T.C.
AYDIN ADNAN MENDERES ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
MATEMATİK ANABİLİM DALI
2020-DR-008

DİFERANSİYEL DENKLEMLERİN YAPAY SİNİR AĞLARI İLE NÜMERİK ÇÖZÜMLERİ

MSc. İclal GÖR

Tez Danışmanı:
Dr. Öğr. Üyesi Korhan GÜNEL

AYDIN

T.C.
AYDIN ADNAN MENDERES ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ MÜDÜRLÜĞÜNE
AYDIN

Matematik Anabilim Dalı Doktora Programı öğrencisi İclal GÖR tarafından hazırlanan "Diferansiyel Denklemlerin Yapay Sinir Ağları ile Nümerik Çözümleri" başlıklı tez, 10.04.2020 tarihinde yapılan savunma sonucunda aşağıda isimleri bulunan jüri üyelerince kabul edilmiştir.

		Ünvanı, Adı Soyadı	Kurumu	İmzası
Başkan	:	Prof. Dr. Adnan MELEKOĞLU	Aydın Adnan Menderes Üniv.	
Üye	:	Doç. Dr. Mehmet Ali BALCI	Muğla Sıtkı Koçman Üniv.	
Üye	:	Doç. Dr. Refet POLAT	Yaşar Üniv.	
Üye	:	Dr. Öğr. Üyesi Rıfat AŞLIYAN	Aydın Adnan Menderes Üniv.	
Üye	:	Dr. Öğr. Üyesi Korhan GÜNEL	Aydın Adnan Menderes Üniv.	

Jüri üyeleri tarafından kabul edilen bu Doktora tezi, Enstitü Yönetim Kurulunun ..... sayılı kararıyla ..... tarihinde onaylanmıştır.

Prof. Dr. Gönül AYDIN
Enstitü Müdürü

T.C.
AYDIN ADNAN MENDERES ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ MÜDÜRLÜĞÜNE
AYDIN

Bu tezde sunulan tüm bilgi ve sonuçların, bilimsel yöntemlerle yürütülen gerçek deney ve gözlemler çerçevesinde tarafımdan elde edildiğini, çalışmada bana ait olmayan tüm veri, düşünce, sonuç ve bilgilere bilimsel etik kuralların gereği olarak eksiksiz şekilde uygun atıf yaptığımı ve kaynak göstererek belirttiğimi beyan ederim.

10.04.2020

İclal GÖR

ÖZET

DİFERANSİYEL DENKLEMLERİN YAPAY SİNİR AĞLARI İLE NÜMERİK ÇÖZÜMLERİ

İclal GÖR

Doktora Tezi, Matematik Anabilim Dalı
Tez Danışmanı: Dr. Öğr. Üyesi Korhan GÜNEL
2020, 91 sayfa

Bu çalışmada, birinci ve ikinci mertebeden lineer başlangıç değer problemleri, Dirichlet sınır koşulları içeren ikinci mertebeden lineer ve lineer olmayan diferansiyel denklemler ve birinci mertebeden lineer diferansiyel denklem sistemlerinin nümerik çözümleri ileri beslemeli tek ara katmanlı yapay sinir ağları kullanılarak elde edilmiştir.

Problemlerin çözümleri için modellenen sinir ağları, popülasyon tabanlı global optimizasyon metotlarından Parçacık Sürü Optimizasyonu, Kütle Çekim Arama Algoritması, Yapay Arı Koloni Algoritması ve Karınca Koloni Optimizasyonu kullanılarak eğitilmiştir. Ek olarak bahsi geçen optimizasyon algoritmaları Parçacık Sürü Optimizasyonu algoritması ile hibritlenerek çözümler elde edilmiştir. Tez çalışması boyunca incelenen optimizasyon yaklaşımlarından elde edilen izlenimler doğrultusunda, bilinen en iyi çözümün komşuluğunda üretilen hiper-küreleri kullanan yeni bir mutasyon operatörü tanımlanmıştır.

Deneysel çalışmalarda elde edilen bulgular, adi diferansiyel denklemlerin nümerik çözümlerini elde etmede yapay sinir ağı kullanımının geleneksel iterasyon tabanlı yöntemlere göre iyi bir alternatif olabileceğini göstermiştir. Yapay sinir ağlarının, çözüm aranan aralığın her noktasında tahmini bir değer üretebilme yetenekleri bu yöntemleri klasik yöntemlere göre tercih edilebilir hale getirmektedir.

Tezde önerilen yaklaşım, farklı sabit adım uzunlukları için değişik tipteki diferansiyel denklemler üzerinde test edilmiş ve diğer yöntemlerle kıyaslandığında genel olarak benzer veya çoğu zaman daha iyi sonuç vermiştir. Bununla birlikte, her tipte diferansiyel denklemi çözebilecek evrensel bir yapay sinir ağı modeli oluşturmanın olası olmadığı kanısına varılmıştır.

Anahtar Sözcükler: Yapay sinir ağları, Diferansiyel denklemler, Sınır değer problemi, Parçacık sürü optimizasyonu, Kütle çekimi arama algoritması, Yapay arı koloni algoritması, Karınca koloni algoritması.

ABSTRACT

NUMERICAL SOLUTIONS OF DIFFERENTIAL EQUATIONS BY ARTIFICIAL NEURAL NETWORKS

İclal GÖR

Ph.D. Thesis, Department of Mathematics
Supervisor: Asst. Prof. Korhan GÜNEL
2020, 91 pages

In this study, first and second order linear initial value problems, second order linear and nonlinear differential equations with Dirichlet boundary conditions and numerical solutions of systems of first order linear differential equation were obtained by using feed forward neural networks with a hidden layer.

The neural networks modeled for the solution of the problems were trained using population based global optimization methods as Particle Swarm Optimization, Gravitional Search Algorithm, Artificial Bee Colony Algorithm and Ant Colony Optimization. In addition, these optimization algorithms were hybridized with Particle Swarm Optimization algorithm and numerical solutions were obtained. A new mutation operator using the hyper-spheres generated in the neighborhood of the best solution known so far was identified from the examination of the optimization methods during the dissemination period.

Findings from experimental studies have shown that the use of artificial neural network to obtain numerical solutions of ordinary differential equations could be a good alternative to traditional iterative methods. The ability of generating an estimated value at every point of the solution makes artificial neural networks preferable to classical methods.

The proposed approach in the thesis has been tested on different types of differential equations for different fixed step size and has generally produces similar or often better results when compared to other methods. However, it is not possible to create a universal neural network model that can solve all types of differential equations.

Key Words: Artificial neural network, Differential equations, Boundary value problem, Particle swarm optimization, Gravitational search algorithm, Artificial bee colony algorithm, Ant colony optimization.

ÖNSÖZ

Tez çalışması boyunca vermiş olduğu desteklerinden; tanıştığımız andan itibaren kişiliği, yaklaşımı ve akademik başarısı ile örnek aldığım kıymetli hocam Dr. Öğr. Üyesi Korhan GÜNEL'e (Aydın Adnan Menderes Üniversitesi, Matematik Bölümü) en içten dileklerimle saygı ve teşekkürlerimi sunarım. Prof. Dr. Adnan MELEKOĞLU (Aydın Adnan Menderes Üniversitesi, Matematik Bölümü) ve Dr. Öğr. Üyesi Rıfat AŞLIYAN (Aydın Adnan Menderes Üniversitesi, Matematik Bölümü) başta olmak üzere doktora eğitimim boyunca emeği geçen hocalarıma teşekkür ederim. Ayrıca tez jüri üyelerinden Doç. Dr. Refet POLAT'a (Yaşar Üniversitesi, Matematik Bölümü) ve Doç. Dr. Mehmet Ali BALCI'ya (Muğla Sıtkı Koçman Üniversitesi, Matematik Bölümü) değerli katkılarından dolayı teşekkür ederim.

14011 nolu ÖYP projesi ile tez çalışmasını destekleyen Yüksek Öğretim Kurulu'na (YÖK), 2211-Yurt İçi Doktora Burs Programı kapsamında sağladığı destekten ötürü TÜBİTAK Bilim İnsanı Destekleme Daire Başkanlığı birimine teşekkür ederim.

Eğitimim için çok fedakarlık yapıp yanımda olan canım anneme ve bugünlere gelmemde en değerli desteğim olan canım babama, hayatıma girdiği andan itibaren mutluluk kaynağım biricik kızım Elif Hazel GÖR'e ve hiçbir zaman manevi desteğini esirgemeyen kıymetli eşim Makina Yüksek Mühendisi Abdulsamet GÖR'e teşekkürü bir borç bilirim.

Tez çalışması boyunca, biri ESCI kapsamında taranan dergide olmak üzere 2 makale yayınlanmış, 7 uluslararası ve 1 ulusal bildiri sunulmuştur.

İclal GÖR

İÇİNDEKİLER

SİMGELER DİZİNİ

ODE : Adi diferansiyel denklem (Ordinary Differential Equation)
YSA : Yapay sinir ağı (Neural Network)
SM : Atış metodu (Shooting Method)
FDM : Sonlu fark metodu (Finite Difference Method)
MLP : Çok katmanlı yapay sinir ağı (Multi Layer Perceptron)
RBF : Radyal tabanlı fonksiyon ağları (Radial Basis Function Network)
DFT : Yoğunluk fonksiyonel teorisi (Density Functional Theory)
LeNN : Legendre sinir ağı (Legendre Neural Network)
PSO : Parçacık sürü optimizasyonu (Particle Swarm Optimization)
PDE : Kısmi türevli diferansiyel denklem (Partial Differential Equation)
SQP : Ardışık kuadratik programlama (Sequential Quadratic Programming)
GSA : Kütle çekim arama algoritması (Gravitational Search Algorithm)
ABC : Yapay arı koloni algoritması (Artificial Bee Colony Algorithm)
ACO : Karınca kolonisi optimizasyonu (Ant Colony Optimization)
RNN : Kendinden tekrarlı yapay sinir ağları (Recurrent Neural Network)
LSTM : Uzun kısa süreli hafıza (Long Short Term Memory)

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

1. GİRİŞ

Sistemlerin tanımlanması günlük yaşam problemlerinin çözümü için önemlidir ve karmaşık sistemlerin modellenmesi için adi diferansiyel denklemler (ODE) sıklıkla kullanılan araçlardan biridir. Diferansiyel denklemlerde çözümün varlığı bilinmesine rağmen analitik çözümlerin elde edilmesinin mümkün olmadığı durumlarda, iteratif yöntemlerle çözüme nümerik olarak yaklaşılmaya çalışılır. Bununla birlikte Lee ve Kang'ın çalışmasıyla, diferansiyel denklemlerin nümerik çözümlerini elde edebilmek için, iterasyon bazlı klasik metotlara alternatif olarak Yapay Sinir Ağları (YSA) önerilmiştir [1].

Runge-Kutta, Atış metodu (SM) ve Sonlu Fark Metodu (FDM) gibi adi diferansiyel denklemleri çözmek için kullanılan geleneksel yöntemlerin ilk adımı her zaman tanım bölgesinin ayrıklaştırılmasıdır. Geleneksel yöntemlerle ODE'lerin sayısal çözümleri yalnızca aralık üzerindeki veya tanım bölgesindeki ayrıklaştırma düğümlerinde elde edilebilir. Düğüm noktaları haricinde çözüm elde edebilmek için genellikle interpolasyon veya eğri uydurma yöntemleri kullanılır. Problemi çözerken, nümerik yönteme ek olarak gerçekleştirilen interpolasyon işlemi hata miktarının artmasına neden olur. Dahası, tanım bölgesi ayrıklaştırıldığında, iterasyon sayısı arttıkça bu tip yöntemlerden kaynaklanan kümülatif hata miktarı da giderek çoğalmaktadır.

Yapay sinir ağlarında ise tanım bölgesinin ayrıklaştırılmasıyla elde edilen düğümler kullanılarak ağın eğitimi sağlanır. Ağın eğitimi tamamlandıktan sonra, tanım bölgesininin her noktası için yapay sinir ağı nümerik bir çözüm üretebilir. Yapay sinir ağları fonksiyonlara yaklaşmada küresel optimal çözümü teorik olarak garanti edememesine rağmen, hemen hemen optimal çözüme sıklıkla erişebilirler. Bundan dolayı, son on yılda diferansiyel denklemlerin sayısal çözümlerini elde etmek için yapay sinir ağları tercih edilen yöntemlerden biri olmuştur.

Literatürde yapay sinir ağları ile diferansiyel denklemlerin nümerik çözümlerinin elde edilmesi üzerine görülen ilk çalışmada, Hopfield Sinir Ağı modeli ile sonlu fark denklemlerinin çözümü gerçekleştirilmiştir [1]. Böylelikle yapay sinir ağları ile fonksiyonlara nümerik olarak yaklaşılabileceği gösterilmiştir. Bu tezde yapılan literatür taraması dört ana madde altında toplanmıştır:

(i) Diferansiyel denklemlerin türüne göre

(ii) Yapay sinir ağı modeline göre

(iii) Optimizasyon metotlarının kullanımına göre

(iv) Uygulama içermesine göre

Meade ve Fernandez (1994), ileri beslemeli yapay sinir ağları ile lineer diferansiyel denklemlerin nümerik çözümünü elde etmiştir [2]. Lagaris vd. bir diğer çalışmada (1998), YSA kullanarak başlangıç ve sınır değer problemini deneme çözümü yardımıyla çözmüşlerdir [3]. Çalışmada önerilen deneme fonksiyonu iki kısımdan oluşur. İlk kısım başlangıç ve sınır koşullarını sağlarken, ikinci kısım koşullardan bağımsız ve yapay sinir ağı çözümüne bağlı olacak şekilde tanımlanmıştır.

Li-yingi Hui ve Zhe-zhao (2007), kosinüs tabanlı fonksiyonlar kullanarak inşa edilmiş yapay sinir ağlarını başlangıç değer problemlerinin çözümünde kullanmıştır [4]. Bunun yanında iteratif yöntemlerden Euler ve Heun metotlarını da kullanarak elde ettikleri sonuçları önerdikleri metot ile karşılaştırmışlardır. Fojdl ve Brause (2008), adi diferansiyel denklemlerin çözümü için, toplam en küçük karesel hataya dayalı öznel hata miktarını en küçüklemeye çalışmışlardır [5].

McFall ve Mahan (2009), sınır koşullarına sahip iki ve üç boyutlu lineer ve lineer olmayan diferansiyel denklemlerin çözümü için YSA kullanımını önermişlerdir [6]. Otadi ve Mosleh (2011), gradyan düşüm algoritması ile eğitilmiş iki katmanlı ileri beslemeli yapay sinir ağı aracılığıyla Riccati diferansiyel denklemini çözmüş ve bazı klasik nümerik metotlar ile karşılaştırmıştır [7]. Bir diğer çalışmada ise Kumar ve Yadav (2011), Çok katmanlı Yapay Sinir Ağı (MLP) modeli ile

Radyal Tabanlı Fonksiyon Ağların (RBF) diferansiyel denklemlerin çözümünde kullanıldığı çalışmaları tanıtan geniş bir çalışma önermiştir [8].

ODE'lerin nümerik çözümlerinin elde edilmesiyle beraber, çalışmalar kısmi türevli diferansiyel denklem (PDE) çözümü üzerine odaklanmıştır. Valasoulis (2002), ODE ve PDE çözümü için iki bölüm içeren çözüm önerisi ve iki dijital sinyal işlemcisi ile bir donanım platformundaki uygulamasını incelemiştir [49]. Sun vd. (2003), PDE'ler için YSA kullanarak sayısal olarak çözebilmek için yeni bir sinyal işleme yöntemi kullanmıştır [50]. Bununla birlikte, Poisson denklemi için çözümün tek ve sürekli olduğunu göstermiştir. Beidokhti ve Malek (2009), keyfi başlangıç ve sınır koşullarını içeren PDE sistemlerinin çözümü için YSA'ları tercih etmiştir [51]. Alharbi (2010), dalga, ısı, Poisson ve difüzyon gibi klasik PDE ve PDE sistemi çözümü için, standart sayısal yöntem, sonlu fark ve Hopfield sinir ağının birleşimiyle önerilen yaklaşımı kullanmıştır [40]. McFall (2013), Naive-Stokes eşitliği dahil olmak üzere, üç farklı termal sıvı problem çözümü için YSA'lar kullanmıştır [52]. Ayrıca çalışmada kısmi türevli diferansiyel denklemlerin bağlı sistemlerini çözerken YSA'nın uzunluk faktörü yaklaşımını seçmiştir. Rudd vd. (2014), lineer olmayan eliptik ve parabolik PDE'lerin çözümü için kısıtlı geri yayılım algoritmasını önermiştir [53]. Rudd ve Ferrari (2015), kısıtlamalı integrasyon yaklaşımı ile PDE'nin çözümünde YSA kullanmıştır [54]. Zjavka ve Pedrycz (2016), genel kısmi diferansiyel eşitliklerinin oluşturulmasında polinomları ve sinir ağlarını kullanarak fonksiyonlara yaklaşmıştır [55]. Zjavka vd. (2016), yeni bir diferansiyel polinom ağları önererek, bilinmeyen sistem ya da desenin modellenmesinde kesirli türev kullanıp genel PDE'leri çözmüştür. [56]. Mall ve Chakraverty (2017), Chebyshev sinir ağı ile eliptik PDE'leri çözmüştür [57].

Diferansiyel denklemler için literatürde farklı türlerin önerilmesiyle beraber, bu tipteki diferansiyel denklemler için YSA çözüm önerisi verilmesi planlanmış ve

bununla ilgili çalışmalar literatüre eklenmiştir [9–13, 15–17, 31]. Bu çalışmalarda Bratu tipli, integro diferansiyel denklem, Volterra-Fredholm integro- diferansiyel denklem, Lane-Emdan, Emden-Fowler gibi denklemlerin çözümü elde edilmiştir. Bunlara örnek olarak verilebilecek çalışmalardan ilkini Lagaris vd. (1997), Yapay Sinir Ağı (YSA) kullanarak diferansiyel ve integrodiferansiyel denklemler için özdeğer problemlerinin çözümünü araştırmıştır [18]. Bu yaklaşım için kuantum mekanikte bilinen Marse potansiyeli için Schrödinger eşitliği kullanılmıştır. Kourakos ve Mantoglou (2009), pompalama optimizasyon problemi için modüler nörol alt sinir ağlarını kullanmıştır [19]. Çalışmada oluşturulan ağlar optimizasyon sırasında uyarlanabilir şekilde eğitilmiştir.

Caetano vd. (2010), çift harmonik asilatör ve hidrojen atomu olmak üzere iki sistem için yoğunluk fonksiyonel teorisi kullanarak (DFT) elde edilen denklemleri çözmüştür [20]. Baymani vd. (2010), Stokes problemini çözmek için ileri beslemeli YSA kullanarak Poisson denklemine dönüştürüp çözmüştür [21]. Mall ve Chakraverty (2014), çalışmalarında Cheyshev YSA modeli ile özel tipteki Lane-Emdan diferansiyel denklemlerin çözümünü elde etmiştir [22]. Anastassi (2014), YSA ile Runge-Kutta metodunu oluşturarak ODE'lerin nümerik çözümlerini sunmuştur [23]. Raja ve Samar (2014), iç nokta metodu ile eğitilmiş üç farklı YSA kullanarak lineer olmayan manyetohidrodinamik Jeffery-Hamel akış problemini çözmüştür [24]. Raja ve Ahmad (2014), Bratu tipteki diferansiyel denklemin çözümünde YSA kullanmıştır [25]. Önerilen metotta yer alan YSA için, logaritmik sigmoid, radyal tabanlı ve tanjant sigmoid fonksiyonlarını tercih etmiştir. Raja vd. (2015), lineer olmayan quadratik Riccati diferansiyel denklemi çözümü için yeni bir metot kullanmıştır [26]. Mall ve Chakraverty (2015), Emden-Fowler tipli denklemlerin tekil başlangıç değer problemlerinin çözümü için Chebyshev Sinir Ağlarını kullanmıştır [27]. Mall ve Chakraverty bir diğer çalışmada ise (2016), Van-der Pol-Puffing asilatör denklemini çözmek için Hermit polinom bazlı fonksiyonel link YSA kullanmıştır [28]. Masood (2017), yakıt

tutuşturma modellerinde, elektrik ileten katılarda ve ısı transfer çalışmalarında yaygın olarak kullanılan Bratu tipi denklem için Mexican Hat Wavelet YSA kullanarak çözüm önermiştir [29]. Kashkaria (2017), mixed koşullar altında lineer olmayan Volterra-Fredholm integro diferansiyel denklemin çözümünde YSA kullanmıştır [30]. Raja vd. (2017), lineer olmayan Pantograph sistemlerin çözümünde ileri beslemeli YSA kullanmıştır [31]. Sinchev vd. (2017), aktivasyon fonksiyonu Gauss dağılımı olan Radyal tabanlı YSA ile 2 boyutlu Naive-Stokes denklemini çözmüştür [32]. Mall ve Chakraverty (2016), başlangıç ve sınır değer problemlerini çözmek için tek katmanlı Legendre Sinir Ağı (LeNN) modeline dayanan yeni bir yöntem geliştirmiştir [33]. Önerdikleri yaklaşımda bir Legendre polinom tabanlı fonksiyonel bağlantı içeren YSA geliştirmiştir. Sabir vd. (2018), tekil Thomas-Fermi sistemlerinin çözümü için ileri beslemeli YSA'lar kullanmıştır [34].

Diferansiyel denklemlerin özel tiplerinden biri olan kesirli diferansiyel denklemlerin nümerik çözümleri için önerilen çalışmalar farklı tipte YSA'lar kullanılarak elde edilmiştir. Raja vd. (2010), kesirli mertebeden lineer olmayan Riccati diferansiyel denkleminin çözümü için stokastik bir teknik önermiştir [35]. Rostami ve Jafarian (2017), yüksek mertebeden lineer kesirli diferansiyel denklem çözümü için YSA'ları tercih etmiştir [36]. Jafarian vd. (2017), sınırlı bir alanda, kesirli mertebeli başlangıç değerli diferansiyel denkleminin yaklaşık seri çözümü için YSA kullanmıştır [17]. Aguilar vd. (2018), kesirli gecikmeli diferansiyel denklem çözümü için ileri beslemeli YSA ile Adams-Bashforth-Moultan yöntemini karşılaştırmıştır [37]. Jafarian vd. (2018), maliyet fonksiyonu olarak genelleştirilmiş sigmoid fonksiyonu tercih edilen üç katmanlı ileri beslemeli YSA ile lineer kesirli adi diferansiyel denklemin çözümünü elde etmiştir [38]. Pakdaman vd. (2017), kesirli diferansiyel denklemi için fonksiyon analizi yaklaşımını kullanarak YSA ile çözüm elde etmiş ve optimizasyon yaklaşımı ile YSA'nın parametrelerini ayarlamıştır [39].

Yapay sinir ağları teorik olarak geliştirilmiş ve geliştirilen ağların kullanılmasıyla birlikte diferansiyel denklemlerin nümerik çözümü için kullanılabilir oldukları farklı çalışmalarda araştırılmıştır [16, 24, 27–29, 32, 33, 40–43]. Yapay sinir ağlarından, hibrit sinir ağları, Hopfield, Legendre, Radyal tabanlı gibi ağlar çözüm elde etmek için kullanılmıştır. Bu çalışmalardan birini Yekrangi (2011) mekanikte bilinen basit sarkaç probleminin çözümü için hibrit sinir ağları ve Parçacık Sürü Optimizasyonu (PSO) kullanarak yapmıştır [44]. Ayrıca bu yöntem karmaşık sarkaç problemleri için de uygulanabilir olduğunu idda etmiştir. Alharbi (2012), beyindeki nöronların aktivitesini modellemek için kullanılan sinir alan denklemlerinin, denklem sistemi olarak modelledikten sonra tekrarlayan YSA ile çözümüne ulaşmıştır [45]. Chakraverty ve Mall (2014), önerdikleri metotta tek girdi ve çıktı modeli içeren regresyon tabanlı sinir ağı kullanarak iki terim toplamı olarak yazılan deneme fonksiyonu ile birinci, ikinci ve dördüncü mertebeden diferansiyel denklemi çözmüştür [46]. Zjavka ve Snasel (2015), benzerlik modeli analiz yöntemleri ve polinom sinir ağlarını kullanarak genel diferansiyel denklemi çözmüştür [42]. Bir diğer çalışmada Rizaner (2018), başlangıç değer probleminin yaklaşık çözümleri için Radyal tabanlı ağlar kullanmıştır [47]. Tan vd. (2018), başlangıç ve sınır değer problemlerini çözmek için modifiye geri yayılımlı çok katmanlı YSA'yı tercih etmiştir [48].

Yapay sinir ağı parametrelerinin belirlenmesinde kullanılan optimizasyon metodu da çözümü etkiler. Dolayısıyla tez konusuyla ilgili çalışmalar optimizasyon metotlarının da eklenmesiyle beraber iyileştirilmeye çalışılmıştır [39, 44]. Malek ve Beidokhti (2006), Nelder-Mead optimizasyon tekniği ile eğitilen ileri beslemeli ağları kullanarak hibrit bir metot inşa etmiş ve önerilen metot ile yüksek mertebeli diferansiyel denklemleri çözmüştür [58]. Vinod vd. (2009), biyolojik bozunma sürecini tanımlayan modelin denklemlerini ileri beslemeli YSA ve genetik algoritma kullanarak çözmüştür [59]. Aquiono vd. (2010), çalışmalarında doğrusal ve karesel programlama problemlerini çözen iki fazlı optimizasyon

sinir ağı kullanmıştır [60]. Bu ağlar ileri Lagrange enerji fonksiyonunun bir dönüşümünden elde edilen bir dizi diferansiyel denklemin çözümüne dayanır. Raja vd. (2013), genetik algoritma ve iç nokta algoritmaları ile optimize edilmiş 2-boyutlu Bratu denklemlerinin çözümünde ileri beslemeli YSA'lar kullanarak stokastik bir teknik geliştirmiştir [9]. Raja (2014), ikinci mertebeden Pantograph fonksiyonel diferansiyel denklem eşitliklerinin çözümünü, YSA, benzetimli tavlama (simulated annealing), örüntü arama (pattern search), genetik algoritma, aktif küme (active-set) algoritması ve bu metotların hibrit modellerini kullanarak bulmuştur [10]. Bir diğer çalışmada Raja vd. (2014), ileri beslemeli YSA modelinin kullanılmasıyla 2-boyutlu Bratu eşitlikleri çözerek, PSO ve sıralı karesel programlama (SQP) optimizasyon metotları ile optimize etmiştir [11]. Raja (2014), Troesch probleminin çözümünü YSA ve optimizasyon metodu aracılığıyla elde etmiştir [12]. Çalışmalarında PSO, aktif-küme ve bu iki algoritmanın hibitlenmesi ile oluşturulan optimizasyon yaklaşımını kullanmıştır. Raja (2014), Troesch sınır değer probleminin çözümü için gözetimsiz öğrenmeye dayalı ileri beslemeli YSA kullanmıştır [13]. Çalışmada global arama için genetik algoritma ve desen arama, yerel arama için ise iç nokta metodunu tercih etmiştir. Raja (2014), doğrusal olmayan Jeffery-Hamel problemi için genetik algoritma ve sıralı karesel programlama ile optimize edilmiş ileri beslemeli YSA yardımıyla yaklaşık çözüm elde etmiştir [24]. Raja vd. (2015), sıralı karesel programlama ile optimize edilmiş üç denetimsiz YSA ile Painleve denklemini çözmüştür [41]. Raja (2016), elektrik ileten katıların ve diğer çeşitli fiziksel durumların modellenmesinde ortaya çıkan Bratu tipli denklemleri çözmek için biyolojiden esinlenen (bio-inspired) bir yaklaşım geliştirmiştir [31]. Majeed vd. (2017), lineer olmayan Troesch sistemi çözümünde genetik algoritma ve SQP ile optimize edilmiş Marlet Dalgacık YSA'yı tercih etmiştir [15]. Khan (2017), Legendre polinomları ve benzetimli tavlama kullanarak önerilen Legendre benzetimli tavlama YSA ile lineer olmayan Lane-Emdan denklemleri çözmüştür [16]. Ahmad vd. (2017), Falkner-Skan

denklemlerinin çözümü için aktif-set, SQP, karesel programlama ve genetik algoritmanın hibritlenmesi ile oluşan metodun kullanılmasıyla optimize edilmiş, logaritmik sigmoid fonksiyonlu YSA'ları tercih etmiştir [43]. Raja vd. (2018), lineer olmayan ısı iletim modelinin sayısal olarak çözümü için YSA, genetik algoritma, aktif-küme ve hibrit yaklaşımlarını kullanmıştır [61]. Bir diğer çalışmada ise Raja vd. (2018), lineer olmayan RC devresi için oluşturulan birinci mertebeden diferansiyel denklemi genetik algoritma, SQP ve bu metotların hibritlenmesi ile optimize edilmiş YSA kullanarak çözmüştür [62]. Raja vd. (2018), benzer bir çalışmayı lineer olmayan tekil sınır değer probleminin çözümü için nöro evrim modeli tasarlayarak gerçekleştirmiştir [63]. Önerdikleri modele göre, genetik algoritma ve SQP kullanarak YSA'nın parametre ayarı yapılmıştır. Ahmad vd. ise (2018), indüksiyon motor modelinde oluşturulan 5. mertebeden sınır değer probleminin çözümünde optimizasyon metotları ile optimize edilmiş YSA modeli kullanmıştır [64].

Diferansiyel denklemlerin nümerik çözümünü elde etmek için YSA'ların tercih edildiği bazı çalışmalarda farklı uygulama alanlarında çalışmalara odaklanılmış ve uygulama alanına göre modelleme yapılmıştır [19, 44, 45, 49, 50, 52, 59–64]. Dua (2011), çalışmasında büyük ve gürültülü veri setleri ve doğrusal olmayan modeller üzerine problem çözmede etkili olan bir YSA yaklaşımını parametre tahmininde kullanmıştır [65]. Ling vd. (2013), sistem biyolojisi alanında gen ve protein etkileşiminden oluşan hücresel ağların kontrolü için tekrarlayan YSA önermiştir [66]. Önerilen yaklaşımda kullanılan ağ, bir sinyalleme için moleküler etkileşimleri temsil eden bir adi diferansiyel denklemi çözmektedir. Xiangdong ve Guanghua (2009), doğrusal olmayan çok seviyeli programlama problemlerinin çözümünde YSA'ların dinamik davranışını baz alan bir yaklaşım geliştirmiştir [67]. Chudzik ve Minkina (2009), çalışmalarında ısı yalıtım problemlerinin termal parametrelerini test etmek için sıcak problu bir ölçüm sistemi üzerine yoğunlaşmıştır [68]. Ayrıca ısı yalıtım malzemesinin bir örneğinde ters ısı aktarım

probleminin çözümünde YSA'nın kullanılabilirliği üzerine odaklanmıştır. Aquiono (2015), ileri beslemeli geri yayılımlı üç tabakalı YSA ile adi diferansiyel denklem yapısına sahip karışım tank problemini çözmüştür [69]. Bu çalışmalar farklı yaklaşımların da eklenmesiyle beraber iyileştirilmeye devam etmektedir.

Tezde birinci ve ikinci mertebeden farklı tipte diferansiyel denklemler ve birinci mertebeden lineer diferansiyel denklem sistemlerinin nümerik çözümlerini elde etmek için ileri beslemeli tek ara katmanlı yapay sinir ağları, popülasyon tabanlı global optimizayon metotlarından Parçacık Sürü Optimizasyonu, Kütle Çekim Arama Algoritması, Yapay Arı Koloni Algoritması, Karınca Koloni Optimizasyonu ve bu metotların hibritlenmesiyle oluşturulan metotlar kullanılarak eğitilmiştir. Ayrıca tezde, bilinen en iyi çözümün komşuluğunda oluşturulan hiper-küreler kullanılarak mutasyon operatörü yardımıyla çözüm iyileştirilmeye çalışılmıştır.

Tez dört bölümden oluşmaktadır. Tezin ikinci bölümünde öncelikle ileri beslemeli yapay sinir ağları ile diferansiyel denklemlerin nümerik çözümünü elde etme probleminin nasıl bir optimizasyon problemine dönüştürüldüğü gösterilecektir. Ardından bu optimizasyon probleminin çözümü için tezde kullanılan popülasyon tabanlı global optimizasyon yaklaşımlarından kısaca bahsedilecektir. Ek olarak, bahsi geçen algoritmaların hibritlenmesi ve yeni bir mutasyon operatörü tanımlanarak elde edilen nümerik çözümlerin nasıl geliştirilebileceği üzerinde durulacaktır. Tezin üçüncü bölümünde farklı tipte başlangıç veya sınır koşulları içeren diferansiyel denklemler ve diferansiyel denklem sistemlerinin çözümü için örnekler sunulmuştur. Yapılan deneysel çalışmalarda elde edilen sonuçlar birbiriyle karşılaştırılmış ve tezin dördüncü bölümünde elde edilen bu sonuçlar tartışılmıştır.

2. MATERYAL VE METOT

Bu bölümde ileri beslemeli yapay sinir ağları ile diferansiyel denklemlerin nümerik çözümünden bahsedilecektir. Bu amaçla başlangıç ve sınır değer problemlerine ek olarak diferansiyel denklem sistemlerinin nümerik çözümleri incelenecektir. Ayrıca tezde kullanılan popülasyon tabanlı global optimizasyon algoritmalarından parçacık sürü optimizasyonu, kütle çekim arama algoritması, yapay arı koloni algoritması ve karınca koloni optimizasyonu kısaca açıklanacaktır.

2.1. İleri Beslemeli Yapay Sinir Ağları ile Diferansiyel Denklemlerin Nümerik Çözümleri

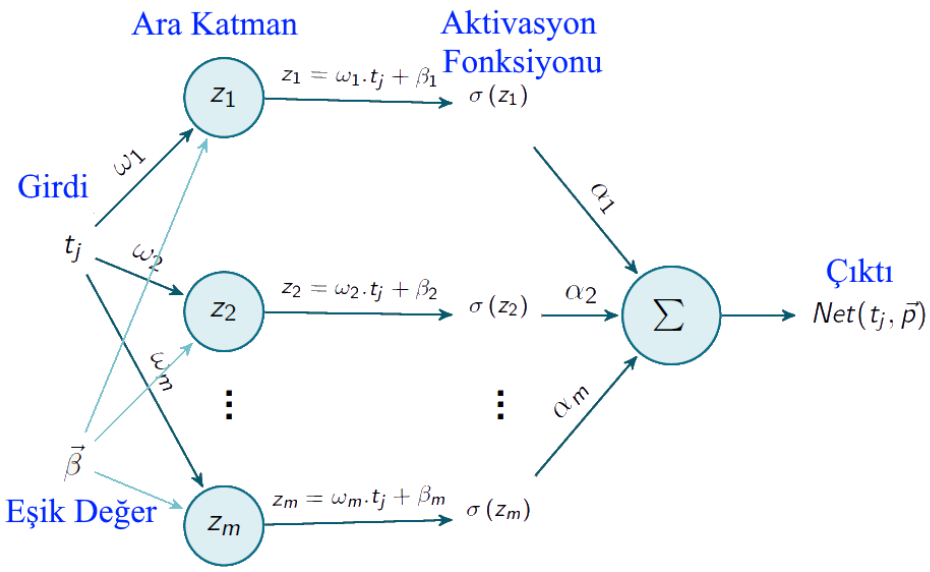
Tanım 2.1. Birinci mertebeden diferansiyel denklem için başlangıç değer problemi $t \in [a,b]$ olmak üzere $t = t_0 = a$ için,

$$\begin{cases} y'(t) = f(t, y(t)), \\ y(t_0) = y_0 \end{cases} \tag{2.1}$$

olarak tanımlıdır.

Tanım 2.1 ile verilen başlangıç değer problemini ele alalım. n toplam girdi sayısı ve m ara katmandaki nöron sayısı olmak üzere, $\vec{\alpha}, \vec{\beta}$ ve $\vec{w} \in \mathbb{R}^m$ olacak şekilde ileri beslemeli yapay sinir ağının parametreleri $\vec{p} = \vec{p}(\vec{\alpha}, \vec{w}, \vec{\beta})$ olsun. $j = 1, 2, \ldots, n$ için, $t_j \in [a,b]$ noktasında yapay sinir ağının çıktısı $\vec{p}$ parametre değerleri için $Net(t_j, \vec{p})$ ile gösterilir. Genel olarak $h > 0$ sabit adım uzunluğu için $t_j = a + j.h$ olacak şekilde $[a,b]$ aralığının parçalanışından elde edilen noktalar ağın eğitimi için kullanılır.

Verilen başlangıç değer probleminin çözümü için yapay sinir ağı çıktısına bağlı olarak başlangıç koşullarını sağlayan deneme fonksiyonu $y_T(t, \vec{p}) = y_0 + (t - t_0) Net(t, \vec{p})$ olarak ifade edilebilir [2].



Şekil 2.1. Adi diferansiyel denklemlerin nümerik çözümleri için oluşturulan ileri beslemeli yapay sinir ağının topolojik yapısı.

Şekil 2.1'de görüldüğü üzere; m ara katmandaki nöron sayısını belirtmek üzere, $i = 1,2,3,\ldots,m$ için z_i ara katmandaki i. nöronu temsil eder. t_j girdisi için z_i nöronunun çıktısı, w_i ağırlık ve β_i eşik değer olmak üzere; $z_i = w_i t_j + \beta_i$ olarak hesaplanır. Her bir nöronun çıktısı aktivasyon fonksiyonu ile işleme tabi tutulduktan sonra ağırlıklandırılır. Ağırlıklandırılmış çıktıların toplamı yapay sinir ağının çıktısı olarak belirlenir. Yani, $\vec{p} = \vec{p}(\vec{\alpha}, \vec{w}, \vec{\beta})$ ağın bilinmeyen parametrelerini göstermek üzere, $Net(t_j, \vec{p}) = \sum_{i=1}^{m} \alpha_i \sigma(z_i)$ ile elde edilen ağırlıklandırılmış toplam değeri ileri beslemeli yapay sinir ağının çıktısını oluşturur.

$Net(t_j, \vec{p})$'de verilen aktivasyon fonksiyonu olarak $\sigma(z_i) = \dfrac{1}{1+\exp(-z_i)}$ sigmoid fonksiyonu tercih edilmiştir. Sigmoid fonksiyonu dışında farklı bir aktivasyon fonksiyonu kullanılabilir. Literatürde en fazla tercih edilen aktivasyon fonksiyonları hiperbolik tanjant ve sigmoid fonksiyonlarıdır. Diğer yandan genel olarak aktivasyon fonksiyonunun farklı seçilmesi, sonucu fazlaca değiştirmez. Bunun temel nedeni bilinmeyen parametrelere dayalı olarak hesaplanan ağın çıktısından kaynaklı hatanın minimize edilerek parametre değerlerinin

ayarlanmasıdır. Dolayısıyla hangi aktivasyon fonksiyonunun kullanıldığından ziyade ağın parametrelerinin nasıl belirlendiği daha önemlidir.

Başlangıçta rassal olarak belirlenen parametre değerleriyle ağın çıktısı hesaplanır. Elde edilen çıktı sonucunda ağda hata oluşacaktır. Oluşan hatayı minimize etmek amacıyla, genellikle geri yayılım algoritması kullanılır ve $\vec{p}$ bilinmeyen parametre değerleri güncellenerek oluşan hata tüm ağa yayılır. İleri beslemeli yapay sinir ağında hata, n girdi sayısı olmak üzere; $E = \frac{1}{2}\sum_{j=1}^{n}(d_j - y_j)^2$ kullanılarak hesaplanmaktadır. Verilen eşitlikte, d_j değeri t_j girdisi için ağın üretmesi beklenen çıktı değeri ve y_j ise ağın gerçekte ürettiği çıktı değeridir. Öte yandan, adi diferansiyel denklem çözümünde oluşan hatanın hesaplanması için öncelikle maliyet fonksiyonu, girdi değeri t_j için $e_j = \frac{1}{2}\left(\frac{\partial y_T}{\partial t_j} - f(t_j, y_j)\right)^2$ eşitliği aracılığıyla hesaplanır. Buna bağlı olarak her girdi için oluşan hata değerlerinin toplamı olan maliyet fonksiyonu $E = \sum_{j=1}^{n} e_j = \sum_{j=1}^{n} \frac{1}{2}\left(\frac{\partial y_T}{\partial t_j} - f(t_j, y_j)\right)^2$ ile verilir. Amacımız diferansiyel denklemi çözmek için maliyet fonksiyonunu minimize etmektir. μ genellikle 0 ile 1 arasında değer alan öğrenme katsayısı olmak üzere; $i = 1, 2, \ldots, n$ için α_i, w_i ve β_i değerlerini güncellemek amacıyla Gradyan Düşüm Algoritması (Gradient Descent Algorithm) kullanılır. Bu yaklaşımda paremetre değerleri sırasıyla $\alpha_i = \alpha_i - \mu\frac{\partial E}{\partial \alpha_i}$, $w_i = w_i - \mu\frac{\partial E}{\partial w_i}$ ve $\beta_i = \beta_i - \mu\frac{\partial E}{\partial \beta_i}$ olacak şekilde güncellenir.

Parametre değerleri güncellenirken maliyet fonksiyonunun bilinmeyen parametrelere göre kısmi türevleri $i = 1, 2, \ldots, m$ için

$$\frac{\partial y_T}{\partial t_j} = Net(t_j, \vec{p}) + (t_j - t_0)\frac{\partial Net(t_j, \vec{p})}{\partial t_j},$$

$$\frac{\partial Net(t_j, \vec{p})}{\partial t_j} = \sum_{i=1}^{m} \alpha_i w_i \sigma(z_i)(1 - \sigma(z_i))$$

ve

$$\frac{\partial^2 Net(t_j, \vec{p})}{\partial t_j^2} = \sum_{i=1}^{m} \alpha_i w_i^2 \sigma(z_i)(1 - \sigma(z_i))(1 - 2\sigma(z_i))$$

olmak üzere sırasıyla 2.2, 2.3 ve 2.4 eşitliklerinde olduğu gibi sıfıra eşitlenir.

$$\frac{\partial E}{\partial \alpha_i} = \sum_{j=1}^{n} \left(\frac{\partial y_T}{\partial t_j} - f(t_j, y_j) \right) \left(\frac{\partial^2 y_T}{\partial \alpha_i \partial t_j} - \frac{\partial f(t_j, y_j)}{\partial \alpha_i} \right) = 0 \tag{2.2}$$

$$\frac{\partial E}{\partial \beta_i} = \sum_{j=1}^{n} \left(\frac{\partial y_T}{\partial t_j} - f(t_j, y_j) \right) \left(\frac{\partial^2 y_T}{\partial \beta_i \partial t_j} - \frac{\partial f(t_j, y_j)}{\partial \beta_i} \right) = 0 \tag{2.3}$$

$$\frac{\partial E}{\partial w_i} = \sum_{j=1}^{n} \left(\frac{\partial y_T}{\partial t_j} - f(t_j, y_j) \right) \left(\frac{\partial^2 y_T}{\partial w_i \partial t_j} - \frac{\partial f(t_j, y_j)}{\partial w_i} \right) = 0 \tag{2.4}$$

Verilen kısmi türevlerde geçen $\frac{\partial^2 y_T}{\partial \alpha_i \partial t_j}$, $\frac{\partial^2 y_T}{\partial \omega_i \partial t_j}$ ve $\frac{\partial^2 y_T}{\partial \beta_i \partial t_j}$ değerleri sırasıyla $\frac{\partial Net(t_j, \vec{p})}{\partial \alpha_i} = \sigma(z_i)$ ve $\frac{\partial^2 Net(t_j, \vec{p})}{\partial \alpha_i \partial t_j} = w_i \sigma(z_i)(1 - \sigma(z_i))$ olmak üzere;

$$\frac{\partial^2 y_T}{\partial \alpha_i \partial t_j} = \frac{\partial Net(t_j, \vec{p})}{\partial \alpha_i} + (t_j - t_0) \frac{\partial^2 Net(t_j, \vec{p})}{\partial \alpha_i \partial t_j}, \tag{2.5}$$

$\frac{\partial Net(t_j, \vec{p})}{\partial \beta_i} = \alpha_i \sigma(z_i)(1 - \sigma(z_i))$ ve $\frac{\partial^2 Net(t_j, \vec{p})}{\partial \beta_i \partial t_j} = \alpha w_i \sigma(z_i)(1 - \sigma(z_i))(1 - 2\sigma(z_i))$ olmak üzere;

$$\frac{\partial^2 y_T}{\partial \beta_i \partial t_j} = \frac{\partial Net(t_j, \vec{p})}{\partial \beta_i} + (t_j - t_0) \frac{\partial^2 Net(t_j, \vec{p})}{\partial \beta_i \partial t_j} \tag{2.6}$$

ve $\frac{\partial Net(t_j, \vec{p})}{\partial w_i} = \alpha t_i \sigma(z_i)(1 - \sigma(z_i))$ ve $\frac{\partial^2 Net(t_j, \vec{p})}{\partial w_i \partial t_j} = \alpha w_i \sigma(z_i)(1 - \sigma(z_i))(1 - 2\sigma(z_i))$ olmak üzere;

$$\frac{\partial^2 y_T}{\partial w_i \partial t_j} = \frac{\partial Net(t_j, \vec{p})}{\partial w_i} + (t_j - t_0) \frac{\partial^2 Net(t_j, \vec{p})}{\partial w_i \partial t_j} \tag{2.7}$$

şeklinde hesaplanır.

Tanım 2.2. İkinci mertebeden diferansiyel denklem için başlangıç değer problemi,

$$\begin{cases} y''(t) = f(t, y(t), y'(t)), \quad t \in [a, b] \\ y(a) = A \\ y'(a) = B \end{cases} \tag{2.8}$$

olarak tanımlıdır.

Tanım 2.8 ile verilen ikinci mertebeden başlangıç değer problemi için önerilen deneme fonksiyonu, $y_T(t,\vec{p}) = A + (t-a)B + \frac{1}{2}(t-a)^2 Net(t,\vec{p})$ olarak tanımlıdır. Birinci mertebeden diferansiyel denklemler için yapılan işlemlere benzer şekilde diferansiyel denklemi çözmek için 2.9 eşitliği ile verilen maliyet fonksiyonu minimize edilir.

$$E = \frac{1}{2}\sum_{j=1}^{n}\left\{\frac{\partial^2 y_T(t_j)}{\partial t_j^2} - f\left(t_j, y_T(t_j), \frac{\partial y_T(t_j)}{\partial t_j}\right)\right\}^2 \tag{2.9}$$

Maliyet fonksiyonunda yer alan deneme fonksiyonunun girdi değerlerine bağlı kısmi türevleri 2.10 ve 2.11 eşitlikleri aracığıyla elde edilir.

$$\frac{\partial y_T(t_j)}{\partial t_j} = B + (t_j - a)Net(t_j,\vec{p}) + \frac{1}{2}(t-a)^2\frac{\partial Net(t,\vec{p})}{\partial t_j} \tag{2.10}$$

$$\frac{\partial^2 y_T(t_j)}{\partial t_j^2} = Net(t_j,\vec{p}) + 2(t_j-a)\frac{\partial Net(t,\vec{p})}{\partial t_j} + \frac{1}{2}(t-a)^2\frac{\partial^2 Net(t,\vec{p})}{\partial t_j^2} \tag{2.11}$$

Tanım 2.3. İkinci mertebeden diferansiyel denklem için Dirichlet sınır değer problemi,

$$\begin{cases} y''(t) = f(t,y(t),y'(t)), \quad t \in [a,b] \\ y(a) = A \\ y(b) = B \end{cases} \tag{2.12}$$

olarak tanımlıdır.

Tanım 2.3 ile verilen Dirichlet sınır değer problemi için sınır değer koşullarını sağlayan deneme fonksiyonu 2.13 eşitliği ile tanımlıdır.

$$y_T(t,\vec{p}) = \frac{(t-b)A - (t-a)B}{(a-b)} + (t-a)(t-b)Net(t,\vec{p}) \tag{2.13}$$

Bilinmeyen parametre değerlerinin belirlenmesi, ikinci mertebeden başlangıç değer problemine benzer olarak 2.9 eşitliği ile verilen maliyet fonksiyonunun

minimizasyonunu gerektirir. Değişen sadece 2.13 eşitliği ile verilen deneme fonksiyonunun girdi değerine bağlı olarak 2.14 ve 2.15 eşitlikleri hesaplanan kısmi türev değerleridir.

$$\frac{\partial y_T(t_j)}{\partial t_j} = \frac{A-B}{a-b} + (2t_j - a - b)Net(t_j,\vec{p}) + (t_j-a)(t_j-b)\frac{\partial Net(t_j,\vec{p})}{\partial t_j} \quad (2.14)$$

$$\frac{\partial^2 y_T(t_j)}{\partial t_j^2} = 2Net(t_j,\vec{p}) + 2(2t_j - a - b)\frac{\partial Net(t_j,\vec{p})}{\partial t_j} + (t_j-a)(t_j-b)\frac{\partial^2 Net(t_j,\vec{p})}{\partial t_j^2} \quad (2.15)$$

Dirichlet sınır problemleri için Gradyan Düşüm Algoritması kullanıldığında, maliyet fonksiyonu E'nin bilinmeyen parametrelere göre türevleri alınması gerekir. Ancak, yüksek boyutlu problemlerde, büyük ihtimalle yerel minimuma takılma durumu oluşacaktır. Bundan dolayı, optimal parametreleri belirlemek için Bölüm 2.3 ile verilen bazı global optimizasyon algoritmaları kullanılmıştır.

2.2. Diferansiyel Denklem Sistemlerinin Çözümleri

Tanım 2.4. Diferansiyel denklem sistemi $t \in [a,b]$ için

$$\begin{cases} x'(t) = f(t,x(t),y(t)), & x(a) = A \\ y'(t) = g(t,x(t),y(t)), & y(a) = B \end{cases} \quad (2.16)$$

olarak tanımlıdır.

Başlangıç değer problemi olarak Tanım 2.4 ile verilen diferansiyel denklem sistemini ele alalım. Bu denklem sisteminin çözümü için, sınır değer koşullarını sağlayan iki farklı deneme fonksiyonu Eşitlik 2.17 ve Eşitlik 2.18 ile verilmiştir.

$$x_T(t_j,\vec{p}_1) = A + (t_j - a)Net_1(t_j,\vec{p}_1) \quad (2.17)$$

$$y_T(t_j,\vec{p}_2) = B + (t_j - a)Net_2(t_j,\vec{p}_2) \quad (2.18)$$

n toplam girdi sayısı ve m ara katmandaki nöron sayısı olmak üzere $i = 1,2,\ldots,m$ ve $j = 1,2,\ldots,n$ için $\vec{p}_1 = \vec{p}_1(\vec{\alpha}_1, \vec{w}_1, \vec{\beta}_1)$ ve $\vec{p}_2 = \vec{p}_2(\vec{\alpha}_2, \vec{w}_2, \vec{\beta}_2)$ sırasıyla iki farklı ileri beslemeli ağ için bilinmeyen parametrelerdir. $Net_1(t_j, \vec{p}_1) = \sum_{i=1}^{m} \alpha_1 \sigma(z_{1i})$ ve $Net_2(t_j, \vec{p}_2) = \sum_{i=1}^{m} \alpha_2 \sigma(z_{2i})$ sırasıyla yapay sinir ağları için çözümdür.

Oluşturulan ilk yapay sinir ağında m ara katmandaki nöron sayısı olmak üzere, $i = 1,2,\ldots,m$ için $z_{1i} = w_{1i} t_j + \beta_{1i}$ olarak elde edilir. z_{1i} ara katmanın girdi nöronları, w_{1i} ağırlık değeri ve β_{1i} değeri ise z_{1i} nöronunun eşik değeridir. İkinci yapay sinir ağı için benzer şekilde, z_{2i} ara katmanın girdi nöronları, w_{2i} ağırlık değeri ve β_{2i} değeri ise z_{2i} nöronunun eşik değeridir.

Diğer yandan her iki yapay sinir ağı için aktivasyon fonksiyonu olarak $\sigma(z_{1i}) = \dfrac{1}{1+\exp(-z_{1i})}$ ve $\sigma(z_{2i}) = \dfrac{1}{1+\exp(-z_{2i})}$ sigmoid fonksiyonları tercih edilmiştir.

Ağların çıktı değerinin hesaplanmasından sonra oluşan hatayı minimize etmek amacıyla ilk ağ için $\vec{p}_1$ ve ikinci ağ için $\vec{p}_2$'nin parametre değerleri güncellenmektedir.

t_j girdi değeri için maliyet fonksiyonu $H_{1j} = \dfrac{\partial^2 x_T(t_j)}{\partial t_j^2} - f(t_j, x_T(t_j), y_T(t_j)$ ve $H_{2j} = \dfrac{\partial^2 y_T(t_j)}{\partial t_j^2} - g(t_j, x_T(t_j), y_T(t_j)$ olmak üzere Toplam Karesel Hata Miktarını (Sum of Squarred Error) gösteren $E = \dfrac{1}{2} \sum_{j=1}^{n} \left(H_{1j}^2 + H_{2j}^2\right)$ eşitliği aracılığıyla elde edilir.

Oluşan hatayı minimize etmek amacıyla $\alpha_1 = \alpha_1 - \lambda \dfrac{\partial E}{\partial \alpha_1}$, $\beta_1 = \beta_1 - \lambda \dfrac{\partial E}{\partial \beta_1}$, $w_1 = w_1 - \lambda \dfrac{\partial E}{\partial w_1}$, $\alpha_2 = \alpha_2 - \lambda \dfrac{\partial E}{\partial \alpha_2}$, $\beta_2 = \beta_2 - \lambda \dfrac{\partial E}{\partial \beta_2}$ ve $w_2 = w_2 - \lambda \dfrac{\partial E}{\partial w_2}$ parametre değerlerinin güncellenme işlemleri gerçekleştirilir. Bu güncelleme işlemleri için kısmi türevler, her bir denklem için, Tanım 2.2 ile verilen birinci mertebeden başlangıç değer probleminde verilen türevlere benzer olarak elde edilir. Diferansiyel denklem sisteminin çözümünü elde etmek için, maliyet

fonksiyonunun iki farklı yapay sinir ağının ayrı ayrı tüm bilinmeyen parametrelere göre kısmi türevleri alınır.

Tanım 2.5. İkinci mertebeden diferansiyel denklem için Neuman sınır değer problemi

$$\begin{cases} y''(t) = f(t, y(t), y'(t)), & t \in [a,b] \\ y'(a) = A \\ y'(b) = B \end{cases} \tag{2.19}$$

olarak tanımlıdır.

Verilen Neuman sınır değer probleminin çözümü için, $y' = z$ dönüşümü yapılarak diferansiyel denklem sistemine dönüştürülür. Buna göre 2.20 eşitliği ile verilen denklem sisteminin çözümü, Tanım 2.16 ile verilen denklem sistemine benzer şekilde elde edilir.

$$\begin{cases} y'(t) = z = g(z), & y'(a) = z(a) = A \\ z'(t) = f(t, y(t), z(t)), & z(b) = B \end{cases} \tag{2.20}$$

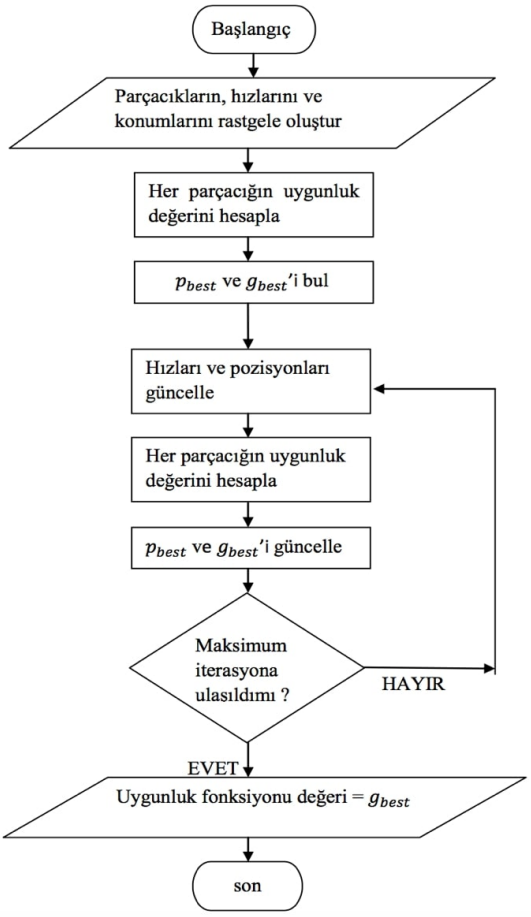
Bir sonraki bölümde ileri beslemeli yapay sinir ağlarının eğitimi aşaması sonrasında oluşan hatayı minimize etmek amacıyla Gradyan Düşüm Algoritmaları yerine kullanılan populasyon tabanlı global optimizasyon yaklaşımları açıklanacaktır.

2.3. Popülasyon Tabanlı Global Optimizasyon Yaklaşımları

Sezgisel yaklaşımlar olan popülasyon tabanlı optimizasyon algoritmaları rassal olarak oluşturulan bir başlangıç popülasyonu ile çözüm aramaya dayanır. Popülasyon her biri birer aday çözüm olan ve bağımsız olarak hareket edebildiği gibi dahil olduğu sürü ile birlikte hareket edebilen bireylerden oluşur. Devam eden kısımda çoğu doğadaki olay ve durumları örnek alarak oluşturulan ve tezde kullanılan bu tip metotlardan bahsedecektir.

2.3.1. Parçacık Sürü Optimizasyonu

Sürü zekasına dayalı global optimizasyon metotlarından biri olan Parçacık Sürü Optimizasyonu (PSO), başlangıçta rastgele konumları ve hızları olan parçacıkların arama uzayında global minimuma veya maksimuma ulaşmaya çalışmalarına dayalıdır [70]. Şekil 2.2'de verildiği üzere, algoritmanın işleyiş prensibi özetle şu şekilde açıklanabilir:



Şekil 2.2. Parçacık sürü optimizasyonu algoritmasının akış şeması [71].

İlk olarak rastgele parçacıklar ve bu parçacıklara ait hız ve konum bilgileri oluşturulur. Oluşturulan her parçacık için uygunluk değeri hesaplanır. Bir

parçacığın uygunluk değeri, bu parçacığın konumundaki optimum değeri bulunacak olan fonksiyonun almış olduğu değerdir.

Algoritmanın akışında yer alan p_{best}, bir parçacığın ulaştığı konumlar arasındaki en iyi uygunluk değerinin elde edildiği konum olarak tanımlıdır ve bu değer kişisel en iyi derece olarak da düşünülebilir. Böylece parçacık sonraki konum ve hızı için en iyi konumundan faydalanmış olacaktır. g_{best} ise, tüm parçacıklar arasında ulaşılan en iyi konum olarak belirlenmiştir. Diğer bir deyişle bu değer, konuma göre elde edilen uygunluk değerlerinin en iyisi olarak tanımlıdır.

Bir sonraki aşamada, parçacıkların hız ve konumları güncellenir. Bu güncelleme işlemi, parçacıkların kendi en iyi konumları (p_{best}) ve sürüde yer alan en iyi konum (g_{best}) bilgisi kullanılarak gerçekleştirilir.

Bu aşama sonrasında, sürüde yer alan her bir parçacık için uygunluk değerleri revize edilir ve buna bağlı olarak p_{best} ve g_{best} değerleri güncellenir. Bu süreçler önceden belirlenen durma kriteri sağlanana kadar tekrarlanır. Algoritma çalışmasını sonlandırdığında g_{best} değeri elde edilen yaklaşık optimal değer olarak belirlenir.

D boyutlu bir arama uzayında N tane parçacık verilsin. PSO algoritmasının çalışma prensibini açıklamak üzere $i.$ parçacığın konumu 2.21 eşitliği ile tanımlansın.

$$\mathbf{X}_i = (x_{i1}, x_{i2}, ..., x_{iD}),\ i = 1, 2, ..., N \tag{2.21}$$

Buna göre sürüde yer alan $i.$ parçacığın kişisel en iyi $\mathbf{p}_{best\,i}$ değerleri 2.22 eşitliği ile tanımlıdır.

$$\mathbf{p}_{best\,i} = (p_{i1}, p_{i2}, ..., p_{iD}),\ i = 1, 2, ..., N \tag{2.22}$$

Diğer yandan, $i.$ parçacığın yer değişim vektörü yani hız vektörü ise 2.23 eşitliği ile verilmiştir.

$$\mathbf{V}_i = (v_{i1}, v_{i2}, ..., v_{iD}),\ i = 1, 2, ..., N \tag{2.23}$$

Bu değerler kullanılarak hız ve konum güncelleme işlemleri 2.24 ve 2.25 eşitlikler kullanılarak hesaplanır.

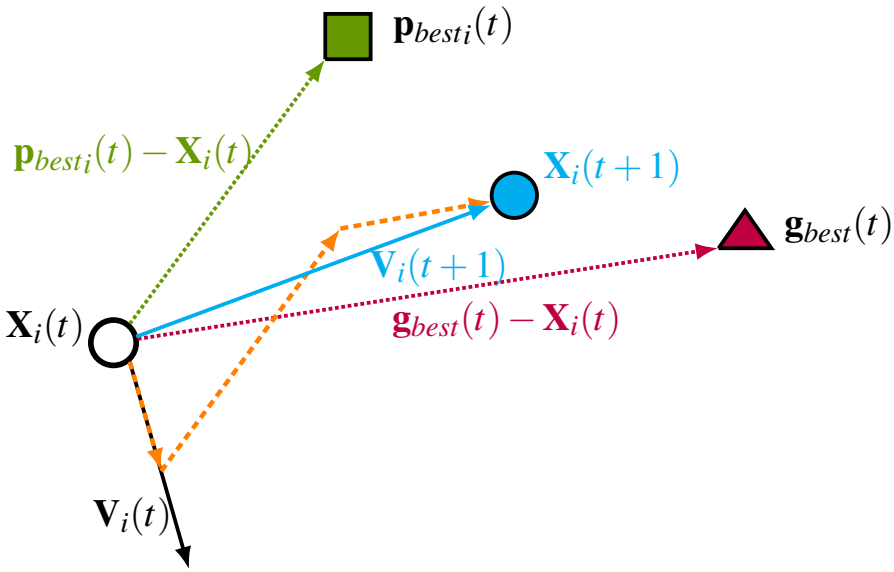
$$\mathbf{V}_i(t+1) = w\mathbf{V}_i(t) + c_1 r_1(t)(\mathbf{p}_{best_i}(t) - \mathbf{X}_i(t)) + c_2 r_2(t)(\mathbf{g}_{best}(t) - \mathbf{X}_i(t)) \quad (2.24)$$

2.24 eşitliğinde, $(t+1)$. iterasyondaki hız vektörü elde edilmiş olur. Eşitlikte yer alan c_1 ve c_2 değerleri hızlandırma katsayıları olarak belirlenmiştir. Ayrıca r_1 ve r_2 değerleri ise $[0,1]$ aralığında seçilen rastgele değerlerdir.

$$\mathbf{X}_i(t+1) = \mathbf{X}_i(t) + \mathbf{V}_i(t+1) \quad (2.25)$$

2.25 eşitliğindeki formül ile $(t+1)$. iterasyondaki konum vektörü bulunur. Elde edilen konum vektörü için probleme yeni bir çözüm önerisi olarak değerlendirilir.

Eğer durma kriteri sağlandıysa en iyi uygunluk değerine sahip parçacık konumu optimal çözüm olarak belirlenir. Aksi halde işlemler durma kriteri sağlanana kadar tekrarlanır.



Şekil 2.3. Parçacık sürü optimizasyonu algoritmasının hız güncelleme işlemi [72].

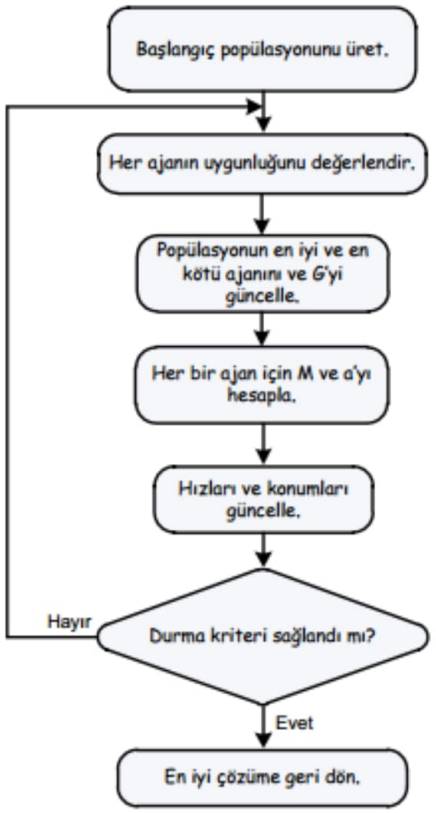
Şekil 2.3 ile görüldüğü üzere, hız güncelleme işlemi toplamda yer alan üç farklı kısımdan oluşmaktadır. Bunlardan ilki olan w$\mathbf{V}_i(t)$ değeri, hız değerinin atalet ağırlığını ifade etmektedir. Bu değer parçacıkların hız değerinin güncellemesi aşamasında ani değişiklikten kaçınılarak bir önceki değere göre güncellemesini sağlamaktadır. Atalet değerinin göz ardı edilmesi durumunda parçacıklar ani yön değiştireceği için uygun bir arama gerçekleşmeyecektir.

Benzer biçimde Şekil 2.3 ile görüldüğü üzere, $c_1 r_1(t)(\mathbf{p}_{best_i}(t) - \mathbf{X}_i(t))$ değeri ile belirlenen parçacıkların kişisel hafızaları sayesinde, konum güncellemede parçacıkların kendi kişisel en iyi konumuna doğru hareket edilmesi sağlanır. Bu değer $c_1 r_1(t)$ aracılığıyla ölçeklendirilmiştir. Böylelikle parçacığın kendi en iyi konumundan daha uzak bir konuma ulaşmaması garantilenir. $c_2 r_2(t)(\mathbf{g}_{best}(t) - \mathbf{X}_i(t))$ ise sosyal hafıza olarak adlandırılır ve parçacıkların konum güncelleme işleminde sürüde bilinen en iyi konumuna doğru hareket edilmesini sağlar. Bu değer $c_2 r_2(t)$ aracılığıyla ölçeklendirilmiştir.

2.3.2. Kütle Çekim Arama Algoritması

Rashedi ve arkadaşları tarafından 2009 yılında önerilen Kütle Çekim Arama Algoritması (GSA), belli bir kütleye sahip nesnelerin birbirleri arasındaki çekim kuvveti ile yerçekimi kanununu baz alan bir optimizasyon metodudur. Bu algoritma yerçekimi kuvveti ile birlikte kütlelerin birbirleri üzerine uyguladıkları toplam çekim kuvvetini kullanarak, kütlesi küçük olan nesnelerin kütlesi daha büyük olan nesneye doğru hareket etmesi esasına dayanır [73].

GSA yaklaşımında tanımlanmış olan nesneler optimum çözümü arayan ajanlar olarak adlandırılmıştır. Ajanlar kütlelerine göre en büyük kütleye sahip ajana doğru farklı hızlarla yaklaşırlar. Hızların belirlenmesindeki unsur ajanların birbirine uyguladıkları çekim kuvvetidir.



Şekil 2.4. Kütle çekim arama algoritmasının işleyiş şeması [74].

Kütle Çekim Arama Algoritmasının işleyiş prensibi Şekil 2.4'de verildiği üzere kısaca şu şekilde açıklanabilir: Öncelikle başlangıç popülasyonu rastgele oluşturulur ve sonra popülasyonda yer alan her bir ajan için uygunluk değeri hesaplanır. Her bir ajan, konum, hız, ivme ve kütle bilgisine sahiptir.

İterasyonun ilk adımında, ajanlar arasındaki mesafeler hesaplanır. Ardından yerçekimi sabiti, herhangi iki ajana ait kütle değerleri ve bu iki ajan arasındaki mesafe bilgileri kullanılarak birbirlerine uyguladıkları çekim kuvveti hesaplanır. Daha sonra herhangi bir ajan üzerine uygulanan toplam çekim kuvveti belirlenir. Toplam çekim kuvveti ve ajanın kütlesine göre ivme hesabı yapılır ve ajan bu

ivmeye bağlı olarak hızlandırılır. Hız bilgisine göre ise ajanın yeni konumu tespit edilir.

Konum güncellemesi yapıldıktan sonra revize edilen uygunluk değerlerinin en iyi ve en kötü değerleri kullanılarak kütle miktarları değiştirilir. Bu güncelleme işleminden sonra kütle miktarları normalize edilir. Böylelikle bilinen en iyi çözüme sahip ajan en büyük kütleye sahip olur. Bu adımlar hedeflenen kriter elde edilinceye kadar tekrar edilir. Algoritma çalışmasını sonlandırdığında en büyük kütleye sahip ajanın konumu optimal çözüme en yakın konum olarak saptanır.

Şimdi, N tane ajana sahip bir sistem olduğunu varsayalım. Belirli bir kütleye sahip bu ajanlar için $i.$ ajanın konumu 2.26 eşitliği ile tanımlıdır:

$$\mathbf{X}_i = (x_{i1}, x_{i2}, \ldots, x_{id}, ..., x_{in}),\ i = 1, 2, ..., N \tag{2.26}$$

2.26 denkleminde, N sürüde yer alan toplam ajan sayısı olmak üzere, x_{id} $i.$ ajana ait konum vektörünün $d.$ bileşeni olarak belirlenmiştir. Ajanlar başlangıçta arama uzayında rastgele konumlandırılır, bir başka deyişle aday çözüm ilk olarak rastgele belirlenir. Verilen bir t zamanında, i ve j kütleleri arasındaki kuvvet olan $\mathbf{F}_{ij}$, kütleler arasındaki öklid mesafesi $R_{ij}(t) = \|\mathbf{X}_i(t) - \mathbf{X}_j(t)\|_2$ olmak üzere, 2.27 eşitliği aracılığıyla elde edilir:

$$\mathbf{F}_{ij} = G(t)\frac{M_{pi}(t) \times M_{ai}(t)}{R_{ij}(t) + \varepsilon}(\mathbf{X}_i(t) - \mathbf{X}_j(t)) \tag{2.27}$$

2.27 eşitliğinde $M_{ai}(t)$, j ajanının aktif çekim kütlesini, $M_{pi}(t)$, i ajanının pasif çekim kütlesini, $G(t)$, t anında yerçekimi sabitini ve ε küçük bir sabit değeri göstermektedir. Aktif çekim kütlesi, bir nesne tarafından uygulanan çekim kuvvetinin büyüklüğü ile orantılı statik bir değerdir. Pasif çekim kütlesi ise bir nesneye başka bir nesne tarafından uygulanan çekim kuvveti ile orantılı statik bir değerdir.

r_j, $[0,1]$ kapalı aralığında rassal bir değer olmak üzere; $i.$ ajan üzerine uygulanan toplam kuvvet 2.28 denklemi ile hesaplanır:

$$\mathbf{F}_i = \sum_{j=1, j\neq i}^{N} r_j \mathbf{F}_{ij}(t) \tag{2.28}$$

Bulunan eşitlik ile hareket kanunu uygulanırsa, $M_{ii}(t)$ atalet kütlesini göstermek üzere, ivme 2.29 eşitliği ile elde edilir. Atalet kütlesi, durağan haldeki nesnenin ne ölçüde hızlanması gerektiğini gösteren dinamik bir değerdir.

$$\mathbf{a}_i = \frac{\mathbf{F}_i(t)}{M_{ii}(t)} \tag{2.29}$$

Böylece ajanın hız ve pozisyonunun hesaplanması sırasıyla 2.30 ve 2.31 eşitlikleri ile gerçekleştirilir:

$$\mathbf{V}_i(t+1) = r_i \mathbf{V}_i(t) + \mathbf{a}_i(t) \tag{2.30}$$

$$\mathbf{X}_i(t+1) = \mathbf{X}_i(t) + \mathbf{V}_i(t+1) \tag{2.31}$$

2.32 eşitliğinde verildiği üzere, G yerçekim kuvveti, G_0 ilk değeri olan ve zamanla değişen bir fonksiyondur.

$$G(t) = G(G_0, t) \tag{2.32}$$

Yerçekimi ve atalet kütlelerinin güncelleme işlemi, $f_i(t)$, t anında $i.$ ajanın uygunluk değeri ve başlangıçta $i = 1,2,\ldots,N$ için $M_{ai} = M_{pi} = M_{ii} = M_i$ olmak üzere, 2.33 ve 2.34 eşitlikleri kullanılarak gerçekleştirilir:

$$m_i(t) = \frac{f_i(t) - f_{enkötü}(t)}{f_{eniyi}(t) - f_{enkötü}(t)} \tag{2.33}$$

$$M_i(t) = \frac{m_i(t)}{\sum_{j=1}^{n} m_j(t)} \tag{2.34}$$

2.33 ve 2.34 eşitliklerinde verilen $f_{eniyi}(t)$ ve $f_{enkötü}(t)$ değerleri sırasıyla tüm ajanlardan elde edilen en iyi ve en kötü uygunluk değerleridir ve minimizasyon

problemleri için 2.35 ve 2.36 eşitlikleri ile elde edilir:

$$f_{eniyi}(t) = \min_{i \in 1,2,\ldots,N} f_i(t) \tag{2.35}$$

$$f_{enkötü}(t) = \max_{i \in 1,2,\ldots,N} f_i(t) \tag{2.36}$$

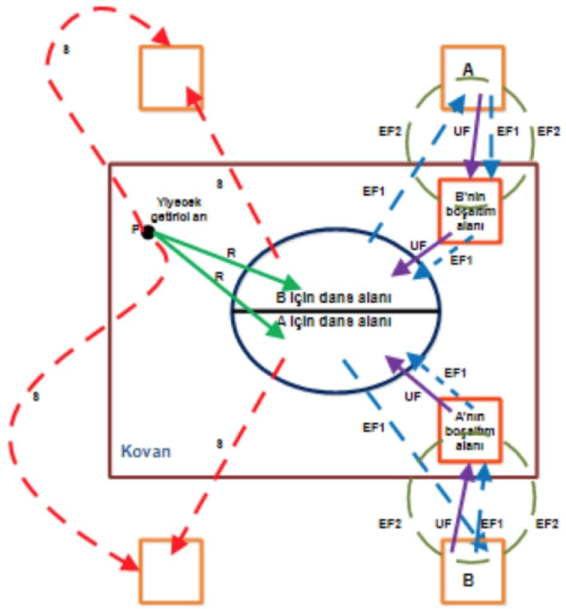
Diğer taraftan maksimizasyon problemleri için ise, 2.35 ve 2.36 eşitliklerinin sağ tarafları yer değiştirilir.

Eğer durma kriteri sağlandıysa, en iyi uygunluk değerine sahip ajanın konumu optimal çözüm olarak belirlenir. Aksi halde işlemler durma kriteri sağlanana kadar tekrarlanır.

2.3.3. Yapay Arı Koloni Algoritması

Sürü zekasına dayalı global optimizasyon algoritmalarından biri olan Yapay Arı Kolonisi algoritması (ABC), arıların doğadaki davranışlarından esinlenerek oluşturulmuştur. Algoritmada yer alan yiyecek kaynağının konumunu belirleme, problemin çözümü olarak belirlenmiştir. Algoritmada işçi, gözcü ve kaşif arı olmak üzere üç farklı arı tipi tanımlanmıştır. Bu arıların davranışı doğada yer alan arıların davranışı ile birebir örtüşmemekle beraber, doğada yer alan özelliklere ek olarak algoritmaya bazı yeni özellikler de eklenmiştir.

Bu özelliklerden ilki, her bir nektarın ya da besin kaynağının çıkarılmasına karşılık sadece bir arının görevlendirilmiş olmasıdır. Dolayısıyla algoritmadaki besin sayısı ile görevli arıların sayısı eşit olarak alınır. Diğer bir yeni özellik ise, işçi arı ile gözcü arıların sayılarının eşit olarak alınmasıdır. Besin kaynağının kalitesi uygunluk değeri ile belirlenir. Böylece maksimizasyon ya da minimizasyon problemleri için besin kaynağının kalitesinin iyi olması araştırılacak ve çözüm elde edilen en yüksek veya en düşük uygunluk değerinin bulunduğu konum olarak alınır.



Şekil 2.5. Yapay arı kolonisi algoritmasının çalışma prensibi [77].

Şekil 2.5'de görüldüğü üzere, kaşif arıların görevi besin kaynağını aramaktır. Bu arayış rastgele olarak gerçekleşmektedir. Keşif sonucu bulunan besin kaynağından kaşif arı kovana nektar taşımaya başlar. Kovandaki kaşif arı üç farklı şekilde rol almaktadır. Bu rollerden ilki, dans alanında besin kaynağı ile ilgili bilgiyi arılarla paylaşmak, diğeri bilgi aktarımı yapmadan besin kaynağına doğru gitmek ve diğer bir rol ise besin kaynağından uzaklaşarak kaşif arı olarak daha iyi bir çözüm aramaya devam etmektir. Kaşif arıların dansını izleyerek bilgi toplayan gözcü arılar ise besin kaynağına yönelir.

Yapay Arı Kolonisi Algoritmasının işleyişinde, $r \in (0,1)$ rassal bir değer olmak üzere, öncelikle 2.37 eşitliği ile besin kaynakları arama uzayında rastgele konumlandırılır.

$$x_{i,j} = \underline{x_j} + r \times (\overline{x_j} - \underline{x_j}) \tag{2.37}$$

2.37 eşitliğinde S_n toplam besin kaynağı sayısını göstermek üzere, $i = 1,2,\ldots,S_n$ için $\underline{x_j}$ ve $\overline{x_j}$ değerleri j. parametre değerinin sırasıyla alt ve üst sınır değerlerini gösterir.

Algoritmada gözcü arılar en uygun besin kaynağını 2.38 eşitliğiyle verilen olasılık hesabına göre seçerler.

$$P_i = \frac{f(\mathbf{x_i})}{\sum_{i=1}^{S_n} f(\mathbf{x_i})} \tag{2.38}$$

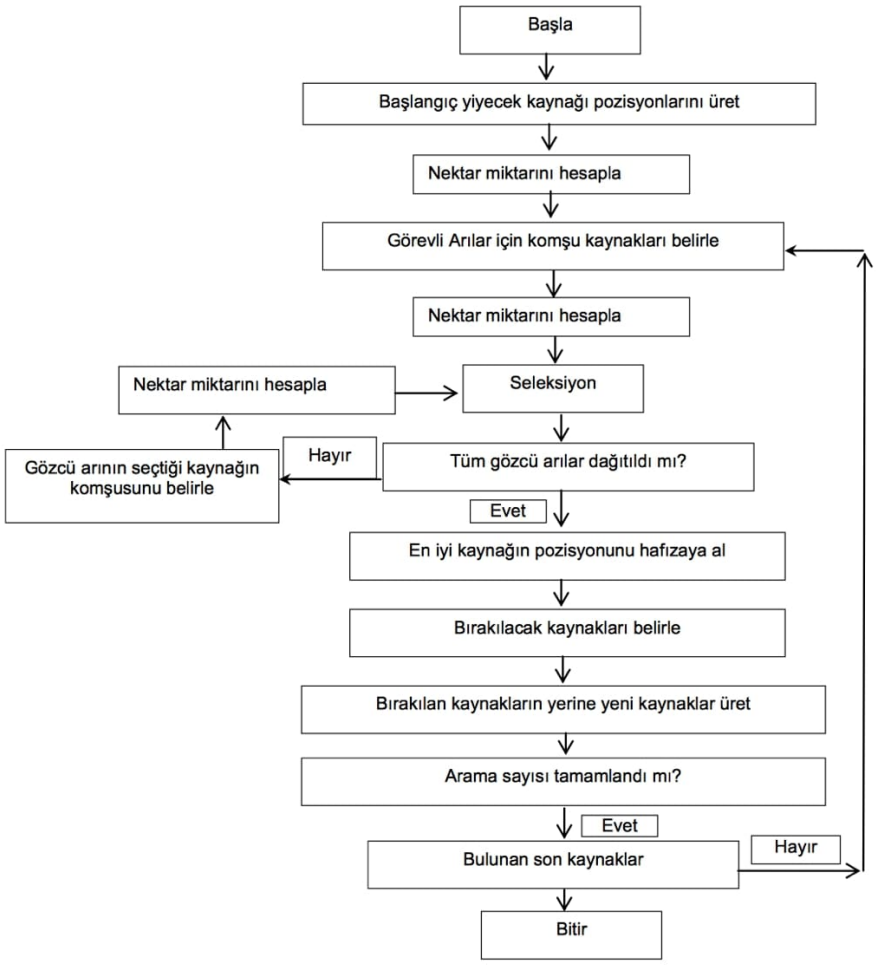
Verilen eşitlikte $f(\mathbf{x_i})$ değeri i. besin kaynağındaki nektar miktarı olarak tanımlanmıştır.

D boyutlu arama uzayında, besin kaynağı $\mathbf{x}_i$ için konum güncelleme işlemi ise, $i,k \in \{1,2,...,S_n\}$ ve $k \neq i$ olmak üzere, 2.39 eşitliği ile verilmiştir.

$$x'_{i,j} = x_{i,j} + \psi(x_{i,j} - x_{k,j}), \quad j = 1,2,\ldots,D \tag{2.39}$$

Verilen eşitlikte $x_{i,j}$ değeri $x_{k,j}$ komşuluğuna göre belirlenir ve ψ değeri $[-1,0]$ aralığında rastgele bir değerdir. Aynı zamanda j ve k değerleri de rastgele belirlenmektedir. Bu işlemin temel sebebi, kaşif arının besin kaynağının konumunu hatırladığı kadarıyla diğer arılara aktarmasıdır. Kaşif arı diğer arılara besin kaynağının konumunu aktarırken konum vektöründe tek bir bileşen haricinde diğer bileşenleri doğru olarak aktarır. Dolayısıyla arama uzayında her iterasyonda tek bir doğrultu üzerinde arama işlemi gerçekleştirilir, bu da algoritmaya arama uzayındaki bilinen en iyi çözüm civarında daha iyi arama yapma imkanı tanır.

Bu süreç istenilen toleransla yaklaşık çözüme ulaşana kadar devam eder. Algoritmanın verilen iş akış diyagramı ise Şekil 2.6 aracılığıyla sunulmuştur.



Şekil 2.6. Yapay arı kolonisi algoritmasının çalışma prensibi [77].

Bir sonraki bölümde tarafımızdan Yapay Arı Kolonisi Algoritmasını iyileştiren bir yaklaşım önerilmiştir.

2.3.3.1. Yapay Arı Koloni Algoritması için Yeni Bir Mutasyon Önerisi

ABC algoritmasının en iyi çözüm civarındaki yerel arama (exploitation) ve tüm arama uzayı üzerinde küresel arama (exploration) yeteneklerini geliştirebilmek için tarafımızdan yeni bir yaklaşım önerilmiştir [76].

Önerilen yaklaşımda en iyi çözümün komşuluğunda başlangıçta rassal bir yarıçapa sahip ve giderek daralan bir hiper-küre oluşturulmuştur. Her iterasyonda gamma fonksiyonu kullanılarak dinamik olarak oluşturulan bu hiper-küre içinde üretilen ve global çözümün mutasyonları olarak adlandırılan yeni bir popülasyon üretilmiştir. Üretilen bu popülasyon içerisinde global çözümden daha iyi uygunluk seviyesine sahip bir birey (arı) olup olmadığı araştırılmıştır. Benzer olarak, hiper-küre dışında üretilen popülasyon içinde arama uzayının geri kalan kısmı keşfedilmeye çalışılmıştır.

Önerilen yaklaşımda orijinal Yapay Arı Kolonisi Algoritmasından farklı olarak uyarlamalı (adaptif) sürü boyutu kullanılmıştır. Uyarlamalı sürüsü kullanan sezgi üstü algoritmalar genellikle tüm arama uzayında düzgün dağılımlı popülasyon üretmektedir. ABC algoritmasında eğer bir çözüm terkedilirse, o terkedilen çözümün etrafında yeni bir popülasyonunun üretilmesi zaman kaybına yol açan gereksiz bir süreçtir.

Bahsedilen durumu önlemek için, tüm arama uzayının alt alanı olarak hiper-kürenin içinde ve dışında eşit popülasyon oluşturulmuştur. Popülasyon üretilirken 2.40 eşitliğinde verilen gamma fonksiyonu kullanılmaktadır.

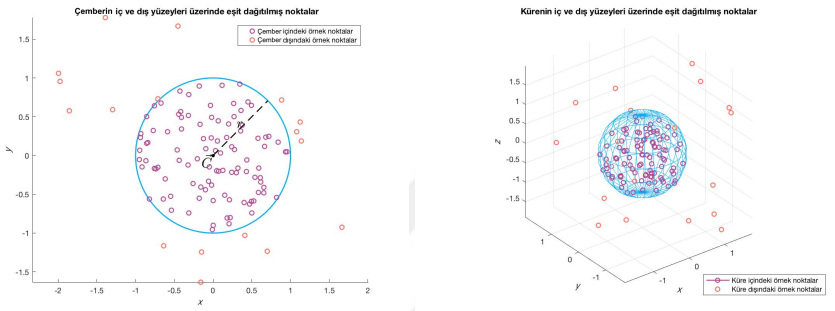
$$\vec{p}_i = \vec{C} + r.\vec{U}_i \frac{\left(\int_0^{S_i^2} t^{\left(\frac{D}{2}-1\right)} e^{-t} dt \right)^{\frac{1}{D}}}{S_i.\Gamma\left(\frac{D}{2}\right)^{\frac{1}{D}}} \tag{2.40}$$

2.40 eşitliğinde $\vec{U}_i$, arama alanı üzerinde sürekli düzgün dağılım ile rastgele oluşturulmuş vektördür. $\vec{C}$ hiper-kürenin merkezidir ve o ana kadar bilinen en iyi çözümün konumunu temsil eder. r hiper-kürenin yarıçapı olarak tanımlıdır.

2.40 eşitliğinde verilen

$$S_i = \sqrt{\sum_{j=1}^{D} p_{i,j}^2}$$

olmak üzere, $i = 1, 2, \ldots, M$ için M değeri yeni popülasyonun boyutunu gösterir ve D arama uzayının boyutudur. $\vec{p}_i$ ise yeni oluşturulan popülasyondaki $i.$ bireyin konum vektörüdür.



(*a*) Birim çember üzerinde iki boyutlu durum. (*b*) Birim küre üzerinde üç boyutlu durum.

Şekil 2.7. Çember ve kürenin iç ve dış yüzeyleri üzerinde düzgün dağılımlı noktalar.

Şekil 2.7'de hiper-kürede rastgele oluşturulan noktalar görülmektedir. Şekil 2.7a'da birim çember üzerinde düzgün dağılımlı noktaların üretimi gösterilmiştir. Benzer şekilde üç boyutlu uzay için Şekil 2.7b'de verilmiştir.

Önerilen yaklaşımda öncelikle 2.40 eşitliği kullanılarak rastgele yeni bir popülasyon üretilmiştir. Böylece yerel arama (exploitation) süreci iyileştirilmeye çalışılmıştır. Ayrıca, kaşif arı aşamasında, terkedilen herhangi bir çözümün etrafında yeni bir aday çözüm üretilmiştir. Oluşturulan yeni popülasyon sonrasında elit bireylerin seçim işlemi yapılır. Bu işlemde yeni oluşturulan nüfusun uygunluk değerleri hesaplanır ve en zayıf olan bireyler, yani uygunluk değeri en yüksek olanlar elenir. Bu durum tüm arama uzayı boyunca global optimuma ulaşma olasılığını arttırır. Her bir iterasyonda hiper-kürenin yarıçapı düşürülerek dinamik bir global arama sağlanmaktadır.

Önerilen metodun klasik yapay arı kolonisi algoritması ile karşılaştırması amacıyla küre ve Rosenbrock fonksiyonlarının her iki algoritma ile elde edilen çözümleri Örnek 2.6'da karşılaştırılmıştır.

Örnek 2.6. d boyutlu uzayda küre fonksiyonu

$$f(x) = \sum_{i=1}^{d} x_i^2 \tag{2.41}$$

ve Rosenbrock fonksiyonu

$$f(x) = \sum_{i=1}^{d-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2] \tag{2.42}$$

olmak üzere tanımlıdır. Buna göre 100 iterasyon için 10 ve 30 boyutlu uzayda algoritmaların çözümleri Çizelge 2.1 ile verilmiştir.

Çizelge 2.1. Küre ve Rosenbrock test fonksiyonları için ABC ve mABC kullanılarak elde edilen en iyi uygunluk değerlerinin ortalaması.

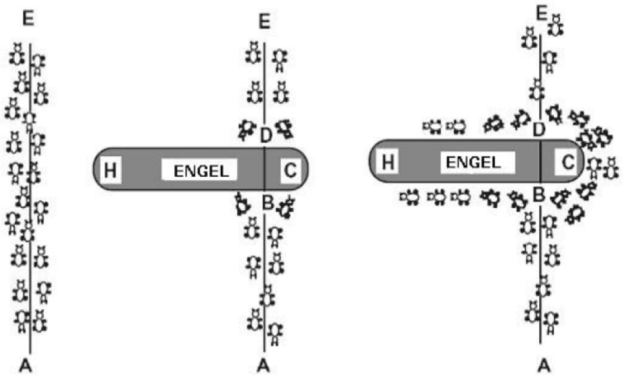
		Boyut (d)	
Yöntem	**Fonksiyon**	**10**	**90**
ABC	Küre	$6.790 \times 10^{-23} \pm 6.835 \times 10^{-23}$	$1.832 \times 10^{+3} \pm 94.36$
	Rosenbrock	4.772 ± 0.1104	$1.141 \times 10^{+7} \pm 1.374 \times 10^{+6}$
mABC	Küre	$1.512 \times 10^{-29} \pm 3.139 \times 10^{-29}$	$5.070 \times 10^{-8} \pm 3.072 \times 10^{-9}$
	Rosenbrock	$2.400 \times 10^{-5} \pm 5.509 \times 10^{-6}$	57.44 ± 15.99

Çizelge 2.1 ile verildiği üzere her iki boyut için önerilen mutasyon yaklaşımlı algoritmanın klasik yapay arı koloni algoritmasının çözümlerini iyileştirdiği gözlemlenmiştir. Ek A kısmında önerilen yaklaşımın algoritması Algoritma 2 ile adım adım verilmiştir. Algoritma 2'in çalışabilmesi için, Algoritma 1'de sunulan fonksiyon çağrılarına ihtiyaç duyulur. Listelenen fonksiyonlar ileri beslemeli yapay sinir ağının oluşturulması ve probleme dair maliyet fonksiyonunun hesaplanması için tanımlanmıştır.

2.3.4. Karınca Koloni Optimizasyonu

Karınca Kolonisi Optimizasyonu (ACO), gerçek karınca koloni davranışlarının matematiksel olarak modellenmesiyle geliştirilen bir algoritmadır. İlk olarak Dorigo ve arkadaşları tarafından gezgin satıcı problemi ve karesel sıralama problemleri gibi zor optimizasyon problemlerinin çözümü için önerilmiştir [78].

Karıncalar, yiyecek temin ederken öncelikle besin kaynağı ile yuva arasındaki kullanabileceği yolları tespit eder. Tespit edilen bu yollardan herhangi birini ilk kullanan karınca feromon adında kimyasal bir salgı bırakır. Yolun kısa olması durumunda bu salgının yoğunluğu daha fazladır. Diğer karıncalar da feromon miktarına göre gideceği yola karar verir. Şekil 2.8'de görüldüğü üzere, karıncalar engel karşısında başlangıçta eşit olarak yollara dağılırken, zamanla feromon miktarı yoğunluğuna göre en kısa yola ulaşma yeteneğine sahiptir.



Şekil 2.8. Karıncaların en kısa yolu bulma aşamaları [79].

Yapay karıncalar ise gerçek karıncalardan bazı özellikler ile ayrışır. Karınca Kolonisi Optimizasyonuna göre oluşturulan yapay feromon izlerinin güncelleme işlemi en iyi çözüme ulaşana kadar tekrarlanır. Algoritmada başlıca yapılan işlemler, feromon miktarının arttırılması veya belli miktarda azaltılması, en iyi çözüme ulaşılması durumunda global feromon miktarının güncellenmesi ve güncellenen bu değere göre karıncaların yeni iterasyonda benzer işlemlere devam etmesidir. Algoritma genel manada feromon izlerinin güncelleştirilmesiyle tekrar eden bir yaklaşıma sahiptir. Bu yaklaşımda en iyi çözüme ulaşmak için oluşturulan bilgiler her isterasyonda güncellenmektedir.

ACO algoritmasında bir turda, j. noktadaki k. karınca için bir sonraki adımda hangi yolu seçeceği iki farklı şekilde gerçekleşir. Bunlardan ilki, feromon miktarının

maksimum olduğu yolun seçilmesidir. İkincisi ise, feromon miktarlarına göre farklı yollardan birinin olasılıksal olarak seçilmesidir. Feromon miktarının daha yüksek olduğu yolun seçilme olasılığı daha yüksektir. Bu yol seçimine göre , i. noktadaki k. karınca için u tane farklı yolun hangisini tercih edeceği 2.43 eşitliği ile belirlenir.

$$j = \max_{u \in J_k(i)} \{[\tau(i,u)]^{\alpha} \times [\eta(i,u)]^{\beta}\} \tag{2.43}$$

Verilen eşitlikte, $\tau(i,u)$, (i,u) yolundaki feromon iz miktarı, d_{ij}, i. nokta ile j. nokta arasındaki uzaklık olmak üzere, $\eta(i,u) = \dfrac{1}{d_{iu}}$ ve $J_k(i)$ değeri ise i. noktadaki k. karınca için henüz kullanılmayan yolların kümesini göstermektedir. Ayrıca α feromon oranı katsayısı ve $\beta > 0$ ağırlıklandırma katsayıdır.

$0 < q < 1$ değeri her aday çözüm için rassal olarak belirlenir ve belirlenen çözüm uzayını araştırmanın önemini gösteren parametredir. q_0 değeri, bu değer için eşik değerdir. Eğer $q < q_0$ ise yol seçimi 2.43 eşitliği ile belirlenirken, aksi durumda kullanılabilecek yolların seçilme olasılıkları 2.44 eşitliği ile belirlenir. Bu eşitlik aracılığıyla i. noktadaki bir karıncanın j. noktasını seçip o noktaya gitme olasılığı elde edilmiş olur.

$$P_{ij}^{k} = \begin{cases} \dfrac{[\tau(i,u)]^{\alpha} \times [\eta(i,u)]^{\beta}}{\sum\limits_{u \in J_k(i)} \{[\tau(i,u)]^{\alpha} \times [\eta(i,u)]^{\beta}\}}, & j \in J_k(i) \\ 0, & \text{aksi taktirde} \end{cases} \tag{2.44}$$

Literatürde yol seçimi ile ilgili olarak Turnuva Seçimi ve Şans Çarkı gibi farklı yaklaşımlara rastlamak mümkündür.

Karıncalar turlarını bitirdiğinde feromon güncelleme işlemi gerçekleşmektedir. Buna göre öncelikle feromon buharlaştırma yapılır. Sonraki aşama ise, karıncaların kullandıkları yollardaki feromon miktarının kullanılan yol uzunluğu ile ters orantılı olacak şekilde arttırılmasıdır. Bunu yapmaktaki amaç, daha kısa yol kullanan karınca için daha fazla feromon miktarı olmasını sağlamaktır. Lokal ve global feromon güncelleme olmak üzere iki farklı şekilde yapılan işlemler 2.45 eşitliğinde

verilmiştir.

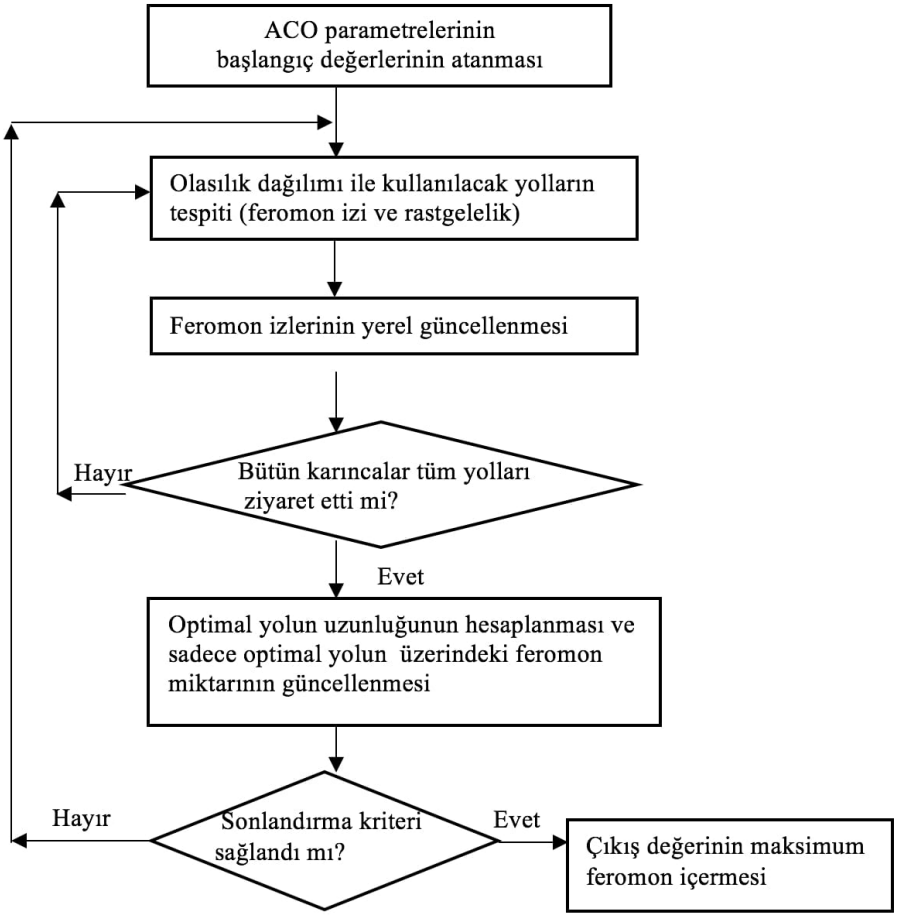
$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \sum_{k=1}^{m} \Delta_{ij}^{k}(t+1) \tag{2.45}$$

Verilen eşitlikte $\Delta_{ij}^{k}(t+1)$ değeri lokal güncelleme yapıldığında 2.46 eşitliği, global güncelleme yapıldığında ise 2.47 kullanılmaktadır.

$$\Delta_{ij}^{k}(t+1) = \begin{cases} \dfrac{1}{L^{k}(t+1)} & k \text{ karıncası (i,j) yolunu kullanmışsa} \\ 0 & \text{aksi taktirde} \end{cases} \tag{2.46}$$

$$\Delta_{ij}^{k}(t+1) = \begin{cases} \dfrac{1}{L_{best}(t+1)} & k \text{ karıncası (i,j) yolunu kullanmışsa} \\ 0 & \text{aksi taktirde} \end{cases} \tag{2.47}$$

Verilen eşitlikte L_{best} değeri o anki iterasyondaki elde edilen en iyi turun uzunluğu olarak belirlenmiştir.



Şekil 2.9. Karınca kolonisi algoritmasının akış diyagramı [80].

Şekil 2.9 ile ACO algoritmasının akış diyagramı verilmiştir. Özetle karınca koloni optimizasyonunun adımları verilen sırada gerçekleşmektedir: Öncelikle başlangıç feromon değerleri tespit edilir. Karıncaların konumu rassal olarak belirlendikten sonra, herbir karınca sonraki yolu denklemde yer alan 2.44 eşitliği ile verilen lokal arama olasılığına göre kendi turunu tamamlar. Sonra herbir karınca için geçilen yolların uzaklığı hesaplanır ve 2.46 eşitliği ile lokal feromon güncelleme işlemi yapılır. Böylece en iyi çözüm elde edilir ve bu değer 2.47 eşitliği ile global feromon güncelleme işleminde kullanılır. İstenilen kritere veya maksimum

iterasyon sayısına ulaşılana kadar ikinci adımdan itibaren algoritma çalışmaya devam eder.

3. DENEYSEL ÇALIŞMALAR

Bu bölümde, başlangıç ve sınır değer problemleri için farklı tipte diferansiyel denklemlerinin nümerik çözümü araştırılmıştır. Bu amaçla birinci ve ikinci mertebeden lineer başlangıç değer problemi, ikinci mertebeden lineer ve lineer olmayan Dirichlet sınır değer problemleri ve birinci mertebeden lineer denklem sistemi ele alınmıştır. Çözümleri elde etmek için tezde yer alan global optimizasyon metotları kullanılmıştır. Deneysel çalışmalarda kullanılan keyfi parametreler ise Çizelge 3.1 ile verilmiştir.

Çizelge 3.1. Deneysel çalışmalarda kullanılan keyfi parametreler.

Parametreler	**Parametrelerin Değerleri**
YSA'daki nöron sayısı, m	6
Popülasyon sayısı, N	30
Arama uzayının alt sınırı	-1
Arama uzayının üst sınırı	1
Arama uzayının boyutu, D	$3m$
Eğitim kümesi için adım uzunluğu, h	0.01
Test kümesi için adım uzunluğu, $\frac{h}{2}$	0.005
İterasyon sayısı	1000

Örnek 3.1. Birinci mertebeden lineer başlangıç değer problemi 3.1 denklemi ile verilmiştir.

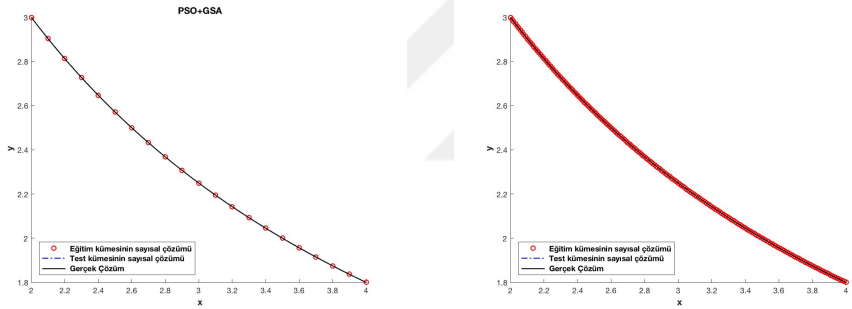
$$\begin{cases} y'(t) + \dfrac{y(t)}{t+1} = 0, \quad t \in [2,4], \\ y(2) = 3 \end{cases} \tag{3.1}$$

Verilen denklemin analitik çözümü $y(t) = \dfrac{9}{t+1}$'dir. Denklemin elde edilen nümerik çözümlerin sonucu oluşan mutlak hata değerleri iteratif metotlar için Çizelge 3.2, sezgisel optimizasyon metotlar için Çizelge 3.3 ve hibrit optimizasyon metotlar için ise Çizelge 3.4 ile sonuçlar verilmiştir.

İteratif metotlardan atış metodunu (shooting method) ve Lobatto IIIa metodunu kullanabilmek için, örnekte verilen denklem t'ye göre tekrar türevlenip ikinci

mertebeden bir diferansiyel denklem elde edilmelidir. Böylece atış metodu ve Lobatto IIIa uygulanabilir. Ancak bu durumda metotların çalışması için $y'(a)$ değerinin tahmin edilmesi gerekmektedir. Tezde sonuçların karşılaştırılabilmesi amacıyla $y'(a)$ sınır koşulu gerçek çözüm değeri olarak alınmıştır.

Örnekte verilen metotlarda en iyi sonuca PSOGSA kullanılarak ulaşılmıştır. PSOGSA hibrit metodunda $h = 0.1$ için eğitim ve test kümesine göre elde edilen nümerik çözüm ile gerçek çözümün karşılaştırıldığı grafik Şekil 3.1 a ile, $h = 0.01$ için ise Şekil 3.1 b ile verilmiştir.



(a) $h = 0.1$ (b) $h = 0.01$

Şekil 3.1. Örnek 3.1 için PSOGSA algoritması kullanılarak 1000 iterasyonda birinci mertebeden başlangıç değer probleminin yaklaşık çözüm grafiği.

Ayrıca elde edilen hatalar için ortalama karesel hata değerleri Çizelge 3.5 aracılığıyla sunulmuştur. Test kümesi için işlem süreleri ise Çizelge 3.6 ile karşılaştırılmıştır.

Çizelge 3.2. Örnek 3.1 için İteratif Metotlarda Mutlak Hata Değerleri.

		İteratif Metotlar		
	x	SM	FDM	Lobatto IIIa
Eğitim Kümesi	2.000	0.000	1.000	6.752×10^{-6}
	2.100	9.260×10^{-2}	9.680×10^{-1}	1.722×10^{-1}
	2.200	1.824×10^{-1}	9.379×10^{-1}	3.391×10^{-1}
	2.300	2.695×10^{-1}	9.096×10^{-1}	5.011×10^{-1}
	2.400	3.543×10^{-1}	8.830×10^{-1}	6.588×10^{-1}
	2.500	4.370×10^{-1}	8.580×10^{-1}	8.125×10^{-1}
	2.600	5.176×10^{-1}	8.343×10^{-1}	9.625×10^{-1}
	2.700	5.965×10^{-1}	8.118×10^{-1}	1.109
	2.800	6.737×10^{-1}	7.906×10^{-1}	1.253
	2.900	7.493×10^{-1}	7.704×10^{-1}	1.393
	3.000	8.235×10^{-1}	7.513×10^{-1}	1.531
	3.100	8.964×10^{-1}	7.330×10^{-1}	1.667
	3.200	9.681×10^{-1}	7.156×10^{-1}	1.800
	3.300	1.039	6.991×10^{-1}	1.931
	3.400	1.108	6.833×10^{-1}	2.060
	3.500	1.176	6.682×10^{-1}	2.188
	3.600	1.244	6.537×10^{-1}	2.313
	3.700	1.311	6.398×10^{-1}	2.437
	3.800	1.376	6.266×10^{-1}	2.559
	3.900	1.442	6.138×10^{-1}	2.680
	4.000	1.506	6.016×10^{-1}	2.800
Test Kümesi	2.000	0.000	1.000	6.752×10^{-6}
	2.105	9.715×10^{-1}	9.663×10^{-1}	1.806×10^{-1}
	2.205	1.868×10^{-1}	9.362×10^{-1}	3.473×10^{-1}
	2.305	2.738×10^{-1}	9.080×10^{-1}	5.091×10^{-1}
	2.405	3.585×10^{-1}	8.814×10^{-1}	6.666×10^{-1}
	2.505	4.411×10^{-1}	8.563×10^{-1}	8.201×10^{-1}
	2.605	5.216×10^{-1}	8.326×10^{-1}	9.699×10^{-1}
	2.705	7.430×10^{-1}	8.102×10^{-1}	1.116
	2.805	6.775×10^{-1}	7.890×10^{-1}	1.260
	2.905	7.531×10^{-1}	7.688×10^{-1}	1.400
	3.005	8.272×10^{-1}	7.497×10^{-1}	1.538
	3.105	9.000×10^{-1}	7.315×10^{-1}	1.673
	3.205	9.716×10^{-1}	7.141×10^{-1}	1.807
	3.305	1.042	6.976×10^{-1}	1.938
	3.405	1.111	6.818×10^{-1}	2.067
	3.505	1.180	6.667×10^{-1}	2.194
	3.605	1.247	6.522×10^{-1}	2.319
	3.705	1.314	6.384×10^{-1}	2.443
	3.805	1.380	6.251×10^{-1}	2.565
	3.905	1.445	6.124×10^{-1}	2.686
	4.000	1.506	6.008×10^{-1}	2.800

Çizelge 3.3. Örnek 3.1 için Sezgisel Metotlarda Mutlak Hata Değerleri.

		Sezgisel Metotlar			
	x	mABC	ACO	GSA	PSO
Eğitim Kümesi	2.000	0.000	0.000	0.000	0.000
	2.100	5.623×10^{-5}	1.269×10^{-3}	4.983×10^{-4}	4.887×10^{-4}
	2.200	3.482×10^{-5}	2.449×10^{-3}	5.922×10^{-4}	5.547×10^{-4}
	2.300	8.548×10^{-6}	3.541×10^{-3}	4.546×10^{-4}	3.925×10^{-4}
	2.400	4.426×10^{-5}	4.570×10^{-3}	2.090×10^{-4}	1.376×10^{-4}
	2.500	6.013×10^{-5}	5.577×10^{-3}	5.965×10^{-5}	1.211×10^{-4}
	2.600	5.483×10^{-5}	6.604×10^{-5}	2.963×10^{-4}	3.299×10^{-4}
	2.700	3.313×10^{-5}	7.689×10^{-3}	4.691×10^{-4}	4.618×10^{-4}
	2.800	2.514×10^{-6}	8.864×10^{-3}	5.634×10^{-4}	5.091×10^{-4}
	2.900	2.916×10^{-5}	1.015×10^{-2}	5.776×10^{-4}	4.775×10^{-4}
	3.000	5.514×10^{-5}	1.155×10^{-2}	5.194×10^{-4}	3.816×10^{-4}
	3.100	7.061×10^{-5}	1.306×10^{-2}	4.026×10^{-4}	2.409×10^{-4}
	3.200	7.311×10^{-5}	1.466×10^{-2}	2.455×10^{-4}	7.701×10^{-5}
	3.300	6.259×10^{-5}	1.632×10^{-2}	6.837×10^{-5}	8.816×10^{-5}
	3.400	4.125×10^{-5}	1.800×10^{-2}	1.074×10^{-4}	2.344×10^{-4}
	3.500	1.323×10^{-5}	1.965×10^{-2}	2.605×10^{-4}	3.441×10^{-4}
	3.600	1.577×10^{-5}	2.121×10^{-2}	3.706×10^{-4}	4.029×10^{-4}
	3.700	3.892×10^{-5}	2.261×10^{-2}	4.185×10^{-4}	4.004×10^{-4}
	3.800	4.864×10^{-5}	2.379×10^{-2}	3.869×10^{-4}	3.300×10^{-4}
	3.900	3.703×10^{-5}	2.467×10^{-2}	2.606×10^{-4}	1.893×10^{-4}
	4.000	3.790×10^{-6}	2.517×10^{-2}	2.646×10^{-5}	2.058×10^{-5}
Test Kümesi	2.000	0.000	0.000	0.000	0.000
	2.105	5.641×10^{-5}	1.330×10^{-3}	5.109×10^{-4}	5.001×10^{-4}
	2.205	3.283×10^{-5}	2.506×10^{-3}	5.897×10^{-4}	5.506×10^{-4}
	2.305	1.068×10^{-5}	3.593×10^{-3}	4.441×10^{-4}	3.811×10^{-4}
	2.405	4.557×10^{-5}	4.621×10^{-3}	1.955×10^{-4}	1.242×10^{-4}
	2.505	6.035×10^{-5}	5.628×10^{-3}	7.260×10^{-5}	1.330×10^{-4}
	2.605	5.408×10^{-5}	6.657×10^{-3}	3.067×10^{-4}	3.384×10^{-4}
	2.705	3.175×10^{-5}	7.746×10^{-3}	4.757×10^{-4}	4.662×10^{-4}
	2.805	8.927×10^{-7}	8.925×10^{-3}	5.660×10^{-4}	5.093×10^{-4}
	2.905	3.065×10^{-5}	1.021×10^{-2}	5.763×10^{-4}	4.741×10^{-4}
	3.005	5.619×10^{-5}	1.162×10^{-2}	5.148×10^{-4}	3.755×10^{-4}
	3.105	7.105×10^{-5}	1.314×10^{-2}	3.956×10^{-4}	2.330×10^{-4}
	3.205	7.288×10^{-5}	1.474×10^{-2}	2.369×10^{-4}	6.861×10^{-5}
	3.305	6.175×10^{-5}	1.640×10^{-2}	5.937×10^{-5}	9.608×10^{-5}
	3.405	3.997×10^{-5}	1.809×10^{-2}	1.158×10^{-4}	2.409×10^{-4}
	3.505	1.175×10^{-5}	1.973×10^{-2}	2.672×10^{-4}	3.483×10^{-4}
	3.605	1.713×10^{-5}	2.128×10^{-2}	3.746×10^{-4}	4.043×10^{-4}
	3.705	3.979×10^{-5}	2.268×10^{-2}	4.189×10^{-4}	3.985×10^{-4}
	3.805	4.863×10^{-5}	2.384×10^{-2}	3.830×10^{-4}	3.247×10^{-4}
	3.905	3.575×10^{-5}	2.470×10^{-2}	2.516×10^{-4}	1.804×10^{-4}
	4.000	3.790×10^{-6}	2.517×10^{-2}	2.646×10^{-5}	2.058×10^{-5}

Çizelge 3.4. Örnek 3.1 için Hibrit Metotlarda Mutlak Hata Değerleri.

		Hibrit Metotlar		
	x	PSOABC	PSOACO	PSOGSA
Eğitim Kümesi	2.000	0.000	0.000	0.000
	2.100	1.113×10^{-4}	1.377×10^{-4}	2.436×10^{-4}
	2.200	5.409×10^{-5}	1.324×10^{-4}	2.668×10^{-4}
	2.300	1.459×10^{-4}	6.433×10^{-5}	1.790×10^{-4}
	2.400	1.384×10^{-4}	8.617×10^{-5}	5.294×10^{-5}
	2.500	9.586×10^{-5}	8.385×10^{-5}	6.659×10^{-5}
	2.600	3.093×10^{-5}	1.240×10^{-4}	1.551×10^{-4}
	2.700	4.010×10^{-5}	1.348×10^{-4}	2.026×10^{-4}
	2.800	1.006×10^{-4}	1.201×10^{-4}	2.092×10^{-4}
	2.900	1.358×10^{-4}	8.657×10^{-5}	1.812×10^{-4}
	3.000	1.345×10^{-4}	4.248×10^{-5}	1.284×10^{-4}
	3.100	8.963×10^{-4}	4.233×10^{-6}	6.209×10^{-5}
	3.200	1.253×10^{-6}	4.668×10^{-5}	6.619×10^{-6}
	3.300	1.360×10^{-4}	7.962×10^{-5}	6.803×10^{-5}
	3.400	3.081×10^{-4}	9.976×10^{-5}	1.146×10^{-4}
	3.500	5.071×10^{-4}	1.058×10^{-4}	1.414×10^{-3}
	3.600	7.189×10^{-6}	9.855×10^{-5}	1.465×10^{-3}
	3.700	9.267×10^{-4}	8.053×10^{-5}	1.311×10^{-4}
	3.800	1.111×10^{-3}	5.601×10^{-5}	9.958×10^{-5}
	3.900	1.251×10^{-3}	3.070×10^{-5}	5.909×10^{-5}
	4.000	1.324×10^{-3}	1.148×10^{-5}	1.974×10^{-5}
Test Kümesi	2.000	0.000	0.000	0.000
	2.105	5.736×10^{-5}	1.400×10^{-4}	2.489×10^{-4}
	2.205	1.166×10^{-4}	1.300×10^{-4}	2.643×10^{-4}
	2.305	1.465×10^{-4}	6.027×10^{-5}	1.731×10^{-4}
	2.405	1.370×10^{-4}	2.078×10^{-5}	4.653×10^{-5}
	2.505	9.302×10^{-5}	8.655×10^{-5}	7.188×10^{-5}
	2.605	2.738×10^{-5}	1.252×10^{-4}	1.585×10^{-4}
	2.705	4.351×10^{-5}	1.346×10^{-4}	2.039×10^{-4}
	2.805	1.030×10^{-4}	1.188×10^{-4}	2.085×10^{-4}
	2.905	1.367×10^{-4}	8.454×10^{-5}	1.790×10^{-4}
	3.005	1.333×10^{-4}	4.014×10^{-5}	1.253×10^{-4}
	3.105	8.618×10^{-5}	6.508×10^{-6}	5.862×10^{-5}
	3.205	6.986×10^{-6}	4.859×10^{-5}	9.935×10^{-6}
	3.305	1.438×10^{-6}	8.095×10^{-5}	7.076×10^{-5}
	3.405	3.175×10^{-4}	1.004×10^{-4}	1.164×10^{-4}
	3.505	5.175×10^{-4}	1.058×10^{-4}	1.421×10^{-4}
	3.605	7.296×10^{-4}	9.788×10^{-5}	1.462×10^{-4}
	3.705	9.367×10^{-4}	7.942×10^{-5}	1.299×10^{-4}
	3.805	1.119×10^{-3}	5.471×10^{-5}	9.770×10^{-5}
	3.905	1.256×10^{-3}	2.953×10^{-5}	5.700×10^{-5}
	4.000	1.324×10^{-3}	1.148×10^{-5}	1.974×10^{-5}

Çizelge 3.5. Örnek 3.1 için Ortalama Karesel Hata Değerleri.

Kategori	**Metot**	**Eğitim Kümesi için Ortalama MSE**	**Test Kümesi için Ortalama MSE**
İteratif Metotlar	SM	8.283×10^{-1}	8.283×10^{-1}
	FDM	6.020×10^{-1}	6.012×10^{-1}
	Lobatto IIIa	$2.912 \times 10^{+1}$	$2.857 \times 10^{+1}$
Sezgisel Metotlar	mABC	$6.182 \times 10^{-5} \pm 6.685 \times 10^{-5}$	$6.169 \times 10^{-5} \pm 6.667 \times 10^{-5}$
	ACO	$9.613 \times 10^{-4} \pm 1.102 \times 10^{-3}$	$9.726 \times 10^{-4} \pm 1.115 \times 10^{-3}$
	GSA	$1.566 \times 10^{-6} \pm 2.043 \times 10^{-6}$	$1.559 \times 10^{-6} \pm 2.033 \times 10^{-6}$
	PSO	$3.233 \times 10^{-7} \pm 1.571 \times 10^{-7}$	$3.217 \times 10^{-7} \pm 1.563 \times 10^{-7}$
Hibrit Metot	PSO + ABC	$8.011 \times 10^{-6} \pm 5.346 \times 10^{-6}$	$8.184 \times 10^{-6} \pm 5.469 \times 10^{-6}$
	PSO + ACO	$2.253 \times 10^{-7} \pm 2.057 \times 10^{-7}$	$2.242 \times 10^{-7} \pm 2.047 \times 10^{-7}$
	PSO + GSA	$\mathbf{1.160 \times 10^{-7} \pm 1.103 \times 10^{-7}}$	$\mathbf{1.154 \times 10^{-7} \pm 1.098 \times 10^{-7}}$

Çizelge 3.6. Örnek 3.1 için saniye cinsinden geçen süre.

Metot	**Geçen Süre**
SM	2.132939
FDM	0.057241
Lobatto IIIa	0.348860
mABC	$5.048 \times 10^{-4} \pm 5.969 \times 10^{-5}$
ACO	$8.671 \times 10^{-4} \pm 8.731 \times 10^{-4}$
GSA	$6.272 \times 10^{-4} \pm 3.423 \times 10^{-4}$
PSO	$1.182 \times 10^{-3} \pm 2.085 \times 10^{-3}$
PSO + ABC	$5.023 \times 10^{-4} \pm 7.921 \times 10^{-5}$
PSO + ACO	$5.508 \times 10^{-4} \pm 8.507 \times 10^{-5}$
PSO + GSA	$5.026 \times 10^{-4} \pm 8.870 \times 10^{-5}$

Örnek 3.2. İkinci mertebeden lineer başlangıç değer problemi 3.2 denklemi ile verilmiştir.

$$\begin{cases} y''(t) + 5y'(t) + 4y(t) = 0, \quad t \in [0,1], \\ y(0) = 0 \\ y'(0) = -1 \end{cases} \tag{3.2}$$

Denklemin analitik çözümü $y(t) = \dfrac{\exp(t) - \exp(4t)}{3}$'dir. Denklem için nümerik çözümler sonucu mutlak hata değerleri sırasıyla iteratif, sezgisel ve hibrit metotlar olmak üzere gruplandırılan metotlar için sonuçlar sırasıyla Çizelge 3.7, Çizelge 3.8 ve Çizelge 3.9 ile sunulmuştur. Çizelgelerde görüldüğü üzere en iyi sonuç mABC yöntemi kullanılarak elde edilmiştir. Bu metot için $h = 0.1$ ve $h = 0.01$ için gerçek çözüm ve nümerik çözümlerin karşılaştırılması sırasıyla Şekil 3.2 *a* ve Şekil 3.2 *b* aracılığı ile gösterilmiştir.

Ek olarak tüm metotlar için ortalama karesel hata değerleri Çizelge 3.5 ile ve test kümesi için geçen süreler Çizelge 3.6 ile verilmiştir.

Çizelge 3.7. Örnek 3.2 için İteratif Metotlarda Mutlak Hata Değerleri.

		İteratif Metotlar		
	x	SM	FDM	Lobatto IIIa
Eğitim Kümesi	0.000	0.000	0.000	4.083×10^{-5}
	0.100	1.271×10^{-1}	3.677×10^{-3}	1.214×10^{-1}
	0.150	2.170×10^{-1}	6.770×10^{-3}	2.074×10^{-1}
	0.200	3.301×10^{-1}	1.107×10^{-2}	3.154×10^{-1}
	0.250	4.714×10^{-1}	1.695×10^{-2}	4.504×10^{-1}
	0.300	6.476×10^{-1}	2.490×10^{-2}	6.188×10^{-1}
	0.350	8.665×10^{-1}	3.555×10^{-2}	8.279×10^{-1}
	0.400	1.138	4.968×10^{-2}	1.087
	0.450	1.473	6.832×10^{-2}	1.407
	0.500	1.887	9.276×10^{-2}	1.803
	0.550	2.397	1.246×10^{-2}	2.290
	0.600	3.024	1.660×10^{-2}	2.890
	0.650	3.796	2.196×10^{-1}	3.627
	0.700	4.743	2.887×10^{-1}	4.532
	0.750	5.906	3.775×10^{-1}	5.643
	0.800	7.332	4.913×10^{-1}	7.006
	0.850	9.080	6.369×10^{-1}	8.676
	0.900	11.220	8.226×10^{-1}	10.720
	0.950	13.840	1.059	13.230
	1.000	1.705	1.360	16.290
Test Kümesi	0.000	0.000	0.000	0.000
	0.005	4.993×10^{-3}	6.294×10^{-5}	4.770×10^{-3}
	0.105	1.352×10^{-1}	2.002×10^{-3}	1.292×10^{-3}
	0.155	2.272×10^{-1}	3.629×10^{-3}	2.171×10^{-1}
	0.205	3.428×10^{-1}	5.891×10^{-3}	3.276×10^{-1}
	0.255	4.874×10^{-1}	8.985×10^{-3}	4.657×10^{-1}
	0.305	6.674×10^{-1}	1.317×10^{-2}	6.377×10^{-1}
	0.355	8.911×10^{-1}	1.877×10^{-2}	8.514×10^{-1}
	0.405	1.168	2.621×10^{-2}	4.083×10^{-5}
	0.455	1.511	3.603×10^{-2}	4.083×10^{-5}
	0.505	1.933	4.890×10^{-2}	1.116
	0.555	2.454	6.571×10^{-2}	2.345
	0.605	3.095	8.754×10^{-2}	2.957
	0.655	3.882	1.158×10^{-1}	3.709
	0.705	4.849	1.523×10^{-1}	4.633
	0.755	6.036	1.992×10^{-1}	5.767
	0.805	7.491	2.594×10^{-1}	7.158
	0.855	9.275	3.365×10^{-1}	8.862
	0.905	11.460	4.349×10^{-1}	10.950
	0.955	14.140	5.603×10^{-1}	13.510
	1.000	17.050	7.022×10^{-1}	16.290

Çizelge 3.8. Örnek 3.2 için Sezgisel Metotlarda Mutlak Hata Değerleri.

		Sezgisel Metotlar			
	x	mABC	ACO	GSA	PSO
Eğitim Kümesi	0.000	0.000	0.000	0.000	0.000
	0.100	4.476×10^{-3}	1.486×10^{-2}	3.708×10^{-2}	3.859×10^{-2}
	0.150	1.003×10^{-2}	3.711×10^{-3}	8.803×10^{-2}	9.116×10^{-2}
	0.200	1.787×10^{-2}	7.252×10^{-2}	1.657×10^{-1}	1.708×10^{-1}
	0.250	2.819×10^{-2}	1.241×10^{-1}	2.751×10^{-1}	2.823×10^{-1}
	0.300	4.145×10^{-2}	1.954×10^{-1}	4.224×10^{-1}	4.316×10^{-1}
	0.350	5.837×10^{-2}	2.908×10^{-1}	6.153×10^{-1}	6.262×10^{-1}
	0.400	7.997×10^{-2}	4.153×10^{-1}	8.632×10^{-1}	8.752×10^{-1}
	0.450	1.075×10^{-1}	5.749	1.178	1.190
	0.500	1.422×10^{-1}	7.769	1.574	1.585
	0.550	1.856×10^{-1}	1.030	2.068	2.077
	0.600	2.392×10^{-1}	1.344	2.682	2.687
	0.650	3.050×10^{-1}	1.732	3.443	3.442
	0.700	3.860×10^{-1}	2.209	4.382	4.373
	0.750	4.862×10^{-1}	2.793	5.540	5.520
	0.800	6.098×10^{-1}	3.509	6.965	6.931
	0.850	7.611×10^{-1}	4.384	8.715	8.665
	0.900	9.464×10^{-1}	54.53	10.86	10.84
	0.950	1.174	67.620	13.500	13.400
	1.000	1.453	8.368	16.730	16.610
Test Kümesi	0.000	0.000	0.000	0.000	0.000
	0.005	1.135×10^{-5}	4.079×10^{-5}	8.452×10^{-5}	8.878×10^{-5}
	0.105	4.933×10^{-3}	1.657×10^{-2}	4.110×10^{-2}	4.275×10^{-2}
	0.155	1.071×10^{-2}	4.002×10^{-2}	9.452×10^{-2}	9.784×10^{-2}
	0.205	1.878×10^{-2}	7.689×10^{-2}	1.751×10^{-1}	1.804×10^{-1}
	0.255	2.938×10^{-2}	1.303×10^{-1}	2.880×10^{-1}	2.954×10^{-1}
	0.305	4.297×10^{-2}	2.038×10^{-1}	4.395×10^{-1}	4.489×10^{-1}
	0.355	6.031×10^{-2}	3.019×10^{-1}	6.374×10^{-1}	6.484×10^{-1}
	0.405	8.244×10^{-2}	4.296×10^{-1}	8.915×10^{-1}	9.035×10^{-1}
	0.455	1.106×10^{-1}	5.931	1.214	1.232
	0.505	1.462×10^{-1}	7.997×10^{-1}	1.618	1.629
	0.555	1.905×10^{-1}	1.058	2.123	2.132
	0.605	2.452×10^{-1}	1.379	2.751	2.756
	0.655	3.124×10^{-1}	1.775	3.528	3.527
	0.705	3.951×10^{-1}	2.262	4.487	4.478
	0.755	4.974×10^{-1}	2.859	5.670	5.649
	0.805	6.236×10^{-1}	3.589	7.124	7.089
	0.855	7.780×10^{-1}	4.481	8.911	8.859
	0.905	9.672×10^{-1}	5.573	11.10	11.03
	0.955	1.200	6.909	13.790	13.700
	1.000	1.453	8.368	16.730	16.610

Çizelge 3.9. Örnek 3.2 için Hibrit Metotlarda Mutlak Hata Değerleri.

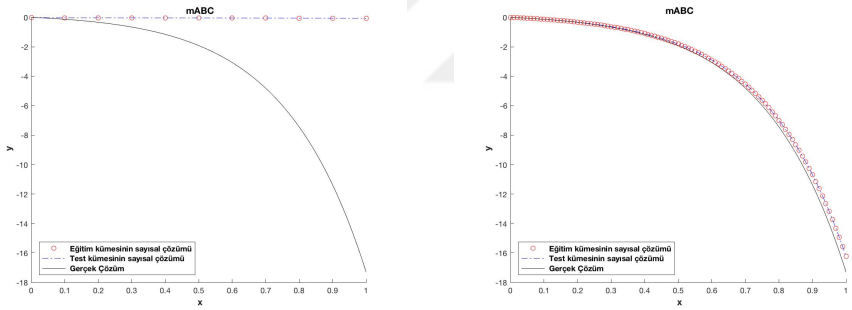
		Hibrit Metotlar		
	x	PSOABC	PSOACO	PSOGSA
Eğitim Kümesi	0.000	0.000	0.000	0.000
	0.100	4.025×10^{-2}	3.833×10^{-2}	3.859×10^{-2}
	0.150	9.459×10^{-2}	9.059×10^{-2}	9.116×10^{-2}
	0.200	1.763×10^{-1}	1.698×10^{-1}	1.708×10^{-1}
	0.250	2.900×10^{-1}	2.807×10^{-1}	2.823×10^{-1}
	0.300	4.415×10^{-1}	4.294×10^{-1}	4.343×10^{-1}
	0.350	6.380×10^{-1}	6.232×10^{-1}	6.262×10^{-1}
	0.400	8.884×10^{-1}	8.715×10^{-1}	8.752×10^{-1}
	0.450	1.204	1.186	1.190
	0.500	1.599	1.580	1.585
	0.550	2.089	2.071	2.077
	0.600	2.697	2.680	2.687
	0.650	3.447	3.435	3.442
	0.700	4.372	4.366	4.373
	0.750	5.511	5.512	5.520
	0.800	6.911	6.923	6.931
	0.850	8.631	8.657	8.665
	0.900	10.740	10.790	10.790
	0.950	13.330	13.400	13.400
	1.000	16.510	16.600	16.610
Test Kümesi	0.000	0.000	5.311	4.083×10^{-5}
	0.005	9.362×10^{-5}	8.815×10^{-5}	8.878×10^{-5}
	0.105	4.457×10^{-2}	4.246×10^{-2}	4.275×10^{-2}
	0.155	1.015×10^{-1}	9.723×10^{-2}	9.784×10^{-2}
	0.205	1.862×10^{-1}	1.794×10^{-1}	1.804×10^{-1}
	0.255	3.034×10^{-1}	2.938×10^{-1}	2.954×10^{-1}
	0.305	4.590×10^{-1}	4.466×10^{-1}	4.489×10^{-1}
	0.355	6.604×10^{-1}	6.454×10^{-1}	6.484×10^{-1}
	0.405	9.168×10^{-1}	8.997×10^{-1}	9.035×10^{-1}
	0.455	1.240	1.221	1.226
	0.505	1.643	1.624	1.629
	0.555	2.144	2.126	2.132
	0.605	2.765	2.749	2.756
	0.655	3.531	3.519	3.527
	0.705	4.476	4.470	4.478
	0.755	5.638	5.641	5.649
	0.805	7.068	7.081	7.089
	0.855	8.823	8.851	8.859
	0.905	10.980	11.020	11.030
	0.955	13.620	13.690	13.700
	1.000	16.510	16.600	16.610

Çizelge 3.10. Örnek 3.2 için Ortalama Karesel Hata Değerleri.

Kategori	**Metotlar**	**Eğitim Kümesi için Ortalama MSE**	**Test Kümesi için Ortalama MSE**
İteratif Metotlar	SM	$3.534 \times 10^{+1}$	$3.534 \times 10^{+1}$
	FDM	1.890×10^{-1}	5.021×10^{-2}
	Lobatto IIIa	$3.651 \times 10^{+1}$	$3.101 \times 10^{+1}$
Sezgisel Metotlar	mABC	$\mathbf{1.334 \times 10^{-1} \pm 9.890 \times 10^{-2}}$	$\mathbf{1.375 \times 10^{-1} \pm 1.020 \times 10^{-1}}$
	ACO	$2.633 \times 10^{+1} \pm 7.882 \times 10^{+0}$	$2.715 \times 10^{+1} \pm 8.127 \times 10^{+0}$
	GSA	$3.284 \times 10^{+1} \pm 8.726 \times 10^{-2}$	$3.387 \times 10^{+1} \pm 8.987 \times 10^{-2}$
	PSO	$3.254 \times 10^{+1} \pm 1.226 \times 10^{-1}$	$3.355 \times 10^{+1} \pm 1.263 \times 10^{-1}$
Hibrit Metotlar	PSO + ABC	$3.249 \times 10^{+1} \pm 3.164 \times 10^{-1}$	$3.350 \times 10^{+1} \pm 3.269 \times 10^{-1}$
	PSO + ACO	$3.264 \times 10^{+1} \pm 2.453 \times 10^{-1}$	$3.366 \times 10^{+1} \pm 2.521 \times 10^{-1}$
	PSO + GSA	$3.242 \times 10^{+1} \pm 1.063 \times 10^{-1}$	$3.342 \times 10^{+1} \pm 1.096 \times 10^{-1}$

Çizelge 3.11. Örnek 3.2 için saniye cinsinden geçen süre.

Metot	**Geçen Süre (sn)**
SM	2.201824
FDM	0.053136
Lobatto IIIa	0.529050
mABC	$3.306 \times 10^{-4} \pm 1.237 \times 10^{-4}$
ACO	$5.287 \times 10^{-4} \pm 7.786 \times 10^{-4}$
GSA	$4.618 \times 10^{-4} \pm 4.324 \times 10^{-4}$
PSO	$4.062 \times 10^{-4} \pm 3.454 \times 10^{-4}$
PSO + ABC	$3.993 \times 10^{-4} \pm 3.328 \times 10^{-4}$
PSO + ACO	$3.249 \times 10^{-4} \pm 1.055 \times 10^{-4}$
PSO + GSA	$6.513 \times 10^{-4} \pm 6.697 \times 10^{-4}$



(*a*) $h = 0.1$ (*b*) $h = 0.01$

Şekil 3.2. Örnek 3.2 için mABC algoritması kullanılarak 1000 iterasyonda ikinci mertebeden başlangıç değer probleminin yaklaşık çözüm grafiği.

Örnek 3.3. İkinci mertebeden lineer Dirichlet sınır değer problemi 3.3 denklemi ile verilmiştir [82].

$$\begin{cases} y''(t) = \dfrac{-2t}{1+t^2}y'(t) + y(t) + \dfrac{2}{1+t^2} - \log(1+t^2), \quad t \in [0,1], \\ y(0) = 0 \\ y(1) = \log(2) \end{cases} \tag{3.3}$$

Verilen denklem için kesin çözüm $y(t) = \log(1+t^2)$'dir. Denklemin iteratif metotlar kullanılarak ulaşılan nümerik çözümleri için mutlak hata değerleri Çizelge 3.12, optimizasyon metotları ile elde edilen nümerik çözümleri için mutlak hata değerleri 3.13 ve bu metotların hibritlenmesiyle elde edilen metotlar

kullanılmasıyla ulaşılan nümerik çözümler için mutlak hata değerleri Çizelge 3.14 ile verilmiştir.

Bununla birlikte en iyi sonuca ulaşan mABC metodu için sırasıyla $h = 0.1$, $h = 0.01$ için nümerik çözümler ve kesin çözümler Şekil 3.3 *a* ve Şekil 3.3 *b* ile sunulmuştur. Metotların tümü için ise ortalama karesel hata değerleri Çizelge 3.15 ile ve test kümesi için işlem süreleri Çizelge 3.16 ile gösterilmiştir.

Çizelge 3.12. Örnek 3.3 için İteratif Metotlarda Mutlak Hata Değerleri.

		İteratif Metotlar		
	x	SM	FDM	Lobatto IIIa
Eğitim Kümesi	0.010	3.448×10^{-12}	2.657×10^{-4}	1.983×10^{-8}
	0.050	1.517×10^{-11}	1.278×10^{-3}	9.910×10^{-8}
	0.100	2.537×10^{-11}	2.341×10^{-3}	1.981×10^{-7}
	0.150	3.124×10^{-11}	3.078×10^{-3}	2.974×10^{-7}
	0.200	3.372×10^{-11}	3.430×10^{-3}	3.970×10^{-7}
	0.250	3.388×10^{-11}	3.385×10^{-3}	4.978×10^{-7}
	0.300	3.275×10^{-11}	2.969×10^{-3}	5.989×10^{-7}
	0.350	3.118×10^{-11}	2.237×10^{-3}	6.999×10^{-7}
	0.400	2.973×10^{-11}	1.266×10^{-3}	7.968×10^{-7}
	0.450	2.871×10^{-11}	1.445×10^{-4}	8.888×10^{-7}
	0.500	2.813×10^{-11}	1.033×10^{-3}	9.778×10^{-7}
	0.550	2.785×10^{-11}	2.173×10^{-3}	1.066×10^{-7}
	0.600	2.762×10^{-11}	3.190×10^{-3}	1.153×10^{-7}
	0.650	2.713×10^{-11}	4.004×10^{-3}	1.239×10^{-6}
	0.700	2.609×10^{-11}	4.550×10^{-3}	1.324×10^{-6}
	0.750	2.427×10^{-11}	4.773×10^{-3}	1.409×10^{-6}
	0.800	2.151×10^{-11}	4.632×10^{-3}	1.494×10^{-6}
	0.850	1.770×10^{-11}	4.097×10^{-3}	1.578×10^{-6}
	0.900	1.282×10^{-11}	3.150×10^{-3}	1.662×10^{-6}
	0.950	6.898×10^{-12}	1.784×10^{-3}	1.746×10^{-6}
	1.000	0.000	0.000	1.830×10^{-6}
Test Kümesi	0.000	0.000	0.000	0.000
	0.005	1.093×10^{-13}	1.333×10^{-4}	9.914×10^{-9}
	0.105	1.6305×10^{-12}	2.431×10^{-3}	2.077×10^{-7}
	0.155	1.971×10^{-12}	3.131×10^{-3}	3.060×10^{-7}
	0.205	2.106×10^{-12}	3.444×10^{-3}	4.027×10^{-7}
	0.255	2.104×10^{-12}	3.36×10^{-3}	5.012×10^{-7}
	0.305	2.026×10^{-12}	2.909×10^{-3}	5.995×10^{-7}
	0.355	1.925×10^{-12}	2.149×10^{-3}	6.975×10^{-7}
	0.405	1.835×10^{-12}	1.159×10^{-3}	7.941×10^{-7}
	0.455	1.772×10^{-12}	2.745×10^{-5}	8.868×10^{-7}
	0.505	1.737×10^{-12}	1.151×10^{-3}	9.772×10^{-7}
	0.555	1.719×10^{-12}	2.282×10^{-3}	1.066×10^{-6}
	0.605	1.7039×10^{-12}	3.282×10^{-3}	1.155×10^{-6}
	0.655	1.670×10^{-12}	4.072×10^{-3}	1.242×10^{-6}
	0.705	1.601×10^{-12}	4.588×10^{-3}	1.328×10^{-6}
	0.755	1.484×10^{-12}	4.776×10^{-3}	1.414×10^{-6}
	0.805	1.307×10^{-12}	4.597×10^{-3}	1.499×10^{-6}
	0.855	1.065×10^{-12}	4.021×10^{-3}	1.584×10^{-6}
	0.905	7.579×10^{-13}	3.033×10^{-3}	1.669×10^{-6}
	0.955	3.8610×10^{-13}	1.6257×10^{-3}	1.754×10^{-6}
	1.000	0.000	0.000	1.830×10^{-6}

Çizelge 3.13. Örnek 3.3 için Sezgisel Metotlarda Mutlak Hata Değerleri.

		Sezgisel Metotlar			
	x	mABC	ACO	GSA	PSO
Eğitim Kümesi	0.010	3.814×10^{-8}	1.142×10^{-4}	1.671×10^{-6}	2.294×10^{-6}
	0.050	3.119×10^{-5}	6.072×10^{-4}	1.464×10^{-6}	1.167×10^{-4}
	0.100	7.458×10^{-6}	1.240×10^{-3}	5.331×10^{-4}	4.260×10^{-4}
	0.150	7.241×10^{-6}	1.819×10^{-3}	9.637×10^{-4}	7.578×10^{-4}
	0.200	2.622×10^{-6}	7.834×10^{-3}	1.306×10^{-3}	1.000×10^{-3}
	0.250	3.491×10^{-6}	2.668×10^{-3}	1.487×10^{-3}	1.095×10^{-3}
	0.300	7.922×10^{-6}	2.923×10^{-2}	1.480×10^{-3}	1.024×10^{-3}
	0.350	8.747×10^{-6}	3.077×10^{-2}	1.296×10^{-3}	8.047×10^{-4}
	0.400	5.836×10^{-6}	3.146×10^{-2}	9.720×10^{-4}	4.743×10^{-4}
	0.450	5.852×10^{-7}	3.146×10^{-2}	5.604×10^{-4}	8.405×10^{-5}
	0.500	4.840×10^{-6}	3.087×10^{-3}	1.202×10^{-4}	3.119×10^{-4}
	0.550	8.396×10^{-6}	2.973×10^{-3}	2.919×10^{-4}	6.632×10^{-4}
	0.600	8.878×10^{-6}	2.805×10^{-3}	6.281×10^{-4}	9.288×10^{-4}
	0.650	6.262×10^{-6}	2.582×10^{-3}	8.532×10^{-4}	1.081×10^{-3}
	0.700	1.635×10^{-6}	2.300×10^{-3}	9.488×10^{-4}	1.109×10^{-3}
	0.750	3.250×10^{-6}	1.963×10^{-3}	9.139×10^{-4}	1.016×10^{-3}
	0.800	6.611×10^{-6}	1.577×10^{-3}	7.662×10^{-4}	8.257×10^{-4}
	0.850	7.303×10^{-6}	1.158×10^{-3}	5.412×10^{-4}	5.728×10^{-4}
	0.900	5.320×10^{-6}	7.316×10^{-3}	2.913×10^{-4}	3.083×10^{-4}
	0.950	2.021×10^{-6}	3.308×10^{-4}	8.397×10^{-5}	9.376×10^{-5}
	1.000	0.000	0.000	0.000	0.000
Test Kümesi	0.000	0.000	0.000	0.000	3.978×10^{-5}
	0.005	8.561×10^{-8}	5.649×10^{-5}	3.187×10^{-6}	3.116×10^{-6}
	0.105	7.674×10^{-6}	1.301×10^{-3}	5.768×10^{-4}	4.605×10^{-4}
	0.155	6.941×10^{-6}	1.872×10^{-3}	1.003×10^{-3}	7.875×10^{-4}
	0.205	2.021×10^{-6}	2.343×10^{-3}	1.332×10^{-3}	1.017×10^{-3}
	0.255	4.055×10^{-6}	2.699×10^{-3}	1.495×10^{-3}	1.095×10^{-3}
	0.305	8.183×10^{-6}	2.943×10^{-3}	1.469×10^{-3}	1.008×10^{-3}
	0.355	8.611×10^{-6}	3.087×10^{-3}	1.269×10^{-3}	7.759×10^{-4}
	0.405	5.383×10^{-6}	3.149×10^{-3}	9.338×10^{-4}	4.371×10^{-4}
	0.455	1.306×10^{-8}	3.143×10^{-3}	5.167×10^{-4}	4.382×10^{-4}
	0.505	5.308×10^{-6}	3.078×10^{-3}	7.686×10^{-5}	3.498×10^{-4}
	0.555	8.593×10^{-6}	2.959×10^{-3}	3.297×10^{-4}	6.942×10^{-5}
	0.605	8.745×10^{-6}	2.785×10^{-3}	6.560×10^{-4}	9.495×10^{-4}
	0.655	5.865×10^{-6}	2.556×10^{-3}	8.688×10^{-4}	1.089×10^{-3}
	0.705	1.126×10^{-6}	2.269×10^{-3}	9.510×10^{-4}	1.105×10^{-5}
	0.755	3.680×10^{-6}	1.926×10^{-3}	9.037×10^{-4}	1.001×10^{-3}
	0.805	6.811×10^{-6}	1.536×10^{-3}	7.464×10^{-4}	8.025×10^{-4}
	0.855	7.213×10^{-6}	1.115×10^{-3}	5.163×10^{-4}	5.460×10^{-4}
	0.905	5.015×10^{-6}	6.897×10^{-4}	2.674×10^{-4}	2.834×10^{-4}
	0.955	1.710×10^{-6}	2.937×10^{-4}	6.859×10^{-5}	7.771×10^{-5}
	1.000	0.000	0.000	0.000	0.000

Çizelge 3.14. Örnek 3.3 için Hibrit Metotlarda Mutlak Hata Değerleri.

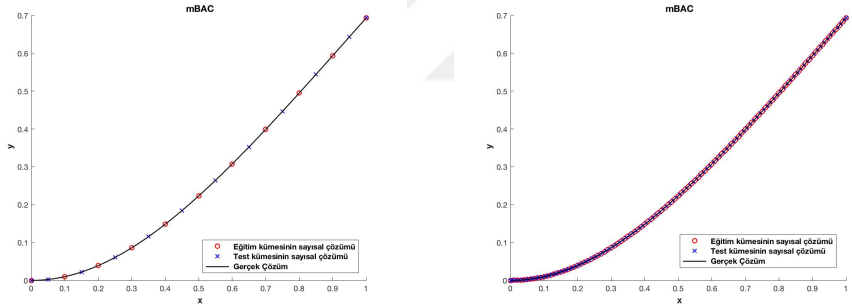
		Hibrit Metotlar		
	x	PSOABC	PSOACO	PSOGSA
Eğitim Kümesi	0.010	8.982×10^{-4}	7.797×10^{-3}	1.139×10^{-6}
	0.050	3.815×10^{-3}	9.776×10^{-3}	1.260×10^{-4}
	0.100	6.118×10^{-3}	1.896×10^{-2}	4.313×10^{-4}
	0.150	7.183×10^{-3}	2.763×10^{-2}	7.550×10^{-4}
	0.200	7.254×10^{-3}	1.085×10^{-2}	9.917×10^{-4}
	0.250	6.547×10^{-3}	1.045×10^{-2}	1.085×10^{-3}
	0.300	5.254×10^{-3}	9.276×10^{-3}	1.021×10^{-3}
	0.350	3.549×10^{-3}	1.728×10^{-2}	8.152×10^{-4}
	0.400	1.588×10^{-3}	5.380×10^{-3}	5.042×10^{-4}
	0.450	4.841×10^{-4}	3.021×10^{-3}	1.369×10^{-4}
	0.500	2.535×10^{-3}	6.099×10^{-4}	2.349×10^{-4}
	0.550	4.442×10^{-3}	1.704×10^{-3}	5.640×10^{-4}
	0.600	6.090×10^{-3}	3.782×10^{-3}	8.120×10^{-3}
	0.650	7.373×10^{-3}	5.497×10^{-3}	9.536×10^{-3}
	0.700	8.1999×10^{-3}	6.736×10^{-3}	9.789×10^{-4}
	0.750	8.487×10^{-3}	7.403×10^{-3}	8.934×10^{-4}
	0.800	8.168×10^{-3}	7.417×10^{-3}	7.183×10^{-4}
	0.850	7.190×10^{-3}	6.717×10^{-3}	4.890×10^{-4}
	0.900	5.514×10^{-3}	5.261×10^{-3}	2.530×10^{-4}
	0.950	3.120×10^{-3}	3.024×10^{-3}	6.825×10^{-5}
	1.000	0.000	0.000	0.000
Test Kümesi	0.000	0.000	0.000	0.000
	0.005	4.580×10^{-4}	3.886×10^{-3}	1.280×10^{-6}
	0.105	6.276×10^{-3}	1.985×10^{-2}	4.6510×10^{-4}
	0.155	7.231×10^{-3}	2.847×10^{-2}	7.839×10^{-4}
	0.205	7.215×10^{-3}	1.085×10^{-2}	1.008×10^{-3}
	0.255	6.441×10^{-3}	1.036×10^{-2}	1.086×10^{-3}
	0.305	5.100×10^{-3}	9.123×10^{-3}	1.006×10^{-3}
	0.355	3.362×10^{-3}	7.323×10^{-3}	7.880×10^{-5}
	0.405	1.384×10^{-3}	5.151×10^{-3}	4.691×10^{-4}
	0.455	6.924×10^{-4}	2.780×10^{-3}	9.913×10^{-5}
	0.505	2.735×10^{-3}	3.717×10^{-4}	2.705×10^{-4}
	0.555	4.621×10^{-3}	1.925×10^{-3}	5.930×10^{-4}
	0.605	6.236×10^{-3}	3.972×10^{-3}	8.312×10^{-4}
	0.655	7.478×10^{-3}	5.644×10^{-3}	9.614×10^{-4}
	0.705	8.254×10^{-3}	6.830×10^{-3}	9.751×10^{-4}
	0.755	8.483×10^{-3}	7.435×10^{-3}	8.795×10^{-4}
	0.805	8.100×10^{-3}	7.380×10^{-3}	6.972×10^{-4}
	0.855	7.054×10^{-3}	6.606×10^{-3}	4.648×10^{-4}
	0.905	5.307×10^{-3}	5.073×10^{-3}	2.312×10^{-4}
	0.955	2.840×10^{-3}	2.757×10^{-3}	5.505×10^{-5}
	1.000	0.000	0.000	0.000

Çizelge 3.15. Örnek 3.3 için Ortalama Karesel Hata Değerleri.

Kategori	**Metot**	**Eğitim Kümesi için Ortalama MSE**	**Test Kümesi için Ortalama MSE**
İteratif Metotlar	SM	$\mathbf{3.046 \times 10^{-24}}$	$\mathbf{3.202 \times 10^{-24}}$
	FDM	9.014×10^{-6}	5.654×10^{-2}
	Lobatto IIIa	3.629979×10^{-12}	1.175×10^{-12}
Sezgisel Metotlar	mABC	$\mathbf{1.045 \times 10^{-9} \pm 1.736 \times 10^{-9}}$	$\mathbf{1.035 \times 10^{-9} \pm 1.719 \times 10^{-9}}$
	ACO	$6.208 \times 10^{-6} \pm 6.445 \times 10^{-6}$	$6.147 \times 10^{-6} \pm 6.381 \times 10^{-6}$
	GSA	$1.087 \times 10^{-6} \pm 1.116 \times 10^{-6}$	$1.076 \times 10^{-6} \pm 1.105 \times 10^{-6}$
	PSO	$5.836 \times 10^{-7} \pm 3.145 \times 10^{-8}$	$5.779 \times 10^{-7} \pm 3.114 \times 10^{-8}$
Hibrit Metotlar	PSO + ABC	$4.557 \times 10^{-5} \pm 5.351 \times 10^{-5}$	$4.454 \times 10^{-5} \pm 5.116 \times 10^{-5}$
	PSO + ACO	$4.493 \times 10^{-3} \pm 4.930 \times 10^{-3}$	$4.694 \times 10^{-3} \pm 5.162 \times 10^{-3}$
	PSO + GSA	$5.272 \times 10^{-7} \pm 3.993 \times 10^{-8}$	$5.220 \times 10^{-7} \pm 3.954 \times 10^{-8}$

Çizelge 3.16. Örnek 3.3 için saniye cinsinden geçen süre.

Metot	**Geçen Süre**
SM	2.842515
FDM	0.035005
Lobatto IIIa	3.629979
mABC	$3.001 \times 10^{-4} \pm 1.011 \times 10^{-4}$
ACO	$4.522 \times 10^{-4} \pm 4.053 \times 10^{-4}$
GSA	$4.890 \times 10^{-4} \pm 4.209 \times 10^{-4}$
PSO	$4.874 \times 10^{-4} \pm 4.437 \times 10^{-4}$
PSO + ABC	$1.490 \times 10^{-3} \pm 2.233 \times 10^{-4}$
PSO + ACO	$1.570 \times 10^{-3} \pm 1.419 \times 10^{-4}$
PSO + GSA	$4.983 \times 10^{-4} \pm 4.651 \times 10^{-4}$



(*a*) $h = 0.1$ (*b*) $h = 0.01$

Şekil 3.3. Örnek 3.3 için mABC algoritması kullanılarak 1000 iterasyonda ikinci mertebeden lineer Dirichlet sınır değer probleminin yaklaşık çözüm grafiği.

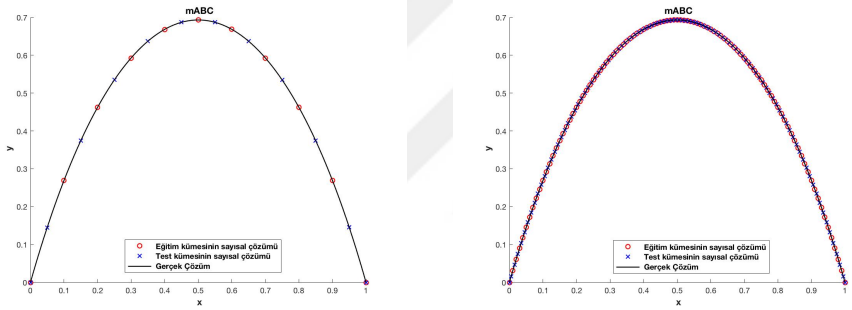
Örnek 3.4. İkinci mertebeden lineer olmayan Dirichlet sınır değer problemi 3.4 denklemi ile verilmiştir [31].

$$\begin{cases} y''(t) + \pi^2 \exp(-y(t)) = 0, \quad t \in [0,1], \\ y(0) = 0 \\ y(1) = 0 \end{cases} \tag{3.4}$$

Denklemin analitik çözümü $y(t) = \log(1+\sin(\pi t))$'dir. İteratif metotlar ile bulunan nümerik çözümlerde elde edilen mutlak hatalar Çizelge 3.17, optimizasyon metotlar ile bulunan nümerik çözümlerdeki mutlak hatalar Çizelge 3.18 ve hibrit

metotlar ile bulunan nümerik çözümlerde oluşan mutlak hatalar Çizelge 3.19 ile gösterilmiştir.

Verilen Çizelgelerde de görüldüğü üzere, en iyi sonucu mABC metodu vermiştir. Şekil 3.4 *a* ve Şekil 3.4 *b* olmak üzere 1000 iterasyon için $h = 0.1$ ve $h = 0.01$ farklı adım uzunlukları ile elde edilen nümerik çözümler verilmiştir. Bunlara ek olarak metotların ortalama karesel hata değerleri Çizelge 3.20 ile ve test kümesi için geçen süreler Çizelge 3.21 ile gösterilmiştir.



(*a*) $h = 0.1$ (*b*) $h = 0.01$

Şekil 3.4. Örnek 3.4 için mABC algoritması kullanılarak 1000 iterasyonda ikinci mertebeden lineer olmayan Dirichlet sınır değer probleminin yaklaşık çözüm grafiği.

Çizelge 3.17. Örnek 3.4 için İteratif Metotlarda Mutlak Hata Değerleri.

		İteratif Metotlar		
	x	SM	FDM	Lobatto IIIa
Eğitim Kümesi	0.010	1.695×10^{-11}	3.051×10^{-2}	9.724×10^{-9}
	0.050	6.926×10^{-11}	2.804×10^{-2}	4.860×10^{-8}
	0.100	1.105×10^{-10}	2.548×10^{-2}	9.736×10^{-8}
	0.150	1.355×10^{-10}	2.340×10^{-2}	1.460×10^{-7}
	0.200	1.508×10^{-10}	2.172×10^{-2}	1.934×10^{-7}
	0.250	1.602×10^{-10}	2.037×10^{-2}	2.380×10^{-7}
	0.300	1.659×10^{-10}	1.931×10^{-2}	2.781×10^{-7}
	0.350	1.693×10^{-10}	1.852×10^{-2}	3.118×10^{-7}
	0.400	1.712×10^{-10}	1.797×10^{-2}	3.372×10^{-7}
	0.450	1.721×10^{-10}	1.764×10^{-2}	3.531×10^{-7}
	0.500	1.724×10^{-10}	1.754×10^{-2}	3.585×10^{-7}
	0.550	1.720×10^{-10}	1.764×10^{-2}	3.531×10^{-7}
	0.600	1.709×10^{-10}	1.797×10^{-2}	3.372×10^{-7}
	0.650	1.688×10^{-10}	1.852×10^{-2}	3.118×10^{-7}
	0.700	1.653×10^{-10}	1.931×10^{-2}	2.781×10^{-7}
	0.750	1.594×10^{-10}	2.037×10^{-2}	2.380×10^{-7}
	0.800	1.500×10^{-10}	2.172×10^{-2}	1.934×10^{-7}
	0.850	1.347×10^{-10}	2.340×10^{-2}	1.460×10^{-7}
	0.900	1.097×10^{-10}	2.548×10^{-2}	9.736×10^{-7}
	0.950	6.873×10^{-11}	2.804×10^{-2}	4.860×10^{-8}
	1.000	1.926×10^{-16}	3.120×10^{-2}	2.220×10^{-16}
Test Kümesi	0.000	0.000	1.565×10^{-2}	0.000
	0.005	8.705×10^{-12}	1.548×10^{-2}	4.751×10^{-9}
	0.105	1.136×10^{-10}	1.266×10^{-2}	1.005×10^{-7}
	0.155	1.374×10^{-10}	1.164×10^{-2}	1.486×10^{-7}
	0.205	1.520×10^{-10}	1.081×10^{-2}	1.954×10^{-7}
	0.255	1.609×10^{-10}	1.015×10^{-2}	2.393×10^{-7}
	0.305	1.663×10^{-10}	9.631×10^{-3}	2.786×10^{-7}
	0.355	1.695×10^{-10}	9.245×10^{-3}	3.113×10^{-7}
	0.405	1.713×10^{-10}	8.979×10^{-3}	3.357×10^{-7}
	0.455	1.722×10^{-10}	8.827×10^{-3}	3.505×10^{-7}
	0.505	1.724×10^{-10}	8.784×10^{-3}	3.548×10^{-7}
	0.555	1.719×10^{-10}	8.849×10^{-3}	3.494×10^{-7}
	0.605	1.707×10^{-10}	9.023×10^{-3}	3.315×10^{-7}
	0.655	1.685×10^{-10}	9.312×10^{-3}	3.054×10^{-7}
	0.705	1.648×10^{-10}	9.723×10^{-3}	2.712×10^{-7}
	0.755	1.587×10^{-10}	1.027×10^{-2}	2.308×10^{-7}
	0.805	1.488×10^{-10}	1.096×10^{-2}	1.862×10^{-7}
	0.855	1.327×10^{-10}	1.183×10^{-2}	1.390×10^{-7}
	0.905	1.065×10^{-10}	1.289×10^{-2}	9.089×10^{-8}
	0.955	6.337×10^{-11}	1.421×10^{-2}	4.286×10^{-8}
	1.000	1.926×10^{-16}	1.565×10^{-2}	2.220×10^{-14}

Çizelge 3.18. Örnek 3.4 için Sezgisel Metotlarda Mutlak Hata Değerleri.

		Sezgisel Metotlar			
	x	mABC	ACO	GSA	PSO
Eğitim Kümesi	0.010	1.126×10^{-6}	6.545×10^{-4}	1.111×10^{-3}	1.187×10^{-3}
	0.050	4.079×10^{-5}	2.965×10^{-3}	2.382×10^{-3}	2.863×10^{-3}
	0.100	5.807×10^{-5}	4.970×10^{-3}	1.363×10^{-3}	2.292×10^{-4}
	0.150	1.991×10^{-6}	5.919×10^{-2}	8.773×10^{-3}	6.932×10^{-3}
	0.200	8.330×10^{-5}	5.891×10^{-3}	1.800×10^{-2}	1.549×10^{-2}
	0.250	1.256×10^{-4}	5.044×10^{-3}	2.764×10^{-2}	2.459×10^{-2}
	0.300	1.026×10^{-4}	3.577×10^{-3}	3.666×10^{-2}	3.324×10^{-2}
	0.350	2.504×10^{-5}	1.714×10^{-3}	4.428×10^{-2}	4.069×10^{-2}
	0.400	7.329×10^{-5}	2.940×10^{-4}	4.994×10^{-2}	4.642×10^{-2}
	0.450	1.522×10^{-4}	2.197×10^{-3}	5.328×10^{-2}	5.004×10^{-2}
	0.500	1.810×10^{-4}	3.761×10^{-3}	5.409×10^{-2}	5.136×10^{-2}
	0.550	1.485×10^{-4}	4.802×10^{-3}	5.235×10^{-2}	5.029×10^{-2}
	0.600	6.630×10^{-5}	5.218×10^{-3}	4.814×10^{-2}	4.688×10^{-2}
	0.650	3.473×10^{-5}	5.021×10^{-3}	4.174×10^{-2}	4.133×10^{-2}
	0.700	1.144×10^{-4}	4.343×10^{-3}	3.355×10^{-2}	3.400×10^{-2}
	0.750	1.384×10^{-4}	3.428×10^{-3}	2.420×10^{-2}	2.540×10^{-2}
	0.800	9.506×10^{-5}	2.574×10^{-3}	1.450×10^{-2}	1.626×10^{-2}
	0.850	8.249×10^{-6}	2.029×10^{-3}	5.561×10^{-3}	7.595×10^{-3}
	0.900	6.313×10^{-5}	1.824×10^{-3}	1.195×10^{-3}	7.113×10^{-3}
	0.950	5.712×10^{-5}	1.537×10^{-3}	3.880×10^{-3}	2.612×10^{-3}
	1.000	2.220×10^{-16}	2.220×10^{-16}	2.220×10^{-16}	2.220×10^{-16}
Test Kümesi	0.000	0.000	0.000	0.000	0.000
	0.005	1.794×10^{-6}	3.302×10^{-4}	7.496×10^{-4}	6.372×10^{-4}
	0.105	5.499×10^{-5}	5.112×10^{-3}	6.007×10^{-4}	7.700×10^{-4}
	0.155	1.033×10^{-5}	5.958×10^{-3}	9.640×10^{-3}	7.729×10^{-3}
	0.205	9.004×10^{-5}	5.840×10^{-3}	1.896×10^{-2}	1.639×10^{-2}
	0.255	1.263×10^{-4}	4.922×10^{-3}	2.859×10^{-2}	2.549×10^{-2}
	0.305	9.696×10^{-5}	3.404×10^{-3}	3.750×10^{-2}	3.405×10^{-2}
	0.355	1.547×10^{-5}	1.516×10^{-3}	4.494×10^{-2}	4.135×10^{-2}
	0.405	8.269×10^{-5}	4.931×10^{-4}	5.038×10^{-2}	4.688×10^{-2}
	0.455	1.577×10^{-4}	2.372×10^{-3}	5.347×10^{-2}	5.028×10^{-2}
	0.505	1.805×10^{-4}	3.891×10^{-3}	5.403×10^{-2}	5.136×10^{-2}
	0.555	1.421×10^{-4}	4.872×10^{-3}	5.204×10^{-2}	5.005×10^{-2}
	0.605	5.641×10^{-5}	5.225×10^{-3}	4.760×10^{-2}	4.642×10^{-2}
	0.655	4.433×10^{-5}	4.972×10^{-3}	4.099×10^{-2}	4.067×10^{-2}
	0.705	1.197×10^{-4}	4.258×10^{-3}	3.266×10^{-2}	3.318×10^{-2}
	0.755	1.370×10^{-4}	3.335×10^{-3}	2.323×10^{-2}	2.449×10^{-2}
	0.805	8.759×10^{-5}	2.503×10^{-3}	1.355×10^{-2}	1.536×10^{-2}
	0.855	7.612×10^{-7}	1.996×10^{-3}	4.759×10^{-3}	6.801×10^{-3}
	0.905	6.671×10^{-5}	1.811×10^{-3}	1.683×10^{-3}	1.849×10^{-3}
	0.955	5.169×10^{-5}	1.467×10^{-3}	3.835×10^{-3}	2.663×10^{-3}
	1.000	2.220×10^{-16}	2.220×10^{-16}	2.220×10^{-16}	2.220×10^{-16}

Çizelge 3.19. Örnek 3.4 için Hibrit Metotlarda Mutlak Hata Değerleri.

		Hibrit Metotlar		
	x	PSOABC	PSOACO	PSOGSA
Eğitim Kümesi	0.010	1.156×10^{-3}	2.486×10^{-2}	1.187×10^{-3}
	0.050	2.500×10^{-3}	1.160×10^{-1}	2.864×10^{-3}
	0.100	1.368×10^{-3}	2.053×10^{-1}	2.279×10^{-4}
	0.150	9.100×10^{-3}	2.234×10^{-1}	6.930×10^{-3}
	0.200	1.880×10^{-2}	2.735×10^{-1}	1.549×10^{-2}
	0.250	2.903×10^{-2}	3.141×10^{-1}	2.459×10^{-2}
	0.300	3.871×10^{-2}	3.463×10^{-1}	3.323×10^{-2}
	0.350	4.700×10^{-2}	3.708×10^{-1}	4.069×10^{-2}
	0.400	5.333×10^{-2}	3.882×10^{-1}	4.641×10^{-2}
	0.450	5.728×10^{-2}	3.989×10^{-1}	5.004×10^{-2}
	0.500	5.862×10^{-2}	4.030×10^{-1}	5.135×10^{-2}
	0.550	5.727×10^{-2}	4.005×10^{-1}	5.028×10^{-2}
	0.600	5.332×10^{-2}	3.914×10^{-1}	4.687×10^{-2}
	0.650	4.699×10^{-2}	3.753×10^{-1}	4.133×10^{-2}
	0.700	3.869×10^{-2}	3.518×10^{-1}	3.399×10^{-2}
	0.750	2.902×10^{-2}	3.203×10^{-1}	2.539×10^{-2}
	0.800	1.878×10^{-2}	2.798×10^{-1}	1.626×10^{-2}
	0.850	9.083×10^{-3}	2.293×10^{-1}	7.591×10^{-3}
	0.900	1.354×10^{-3}	1.672×10^{-1}	7.086×10^{-4}
	0.950	2.508×10^{-3}	9.164×10^{-2}	2.613×10^{-3}
	1.000	2.220×10^{-16}	2.220×10^{-16}	2.220×10^{-16}
Test Kümesi	0.000	0.000	0.000	0.000
	0.005	6.245×10^{-4}	1.254×10^{-2}	6.373×10^{-4}
	0.105	2.003×10^{-3}	2.138×10^{-1}	7.686×10^{-4}
	0.155	1.000×10^{-2}	2.289×10^{-1}	7.727×10^{-3}
	0.205	1.982×10^{-2}	2.780×10^{-1}	1.639×10^{-2}
	0.255	3.004×10^{-2}	3.177×10^{-1}	2.549×10^{-2}
	0.305	3.961×10^{-2}	3.491×10^{-1}	3.404×10^{-2}
	0.355	4.773×10^{-2}	3.729×10^{-1}	4.134×10^{-2}
	0.405	5.384×10^{-2}	3.896×10^{-1}	4.687×10^{-2}
	0.455	5.753×10^{-2}	3.996×10^{-1}	5.028×10^{-2}
	0.505	5.861×10^{-2}	4.030×10^{-1}	5.136×10^{-2}
	0.555	5.699×10^{-2}	3.999×10^{-1}	5.004×10^{-2}
	0.605	5.279×10^{-2}	3.901×10^{-1}	4.641×10^{-2}
	0.655	4.624×10^{-2}	3.733×10^{-1}	4.067×10^{-2}
	0.705	3.777×10^{-2}	3.490×10^{-1}	3.318×10^{-2}
	0.755	2.800×10^{-2}	3.167×10^{-1}	2.449×10^{-2}
	0.805	1.777×10^{-2}	2.752×10^{-1}	1.535×10^{-2}
	0.855	8.194×10^{-3}	2.236×10^{-1}	6.797×10^{-3}
	0.905	7.585×10^{-4}	1.603×10^{-1}	1.823×10^{-4}
	0.955	2.592×10^{-3}	8.325×10^{-2}	2.665×10^{-3}
	1.000	2.220×10^{-16}	2.220×10^{-16}	2.220×10^{-16}

Çizelge 3.20. Örnek 3.4 için Ortalama Karesel Hata Değerleri.

Kategori	**Metot**	**Eğitim Kümesi için Ortalama MSE**	**Test Kümesi için Ortalama MSE**
İteratif Metotlar	SM	$\mathbf{1.853 \times 10^{-21}}$	$\mathbf{3.124 \times 10^{-21}}$
	FDM	4.891×10^{-4}	1.229×10^{-4}
	Lobatto IIIa	5.627×10^{-14}	5.908×10^{-14}
Sezgisel Metotlar	mABC	$\mathbf{7.736 \times 10^{-9} \pm 4.153 \times 10^{-9}}$	$\mathbf{7.660 \times 10^{-9} \pm 4.112 \times 10^{-9}}$
	ACO	$2.717 \times 10^{-4} \pm 2.570 \times 10^{-4}$	$2.691 \times 10^{-4} \pm 2.545 \times 10^{-4}$
	GSA	$1.139 \times 10^{-3} \pm 3.025 \times 10^{-5}$	$1.128 \times 10^{-3} \pm 2.995 \times 10^{-5}$
	PSO	$9.722 \times 10^{-4} \pm 3.458 \times 10^{-4}$	$9.627 \times 10^{-4} \pm 3.425 \times 10^{-5}$
Hibrit Metotlar	PSO + ABC	$1.268 \times 10^{-3} \pm 1.321 \times 10^{-5}$	$1.256 \times 10^{-3} \pm 1.308 \times 10^{-5}$
	PSO + ACO	$1.516 \times 10^{-1} \pm 3.034 \times 10^{-2}$	$1.501 \times 10^{-1} \pm 3.005 \times 10^{-2}$
	PSO + GSA	$8.893 \times 10^{-4} \pm 1.245 \times 10^{-4}$	$8.806 \times 10^{-4} \pm 1.233 \times 10^{-4}$

Çizelge 3.21. Örnek 3.4 için saniye cinsinden geçen süre.

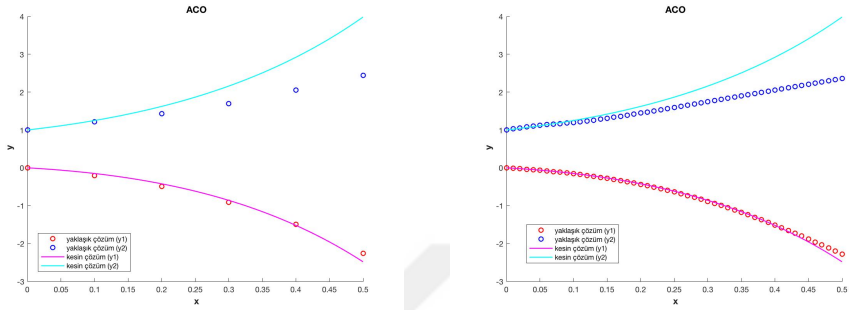
Metot	**Geçen Süre**
SM	21.522264
FDM	69.003596
Lobatto IIIa	3.557704
mABC	$3.543 \times 10^{-4} \pm 1.628 \times 10^{-4}$
ACO	$4.676 \times 10^{-4} \pm 3.405 \times 10^{-5}$
GSA	$4.379 \times 10^{-4} \pm 3.687 \times 10^{-4}$
PSO	$4.386 \times 10^{-4} \pm 3.552 \times 10^{-4}$
PSO + ABC	$1.601 \times 10^{-3} \pm 5.094 \times 10^{-4}$
PSO + ACO	$1.666 \times 10^{-3} \pm 4.727 \times 10^{-4}$
PSO + GSA	$3.305 \times 10^{-4} \pm 1.085 \times 10^{-4}$

Örnek 3.5. Birinci mertebeden lineer diferansiyel denklem sistemi 3.5 denklemi ile verilmiştir.

$$\begin{cases} y_1'(t) = 2y_1 - y_2 - 5t, \quad t \in [0, 1/2], \\ y_2'(t) = 3y_1 + 6y_2 - 4 \\ y_1(0) = 0 \\ y_2(0) = 1 \end{cases} \tag{3.5}$$

Denklem sisteminin analitik çözümleri sırasıyla $y_1(t) = 2t - \exp(3t) + 1$ ve $y_2(t) = \exp(3t) - t$ 'dir. Denklemde $y_1(t)$ için, iteratif, sezgisel ve hibrit metotlar kullanılarak ulaşılan nümerik çözümler sonucu elde edilen mutlak hata değerleri sırasıyla Çizelge 3.22, Çizelge 3.24 ve Çizelge 3.26 kullanılarak sunulmuştur. Ayrıca $y_2(t)$ için, iteratif, sezgisel ve hibrit metotlar kullanılarak ulaşılan nümerik çözümler sonucu elde edilen mutlak hata değerleri sırasıyla Çizelge 3.23, Çizelge 3.25 ve Çizelge 3.27 ile verilmiştir.

En iyi sonucu veren ACO metodu için 1000 iterasyon sonunda elde edilen nümerik çözümler ve kesin çözümler $h = 0.1$ ve $h = 0.01$ adım uzunlukları için sırasıyla Şekil 3.5 *a* ve Şekil 3.5 *b*'de gösterilmiştir. Metotların ortalama karesel hata değerleri ise $y_1(t)$ ve $y_2(t)$ için sırasıyla Çizelge 3.28 ve Çizelge 3.29 ile ve test kümesi için işlem süreleri Çizelge 3.30 ile verilmiştir.



(*a*) $h = 0.1$ (*b*) $h = 0.01$

Şekil 3.5. Örnek 3.5 için ACO algoritması kullanılarak 1000 iterasyonda diferansiyel denklem sisteminin yaklaşık çözüm grafiği.

Çizelge 3.22. Örnek 3.5 için İteratif Metotlarda y_1 için Mutlak Hata Değerleri.

		İteratif Metotlar		
	x	SM	FDM	Lobatto IIIa
Eğitim Kümesi	0.010	1.021	3.051×10^{-2}	1.021
	0.050	1.118	2.804×10^{-2}	1.118
	0.100	1.265	2.548×10^{-2}	1.265
	0.150	1.439	2.340×10^{-2}	1.439
	0.200	1.639	2.172×10^{-2}	1.639
	0.250	1.863	2.037×10^{-2}	1.863
	0.300	2.105	1.931×10^{-2}	2.105
	0.350	2.356	1.852×10^{-2}	2.356
	0.400	2.601	1.797×10^{-2}	2.601
	0.450	2.819	1.764×10^{-2}	2.819
	0.500	2.982	1.754×10^{-2}	2.982
Test Kümesi	0.000	0.000	1.565×10^{-2}	1.000
	0.005	1.011	1.548×10^{-2}	1.011
	0.105	1.281	1.266×10^{-2}	1.281
	0.155	1.457	1.164×10^{-2}	1.457
	0.205	1.660	1.081×10^{-2}	1.660
	0.255	1.887	1.015×10^{-2}	1.887
	0.305	2.130	9.631×10^{-3}	2.130
	0.355	2.381	9.245×10^{-3}	2.381
	0.405	2.624	8.979×10^{-3}	2.624
	0.455	2.839	8.827×10^{-3}	2.839
	0.495	2.969	8.784×10^{-3}	2.969

Çizelge 3.23. Örnek 3.5 için İteratif Metotlarda y_2 için Mutlak Hata Değerleri.

		İteratif Metotlar		
	x	SM	FDM	Lobatto IIIa
Eğitim Kümesi	0.010	7.090×10^{-2}	3.051×10^{-2}	7.090×10^{-2}
	0.050	4.264×10^{-2}	2.804×10^{-2}	4.264×10^{-2}
	0.100	2.262×10^{-1}	2.548×10^{-2}	2.262×10^{-1}
	0.150	4.757×10^{-1}	2.340×10^{-2}	4.757×10^{-1}
	0.200	8.189×10^{-1}	2.172×10^{-2}	8.189×10^{-1}
	0.250	1.293	2.037×10^{-2}	1.293
	0.300	1.948	1.931×10^{-2}	1.948
	0.350	2.848	1.852×10^{-2}	2.848
	0.400	4.081	1.797×10^{-2}	4.081
	0.450	5.760	1.764×10^{-2}	5.760
	0.500	8.034	1.754×10^{-2}	8.034
Test Kümesi	0.000	0.000	1.565×10^{-2}	0.000
	0.005	8.348×10^{-2}	1.548×10^{-2}	8.348×10^{-2}
	0.105	2.478×10^{-1}	1.266×10^{-2}	2.478×10^{-1}
	0.155	5.053×10^{-1}	1.164×10^{-2}	5.053×10^{-1}
	0.205	8.597×10^{-1}	1.081×10^{-2}	8.597×10^{-7}
	0.255	1.349	1.015×10^{-2}	1.349
	0.305	2.025	9.631×10^{-3}	2.025
	0.355	2.955	9.245×10^{-3}	2.955
	0.405	4.227	8.979×10^{-3}	4.227
	0.455	5.958	8.827×10^{-3}	5.958
	0.495	7.775	8.784×10^{-3}	7.775

Çizelge 3.24. Örnek 3.5 için Sezgisel Metotlarda y_1 için Mutlak Hata Değerleri.

		Sezgisel Metotlar			
	x	mABC	ACO	GSA	PSO
Eğitim Kümesi	0.010	0.000	2.478×10^{-3}	1.349×10^{-2}	1.155×10^{-2}
	0.050	1.710×10^{-2}	1.212×10^{-2}	5.938×10^{-2}	6.975×10^{-2}
	0.100	5.807×10^{-5}	2.417×10^{-2}	9.622×10^{-2}	1.718×10^{-1}
	0.150	9.790×10^{-2}	3.696×10^{-2}	1.063×10^{-1}	3.105×10^{-1}
	0.200	1.687×10^{-1}	5.083×10^{-2}	8.457×10^{-2}	4.908×10^{-1}
	0.250	2.658×10^{-1}	6.524×10^{-2}	2.537×10^{-2}	7.184×10^{-1}
	0.300	3.950×10^{-1}	7.830×10^{-2}	7.802×10^{-2}	1.000
	0.350	5.634×10^{-1}	8.623×10^{-2}	2.334×10^{-1}	1.343
	0.400	7.794×10^{-1}	8.282×10^{-2}	4.497×10^{-1}	1.757
	0.450	1.053	5.882×10^{-2}	7.375×10^{-1}	2.252
	0.500	1.397	1.528×10^{-3}	1.109	2.840
Test Kümesi	0.000	0.000	0.000	0.000	0.000
	0.005	1.137×10^{-3}	1.244×10^{-3}	6.843×10^{-3}	5.630×10^{-3}
	0.105	5.274×10^{-2}	2.540×10^{-2}	9.850×10^{-1}	1.840×10^{-1}
	0.155	1.039×10^{-1}	3.830×10^{-2}	1.056×10^{-1}	3.266×10^{-1}
	0.205	1.771×10^{-1}	5.227×10^{-2}	8.044×10^{-2}	5.113×10^{-1}
	0.255	2.772×10^{-1}	6.665×10^{-2}	1.713×10^{-2}	7.440×10^{-1}
	0.305	4.100×10^{-1}	7.940×10^{-2}	9.108×10^{-2}	1.031
	0.355	5.827×10^{-1}	8.653×10^{-2}	2.521×10^{-1}	1.381
	0.405	8.040×10^{-1}	8.153×10^{-2}	4.751×10^{-1}	1.803
	0.455	1.084	5.484×10^{-2}	7.706×10^{-1}	2.307
	0.495	1.359	9.197×10^{-3}	1.068	2.777

Çizelge 3.25. Örnek 3.5 için Sezgisel Metotlarda y_2 için Mutlak Hata Değerleri.

		Sezgisel Metotlar			
	x	mABC	ACO	GSA	PSO
Eğitim Kümesi	0.010	1.236×10^{-2}	2.468×10^{-3}	1.523×10^{-2}	1.155×10^{-2}
	0.050	6.882×10^{-2}	1.516×10^{-2}	8.556×10^{-2}	6.975×10^{-2}
	0.100	1.573×10^{-1}	3.644×10^{-2}	1.969×10^{-1}	1.718×10^{-1}
	0.150	2.695×10^{-1}	6.533×10^{-2}	3.383×10^{-1}	3.105×10^{-1}
	0.200	4.099×10^{-1}	1.049×10^{-1}	5.147×10^{-1}	4.908×10^{-1}
	0.250	5.841×10^{-1}	1.590×10^{-1}	7.318×10^{-1}	7.184×10^{-1}
	0.300	7.988×10^{-1}	2.330×10^{-1}	9.962×10^{-1}	1.000
	0.350	1.062	3.336×10^{-1}	1.316	1.343
	0.400	1.382	4.693×10^{-1}	1.699	1.757
	0.450	1.770	6.506×10^{-1}	2.157	2.252
	0.500	2.238	8.904×10^{-1}	2.701	2.840
Test Kümesi	0.000	0.000	0.000	0.000	0.000
	0.005	6.095×10^{-3}	1.190×10^{-3}	7.503×10^{-3}	5.630×10^{-3}
	0.105	1.674×10^{-3}	3.894×10^{-2}	2.096×10^{-1}	1.840×10^{-1}
	0.155	2.821×10^{-1}	6.875×10^{-2}	3.543×10^{-1}	3.266×10^{-1}
	0.205	4.257×10^{-1}	1.095×10^{-1}	5.345×10^{-1}	5.113×10^{-1}
	0.255	6.037×10^{-1}	1.654×10^{-1}	7.560×10^{-1}	7.440×10^{-1}
	0.305	8.228×10^{-1}	2.418×10^{-1}	1.025	1.031
	0.355	1.091	3.454×10^{-1}	1.351	1.381
	0.405	1.417	4.852×10^{-1}	1.741	1.803
	0.455	1.813	6.717×10^{-1}	2.207	2.307
	0.495	2.187	8.634×10^{-1}	2.643	2.777

Çizelge 3.26. Örnek 3.5 için Hibrit Metotlarda y_1 için Mutlak Hata Değerleri.

		Hibrit Metotlar		
	x	PSOABC	PSOACO	PSOGSA
Eğitim Kümesi	0.010	3.016×10^{-5}	1.243×10^{-2}	1.264×10^{-2}
	0.050	9.099×10^{-3}	5.506×10^{-2}	5.611×10^{-2}
	0.100	4.362×10^{-2}	9.014×10^{-2}	9.227×10^{-2}
	0.150	1.078×10^{-1}	1.010×10^{-1}	1.043×10^{-1}
	0.200	2.066×10^{-1}	8.271×10^{-2}	8.713×10^{-2}
	0.250	3.458×10^{-1}	2.956×10^{-2}	3.518×10^{-2}
	0.300	5.321×10^{-1}	6.513×10^{-2}	5.830×10^{-2}
	0.350	7.731×10^{-1}	2.091×10^{-1}	2.011×10^{-1}
	0.400	1.078	4.115×10^{-1}	4.022×10^{-1}
	0.450	1.457	6.827×10^{-1}	6.721×10^{-1}
	0.500	1.922	1.035	1.023
Test Kümesi	0.000	0.000	0.000	0.000
	0.005	1.254×10^{-4}	6.299×10^{-3}	6.402×10^{-3}
	0.105	4.863×10^{-3}	9.239×10^{-2}	9.463×10^{-2}
	0.155	1.161×10^{-1}	1.006×10^{-1}	1.039×10^{-1}
	0.205	2.186×10^{-1}	7.907×10^{-2}	8.361×10^{-2}
	0.255	3.622×10^{-1}	2.208×10^{-2}	2.781×10^{-2}
	0.305	5.536×10^{-1}	7.717×10^{-2}	7.022×10^{-2}
	0.355	8.005×10^{-1}	2.266×10^{-1}	2.184×10^{-1}
	0.405	1.112	4.353×10^{-1}	4.259×10^{-1}
	0.455	1.499	7.140×10^{-1}	7.033×10^{-1}
	0.495	1.871	9.956×10^{-1}	9.839×10^{-1}

Çizelge 3.27. Örnek 3.5 için Hibrit Metotlarda y_2 için Mutlak Hata Değerleri.

		Hibrit Metotlar		
	x	PSOABC	PSOACO	PSOGSA
Eğitim Kümesi	0.010	6.774×10^{-3}	1.176×10^{-2}	1.150×10^{-2}
	0.050	3.166×10^{-2}	7.067×10^{-2}	6.951×10^{-2}
	0.100	5.788×10^{-2}	1.734×10^{-1}	1.714×10^{-1}
	0.150	7.873×10^{-2}	3.124×10^{-1}	3.099×10^{-1}
	0.200	9.401×10^{-2}	4.926×10^{-1}	4.899×10^{-1}
	0.250	1.030×10^{-1}	7.200×10^{-1}	7.173×10^{-1}
	0.300	1.037×10^{-1}	1.001	9.988×10^{-1}
	0.350	9.276×10^{-2}	1.343	1.342
	0.400	6.412×10^{-2}	1.756	1.756
	0.450	8.767×10^{-3}	2.250	2.250
	0.500	8.603×10^{-2}	2.836	2.838
Test Kümesi	0.000	0.000	0.000	0.000
	0.005	3.415×10^{-3}	1.254×10^{-2}	5.606×10^{-3}
	0.105	6.021×10^{-2}	5.736×10^{-3}	1.835×10^{-1}
	0.155	8.051×10^{-2}	3.284×10^{-1}	3.259×10^{-1}
	0.205	9.522×10^{-2}	5.131×10^{-1}	5.104×10^{-1}
	0.255	1.035×10^{-1}	7.455×10^{-1}	7.429×10^{-1}
	0.305	1.032×10^{-1}	1.032	1.030
	0.355	9.081×10^{-2}	1.381	1.380
	0.405	5.996×10^{-2}	1.802	1.801
	0.455	1.308×10^{-3}	2.304	2.305
	0.495	7.433×10^{-2}	2.773	2.775

Çizelge 3.28. Örnek 3.5'te y_1 için Ortalama Karesel Hata Değerleri.

Kategori	**Metot**	**Eğitim Kümesi için Ortalama MSE**	**Test Kümesi için Ortalama MSE**
İteratif Metotlar	SM	4.084	4.076
	FDM	6.128×10^{-1}	5.951×10^{-1}
	Lobatto IIIa	4.084	4.076
Sezgisel Metotlar	mABC	$3.514 \times 10^{-1} \pm 1.349 \times 10^{-2}$	$3.305 \times 10^{-1} \pm 1.266 \times 10^{-2}$
	ACO	$\mathbf{1.740 \times 10^{-2} \pm 2.043 \times 10^{-2}}$	$\mathbf{1.707 \times 10^{-2} \pm 2.043 \times 10^{-2}}$
	GSA	$1.783 \times 10^{-1} \pm 2.889 \times 10^{-2}$	$1.638 \times 10^{-1} \pm 2.715 \times 10^{-2}$
	PSO	$1.297 \times 10^{-1} \pm 1.557 \times 10^{-2}$	$1.186 \times 10^{-1} \pm 1.457 \times 10^{-2}$
Hibrit Metotlar	PSO + ABC	$5.786 \times 10^{-2} \pm 5.686 \times 10^{-2}$	$5.384 \times 10^{-2} \pm 5.344 \times 10^{-2}$
	PSO + ACO	$1.449 \times 10^{-1} \pm 3.385 \times 10^{-2}$	$1.328 \times 10^{-1} \pm 3.160 \times 10^{-2}$
	PSO + GSA	$1.503 \times 10^{-1} \pm 2.702 \times 10^{-2}$	$1.379 \times 10^{-1} \pm 2.529 \times 10^{-2}$

Çizelge 3.29. Örnek 3.5'te y_2 için Ortalama Karesel Hata Değerleri.

Kategori	**Metot**	**Eğitim Kümesi için Ortalama MSE**	**Test Kümesi için Ortalama MSE**
İteratif Metotlar	SM	9.777	9.544
	FDM	$1.785 \times 10^{+2}$	$2.000 \times 10^{+2}$
	Lobatto IIIa	9.777	9.544
Sezgisel Metotlar	mABC	$1.021 \pm 1.246 \times 10^{-2}$	$9.698 \times 10^{-1} \pm 1.186 \times 10^{-2}$
	ACO	$\mathbf{5.389 \times 10^{-1} \pm 1.909 \times 10^{-1}}$	$\mathbf{5.064 \times 10^{-1} \pm 1.802 \times 10^{-1}}$
	GSA	$1.648 \pm 1.207 \times 10^{-1}$	$1.570 \pm 1.161 \times 10^{-1}$
	PSO	$1.598 \pm 5.211 \times 10^{-2}$	$1.518 \pm 4.991 \times 10^{-2}$
Hibrit Metotlar	PSO + ABC	$1.029 \pm 3.048 \times 10^{-1}$	$9.750 \times 10^{-1} \pm 2.919 \times 10^{-1}$
	PSO + ACO	$1.561 \pm 1.559 \times 10^{-1}$	$1.484 \pm 1.491 \times 10^{-1}$
	PSO + GSA	$1.679 \pm 8.855 \times 10^{-2}$	$1.596 \pm 8.505 \times 10^{-2}$

Çizelge 3.30. Örnek 3.5 için saniye cinsinden geçen süre.

Metot	**Geçen Süre**
SM	10.028984
FDM	0.033099
Lobatto IIIa	0.147758
mABC	$1.989 \times 10^{-4} \pm 1.524 \times 10^{-4}$
ACO	$1.884 \times 10^{-4} \pm 1.293 \times 10^{-4}$
GSA	$2.355 \times 10^{-4} \pm 2.331 \times 10^{-4}$
PSO	$1.895 \times 10^{-4} \pm 1.750 \times 10^{-4}$
PSO + ABC	$2.323 \times 10^{-4} \pm 2.049 \times 10^{-4}$
PSO + ACO	$1.988 \times 10^{-4} \pm 1.361 \times 10^{-4}$
PSO + GSA	$1.620 \times 10^{-4} \pm 1.308 \times 10^{-4}$

Tezin takip eden bölümünde, deneysel çalışmalar kapsamında elde edilen bulgular üzerinde yorumlara yer verilecektir. Bu bağlamda, çalışmanın hangi kısıtlar altında gerçekleştirildiği ve karşılaşılan problemlerin üstesinden gelebilmek için neler yapılabileceği hususunda ipuçları verilecektir.

4. TARTIŞMA VE SONUÇ

Bu tezde, ileri beslemeli yapay sinir ağları kullanılarak başlangıç veya Dirichlet sınır koşullarına sahip birinci ve ikinci mertebeden adi diferansiyel denklemlerin nümerik çözümleri elde edilmiştir. Çalışmanın özgünlüğü, diferansiyel denklem çözümlerinin teorik yöntemler kullanılmadan bilgisayarlara sezgisel olarak öğretilmesinden kaynaklanmaktadır. Ek olarak tezde tüm sezgisel optimizasyon yöntemlerine kolaylıkla uyarlanabilecek yeni bir mutasyon yaklaşımı sunulmuştur.

Deneysel çalışmalar kısmında seçilen problemler lineer veya lineer olmayan diferansiyel denklem türlerini kapsayacak şekilde örneklenmiştir. Çözümü gerçekleştirilen tüm örneklerde Çizelge 3.1 ile verilen parametre değerleri kullanılmıştır. Böylelikle tezde kullanılan tüm yöntemlerin birbiriyle daha sağlıklı biçimde karşılaştırılmasına olanak sağlanmıştır. Örnekler sezgisel optimizasyon yöntemleri ile eğitilen yapay sinir ağları haricinde, klasik metotlardan atış yöntemi, sonlu farklar ve Lobatto IIIa yöntemleri ile de çözülmüş ve elde edilen sonuçlar birbiriyle karşılaştırılmıştır.

Klasik yöntemlerin en büyük dezavantajı, belli bir aralıkta aranan çözümün sadece belirtilen aralığın parçalanışından elde edilen düğüm noktalarında bulunmasıdır. Düğüm noktaları haricindeki noktalarda diferansiyel denklemin nümerik çözümünü elde edebilmek için ise genellikle interpolasyon teknikleri kullanılır. Ancak bu yaklaşım kümülatif hatanın artmasına neden olur. Aksine, yapay sinir ağlarında aralığın parçalanışından elde edilen düğüm noktaları sadece ağın eğitimi için kullanılmaktadır. Ağ eğitimini tamamladıktan sonra aralık üzerindeki her noktada çözüm üretir. Bununla birlikte yapay sinir ağları eğitimini tamamladıktan sonra çok hızlı biçimde çıktı üretir. Ancak ağın eğitim süresi zaman alır.

Tezde yapay sinir ağlarının eğitiminde Parçacık Sürü Optimizasyonu, Yapay Arı Kolonisi, Yapay Karınca Kolonisi ve Kütle Çekim Arama Algoritmaları kullanılmıştır. Bahsi geçen optimizasyon algoritmalarının birbirlerine göre sağladıkları üstünlükleri kullanarak, elde edilen sonuçların iyileştirilmesi için hibrit yöntemler de yine bu tezde denenmiştir. Bununla birlikte incelenen optimizasyon algoritmaları için bilinen en iyi çözümün civarında dinamik olarak oluşturulan hiper-küreleri kullanan yeni bir mutasyon operatörü tanımlanmıştır. Önerilen mutasyon operatörü Yapay Arı Kolonisi algoritması üzerinde test edilmiş ve elde edilen sonuçları iyileştirdiği görülmüştür. Bu nedenle, tezde verilen örneklerin çözümlerinde klasik ABC algoritması yerine tezde önerilen mutasyon operatörü ile zenginleştirilen mABC algoritması ile elde edilen sonuçlara yer verilmiştir.

Deneysel çalışmalarda her örnek, başlangıçta rassal olarak üretilen bir popülasyon kullanılarak çözülmüştür. Performans analizinin daha sağlıklı yapılabilmesi adına örnekler onar kez çalıştırılmış ve elde edilen ortalama karesel hataların ortalaması ve standart sapmaları elde edilmiştir.

Örnek 3.1 ile verilen ilk deneysel çalışmada birinci mertebeden lineer homojen bir diferansiyel denklem çözümü elde edilmiştir. Örnek için genel olarak hibrit yaklaşımların daha iyi sonuç verdiği görülmüştür. Elde edilen nümerik çözümler incelendiğinde, sezgisel optimizasyon metotlarının tamamının ortalama karesel hata miktarları bağlamında iteratif yöntemlere üstünlük sağladığı görülmüştür. Popülasyon tabanlı sezgisel optimizasyon yöntemleri kendi aralarında karşılaştırıldığında en iyi sonucun hibrit metotlardan PSOGSA algoritması tarafından sağlandığı saptanmıştır. Benzer olarak sezgisel metotların da iteratif yöntemlere göre daha iyi sonuç ürettiği görülmüştür.

Belirtilen Örnek 3.1 birinci mertebeden olduğu için atış yöntemi diferansyel denklemin bağımsız değişkene göre ikinci kez türevi alınarak 2. mertebeden bir diferansiyel denkleme dönüştürülür. Ancak bu durumda başlangıç noktasında

bilinmeyen fonksiyonun türev değerinin ($y'(a)$) tahminlenmesine ihtiyaç duyulur. Çalışmada diğer metotlarla karşılaştırma yapabilme adına diferansiyel denklemin gerçek çözümünün türevinin bu noktadaki değeri kullanılmıştır. Gerçekte atış metotu birinci mertebeden başlangıç değer problemleri için $y'(a)$ değeri bilinmediğinden kullanılamaz. Benzer şekilde Lobatto IIIa metodu için de problem 2. mertebeden başlangıç değer problemine dönüştürülerek çözüm araştırılmıştır. Ayrıca birinci mertebeden başlangıç değer problemleri için sonlu farklar yöntemi ise Euler yöntemine karşılık gelir.

Örnek 3.2 ile verilen deneysel çalışmada ikinci mertebeden lineer olmayan bir diferansiyel denklemin çözümü elde edilmiştir. Örnekte elde edilen sonuçların istenilen düzeyde olmadığı söylenebilir. En iyi sonucun mABC algoritması ile elde edildiği görülmüştür. Bu algoritma ile verilen nümerik çözümlerin karşılaştırıldığı Şekil 3.2a'da görüldüğü üzere, $h = 0.1$ adım uzunluğu için ağın öğrenmediği gözlemlenirken, $h = 0.01$ için ağın çözümü öğrenebildiği görülmüştür. $h = 0.1$ için ağın çözümü öğrenememe nedeni, ağın eğitimi için yeterli sayıda örneğin ağa girdi olarak verilmemesi olarak yorumlanmıştır. Tüm metotlar arasında en düşük ortalama karesel hata mABC yaklaşımıyla elde edilmiştir.

Örnek 3.3 ve Örnek 3.4 ile verilen sırasıyla ikinci mertebeden lineer ve lineer olmayan Dirichlet sınır değer problemleri için, elde edilen nümerik çözümlerde en iyi sonucu iteratif yöntemlerden atış metodu vermiştir. Sezgisel algoritmalardan mABC algoritmasının ise diğer sezgisel algoritmalara göre daha iyi sonuç verdiği gözlemlenmiştir. Klasik yöntemler iyi sonuç vermesine rağmen, aralığın parçalanışından elde edilen düğümler harici noktalarda çözüm üretemediklerinden sezgisel yöntemlerin kullanılması daha cazip olabilir.

Örnek 3.5 birinci mertebeden diferansiyel denklem sisteminin en iyi sonucu sezgisel metotlardan ACO metodu vermiştir. h sabit adım uzunluğu azaltıldıkça çözüme yakınsama artacaktır. Bu örnekte diferansiyel denklem sisteminin her

bir bilinmeyen fonksiyonu için ayrı bir maliyet fonksiyonu tanımlıdır. En iyileştirilmesi istenilen maliyet fonksiyonu ise bu fonksiyonların toplamı olarak ifade edilmiştir. Bahsi geçen ağırlıklandırılmış maliyet fonksiyonunun optimize edilme problemi yerine çoklu optimizasyon yaklaşımlarının kullanılabileceği düşünülmektedir. Bununla birlikte birinci mertebeden bir denklem sistemi rahatlıkla yüksek mertebeden bir başlangıç değer problemine dönüştürülebilir. Dolayısıyla problem gerekli dönüşümler yapıldıktan sonra Örnek 3.1 ve Örnek 3.2 ile benzer olarak da çözülebilir. Ancak bu çözümlerin tezde elde edilen çözümlerden çokça farklı olacağı düşünülmemektedir.

Genel olarak Dirichlet sınır değer problemlerinde başlangıç değer problemlerine göre daha başarılı sonuçlar elde edildiği görülmüştür. Bu durum, başlangıç değer problemlerinde aralığın en küçük üst sınırındaki çözümün bilinmemesinden kaynaklanır. Her iki sınır değeri bilinen bir problem için ağın bilinmeyen parametreleri daha dengeli biçimde ayarlanabilir, haliyle Dirichlet sınır değer problemlerinin çözümlerini yapay sinir ağına öğretmek daha kolaydır.

Çalışmada farklı her diferansiyel denklem için ayrı bir yapay sinir ağı oluşturma gerekliliği ortaya çıkmıştır. Bunun temel nedeninin, diferansiyel denklemlerin nümerik çözümlerinde yapay sinir ağının çözümünü de içeren ve diferansiyel denklemin başlangıç veya sınır koşullarını sağlayan farklı tipte deneme fonksiyonlarının kullanılması olduğu düşünülmektedir.

Mevcut çalışmanın Çok Katmanlı Algılayıcı (Multi Layer Perceptron, MLP), Kendini Tekrarlayan Ağ (Recurrent Neural Network, RNN), Uzun kısa süreli hafıza (Long Short-Term Memory, LSTM) ağları gibi farklı tipteki ağ modelleri ile tekrarlanabileceği ve problemin makine öğreniminin bir alt dalı olan derin öğrenmeye taşınabileceği düşünülmektedir. Böylelikle tezin temel kısıtlarından biri olan az sayıda nöron kullanımı probleminin aşılabileceği ve daha hızlı çözüme yakınsanabileceği tahmin edilmektedir.

KAYNAKLAR

[1] Lee, H. Kang, I.S. 1990. Neural algorithms for solving differential equations. **Journal of Computational Physics**, 91(1): 110-131.

[2] Meade, A., Fernandez, A. 1994. The numerical solution of linear ordinary differential equations by feedforward neural networks. **Mathematical and Computer Modelling**, 19(12): 1-25.

[3] Lagaris, I.E., Likas, A., Fotiadis, D.I. 1998. Artificial neural networks for solving ordinary and partial differential equations. **IEEE Transactions on Neural Networks**, 9(5): 987-1000.

[4] Li, Z., Wang, W., Yan, Y., Li, Z. 2015. Psabc: A hybrid algorithm based on particle swarm and artificial bee colony for high-dimensional optimization problems. **Expert Systems with Applications**, 42(22): 8881-8895.

[5] Fojdl, J., Brause, R.W. 2008. The performance of approximation ordinary differential equations by neural nets. **20th IEEE International Conference on Tools with Artificial Intelligence**, ISBN: 978-0-7695-3440-4.

[6] McFall, K.S., Mahan, J.R. 2009. Artificial neural network method for solution of boundary value problems with exact satisfaction of arbitrary boundary conditions. **IEEE Transactions On Neural Networks**, 20(8).

[7] Otadi, M., Mosleh, M. 2011. Numerical solution of quadratic Riccati differential equation by neural network. **Mathematical Sciences**, 5(3): 249-257.

[8] Kumar, M., Yadav, N. 2011. Multilayer perceptrons and radial basis function network methods for the solution of differential equations: A survey. **Computer and Mathematics with Applications**, 62: 3796-3811.

[9] Raja, M.A.Z., Ahmad, S.I., Samar, R. 2013. Neural network optimized with evolutionary computing technique for solving the 2-dimensional Bratu problem. **Neural Computing and Applications**, 23(7-8): 2199-2210.

[10] Raja, M.A.Z. 2014. Numerical treatment for boundary value problems of Pantograph functional differential equation using computational intelligence algorithms. **Applied Soft Computing**, 24: 806-821.

[11] Raja, M.A.Z., Ahmad, S.I., Samar, R. 2014. Solution of the 2-dimensional Bratu problem using neural network, swarm intelligence and sequential

quadratic programming. **Neural Computing and Applications**, 25(7-8): 1723-1739.

[12] Raja, M.A.Z. 2014. Stochastic numerical treatment for solving Troesch's problem. **Information Sciences**, 279: 860-873.

[13] Raja, M.A.Z. 2014. Unsupervised neural networks for solving Troesch's problem. **Chinese Physics B**, 23(1): 018903.

[14] Raja, M.A.Z., Samar, R., Alaidarous, E.S., Shivanian, E. 2016. Bio-inspired computing platform for reliable solution of Bratu-type equations arising in the modeling of electrically conducting solids. **Applied Mathematical Modelling**, 40: 5964-5977.

[15] Majeed, K., Masood, Z., Samar, R. Raja, M.A.Z. 2017. A genetic algorithm optimized Morlet wavelet artificial neural network to study the dynamics of nonlinear Troesch's system. **Applied Soft Computing**, 56: 420-435.

[16] Khan, N., Shaikh, A. 2017. A smart amalgamation of spectral neural algorithm for nonlinear Lane-Emden equations with simulated annealing. **Journal of Artificial Intelligence and Soft Computing Research**, 7(3): 215-224.

[17] Jafarian, A., Rostami, F., Golmankhaneh, A. K., Baleanu, D. 2017. Using ANNs approach for solving fractional order volterra integro-differential equations. **International Journal of Computational Intelligence Systems**, 10: 470-480.

[18] Lagaris, I.E., Likas, A., Fotiadis, D.I. 1997. Artificial neural network methods in quantum mechanics. **Computer Physics Communications**, 104: 1-14.

[19] Kourakos, G., Mantoglou, A. 2009. Pumping optimization of coastal aquifers based on evolutionary algorithms and surrogate modular neural network models. **Advances in Water Resources**, 32: 507-521.

[20] Caetano,C., Reis Jr. J. L., Amorim, J., Ruv Lemes M., Dal Pino Jr. A. 2010. Using neural networks to solve nonlinear differential equations in atomic and molecular physics. **International Journal of Quantum Chemistry**, 111: 2732-2740.

[21] Baymani, M., Effati, S., Kerayechian, A. 2010. A feed-forward neural network for solving stokes problem. **Acta Applicandae Mathematicae**, 116(1): 55-64.

[22] Mall, S., Chakraverty, S. 2014. Chebysev neural network based model for solving Lane-Emdan type equations. **Applied Mathematics and Computaion**, 247: 100-114.

[23] Anastassi, A.A., 2014. A feed-forward neural network for solving stokes problem, **Neural Computing and Applications**. 25(1): 229-236.

[24] Raja, M.A.Z., Samar, R.. 2014. Numerical treatment for nonlinear MHD Jeffery-Hamel problem using neural networks optimized with interior point algorithm. **Neurocomputing**, 124: 178-193.

[25] Raja, M.A.Z., Ahmad S.I. 2014. Numerical treatment for solving one-dimensional Bratu problem using neural networks, **Neural Computing and Applications**. 24: 549-561.

[26] Raja, M.A.Z., Manzar, M., Samar, R. 2015. An efficient computational intelligence approach for solving fractional order Riccati equations using ANN and SQP. **Applied Mathematical Modelling**, 39: 3075-3093.

[27] Mall, S., Chakraverty, S. 2015. Numerical solution of nonlinear singular initial value problems of Emden-Fowler type using Chebyshev neural network method. **Neurocomputing**, 149: 975-982.

[28] Mall S., Chakraverty S. 2016. Hermite functional link neural network for solving the Van der Pol-Duffing oscillator equation. **Neural Computing**, 28(8): 1574-98.

[29] Masood, Z., Majeed, K., Samar, R., Raja, M.A.Z. 2017. Design of mexican hat wavelet neural networks for solving Bratu type nonlinear systems. **Neurocomputing**, 221.

[30] Kashkari, B., Syam, M. 2017. Evolutionary computational intelligence in solving a class of nonlinear Volterra-Fredholm integro-differential equations. **Journal of Computational and Applied Mathematics**, 311: 314-323.

[31] Raja, M.A.Z., Iftikhar, A., Khan, I., Syam, M., Wazwaz, A.M. 2016. Neuro-heuristic computational intelligence for solving nonlinear pantograph systems. **Frontiers of Information Technology and Electronic Engineering**, 18(4): 464-484.

[32] Sinchev, B., Sibanbayeva, S.E., Mukhanova, A.M., Nurgulzhanova, A.N., Zaurbekov, N.S., Imanbayev, K.S., Gagarina, N.L., Baibolova, L.K. 2017. Some methods of training radial basis neural networks in solving the Navier-Stokes equations. **International Journal for Numerical Methods in Fluids**, 86:625-636.

[33] Mall, S., Chakraverty, S. 2016. Application of Legendre neural network for solving ordinary differential equations. **Applied Soft Computing**, 43: 347-356.

[34] Sabir, Z., Manzar, M., Raja, Raja, M.A.Z., Sheraz, M., Wazwaz, A.M. 2018. Neuro-heuristics for nonlinear singular Thomas-Fermi systems. **Applied Soft Computing**, 65: 152-169.

[35] Raja, M.A.Z., Junaid, A.K., Qureshi, I.M. 2010. A new stochastic approach for solution of Riccati differential equation of fractional order. **Annals of Mathematics and Artificial Intelligence**, 60: 229-250.

[36] Rostami, F, Jafarian, A. 2017. A new artificial neural network structure for solving high-order linear fractional differential equations. **International Journal of Computer Mathematics**, 95: 1-15.

[37] Zúñiga-Aguilar, C., Coronel-Escamilla, A., Gómez-Aguilar, J.F., Alvarado-Martinez, V.M., Romero Ugalde, H.M.2018. New numerical approximation for solving fractional delay differential equations of variable order using artificial neural networks. **European Physical Journal Plus**, 133: 75.

[38] Jafarian, A., Nia, S.M., Golmankhaneh, A.K., Baleanu, D. 2018. On artificial neural networks approach with new cost functions. **Applied Mathematics and Computation**, 339: 546-555.

[39] Pakdaman, M., Ahmadian, A., Effati, S., Salahshour, S., Baleanu, D. 2017. Solving differential equations of fractional order using an optimization technique based on training artificial neural network. **Applied Mathematics and Computation**, 293: 81-95.

[40] Alharbi, A. 2010. An artificial neural networks method for solving partial differential equations. **AIP Conference Proceedings**, 1281: 1425.

[41] Raja, M.A.Z., Khan, J.A., Shah, S.M., Samar, R., Behloul, D. 2015. Comparison of three unsupervised neural network models for first Painleve transcendent. **Neural Computing and Application**, 26: 1055-1071.

[42] Zjavka, L., Snasel, V. 2015. Composing and solving general differential equations using extended polynomial networks. **International Conference on Intelligent Networking and Collaborative Systems**.

[43] Ahmad, I., Ahmad, S.I., Bilal, M., Anwar, N. 2016 . Stochastic numerical treatment for solving Falkner-Skan equations using feedforward neural networks. **Neural Computing and Applications**, 28 (1): 1131-1144.

[44] Yekrangi, A., Ghalambaz, M., Noghrehabadi, A., Beni, Y.T., Abadyan, M., Abadi, M.N., Abadi, M.N., 2011. An approximate solution for a simple pendulum beyond the small angles regimes using hybrid artificial

neural network and particle swarm optimization algorithm. **Procedia Engineering**, 10: 3734-3740.

[45] Alharbi, A. 2012. A solution to neural field equations by a recurrent neural network method. **AIP Conference Proceedings**, 1479: 772-776.

[46] Chakraverty, S. 2014. Regression-based weight generation algorithm in neural network for solution of initial and boundary value problems. **Neural Computing and Applications**, 25(3), 585-594.

[47] Rizaner, F.B., Rizaner, A. 2018. Approximate solutions of initial value problems for ordinary differential equations using radial basis function networks. **Neural Process Letter**, 48:1063-1071.

[48] Tan, L.S., Zainuddin, Z., Ong, P. 2018. Solving ordinary differential equations using neural networks. **AIP Conference Proceedings**, 1974, 020070.

[49] Valasoulis, K., Fotiadis, D.I., Lagaris, I.E., Likas, A. 2002. Solving differential equations with neural networks: implementation on a DSP platform. 2: 1265-1268.

[50] Sun, M., Yan, X., Sclabassi, R.J. 2004. Solving partial differential equations in real-time using artificial neural network signal processing as an alternative to finite-element analysis. Proceedings of **2003 International Conference on Neural Networks and Signal Processing**, 1: 381-384.

[51] Beidokhti, R.S., Malek, A. 2009. Solving initial-boundary value problems for systems of partial differential equations using neural networks and optimization techniques. **Journal of the Franklin Institute**, 346: 898-913.

[52] McFall, K. 2013. Automated design parameter selection for neural networks solving coupled partial differential equations with discontinuities. **Journal of the Franklin Institute**, 350: 300-317.

[53] Rudd, K., Di Muro, G., Ferrari, S. 2013. A Constrained back propagation approach for the adaptive solution of partial differential equations. **IEEE transactions on neural networks and learning systems**, 25(3): 571-584.

[54] Rudd, K. 2015. A constrained integration (CINT) approach to solving partial differential equations using artificial neural networks. **Neurocomputing**, 155: 277-285.

[55] Zjavka, L., Pedrycz, W. 2016. Constructing general partial differential equations using polynomial and neural networks. **Neural Networks**, 73: 58-69.

[56] Zjavka, L., Snasel, V., Ojha, V., Pedrycz, W.2016. A substitution of the general partial differential equation with extended polynomial networks. **IEEE Joint Conference on Neural Network**.

[57] Mall, S., Chakraverty, S. 2017. Single layer Chebyshev neural network model for solving elliptic partial differential equations. **Neural Processing Letters**. 45: 825-840.

[58] Malek, A., Beidokhti, R.S. 2006. Numerical solution for high order differential equations using a hybrid neural network optimization method. **Applied Mathematics and Computation**, 183(1): 260-271.

[59] Vinod, A.V., Kumar, K.A., Reddy, G.V. 2009. Simulation of biodegradation process in a fluidized bed bioreactor using genetic algorithm Trained Feed Forward Neural Network. **Biochemical Engineering Journal**, 46: 12-20.

[60] Aquino, R.R.B., Carvalho Jr.,M.A., Neto,O.N., Lira, M.M.S., Almeida, G.J., Tiburcio, S.N.N. 2010. Recurrent neural networks solving a real large scale mid-term scheduling for power plants. **International Joint Conference on Neural Networks** (IJCNN), 18-23 July 2010, Barselona, İspanya.

[61] Raja, M.A.Z., Umar, M., Sabir, Z., Khan, J.A., Baleanu, D. 2018. A new stochastic computing paradigm for the dynamics of nonlinear singular heat conduction model of the human head. **The European Physical Journal Plus**, 133: 364.

[62] Raja, M.A.Z., Mehmood, A., Niazi, S.A., Shah, S.M. 2018. Computational intelligence methodology for the analysis of RC circuit modelled with nonlinear differential order system. **Neural Computing and Applications**, 30: 1905-1924.

[63] Raja, M.A.Z., Abbas, S. Syam, M.I. Wazwaz, A.M. 2018. Design of neuro-evolutionary model for solving nonlinear singularly perturbed boundary value problems. **Applied Soft Computing**, 62: 373-394.

[64] Iftikhar, A., Ahmad, F., Raja, M.A.Z., Ilyas, H., Anwar, N., Azad, Z. 2016. Intelligent computing to solve fifth-order boundary value problem arising in induction motor model. **Neural Computing and Applications**, 29: 1-18.

[65] Dua, V. 2011. An Artificial Neural Network approximation based decomposition approach for parameter estimation of system of ordinary differential equations. **Computers and Chemical Engineering**, 35(3): 545-553.

[66] Ling, H., Samarasinghe, S., Kulasiri, D. 2013. Novel recurrent neural network for modelling biological networks: Oscillatory p53 Interaction Dynamics. **BioSystems**. 114: 191-205.

[67] Xiangdong, F., Guanghua, H. 2013. A neural network for solving nonlinear multilevel programming problems. **2009 Chinese Control and Decision Conference**.

[68] Chudzik, S., Grys, S., Minkina, W. 2009. The application of the artificial neural network and hot probe method in thermal parameters determination of heat insulation materials Part 1 - thermal model consideration. **IEEE International Conference on Industrial Technology**.

[69] Aquino, A.U., Dadios, E.D. 2015. Solution to the ODE-mixing tank problem using artificial neural networks. In **8th IEEE Humanoid Nanotechnology Information Technology Communication and Control Environment and Management (HNICEM)**.

[70] Kennedy, J., Eberhart, R. 1995. Particle swarm optimization. In **Proceedings of the IEEE International Conference on Neural Networks**, 1942-1948.

[71] Cantaş, Y. 2014. Parçacık Sürü Optimizasyonu ile Tornalama İşlemlerinde Kesme Koşullarının Belirlenmesi, Dumlupınar Üniversitesi Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, Kütahya.

[72] http://yarpiz.com, Erişim tarihi: 23.12.2018.

[73] Rashedi, E., Nezamabadai-pour, H., Saryazdi, S. 2009. GSA: A Gravitational Search Algorithm. **Information Sciences**, 179: 2232-2248.

[74] Guvenc, U., Katatcıoğlu, F. 2015. GSA algoritmasının değişkenlerinin incelenmesi ve en uygun değerlerinin tespiti. **Journal of Advanced Technology Sciences**, 4(1): 24-35.

[75] Günel, K. Gör, İ. 2019. Gravitational search algorithm solutions of initial value problems for ordinary differential equations. **Advanced Mathematical Models Applications**, 4(3): 232-242.

[76] Günel, K. Gör, İ. 2019. A modification of artificial bee colony algorithm for solving initial value problems. **TWMS J. App. Eng. Math.**, 9(4): 810-821.

[77] Karaboğa, D. 2011. Yapay Zeka Optimizasyon Algoritmaları, Nobel yayın, Genişletilmiş 2. basım, 206.

[78] Dorigo M., DiCaro G., Gambardella L. M. 1999. Ant algorithms for discrete optimization. **Artificial Life**, 5: 137-172.

[79] Colorni A., Dorigo M., Maniezzo V. 1991. Distributed optimization by ant colonies, Paris, France. **Proceedings of ECAL-European Conference on Artificial Life**, 134-142.

[80] Yun H., Jeong S., Kim, K. 2013. Advanced harmony search with ant colony optimization for solving the traveling salesman problem. Hindawi Publishing Corporation. **Journal of Applied Mathematics**.

[81] Yıldız, A.R. 2011. Yapay arı koloni algoritması ile taşıt salıncak kolunun optimum boyutlarının bulunması, **3. Ulusal Tasarım İmalat ve Analiz Kongresi**.

[82] Atkinson, K., Han, W., Stewart, D.E. 2009. Numerical Solution of Ordinary Differential Equations, John Wiley & Sons, pp.190.

A. EKLER DİZİNİ

Algoritma 1 Başlangıç Değer Probleminin mABC algoritması ile çözümü için gerekli fonksiyonlar

function NET($t,\vec{\alpha},\vec{w},\vec{\beta}$) ▷ t noktasında yapay sinir ağı çözümü
    $m \leftarrow$ the length of $\vec{\alpha}$ ▷ m yapay sinir ağının nöron sayısı
    return $\sum_{i=1}^{m} \alpha_i \sigma(w_i t + \beta_i)$
end function

function DNET($t,\vec{\alpha},\vec{w},\vec{\beta}$) ▷ t noktasında yapay sinir ağının türevi
    $m \leftarrow \vec{\alpha}$'nın uzunluğu ▷ m yapay sinir ağının nöron sayısı
    return $\sum_{i=1}^{m} \alpha_i w_i \sigma(w_i t + \beta_i)(1 - \sigma(w_i t + \beta_i))$
end function

function TRIALY($t,t_0,y_0,\vec{p}$) ▷ t noktasında $\vec{p}$ parametre değerlerine göre deneme çözümleri
    return $y_0 + (t - t_0)$NET$(t,\vec{p})$
end function

function DTRIALY($t,t_0,y_0,\vec{p}$) ▷ $\frac{\partial y_T}{\partial t}$ değeri
    return NET$(t,\vec{\alpha},\vec{w},\vec{\beta}) + (t - t_0)$DNET$(t,\vec{\alpha},\vec{w},\vec{\beta})$
end function

function COST($\vec{t},t_0,y_0,\vec{p}$) ▷ Tüm girdi değerlerine göre uygunluk fonksiyonu
    $n \leftarrow, \vec{t}$ vektörünün uzunluğu ve girdi sayısı
    $E \leftarrow \frac{1}{n} \sum_{i=1}^{n} \{$ DTRIALY$(t_j,t_0,y_0,\vec{p}) - f(t_j,$TRIALY$(t_j,t_0,y_0,\vec{p}))$ $\}^2$
    return E
end function

Algoritma 2 Başlangıç Değer Probleminin Çözümü için mABC algoritması

procedure mABC_IVP ▷ IVP çözümü için mABC algoritması
  $[a,b]$ kapalı aralığını arama uzayı olarak belirle
  $h \leftarrow$ adım uzunluğu ▷ $h > 0$
  $t_0 \leftarrow a, y_0 \leftarrow$ problem için başlangıç şartları
  $m \leftarrow$ Yapay sinir ağının nöron sayısı
  Hiperkürenin yarıçapının başlangıç değeri, r
  for $j \leftarrow 0$ **to** n **do**
    $t_j \leftarrow a + hj$ ▷ Arama uzayında parçalanış oluşturma
  end for

  for $i \leftarrow 1$ **to** m **do**
    $\vec{\alpha}, \vec{\beta}, \vec{w} \in \mathbb{R}^m$ olmak üzere; $\vec{p}_i = (\vec{\alpha}_i, \vec{\beta}_i, \vec{w}_i)$ yapay sinir ağı parametrelerini yapay arı popülasyonu olarak başlangıçta kabul et
    COST($\vec{t}$,t_0,y_0,$\vec{p}_i$) fonksiyonu ile uygunluk değerleri aday çözümler için hesapla
  end for

  iterasyon $\leftarrow 1$
  repeat
    for all işçi arı $\vec{p}_i$ **do** ▷ işçi arı aşaması
      k, i'den farklı olacak şekilde, Eşitlik 2.39 kullanarak, işçi arı $\vec{p}_i$'nin komşuluğunda $\vec{p}_i'$ yeni çözüm üret
      COST($\vec{t}$,x_0,y_0,$\vec{p}_i'$) fonksiyonu kullanarak yeni çözümlerin uygunluk değerleri hesapla
      Yeni çözüm $\vec{p}_i'$'nin ve $\vec{p}_i$ işçi arıların uygunluk değerlerinin karşılaştırılması için greedy selection uygula ve yeni popülasyon belirle
    end for
    Eşitlik 2.38 ile verilen olasılık değerleri kontrol et ve gözlemci arı belirlemek için normalize et

    for all gözlemci arı $\vec{p}_i$ **do** ▷ gözlemci arı aşaması
      P_i olasılığına göre, gözlemci $\vec{p}_i$ arısının komşuluğunda yeni bir $\vec{p}_i'$ çözümü oluştur
      COST($\vec{t}$,x_0,y_0,$\vec{p}_i'$) fonksiyonu kullanarak yeni çözümlerin uygunluk değerlerini hesapla
      Yeni çözüm $\vec{p}_i'$'nin ve $\vec{p}_i$ gözlemci arıların uygunluk değerlerinin karşılaştırılması için greedy selection uygula ve yeni popülasyon belirle
    end for

27: **for all** aday çözüm $\vec{p}_i$ **do** ▷ kaşif arı aşaması
28: **if** aday terkedilen bir çözüm ise **then**
29: Eşitlik 2.40 kullanarak bu değeri yeni bir rastgele değer ile değiştir ve bu değer terkedilen çözüm hiperkürenin dışında olacak şekilde garantilemesi gerekir
30: **end if**
31: **end for**
▷ eleme aşaması
32: Eşitlik 2.40 kullanarak, yeni bir arı popülasyonu yarat ve bu değer bulunan en iyi çözüme komşu olması garanti olacak şekilde belirle
33: Yeni çözümlerin uygunluk değerlerini hesapla
34: Uygunluk değerlerine göre en zayıf çözüm dikkate alınmaz

35: En iyi besin kaynağı tespit et ve hafızada tut
36: Sönümleme oranı kullanarak hiperküre çapı azalt
37: iterasyon $\leftarrow$ iterasyon $+1$
38: **until** Maksimum iterasyona ulaşınca
39: **end procedure**

ÖZ GEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı	:	İclal GÖR
Doğum Yeri ve Tarihi	:	İspir, 01.03.1991

EĞİTİM DURUMU

Lisans Öğrenimi	:	Mimar Sinan Güzel Sanatlar Üniv. Fen-Edebiyat Fak., Matematik Böl.
Yüksek Lisans Öğrenimi	:	Aydın Adnan Menderes Üniversitesi Fen-Edebiyat Fak., Matematik Böl.
Doktora Öğrenimi	:	Aydın Adnan Menderes Üniversitesi Fen-Edebiyat Fak., Matematik Böl.
Bildiği Yabancı Diller	:	İngilizce

BİLİMSEL FAALİYETLERİ

a) Yayınlar
-Diğer :

Günel, K., Aşlıyan, R., Gör, İ., A Geometrical Modification of Learning Vector Quantization Method for Solving Classification Problems, **Suleyman Demirel University Journal of Natural and Applied Sciences**, Volume 20, Issue 3, 414-420, 2016.

Günel, K. Gör, İ. 2019. A modification of artificial bee colony algorithm for solving initial value problems, **TWMS J. App. Eng. Math.**, 4(1), 24-35.

Günel, K. Gör, İ. 2019. Gravitational Search Algorithm Solutions of Initial Value Problems for Ordinary Differential Equations. **Advanced Mathematical Models Applications**, 4(3): 232-242.

b) Bildiriler
-Uluslararası :

Gör, İ., Günel K. A Modification of Gravitational Search Algorithm with Hyper-Ellipsoids, International Conference of Mathematical Sciences (ICMS 2019), İstanbul, Turkey, September 04-08, 2019.

Günel K., Gör, İ. A new mutation approach for particle swarm optimization, International Conference on Mathematics and Mathematics Education (ICMME 2019), Konya, Turkey, July 11-13, 2019.

Günel K., Gör, İ., Tekeli, K. Swarm Intelligence Enchanged by Oblique Section Planes for solving Dirichlet Boundary Problems for ODEs, International Conference on Applied Mathematics in Engineering (ICAME), Balikesir, Turkey, June 27-29, 2018.

Gör, İ.,Günel K., Numerical Implementation of IVPs with ABC Algorithm, 24th Conference on Applied and Industrial Mathematics - CAIM 2016, Verbal, 15-18.09.2016.

Günel K., Gör, İ., Application of Gravitational Search Algorithm for Obtaining Numerical Solutions of IVPs, International Conference on Mathematics and Mathematics Education (ICMME 2016), Verbal, Elazig, Turkey, 14.05.2016.

Gör İ., Aşlıyan, R., Kalfa, Ö. Textile Image Classification and Multi-layer Perceptron, International Conference on Pure and Applied Mathematics (ICPAM 2015), Verbal, Van, 27.08.2015.

Gör, İclal., Günel K., Solving Systems of Linear Differential Equations by using Artificial Neural Networks, International Conference on Pure and Applied Mathematics (ICPAM 2015), Verbal, 27.08.2015.

Günel K., Gör, İclal., Artificial Neural Network Solutions of First Order Linear Differential Equations, 7th International Workshop on Differential Equations and Applications (WDEA 2015), Verbal, 28.07.2015.

Aşlıyan R., Günel K., Gör, İclal., Textile Image Classification Using Artificial Neural Networks, Global Journal on Technology, Vol 4 (2013), 3rd World Conference on Innovation and Computer Science (INSODE - 2013), Verbal, 27.04.2013.

Bayrak, M.A., Ulvi, İ. Identifying an unknown function in a parabolic equation by homotopy analysis method and comparison with the Adomian decomposition method, AIP (American Institute of Physics) Conference Proceedings, First International Conference on Analysis and Applied Mathematics (ICAAM), Gumushane, Turkey, 1470, 88-91, (2012).

-Ulusal :

Gör, İ., Çok Katmanlı Algılayıcı Yapay Sinir Ağı ile Lineer Diferansiyel Denklem Sisteminin Çözümü, XVIII. Akademik Bilişim Konferansı, Sözlü, 30.01.2016-05.02.2016.

Gör, İ., Günel K., Vektör Nicemleme Metoduna Geometrik Bir Yaklaşım, 28. Ulusal Matematik Sempozyumu, Sözlü, 08.09.2015.

c) Katıldığı Projeler

Gör, İ., Günel K., Diferansiyel Denklemlerin Yapay Sinir Ağları ile Nümerik Çözümleri, ÖYP-14011, Aydın Adnan Menderes Üniversitesi, Aydın, Türkiye, 14.09.2014-10.04.2020.

Gör, İ., Günel K., A Design and Implementation of Geometrical Learning Algorithm for Vector Quantization , ÖYP-12006, Aydın Adnan Menderes Üniversitesi, Aydın, Türkiye, 10.09.2012- 10.09.2014.

İŞ DENEYİMİ

Çalıştığı Kurumlar ve Yıl : Aydın Adnan Menderes Üniversitesi, Fen-Edebiyat Fakültesi, Matematik Bölümü (2012 - ...)

İLETİŞİM

E-posta Adresi : iclal@adu.edu.tr
Tarih : 10.04.2020