PARÇACIK SÜRÜ OPTİMİZASYONU

BMÜ-579 METASEZGİSEL YÖNTEMLER

YRD. DOÇ. DR. İLHAN AYDIN

GENEL BİLGİLER

- 1995 yılında Dr.Eberhart ve Dr.Kennedy tarafından geliştirilmiş popülasyon temelli sezgisel bir optimizasyon tekniğidir.
- PSO'nun temelinin sosyolojik esinlemeli olduğu söylenebilir.
- Çünkü algoritmanın orijinal fikri, kuşların sürü halinde toplanmasıyla ilişkilendirilmiş sosyolojik davranışlarına dayanır.
- Kuş, balık ve hayvan sürülerinin bir "bilgi paylaşma" yaklaşımı uygulayarak çevrelerine adapte olabilme, zengin yiyecek kaynağı bulabilme ve avcılardan kaçabilme yeteneklerinden esinlenmiştir.

GENEL BİLGİLER

- PSO, optimum ya da optimuma yakın çözüm bulmak için önce her biri aday çözümü sunan bireyler (parçacıklar) oluşturur.
- Bu bireylerin oluşturulması gelişigüzel, düzenli ya da her iki şekilde yapılabilir.
- Bireylerin bir araya gelmesinden çözüm için gerçekleştirilen popülasyonumuz (sürü) meydana gelir.
- Pratikte, 2 ve 100 arası boyuttaki çoğu gerçek problemler için 20 parçacıklı bir sürü oldukça iyi çalışmaktadır.
- Uyarlamalı sürü boyutu da kullanılabilir.
- PSO, bireyler arasındaki bilginin paylaşımını esas alır. Her bir parçacık kendi pozisyonunu sürüdeki en iyi pozisyona doğru ayarlarken, bir önceki tecrübesinden de yararlanır.

GENEL BİLGİLER

- PSO, fonksiyon optimizasyonu, çizelgeleme, yapay sinir ağlarının eğitimi, bulanık mantık sistemleri, görüntü işleme vb. pek çok alanda yaygın olarak kullanılmaktadır.

- Belirli bir alanda, sadece bir bölgede yiyecek olduğu, bir kuş grubunun bu alanda yiyecek aradığı ve başlangıçta yiyeceğin nerede olduğunu bilmediği kabul edilir ise, yiyecek bulmak için en iyi çözüm ne olabilir?

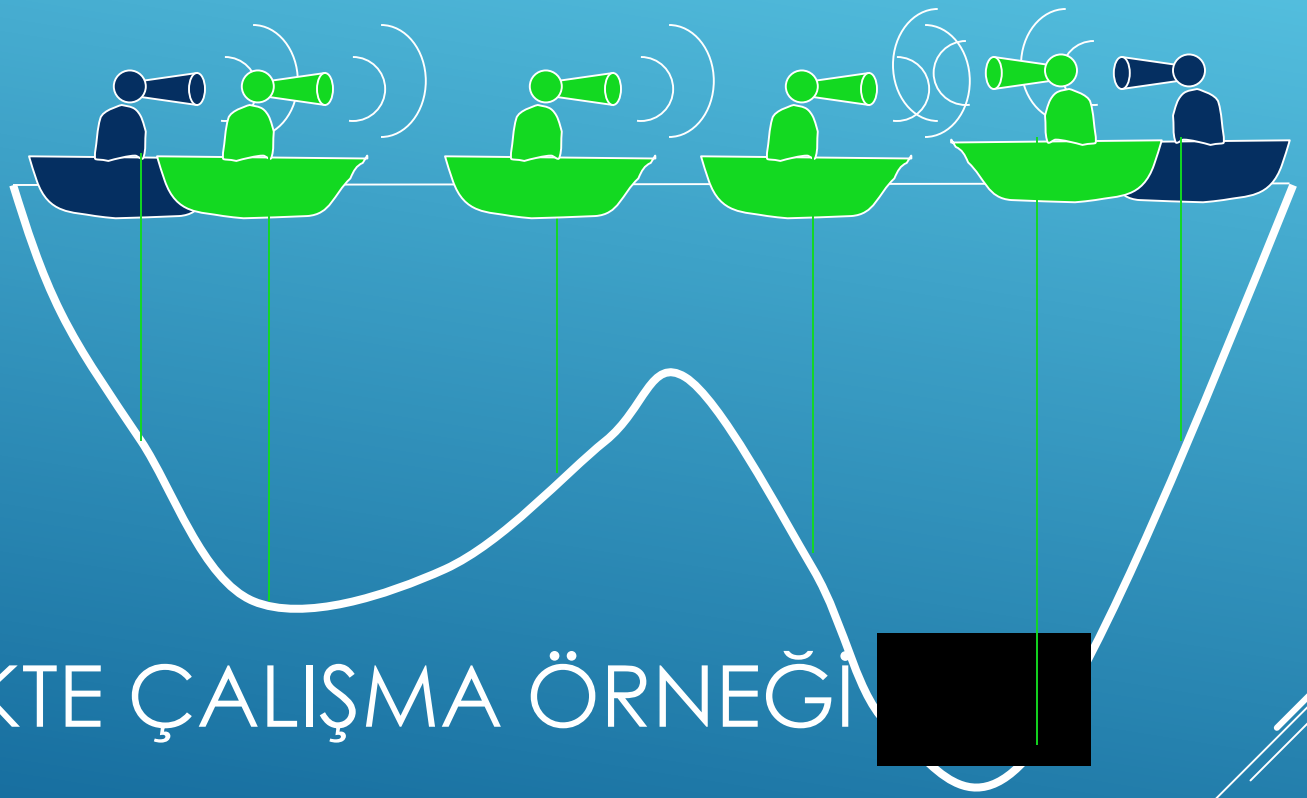
ALGORİTMA

ALGORİTMA

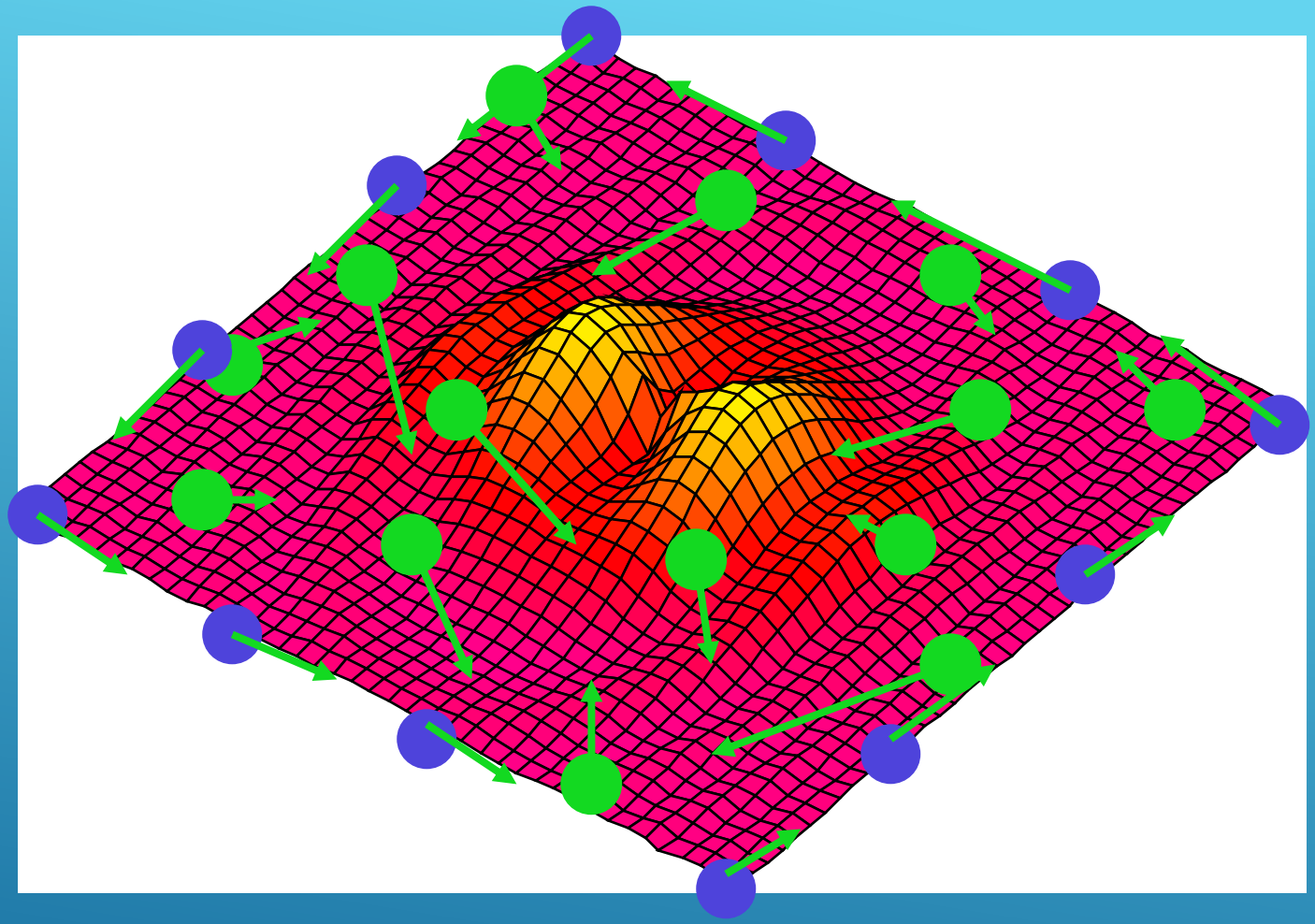
- PSO'da her bir parçacık bir kuşu ifade eder ve her parçacık bir çözüm sunar.
- Tüm parçacıkların uygunluk fonksiyonu ile bulanan uygunluk değerleri vardır.
- Parçacıklar, kuşların uçuşlarını yönlendiren hız bilgisine benzer bir bilgiye sahiptir.

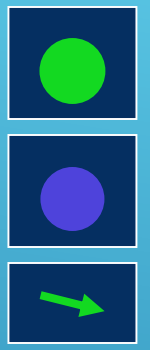
ALGORİTMA

- PSO rastgele üretilmiş belirli sayıda çözümle (parçacıkla) başlatılır ve parçacıklar güncellenerek en uygun çözüm değeri araştırılır.
- Parçacıkların her biri, parçacığın en iyi kendi çözümü (pbest) ve tüm parçacıkların en iyi çözümü (gbest) kullanılarak güncellenir. Bu değerler hafızada saklanır.

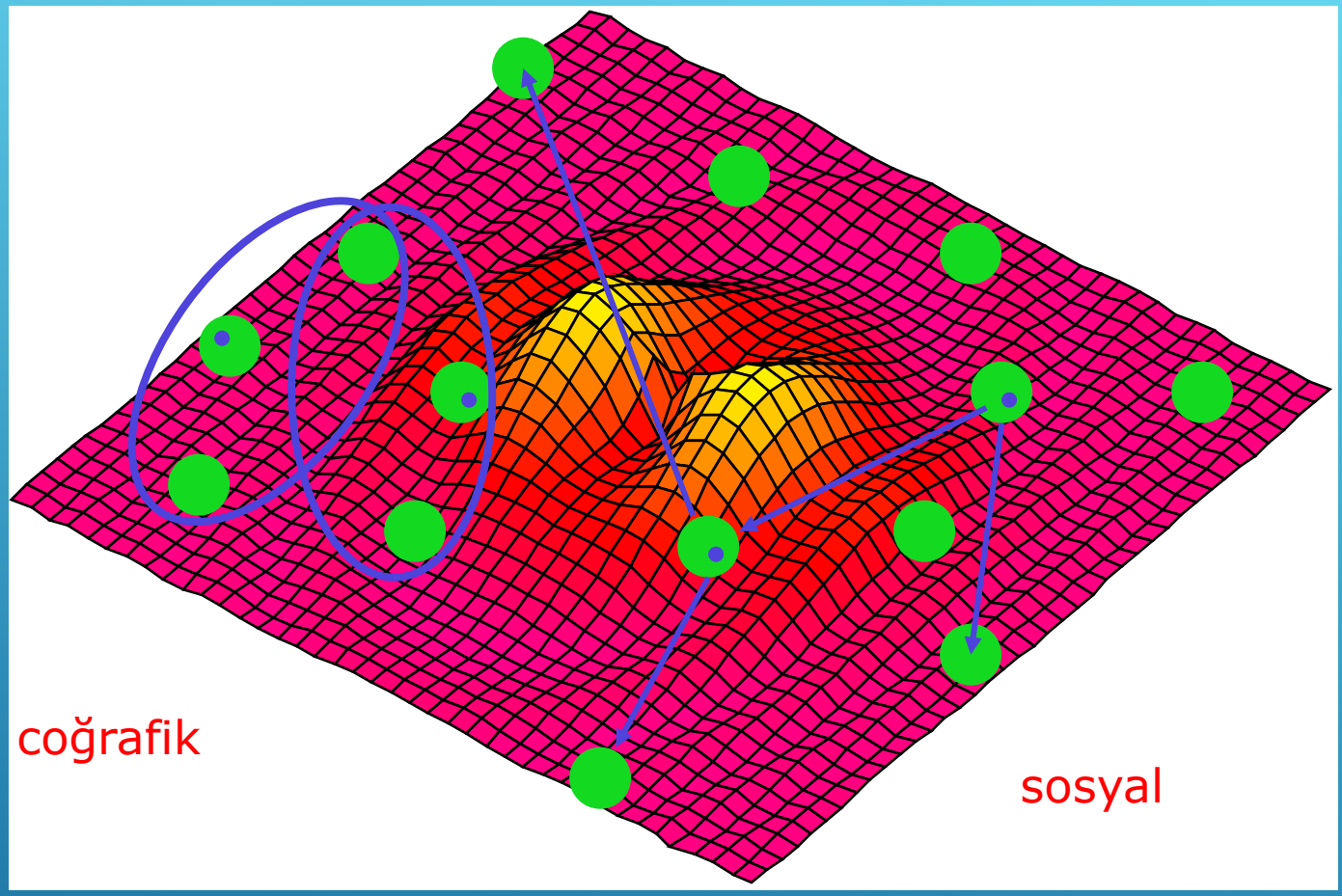


BİRLİKTE ÇALIŞMA ÖRNEĞİ



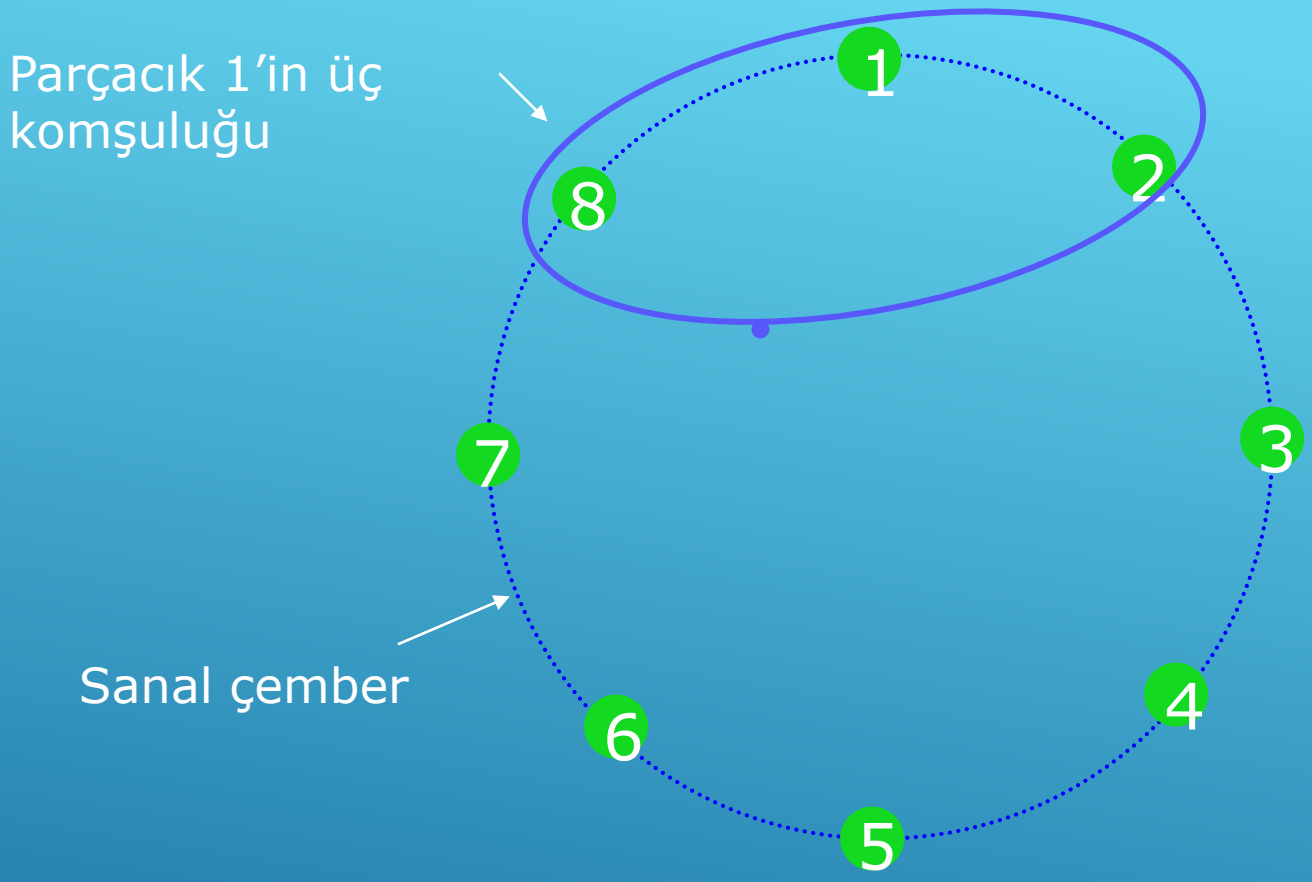


HIZ VE POZİSYONLARIN BAŞLATILMASI

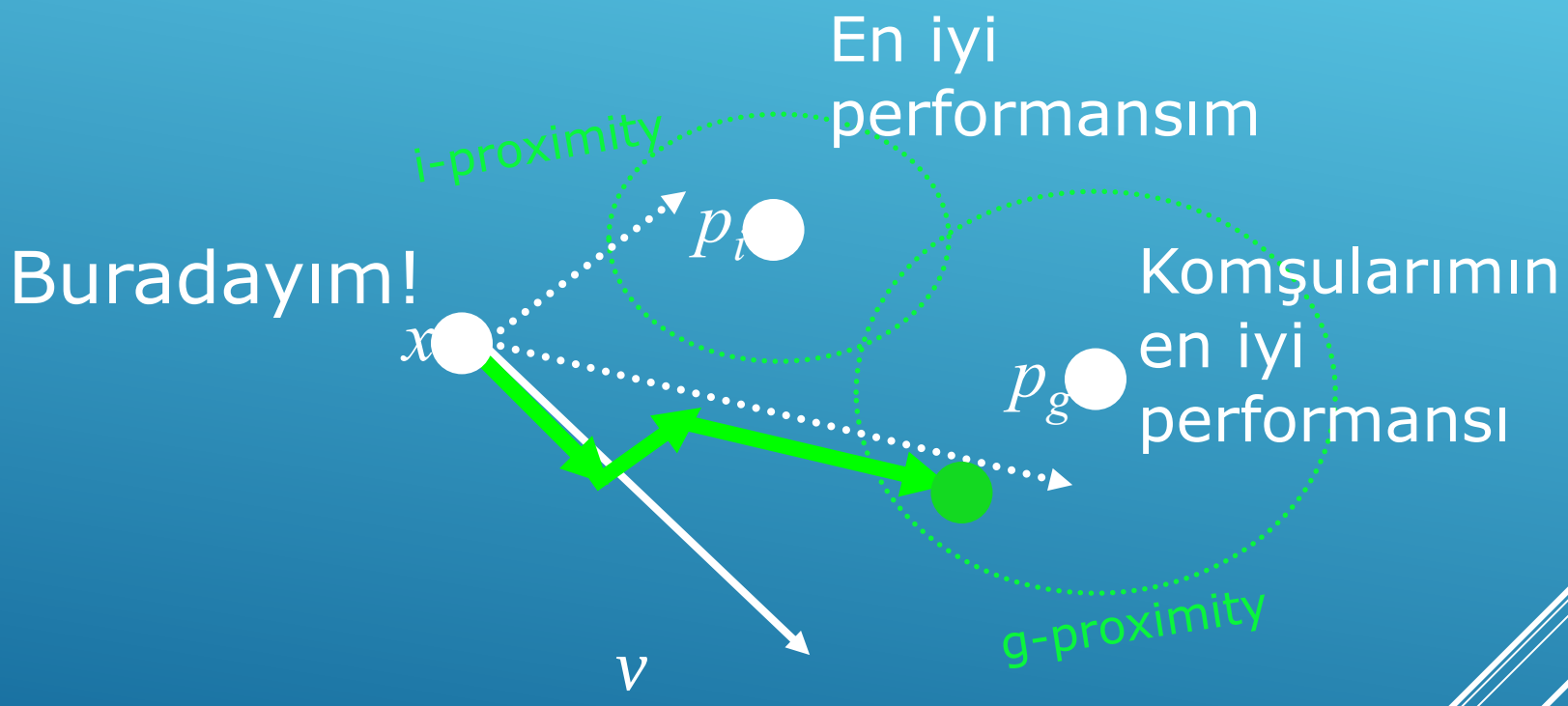


KOMŞULUK

DAİRESEL KOMŞULUK



PSİKOLOJİK UZLAŞMA



s sürünün boyutu olsun. Her i parçacığı, birkaç karakteristiğe sahip bir nesne olarak temsil edilebilir.
Bu karakteristikler aşağıdaki sembollere gösterilir:
xi: Parçacığın mevcut pozisyonu;
vi: Parçacığın mevcut hızı;
yi: Parçacığın kişisel en iyi pozisyonu.

- i parçacığıyla ilişkilendirilmiş kişisel en iyi pozisyon, parçacığın ziyaret ettiği (bir önceki xi değeri) ve bu parçacık için en yüksek uygunluk değerini veren en iyi pozisyondur.
- Bir minimizasyon işi için daha düşük bir fonksiyon değeri sağlayan bir pozisyon daha yüksek uygunluğa sahip kabul edilir.
- f sembolü minimize edilen amaç fonksiyonunu göstermek üzere kişisel en iyi pozisyon için güncelleme denklemi t zaman aralığına bağlı olarak aşağıda gösterilmiştir.

$$y_i(t+1) = \begin{cases} y_i(t) & , f(x_i(t+1)) \geq f(y_i(t)) \\ x_i(t+1), & f(x_i(t+1)) < f(y_i(t)) \end{cases}$$

- PSO'nun global ve lokal adlı iki versiyonu bulunmaktadır.
- İki algoritma arasındaki fark verilen bir parçacığın direkt olarak etkileşim içinde olduğu parçacıkların kümesine bağlıdır ve $\hat{y}$ sembolü, bu etkileşimi temsil için kullanılacaktır.
- Global modelde kullanılan $\hat{y}$ 'nin tanımı aşağıdaki eşitlikte sunulmuştur.

$$\hat{y}(t) \in \{y_0(t), y_1(t), \ldots, y_s(t)\} \mid f(\hat{y}(t)) = \min\{f(y_0(t)), f(y_1(t)), \ldots, f(y_s(t))\}$$

- Bu tanım, yˆ ’nin herhangi bir parçacık tarafından şimdiye kadar keşfedilen en iyi pozisyon olduğunu belirtir.
- Algoritma iki bağımsız gelişigüzel diziyi kullanır, r1 ~ U(0, 1) ve r2 ~ U(0, 1).
- Bu diziler aşağıdaki denklemde gösterildiği gibi algoritmanın stokastik doğasını etkilemek için kullanılır.
- Burada c1 ve c2 katsayıları öğrenme faktörleridir ve hızlanma katsayıları olarak da adlandırılır.
- Bu katsayılar, bir iterasyonda bir parçacığın alabileceği adımın maksimum boyutunu etkiler ve her parçacığı kişisel en iyi ve global en iyi pozisyonlarına doğru çeken, stokastik hızlanmayı ifade eder.
- Düşük değerlerin seçilmesi parçacıkların hedef bölgeye doğru çekilmeden önce, bu bölgeden uzak yerlerde dolaşmalarına imkân verir.
- Ancak hedefe ulaşma süresi uzayabilir.
- Diğer yandan, yüksek değerlerin seçilmesi, hedefe ulaşmayı hızlandırırken, beklenmedik hareketlerin oluşmasına ve hedef bölgeye ulaşılmamasına sebep olabilir.

$$v_{i,j}(t+1) = v_{i,j}(t) + c_1 r_{1,j}(t)\left[y_{i,j}(t) - x_{i,j}(t)\right] + c_2 r_{2,j}(t)\left[\hat{y}_j(t) - x_{i,j}(t)\right]$$

- c1 ve c2'nin değerleri $0 < c1, c2 \leq 2$ sabitleriyle sınırlanır.
- Ancak bilişsel katsayının olarak da isimlendirilen c1'in biraz daha büyük seçilmesi ve $c1 + c2 = 4$ durumunun sağlanması halinde daha iyi sonuçların alınabileceği gösterilmiştir.
- Buradan c1'in bu parçacığın kişisel en iyi pozisyonunun yönünde adım büyüklüğünü ayarladığı, c2'nin de global en iyi parçacığın yönünde maksimum adım büyüklüğünü ayarladığı açıktır.
- vi,j değeri, parçacığın arama uzayından ayrılma olasılığını azaltmak için [-vmaks, vmaks] değerine sıkıştırılmıştır.
- Eğer arama uzayı [-xmaks, xmaks] sınırları ile tanımlanmışsa, vmaks'ın değeri tipik olarak $vmaks = k \times xmaks$ değerine ayarlanır ve $0.1 \leq k \leq 1.0$ olur.
- Ayrıca vmaks arama uzayı büyüklüğünün yarısına eşitlenebilir

$$v_{i,j}(t+1) = v_{i,j}(t) + c_1 r_{1,j}(t)\left[y_{i,j}(t) - x_{i,j}(t)\right] + c_2 r_{2,j}(t)\left[\hat{y}_j(t) - x_{i,j}(t)\right]$$

Her parçacığın pozisyonu, bu parçacık için yeni hız vektörü kullanılarak

x (t +1) = x (t)+ v (t +1) şeklinde güncellenir.

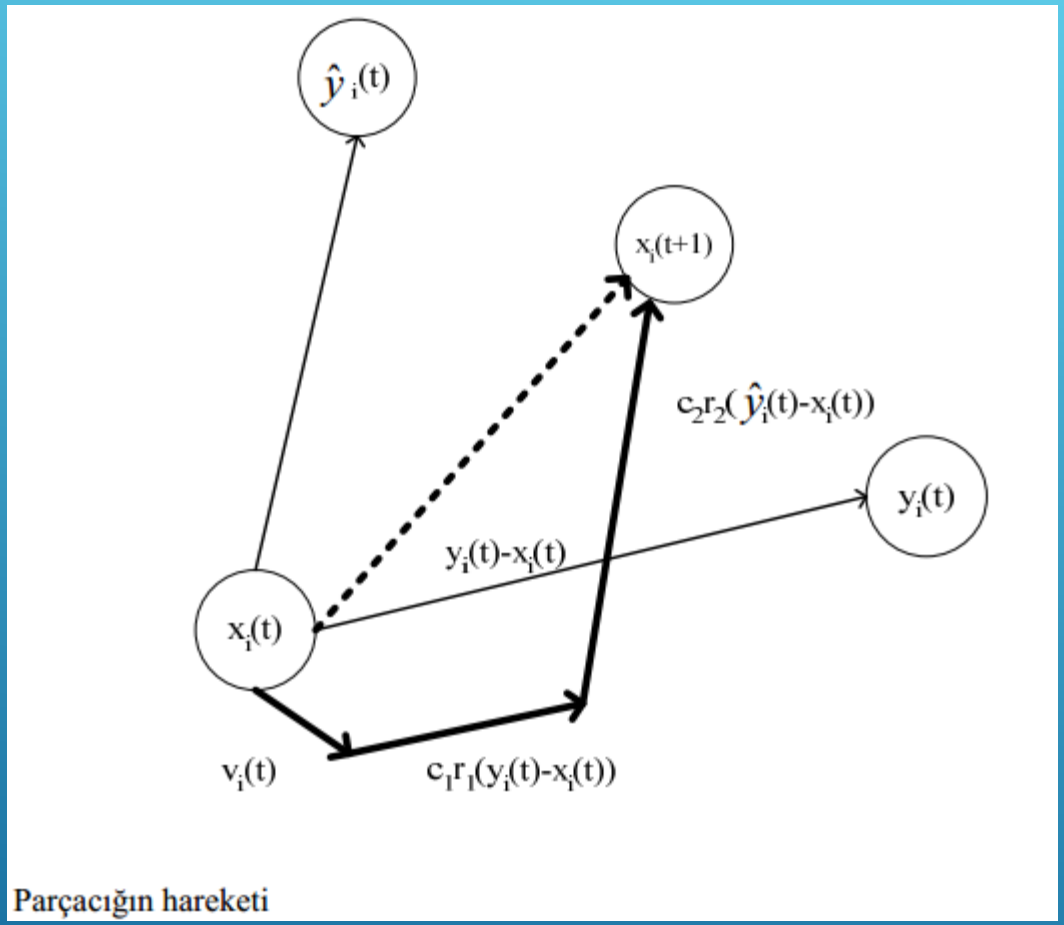
xi,n(t)∈[ln, un] ve 1 ≤ n ≤N olmak üzere ln ve un n. boyut için alt ve üst sınırları göstermektedir.
PSO algoritmasının çalışması esnasında, boyutlar için alt sınır ya da üst sınırın aşılması durumunda bir düzeltme işlemi uygulanması gerekmektedir.
Bu işlem alt sınır ve üst sınır aşılması durumunda sırasıyla

$$x_i = x_i + \alpha \times r \times (u_n(x_i) - l_n(x_i))$$

$$x_i = x_i - \alpha \times r \times (u_n(x_i) - l_n(x_i))$$

Burada $\alpha \in [0, 1]$ aralığında kullanıcı tanımlı bir değerdir ve r ~ U(0, 1) olarak seçilir

Bir parçacığın hız denklemindeki üç terime bağlı olarak ve pozisyon güncelleme denklemine göre hareketi aşağıdaki gibi gösterilmiştir.



Parçacığın hareketi

$x_{i,j}$ ve $v_{i,j}$'ye gelişigüzel başlangıç değerleri vererek sürüyü oluştur

Do

For $i = 1$ **to** Parçacık sayısı

if $f(x_i) < f(y_i)$ **then** $y_i = x_i$ // lokal (kişisel) en iyiyi güncelle

$\hat{y}_i = \min(x_{komşular})$ // global en iyiyi güncelle

For $j = 1$ **to** Optimize edilen boyut sayısı

$v_{i,j} = v_{i,j} + c_1 r_{1,j}\left[y_{i,j} - x_{i,j}\right] + c_2 r_{2,j}\left[\hat{y}_j - x_{i,j}\right]$ // hız vektörünü güncelle

$x_{i,j} = x_{i,j} + v_{i,j}$ // pozisyon vektörünü güncelle

Next j

Next i

Until Sonlandırma kriteri

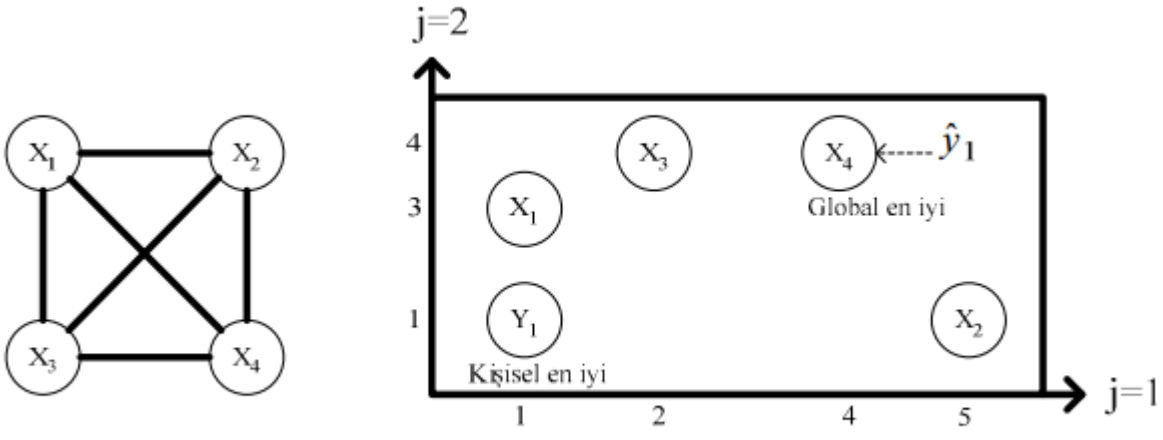
PSO ALGORİTMASI

Algoritmanın ilk adımında ifade edilen başlatma, aşağıda verilen adımlardan oluşur:

1. Her $x_{i,j}$ koordinatını tüm $i \in 1 \ldots s$ ve $j \in 1 \ldots n$ için [-xmaks, xmaks] aralığında düzenli rassal dağılımdan türetilmiş bir değere ata (Parçacıkların başlangıç pozisyonları arama uzayı boyunca dağıtılır).
2. Her $v_{i,j}$'yi $i \in 1 \ldots s$ ve $j \in 1 \ldots n$ için [-vmaks, vmaks] aralığında düzenli rassal dağılımdan türetilmiş bir değere ata (Alternatif olarak parçacıkların hızları 0'a atanabilir çünkü başlangıç pozisyonları zaten gelişigüzel atanmıştır).
3. $y_i = x_i \; \forall \; i \in 1 \ldots s$ olarak ata (Alternatif olarak her parçacık için iki rassal vektör üretilebilir ve en uygun vektör y_i'ye daha az uygun olanı da x_i'ye atanabilir.

Bu, ek fonksiyon değerlendirmesi gerektirir, bu yüzden genellikle ilk önce açıklanan daha basit yöntem kullanılır).

- Sadece iki değişkenli basit bir problem ele alınırsa xi parçacığı iki reel sayıdan oluşan bir vektör olarak temsil edilir. Yani Xi = <xi1, xi2> olur.
- Şekilde, dört parçacıktan oluşan sürünün global komşuluk yapısı ve arama uzayında parçacıkların pozisyonları görülmektedir.
- Pozisyonlar için x ekseni birinci boyut (j=1), y ekseni de ikinci boyut (j=2) olarak seçilmiştir.



Komşuluk yapısı ve arama uzayında parçacıkların *t* anında pozisyonları

ORİJİNAL PSO ALGORİTMASINDA BİR PARÇACIĞIN HAREKETİNİN SAYISAL ÖRNEĞİ

Sadece birinci parçacık (X1) göz önüne alınıp bir adım sonra nasıl hareket edeceğine bakılırsa parçacığın hızı ve bir sonraki pozisyona gelecektir. c1 = c2 = 2 seçildiği varsayılırsa;

j = 1 için v1,1(t) = 1; r1,1(t) = 0.4; r2,1(t) = 0.1
j = 2 için v1,2(t) = 2; r1,2(t) = 0.25; r2,2(t) = 0.25

j=1 boyutunda

$$v_{1,1}(t+1) = 1 + 2 \times 0.4 \times [1-1] + 2 \times 0.1 \times [4-1] = 1.6$$

$$x_{1,1}(t+1) = 1 + 1.6 = 2.6$$

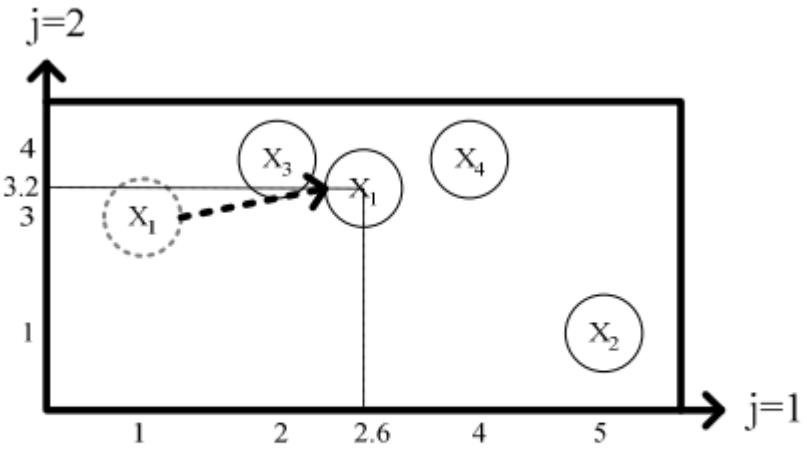
j=2 boyutunda

$$v_{1,2}(t+1) = 2 + 2 \times 0.25 \times [1-3] + 2 \times 0.1 \times [4-3] = 0.2$$

$$x_{1,2}(t+1) = 3 + 0.2 = 3.2$$

ORİJİNAL PSO ALGORİTMASINDA BİR PARÇACIĞIN HAREKETİNİN SAYISAL ÖRNEĞİ

- Böylece X1 parçacığı PSO güncelleme denklemlerinden sonra yeni (2.6, 3.2) pozisyonuna hareket edecektir.
- Bu durumda X1'in yeni pozisyonu Şekil 2.6'da gösterildiği gibi olur.
- Bazı boyutlarda sınır aşma problemi ile karşılaşılınca X1'in hızı için bazı sınırlamalar getirilebilir ve boyut sınırlarında kalması sağlanabilir.



t+1 anında X_1 parçacığının pozisyonunun güncellenmesi

ORİJİNAL PSO ALGORİTMASINDA BİR PARÇACIĞIN HAREKETİNİN SAYISAL ÖRNEĞİ

PSO PARAMETRELERİ

- Parçacık sayısı: genelde 20-40 arasındadır.
- Parçacık boyutu: probleme göre değişkenlik gösterir.
- V_{max}= bir iterasyonda, bir parçacıkta meydana gelebilecek maksimum değişikliği (hız) belirler.
- Parçacık aralığı: örneğin x1 için(-5,5)
- Öğrenme faktörleri: c_1, c_2 olup [0,4] aralığında seçilebilir.
- Sonlandırma kriteri: minimum hata veya iterasyon sayısı olabilir.

SAYISAL ÖRNEK

- İkisi küresel olmak üzere toplamda 6 yerel minimuma sahip bir fonksiyon

$f(x)=4x_1^2 - 2{,}1x_1^4 + 1/3\,(x_1^6) + x_1x_2 - 4x_2^2 + 4x_2^4$

Fonksiyonun minimum olduğu nokta araştırılacaktır.

SAYISAL ÖRNEK

- Boyut=D=2 (x_1,x_2)
- Parçacık sayısı=10
- Öğrenme faktörü=c_1,c_2=2
- Parçacıklar parametre aralıklarına uygun olarak seçilerek, her parçacık için uygunluk değeri hesaplanır.
- İlk iterasyon için her parçacığın pbest değeri kendisine eşittir. Başlangıç hızı 0 olarak alınmıştır.

SAYISAL ÖRNEK

Parçacık	$x_{1,1}^{0}$	$x_{1,2}^{0}$	Uygunluk Değeri	pbest
P1	-3,7060195	2,81307351	707,5444175	$p1^{1}$
P2	1,30101908	-1,6908636	24,47949005	$P2^{1}$
P3	3,5723814	-3,3609453	156,3683986	$P3^{1}$
P4	1,2363086	2,8388213	258,3875547	$P4^{1}$
P5	-2,0099463	1,61072044	-10,58678451	$P5^{1}$
P6	-2,6013287	0,75244956	-96,83639806	$P6^{1}$
P7	-1,5921306	2,24163608	83,9370077	$P7^{1}$
P8	-4,9807337	-4,3846604	207,8923222	$P8^{1}$
P9	-3,5026999	-2,6985498	-94,5324658	$P9^{1}$
P10	-3,3496718	4,677583	1634,851374	$P10^{1}$
			gbest	**p5**

$$pbest_1^1 = [-3.7060195, 2.81307351]$$

$$gbest^1 = \overrightarrow{P_5^1} = [-2.0099463, 1.61072044]$$

$$v_i^{k+1} = v_i^k + c_1 rand_1^k \left(pbest_i^k - x_i^k\right) + c_2 rand_2^k \left(gbest^k - x_i^k\right)$$

$$v_{1,1}^1 = 0 + c_1 rand_1^1 \left(pbest_{1,1}^1 - x_{1,1}^1\right) + c_2 rand_2^1 \left(gbest^1 - x_{1,1}^1\right)$$

$$v_{1,1}^1 = 0 + 2x0.4x\left(-3.7060195 - (-3.7060195)\right) + 2x0.6x\left(-2.0099463 - (-3.7060195)\right)$$

$$v_{1,1}^1 = 2,03528784$$

$$v_{1,2}^1 = 0 + c_1 rand_1^1 \left(pbest_{1,2}^1 - x_{1,2}^1\right) + c_2 rand_2^1 \left(gbest^1 - x_{1,2}^1\right)$$

$$v_{1,2}^1 = 0 + 2x0.4x\left(2.81307351 - 2.81307351\right) + 2x0.6x\left(1.61072044 - 2.81307351\right)$$

$$v_{1,2}^1 = -1,442823684$$

$v_1^1 = [2,03528784, -1,442823684]$ olarak bulunur.

SAYISAL ÖRNEK

Birinci iterasyon için hesaplanan hız değerleri

Parçacık	$v_{1,1}^{1}$	$v_{1,2}^{1}$	rand1	rand2
P1	2,035288	-1,44282	0,4	0,6
P2	-5,29754	5,282534	0,3	0,8
P3	-8,57238	8,360945	0,1	0,9
P4	-5,19401	-1,96496	0,7	0,8
P5	0	0	0,5	0,1
P6	0,473106	0,686617	0,4	0,4
P7	-0,25069	-0,37855	0,9	0,3
P8	0	0	0,3	0
P9	0,597101	1,723708	0,1	0,2
P10	1,339726	-3,06686	0,6	0,5

SAYISAL ÖRNEK

Parçacıkların yeni konumları;

$x_1^{k+1} = x_1^k + v_1^{k+1}$

$x_{1,1}^1 = x_{1,1}^0 + v_{1,1}^1$;

$x_{1,1}^1 = (-3,7060195) + 2,035288 = -1,6707315$

$x_{1,2}^1 = x_{1,2}^0 + v_{1,2}^1$;

$x_{1,2}^1 = 2,81307351 + (-1,44282) = 1,37025351$

$x_1^1 = [-1,6707315,\ 1.37025351]$

SAYISAL ÖRNEK

Birinci iterasyon için hesaplanan konum değerleri

Parçacık	$X_{1,1}^{0}$	$v_{1,1}^{1}$	$X_{1,2}^{0}$	$v_{1,2}^{1}$	$X_{1,1}^{1}$	$X_{1,2}^{1}$
P1	-3,70602	2,035288	2,813074	-1,44282	**-1,67073**	**1,37025**
P2	1,301019	-5,29754	-1,69086	5,282534	**-3,99653**	**3,591671**
P3	3,572381	-8,57238	-3,36095	8,360945	**-5**	**5**
P4	1,236309	-5,19401	2,838821	-1,96496	**-3,9577**	**0,87386**
P5	-2,00995	0	1,61072	0	**-2,00995**	**1,61072**
P6	-2,60133	0,473106	0,75245	0,686617	**-2,12822**	**1,439066**
P7	-1,59213	-0,25069	2,241636	-0,37855	**-1,84282**	**1,863087**
P8	-4,98073	0	-4,38466	0	**-4,98073**	**-4,38466**
P9	-3,5027	0,597101	-2,69855	1,723708	**-2,9056**	**-0,97484**
P10	-3,34967	1,339726	4,677583	-3,06686	**-2,00995**	**1,61072**

SAYISAL ÖRNEK