T.C
SAKARYA ÜNİVERSİTESİ
BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BSM 310 – YAPAY ZEKA

TEPE TIRMANMA ALGORİTMASI

HILL CLIMBING ALGORITHM

Grup üyeleri:

Zeynep İpek TAYYAR B171210015

Özüm BOZDAĞ B171210039

Rabia ÖZÇELİK B171210021

Ferhat ACAR B161210051

Miray ÇOBAN B171210035

Sakarya

2020

İÇİNDEKİLER

YEREL ARAMA ALGORİTMALARI

Eğer sonuca giden yol önemli değilse bu tür algoritmalar kullanılır.

Yerel arama algoritmaları bir tane şimdiki düğüm ile çalışır ve genel olarak sadece o düğümün komşularına uğrar.

Yerel arama algoritmaları sistematik olmasa da bize sunduğu iki avantaj vardır:

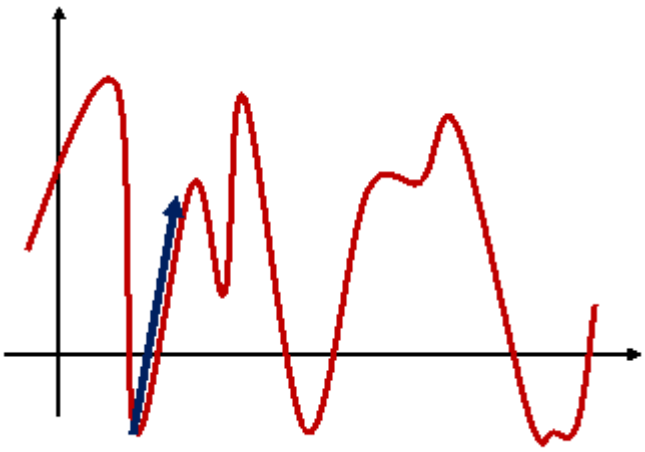
1. Bellekte çok az yer kaplarlar (genelde sabit bir değer) ve
2. Büyük ya da sonsuz durum uzaylarında genelde uygun çözümlere ulaşırlar.

Yerel arama algoritmaları hedefi bulma dışında optimizasyon problemlerinde de kullanılır. Verilen amaç fonksiyonu doğrultusunda en iyi durumu bulmaya çalışır.

- Optimizasyon problemlerinde hedefe giden yol önemsizdir ve hedefin kendisi çözümdür.
- Bazı optimizasyon problemlerinde hedef bilinmez ve amaç en iyi durumu bulmaktır.

TEPE TIRMANMA ALGORİTMASI

Bilgisayar bilimlerinde kullanılan arama algoritmalarından birisidir. Tepe tırmanma algoritması yerel algoritmalardandır. Arama işleminin yapıldığı grafikteki tepelerden ismini alır. Basitçe bir grafikte bulunan en düşük noktanın aranması sırasında grafikte yapılan hareketin aslında tepe tırmanmaya benzemesinden ismini almaktadır.



Örneğin yukarıdaki şekilde gösterilen ok temsili bir tepe tırmanma işlemidir. Burada arama yapan algoritma aslında bir çukur bulmuş ancak daha iyisi için tepe tırmanmaktadır denilebilir.

TEPE TIRMANMA ALGORİTMASI (En dik çıkma/inme)

Tepe tırmanma algoritması her iterasyonda amaç fonksiyonuna göre bulunan düğümün en başarılı çocuk düğümüne gider.

- En başarılı çocuk düğüm amaç fonksiyonuna göre en iyi değere (en yüksek ya da en düşük) sahip olan düğümdür.

- Eğer çocuk düğümlerin hiçbiri bulunulan düğümden iyi değere sahip değilse bulunulan düğümü döndürür.
- Tepeye tırmanır gibi ilerler.
- Zirveye ulaştığında, yani komşu düğümlerden hiçbiri daha yüksek değere sahip olmadığında durur.
- Eğer birden fazla ardıl durum varsa aralarından en iyisini rastgele seçer.
- Nerede olduğunu bilmediği için(hafıza kullanımı) graflarda olduğu gibi önceki duruma dönme olasılığı yoktur.

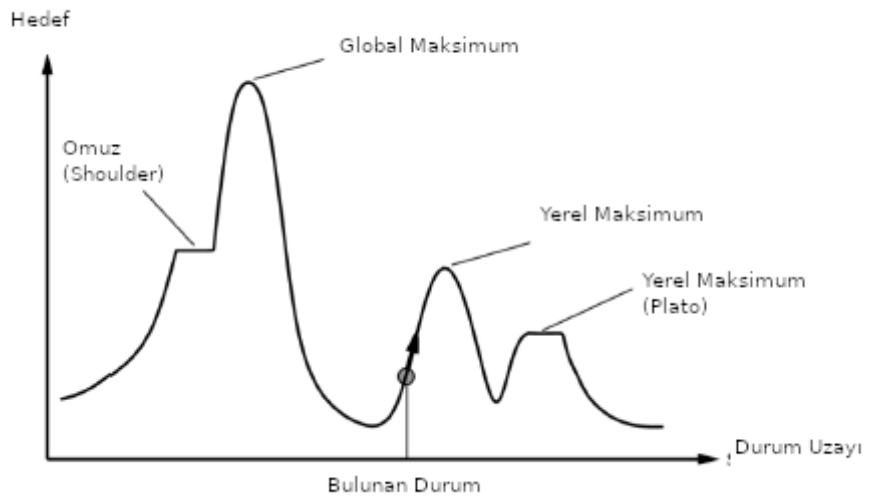
Algoritma bir arama ağacı kullanmadığından sadece bulunulan düğümü ve amaç fonksiyonunun değerini tutar.

Tepe tırmanma en yakın komşularından başka düğümlere bakmaz.

Tepe tırmanma algoritması bazen açgözlü yerel arama (greedy local search) olarak adlandırılır çünkü şuanki en iyi durumu seçer ileriyi düşünmez.

- Açgözlü algoritmalar genelde iyi performans gösterirler ve
- Tepe tırmanma algoritması genelde sonuca doğru hızlı bir gelişme gösterir.

TEPE TIRMANIŞI İÇİN DURUM UZAYI DİYAGRAMI



Durum uzayı diyagramı, arama algoritmamızın objektif fonksiyonumuzun (en üst düzeye çıkarmak istediğimiz fonksiyon) değerine ulaşabileceği durum kümesinin grafiksel bir temsilidir.

X ekseni: durum alanını, yani algoritmamızın erişebileceği durumları veya yapılandırmayı belirtir.

Y ekseni: belirli bir duruma karşılık gelen nesnel işlevin değerlerini belirtir. En iyi çözüm, objektif fonksiyonun maksimum değere (küresel maksimum) sahip olduğu durum alanı olacaktır.

Yerel maksimum: Komşu durumundan daha iyi bir durumdur, ancak bundan daha iyi bir durum vardır (küresel maksimum). Bu durum daha iyidir çünkü burada nesnel işlevin değeri komşularından daha yüksektir.

Küresel maksimum: Durum uzayı diyagramında mümkün olan en iyi durumdur. Çünkü bu durumda objektif fonksiyon en yüksek değere sahiptir.

Plato / düz yerel maksimum: Komşularının aynı değere sahip olduğu düz bir bölgedir.

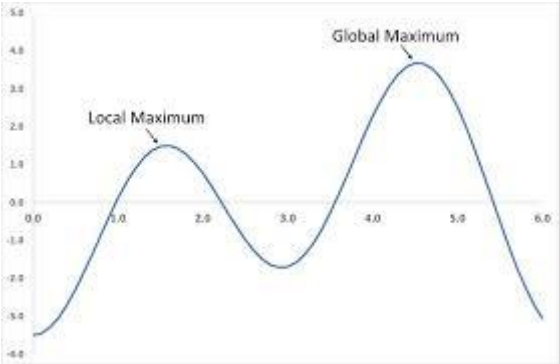
Mevcut durum: Arama sırasında şu anda bulunduğumuz durum uzay diyagramı bölgesi.

Omuz (Shoulder): Yokuş yukarı kenarı olan bir platodur.

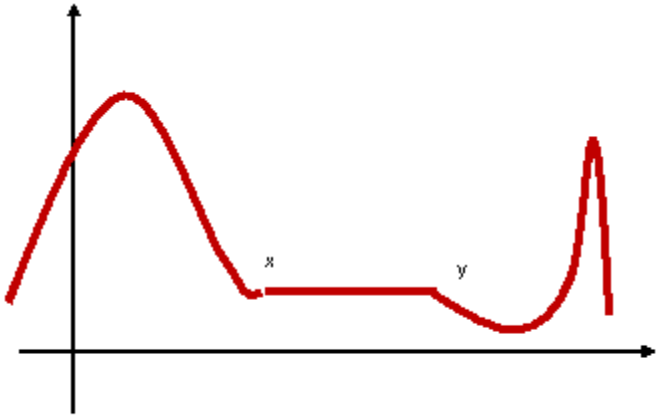
TEPE TIRMANIŞINDA FARKLI BÖLGELERDEKİ SORUNLAR

Aşağıdaki bölgelerden herhangi birine girerse, tepe tırmanışı en iyi duruma (küresel maksimum) ulaşamaz:

- **Yerel maksimum**: Yerel maksimumda tüm komşu durumların geçerli durumdan daha kötü bir değeri vardır. Tepe tırmanışı açgözlü bir yaklaşım kullandığından, daha kötü duruma geçmeyecek ve kendini sonlandıramayacaktır. Daha iyi bir çözüm bulunabilse de süreç sona erecek.



- **Plato (Yayla):** Platoda tüm komşular aynı değere sahiptir. Bu nedenle, en iyi yönü seçmek mümkün değildir.



Örneğin yukarıdaki şekilde x ve y noktaları arasında bir düzlük bulunmaktadır. Başlangıç noktası olarak bu aralıktaki herhangi bir noktadan başlanırsa algoritma komşuları aradığında daha iyi veya daha kötü bir sonuç bulamayacağı için hatalı karar verebilir.

- **Sırt (Ridge):** Sırttaki herhangi bir nokta tepe gibi görünebilir, çünkü olası tüm yönlerde hareket aşağı doğrudur. Dolayısıyla algoritma bu duruma ulaştığında durur.

TEPE TIRMANMA ALGORİTMASI PSEUDO KODU

```
Ayrık Uzay Tepe Tırmanma Algoritması:
    currentNode := startNode
    loop do
        L := NEIGHBORS(currentNode)
        nextEval := -INF
        nextNode := NULL
        for all x in L do
            if EVAL(x) > nextEval then
                nextNode := x
                nextEval := EVAL(x)
        if nextEval ≤ EVAL(currentNode) then
            // Return current node since no better neighbors exist
            return currentNode
        currentNode := nextNode
```

```
Kesintisiz Uzay Tepe Tırmanma Algoritması
    currentPoint := initialPoint    // the zero-magnitude vector is
common
    stepSize := initialStepSizes    // a vector of all 1's is common
    acceleration := someAcceleration // a value such as 1.2 is
common
    candidate[0] := -acceleration
    candidate[1] := -1 / acceleration
    candidate[2] := 0
    candidate[3] .= 1 / acceleration
    candidate[4] := acceleration
```

```
    loop do
        before := EVAL(currentPoint)
        for each element i in currentPoint do
            best := -1
            bestScore := -INF
            for j from 0 to 4 do      // try each of 5 candidate
locations
                currentPoint[i] := currentPoint[i] + stepSize[i] ×
candidate[j]
                temp := EVAL(currentPoint)
                currentPoint[i] := currentPoint[i] − stepSize[i] ×
candidate[j]
                if temp > bestScore then
                   bestScore := temp
                   best := j
            if candidate[best] is 0 then
                stepSize[i] := stepSize[i] / acceleration
            else
                currentPoint[i] := currentPoint[i] + stepSize[i] ×
candidate[best]
                stepSize[i] := stepSize[i] × candidate[best] //
accelerate
        if (EVAL(currentPoint) − before) < epsilon then
            return currentPoint
```

TEPE TIRMANMA ALGORİTMASI ÇEŞİTLERİ

1. **Steepest Ascent Hill Climbing (En Dik Tepe Tırmanma)**
   Klasik tepe tırmanma algoritmasından farklı olarak bu algoritmada, bulunabilen bütün sonuçlar arasından bir seçim yapılır.
   Bu algoritmada da klasik tepe tırmanma algoritmasında da sorun aynıdır. Şayet arama işlemi sırasında bir yerel çukura (local minimum) rastlanılırsa bu durumdan algoritma kendisini kurtaramayarak en doğru sonucu bulamayabilir.
2. **Stochastic Hill Climbing (Olasılıksal Tepe Tırmanma)**

Olasılıksal tepe tırmanma algoritmasında (stochastic hill climbing algorithm) bütün komşuların aranması ve komşuların verdiği sonuca göre hareket etmek yerine, rast gele olarak bir komşunun seçilmesi söz konusudur. Şayet gidilen bu komşu beklenen yönde bir iyileştirme sağlıyorsa, bu yönde aramaya (tırmanmaya) devam edilir, beklenen iyileştirme sağlanamıyorsa, bu durumda daha farklı bir komşu denenir.

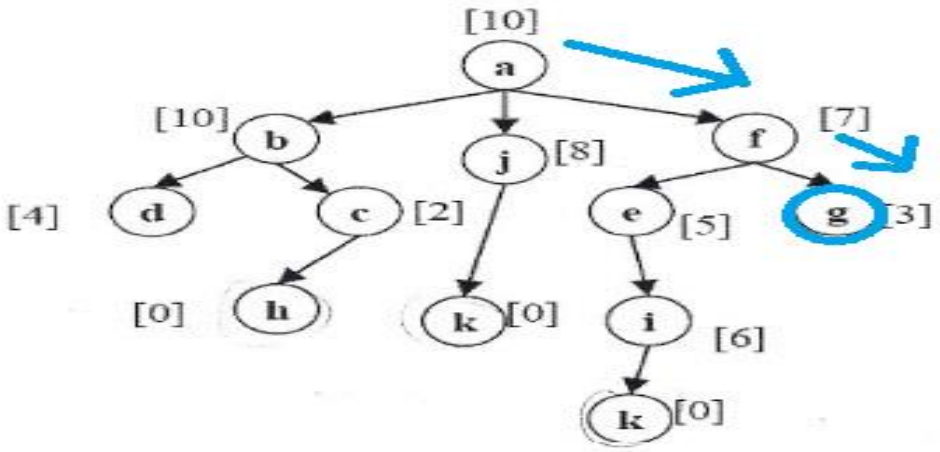
3. **Random Restart Hill Climbing (Rastgele Başlangıç Tepe Tırmanma)**

Yukarıdaki tepe tırmanma algoritmalarının yanında rastgele başlangıç tepe tırmanma algoritması (random restart hill climbing algorithm) şaşırtıcı derecede iyi sonuç veren bir algoritmadır. Bu algoritma basitçe bir x durumunu başlangıç kabul eder ve daha iyi bir durum bulunca başlangıç durumunu bu daha iyi duruma kaydırır. Algoritma iyi durum buldukça başlangıç durumunu kaydıran ancak bulamadığı durumlarda da aramaya devam eden bir

yapıya sahiptir. Rastgele başlangıç tepe tırmanma algoritmasına bazı kaynaklarda pompalı tüfek tepe tırmanma algoritması (shotgun hill climbing algorithm) ismi de verilmektedir.

TEPE TIRMANMA ALGORİTMASI ÖRNEKLERİ

1.Örnek



Örneğin bu ağaçta en az maliyetle sonuca ulaşmaya çalıştığımızda ilk durumda a düğümüne uğruyoruz. Düğümlerde ki değerler sezgisel sonuçlar olarak adlandırılıyor.

H ve k düğümlerinin sezgisel maliyetleri 0 verildiği için onları son nokta, sonuç durumu olarak kabul edebiliriz.

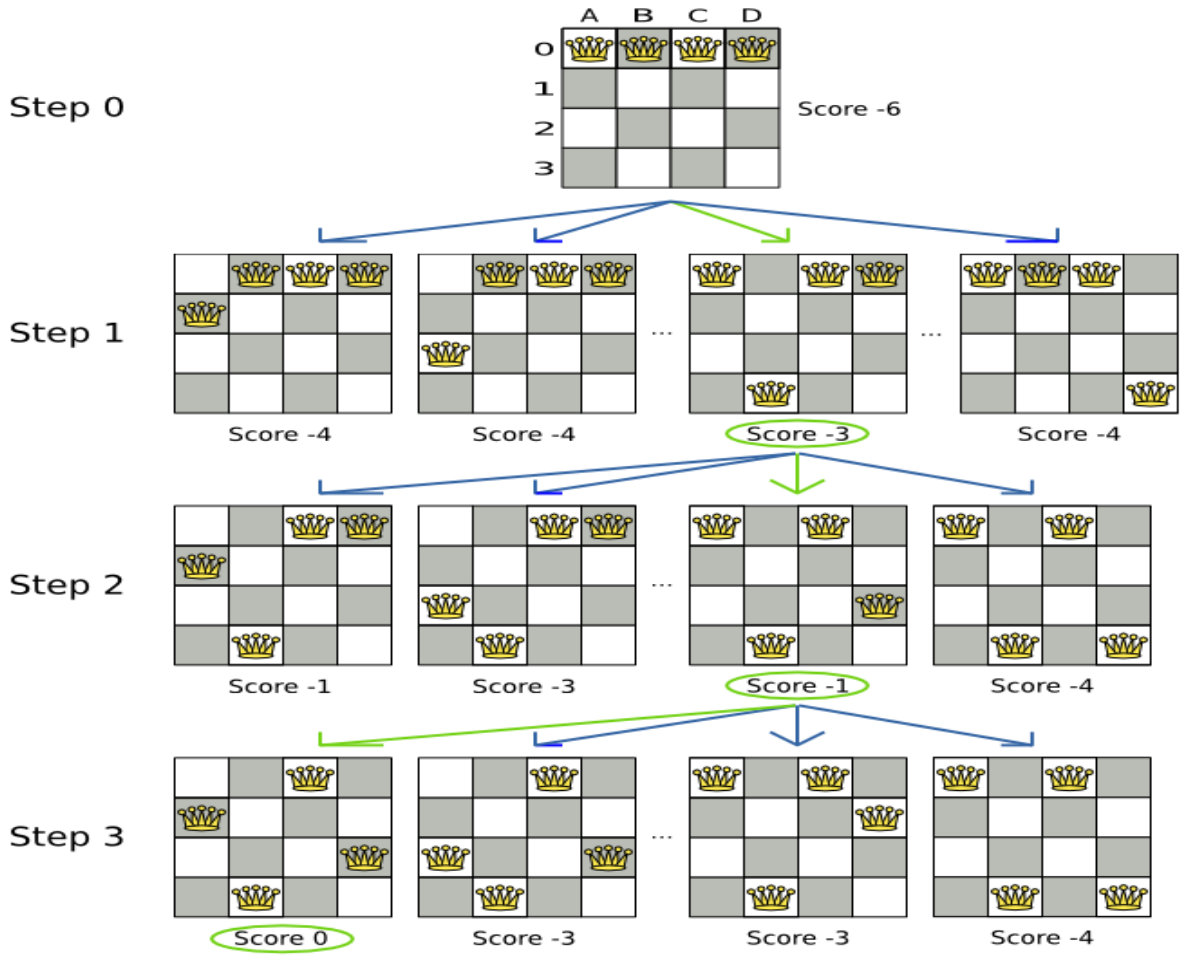
Burada a düğümünden başlayarak tepe tırmanması yapmayı denediğimizde iki alternatifimiz var b ya da f düğümü.

B de sezgisel maliyet 10, f de ki sezgisel maliyet 7'dir. Az olanı tercih edeceğimiz için f düğümüne gidiyoruz.

F düğümünden sonra algoritma aç gözlü bir yaklaşım yaparak g düğümüne ilerleyecektir. Fakat g bizim bir sonuç durumumuz değildi yani geldiğimiz noktada sonuca ulaşamamış oluyoruz. Tepe Tırmanma Algoritmasında böyle bir sorun vardır. Her zaman sonuca ulaşamayabiliriz. Bu örnekte sonuç durumuna ulaşamama sebebimiz algoritmanın local minimuma takılıyor olmasıdır.

Tepe tırmanma algoritmasında local minimum ve local maximumumu çözmek için geliştirmeler vardır.

2.ÖRNEK: N VEZİR PROBLEMİ (N QUEENS PROBLEM)



KAYNAKÇA

- https://web.cs.hacettepe.edu.tr/~ilyas/Courses/VBM688/lec05_localsearch.pdf
- https://www.seas.upenn.edu/~cis391/Lectures/informed-search-II.pdf
- http://bilgisayarkavramlari.sadievrenseker.com/2009/12/02/tepe-tirmanma-algoritmasi-hill-climbing-algorithm/
- https://www.geeksforgeeks.org/introduction-hill-climbing-artificial-intelligence/
- https://docs.optaplanner.org/6.3.0.Beta1/optaplanner-docs/html/ch10.html
- https://www.javatpoint.com/hill-climbing-algorithm-in-ai
- https://en.wikipedia.org/wiki/Hill_climbing