Sezgisel (Bilgili) arama Yöntemleri

Konular

Sezgisel arama

- En iyisini arama-Best-first search (istekli en iyisini arama-Greedy best-first search)
- A* arama
- Yerel arama algoritmaları-Local search
  - Dağa Tırmanma-Hill-climbing search
  - Yerel ışın arama-Local beam search
  - Genetik algoritmalar - Genetic algorithms

Sezgisel Arama Yöntemleri

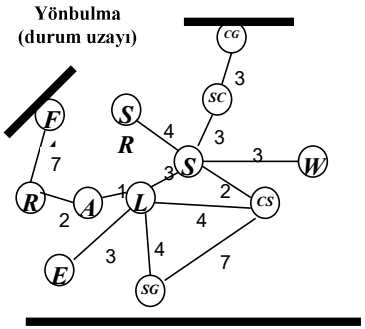
- Kör arama yöntemleri basittir, fakat çoğu zaman pratik değildir.
- **Kör arama yöntemleri bilgisiz** yöntemlerdir. Yani, bu yöntemlerle arama, durum uzayı hakkında bilgi olmadan gerçekleştirilir.
- Sezgisel arama yöntemleri , ilk önce **en umut verici yolu** incelemekle aramanın etkisini yükseltiyor

Değerlendirme fonksiyonu

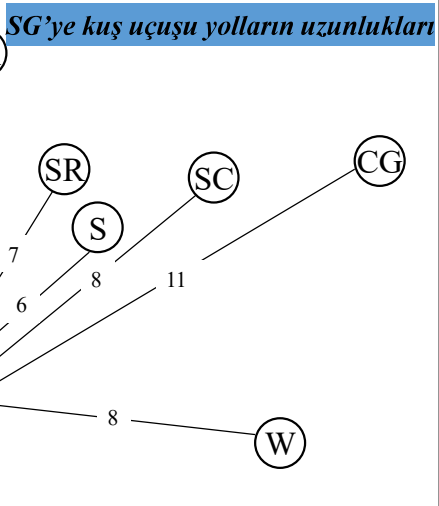
Sezgisel aramayı kullanmak için bize değerlendirme fonksiyonu (evaluation function) gerekmektedir. Değerlendirme fonksiyonu , hedef/amaç durumuna yakınlığı değerlendirmek için arama ağacında düğümleri inceler

- **Sezgisellik bir tahmindir, fakat aramayı gerçekleştirmek için yararlı bir yol olabilir.**
- **Temel düşünce:**
  - Tüm mümkün arama yollarını denemek yerine, hedefe/amaca yaklaştırdığı düşünülen yolları denemek

Sezgisel Arama



- Örneğin, arama uzayında kentler arasındaki kuş uçuşu mesafeler, sezgisel değerlendirme için kullanılabilir. Üstünlük verilen yollar SG'ye daha yakın olan yollardır.



Sezgisel Arama Algoritması

- Sezgisel değerlendirme fonksiyonu, yeni yolların arama kuyruğuna eklenmeden önce incelenmesini sağlar

HeuristicSearch (initial, goal, queuing-fn, eval-fn)

```
Sıfır uzunluklu, yalnız başlangıç
durumu içeren düğümden oluşan
kuyruğu oluşturmalı (ağacın kökü)

Kuyruktaki birinci yol amaç durumda
sonlanana dek veya kuyruk boş olana
dek aşağıdaki işlemleri yapmalı:

    Birinci yolu, uç durumun tüm
    ardıllarına dek genişletmekle
    yeni yollar oluşturmalı

    Döngülü tüm yeni yolları
    elemeli

    * Yeni yolları değerlendirme
    fonksiyonuna göre sıralamalı

    * Yeni yolları kuyruğa eklemeli

Eğer amaç durum bulunmuşsa arama
başarılıdır, eksi halde arama
başarısız sonuçlanmıştır
```

En İyisini arama algoritmaları

Bazı hallerde amaca doğru herhangi bir yolun bulunması yeterli olsa da, bazı zamanlarda en iyi yolun bulunması gerekebilir.

En hızlı, en düşük maliyetle ve en kolay yolla amaca ulaşılması için optimal arama yapılmalıdır.

En iyisini arama algoritmaları

- Mantıklı bir zaman diliminde en iyi çözümün bulunması yöntemleri:
  - En iyisini arama (Best-first search)
  - A*
- Sezgisel tahminler(değerlendirmeler) kullanılıyor, fakat bu tahminler yakın çözüme doğru değil, genel çözüme doğru yöneltilmiştir

En iyisini arama-Best-first search

- Temel düşünce: değerlendirme fonksiyonu *f(n) kullanmakla her bir düğüm için*
  - "-"arzu edilmenin" (desirability) tahmin edilmesi
  - →En çok arzu edilen düğümü genişletmeli

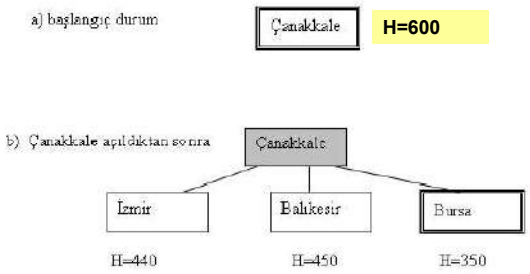
Düğümleri, "arzu edilme"lerine göre azalarak sıralamalı

En iyisini arama

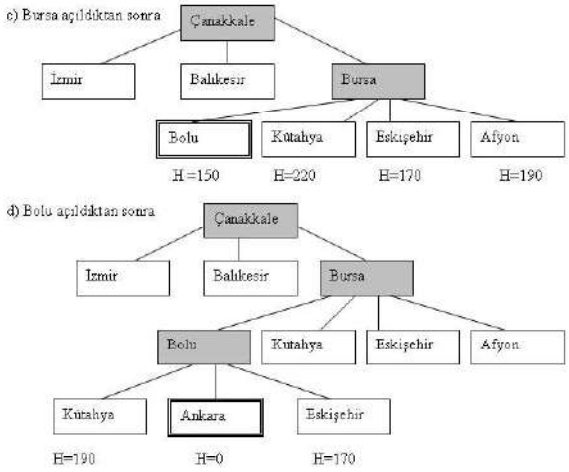
Ankara ile diğer kentler arasındaki kuş uçuşu mesafeler

Çanakkale	600	Eskişehir	170
Bursa	350	Afyon	190
Balıkesir	450	Kütahya	220
Manisa	420	Uşak	250
İzmir	440	Isparta	240
Aydın	430	Konya	200
Denizli	410	Bolu	150

En iyisini arama



En iyisini arama



En iyisini (istekli) arama

- Değerlendirme fonksiyonu *f(n) = h(n)*
- h(n): n düğümünden amaca dek tahmin edilen maliyet (sezgisellik fonksiyonu)

Örnek, *h(n)* = n'den Ankara'ya dek kuş uçuşu mesafesi

Algoritma, amaca en yakın saydığı düğüme doğru genişletme yapıyor

En iyisini arama algoritmasının özellikleri

- Tam? Değil – sonsuz döngüler olabiliyor
- Zaman? $O(b^m)$, iyi bir sezgisel algoritma işlem zamanının küçülmesine neden olabiliyor
- Uzay? $O(b^m)$ – tüm düğümler bellekte tutuluyor
- Optimal? Değil

A* arama

- Temel düşünce: yüksek maliyetli yollara doğru genişletme yapmamalı
- Değerlendirme fonksiyonu *f(n) = g(n) + h(n)*
- Toplam Değer(Çözüm)= Yol Değeri(gidilen) + Tahmin Değer(Kalan)

- *n: aramadaki her hangi durumdur*
- *g(n): başlangıç durumdan n durumuna dek gidilmiş yolun maliyetidir*
- *h(n): n durumundan amaç durumuna dek gereken maliyetin sezgisel tahminidir.*
- *h(n), n durumundan amaç durumuna dek en kısa yolun maliyetine eşit veya ondan küçüktür*

8 taş bulmacası örneği

Değerlendirilecek Durum

2*	8*	3
1*	6*	4
0	7*	5

Amaç Durumu

1	2	3
8		4
7	6	5

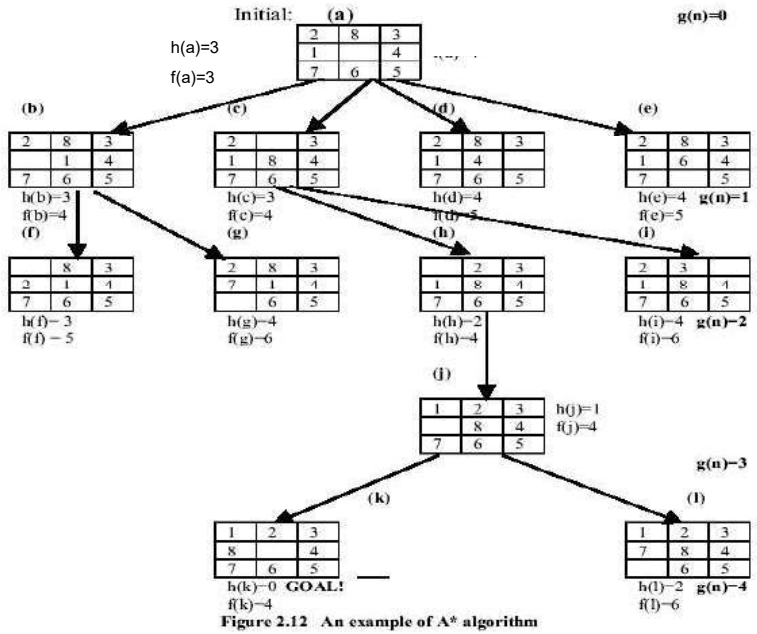


Figure 2.12 An example of A* algorithm

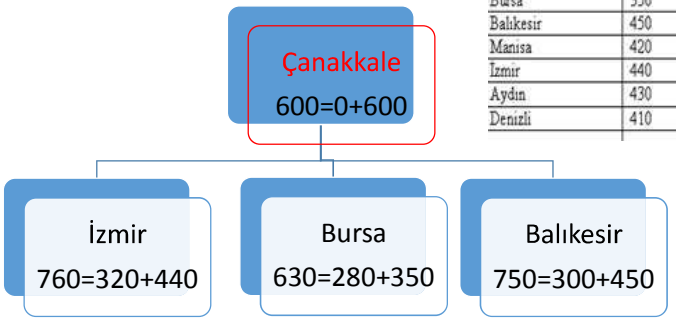
A* arama-örnek

Çanakkale	600	Eskişehir	170
Bursa	350	Afyon	190
Balıkesir	450	Kütahya	220
Manisa	420	Uşak	250
İzmir	440	Isparta	240
Aydın	430	Konya	200
Denizli	410	Bolu	150

Çanakkale
600=0+600

Sol değer (600) : değerlendirme fonksiyonunun değeri
Sağ 1. değer (0): başlangıç durumdan o anki duruma dek harcanan çaba
Sağ 2.değer (600): o anki durumdan amaç durumuna dek tahmin edilen çaba

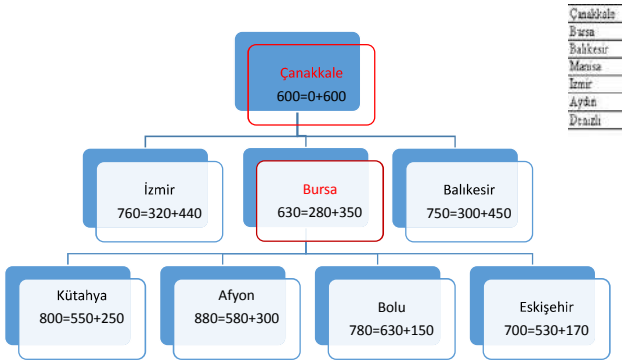
A* arama-örnek



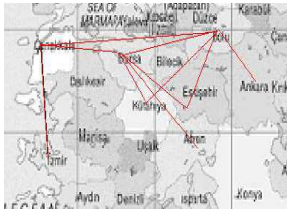
Çanakkale	600	Eskişehir	170
Bursa	350	Afyon	190
Balıkesir	450	Kütahya	220
Manisa	420	Uşak	250
İzmir	440	Isparta	240
Aydın	430	Konya	200
Denizli	410	Bolu	150

Toplam sayı: değerlendirme fonksiyonunun değeri
Sağ 1. sayı: başlangıç durumdan o anki duruma dek harcanan çaba
Sağ 2.sayı: o anki durumdan amaç durumuna dek tahmin edilen çaba

A* arama-örnek

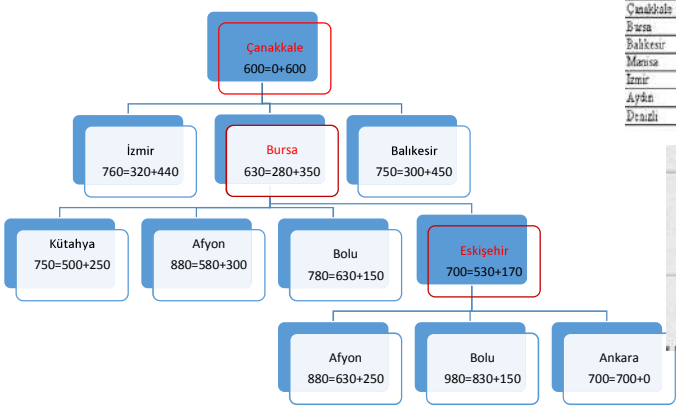


Çanakkale	600	Eskişehir	170
Bursa	350	Afyon	190
Balıkesir	450	Kütahya	220
Manisa	420	Uşak	250
İzmir	440	Isparta	240
Aydın	430	Konya	200
Denizli	410	Bolu	150



Toplam sayı: değerlendirme fonksiyonunun değeri
Sağ 1. sayı: başlangıç durumdan o anki duruma dek harcanan çaba
Sağ 2.sayı: o anki durumdan amaç durumuna dek tahmin edilen çaba

A* arama-örnek



Çanakkale	600	Eskişehir	170
Bursa	350	Afyon	190
Balıkesir	450	Kütahya	220
Manisa	420	Uşak	250
İzmir	440	Isparta	240
Aydın	430	Konya	200
Denizli	410	Bolu	150

Toplam sayı: değerlendirme fonksiyonunun değeri
Sağ 1. sayı: başlangıç durumdan o anki duruma dek harcanan çaba
Sağ 2.sayı: o anki durumdan amaç durumuna dek tahmin edilen çaba

A* algoritmasının özellikleri

- Tam? Evet
- Zaman? $O(b^m)$
- Uzay? $O(b^m)$ – tüm düğümler bellekte tutuluyor
- Optimal? Evet

Yerel arama algoritmaları

- Pek çok optimalleştirme probleminde amaca götüren yolun hiçbir önemi olmayabilir; yani çözüm amaç durumun bizzat kendisidir.
- Bu durumlarda, yerel arama algoritmalarını kullanabiliriz
  - Yalnız "şimdiki" durumu akılda tutmalı ve onu iyileştirmeye çalışmalı

Dağa Tırmanma(Hill-Climbing)

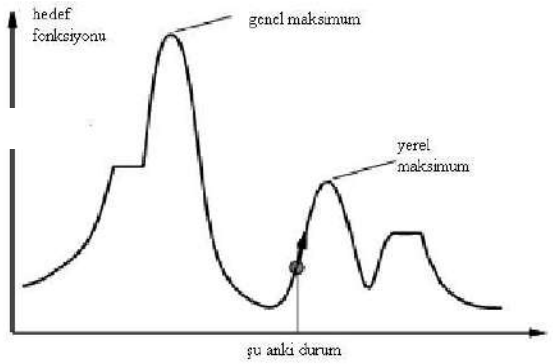
- **"Dağa tırmanma" yönteminin ana fikrinde şöyle bir varsayım dayanmaktadır:**
  - Ormancı gece dağda yolunu kaybetmiştir. Onun evi dağın zirvesindedir.
  - Karanlık olsa da ormancı, her adımının onu amacına yakınlaştırdığını bilmektedir
  - Dağa tırmanma yönteminde aramanın yönü her zaman amaca daha yakın düğüme doğrudur.

Dağa tırmanma algoritması

```
function HILL-CLIMBING( problem) returns a state that is a local maximum
   inputs: problem, a problem
   local variables: current, a node
                    neighbor, a node
   current ← MAKE-NODE(INITIAL-STATE[problem])
   loop do
      neighbor ← a highest-valued successor of current
      if VALUE[neighbor] ≤ VALUE[current] then return STATE[current]
      current ← neighbor
```

Dağa tırmanma

- Sorun: arama başlangıç durumuna bağlıdır; yerel maksimumda takılma olabilir



Dağa Tırmanma

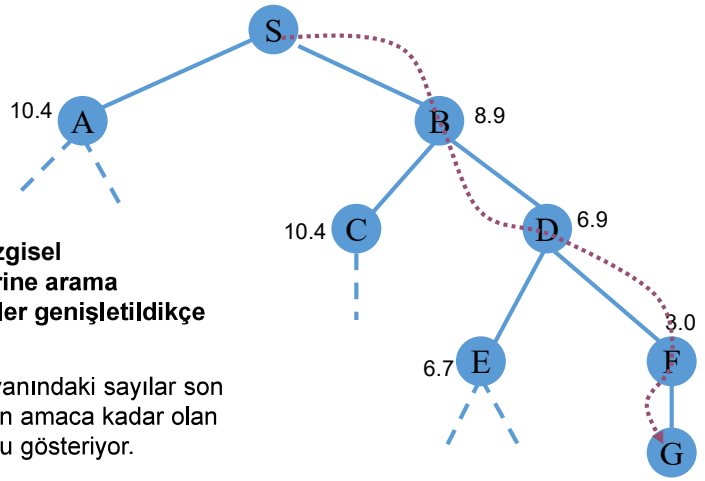
- **Sezgisel değerlendirme, derinine arama yöntemini dağa tırmanma yöntemine dönüştürür**
  - son durum ve amaç durum yolları arasındaki "mesafeyi" ölçmek için her bir yeni yola değerlendirme fonksiyonu uygulanıyor

Zaman karmaşıklığı	Uzay karmaşıklığı	tam?	Optimal?
$O(b^d)$	O(bd)	değil	değil

Dağa Tırmanma

- **Önemli noktalar**
  - Arama uzayında döngüler varsa dağa tırmanma yöntemi kullanılamaz
  - Mevcut durumdan daha iyi ardıl (komşu) durumlar yoksa arama sonlandırılıyor. Bu, arama uzayında yerel maksimum bulunması sorununu doğuruyor. Başka bir deyişle, etrafındaki durumlardan daha iyi olan durum , çözüm olmayabilir.
  - Dağa tırmanma yöntemi, çözümün tahmin edilen değerini doğru hesaplayan değerlendirme fonksiyonunun verilmiş olduğu problemler için uygulanmaktadır.

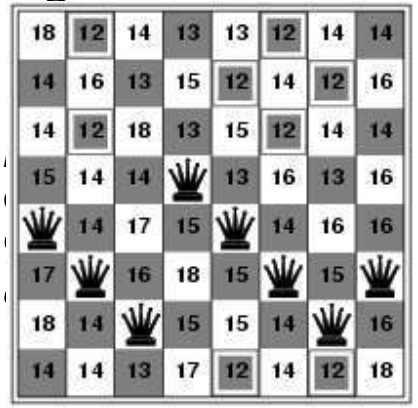
Dağa Tırmanma

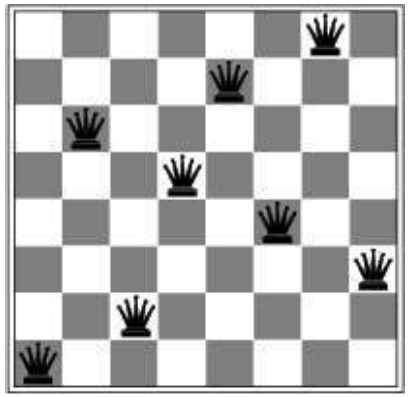


Dağa tırmanma, sezgisel değerlendirmeli derine arama yöntemidir. Düğümler genişletildikçe seçenek sunuyor.

Şekilde düğümlerin yanındaki sayılar son (o anki) düğümlerden amaca kadar olan düz yolun uzunluğunu gösteriyor.

Dağa tırmanma – 8 vezir sorunu





h = doğrudan veya dolaylı olarak bir-birine hamle eden vezir çiftleri sayısı
örnekte h = 17

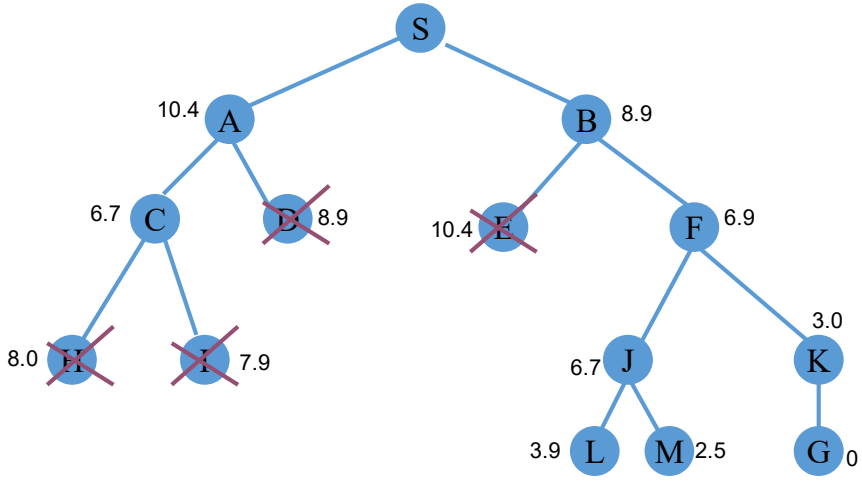
Yerel minimum: *h = 1*

Işın arama (Beam Search)

- Enine aramaya benzer. Fakat her seviyede genişletilen düğümler sayısına kısıtlama getirmekle zaman karmaşıklığı azaltılıyor.
- Enine genişletme sınırı w (b >> w) belirlenir
- Bir sonraki genişletmeler için en iyi w düğüm seçiliyor

Işın Arama

- **Arama seviye-seviye gerçekleştiriliyor. Fakat her seviyede en iyi w (enine genişletme sınırı) sayıda düğüm genişletilebilir**
- **Örnekte** `w=2`.



Genetik algoritmalar

- Rasgele üretilmiş k sayıda durumla (popülasyon) başlamalı
- Ardıl durum iki baba durumu birleştirmekle üretiliyor
- Durum, sınırlı alfabe ile (genelde 0 ve 1) oluşturulmuş satırla ifade edilir
- Değerlendirme fonksiyonu (uygunluk fonksiyonu). Daha iyi durumlar için daha yüksek değerler verir.
- Durumların bir sonraki nesillerini üretme, çaprazlama ve mutasyon kullanmakla yapılıyor

Genetik algoritmalar-örnek

- Uygunluk fonksiyonu: bir-birine hamle etmeyen vezir çiftleri sayısı
- (min = 0, max = (8 × 7)/2 = 28)