TAVLAMA BENZETİMİ ALGORİTMASI

TAVLAMA BENZETİMİ ALGORİTMASI

- Tavlama Benzetimi, Benzetimli Tavlama, Benzetilmiş Tavlama, Tavlama Benzeştirmesi gibi farklı tanımlamaları olan bu algoritma İngilizcede Simulated Annealing“ olarak geçmektedir.
- Katı bir malzemenin önce belirli bir sıcaklığa kadar ısıtılıp sonra soğutulması yoluyla malzemenin kalitesinin arttırılmasını sağlayan fiziksel tavlama sürecinin taklidine dayanmaktadır.

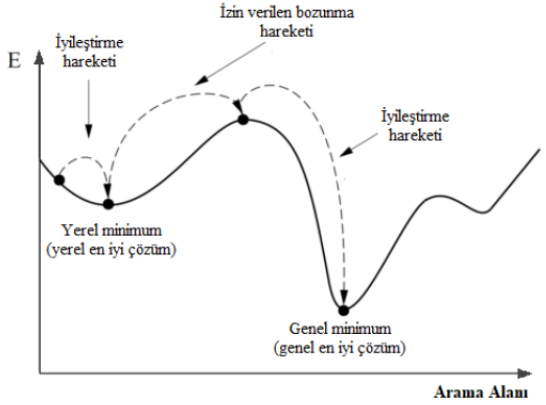
TAVLAMA BENZETİMİ ALGORİTMASI

- Bu yaklaşımda, metal gibi bir katı madde yüksek bir sıcaklığa kadar ısıtılır ve erimiş bir hale getirilir. Atomlar, bu durumda özgürce hareket edebilirler. Bununla birlikte, atomların hareketleri sıcaklığın düşürülmesi yoluyla kısıtlanır. Sıcaklık azaldıkça, atomlar daha az hareket etmeye eğilimlidir ve kristal formları mümkün olan en az dahili enerjiye sahiptir. Kristallerin oluşumu temelde soğutma oranı ile ilgilidir. Erimiş metallerin soğutma hızı çok hızlı olduğunda, bu sıcaklığın çok hızlı bir şekilde azaldığı anlamına geleceğinden kristal halini elde edemeyebilir. Bunun yerine, kristal duruma kıyasla daha yüksek bir enerji durumuna sahip olan bir polikristalin durumuna ulaşabilir. Mühendislik uygulamalarında, hızlı soğutmanın sonunda malzemenin içinde kusurlar meydana gelir. Bu nedenle en düşük enerji durumuna (iç enerji) ulaşmak için ısıtılmış katı (erimiş metal) sıcaklığının yavaş ve kontrollü bir oranda azaltılması gerekir. Yavaş bir oranda bu soğutma, tavlama olarak adlandırılır.

Genel bir minimum noktası bulmak amacı ile problemin farklı zamanlarındaki sonuçlarını elde ederek bu sonuçlardan iyi olan değere doğru hareket edip birden fazla çözümü test ederek en iyi sonucu bulmaya yardımcı olan yöntem olan Simulated Annealing (Benzetimli Tavlama) yöntemi de bir optimizasyon problemi çözüm yöntemidir.

Tavlama benzetiminde soğutma işleminin kontrolünü, bir diğer deyişle hangi hızda yapılacağını ayarlamayı sağlayan fonksiyon, optimum çözümün aranması esnasında daha iyi olmasalar da genel en iyi çözümün bulunmasına yardımcı olacak çözümlerin kabul edilebilirliğine ilişkin olasılığı ifade eder. Aramaya ilk başlandığında bu olasılığın daha iyi olmayan çözümlerin kabul edildiği ve bu doğrultuda yerel en iyi çözümlerden kaçınarak genel en iyi çözümü bulmayı sağlayan yeterince yüksek bir değerde olacak şekilde ayarlanması gerekir. En iyi çözümün aranması süresince, olasılığın gittikçe azaldığı dolayısıyla yerel en iyi çözümden kaçmanın zorlaştığı görülür. Burada amaçlanan, eldeki yerel en iyi çözümün komşuları aranarak yeni bir yerel en iyi çözüme geçmektense genel en iyi çözüme ulaşılmaya çalışılmasıdır.

Tavlama benzetimi algoritması rastgele başlangıç çözümüyle başlar ve her yinelemede yerel komşulukla sonraki çözümü üretir. Sistemin enerji düzeyini temsil eden 'E' amaç fonksiyonunu geliştiren (maddenin/sistemin enerjisini azaltan) yani eniyileyen yeni çözüm her zaman kabul edilir. Öte yandan, sistemin sıcaklığını arttırmaya müsaade eden ya da sistemdeki amaç fonksiyonundan uzaklaşmaya/bozunmaya belirli bir düzeyde izin veren geçici çözüm önerileri de kabul edilmektedir.



Mevcut ve yeni en iyi çözüm arasındaki enerji farkı *(ΔE)*, amaç fonksiyonunun mevcut çözümün komşuluğu yoluyla oluşturulan yeni çözümü ile amaç fonksiyonunun mevcut değeri arasındaki farkı, yani ilk bulunan en iyi çözüm ile sonrasında bulunan en iyi çözüm arasındaki farkı gösterir. Daha iyi olmayan çözümün kabul edilme olasılığı aşağıda gösterildiği gibi Boltzmann dağılımına dayanır:

$$P(\Delta E, T) = e^{\frac{\Delta E}{T}} = e^{\frac{E(n+1) - E(n)}{T}}$$

Burada; *E(n)* mevcut çözümün amaç fonksiyonunu (mevcut enerji düzeyini), *E(n+1)* yerel komşulukla elde edilen yeni çözümün enerji düzeyini, *T* ise sistemin gerçek sıcaklığını temsil eder. *T* değeri yani sıcaklık arttığında *P* değeri, yani daha iyi olmayan bir komşu çözümün kabul edilme olasılığı azalır. Bu noktada belirli bir sıcaklık seviyesinde toplamda kaç yeni çözümün araştırılacağı araştırmacı tarafından belirlenir. Ancak kararlı bir duruma ulaşıldığında (çözümlerde daha fazla ilerleme olmadığı durumlarda), sıcaklık genellikle azaltılmış olur.

Algoritma adımları

1. Bir ilk vektör x_1 ile başlanır (f_{xi}) ve amaç fonksiyonu değeri en iyi kabul edilir. Ardından fonksiyona yüksek bir sıcaklık değeri (T_{max}) atanır.

2. Rastgele yeni bir çözüm $f(x_{(i+1)})$ oluşturularak yeni çözüm ile mevcut fonksiyon değeri arasındaki fark bulunur. Bunun için $\Delta f=f_{(i+1)}-f_i=f(x_{(i+1)})-f(x_i)$ hesaplanır.

3. Yeni çözümün mevcut çözümden daha iyi olup olmadığı belirlenir. Eğer fark (Δf) istenen yöndeyse yani yeni çözüm daha iyiyse kabul edilir, değilse bir sonraki aşamaya geçilir (Bazı durumlarda minimize etme bazı durumlarda ise maksimize etme amaçlandığından minimizasyon durumunda farkın negatif değerde olması yani yeni bulunan çözümün daha düşük değerde olması, maksimizasyon durumunda ise farkın pozitif değerde olması yani yeni bulunan çözümün daha yüksek değerde olması istenir).

4. $1/(1+e^{(-\Delta f/T)})$ değeri rastgele üretilen bir sayının (Z) değerinden büyükse yeni çözüm kabul edilir, değilse eski çözüm halen en iyi çözüm olmaya devam eder.

5. x_{i+1} noktası reddedilirse eski çözüm yani (f_{xi}) halen en iyi çözüm olmaya devam eder, ardından algoritma tarafından rastgele yeni bir çözüm x_{i+1} üretilir (Algoritmanın kabul olasılığına göre daha kötü bir noktayı kabul edebileceği akılda bulundurulmalıdır).

SA- Özet

- **Metallerin tavlama sürecinden esinlenerek geliştirilmiştir.**
- **Metallerin ısıtılarak şekil verilmesi** ve **kontrollü bir şekilde soğutularak şeklin kalıcı hale getirilmesi** sürecine benzetim yapılmıştır.
- Tavlama benzetimi **sürekli fonksiyonların optimizasyonunda** veya **kombinatoryal optimizasyon problemlerinde** kullanılabilir.
- **Soğutma süresi uzun olursa** algoritma **global optimuma yakınsar.**
- **Komşu çözüm üretme maliyeti** diğer algoritmalara göre **genellikle daha düşüktür.**

SA -Özet

- **Isının yüksek olduğu durumlarda** çözüm uzayında **geçişler kolay yapılır** (yüksek ısıda metale kolay şekil verme).
- **Isı düştükçe** çözüm uzayında **lokal bölgelerde arama yapılır** (düşük ısıda metalin şekli çok değişmez).
- **Isı** belirlenen **eşik değerin altında düştüğünde** optimum değerin elde edilmesi amaçlanır.
- Isı değerine bağlı **rastgele bir çözüme geçme olasılığı her zaman vardır.**

SA-Özet

- Tavlama benzetimi algoritması **yüksek bir sıcaklık değeriyle başlar.**
- Her bir hesaplama adımında **mevcut çözümün komşuları arasından çok sayıda çözüm üretilir.**
- **Yeni çözümler** belirlenen kriterlere göre **kabul edilir** veya **reddedilir.**
- Her bir hesaplama adımından sonra **sıcaklık belirlenen bir fonksiyona göre azaltılır.**
- Algoritmayı sonlandırma şartları:
  - **İstenen iterasyon sayısına ulaşılması**
  - **Sıcaklığın minimum değerine ulaşması**
  - **İstenen bir çözüme ulaşılması**

SA-Özet

Pseudocode for Simulated Annealing

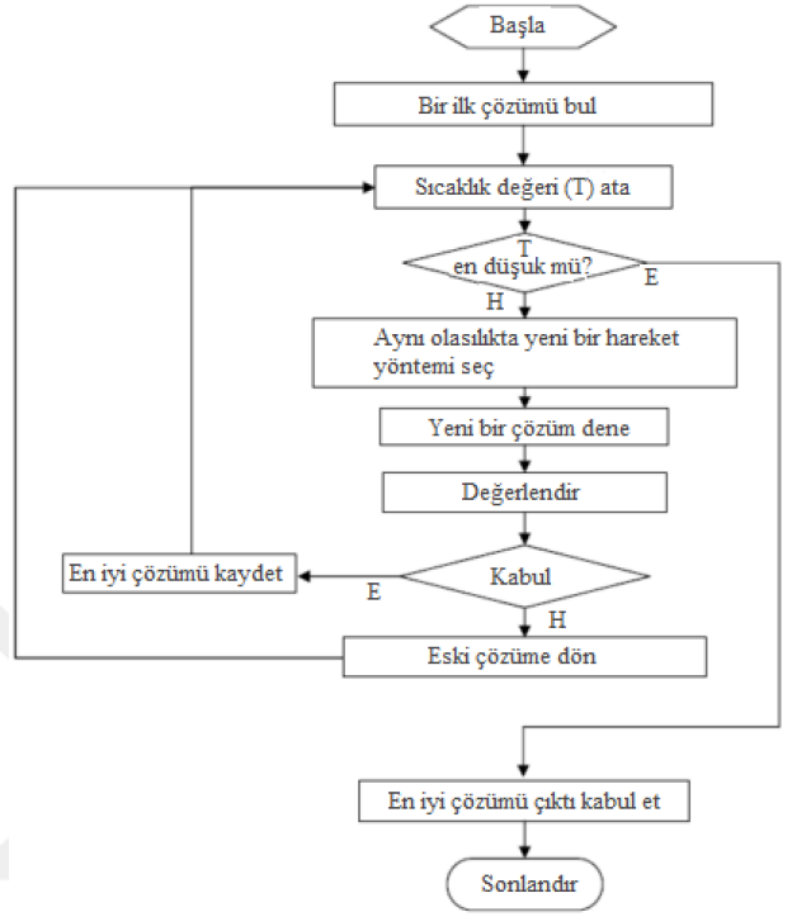
Input: ProblemSize, $iterations_{max}$, $temp_{max}$
Output: S_{best}

$S_{current} \leftarrow$ `CreateInitialSolution(ProblemSize)`;
$S_{best} \leftarrow S_{current}$;
for $i = 1$ **to** $iterations_{max}$ **do**
  $S_i \leftarrow$ `CreateNeighborSolution(`$S_{current}$`)`;
  $temp_{curr} \leftarrow$ `CalculateTemperature(`i, $temp_{max}$`)`;
  if `Cost(`S_i`)` $\leq$ `Cost(`$S_{current}$`)` **then**
    $S_{current} \leftarrow S_i$;
    if `Cost(`S_i`)` $\leq$ `Cost(`S_{best}`)` **then**
      $S_{best} \leftarrow S_i$;
    end
  else if `Exp(` $\frac{\texttt{Cost}(S_{current}) - \texttt{Cost}(S_i)}{temp_{curr}}$ `)` $>$ `Rand()` **then**
    $S_{current} \leftarrow S_i$;
  end
end
return S_{best};

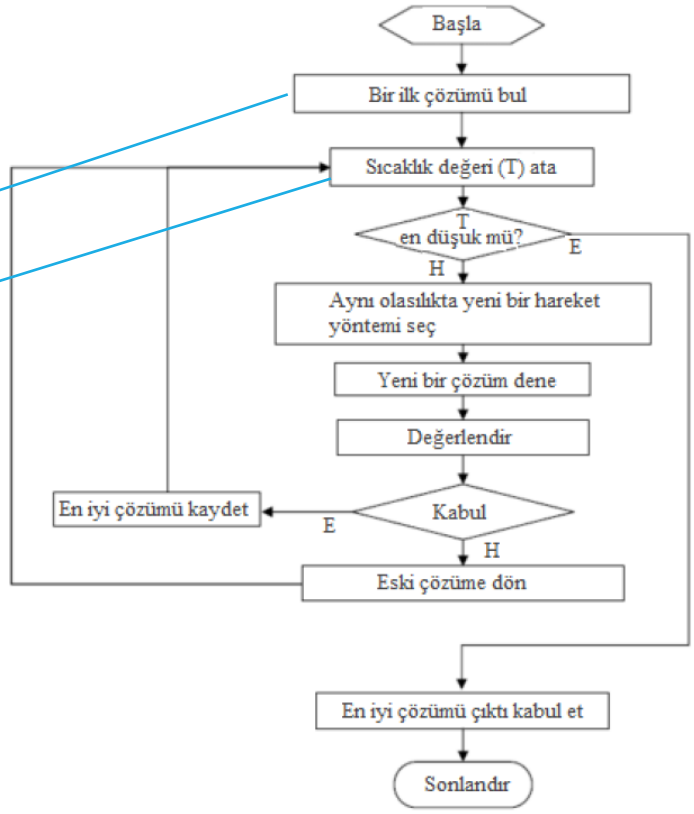
SA-Özet

- **Parametreler**
  - Başlangıç sıcaklığı
  - Her sıcaklıkta üretilecek çözüm sayısı fonksiyonu
  - Sıcaklık azaltma fonksiyonu
  - Algoritmayı durdurma şartı

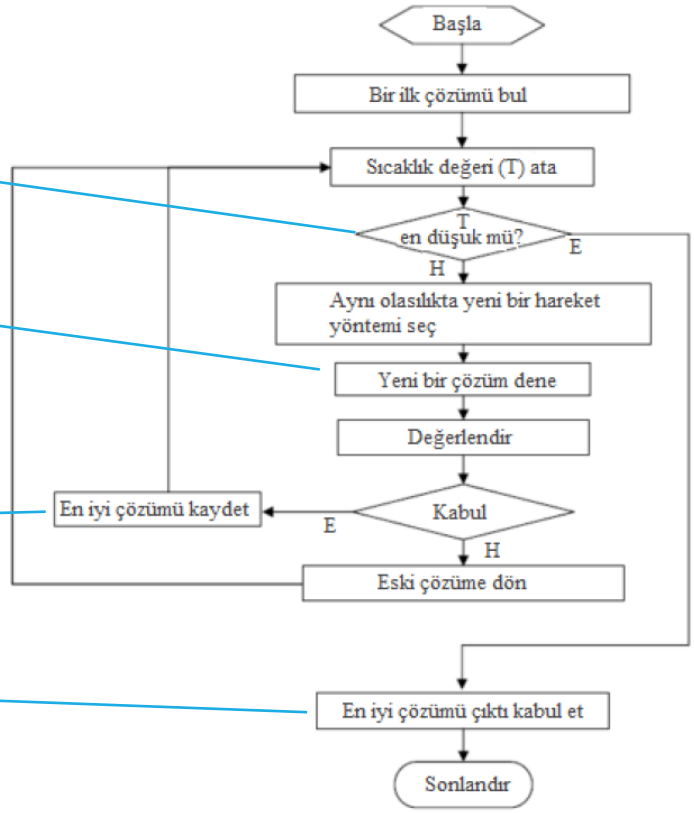
Akış Şeması


Başla
Bir ilk çözümü bul
Sıcaklık değeri (T) ata
T
en düşük mü?
E
H
Aynı olasılıkta yeni bir hareket yöntemi seç
Yeni bir çözüm dene
Değerlendir
En iyi çözümü kaydet
E
Kabul
H
Eski çözüme dön
En iyi çözümü çıktı kabul et
Sonlandır

```python
import numpy as np
import matplotlib.pyplot as plt

def fun(x):
    return sum(np.power(x,2))

alt = -5
ust = 5
problem_boyutu = 4
cozum = np.random.ranf([problem_boyutu]) * (ust-alt) + alt
obj = fun(cozum)

objit = list()
iterasyon = 0

cozumeniyi = cozum;
objeniyi = obj;

T = 100000
Tend = 0.1
delta = 0.10
soguma_katsayisi = 0.99

alt_degisim = (alt-ust) * delta/2
ust_degisim = (ust-alt) * delta/2
```



```
while T>Tend:

    degisim_mikrari = np.random.ranf(problem_boyutu) * (ust_degisim-alt_degisim) + alt_degisim
    komsu = cozum + degisim_mikrari;
    obj_komsu = fun(komsu);

    if obj_komsu<=obj:
       cozum=komsu;
       obj=obj_komsu;
    else:
        de = obj_komsu-obj;
        pa = np.exp(-de/T);
        rs=np.random.ranf()
        if rs<pa:
            cozum=komsu;
            obj=obj_komsu;


    T=T*soguma_katsayisi;

    objit.append(obj)

    if objit[iterasyon] < objeniyi:
        objeniyi = obj;
        cozumeniyi = cozum;

    iterasyon=iterasyon+1;

print("Optimum Karar değişkenleri :",cozumeniyi)
print("Optimum Çözüm :",objeniyi)

plt.plot(objit)
plt.xlabel("iterasyon")
plt.show()
```



TAVLAMA BENZETİMİ (PYTHON KOD)

```python
import numpy as np
import matplotlib.pyplot as plt

def fun(x):
    return sum(np.power(x,2))

alt = -5
ust = 5
problem_boyutu = 4
cozum = np.random.ranf([problem_boyutu]) * (ust-alt) + alt
obj = fun(cozum)

objit = list()
iterasyon = 0

cozumeniyi = cozum;
objeniyi = obj;

T = 100000
Tend = 0.1
delta = 0.10
soguma_katsayisi = 0.99

alt_degisim = (alt-ust) * delta/2
ust_degisim = (ust-alt) * delta/2
```

```python
while T>Tend:

    degisim_mikrari = np.random.ranf(problem_boyutu) * (ust_degisim-alt_degisim) + alt_degisim
    komsu = cozum + degisim_mikrari;
    obj_komsu = fun(komsu);

    if obj_komsu<=obj:
       cozum=komsu;
       obj=obj_komsu;
    else:
        de = obj_komsu-obj;
        pa = np.exp(-de/T);
        rs=np.random.ranf()
        if rs<pa:
            cozum=komsu;
            obj=obj_komsu;


    T=T*soguma_katsayisi;

    objit.append(obj)

    if objit[iterasyon] < objeniyi:
        objeniyi = obj;
        cozumeniyi = cozum;

    iterasyon=iterasyon+1;

print("Optimum Karar değişkenleri :",cozumeniyi)
print("Optimum Çözüm :",objeniyi)

plt.plot(objit)
plt.xlabel("iterasyon")
plt.show()
```

TABU Arama Algoritması

- TA algoritması, temelde son çözüme götüren adımın dairesel hareketler oluşturmasını engellemek için cezalandırılarak bir sonraki döngüde tekrar edilmesinin yasaklanması hedeflemektedir.

TABU

- TA algoritması döngü ya da çalışma süresi boyunca üretilen çözümler ile ilgili bilgileri kayıt eder yani hafızasında tutar. Tabu listesi olarak da adlandırılan bu hafızada saklanan bilgiler, araştırma uzayında yeni çözüm kümelerinin oluşturulmasında karşılaştırma amaçlı kullanılır. TA algoritması, mevcut çözümlerden küçük bir değişim ile elde ettiği denenmemiş bir çözüm kümesi üreterek optimizasyona başlamaktadır.

TABU

- TA algoritmasında, çözüm uzayında bir yerel minimum noktasında takılmayı engelleyebilmek için oluşturulan yeni çözüme, o anki çözümden daha kötü olsa bile izin verilir. Ancak kötü çözüme izin verilmesi algoritmayı bir kısır döngüye girmesine neden olabilir. Algoritmanın kısır döngüye girmesine engel olmak için, bir tabu listesi oluşturulur ve o anki çözüme uygulanmasına izin verilmeyen tüm yasaklı hareketler tabu listesinde saklanır.

TABU

- Bir hareketin tabu listesine alınıp alınmayacağını belirlemek için, tabu kısıtlamaları adı verilen bazı kriterler kullanılmaktadır. Tabu listesinin kullanılması, belirli bir iterasyon sayısınca daha önce denenmiş çözümlerin tekrar edilmesini engellediği için arama esnasında bir bölgede takılma ihtimalini azaltmaktadır.

TABU

- TA, mümkün bir çözüm ile başlar. Bu çözüm, problemin matematiksel ifadesinde geçen kısıtları tatmin eden bir çözümdür. Tabu aramanın performansı başlangıç çözümüne bağlıdır. Bu nedenle mümkün olduğunca iyi olan bir çözüm ile başlanılır. Tabu listesi ilk giren ilk çıkar (first in first out = FIFO) mantığında çalışan bir listedir. Algoritmada belirlenen karakteristik özelliklere göre tabu listesi sürekli olarak yukarıdan doldurulmaya başlar. Bulunan elemanların sayısı liste uzunluğunu aştığında yeni gelen elemanın listeye eklenmesi için listenin en sonundaki eleman listeden çıkarılır.

TA-Örnek

TABU ARAMA

⊛ Bir sırt çantası problemının en iyi çözümünü $t_{max} = 5$ için tabu arama ile inceleyiniz.

(Z) max $18x_1 + 25x_2 + 11x_3 + 14x_4$

Kst $2x_1 + 2x_2 + x_3 + x_4 \leq 3$

$x_1, x_2, x_3, x_4 = 0$ veya 1

$X^{(0)} = (1,0,0,0)$

⊛ Böyle bir problemde değişkenlerin değerini 0 ise 1 ve 1 ise 0 yaparak komşu çözümleri bulmaya çalışırsak

t	x^t	amaç fonk. Z	mevcut z^*	Hamle	Δz
0	(1,0,0,0)	18	18	J=4	14
1	(1,0,0,1)	32	32	J=1	-18
2	(0,0,0,1)	14	32	J=2	25
3	(0,1,0,1)	39	39	J=4	-14
4	(0,1,0,0)	25	39	J=3	11
5	(0,1,1,0)	36	39	stop	—