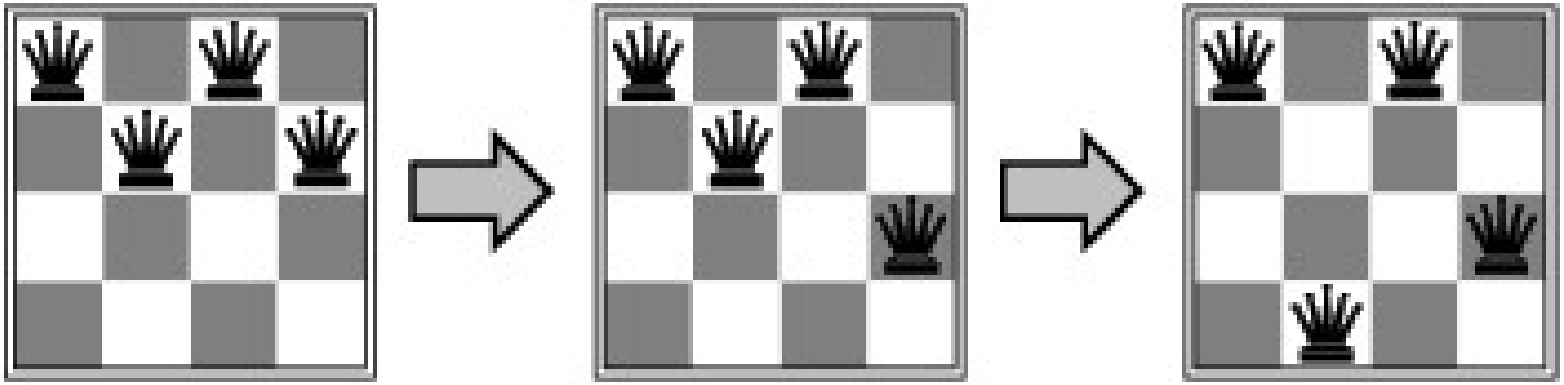
Lokal Arama Algoritmaları

Lokal Arama Algoritmaları
Local search algorithms

- Birçok optimizasyon probleminde, hedefe giden yol / uygulanan hareketler önemsizdir. Hedef durumun kendisi istenen çözümdür.
- Amaç arama uzayında istenen kısıtlara / özelliklere sahip / fayda fonksiyonunu maksimum yapan durumu bulmaktır. Örnek: n-vezir, zirve bulmak
- Bu durumlarda lokal arama algoritmaları kullanılır.
- Hafızada sadece mevcut durumu tut. Onu düzeltmeye çalış.
- Çok az hafıza gereksinimi

n-vezir *n*-queens

- N veziri $n \times n$ lik bir satranç tahtasına hiçbiri birbirini tehdit etmeyecek şekilde yerleştir.
- Hiçbir satır sütun ve diyagonalde birden fazla vezir olmamalı.
- Bir durumdan başla onu iyileştirerek devam et.



Tepe Tırmanma
Hill Climbing

- Yoğun bir siste, Everest Dağına tırmanmaya benzer. Sadece etkin durumun bilgisini tutar.
- Ana düşünce : Her zaman, şimdiki durumu en fazla geliştiren yönde adım at.
- Best-first Search'e benzer.
- Öğrenme algoritmalarında (ör: YSA) popülerdir.
- Yaylada ve sıralı tepelerde şaşabilir.

Çok nadiren

Genelde

Tepe Tırmanma Algoritması*

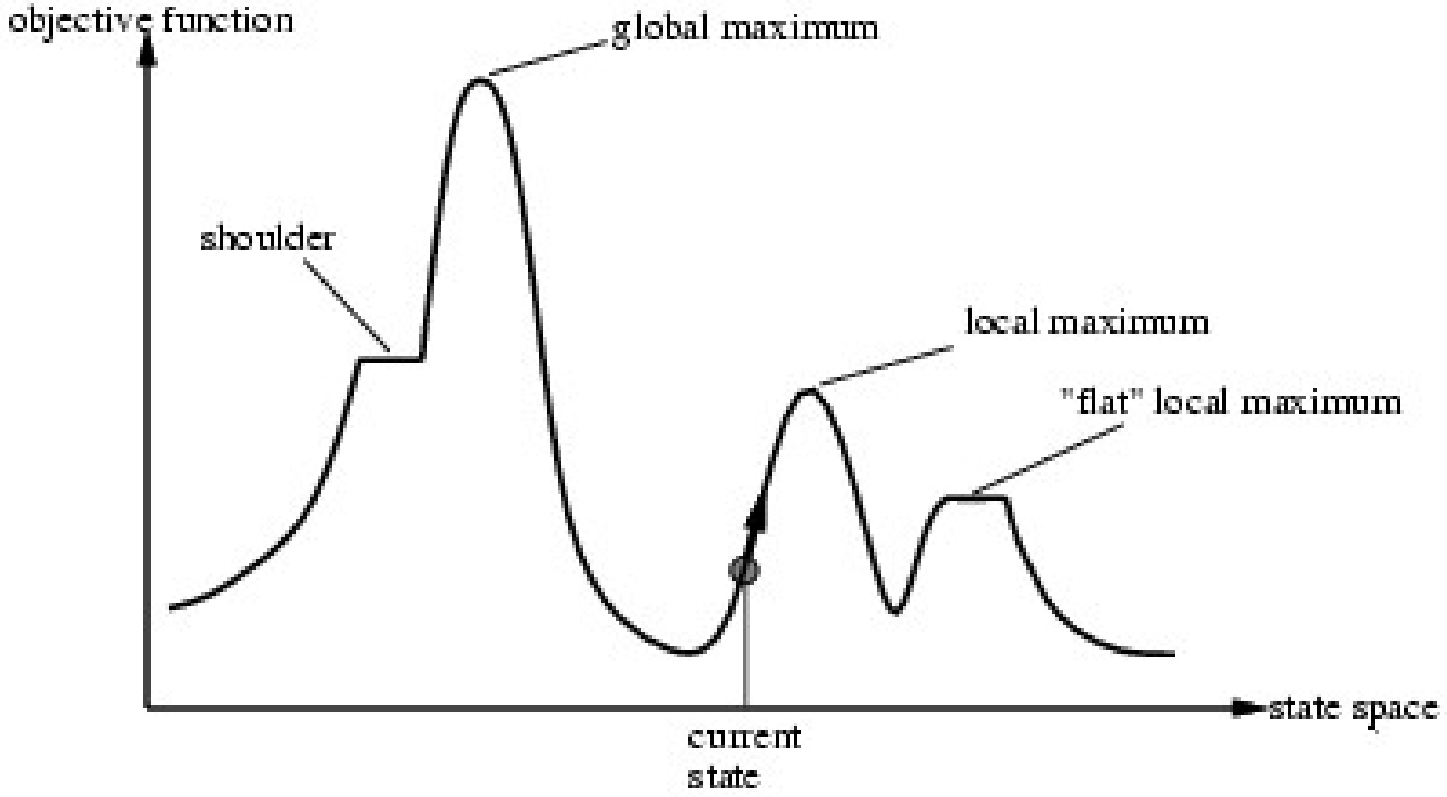
function HILL-CLIMBING(*problem*) **return** a state that is a local maximum
    input: *problem*, a problem
    local variables: *current*, **a node.**
                neighbor, **a node.**

    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*

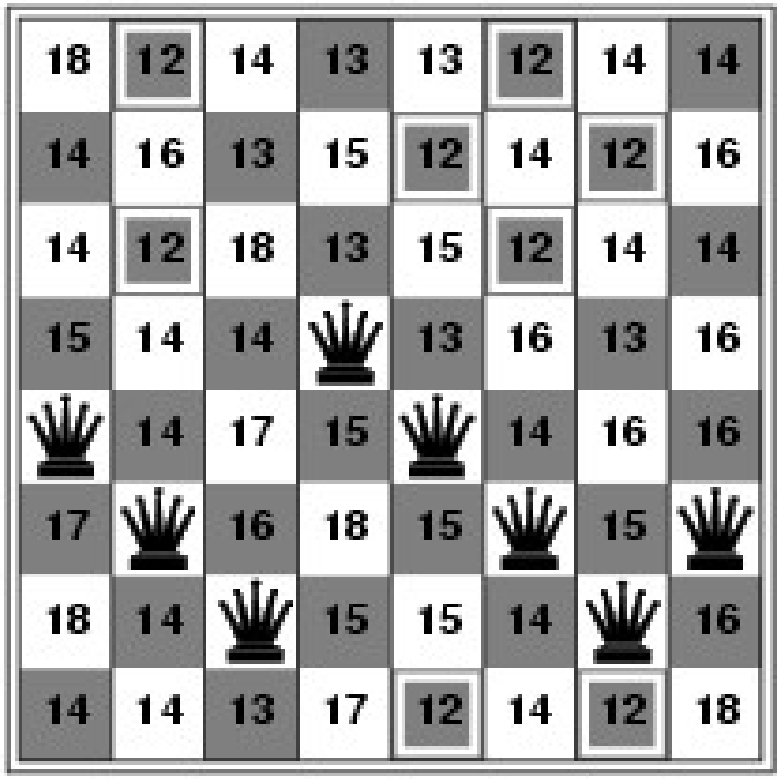
[*] https://github.com/aimacode/aima-pseudocode/blob/master/md/Hill-Climbing.md

Tepe Tırmanmanın Problemleri

- İlk duruma bağlı
- Lokal maksimum, plato ve sırtlara takılabilir

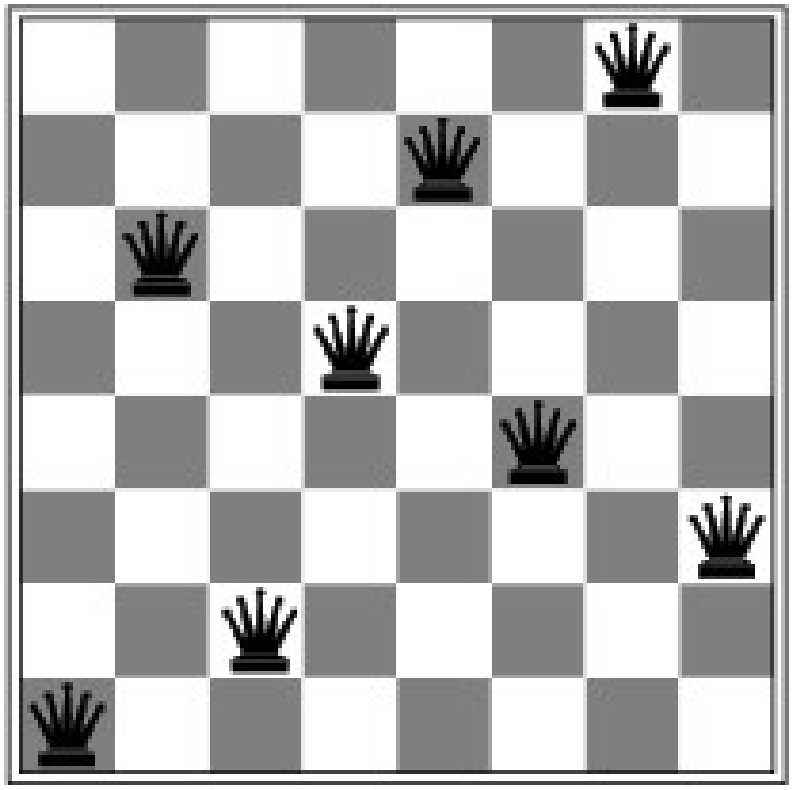


8-Vezir problemini Tepe Tırmanma ile çözmek



- *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*

8-vezir çözüm örneği



- $h = 1$

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

Bazı Çözüm Alternatifleri

- Tepe tırmanmayı farklı başlangıçlarla tekrarlamak - Random-restart hill climbing
- Benzetimli Tavlama- Simulated annealing
- Paralel Tepe Tırmanma - Local beam search

Benzetimli Tavlama- Simulated annealing

- Ana fikir : Yerel Maksimum'dan kaçmak için, istenmeyen hareketlere izin ver.
- Rasgele bir hareket üret. İyileşme varsa kabul et. Yoksa zamanla ve kötüleşme miktarıyla azalan bir olasılıkla kabul et.
- Zaman içinde rasgele hareketin boyutu (yeni noktanın uzaklığı bir dağılımdan üretilirse) ve kabul olasılığı azaltılır.

Benzetimli Tavlama Algoritması*

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 ← *schedule(t)*

if *T* = 0 **then return** *current*

next ← a randomly selected successor of *current*

ΔE ← *next*.VALUE - *current*.VALUE

if *ΔE* > 0 **then** *current* ← *next* *(iyileşme varsa kesin kabul)*

else *current* ← *next* only with probability $e^{\Delta E/T}$ *(iyileşme yoksa belki)*

Schedule: $T=T_0*0.95^t$ Belki, kötüleşme miktarına (*ΔE) ve geçen zamana bağlı*

[*] https://github.com/aimacode/aima-pseudocode/blob/master/md/Simulated-Annealing.md

Paralel Tepe Tırmanma - Local beam search

- Ana Fikir: Tek bir durumu izlemek yerine K taneyi izle
- K adet rasgele üretilmiş durumla başla
- Her bir iterasyonda k durumun hepsiden gidilebilecek tüm durumları üret.
- Bu durumlardan biri hedefse dur. Değilse, en iyi k tanesini mevcut durumlar olarak ata ve bir önceki adıma dön.

Bozuk TV

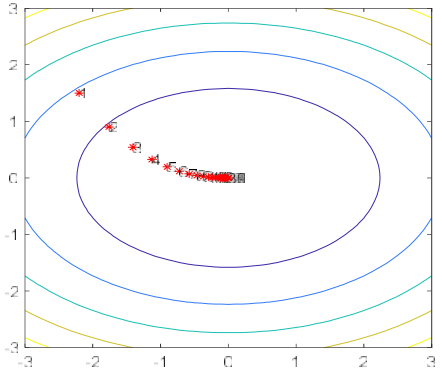
- Televizyonunuzun görüntüsü bozuk.
- Görüntü ayarı için 4 kontrol düğmesi var.
- Her bir düğmenin 100 farklı pozisyonu var.
- Nasıl bir yol izlersiniz?

- Peki ya kasa açsanız ☺

Ayrık / sürekli uzaylar

- Ayrık uzayda lokal arama: Bir noktadan gidilebilecek noktaların sayısı sınırlı. Hepsi denenebilir ☺ (8 vezir, labirent, çizge).
- Sürekli uzayda lokal arama: Bir noktadan sonsuz noktaya gidilebilir. Hepsi denenemez ☹ Hangi yöne, ne kadar? Eğim bize yol gösterir ☺ (gradient descent, konveks opt., YSA opt.)

$$f(x_1, x_2) = x_1^2 + 2x_2^2$$



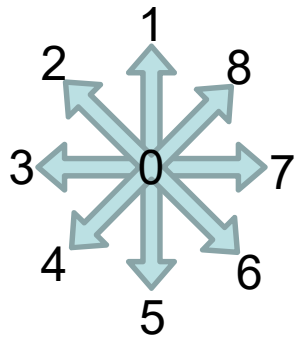
Uygulama: bir alanı tarama

- Amaç: N*N lik bir alanda 8 yöne hareket, N*N-1 adet hareket, minimum açıda dönüşle maksimum alanı gez
- Hareketler arası açı miktarı az olsun, yumuşak dönüşler yapsın.
- İyilik fonksiyonu 2 bileşene sahip: min(açı) ve max(alan)
- Temsil: (N*N-1) adet 0-8 arası yönleri belirten sayılar

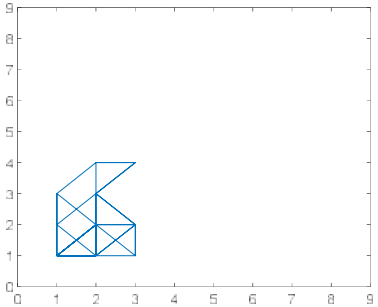
Tepe Tırmanma ile
tepe_tirmanma_tarama_1robot.m

- Rasgele bir çözümle başla
- G kez:
  - Çözümden (%mu kadar) rasgele değişiklerle P adet yeni çözüm üret
  - Üretilen çözümlerin iyilik değerlerini hesapla
    - Dönüş açılarını topla, gidilen farklı nokta sayısını topla
  - Üretilenlerden en iyisi mevcuttan iyi ise çözüme ata, mu oranını azalt, kötü ise mu oranını arttır
  - tk_max kez daha iyisi üretilmediyse çözümü rasgele başlat

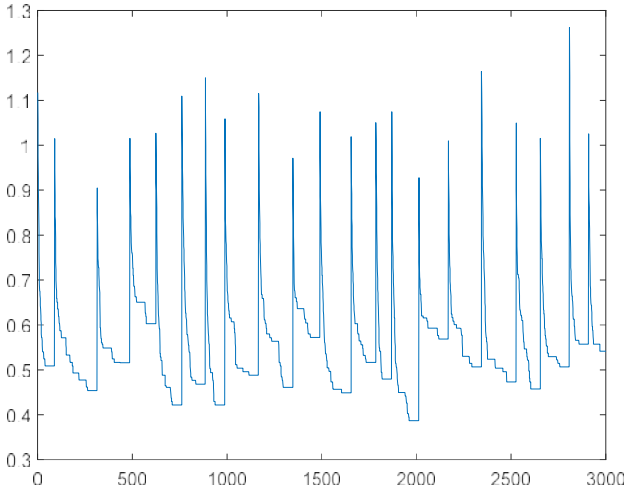
Bulunan çözümlerden biri



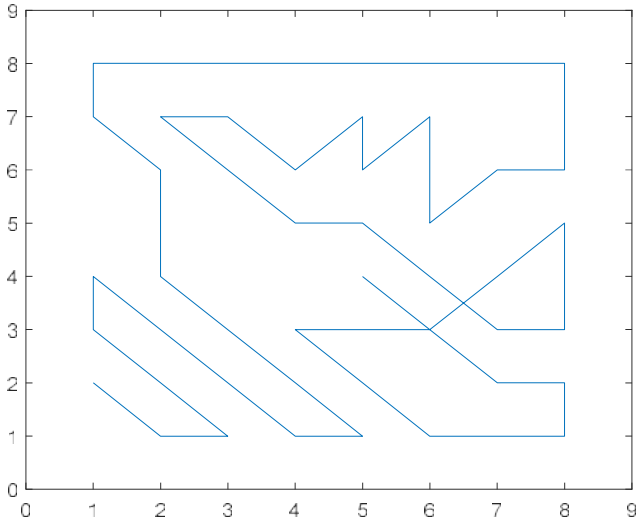
Rasgele bir başlangıç



Sürecin ilerleyişi



Bulunan en iyi çözüm



P=200
mu=0.01; % degisim orani
mu_dec=0.99; % azalma orani
mu_inc=1.01; % artma orani
tk_max=50; % tk_max kez ayni kaldıysa restart
G=3000; % iterasyon sayisi

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

Uygulama: labirentten çıkış

paralel_tepe_tirmanma_labirent.m

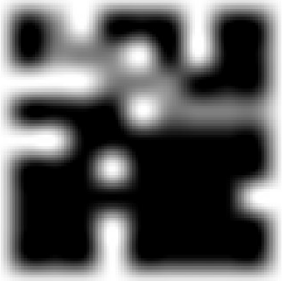
- İçinde engeller olan bir ortam var. En sağ bütün hücreler çıkış
- Her çözüm: P adet (pattern (K*K), hareket) ikilisi
- PS adet rasgele çözümle başla
- GS kez:
  - Çözümlerin iyilik değerlerini hesapla
    - (pattern, hareket) ikililerıyle HS kez hareket et. Pattern'lerinden hangisi bulunduğun yere en çok benziyorsa onun hareketi yap. Son vardığın noktanın en sağa yakınlığını hesapla
  - Çözümlerin en iyilerinin bir kısmını tut (best_keep). Diğerlerini mutasyonla değiştir.

Bir Çözüm

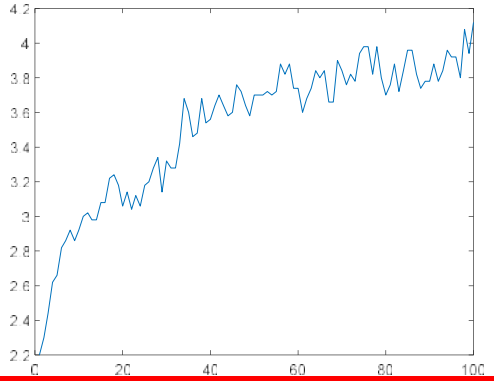
```
K=3; % pattern buyuklugu
P=4; % pattern sayisi
PS=50; % populasyon buyuklugu
GS=100; % generation sayisi
HS=30; % her bireyin hareket sayisi
mut_r= 0.2; % mutasyon orani
best_keep = 0.2;
```

Hareketler

```
  2
3   1
  4
```



Her generasyonun ortalama iyiliği




```
1 0 0
0 1 0 1
1 1 1

1 1 1
0 1 1 4
1 0 0

0 1 0
0 1 1 1
0 0 0

1 1 1
1 0 0 1
0 0 0
```

Daha kısa çözümler bulmaya nasıl zorlarız?

Bir labirentte öğrendiğini başka bir labirente kullanabilir mi? Hangi durumda kullanabilir?

Zorlayalım ☺

P=4 P=40 P=100

Sorun nerede?

1 0 1
1 0 1
1 0 1

Aşağı mı yukarı mı?

1 1 1
0 0 0
1 1 1

Sola mı sağa mı?

Çözüm nerede?

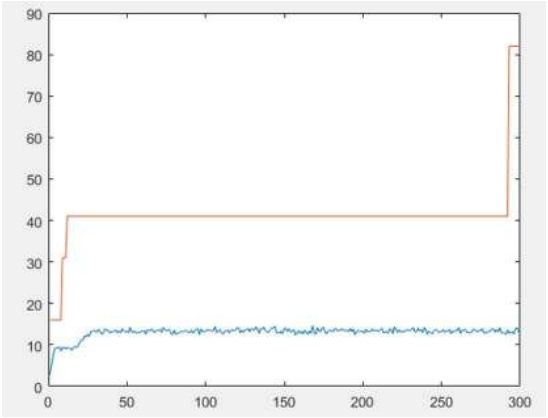
Daha fazla pattern (P) ?
Daha çok çözüm ?
Daha çok adım sayısı?
Mutasyon, best_keep oranlarında değişim?
Paralel tepe tırmanma yerine genetik alg?
Pattern büyüklüğünü (K) arttırmak ?
Tüm alanı mı görmeli?
Kaç pattern gerekecek?
CNN? Küçük pattern'lerin birleşimleriyle büyük pattern
Önceki hareketi de kullanmak

paralel_tepe_tirmanma_labirent_v3.m

- En iyileri tutup, diğerlerini mutasyona uğratmak → En iyilerden mutasyonla yenileri üretmek
- Geçmiş hareketi de kullanmak
- Ödüle gezdiği farklı hücre sayısını eklemek

şimdiki	1	4	2
önceki	1	4	2
	0 1 0	1 1 1	1 0 1
	1 0 0	1 0 1	1 0 1
	1 1 1	1 0 0	1 0 0
şimdiki	3	2	3
önceki	4	4	3
	0 1 0	1 0 1	0 0 0
	0 0 0	1 1 1	1 1 1
	1 1 1	0 1 0	1 0 1

```
K=3; % pattern buyuklugu
P=6; % pattern sayisi
PS=500; % populasyon buyuklugu
GS=300; % generation sayisi
HS=100; % her bireyin hareket sayisi
mut_r= 0.04; % mutasyon orani
best_keep = 0.1;
```



Kaynaklar

- http://aima.cs.berkeley.edu/figures.pdf