

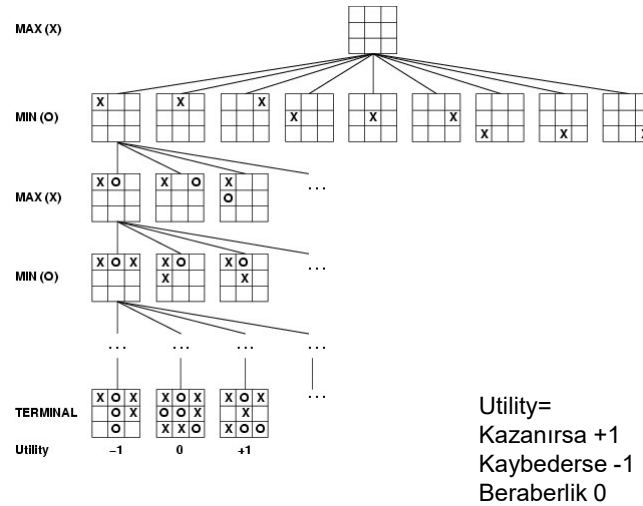
# Oyunlar

## Oyun türleri

	deterministic	chance
perfect information	chess, checkers, go, othello	backgammon monopoly
imperfect information		bridge, poker, scrabble nuclear war

imperfect information= eksik bilgi Ör: karşı tarafın elindekini bilmeme

## Oyun ağacı – Game tree (2-oyuncu)

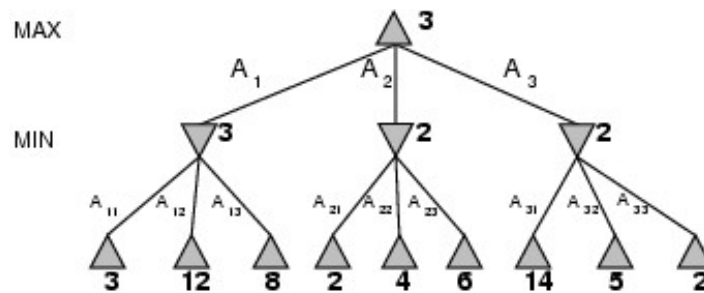


Mehmet Fatih AMASYALI Yapay Zeka Ders Notları

YILDIZ TEKNİK ÜNİVERSİTESİ BİLGİSAYAR

## Minimax

- Rasgele olmayan oyunlar için ideal
- Yaklaşım: En yüksek **minimax değerini** sağlayan hamleyi yap
- Örnek: 2-oyunculu oyun:



Mehmet Fatih AMASYALI Yapay Zeka Ders Notları

YILDIZ TEKNİK ÜNİVERSİTESİ BİLGİSAYAR

## Minimax bileşenleri

- MAX = Bizim olası hamlelerimizden kendi açımızdan en iyi sonucu verenin seçilmesi
- MIN = Rakibin olası hamlelerinden kendi açısından en iyi sonucu veren, bizim açımızdan en kötü sonucu verenin seçilmesi
- UTILITY = Oyunun sonucunu gösteren durum (Ör: kazanırsak +1, berabere 0, kaybedersek -1)

## Minimax değeri

$$\text{MINIMAX-VALUE}(n) = \begin{cases} \text{UTILITY}(n) & \text{eğer } n \text{ oyun sonu seviyesinde ise} \\ \max ( \text{Ardıl} (n) ) & \text{eğer } n \text{ MAX seviyesinde ise} \\ \min ( \text{Ardıl} (n) ) & \text{eğer } n \text{ MIN seviyesinde ise} \end{cases}$$

Ardıl (n) = n durumundan gidilebilecek tüm durumların değerleri

# Minimax algoritması

```

function MINIMAX-DECISION(state) returns an action
   $v \leftarrow \text{MAX-VALUE}(\textit{state})$ 
  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)
   $v \leftarrow -\infty$ 
  for a, s in SUCCESSORS(state) do
     $v \leftarrow \text{MAX}(v, \text{MIN-VALUE}(s))$ 
  return v



---


function MIN-VALUE(state) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
   $v \leftarrow \infty$ 
  for a, s in SUCCESSORS(state) do
     $v \leftarrow \text{MIN}(v, \text{MAX-VALUE}(s))$ 
  return v

```

## Minimax özellikleri

- Bütünlük? Evet (eğer ağaç sınırlı ise)
- Zaman karmaşıklığı?  $O(b^m)$
- Bellek karmaşıklığı?  $O(bm)$  (derinlemesine arama)
- En iyi çözüm? Evet (optimal –kendisi için her zaman en iyisini yapan- rakibe karşı)

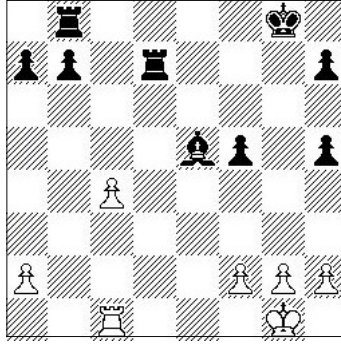
## Satrançta Sınırlar

- Arama uzayının büyüklüğü ( $32^{40}$ )
  - Ortalama hamle sayısı = 40
  - Her adımda yapılabilecek farklı hamle sayısı ortalaması = 32
  - $32^{40} = 2^{200} \approx 10^{60}$
- Saniyede 3 milyar durum işlersek
  - Bir yıldaki saniye sayısı  $\approx 32 \cdot 10^6$
  - Bir yılda işlenecek durum sayısı  $\approx 10^{17}$
  - Tüm durumların değerlendirilmesi  $10^{43}$  yıl sürer.
  - Evrenin yaşı  $\approx 10^{10}$  yıl

## Sınırları Aşmak İçin

- Sınırlı Derinlikte Arama ve değerlendirme fonksiyonu (cutoff test, evaluation function)
- Alfa beta budaması

## Satranç için değerlendirme fonksiyonu



- Taş değerleri:

Piyon=1, at=fil=3, kale=5, vezir=9

- Siyahlar:

– 5 piyon, 1 fil, 2 kale

- $\text{Skor} = 1 \cdot (5) + 3 \cdot (1) + 5 \cdot (2)$   
 $= 5 + 3 + 10 = 18$

- Beyazlar:

– 5 piyon, 1 kale

- $\text{Skor} = 1 \cdot (5) + 5 \cdot (1)$   
 $= 5 + 5 = 10$

Bu durumun iki taraf için skorları:

siyahlar için =  $18 - 10 = 8$

beyazlar için =  $10 - 18 = -8$

## Değerlendirme fonksiyonları

- Oyunun sonucunun çok derinlerde olduğu durumlarda durumların değerlendirilmesi için kullanılan fonksiyonlardır.

- Satranç için bu fonksiyon genellikle önceden belirlenen **özniteliklerin doğrusal** toplamı olarak düşünülür.

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

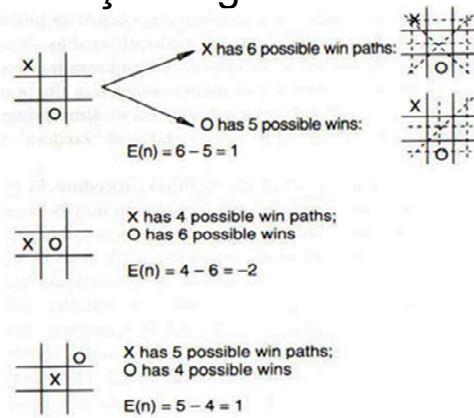
- Örnek,  $w_1 = 9$  ve
- $f_1(s) = (\text{beyaz vezir sayısı}) - (\text{siyah vezir sayısı})$ , vs.
- Ağırlıklı toplam, bileşenlerin birbirinden bağımsız olduğunu varsayar.

## Satranç için değerlendirme fonksiyonu

- **Asıl iş: f'leri ve w'leri belirlemek:**

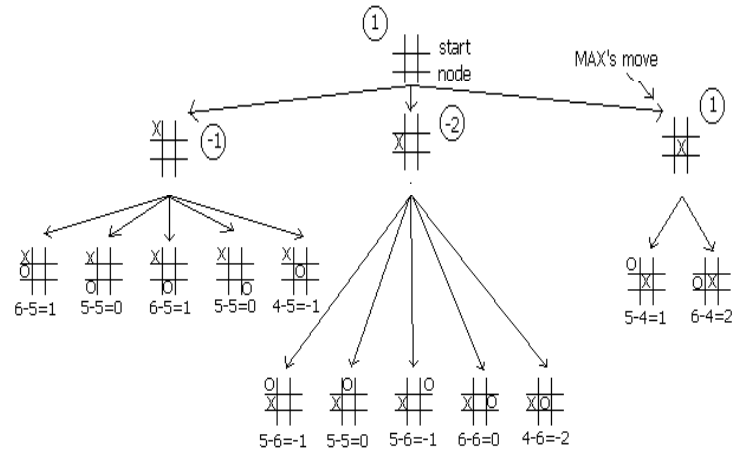
- Etrafı boş olan piyonlar kötüdür.
- Şahın korumaları var mı?
- Hakeret kabiliyetin nasıl?
- Tahtanın ortasının kontrolü sende mi?
- Oyun süresince w lerin değerleri değişir mi?

## Tic-Tac-Toe için Değerlendirme fonksiyonu

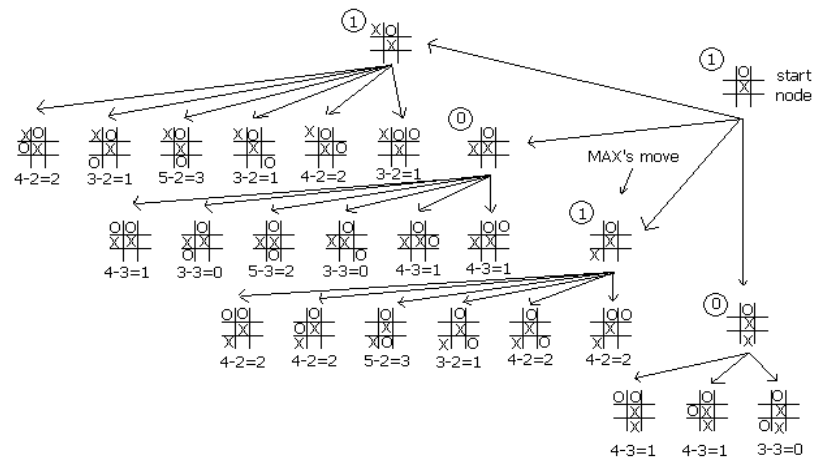


Heuristic is  $E(n) = M(n) - O(n)$   
 where  $M(n)$  is the total of My possible winning lines  
 $O(n)$  is total of Opponent's possible winning lines  
 $E(n)$  is the total Evaluation for state  $n$

## Tic-Tac-Toe için Değerlendirme fonksiyonu (ilk hamlemiz, 2 derinlikli arama)



## Tic-Tac-Toe için Değerlendirme fonksiyonu (2. hamlemiz, 2 derinlikli arama)





## Sınırlı Derinlikte Arama'da Ufuk Etkisi - Horizon Effect



Aramayı farklı derinliklerde yapmanın etkisi:  
C2'deki beyaz kale a2'deki piyonu alır mı?

1 derinlik  
Bir piyon kazançta olduğundan alır.

2 derinlik  
Bir piyona bir kale değmez. Almaz.

3 derinlik  
Bir piyon + bir kaleye, bir kale değer. Alır.

4 derinlik  
bir piyon + bir kaleye, 2 kale değmez. Almaz.

## Sınırlı Derinlikte Arama (Cutoff search)

*MinimaxCutoff* *MinimaxValue* ile aynıdır.

1. *Terminal?* yerine *Cutoff?* Kullanılır.
2. *Utility* yerine *Eval* kullanılır.

Satrançta yeterli mi?

Hamle yapma süresi eğer 100 saniye ise ve  $10^4$  düğüm/saniye hızda inceleme yapabiliyorsak  
→ her hareket için  $10^6$  düğüm inceleyebiliriz.

$b^m = 10^6$  ile sınırlıysa  $b=35$  ise  $m=4$  olabilir.

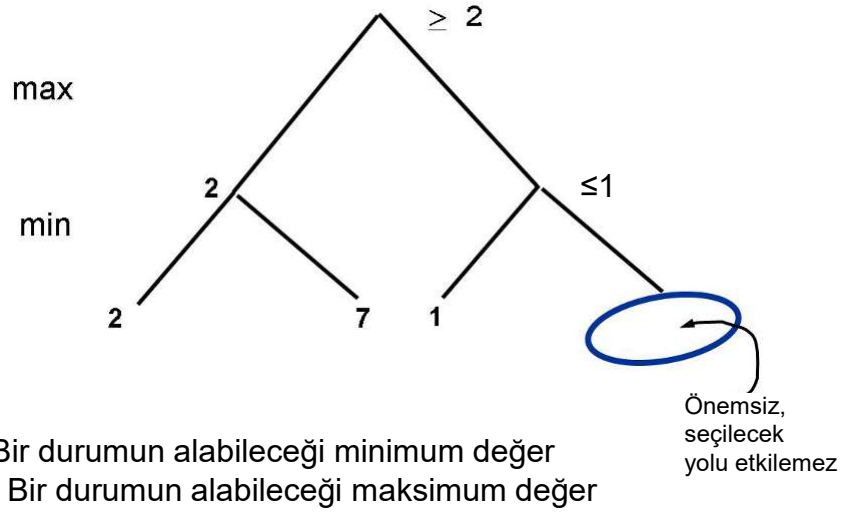
4-katlı ileri-bakış (lookahead) → amatör bir satranç oyuncusu

4 kat ileri bakan bir program en iyi ihtimalle amatör oyuncularını yenebilir.

- 4-kat ≈ amatör oyuncu - insan
- 8-kat ≈ master seviyesinde insan
- 12-kat ≈ Deep Blue, Kasparov

Çözüm: Sınırlı derinlikte aramaya alfa beta budama eklemek

## Tüm durumları değerlendirmek gerekir mi? $\alpha$ - $\beta$ budama örneği

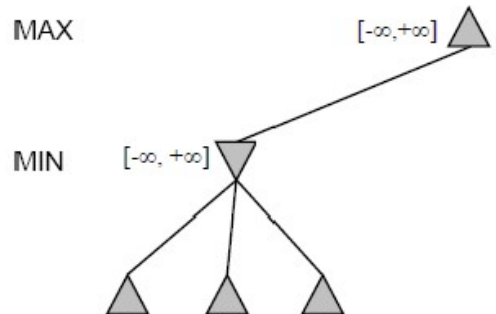


Mehmet Fatih AMASYALI Yapay Zeka Ders Notları

YILDIZ TEKNİK ÜNİVERSİTESİ BİLGİSAYAR

## $\alpha$ - $\beta$ budama örneği

Depth first le ilk yaprağa kadar in.

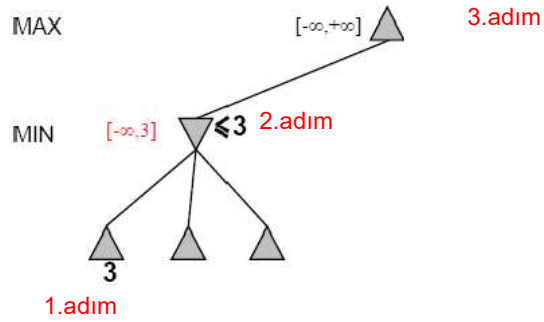


Mehmet Fatih AMASYALI Yapay Zeka Ders Notları

YILDIZ TEKNİK ÜNİVERSİTESİ BİLGİSAYAR

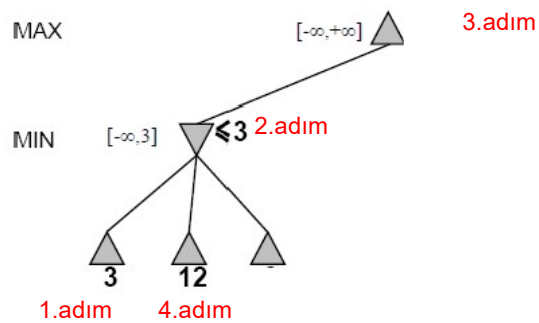
## $\alpha$ - $\beta$ budama örneği

İlk yaprağı değerlendir buna göre [alfa, beta] değerlerini güncelle.



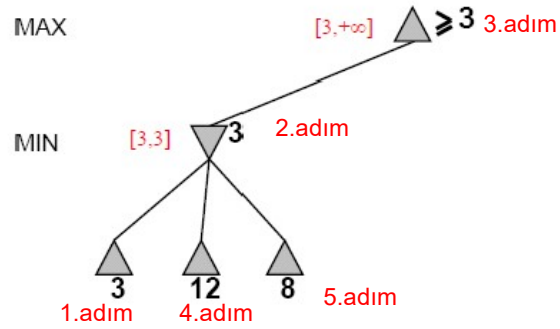
## $\alpha$ - $\beta$ budama örneği

İkinci yaprağı değerlendir buna göre alfa, beta değerlerini güncelle.



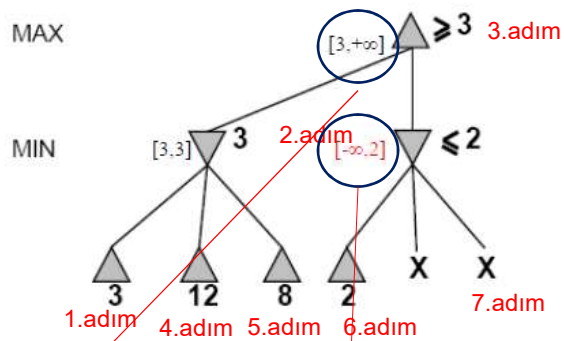
## $\alpha$ - $\beta$ budama örneği

3.yaprağı değerlendir buna göre alfa, beta değerlerini güncelle.



## $\alpha$ - $\beta$ budama örneği

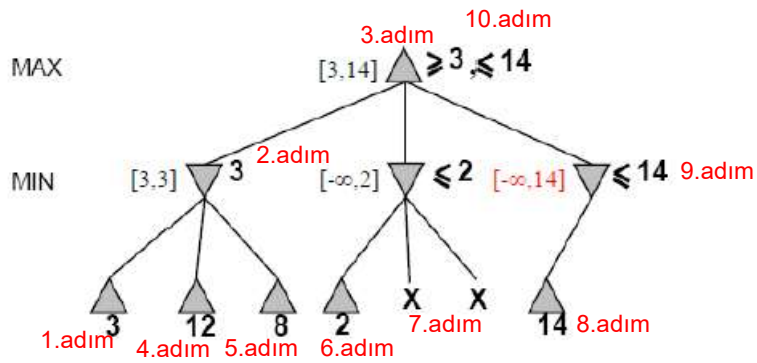
4.yaprağı değerlendir buna göre alfa, beta değerlerini güncelle.



7.adım: En küçük değer 3, altının en büyük değeri 2, o halde X ile işaretli lerin değerlerini bulmaya gerek yok, Zaten alfa ve beta yı değiştiremezler

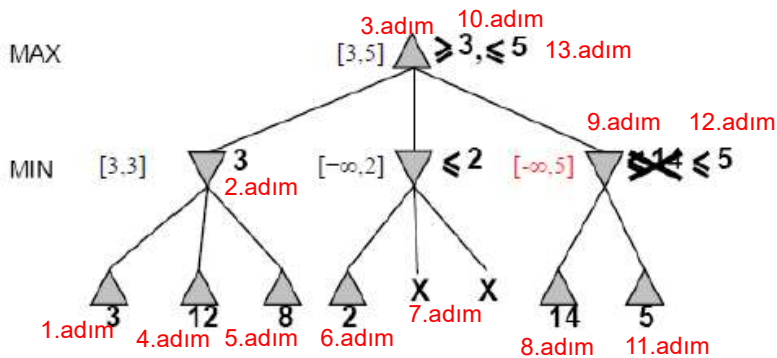
## $\alpha$ - $\beta$ budama örneği

7.yaprağı değerlendir buna göre alfa, beta değerlerini güncelle.



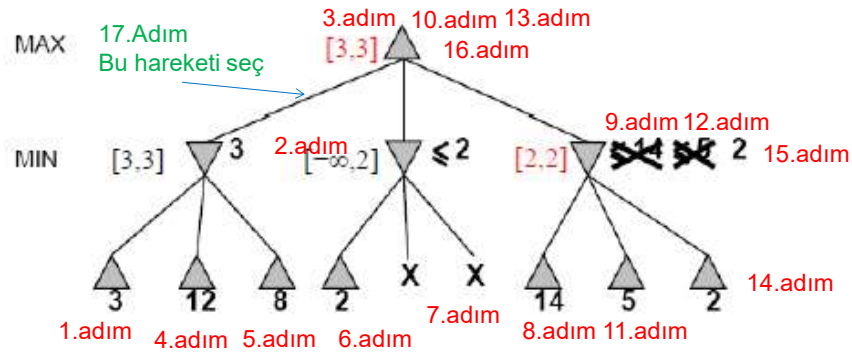
## $\alpha$ - $\beta$ budama örneği

8.yaprağı değerlendir buna göre alfa, beta değerlerini güncelle.



## $\alpha$ - $\beta$ budama örneği

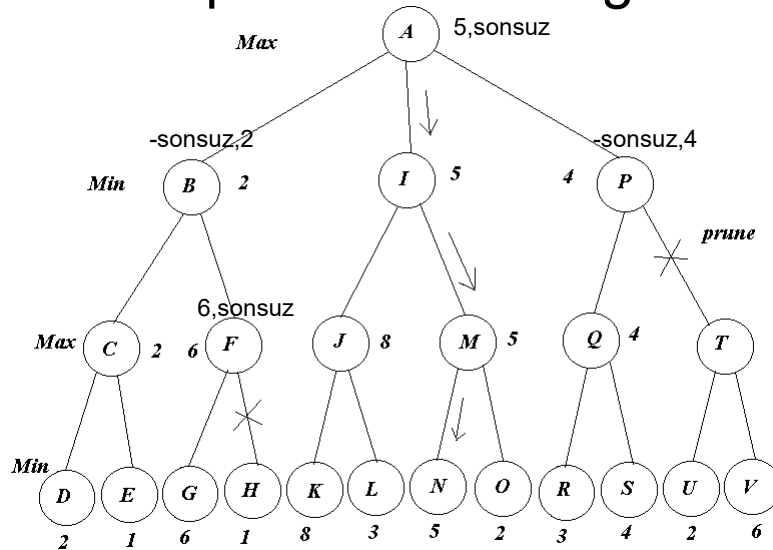
9.yaprağı değerlendir buna göre alfa, beta değerlerini güncelle.



Mehmet Fatih AMASYALI Yapay Zeka Ders Notları

YILDIZ TEKNİK ÜNİVERSİTESİ BİLGİSAYAR

## $\alpha$ - $\beta$ budama örneği



**Adım adım çözünüz**

Mehmet Fatih AMASYALI Yapay Zeka Ders Notları

YILDIZ TEKNİK ÜNİVERSİTESİ BİLGİSAYAR

## $\alpha$ - $\beta$ özellikleri

- Budama seçilen hareketi **etkilemez**.
- Karmaşıklığı  $O(b^{m/2})$  'ye kadar düşürebilir.  
 → arama derinliğini iki katına çıkarır.  
 $4 \cdot 2 = 8$  (satrançta master seviyesi)

## $\alpha$ - $\beta$ algoritması

```

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

```

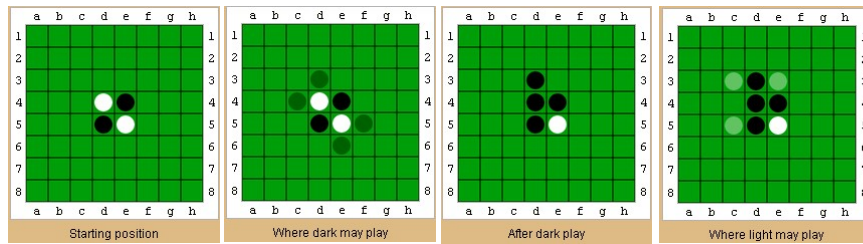
## Rastgele olmayan oyun uygulamaları

- Dama (Checkers): Chinook 1994'de dünya şampiyonu Marion Tinsley ile başabaş oynadı, aynı yıl Tinsley sağlık sebeplerinden oyunlardan çekildi ve Chinook o zamandan beri dünya şampiyonu. Günümüzde dama tamamen çözülmüşü durumda.
- Satranç: Deep Blue 1997'de 6 oyun sonunda dünya satranç şampiyonu Garry Kasparov'u yendi. Deep Blue saniyede 200 milyon düğümü arayabilmekte, oldukça iyi bir değerlendirme fonksiyonu kullanmakta, gerektiğinde 40-kat hamle ileriye görebilecek derecede arama yapabilmektedir.



## Rastgele olmayan oyun uygulamaları

- Othello (reversi): Günümüzde insan dünya şampiyonları bilgisayarlara karşı yarışmamaktadırlar, bilgisayarlar bu alanda çok üstündür.

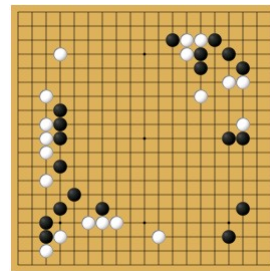
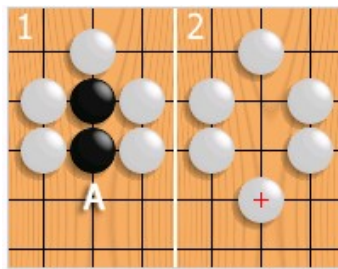


Mehmet Fatih AMASYALI Yapay Zeka Ders Notları

YILDIZ TEKNİK ÜNİVERSİTESİ BİLGİSAYAR

## Rastgele olmayan oyun uygulamaları

- Go: İnsanlar anca 2016'da üstünlüklerini kaybettiler. Bunun sebebi 19x19'luk oyun tahtasında  $b \sim 250$ ,  $d \sim 150$  olmasıdır. Örüntü tanıma teknikleri daha yaygın kullanılmaktadır. Ağustos 2008'de her biri 32 işlemciye sahip 25 sunucuda çalışan Mygo profesyonel bir go oyuncusunu yendi. Mart 2016'da AlphaGo dünya şampiyonunu 4-1 yendi.



Mehmet Fatih AMASYALI Yapay Zeka Ders Notları

YILDIZ TEKNİK ÜNİVERSİTESİ BİLGİSAYAR

## Bilgisayarla satranç oynamak

- 1957 - Newell ve Simon: 10 yıl içinde dünya satranç şampiyonunun bir bilgisayar olacak
- 1958 : Satranç oynayan ilk bilgisayar IBM 704
- 1967 : Mac Hack programı insanların katıldığı bir turnuva da başarıyla yarıştı
- 1983 : Belle programı, Amerika Satranç Federasyonundan master ünvanını aldı.
- 1980'lerin ortaları : Carnegie Mellon üniversitesinde bilim adamları sonradan Deep Blue'ya dönüşecek çalışmayı başlattı.
- 1989: Projeyi IBM devraldı.

## Bilgisayarla satranç oynamak

- 11 Mayıs 1997, Gary Kasparov, 6 oyunluk maçta Deep Blue'ya 3.5 a 2.5 yenildi.
  - 2 oyun deep blue, 1 oyun Kasparov aldı, 3 oyun berabere

<http://www.research.ibm.com/deepblue/meet/html/d.3.html>



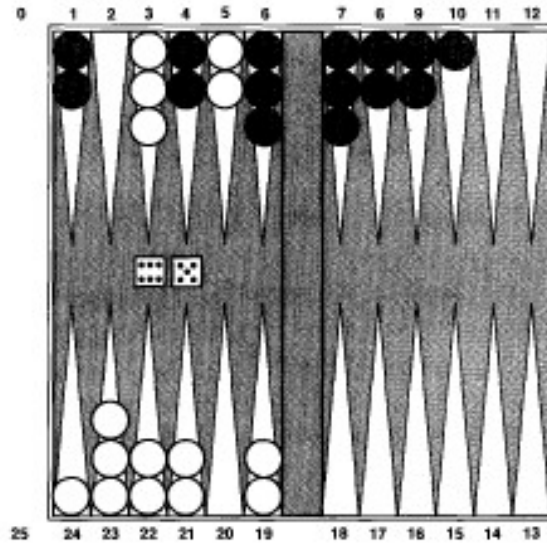
## Metotların Tarihi

•Shannon, Turing Minimax arama		1950
•Kotok/McCarthy Alpha-beta budama		1966
•MacHack	Dönüşüm tabloları	1967
•Chess 3.0+	Iterative-deepening	1975
•Belle	Özel donanım	1978
•Cray Blitz	Paralel arama	1983
•Hitech	Paralel değerlendirme	1985
•Deep Blue	yukarıdakilerin hepsi	1997

## İçinde şans faktörü olan oyunlar

- Tavla – atılan zara göre oyun oynanır.
- Kağıt oyunları – kağıtların dağılımına göre oyun oynanır.
- Oyun ağacını oluşturmak için her türlü varyasyonun göz önüne alınması gerekir.

## Tavla



Mehmet Fatih AMASYALI Yapay Zeka Ders Notları

YILDIZ TEKNİK ÜNİVERSİTESİ BİLGİSAYAR

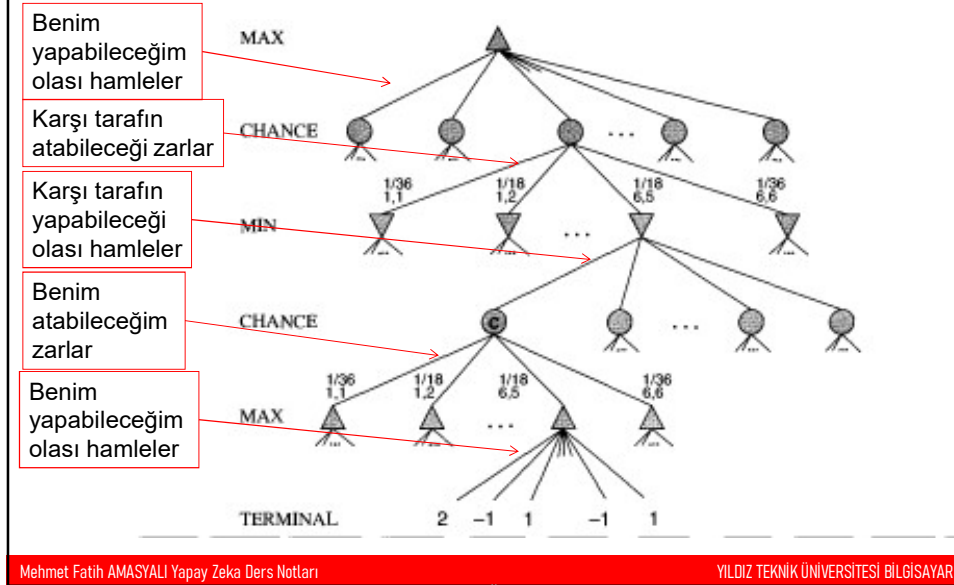
## Expectiminimax bileşenleri

- Terminal (Sonuç) düğümü
- min düğümü
- max düğümü
- Şans düğümü
  - Şans düğümünde gelebilecek zar, kağıt, vs. gibi durumlar olasılıklarına göre dizilirler. Örnek: Tavlada atılan çift zar için 21 değişik durum (düğüm) oluşabilir. (6 aynı çift zar, 15 değişik çift zar), yani her seviye için 21 şans düğümü vardır.

Mehmet Fatih AMASYALI Yapay Zeka Ders Notları

YILDIZ TEKNİK ÜNİVERSİTESİ BİLGİSAYAR

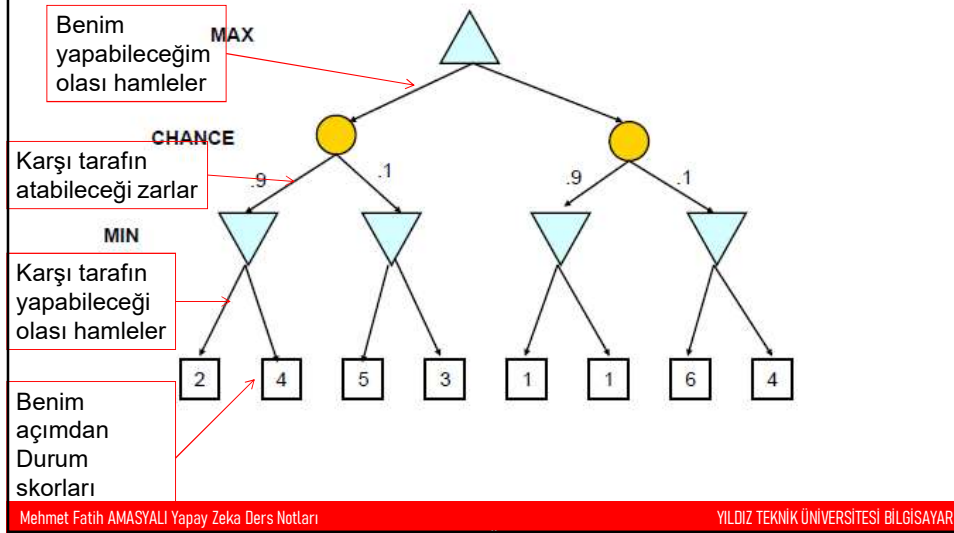
## Expectiminimax



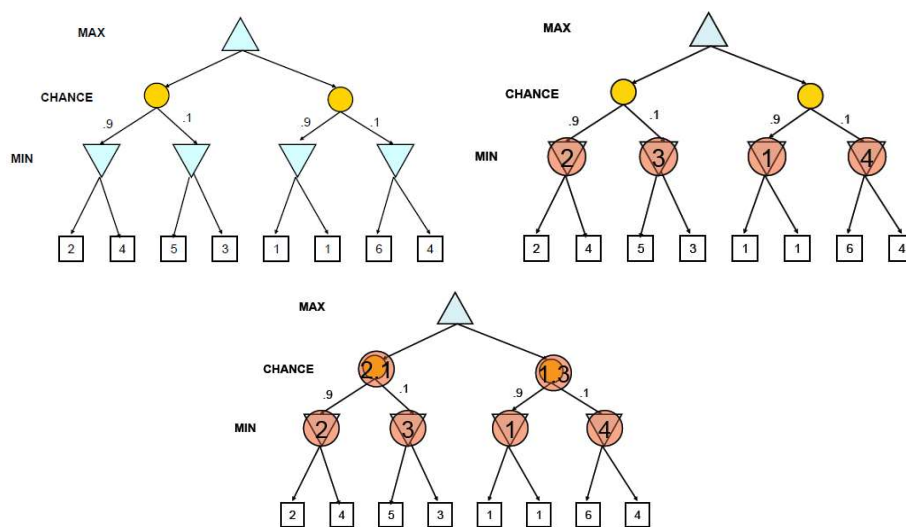
## Expectiminimax

- EXPECTIMINIMAX(n) =
  - UTILITY(n)      eğer n oyun sonu (terminal) durumu ise
  - max ( Ardıl (n) )      eğer n MAX seviyesinde ise
  - min ( Ardıl (n) )      eğer n MIN seviyesinde ise
  - $\sum P(s) * \text{Ardıl}(s)$       eğer n şans düğümü ise

## Expectiminimax Örnek



## Expectiminimax Örnek



## Expectiminimax özellikleri

- Bütünlük? Evet (eğer ağaç sınırlı ise)
- En iyi çözüm? Evet (şans faktörü göz önünde olmalı)
- Zaman karmaşıklığı?  $O(b^m n^m)$
- Bellek karmaşıklığı?  $O(n^m b m)$  (derinlemesine arama)
- $n \rightarrow$  şans faktörünün dallanma sayısı
- Tavla için  $b \approx 20$ ,  $n = 21 \rightarrow$  tam çözüm şu anki şartlarda imkansız. En fazla 3-seviye pratikte mümkün.
- TDGammon adlı oyun 2 derinlikli arama yapar. Çok iyi bir değerlendirme fonksiyonu var. Sonuç: dünya şampiyonu

## Tahmin ve Öğrenme

- Bilgisayar kullanıcının hareket loglarını kaydedebilir.
- Kullanıcının bir sonraki hareketini bu loglara göre tahmin edebilir. Tüm olasılıkları hesaba katmaya gerek kalmaz.
- Bu tahmine göre yapması en iyi hareketi yapabilir.
- Önceki hareketler
 

Low Kick, Low Punch, Uppercut	10	50%
Low Kick, Low Punch, Low Punch	7	35%
Low Kick, Low Punch, Low Kick	3	15%
- Şimdiki durum:  
Low Kick, Low Punch, ?
- Ör: Virtual Fighter 4

## Değerlendirme fonksiyonunu öğrenmek

- Deep Blue durum değerlendirme fonksiyonundaki ağırlıkları ( $w_1, w_2, \dots, w_n$ ) öğrenmiştir.

$$f(p) = w_1 f_1(p) + w_2 f_2(p) + \dots + w_n f_n(p)$$

- Nasıl?

## Değerlendirme fonksiyonunu öğrenmek

- Grand master seviyesindeki oyuncuların 1000'den fazla oyunundaki her hamlenin yer aldığı “şu durumda şunu yaptı” database’i
- Tüm hamleler için, önceki durumdan gidilebilecek tüm durumların içinde yapılanın (database’de yer alanın) seçileceği ( $f(p)$ ’si büyük) şekilde  $w$ ’lerin düzenlenmesi



## Kendi kendine oynamak

- Monte Carlo Tree Search
- Yapılacak hamleyi seçerken rasgele hareketlerle oyun sonuna kadar ilerler. Oyunun sonucuna göre yaptığı hamleyi değerlendirir.

## Monte Carlo Arama Ağacı

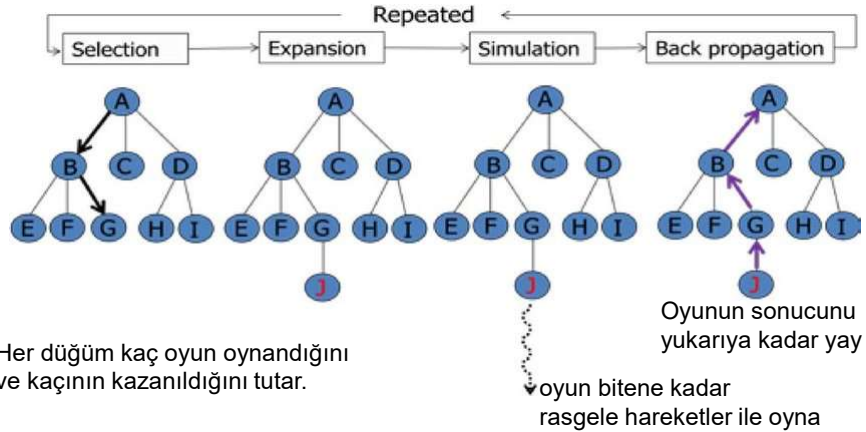
- 2006 başlarında Go ile popüler olmaya başladı.
- Büyük arama uzayları için tasarlanmış
- Sınırlı bir derinlikteki her olasılığı denemek yerine oyun bitene kadar rasgele hamleler üretir.

[\*] <https://dergipark.org.tr/tr/download/article-file/486013>

[\*] <https://www.youtube.com/watch?v=Fbs4InGLS8M>

## Monte Carlo Arama Ağacı

- A'dayız yapabileceğimiz 3 hareket var. hangisini yapmak iyi?



[\*] Yen, Shi-Jim & Chou, Cheng-Wei & Chen, Jr-Chang & Wu, I-Chen & Kao, Kuo-Yuan. (2014). Design and Implementation of Chinese Dark Chess Programs. IEEE Transactions on Computational Intelligence and AI in Games. 7. 1-1. 10.1109/TCIAIG.2014.2329034.

Mehmet Fatih AMASYALI Yapay Zeka Ders Notları

YILDIZ TEKNİK ÜNİVERSİTESİ BİLGİSAYAR

## Monte Carlo Arama Ağacı

- Seçimde ikilem (az oyun oynanmışlar vs. büyük oranda kazanmışlar)
- Seçimde sezgisel yaklaşımlar kullanılıyor. Rasgele seçim uygulanabilir değil.
  - Upper Confidence Bound for Trees
    - Her düğüm için
    - kazanılan oyun / sonrasında oynanan oyun oranı
    - Tüm oynanan oyun sayısı tabanlı bir formül
  - CNN
    - Oyun tahtası (düğüm) CNN'e verilir, ağ tahtanın değerini tahmin eder

[\*] <https://nikcheerla.github.io/deeplearningschool/2018/01/01/AlphaZero-Explained/>

Mehmet Fatih AMASYALI Yapay Zeka Ders Notları

YILDIZ TEKNİK ÜNİVERSİTESİ BİLGİSAYAR

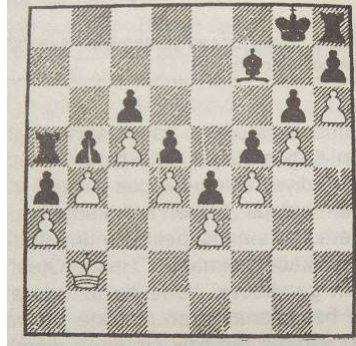
## Monte Carlo Arama Ağacı

- Oyun algoritmalarında temel yaklaşım
- Kolaylıkla paralelleştirilebilir (kökten ya da yapraktan)
- Uygulama alanları: Bilgisayarlara karşı oynanabilecek her türlü oyun, bilgisayarla taklit edilebilen dünyalardaki arama problemleri (n eklemlı bir robot kolunu istediğimiz konuma getirmek için gerekli hareketler)

## Sorular

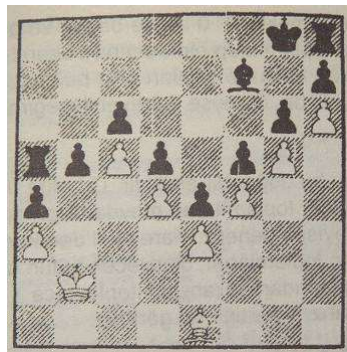
- Deep Blue'nun yaptığı şey sadece tasarımcıların fikirlerini büyük hesap kabiliyetiyle uygulamak. Yenen tasarımcılar ve teknoloji. Yapılan sadece özel bir hesap makinesi.
- Drew McDermott:
  - Deep Blue'nun gerçekte satrançtan anlamadığını söylemek, bir uçağın kanatlarını çırpmadığı için uçmadığını söylemeye benzer.
- Uzaylılar dünyaya gelse ve bize satrançta meydan okusalar
  - Kapişmaya Deep Blue'yu mu Kasparov'u mu gönderirdiniz?

## Deep Thought kaleyi aldı!



- Oyun sırası beyaz'da
- Çözüm yeni bir f'in eklenmesi

## Deep Thought kaleyi yine aldı!



- Oyun sırası beyaz'da
- Çözüm yeni bir f'in daha eklenmesi
- Nereye kadar?

## Özet

- Oyunlar zevklidir.
- YZ hakkında **pratik** yeni fikirler üretirler.
- Hesaplama karmaşıklığı sebebiyle mükemmellik genelde sağlanamaz. Bunun yerine yaklaşımlar, kestirimler kullanılır.
- Günümüzde RL ve MCTS popüler.

## Referanslar

- <http://www.cs.pomona.edu/~sara/classes/cs030-spring13/notes/abslides.pdf>
- <https://deepmind.com/research/alphago/>