YAPAY ZEKA

* Yapay zeka ilk defa 1950 yılında Alan Turing tarafından ortaya atılmıştır. O zamandan bu zamana 4 tane yapay zeka tanımı yapılmıştır.

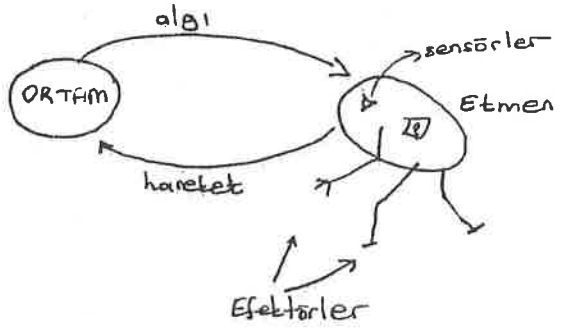
1. İnsan gibi düşünen sistemler
2. İnsan gibi hareket eden sistemler
3. mantıklı bir şekilde düşünen sistemler
4. Mantıklı bir şekilde hareket eden sistemler

- Ajan (Agent)
- Robot
- Aracı
- Etmen

Yapay zekanın oluşturulması için:

⇒ 1. Doğal Dil işleme (Natural Language processing)
2. Bilgi Temsili (Knowledge Representation)
3. Otomatikleştirilmiş muhakeme (Automated Reasening)
4. Makine öğrenmesi (Machine Learning)

* Socrates bir insandır → Bütün insanlar ölümcüldür → Socrates ölümcüldü
mantıklı bir şekilde düşünen sistemlere örnektir.



Etmen Çeşitleri

1. Basit refleks etmenler: Bir olay karşısında verilen ani tepki
2. Dünyanın durumunu izleyen etmenler:
3. Amaç tabanlı etmenler
4. Fayda tabanlı etmenler

ORTAMLAR

1. Erişilebilir: Eğer bir etmenin sensörleri ona ortamın tam durumuna erişim verirse ortam erişilebilir.
2. Belirli: Eğer ortamın sonraki durumu şu anki durum ve etmenin hareketine bağlı olarak tahmin edilirse belirlidir.

3. Statik- Dinamik: Eğer müzakere ederken ortam değişirse ortam etmen için dinamiktir.

4. Ayrık- Sürekli: Algı ve hareketlerin sayısı belirli ise ayrıktır.

	Erişilebilir	Belirli	Statik	Ayrık
Santranç	✓	✓	✓	✓
Poker	×	×	✓	✓
Tavla	✓	×	✓	✓
Taksi sürme Etmeni	×	×	×	×

→ yapacağın hareketlerin sayısı belirli ise ayrıktır.

Arama ile Etmen Güzme

Örnek Problemleri

1. Toy Problemleri: Oyuncak, gerçek olmayan problemler

- 8 puzzle problemi

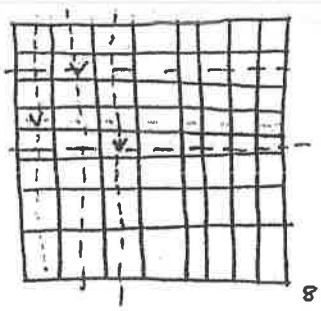
5	4	
6	1	8
7	3	2

⇒ amaç ortadaki boş duruma getirme

1	2	3
8		4
7	6	5

- Durum
- Hareket
- amaç
- Maliyet (Path Cost)

2. 8 vezir problemi



8 vezir öyle yerleştireceksinki hiçbiri birbirini yemiyecek.

8x8 matris üzerinde santranç oynanır.

Gerçek Dünya Problemleri

1. Rota bulma problemleri
   Navigasyonlar

2. Gezgin Satıcı problemi

Bir noktadan satıcı başlıyor şehirleri gezmeye ve bütün şehirlere uğrayacak ama sadece bir defa başlangıç noktasına döndüğünde kat etdiği mesafe minumum olmalı

3. VLSI Tasarımı:

Amaç minumum alan işgal edecek şekilde devreleri yerleştirmek.

<u>NOT</u>: Bu problemler arama ile çözüm bulunmuştur

Arama Stratejileri : 4 kritere göre değerlendirilir.

1. Tamlık (Completes): Eğer bir tane çözüm varsa onu bulmayı garanti eder.

2. Zaman Karmaşıklığı: Çözümü bulmak ne kadar zaman alır
   (Time Complexity)

3. Hafıza Karmaşıklığı: Arama yapmak için ne kadar hafıza gerekli
   (Space Complexity)

4. Optimallik: Birkaç farklı çözüm varsa strateji en kaliteli çözümü bulur mu?

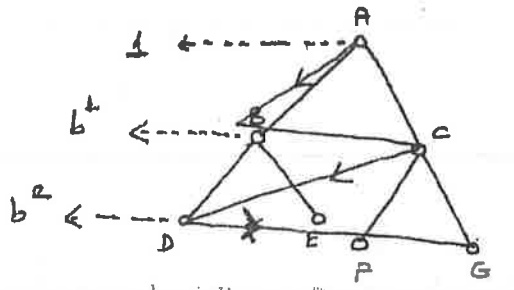
Arama Yöntemleri (Stratejileri) ikiye ayrılır.

1. Bilgili Arama (Informed Search): Arama yaparken size bilgi verilir.

2. Bilgisiz Arama (Uninformed Search): Arama yaparken size bilgi verilmez.

Bilgisiz Arama Yöntemleri

1. İlk Önce [illegible] arama (Breadth - First - Search):



Hafıza: A B C D E F G

Kuyruk: ~~B C D E~~

⇒ önce B hafıza gider kuyruktan ve kuyruğa B'nin çocukları yazılır daha sonra sıra C'ye gelir aynı işlem bitene kadar devam eder ve seviye seviye arama yapar.

BFS Tamdır

Toplam dal sayısı $= 1 + b^1 + b^2 + \dots + b^d$ ($d \to$ derinlik)

çizdiğimiz ağaçta b=2, derinlik 2 düğüm.

⇒ Bütün hareketler, hareketlerin maliyeti aynıysa optimaldir.

Örneğin şehir arası mesafe birbirinden farklı olduğu için optimal değildir.

NOT: [illegible]

Matematiksel olarak b ile d arasını belirlemede kullanılır. Örneğe değeri verilmiş.

b ile d arasındaki ilişki

Zaman ve hafıza karmaşıklığı: b^d

7. bana düğüme gidiyor. 7. birim zaman harcıyor.

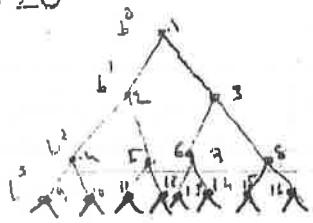
ÖR: b=2, 4. derinlikte bastan 5. düğümde çözüm?

↳ her bir düğüm 2'ye ayrılıyor demek

$1 + b + b^2 + b^3 + 5$

$1 + 2 + 2^2 + 2^3 + 5 = 20$

$2^4 = 16$ zaman ve hafıza karmaşıklığı

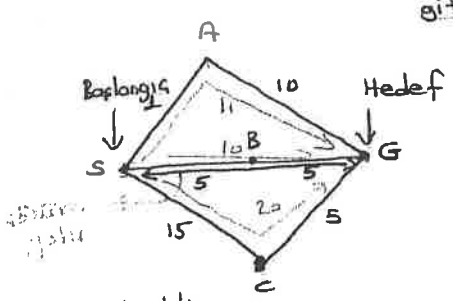


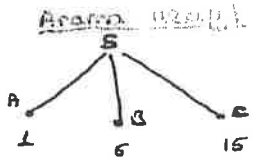
16 düğüme gidiyor

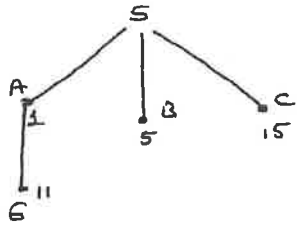
16 birim zaman harcıyor.

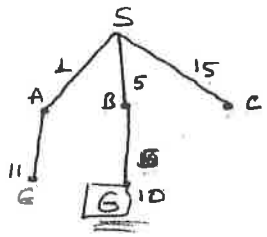
2. Derine Arama Algoritması

Bilgili Olmayan Arama: Bilgili arama yöntemi değildir çünkü gittiği mesafe gittikten sonra verilir.







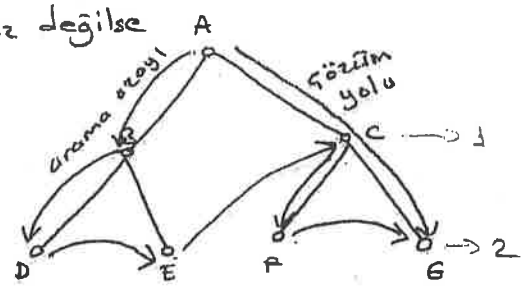


- Optimal değildir.
- Tamdır
- Complete'dir.
- Zaman ve hafıza $= b^d$ dır.

$3^2 = 9$

3. İlk Önce Derine Arama Algoritmasını (Kaplan - Pinsk - Sonuk):

Tam => b sonsuz değilse tamdır.



Hafıza: A B

Kuyruk: ~~D~~ E ~~B~~ C

- Tam ~~b~~ değildir => Sonsuz derinlikteki yada tekrarlanan durumlu ağaçlar sonuca ulaşamaz
- Optimal değildir
- Complete değildir

Kökten mümkün olduğunca uzaklaştığı için optimal değil

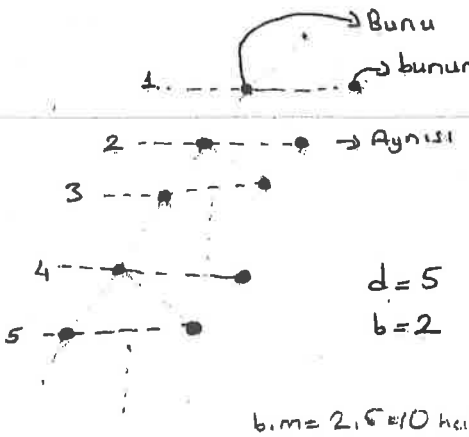
Tekrarlama faktörü

Hafıza gereksinimi bir

Tamamlanmamıştır.

$2.2 = 4$

b.m Derine Kaplama Algoritması: Tek fark bu hafıza gereksinimi az.



tüm düğümler için geçerli

Dezavantajı:
Sonsuz derine gittiğinde çözümü bulamıyor.

$d = 5$

$b = 2$

$1 + b^1 + b^2 + b^3 + b^4 + b^5$

$b.m = 2.5 = 10$ hafıza gereksinimi

4. Sınırlı [illegible]

Zaman: $O(b^l)$ l: sınırlı derinlik

Hafıza: $O(b.l)$ d: çözüm derinliği

Derinlik sınırı(l) çözüm yolu uzunluğu en fazla durum sayısı kadar

- Eğer $l \geq d$ ise tamdır.
- İnebileceğin derinlik çözüm değinliğinden büyük olmalıki çözümü bulabilsin.
- Optimal değildir.

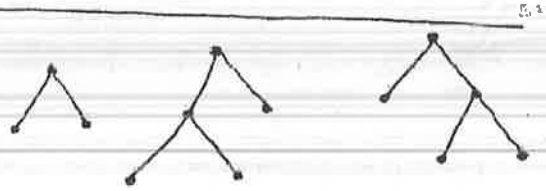
5. İteratif [illegible]

optimaldir. Çünkü kök düğüme en yakın çözümü bulur

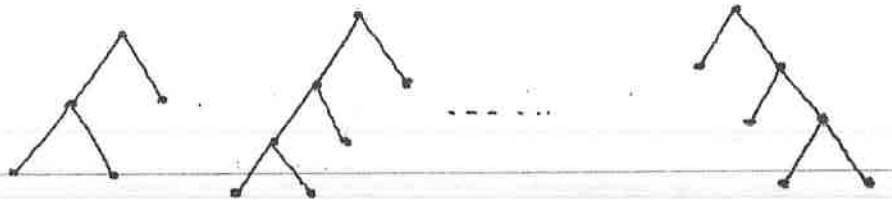
limit 0 •

limit 1 •

limit 2 •



limit 3 •



[illegible]

$(d-1)b^2 + (d-2)b^3 + (d-3)b^4 + (d-4)b^5$

[illegible]

$1 + 10 + 10^2 + 10^3 + 10^4 + 10^5 = 111.111$

$(5+1).1 + (5).10 + (5-1).10^2 + (5-2).10^3 + (5-3).10^4 + (5-4).10^5$

Bilgili Arama Yöntemleri

(Heuristik = Sezgisel Yöntemler)

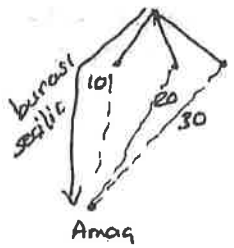
Bilgili arama yöntemlerinde strateji Best-First-Search (En iyi-ilk önce-arama)

iki türdür Zaman ve hafıza = $O(b^m)$

1. Gready arama

2. A^* arama

1. Gready Arama: Amaca en yakın yol sezgisel olarak bulunur. Değerleri karşılaştırmak için bütün düğümleri bellekte tutar.



* Algoritma her adımda hedefe en yakın yoldan ilerliyor.

$f(n)$ ve $h(n)$ → heuristik değer (kuş uçuşu mesaf

Tam değildir. (Döngülere takılabildiği için)

Time ve space = $O(b^m)$ dir.

Optimal değildir

2. A^* arama: Hem yolu hem kuş uçuşu mesafeyi toplayarak işlem görür. optimaldir.

Kökten itibaren toplam maliyeti yüksek durumlara gidişi engellemek

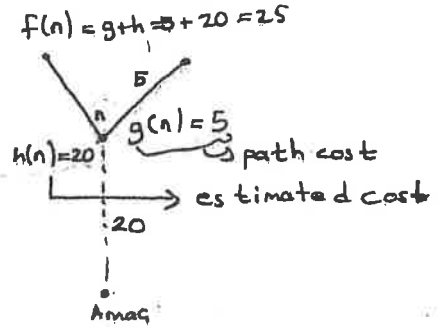
$$f(n) = g(n) + h(n)$$

Heuristik

$g(n)$: Kökten mevcut duruma (n) gelişin maliyeti

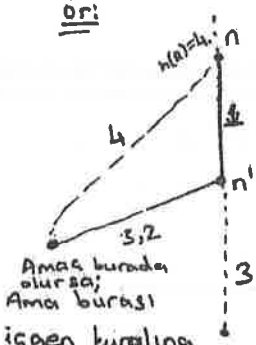
$h(n)$: Mevcut durumdan (n) hedefe gidişin tahmini maliyeti

$f(n)$: Kökten hedefe n'den geçilerek gidişin tahmini maliyeti



Yaprak düğümlere bakılarak arama yapılır her zaman için en küçüğü bulunur ve o küçük düğüm açılır. Daha sonra bakılır yaprak düğümlerde en küçük hangisi olursa ondan devam edilir hedefe ulaşana kada

Ör:



$g(n) = 3$ — 1 arttı ise $h(n)$'nde 1 azalması gerekir.

$g(n') = 4$

$h(n') = 3$ olabilirmi?

Amaç burada olursa doğrusal olduğu için $h(n)=4$ burası en az 3 olabilir.

içgen kuralına göre 3,... bir değer alır.
ki şeyin toplamı liğerinden fazla olmak zorunda.

ÖR:

5	4	
6	1	8
7	3	2

Başlangıç durum

1	2	3
8		4
7	6	5

Amaç durum

yürüttik ⇐ eğeri 7'dir.

5 yerine 4 adımda gitti
1 yerine 2 adımda gitti

1. h_1 = yanlış durumdaki (pozisyondaki) karelerin sayısı

$h_1 = 7$

2. h_2 = Karelerin amaç pozisyonlarından manhattan uzaklıklarının toplamı

$h_2 = 2 + 3 + 3 + 2 + 4 + 2 + 0 + 2 = 18$

h2 için

1 yerine 2 birim uzaklıkta
2 yerine 3 birim uzaklıkta
3 yerine 3 birim uzaklıkta
4 yerine 2 birim uzaklıkta
5 yerine 4 birim uzaklıkta
6 yerine 2 birim uzaklıkta
7 yerine 0 " "
+ 8 " 2 birim uzaklıkta

h2 = 18

Performans Üzerine Heuritkin Etkisi

b^* : Etkili dallanma faktörü

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

<u>ÖR:</u> Eğer A^+, 52 düğüm kullanarak derinliği 5'de bir çözüm bulmuşsa

$N = 52$, $d = 5$

$b^* = 1,91$

b^*; 1'e ne kadar yakın olursa heurntik o kadar etkin

$h(n) = \max(h_1(n) \dots h_m(n))$

$b=1$ olursa bir düğümden diğerine gitmek için 1 yol vardır.

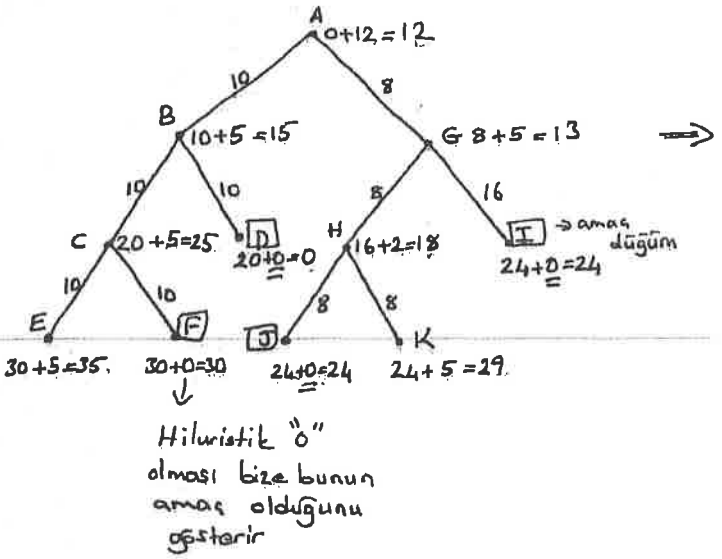
Hafıza Sınırlı Arama

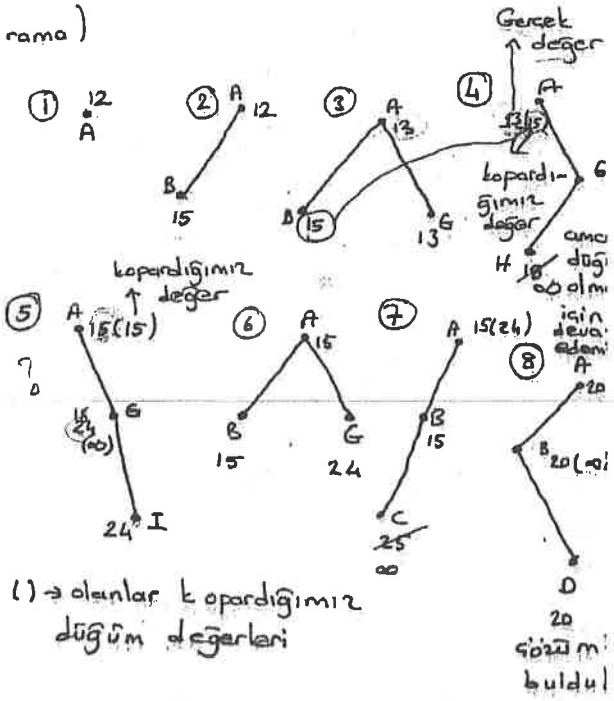
1. İteratif Derinleştirerek A^* Arama (IDA*)

$f = g + h$ (derinleştirerek arama yapar. f'i git gide artırıyoruz

$f = 100 - 1000$

2. SMA* Arama (Basit Hafıza Sınırlı A* Arama)





3 düğümlük hafıza

→ sınavda soruda değerler verilir ağaç ve çözüme giden yolları bizim çizmemiz istenir.

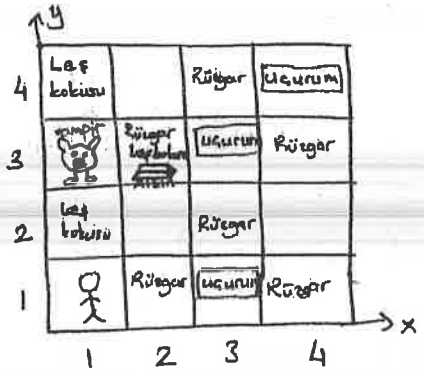
Lokal Arama Yöntemleri

1. Hill-Climbing (Tepeye Tırmanma):

2. Simulated Annealing (Benzetimli Tavlama):

BİLGİ VE MUHAKEME

⇒ Vampir Dünyası:

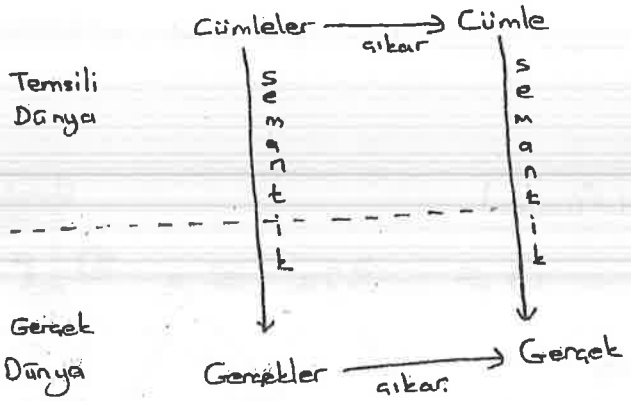


⇒ küçük kediler ve köpekler burada kedilerin küçük olduğunu çıkarır ama köpeklerin büyükmü yoksa küçükmü olduğu tam net değildir. Bunu şöyle ifade edersek eğer daha anlaşılır olur:

$-c + d$

cat (olumsuz), dog (olumlu)

böylece çıkan sonuç kedi küçük köpek büyük muhakemesi yaparız.



Lojik Türleri

Dil	Ontolojik Anlamı	Gerçeklik Değeri
Önermesel lojik	Gerçekler	T / F / Bilinmeyen
Birinci derece lojik	Gerçekler, nesneler ilişkiler	T / F / Bilinmeyen
~~Olasılık Teorisi~~ Bağıcı Lojik	Gerçekler, nesneler, ilişkiler, zaman	T / F / Bilinmeyen
~~Bulanık lojik~~ Olasılık Teorisi	Gerçeklik	0-1 arası değer
Bulanık Lojik	Doğruluk derecesi	0-1 arası değer

Cümleler → Atomik Cümle | Karmaşık Cümle

Atomik cümleler → T | F (T: True, F: false)

P | Q | R ....

Karmaşık Cümle → (Cümle)

| Cümle Bağıntı cümle

| ¬ cümle (¬: değil işareti)

Bağıntı → ∧ | ∨ | ⟺ | ⟹

(∧: ve, ∨: veya, ⟺: ancak ve ancak, ⟹: ise)

(önermesel lojiğin BNF yapısı)

İSE TABLOSU

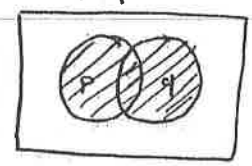
p	q	p ⟹ q
0	0	1
0	1	1
1	0	0
1	1	1

ANCAK VE ANCAK TABLOSU

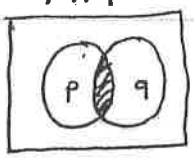
p	q	p ⟺ q
0	0	1
0	1	0
1	0	0
1	1	1

$p \Rightarrow q \equiv \neg(p \vee q)$

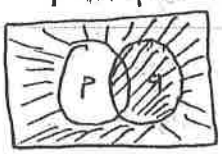
p∨q



p∧q



$p \Rightarrow q \equiv \neg(p \vee q)$



Önermesel Lojiğin Çıkarım Kuralları

1. Modus Ponens

$$\frac{\underbrace{\alpha \Rightarrow \beta}_{\text{Kural}},\ \alpha}{\beta}$$

(α: gerçek; β: çıkan sonuç)

Eğer kuralım varsa ve α gerçeğimde varsa β dışarı çıkar.

2. And-Elimination

$$\frac{\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n}{\alpha_i}$$

(α_i: çıkan sonuç)

→ Eğer böyle bir kuralım varsa her bir α_i değerim ayrı ayrı dışarı çıkar.

3. And- Introduction

$$\frac{\alpha_1, \alpha_2, \dots \alpha_n}{\alpha_1 \wedge \alpha_2 \wedge \alpha_3 \dots \wedge \alpha_n}$$

4. Or - Introduction

$$\frac{\alpha_i}{\alpha_1 \vee \alpha_2 \vee \dots \vee \alpha_n}$$

5. Değilin - Değili

$$\frac{\neg\neg\alpha}{\alpha}$$

6. Unit Resolution

$$\frac{\alpha \vee \beta, \ \neg\beta}{\alpha}$$

7. Resolution

$$\frac{\alpha \vee \beta, \ \beta \vee \gamma}{\alpha \vee \gamma} \equiv \frac{\neg\alpha \Rightarrow \beta, \ \beta \Rightarrow \gamma}{\neg\alpha \Rightarrow \gamma}$$

Arama yöntemleri
lojikten soru çıkabilir.
Tanım

vampir örneğini ifade etmeye çalışırsak

Bilgi tabanı:

$S_{1,2} \longrightarrow [1,2]$ koku var

$B_{2,1} \longrightarrow [2,1]$ rüzgar var

$\neg \underbrace{S_{11}}_{\text{Hücre kokusu ve rüzgar var}}$, $\neg \underbrace{B_{11}}_{}$

$\neg \underbrace{S_{21}}_{\text{koku yok}}$, $\underbrace{B_{21}}_{\text{rüzgar var}}$

$\underbrace{S_{12}}_{\text{koku var}}$ $\neg \underbrace{B_{12}}_{\text{rüzgar var}}$

4x16 = 64 tane kuralım var normalde

Kuralları

R1) $\neg S_{11} \Rightarrow \neg W_{11} \wedge \neg W_{12} \wedge \neg W_{21}$

R2) $\neg S_{21} \Rightarrow \neg W_{11} \wedge \neg W_{21} \wedge W_{12} \wedge \neg W_{31}$

R3) $\neg S_{12} \Rightarrow \neg W_{11} \wedge \neg W_{12} \wedge W_{22} \wedge \neg W_{13}$

R4) $S_{12} \Rightarrow \boxed{W_{13}}^{?} \vee W_{12} \vee W_{22} \vee W_{11}$

1) $\neg S_{11}$ ve R1'e Modus Ponens uygulansak

$\neg W_{11} \wedge \neg W_{12} \wedge \neg W_{21}$

2) 1'e and-elimition uygularsak

$\neg W_{11}$, $\neg W_{12}$, $\neg W_{21}$ ⇒ her biri ayrı ayrı dışarı çıkar and-elimitionda.

3) $\neg S_{21}$ ve R2'e Modus Ponens + And elimition

$\neg W_{11}$, $\neg W_{21}$, $\neg W_{12}$, $\neg W_{31}$

4) S_{12} ve R4'e Modus Ponens

$\underbrace{W_{13} \vee W_{12} \vee W_{22}}_{\beta} \vee \underbrace{W_{11}}_{\alpha}$ → W_{11}'da yukarıda değilini bulmuştuk.

5) Unit resulation uygulanırsa

$\neg W_{11}$ 2. adımda çıkarılmıştı

$\underbrace{W_{13} \vee W_{12}}_{\alpha} \vee \underbrace{W_{22}}_{\beta}$

6.) $\neg W_{22}$ 3. adımda çıkarılmıştı

$\underbrace{W_{13}}_{\alpha} \vee \underbrace{W_{12}}_{\beta}$

7) $\neg W_{12}$ 2. adımda çıkarıldı. $\boxed{W_{13}}$ vampir 13 karesinde ? W_{13}

BİRİNCİ DERECE LOJİK (Yüklem Aritmetiği)

Cümle → Atomik Cümle

| Cümle BAĞINTI Cümle | Niceleyici ... cümle

| ¬ Cümle | (Cümle)
↳ değil

Atomik Cümle → Yüklem(Terim, ...) | Terim = Terim

Terim → Fonksiyon(Terim, ...) | Sabit | Değişken

BAĞINTI → ⇒ | ∧ | ∨ | ⇔

Niceleyici → ∀ | ∃

Sabit → A | x_1 | John ...

Değişken a | x | s ...

Yüklem → Önce | Renklere sahip | Yağmur yağıyor | ...

Fonksiyon → Annesi | Solbacağı | ...

Atomik Cümle:

Erkek kardeşi (Richard, John)

Komplex Cümle:

Daha yaşlı (John, 30) ⇒ ¬ Daha Genç (John, 30)

Niceleyiciler:

- ∀x Kedi (x) ⇒ memelidir (x) : Tüm kediler memelidir.
- ∀x, y Ebeveyn (x, y) ⇒ çocuğu (y, x)
- $\underbrace{\text{Herkes}}_{x}$ $\underbrace{\text{birini}}_{y}$ sever ∀x, ∃y Sever(x, y)
- $\underbrace{\text{Herkes}}_{x}$ tarafından sevilen $\underbrace{\text{bir insan}}_{y}$ vardır. ∃y, ∀x Sever(x, y)
- ∀x . ¬ Sever (x, Havuç) ≡ ¬ ∃x Sever(x, Havuç)
  (Herkes havucu sevmez.) (Havucu seven kimse yok)

Herkes dondurmayı sever ⇒ ∀x Sever (x, Dondurma) ≡ ¬∃x ¬ Sever(x, dondurma)
Dondurmayı sevmeyen kimse yok.

De Morgan Kuralları:

1. ∀x ¬P ≡ ¬∃x P
2. ¬∀x P ≡ ∃x ¬P
3. ∀x P ≡ ¬∃x ¬P
4. ∃x P ≡ ¬∀x ¬P

Birinci Derece Lojikte Çıkarsama

Önermesel lojikte çıkarmasal kurallara ek olarak;

.. Universal Elimination:

$$\frac{\forall v\ \alpha}{SUBST(\{v/g\}, \alpha)}$$

α → cümle
v → değişken
g → ground terim (x yerine herhangi bir nesne yazılabilir (g))

ÖR: ∀x Sever(x, Dondurma), {x/Ben} Sever(Ben, Dondurma)

. Existential Elimination:

$$\frac{\exists v, \alpha}{SUBST(\{v/k\}, \alpha)}$$

α → cümle
v → değişken
k → sabit sembol (Bilgi tabanından daha önce geçmeyen)

ÖR: ∃x öldürür(x, Kurban), öldürür(katil, kurban)

.. Existantial Introduction:

$$\frac{\alpha}{\exists v\ SUBST(\{g/v\}, \alpha)}$$

ÖR: sever(Jerry, Dondurma)' dan

∃x Sever(x, Dondurma)

$\neg P \wedge \neg q \equiv \neg(P \vee q)$

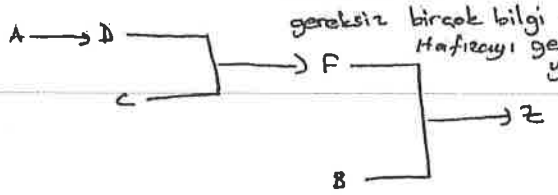
$\neg(P \wedge q) \equiv \neg P \vee \neg q$

$P \wedge q \equiv \neg(\neg P \vee \neg q)$

$P \vee q \equiv \neg(\neg P \wedge \neg q)$

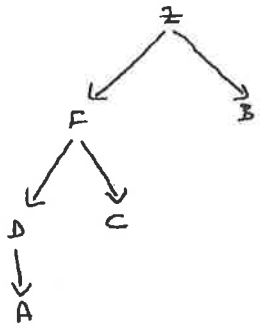
İLERİ ZİNCİRLEME YAPISI

Dezavantajı: Çok zaman alır gereksiz birçok bilgi veri verir. Hafızayı gereksiz yorar.

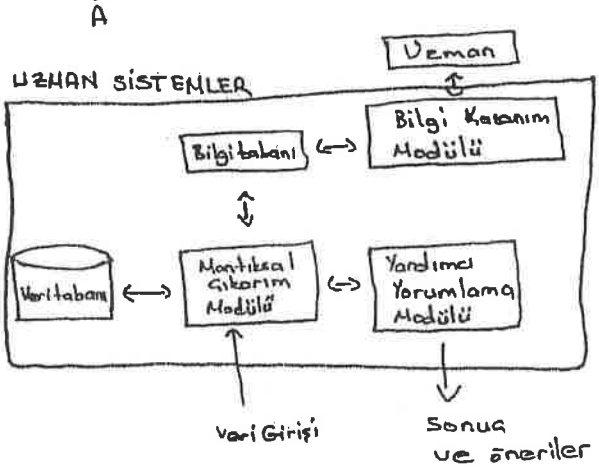


Gerçekler	Gerçekler	Gerçekler	Gerçekler
(A) B	A B	A (B)	A B
C H	(C) H	(F) C H	C H F
E G	E G (D)	D E G	DE G
			(Z)
F∧B → Z	F∧B → Z	F∧B → Z	
C∧D → F	C∧D → F	C∧D → F	
A → D	A → D → D	A → D	
Kurallar	Kurallar	Kurallar	elde ettik.

GERİYE ZİNCİRLEME YAPISI



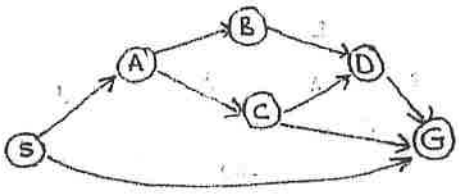
UZMAN SİSTEMLER



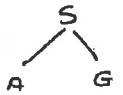
İnsan	Uzman Sistem
Değişkenlik	Sabit
Bilgilerin Arttırılma Zorluğu	Basit Bilgi Nakli
Bilgilerin Sistemleştirilmesi zordur	Basit Belgeleme İşlemi
Pahalı	Ucuz
Aktif Yaşam Süresi az	Çok

Vize Soruları

①

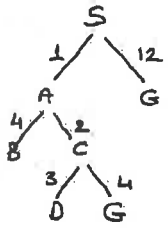


ilk-önce-enine



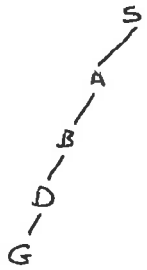
Yol : S-G

b) Uniform maliyetli arama



Yol : S-A-C-G

ilk-önce-derine



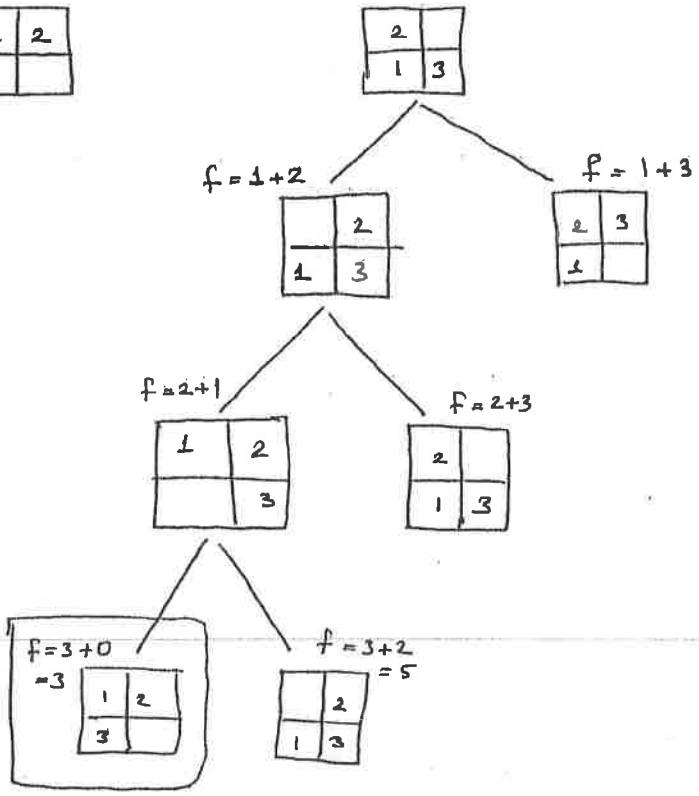
Yol: S-A-B-D-G

②

h=3

f = h+g
f = 0+3
= 3

h = yanlış yerdeki taşların sayısı



③ $f(n) = (100-w).g(n) + w.h(n)$

a) $w = 0$ $\quad f(n) = 100g(n)$ ⇒ uniform maliyetli arama

b) $f(n) = 50g(n) + 50h(n)$ $\quad A^*$ arama

c) $f(n) = 100h(n)$ → Greedy arama

d) $0 \leq w \leq 50$ $\quad$ 50'den sonrası için optimallik bozuluyor.

GENETİK ALGORİTMALAR

Bir arama yöntemidir.

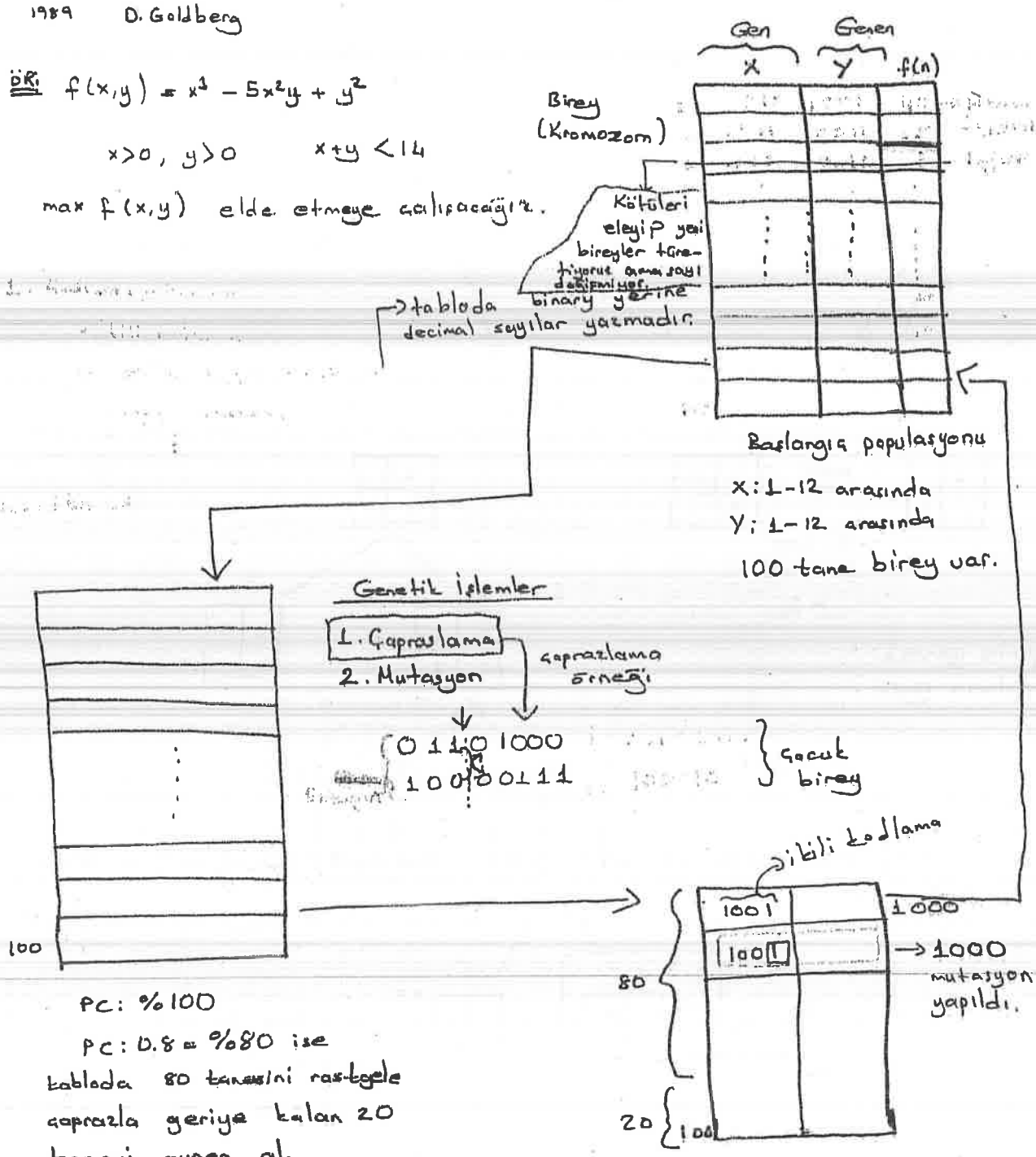
1975 J. Holland

1989 D. Goldberg

ÖR: $f(x,y) = x^3 - 5x^2y + y^2$

$x>0,\ y>0 \qquad x+y<14$

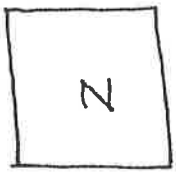
max $f(x,y)$ elde etmeye çalışacağız.



*Mutasyon rastgele bir bireyin rastgele bir geninde rastgele bir bitinde 0 ise 1, 1 ise 0 yapılması olayıdır.

Mutasyon binde bir derilirse populasyonun 100 buda 10 jerarosyond bir mutasyon yapma anlamına gelir.

ÖR: [illegible]



k tane nesne var.

Hacim: $V_1, V_2, \ldots V_k$

Fayda: $U_1, U_2, \ldots U_k$

Hacim ve fayda arasında nasıl bir ilişki olduğu bilinmiyor.

NOT: Kromozom uzunluğu oldukça kısa elde edilmeye çalışır. Ne kadar kısa olursa o kadar hızlı çalışır.

1: Nesnenin alındığını
0: Nesnenin alınmadığını

Hacim → 1 0 1 ... 1

→ Bunların toplamı benim çantama sığıyormu sığmıyormu. Sığmıyorsa fitnessına negatif değer verilerek elenir.

$$\sum_{\text{Alınan}} V_i \leq N$$

Çaprazlama ve mutasyon sonucunda istenmeye bir biray elde edilirse onu elemek için fitnessına negatif değer verilerek kötü değer olarak gösterilir ve elenir.

$$\text{Fitness} = \max \sum_{\text{Alınan}} U_i$$

ÖR: 8 vezir probleminin genetik algoritma olarak çözümü (ikili kodlama kullanarak)

	1	2	3	4	5	6	7	8
000								
001	V							
010								
011		V						
100								
101								
110								
111								

Kromozomu

3x8=24 en kısa bu bulunur

1	2	3		8
001	011		...	

1. konumda 001 satırında vezir bulunmaktadır.

fitness: Birbirini tehdit etmeyen vezir sayısı

Amaç fitness: 4 olarak varsaydı max(fitness) = fitness: 8 yapmak.

BİYOMETRİK TANIMA SİSTEMLERİ

1. Yüz tanıma
2. Parmak izi tanıma
3. Avuç içi Tanıma
4. Iris (Göz retinası) tanıma
5. İmza tanıma
6. Ses tanıma

⇒ 1. Yüz tanıma: Lambrezo tarafından bulunmuştur. Mahkumlar üzerinde yüz tanımayı kullanmıştır.

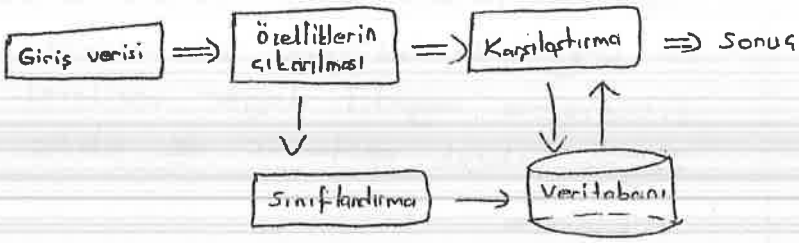
Altın Oranı: 1,618

Yüz yüksekliği / Yüz genişliği

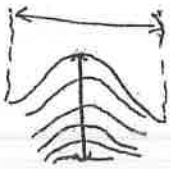
Alın genişliği / Burun boyu

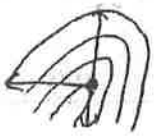
Yüz genişliği / Göz bebekleri arası mesafe

Blog diyagramı;



⇒ 2: Parmak izi tanıma:





⇒ 3. Avuç içi tanıma: Avuç içindeki çizgilere göre tanır

⇒ 4. Göz Retinası tanıma: Gözdeki kılcal damarlara göre tanıma yapıyor.

⇒ 5. İmza Tanıma:

A A A

→ A harfini tanıma

⇒ 6. Ses tanıma: Başarımı en düşük tanımadır. Çünkü sesi işlemek zordur.