

PARÇACIK SÜRÜ OPTİMİZASYONU

(Particle Swarm Optimization)

Hazırlayan: Ecrin Nazar AYBAY

21253043

Parçacık Sürü Optimizasyonu Nedir?

Parçacık Sürü Optimizasyonu, **1995 yılında James Kennedy ve Russell Eberhart** tarafından geliştirilmiş bir optimizasyon algoritmasıdır. Algoritma, doğadan esinlenerek geliştirilmiştir ve kuşların sürü davranışlarını veya balık sürülerinin hareketlerini taklit eder. PSO, **popülasyon temelli bir optimizasyon tekniğidir** ve bir hedef fonksiyonun en iyi çözümünü bulmayı amaçlar.

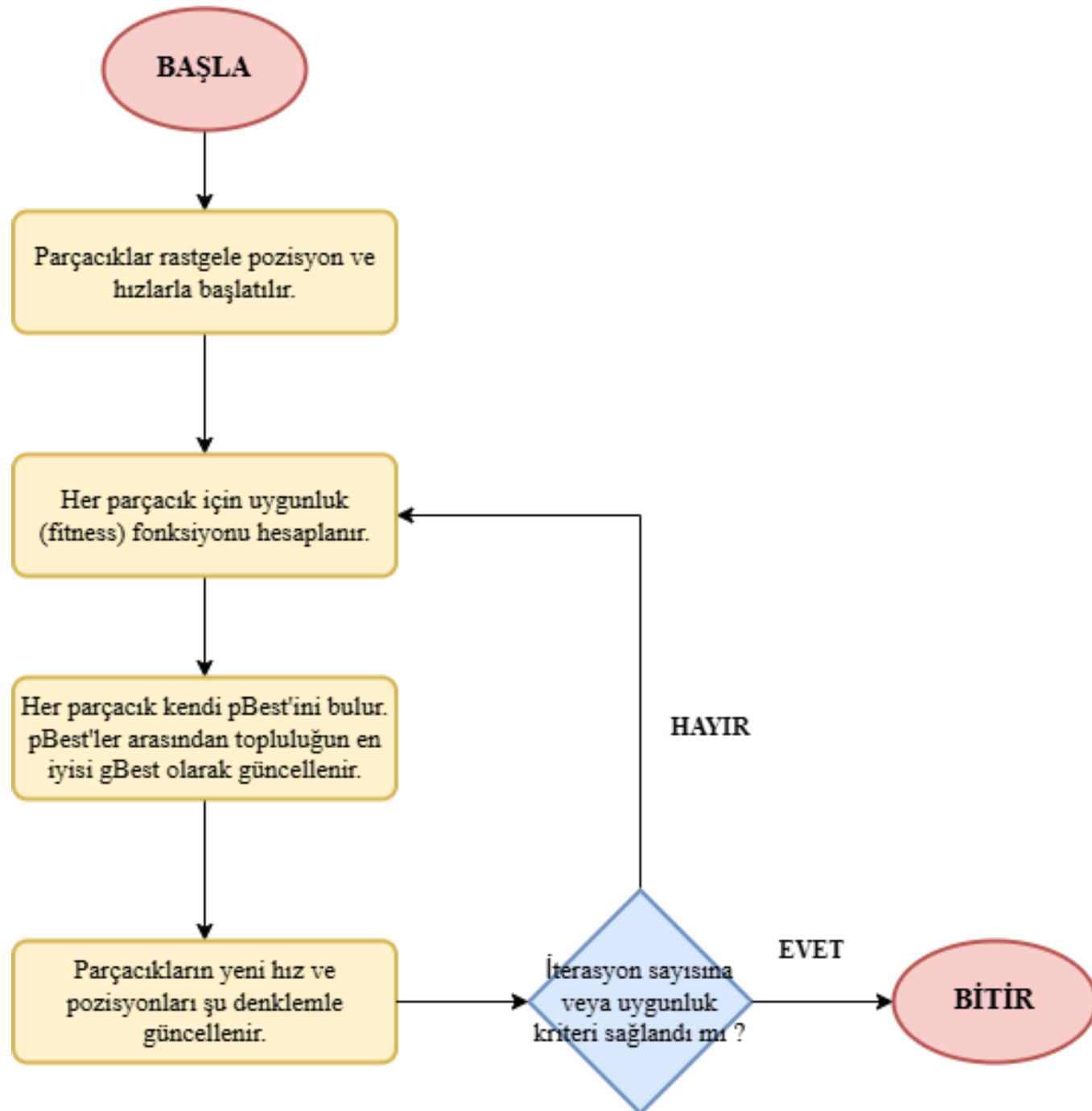
Her bireye parçacık (particle) denir ve parçacıklardan oluşan popülasyona da sürü (swarm) denir. PSO, sürüde bulunan bireylerin pozisyonunun, sürünün en iyi pozisyona sahip olan bireyine yaklaştırılmasına dayanır ve yaklaşma hızı rasgele gelişen durumdur.[1]

[1] Özsağlam, M. Y., & Çunkaş, M. (2008). Optimizasyon problemlerinin çözümü için parçacık sürü optimizasyonu algoritması. *Politeknik Dergisi*, 11(4), 299-305.

PSO'nun Temel Bileşenleri

- 1. Parçacıklar (Particles):** Çözüm uzayında hareket eden bireysel çözüm adaylarıdır. Her parçacığın pozisyonu bir potansiyel çözümü temsil eder.
- 2. Hız (Velocity):** Parçacıkların çözüm uzayındaki yönünü ve büyüklüğünü belirler.
- 3. Pozisyon (Position):** Parçacığın çözüm uzayındaki mevcut yerini ifade eder.
- 4. Fitness Fonksiyonu:** Her parçacığın ne kadar iyi bir çözüm sunduğunu ölçen matematiksel bir fonksiyondur.
- 5. Kişisel En İyi (pBest):** Parçacığın şimdiye kadar ulaştığı en iyi pozisyon.
- 6. Küresel En İyi (gBest):** Tüm parçacıkların şimdiye kadar ulaştığı en iyi pozisyon.

Akış Diyagramı



Hız ve Pozisyon Güncellemesi:

Parçacıkların yeni hız ve pozisyonları şu denklemlerle güncellenir:

Hız Güncellemesi:

$$v_i(t + 1) = w \cdot v_i(t) + c_1 \cdot r_1 \cdot (pBest_i - x_i(t)) + c_2 \cdot r_2 \cdot (gBest - x_i(t))$$

w : Atalet ağırlığı (denge sağlamak için).

c_1, c_2 : Öğrenme katsayıları.

r_1, r_2 : Rastgele değerler (0 ile 1 arasında).

Pozisyon Güncellemesi:

$$x_i(t + 1) = x_i(t) + v_i(t + 1)$$

PSO'nun Parametreleri Nasıl Ayarlanmalıdır?

PSO'nun performansı, doğru parametre ayarlarına bağlıdır. En yaygın parametreler şunlardır:

- **Parçacık sayısı:** Çok az parçacıkla çözüm kalitesi düşebilir, çok fazla parçacıkla ise hesaplama süresi artabilir.
- **İnertlik ağırlığı:** Parçacıkların geçmiş hızını ne kadar koruyacaklarını belirler.
- **Kognisyon katsayısı:** Parçacığın kendi en iyi deneyimini ne kadar takip edeceğini belirler.
- **Sosyal katsayı:** Parçacığın komşularının en iyi deneyimini ne kadar takip edeceğini belirler.

Parametre ayarları: Genellikle deneysel olarak belirlenir. Grid search, random search veya daha gelişmiş teknikler kullanılarak en iyi parametre kombinasyonu bulunabilir.

PSO'nun Katsayılarının Detaylı Anlatımı

$$v_i(t + 1) = w \cdot v_i(t) + c_1 \cdot r_1 \cdot (pBest_i - x_i(t)) + c_2 \cdot r_2 \cdot (gBest - x_i(t))$$

Parçacık Sürü Optimizasyonu (PSO) algoritmasında kullanılan katsayılar, parçacıkların hareketlerini ve dolayısıyla çözüm uzayında nasıl ilerlediklerini belirleyen önemli parametrelerdir. Bu katsayılar, parçacıkların hem kendi geçmiş deneyimlerine hem de sürünün genel deneyimine ne kadar önem vereceğini belirler. Şimdi, bu katsayıların formüldeki rollerini ve anlamlarını daha detaylı inceleyelim:

$$v_i(t + 1) = w \cdot v_i(t) + c_1 \cdot r_1 \cdot (pBest_i - x_i(t)) + c_2 \cdot r_2 \cdot (gBest - x_i(t))$$

1. İnertlik Ağırlığı (w)

- **Anlamı:** Parçacığın önceki hızını ne kadar koruyacağıdır. Yüksek bir inertik ağırlığı, parçacığın mevcut hızını daha uzun süre korumasına ve daha geniş bir alanı keşfetmesine neden olur. Düşük bir inertik ağırlığı ise parçacığın hızını daha hızlı değiştirmesine ve daha hızlı bir şekilde optimal çözüme yakınsamasına neden olur.
- **Formüldeki Yeri:** Hız güncelleme formülünde, parçacığın önceki hızını çarpan olarak ifade eder.
- **Etki:**
 - **Yüksek w:** Daha geniş arama alanı, yerel minimuma takılma riski az.
 - **Düşük w:** Daha hızlı yakınsama, ancak yerel minimumlara takılma riski artar.

$$v_i(t + 1) = w \cdot v_i(t) + c_1 \cdot r_1 \cdot (pBest_i - x_i(t)) + c_2 \cdot r_2 \cdot (gBest - x_i(t))$$

2. Kognisyon Katsayısı (c1)

- **Anlamı:** Parçacığın kendi en iyi çözümünü (kişisel en iyi) ne kadar takip edeceğidir. Bu katsayı, parçacığın kendi deneyimlerine ne kadar önem verdiğini gösterir.
- **Formüldeki Yeri:** Hız güncelleme formülünde, parçacığın kişisel en iyi çözümü ile mevcut konumu arasındaki farkı çarpan olarak ifade eder.
- **Etki:**
 - **Yüksek c1:** Parçacık, kendi en iyi çözümüne daha çok odaklanır ve daha hızlı bir şekilde kişisel en iyi çözümünü iyileştirebilir. Ancak, sürünün diğer parçacıklarının bilgilerinden yararlanma olasılığı azalır.

$$v_i(t + 1) = w \cdot v_i(t) + c_1 \cdot r_1 \cdot (pBest_i - x_i(t)) + c_2 \cdot r_2 \cdot (gBest - x_i(t))$$

3. Sosyal Katsayı (c2)

- **Anlamı:** Parçacığın sürünün en iyi çözümünü (küresel en iyi) ne kadar takip edeceğidir. Bu katsayı, parçacığın sürünün genel deneyimlerine ne kadar önem verdiğini gösterir.
- **Formüldeki Yeri:** Hız güncelleme formülünde, parçacığın küresel en iyi çözüm ile mevcut konumu arasındaki farkı çarpan olarak ifade eder.
- **Etki:**
 - **Yüksek c2:** Parçacık, sürünün en iyi çözümüne daha çok odaklanır ve sürünün diğer parçacıklarının bilgilerinden daha fazla yararlanır. Ancak, yerel minimumlara takılma riski artabilir.

PSO'nun Uygun Olduğu Problem Türleri

- **Sürekli Optimizasyon Problemleri:** PSO, sürekli bir çözüm uzayında en iyi çözümü bulmak için oldukça etkilidir. Bu tür problemlere örnek olarak, mühendislikte tasarım optimizasyonu, ekonomide portföy optimizasyonu ve yapay sinir ağlarında ağırlıkların ayarlanması verilebilir.
- **Çok Modlu Optimizasyon Problemleri:** Birden fazla yerel minimumun olduğu problemlerde, PSO'nun farklı başlangıç noktalarından arama yapma özelliği sayesinde global minimumu bulma olasılığı artar.
- **Kısıtlı Optimizasyon Problemleri:** PSO, bazı kısıtlar altında en iyi çözümü bulmak için uyarlanabilir. Ancak, karmaşık kısıtlar için ek mekanizmalara ihtiyaç duyulabilir.
- **Büyük Boyutlu Problemler:** PSO, büyük boyutlu problemlerde de kullanılabilir ancak performansı, problem yapısına ve parametre ayarlarına bağlı olarak değişebilir.

PSO Örnek

$$f(x) = x^2 + 2x - 3$$

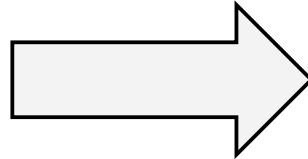
Örnek problemimiz verilen denklemin sonucunu 0 yapacak x değerini bulmak olsun.

1- Parçacık sayısı belirlenir.

Bu sayı arama uzayının genişliğine ve isteğe bağlı olarak belirlenir. Bu örnekte parçacık sayısını 3 olarak belirleyelim. Parçacıklar rastgele belirlenir. Parçacıkları aşağıda verildiği gibi sırasıyla 3, 7, 5 olarak belirlensin.

2 - Her parçacığın uygunluk değeri hesaplanır.

Bu örnekte amacımız problemin denklemini 0'a yaklaştırmak olduğundan uygunluk fonksiyonumuz problemin denkleminin ta kendisidir. Görüldüğü üzere belirlediğimiz x değerlerini denklemde yerine koyarak uygunluk değerlerini kolayca elde ettik.

$P_1 = 3$		$f(3) = 12$
$P_2 = 7$		$f(7) = 60$
$P_3 = 5$		$f(5) = 32$

3- pbest ve gbest değerleri hesaplanır.

Şu an ilk iterasyonda olduğumuzdan parçacıkların kendileri zaten pbest'leridir. gbest ise 0'a en yakın olan P1 parçacığdır.

4- c1, c2 değerleri ve rastgele olarak rand1, rand2 değerleri belirlenir.

c1 ve c2 değerleri parametreleri 2 olarak belirlendi(genelde böyle yapıyorlarmış). rand1 ve rand2 değerini de hesaplama kolaylığı açısından 2 belirlendi. Ayrıca daha ilk iterasyonda olduğundan parçacıklar herhangi bir hıza sahip değildir. Bu nedenle ilk değişim hızı V0'ı 0 kabul ediyoruz.

5- Parçacıkların değişim hızları hesaplanır.

$$P_1 \rightarrow 0 + 2 * 2 * (3 - 3) + 2 * 2 * (3 - 3) = 0$$

$$P_2 \rightarrow 0 + 2 * 2 * (7 - 7) + 2 * 2 * (3 - 7) = -16$$

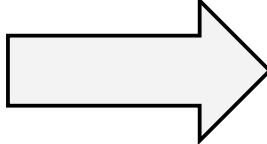
$$P_3 \rightarrow 0 + 2 * 2 * (5 - 5) + 2 * 2 * (3 - 5) = -8$$

Böylece formülümüzü kullanarak parçacıkların değişimi hesaplanmış oldu.

6- Parçacıkların yeni değerleri belirlenir.

Bu aşamada parçacıklar, değişim değerleri ile toplanarak yeni parçacıklar belirlenir.

7- Yeni parçacıkların uygunluk değerleri bulunur.

$$\begin{array}{ll} P_1 \rightarrow 3 + 0 = 3 & f(3) = 12 \\ P_2 \rightarrow 7 - 16 = -9 & f(-9) = 34 \\ P_3 \rightarrow 5 - 8 = -3 & f(-3) = 0 \end{array}$$


Denklemimizi 0 yapan değer -3 olarak bulunmuş oldu. Eğer çözüme ulaşamamış olsaydı; yeni parçacıklar da göz önüne alınarak yeniden pbest değeri ve bu zamana kadar gelmiş tüm pbest'lerin en iyisi olan gbest değeri belirlenecekti. Ek olarak rand1 ve rand2 değerleri tekrar belirlenip parçacıkların değişimi hesaplanacak ve yeni parçacıklar bulunacaktı. Ve bir kez daha uygunluk değerini bulup çözüme ne kadar yaklaştığımızı değerlendirecektik.