T.C

SAKARYA ÜNİVERSİTESİ

BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BSM 310 – YAPAY ZEKA

TEPE TIRMANMA ALGORİTMASI
(HLL CLIMBING SEARCH-HCS)

Sunum Raporu

Grup üyeleri:

B191200702 – Fatemeh FARSHIDKIA
G171210026 – İrem YALÇIN
G171210058 – Ertuğrul KARABABA

DANIŞMAN

DR.ÖĞR.ÜYESİ İSMAİL ÖZTEL

Sakarya

2020

Yapay Zekada Tepe Tırmanma Algoritması
(Hill-Climbing Search – HCS)

Tepe tırmanma algoritması, dağın zirvesini veya soruna en iyi çözümü bulmak için sürekli olarak yükseklik / değer artırma yönünde hareket eden yerel bir arama algoritmasıdır. Hiçbir komşunun daha yüksek bir değere sahip olmadığı bir zirve değerine ulaştığında sona erer. [1]

Tepe Tırmanışı, Yapay Zeka alanındaki matematiksel optimizasyon problemleri için kullanılan buluşsal bir araştırmadır. Geniş bir girdi seti ve iyi bir sezgisel işlev göz önüne alındığında, soruna yeterince iyi bir çözüm bulmaya çalışır. Bu çözüm küresel optimum maksimum olmayabilir. Yukarıdaki tanımda, matematiksel optimizasyon problemleri, tepeye tırmanmanın, verilen girdilerden değerler seçerek belirli bir gerçek işlevi en üst düzeye çıkarmak veya en aza indirmemiz gereken sorunları çözdüğü anlamına gelir. [1]

Sezgisel Arama Stratejileri

(Heuristic-Informed Search Strategies)

Sezgisel arama işlemini gerçekleştirmenin en kolay yolu Tepe Tırmanma (hill climbing) olarak adlandırılan bir arama stratejisidir. Tepe tırmanma stratejisinde arama işleminde o anda aktif olan düğüm genişletilir ve çocuk düğümler değerlendirilir. Bunlardan en iyi değerle sahip olan düğüm bir sonraki genişletme işleminde kullanılmak üzere seçilir. Genişletme, ilk en iyi bulunduğunda ya da en iyi bulunduğunda olabilir. Sezgi: Daima daha iyi bur duruma hareket eder. Arama işlemi çocukların hepsinden daha iyi bir duruma erişildiğinde sona erer. Bu sistemde hatalı sezgisel değerler sonsuz yollara, dolayısıyla arama işleminin başarısızlıkla sonuçlanmasına neden olur. Bu yöntemin başarısı için sezgisel değerlerinin doğru belirlenmesi büyük önem taşır. Algoritmada önceki basamaklarla ilgili bilgi tutulmamaktadır, dolayısıyla hata durumu ile karşılaşıldığında eldeki yanlış çözüm üzerinde bir düzeltme yapılabilmesi söz konusu değildir. Algoritma tüm çocuklardan daha iyi durumda bir düğüme erişildiğinde sona erdiğinden yerel maksimum noktaları da sorun yaratmaktadır. İyi mutlak açıdan en iyi olmak zorunda değildir.

Tepe tırmanma algoritması yerel ve global maksimum arasındaki ayrımı yapamamaktadır.[2]

Bir sistemin ya da bir programın daha iyi hale getirilmesi isteniyorsa, sistemin verdiği sonuçlara göre arama işlemi yapılarak iyileştirme amaçlanabilir. İşte bu noktada tepe tırmanma algoritması (hill climbing algorithm) de dahil olmak üzere çeşitli arama algoritmaları devreye girer. Örneğin literatürde sıkça kullanılan seyyar satıcı problemi bu problemlerden birisidir. Tepe tırmanma algoritması verilen bir harita için verilen bir sonucu iyileştirmeye çalışır. Elbette amaç bütün ihtimalleri denemeden iyi bir sonuç bulabilmektir. Tepe tırmanma algoritması, arama algoritmaları arasındaki en iyi sonucu verene algoritma değildir. Ancak kodlanması ve tasarımının basit oluşundan dolayı sıklıkla kullanılır. [2]

Tepe Tırmanma teknikleri keşfedilmemiş bölgenin özellikleri üzerinde temel olarak bilgilendirilmemiş olmalarına rağmen, belirli bir düğüm hakkında yerel (local) bilgiden faydalanır.(Dr.R.Kandemir 2017).

Tepe Tırmanışı, **Genetik Algoritmanın** en basit uygulamasıdır . Uygulama kolaylığına odaklanmak yerine, tamamen nüfus ve çaprazlama gibi kavramlara biniyor. Daha geleneksel genetik algoritmalara kıyasla daha hızlı yinelemelere sahiptir, ancak karşılığında geleneksel olanlardan daha az kapsamlıdır.[10]

Tepe Tırmanma metodu bir iteratif iyileştirme (yerel arama) yöntemidir. iniş ya da adım adım iniş strateji olarak da adlandırılmaktadır. Bu metodun temelinde, tanımlanan bazı kurallara göre bir çözümden bir diğer komşu çözüme ulaşma vardır. Tepe tırmanma metodunun etkinliğinde iyi bir komşuluk yapısı seçiminin önemi büyüktür. İyi mutlak açıdan en iyi olmak zorunda değildir. Tepe tırmanmanın zayıf yanı yerel ve genel en iyi arasındaki ayrımı yapamaması sonucu yerel optimumdan kaçamamasıdır. Yerel optimumdan genel optimuma geçebilmek için modern sezgiseller geliştirilmiştir. Tepe tırmanma algoritmasının adımları aşağıdaki gibi gösterilebilir:

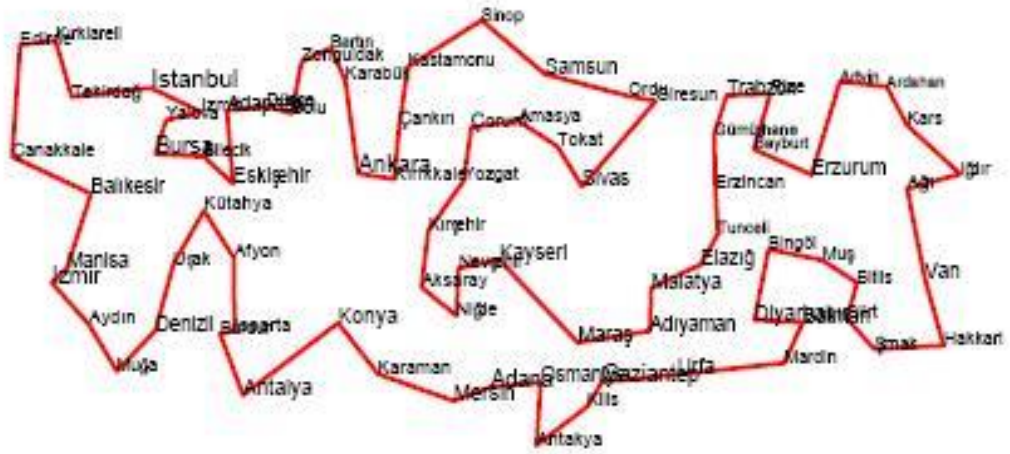
1. Başlangıç çözümü üret; x0: mevcut çözüm ve $x0 \in R$
2. Aşağıdaki adımları tekrarla:

a) N(xn) komşu seti içinden en iyi x′ komşusunu seç. b) f(x′) ≤ f(xn) ise x′ çözümünü yeni mevcut çözüm olarak ata: xn+1 = x′ c aksi halde dur. [1]

Seyahat Satıcı Problemi (Traveling Salesman Problem)

Bu problemde Satıcı tarafından kat edilen mesafeyi en aza indirmemiz gerekiyor. "Sezgisel arama", bu arama algoritmasının soruna en uygun çözümü bulamayabileceği anlamına gelir. Ancak, makul bir sürede iyi bir çözüm verecektir. Sezgisel işlev, arama algoritmasındaki herhangi bir dallanma adımında mevcut tüm bilgilere dayanarak olası tüm alternatifleri sıralayacak bir işlevdir. Algoritmanın olası rotalardan en iyi rotayı seçmesine yardımcı olur.[1]

Pekçok farklı yer gezen bir tüccarın en az yol katederek bütün gezeceği yerleri nasıl tamamlayacağının hesaplanmasıdır. (gezgin satıcı problemi) Örneğin aşağıda verilen türkiye haritasında farklı iller ve bu illerin coğrafi konumları işaretlenmiştir. Kolaylık olsun diye iller arasındaki mesafeleri kuş uçuşu gitmek istersek bu durumda bütün illeri dolaşan en kısa yol aşağıdaki şekilde olur.[3]



Şekil 1

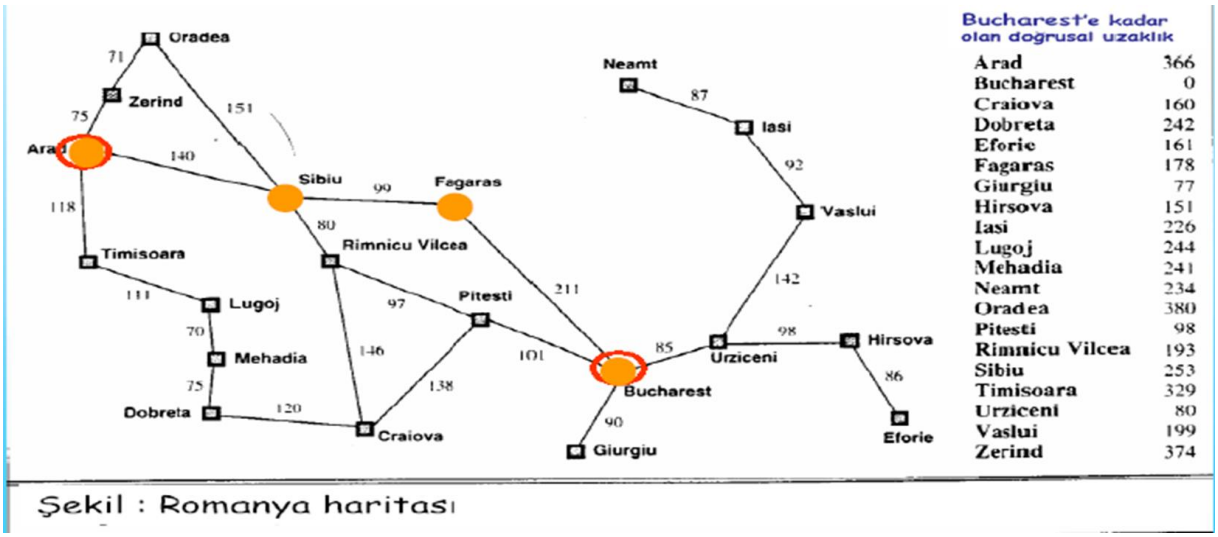
Bu haritada bütün noktaları dolaşan en kısa yol bulunmuştur. Bu işlemi yaparken kullanılabilecek bir iki yol aşağıda açıklanmıştır:

[1] Gazi Üniv. Müh. Mim. Fak. Der. Cilt 18, No 4, 2013.

Açgözülü yaklaşımı (greedy approach): Bu yaklaşımda başlangıç olarak seçilen herhangi bir şehire en yakın mesafedeki diğer şehir seçilir ve gezi listesine eklenir. Bu şekilde bütün şehirler gezi listesine eklenene kadar hep en son eklenen şehre en yakın şehir seçilir.[3]

Enküçük artış yöntemi (Smallest increase): Bu yöntemde ise toplam gezilecek mesafe her seferinde yeniden hesaplanmakta ve alternatiflerden hangisi eklenirse toplam gezi mesafesi en az olur diye sorglunarak yeni şehirler eklenmektedir. Örneğin başlangıçtan alınan 3 şehir gezi listesinde bulunurken 4. şehri seçecek olalım. Bu durumda bütün ihtimaller teker teker listeye eklenerek en az mesafe hangisinde bulunuyorsa bu şehir seçilir.[3]

Tepe Tırmanma (Hill Climbing) Aramaya göre
Arad-Sibiu-Fagares-Bucharest Toplam Uzaklık Örneği



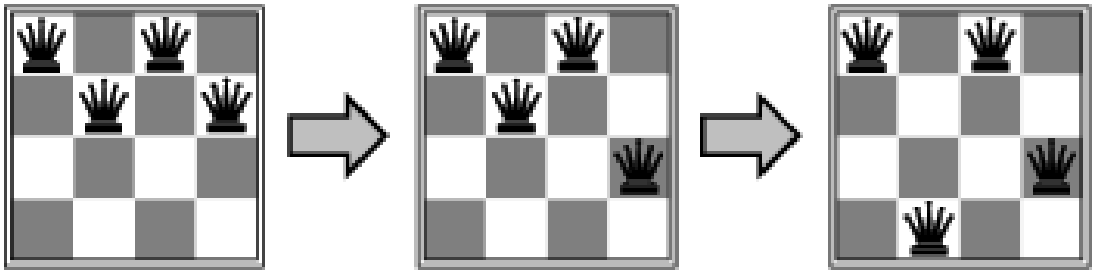
Şekil 2

Arad-Sibiu-Fagares-Bucharest Toplam Uzaklık **:**

140+99+211=450

N-Vezir N-Queens Problemi [4]

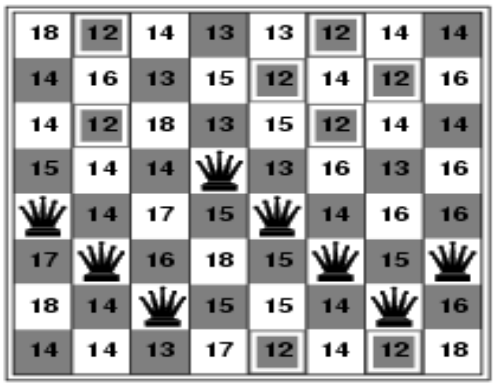
- **N** veziri **n × n** lik bir satranç tahtasına hiçbiri birbirini tehdit etmeyecek şekilde yerleştir.
- Hiçbir satır sutun ve diagonalde birden fazla vezir olmamalı.



Şekil3

8-Vezir problemini Tepe Tırmanma ile çözmek:

- Herbir sütuna bir rasgele bir vezirle başla. Her bir adımda sadece bir veziri sadece aşağı ya da yukarı x adım hareket ettirerek çözümü ara.
- h = birbirini tehdit eden vezir çifti sayısı
- Yukarıdaki tahta / durum için h = 17



Şekil4

8-Vezir'de Tepe Tırmanmanın performansı:

- Rasgele başlangıç değerleriyle
- Denemelerin %14'ünde çözer
- %86'sında lokal bir maksimuma takılır

- 8^8 = 2^24 ~ 17 milyon durum . [4]

Tepe Tırmanma Algoritması (Türkçe Açıklaması) [4]

- AÇIK listesine Başlangıç düğümünü **S** ‘yi ekle
- KAPALI listesini boşalt
- AÇIK listesi boş değilse, AÇIK listesinin en solundaki elemanı listeden çıkart . Bu elemanı *x* olarak adlandıralım.
- *x* ‘in çocuklarını bul
- *x* 'in tüm çocukları için

Eğer çocuk AÇIK ve KAPALI listesinde değil ise

Çocuğa bir sezgisel değer ata ve AÇIK listesine ekle.

Eğer çocuk AÇIK listesinde ise

Eğer daha kısa bir yol ile ulaşılmış ise AÇIK lisesindeki çocuk kaydının sezgisel değerini değiştir.

Çocuğun ebeveyn bilgisini güncelle.

Eğer çocuk KAPALI listesinde ise

Eğer daha kısa bir yol bulunmuş ise KAPALI listesinden çıkartıp yeni bilgilere göre güncelleyip AÇIK listesine ekle.

- *x* düğümünü KAPALI listesine ekle
- AÇIK listesini sezgisel değerlerine göre yeniden sırala (En iyi seçenek sola gelecek şekilde)
- 3. adıma git

Tepe Tırmanma Özellikleri

Tepe Tırmanma Algoritmasının bazı temel özellikleri şunlardır: [5]

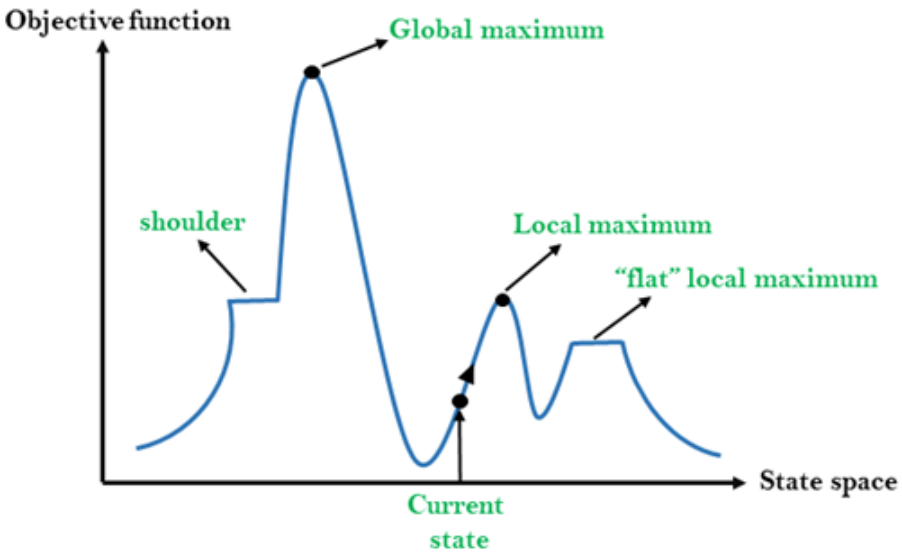
- **Üret ve Test değişkeni:** Tepe Tırmanışı, Üret ve Test yönteminin bir çeşididir. Üret ve Test Et yöntemi, arama alanında hangi yöne hareket edeceğinize karar vermeye yardımcı olan geri bildirim üretir.

- **Açgözlü yaklaşım:** Tepe tırmanma algoritması araması, maliyeti optimize eden yönde hareket eder.
- **Geri izleme yok:** Önceki durumları hatırlamadığı için arama alanını geri izlemez.

Tepe Tırmanışı için Durum-Uzay Diyagramı: [6]

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 mevcut. Y ekseni üzerindeki fonksiyonun maliyeti varsa, aramanın amacı küresel minimum ve yerel minimumları bulmaktır. Y ekseni işlevi Amaç işlevi ise, aramanın amacı küresel maksimum ve yerel maksimum değerlerini bulmaktır.



Şekil 5

Not: Bir tepe tırmanma algoritması düğümü, **durum** ve **değer** olmak üzere iki bileşene sahiptir. **Evolution :**Değişim , **Current state:** Mevcut durum

Eyalet uzay manzarasındaki farklı bölgeler:

Yerel Maksimum: Yerel maksimum, komşu eyaletlerinden daha iyi olan bir durumdur, ancak bundan daha yüksek olan başka bir durum da vardır.

Global Maximum: Global maximum, mümkün olan en iyi durum uzay manzara durumudur. En yüksek objektif fonksiyon değerine sahiptir.

Mevcut durum: Diyagramda bir ajanın halihazırda mevcut olduğu bir durumdur.

Düz yerel maksimum: Mevcut eyaletlerin tüm komşu devletlerinin aynı değere sahip olduğu manzaradaki düz bir alandır.

Omuz: Yokuş yukarı kenarı olan bir plato bölgesidir.

Temel olarak bir grafikte rastgele seçilen bir nokta için 3 farklı ihtimal bulunmaktadır: (M.Fatih AMASYALI 2012)

1. Noktanın bir tarafında problem iyileşirken diğer tarafında kötüleşmektedir. Bu durumda iyi tarafa doğru tırmanma algoritması devam eder.
2. Noktanın iki tarafında da problem sonucu kötüleşmektedir. Bu durumda bulunulan nokta problem için en iyi noktalardan (optimum points) birisidir. Elbette bu en iyi sonuç olmayabilir yani bu sonuçtan daha iyi sonuçlar olabilir ancak klasik tepe tırmanma algoritması bu diğer sonuçları bulamaz ve bu noktada kalır.
3. Noktanın iki tarafında da problem iyileşiyordur. Yani tesadüfen bulunulan nokta aslında problem için ulaşılabilecek en kötü noktalardan birisidir.Bu durumda tepe tırmanma algoritması yönlerden birisini seçerek tırmanmaya devam eder.

Tepe Tırmanma Algoritması Türleri: [6]

- **Basit Tepe Tırmanışı**
- **En dik çıkış Tırmanma**
- **Rassal-tekrar başlatmalı Tepe Tırmanma**

- **Stokastik Tepe Tırmanma**
- **İlk-seçimli Tepe Tırmanma**
- **Paralel Tepe Tırmanma**

Basit Tepe Tırmanışı

Basit tepe tırmanma, bir tepe tırmanma algoritması uygulamanın en basit yoludur. Yalnızca bir kerede komşu düğüm durumunu değerlendirir ve geçerli maliyeti optimize eden ve onu geçerli durum olarak ayarlayan ilk düğümü seçer . Sadece bir durum olup olmadığını kontrol eder ve mevcut durumdan daha iyi durum buluyorsa, başka bir durumda o durumda hareket eder. Bu algoritma aşağıdaki özelliklere sahiptir:

- Daha az zaman alıcı
- Daha az optimum çözüme sahiptir ve çözüm garanti edilmez

Basit Tepe Tırmanışı için Algoritma:

Adım 1: Başlangıç durumunu değerlendirin, eğer hedef durum ise, başarıyı ve Durdur'u döndürün.

Adım 2: Döngü Bir çözüm bulunana veya uygulanacak yeni bir operatör kalmayıncaya kadar.

Adım 3: Bir operatör seçin ve geçerli duruma uygulayın.

4. Adım: Yeni durumu kontrol edin:

a. Hedef durum ise, başarıyı iade edin ve çıkın.
b. Değil ise, mevcut durumdan daha iyiyse, yeni bir durumu geçerli durum olarak atayın.
c. Mevcut durumdan daha iyi değilse, 2. adıma dönün. Adım 5: Çıkış.

En dik Tepe Tırmanışı

En dik Yükseliş algoritması, basit yokuş tırmanma algoritmasının bir varyasyonudur. Bu algoritma, mevcut durumun tüm komşu düğümlerini inceler ve hedef duruma en yakın olan bir komşu düğümü seçer. Bu algoritma birden fazla komşu ararken daha fazla zaman harcar.

Dik yokuş yukarı tırmanma algoritması: [8]

Adım 1: Başlangıç durumunu değerlendirin, eğer hedef durum ise başarıyı döndürün ve durun, aksi takdirde mevcut durumu başlangıç durumu yapın.

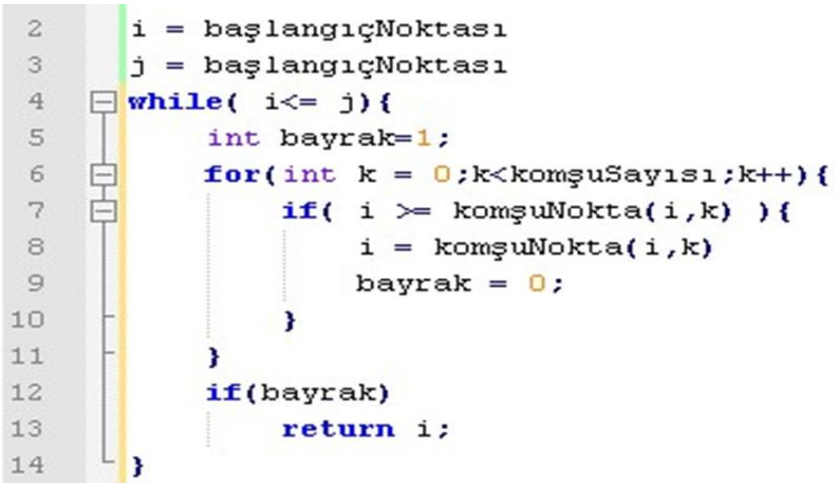
Adım 2: Bir çözüm bulunana veya geçerli durum değişmeyene kadar döngü yapın.

a. SUCC, mevcut durumun herhangi bir halefi bundan daha iyi olacak bir durum olsun.
b. Mevcut durum için geçerli olan her operatör için:

    Yeni işleci uygulayın ve yeni bir durum oluşturun.

    i. Yeni durumu değerlendirin.
    ii. Hedef durum ise, geri dönün ve çıkın, aksi takdirde SUCC ile karşılaştırın.
    iii. SUCC'den daha iyiyse, yeni durumu SUCC olarak ayarlayın.
    iv. SUCC geçerli durumdan daha iyiyse, mevcut durumu SUCC olarak ayarlayın.

c. Adım 5: Çıkış.



```
i = başlangıçNoktası
j = başlangıçNoktası
while( i<= j){
    int bayrak=1;
    for(int k = 0;k<komşuSayısı;k++){
        if( i >= komşuNokta(i,k) ){
            i = komşuNokta(i,k)
            bayrak = 0;
        }
    }
    if(bayrak)
        return i;
}
```

Şekil 6

Rassal-tekrar Başlatmalı Tepe Tırmanma

Algoritma, eğer tek tepe varsa çalışır. Lokal maksimumlara takılıp kalmayı önlemek için ve bulunan çözümü iyileştirmek için kullanılır. Tekrarlı olarak tepe tırmanmayı uygulanır (çoklu tepe varsa tekrar başlatılır). Olasılıksal tepe tırmanma sistemi olan PALO da kullanılmaktadır. PALO ‘’Sezgisel’’ düşünmeyi gerektiren bir tümevarımsal öğrenme modellemesidir.

NOT: Rastgele bir nokta seçmek için önerilen tepe tırmanma varyasyonu, ancak önceden ziyaret edilen düğümler dışındaki en düşük maliyeti seçmek, rastgele seçmekten daha iyidir. [9]

İlk-seçimli Tepe Tırmanma

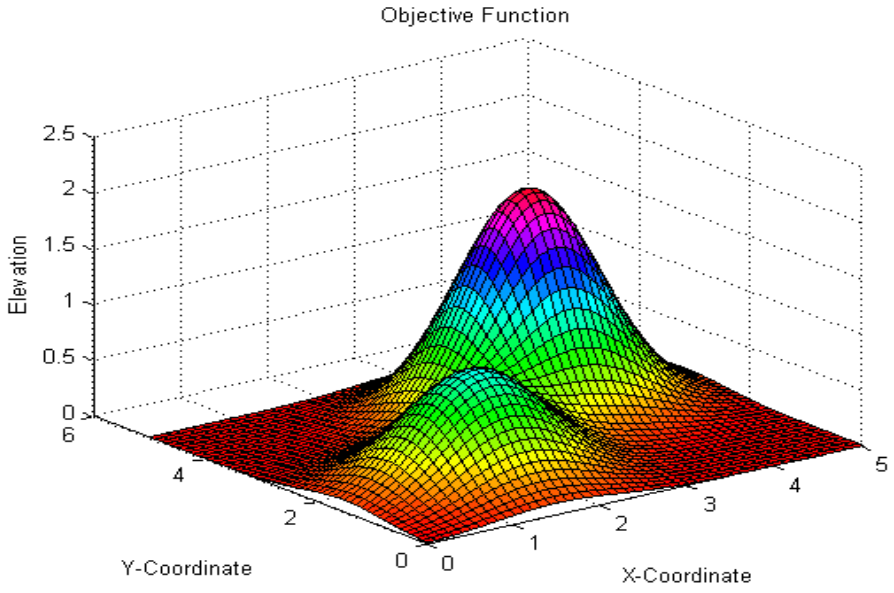
Stokastiktepe tırmanmayı, daha iyi bulunana kadar çocukları gelişigüzel üreterek kullanır.

Stokastik Tepe Tırmanışı

Stokastik tepe tırmanışı, hareket etmeden önce tüm komşusunu incelememektedir. Daha ziyade, bu arama algoritması bir komşu düğümü rastgele seçer ve geçerli durum olarak seçmeye veya başka bir durumu incelemeye karar verir. Stokastik bir tepe tırmanma algoritması bitişikteki tüm komşulara bakar, yokuş

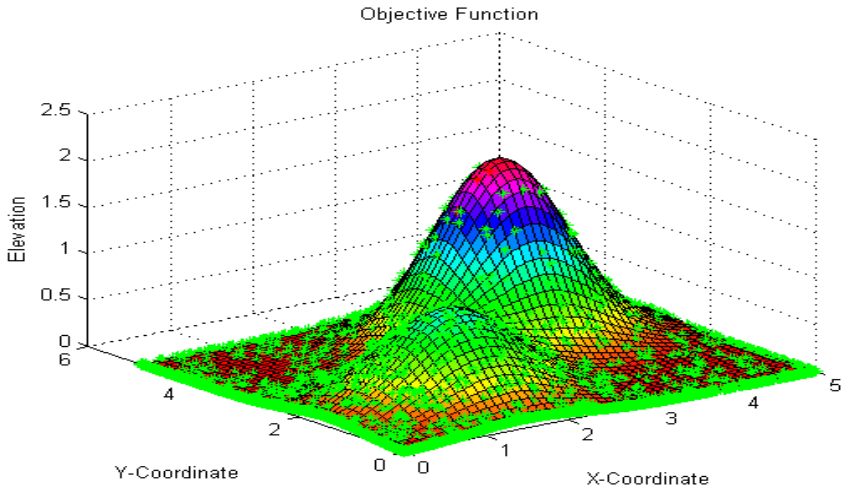
aşağı giden tüm komşuları yok sayar ve yokuş yukarı seçenekler arasında rastgele seçim yapar. Bu, yerel minimada yakınlaşmaya yatkındır.[11]

Stokastik tepe tırmanışı örnek



Şekil 7

Örneğin böyle bir yüzeyde rastgele bir noktadan başlarsak ne kadar yüksek olduğumuzu kontrol edebilir ve görebiliriz, ama en tepeden başladığımızı gerçekten varsayamayız bu durumda yapılacak en basit şey, bitişik noktalara bakmak, yokuş yukarı olanı bulmak ve artık yokuş yukarı gidene kadar bu şekilde ilerlemektir. Bu durumda sadece 1 tepe varsa iyi çalışır, ancak bu durumda alt zirveye takılmaya özellikle duyarlıdır rastgele bir yönde rastgele bir mesafeyi 'atlamak' ve orada kontrol etmek daha iyi bir yöntemdir. [11]



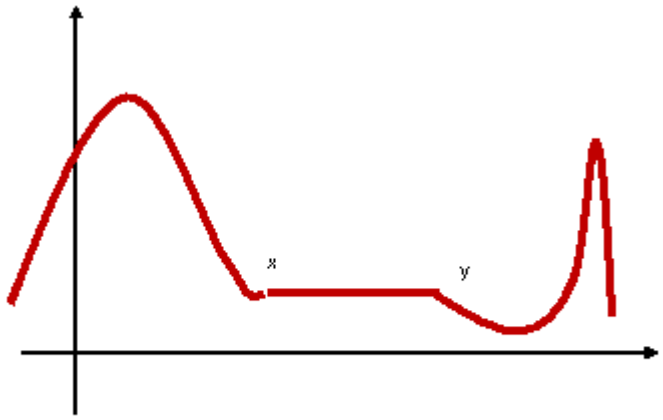
Şekil 8

Bu şekilde algoritmanın nasıl çalıştığı gösteriliyor. Yeşil yıldızların karmaşası, algoritmanın yeni bir noktaya 'atladığı' ve bir önceki noktadan daha düşük olduğunu bulduğu örneklerdir, bu yüzden algoritma geri döndüyor. Doğru ilerleme kırmızı renkte gösterilir ve rastgele olması nedeniyle görülmesi neredeyse imkansızdır. Bu yüzeyde, stokastik tepe tırmanma algoritması, rastgele başlangıç noktaları varsayarak, verilen zamanın sadece% 56'sında doğru zirveyi belirleyecektir. Yeterli zaman / yineleme verildiğinde, uygulamam sonunda her seferinde gerçek zirveyi bulacaktır.[11]

Tepe Tırmanma Algoritmasında Riskler: [7]

Tepe tırmanma algoritmaları genel olarak yerel bir başarı noktasında (local optimum point) takılmak gibi bir zafiyete sahiptirler.

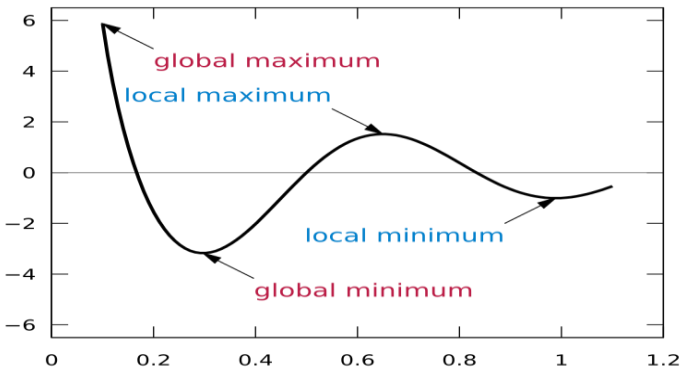
Örneğin aşağıdaki şekilde **X** ve **Y** noktaları arasında bir düzlük bulunmaktadır. Başlangıç noktası olarak bu aralıktaki herhangi bir noktadan başlanırsa algoritma komşuları aradığında daha iyi veya daha kötü bir sonuç bulamayacağı için hatalı karar verebilir.



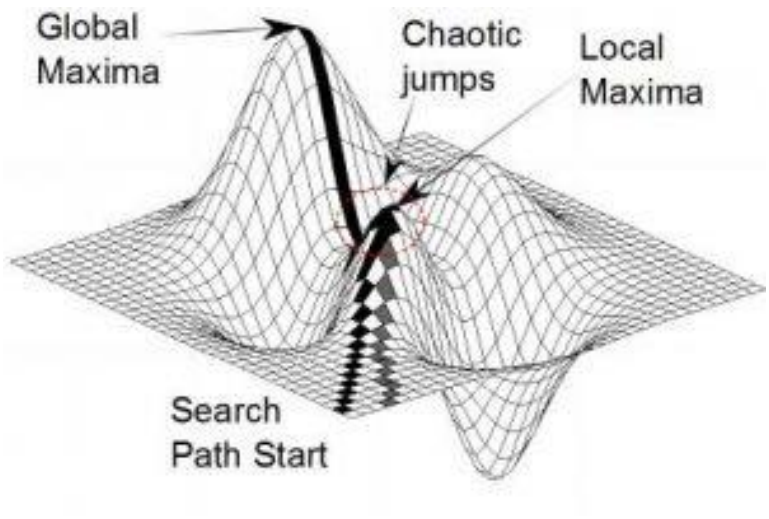
Şekil 9

Yerel Maksimum: Yerel maksimum, manzaradaki komşu durumların her birinden daha iyi olan bir tepe durumdur, ancak yerel maksimumdan daha yüksek başka bir durum da vardır. Tepe tırmanışı mutlaka küresel maksimumu bulamaz, bunun yerine yerel maksimuma yaklaşabilir . Buluşsal yöntem dışbükey ise bu sorun oluşmaz. Bununla birlikte, birçok fonksiyon dışbükey olmadığından, tepe tırmanışı genellikle küresel bir maksimuma ulaşamayabilir. Diğer yerel arama algoritmaları **stokastik tepe tırmanışı**, **rasgele yürüyüşler** ve **tavlama benzetimi** gibi bu sorunun üstesinden gelmeye çalışmaktadır .

Çözüm: Geri izleme tekniği, durum uzayı peyzajında yerel maksimum değerin bir çözümü olabilir. Geleceğin yolunun bir listesini oluşturulur, böylece algoritma arama alanını geri izleyebilir ve diğer yolları da keşfedebilir.



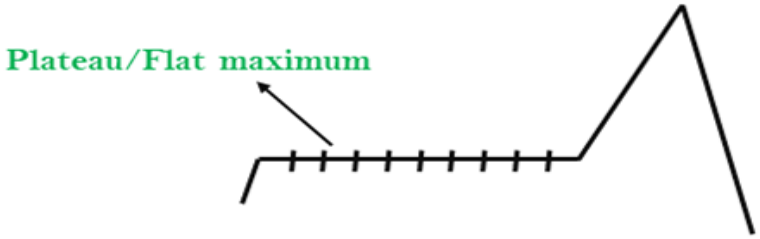
Şekil 10



Şekil 11

Plato: Bir plato, mevcut durumun tüm komşu durumlarının aynı değeri içerdiği arama alanının düz alanıdır, bu durumda algoritma hareket etmek için en iyi yönü bulamaz. Yayla bölgesinde bir tepe tırmanışı araştırması kaybolabilir.

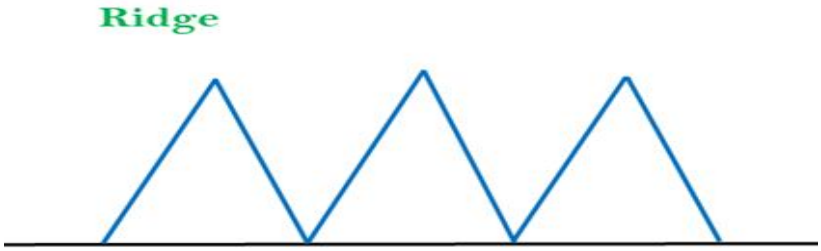
Çözüm: Plato için çözüm arama yaparken büyük adımlar veya çok küçük adımlar atmaktır. Mevcut durumdan çok uzakta olan bir durumu rastgele seçilir, böylece algoritma plato olmayan bölgeyi bulabilir.[7]



Şekil 12

Sırtlar ve sokaklar: Sırt, yerel maksimumun özel bir şeklidir. Çevresinden daha yüksek bir alana sahiptir, ancak kendisi bir eğime sahiptir ve tek bir hareketle ulaşılamaz.[7]

Çözüm: Çift yönlü arama kullanarak veya farklı yönlerde hareket ederek bu sorunu.iyileştirebiliriz.



Şekil 13

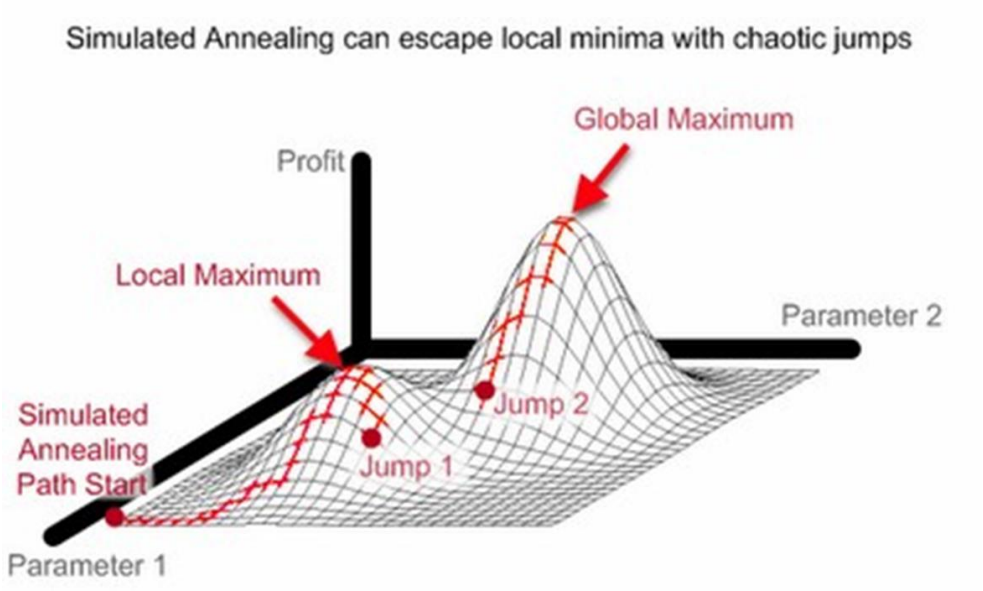
Not : Daha iyi sonuçlar için

- **Tepe tırmanmayı farklı başlangıçlarla tekrarlamak -Random-restart hill climbing**
- **Benzetimli Tavlama- Simulated annealing**
- **Paralel Tepe Tırmanma - Local beam search** gibi yöntemler kullanılır.

Tavlama Benzetimli

Tavlama Benzetimi (TB), karesel optimizasyon problemleri için iyi çözümler veren olasılıklı karar verme yöntemi olarak görülebilir. Kontrol parametresi "sıcaklık"tır ve minimizasyon problemleri için çözümün daha iyi bir çözüme ulaşmasının kabulu olasılığını değerlendirir. "Tavlama Benzetimi" ismi, katıların fiziksel tavlanma süreci ile olan benzerlikten ileri gelmektedir. TB algoritması, birbirlerinden bağımsız olarak, Kirkpatrick, Gelatt ve Vecchi (1983) ve Cerny (1985) tarafından ortaya konmuştur. Günümüze kadar, bilgisayar tasarımı, görüntü işleme, moleküler fizik ve kimya, çizelgeleme gibi farklı alanlardaki bir çok optimizasyon problemine uygulanmıştır.[2]

[2] Gazi Üniv. Müh. Mim. Fak. Der. Cilt 18, No 4, 2013



Şekil 14

Benzetimli tavlama aslında daha düşük bir değere doğru hareket etmeyen bir tepe tırmanma algoritmasıdır, eksik olmaması garanti edilmez, çünkü yerel bir maksimumda sıkışabilir. Ve eğer algoritma bir halefi hareket ettirerek rastgele bir yürüyüş uygularsa, tamamlanabilir ancak etkili olmayabilir. Simüle Tavlama , hem verimlilik hem de bütünlük sağlayan bir
algoritmadır.

function SIMULATED-ANNEALING(*problem*, *schedule*) **returns** a solution state
  inputs: *problem*, a problem
    schedule, a mapping from time to "temperature"

  current ← MAKE-NODE(*problem*.INITIAL-STATE)
  for $t = 1$ **to** ∞ **do**
    $T \leftarrow schedule(t)$
    if $T = 0$ **then return** *current*
    next ← a randomly selected successor of *current*
    $\Delta E \leftarrow next.\text{VALUE} - current.\text{VALUE}$
    if $\Delta E > 0$ **then** *current* ← *next*
    else *current* ← *next* only with probability $e^{\Delta E/T}$

Şekil 15

Mekanik terimde Tavlama , bir metalin veya camın yüksek bir sıcaklığa sertleştirilmesi ve ardından kademeli olarak soğutulması işlemidir, böylece bu,

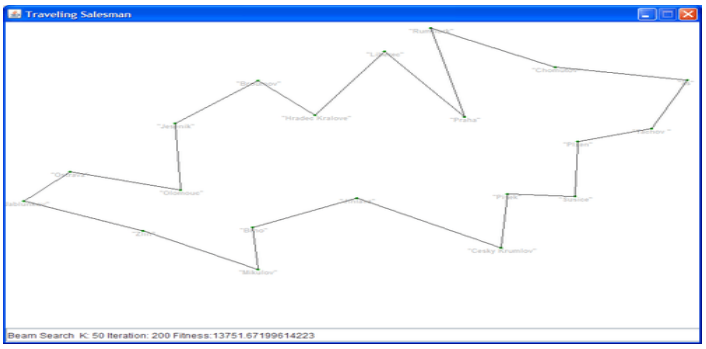
metalin düşük enerjili kristal bir duruma ulaşmasını sağlar. Aynı işlem, algoritmanın en iyi hareketi seçmek yerine rastgele bir hareket seçtiği simüle edilmiş tavlamada kullanılır. Rastgele hareket durumu iyileştirirse, aynı yolu izler. Aksi takdirde, algoritma olasılığı 1'den az olan yolu izler veya yokuş aşağı hareket eder ve başka bir yol seçer.

Paralel Tepe Tırmanma

Kısaca bu algoritmada ana fikir tek bir durumu izlemek yerine K taneyi izlemektir.

Paralel Tepe Tırmanma ile Traveling Salesman: [10]

TSP yapısı gereği sadece son durumla ilgilenen, geçilen ara adımların önemsiz olduğu bir problem. Örneğin satrançta ara adımların tek tek önemi vardır, çünkü rakip de sizin yaptığınıza göre bir hamle yapacaktır. Dolayısıyla TSP gibi bir problemde "bir adet durum tut, bu durumu en iyi hale getirmeye çalış" mantığı benimsenebilir.
Bu mantığın bir adım ilerisi "k adet durumu tut ve en iyi hale getirmeye çalış" mantığıdır. Paralel Tepe Tırmanma (Local Beam Search) bu mantıkla çalışır. O durumun iyiliği, aynı genetik algoritmadaki gibi bir fitness fonksiyonu ile ölçülür. Genetik Algoritma için belirlediğimiz fonksiyon burada da geçerli. (Rotanın toplam uzunluğu; kısaysa iyi, uzunsa kötü).



Şekil 16

Paralel Tepe Tırmanma ile TSP genel adımları:

- K adet rasgele rota oluşturulur. Aynı rotanın iki kez oluşmadığından emin olunur.
- Bu aşamada yukarıdaki rota kümesinde rasgele swap'lar (iki şehrin yerini değiştirmek) yapılarak yeni oluşan rotalar listeye eklenecek. Burada farklı yöntemler kullanılabilir. 1. adımda oluşturduğumuz her rota için, ardışık komşuların yerlerini değiştirerek yeni rotalar oluşturulur.
- Bu rotalar fitness değerlerine göre sıralanarak en iyi K tanesi seçilir.
- İstenen döngü sayısına ulaşılıncaya veya istenen düşüklükte fitness değeri elde edilene kadar 2. adımdan devam edilir.

20 şehirli haritada K=100 için 200'lü döngü içerisinde çalıştırılınca 9 sn. gibi bir sürede, Genetik Algoritma'nın 28 sn.'de ulaştığı sonuca ulaşılabiliyor.
K=50 için 200'lü döngüde çalıştırıldığında 2 sn. sonunda yine GA'nın 28 sn.'de verdiğinden daha iyi bir sonuç alınıyor. K değeri şehir sayısına bağlı olarak, iterasyon sayısı da K değerine bağlı olarak doğru seçildiğinde oldukça hızlı sonuç alınabiliyor.

Özet

Açgözlü ve içdeğişim sezgiselleri birçok yaklaşım algoritmasının temelidir. Her ikisi de gerekli başlangıç olurlu çözümü veya olurlu çözümleri kullanarak kesin algoritmalara yardımcı olurlar . Bununla birlikte kullanıcının sezgisel çözüm değerinin optimal çözüm değerinden uzaklaşmadığı noktasında tatmin olması için, nihai çözümde sezgisellerin kullanımında üst sınır olmalıdır Bilgisiz (kör) arama algoritmaları, düğümlerin amaca uzaklığı hakkında bir bilgiye sahip değildirler. Eğer belirli sayıda düğüm varsa, uygun çözüm bulunabilir.Bilgili (sezgisel) arama yaklaşımlarında, amaca olan tahmini uzaklık kullanılır. Amaca yakın olan düğümler ilk önce açılır. Çözümün bulunması garanti değildir. Bu yöntemlerin başarılı olmasında doğru değerlendirme fonksiyonunun seçilmesi önemli rol oynar.

Tepe tırmanma lokal bir yöntemdir, bir sonraki adımda ne yapacağını sadece seçimlerinin "anlık" sıralarına bakarak karar verir ve global bilgi sezgisel fonksiyona kodlanabilir. Büyük, inişli çıkışlı problem uzayları için çok etkili olabilir. Global

sezgisel ile hesapsal karmaşıklık artabilir ve diğer yöntemlerle birleştirildiğinde kullanışlı olabilir.

Kaynaklar

[1] From https://www.javatpoint.com/hill-climbing-algorithm-in-ai [Erişim Tarihi: 10.04.2020].

[2] From http://bilgisayarkavramlari.sadievrenseker.com/2009/12/02/tepe-tirmanma-algoritmasi-hill-climbing-algorithm/ [Erişim Tarihi: 09.04.2020].

[3] From http://www.bilgisayarkavramlari.com/2008/08/28/seyyar-tuccar-problemi-traveling-salesman-problem/ [Erişim Tarihi: 09.04.2020].

[4] From https://tr.qwe.wiki/wiki/Hill_climbing [Erişim Tarihi: 10.04.2020].

[5]Fromhttps://www.tutorialspoint.com/design_and_analysis_of_algorithms/design_and_analysis_of_algorithms_hill_climbing.htm [Erişim Tarihi: 11.04.2020].

[6] From https://www.geeksforgeeks.org/introduction-hill-climbing-artificial-intelligence/'/ [Erişim Tarihi: 11.04.2020].

[7] From https://medium.com/@bhavek.mahyavanshi50/introduction-to-hill-climbing-artificial-intelligence-a3714ed2d8d8 [Erişim Tarihi: 11.04.2020].

[8] https://www.it-swarm.dev/tr/artificial-intelligence/tepe-tirmanma-algoritmasi [Erişim Tarihi:14.04.2020].

[9]From https://www.edureka.co/blog/hill-climbing-algorithm-ai/#hillclimbingapplications [Erişim Tarihi:22.04.2020].

[10] From https://akaya.wordpress.com/2010/01/28/paralel-tepe-tirmanma-ile-traveling-salesman/[Erişim Tarihi:22.04.2020].

[11]From https://wmnoise.wordpress.com/2012/05/12/272/ [Erişim Tarihi:22.04.2020].