YAPAY BAĞIŞIKLIK SİSTEMİ İLE BULANIK YAKLAŞTIRICI KURAL TABANI OPTİMİZASYONU

Erkan DUMAN[1] **Erhan AKIN[2]**

[1,2]Fırat Üniversitesi Mühendislik Fak. Bilgisayar Müh. Bölümü, 23119-Elazığ
[1]e-posta: erkanduman@firat.edu.tr [2] e-posta: eakin@firat.edu.tr

ÖZET : Bu çalışmada, popüler bir optimizasyon aracı olan Yapay Bağışıklık Sistemi(YBS) ile, bulanık denetleyici tasarımındaki parametrelerin belirlenmesi gibi deterministik olmayan bir işlevin, nasıl gerçekleştirilebileceği ve nasıl uygulanabileceği incelenmiştir. Önerilen yöntemin performansını değerlendirmek için doğrusal olmayan ve daha önce literatürde benzeri uygulamalarda tercih edilen bir fonksiyon yaklaştırma örneği gerçekleştirilmiştir. MATLAB ortamında gerçekleştirilen bu deneysel benzetim ortamında hangi parametrelerin ne etkileri olduğu da kolaylıkla irdelenmiştir. Ayrıca aynı yöntemin, tasarımında bilgi veya deneyim gerektiren diğer yumuşak hesaplama teknikleri için de kullanılabileceği gösterilmektedir.

***Anahtar sözcükler**: Yapay Bağışıklık Sistemi, Bulanık Yaklaştırıcı, Optimizasyon*

1. GİRİŞ

Bulanık mantık tabanlı denetleyiciler, klasik kontrol algoritmaları üzerine kurulan denetleyicilerin sonuç vermediği çoğunlukla doğrusal olmayan, kompleks sistemlerde basit deneysel kurallar üzerine kolaylıkla inşa edilip uygulanabilmektedir. Bu nedenle bir çok farklı alanda sayısız uygulama örneği ile kullanıla gelmektedir. Tasarımcı sistemin modelini bilmek zorunda kalmadan deneyimleri ile sistemin kontrolü için gerekli kuralları çıkarabilir. Ancak her zaman için bu deneyim veya bilgi mevcut olmayabilmektedir.

Bulanık mantık tabanlı bir denetleyici tasarlarken iki farklı durumla karşılaşılabilir: birincisi kontrol edilecek sistemin çalışması ve davranışı hakkında herhangi bir deneyime veya bilgiye sahip olunmadığı için gerekli kuralların ortaya çıkarılamaması. İkincisi ise bulanık denetleyici için gerekli kuralların mevcut olmasına rağmen denetleyicinin parametrelerinin nasıl seçileceği konusunda bilgi sahibi olunamamasıdır. Bu iki durumla baş edebilmek için çeşitli optimizasyon araçları denenmiş ve başarılı sonuçlar elde edilmiştir. YBS gibi popüler bir optimizasyon aracının bu alanda nasıl bir performans göstereceği sorusu da ilgi çekicidir.

Optimizasyon araçları ile bulanık denetleyicilerin kural tabanları, üyelik fonksiyonlarının sayısı, biçimleri ve üyelik fonksiyonlarının parametreleri maksimum performans elde edilecek şekilde hesaplanabilmektedir. Bu çalışmada ise önerilen yöntem ile bir bulanık denetleyicinin bahsedilen bütün parametreleri YBS kullanılarak optimuma yaklaştırılmaya çalışılmıştır.

YBS, genetik algoritmalar, yapay sinir ağları, karınca kolonisi gibi sezgisel yöntemlere benzer şekilde doğadan esinlenerek üretilmiş bir yöntemdir. Canlılardaki bağışıklık sisteminin özet bir modeli çıkarılmış ve bir çok farklı alanlarda uygulanmıştır[4]. Ortaya çıkarılan standart bir model olmamasına rağmen uygulamalarda kullanılan modeller, bağışıklık sistemlerinin temel biyolojik özellikleri olan öğrenme, hatırlama, sınıflandırma ve algılama özelliklerini sergilemektedirler[3]. Bu özellikleri bünyesinde bulunduran bir hesapsal yöntemin başarısız olması pek mümkün değildir. Bu nedenle YBS, yumuşak hesaplama teknikleri içerisinde yer almaya aday bir yöntemdir ve örüntü tanıma, iş programlama, robotik, veri analizi gibi bir çok alanda başarılı sonuçlar vermiştir[2].

Bu makalede öncelikle YBS'nin ne olduğu, nasıl çalıştığı ve algoritması verilecektir. Daha sonra bu algoritmanın bulanık denetleyici tasarımına nasıl adapte edildiği gösterilecektir. Sonunda ise önerilen yöntemin performansını test etmek için MATLAB ortamında gerçekleştirilen örnek bir uygulama sonuçları verilecek ve çalışma ile ilgili önerilere ve düşüncelere değinilecektir.

2. YAPAY BAĞIŞIKLIK SİSTEMİ

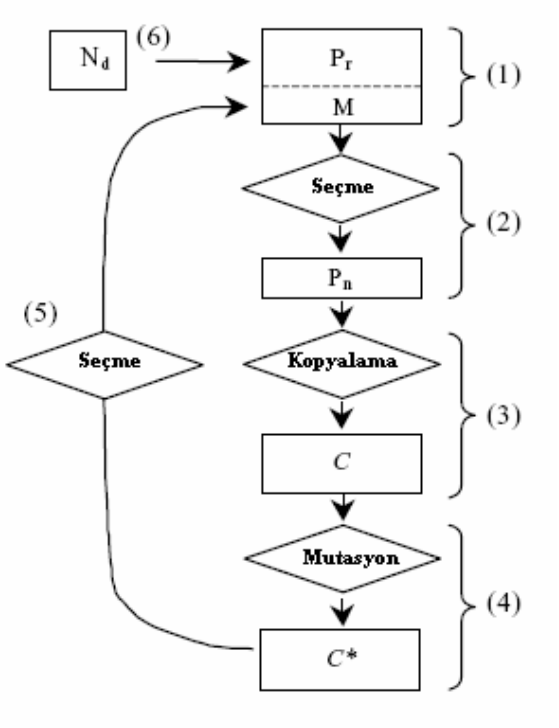
YBS, yapay sinir ağları ve genetik algoritmalarda olduğu gibi doğadan taklit edilerek ortaya çıkarılmış genel amaçlı bir sezgisel yöntemdir[7]. Canlılardaki savunma mekanizması özetlenip modellenerek oluşturulmuş bu optimizasyon algoritmasının, mühendislik alanındaki bir çok kompleks problemin çözümünde nasıl performans göstereceği incelenmiştir[3,4]. Yapay sinir ağları ve genetik algoritmalar kadar eski olmayan bu yöntemin popülerliği gittikçe artmaktadır. Çünkü her iki yönteme göre de avantajları olduğu yapılan uygulamalarda gözlemlenmiştir. Ancak görünen o ki; uygulama bazlı seçim yapmak ve bunun üzerine artıları ve eksileri göz önüne almak daha sağlıklıdır.

YBS, genetik algoritmalar ile benzerlikler göstermektedir. Çözüm uzayının kodlanması,

uygunluk fonksiyonunun belirlenmesi, mutasyon vb. ortak operatörler ve süreçler her iki yöntemde de ortaktır. Çaprazlama operatörü YBS'de kullanılmamaktadır. YBS, lokal optimumlara daha hızlı yönelmekte ancak bir genetik algoritma kadar global optimuma yakınsamayabilmektedir. YBS ise doğası gereği örüntü tanıma, bilgisayar güvenliği, ağ güvenliği ve dinamik iş-programlama vb. alanlarda genetik algoritma ve diğer optimizasyon tekniklerine göre daha iyi sonuçlar vermiştir[1,2].

YBS, yapay sinir ağlarında olduğu gibi hafıza ve öğrenme özelliklerine sahiptir . Her iki sistemde içerisinde çok sayıda ve farklı hücrelerden oluşmaktadır. Her iki sistem için de belirleyici bir eşik değeri vardır ve bu değer kesin tanıma sağlar. Ancak yapıları incelendiğinde farklı alanlarda benzer performanslar için üretildikler göze çarpmaktadır.

YBS ile bir problem çözümünün hangi aşamalardan oluştuğu şekil 1'de sırasıyla akış diyagramı şeklinde gösterilmektedir. Toplam 6 adımdan oluşan bu algoritma ile bir fonksiyonun maksimumunu bulmak veya bir bulanık denetleyicinin parametrelerini optimuma yaklaştırmak mümkündür.



Şekil 1: YBS Akış Diyagramı.

Yukarıdaki şekilde verilen akış diyagramını algoritmik adımlar halinde yazacak olursak;

1. Hafıza hücrelerinin bir alt kümesi(M) ile bir önceki iterasyondan kalan kümeyi(P_r) birleştirerek aday çözümler kümesi olan P'yi oluştur ($P = P_r + M$).
2. Oluşturulan popülasyondan benzerlik ölçümüne bağlı olarak n tane en iyi bireyi seç(P_n).
3. Bu n tene en iyi bireyi kopyalayarak C geçici popülasyonunu oluştur. Bu geçici popülasyondaki klonların sahip oldukları alanlar, antijene benzerliklikleri ile doğru orantılıdır.
4. C içerisindeki klonların her birini bir mutasyon oranına ve klonun benzerlik değerine bağlı olarak mutasyona tabi tut. Bu sayede olgunlaşmış antikor popülasyonunu(C^*) oluştur.
5. Hafıza kümesini, gelişmiş hücrelerden C^* dan tekrar seçerek oluştur(P kümesinin bazı hücreleri C^* nin bazı gelişmiş hücreleri ile yer değiştirebilir).
6. Farklılaşmayı sağlamak için popülasyonun düşük benzerlikli antikorlarını (d tane), yenileri ile değiştir.

Genetik algoritmalarda olduğu gibi YBS'de de sonuca etki eden parametreler mevcuttur:

- Popülasyonun boyutu
- Fitness fonksiyonunun uygunluğu ve çözünürlülüğü
- n değerinin seçimi
- Mutasyon oranının seçimi(genelde genetik algoritmaya göre çok daha düşük bir oran)
- Yenileme oranı
- Sonlandırma koşullarının belirlenmesi (Maksimum iterasyon sayısı veya amaçlanan doğruluk)

3. ÖNERİLEN YÖNTEM

Bu çalışmada örnek uygulama olarak yüksek dereceden doğrusal olmayan bir fonksiyon yaklaştırma seçildiği için böyle bir bulanık denetleyicinin giriş çıkışları ve bu giriş çıkışlara ait üyelik fonksiyonları çok açıktır. Burada optimum yaklaştırılması gereken olay kural tabanındaki kuralların sayısı ve içerikleridir.

$$y = \left(1 + x_1^{05} + x_2^{-1} + x_3^{-15}\right)^2 \qquad (1)$$

Denklem 1'de verilen fonksiyon daha önce de literatürde benzer çalışmaların performansını test etmek için kullanılmıştır[5,6]. x_1, x_2 ve x_3 değişkenleri [1-,5] aralığında kabul edilmiştir.

Genetik algoritmalarda olduğu gibi YBS'de de işe öncelikle arama uzayının kodlanması ile başlanır. Bir kural tablosu kurallar listesinden oluşur. Örnek uygulamadaki denetleyici sistemde her bir kural ise 3 tane giriş değişkeni, 1 tane çıkış değişkeni ve bir tane kural ağırlık değeri ve 1 tane de bağlaç operatör türünü içermektedir. Örneğin [1 2 2 1 0,5 1] kuralının anlamı;

EGER x1 is *mf1 and* x2 is *mf2 and* x3 is *mf2 THEN* y is *mf1* (weight = *0,5*)

şeklindedir. Bu şekildeki kuralların bir listesi oluşturmak demek nxm lik bir tablo oluşturmaktır. Örnek için m sayısının yukarıdaki gibi 6 olduğu görülmektedir. n ise kural tabanındaki kural sayısını göstermektedir. Mesela kural sayısını 10 ile sınırlarsak her bir klon 10x6 lık bir tablo ile ifade edilebilir. Popülasyon boyutunu da 10 olarak belirlersek popülasyon havuzumuz aslında 10x10x6 lık bir tablo anlamına gelmektedir. Tabi yukarıda gösterildiği gibi her bir satır, kendine özgü bir durumu ifade etmektedir. Genetik algoritmalarda olduğu gibi başlangıç popülsyonu rasgele üretilmektedir.

Algoritmanın 2.aşamasında popülasyonun yüzde kaçının hayata devam edeceğini belirlemek gerekmektedir. Bu ve daha sonraki seçme işlemleri fitness fonksiyonunun geri gönderdiği sonuca göre yapılacağından öncelikle fitness fonksiyonunu açıklamamız gerekir. Bir bulanık denetleyicinin fonksiyon yaklaştırma örneği için performansını hesaplamak aslında karmaşık bir iş değildir. Fonksiyon eğrisi üzerinde belirlenen sayıda örnek nokta alınır ve daha sonra bu noktalar ile bulanık denetleyicinin çıkışları arasındaki farklar hesaplanır. Bu aradaki fark ne kadar küçük ise denetleyici o kadar iyi yakınsamaktadır. Bu işlev bir fonksiyon halinde denklem 2'de gösterilmektedir.

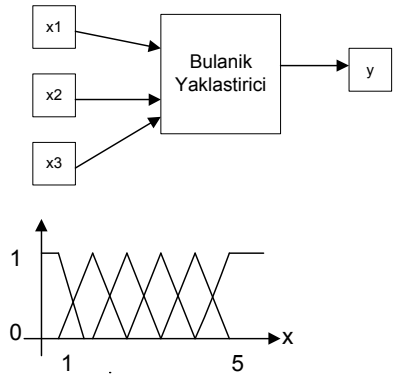
$$f(i) = S - (yreal - yfuzzy) \qquad (2)$$

S sabiti fitness değerinin sürekli pozitif çıkmasını sağlayacak bir değerdir ve maksimum uygunlukda f(i) = S olacaktır.

Bütün klonların yukarıdaki gibi fitness ları hesaplandıktan sonra aralarından %80 i hayatını ikinci iterasyonda da devam ettirecektir. Bu bireyler seçilirken f lerinin büyüklüğüne dikkat edilecektir. Geriye kalan % 20 si ise ikinci iterasyon başlamadan önce silinecek ve yerlerine yen bireyler rasgele oluşturulacaktır.

3.aşamada ise popülasyonun %80 lik kısmı geçici bir havuza alınıp burada kopyalama ve mutasyon işlemine tabi tutulmaktadır. Geçici havuzun boyutu doğal olarak artmakta ancak ikinci iterasyona başlamadan önce gelişen klonların fitness ları tekrar hesaplanarak hangilerinin eleneceği belirlenmektedir.

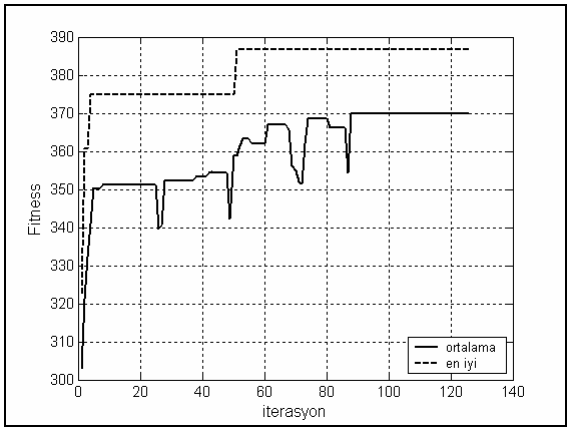
İterasyon sayısı başlangıçta belirtilen maksimum değere ulaştığında veya en iyi performansa sahip birey istenen doğrulukla fonksiyona yaklaştığında algoritma sonlandırılır. Aksi halde aynı aşamalara tekrar gerçekleştirilir.



Şekil 2 : Bulanık Denetleyicinin Yapısı

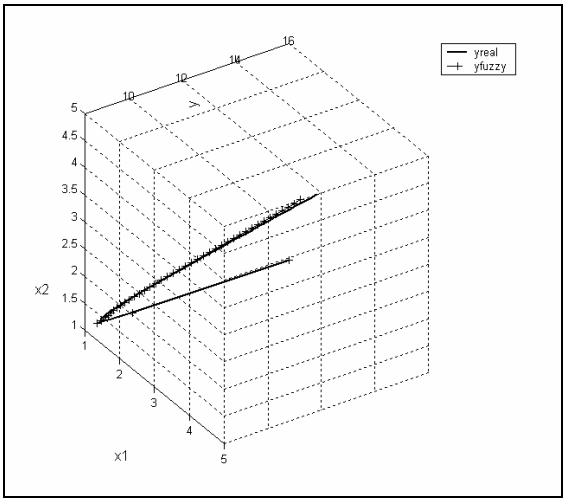
4. DENEYSEL SONUÇLAR

Önerilen yöntemin uygunluğunu göstermek için MATLAB ortamında şekil 2'de gösterilen bulanık denetleyici yapısı oluşturulmuş ve bu denetleyicinin kural tabanı YBS ile optimuma yaklaştırılmaya çalışılmıştır. Şekil 2'den de görüldüğü gibi bulanık denetleyici sistemin, 3 tane giriş değişkeni, 1 tane çıkış değişkeni ve bunlara ait 6'şar tane de üyelik fonksiyonları mevcuttur.



Şekil 3: Ortalama uygunluk ve en iyi uygunluk

Şekil3'de iterasyon adımları ilerledikçe ortalama uygunluk ve en iyi uygunluk değerlerinin de iyileştiği görülmektedir. Bu ifade bize YBS algoritmasının iyi tasarlandığını yani parametrelerinin düzgün seçildiğini göstermektedir.



Şekil 4 : Bulanık yaklaştırıcının çıkışı ile gerçek fonksiyonun karşılaştırılması.

Şekil 4'de algoritmanın 125 iterasyon çalıştırılmasından sonra elde edilen en iyi bulanık denetleyici ile gerçek fonksiyonun üst üste çizdirilmiş hali gösterilmektedir. Bu şekilden de anlaşılacağı gibi 125 gibi genetik algoritmaya göre küçük sayılacak bir iterasyon sayısında bile çok başarılı bir sonuç elde edilmiştir.

5. SONUÇ

Bu çalışmada bulanık denetleyici tasarımını otomatikleştirilmesi veya parametrelerinin optimuma yaklaştırılması için YBS nin kullanılabileceği öne sürülmüş ve test edilmiştir. Yapılan uygulama kompleks bir uygulama olmamakla birlikte yöntemin çalışabilirliğini vurgulamaktadır. YBS'nin Türkçe hazırlanmış nümerik bir örneğini oluşturacak olan bu çalışmanın, ülkemizdeki bir çok araştırmacıya da kaynak özelliği teşkil edeceği düşünülmektedir.

Ayrıca bu yöntem çok daha karmaşık ve farklı uygulamalara da adapte edilebilir. Örneğin yumuşak hesaplama ve yapay zeka tekniklerinde uzman deneyim veya tecrübesi gerektiren bir çok faklı algoritma veya teknik mevcuttur. Benzer şekilde bu teknikler için de bu yöntemin iyi sonuçlar verileceği ön görülmektedir.

Bu çalışmada sadece bulanık denetleyicinin kural tabanı sıfırdan oluşturulmuş, uygulama karmaşık olmadığı için üyelik fonksiyonları ve bunların parametreleri baştan belirlenmiştir. Aslında daha kompleks olan farklı bir uygulamada bulanık denetleyicinin hem kural tabanı hem de diğer parametrelerinin birlikte YBS ile oluşturulabileceği görülmüştür.

KAYNAKLAR

1. L. N. De Castro, J. Timmis, "Artificial Immune Systems: A novel paradigm to pattern recognition", Artificial Neural Networks in Patttern Recognition, SOCO-2002.
2. L. N. De Castro, F. J. Von Zuben, "The Clonal Selection Algorithm with Engineering Applicaitons", Workshop proceedings of GECCO'00.
3. L. N. De Castro, F. J. Von Zuben, "Artificial Immune Systems: Part-I Basic theory and applicaitons", Technical Report, 1999.
4. L. N. De Castro, F. J. Von Zuben, "Artificial Immune Systems: Part-II A Survey of Applicaitons", Technical Report, 2000.
5. J.J. Blake, L.P. Maguire, T.M. McGinnity, B. Roche, L.J. McDaid, "The implementation of fuzzy systems, neural Networks and fuzzy neural Networks using FPGA", Information Sciences, Elsevier-98.
6. S. Horikawa, T. Furuhashi, Y. Uchikawa, "On fuzzy modelling using fuzzy neural Networks with the backpropogation algorithm", IEEE Trans. Neural Networks 3, 1992.
7. Derviş Karaboğa, "Yapay Zeka Optimizasyon Algoritmaları", Atlas yayın-dağıtım,2004.