T.C

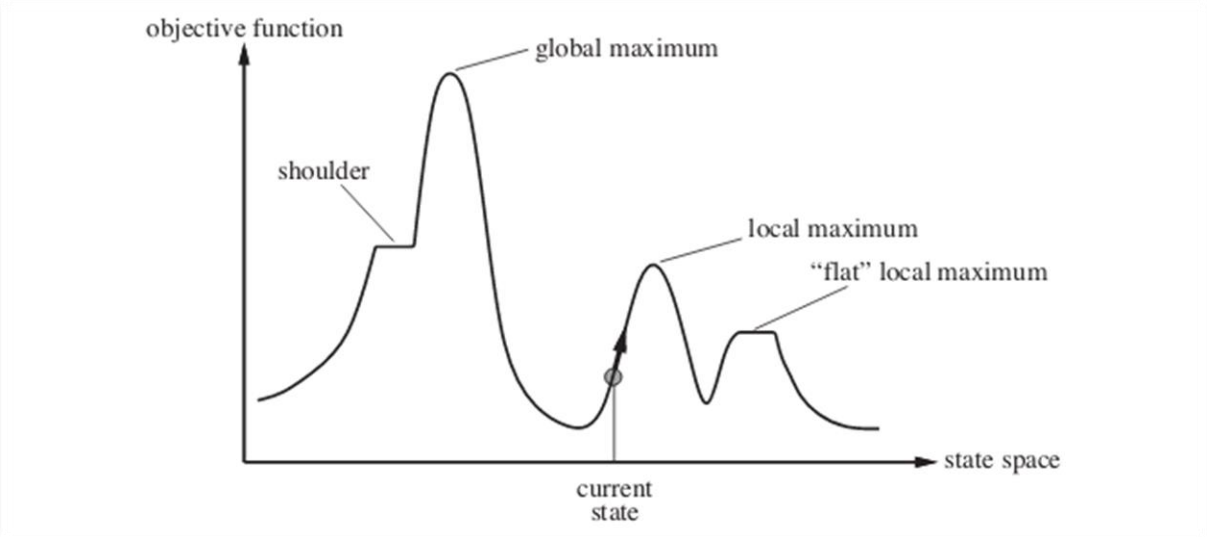
SAKARYA ÜNİVERSİTESİ
BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BSM 310 – YAPAY ZEKA

Grup üyeleri:
G1712.10010 AHMET KAYACI
G1712.10106 BERKECAN AYHAN
G1712.10064 BURAK KALKAVAN
G1812.10376 İRFAN AŞKIN KILIÇ
B1712.10044 ÖMER FARUK AKKAYA

Sakarya
2020

TEPE TIRMANMA ALGORİTMASI



Basitçe açıklamak gerekirse **Tepe Tırmanma Algoritması**, sürekli artan değerin olduğu yere doğru hareket eden bir döngüdür. Algoritma, hafızasında bir arama ağacı tutmaz, bu durumda temel düğüm (node) veri yapısı sadece o anki ve bir sonraki gidilebilecek durumları içerir. Tepe Tırmanma Algoritması'nın 3 varyasyonu vardır. Bunlardan ilki Temel Tepe Tırmanma, ikincisi En-Dik Çıkış Tepe Tırmanma ve son olarak Rastlantısal Tepe Tırmanma olmak üzere 3 ana başlık altında incelenir. Gidilebilecek daha iyi birden fazla durum varsa algoritma aralarından rastgele birine gider. Bu temel yaklaşımın üç adet dezavantajı vardır.

Tepe Tırmanışı için Durum-Uzay Diyagramı:

Durum-uzay manzarası, çeşitli algoritma durumları ve Amaç fonksiyonu / Maliyet arasında bir grafik gösteren tepe tırmanma algoritmasının grafiksel bir temsilidir.

Y ekseninde objektif bir fonksiyon veya maliyet fonksiyonu olabilen fonksiyonu ve X eksenindeki durum uzayını aldık. Y ekseni üzerindeki fonksiyonun maliyeti varsa, aramanın amacı küresel minimum ve yerel minimumları bulmaktır. Y ekseni işlevi Amaç fonksiyonu ise, aramanın amacı küresel maksimum ve yerel maksimum değerlerini bulmaktır.
(Üstteki grafik incelenmelidir)

Tepe Tırmanma Algoritması'nın Özellikleri

Tepe Tırmanma Algoritmasının bazı temel özellikleri şunlardır:

1) **Üretme ve Test değişkeni**: Tepe Tırmanışı, Üretme ve Test yönteminin bir çeşididir. Üret ve Test Et yöntemi, arama alanında hangi yöne hareket edeceğinize karar vermeye yardımcı olan bir geri bildirim üretmektedir.

2) **Açgözlü yaklaşım:** Tepe tırmanma algoritması araması, maliyeti optimize eden yönde hareket etmektedir.

3) **Geri izleme yok:** Önceki durumları hatırlamadığı için arama alanını geri izlemez ve hareketine devam eder.

TEPE TIRMANMA ALGORİTMASININ VARYASYONLARI

1)Temel Tepe Tırmanma

Temel Tepe Tırmanma'da amaç, başlangıç noktası olarak belirlenen bir noktadan komşu noktalara gezerek daha iyi sonuçlar elde etmektir. Daha hızlı sonuç alır ancak daha düşük bir optimal çözüm verir ve garanti değildir.

Eğer belirlediğimiz başlangıç noktasının bir tarafında problem iyileşirken diğer tarafında kötüleşiyorsa, algoritma iyi olan tarafa doğru hareket eder.

Eğer belirlediğimiz başlangıç noktasının iki tarafında da problem iyileşiyorsa, algoritma taraflardan birini seçer. (Hangisini seçtiği problem çözücünün kararına bağlıdır.) Görüldüğü gibi her iki tarafa hareketinde de sonuç bizim için olumlu yönde optimuma doğru hareket eder.

Eğer belirlediğimiz başlangıç noktasının iki tarafında da problem kötüleşiyorsa, algoritma bu noktada kalır ve çalışmasını tamamlar. Bu artık programın belirli bir maksimuma ulaştığı anlamına gelir. (Global veya Lokal maksimum)

Algoritması:

Adım 1: Başlangıç durumunu değerlendirin, eğer hedef durumda ise, başarılı döndürün ve durun.

Adım 2: Döngüye bir çözüm buluncaya kadar veya uygulanacak yeni bir operatör kalmayıncaya kadar devam edin.

Adım 3: Bir operatör seçin ve geçerli duruma uygulayın.

Adım 4: Yeni durumu kontrol edin:
Hedef durum da ise, başarılı döndürün ve çıkın.
Eğer mevcut durumdan daha iyiyse, yeni bir durumu mevcut durum olarak atayın.
Mevcut durumdan daha iyi değilse, 2. adıma dönün.

Adım 5: Çıkış.

2)En Dik-Çıkış Tepe Tırmanma

Temel Tepe Tırmanma'nın aksine seçilebilecek daha iyi olan ilk noktayı değil, daha iyi olan noktalar arasındaki en iyi noktayı bulmaya çalışır. Bulunduğu noktadan gidilebilecek tüm noktaları değerlendirmeye alır. Sonraki nokta olarak en iyi noktayı seçer. Algoritmanın bu türü daha aceleci ve hızlı olarak bi sonuca varmaktansa biraz daha seçici davranır ve bizi daha doğru bir sonuca götürme ihtimali fazladır.

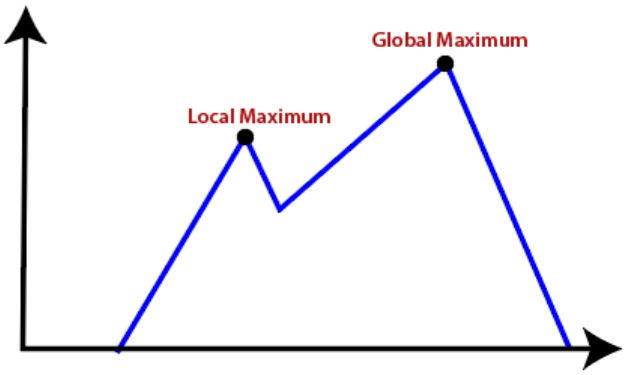
3)Rastlantısal Tepe Tırmanma

Başlangıç noktasından iyiye doğru olan taraf seçilir. Şayet bu nokta beklenildiği gibi bir iyileştirme sağlıyorsa yola devam edilir. İyileştirme sağlanamadıysa rastgele yeni bir başlangıç noktası seçilir. Algoritmanın bu türü deneme-yanılma payını arttırmaktadır, bize ulaşmamız gereken herhangi bir maksimum değer için çok daha fazla opsiyon sağlayabilmektedir.

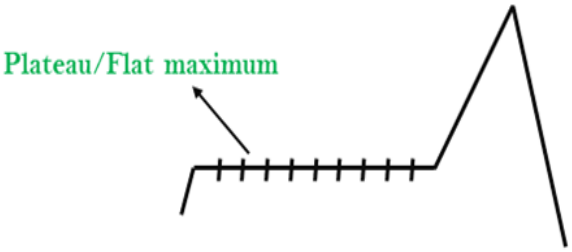
Özetle; 3 farklı türde incelediğimiz Tepe Tırmanma algoritmamız genel anlamda hızlı ve doğruluğu her zaman kesin olmayacak bir çözüme bize götürebilecek olsada, farklı türlerinde daha isabetli sonuçlar için farklı yollar izlenildiği de görülebilmektedir. Böylelikle bizi sonuç olarak daha iyi bir maksimum sonuca ulaştırmaya çalışır.

TEPE TIRMANMA ALGORİTMASI SORUNLARI

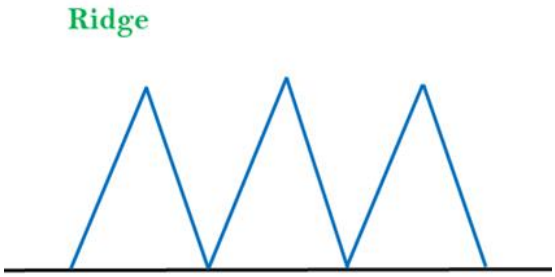
1.Yerel Maksimum (Local Max):Yerel Maksimum Global Maksimuma benzer. En büyük tepe noktası (yani global maksimum) olmaması durumunda algoritma başarım sağladığını düşünerek çalışmayı durdurur.



2. Plato (Plateau): Plato, durum uzayında çalışan fonksiyonun düz (eğimi=0) olduğu bölgedir. Algoritma buraya girdiği vakit rastgele bir yürüyüşe çıkacaktır.



3. Sırt (Ridge): Bir sırt aniden dikleşebilir, bu durumda algoritma sırtın tepesine kolaylıkla ulaşabilir, ama tepe sadece çok hafif kıvrılarak da hakiki bir tepe noktası gibi görünebilir. Bu duruma özel bir ekstra işlem algoritmaya dahil edilmemişse (sürekli aynı yöne gitmek gibi) algoritma bir o tarafa bir bu tarafa verimsiz bir şekilde hareket eder.



Tüm bu durumlarda algoritma, daha fazla ilerlemenin kat edilmediği bir noktaya ulaşır. Böyle bir şey olursa yapılabilecek en mantıklı şeylerden biri farklı bir başlangıç noktasından yeniden başlamak olabilir. Rastgele Tepe Tırmanma Algoritması bu iş için çok uygundur. Bir dizi Tepe Tırmanma Algoritmasını (her biri rastgele, farklı başlangıç noktalarından) çalıştırır. Tüm bu algoritma dizisi arasından en iyi sonuca sahip algoritmayı seçer.
Sabit sayıda iterasyonla bu işlemi gerçekleştirebilir veya sürekli çalışarak, belli bir sayıdan sonra en iyi sonucun değişmemesi üzerine bir yaklaşım da geliştirilebilir.

Sonuç olarak, yeterli sayıda iterasyona izin verilirse, Rastgele Tepe Tırmanma Algoritması eninde sonunda en iyi sonuca ulaşacaktır. Tepe Tırmanma Algoritmasının başarısı çok büyük oranda durum uzayının şekline (yani araziye) bağlıdır. Eğer sadece birkaç tane Yerel Maksimum varsa, Rastgele Tepe Tırmanma iyi bir sonucu hızlı bir şekilde bulacaktır.

Gerçek hayat problemlerinde arazi çok daha düzensiz ve dağınıktır. Eğer problem NP-Complete (karmaşıklık düzeyi arttıkça üstel olarak zorlaşan problemler) bir problem ise, üstel bir çalışma zamanından daha iyisini beklemek doğru olmayacaktır. Neyse ki çoğu zaman az sayıda iterasyonla yeterince iyi bir çözüme ulaşılır.

Pseudo-Code Örneği

Şimdi de Tepe Tırmanma Algoritmasının basit bir **Pseudo-Code** örneğini inceleyelim.

```
function HILL-CLIMBING(problem) returns a solution state
   inputs: problem, a problem
   static: current, a node
           next, a node

   current ← MAKE-NODE(INITIAL-STATE[problem])
   loop do
      next ← a highest-valued successor of current
      if VALUE[next] < VALUE[current] then return current
      current ← next
   end
```


```
function HILL-CLIMBING(problem) returns a solution state
   inputs: problem, a problem
   static: current, a node
           next, a node

   current ← MAKE-NODE(INITIAL-STATE[problem])
```

Fonksiyon problemi parametre olarak alıyor, yapılan işlemler sonucu çözüm durumunu döndürüyor. Daha önce de bahsedilen temel düğüm veri yapısından iki elemanı – o an bulunduğu düğüm ve bir sonraki düğüm – kullanıyor.

Fonksiyon, problemin başlangıç durumundan bir düğüm üreterek ve ürettiği düğümü o an bulunduğu düğüm değişkenine atayarak çalışmaya başlıyor. Bir sonraki adımda, Tepe Tırmanma Algoritmasının davranışını belli eden bir döngü var. Şimdi bu döngüyü detaylıca inceleyelim.

```
loop do
    next ← a highest-valued successor of current
    if VALUE[next] < VALUE[current] then return current
    current ← next
end
```

Döngüye bir giriş koşulu yok gibi gözüküyor, ama bir tane çıkış koşulu var. Bu döngünün en az bir kere çalışacağı anlamına geliyor. Döngü bir sonraki düğüm değişkenine o an bulunduğu düğüm değişkeninin erişebildiği en yüksek değerli düğüm değişkenini atıyor.

Bunu takip eden adımda çıkış koşulu var. Algoritmanın daha önce de bahsettiğimiz zayıflıklarının temel sebebi bu koşul. Koşul, bir sonraki düğüm değişkenine atadığımız düğüm ile o an bulunduğu düğüm değişkenine atadığımız düğüm değerlerini karşılaştırıyor. Bu durumda o an bulunduğu düğüm, bir sonraki düğümün değerinden büyükse bir tepeye varmışız demektir. Bir önceki adımda halihazırda olabilecek en yüksek değerli bir sonraki düğüm değişkenini belirlemiştik. En yüksek değeri bulduğumuz halde o an bulunduğumuz düğümün değeri daha büyükse bir tepe noktasında bulunduğumuzu anlamamız gerekir. Ve bir tepe noktasına vardıysak algoritma işini bitirmiş demektir, yani bizim çıkış noktamız bu adım olmalıdır.

Bir sonraki adımda, koşulun true döndürmemesi durumunda, daha önce seçmiş olduğumuz bir sonraki düğüm değişkenini yeni o an bulunduğumuz düğüm değişkeni yaparız. Döngü bu işlemleri tekrar ede ede bir tepe noktası bulana kadar dönmeye devam eder.

TEPE TIRMANMA ALGORİTMALARININ KULLANIM ALANLARI

Uygun araziye sahip bir sezgisel fonksiyon kullanılırsa Tepe Tırmanma Algoritmaları birçok alanda kullanılabilir. Bunları maddesel olarak incelersek;

1) **Ağ Trafiği:** Ağ trafiği, belirli bir zamanda bir ağda hareket eden veri miktarını ifade eder. Ağ verileri çoğunlukla ağdaki yükü sağlayan ağ paketlerinde kapsüllenir. Ağ trafiği, ağ trafiği ölçümü, ağ trafiği kontrolü ve simülasyon için ana bileşendir

2) **Gezgin Satıcı Problemleri:** Seyahat Eden Satıcı Sorunu (seyahat eden satış elemanı sorunu olarak da bilinir) şu soruyu sormaktadır: "Bir şehirler listesi ve her bir şehir çifti arasındaki mesafeler göz önüne alındığında, her bir şehri ziyaret eden ve başlangıç noktasına geri dönen olası en kısa yol nedir?"

3) **8 Vezir Problemi:** 8 Vezir Problemi, 8x8'lik bir satranç tahtasına 8 adet vezirin hiçbiri olağan vezir hamleleriyle birbirini alamayacak biçimde yerleştirmesi sorunudur. Her bir vezirin konumunun diğer bir vezire saldırmasına engel olması için hiçbir vezir başka bir vezirle aynı satıra, aynı kolona ya da aynı köşegene yerleştirilemez.

4) **Entegre Devre Tasarımı:** Entegre devre tasarımı veya IC tasarımı, entegre devreleri veya IC'leri tasarlamak için gereken belirli mantık ve devre tasarım tekniklerini kapsayan bir elektronik mühendisliğinin alt kümesidir. Entegre Devre Tasarımı, taşınabilir, hızlı ve enerji tasarruflu iletişim sistemleri oluşturmak için akıllı IC tasarım tekniklerine odaklanır.

5) **Deneyerek Öğrenme Metodu:** Deneyerek öğrenme, ilk elden deneyim yoluyla bir eğitim yöntemidir. Beceri, bilgi ve deneyim, geleneksel akademik sınıf ortamının dışında edinilir ve stajları, yurtdışındaki çalışmaları, saha gezilerini, saha araştırmalarını ve hizmet-öğrenme projelerini içerebilir.

6) **Robot Koordinasyonu:** Çok robotlu bir sistemde, her robot iki temel davranış arasından seçim yapabilir: engellerden kaçınma ve çatışma çözme davranışları. Çarpışmayı önlemek için robotlar arasında koordinasyon gereklidir:
Her robot, en yakın robotla uyumlu yörünge modifikasyonu için kullanılan bir açısal veriyi iletmektedir.

gibi başlıca örnek gösterilebilecek uygulamaların yanında Tepe Tırmanma Algoritmaları farklı birçok alanda kullanılmakta ve bizlere problem üzerinde optimal bir sonuca ulaşmakta yardımcı olmaya çalışmaktadır.

KAYNAKÇA

[1] http://bilgisayarkavramlari.sadievrenseker.com/2009/12/02/tepe-tirmanma-algoritmasi-hill-climbing-algorithm/

[2] https://docplayer.biz.tr/92622168-Yerel-arama-teknikleri-ve-optimizasyon-local-search-and-optimisation.html

[3] https://www.geeksforgeeks.org/introduction-hill-climbing-artificial-intelligence/

[4] https://www.javatpoint.com/hill-climbing-algorithm-in-ai

[5] https://medium.com/@bhavek.mahyavanshi50/introduction-to-hill-climbing-artificial-intelligence-a3714ed2d8d8

[6] https://www.edureka.co/blog/hill-climbing-algorithm-ai/

[7] https://web.cs.hacettepe.edu.tr/~ilyas/Courses/VBM688/lec05_localsearch.pdf

[8] https://www.mygreatlearning.com/blog/an-introduction-to-hill-climbing-algorithm/

[9] https://www.sciencedirect.com/science/article/pii/S0888613X97000042

[10] https://study.com/academy/lesson/what-is-experiential-learning-definition-theories-examples.html

[11] https://www.utwente.nl/en/education/master/programmes/electrical-engineering/specialization/integrated-circuit-design/

[12] http://www.brainmetrix.com/8-queens/

[13] https://www.geeksforgeeks.org/travelling-salesman-problem-set-1/

[14] https://www.techopedia.com/definition/29917/network-traffic