

## Rekabet ortamında arama Adversarial Search

### Rekabet ortamında arama

- ◎ Çoklu vekil ortamı- her bir vekil karar verirken diğer vekillerin de hareketlerini dikkate almalı ve bu vekillerin onun durumunu nasıl etkileyeceğini bilmelidir
- ◎ Olasılık- diğer vekillerin hareketlerinin tahmin edilememesi
  - “Önceden tahmin edilemeyen” karşı taraf
  - Rakibin her olası cevabına karşı bir hareketin belirlenmesi
- ◎ İşbirlikçi ve rakip vekiller
- ◎ Rekabet ortamında arama-oyun
- ◎ Zaman sınırlamaları

2

### Oyun neden öğrenilmeli?

- ◎ Yapay Zekanın en eski alanlarından birisi (Shannon and Turing, 1950)
  - Zeka gerektiren rekabetin soyut ifadesi
  - Durum ve faaliyetlerin kolay ifade edilebilirliği
  - Dış dünyadan çok az bilginin gerekliliği
- ◎ Oyun oynama, bazı yeni gereksinimlerle aramanın özel halidir.

### Oyun türleri

- Tam bilgili; tam olmayan bilgili
- Deterministik (Belirli)
- Şans (Talih)

Satranç,dama,go- tam bilgili, deterministik  
Tavla- tam bilgili, şans  
Kağıt oyunları- tam olmayan bilgili, şans

## Oyunla ilgili sorunlar

### ① Oyunların çözümü zordur

#### ② "Olasılık" sorunu

→ Rakibin hareketini bilmiyoruz!

#### ③ Arama uzayının boyutu:

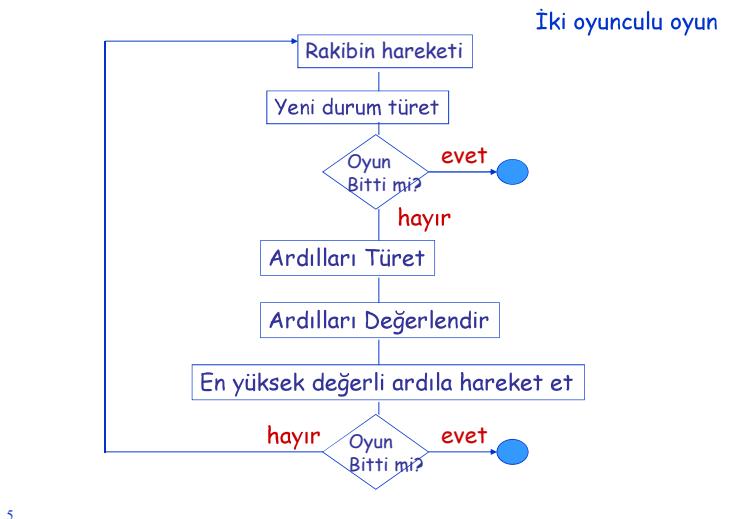
→ Satranç : her durumda yaklaşık ~15 hareket, 80 karşılıklı hamle

◆ ağaçta  $15^{80}$  düğüm

→ Go : her durumda ~200 hareket, 300 karşılıklı hamle

◆ ağaçta  $200^{300}$  düğüm

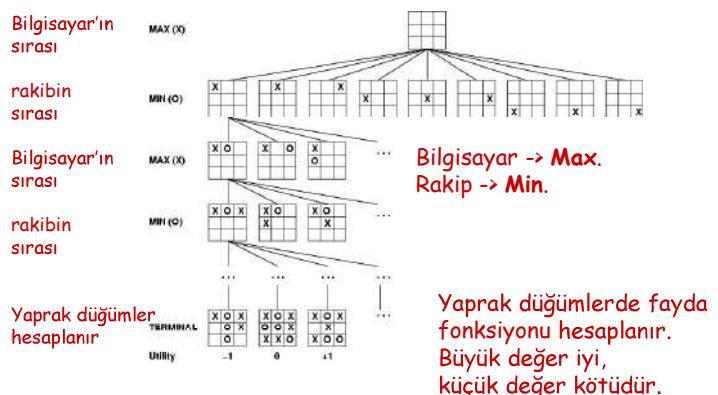
#### ④ Optimal çözümün çoğu zaman mümkün olmaması



## Oyun-arama sorunu

- ① Başlangıç durum-ilk pozisyon ve birinci hamle yapacak oyuncu
- ② Ardıl fonksiyonu- (*hareket,durum*) çiftleri listesini veriyor; yasal hareket ve bu hareket sonucu durum
- ③ Uç düğüm (terminal) denemesi -oyunun bittiğini belirler. Oyunun son bulduğu durumlara uç durumlar denir
- ④ Fayda fonksiyonu - uç durumlar için sayı değeri
- ⑤ Oyun ağacı- başlangıç durum ve her iki tarafın yasal hareketleri

## Oyun Ağacı(2-oyuncu, Deterministik, Sıralı)

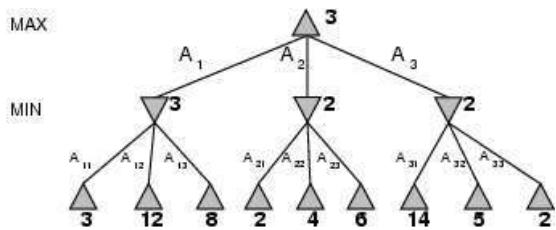


## Oyun oynama algoritmaları:

- ◎ Minimax algoritması
- ◎ Alpha-beta budama
  - Değerlendirme fonksiyonu
  - Aramayı kesme
  - her hangi derinlik sınırına kadar arama
  - Derinlik sınırında değerlendirme fonksiyonunun kullanılması
  - Değerlendirmenin tüm ağaç boyunca yayılması

## Minimax yöntemi

- ◎ Deterministik oyunlar için mükemmel taktik
- ◎ Temel fikir: en yüksek minimax değerli hareketi seçmeli = en iyi ulaşılabilir sonuç



9

10

## Minimax değer

- ◎ Minimax değer ( $n$ )=

Fayda( $n$ ) , eğer  $n$  son durum ise

Max (Minimaxdeğer( $s$ )),  $n$ -Max düğüm ise  
 $S \in \text{ardıllar}(n)$

Min (Minimaxdeğer( $s$ ))),  $n$  - Min düğüm ise

$S \in \text{ardıllar}(n)$

## Minimax algoritması

```

function MINIMAX-DECISION(state) returns an action
  v ← MAX-VALUE(state)
  return the action in SUCCESSORS(state) with value v

function MAX-VALUE(state) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← −∞
  for a, s in SUCCESSORS(state) do
    v ← MAX(v, MIN-VALUE(s))
  return v

function MIN-VALUE(state) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← ∞
  for a, s in SUCCESSORS(state) do
    v ← MIN(v, MAX-VALUE(s))
  return v
  
```

11

12

## minimax'in özellikleri

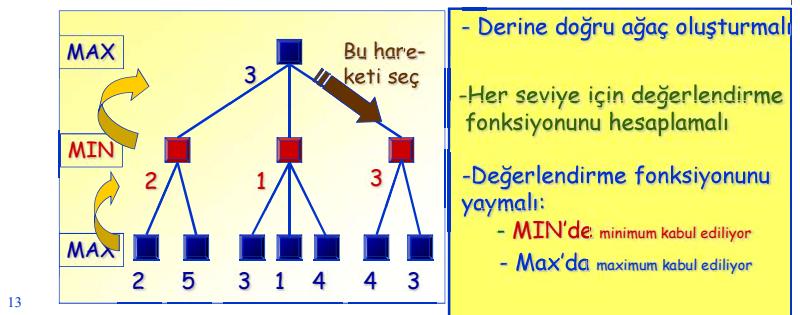
- tam? Evet (eğer ağaç sonlu ise)
- Optimal? Evet (optimal rakibe karşı)
- Zaman karmaşıklığı?  $O(b^m)$
- Uzay karmaşıklığı?  $O(bm)$  (derinine aramada)
- m- ağaçın en fazla derinliği
- b- her noktada mümkün hamleler sayısı
- Satranç için  $b \approx 35$ ,  $m \approx 100$

## MINI MAX

### © Kısıtlar:

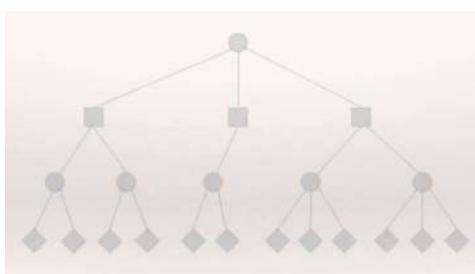
- 2 oyuncu: MAX (bilgisayar) ve MIN (rakip)
- deterministik, tam bilgi

### © Derinine arama ve değerlendirme fonksiyonu

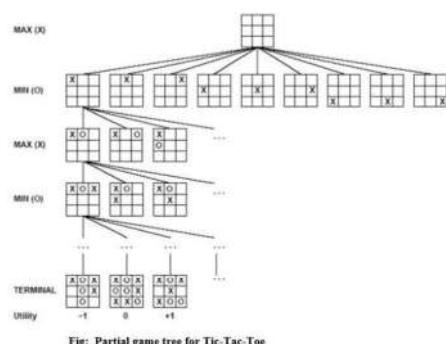


13

## MINI MAX



<https://www.youtube.com/watch?v=zDskcx8FStA>



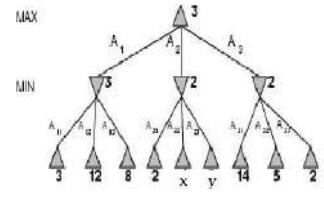
## Alpha-Beta budama

- ④ Tüm ağacın (yukarıdan aşağıya doğru derinine) oluşturulmasına ve değerlerin tüm ağaç boyu yayılmasına gerek kalmayabilir
- ④ Edinilmiş bazı değerler, ağacın üretilmemiş kısımlarının fazla olduğu ve üretilmesine gerek kalmadığı bilgisini verebilir

## $\alpha$ - $\beta$ budama için minimax değerinin bulunması

**Temel fikir:** oyun ağacında her bir düğüme bakmadan da doğru çözümü bulmak mümkün değildir. Bu halde ağacın bakılmayan kısmı budanmış oluyor

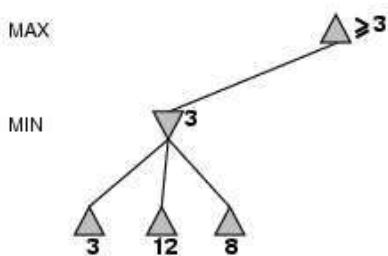
**Minimax Değer(kök)=**  
 $\max(\min(3,12,8),\min(2,x,y),\min(14,5,2))$   
 $=\max(3,\min(2,x,y),2)=\max(3,z,2)$  ;  
 $z=\min(2,x,y)$  kabul ettik. Buradan  $z \leq 2$  olduğu anlaşılıyor.  
 O zaman Minimax Değer(kök)= 3 alırız



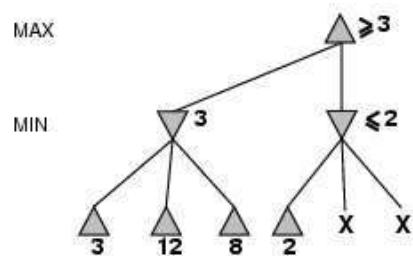
17

18

## $\alpha$ - $\beta$ budama örneği



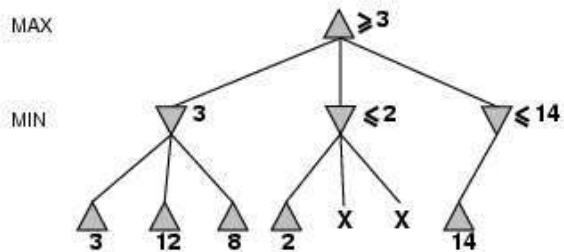
## $\alpha$ - $\beta$ budama örneği



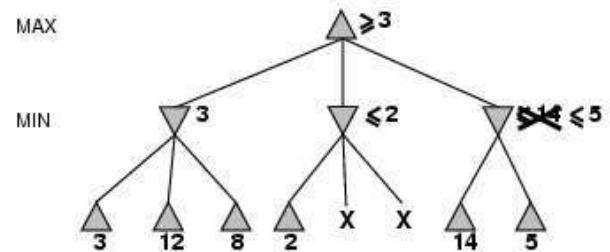
19

20

### $\alpha$ - $\beta$ budama örneği



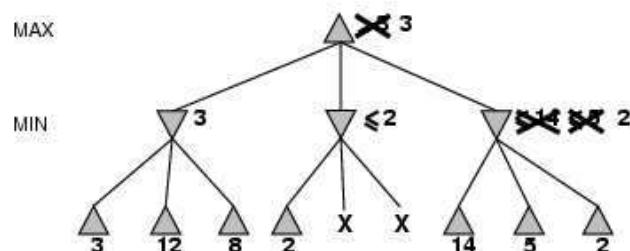
### $\alpha$ - $\beta$ budama örneği



21

22

### $\alpha$ - $\beta$ budama örneği



23

### Alpha-Beta budama



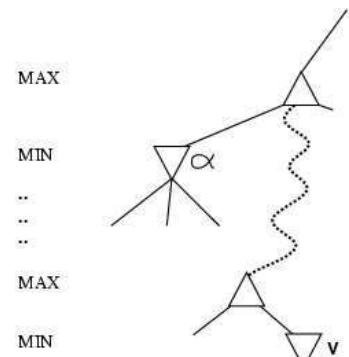
<https://www.youtube.com/watch?v=l-hh51ncgDI>  
<https://www.youtube.com/watch?v=Ewh-rF7KSEg>

## $\alpha$ - $\beta$ 'nın Özellikleri

- ◎ Budama son neticeyi etkilemez
- ◎ Hareketlerin iyi sıralanması budamanın etkiliğini yükseltir
- ◎ "mükemmel sıralamada," zaman karmaşıklığı =  $O(b^{m/2})$

## Neden $\alpha$ - $\beta$ ?

- ◎  $\alpha$ , max için yol boyunca seçilmiş en iyi (en yüksek) değer
- ◎ Eğer  $v$   $\alpha$ 'dan kötü ise max onu iptal edecek  
→ uygun dal budanacak
- ◎ Min için  $\beta$ , benzer yolla değerlendirilir



25

26

## $\alpha$ - $\beta$ algoritması

```
function ALPHA-BETA-SEARCH(state) returns an action
  inputs: state, current state in game
   $v \leftarrow \text{MAX-VALUE}(state, -\infty, +\infty)$ 
  return the action in SUCCESSORS(state) with value  $v$ 

function MAX-VALUE(state,  $\alpha, \beta$ ) returns a utility value
  inputs: state, current state in game
     $\alpha$ , the value of the best alternative for MAX along the path to state
     $\beta$ , the value of the best alternative for MIN along the path to state
  if TERMINAL-TEST(state) then return UTILITY(state)
   $v \leftarrow -\infty$ 
  for  $a, s$  in SUCCESSORS(state) do
     $v \leftarrow \text{MAX}(v, \text{MIN-VALUE}(s, \alpha, \beta))$ 
    if  $v \geq \beta$  then return  $v$ 
     $\alpha \leftarrow \text{MAX}(\alpha, v)$ 
  return  $v$ 
```

## $\alpha$ - $\beta$ algoritması

```
function MIN-VALUE(state,  $\alpha, \beta$ ) returns a utility value
  inputs: state, current state in game
     $\alpha$ , the value of the best alternative for MAX along the path to state
     $\beta$ , the value of the best alternative for MIN along the path to state
  if TERMINAL-TEST(state) then return UTILITY(state)
   $v \leftarrow +\infty$ 
  for  $a, s$  in SUCCESSORS(state) do
     $v \leftarrow \text{MIN}(v, \text{MAX-VALUE}(s, \alpha, \beta))$ 
    if  $v \leq \alpha$  then return  $v$ 
     $\beta \leftarrow \text{MIN}(\beta, v)$ 
  return  $v$ 
```

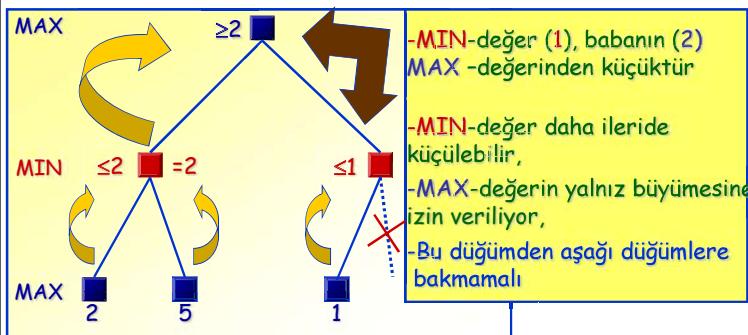
27

28

## Alpha-Beta budama ilkeleri

### © İlkeler:

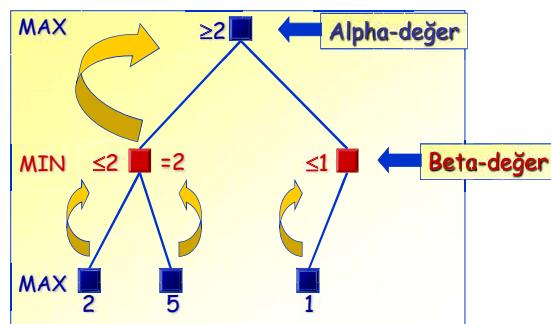
- Derinine, soldan sağa ağaç üretmeli
- son düğümlerin değerlerini baba düğümleri için başlangıç tahminler kabul etmeli.



29

## Alpha-Beta budama ilkeleri (devami)

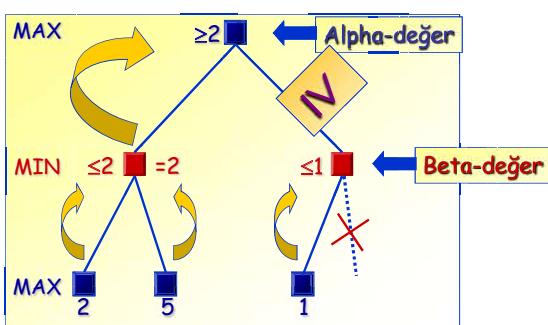
- MAX-düğümlerde (geçici) değerler ALPHA-değerlerdir
- MIN-düğümlerde (geçici) değerler BETA-değerlerdir



30

## Alpha-Beta ilkeleri (1):

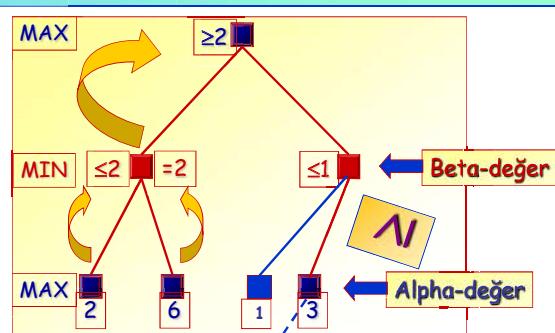
- Eğer ALPHA-değer oğul düğümün Beta-değerinden büyük veya ona eşitse:  
uygun soydan düşümlerin üretimi sürdürülmeli



31

## Alpha-Beta ilkeleri (2):

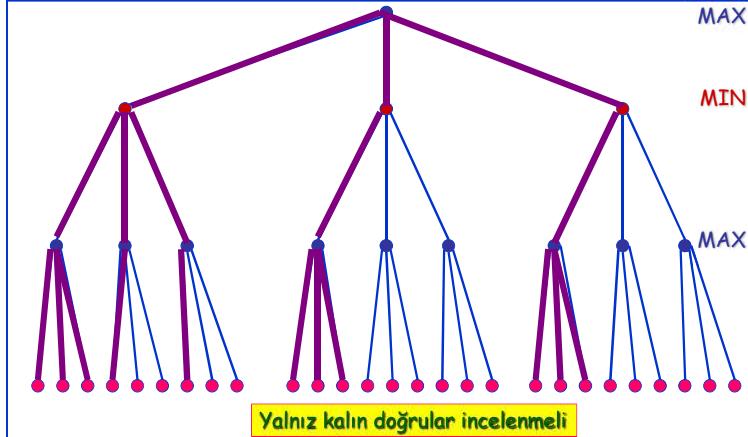
- Eğer Beta-değer, oğul düğümün Alpha-değerinden küçük veya ona eşitse:  
uygun soy üzerinden düşümlerin üretimi durdurmalı



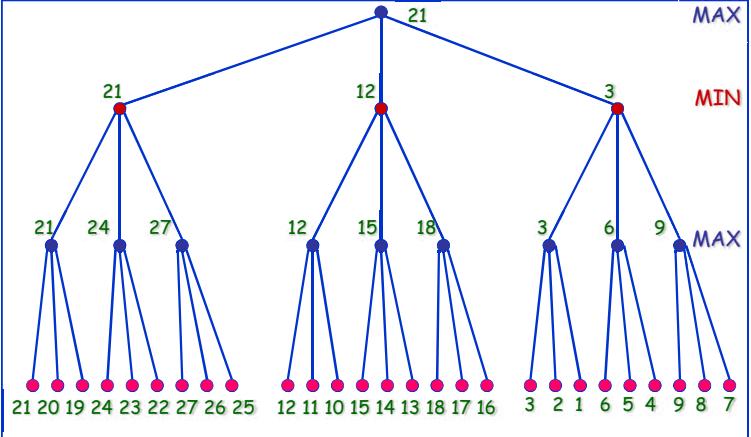
32

## En iyi durum

- Eğer her seviyede: en iyi düğüm en soldaki düğüm ise



## Mükemmel sıralanmış ağaç örneği



## Değerlendirme fonksiyonu

### Ⓐ Ağırlıklı doğrusal fonksiyon

$$Eval(s) = w_1 f_1(s) + w_2 f_2(s) + \dots + w_n f_n(s)$$

w-özelliğin ağırlığı

f-özellik

Örnek: satrançta

$$f(s) = \text{aynı türden taşların sayısı}$$

w-uygun taşın ağırlığı (örn., piyon için 1)

## Kısıtlar

Örnek: arama için 100 saniyelik zaman tanımıdır. Her saniyede  $10^4$  düğüm araştırılmalıdır  
→ her harekette  $10^6$  düğüm

yaklaşımalar:

### Ⓐ Kesme denemesi (cutoff test):

Derinlik sınırı

### Ⓐ Değerlendirme fonksiyonları

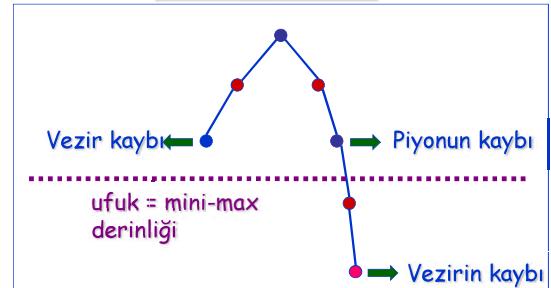
## Kesmekte arama (cutting off)

Aşağıdaki değerlerle çalışmak mümkün mü?  
 $b^m = 10^6$ ,  $b=35 \rightarrow m=4$

Yalnız 4 hamle ileriyi görmek satranç oyuncusu için başarısızlıktır!

- 4 hamle  $\approx$  acemi oyuncu
- 8 hamle  $\approx$  tipik bir program, usta oyuncu
- 12 hamle  $\approx$  Deep Blue, Kasparov

## Ufuk etkisi



Derinine ilerlemekle  
felaketi önlüyor olamayız de onu geciktirebiliriz

→ çözüm: sezgisel devam

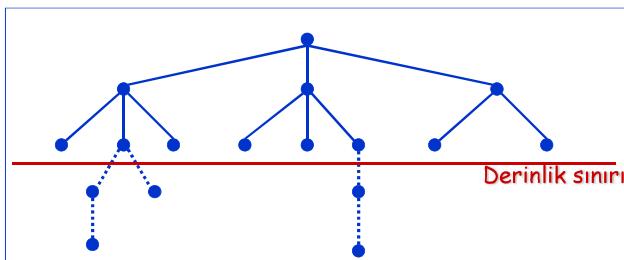
37

38

## Sezgisel Devam

Stratejik durumlarda çok önemli  
oyun taşının kaybı, piyonun vezire çevrilmesi, ...

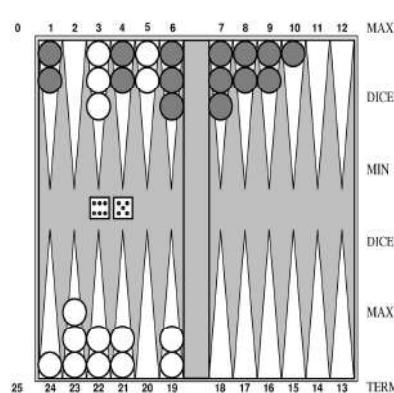
Arama derinlik sınırının dışında da yapmalı!



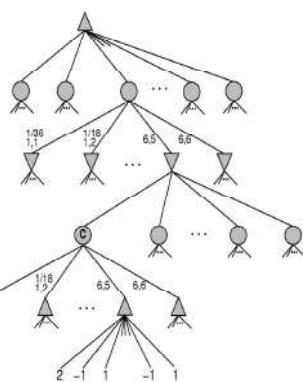
39

## Şans oyunları

Örnek: Tavla:



Oyun ağacının biçimi:



## Sans oyunlarında 'Fayda'nın yayılması

**C düğümü için fayda fonksiyonu**

Di- zarın değeri

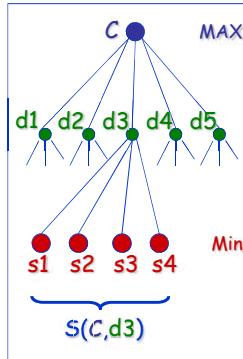
P(di)-Di'nin oluşma olasılığı

S(C,Di)-Di değerinde C'den ulaşılabilen durum

Fayda(s)-s'in değerlendirilmesi

Beklenen\_max( C ) =

$$\sum_i P(d_i) \max_{s \in S(C, d_i)} utility(s)$$



## Bazi oyunları



### BACKGAMMON

- 2 players
- 15 pieces each
- Goal: Move all pieces off the board
- Rules:
  - Dice roll determines number of moves
  - Players move in opposite directions
  - Piece cannot land on a point occupied by 2 or more of opponent's pieces
  - Single piece can be "hit" if landed on by opponent; hit piece must start anew
- Program: TD-Gammon\*
- Web site: [www.research.ibm.com/massdistn.html](http://www.research.ibm.com/massdistn.html)
- Advantage: Too close to call



### BRIDGE

- 4 players in 2 teams
- 13 cards dealt to each player
- Goal: Make 2 "game contracts," or a "rubber"
- Rules:
  - The bid: Each player predicts how many times his or her card will be the highest (a trick)
  - The play: Put down 1 card at a time and compare it with others; this occurs 13 times
  - The scoring: Points scored if bid is met or exceeded; otherwise points go to the opposing team
- Program: GIB\*
- Web site: [www.gibware.com/massdistn.html](http://www.gibware.com/massdistn.html)
- Advantage: Human



### CHECKERS

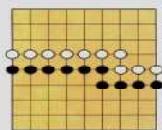
- 2 players
- 12 pieces each
- Goal: Avoid being the player who can no longer move (usually when a player has no pieces left)
- Rules:
  - Move forward on dark diagonal, 1 square at a time
  - Opponent's piece captured when jumped to empty square diagonally behind opponent's piece
  - Capturing a "King": a piece that can move sideways and forward, occurs when piece is moved to opponent's last row
- Program: Chinook
- Web site: [www.cs.ulb.ac.be/~chinook](http://www.cs.ulb.ac.be/~chinook)
- Advantage: Machine



### CHESS

- 2 players
- 16 pieces each (1 king, 1 queen, 2 rooks, 2 bishops, 2 knights, 8 pawns)
- Goal: Capture opponent's king (checkmate)
- Rules:
  - Pieces are captured when landed on by opponent's piece
  - Type of piece dictates movement options
- Program: Deep Blue
- Web site: [www.chess.ibm.com/massdistn.html](http://www.chess.ibm.com/massdistn.html)
- Advantage: Too close to call

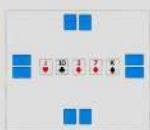
## Bazi oyunları



- GO**
- Black-and-white stones
  - Grid size of board can vary; typical game is on 19-by-19 grid points
  - Game ends when a larger part of the board (conquered part) encompasses stones placed on board plus stones that could be added safely—that is, within the player's walls
  - Rules:
    - Both sides alternate in placing stones on the board
    - Stones surrounded by an opponent's stones are captured and removed from the board
  - Program: Hanfalk\*
  - Web site: [www.wetwind.com/go](http://www.wetwind.com/go)
  - Advantage: Human, by a huge margin



- OTHELLO**
- 2 players
  - Black-and-white disks
  - Goal: Have most disks on the board at the end of the game
  - Rules:
    - Players alternate placing disks on unoccupied board squares
    - If opponent's disks are trapped between other player's disks, opponent's disks are flipped to the other player's color
  - Program: Logistello
  - Web site: [www.neci.nj.nec.com/~hwangpage/otello.html](http://www.neci.nj.nec.com/~hwangpage/otello.html)
  - Advantage: Machine



- POKER (Texas Hold 'Em)**
- 3 to 20 players
  - 2 cards dealt to each player; 5 cards placed in center of table
  - Goal: Obtain the best hand and win the "pot"
  - Rules:
    - 5 center (community) cards start face down
    - First round of betting ensues; 3 community cards are turned over
    - Subsequent rounds of betting ensue: 4th and 5th community cards turned over
    - Player may discard 5 from the community cards and then draw 2 to obtain identical kinds of cards (pairs, 3- and 4-of-a-kind), flushes (all same suit), straights (sequential) or their combinations
    - Final round of betting ensues
  - Program: LOKI\*
  - Web site: [www.cs.ulb.ac.be/~games/poker](http://www.cs.ulb.ac.be/~games/poker)
  - Advantage: Human, by a huge margin



- SCRABBLE**
- 2 to 4 players
  - 100 tiled letters
  - Goal: Accumulate most points by creating high-scoring words
  - Rules:
    - Each player draws 7 letters
    - Each letter has a value
    - Squares on the board have values
    - Words created must join an array
  - Program: Maven\* (used in Scrabble CD-ROM)
  - Web site: [www.hasbroscrabble.com/cdcd.html](http://www.hasbroscrabble.com/cdcd.html)
  - Advantage: Machine, by a slight margin

## Satranç ustası ve satranç programı arasındaki farklar:

1. Deep Blue saniyede 200,000,000 'in üzerinde pozisyonu değerlendirebilir

Bir usta ise saniyede 3 pozisyon değerlendirebilir

2. Satranç programının bilgisi azdır, ama hesaplama yeteneği çok yüksektir

Ustanın çok yüksek satranç bilgisi var, ama hesaplama yeteneği sınırlıdır.

3. İnsan satranç oynadığı zaman hislerinden, öneşzilerinden yararlanıyor.

Programın duyma, sezme yeteneği yoktur.

\*Indicates commercial software that runs on personal computers

## Satranç ustası ve satranç programı arasındaki farklar:

4. İnsan kendi hatalarından ve başarılarından öğrenebilme yeteneğine sahiptir.

Deep Blue, bugünkü haliyle, öğrenme sistemi değildir; bu nedenle, rakibinden öğrenmek veya satranç tahtasındaki durumu "düşünmek" için yapay zeka kullanma yeteneğine sahip değildir.

5. Programın korku duygusu, fikrinin dağıtilması endişesi yoktur (örneğin, Kasparov'un sabit bakişlarından).

Bir ustanın ise insanı zafiyeti var, canı sıkılabilir, kafa dağılabilir vs.

6. Programın oyun anlayışındaki değişimler, geliştirme ekibi tarafından yapılmalıdır

Usta ise her oyundan önce, sonra, oyun içinde oyununda değişiklik yapabilir.

## Satranç ustası ve satranç programı arasındaki farklar:

7. İnsan rakibini değerlendirebilir, onun zayıf yönlerini öğrenebilir ve bundan yararlanabilir.

Program ise satranç pozisyonlarını çok iyi değerlendirse de rakibinin zayıf yönlerinden yararlanamaz.

8. İnsan, değerlendirebildiği pozisyonlar içinden seçim yapar

Program ise mümkün pozisyonlar içinden en iyisini seçebiliyor (Deep Blue saniyede 200 milyon pozisyon içinde arama yapabiliyor)