**Parçacık Sürüsü Optimizasyon (*Particle Swarm Optimization* – PSO) algoritmasının iki temel denklemi:**

Hız (*velocity*) Güncellemesi:



Konum (*position*) Güncellemesi:



**Tanımlar** (implementasyon esnasında kolaylık olması açısından değişken isimlerini de aşağıdaki gibi tanımlayarak kullanmanız hararetle önerilir):

*positioni*  : *i*'inci parçacığın mevcut konumu

*positioni* = (*xi*, *yi*)

*velocityi* : *i*'inci parçacığın mevcut hız vektörü

*velocityi* = (v\_*xi*, v\_*yi*)

*pbesti* : *i*'inci parçacığın o ana kadar deneyimlediği en iyi konum

*pbesti* = (*xpbest\_i*, *ypbest\_i*)

*val\_pbesti* : *i*'inci parçacığın o ana kadar deneyimlediği en iyi konumda fonksiyonun aldığı değer (karşılaştırmalarda kullanılacak)

*gbest*: sürününo ana kadar deneyimlediği en iyi konum

*gbest*= (*xgbest*, *ygbest*)

*val\_gbest*: sürününo ana kadar deneyimlediği en iyi konumda fonksiyonun aldığı değer (karşılaştırmalarda kullanılacak)

Dolayısıyla yukarıdaki hız ve konum güncelleme denklemleri, açık formda şu şekilde yazılabilir:









Yukarıdaki denklemlerde:

*w* : atalet faktörü (*inertia factor*) olup; iterasyonlar boyunca 0.95'ten 0.4'e lineer olarak azaltılacaktır.

*c*1 : bilişsel davranış katsayısı (*cognitive behavior coefficient*) olup 1.494 olarak alınabilir.

*c*2 : sosyal davranış katsayısı (*social behavior coefficient*) olup 1.494 olarak alınabilir.

*r*1 : 0 ile 1 arasında rastgele bir sayıdır.

*r*2 : 0 ile 1 arasında rastgele bir sayıdır.

Δ*t* : zaman adımı olup 1 olarak seçilebilir.

Algoritma için gerekli veri yapıları ve parametreler:

* *number\_of\_particles* : sürüdeki toplam parçacık sayısı
* *number\_of\_iterations* : toplam iterasyon sayısı
* *w* : atalet faktörü
* *c*1 : bilişsel davranış katsayısı
* *c*2 : sosyal davranış katsayısı
* *r*1 : 0 ile 1 arasında rastgele sayı, her kullanıldığında yeni değer üretilecek
* *r*2 : 0 ile 1 arasında rastgele sayı, her kullanıldığında yeni değer üretilecek
* Δ*t* : zaman adımı, 1 olarak seçilebilir
* *positions\_matrix* : tüm parçacıkların konum değerlerinin (yani *xi* ve *yi* değerlerinin) tutulduğu 2 boyutlu bir array veya list
* *velocities\_matrix* : tüm parçacıkların konum değerlerinin (yani *v\_xi* ve *v\_yi* değerlerinin) tutulduğu 2 boyutlu bir array veya list
* *current\_values\_matrix* : parçacıkların her birinin mevcut konumunda fonksiyonun aldığı değerlerin tutulduğu 1 boyutlu bir array veya list
* *personel\_bests\_matrix* : tüm parçacıkların bireysel en iyi konum değerlerinin (yani *xpbest\_i* ve *ypbest\_i* değerlerinin) tutulduğu 2 boyutlu bir array veya list
* *personel\_best\_values\_array* : tüm parçacıkların her birinin bireysel en iyi konum değerinde fonksiyonun aldığı değerin (yani *val\_pbesti* değerlerinin) tutulduğu 1 boyutlu bir array veya list
* *gbest*'in, yani sürününo ana kadar deneyimlediği en iyi konumun değerleri : *gbest*= (*xgbest*, *ygbest*)
* *val\_gbest*: sürününo ana kadar deneyimlediği en iyi konumda fonksiyonun aldığı değer

**Algoritmanın Sözde Kodu:**

def ParcaciklariIlklendir (alt\_sinir\_x, ust\_sinir\_x, alt\_sinir\_y, ust\_sinir\_y, alt\_sinir\_v\_x, ust\_sinir\_v\_x, alt\_sinir\_v\_y, ust\_sinir\_v\_y,):

# tüm parçacıkları, pozisyonları ve hız bileşenleri belirtilen sınırlar içinde kalacak şekilde rastgele

olarak ilklendir.

def BirParcacigiGuncelle (i, w, c1, c2, delta\_t):

# bu fonksiyonda, ilgili (yani i'inci) parçacığın hızı ve konumu güncellenecek (yukarıdaki

denklemlerle)

# mevcut konumda fonksiyonun aldığı değer hesaplanacak (buna *value* diyelim)

# *value* değeri *personel\_best\_values\_array* içerisinde bulunan *val\_pbesti* ile karşılaştırılacak; gerekli

durumda *val\_pbesti =value* yapılarakgüncellenecek

# *value* değeri *gbest* ile de karşılaştırılacak; gerekli durumda *gbest* *=value* yapılarakgüncellenecek

def TumParcaciklariGuncelle (number\_of\_particles):

# bir for dongüsü içinde tüm parçacıklar için BirParcacigiGuncelle(...) fonksiyonunu çağır

def AmacFonksiyonuHesapla(number\_of\_particles):

# bir for döngüsü içerisinde

# tüm parçacıkların mevcut konumlarında, minimize edilecek amaç fonksiyonunun değerini hesapla

# minimize edilecek amaç fonksiyonu *x*2 + *y*2 olarak seçilebilir. Parçacıkların iterasyonlar sonunda

(0,0) noktasına üşüşüp üşüşmediği kontrol edilebilir.

# minimize edilecek amaç fonksiyonu alternatif olarak (*x* – 5)2 + (*y* – 3)2 olarak seçilebilir.

Parçacıkların iterasyonlar sonunda (5,3) noktasına üşüşüp üşüşmediği kontrol edilebilir.

def TumParcacikKonumlariniCizdir (number\_of\_particles):

# bir scatter plot ile tüm parçacıkların mevcut konumunu çizdir

def PerformParticleSwarmOptimization(number\_of\_particles, number\_of\_iterations, w, c1, c2, delta\_t, alt\_sinir\_x, ust\_sinir\_x, alt\_sinir\_y, ust\_sinir\_y, alt\_sinir\_v\_x, ust\_sinir\_v\_x, alt\_sinir\_v\_y, ust\_sinir\_v\_y)

# Adım 1:

ParcaciklariIlklendir (alt\_sinir\_x, ust\_sinir\_x, alt\_sinir\_y, ust\_sinir\_y, alt\_sinir\_v\_x, ust\_sinir\_v\_x,

alt\_sinir\_v\_y, ust\_sinir\_v\_y)

# Adım 2:

for iter in (0 : number\_of\_iterations)

AmacFonksiyonuHesapla(number\_of\_particles)

TumParcaciklariGuncelle (number\_of\_particles)

TumParcacikKonumlariniCizdir(number\_of\_particles)

return gbest*,* val\_gbest