ölçüsü denir. Performans ölçüsünün dikkatli bir şekilde saptanması gerekmektedir. Örneğin kirli zemini temizleyen bir ajanı gözönüne alalım. Burada performans ölçüsü olarak 8 saatlik çalışmada temizlenen kir miktarı alınabilir. Daha karmaşık performans ölçüsü yalnız temizlenen kir miktarını değil aynı zamanda tüketilen enerji ve çıkarılan gürültüyü de dikkate alabilir.

Değerlendirme Zamanı

Performansın ne zaman değerlendirildiği de önemlidir. Örneğin temizlik ajanının performansını ilk saat sonunda değerlendirirsek işe hızlı başlayan ama sonra yavaşlayan belki de hiç çalışmayan ajan ödüllendirilmiş, sürekli çalışan cezalandırılmış olur. Bu nedenle performansı günlük veya ömür boyu şeklinde daha uzun vadede değerlendirmek gerekir.

Mükemmellikle rasyonelliğin ayırt edilmesi gerekir. Mükemmel ajan davranışlarının vereceği sonucu bilir ve buna göre davranır. Fakat gerçekte mükemmellik imkansızdır. Aşağıdaki örneği göz önüne alalım:

Okula gelirken demiryolunun karşısında bir arkadaşınızın gitmekte olduğunu gördünüz. Demiryolu geçiti açıktı ve arabalar geçiyordu. Arkadaşınıza başka türlü erişemeyeceğiniz için demiryolundan geçmeye başladınız. Bu sırada bir yolcu uçağının kapısı üzerinize düştü.

Demiryolundan geçmek rasyonel olmayan bir davranış mıdır?

Rasyonellik algılananlara bağlı olarak beklenen başarı ile ilgilidir. Çoğu zaman demiryolundan geçmek başarılı sonuç verdiği ve üzerinize uçak kapısı düşeceğini tahmin edemeyeceğiniz için demiryolundan geçmek rasyonel bir davranıştır. Eğer ajanın düşen kapıyı

saptayabilecek bir radarı varsa o zaman hareket rasyonel değildir. Benzer şekilde demiryolu geçidi kapalı iken geçmek de rasyonel olmayan bir davranıştır. Diğer bir deyişle ajanı algılayabileceği bir durumu dikkate almayarak başarısız olduğu zaman suçlayabiliriz.

Sonuç olarak herhangi bir andaki rasyonellik dört şeye bağlıdır:

- Başarı derecesini belirten performans ölçüsü,
- Ajanın o ana kadar algıladığı her şey (algı serisi-percept sequence),
- Ajanın çevre hakkında bildikleri,
- Ajanın yapabileceği hareketler.

İdeal Rasyonel Ajan

Yukarıdaki sonuçlardan yola çıkarak ideal rasyonel ajan aşağıdaki şekilde tanımlayabiliriz:

Her olası algı serisi için, algı serisi ve sahip olduğu bilgileri kullanarak performans ölçüsünü maksimize edecek şekilde davranan ajan ideal ajandır.

Demiryolu geçidinin kapalı olup olmadığına bakmaz isek algı serisi bize hızla yaklaşan bir treni söyleyemez. Demiryolu geçidine bakmadan geçmek riski çok yüksek olduğu için rasyonel bir davranış değildir. İdeal rasyonel ajan adımını atmadan önce bakma eyleminde bulunmalıdır. Çünkü bakmak beklenen performansı artırır. Bu nedenle yararlı bilgileri elde etmek amacıyla yapılan eylemler rasyonelliğin bir parçasıdır.

Algı Serisinden Eyleme İdeal Eşleme

Ajanın davranışı yalnız algı serisine bağlı ise olası tüm algı serilerine karşı gelen eylemler tablo haline getirilerek bir ajan tanımlanabilir. Çoğu zaman bu tablo çok uzun olacaktır. Oluşturulan tabloya algı serisinden eyleme eşleme denir. Eğer eşleme ajanı tanımlıyorsa ideal eşleme de ideal ajanı tanımlar. Eşleme için tablonun her bir elemanının ayrı ayrı belirtilmesi gerekmez. Örneğin hesap makinesindeki karekök fonksiyonunu basit bir ajan olarak göz önüne alalım. Bu ajanın algı serisi basılan tuşlardır. İdeal eşleme; girilen pozitif sayı x ise $z^2 \approx x$ olacak şekilde 4 basamak doğrulukta z'yi göstermektir. Bu amaçla tablo kullanmak yerine Newton yöntemi kullanılarak yazılan program ile ajan tanımlanabilir. Tablo çok uzun olmasına karşın ajan çok kısa bir programdır. Aşağıda tablo ve program görülmektedir:

Algı X	Eylem Z
1.0	1.0000
1.1	1.0488
1.2	1.0954
1.3	1.1401
...	

```
Function SQRT(X)
  z←1.0    /* ilk tahmin */
   repeat until  |z^2-x|<10^-4
      z←z-(z^2-x)/(2z)
end
return z
```

Otonomi

Eğer bir ajanın eylemleri tamamen sahip olduğu bilgilere bağlıysa yani algı serisini dikkate almıyorsa otonom değildir. Eğer bir sistemin davranışları kendi deneyimleri ile saptanıyorsa otonomdur. Otonom kelimesi insanın doğrudan kontrolü altında olmayan anlamında da kullanılır. Örneğin otonom kara aracı (insansız). Otonom olan ajanlar çevre koşulları değiştiğinde yeni koşullara adapte olarak görevini başarı ile sürdürebilir. Eğer sadece önceden verilen bilgileri kullanırsa başarısız olma olasılığı yüksektir.

2.3 ZEKİ AJANLARIN YAPISI

YZ ajan programı yazma işidir. Bu programlar algıdan eyleme eşleme yapan programlardır. Yazılan programlar bir yapı üzerinde çalıştırılacaktır. Bu yapı bilgisayar olabileceği gibi özel amaçlı donanım da olabilir. Yapı sensorlardan gelen algıları programa aktarır, programı çalıştırır ve efektörler ile programın davranışını gerçekleştirir. Ajan yapı ve programdan meydana gelmektedir:

Ajan = Yapı + Program

Ajan programı tasarlamadan önce ajanın çalışacağı çevrenin, algılamaların, eylemlerin ve amaçların tanımlanması gerekir. Aşağıdaki tabloda bazı ajan tipleri verilmiştir.

Ajan Tipi	**Algılama**	**Eylem**	**Amaç**	**Çevre**
Tıbbi teşhis sistemi	Bulgular, hastanın cevapları	Sorular, testler	Sağlıklı hasta, minimum maliyet	Hasta, hastane
Uydu görüntü analiz sistemi	Değişik yoğunluk ve renkte pikseller	Görüntüyü sınıflandır, bas	Doğru sınıflandırma	Uydudan gelen görüntü
Rafineri Kontrolörü	Sıcaklık, basınç okumaları	Valfleri aç, kapa; sıcaklığı ayarla	Saflığı ve emniyeti maksimum yap	Rafineri

İnteraktif İngilizce öğreticisi	Yazılı kelimeler	Alıştırmaları, önerileri,düzeltmeleri yaz	Öğrencinin notunu maksimize et	Öğrenci kümesi

Ajan Programları

Zeki ajanlar oluşturulurken aynı iskelet kullanılacaktır: çevreyi algılamak ve eylem üretmek. Ajan programının en basit hali aşağıda görülmektedir:

```
Function İskelet_Ajan( algı)
static : bellek   /* ajanın belleği */

bellek← Güncelle_Bellek(bellek, algı)
eylem← Eniyi_Eylemi_Seç(bellek)
bellek← Güncelle_Bellek(bellek, eylem)
return eylem
```

Eşleme algı serisinden eyleme yapılmaktadır ama ajana yalnız o anki algı gelir. Bu algıların seri halinde saklanıp saklanmaması ajana bağlıdır. Bazı çevrelerde algı serisi saklanmaksızın oldukça başarılı olunabilir. Karmaşık çevrelerde ise tüm algıların saklanması fizibil olmamaktadır.

Ajan programı yazmanın en basit yolu tablo kullanmaktır (look-up table). Bu durumda olası tüm algı serisinin bellekte tutulması ve indeks kullanarak erişilmesi gerekir. Tablo kullanımında aşağıdaki olumsuzluklar ortaya çıkar:

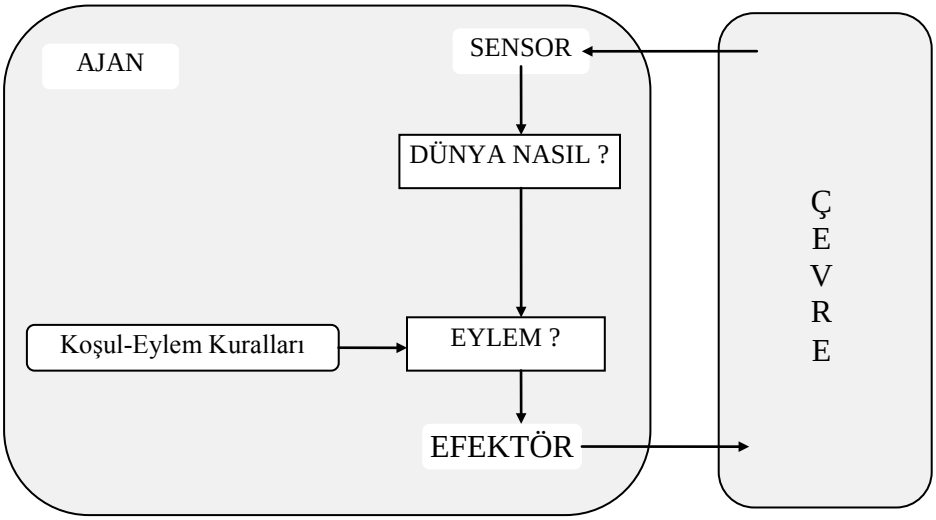
- Yalnız satranç oynayabilen basit bir ajan için bile gerekli tablonun 35^{100} girişi olması gerekir.
- Tasarımcının tabloyu oluşturması çok uzun zaman alır.
- Ajan otonom değildir. Çünkü en iyi eylem hesabı tamamen önceden girilmiştir.
- Ajana bir derece otonomi tanınarak öğrenme mekanizması oluşturulsa bile tüm girişler için tablonun doğru değerlerini bulması sonsuza kadar sürer.

Basit Refleks Ajanlar

Bir kameradan gelen görüntü 50 Mbyte/sn. hızındadır (saniyede 25 çerçeve, her çerçeve 1000*1000 piksel ve her piksel 8 bit renk ve 8 bit yoğunluk bilgisi). Bir saat için gerekli look-up tablosu $2^{60*60*50M}$ girişli olacaktır. Genel giriş çıkış ilişkileri kullanılarak tablo kısaltılabilir. Örneğin öndeki araç fren yaparsa fren lambaları yanar ve sürücü buna dikkat ederek frene basar. Aynı işlem görsel giriş kullanılarak "öndeki araç fren yapıyor" koşulu ile ajan programındaki "fren yap" eylemi ilişkilendirilebilir. Bu ilişkiye koşul-eylem (condition-action) kuralı denir ve aşağıdaki şekilde yazılabilir:

EĞER Öndeki_Araç_Frenliyor İSE frenle (if/then)

İnsanlarda benzeri davranışlar bir öğrenmenin sonucunda (araba sürme gibi) veya refleks olarak (kızgın sobadan elin çekilmesi gibi) yapılır. Aşağıda koşul-eylem kuralının ajana algıdan eyleme bağlantıyı nasıl sağladığı görülmektedir.



Basit Refleks ajanın şematik diyagramı.

```
Function Basit_Refleks_Ajan(Algı)
static : Kurallar; /* koşul-eylem kuralları */

durum ← Giriş_Yorumla( algı )
kural ← Kural_Eşleme(durum, kurallar)
eylem ← Kural_Eylem[kural]
return eylem
```

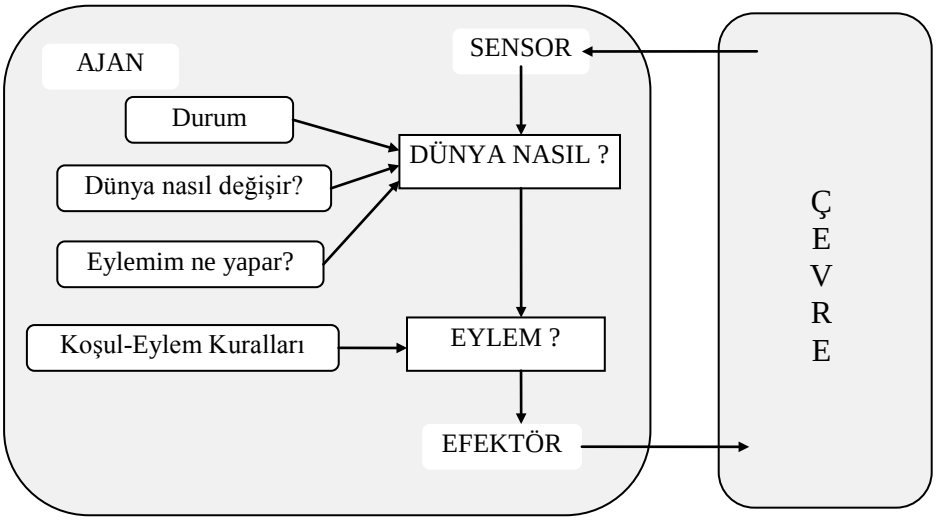
Basit Refleks ajan. Algılamayla tanımlanan mevcut duruma uyan kuralı bularak çalışır.

- Giriş_Yorumla : Algılanan mevcut durumu soyut olarak tanımlar (abstraction).
- Kural_Eşleme : Kural kümesinde mevcut duruma uyan ilk kuralı verir.
- Kural_Eylem : Kurala bağlı olarak yapılacak eylemi verir.

Değişimleri İzleyen Ajan

O anki algılamaya bağlı olarak doğru karar verilebiliyor ise basit refleks ajanlar başarılı olabilir. Arabanın arkasında fren lambalarına ek olarak dönüş sinyal lambaları da yer almaktadır. Frene basılıp basılmadığını saptamak için arabanın her iki kenarındaki lambanın kontrol edilmesi gerekmektedir. Bu amaçla bir önceki görüntünün saklanması gerekmektedir. Bir önceki görüntüde her iki lamba sönükse ve o anki görüntüde ikisi de yanıyor ise frene basıldığını söyleyebiliriz. Bu nedenle doğru eylemin seçilebilmesi için bazı bilgilerin saklanması gerekmektedir. Buna iç durum (internal state) adı verilir.

Sensorlardan gelen bilgi daha önceki duruma bağlı olarak farklı sonuçlar verebiliyor ise önceki durumun da saklanması gerekir. Dünyanın durumu yalnız o anki girişe değil bir önceki duruma da bakılarak saptanır. Aşağıda iç durumlu bir refleks ajan görülmektedir.




```
Function Durumlu_Refleks_Ajan(Algı)
  static : Kurallar; /* koşul-eylem kuralları */
           Durum;  /* mevcut dünya durumu */

  durum ← Yenile_Durum(durum, algı )
  kural ← Kural_Eşleme(durum, kurallar)
  eylem ← Kural_Eylem[kural]
  durum ← Yenile_Durum(durum, eylem )
return eylem
```

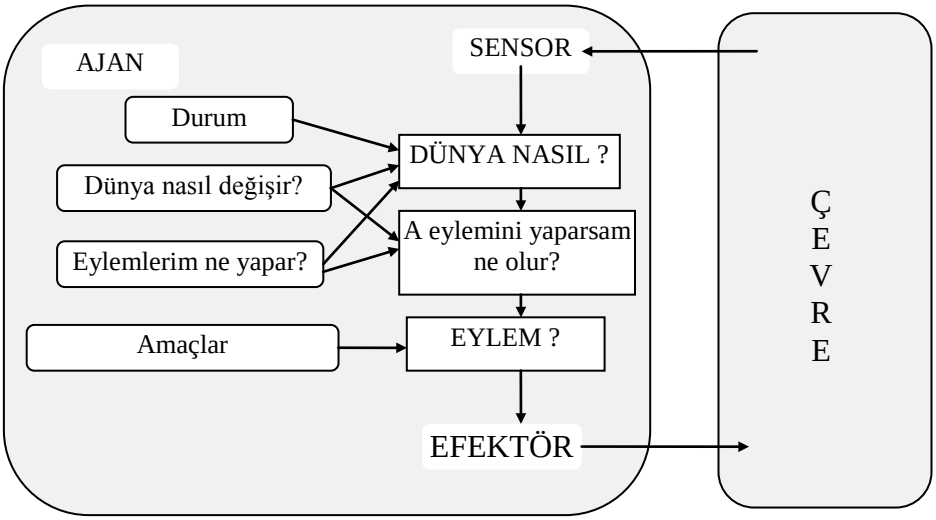
Yenile_Durum: Sensorlardan gelen bilgiye veya yapılan eyleme bağlı olarak yeni iç durumu oluşturur.

Amaç tabanlı Ajanlar

Çevrenin o anki durumunu bilmek ne yapılacağına karar vermeye yetmeyebilir. Örneğin bir kavşakta araba sağa, sola dönebilir veya doğru gidebilir. Doğru karar arabanın nereye gideceğine bağlıdır. Diğer deyişle ajan o anki durumla birlikte amaçla ilgili bilgilere de gereksinim duyar. Ajan programı, olası kararların sonuçları ile bu bilgiyi birleştirerek doğru eylemde bulunabilir. İstenen amaca bazı durmlarda kolayca erişilebileceği gibi bazen çok karmaşık işlemler yapılması gerekebilir. Ajanın amacına erişebilmesi için yapacağı eylem sırası YZ'nin alt alanları olan arama (search) ve planlama (planning) da incelenmektedir.

Bu şekilde karar verme daha önce anlatılan koşul-eylem kurallarından temel olarak farklıdır. Refleks ajan fren lambasını gördüğü zaman fren yapar. Amaç tabanlı ajan ise öndeki aracın fren lambaları yandığı zaman onun yavaşlayacağını çıkarır. Öndeki araca çarpmama

amacını gerçekleştirecek eylem ise fren yapmaktır. Her ne kadar amaç tabanlı azan etkin görünmese de esnektir. Örneğin yağış başladığı zaman frenlerin etkin bir şekilde kullanılabilmesi için bilgisini yenileyebilir. Diğer yandan refleks ajan için çok sayıda koşul-eylem kuralı yazmak gerekir. Amaç tabanlı ajanlarda amacı değiştirerek farklı noktalara erişmek mümkündür. Refleks ajanlar ise sadece bir noktaya giderler. Aşağıdaki şekilde amaç tabanlı ajanın yapısı görülmektedir.

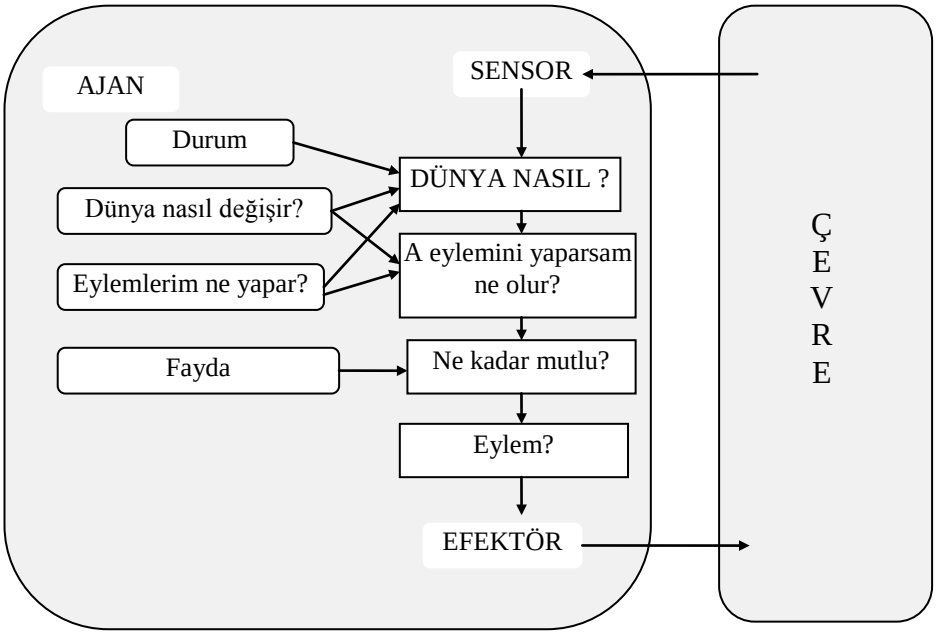


Amaç tabanlı ajan.

Fayda Tabanlı Ajanlar

Amaçlar yalnız başına yüksek kalitede davranış üretmek için yeterli değildir. Örneğin arabanın istenen yere gidebilmesi (amaca erişmek) için izleyebileceği bir çok yol olabilir. Bu yolların bazıları diğerlerinden daha hızlı, daha güvenli veya daha ucuz olabilir. Amaçlar "mutlu" ve "mutsuz" durumları arasında kaba bir ayırım sağlar. Oysa daha genel performans ölçüleri amaca erişirken ne kadar mutlu olduğunu gösterebilir. Mutlu kelimesi bilimsel görülmediği için bunun yerine fayda (utility) kelimesi kullanılmaktadır. Fayda fonksiyonu; mevcut durumu, mutluluk derecesini gösteren bir gerçel sayıya dönüştürür. Fayda fonksiyonları tanımlanırken çelişkiye düşülebilir. Örneğin hız ve emniyetin ağırlıkları uygun şekilde seçilmelidir. Birden fazla amaç olduğu zaman bu amaçların da önem sırasına göre sıralanması gerekebilir.

Amaçlar her ne kadar kaba olsa da istenen amaca erişmek için gereken eylemi kolayca bulurlar. Bazı durumlarda fada fonksiyonu bir amaç kümesi şeklinde de ifade edilebilir. Aşağıdaki şekilde fayda tabanlı ajanın yapısı görülmektedir.



Fayda tabanlı ajan.

2.4 ÇEVRE

Bu bölümde ajan ve çevre ilişkileri incelenecektir. Önce ajanların çalışacağı farklı çevreler ve bu çevrelerin ajan tasarımındaki etkileri incelenecektir.

Çevre Özellikleri

Çevre aşağıdaki temel özelliklere bağlı olarak sınıflanlandırılabilir.

- **Erişimli / Erişimsiz** (Accessible vs. inaccessible)

Eğer ajanın sensorları çevrenin durumunu tamamen saptayabiliyorsa çevre ajan için erişilebilirdir. Çevrenin erişilebilir ise değişimleri izlemek için durum saklaması gerekmez.

- **Deterministik/ Deterministik olmayan** (Deterministic vs. nondeterministic)

Eğer çevrenin gelecek durumu o anki durum ve ajanın eylemine bağlı olarak tespit edilebiliyor ise çevre deterministiktir. Eğer çevre erişimli değil ise deterministik olmadığı kabul edilebilir.

- **Bölümlü/ Bölümsüz** (Episodic vs. nonepisodic)

Bölümlü çevrede ajanın deneyimleri bölümlere ayrılmıştır. Her bir bölüm ajanın algı ve eyleminden oluşur. Eylemin kalitesi sadece o bölüme aittir. Bir önceki bölümdeki eylemin o anki bölümde bir etkisi yoktur. Bölümlü çevre, ajanın ileriyi düşünmesi gerekmediği için daha basittir.

- **Statik / Dinamik** (Static vs. dynamic)

Ajan karar verirken çevre değişiyorsa dinamik değişmiyor ise statiktir. Ajan karar verirken dünyanın değişimini izlemesi gerekmediğinden ve zaman önemli olmadığından statik çevrede çalışmak daha kolaydır. Eğer çevre zamanla değişmiyorsa ama ajanın performansı zamana göre değişiyorsa çevre yarı dinamik (semidynamic) olarak adlandırılır.

- **Ayrık / Sürekli** (Discrete vs. continuous)

Eğer sınırlı sayıda farklı ve iyi tanımlanmış algı ve eylemler var ise çevre ayrıktır. Satranç ayrıktır çünkü her hamlede olası hareketler sabit sayıdadır. Araba sürme süreklidir, çünkü arabanın ve diğer araçların pozisyonu ve hızı sonsuz sayıda sürekli değerler alabilirler.

Aşağıdaki tabloda bazı çevreler ve karakteristikleri verilmiştir.

Çevre	Erişilebilir	Deterministik	Bölümlü	Statik	Ayrık
Santranç(saatle)	Evet	Evet	Hayır	Yarı	Evet
Satranç (saatsiz)	Evet	Evet	Hayır	Evet	Evet
Poker	Hayır	Hayır	Hayır	Evet	Evet
Tavla	Evet	Hayır	Hayır	Evet	Evet
Araba sürme	Hayır	Hayır	Hayır	Hayır	Hayır
Tıbbi teşhis	Hayır	Hayır	Hayır	Hayır	Hayır
Görüntü analiz	Evet	Evet	Evet	Yarı	Hayır
Parça toplama robotu	Hayır	Hayır	Evet	Hayır	Hayır
Rafineri kontrolörü	Hayır	Hayır	Hayır	Hayır	Hayır
İnteraktif İng.öğretici	Hayır	Hayır	Hayır	Hayır	Evet

Çevre Programları

Ajan programlarını test etmek için çevre simülatörü kullanılır. Simülatörler bir veya daha fazla ajanı giriş olarak alır, her bir ajana doğru algıları ileterek geriye eylemlerini alır. Simülatör ajanların eylemine bağlı olarak çevreyi yeniler. Yenileme işleminde ajan eylemine ek olarak bazı dinamik özellikler de eklenebilir. Bu nedenle çevre başlangıç durumu ve yenileme fonksiyonu ile tanımlanır. Aşağıda çevre simülatör programı görülmektedir.

```
Procedure Run-Çevre (durum, yenile-fn, ajanlar,
kesme)
  girişler: durum,  çevrenin ilk durumu
            yenile-fn,  çevreyi yenileme fonksiyonu
            ajanlar,    ajan kümesi
            kesme,      programdan çıkış testi
repeat
  for her ajan do
        Algı[ajan] ← Algı-Al(ajan,durum)
for her ajan do
        Eylem[ajan] ← Program[ajan](Algı[ajan])
durum ← Yenile-fn(eylemler, ajanlar, durum)
until kesme(durum)
```

Simülatör programına her bir ajanın performansını değerlendirecek performans fonksiyonu da eklenebilir.

BÖLÜM 3
ARAMA İLE PROBLEM ÇÖZME

Temel refleks ajan daha sonraki durumları göz önüne almaz. Refleks ajanın eylemi o anki algılamalarının bir sonucudur. Üstelik ne eylemlerinin sonucu hakkında bir bilgisi ne de amacını bilmektedir. Bu bölümde amaç tabanlı ajanın bir türü olan problem çözme ajanı (PÇA) tanımlanacaktır. PÇA istenen duruma erişmek için ne yapmak gerektiğine eylem sırasını bularak karar verir.

3.1 PROBLEM ÇÖZME AJANI - PÇA

Çevre durum değiştirirken zeki ajanların performans ölçüsünü maksimize edecek şekilde davrandıkları kabul edilir. Bu tanıma bağlı olarak başarılı ajan tasarımı güçtür. Ajana bir amaç verilerek tasarım basitleştirilebilir. İzmit'ten İzmir'e gitmek bir amaç olarak verilirse izlenebilecek çok sayıda yol bulunabilir. Yürüyerek gitmek de seçeneklerden biri olabilir. Ama İzmir'e varılması gereken zaman belirtilirse ajan başarısızlıkla sonuçlanacak yürüme seçeneğini dikkate almayacaktır. Amaçlar ajanın göz önüne alacağı seçenekleri sınırlayarak davranışlarını organize etmede yardımcı olur. Bu işlem için amacın formülize edilmesi gerekir. ***Amaç formülasyonu*** problem çözmede ilk adımdır.

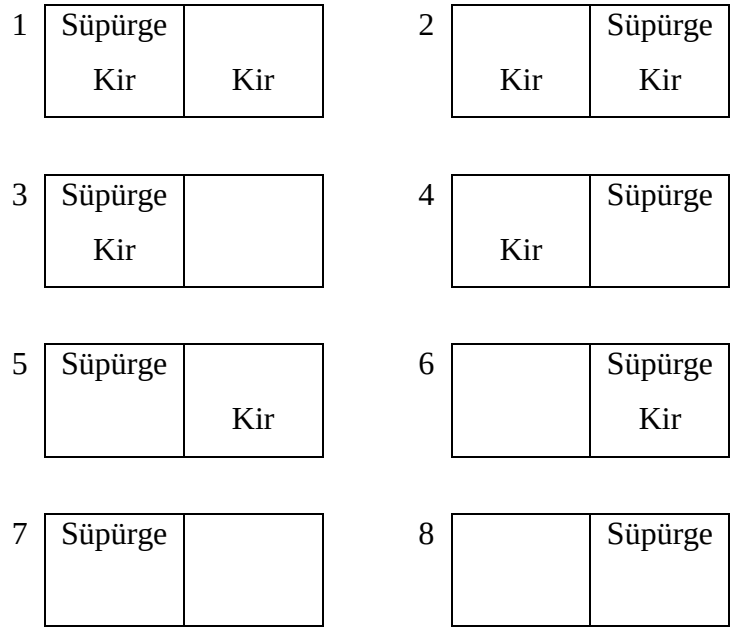
Amaç bir durum olarak düşünülebilir. Eylemler ise durumlar arasında geçiş yapmayı sağlar. Bir şehirden diğerine gitmek için 100m ileri git, sağa dön, 50m ilerle şeklinde çok ayrıntılı eylemleri dikkate alırsak park yerinden dışarı çıkamayız. Ajanın eylemini bir şehirden diğerine gitmek şeklinde üst seviyede düşünürsek, durumlar herhangi bir şehirde bulunmak olacaktır. ***Problem formülasyonu*** istenen amaca erişmek için hangi durum ve eylemlerin göz önüne alınacağını belirleme işlemidir.

Herhangi bir durumdan istenilen duruma varmak için olası eylem sıralarını deneyerek en iyi olanını seçme işlemine ***arama*** (searching) denir. Arama algoritmaları problemi giriş olarak alır ve çözümü eylem sırası olarak verir. Çözüm bulunduktan sonra önerilen eylemler icra edilir (icra aşaması). Böylece ajan için "***formülize etme, arama ve icra***" şeklinde bir tasarım elde edilir.

3.2 PROBLEMLERİN FORMÜLİZE EDİLMESİ

Bilgi ve Problem Tipleri

Çevre olarak temizlik dünyasını ele alalım. Bu dünyada yalnız iki yer vardır. Herbir yerde kir olabilir veya olmayabilir. Ajan iki yerin birinde bulunabilir. Aşağıdaki şekilde görüldüğü gibi 8 farklı dünya durumu vardır.



Basitleştirilmiş temizlik dünyasında 8 olası durum

Temizlik dünyasında ajan 3 farklı eylemde bulunabilir: ***sağ, sol*** ve ***em***. Şimdilik emmenin %100 başarılı olduğunu kabul edelim. Amaç bütün kiri temizlemektir. Yani amaç {7,8} durum kümesine eşittir.

İlk olarak ajanın sensorları ile hangi durumda olduğunu tespit edebildiğini kabul edelim (erişimli dünya). Aynı zamanda ajan eylemlerinin ne sonuç vereceğini de kabul edelim. Bu durumda ajan izleyeceği herhangi bir eylem sırası sonucunda hangi duruma erişeceğini hesaplayabilir. Örneğin başlangıçta 5. Durumda ise [sağ, em] eylem sırasının amaç durumuna getireceğini hesaplayabilir. Bu en basit durumdur ve ***tek durumlu problem*** (single-state problem) olarak adlandırılır.

İkinci olarak, ajanın eylemlerinin sonucunu bildiğini ama bulunduğu durumu tespit edemediğini kabul edelim. Ajan başlangıç durumunun {1,2,3,4,55,6,7,8} durum kümesindeki herhangi bir durum olduğunu bilir. Bu şartlar altında bile ajan çalışabilir. Çünkü ***sağ*** eyleminin sonucunda {2,4,6,8} durumlarından birine erişeceğini hesaplayabilir. Aslında başlangıç durrumu ne olursa olsun [sağ, em, sol, em] eylemleriyle ajan amaç duurumuna erişir. Özetle, dünya tam olarak erişimli değil ise ajan bir durum yerine içinde bulunabileceği durumları tahmin edebilmelidir. Bu ***çok durumlu problem*** (multiple-state problem) olarak adlandırılır.

Ajanın pozisyon ve kir sensörü olduğunu ama diğer karedeki kiri algılayamadığını kabul edelim. Sensörün {1,3} durumlarından birinde buluunulduğunu söylediğini kabul edelim. Ajan eylem sırasını [em, sağ, em] şeklinde formülize edebilir. Emme eylemiyle {5,7} durumlarından birine geçilir. Sağa hareket edildiğinde {6,8} durumlarından birine geçilir. Gerçekte 6 durumunda ise eylem sırası başarılı olacak 8 durumunda ise başarısız olacaktır. Bu nedenle problemin çözümünü garanti eden sabir bir eylem sırası yoktur.

Ajanın {1,3} durumunun birinden başlayarak problemi çözmesinin yolu vardır: ***em***, ***sağ***a git, **eğer** kir var **ise** ***em***. Yani problemin çözülmesi ***icra aşamasında algılama*** gerektirir. Ajanın bir eylem yerine tüm eylem ağacını hesaplaması gerektiğine dikkat ediniz. Genel olarak, ağacın her dalı doğabilecek koşula bağlıdır. Bu tip problemler koşullu problem (contingency problem) olarak adlandırılır. Gerçek dünyadaki bir çok problem koşullu problemdir, çünkü mutlak tahmin imkansızdır. Bu nedenle insanların çoğu yürürken veya araba kullanırken gözlerini açık tutar.

Tek durumlu ve çok durumlu problemler benzer arama teknikleri kullanılarak çözülürler. Koşullu problemler ise çok daha karmaşık algoritmalar gerektirir. Ajan garantili çözümü bulmadan eylem yapmak zorunda kalabilir. Ajan tam çözümü bulmadan eylem yapmaya başlar icra aşamasında önüne çıkan koşullara bağlı olarak yeni eylemde bulunur. Diğer deyişle arama ve icra ***dönüşümlü*** olarak yapılır.

Son olarak ajanın eylemlerinin etkileri hakkında bilgisi olmadığını kabul edelim. Bu bilinmeyen bir ülkede haritasız dolaşmaya benzer. Ajan gerçek dünyada eylemlerinin sonuçlarını ve farklı durumları yaşayarak öğrenecektir. Bu zeki ajanlar için en zor görevdir. Dikkatsiz bir ajan büyük tehlikelerle karşılaşır eğer ölmezse çevrenin haritasını çıkararak daha sonraki eylemleri için kullanabilir. Bu tip problemler ***keşif problemi*** (exploration) olarak adlandırılır.

İyi Tanımlanmış Problemler ve Çözümleri

Problem, ajanın ne yapacağına karar verirken kullanacağı bilgiler kümesidir. Problem tanımlamanın temel elemanları durumlar ve eylemlerdir:

- Ajanın içinde bulunduğunu bildiği duruma ***başlangıç durumu*** (initial state) denir.
- ***İşlem*** herhangi bir durumda yapılacak eylemle erişilecek durum şeklinde eylemi tanımlanamak için kullanır. Alternatif formülasyon S ***izleme fonksiyonu*** (successor function)dur. S(x); bir eylemle verilen x durumundan geçilebilecek durumların kümesini verir.

- Başlangıç durumundan eylem sırası ile erişilebilecek durumlara problemin ***durum uzayı*** (state space) denir. Durum uzayında ***yol*** (path) bir durumdan diğerine geçişi sağlayan eylem sırasıdır.
- ***Amaç testi***, ajanın istenen duruma erişilip erişilemediğini saptamada kullanacağı durum tanımıdır. Amaç durumu birden fazla ise herhangibirine erişilip erişilemediği test edilir. Amaç bazen soyut bir özellik de olabilir. Örneğin santrançta amaç şah mat durumuna erişmektir.
- Birden fazla yol izlenerek amaca erişilebiliyor ise daha kısa veya az maliyetli olan seçilir. ***Yol maliyet*** (path cost) fonksiyonu yola bir maliyet atar.

Başlangıç durumu, İşlem kümesi, amaç testi ve yol maliyet fonksiyonu problemi tanımlar. Problemi temsil etmek için bu elemanlardan oluşan bir veri tipi tanımlanabilir.

veri tipi : problem elemanları : Başlangıç-Durumu, İşlem-Kümesi, Amaç-Testi, Yol-Maliyet-Fn

Arama algoritmasının çıkışı başlangıç durumundan amaç testini sağlayan duruma olan yoldur yani ***çözüm*** dür.

Problem Çözme Performansının Ölçülmesi

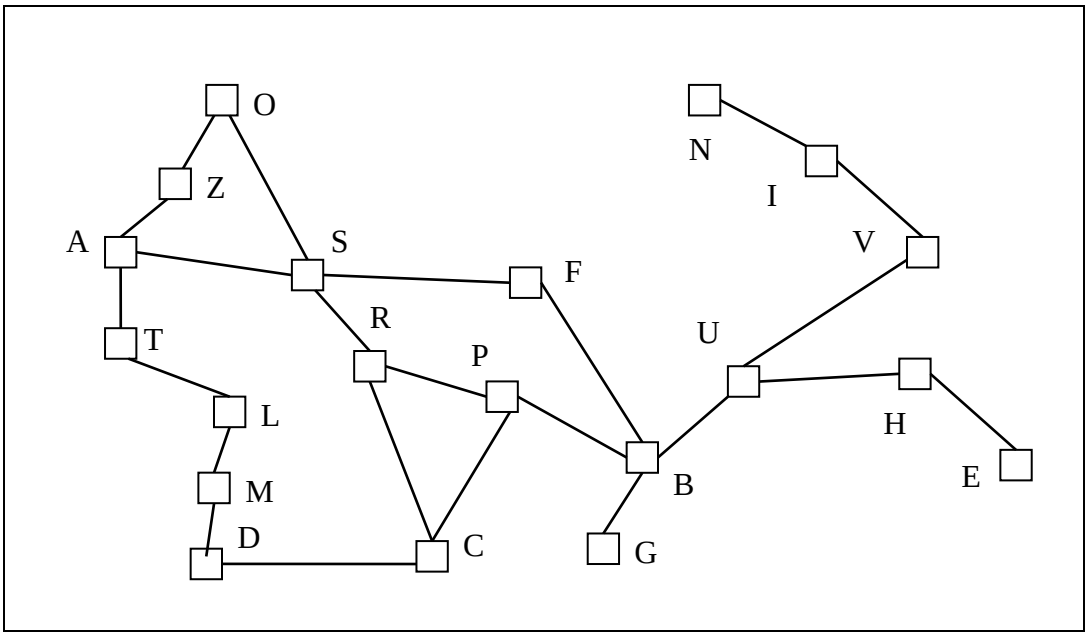
Aramanın etkinliği en az üç şekilde ölçülebilir:

- Hiç çözüm buldu mu ?
- Düşük yol maliyetli iyi bir çözüm mü ?
- Problemi çözmek için gerekli zaman ve belleğe ilişkin ***arama maliyeti*** (search cost) nedir?

Toplam maliyet (total cost), arama maliyeti ile yol maliyetinin toplamıdır.

İki şehir arasında izlenecek güzergahı tespit etmede yol maliyeti kilometre olabilir. Arama maliyeti ise yolun durumu veya yapılabilecek hız olabilir. Toplam maliyeti hesaplarken kilometre ile dakikayı toplamak gerebilir. Bu durumda ajan hangisine daha fazla ağırlık vereceğini tespit etmelidir.

Durum ve Eylemlerin Seçilmesi



Romanya'nın basitleştirilmiş yol haritası.

Yukarıdaki şekilde verilen haritada A'dan B'ye gitmek şeklinde basit bir problemle başlayalım. Uygun durum uzayı her durumun bir yeri gösterdiği 20 duruma sahiptir. Başlangıç durumu "A"dır. Amaç testi "B mi?" dir. İşlemler şehirler arasındaki yollarda araba sürmeye karşı gelir.

Bir çözüm yolu A→S→R→P→B şeklindedir. Çözüm için bir çok yol vardır. Bu çözümlerin en iyisinin hangisi olduğuna karar vermek için yol maliyet fonksiyonunun neyi ölçtüğünü bilmek gerekir. Bu mesafe veya beklenen zaman olabilir. Mevcut haritada bunların hiç biri tanımlanmadığı için adım sayısını maliyet fonksiyonu olarak alabiliriz. S ve F'den geçen yolun maliyeti 3 olduğu için en iyi çözümdür.

Burada, durum tanımında yol ve hava durumu sürücünün aç olup olmadığı gibi detaylar B'ye giden bir yol bulma problemi ile ilgili olmadığı için dikkate alınmamıştır. Gösterimden detayların çıkarılması işlemine ***soyutlama*** (abstraction) denir. Durum tanımı soyutlandığı gibi eylemler de soyutlanmalıdır. A'dan Z'ye gitmenin aracın yer dağiştirmesi, yakıt eksilmesi, kirlilik yaratması gibi bir çok etkisi vardır. Formülasyonda yalnız yer değiştirme dikkate alınacaktır. İyi bir soyutlama, soyut eylemlerin kolayca yapılabileceği şekilde mümkün olduğu kadar detayın çıkarılmasını gerektirir.

3.3 ÖRNEK PROBLEMLER

Problemler; değişik problem çözme yöntemlerini denemek amacıyla kullanılan ***oyuncak problemler*** (toy problems) ve çözümü daha güç olan ve insanlar için önemli olan ***gerçek dünya problemleri*** (real-world problems) olamak üzere ikiye ayrılabilir. Oyuncak problemlerin tanımı

iyi bir şekilde yapılabildiği için algoritmaların performansını karşılaştırmak için değişik araştırmacılar tarafından kullanılabilir. Gerçek dünya problemlerinin ise üzerinde anlaşılmış tek bir tanımı yoktur.

Oyuncak Problemler

8-puzzle

8-puzzle, 3*3'lük tahtada 1'den 8'e kadar numaralandırılmış taşlar ve bir boşluk yer alır. Taş bitişikteki boşluğa hareket edebilir.

5	4	
6	1	8
7	3	2

Başlangıç durumu

1	2	3
8		4
7	6	5

Amaç durum

4'ü boş yere götür yerine, boşluğu solundaki taşla değiştir demek daha az sayıda İşlem gerektirir ve daha mantıklıdır. Oyunun formülasyonu aşağıda verilmiştir:

- **Durumlar**: Durum tanımı 8 taşın herbirinin 9 kareden birinde bulunduğunu belirtir. Boşluğun yerini belirmek de faydalı olur.
- **İşlemler** : Boşluk sol, sağa, yukarı veya aşağı hareket edebilir.
- **Amaç testi** : Yukarıdaki şekilde gösterilen durum.
- **Yol maliyeti** : Her bir adımın maliyeti 1'dir. Yol maliyeti de yolun uzunluğuna eşittir.

8-Vezir Problemi

Amaç satranç tahtasına 8 veziri birbirini tehdit etmeyecek şekilde yerleştirmektir. Bu amaçla geliştirilmiş özel algoritmalar olmasına karşın n-Vezir problemleri arama algoritmalarının testinde ilgiyi korumaktadır. Temelde iki çeşit formülasyon vardır: artışsal (incremental) formülasyonda vezirler tek tek yerleştirilmektedir, tam durum (complete-state) formülasyonunda ise 8 vezir tahta üzerine yerleştirilip hareket ettirilmektedir. Sadece son durum dikkate alındığı için yol maliyeti dikkate alınmaz yalnız arama maliyetine bakılır:

- **Amaç testi:** Tahtada birbirini tehdit etmeyen 8-vezir.
- **Yol maliyeti:** sıfır.

Farklı durum ve işlemler de vardır:

- **Durumlar:** 0-8 vezirin herhangi bir düzenlemsi.

- **İşlemler**: Herhangi bir kareye vezir koymak.

Bu formülasyonda araştırılacak 64^8 olası sıra vardır. Eğer İşlem:

- **İşlemler**: Tehdit edilmeyen en soldaki boş kareye vezir koy.

Bu şekilde tehdit edilmeyen durumları tespit etmek kolaydır. Ama 7 vezirden sonra mümkün eylem kalmamaktadır. Bu nedenle arama işlemi başka bir seçeneği denemelidir. Hızlı bir hesap araştırılacak yalnız 2057 olası sıra olduğunu gösterir. *Doğru formülasyon arama uzayının boyutunu büyük ölçüde küçültür*.

Tehdit edilen veziri aynı kolonda başka bir kareye götürme işlemi de zaman alsa da sonunda çözümü bulur.

n-Vezir Problemi

Problem N adet veziri N*N'lik tahtaya hiç bir vezir diğerini tehdit etmeyecek şekilde yerleştirmektir. Vezir; yatay, düşey ve çapraz hareket edebilir. N=4 için aşağıdaki şekil çizilebilir:

		V	
V			
			V
	V		

Problemin çözümü [1,2,3,4] listesinin özel bir permütasyonu ile verilebilir. Örneğin yukarıdaki şekil [3,1,4,2] şeklinde yazılabilir. Burada vezirin; birinci satırda 3. sütuna, ikinci satırda 1. sütuna, üçüncü satırda 4. sütuna, dördüncü satırda 1. sütuna koyulduğu belirtilmektedir. Verilen permütasyonun çözüm olup olmadığını test etmek için iki veya fazla vezirin aynı köşegende olup olmadığını hesaplamak gerekir. Liste şeklinde gösterim iki veya daha fazla vezirin aynı satır veya sütun üzerinde olmasını engeller. Bir vezirin satır ve sütununun toplamı diğer vezirin satır ve sütunun toplamına eşit ise iki vezir aynı "/" köşegendedir. Bir vezirin satır ve sütununun farkı diğer vezirin satır ve sütunun farkına eşit ise iki vezir aynı "\" köşegendedir.

Bir listenin olası permütasyonları aşağıdaki tanım ile bulunabilir:

```
perm([X|Y],Z) :- perm(Y,W), remove(X,Z,W).
perm([],[]).
```

Remove(X,Z,W), X W'ya eklendiğinde Z'yi verir şeklinde düşünülebilir. Permütasyon tanımı aşağıdaki şekilde açıklanabilir:

- W Y'nin pernütasyonu ise ve X W'ya eklendiğinde Z elde ediliyorsa , Z [X|Y]'nin permütasyonundur.
- []'nin permütasyonu []'dir.

Kripto Aritmetik

Kripto aritmetik (cryptarithmetic) problemlerinde rakamların yerini harfler almıştır. Amaç toplama işleminin doğru olabilmesi için harflerin rakam karşılığını bulmaktır:

Çözüm:

FORTY	29786
TEN	850
+ TEN	+ 850
SIXTY	31486

F = 2, O = 9, R = 7 ...

Diğer örnekler:

SEND	CROSS
+MORE	+ROADS
MONEY	DANGER

En basit formülasyon aşağıdaki gibi olabilir:

- **Durumlar**: Bazı harflerin rakamlarla değiştirildiği kripto aritmetik bulmaca.
- **İşlemler:** Bir harfin bulunduğu tüm yerleri bulmacada daha önce kullanılmayan bir rakam ile değiştir.
- **Amaç testi:** Yalnız rakamlardan oluşan bulmaca doğru aritmetik sonucu veriyor.
- **Yol maliyeti:** Sıfır. Tüm çözümler aynı geçerlilikte.

Misyonerler ve Yamyamlar (Missionaries & cannibals)

Üç misyoner ve üç yamyam nehri geçmek istemektedir. Yalnız iki kişiyi taşıyabilecek bir bot vardır. Nehir yüzülerek geçilememektedir, bot en az bir kişi tarafından karşı kıyıya götürülebilmektedir. Nehrin her iki kıyısında yamyamların sayısı misyonerleri geçerlerse misyonerlerin yenilme tehlikesi vardır. Problem misyonerlerin yenilmeden herkesin karşı kıyıya geçmesidir

Bu problem, problem formülasyonuna analitik açıdan yaklaşan ilk makalenin konusu olduğu için çok ünlüdür (Amarel,1968). Problem çözme yöntemlerini uygulamadan önce gerçek yaşam problemlerinin soyutlanması gerekir.

Şöyle bir sahne hayal edelim: Arawaskan kabilesinin üç üyesi, Alice, Bob ve Charles timsah kaynayan Amazon nehrinin kenarında yeni arkadaşları Xavier, Yolanda ve Zelda ile durmaktadır. Bu esnada yağmur yağmakta ve kuşlar ötmektedir. Xavier, Yolanda ve Zelda misyonerdir ve yeni arkadaşlarına yalnız veya az sayıda yakalanırlarsa ne olacağı konusunda şüpheleri vardır. Hepsi kıyaya bağlı küçük teknenin nehrin iki yakasında geçişi sağladığı konusunda emin değillerdir.

Bu problemi formülize etmek için: ilk adım yağmur, timsah gibi çözüme etkisi olmayan tüm detayları ihmal etmektir. Bir sonraki adım doğru işlem kümesinin ne olduğuna karar vermektir. İşlemin bir veya iki kişiyi botla nehrin karşısına geçirmek olduğunu biliyoruz. Ama, botun içinde geçen zamanı gösteren bir duruma gereksinim olup olmadığına karar vermemiz gerekir. Botta iki kişi olabileceği için yamyamlar sayıca misyonerlerden fazla olamaz. Bu nedenle geçişin sadece uç noktaları önemlidir. Bireylerin de soyutlanması gerekir. Çözüm için misyonerlerin veya yamyamların isimleri gereksizdir. 3 misyoner ve 3 yamyamın herhangi bir permütasyonu aynı sonucu verir. Bu yaklaşımla problem aşağıdaki şekilde tanımlanabilir:

- Durumlar: Durum, misyonerlerin ve yamyamların sayısı ile botun nehrin hangi kenarında olduğunu gösteren 3 rakamdan oluşur. Başlangıç durumu (3,3,1) dir.
- İşlemler: Her durumda olası işlemler; bir misyoner, bir yamyam, iki misyoner, iki yamyam veya herbirinden birer kişiyi botla karşı kıyıya geçirmektir.
- Amaç testi: (0,0,0) durumuna erişmek.
- Yol maliyeti: Nehri geçiş sayısı.

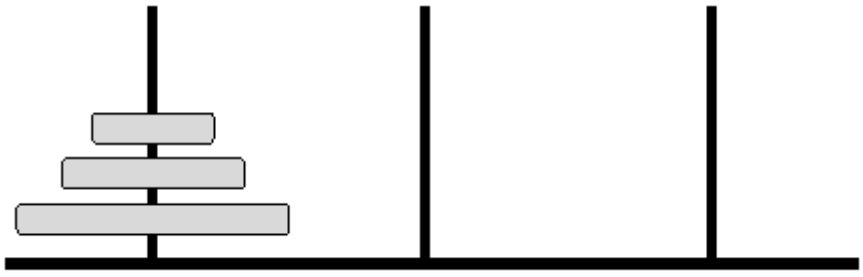
Durum uzayı oldukça küçük olduğu için bu problem bilgisayarla çözülebilir.

Temizlik Dünyası (Vacuum world)

Bu problem daha önceki bölümde anlatılmıştı.

Hanoi Kulesi (Tower of Hanoi)

Bu ünlü bilmecinin amacı, sol çubuktaki N adet diski ortadaki yardımcı çubuğu kullanarak sağdaki çubuğa taşımaktır. Hiç bir zaman büyük disk daha küçük olan diskin üzerine koyulamaz. Aşağıda N=3 için başlangıç durumu görülmektedir:



Maymun ve Muz problemi (Monkey & Bananas)

Aç bir maymun tavandan muzların asılı olduğu bir odadadır. Odada sandalye ve sopa vardır. Maymunun sandalye üzerine çıkarak sopayla muzları düşürüp yiyebilmesi için izleyeceği en iyi eylem sırası nedir.

Gerçek Dünya Problemleri

Rota Bulma

Rota bulma algoritmaları; bilgisayar ağları, otomatik seyahat tavsiye sistemleri, havayolu seyahat planlama sistemleri gibi değişik alanlarda kullanılmaktadır. Havaylu uygulaması çok karmaşıktır çünkü yol maliyeti çok karmaşıktır; para, yer kalitesi, zaman, uçak tipi, indirimler .. .Üstelik problemdeki eylemler de tamamen bilinen çıktıları vermez: uçuş geçikebilir, bağlantılar kaçırılabilir, sis veya acil durumlar gecikmeye neden olabilir.

Turlama ve gezen satıcı (travelling salesperson) problemi

Haritada verilen her şehri, B şehrinden başlayıp bitecek şekilde, en az bir kere ziyaret et şeklinde probleme bakalım. Bu rota bulma problemine benzemektedir, her ikisinde de iki şehir arasında seyahat edilmektedir. Ama bu problemde daha fazla bilginin kaydedilmesi gerekir. Ajanın yerine ek olarak, her bir durum ajanın ziyaret ettiği şehirleri de izlemelidir. Amaç testi 20 şehri de ziyaret ettikten sonra B' de olmaktır.

Gezen satıcı problemi (TSP) ünlü bir turlama problemidir. Amaç en kısa turu bulmaktır. Satıcıların gezilerini planlamaya ek olarak bu algoritmalar otomatik baskılı devre delme düzeneklerinin hareketlerinin planlanmasında da kullanılır.

VLSI çip tasarımı

Silikon çiplerin tasarımı en karmaşık mühendislik işlemlerinden biridir. VLSI çipinde milyonlarca kapı yer alır. Herbir kapının pozisyonu ve bağlantıları çipin başarılı çalışması için hayati önem taşır. İşlrmin her aşamasında CAD kullanılır. En güç işlerden ikisi hücrelerin yayılımı (cell layout) ve bağlantılarıdır (channel routing). Yayılımdan amaç, kullanılan alanı ve bağlantı uzunluklarını en aza indirerek hızı artırmaktır. Bağlantılar hücreler arasndaki boşlukları kullanarak yapılır. Bu problemler çok karmaşık olmasına rağmen çözmeye değer.

Robot Hareketi

Robot hareketi rota bulma probleminin genel halidir. Düz bir yüzeyde hareket eden dairesel robotda uzay iki boyutludur. Robotun kontrol edilecek kol ve bacakları var ise arama uzayı çok boyutlu olur. Arama uzayını sınırlandırmak için ileri teknikler kullanmak gerekir.

Problemin karmaşıklığına ek olarak gerçek robot sensorlardan ve motor kontrolünden gelen hatalarla da uğraşmalıdır.

Montaj Sırası (Assembly sequencing)

Karmaşık nesnelerin robotla birleştirilmesi ilk kez FREDDY robotuyla (Michie,1972) gerçekleştirilmiştir. İlerleme yavaş olmasına rağmen, elektrik motorlarının montajı ekonomiktir. Montaj problemlerinde, problem bir nesnenin parçalarını birleştirmek için gerekli sırayı bulmaktır. Eğer sıra yanlış seçilirse sonradan bazı parçaların yerleştirilmesi olanaksız olacaktır. Tekrar başa dönmek gerekecektir.

3.4 ÇÖZÜM İÇİN ARAMA

Şimdiye dek problemi nasıl tanımlayabileceğimizi ve çözümün ne olduğuna baktık. Geriye çözümün bulunması kalmıştır. Bu da durum uzayında yapılacak arama işlemidir.

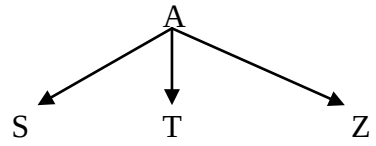
Eylem Sırasını Üretmek

Daha önce verilen harita üzerinde A'dan B'ye rota bulma problemini çözmek için başlangıç durumu A'dan yola çıkılır. İlk adım bunun amaç durum olup olmadığını test etmektir. Böylece "A'dan başlayarak A'ya gitmek" gibi hileli sorular da çözülebilir. Bu amaç durum olmadığı için diğer durumlara bakılması gerekmektedir. O anki durum üzerinde işlemler yaparak yeni durum kümeleri ***üretilir***. Bu işleme durumun ***açılması*** (expanding) denir. Bu şekilde 3 yeni duruma erişilir: S, T ve Z. Yeni durumlar A'ya bir adım uzaklıktadır. Eğer yalnız bir seçenek varsa o durum seçilir ve işleme devam edilir. Birden fazla seçenek varsa hangisinin dikkate alınacağına karar vermeliyiz.

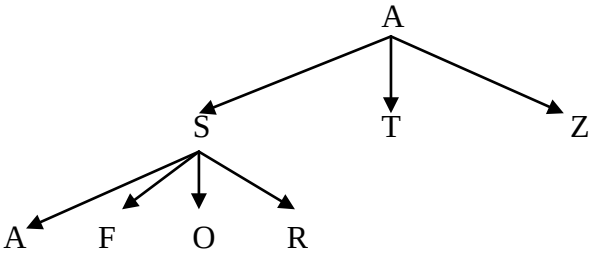
Aramanın temeli: seçeneklerden birini seçmek diğerlerini kenara koymak, eğer yapılan seçim istenen sonucu vermiyorsa diğer seçeneklerden birini denemektir. Seçme, test etme ve açma işlemi çözüm bulunana veya açılacak durum kalmayana dek devam eder. Hangi durumun önce açılacağı ***arama stratejisi*** ile saptanır.

a) Başlangıç durumu

b) A açıldıktan sonra



c) S açıldıktan sonra



A'dan B'ye rota bulmak için kısmi arama ağacı.

Arama işlemini durum uzayının üzerine yerleştirilmiş ***arama ağacı*** olarak düşünmek yararlı olur. Arama ağacının kökü başlangıç durumuna denk gelen ***arama düğümü***dür. Ağacın yaprak düğümleri izleyeni olmayan durumlardır. Bunlar ya henüz açılmamıştır ya da açılmış ama boş küme üretilmiştir.

Arama uzayı ve arama ağacını ayırtetmek gerekir. Rota bulma probleminde durum uzayında yalnız 20 durum vardır. Her şehir için bir durum vardır. Fakat durum uzayında sonsuz sayıda yol (path) vardır. Bu nedenle arama ağacında sonsuz sayıda düğüm vardır. Örneğin A→S→A dalı A→S→ A→S şeklinde devam eder.

Arama Ağaçları için Veri Yapıları

Düğümleri göstermenin birçok yolu vardır. Ama burada düğümü beş elemanlı bir veri yapısı olarak ele alacağız:

- Düğümün durum uzayındaki ***durum*** karşılığı,
- Bu düğüm (***parent node***)tarafından üretilen arama ağacındaki düğüm,
- Bu düğümü üretmek için yapılan ***işlem***,
- Kökten bu düğüme kadar olan yoldaki düğüm sayısı (***derinlik***-depth)
- Başlangıç durumundan bu düğüme olan yolun ***yol maliyeti***.

Veri tipi : Node Bileşenleri : State , Parent-Node, Operator, Depth, Path-Cost

Düğüm ve durum farklıdır. Düğüm; özel bir algoritma tarafından bir problem için üretilen arama ağacını gösteren kayıt veri yapısıdır. Durum dünyanın konfigürasyonunu temsil eder. Düğümde derinlik ve parent vardır, durumda ise yoktur.

Açılmayı bekleyen düğümlerin de (fringe, frontier) belirtilmesi gerekmektedir. En basit gösterim düğüm kümesi olabilir. Bu düğümlerin queue olarak gerçekleştirildiğini kabul edelim. Queue üzerinde aşağıdaki işlemler yapılır:

- Make-Queue(Element) :Verilen eleman ile queue oluşturur.
- Empry?(Queue) : Queue'da hiç eleman yok ise true değeri verir.
- Remove-Front(Queue) : Queue'nun önündeki elemanı çıkartır ve o elemanı verir.
- Queuing-Fn(Elements, Queue) : Queue'ya eleman kümesi ekler. Farklı fonksiyonlar farklı arama algoritmaları üretir.

1.5 ARAMA STRATEJİLERİ

Arama alanındaki işlerin büyük çoğunluğu bir problem için doğru arama stratejisini bulmaya gider. Stratejiler dört kritere göre değerlendirilebilir.

- ***Tamlık*** (Completeness): Eğer çözüm varsa strateji çözümü garanti ediyor mu?
- ***Zaman Karmaşıklığı*** (time complexity): Çözümü bulmak ne kadar zaman alıyor?
- ***Bellek Karmaşıklığı*** (space complexity): Aramayı gerçekleştirmek için ne kadar bellek gerekiyor?
- ***Optimallik*** (optimality): Birden fazla çözüm olduğu zaman strateji en iyi olanını buluyor mu?

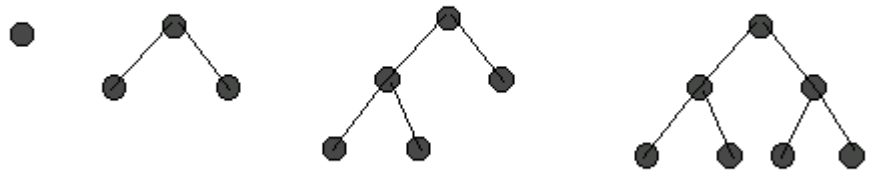
Bu bölümde bilgisiz arama(uninformed search) veya kör arama (blind search) başlığı altında incelenen 6 arama stratejisi incelenecektir. Bilgisiz terimi, mevcut durumdan amaç duruma erişmek için kaç adım gerektiği veya yol maliyeti hakkında bilgi olmadığını belirtir.

Rota bulma probleminde A başlangıç durumunda A,S ve Z durumlarına giden üç eylem vardır. Bilgisiz aramada bu üç yeni durum arasında bir fark yoktur ama daha zeki bir ajan B amaç durumunun A' nın güney batısında olduğu ve bu yönde yalnız S olduğu için S' ye gider. Bu gibi ek bilgileri dikkate alan arama stratejilerine bilgili arama (informed search) veya arama(heuristic search) stratejileri denir.

Önce Genişlik (Breadth-First) Arama

Bu yöntemde önce kök düğümü açılır. Sonra, kök düğümü tarafından üretilen düğümler açılır. Daha sonra bunların izleyenleri açılır. Genel olarak, arama ağacında d derinliğindeki düğümler d+1 derinliğindeki düğümlerden önce açılır. Genişlik öncelikli arama sısrasıyla 1 uzunluğunda, 2 uzunluğunda, .. , tüm yolları dikkate aldığından çok sistematik bir yöntemdir. Eğer bir çözüm var ise bu yöntem çözümün bulunmasını garanti eder(tamlık). Birden fazla

çözüm var ise en sığ olanı yani derinliği en az olanı verir (optimallik). Aşağıdaki şekilde önce genişlik aramalı ağacın gelişimi görülmektedir.



Önce genişlik aramada 0, 1, 2 ve 3 düğümlerinin açılması

Yöntemin zaman ve bellek karmaşıklığını incelemek için her durumun b yeni duruma açıldığını kabul edelim. Yani dallanma faktörü (branching factor) b olsun. Arama ağacının kökü ilk seviyede b adet düğüm üretir, bu düğümlerin her biri de b tane yeni düğüm üretir. İkinci seviyede b^2 , üçüncü seviyede b^3, .., düğüm üretilir. Yol uzunluğu d olan bir problemin çözümü bulmadan önce açılacak düğüm sayısı aşağıdaki şekilde yazılabilir:

$1 + b + b^2 + b^3 + ... + b^d$

Önce genişlik aramada dallanma faktörü b=10 için değişik derinliklerde zaman ve bellek karmaşıklığı aşağıdaki tabloda görülmektedir.

Derinlik	Düğümler	Zaman		Bellek	
0	1	1	msn	100	byte
2	111	0.1	sn	11	Kbyte
4	11111	11	sn	1	Mbyte
6	10^6	18	dk.	111	Mbyte
8	10^8	31	saat	11	Gbyte
12	10^{12}	35	yıl	111	Terabyte
14	10^{14}	3500	yıl	11111	Terabyte

Tabloda dallanma faktörü b=10, 1000 düğüm/sn, 100 byte/düğüm alınmıştır.

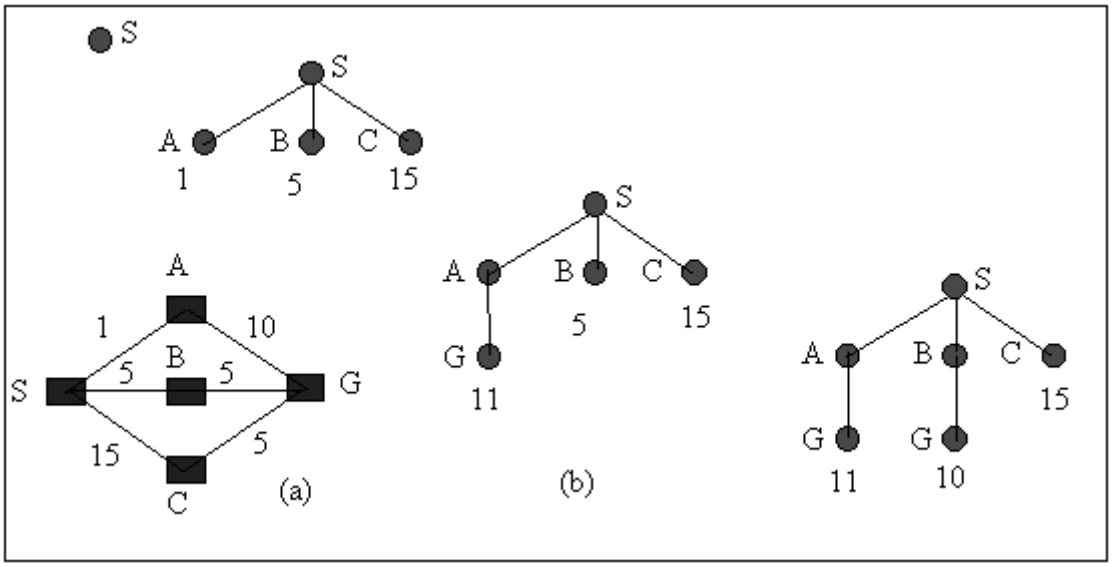
Bu yöntemde açılan tüm düğümler bellekte saklandığı için bellek karmaşıklığı zaman karmaşıklığı gibidir. Tabloya baktığımız zaman genişliğine arama yönteminde bellek geeksinimi icra süresinden daha büyük bir problem ortaya çıkarmaktadır. 6. Seviyedeki sonucu bulmak için 18dakika beklenebilir ama 111 Mbyte RAM biraz zor bulunur.

Zaman gereksinimi de önemli bir faktördür. 14. Derinlikteki bir çözüm için 3500 yıl! beklemek gerekiyor. 10 yıl sonra 100 kat daha hızlı bir bilgisayar alırsanız sonucu 35 yılda bulabilirsiniz.

Uniform Maliyetli Arama

Genişliğine aramada en sığ amaç durum elde edilir. Ama çözüm en düşük maliyetli yol olmayabilir.Uniform maliyetli aramada en düşük derinlikteki düğüm yerine açılmayı bekleyen düğümlerden en düşük maliyetli olanı açılır. Maliyet fonksiyonu g(n) ve derinlik Derinlik(n) olarak tanımlanırsa g(n)=Derinlik(n) için Uniform maliyetli arama genişliğine aramaya eşit olur.

Eğer daha az maliyetli yollar varsa bu yollar daha önce açıldığı için ilk bulunan çözüm en düşük maliyetli çözüm olur. Aşağıdaki şekilde verilen S'den G'ye rota bulma problemini göz önüne alalım:



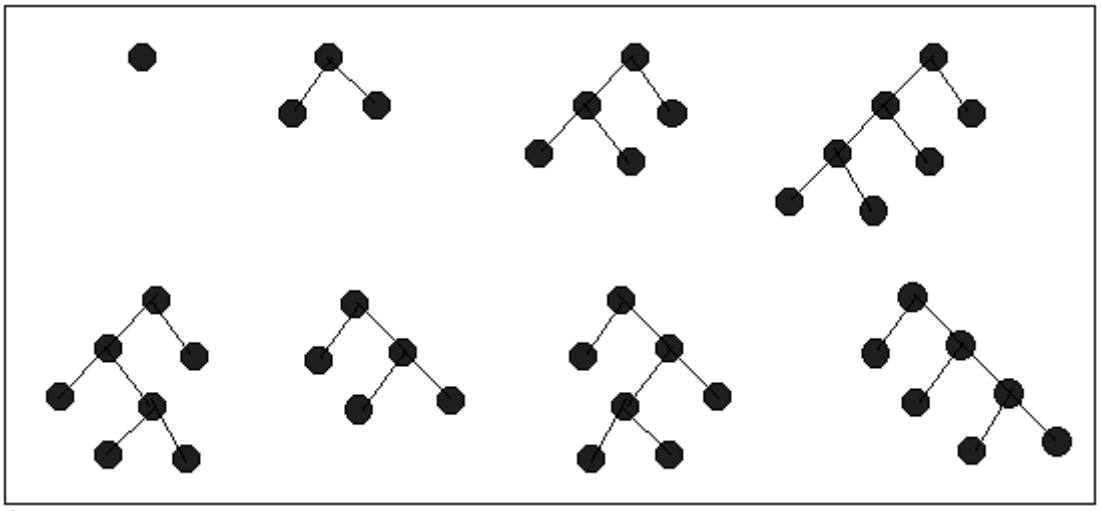
Rota bulma problemi.

a) Her işlemin maliyetini gösteren durum uzayı,
b) Aramanın ilerleyişi.

Bu yöntemde; başlangıç durumu açılarak A, B ve C yolları sağlanır. En ucuz A olduğu için A açılarak SAG çözümü elde edilir. Maliyeti 11 olduğu için algoritma bunu çözüm olarak kabul etmez. Bir sonraki adımda SB açılarak SBG çözümü bulunur. Ağaçta daha düşük maliyetli bir yol kalmadığı için SBG problemin çözümü olarak geriye verilir. Yol maliyeti yol arttıkça azalmıyor ise Uniform maliyetli arama en ucuz çözümü verir.

Derinlik Öncelikli Arama

Derinlik öncelikli aramada daima ağacın en derin düğümlerinden biri açılır. Eğer amaçlanmayan düğüme erişilmiş ise veya açılacak düğüm kalmamış iseaçma işlemine daha sığ seviyelerden devam edilir. Derinlik öncelikli aramanın ilerleyişi aşağıdaki şekilde görülmektedir.



İkili arama ağacı için derinliğine arama ağaçları.

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 sığ çözümün bulunduğu derinliktir. 12. Derinlik için genişliğine aramada 111 terabyte gerekirken derinliğine aramada 12 kilobyte yeterlidir. 10 milyarda biri kadar bellek!

Derinliğine aramanın zaman karmaşıklığı $O(b^m)$ dir. Birden fazla çözümü olan problemlerde derinliğine arama genişliğine aramadan daha iyi sonuç verir. Sonsuz döngüye girme, fazla derine gitme veya daha sığ çözümleri bulamama gibi problemler nedeniyle derinliğine arama tam veya optimal değildir.

Derinlik Sınırlı Arama

Derinliğine aramada karşılaşılan problemi gidermek için yolun maksimum derinliği belirtilerek derinlik sınırlı arama yapılabilir. Örneğin 20 şehrin bulunduğu haritada bir şehirden diğerine gitmek için yapılacak bir aramada maksimum derinlik 19 olarak verilebilir. Yeni işlem formu "eğer A'da iseniz ve seyahat ettiğiniz şehir sayısı 19'dan küçük ise yol uzunluğu bir fazla olan B şehrinde yeni bir durum oluştur" şeklinde ifade edilebilir. Bu yeni işlem kümesiyle çözüm var ise bulunması garanti edilmiş olur. Ama en kısa olanın bulunduğu garanti edilemez. Sonuç olarak derinlik sınırlı arama tamdır ama optimal değildir. Eğer derinlik az seçilirse tamlık da tehlikeye girer. Derinliğine aramada olduğu gibi derinlik sınırlamalı aramada da zaman karmaşıklığı $O(b^l)$, bellek karmaşıklığı $O(bl)$'dir. Burada *l* derinlik sınırıdır.

İteratif Derinleşerek Arama

Derinlik sınırlı aramanın en zor kısmı uygun derinliğin tespitidir. Örneğin harita probleminde derinlik sınırı 19 olarak alınmıştı ama harita incelendiğinde bir şehirden diğerine en

fazla 9 adımda gidilebildiği görülmektedir. Bu sayı durum uzayının ***çapı*** olarak bilinir ve daha iyi derinlik sınırı verir. Ama bir çok problemde problem çözülmeden iyi derinlik sınırının ne olduğu bilinemez.

İteratif derinleşmede en iyi derinliğin seçimi bir kenara koyularak olası tüm derinlik sınırları denenir: önce 1, sonra 2, 3 .. şeklinde. Aslında iteratif derinleşme, derinliğne arama ve genişliğine aramanın faydalarını birleştirir. Genişliğine arama gibi optimal ve tamdır, derinliğine arama gibi az bellek gerektirir.

İteratif derinleşmede köke yakın durumlar çok kez açılırlar. Bu kayıp olarak görülse de derinlik sınırlı arama ile fazla farklı değildir. Örneğin b=10 ve d=5 için derinlik sınırlı arama açma sayısı

$1+b + b^2 + b^3 + ...+ b^d = 1+10+100+1000+10000+100000= 111.111$

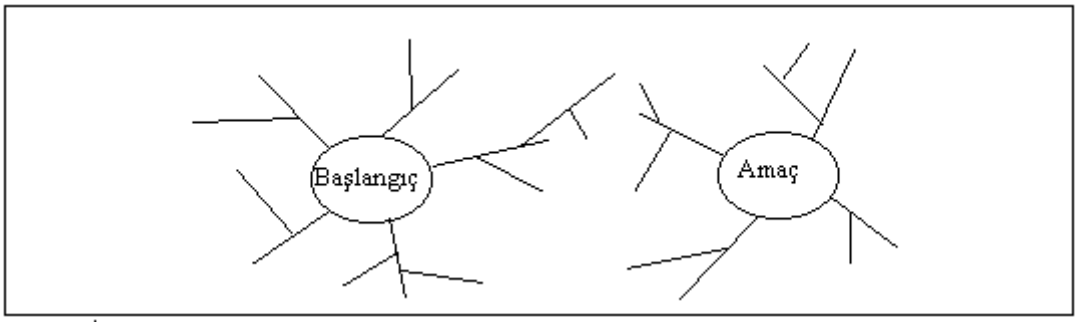
iken derinliğine aramada

$(d+1)\ 1 + (d)\ b +(d-1)b^2+...+ 2b^{d-1}+ 1\ b^d = 6+50+400+3000+20000+100000=123.456$

İteratif derinleşme algoritması genişliğine veya derinliğine aramadan yalnız %11 daha fazla düğüm açar. Bu nedenle iteratif derinleşmenin zaman karmaşıklığı $O(b^d)$, bellek karmaşıklığı ise O(bn)'dir. Arama uzayı büyük olduğunda ve çözümün derinliği bilinmediği zaman iteratif derinleşme algoritması kullanılır.

İki yönlü Arama

İki yönlü aramanın aramada, arama işlemine başlangıç durumu ve amaç durumundan aynı anda başlanır. İki arama ortada karşılaştığı zaman işlem biter. Dallanma faktörü her iki yönde de *b* ise iki yönlü aramada karmaşıklık açısından büyük fark olur. Eğer çözüm d derinliğinde ise çözüm $O(2b^{d/2}) = O(b^{d/2})$ adımda bulunacaktır. b=10 ve d=6 için genişliğine aramada 1.111.111 düğüm üretilirken iki yönlü aramada derinlik iki yönde de 3 olacağı için 2.222 düğüm üretilir. Aşağıdaki şekilde iki yönlü genişliğine arama görülmektedir.



İki yönlü genişliğine arama.

Algoritma gerçekleştirilmeden önce bazı soruların cevaplandırılması gerekmektedir:

- Amaçtan geriye doğru aramak ne demektir? Amaç düğümden başlayarak önceki düğümleri (predecessor) sırayla üretmektir.
- İşlemlerin tersi mümkün olduğunda, önceki düğümler kümesi ve sonraki düğümler (successor) kümesi aynı olacaktır. Bazı problemlerde öncekileri bulmak zor olabilir.
- Birden fazla amaç durum var ise ne yapılabilir? Amaç durum yerine amaç durum kümesine aynı işlemler uygulanabilir. Amaç durumların tespiti güç olabilir. Örneğin satrançta şahmat amacını üretecek durumlar nelerdir?
- Yeni oluşturulacak bir düğümün arama ağacının diğer yanında yer alıp almadığını kontrol etmenin etkin bir yolu olmalıdır.
- Her iki yarıda ne çeşit aramanın yapılacağına karar vermek gerekir. Şekilde gösterilen iki tarafta da genişliğine arama en iyi seçim midir?

Arama Yöntemlerinin Karşılaştırılması

Verilen dört kritere göre anlatılan arama yöntemlerinin karşılaştırılması aşağıdaki tabloda görülmektedir:

Kriter	*Genişlik öncelikli*	*Uniform Maliyet*	*Derinlik Öncelikli*	*Derinlik Sınırlı*	*İteratif Derinlik*	*İki Yönlü*
Zaman	b^d	b^d	b^m	b^l	b^d	$b^{d/2}$
Bellek	b^d	b^d	bm	bl	bd	$b^{d/2}$
Optimal?	Evet	Evet	Hayır	Hayır	Evet	Evet
Tam?	Evet	Evet	Hayır	$l \geq d$ ise evet	Evet	Evet

Arama yöntemlerinin değerlendirilmesi. Burada; b dallanma faktörü, d çözümün derinliği, m arama ağacının maksimum derinliği, l derinlik sınırıdır.

BÖLÜM 4
BİLGİLİ ARAMA

Bilgili arama (informed search) durum uzayı hakkındaki bilgileri kullanarak aramanın karanlıkta rastgele dolaşma şeklinde yapılmasını engeller. Bilgili arama yöntemleri çözümü daha etkin olarak bulurlar.

4.1 En iyi Öncelikli Arama

Hangi düğümün öncelikle açılacağını saptamada kullanılabilecek bilgi ***değerlerndirme fonksiyonu*** (evaluation function) ile elde edilebilir. Düğümlerin değerlendirme fonksiyonu kullanarak sıralanması ve açma işlemine en iyi değeri veren düğümden başlanmasına ***en iyi öncelikli arama*** (best-first search) yöntemi denir. En iyi olan değerlendirme fonksiyonuna göre tespit edilmektedir. Değerlendirme fonksiyonu hatalı sonuçlar da verebilir. Aslında bu arama en iyi sanılanın öncelikle açılmasıdır. Amaç düşük maliyetli çözümler bulmak olduğu için bu algoritmalar maliyet için tahmini bir ölçü kullanarak onu minimize etmeye çalışır. Bu yöntemde iki yaklaşım kullanılmaktadır. Birincisi, amaca en yakın düğümü açmaya çalışır. İkincisi ise, en düşük maliyetli çözüm yolunda düğümü açmaya çalışır.

Greedy Arama

En iyi öncelikli aramanın en basit şeklinden birisi amaca erişmek için tahmini maliyetin minimize edilmesidir. Yani, durumu amaç duruma en yakın olduğu sanılan düğüm öncelikle açılır. Herhangi bir durumdan amaç duruma erişmenin maliyeti tahmin edilebilir ama kesin olarak saptanamaz. Maliyet tahminini hesaplayan fonksiyona bulma (heurestic) fonksiyon denir ve genellikle h ile belirtilir:

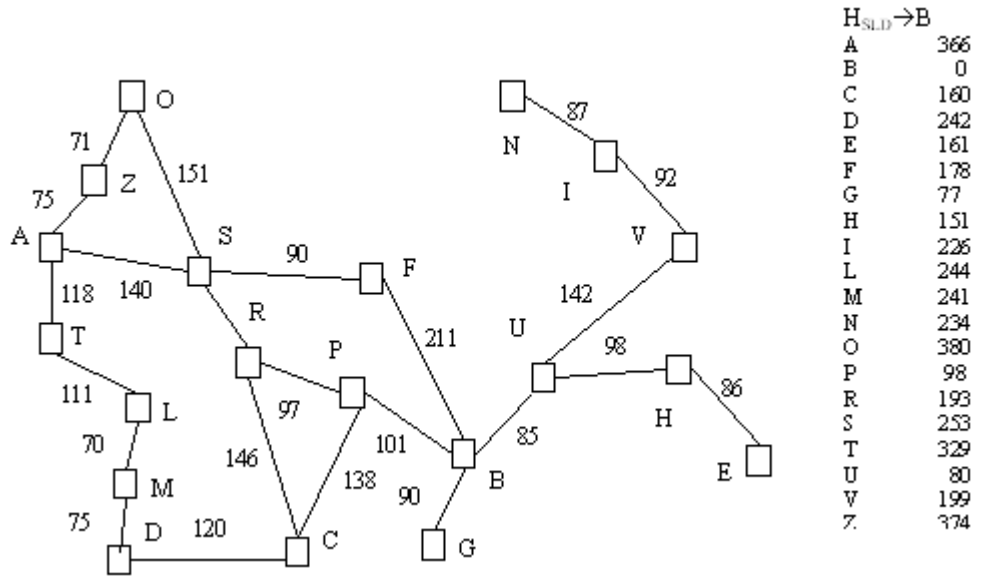
h(n) = n düğümündeki durumdan amaç duruma en ucuz yolun tahmini maliyeti.

Açılacak düğümü h'i kullanarak tespit eden en iyi öncelikli aramaya ***greedy arama*** adı verilir. H herhangi bir fonksiyon olabilir. n amaç ise h(n)=0 olması gerekir.

Rota bulma problemleri için iki nokta arasındaki direkt mesafe (straight-line distance, SLD) iyi bir bulma fonksiyonu olarak kullanılabilir. Yani

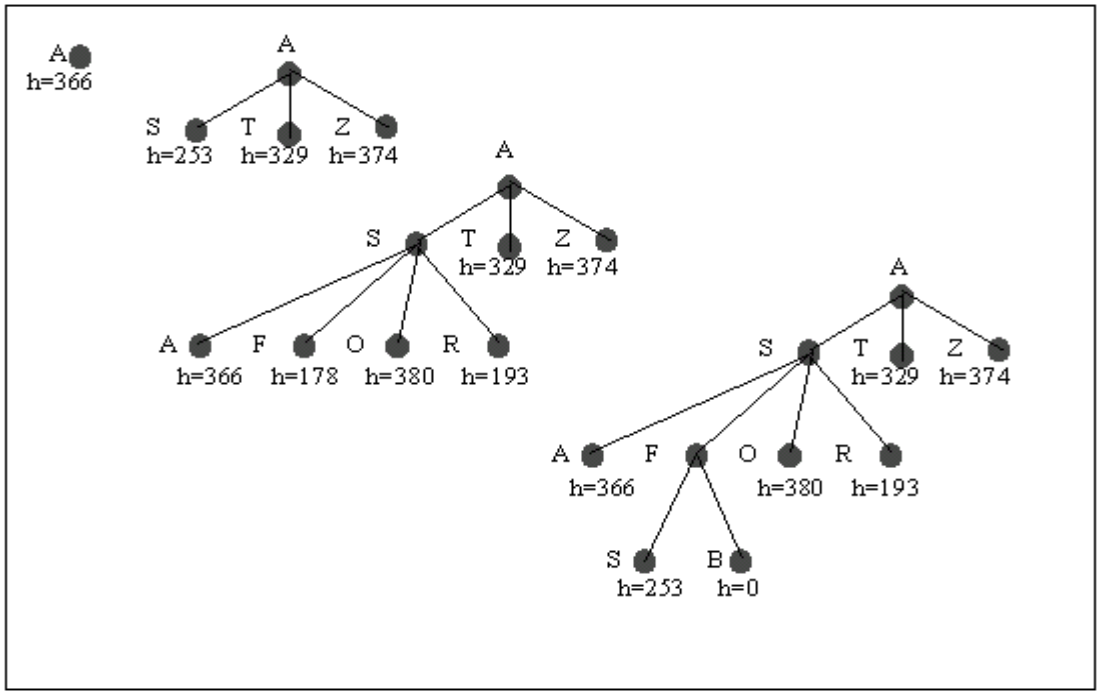
$h_{SLD}(n)$ = n ile amaç yer arasındaki direkt mesafe.

Eğer şehirlerin koordinatları biliniyor ise h_{SLD} değerleri hesaplanabilir. Bu şekilde ek bilgiler arama maliyetini azaltır. Daha önce vermiş olduğumuz haritada şehirler arası mesafe ve bu şehirlerle B arasındaki direkt mesafe, h, aşağıdaki şekilde görülmektedir.



$H_{SLD}\rightarrow B$	
A	366
B	0
C	160
D	242
E	161
F	178
G	77
H	151
I	226
L	244
M	241
N	234
O	380
P	98
R	193
S	253
T	329
U	80
V	199
Z	374

Yol uzunlukları ve B' ye direkt mesafeler.



B şehri için Greedy aramanın aşamaları .

Direkt mesafe fonksiyonu ile A' da açılacak ilk düğüm S olacaktır. Çünkü S B' ye Z veya T' en daha yakındır. Bir sonraki açılacak düğüm ise F olacaktır çünkü B' ye en yakın olan F' dir. F amaç durum olan B' i üretir. Bu problem için bulma fonksiyonu en az arama maliyetini sağlar. Çözüm yolunda olmayan düğümlerin hiç biri açılmamıştır. Ama tam olarak optimal değildir. Çünkü A→S→R→P→B yolu 32 km. daha kısadır. Bu yöntem amaca ulaşmak için en büyük adımı atmaya çalışır. Uzun vadeyi düşünmez.

I' dan F' ye gitme problemini gözönüne alalım. Greedy arama önce N' yi açacaktır ama bu çıkmaz yoldur. Çözüm önce V'ye gitmektir. Bu durumda gereksiz bir düğüm açılmış oldu. Eğer tekrarlanan durrumlara dikkatt edilmez ise arama N ve I arasında salınım yapar.

Greedy arama, derinliğine arama yöntemi gibi çalışır. Çıkmaz bir yolla karşılaşınca geri döner. Derinliğine aramaya benzer şekilde ne tam ne de optimaldir.

A* Arama - Toplam yol maliyetinin azaltılması

Greedy arama amaca erişmek için tahmini maliyeti minimize eder ve böylece arama maliyetini azaltır. Diğer yandan tam ve optimal olan uniform arama o ana kadar olan yolun maliyetini, g(n), azaltır. İki yöntemin birleştirilmesi ile daha iyi arama yapılabilir. Bu işlem iki değerlendirme fonksiyonunu toplayarak yapılabilir:

$f(n) = g(n) + h(n)$ → n' den geçen en ucuz çözümün tahmini maliyeti.

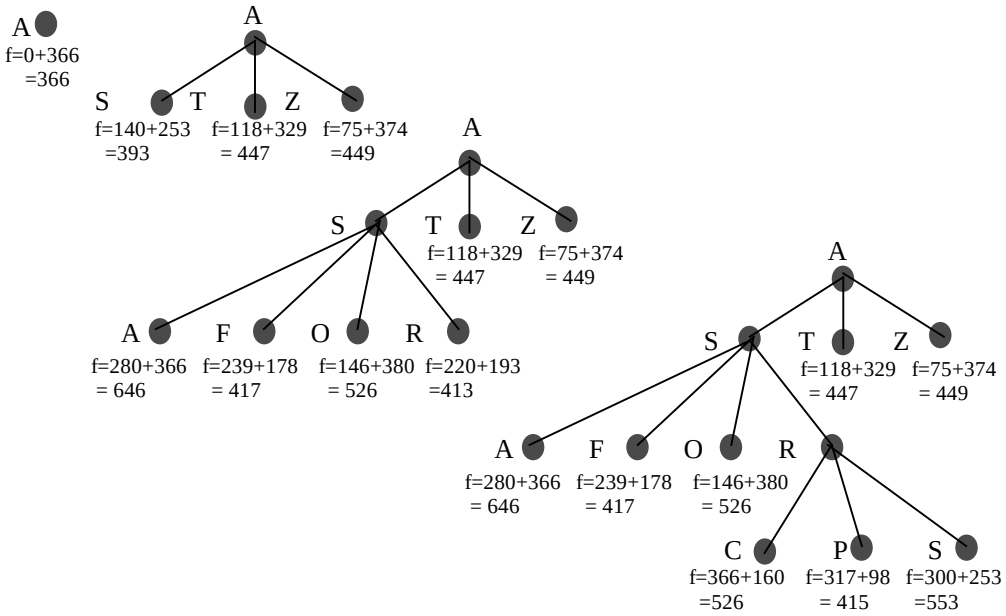
En ucuz çözümü bulmak için en düşük f değerli düğümü öncelikle denemek gerekir. h fonksiyonuna bir sınırlama getirerek yöntemin tam ve optimal olması sağlanabilir. Sınırlama, h fonksiyonun amaca erişmenin maliyetini asla fazla tahmin etmemesidir. Böyle h, kabul edilebilir bulma (***admissible hueristic***) olarak adlandırılır. Kabul edilebilir bulma fonksiyonu maliyeti gerçekte olduğundan az verdiği için iyimserdir. Bu iyimserlik f fonksiyonu için de geçerlidir. Eğer h kabul edilebilir ise f(n) n'den geçen en iyi çözümün gerçek maliyetini asla fazla tahmin etmez. En iyi öncelikli aramada f fonksiyonu değerlendirme fonksiyonu olarak kullanılıyorsa bu arama ***A* arama*** olarak isimlendirilir.

Aşağıda verilen arama ağacına bakıldığı zaman, kökten başlayan herhangi bir yol boyunca f maliyeti asla azalmamaktadır. Burada bulma fonksiyonu ***monotonik*** (monotonocity) özelliktedir. h, üçgen eşitsizliğine uyuyor ise monotoniktir. Örneğimizde direkt mesafe üçgen eşitsizliğine uyduğu için monotoniktir.

Eğer h monotonik değil ise küçük bir düzeltme ile monotik hale getirilebilir. n düğümü n' düğümünün parentı olsun. g(n)=3, ve h(n)=4 olduğunu kabul edersek f(n)= 7 olur. Yani n'den geçen çözümün maliyeti en az 7'dir. g(n')=4 ve h(n')=2 olduğunu kabul edersek f(n')=6 olur. Bu örnek monotonik değildir. Ama n' düğümünden geçen yol aynı zamanda n düğümünden de geçmesi gerektiği için 6 anlamsızdır. Çünkü gerçek maliyetin en az 7 olduğu bilinmektedir. Bu nedenle yeni bir düğüm üretildiği zaman f-maliyetinin parentın maliyetinden düşük olup olmadığı kontrol edilir eğer düşük ise parentın maliyeti kullanılır:

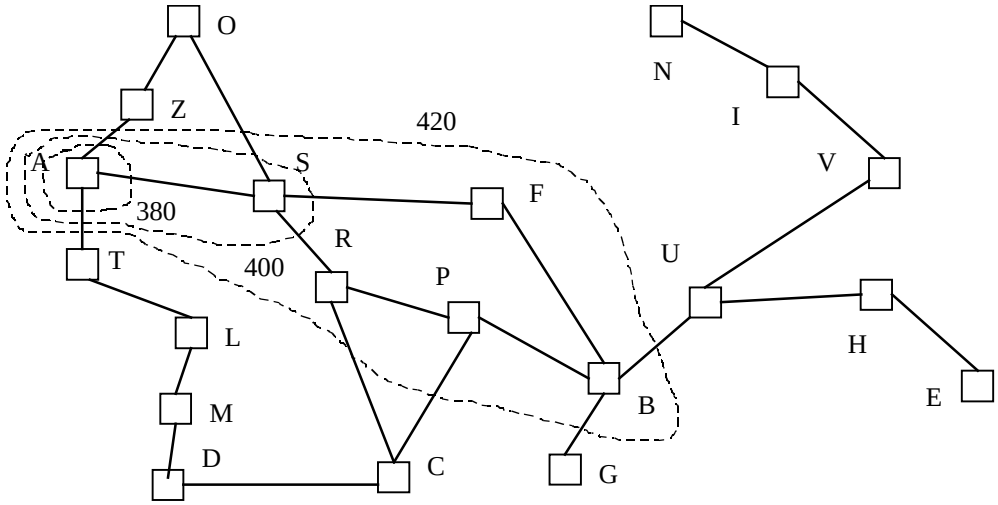
$f(n') = \max(f(n), g(n')+h(n'))$.

Bu eşitlik maksimum yol (pathmax) eşitliği olarak isimlendirilir. Bu eşitlik kullanılır ise kökten başlayarak her yolda f devamlı artar



A' dan B' ye gitmek için A* arama. f=g+h ve h= A' ya direkt mesafe.

Kökten başlayan herhangi bir yol boyunca f asla azalmıyor ise durum uzayında eş uzaklık eğrileri (contour) çizilebilir. Aşağıdaki şekilde 400 ile etiketlenen eğrinin içindeki tüm düğümlerde f(n) 400 den küçük veya eşittir.



A başlangıç durumundan f=380, f=400 ve f=420 eğrileri.

A* arama en düşük f değerli yaprak düğümü açtığı için düğümler bir merkez etrafında bant şeklinde açılır. Uniform maliyetli aramada h=0 olduğu için bantlar başlangıç durumu etrafında dairesel olacaktır.

4.2. BULMA FONKSİYONLARI (SEZGİSEL FONKSİYONLAR)

"Heuristic", Yunanca kökenlidir ve bulmak, keşfetmek anlamına gelmektedir. Ancak bugün sezgisel, herhangi bir şeyin bulunmasını garanti etmeyen bir "arama" (seeking) metodu olarak tanımlanmaktadır. Kombinatoryal optimizasyonda "sezgisel" terimi, bir tümel optimumun bulunmasını garanti eden metotların aksini ifade etmektedir. Kombinatoryal terimi, karar

değişkenlerinin kesikli olmasını yani problem çözümünün tamsayıların ya da diğer kesikli nesnelerin bir kümesi veya bir sırası olmasını ifade etmektedir.

Heuristic Tanımı: Sezgisel, iyi yani en iyi çözüme yakın çözümleri araştıran bir tekniktir ve bu işlemi fizibiliteyi ya da en iyi sonucu bulmayı garanti etmeksizin makul bir hesaplama maliyeti ile yapar.

Bir ağacı arama işlemi oldukça basit bir işlemdir ve bunu nasıl yapacağımıza dair çeşitli yöntemler türetmek elimizdedir. Örneğin, seyahat eden işadamı probleminde her bir yolu takip etmek ve bu sırada ne kadar yol kat ettiğimizi tutmak, bizi sonuca ulaştıracak tam ve kesin çözümdür. Ne yazık ki, seyahat eden işadamı probleminde ve yapay zekada ağaç aramasıyla çözülen daha birçok problemde böyle algoritmalar kuramsal olarak mümkünse de uygulamada kullanışsızdırlar. Örneğin, seyahat eden işadamı problemi için önerilen çözüm, (N-1)! deneme gerektirir. N, şehir sayısıdır. Eğer N=10 ise 3 milyonun üzerinde farklı yol vardır ve N arttıkça algoritmayı sonuçlandırmak oldukça uzun zaman alır, öyle ki bu zaman dilimi, belki de bir işadamının yaşayabileceğinden daha uzundur. Buna benzer problemlerde en iyi çözüm olmasa da ille de bir sonuç bulmak istiyorsak bazı ihtimalleri elemek zorundayız.

Bazen insanlar algoritmalar kullanırlar ama öyle görünüyor ki zekice problem çözmenin birçok aşaması, daha az kesin olan şeyler içerir. Eğer kendi kendinizin problemleri nasıl çözdüğünüzü inceleyecek olsaydınız yaptığınız şeyin tam bir algoritma yerine kuralların çalışacak gibi görünen seyreltilmiş bir derlemesini kullanmak olduğunu görürdünüz. Örneğin, satranç oynadığınızda tahtanın ortasının denetimi kuralını uygularsınız. Bu kural, sizin bir sonuca ulaşmanızı güvencelemez (kazanmak gibi) ama sizi ona yaklaştırır. Sonuca yaklaşılmasına yönelik olan bu kural, **sezgisel** (heuristic) olarak adlandırılır ve size ikinci sınıf bir algoritmaymış gibi gelebilir. Eğer güç bir problemle bir işlem gerçekleştirilecekse sezgisel kullanmak, genellikle tek çözüm yoludur. Bir sezgisel, problemin çözümünü güvencelemeyebilir ve bir çözümün olmadığını da söyleyemez ama sezgiseller, çok geniş bir alanda kullanılabilirler ve onlar kullanılarak bir çözüm bulunabilirse bu çözümü bulmak aynı problem için olan tam algoritmanınkinden daha kısa sürer. Gelecekte bilgisayar biliminin ve programlamanın problem çözmede sezgisel ve birleştirilmiş sezigisel-algoritmasal yaklaşımları daha verimli kullanacakları şüphe götürmez bir gerçektir.

Basit bir sezgisel ve aynı zamanda etkili bir örnek olan seyahat eden işadamı problemini tekrar ele alalım. Her aşamada en yakın şehre gidersek mantıken kısa bir yol bulmuş olacağız. Bu, bir sezgisel olarak "her zaman en yakın mesafeyi temsil eden düğüme git" şeklinde ifade edilebilir. Bu sezgiselin ürettiği yol pek tabi kötü değildir, (daha kötülerini göstermek oldukça

kolay) ancak bu yolun da en kısa yol olduğunu söylemek yanlış olur. Bu "en yakın şehir" sezgiselinin bulduğu yolun en iyi yoldan %20 daha uzun olduğunun delilleri vardır.

Bir önceki örnekte iki şehir arasındaki direkt mesafe bulma (heuristic) fonksiyonu olarak kullanılmıştı. Bu bölümde 8-puzzle problemi incelenecektir.

5	4	
6	1	8
7	3	2

Başlangıç durumu

1	2	3
8		4
7	6	5

Amaç durum

Problemin tipik çözümü 20 adımdır. Bu başlangıç durumuna göre değişebilir. Dallanma faktörü 3 kadardır (boşluk ortada olduğunda 4 olası hareket, köşelerde olduğunda 2 olası hareket ve kenarlarda 3 olası hareket vardır). Bunun anlamı 20 derinlikli aramada yaklaşık $3^{20} = 3.5*10^9$ durumdur. Tekrarlanan durumlar dikkate alınır ise 9 karenin 9! = 362.880 farklı düzenlemesi olacaktır. Durum oldukça azalmasına rağmen hala çok fazladır. Bu nedenle iyi bir bulma fonksiyonu gerekmektedir. Eğer en kısa çözümü bulmak istiyor isek bulma fonksiyonu amaç durum için gerekli adımları fazla tahmin etmemesi gerekir. Aşağıdaki fonksiyonlar bulma fonksiyonu olarak kullanılabilir:

- h_1= Yanlış pozisyondaki taşların sayısı. Yukarıdaki şekilde başlangıç durumunda 8 taşın hiçbiri doğru yerde olmadığı için h_1=8 dir. Yerinde olmayan taşlar en az bir kez hareket ettireleceği için h_1 kabul edilebilir bir bulma fonksiyonudur.
- h_2= Taşların amaç pozisyonundan uzaklıklarının toplamı. Taşlar çapraz hareket edemeyeceği için uzaklık yatay ve düşey mesafelerin toplamı olacaktır. Bu uzaklık bazen Manhattan uzaklığı olarak isimlendirilir. H_2 kabul edilebilir bir bulma fonksiyonudur. Çünkü herhangi bir hareket bir taşı amaca bir adım daha yaklaştırır. 8 taşın başlangıç durumundaki Manhattan mesafesi:

  h_2= 2+3+2+1+2+2+1+2 = 15 tir.

Bulma Fonksiyonunun Performans Üzerindeki Etkisi

Bulma kalitesini belirtmenin bir yolu ***etkin dallanma faktörü***dür (b^*). Eğer bir problem için A^* ile açılan düğümlerin sayısı N ve çözüm derinliği d ise b^* d derinliğindeki dengeli (uniform) bir ağacın N düğüm içermesi için gerekli dallanma faktörüdür:

$N = 1 + b^* + (b^*)^2 + ... + (b^*)^d$

Örneğin A^* 5 derinliğindeki bir çözümü 52 düğüm kullanarak buluyor ise etkin dallanma faktörü 1.92'dir. İyi tasarlanmış bulma fonksiyonu ile b^* 1'e yaklaşır.

8-bulmacası için yukarıda verilen h_1 ve h_2 bulma fonksiyonlarını karşılaştırmak için çözüm uzunluğu 2 ... 16 aralığında rastgele 100'er problem üretilerek A^* ve iteratif derinleşme(IDS) ile çözülmüştür. Çözümler aşağıdaki tabloda görülmektedir.

	Arama Maliyeti			Etkin Dallanma Faktörü		
D	**IDS**	**$A^*(h_1)$**	**$A^*(h_2)$**	**IDS**	**$A^*(h_1)$**	**$A^*(h_2)$**
2	10	6	6	2.45	1.79	1.79
4	112	13	12	2.87	1.48	1.45
6	680	20	18	2.73	1.34	1.30
8	6384	39	25	2.80	1.33	1.24
10	47127	93	39	2.79	1.38	1.22
12	374404	227	73	2.78	1.42	1.24
14	3473941	539	113	2.83	1.44	1.23
16	-	1301	211	-	1.45	1.25

h_2 daima h_1'den daha iyidir. Çünkü daha az düğüm açılmaktadır.

Bulma Fonksiyonlarının Çıkartılması

h_1 ve h_2 8 bulmacası için iyi birer bulma fonksiyonudur ve h_2 daha iyidir. Herhangi bir problem için bu fonksiyonlar nasıl bulunabilir, bilgisayar bulabilir mi?

Eğer 8 bulmacasında kurallar,taşlar yalnız bitişikteki boş kareye değil dolulara da geçebilecek şekilde gevşetilir ise h_1 ve h_2 en kısa çözüm için gerekli adımı doğru bir şekilde verir. İşlemlerdeki kısıtlamaların azaltıldığı problemlere gevşek problem (relaxed problem) adı verilir. Gevşek problemin çözümü orijinal problemin çözümü için iyi bir bulma fonksiyonu verir.

Eğer problemin tanımı formal bir dille yazılır ise gevşek problemler otomatik olarak oluşturulabilir. 8-bulmacası için aşağıdaki tanımı yapabiliriz:

Eğer A B' ye bitişik ise ve B boş ise taş A' dan B' ye hareket edebilir.

Bir veya daha fazla koşulu göz ardı ederek aşağıdaki gevşek problemleri yazabiliriz:

a) *Eğer A B' ye bitişik ise taş A' dan B' ye hareket edebilir.*

b) *Eğer B boş ise taş A' dan B' ye hareket edebilir.*

c) *Taş A' dan B' ye hareket edebilir.*

Bu yöntemi kullanan ABSOLVER programı (1993) problem tanımından bulma fonksiyonları çıkartabilmektedir. Bu program 8-bulmacası için mevcut fonksiyonlardan daha iyi bir fonksiyon üretmiştir. Rubik küp için de ilk faydalı fonksiyonu bulmuştur.

Eğer kabul edilebilir fonksiyonlar birden fazla ise maksimum değer vereni seçilir:

$h(n) = \max(h_1(n), ... , h_m(n))$.

4.3 BELLEK SINIRLI ARAMA

Zeki arama algoritmalarının bulunmasına karşın bazı problemlerin çözülmesi güçtür. Karşılaşılan güçlüklerden birisi bellek gereksinimidir. Kullanılabilecek bellek büyüklüğünü gözönüne alarak problemleri çözmeye çalışan iki algoritma bu kısımda incelenecektir: IDA* ve ISMA*.

İteratif Derinleşen A* Arama (IDA*)

Daha önceki bölümde iteratif derinleşme algoritmasının bellek gereksinimini azaltan yararlı bir algoritma olduğu görülmüştü. Benzer şekilde, A* arama iteratif A* arama (IDA*) şekline dönüştürülebilir. Bu algoritmada her bir iterasyon derinliğine aramadır. Derinliğine aramada derinlik sınırı yerine f-maliyet sınırı kullanılır. Verilen eş uzaklık eğrileri içindeki arama tamamlanınca bir sonraki eğriye geçilir.

IDA* ın zaman karmaşıklığı bulma fonksiyonunun alacağı farklı değerlere bağlıdır. 8-bulmacasında, Manhattan uzaklığı bulma fonksiyonu olarak alındığında, f herhangi bir çözüm yolunda 2 veya üç kez artar. Böylece IDA* 2-3 iterasyona girer. Birçok pratik problemin optimal çözümü ilk olarak IDA* ile bulunmuştur. IDA* birkaç yıl, tek optimal bellek-sınırlı bulma algoritması olarak kalmıştır.

IDA* karmaşık problemlerde iyi çözüm vermemektedir. Örneğin TSP probleminde bulma değeri her durum için farklıdır. Yani her eğri bir önceki eğriden yalnız bir fazla durum içermektedir. Eğer A* N düğüm açıyorsa IDA* N ietrasyon yapmak zorunda kalacaktır: $1+2+...+N=O(N^2)$. N bellek için büyük ise N^2 çok çok büyük olacaktır.

Bu durumu aşmak için f maliyet limiti her iterasyonda ε kadar artırılarak toplam iterasyon sayısı $1/\varepsilon$ ile orantılı hale getirilebilir. Bu arama maliyetini düşürür ama bulunan çözüm optimal çözümden ε kadar kötü olabilir. Bu algoritmalara ***kabul edilebilir ε*** (ε-admissible)

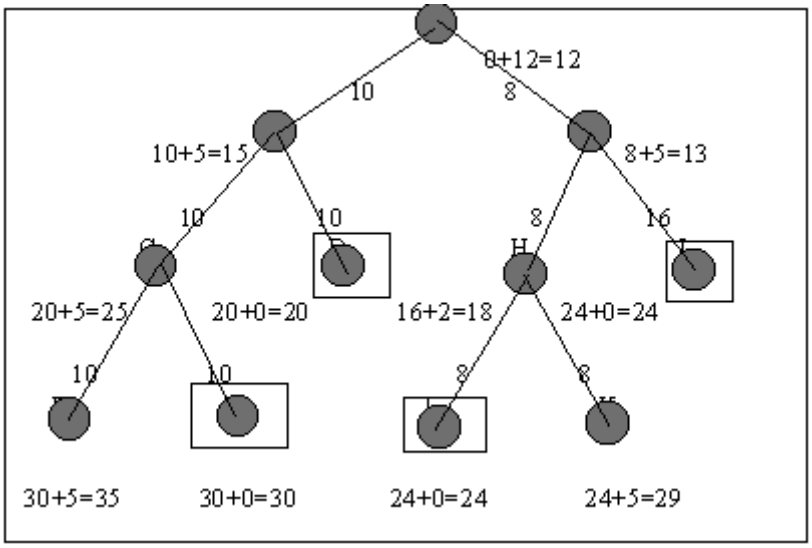
Basitleştirilmiş Bellek Sınırlı A* Arama (SMA*)

IDA* iterasyonlar arasında sadece o anki f-maliyet değerini saklamaktadır. Açtığı düğümler saklanmamaktadır. Bu nedenle aynı düğümleri tekrar tekrar açması gerekebilmektedir. Diğer bir deyişle IDA* çok az bellek kullanmaya çalışmaktadır.

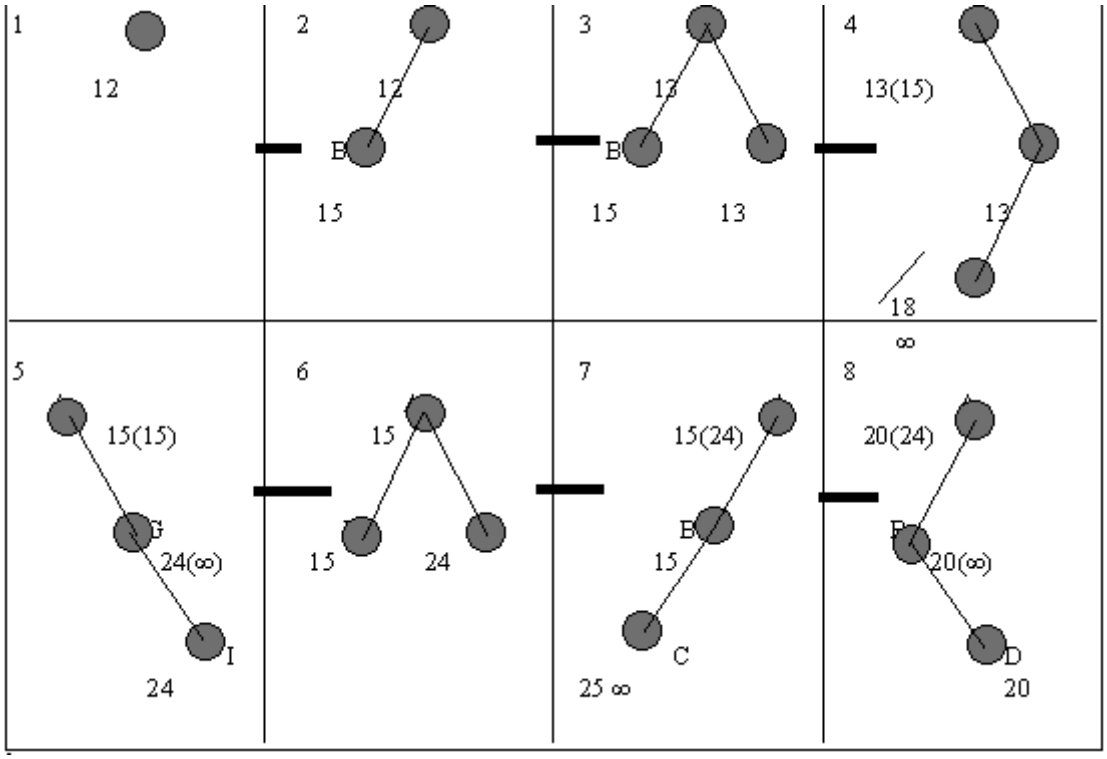
SMA*(Simplified Memory-Bounded A*) ise arama için var olan belleğin tamamını kullanmaktadır. Daha fazla bellek kullanmak sadece aramanın etkinliğini artırır. Gerektiği zaman bir düğümü hatırlamak onu tekrar üretmekten daha iyidir. SMA* ın özellikleri aşağıda verilmiştir:

- Ayrılan belleğin tamamını kullanır.
- Tekrarlanmış durumlardan sakınır.
- Eğer ayrılan bellek en sığ çözüm yolunu saklamaya yeterli ise ***tam***dır.
- Eğer ayrılan bellek en sığ optimal çözüm yolunu saklamaya yeterli ise ***optimal***dir.
- Tüm arama ağacı için yeterli bellek olduğunda arama etkindir.

Yeni bir düğüm üretileceği zaman bellekte yer kalmamış ise queue'daki bir düğüm çıkartılır. Bu düğüme ***unutulmuş düğüm*** (forgetten node) denir. Gelecek vaat etmeyen yani en yüksek f-maliyetli düğümlerin unutulması tercih edilir. SMA* örnek üzerinde açıklanacaktır.



Her düğüm f=g+h ile etiketlenmiştir. Amaç düğümler D,F,I ve J dikdörtgen içinde belirtilmiştir. Amaç yalnız üç düğüm saklanabilen bellek ile en düşük maliyetli amaç düğümü bulmaktır.



SMA* Algoritması aşağıdaki şekilde ilerler:

1. Her aşamada, en derin en düşük f maliyetli düğüme izleyen (successor) eklenir. İzleyen ağaçta olmamalıdır. A düğümüne B eklenmiştir.
2. f(A) hala 12 olduğu için G (f=13) eklenir. A' nın izleyenlerinin minimum maliyeti 13 olduğu için f(A) 13'e eşitlenir. Üç düğüm olduğu için bellek dolmuştur.
3. Düşük maliyetli G açılmadan önce bellekte yer boşaltılmalıdır. En sığ en yüksek değerli B yaprağı unutulur. A' nın en iyi unutulanın değeri f=15 parantez içinde gösterilir. Sonra H eklenir. f(H)=18'dir. H çözüm olmadığı için ve belleğin tamamı kullanıldığı için H' dan geçen bir çözüm bulunamaz. Bu nedenle f(H)=∞'e eşitlenir.
4. H iptal edilerek I açılır. f(I)=24'dür. G' nin izleyenleri 24 ve ∞ değerlerine sahiptir. Bu nedenle f(G)=24 olur. 24 ve 15 değerlerinden küçük olan 15 f(A)' nın değeri olur. I çözüm olmasına karşın f(A)=15 olduğundan en iyi çözüm olmayabilir.
5. Beklentisi olan B düğümü tekrar üretilir.
6. B' nin ilk izleyeni, maksimum derinlikteki C amaç düğüm olmadığından f(C)=∞ olur.
7. C' yi çıkartarak ikinci izleyen D' ye bakılır. f(D)=20'dir.
8. En derin en düşük maliyetli düğüm D' dir. D amaç düğüm olduğu için seçilerek arama işlemine son verilir.

4.4 İTERATİF İLERLEME ALGORİTMALARI

Birkaç iyi bilinen problemlerde mesela 8 vezir ve VLSI tasarımında durumun tanımının kendisi bir çözüm için ihtiyaç duyulan tüm bilgileri içermektedir. Çözümün hangi yolla erişildiği konu dışıdır. Bu gibi durumlarda iteratif ilerleme algoritmaları en pratik yaklaşımı sağlar. Genel fikir tam bir configürasyonla başlamak ve kalitesini arttırmak için değişiklik yapmaktır.

Tavlama Benzetimi Algoritması

Sezgiseller genellikle aşağıdaki kategorilere ayrılmaktadır:

- Açgözlü yapılanma (greedy construction metotlar)
- Komţu yöresi arama (neighborhood search)
- Gevţetme (relaxation teknikleri)
- Kısmi birerleme (partial enumeration)
- Ayrıştırma (decomposition)
- Bölüştürme (partition yaklaşımları)

Sezgisellerin çoğu probleme özel bir yapıya sahiptirler. Bir problem için kullanılan bir sezgisel, başka bir problem için kullanılamamaktadır. Bununla beraber, problemlere uygulanmaları açısından daha genel özelliğe sahip olan tekniklere ilgi son yıllarda giderek artmıştır. Bu teknikler çok sayıda problemlere uygulanmışlar ve oldukça güçlü bulunmuşlardır.

Tavlama benzetimi ve tabu arama lokal komşuluk arama sezgiselinden hareketle geliştirilmişler ve oldukça güçlü oldukları gösterilmiştir. Lokal komşuluk arama sezgiseli esas olarak şu şekilde çalışmaktadır: Proses herhangi bir başlangıç çözümü ile başlar ve bu çözümün tanımlanan komşuluğunu arayarak kendisinden daha iyi bir komşu çözümü olup olmadığını tespit eder. Eğer böyle bir çözüm varsa aynı işlemleri bu çözümün komşuluğu için tekrar eder, aksi halde proses durur. Bu durumda en son bulunan çözüm lokal olarak optimal bir sonuç olmaktadır.

Lokal optimaliteden kurtulmak için çeşitli yöntemler kullanılmaktadır. Bunlardan birisi komşuluğu genişletip içerisindeki komşu çözümleri çeşitlendirmektir. Bir diğeri ise, prosesi farklı başlangıç çözümleri kullanarak çok kez çalıştırmak ve elde edilen sonuçlar içerisinden en iyisini seçmektir. Son zamanlarda üzerinde durulan üçüncü bir yaklaşım ise “uphill” (yokuş-yukarı) hareketlere izin vererek lokal bir optimumda durmadan mevcut çözümden daha kötü olsa bile onun komşu çözümü üzerinden aramaya devam etmektir. Ancak “uphill” hareketlerin kabul edilmesi üzerinde bazı kısıtların olması zorunludur, aksi halde prosedür tüm çözüm uzayını saymaya çalışacaktır. Fakat bu prensibi kullanarak lokal bir optimumdan sıçrama ve daha iyi bir çözümü bulma şansı ortaya çıkmaktadır. Tabu arama ve tavlama benzetimi lokal optimumlardan sakınmak için bu üçüncü yöntemi farklı yaklaşımlarla kullanmaktadırlar.

Sezgisel teknikler bazı doğal oluşum proseslerini taklit etmeye çalışmaktadırlar. Tavlama benzetimi, termo-dinamik prosesleri taklit etmekte, tabu arama ise bir çeşit "hafıza" (memory) uygulamasını dikkate alarak "zeki" (intelligent) prosesleri taklit etmektedir. Genetik algoritmalar da, genetik yapılara benzer bir şekilde problemleri formüle ederek doğanın en iyi olan hayatta kalır prensibini taklit etmektedirler. Yapay sinir ağları ise insan beyninin hesaplama özelliklerini taklit etmektedir.

Tavlama Benzetimi (TB), kombinatoryal optimizasyon problemleri için iyi çözümler veren olasılıklı bir arama yöntemidir. "Tavlama Benzetimi" ismi, katıların fiziksel tavlanma süreci ile olan benzerlikten gelmektedir. TB algoritması, birbirlerinden bağımsız olarak, Kirkpatrick, Gelatt ve Vecchi (1983) ile Cerny (1985) tarafından hemen hemen aynı yıllarda önerilmiş ve literatüre girmiştir. Bilgisayar tasarımı, görüntü işleme (*image processing*), moleküler fizik ve kimya, çizelgeleme problemleri, haberleşme şebekeleri tasarımı gibi farklı alanlardaki bir çok optimizasyon problemine uygulanmıştır.

Basit bir TB algoritmasının adımları:

Adım 1. Uygun (feasible) bir başlangıç durumu seç: $i \in S$

Bir başlangıç sıcaklığı seç: $T > 0$

Sıcaklık değişim sayacını sıfırla: $t = 0$

Adım 2. Durdurma koşulu sağlanmışsa DUR, değilse tekrar sayacını

sıfırla: $n = 0$ ve devam et.

Adım 3. i'nin bir komţusu olan j durumunu rassal olarak üret.

$\Delta = f(j) - f(i)$

Eğer $\Delta < 0$ ise $i = j$, değilse ve $U(0, 1) < \exp(-\Delta /T)$ ise $i = j$.

Adım 4. $n = n + 1$. Eğer $n < M$ ise adım 3'e git, değilse $t = t + 1$,

$T = T(t)$ ve adım 2'ye git.

Tavlama Benzetimi Algoritması

Yukarıda, M her sıcaklık değerinde denenecek hareket sayısını ve T(t) t. sıcaklık değerini göstermektedir. TB algoritmasının global optimum çözümlere yakınsama hızı, M ve T(t), t = 0, 1, 2, ... parametreleri tarafından belirlenmektedir. Ancak pratikte, algoritmanın parametre değerlerinin uygulamaya yönelik seçimi "*tavlama*" veya "*soğutma planı*" ile belirlenmektedir. TB algoritmasında, başlangıç sıcaklığının, sıcaklık azaltma oranının, her sıcaklıktaki tekrar sayısının (komşu çözüm sayısı) ve durdurma koşulunun belirlenmesi tavlama veya soğutma planı

olarak tanımlanmaktadır. Soğutma planının seçimi, algoritmanın performansı üzerinde çok önemli bir etkiye sahiptir.

Tavlama Benzetimi Algoritmasının Uygulanması

TB algoritmasının uygulanması için verilmesi gereken bazı önemli kararlar; probleme özgü seçimler ve tavlama (veya soğutma) planına ait seçimler olmak üzere iki gruba ayrılabilir (Johnson ve diğerleri, 1989).

Probleme Özgü Seçimler: Problem, mümkün çözümlerin kümesi tanımlanabilecek şekilde formüle edilmelidir. Ayrıca, herhangi bir çözümün gösterimi (kodlama yapısı), herhangi bir komşu çözümün nasıl elde edileceği (hareket mekanizması) ve çözümlerin amaç fonksiyonu değerlerinin hesaplanma yöntemi tanımlanmak zorundadır. Bunun yanı sıra, bir başlangıç çözümünün üretilmesi gerekmektedir. Bazı araştırmacılar (Johnson ve diğerleri, 1991; Woodruff, 1994; Shapiro ve Alfa, 1995), kullanılan komşu yapısının TB algoritmasının performansını etkilediğini göstermişlerdir. Genelde, yerel optimumların sığ olduğu "düzgün" bir arama uzayını gösteren komşu yapısı, yerel optimumların derin olduğu "engebeli" bir arama uzayını gösteren komşu yapısına tercih edilmektedir.

Eğer kısıtlı bir problem söz konusuysa, çözüm uzayı sadece kısıtları sağlayan çözümler ile sınırlandırılmalıdır veya kısıtları bozan çözümler uygun bir ceza fonksiyonu dikkate alınarak çözüm uzayına dahil edilmelidir (Eglese, 1990).

Tavlama Planına Ait Seçimler: Sıcaklık parametresinin azalan değerlerinin sonlu bir sırası için sonlu uzunluklu homojen Markov zincirleri üretilerek oluşturulan bir TB algoritmasının, uygulaması da sonlu bir zaman alacaktır. Yakınsamanın sağlanabilmesi için bu algoritmada kullanılan parametreler kümesi uygun bir şekilde belirlenmek zorundadır. TB algoritmasında kullanılan parametrelerin ve değerlerinin belirlenmesi tavlama veya soğutma planı olarak tanımlanmaktadır. Tavlama planı ile aşağıdaki parametreler belirlenmektedir (Aarst ve Korst, 1989; Eglese, 1990):

1. T sıcaklık parametresinin başlangıç değeri,
2. Sıcaklığın hangi yöntemle azaltılacağını belirlemek için kullanılan T(t) sıcaklık fonksiyonu,
3. Her sıcaklıkta gerçekleştirilmesi gereken M tekrar sayısı,
4. Algoritmayı durdurmak için T sıcaklık parametresinin son değeri.

Günümüze kadar çeşitli tavlama planları önerilmiştir (Johnson ve diğerleri, 1989; 1991; Kouvelis ve Chiang, 1992). Önerilen en eski plan Kirkpatrick ve diğerlerinin (1983) fiziksel tavlama ile olan benzerliğe dayanarak ileri sürdükleri plandır. Bu tavlama planına göre,

maddenin sıvı safhaya ulaştığında tüm parçacıklarının rassal olarak düzenlenmesini taklit etmek için, T sıcaklık parametresinin başlangıç değeri, denenen tüm hareketler kabul edilecek kadar yüksek seçilmiştir. Sıcaklık parametresinin değerini azaltmak için ise oransal bir sıcaklık fonksiyonu kullanılarak sabit bir *r* için $T(t + 1) = r.T(t)$ dikkate alınmıştır. Burada *r*, 1'den küçük fakat 1'e yakın bir sabittir ve pratikte genellikle 0.80 ile 0.99 arasında bir değer almaktadır. Bu sıcaklık fonksiyonu ile sıcaklık parametresinin değeri, sıfıra yaklaştıkça daha da yavaş azalmaktadır. Sıcaklık parametresinin her değerinde gerçekleştirilecek M tekrar sayısı, sabit bir üst sınıra göre kabul edilen yeterli sayıda geçişler (hareketler) tarafından belirlenmiştir. Böylece problemin, fiziksel tavlamdaki ısıl dengeye karşılık gelen bir denge durumuna ulaşması amaçlanmaktadır. M tekrar sayısı, sabit veya problemin yada komşu yapısının boyutuyla orantısal alınabilmektedir. Bu tavlama planı ile, sıcaklık parametresinin her değerinde elde edilen çözüm, belli sayıda ardıl sıcaklık değişimleri boyunca aynı kalırsa TB algoritması durdurulmaktadır. Buna göre elde edilen son durum, fiziksel tavlamadaki "*donma durumuna* (*frozen state*)" karşılık gelmektedir.

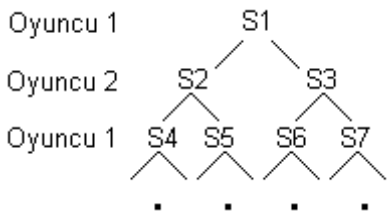
BÖLÜM 5
OYUN OYNAMA

Şu ana kadar ayrıntılı yaklaşımın veya sezgisel yaklaşımın bir sonucu ortaya çıkarmak için olası tüm eylemleri temsil eden bir ağaç üzerinde arama yapmaktan daha fazlası olduğunu düşünmedik. Bununla birlikte dünya, problemlerin doğasını değiştirmeden hareketler gerçekleştirilebilen durağan bir ortam değildir. Kimyada iyi bilinen bir ilke vardır - Le Chatelier ilkesi -. Bu ilkeye göre üzerinde herhangi bir değişiklik yapmaya çalıştığımız bir sistem, buna karşı koyabilmek için kendini değiştirecektir. Özellikle, eğer iki zeka bir yarışmada bir ödül için çekişmekteyseler bu karşı koyma ilkesi daha da belirginleşir. Çatışma içerisinde olan zekaya verilebilecek en iyi örnekler, satranç ve dama gibi oynanan oyunlardır ve buda yapay zeka için oyunların bu kadar çok çalışılmış olmasının sebeplerinden biridir. Bir oyunda sadece iyi bir hamle yapmanız yeterli değildir, çünkü rakibiniz, sizin hamlenizin sebep olduğu üstünlüğü ortadan kaldırmak için karşılık verecektir. Başka bir deyişle oyuncular arasında şimdiye kadar hesaba kattığımız şeylerden daha karmaşık ve çözümü daha zor olan bir etkileşim vardır.

Oyunlar, bir zekanın diğer bir zekayla rekabet halinde olmasının en açık örnekleri olmalarına rağmen bu şekil rekabet, insanların günlük yaşantılarında birbirleriyle olan etkileşme biçiminin de önemli bir bileşendir. Biz insanlar, her zaman, daha büyük veya daha küçük ölçülerde başkalarının davranışlarını nasıl etkileyeceğimizi hesaba katarak davranışlarımızı gerçekleştiririz.

Hamle Ağacı

Rekabete dayalı bir oyunu bir **hamle ağacı** (move tree) yapısında göstermek mümkündür. Bu çeşit ağaç ile daha önce bahsettiğimiz ağaç arasındaki fark, öncekinde her düzeydeki hamle seçimi hakkının yalnızca bize ait olmasıdır. Şimdikinde ise önce biz hamle yaparız, daha sonra rakibimiz hamle yapar ve oyun bitene kadar bu şekilde devam eder. Her sıra değişiminde sadece iki olası hamle olduğu varsayılırsa hamle ağacı, aşağıdaki şekle benzeyecektir.



Şekildeki S1, S2 . . . oyunun belirli durumlarını gösterirler. Tabi ki çoğu oyunun her sıra değişimi için ikiden çok daha fazla seçeneği vardır. Oyunu kazanma sorununu hala oyunun belirli bir durumuna yol açan patikalar için ağacı aramak olarak düşünebilirsiniz. Ama şimdi

şunu da hesaba katmak zorundayız; bizim bir sonraki hamle hakkımız, rakibimizin istediğimiz duruma ulaşmamızı engellemeye çalışacağı hamlesinden sonradır.

Durağan Değerlendirme İşlevi

Rekabet sırasında hamle ağacının nasıl aranacağını düşünmeye başlamadan önce bir hamle veya onun karşılığı olan oyun durumunun niteliğini değerlendirme yollarını bulmalıyız ki hamle yapmadan önce oyunun durumunu değerlendirerek bir hamlenin değerini bulabilelim. Bir hamlenin değerlendirilmesi güç görünse de bunun için tüm gerekli olan oyun akışı içerisinde o hamlenin kaba değerini gösteren bir sezgiseldir. Başka bir deyişle sezgisel bir değerlendirme yeterli olacaktır. Oyunu kesin olarak kazanma veya kaybetme koşulları için hamlelere değer biçmek tam olarak mümkünse oyun bir algoritma kullanılarak oynanacak kadar sıradandır. Böyle bir durumda tüm yapılması gereken bir özel hamlenin beraberinde ne kadar üstünlük getirdiğini toplamaktır. Tabiki buda neyin bir üstünlük olduğunun daha önceden bildirilmesini gerektirir, ancak birçok oyun için zaten bir açıklama bulunur. Eğer yukarıda bahsedilen gibi bir şart yoksa bir hamlenin seçimi, rasgele bir işlemdir. Çünkü oyuncu için her bir hamle, en az diğer hamleler kadar, bir kazanma nedenidir. Eğer insanlar oyunda ustalığın değişik düzeylerine ulaşabilirlerse, o zaman **iyi hamle**nin (good move) uygun bir açıklaması yapılabilir.

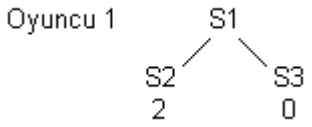
Durağan değerlendirme işlevi (static evaluation function), **$E(S_n)$**'dir. Bu işlev, oyunculardan biri için verilen bir S_i durumuyla ilgili ne kadar üstünlüğün var olduğunu gösteren bir değeri geri döndürür. Örneğin, eğer $E(S_2)$, $E(S_3)$'ten daha büyükse S_2'ye sebep olan hamle S_3'e sebep olandan daha iyidir. Uygulamada durağan değerlendirme işlevleri, çoklukla bir oyuncunun konumunu etkilemek için düşünülmüş birtakım etkenlerin bir ağırlıklı ortalaması yapısındadır. Örneğin, dama oyununda şunun gibi bir değerlendirme işleviyle karşılaşırsınız:

c_1 x (taş üstünlüğü) + c_2 x (merkezin denetimi) + c_3 x (yer değiştirme)

Terimlerden her biri, (taş üstünlüğü vb.) özel bir tür üstünlüğün ölçüsüdür ve c_1, c_2, c_3 . . . ise farklı ölçüleri tek bir üstünlük ölçüsü içinde birleştirmeye yarayan değişmezlerdir. Birçok durumda c_1, c_2, c_3 . . , c_n'ler için en iyi değerler bilinemezler ve deneme yanılma yoluyla bulunmaları gerekir. Eğer bir oyun programı yenilmeye eğilim gösterirse muhtemelen durağan değerlendirme işlevinin değiştirilme zamanı gelmiştir. Açıkçası birçok hamlenin ileride nelere sebep olacağını hesaba katan bir durağan değerlendirme işlevi yapılandırmaya gerek yoktur. İşlevin tüm yapması gereken, oyunun şimdiki durumuna ilişkin mümkün olduğunca iyi bir işaret vermektir.

Tek Kat Arama

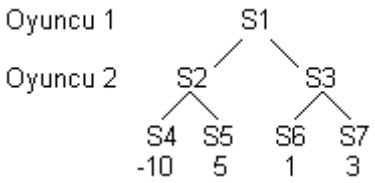
Eğer durağan değerlendirme işlevi $E(S_n)$, Oyuncu1'in üstünlüklerini, artı (positive) değerler ve oyuncu2'nin üstünlüklerini de eksi (negative) değerler olarak gösterirse ağacın bir sonraki düzeyinde Oyuncu1'in $E(S_i)$'nin en büyük değerini veren hamleyi seçmesi gerektiğini görmek zor değildir. Örneğin, Oyuncu1, S_2 ve S_3 hamlelerinden birini seçme durumundaysa ve $E(S_2)=2$ ve $E(S_3)=0$'sa bu oyncunun tek seçeneği S_2 hamlesini yapmaktır. Değerlendirme işlevi, bir hamle ağacının parçası olarak şekildeki gibi olabilir.



Çoklukla bir sonraki hamle için iki seçenekten daha fazlası vardır ama en büyük durağan değerlendirme işlevini üreten hamlenin seçilmesi ilkesi temelde aynı kalır. Oyuncu, her sıra değişiminde oyun ağacının yalnızca bir düzey veya kat aşağısını aradığı için bu yöntem **tek kat arama** (one ply search) stratejisi olarak anılmaktadır.

Aramayı Genişletmek - minimax ilkesi

En iyi bir sonraki hamlenin seçilmesi işleminde iyileştirme yapmak için en açık yol, hamle ağacının daha aşağılarına inerek seçilmesi düşünülen hamlenin etkisini incelemektir. Durağan değerlendirme işlevi tek kat aşağısını değerlendirirken iyi gibi görünen bir hamle, sonraki düzeylerde çok kötü bir hamleye dönüşebilir. Bunu daha iyi anlamak için aşağıdaki şekli inceleyelim:

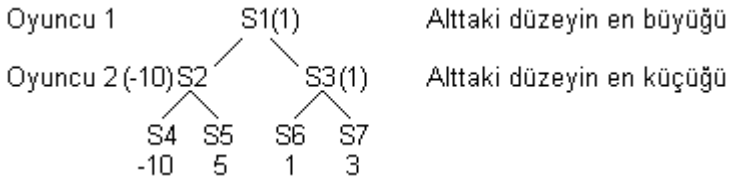


Eğer biz her bir hamlenin etkisini hamle ağacının iki düzey ilerisi için deniyorsak bu yaptığımız, **çift kat arama**dır (two ply search). Tek kat arama, S_2'nin en iyi hamle olduğuna karar versin ama bir sonraki düzeyde Oyuncu2'nin S_4 ile yapacağı -10, bunun Oyuncu1 için çok kötü bir durum olduğunu gösterir. Durağan değerlendirme işlevi iki hamle ileriye bakmak için kullanıldığında, sıra değişiminden sonra Oyuncu2'nin yapabileceği hamleler de hesaba katılacaktır. Oyuncu1, hangi hamleyi yaparsa yapsın, Oyuncu2, buna yine bir hamleyle karşılık verecektir. Bu karşı hamle, durağan değerlendirme işlevini olduğunca küçük yapar (eksi yönde). Oyuncu1'in yapabileceği en iyi şey, sonraki hamlelerin en küçük skorunu en büyültmektir.

Oyuncu2 tarafından seçilmiş en küçük skorlardan en büyük olanının tercih edilmesi bu oyun oynama stratejisinin ismini verir - **minimax** -. Yukarıda verilen oyun ağacı örneğinde, eğer Oyuncu1, S_2'yi seçerse Oyuncu2, -10 skorlu S_4'ü seçecektir. Ancak Oyuncu1, S_3'ü seçecek olursa buna 1 skorlu S_6 ile cevap verilecektir. -10 ve 1, Oyuncu2'nin başarabileceği en küçük skorlardır ve bu yüzden Oyuncu1'in başarabileceği en büyük skor, S_3 ile sonuçlanan hamleyi yaparak, 1'dir.

Yedeklenmiş Skor

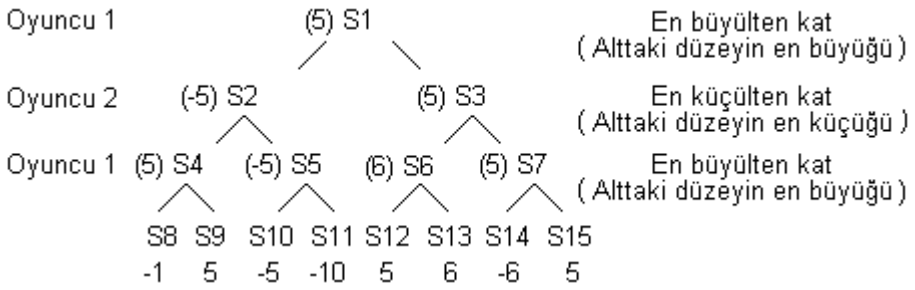
Durağan değerlendirme işlevini yedekleme düşüncesi anlatıldığında minimax stratejisi daha iyi anlaşılacaktır. Örneğin, önceki konuda bahsedilen hamle ağacı üzerinde S_2 ile sonuçlanan hamleyi yapalım. Oyuncu2, S_4 hamlesiyle karşılık verdiği taktirde S_2 hamlesinin -10'luk bir skorla neticeleneceği kolaylıkla görülebilir. -10 skoru, S_2 hamlesinin ne kadar iyi olduğunun değerlendirmesi olduğundan bu skoru S_2'nin yanına yazabiliriz. Aynı şekilde 1 skorunu da ne derece iyi olduğunu göstermek için S_3'ün yanına yazabiliriz. Bu skorlar, **durağan değerlendirme işlevinin yedeklenmiş değerleri** olarak adlandırılırlar ve tümü de daha sonraki hamlelerin skorlarının en küçükleridir. S_1'den başlayarak Oyuncu1'e ait tüm düzeylerde de en büyük skorların yedeklenmiş değer olarak seçilmesi akla yatkındır. Skorları bu şekilde yedekleyerek şekildeki çizenek oluşturulur.



Durağan değerlendirme işlevi, hamle ağacının en ucundaki durumları değerlendirmek için kullanılır.Ağaçtaki diğer tüm değerlerse hemen bir kat aşağısının en büyük yada en küçük değerleridir.

N Kat Arama

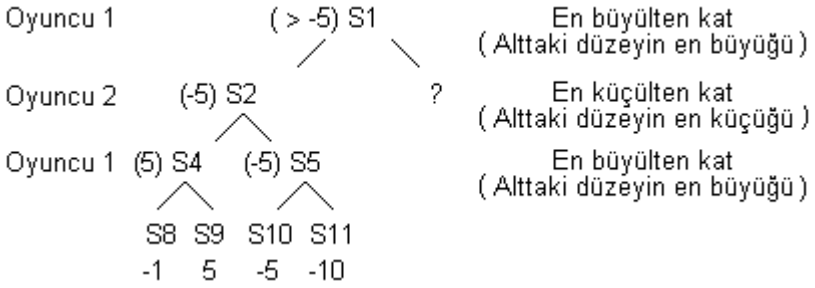
Yukarıda anlatılan düşünceyi herhangi sayıda hamlenin ileriye doğru aranması için genişletmek mümkündür. Katlardan her biri, ya Oyuncu1'e ait olan en büyülten bir kat, yada Oyuncu2'ye ait olan en küçülten bir kattır. Durağan değerlendirme işlevi, hamle ağacının en son katındaki durumlara uygulanır ve daha önceki katlardaki tüm durumlar bu değerleri yedeklemek yoluyla değerlendirilebilir. Bir durumun yedeklenmiş değeri, eğer o durum bir en küçülten kattaysa ondan elde edilebilecek skorların en küçüğüdür, ancak o durum bir en büyülten kattaysa ondan elde edilebilecek skorların en büyüğüdür. Örneğin, aşağıdaki şekildeki hamle ağacı iki en büyülten ve bir de en küçülten düzeyiyle üç katlı bir aramayı temsil eder.



Yedeklenmiş skorlar şekil üzerinde her bir duruma bitişik olarak ayraç içinde gösterilmiştir. Bu örnekte S_3 ile sonuçlanan hamleyi yapması koşuluna bağlı olarak Oyuncu1'in gerçekleştirmeyi umduğu en iyi skor, 5'tir. Oyuncu2 de ağacın bir düzey daha aşağısını aradığından S_7 ile sonuçlanan hamleyi yapacağı rahatlıkla söylenebilir. Aynı yolla eğer Oyuncu2, S_7 ile sonuçlanan hamleyi yaparsa Oyuncu1'in bunun ardından S_{15}'e oynayacağı hakkında güvence verilemez, çünkü S_7'den 3 kat aşağı yapılan bir arama S_{14}'ün daha iyi bir hamle olduğunu ortaya koyabilir. Hamle ağacı çözümlemesindeki gaye, hangi hamlenin en iyi bir sonraki hamle olduğunu saptamaktır ve sonraki hamlelerin yoklanması da sadece bu hamlelerin değerlendirilmesini iyileştirmek içindir.

Arama Sırası

Buraya kadarki konularda hamle ağacını belirli bir derinliğe ait tüm durumlar için tüm yedeklenmiş değerleri ve değerlendirme işleviyle birlikte sunduk. Uygulamada hamle ağacının tam görüntüsünün oluşturulabilmesi için ağaç hamle hamle keşfedilmelidir. Bunu yapmak için birçok yol vardır, ancak en uygun olanı bir **derinlik önde**(depth first) aramadır. Bu arama kuralının nasıl olduğunu bir örnek üzerinde görelim:



Daha önce verdiğimiz hamle ağacını aramak için öncelikle S_1'den ayrılan yollardan birini seçin, burada S_2 seçilmiş. Ardından S_2'den bir hamle seçin, burada S_4 seçilmiş. S_4'ten bir aşağısı ağacın sonuncu düzeyidir ve bu aşamada durağan değerlendirme işlevi S_8'in ve S_9'un değerlendirilmesinde kullanılıp S_4'e 5 değeri yedeklenir. Ardından S_5'in altındaki ağaç parçası incelenerek S_5'e de -5 değeri yedeklenir. S_4'e ve S_5'e ait yedeklenmiş skorlar bilindiğine göre S_2'nin skoru artık hesaplanabilir. S_1 için olan skor henüz hesaplanamaz çünkü S_3 ile başlayan

ağaç parçası henüz incelenmemiştir. Ancak şunu biliyoruz ki S_3'e yedeklenen skor en azından -5 olmalıdır ve bu skor daha büyük bir skorla karşılaşana kadar S_1'in yanına yazılabilir. Hamle ağacının üç kat değerlendirilmesinin buraya kadar anlatılan kısmı yukarıdaki şekilde gösterilmiştir.

Dal ve Sınır Yöntemleri

Başlangıç

Derinlik önde aramayı anlamak kolaydır ama sadece birkaç katın aranması bile başa çıkılması gereken büyük bir sorundur. Örneğin, sıfırlar ve çarpılar oyununda tek kat için saptanabilecek en fazla 9 tane hamle vardır ve bu sayı çift kat aramada 72 ve üç kat aramadaysa 504 olur. Denenmek zorunda olan durumların sayısını azaltmak için yapılabilecek herhangi bir şey kullanışlı bir yaklaşım olarak aramanın hızlanmasında yardımcı olacaktır. Bir altağacı aramaya değip değmeyeceğini saptamak maksadıyla belirli bir kat için tam arama yapmaktan sakınma yöntemleri **dal ve sınır** (branch and bound) ilkeleri üzerine kurulmuştur. Eğer bir altağaç arayacak kadar kıymetli değilse bu altağacın tümünün atılması aynı zamanda **budama** (pruning) olarak da bilinir. Bir dal ve sınır yöntemi, halihazırda elde edilmiş bir sonuçtan daha iyi bir sonuç vermeyecek olan kısmi çözümleri atmak için bir koşul belirler. Örneğin, seyahat eden işadamı probleminde, eğer yolun uzunluğu bilinen en kısa tam yolunkinden daha uzunsa bu yolu devam ettirmenin bir anlamı yoktur. Burada o ana kadar bulunmuş en kısa tam yol bir alt eşik görevi görmektedir. Dal ve sınır yöntemleri çoklukla probleme özeldirler ama bir minimax aramasında kullanmaya değecek ölçüde de etkilidirler.

Alfa-Beta Budaması

Bir dal ve sınır yöntemi, minmax araması için en büyülten ve en küçülten katların varlığını hesaba katmak zorundadır. Bu yüzden alfa ve beta olmak üzere iki tip eşik tutmalıyız. **Alfa** (alpha), o ana kadar bir en büyülten düğümde saptanmış en büyük değerdir ve **beta** (beta), arama sırasında o ana kadar bir en küçülten düğümde saptanmış en küçük değerdir. Daha açık bir ifadeyle alfa, bir en büyülten düğümde en sonunda saptanacak olan skor için bir **alt sınır** (lower bound) ve beta da bir en küçülten düğümde en sonunda saptanacak olan skor için bir **üst sınır**dır (upper bound).

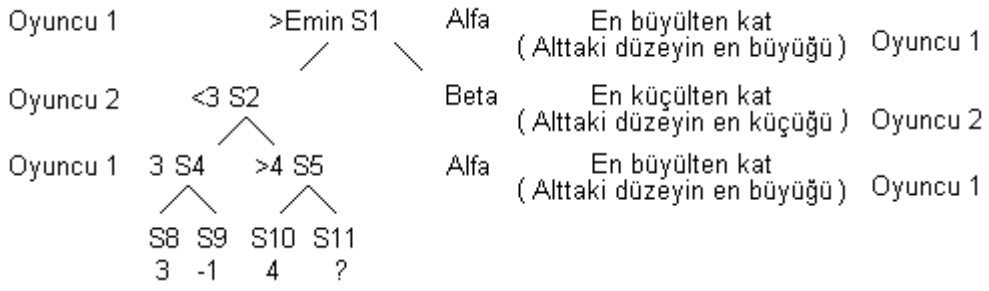
Alfa ve beta değerlerinin bir dept first arama sırasında durumlara atanmış geçici skorlar olduklarının farkına varmışsınızdır. Alfa, bir en büyülten kattaki bir durum için, beta ise bir en küçülten kattaki bir durum için geçici skorlardır. Geçici olan ve kalıcı olarak yedeklenmiş olan skorlar arasındaki ayrımı ortadan kaldırdığımızda bir en büyülten kattaki tüm değerlere alfa ve

bir en küçülten kattaki tüm değerlere de beta diyebiliriz. Bu durumda bir değerin henüz tam olarak saptanmadığını göstermek için küçüktür (<) ve büyüktür (>) işaretlerini kullanacağız. Bir başka yalınlaştırma da şudur; aramaya başlamadan önce bir en büyülten düzeye ait bir durumun alt sınırı, değerlendirme işlevinin döndürebileceği en küçük değer (Emin) ve bir en küçülten düzeye ait bir durumun üst sınırı da değerlendirme işlevinin döndürebileceği en büyük değer (Emax) olduğundan her düzeyde **Emin**'i alfanın başlangıç değeri ve **Emax**'ı da betanın başlangıç değeri olarak almalıyız. Böylece şu iki tanımı yapabiliriz:

- Bir en büyülten kattaki bir durum için alfa, bir sonraki katta ona bağlı olan durumlara ait beta değerlerinin en büyüğüdür.
- Bir en küçülten kattaki bir durum için beta, bir sonraki katta ona bağlı olan durumlara ait alfa değerlerinin en küçüğüdür.

Eğer bu iki tanımı da zor buluyorsanız alfa ve betanın derinlik önde arama bittiği zaman yedeklenen skorun alabileceği en büyük ve en küçük değerleri temsil ettiklerini bilmeniz şimdilik yeterlidir.

Bir hamle ağacının budanmasında alfa ve betanın nasıl kullanıldıklarnı görmek için aşağıdaki şekildeki hamle ağacı arama sahnesini inceleyelim.

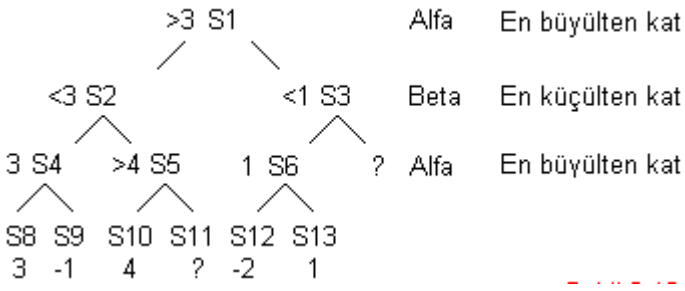


Biz bu sahnede sadece S4, S8 ve S9'u tam olarak değerlendirdik. Bu değerlendirmenin sonucunda şunu öğrendik ki S2'deki değer, 3'e eşit veya 3'ten küçük olmalıdır (S2'deki beta değeri). S10'un değerlendirilmesinden sonra S2'deki beta değeri S5'ten geçen alt ağacın aranmasında daha ileri bir nokta olmadığını gösterir. Bunun sebebi, S11'in değerinin neyi gösterdiğinin bir önemi olmaksızın bir en büyülten kat olarak S5'in değerinin en azından 4 olması gerektiği ama S2'nin de en fazla 3 olması gerektiğinden S5'in değerinin bir etki almak için halihazırda çok büyük olduğunun güvencelenmesidir.

S11'in değerlendirilmesine gerek duymamak çok büyük bir kazanç gibi görünmeyebilir, ancak bu tartışmanın hamle ağacındaki herhangi bir noktaya uygulanabileceğine ve atılmış durumun tam bir ağacın değerlendirilmesini içerebileceğine dikkat edin. Bu anlatılanlardan şu sonuçları çıkarabiliriz:

1. Herhangi bir anda bir en büyülten kattaki bir durumun alfa değeri hemen yukarısındaki durumun beta değerinden daha büyükse bunun devamını takip etmeye gerek yoktur.
2. Herhangi bir anda bir en küçülten kattaki bir durumun beta değeri hemen yukarısındaki durumun alfa değerinden daha küçükse bunun devamını takip etmeye gerek yoktur.

Yukarıdaki 2. kurala örnek olarak bir ağacı budamak için alfa eşiğini içeren yukarıdaki şekildeki hamle ağacının devamını dikkate alalım. Derinlik önde arama S3 üzerinden S6'ya iner. Ardından S12 ve S13'ün değerlendirilmeleri aşağıdaki şekilde gösterilen hamle ağacını üretir.



Daha yukarıda anlattığımız beta budaması, S11'in değerlendirilmesinin gerekli olmadığına karar vermişti. Bu sayede S2 değeri tam olarak bilinmekte ve bu değer S1'e atanacak alfa değerinin 3'ten büyük (>3) olmasına olanak tanımakadır. Bu noktada S6'nın yedeklenmiş skoru da biliniyor olsun. Bu bilgi kullanılarak S3'teki beta değeri, <1 olarak hesaplanır. Bu değer, S3'ün altındaki ağacı arayarak elde edeceğimiz değer ne olursa olsun (? ile işaretli) S3'ün değeri 1'e eşit veya 1'den küçük olmalıdır anlamına gelir. Buradan yola çıkarak S3'e bağlı durumları değerlendirmeye devam etmenin gereksiz olduğu, çünkü S1, S2'den kaynaklanan 3 değerine halihazırda sahip olduğundan S3'ün en son değeri ne olursa olsun S2'nin daha iyisi için teminat vediği söylenebilir. Başka bir deyişle beta değeri bir üstteki durumun alfa değerinden daha küçük olduğundan S3'e ait altağacı daha fazla aramak zorunda değiliz. Böylelikle alfa budaması iki seviyeye ait üç fazladan durumu arama işinden bizi kurtarır.

Ek sezgiseller

Rekabete dayalı bir oyun oynarken alfa-beta budaması ile birleştirilmiş bir minimax araması kullanmak çok etkili bir yöntemdir. Ama, hala bir takım basit yöntemler eklemek yoluyla bazı kazançlar elde etmek mümkündür. Bunların çoğu sezgisel kurallar bölümüne girer, çünkü oyunun ilerlemesini güvence altına almaktan ziyade ona yardımcı olurlar.

Minimax aramasının amacı, tüm oyun ağacını verilen bir derinliğe göre arayarak bir sonraki en iyi hamleyi bulmaktır. Uygulamada çok derin bir arama gerçekleştirmek mümkün değildir, çünkü zaman sınırlaması vardır. Ama en iyi hamleyi seçtikten sonra acaba her şey yolunda mı

diye bu hamleye ait altağacın birkaç hamle daha derine kadar aranmasının ne sakıncası olabilir? Bu, **ikincil arama** (secondary search) olarak adlandırılır ve sabit arama derinliğinin hemen önünde olan tuzaklara düşmemek için iyi bir yöntemdir. Eğer bir ikincil arama bir tuzağı ortaya çıkarırsa yapılması gereken şey, en iyi ikinci hamleyi seçmek ve aynı değerlendirmeye tabi tutmaktır.

Arama derinliğini değiştirmek iyi bir düşüncedir ve iyi olan hamlelerin üzerine yoğunlaşmanın bir yoluymuş gibi düşünülebilir. Daha aşağı seviyelere kadar arama yapmanın bir diğer ölçütü de **suskunluk**tur (quiescence). Eğer derine indikçe arama sonuçları çılgıncasına değişiyorsa oyunda uzun dönem stratejisiyle bir şey yapılamayacak bir durum vardır. Örneğin, satrançta bir taş değişimi meydana geliyorsa Oyuncu1'in taş kaybedeceği düzeyi inceleyince bunun kötü bir hamle olduğu görülür ama Oyuncu1'in taş kazanacağı bundan bir aşağıdaki düzey arandığında ise çok iyi bir hamle olduğu görülür. Taş değişimi gibi oyunu değerlendirmemizi etkileyen kısa vadeli olaylardan etkilenmemek için hamle ağacını hamlenin değerlendirmesinin her bir arama derinliğinde aynı olduğu bir derinliğe kadar aramamız gerekir. Buna, **suskunluk için bekleyiş** denir.

Tüm minimax aramaları için en açık ve dolaysız iyileştirme, oyuna kitap hamlelerinin dahil edilmesiyle olur. Bunlar genellikle minimax aramasının yerine uygulanmak için algoritmasal olarak şifrelenmiş yaygın açılış ve kapanış stratejileridir. Kitap hamlelerinin en iyi bilinen örnekleri satranç açılışlarıdır ve daha önemsiz örnekler sıfırlar ve çarpılar oyunu için verilebilir (merkezde veya köşelerden birinde taş bulundurmak gibi). Kitap hamlelerini gerçekleştirmekle ilgili temel sorun bu hamlelerin yerine minimax aramasını yapmaya ne zaman başlanacağıdır.

BÖLÜM 6
YAPAY ZEKA SİSTEMLERİ

Yönetim bilimleri, yapay zeka alanındaki gelişmelerden hızla etkilenmektedir. Bu etkileşimin bir sonucu olarak, doğal dil arabirimleri, endüstriyel robotlar, uzman sistemler ve zeki yazılımlar gibi uygulamalar ortaya çıkmıştır. Her seviyeden yöneticiler ve çalışanlar, direk veya dolaylı da olsa son kullanıcı olarak bu gelişmelerden haberdar olmak durumundadır. Çünkü bir çok işyeri ve organizasyonda, gittikçe artan bir oranda yapay zeka teknikleri kullanılmakta ve bu yolla verimlilik artışı sağlanmaya çalışılmaktadır.

Bilgisayar Bilimleri:

Uygulamaların bu alanı bilgisayar yazılım ve donanımı üzerine odaklanmıştır. Çünkü yapay zeka uygulamalarının çoğu için, çok güçlü süper bilgisayarların üretilmesine gereksinim duyulmaktadır. Bunun ilk aşamasını beşinci nesil olarak anılan zeki bilgisayarlar oluşturmaktadır. Bu bilgisayarlar optimum seviyede mantıksal anlam çıkarma işlemi için tasarlanmaktadırlar. Bu anlam çıkarma, geleneksel bilgisayarlardaki nümerik işlem yerine sembolik işlemin kullanılması anlamına gelmektedir. Diğer çalışma ise, sinirsel ağların geliştirilmesi için yapılmaktadır. Neurocomputer sistemleri, insan beynindeki nöronların ağ yapılarına göre şekillendirilmiş bir yapıdadır Bu bilgisayarlar bilginin bir çok farklı kısmını aynı anda işleyebilirler. Sinirsel ağ yazılımlarının, basit problem ve çözümleri gösterilerek öğrenmesi sağlanabilmektedir. Örneğin resimleri tanıyabilmekte ve problemleri çözmek için program yapabilmektedirler.

Robotik :

Yapay zeka, mühendislik ve psikoloji robotiğin temel disiplinleridir. Robotik teknolojisi, insan gibi fiziksel kapasitelere sahip, bilgisayar kontrollü robot üretiminin gerçekleştirilmesi için geliştirilmiştir ve yapay zeka alanındaki gelişmelere paralel olarak ilerlemektedir. Bu alandaki uygulamalar robotlara, görme yeteneği veya görsel algılama, dokunsal algılama, idare etmede beceri ve hüner, hareket kabiliyeti ve yol bulabilme zekası kazandırmaktadır. Bazı uygulama örnekleri aşağıda verilmiştir.

Stuttgart Üniversitesinde bir çalışma grubu Aramis (adını monte edilmiş olan kolundan alıyor), Porthos (yük taşıyıcısı) ve Athos (bir stereo kameraya sahip ve gurubun gözcüsü) isimli üç robot üretmiştir. Bu robotlar küçük sorunlarını tekbaşlarına çözebilmektedir. Fakat bu

robotlarda diğerlerinde olmayan bir özellik vardır, kooperasyon yeteneği. Şöyleki; kimin hangi görevi hangi sırayla yapacağını aralarında kararlaştırıyorlar. Aslında makineler bit ve byte'lar düzleminde anlaşmalarına rağmen, çalışma esnasında kadın ve erkek sesleriyle gerçekleşen sözlü diyaloglar ortaya çıkmaktadır.

Bir başka örnek ise MIT'den Rodney Brooks'un tasarladığı ATTİLA isimli böcek robot. 30 cm. boyutundaki bu robot üzerinde 23 motor, 10 mikro işlemci ve 150 adet algılayıcı bulunuyor. Her bacağın üç bağımsız hareketi sayesinde engellerin üstüne tırmanıyor, dik inişler yapıyor ve tutunarak kendisini 25 cm. yüksekliğe çekebiliyor. Brooks'un yapay zeka anlayışında izleme, avlanma, ileri gitme ve gerileme gibi bir takım ilkel içgüdü ve refleksler yer alıyor. Öte yandan onun robotlarında bunları seçen ve bu basit hareketleri yönlendiren bir beyin modeli yer almıyor. Bunun yerine, her davranış, robotun kontrolünde yarışan bireysel zekalar olarak işliyor. Kazananı, robotun alıcılarının o anda ne hissettiği belirliyor ve bu noktada diğer tüm davranışlar geçici olarak bastırılıyor. Kurulan mantıkta, "gerile" gibi tehlikeden sakınma davranışları, "avı izle" gibi daha üst seviyedeki fonksiyonları bastırıyor. Davranış hiyerarşisindeki her seviyenin gerçekleşmesi için bir alttakinin aşılması gerekiyor. Böylece bir böcek robot, örneğin "odadaki en uzak köşeyi belirle ve oraya git" gibi yüksek düzeyde bir komutu, bir yerlere çarpıp başına kaza gelme korkusu olmadan yerine getirebiliyor.

Robotlar gelecekte yalnızca basit ve monoton görevlerle sınırlanmayıp, insanlara karmaşık ve tehlikeli görevlerde de yardımcı olacakları için, akıllı ve daha esnek kullanımlı bir kavrama sisteminin geliştirilmesine yönelik olarak , DLR (Alman Hava ve Uzay Uçuşları Araştırma Kurumu) tarafından insan elini örnek alan üç parmaklı ve çok sensörlü bir robot el geliştirilmiştir.

Doğal Arabirimler :

Doğal arabirimlerin gelişimi yapay zekanın önemli bir alanını göz önüne alır. Doğal arabirimlerin gelişimi, insan tarafından bilgisayarların daha doğal kullanımına yönelik bir kolaylık sağlar. Bu alanda yapay zeka araştırmacılarının en büyük amacı, insan konuşma dilinde bilgisayar ve robotların konuşmaya başlaması ve bizim onları anladığımız gibi onların da bizi anlayabilmesidir. Uygulamalar dil bilim, psikoloji, bilgisayar bilimleri gibi disiplinleri içine alan bir kollektif çalışma alanı içinde yapılmaktadır. Bazı uygulama alanları olarak insan dilini

anlama, konuşmayı tanıma, beden hareketlerinin şekillerini kullanan çok algılayıcılı cihazların geliştirilmesi gösterilebilir.

Bilgisayar ile ilişki kurmak için bir anadilin kullanılması aslında yapay zekanın en kuvvetli yanlarından birini temsil eder. Yazılı anadilin işlenmesi uygulamaları ise çok sayıda bulunmaktadır. Bu konudaki başlıca uygulamalar şunlardır:

- *Bilgisayar yardımıyla tercüme,*
- *Metin özetlerinin otomatik olarak hazırlanması,*
- *Metinlerin otomatik olarak üretilmesi (anlamlı bir sözdizimsel form olarak),*
- *Dökümanların hazırlanmasına yardım (hataların ve tutarsızlıkların bulunması ve gerektiğinde düzeltilmesi, örnek: MSWord programı).*

İnsan sesini algılayan bir uygulama örneği olarak da, NaturallySpeaking isimli bir program seti verilebilir. Program erkek/bayan ayrımı yapmamak için ses girişlerini nötr sinyallere çevirir. Bir batch işlemi, konuşmaları konuşmacıdan bağımsız olarak kendi iç modeliyle karşılaştırarak, süreklilik ve vurgulama gibi ince ayarları yapar. Farklı kullanıcıların telaffuz farklılıklarındaki tutarlılık bu sayede sağlanır. Program ayrıca zaman kaybetmemek için, söylenen bir kelimenin ardından gelebilecek kelimeleri tahmin eder ve tarama alanını daraltır. Mesela, sayın kelimesinden sonra, büyük bir ihtimalle isim gelecektir, tarama alanı buna göre isim alanına yönlendirilir. Bunun ötesinde tüm cümlenin anlamına bakılarak, kelimenin cümlede uygun yerde olup olmadığı da kontrol edilir. Programın elindeki bilgiler arttıkça eskisine göre farklı kararlar verdiği görülmektedir. Gündelik konuşmalarda rastlanan cümlelerde program mükemmel bir performans sergilemektedir. Bir günlük düzenli bir çalışma sonrasında doğruluk oranı %95'lere ulaşmaktadır.

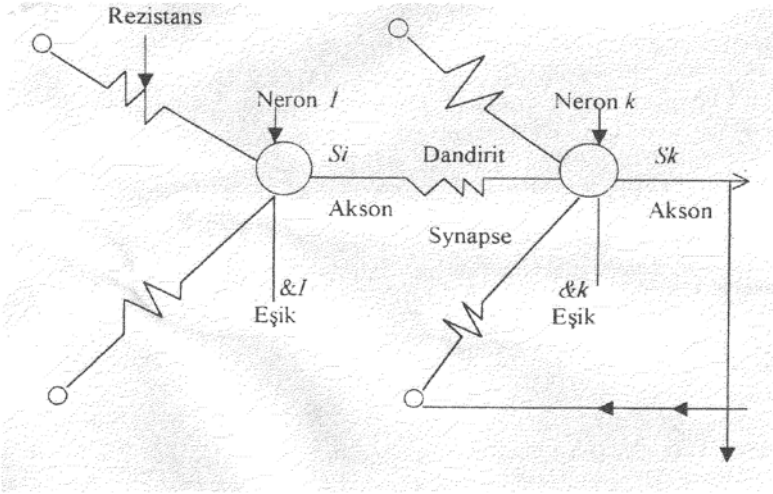
Sinirsel Ağlar:

Sinirsel ağlar çeşitli yollarla birbirine bağlı birimlerden oluşmuş topluluklardır. Her birim iyice basitleştirilmiş bir nöronun niteliklerini taşır. Nöron ağları sinir sisteminin parçalarında olup biteni taklit etmekte, işe yarar ticari cihazlar yapmakta ve beynin işleyişine ilişkin genel kuramları sınamakta kullanılır. Sinirsel ağ içindeki birimler, herbirinin belli işlevi olan katmanlar şeklinde örgütlenmiştir ve bu yapıya yapay sinir ağı mimarisi denir.

Yapay sinir ağlarının temel yapısı, beyne, sıradan bir bilgisayarınkinden daha çok benzemektedir. Yine de birimleri gerçek nöronlar kadar karmaşık değil ve ağların çoğunun yapısı, beyin kabuğundaki bağlantılarla karşılaştırıldığında büyük ölçüde basit kalmaktadır.

Şimdilik, sıradan bir bilgisayarda, akla uygun bir sürede taklit edilebilmesi için bir ağın son derece küçük olması gerekiyor. Gittikçe daha hızlı ve daha koşut çalışan bilgisayarlar piyasaya çıktıkça zamanla gelişmeler sağlanacaktır

Aşağıdaki şekilde, bir insanın sinir sistemindeki axon,dandirit ve nöron ilişkisi gibi tasarlanan yapay ağ mimarisi görülmektedir.



.

Yapay sinir ağlarındaki her bir işlem birimi, basit anahtar görevi yapar ve şiddetine göre, gelen sinyalleri söndürür ya da iletir. Böylece sistem içindeki her birim belli bir yüke sahip olmuş olur. Her birim sinyalin gücüne göre açık ya da kapalı duruma geçerek basit bir tetikleyici görev üstlenir. Yükler, sistem içinde bir bütün teşkil ederek, karakterler arasında ilgi kurmayı sağlar. Yapay sinir ağları araştırmalarının odağındaki soru, yüklerin, sinyalleri nasıl değiştirmesi gerektiğidir. Bu noktada herhangi bir formdaki bilgi girişinin, ne tür bir çıkışa çevrileceği, değişik modellerde farklılık göstermektedir. Diğer önemli bir farklılık ise, verilerin sistemde depolanma şeklidir. Nöral bir tasarımda, bilgisayarda saklı olan bilgiyi, tüm sisteme yayılmış küçük yük birimlerinin birleşerek oluşturduğu bir bütün evre temsil etmektedir. Ortama yeni bir bilgi aktarıldığında ise, yerel büyük bir değişiklik yerine tüm sistemde küçük bir değişiklik yapılmaktadır.

Günümüzde sinirsel ağ uygulamaları ya geleneksel bilgisayarlar üzerinde yazılım simülatörleri kullanılarak, veya özel donanım içeren bilgisayarlar kullanarak gerçekleştirilmektedir. Kredi risk değerlemesinden imza kontrolü, mevduat tahmini ve imalat kalite kontrolüne kadar uzanan uygulamalar yazılım paketlerinden faydalanılarak yapılmaktadır.

Bulanık Mantık :

Bulanık mantık (Fuzzy Logic) kavramı ilk kez 1965 yılında California Berkeley Üniversitesinden Prof. Lotfi A.Zadeh'in bu konu üzerinde ilk makallelerini yayınlamasıyla duyuldu. O tarihten sonra önemi gittikçe artarak günümüze kadar gelen bulanık mantık, belirsizliklerin anlatımı ve belirsizliklerle çalışılabilmesi için kurulmuş katı bir matematik düzen olarak tanımlanabilir. Bilindiği gibi istatistikte ve olasılık kuramında, belirsizliklerle değil kesinliklerle çalışılır ama insanın yaşadığı ortam daha çok belirsizliklerle doludur. Bu yüzden insanoğlunun sonuç çıkarabilme yeteneğini anlayabilmek için belirsizliklerle çalışmak gereklidir.

Fuzzy kuramının merkez kavramı fuzzy kümeleridir. Küme kavramı kulağa biraz matematiksel gelebilir ama anlaşılması kolaydır. Örneğin "orta yaş" kavramını inceleyerek olursak, bu kavramın sınırlarının kişiden kişiye değişiklik gösterdiğini görürüz. Kesin sınırlar söz konusu olmadığı için kavramı matematiksel olarak da kolayca formüle edemeyiz. Ama genel olarak 35 ile 55 yaşları orta yaşlılık sınırları olarak düşünülebilir. Bu kavramı grafik olarak ifade etmek istediğimizde karşımıza şekil deki gibi bir eğri çıkacaktır. Bu eğriye "aitlik eğrisi" adı verilir ve kavram içinde hangi değerin hangi ağırlıkta olduğunu gösterir.

Bir fuzzy kümesi kendi aitlik fonksiyonu ile açık olarak temsil edilebilir. Şekilde görüldüğü gibi aitlik fonksiyonu 0 ile 1 arasındaki her değeri alabilir. Böyle bir aitlik fonksiyonu ile "kesinlikle ait" veya "kesinlikle ait değil" arasında istenilen incelikte ayarlama yapmak mümkündür.

Bulanık mantıkta fuzzy kümeleri kadar önemli bir diğer kavramda linguistik değişken kavramıdır. Linguistik değişken "sıcak" veya "soğuk" gibi kelimeler ve ifadelerle tanımlanabilen değişkenlerdir. Bir linguistik değişkenin değerleri fuzzy kümeleri ile ifade edilir. Örneğin oda sıcaklığı linguistik değişken için "sıcak", "soğuk" ve "çok sıcak" ifadelerini alabilir. Bu üç ifadenin her biri ayrı ayrı fuzzy kümeleri ile modellenir.

Bulanık mantığın uygulama alanları çok geniştir. Sağladığı en büyük fayda ise "insana özgü tecrübe ile öğrenme" olayının kolayca modellenebilmesi ve belirsiz kavramların bile matematiksel olarak ifade edilebilmesine olanak tanımasıdır. Bu nedenle lineer olmayan sistemlere yaklaşım yapabilmek için özellikle uygundur.

Bulanık mantık konusunda yapılan araştırmalar Japonya'da oldukça fazladır. Özellikle fuzzy process controller olarak isimlendirilen özel amaçlı bulanık mantık mikroişlemci çipi'nin üretilmesine çalışılmaktadır. Bu teknoloji fotoğraf makineleri, çamaşır makineleri, klimalar ve otomatik iletim hatları gibi uygulamalarda kullanılmaktadır. Bundan başka uzay araştırmaları ve havacılık endüstrisinde de kullanılmaktadır. TAI'de araştırma gelişme kısmında bulanık mantık konusunda çalışmalar yapılmaktadır. Yine bir başka uygulama olarak otomatik civatalamaların

değerlendirilmesinde bulanık mantık kullanılmaktadır. Bulanık mantık yardımıyla civatalama kalitesi belirlenmekte, civatalama tekniği alanında bilgili olmayan kişiler açısından konu şeffaf hale getirilmektedir. Burada bir uzmanın değerlendirme sınırlarına erişilmekte ve hatta geçilmektedir.

Sanal Gerçeklik :

Sanal gerçeklik bilgisayar ortamında oluşturulan bir gerçekliktir ve cyberspace olarak da bilinir. Yapay zekanın bu alanında doğal gerçekliğe uygun, insan/bilgisayar arabirimlerinin kullanıldığı bir ortam oluşturulur. Sanal gerçeklik, gözlük ve stereo kulaklıktan oluşan başlık seti, vücut hareketlerini algılayan özel bir giysi veya eldivenden oluşan, çok algılayıcılı giriş-çıkış cihazlarına bağlı olarak oluşturulmaktadır. Böylelikle üç boyutlu sanal dünyayı görebilir ve dokunabilirsiniz. Sanal gerçeklik sizin bilgisayar benzetimli nesneler ve varlıklar ile etkileşim içine girebilmenize olanak sağlamaktadır.

Sanal gerçeklik uygulamaları geniş bir alana yayılmıştır. Bilgisayar destekli tasarımda (CAD), tıbbi teşhis ve tedavide, fiziksel ve biyoloji bilimlerindeki bilimsel deneyimlerde, pilot ve astronotların eğitimi için uçuş simülatörlerinde ve eğlence olarak üç boyutlu video oyunlarında kullanılmaktadır. CAD en geniş şekliyle endüstriyel sanal gerçeklik uygulamalarında kullanılmaktadır. Mimarlar ve tasarımcılar, ürünlerin ve yapıların üç boyutlu modelleri üzerinde test ve tasarım işlemleri yapmakta kullanırlar.

Mimarlar ve tasarımcılar, ürünlerin ve yapıların üç boyutlu modelleri üzerinde test ve tasarım işlemleri yapmakta kullanırlar. Bu teknoloji ayrıca ecza ve biyoteknoloji firmaları tarafından yeni ilaçların compüterize edilmiş davranışlarını geliştirmek ve gözlemlemek için kullanılmaktadır. Ayrıca doktorlar hasta vücudunun sanal bir modelinin oluşturulup sorgulanmasında faydalanmaktadır. Şimdi sanal gerçeklik uygulaması ile ilgili daha somut örnekler verilecektir.

- 1998 yılında kullanıma açılacak olan Paris yakınlarındaki stadyum, IBM Fransa tarafından yapımından önce sanal olarak inşa edilmiştir. Amaç tasarım aşamasında insan akınlarını ve onların davranışlarını analiz etmektir. Ayrıca sağlık ve güvenlik kuruluşlarını ihtiyaç duyulan yerlere yerleştirmek ve ziyaretçilere mümkün olduğunca konfor ve hareket serbestliği sağlayabilmektir. Bunların yanısıra müdahale olanaklarını ve etkilerini daha iyi tahmin etme imkanı olmaktadır. Gelecekte bu simülasyonun, havaalanları, resmi binalar ve alışveriş merkezlerinin tasarımında kullanılacağı belirtilmektedir.

- Almanya Frauenhofer Enstitüsünde, yolcuların uçuş korkusunu yenebilecekleri, yolculara yönelik ilk uçuş simülatörü gerçekleştirilmiştir. Bu proje, sanal gerçeklikle psikolojinin, fobilerin tedavisi için ilişkilendirilmesi fikrinden doğmuştur. Sanal ortama, yürüyen bir bant üzerindeymişcesine pencerelerin yanından geçip hafif eğimli olan kapıya vararak giriyorsunuz. Uçağa biniyor, doğru yeri buluyor ve oturuyorsunuz. Klima çalışıyor ve hoparlörlerden müzik sesi geliyor. Hafif bir sarsıntıyla uçak kapıdan ayrılıyor ve piste doğru yol alıyor. Makinelerden uğultulu bir ses geliyor, ivme sizi koltuğa bastırıyor ve Take-off. Yolcu, uçuşu, sanal gerçeklik kaskı ve kulaklık vasıtasıyla yaşıyor, gerçek uçuş duygusunu ise podestin altındaki performansı yüksek elektromotorlar sağlıyor. Uçuş deneye katılan yolcuların sorgulanması ile birlikte yaklaşık kırk dakika kadar sürüyor.
- Avrupa orjinli bir oto üreticisi firma, dağıtım masraflarının yüksek olduğunu düşünmekte ve bu nedenle Kuzey Amerika'daki dağıtım sistemini yeniden ele alıp olası iyileştirme olanaklarını değerlendirmek istemekteydi. Söz konusu firma, ABD dışındaki iki fabrikada ürettiği arabaları deniz ya da demiryoluyla ABD'de yer alan beş dağıtım merkezine göndermekteydi. Araçlar dağıtım merkezlerinden ABD'deki 52 değişik metropoliten pazara dağıtılmaktaydı.

Üretici firma, dağıtım merkezlerinden satıcı acentalara kadar olan ulaştırma maliyetlerinin, dağıtım merkezlerinin acentalara daha yakın yerlerde kurulmasıyla düşürülebileceğini savunmaktaydı. Bu arada, müşterilerin ilk tercihlerini hemen karşılama oranlarını yükselterek müşteri tatmini arttırılmak istenmekteydi. Bu sorunları çözebilecek, maliyet açısından etkin ve kabul edilebilir bir dağıtım sisteminin tasarlanması istenmekteydi.

VISION

İnsanın gözüyle yapabildiğini bilgisayara yaptırmayı amaç edinen bilim dalıdır. Bunun da bazı prensiplere göre yapılması gerekmektedir. Özellikle sanayi alanında kullanım alanı bulunmaktadır. Ekranda bulunan resim görüntü olarak adlandırılmaktadır. Bir bilgisayar ekranının çözünürlüğü ekrana x ve y ekseni boyunca sığabilecek piksel sayısıdır. Mesela 400*480 ' lik bir ekranın x ekseni boyunca 400, y ekseni boyunca 480 aldığını anlıyoruz. Buda bizde çözünürlük (rezülasyon) olarak geçer. Rezülasyon yüksek olursa hassasiyet artar pikseller küçülür daha net bir görüntü oluşur. Bir piksel koordinat ve rengiyle belirlenir. 3 Çeşit resim (görüntü) vardır;

1- Binary Resim:

Bu resim anlayışında siyah ve beyaz renk kullanılır. Bu filmler ya siyahtır yada beyazdır ara renkler ve grinin tonları kullanılmaz.

2- Gray Level :

Burada Binary resme göre ilave olarak ara renkler yani grinin tonları da kullanılır. Burada tonlar kodla ifade edilir ve her rengin bir kod olarak karşılığı vardır. Bu kodlar 0'dan başlar ve 255'e kadar devam eder. Genelde parlak beyaz 0 olarak alınır ve koyu siyahta 255 değerini alır diğer tonlar bu kod değerleri arasında bir kodla ifade edilir. Renklerin kod karşılıkları bu şekilde olabileceği gibi tam terside olabilir fakat en sık kullanılan sıralama biçimi yukarıda ifade ettiğimiz tarzdaki sıralamadır.

3- Renkli Resim:

Gerçek ve esas olan görüntü renkli görüntüdür ve burada milyonlarca tonda farklı renk vardır. Görüntü işlemede renkli görüntü işleme en zor iştir çünkü bilgisayarda renklerin tonunu tutturmak çok zordur. Çünkü burada ekranın çözünürlüğü ekran kartı gibi faktörler işin içine girmektedir ve bilinmektedir ki işlem sürecine ne kadar fazla etken katılırsa o kadar daha toplam hata miktarı artmakta ve mükemmelliğe ulaşma şansı azalmaktadır.

Gray level işlenen en kolay resimdir ve endüstriyel uygulamalarda fazlaca uygulama şansı bulur. Renkli resim pek fazla kullanılmaz sebebi de çok pahalı olması ve işlem yapmanın güçlüğüdür. Şayet bir uygulamada Binary resim yetiyorsa onu yetmiyorsa sırasıyla Gray Level ve daha sonrada Renkli Resim kullanılır.

GÖRÜNTÜ İŞLEMENİN TEMEL ADIMLARI

1- SENSİNG:

Kısaca alma yada görüntüyü elde etme işlemi olarak tanımlanabilir. Görüntüyü elde etmenin çeşitli yolları vardır; bunlardan en basiti sensör kullanmak ve sırasıyla fotoğraf makinası ve kamera kullanmaktır. Burada dikkat edilecek birkaç önemli husus vardır. Bunlar;

a) Bütün resimler aynı şartlar altında ve aynı aydınlatma ortamında elde edilmiş olmalıdır.
b) Aydınlatma yapılırken özellikle yansımaya dikkat edilmelidir. Yanlış yansıma kullanımında parça üzerinde yanlış resimler oluşabilir.
c) Görüntü ve ışığın açısı önemlidir, elde edilmek istenen bilgiye göre ayarlanmalıdır.
d) Sistem kurulurken sistemin hızıyla görüntü işlemenin hızı uygun olmalıdır.
e) Bir görüntüyü işlerken dikkat edilecek bir başka noktada flaş sayısıdır. Parçayı sıhhatli izleyebilmek için kaç kere fotoğrafının çekileceği önemlidir.

f) Gölgelere dikkat edilmelidir ve görüntü üzerine gölge düşmemelidir.

2- PREPROCESSİNG:

Ön İşleme adını verdiğimiz bu bölüm iki aşamadan oluşur. Bunlar sırasıyla Noise Reduction adını verdiğimiz gürültü giderme ile Edge Detection adını verdiğimiz kenar bulma aşamalarıdır.

a) Noise Reduction:

Gürültü resimde kirliliktir. Resimde olmaması gereken noktalardır. Bunu gidermek için gürültü giderme teknikleri geliştirilmiştir. Bu operatörler kullanılarak kirli resim verilip işlem sonunda gürültüsü giderilmiş resim almak mümkündür. Bunu bir örnekle açıklamak gerekirse diyelim ki bir ceket görüntüsü var ve ceket üzerinde bir gürültü var, bu bir toz lekesi olabilir. Ayrıca ceketin kenarları var ve burada da çeşitli renk tonları var. Ceketin resmi çekildiğinde daha koyu olacaktır. Gürültünün anlaşılması bir noktaya bakıldığında bu noktanın renk değeri olacaktır. 3x3 gürültü gidermek için nokta etrafında ki 8 noktaya matrisin orta noktasına bakılır. Eğer bu orta nokta gürültü değilse diğer 8 noktayla aynı renk olmalıdır. Farklı renkteyse rengi etrafına yaklaştırırız. Bunun için bunların her birinin rengine bakılır ve renk değerleri toplanarak 9'a bölünür. Aşağıdaki örnekte resimde olması gereken renk kodu 8 olsun ve maskın ortasındaki gürültünün renk kodunun 15 olduğunu düşünelim ve bu şartlar altında gürültü giderme işlemini inceleyelim.

8	8	8
8	15	8
8	8	8

$$\frac{8\times 8+15}{9}=\frac{79}{9}=8{,}5$$

8	8	8
8	8,5	8
8	8	8

İşlem sonucunda elde edilen gürültü giderme sonucu yeterli değilse tekrar aynı işlem yinelenir. Bu işlem tekrarlandıkça esas resme daha fazla yaklaşılır. Sonuçta elde edilen resim gittikçe açığa gidiyorsa rengin biraz daha açığı resim koyuya gidiyorsa rengin biraz daha koyusu

seçilir. Gürültü giderirken bazen farklı resimler elde edilebilir. Renkler değişebilir. Ana gürültü kesin azaltılmış olur. Bundan başka operatörlerde bulunmasına rağmen genel mantık şudur; Resmi sağdan sola, yukardan aşağıya tararız noktayı toplayıp böleriz. Resim Binary olduğunda ise iki renk vardır ve bu durumda eşik değer kullanılır. Bu değer deneme sonucunda ortaya çıkar. Bulduğumuz toplam sonuç eğer eşik değerinden büyükse siyah , küçükse beyaz yaparız. Bu maskların uygulanması için resmin kenarlarının bulunması gerekmektedir.

b) Edge Detection:

Bilgisayarlar bir resimden bizim anladığımız gibi sonuçlar çıkarır. Bilgisayara piksel, geometrik şekiller v.b gibi bilgileri anlatabiliriz. Elimizdeki resmi bizim geometrik şekillere çizgilere dönüştürmemiz gerekir. Biz gerekli bilgi çizgilerdir ve bu çizgiler kenarlarda saklıdır. Kenarlar bulununca gerekli resim elde edilmiş olunur. Buna kenar bulma denir. Kenar bulma için bazı operatörler geliştirilmiştir. Kenarlar kullanılarak güzel bir resim ortaya çıkarılır.

Kenar Bulma Operatörleri:

1- Gradient Operatörü:

g_x=

-1		+1
-1		+1
-1		+1

g_y=

+1	+1	+1
-1	-1	-1

- g_x ve g_y 3x3 ‘lük birer matristir.

$$g = \sqrt{gx^2 + gy^2}$$

g_x= -1.renk(x-1,y-1) -1.renk(x-1,y) –1.renk(x-1,y+1) +1.renk(x+1,y-1) +1.renk(x+1,y) +1.renk(x+1,y+1)

g_y= +1.renk(x-1,y-1)+1.renk(x,y-1)+1.renk(x+1,y-1)-1.renk(x-1,y+1)-1.renk(x,y+1)

-1.renk(x+1,y+1)

x-1,y-1	x,y-1	x+1,y-1
x-1,y	x,y	x+1,y
x-1,y+1	x,y+1	x+1,y+1

Örneğin; Qbasic'te şöyle bulunur.

g_x=-point(x-1,y-1)-point(x-1,y)-point(x-1,y+1)+point(x+1,y-1)+point(x+1,y)+point(x+1,y+1)

g_y=+point(x-1,y-1)+point(x,y-1)+point(x+1,y-1)-point(x-1,y+1)-point(x,y+1)-point(x+1,y+1)

2- Laplacien Operatörü:

renk=

	-1/4	
-1/4	1	-1/4
	-1/4	

renk=point(x,y)-(1/4)*[point(x,y+1)+point(x,y-1)+point(x+1,y)+point(x-1,y)]

Basic'te yazılışı;

```
SUB         use laplacien
SHARED      x,y,x1,y1,x2,y2
FOR         x=x1 TO x2+1
FOR         y=y1-1 TO y2+1
            İntensity=0
            İntensity   point(x,y)-(1/4)*[point (x,y+1)+point(x,y-1)
                        point(x+1,y)+point(x-1,y)]
            PSET        (x+x2-x1+2θ,y) , INT (intensity)
            NEXT        y,x
```

END SUB

- intensity = renk yoğunluğu
- PSET = en son bulunan rengi yerine koyar
- SHARED = istenen noktanın rengini bulmak için kullanılıyor.

Renk sonucu bulunduğunda o noktadaki renk değerine bulduğumuz rengi koyarız ve böylece o kenar bulunmuş olur.

4- Sabel Operatörü:

g_x=

-1	-2	-1
+1	+2	+1

g_y=

-1		+1
-2		+2
-1		+1

$$renk = \sqrt{g_x^2 + g_y^2}$$

g_x= point(x-1,y+1)+2*point(x,y+1)+point(x+1,y+1)-point(x-1,y-1)-2*point(x,y-1)
-point(x+1,y+1)

g_y= -point(x-1,y-1)-2*point(x-1,y)-point(x-1,y+1)+point(x+1,y-1)+2*point(x+1,y)
+point(x+1,y+1)

Basic'te yazılışı;

SUB use Sabel
SHARED x,y,x_1,y_1,x_2,y_2

```
            FOR x=x1   TO   x2+1
            FOR y=y1-1    TO   y2+1
            gx=0
            gx= point(x-1,y+1)+2*point(x,y+1)+point(x+1,y+1)-point(x-1,y-1)
                -2*point(x,y-1)-point(x+1,y-1)
            gy=0

            gy=-point(x-1,y-1)-2*point(x-1,y)-point(x-1,y+1)+point(x+1,y-1)
                +2*point(x+1,y)+point(x+1,y+1)
            intensity=INT [ SQR ( gx^2+gy^2)]
            PSET (x+x2-x1+2θ , y) , intensity
        NEXT y
    NEXT  x
END   SUB
```

5- Robert Operatörü:

2x2 'lik bir operatördür.

g_x=

x,y	x+1,y
x,y+1	x+1,y+1

g_x=

	+1
-1	

g_y=

+1	
	-1

$$renk = \sqrt{g_x^2 + g_y^2}$$

g_x= point (x+1, y) – point (x, y+1)

g_y= point (x,y) – point (x+1 , y-1)

Basic'te yazılışı;

```
SUB          use Roberts
SHARED       x,y,x1,y1,x2,y2
     FOR     x=x1  TO  x2+1
             FOR   y=y1-1   TO   y2+1
             gx=0
             gx= point (x+1,y) – point (x, y+1)
             gy=0
             gy= point (x,y) – point (x+1,y+1)
             intensity = INT  [ SQR ( gx^2 + gy^2 )]
             PSET  ( x+x2-x1+2θ , y) , intensity
     NEXT y
NEXT x
END  SUB
```

3- SEGMANTATİON:

Resmi (görüntüyü) anlamlı bölgelere ayırma işlemidir. Elimizdeki görüntü veya resim iki renkli olursa resmi siyah veya beyaz iki bölgeye ayırırsak istediğimiz şekilde işleyebiliriz. Resmi anlamlı bölgelere ayırırsak işlemesi daha kolay olacaktır.

4- RECOGNİTİON:

Resmi tanıma işlemidir. Resmi bölgelere ayırınca acaba bunu nasıl tanıtalım sorusu geliyor. Resmi tanırken bunu bilgisayara geometrik şekiller olarak tanıtmak gerekmektedir.

6- INTERPRETATİON:

Resmi ve görüntüyü yorumlama işlemidir. Resme etiket tanımlamak yani resmi yorumlamak gerekir. Örnek olarak küre dediğimizde acaba bu dünyamıdır, bir bilyemidir gibi sonuç olarak görüntü üzerinde nihai yorum yapılmalıdır ve yorumlanmalıdır.

ENDÜSTRİYEL GÖRÜNTÜ İŞLEME

Bir fabrikada üretilen mamüllerin gözle kontrolünün bazı dezavantajları vardır. İnsanın performansı her zaman aynı değildir. Ayrıca moral , hastalık ve benzeri ruhi durumlar kişinin performansına direkt olarak yansır. Sürekli aynı işi yapmak kişiyi sıkmakta ve çabuk yorulmasına , dikkatinin dağılmasına sebep olmaktadır. Özellikle seri üretim yapan bir fabrikada birim zamanda üretilen mamül sayısının çok fazla olduğu durumlarda mamüllerin tümünün gözle kontrol edilmesi hem çok zaman almakta hem de pahalı olmaktadır. Bu bakımdan örnekleme yoluyla tüm mamüllerin kontrolü yerine seçilen belirli sayıdaki örnek mamül kontrol edilmekte ve elde edilen sonuçlar genelleştirilmektedir. Bu ise kalitenin gittikçe önem kazandığı günümüzde %100 kalite kontrol yapan yöneticilere karşı rekabet gücünü oldukça zayıflatacaktır. Göz ile kontrol yavaş olduğundan birim zamanda üretilen mamül sayısı az olmaktadır. Bunun yerine hızlı bir E.G.İ (Endüstriyel Görüntü İşleme) sistemi kurulduğunda hız artacağından birim zamanda üretilen mamül sayısı artacaktır. Bu da daha fazla kar demektir.

Bir Endüstriyel görüntü işleme sistemi için temel donanım şudur;

a) Görüntüyü elde etmek için bir sensör (genellikle bir T.V kamera)
b) Bu görüntüyü sayısal hale getirecek bir sayısallaştırıcı
c) Genel amaçlı veya işe göre imal edilmiş bir işlemci
d) Bir aydınlatma sistemi

Elde edilen görüntü sayısallaştırıcı tarafından sayısal hale dönüştürüldükten sonra uygun bazı teknikler kullanılarak işlemci tarafından değerlendirilmekte ve karar verilmektedir. İşlemci sensörün focusunu ayarlamak , cismin hareketini yavaşlatmak, aydınlatmayı ayarlamak vb birtakım kontrollere sahiptir. Bir E.G.İ sisteminden beklenenler şunlardır;

1) Çok düşük hata seviyesi
2) Yüksek performans
3) Güçlü ve kullanışlı bir karar verme mekanizması
4) Düşük kuruluş ve işletme maliyeti
5) Sistem esnek olmalı sadece bir ürün tipi için olmamalıdır.
6) Kolay adapte edilebilir olmalı. Birim zamandaki ürün sayısı veya ürünün şekli , boyutları değiştiğinde değişime kolayca uyum sağlayabilmelidir.
7) Üretim hattında kolay ve hızlı şekilde kurulmalı ve farklı ürünler için gerektiğinde kolay ve hızlı bir şekilde değiştirilebilmelidir.

E.G.İ 'nin Uygulama Alanları

1) Gıda sanayinde (Örneğin Bisküvi kontrolünde)
2) Bir üretim hattında metal parçaların kontrolü (metal silah kılıfı, mermi vb.)
3) Rulo halinde üretilen mamüllerin yüzey kontrolü
4) Askeri uygulamalar.

Maliyet

Bir E.G.İ sisteminin maliyeti için sabit bir rakam vermek imkansızdır. Çünkü maliyet kurulacak sistemin özelliklerine , kullanılacak donanımın cinsine ve sistemin ne için kullanılacağına göre değişmektedir. Bir fikir vermesi açısından aydınlatma sistemi, optik gereçler, kamera, gerçek zamanlı işlemci bordu, merkezi işlem bordu, iki adet yüksek hızlı sinyal işleme bordu, güç kaynağı, monitör, klavye vs. için yaklaşık olarak 40.000 $ gerekmektedir.

E.G.İ 'nin Avantajları

Kalite kontrolde kullanılan işçilerin maliyeti yüksektir. Bir E.G.İ sistemin maliyeti yüksek görünmekle beraber kısa sürede kendini amorti edeceği beklendiğinden maliyeti düşecektir. Ayrıca işçilerin maliyetinin gittikçe artmasına karşılık böyle bir sistemin maliyeti ise gittikçe azalacaktır. İnsanlar böyle bir sisteme göre çok yavaş kalmakta ve hata oranı daha yüksek olmaktadır. Böyle bir sistemin hızı ve kontrol kabiliyeti daha yüksek olacaktır. Endüstriyel görüntü işlemenin diğer avantajları ise şunlardır;

a) İnsanları rutin ve sıkıcı işlerden kurtarır.
b) İşçilik maliyetlerini ve işçilik giderlerini düşürür.
c) İnsanların çalışmasının zararlı ve tehlikeli olduğu ortamlarda insanların yerine kullanılabilir. Mesela ilaç sanayi veya demir çelik endüstrisinde .
d) Tecrübeli ve uzman kalite kontrol elemanlarının sayısını azaltır. Çünkü bu elemanların maliyeti yüksektir.
e) Bu sistemlerle kontrol yapıldığında sadece parçanın iyi (sağlam) veya kötü (bozuk) olmasının tespit edilmesinden öte kontrol sırasında istenen bazı parametre veya kaydedilmesi suretiyle ilerde daha iyi bir üretim planının yapılmasında bir analiz yapma imkanı da sağlanmaktadır.

Görüntü işleme yöntemleri;

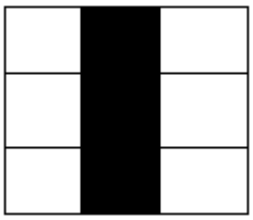
1) Resim çıkarma yada görüntüleri birbirinden çıkarma tekniği. Diyelim ki elimizde endüstriyel bir görüntü var ve bu görüntü sabit bir malzemenin resmi olacağı için fotoğrafını

çekmemiz gerekir. Resim standart olmalıdır ve resim çekildikten sonra bilgisayara kaydedilir. Kaydettikten sonra yeni resim geldiğinde bu görüntüyü elimizdeki , kaydettiğimiz referans resimle karşılaştırıp bunları birbirinden çıkarmak gerekir. Birbirinden doğru çıkarmak için resimleri önce üst üste koyarak tamamen çakıştırmak lazımdır ki bu işlem oldukça zordur. Bu resimlerin çakışan yerlerini çıkarırız geriye fark kalır. Daha sonra yüzde olarak belli bir eşik değeri koyarız, elde ettiğimiz fark eşikten büyükse parça hatalıdır eşikten küçükse mamül kabul edilebilir durumdadır.

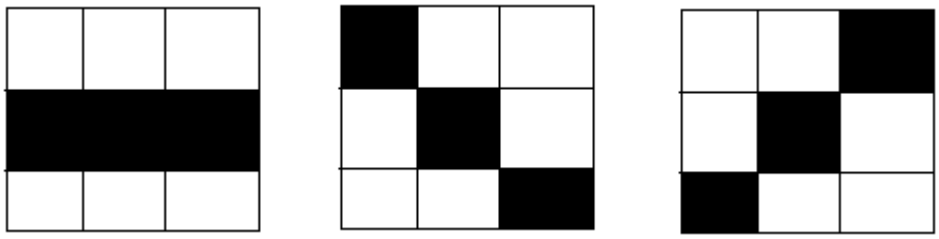
2) Geometrik olarak resmi tanımlarız. Burada zorluk kaç tane geometrik özelliği nasıl kullanacağımızdır

3) Mask tekniği;

Görüntüyü soldan sağa, yukarıdan aşağıya bazı masklarla tarayalım. Her bir maskın kaç kere görüntüde tekrar ettiğini bulalım ve bilgisayarın anlayacağı şekilde dijite edelim.



Bunu şeffaf kabul edelim bu maska ‘3x3 piksel görüntü maskı’ denir. Görüntü maskı düz çizgileri tarar. Burada 3 tane pikseli tarar.



Başka mask çeşitleri;

Bununla beraber bütün masklarda orta piksek doludur. Resmi sağdan sola, yukarıdan aşağıya bunlarla tararız ve köşeleri de elde ederiz.

Resmi tararken kaç çeşit mask yeterli ise o kadarını kullanırız. Genelde 20 tanesi yeterli olmaktadır. Yukarıdakiler 3x3’lük masklardır. Daha büyüğünü yapmak istersek 5x5 veya 7x7 ‘lik masklarda kullanılmaktadır. Yeterli ise az olan kullanılmaktadır. Çünkü büyük masklar kullanılırsa her seferinde örneğin 7x7 ‘lik mask kullanılırsa 7x7=49 noktayı incelemek

gerekmektedir. 3x3=9 noktayı incelemek ise daha basit ve daha hızlıdır. Örneğin tıpta kullanılacaksa hassasiyet önemlidir ve hızlı olmalıdır. Bunun için 5x5 'lik yeterli olur.

Gayemiz şu;

Bilgisayar görüntüyü direk olarak anlayamaz. Görüntüyü bilgisayarın anlayacağı hale getirmeli yani dijite etmeliyiz. Dijite edilmiş bilgiyi kullanarak bilgisayarın karar vermesini sağlayalım. Giriş ve çıkış analogtur. Ama sonuçta biz bilgilerle dijital ortamda uğraşacağız. Her bir maskın görüntüde görüntüyü işlemesi için birer birer taradığımızda üst üste koyup çakışıp çakışmadığına bakarız. Görüntü işlenmeden önce gürültü giderilir. Kenarları bulup iki renkli (binary) hale getiririz. Daha sonra bir maskın uyması demek ortanın dolu yanların açık olmasıdır. Maskın uyup uymadığını anlamak için maskın frekansını buluruz.

Frekans = $\Sigma\Sigma f^{m}_{xy}$

X=1 iken y= 300 kere dönüyor.

Y= 1 iken x= 300 kere dönüyor.

Maskı alıp resmin üzerine koyarız. Maskı koyduğumuzda dolu olan kısımlarda dolu, boş olanlarda boş kısımlar uyuyor mu bakarız. Orada nokta varsa çakışıyorsa uyuyordur. Bu durumda mask 1 ise maskın frekansını bir arttırırız. Sonunda bütün resmi bununla tararız. Tarama yaparken önce soldan sağa hepsi taranır. Sonra bir aşağı satıra inip tekrar bütün satır taranır. Daha sonra yukardan aşağıya aynı şekilde tarama işlemine devam edilir. Bu şekilde görüntünün kaç defa tekrar ettiğini buluruz. Yani görüntüyü elde ederiz. Önce sağdan görüntüyü bununla taratıp her biri için bir değer buluruz. Bunu bilgisayara tanıtırız.

$m_1, m_2, m_3, m_4, m_5, m_6, m_7$.....
şeklinde frekansı tespit ederiz.

$m_1=10$ $m_2=20$ $m_3=30$ $m_4=400$ $m_5=500$ $m_6=0$ .....

Böylece görüntüyü elde etmiş olduk. Bizde farklı 20 tane görüntü var. Mesela bu şekilde 20 tane vektör elde ederiz. Bunların her birini bilgisayara öğretmek için endüktif öğrenme kullanırız.

İşlenecek görüntü aralığa gelirse tanıyacak yani aralık bulacak. Kuralları çıkarıp bilgi tabanı elde ederiz. Yani vektör elde ederiz.

UZMAN SİSTEMLER

1 Tanım

Bir uzman sistem, insan uzmanların becerilerini gerektiren problemlerin çözümü için bilgileri, olayları ve sorgulama tekniklerini kullanan bilgisayar tabanlı bir sistemdir. Bu sistemler, belli donanım veya yazılım konfigürasyonları için dizayn edilebilir. Ya da genel amaçlı bir bilgisayar üzerinde çalışabilecek yazılım sistemleri olarak tasarlanabilir.

Uzman sistemlerin kullandığı bilgiler, ya kurallardan ya da belli bir ana konu ile ilgili bileşenlerin davranışları hakkındaki deneyim bilgisi ile teşkil edilir. Kurallar genelde, belli bir durum tanımlarını verir. Örneğin; "Eğer bir futbol takımı teknik ve kondisyon açısından iyi durumda ise ve eğer kaliteli bir teknik adama sahipse ve eğer dengeli bir yönetici kadrosu mevcutsa, o futbol takımının maçlarını kazanması ve ligde şampiyon olması kuvvetle muhtemeldir". Deneyim bilgisi, belli bir konu üzerindeki tecrübelerin birikimi olarak yorumlanabilir.

Uzman sistem tasarımcılarının bu sistemleri geliştirmede gözettikleri esas ilkeler arasında; o an için mevcut olmayan bir uzmanın yerini alabilmesi, birçok insan uzmanın bilgi ve tecrübe birikimini bir araya getirebilmesi, yeni uzman adaylarını eğitebilmesi, uzmanları ilgilendirmeyen veya onlara cazip gelmeyen veya uzmanların pahalı olduğu projeler için gerekli uzmanlığı sağlamak gelmektedir.

Uzmanlar özellikle birden çok ana-konuda ihtisaslaşmışlarsa, birçok uzmanın bilgi ve tecrübe birikimini bir araya getirmek o kadar kolay bir iş değildir. Uzman sistemler, herhangi bir uzmanın yapabileceğinden çok daha fazla bilgi ve olayı depolayıp kullanabilirler. Bu durum, uzman sistemlerin, insan uzmanlarla yapılamayacak çıkarım işlemlerinin kolaylıkla altından kalkabileceklerini göstermektedir. Uzman sisteme sağlanan bilgiler, belli bir konudan veya birbiri ile yakından ilişkili birkaç konudan sağlanabilir. Fakat her halde uzman sisteme dahil edilen bilgi, ayrı ayrı sistem bütününe katkıda bulunan her bir uzmanın bilgi birikiminden daha fazla olacaktır.

Geleceğin uzman adayları, belli hususlarda eğitim görmek ve belli olayların tecrübesini yaşamak durumundadır. Bir şirketin belli bir konuda uzmana ihtiyaç duyması halinde, uzman sistemler oldukça etkin bir biçimde eğitime destek birimleri olarak kullanılabilir. Eğer belli bir konuda bir uzman sistem mevcutsa, uzman adayı sisteme bir problem sunup cevabını isteyebilir. Bunun üzerine sistem gerekli çözümü sunacaktır. Eğer uzman adayı, sunulan çözümü anlarsa,

sisteme yeni bir problem verir. Eğer anlayamaz ise sistemden sonuca nasıl ulaştığını ya da işlem için kullandığı yöntemin ne olduğu hakkında açıklama isteyebilir. Bunun üzerine sistem çözüme ulaşırken kullandığı kural ve olaylar zinciri ile ilgili detaylı bilgiyi verecektir.

2. Uzman Sistem Bileşenleri

Uzman sistemler, kendilerinden beklenenler doğrultusunda, birçok farklı yolla yapılandırılabilir ve bu farklı yapı mimarileri değişik bileşenler içerir. Bununla beraber, belli bazı unsurlar birçok uzman sistemde ortak özellik olarak göze çarpmaktadır. Bunlar; kullanıcı arabirimi (user interface), bilgi tabanı (knowladge base), bilgi kazanım modülü (knowledge aquisition module) ve karar verme mekanizması (infrence engine) dır.

Kullanıcı arabirimi, sistem kullanıcısı ve sistem (ya da sistem grubu) arasında iletişim, aktarım ve değişimi (bilgi, görüş, yorum, veri vs.) sağlayan yazılımdır. Bunun vasıtası ile, sistemin ana-uğraş konusuna ilişkin belli bir durum hakkındaki olaylar ve veriler, kullanıcı tarafından girilebilir ve kullanıcı uzman sisteme söz konusu alanda sorular yöneltebilir.

Uzman sistem bilgi tabanı, özel bir konu hakkında, uzman-seviyeli bilgi içermektedir. Söz konusu bilgi, bir veya daha fazla insan uzmandan elde edilmekte ve uzman sistem tasarımına mahsus olan bir bilgi sunum formunda saklanmaktadır. Aslında uzman sistemler, bir bilgi veri tabanı ihtiva etmeleri nedeni ile çoğu kere ve çoğu yerde bilgi-tabanlı sistemler olarak algılanmakta ve kabul edilmektedir.

Bilgi kazanım modü/ü, insan uzmanlardan bilgi toplamak amacı ile, uzman sistemler ve uzman bireyler arasında diyalog kurulmasını sağlayan yazılımdır. Bu birim, elde edilen bilgiyi, sistemin veri tabanına yerleştirir. Uzman sistem ve uzman birey arasındaki arabirim, bazen kullanıcı arabirim olmakta, bazen de bilgi-tedarik usulüne özgü olarak yapılandırılmaktadır.

Karar Verme mekanizması, uzman sistemin, sahip olduğu veriler ve imkanlar dahilinde sonuçlara erişme sırasında kullanıldığı mantık sürecini sağlayan yazılımdır. Bu çoğunlukla, (inference engine) olarak adlandırılır. Mekanizma, yeni bilgiler oluşturmak veya bir sorunun cevabına erişebilmek için, uzman sistemin veri tabanından ya da kullanıcı tarafından sağlanan verilerden yararlanır.

Yukarıda bahsedilen temel uzman sistem bileşenleri, değişik yollarla tasarlanmış ve uygulamaya

sokulmuştur. Bazen koşullara ve beklentilere göre, özel tasarımlı bileşenler geliştirilmiştir. Tüm bu bileşenlerin farklı bileşimleri, muhtelif uzman sistem(US) mimarilerinin gelişmesine yol açmıştır.

3. Belirgin Karakteristikler

Bu sistemler, insan uzmanların bilgilerinin, bir bilgisayar içerisine toplanmasına, belli formatlarda depolanmasına ve başkalarının bu bilgiye erişmelerine imkan tanır. Düzen, disiplin ve paylaşım sistem kullanımında ilk ortaya çıkan ilkelerdir. Uzman sistem teknolojisi bir süredir üniversite araştırma-geliştirme merkezlerinde kullanılmakla beraber, bu sistemlere dayalı ticari ürünlerin ortaya çıkması hayli yenidir.

Tanımlar ve tavsiyeler ne kadar farklı olursa olsun, bir sistemin uzman sistem olarak hüviyet kazanabilmesi için, şu özelliklere sahip olması gerektiği ileri sürülebilir:

a. Sistem,bir uzman potansiyeli ve yeterliğinde fonksiyon görme/idir. Uzman sistem ler, sistemin bilgi bankasından edinilmiş özel bilgilere sahip olmalı ve bunları, çıkarımlar geliştirebilmek üzere kullanıma sokabilmelidir. Sahip olunması gereken ve bilgi-bankasında saklanacak bilgiler, uzman sistemin hizmet sunduğu ilgili konuda uzman olan bireylerin birikimlerinden yola çıkılarak teşkil edilmelidir. Bilgilerin, sistemin sonuçlara erişebilmesine imkan verecek yeterlilik ve anlaşılabilirlik seviyesinde olması gerekir. Öyle ki, varılacak sonuçlar veya çıkarılacak yeni bilgiler, sistem kullanıcısında bulunmalı ve ona yardımcı olmalı (aksi takdirde, belli bilgileri hatırlamak ve karmaşık sayısal işlemleri yapmak için uzman sistemler kullanmanın bir anlamı yoktur) ve ayrıca benzeri çıkarımları, sadece ilgili alanda uğraş veren bir uzman kişi yapabilmelidir. Dolayısıyla uzman sistem, hizmet verdiği tüm sistemin faaliyet gösterdiği ana-uğraş alanında bilgi kapasitesi, işlem hızı ve çıkarım gücü ile şekillenen uzmanlık sahibi olmalıdır. Aşağıda özetlenen iki karakteristik, bu ilk özelliğin sonucu olarak yorumlanabilir.

b. Bır uzman sistem, genelden öze/e indirgeme veya eriş/m oluşturabilmek ıçin, bir karar verme mekanizması kullanır. Söz konusu mekanizma, çoğunlukla uzman sistemin karar verme mekanizması (inference engine) veya kontrol yapısı olarak kabul edilir. Bir uzman sistem, en ahenkli ve isabetli hükmü çıkaracak şekilde mantık yürütmesine fırsat veren çıkarım kontrol yapısına sahip olmalı ve geliştirdiği çıkarımları, karşı tarafça kabul edilebilir formda ve inandırıcılıkta sunmalıdır.

c. Bir uzman sistemin uzmanlık birikimi, salıibi olduğu ve edindiği özel bilgi üzerine kuruludur.

Bu sistemler, insan uzmanlar tarafından sağlanan bilgi ve tecrübe birikimini kullanırlar. Söz konusu birikim, ana-uğraş konusu hakkındaki olayları ve yol gösterici verilen kapsar. Çıkarım ünitesi tarafından yönlendirilen ve kontrol edilen uzman sistem bilgileri, bilgi tabanında (knowledge base) depolanır. Bu banka uzman sistemin tipik bir karakteristiğidir.

Sonuç olarak, değerlendirmeye tabi tutulacak bilgilerin incelenmesi, değiştirilmesi ve gerektiğinde kapsamının geliştirilmesi çok kolaylaşır

karakteristiği de, *sistemin kendisine sunulan bilgi/en nasıl ku//anılacağına dair bır programın veya program/ama yapısının bu/unmamasıdır.* Sistemin çıkarım kontrol mekanizması, kendisine sağlanan bilgiyi nasıl ve ne zaman kullanacağının kararını verecektir.

Yukarıda sayılan özellikler, uzman sistemlerin, diğer bilgisayar tabanlı sistemlerden ayrımını yapmada yardımcı olmaktadır.

Pek çok uzman sistem de, tıpkı insan uzmanlar gibi, eksik ve belirsiz verilerle uğraşırlar. Farklı uzman sistemler, bu tip verilerle ilgilenmede değişik yollar takip ederler.

Bir uzman sistem, *belıısız bilgileri,* belli bir kesinlik (doğruluk) faktörü ile kabul eder. Bu faktör; uzman şahsın, sahibi olunan veri ya da bilgilerin doğruluk derecesi hakkındaki fikrinin resmi olmayan ölçüsüdür. Kesinlik faktörleri, yapılan çıkarımlar neticesinde tayin edilir ve her yeni çıkarım sonrasında yenilenir. Faktörler, sonuçlarla beraber sunulur.

Eksik bilgi, birçok uzman sistem tarafından basit şekilde karşılanır. Bir uzman sistemin, bir sonuç elde edebilmesi veya çıkarım yapabilmesi için ihtiyaç duyduğu bir olay veya önerme eksik kalırsa, gerekli bilgiyi sağlaması için kullanıcıyı uyarır. Eğer gerekli bilgi verilirse, uzman sistem işlemini devam ettirir. Eğer kullanıcı gerekli bilgiyi sağlayamazsa, uzman sistemin çıkarım mekanizması, geliştirmeye çalıştığı sonucu ortaya koyma işlemini sona erdirip, farklı bir mantık sürecini uygulamaya koyarak çıkarımda bulunmaya çalışır. Eğer sunulan bilgide büyük miktarda yetersizlik varsa, sistem mevcut problem için çözüm bulma işlemini askıya alarak, kendisine yöneltilen bir sonraki probleme geçer.

Rasgele (Random) olmayan bir usulde, birçok alternatif arasından çözüm veya hedef bulmaya yönelik arama yeteneği, *akdhı problem çözme a/goritmasının* belirgin bir özelliğidir. Uzman

sistemler, bir dizi arama tekniğinden faydalanırlar. Örneğin heuristik (bulmaya, anlamaya ve keşfetmeye yarayan) arama tekniği kullanan bir uzman sistem, tüm ihtimallerin aranması işlemi yerine, baş parmak kurallar zincirini kullanarak dikkate alınacak ihtimaller kümesini azaltır ki, bu yöntem tipik olarak bir uzman şahsın belli bir konudaki problemin çözümü sırasında takip edeceği usuldür.

Bir kullanıcı ile bir uzman sistem arasındaki konsültasyon sırasında, kullanıcıdan, uzman sistemin belli bir konuda açıklama yapması isteği gelebilir. Uzman sistemler genelde tipik olarak interaktif (karşılıklı etkileşime müsait) yapıda olup, eriştikleri hemen hemen bütün neticelerin haklılığını ortaya koyabilmektedirler. Hatta bir çalışma veya uygulamanın tam ortasında verdikleri ara sonuçlarla oldukça değerli bilgiler sağlayabilmektedir.

4. Temsili Mimari

- **a.** Kurallar Halinde
- **b.** Çatı (frame)
- **c.** Anlamlı ağ (Semantic net)

Farklı uzman sistem bileşenler ile, değişik uzman sistem mimarileri oluşturulur. Bu mimarilerden en göze çarpanı temsili mimarıdır. Günümüz uzman sistem mimarileri, bilgisayar tabanlı bir sistem desteğinde bilgiler ile yakından ilgili olan bireylerin, bilgiyi nasıl temsil edecekleri ve akıllı-karar-verme (intelligent decision making) görevlerini nasıl yerine getireceklerini algılama biçimlerini yansıtmaktadır. Laboratuvarlardaki deney çalışmaları ve endüstrideki pratik uygulamalar yoluyla, bilginin nasıl temsil edileceği ve kullanıma sokulacağı hakkında gün geçtikçe ortaya yeni yaklaşımlar konulmaktadır. Mimari yapıların temel nitelikleri artık iyice belirginleşmeye başlamıştır.

Kullanıcı arabirimi; sistem kullanıcısının belli bir durum veya konu ile ilgili olarak kurallar ve olaylar girmesine ve sisteme sorular sormasına imkan verir. Kullanıcı ihtiyaçlarına çözümler bulur. Sistem ve kullanıcı arasında her türlü iletişimin kurulmasına destek verir.

Bilgi bankası; bir uzmanın belli bir konu hakkındaki bilgilerini, kodlanmış bir format da ihtiva eder. Söz konusu kodlama, doğal olarak, kolay okunabiiir ve anlaşılabilir yapıdadır. Bilgi bankaları, belli konuların uzmanları tarafından hazırlanmakla beraber, içerikleri, ana-konu hakkında malumat sahibi olan herhangi biri tarafından anlaşılabilir olmalıdır.

Bilgi Kazanım birimi; yeni olaylar ve yorumlar oluşturabilmek üzere, bilgi bankası ve kullanıcı tarafından kendisine sağlanan bHgileri kullanır. Bunu yaparken bir uzmanın dedüktif (tümdengelimci) düşünce tarzını takip eder. Sistemi geliştiren olaylar, sistem kullanıcısının

ihtiyaç duyduğu tavsiyeler olabilir.

Basit uzman sistem mimarisinin yapısı geliştirilebilir. Yaygın olan bir genişletme yolu, veri bankasının, bir bilgi veritabanı ve bir ana veritabanı haline getirilmesidir. Bu iki veritabanının yönetimi, bir veritabanı yönetim sistemi vasıtası ile gerçekleştirilir.

Bilgi veritabanı, belli bir konuya ait bileşenlerin davranış ve karekteristlikleri ile ilgili kuralları içerir. Bu kurallar, uzman sistemler tarafından kullanılabilmelerine imkan sağlayacak usulde şekillendirilirler. Bilgilerini, bu kural formatı içinde temsil eden/sunan uzman sistemler, çoğunlukla kural-tabanlı sistemler olarak adlandırılır.

Ana-veritabanı, uzman sistemin hizmet sunduğu konu hakkındaki olayları içerir. Daha etkin veri kullanımı ve yönetimi sağlamak için, muhtelif biçimleme tekniklerini uygulamaya koyar.

Uzman sistemlerin birçoğundaki veriler ve bilgiler, sistem ömrü boyunca güncelleştirilecek ve gerektiğinde revize edilerek genişletilecektir. Bu kullanıcının, kendisine en güncel ve eksiksiz yardımın temin edildiği hususunda ikna edilmesi için yapılır. Bu amaçla birçok uzman sistemin, benzeri bir güncelleme işlemi gerçekleştiren modülü bulunur. Söz konusu modüle veya modüller grubuna bilgitedarik ünitesi adı verilir. İlgili ünite, ana veritabanı tarafından, uzman şahıstan bilgisayar-tabanlı uzman sisteme bilgi transferinin kolay ve seri kılınması amacı ile kullanılır.