PARÇACIK SÜRÜSÜ OPTİMİZASYON ALGORİTMASI VE BENZETİM ÖRNEKLERİ

Seçkin TAMER, Cihan KARAKUZU
seckintamer@gmail.com, cihankk@kou.edu.tr

Kocaeli Üniversitesi, Müh. Fak., Elektronik ve Haberleşme Mühendisliği Bölümü İzmit/KOCAELİ

ÖZET
Bu bildiride kuş sürülerinin davranışlarından esinlenilerek ortaya çıkarılmış, populasyon tabanlı bir optimizasyon tekniği olan "Parçacık Sürüsü Optimizasyon Algoritması" anlatılmıştır. Sürü davranışlarının benzetimi, algoritmanın yapısı ve çeşitli uygulamaları sunulmuştur. Yapılan benzetimlerde algoritmanın fonksiyon optimizasyonunda ve yapay sinir ağlarının eğitiminde kullanılabileceği görülmüştür.

Anahtar Kelimeler: *parçacık sürüsü, optimizasyon algoritmaları, yapay sinir ağı eğitimi*

1. GİRİŞ
Günümüzde biyolojik sistemlerden esinlenilerek ortaya çıkarılmış birçok yöntem hesaplama problemlerinin çözümünde kullanılmaktadır. Örneğin yapay sinir ağları insan beyninin basitleştirilmiş bir modelidir, genetik algoritma ise insan evriminden esinlenilerek ortaya çıkarılmıştır. Biyolojik sistemlerin başka bir çeşidi olan sosyal sistemler, özellikle bireyin çevresiyle ve diğer bireylerle olan etkileşimini ve kollektif (ortak) davranışlarını incelemektedir. Bu davranışlar sürü zekası olarak adlandırılmaktadırlar.

Hesaplama alanında kullanılan ve sürülerden esinlenilerek ortaya konulan iki popular metot vardır. Bunlar "karınca koloni optimizasyonu"(KKO) ve "parçacık sürü optimizasyonu" (PSO) dur. KKO karıncaların davranışlarından esinlenilerek ortaya konmuş bir yöntemdir ve ayrık optimizasyon problemlerinde başarılı uygulamaları vardır [1]. PSO kavramı esasında sosyal yaşamın basitleştirilmiş bir benzetimidir. Daha sonra bu benzetim bir optimizasyon yöntemi olarak kullanılmaya başlanmıştır.

Parçacık Sürüsü (particle swarm) Optimizasyonu (PSO); 1995 yılında J.Kennedy ve R.C.Eberhart tarafından; kuş sürülerinin davranışlarından esinlenilerek geliştirilmiş populasyon tabanlı stokastik optimizasyon tekniğidir [2]. Doğrusal olmayan problemlerin çözümü için tasarlanmıştır. Çok parametreli ve çok değişkenli optimizasyon problemlerine çözüm bulmak için kullanılmaktadır. PSO, genetik algoritmalar gibi evrimsel hesaplama teknikleriyle bir çok benzerlik gösterir. Sistem rasgele çözümler içeren bir populasyonla başlatılır ve nesilleri güncelleyerek en optimum çözümü araştırır. PSO da parçacık olarak adlandırılan olası muhtemel çözümler, o andaki optimum parçacığı izleyerek problem uzayında dolaşırlar. PSO'nun klasik optimizasyon tekniklerinden en önemli farklılığı türev bilgisine ihtiyaç duymamasıdır. PSO'yu uygulamak, algoritmasında ayarlanması gereken parametre sayısının az olması sebebiyle oldukça basittir. PSO; fonksiyon optimizasyonu, bulanık sistem kontrolü, yapay sinir ağı eğitimi gibi bir çok alanda başarıyla uygulanabilmektedir [3, 4, 5, 6].

2. PARÇACIK SÜRÜSÜ OPTİMİZASYON ALGORİTMASI
Önceki bölümde de bahsedildiği gibi PSO kuş sürülerinin davranışlarının bir benzetimidir. Kuşların uzayda, yerini bilmedikleri yiyeceği aramaları, bir probleme çözüm aramaya benzetilir. Kuşlar yiyecek ararken yiyeceğe en yakın olan kuşu takip ederler. Parçacık olarak adlandırılan her tekil çözüm, arama uzayındaki bir kuştur. Parçacık hareket ettiğinde, kendi koordinatlarını bir fonksiyona gönderir ve böylece parçacığın uygunluk değeri ölçülmüş olur. (Yani yiyeceğe ne kadar uzaklıkta olduğu ölçülmüş olur.) Bir parçacık, koordinatlarını, hızını (çözüm uzayındaki her boyutta ne kadar hızla ilerlediği), şimdiye kadar elde ettiği en iyi uygunluk değerini ve bu değeri elde ettiği koordinatları hatırlamalıdır. Çözüm uzayındaki her boyuttaki hızının ve yönünün her seferinde nasıl değişeceği, komşularının en iyi koordinatları ve kendi kişisel en iyi koordinatlarının bir birleşimi olacaktır.

Çözüm uzayı problemdeki değişken veya bilinmeyen sayısına bağlı olarak çok boyutta olabilir. Örneğin; $5x^2 + 2y^3 - (z/w)^2 + 4$ fonksiyonunun çözüm uzayı x, y, z ve w bilinmeyenlerinden dolayı 4 boyutludur. Bu problemin çözüm uzayında tanımlanan bir parçacığın pozisyonu 4 koordinat ile P=[x, y, z, w] şeklinde belirtilmektedir. Görsel olarak insanların resmedemediği 4 veya daha fazla boyutlu karmaşık problemlerde çalışmanın PSO için herhangi bir zorluğu bulunmamaktadır. Örneğin PSO, bir yapay sinir ağının eğitiminde kullanılacaksa ve ağda 50 tane bağlantı ağırlığı mevcutsa problem 50 boyutlu bir uzayda çözülecektir.

PSO, bir grup rasgele çözümle (parçacık sürüsü) başlatılır ve güncellemelerle optimum çözüm bulunmaya çalışılır. Her tekrarlamada (iterasyonda), parçacık konumları, iki en iyi değere göre güncellenir. İlki; o ana kadar parçacığın elde ettiği en iyi çözümü sağlayan koordinatlardır. Bu değer *"pbest"* olarak adlandırılır ve hafızada saklanmalıdır. Diğer en iyi değer ise, populasyonda o ana kadar tüm parçacıklar tarafından elde edilen en iyi çözümü sağlayan koordinatlardır. Bu değer global en iyidir ve *"gbest"* ile gösterilir. Örneğin D adet parametreden oluşan n adet parçacık olduğunu varsayalım. Bu durumda populasyon parçacık matrisi eşitlik (1)'deki gibidir.

$$x = \begin{bmatrix} x_{11} & x_{12} & \dots & \dots & x_{1D} \\ x_{21} & x_{22} & \dots & \dots & x_{2D} \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ x_{n1} & x_{n2} & \dots & \dots & x_{nD} \end{bmatrix}_{n \times D} \quad (1)$$

Yukarıdaki matriste, i'ninci parçacık $x_i = [x_{i1}, x_{i2}, ...x_{iD}]$ olarak ifade edilir. Önceki en iyi uygunluk değerini veren i'ninci parçacığın pozisyonu $pbest_i = [p_{i1}, p_{i2}, ...p_{iD}]$ olarak ifade edilir. *gbest* ise her iterasyonda tüm parçacıklar için tektir ve $gbest = [p_1, p_2, ...p_D]$ şeklinde gösterilir. i'ninci parçacığın hızı (her boyuttaki konumunun değişim miktarı) $v_i = [v_{i1}, v_{i2}, ...v_{iD}]$ olarak ifade edilir. İki en iyi değerin bulunmasından sonra parçacık hızları ve konumları aşağıda verilen (2) ve (3) nolu denklemlere göre güncellenir.

$$v_i^{k+1} = v_i^k + c_1.rand_1^k.\left(pbest_i^k - x_i^k\right) + c_2.rand_2^k.\left(gbest^k - x_i^k\right) \quad (2)$$

$$x_i^{k+1} = x_i^k + v_i^{k+1} \quad (3)$$

Denklem (2)'de, c_1 ve c_2 öğrenme faktörleridir. c_1 ve c_2, her parçacığı *pbest* ve *gbest* pozisyonlarına doğru çeken, stokastik hızlanma terimlerini ifade eden sabitlerdir. c_1, parçacığın kendi tecrübelerine göre hareket etmesini, c_2 ise sürüdeki diğer parçacıkların tecrübelerine göre hareket etmesini sağlar. Düşük değerler seçilmesi parçacıkların hedef bölgeye doğru çekilmeden önce, bu bölgeden uzak yerlerde dolaşmalarına imkan verir. Ancak hedefe ulaşma süresi uzayabilir. Diğer yandan, yüksek değerler seçilmesi, hedefe ulaşmayı hızlandırırken, beklenmedik hareketlerin oluşmasına ve hedef bölgenin es geçilmesine sebep olabilir. Bu algoritma üzerinde araştırmacıların yaptığı denemelerde $c_1=c_2=2$ olarak almanın iyi sonuçlar verdiği belirtilmiştir. Denklemdeki $rand_1$ ve $rand_2$, [0,1] arasında düzgün dağılımlı rasgele sayılardır. k ise iterasyon sayısını belirtmektedir. Aşağıda PSO algoritması için gerekli olan prosedür özetlenmiştir.

For *her parçacık için başlangıç koşullamaları*
End
Do **For** *her parçacık için uygunluk değerini hesapla*
eğer uygunluk değeri, pbest ten daha iyi ise; şimdiki değeri yeni pbest olarak ayarla
End
Tüm parçacıkların bulduğu pbest değerlerinin en iyisini, tüm parçacıkların gbest'i olarak ayarla
For *her parçacık için*
(2) denklemine göre parçacık hızını hesapla
(3)denklemine göre parçacık pozisyonunu güncelle
End
While *maksimum iterasyon sayısına veya minimum hata koşulu sağlanana kadar devem et*

2.1 PSO Parametre Kontrolü

PSO da ayarlanması gereken çok fazla sayıda parametre yoktur. Aşağıda parametrelerin listesi ve tipik değerleri verilmiştir.

Parçacık sayısı: 20 ile 40 arasındadır. Birçok problem için 10 parçacık kullanmak yeterlidir. Bazı zor veya özel problemlerde ise 100 veya 200 parçacık kullanılması gerekebilir.

Parçacık boyutu: Optimize edilecek probleme göre değişmektedir.

Parçacık aralığı: Optimize edilecek probleme göre değişmekle birlikte farklı boyutlarda ve aralıklarda parçacıklar tanımlanabilir.

Vmax: Bir iterasyonda, bir parçacıkta meydana gelecek maksimum değişikliği (hız) belirler. Genellikle parçacık aralığına göre belirlenir. Örneğin X1 parçacığı (-10,10) aralığında ise Vmax=20 sınırlandırılabilir.

Öğrenme Faktörleri: c_1 ve c_2 genellikle 2 olarak seçilir. Fakat farklı da seçilebilir. Genellikle c_1, c_2 ye eşit ve [0, 4] aralığında seçilir.

Durma Koşulu: Maksimum iterasyon sayısına ulaşıldığında veya değer fonksiyonu istenilen seviyeye ulaştığında algoritma durdurulabilir.

PSO'nun şu andaki güncel versiyonu elde edilmeden önce çeşitli denemeler yapılmış ve bir çok yöntem ortaya atılmıştır. Algoritmanın başarımını arttırmak için Yuhui Shi ve R.C. Eberhart tarafından yapılan çalışmalar sonucunda 1998 yılında *Geliştirilmiş PSO* algoritması ortaya çıkarılmıştır[7, 8].

2.2 Geliştirilmiş PSO

Bu versiyonun en önemli farklılığı; parçacığın, komşularının ve kendisinin geçmişteki en iyi pozisyonları arasında arama yapmamasıdır. Komşularının geçmiş en iyi pozisyonu (gbest) ile tüm

parçacıkların en iyi orta pozisyonu arasında arama yapılır. Yani elde edilen pbest değerlerinin ortalaması alınır.

$p_i^* = [p_{i1}^*, p_{i2}^*, ...p_{iD}^*]$ olmak üzere; bireyin en iyi değeri aşağıdaki gibi hesaplanır.

$p_{ij}^* = (p_{1j} + p_{2j} + ...p_{nj})/n$ (j=1, 2,....,D) yani n adet paçacığın her boyuttaki en iyi (pbest) değerlerinin ortalaması alınmakta ve yeni p_i^* vektörü elde edilmektedir. Buna göre (2) denklemi aşağıdaki denklem (4)'e dönüşmektedir.

$$v_i^{k+1} = w.v_i^k + c_1.rand_1^k.\left(p_i^{*k} - x_i^k\right) + c_2.rand_2^k.\left(gbest^k - x_i^k\right) \quad (4)$$

Denklemde w eylemsizlik ağırlığıdır, w<1 olarak seçilmeli ve her iterasyonda doğrusal olarak azaltılmalıdır. PSO da eylemsizlik ağırlığı global ve yerel arama yeteneğini dengelemek için kullanılır. Büyük eylemsizlik ağırlığı global arama, küçük ağırlık ise yerel arama yapılmasını kolaylaştırır. Eylemsizlik ağırlığı yerel ve global araştırma arasındaki dengeyi sağlar ve bunun sonucunda yeterli optimal sonuca daha az iterasyonda ulaşılır. Buradaki her parçacık; sürüdeki sadece en iyi parçacığın değil sürüdeki diğer tüm parçacıkların tecrübelerinden de yararlanmış olur.

3. BENZETİM ÖRNEKLERİ

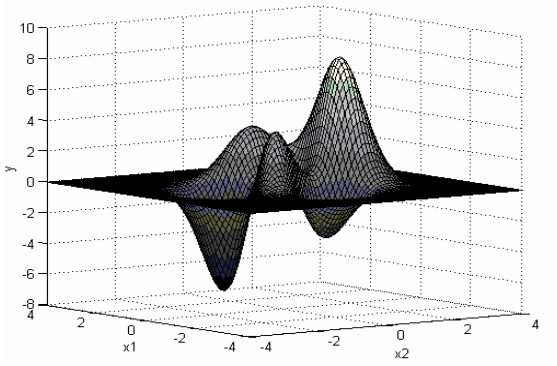
PSO algoritmasının başarımını test edebilmek için; Matlab programı yardımıyla çeşitli benzetimler yapılmıştır. Yapılan benzetimlerde Geliştirilmiş PSO algoritması kullanılmıştır. İlk olarak matematiksel ifadesi bilinen ve birçok yerel maksimum ve minimum içeren *"peaks"* fonksiyonu tarafından tanımlanan yüzeyde küresel minimum ve maksimum noktası çözümü gerçekleştirilmiştir. Bu problemde y fonksiyonu iki değişkene bağlı olduğundan parçacıklar P=[x_1,x_2] şeklinde alınmıştır. Peaks fonksiyonunun matematiksel ifadesi eşitlik (5)'te verilmiştir.

$$y = 3(1-x_1)^2.e^{\left[-x_1^2-(x_2+1)^2\right]} - 10.\left(\frac{x_1}{5} - x_1^3 - x_2^5\right).e^{\left[-x_1^2-x_2^2\right]} - \frac{1}{3}.e^{\left[-(x_1+1)^2-x_2^2\right]} \quad (5)$$

Uygulamada 10 adet parçacık kullanılmış olup PSO'nun minimum noktası için bulduğu çözüm eşitlik (6)'de, max noktası için bulduğu çözüm eşitlik (7)'da verilmiştir. Bu sonuçların fonksiyonun gerçek min/max noktalarına çok yakın olduğu Şekil 1 ve 2'den görülebilir.

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0,2283 \\ -1,6255 \end{pmatrix} \Rightarrow Min_{\ x}\, y = -6,5511 \quad (6)$$

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} -0,0093 \\ 1,5814 \end{pmatrix} \Rightarrow Max_{\ x}\, y = 8,1062 \quad (7)$$



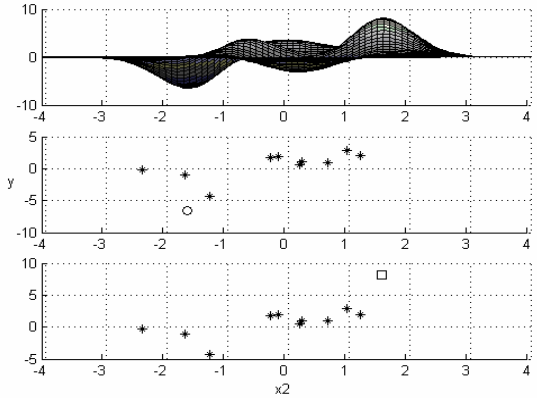
Şekil 1. Peaks fonksiyonunun tanımladığı yüzey

Tablo 1'de parçacıkların rasgele seçilen başlangıç konumları ve küresel çözümü ararken geldikleri en son konumları görülmektedir. Görüleceği üzere parçacıkların hepsi sürü mantığına uygun olarak global çözümün etrafında toplanmışlardır.

Tablo 1. Birinci örnek için başlangıç ve sonuç parçacık konumları

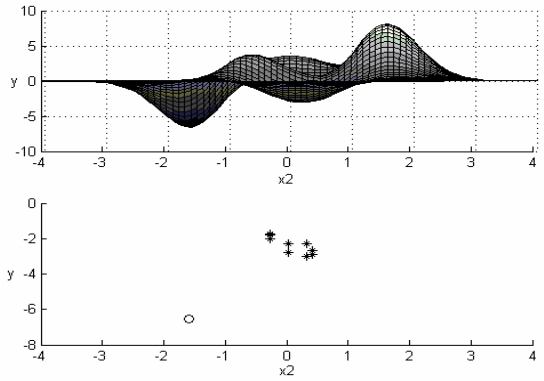
P0 (başlangıç)		**P (min)**		**P (max)**	
X1	**X2**	**X1**	**X2**	**X1**	**X2**
0.0559	0.6804	0.2283	-1.6255	-0.0093	1.5814
-1.1071	-2.3646	0.2283	-1.6255	-0.0093	1.5813
0.4855	0.9901	0.2282	-1.6256	-0.0093	1.5814
-0.0050	0.2189	0.2283	-1.6255	-0.0091	1.5814
-0.2762	0.2617	0.2283	-1.6255	-0.0093	1.5814
1.2765	1.2134	0.2283	-1.6255	-0.0093	1.5814
1.8634	-0.2747	0.2283	-1.6255	-0.0092	1.5814
-0.5226	-0.1331	0.2285	-1.6263	-0.0093	1.5814
0.1034	-1.2705	0.2286	-1.6253	-0.0093	1.5814
-0.8076	-1.6636	0.2282	-1.6256	-0.0094	1.5814

Şekil 2'de fonksiyon yüzeyinin ve parçacık pozisyonlarının x düzleminden yan görünüşü verilmiştir. Yıldızla işaretliler parçacıkların başlangıç konumlarını göstermektedir. Yuvarlakla işaretli konum parçacıkların küresel minimum çözümünde son adımda geldikleri yeri (Şekil 2b), kare ise parçacıkların küresel maksimum çözümünde son adımda geldikleri yeri göstermektedir (Şekil 2c).



Şekil 2 (a). Fonksiyonun yüzeyi, ***(b)*** Global minimum çözümündeki parçacıkların başlangıç ve son konumları (***c)*** Global maksimum çözümündeki parçacıkların başlangıç ve son konumları

PSO'nun başarımını daha detaylı test etmek için başlangıçta tüm parçacıklar yerel minimuma yerleştirilmiş (Şekil 3b yıldızla gösterilenler) ve bu durum için algoritma küresel minimumu bulmak için koşturulmuştur. Sonuç olarak bu başlangıç durumu için bile parçacıklar global minimuma çok yakın bir bölge etrafında toplanmışlardır (Şekil 3b yuvarlakla gösterilenler).



Şekil 3(a) Fonksiyonun yüzeyi, (***b)*** Global minimum çözümündeki parçacıkların başlangıç (yerel minimuma yerleştirilmiş) ve son konumları

Uygulamada en iyi çözüme ulaşmak için tüm parçacıkların bir araya toplanmasını beklemek gerekmez. Her iterasyon sonucunda bulunan gbest değeri istediğimiz değer fonksiyonunu sağlıyorsa, bu gbest değerini bulan parçacık çözüm olarak alınabilir.
PSO algoritmasının başarımını test etmek için yapılan ikinci benzetimde ise 2 katmanlı bir yapay sinir ağının eğitimi gerçekleştirilmiştir. PSO algoritmasıyla eğitilen ağ ile mantıksal EXOR problemi çözülmeye çalışılmış ve elde edilen sonuç, ağ eğitiminde kullanılan *klasik geri yayılım öğrenme algoritması* ile karşılaştırılmıştır. EXOR probleminde 2 giriş, 1 çıkış ve 4 adet örnek bulunmaktadır. Bu problem Şekil 4'de verilen ağın parametrelerinin PSO ile belirlenmesi sonucunda çözülmüştür. PSO algoritması gradyen bilgisine ihtiyaç duymadığı için hücre aktivasyon fonksiyonu seçiminde esneklik sağlar. Bu uygulamada 0-1 aralığında çıkış veren "sigmoidal" aktivasyon fonksiyonu kullanılmıştır. Sonlandırma kriteri; değer fonksiyonunun binde 1'e inmesi veya maksimum iterasyon sayısına (100) ulaşılması olarak seçilmiştir.

Değer fonksiyonu ise toplam karesel hata olarak denklem (7)'de verildiği gibi tanımlanmıştır.

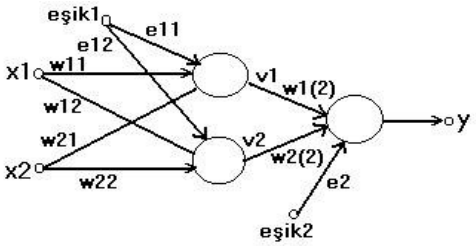
$$\cos t = E = \frac{1}{2}\sum_i e_i^2 \qquad (7)$$

Denklem (7)'deki hata (e_i) denklem (8) ile tanımlıdır.

$$e_i = yd_i - y_i \qquad (8)$$

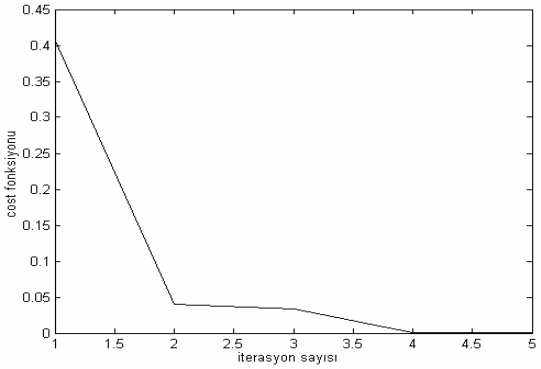
i ise örnek sayısını göstermektedir. Uygulamada 100 adet parçacık kullanılmıştır.
Ağda ayarlanması gereken eşitlik (9)'da verilen 9 adet parametre bulunduğundan, problem 9 boyutlu bir uzayda çözülecektir. Parçacıklar (1x9) yapısında aşağıdaki gibi düzenlenmiştir.

$$P = [w_{11} \quad w_{12} \quad w_{21} \quad w_{22} \quad e_{11} \quad e_{12} \quad w_1 \quad w_2 \quad e_2] \qquad (9)$$



Şekil 4. Eğitilecek ağ yapısı

Şekil 5'te PSO koşturulurken değer fonksiyonunun iterasyonla değişimi görülmektedir. PSO ile ağ eğitiminde 5 iterasyon sonucunda değer fonksiyonu 3.3172e-005'e kadar düşmüştür. EXOR problemi için elde edilen sonuçlar Tablo 2'de özetlenmiştir.



Şekil 5. PSO ile ağ eğitimi sırasında değer fonksiyonunun iterasyonla değişimi

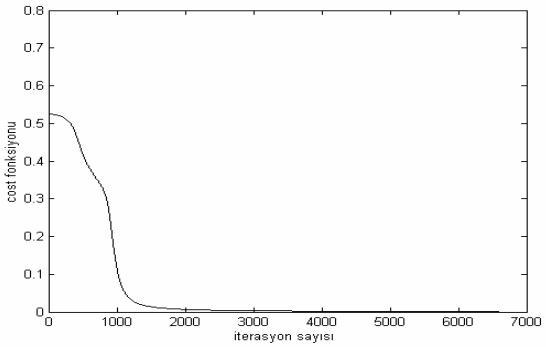
Tablo 2. PSO algoritması ile eğitilen ağın başarım göstergeleri

X1	X2	Yd	Y	Hata
0	0	0	0.0045	-0.0045
0	1	1	1.0000	0.0000
1	0	1	0.9949	0.0051
1	1	0	0.0045	-0.0045

Parçacıkların yani ağ parametrelerinin başlangıçtaki rasgele değerlerine bağlı olarak iterasyon sayısı değişebilmektedir. Genel olarak Şekil 6'deki eğitim seyri ile sonuçlanan geri yayılım algoritması ile kıyaslandığında PSO algoritmasının başlangıç parametreleri ne olursa olsun oldukça hızlı sonuç verdiği görülmektedir. Geriye yayılım ile ağ eğitiminde 6635 iterasyon sonucunda değer fonksiyonu 9.9994e-004 olarak elde edilmiştir. Bu eğitim ile elde edilen başarım Tablo 3'de özetlenmiştir.

Tablo 3. Geriye yayılım algoritması ile eğitilen ağın başarım göstergeleri

X1	X2	Yd	Y	Hata
0	0	0	0.0202	-0.0202
0	1	1	0.9767	0.0233
1	0	1	0.9766	0.0234
1	1	0	0.0224	-0.0224



Şekil 6. Geri yayılım algoritması ile ağ eğitimi sırasında değer fonksiyonunun iterasyonla değişimi

4. SONUÇLAR

Parçacık sürüsü optimizasyon algoritması kullanılan en yeni akıllı hesap tekniklerden biridir. Çok boyutlu problemlere oldukça hızlı ve yeterli sonuçlar verdiği bu bildiride verilen benzetim çalışmalarında gösterilmiştir. İşlem yükü ve gerekli hafıza miktarı parçacık sayısına ve probleminin boyutuna bağlıdır. Ancak algoritma; herhangi bir diferansiyel denklem çözümü ve gradyen hesabı gerektirmediğinden, ayrıca ayarlanması gereken az sayıda parametre bulunduğundan yürütülmesi kolaydır. PSO'nun; fonksiyon optimizasyonunda, global çözüme kolayca ulaştığı yapılan benzetimlerde görülmüştür.

Yukarıda verilen sonuçlardan da görüleceği üzere, EXOR problemini çözebilmek için gerekli ağ parametreleri PSO algoritması ile geri yayılım algoritmasından oldukça hızlı bir şekilde belirlenmiştir. Bu sonuç, yapay sinir ağlarının eğitimi için kullanılan en popüler yöntem olan geri yayılım öğrenme algoritması yerine PSO'nun kullanılabileceğini göstermektedir.

KAYNAKÇA

[1] Kalınlı A., Karaboğa N., Karaboğa D.; "A Modified Touring Ant Colony Optimization Algorithm for Contounous Functions", 16th International Symposium on Computer and Informatıon Science (ISCIS XVI), Işık University, Antalya (2001) s:437-444

[2] Kennedy, J. and Eberhart, R. C. "Particle swarm optimization" Proc. IEEE int'l conf. on neural networks Vol. IV, pp. 1942-1948. IEEE service center, Piscataway, NJ, 1995.

[3] Fábio A. Guerra and Leandro dos S. Coelho, "Multi-step ahead nonlinear identification of Lorenz's chaotic system using radial basis neural network with learning by clustering and particle swarm optimization", *Chaos, Solitons & Fractals, In Press, Corrected Proof, Available online 7 July 2006,*

[4] Jialin Zhou, Zhengcheng Duan, Yong Li, Jianchun Deng and Daoyuan Yu, "PSO-based neural network optimization and its utilization in a boring machine", *Journal of Materials Processing Technology, In Press, Corrected Proof, Available online 8 June 2006*

[5] Cosmin Danut Bocaniala and Jose Sa da Costa, "Application of a novel fuzzy classifier to fault detection and isolation of the DAMADICS benchmark problem", *Control Engineering Practice, Volume 14, Issue 6, June 2006, Pages 653-669*

[6] S. P. Ghoshal, "Optimizations of PID gains by particle swarm optimizations in fuzzy based automatic generation control", *Electric Power Systems Research, Volume 72, Issue 3, 15 December 2004, Pages 203-212*

[7] Shi, Y. and Eberhart, R. C. "A modified particle swarm optimizer." Proceedings of the IEEE International Conference on Evolutionary Computation pp. 69-73. IEEE Press, Piscataway, NJ, 1998

[8] Shi, Y. and Eberhart, R. C. "Parameter selection in particle swarm optimization.", Evolutionary Programming VII: Proc. EP 98 pp. 591-600. Springer-Verlag, New York, 1998.