Karma (PSO-GA)Rasgele Arama Algoritmasıyla Bulanık-Nöral Kontrolör Eğitimi

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

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

Anahtar sözcükler: Parçacık Sürüsü Optimizasyonu, Genetik Algoritma, Bulanık-Nöral Ağ Eğitimi, Kontrolör Parametrelerinin Optimizasyonu

Özet

Bu bildiride Parçacık Sürüsü Optimizasyon (PSO) algoritmasının başarımını artırmak için, Genetik Algoritma (GA) daki çaprazlama (crossover) ve değişim (mutasyon) işlevlerinin PSO algoritmasının içersine dahil edilmesiyle ortaya konulmuş karma algoritma (HPSOGA) ile çeşitli mimarilerdeki bulanık-nöral (Neuro-Fuzzy) kontrolörlerin eğitilmesi ve bu kontrolörlerin iki örnek sistem üzerindeki başarımı sunulmuştur. İlk olarak dinamik bir sistemi; kontrol etmek için RFNN (Recurrent Fuzzy Neural Network) yinelemeli bulanık ağ yapısındaki kontrolörün parametreleri klasik PSO ve HPSOGA ile optimize edilmiş, Karma algoritmanın üstünlüğü görülmüştür. İkinci örnek olarak ise, DC motor hızını kontrol edebilmek için üç farklı ağ yapısındaki (ANFIS, RFNN1, RFNN2) kontrolör, HPSOGA ile eğitilmiş ve ağ yapılarının başarımları karşılaştırılmıştır.

1. Giriş

Parçacık Sürüsü (particle swarm) Optimizasyonu (PSO); 1995 yılında J.Kennedy ve R.C.Eberhart tarafından; kuş sürülerinin davranışlarından esinlenilerek geliştirilmiş populasyon tabanlı rasgele arama algoritmasıdır [1, 2]. Doğrusal olmayan, çok parametreli optimizasyon problemlerine çözüm bulmak için kullanılmaktadır. 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; fonksiyon optimizasyonu, bulanık sistem kontrolü, yapay sinir ağı eğitimi gibi bir çok alanda başarıyla uygulanabilmektedir [3, 4, 5].

Bulanık-nöral kontrolör parametrelerinin, sistemi kontrol edebilecek şekilde belirlenmesi, çok parametreli bir optimizasyon problemidir. Bu bildiri; çeşitli yapılardaki ağlar için bu problemin PSO algoritmasıyla başarılı bir şekilde çözülebileceğini göstermeyi amaçlamıştır.

Bildirinin ikinci bölümünde klasik PSO algoritması tanıtılmıştır. Üçüncü bölümde karma algoritma, dördüncü bölümde ise kontrolör olarak eğitilen bulanık-nöral ağ yapıları açıklanmıştır. Beşinci bölümde iki farklı sistemin, farklı algoritmalarla eğitilen, farklı yapılardaki ağlarla kontrol edilmesinin benzetimine ait karşılaştırmalı sonuçlar verilmiştir. Son bölümde ise elde edilen sonuçlar yorumlanmıştır.

2. PSO Algoritması

PSO; parçacık olarak adlandırılan, başlangıçta rasgele atanmış bir grup aday çözümün çok boyutlu bir arama uzayında dolaşarak en iyi çözümü aramasına dayalı bir yöntemdir. Parçacıkların boyutu, problemin çözüm uzayının boyutuna eşittir. Örneğin D boyutlu bir problemin çözümünde kullanılacak sürüdeki i. parçacık aşağıdaki gibi ifade edilir.

$$p_i = [p_{i1}, p_{i2}, ...p_{iD}] \quad (1)$$

Her iterasyonda, tüm parçacıkların pozisyonları, iki en iyi değere göre güncellenir. İlki; o ana kadar parçacığın elde ettiği en iyi çözümü sağlayan değerdir. Bu değer *"pbest"* olarak adlandırılır. Diğer en iyi değer ise, populasyonda o ana kadar tüm parçacıklar tarafından elde edilen en iyi çözümü sağlayan değerdir. Bu değer global en iyidir ve *"gbest"* ile gösterilir.
i. parçacığın pbest değeri (2) deki gibi gösterilmektedir. Gbest değeri tüm parçacıklar için tektir ve (3) teki gibi gösterilmektedir.

$$pbest_i = [p_{i1}, p_{i2}, ...p_{iD}] \quad (2)$$

$$gbest = [p_1, p_2, ...p_D] \quad (3)$$

i'ninci parçacığın hızı (her boyuttaki konumunun değişim miktarı) (4) teki gibi ifade edilir.

$$v_i = [v_{i1}, v_{i2}, ...v_{iD}] \quad (4)$$

İki en iyi değerin bulunmasından sonra parçacık hızları ve konumları aşağıda verilen (5) ve (6) nolu denklemlere göre güncellenir.

$$v_i^{k+1} = w.v_i^k + c_1.rand_1^k.\left(pbest_i^k - p_i^k\right) + c_2.rand_2^k.\left(gbest^k - p_i^k\right) \quad (5)$$

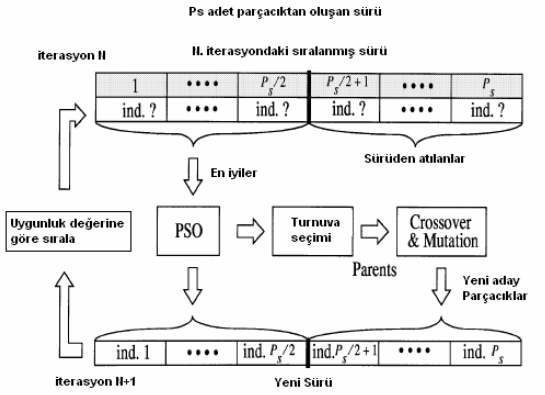
$$p_i^{k+1} = p_i^k + v_i^{k+1} \quad (6)$$

Denklem (5)'de, c_1 ve c_2 öğrenme faktörleridir. c_1 ve c_2, her parçacığı *pbest* ve *gbest* pozisyonlarına doğru çeken, rasgele değişen hızlanma terimlerini ifade eden sabitlerdir. Denklemdeki $rand_1$ ve $rand_2$, [0,1] arasında düzgün dağılımlı rasgele sayılardır. Denklemde w eylemsizlik ağırlığı olarak tanımlanmış olup, global ve yerel arama yeteneğini dengelemek ve yeterli optimal sonuca daha çabuk ulaşmak için kullanılmaktadır. k ise iterasyon sayısını göstermektedir. Sürüdeki parçacık sayısı, eylemsizlik ağırlığı w ve öğrenme oranları c1, c2' nin uygun değerler seçilmesi algoritmanın başarımını artırmaktadır [6].

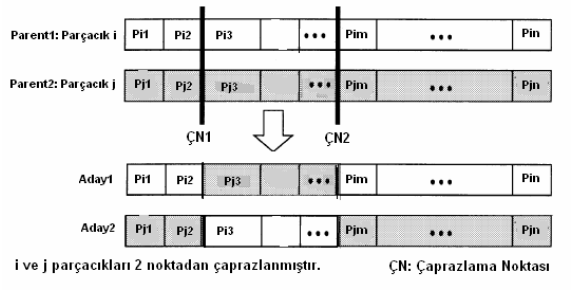
3. Karma PSO-GA Algoritması

Algoritmanın başarımını arttırmak için araştırmacılar tarafından bir çok yöntem ortaya atılmıştır. Algoritmayla ilgili yapılan bu çalışmaların ortak amacı, parçacıkların yerel çözümlere takılmadan, yeterli optimum çözüme daha hızlı bir şekilde yakınsamasını sağlamaktır. Algoritma yürütülürken ilerleyen iterasyonlarda en iyi olan parçacık hızı güncellenirken, (5) nolu ifadenin sağ tarafındaki ikinci ve üçüncü terimler sürekli sıfır olur. Bu yüzden başka bir parçacık en iyi (gbest) olana kadar şu andaki en iyi parçacığın konumunda bir değişiklik olmaz (bu sebeple ifadedeki birinci terimde sıfır olur). Diğer parçacıklar da en iyi parçacığı izleyecekleri için parçacıklar belirli bir yerde kümelenmeye başlar fakat kümelendikleri bu yer her zaman optimum çözüm olmayabilir. Bunu önlemek için araştırmacılar tarafından çeşitli çalışmalar yapılmış ve günümüzde de yapılmaya devam etmektedir. Çeşitli algoritmaların birleşimiyle elde edilen karma algoritma yapıları ile daha iyi sonuçlar alınabileceği düşünülmektedir. Örneğin *Gradient Descent* GD (eğim iniş) metoduyla PSO'nun birleştirilmesiyle karma bir algoritma ortaya konmuştur [7]. PSO algoritmasına benzer şekilde populasyon tabanlı bir algoritma olan *Genetik Algoritmanın* PSO ile birleştirilmesiyle de karma yapılar elde edilmiştir [9, 10]. Bu sayede algoritmaların birbirlerine karşı üstün yanları birleştirilerek, çözülecek probleme özgü yeni yaklaşımlar ortaya konmaktadır.

Karma algoritmada, sürüdeki bireylerin konumu güncellenirken PSO algoritmasına, GA'daki çaprazlama ve değişim operatörleri dahil edilmiştir. Karma algoritmanın işleyişi şu şekildedir: Öncelikle sürüdeki tüm parçacıklar sisteme uygulanır ve parçacıkların uygunluk değerleri hesaplanır. Hesaplanan bu uygunluk değerine bağlı olarak parçacıklar için *pbest* ve *gbest* değerleri belirlenir. Parçacıklar uygunluk değerlerine göre sıralanır. Sürüdeki parçacıkların iyi uygunluk değerine sahip olan yarısı seçilir, kötü uygunluk değerine sahip olan parçacıklar ise sürüden atılır. İyi uygunluk değerine sahip olan parçacıklar arasından, GA'daki "turnuva seçimi" yöntemiyle, çaprazlama olasılığına bağlı olarak 2 adet *parent* (ebeveyn) parçacık seçilir. Seçilen 2 adet ebeveyn parçacığın 2 noktalı çaprazlanması ile 2 yeni parçacık elde edilir. Çaprazlama işlemi şekil 2'de gösterilmiştir. Bu yeni parçacıkların bazı parametreleri değişim olasılığına bağlı olarak değiştirilir. Elde edilen bu yeni bireyler bir sonraki iterasyon için sürüye dahil edilir. Sürüdeki iyi uygunluk değerine sahip olan bireyler ise PSO algoritmasıyla (5) ve (6) eşitliklerine göre güncellenir. Karma algoritmanın işleyişi Şekil 1'de görülmektedir.



Şekil 1: HPSOGA algoritmasının işleyişi [9]



Şekil 2: İki noktalı çaprazlama

Algoritmanın her iterasyonunda *gbest* ten bağımsız yeni aday parçacıkların sürüye dahil edilmesiyle, parçacıkların yerel çözümlere takılması engellenmiş olup, yeterli optimum çözümün daha çabuk bulunması sağlanmıştır.

4. Bulanık-Nöral Ağlar

Bu bölümde kontrolör olarak eğitilecek olan çeşitli bulanık-nöral ağ yapıları anlatılacaktır. Bildiride 2 adet yinelemeli ağ ve standart ANFIS olmak üzere 3 ağ yapısı üzerinde durulmuştur. Bunlar;

- ANFIS (Adaptif Neural-Fuzzy Inference System)
- RFNN1: TRFN (Takagi-Sugeno-Kang(TSK) Recurrent Fuzzy Network)
- RFNN2 (Recurrent Fuzzy Neural Network)

Ağın katmanları ve ağda ayarlanması gerekli olan parametreler açıklanacaktır.

4.1 ANFIS yapısındaki ağ

Ağın birinci katmanındaki her bir düğüm A_i ve B_i gibi bir bulanık kümeyi ifade eder. Bu katmanın çıkışı kullanılan üyelik fonksiyonlarına (ÜF) bağlı olarak, girişlerin bulanık kümelere üyelik dereceleridir. Örneğin, A_i bulanık kümesi Gauss üyelik fonksiyonu ile tanımlanabilir.

$$\mu_{Ai}(x) = \exp\left[-\left(\frac{x - c_i}{\sigma_i}\right)^2\right] \tag{7}$$

Bu denklemde $\{\sigma_i, c_i\}$ Gauss fonksiyonunun varyansını ve merkezini belirleyen parametrelerdir.

İkinci katmandaki her bir düğüm (Π) ile tanımlanmış olup, gelen işaretleri çarparak sonuca ulaşan bir düğümdür. Bu katmandaki her bir düğümün çıkışı, bir kuralın aktiflik derecesini (ateşleme gücü) vermektedir. Aslında bu katmanda T-norm işlemi gerçekleştirilmektedir ve herhangi bir T-norm operatörü bu katmanda düğüm fonksiyonu olarak kullanılabilir.

$$w_i = \mu_{Ai}(x) * \mu_{Bi}(y) \ , \ i=1,2 \tag{8}$$

Üçüncü katmanda kural ateşleme derecelerinin normalizasyonu gerçekleştirilmektedir.

$$\overline{w_i} = \frac{w_i}{w_1 + w_2}, i = 1,2 \tag{9}$$

Dördüncü katmandaki her bir düğümde ilgili kuralın sonuç üzerindeki etkisi hesaplanmaktadır.

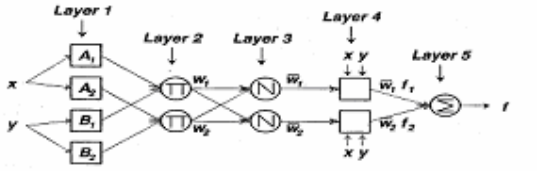
$$\overline{w_i}.f_i = \overline{w_i}.(p_i x + q_i y + r_i) \tag{10}$$

Burada $\overline{w_i}$, 3. katmanın çıkışıdır. { p_i , q_i , r_i } ise parametre kümesidir.

Beşinci katmandaki tek düğüm Σ ile tanımlanmış olup, gelen bütün işaretleri toplayarak ağ çıkışını hesaplar. Ağ çıkısı ağırlıklı ortalama ile bulunmuş olur.

$$f = \sum_i \overline{w_i}.f_i = \frac{\sum_i w_i.f_i}{\sum_i w_i} \qquad (11)$$

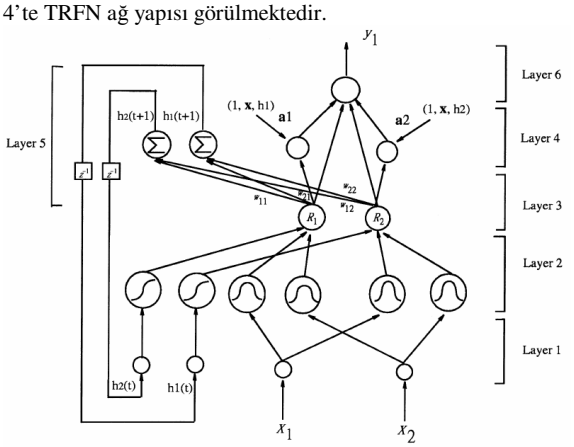
Aşağıdaki şekil 3'te iki giriş, iki kural, tek çıkışlı sugeno bulanık çıkarımına eşdeğer ANFIS mimarisi görülmektedir [11].



Şekil 3: İki-girişli birinci dereceden Sugeno bulanık çıkarımına eşdeğer ANFIS mimarisi [11]

4.2 RFNN1 (TRFN) yapısındaki ağ

Zamansal bilgi işleme problemine; kontrol, haberleşme, örüntü tanıma gibi bir çok alanda rastlanmaktadır. Bu tarz problemleri çözmenin etkili bir yolu yinelemeli ağ yapıları kullanmaktır [10]. Bu tür ağlarda geçmiş zamana ait bilgilerin kullanılması için geri besleme döngüleri bulunmaktadır. Şekil 4'te TRFN ağ yapısı görülmektedir.



Şekil 4: TRFN yapısındaki ağ [9]

Ağın birinci katmanında harici giriş değişkenleri ve geçmişteki kural çıkışlarına ait geri besleme değişkenleri (dahili) bulunmaktadır. İkinci katmandaki her bir düğüm bir üyelik fonksiyonunu temsil eder. Örneğin harici girişler için Gauss biçimli (eşitlik 7), dahili değişkenler için sigmoidal üyelik fonksiyonları kullanılabilir.

$$G(x) = \frac{1}{1 + e^{-s(x - c)}} \qquad (12)$$

Bu katmanda eğitim sırasında ayarlanması gereken; üyelik fonksiyonlarına ait (merkez, varyans, eğim) parametreler bulunmaktadır. Ağın üçüncü katmanında kural düğümleri bulunmaktadır. Her kural düğümü harici girişlerin ÜF'lerine ve o kuralın daha önceki çıkışına bağlı bir çıkış verir. Yani bu katmanda T-norm işlevi gerçekleştirilmiş olur (eşitlik 13). Üçüncü katmandaki her bir kural çıkışı, beşinci katmanda ağırlıklı toplama işlemine tabi tutulur. Böylece sonraki adımda kullanılacak dinamik geri besleme sinyali elde edilir (eşitlik 14). Her kural için bir geri besleme değişkeni bulunmaktadır. Burada; N_r kural sayısı olmak üzere (N_r^2) adet parametre bulunmaktadır. Dördüncü katmandaki düğümlerde, her kural için, harici giriş değişkenlerinin, geri besleme değişkeninin ve bir sabitin ağırlıklı doğrusal birleşiminden bir çıkış elde edilir (eşitlik 15). Burada ayarlanması gereken, her bir kural için, n giriş değişkeni için (n+2) adet parametre bulunmaktadır. Son katmanda ise ağırlıklı ortalama yöntemi ile durulandırma yapılır ve ağ çıkışı elde edilir (eşitlik 16).

$$R_i(t) = \mu(x(t), h_i(t)).....$$
$$..... = \frac{1}{1 + e^{-s(h_i(t) - c)}} . \exp\left[- \sum_{j=1}^{n} \left(\frac{x_j(t) - c_{ij}}{\sigma_{ij}} \right)^2 \right] \qquad (13)$$

$$h_i(t+1) = \left(\sum_{k=1}^{Nr} \mu_k(x(t), h_i(t)) \right).w_{ik} \qquad (14)$$

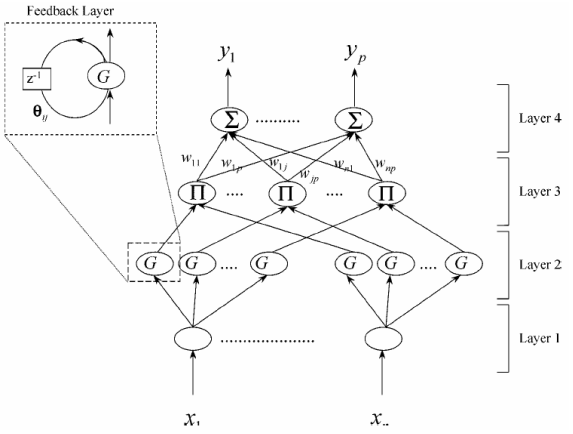
$$f_i(t) = a_{i0} + \left(\sum_{j=1}^{n} a_{ij}.x_j(t) \right) + a_{i(n+1)}.h_i \qquad (15)$$

$$y(t+1) = \frac{\sum_i^{N_r} R_i.f_i}{\sum_i^{N_r} R_i} \qquad (16)$$

Ağda eğitim sırasında ayarlanması gereken toplam $N_r(N_r+3n+2)$ adet parametre bulunmaktadır.

4.3 RFNN2 yapısındaki ağ

Bu yapısında geçmiş zamana ait bilgiler geri besleme döngüleriyle ağ içerisinde saklanmaktadır. Ağ yapısı Şekil 5'te verilmiştir [11]. Bu ağın farklılığı; giriş değişkenlerinin, geçmişte bulanık kümelere üyelik derecelerinin geri beslenerek, girişlere eklenmesidir. Yani kural çıkışları değil, kural aktiflik derecelerini belirleyen, giriş üyelik dereceleri geri besleme yapılarak ağ içinde tutulmaktadır.



Şekil 5: RFNN yapısındaki ağ [11]

Ağın birinci katmanında harici girişler bulunmaktadır. İkinci katmanda üyelik fonksiyonlarını temsil eden düğümler bulunmaktadır. Bu düğümlerde diğer ağ yapılarında olduğu gibi Gauss biçimli üyelik fonksiyonu kullanılabilir. Bu düğümlerin çıkışı, geri besleme yapılarak, ağırlıklı olarak giriş değişkenlerine eklenmekte ve düğüm girişi elde edilmektedir (eşitlik 17). Bu katmandaki her düğüm için 3 (2 ÜF parametresi, 1 bağlantı ağırlığı) parametre bulunmaktadır. Üçüncü katmanda T-norm işleviyle kural çıkışları elde edilmektedir (eşitlik 18). Dördüncü katmanda kural

çıkışlarının ağırlıklı toplamıyla çıkış değişkenleri hesaplanmaktadır. Bu katmanda her çıkış için kural sayısı kadar bağlantı ağırlığı parametresi bulunmaktadır (eşitlik 19).

$$u_{ij}^{2}(k) = x_i(k) + \theta_{ij}.O_{ij}^{2}(k-1) \quad (17)$$

(θ_{ij}: bağlantı ağırlığı)

u^2_{ij}, O^2_{ij}: 2 katmandaki i'ninci girişin, j'ninci bulanık kümesinin girişi ve çıkışı

$$O^{3}_{i} = \prod_{i} u^{3}_{i} \text{ , (T-norm: cebirsel çarpım)} \quad (18)$$

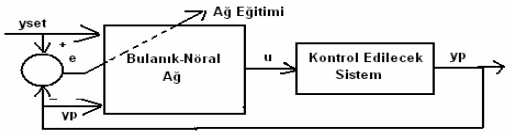
$$y_j = O^{4}_{i} = \sum_{i}^{m} u^{4}_{ij}.w^{4}_{i} \text{ , } (w^4_i\text{: bağlantı ağırlığı}) \quad (19)$$

5. Bulanık-Nöral Kontrolör Eğitimi

Kontrol edilecek sistemin modelinin bilinmediği veya eğiticili öğrenme ile kontrolör tasarımı için eğitim verisinin bulunmadığı durumlarda, rasgele arama algoritmalarının özellikle türev bilgisine ihtiyaç duymaksızın alışılmış ağ eğitim yöntemlerine göre sağladığı üstünlük tartışılmazdır. Zaten literatürdeki [9] gibi çalışmalar, bu algoritmaların geleneksel türev tabanlı öğrenme algoritmalarından daha iyi ve hızlı eğitim yaptıklarını göstermiştir. Bu sebeple bu çalışmada, bulanık-nöral kontrolör eğitimi için iki rasgele arama algoritması kullanılmış ve iki örnek sistem üzerindeki sonuçları irdelenmiştir.

Bu bölümde PSO ve HPSOGA algoritmalarıyla eğitilen, farklı yapılardaki bulanık-nöral kontrolörlerin, örnek iki sistem üzerindeki başarımlarını görmek için yapılan benzetimler sunulmuştur. Sistem modellemeleri, ağ eğitimleri ve benzetimler MATLAB programıyla gerçekleştirilmiştir. Ağ yapılarının başarımların karşılaştırılabilmesi için ağların giriş ve ÜF yapıları aynı alınmıştır. Ağ girişleri; yset (set değer/ön ayar değeri) ve yp (sistemin o anki çıkışı), ağ çıkışı ise sisteme uygulanacak kontrol sinyali (u) olarak tanımlanmıştır. Kontrolör eğitiminin blok yapısı Şekil 6'da gösterilmiştir. Eğitimlerde PSO ve HPSOGA algoritmalarına ait aşağıda verilen parametreler kullanılmıştır. Eğitimlerde parçacıklar için herhangi bir sınırlama getirilmemiştir. Yani parçacıklar arama uzayında serbest bırakılmıştır.

Parçacık sayısı=60
Eylemsizlik ağırlığı(w)=[1,2 0,8] aralığında iterasyona bağlı doğrusal azalan değer
Öğrenme oranları $c_1=c_2=2$
Mutasyon olasılığı=0.1
Çaprazlama olasılığı=0.5



Şekil 6: Bulanık-Nöral ağ ile sistem kontrolü

Algoritmanın; bulanık-nöral kontrolör eğitimindeki başarımını görebilmek için ağlar rasgele başlangıç koşulları için 100'er kez eğitilmişlerdir. Farklı sistemlerin, farklı algoritmalarla eğitilen, farklı yapıdaki ağlarla kontrolünde elde edilen en iyi ve ortalama sonuçlar sunulmuştur.

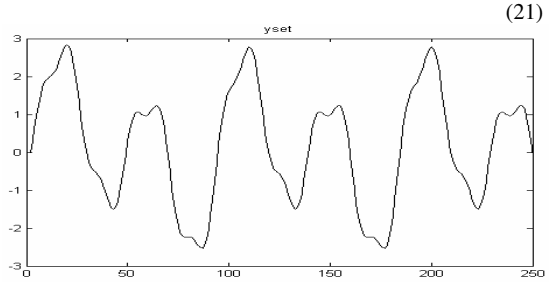
ÖRNEK 1: Tek giriş, tek çıkışlı ayrık dinamik sistem kontrolü

Ayrık zaman ifadesi eşitlik 20'de verilen dinamik sistemi kontrol edebilmek için TRFN ağ yapısında PSO ve HPSOGA algoritmalarıyla bulanık-nöral kontrolör eğitimi yapılmış, iki algoritmanın başarımları karşılaştırılmıştır. Eğitimler sistemi, eşitlik 21'de ifade edilen oldukça değişken bir set değer yörüngesinde tutmak için gerçekleştirilmiştir.

$$y_p(k+1) = \frac{y_p(k).y_p(k-1).(y_p(k-2)+2{,}5)}{1+y_p^2(k)+y_p^2(k-1)} + u(k) \quad (20)$$

Set değer Şekil 7'de görülmektedir. Eğitimler; yukarıda verilen parametrelerle, rasgele başlangıç değerleri için 100 kez tekrarlanmış elde edilen ortalama ve en iyi sonuçlar verilmiştir. Kontrol edilmek istenen sistem dinamik olduğu için başlangıç konumuna göre sistem girişine hiçbir şey uygulanmadığında bile çıkış üretmektedir.

$$y_r(k+1) = 0{,}6.y_r(k) + 0.2y_r(k-1) + 0{,}5.\sin(2\pi\pi/45) + 0{,}2.\sin(2\pi\pi/15) + 0{,}2.\sin(2\pi\pi/90) \quad (21)$$

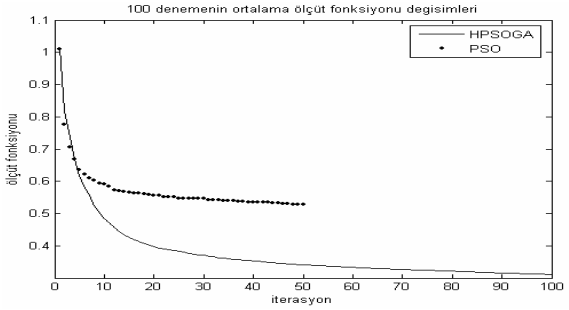


Şekil 7: Değişken set değer

Ölçüt fonksiyonu olarak eşitlik 22'de verilen fonksiyon kullanılmıştır. Benzetimlerde N=250 ayrık örnek değeri kullanılmıştır..

$$J = \left(\frac{1}{N} \sum_{k=1}^{N} \left(y_r(k) - y_p(k) \right)^2 \right)^{1/2} \quad (22)$$

Rasgele başlangıç pozisyonlarında 100 kez tekrarlanan eğitimler sonucunda elde edilen ortalama ve en iyi ölçüt değerleri tablo 1'de verilmiştir. Eğitimler seyrince ölçüt fonksiyonlarının ortalama değişimi şekil 8'de görülmektedir.



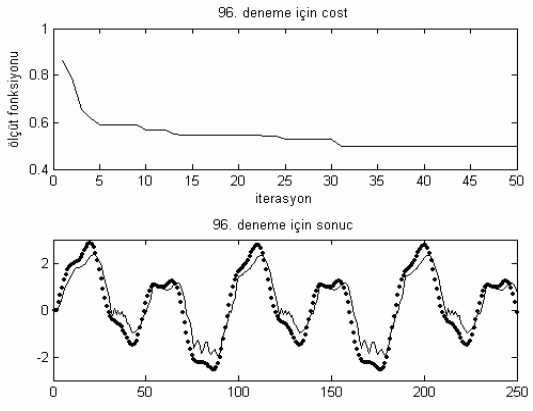
Şekil 8: Ölçüt fonksiyonunun değişimi

Tablo 1: Elde edilen en düşük ölçüt değerleri

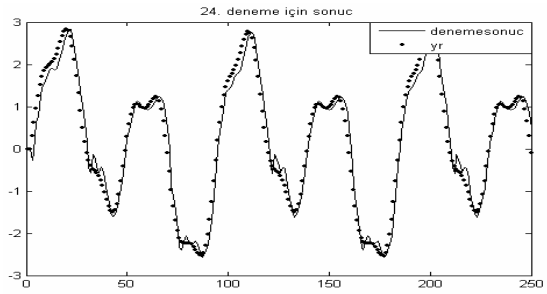
	HPSOGA	PSO
Ortalama Ölçüt Değeri	0,3113	0,5269
En iyi ölçüt değeri	0,2130	0,3088

HPSOGA algoritmasıyla yapılan eğitimler 100 iterasyon sürdürülmüştür. PSO algoritmasıyla yapılan eğitimler ise 50 iterasyonda sonlandırılmıştır. PSO algoritması yerel çözümlere takılabildiği için uzun iterasyon sayılarında fazla bir iyileşme sağlanamamaktadır aksine bu durum fazla işlem

yüküne sebep olmaktadır. Bu duruma örnek bir iterasyonun seyri Şekil 9(a)'da görülmektedir. HPSOGA algoritmasının, yerel çözümlere takılma olasılığı, PSO algoritmasından daha az olduğu için gerçekleştirilen 100 denemenin büyük bir çoğunluğunda iyi sonuçlar elde edilmiştir. PSO algoritmasıyla gerçekleştirilen eğitimlerde ise bu oran daha düşüktür. HPSOGA algoritmasıyla gerçekleştirilen eğitimler sonucunda elde edilen kontrol sonuçlarından biri Şekil 10'da görülmektedir. Sistem çıkışının istenilen set değer yörüngesini oldukça iyi bir şekilde takip ettiği görülmektedir. PSO algoritmasıyla gerçekleştirilen eğitimler sonucunda elde edilen kontrol sonuçlarında ise ölçüt fonksiyonunun yeterince azaltılamadığı, sistem çıkışının set değer yörüngesini salınımlarla takip ettiği görülmüştür. Bu durum Şekil 9(b)'de görülmektedir.



Şekil 9: PSO algoritmasıyla gerçekleştirilen 96. deneme için elde edilen sonuçlar: a) eğitim boyunca ölçüt fonksiyonunun değişimi. b) eğitilen ağın kontrol başarımı



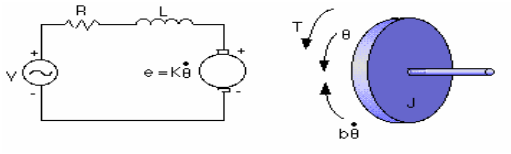
Şekil 10: HPSOGA algoritmasıyla gerçekleştirilen 24. deneme sonunda eğitilen ağla kontrol edilen sistem çıkışı

ÖRNEK 2: DC Motor hız kontrolü

DC motorlar, kontrol sistemlerinde sıklıkla denetim elemanı olarak kullanılmaktadırlar. Sağladıkları dönme hareketi sırasında açısal hızlarının ve açısal konumlarının kontrol edilmesi gerekir. Bu bildiride HPSOGA algoritmasıyla eğitilen 3 farklı bulanık-nöral ağ ile DC motor hızı kontrol edilmeye çalışılmıştır. DC motorun aşağıda verilen fiziksel parametrelere sahip olduğu varsayılarak modellenmesi gerçekleştirilmiştir.

* Rotorun eylemsizlik momenti (J) = 0.01 kg.m^2/s^2
* Mekanik sistemin sönümleme sabiti (b) = 0.1 Nms
* Elektromotor kuvveti sabiti (K=Ke=Kt) = 0.01 Nm/Amp
* Stator direnci (R) = 1 ohm
* İndüktans (L) = 0.5 H

DC motorun armatürüne ait elektrik diyagramı ve rotora ait serbest cisim diyagramı Şekil 11'de verilmiştir.



Şekil 11: DC motor modeli

Motorun torku T, Kt sabitiyle armatür akımına(i) bağlıdır (eşitlik 23). Ters elektromotor kuvveti(e) ise Ke sabitiyle açısal hıza bağlıdır (eşitlik 24).

$$T = K_t.i \tag{23}$$

$$e = K_e.\dot{\theta} \tag{24}$$

Yukarıdaki şekilden; Newton ve Kirchoff yasalarına göre aşağıdaki eşitlikler elde edilir.

$$J.\ddot{\theta} + b.\dot{\theta} = K.i \tag{25}$$

$$L.\frac{di}{dt} + R.i = V - K.\dot{\theta} \tag{26}$$

Açısal hız ve armatür akımı durum değişkeni olarak, giriş değişkeni olarak gerilim, çıkış değişkeni olarak açısal hız alındığında DC motor aşağıdaki durum-uzay eşitlikleriyle modellenebilir.

$$\frac{d}{dt}\begin{bmatrix}\dot{\theta}\\ i\end{bmatrix} = \begin{bmatrix}-\frac{b}{J} & \frac{K}{J}\\ -\frac{K}{L} & -\frac{R}{L}\end{bmatrix}.\begin{bmatrix}\dot{\theta}\\ i\end{bmatrix} + \begin{bmatrix}0\\ \frac{1}{L}\end{bmatrix}.V \tag{27}$$

$$\dot{\theta} = \begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}\dot{\theta}\\ i\end{bmatrix} \tag{28}$$

DC motor hızını kontrol edebilmek için eğitilecek bulanık-nöral ağlar, karşılaştırma yapabilmek için benzer giriş ve ÜF yapısında tasarlanmış ve HPSOGA algoritmasının yukarıda verilen parametreleriyle eğitilmiştir. Sistemin benzetimi [0 0] başlangıç koşulu altında, 5sn boyunca işletilmiştir. Eğitimler sinüs biçimli sürekli değişen bir set değer için rasgele parametrelerle başlanıp, 50 kez tekrarlanmış ve elde edilen ortalama ve en iyi sonuçlar verilmiştir.

ANFIS yapısındaki ağda; giriş kısmında Gauss biçimli 4 adet üyelik fonksiyonu için 8 parametre, çıkış kısmında ise 4 kural için 12 adet parametre olmak üzere ağda toplam 20 parametre bulunmaktadır. Sürüdeki her bir parçacık 20 adet parametre içerecek şekilde aşağıdaki gibi düzenlenmiştir.

$$P_i = [c_1\ \sigma_1\ .....c_4\ \sigma_4\ p_1\ q_1\ r_1\ .....p_4\ q_4\ r_4] \tag{29}$$

TRFN yapısındaki ağda; 2 giriş için Gauss biçimli 4 üyelik fonksiyonu, 4 geri besleme girişi için de sigmoidal biçimli 4 üyelik fonksiyonu kullanılmıştır. Bu fonksiyonlar toplam 16 adet parametre içermektedir. Ağın üçüncü katmanında 4 adet kural düğümünden, 4 adet geri besleme birimine 16 adet bağlantı ağırlığı parametresi bulunmaktadır. Çıkış katmanında her bir kural için, 4 parametre (1 sabit, 2 giriş ve 1 kural çıkışı bağlantı ağırlıkları), 4 kural için toplam 16 parametre bulunmaktadır. Yani ağda ayarlanması gereken 48 adet parametre bulunmaktadır. Parçacıklar aşağıdaki gibi düzenlenmiştir.

$$P_i = [c_1\ \sigma_1\ .....c_4\ \sigma_4\ .....w_{11}w_{12}....w_{44}\ a_1\ a_2.....a_{16}] \tag{30}$$

RFNN yapısındaki ağda; giriş kısmında Gauss biçimli 4 adet üyelik fonksiyonu için 8 parametre, üyelik fonksiyonlarının geri besleme bağlantı ağırlıkları için 4 parametre, çıkış katmanında 4 kural için 4 adet bağlantı ağırlığı parametresi

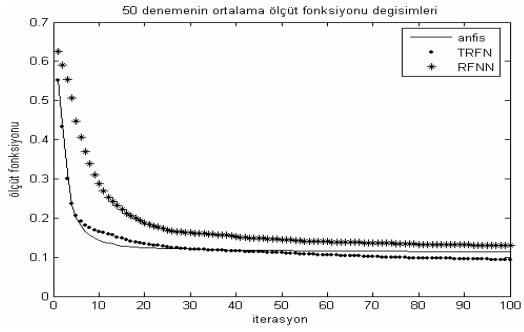
olmak üzere toplam 16 adet parametre bulunmaktadır. Parçacıklar aşağıdaki gibi düzenlenmiştir.

$$P_i = [c_1 \ \ \sigma_1 \ \ .....c_4 \ \ \sigma_4 \ \ \theta_1 ..... \theta_4 \ \ w_1 \ \ ..... w_4] \quad (31)$$

100 iterasyonluk ağ eğitimleri; farklı başlangıç koşulları için 50 kez tekrarlanmış, elde edilen en iyi ve ortalama ölçüt değerleri Tablo 2'de verilmiştir. Farklı ağ yapıları için 50 kez tekrarlanan eğitimler sırasında ölçüt fonksiyonunun değişimlerinin ortalaması Şekil 12'de görülmektedir.

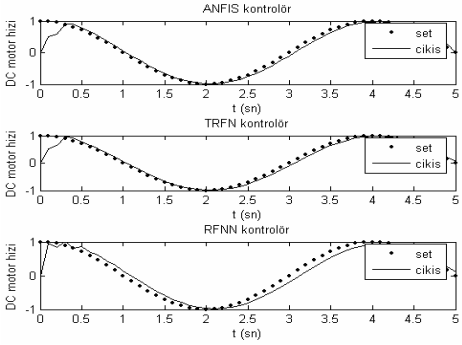
Tablo 2: Elde edilen en düşük ölçüt değerleri

	ANFIS	TRFN	RFNN
Ortalama Ölçüt Değeri	0,1133	0,0934	0,1295
En Düşük Ölçüt Değeri	0.1112	0,0282	0,0483



Şekil 12: Ağların eğitimi sırasında ölçüt fonksiyonunun değişimi

Eğitimler sonucunda sistem çıkışının yani DC motor hızının istenilen set değer yörüngesini takip ettiği şekil 13'te görülmektedir. Eğitim sonundaki sistemin birim basamak yanıtları da şekil 14'te verilmiştir. Şekillerden de görüldüğü gibi sistem; 0.5 sn gibi kısa bir sürede, kalıcı durum hatası ve aşma gerçekleşmeden istenilen set değere ulaşmıştır. Bu sonuçlara göre algoritmanın farklı yapılardaki kontrolör ağ parametrelerini başarıyla bulduğu görülmektedir. 50 kez tekrarlanan eğitimler sonunda elde edilen sonuçlara göre TRFN yapısındaki ağın daha iyi sonuç verdiği görülmektedir.

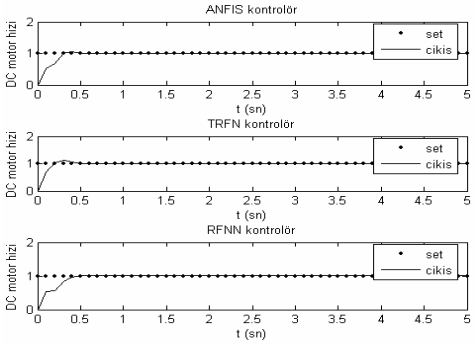


Şekil 13: Eğitimler sonucunda elde edilen ağ yapılarıyla kontrol edilen DC motor hızının değişimi

6. Sonuçlar

Bu bildiride PSO algoritmasının başarımını artırmak için geliştirilmiş, HPSOGA algoritması kullanılarak eğitilen bulanık-nöral kontrolörlerle çeşitli sistemlerin denetlenmesine ait benzetim sonuçları sunulmuştur. Yapılan benzetimlerde dinamik bir sistemi, oldukça değişken bir set değer yörüngesinde tutabilmek için TRFN yapısındaki bir ağ HPSOGA algoritmasıyla eğitilmiş ve PSO ile yapılan eğitim sonuçlarına göre oldukça başarılı sonuçlar elde edilmiştir.

Gerçekleştirilen ikinci benzetimde ise 3 farklı yapıdaki ağ; DC motor hızını kontrol edebilmek için HPSOGA algoritmasıyla eğitilmiştir. Sinüzoidal şekilde değişen set değeri için gerçekleştirilen eğitimler sonucunda oldukça başarılı sonuçlar elde edilmiştir. TRFN yapısındaki ağın, diğer ağlara göre daha iyi sonuç verdiği görülmüştür. Ağ yapılarının birbirlerine karşı üstünlüklerini test etmek için başka sistemler üzerinde, daha kapsamlı çalışmalar yapılması gerekmektedir.



Şekil 14: Eğitilen ağlarla kontrol edilen DC motorun birim basamak yanıtı

7. Kaynaklar

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

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

[3] 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, s: 203-212

[4] Jih-Gau Juang, Bo-Shian Lin, Kuo-Chih Chin, "Automatic Landing Control Using Particle Swarm Optimisation", Proceedings of the IEEE International Conference on Mechatronics, July 10-12 2005, Taipei, Taiwan

[5] Hamdi A. Awad, "A Novel Particle Swarm-Based Fuzzy Control Scheme", 2006 IEEE International Conference on Fuzzy Systems Canada July 16-21, 2006

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

[7] M. Aliyari Shoorehdeli, M. Teshnehlab, A. K. Sedigh, "A Novel Training Algorithm in ANFIS Structure", Proceedings of the 2006 American Control Conference Minneapolis, Minnesota, USA

[8] A. A. Esmin, G. Lambert-Torres, "Fitting Fuzzy Membership Functions using Hybrid Particle Swarm Optimization", 2006 IEEE International Conference on Fuzzy Systems, July 16-21, 2006

[9] Chia-Feng Juang, "A Hybrid of Genetic Algorithm and Particle Swarm Optimization for Recurrent Network Design", IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics, Vol. 34, No:2, April 2004

[10] Jyh-Shing Roger Jang, "ANFIS : Adaptive-Network-Based Fuzzy Inference System", Transactions on Systems, Man, and Cybernetics, Vol. 23, No: 3, May/June 1993

[11] Ching-Hung Lee ve Ching-Cheng Teng, "Identification and Control of Dynamic Systems Using Recurrent Fuzzy Neural Networks", IEEE Transactions on Fuzzy Systems, Vol. 8, No: 4, August 2000