Differential Evolution Algorithm and Software Development: A Literature Review

1st Fatih Bildirici
Phd Student in Artificial Intelligence
Ankara University
Ankara, Turkiye
0000-0002-1730-4268

2nd Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

***Abstract*—Diferansiyel Evrim (DE) algoritması, evrimsel algoritmaların önemli bir alt kümesidir ve genetik operatörlerin etkin kullanımı sayesinde global optimizasyon problemlerinde yüksek performans göstermektedir. Bu çalışmada, DE algoritmasının yazılım geliştirme alanındaki potansiyel uygulamaları ve performans değerlendirme kriterleri incelenmiştir. DE algoritmasının yazılım bileşenlerinin optimizasyonu, test veri üretimi, hata ayıklama ve performans yönetimi gibi yazılım mühendisliği problemlerindeki kullanımı ele alınmıştır. Literatür taraması yöntemiyle mevcut araştırma boşlukları belirlenmiş ve gelecekteki araştırmalara yönelik öneriler sunulmuştur. DE algoritmasının yazılım geliştirme süreçlerinde geniş bir perspektiften incelenmesi, alan yazınına önemli katkılar sağlayacaktır.**

***Index Terms*—Differential Evolution, Software Development, Optimization, Performance Management, Literature Review**

I. Introduction

Giriş Evrimsel algoritmalar, doğadan esinlenerek geliştirilmiş, popülasyon tabanlı ve meta-sezgisel optimizasyon teknikleridir. Bu algoritmalar, genetik operatörler kullanarak çözüm uzayında global optimizasyon problemlerini çözmeyi amaçlar. Evrimsel algoritmalar ailesi, genetik algoritmalar (GA), genetik programlama (GP), evrim stratejileri (ES) ve diferansiyel evrim (DE) gibi çeşitli teknikleri içerir. Bu algoritmalar, karmaşık ve çok boyutlu problemler için etkin ve esnek çözüm yöntemleri sunarak geniş bir uygulama yelpazesinde başarılı sonuçlar elde etmişlerdir (Petrowski Hamida, 2017).

Diferansiyel evrim (DE) algoritması, evrimsel algoritmaların önemli bir alt kümesini oluşturur ve sürekli alanlarda global optimizasyon problemlerinin çözümünde büyük başarı sağlamıştır. İlk kez 1995 yılında Storn ve Price tarafından tanıtılan DE algoritması, popülasyon tabanlı doğası ve genetik operatörlerin etkin kullanımı sayesinde diğer evrimsel algoritmalardan ayrılır. DE algoritması, popülasyondaki aday çözümler üzerinde mutasyon, çaprazlama ve seçilim operatörlerini kullanarak yeni çözümler üretir ve bu çözümler arasından en iyilerini seçerek iteratif olarak çözüm kalitesini artırır (Price et al., 2015).

DE algoritmasının belirgin avantajları arasında basit yapısı, kolay uygulanabilirliği ve genellikle yüksek performans göstermesi bulunmaktadır. Bu özellikler, DE algoritmasını geniş bir uygulama yelpazesinde popüler bir optimizasyon aracı haline getirmiştir. DE algoritmasının etkili olduğu başlıca alanlar arasında parametrik optimizasyon, makine öğrenimi, veri madenciliği, görüntü işleme, sistem tanılama ve mühendislik problemleri yer almaktadır (Plagianakos et al., 2018).

Yazılım geliştirme alanında, diferansiyel evrim algoritmasının kullanımı ve potansiyel uygulamaları dikkat çekici bir araştırma alanı olarak ortaya çıkmaktadır. DE algoritması, yazılım geliştirme performans yönetimi, yazılım bileşenlerinin optimizasyonu, proje planlama, yazılım hata ayıklama ve performans ayarlaması gibi çeşitli yazılım mühendisliği problemlerinde kullanılabilir. Özellikle, yazılım bileşenlerinin geliştirilmesinde optimal performans yönetiminin sağlanması ve tahmin edilmesi başta birçok alanda kullanımı mevcuttur (Amiri & Karbin, 2015).

II. Evolutionary Algorithms and the Role of Differential Evolution Algorithm in Optimization Problems

Meta-sezgisel algoritmalar, karmaşık ve çok boyutlu optimizasyon problemlerinin çözümünde geniş uygulama alanı bulan esnek ve genel arama stratejileridir. Bu algoritmalar, problem uzayında global optimum çözümleri keşfetmek amacıyla çeşitli sezgisel yöntemler kullanarak yerel optimumlardan kaçınma yeteneğine sahiptirler (Yang, 2010). Meta-sezgisel algoritmalar arasında, biyolojik evrim süreçlerinden ilham alan evrimsel algoritmalar (EA) önemli bir yer tutar. Evrimsel algoritmalar, genetik operatörleri kullanarak popülasyon tabanlı bir yaklaşımı benimser ve nesiller boyunca çözümleri iteratif olarak iyileştirir (Malik et.al, 2021).

Evrimsel algoritmaların temel bileşenlerinden biri olan genetik algoritmalar (GA), doğal seçilim ve genetik çaprazlama süreçlerini model alır. Genetik algoritmalar, başlangıç popülasyonuyla başlar ve bu popülasyondaki bireyler üzerinde mutasyon, çaprazlama ve seçilim operatörlerini kullanarak yeni nesiller üretir. Bu süreç, problem uzayında daha iyi çözümler bulma amacıyla tekrarlanır. Genetik programlama (GP) ise benzer prensipleri takip ederek bilgisayar programlarının evrimini sağlar ve özellikle otomatik programlama ve makine öğrenimi alanlarında uygulanır (Poli & Koza, 2013).

Diferansiyel evrim (DE) algoritması, evrimsel algoritmalar ailesinin kritik bir üyesi olarak öne çıkar. DE al-

goritması, sürekli optimizasyon problemlerine odaklanarak popülasyondaki aday çözümler üzerinde diferansiyel mutasyon, çaprazlama ve seçilim operatörlerini kullanır. DE algoritmasının belirgin özelliklerinden biri, mutasyon operatörünün diferansiyel vektör farklarını kullanarak yeni çözümler üretmesidir. Bu yöntem, DE algoritmasının hızlı ve etkin bir şekilde global optimuma yakın çözümler bulmasını sağlar (Mayer et.al, 2015).

Diferansiyel evrim algoritması, diğer evrimsel algoritmalardan farklı olarak, özellikle yüksek boyutlu ve karmaşık optimizasyon problemlerinde yüksek performans göstermesi ile tanınır. DE algoritmasının basit yapısı ve kolay uygulanabilirliği, geniş bir uygulama yelpazesinde yaygın olarak kullanılmasını sağlamıştır. Özellikle mühendislik optimizasyonu, veri madenciliği ve makine öğrenimi gibi alanlarda DE algoritmasının üstün performansı dikkat çekmektedir (Qing, 2009). Bu bağlamda, diferansiyel evrim algoritması, evrimsel algoritmalar ailesinin etkin ve güçlü bir üyesi olarak optimizasyon problemlerinin çözümünde önemli bir araç olarak kabul edilmektedir.

III. Diferansiyel Evrim Algoritmasının Adımları

Diferansiyel Evrim (DE) algoritması, evrimsel hesaplama tekniklerinin bir alt kümesi olup, sürekli alanlarda global optimizasyon problemlerinin çözümünde etkinliği kanıtlanmış bir yöntemdir. DE algoritması, iteratif bir süreç olup, başlangıç popülasyonunun oluşturulması, mutasyon, çaprazlama ve seçilim aşamalarından oluşur. Bu bölümde, DE algoritmasının temel adımları ayrıntılı olarak ele alınacaktır.

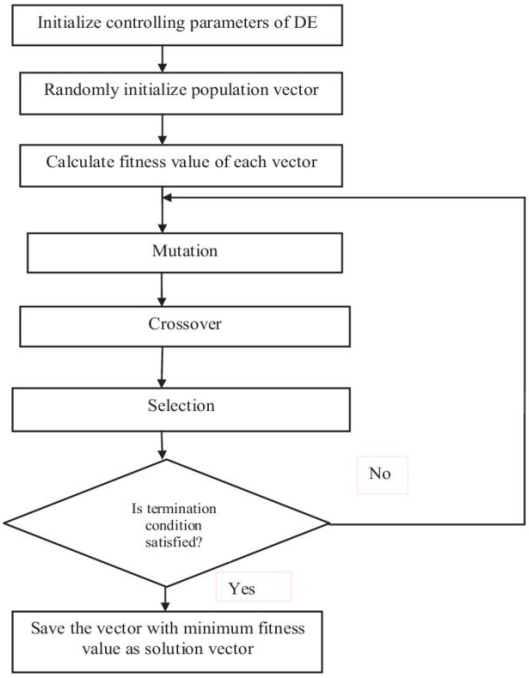

Fig. 1. Diferansiyel Evrim Algoritmasının Adımları

A. Başlangıç Popülasyonunun Oluşturulması

DE algoritması, D-boyutlu bir çözüm uzayında rastgele seçilen konumlardan oluşan başlangıç popülasyonunun oluşturulması ile başlar. Popülasyon büyüklüğü NP ile ifade edilir ve her birey $x_{i,G}$ şeklinde gösterilir. Burada $i = 1, 2, \ldots, NP$ ve G nesil sayısını temsil eder. Başlangıç popülasyonu şu şekilde oluşturulur:

$$x_{i,0} = x_{\min} + r_i \cdot (x_{\max} - x_{\min})$$

Burada $x_{\min}$ ve $x_{\max}$ çözüm uzayının alt ve üst sınırlarını, r_i ise [0,1] aralığında rastgele bir vektörü temsil eder. Başlangıç popülasyonunun çeşitliliği, algoritmanın global optimuma ulaşma olasılığını artırır (reference needed).

B. Mutasyon

Mutasyon aşaması, DE algoritmasının en belirgin ve kritik aşamasıdır. Bu aşamada, mevcut popülasyondan rastgele seçilen üç farklı birey kullanılarak bir mutasyon vektörü oluşturulur. Mutasyon vektörü $v_{i,G+1}$ ile gösterilir ve şu şekilde hesaplanır:

$$v_{i,G+1} = x_{r1,G} + F \cdot (x_{r2,G} - x_{r3,G})$$

Burada $r1, r2$ ve $r3$, i den farklı ve rastgele seçilen birey indeksleridir; F ise skalalama faktörüdür ve genellikle [0, 1] aralığında bir değerdir. Diferansiyel mutasyon mekanizması, çözüm uzayında daha yönlendirilmiş bir arama yapılmasına olanak tanır ve yerel minimumlardan kaçınmayı sağlar (reference needed).

C. Çaprazlama

Çaprazlama aşamasında, mutasyon sonucu elde edilen vektör ile mevcut birey arasında bir kombinasyon yapılır. Bu işlem, her bir çözüm bileşeninin belirli bir olasılıkla mutasyon vektöründen mi yoksa mevcut bireyden mi alınacağını belirler. Çaprazlama vektörü $u_{i,G+1}$ ile gösterilir ve şu şekilde hesaplanır:

$$u_{j,i,G+1} = \begin{cases} v_{j,i,G+1} & \text{eğer } rand_j \leq CR \text{ veya } j = j_{rand} \\ x_{j,i,G} & \text{aksi halde} \end{cases}$$

Burada $j = 1, 2, \ldots, D$ çözüm bileşenlerini, CR çaprazlama oranını ve $rand_j$ [0,1] aralığında rastgele bir sayıyı temsil eder. j_{rand} ise rastgele seçilen bir indekstir ve en az bir bileşenin mutasyon vektöründen gelmesini garanti eder (reference needed). Çaprazlama işlemi, popülasyonun genetik çeşitliliğini korur ve çözümlerin daha geniş bir uzayda arama yapmasını sağlar.

D. Seçilim

Seçilim aşaması, yeni oluşturulan bireyler ve mevcut bireyler arasında bir rekabet sürecini başlatır. Bu aşamada, uygunluk fonksiyonu $f(x)$ kullanılarak her bireyin kalitesi değerlendirilir ve daha iyi uygunluk değerine sahip olan bireyler bir sonraki nesile aktarılır. Seçilim işlemi, aşağıdaki gibi ifade edilir:

$$x_{i,G+1} = \begin{cases} u_{i,G+1} & \text{eğer } f(u_{i,G+1}) \leq f(x_{i,G}) \\ x_{i,G} & \text{aksi halde} \end{cases}$$

Bu mekanizma, algoritmanın iteratif olarak iyileşmesini ve en iyi çözümleri bulmasını sağlar. Seçilim süreci, DE'nin daha etkin bir şekilde en iyi çözümleri bulmasını ve global optimuma ulaşma olasılığını artırır (reference needed).

E. Termininasyon Kriteri

Her iterasyon sonunda, algoritmanın durma kriterleri kontrol edilir. Bu kriterler arasında maksimum nesil sayısına ulaşma, belirli bir uygunluk değerine ulaşma veya çözümdeki değişikliklerin belirli bir eşik değerinin altına düşmesi gibi durumlar bulunabilir. Eğer durma kriterleri sağlanmışsa, algoritma sonlandırılır ve en iyi çözüm olarak kabul edilen birey raporlanır.

Diferansiyel evrim algoritmasının bu temel adımları, algoritmanın güçlü performansını ve esnekliğini sağlayan kritik bileşenlerdir. Başlangıç popülasyonunun rastgele oluşturulması, diferansiyel mutasyon ile yönlendirilmiş arama, çaprazlama ile genetik çeşitliliğin korunması ve seçilim ile iteratif iyileştirme, DE algoritmasının yüksek performans göstermesini sağlar. Bu adımların doğru bir şekilde uygulanması, DE algoritmasının geniş bir yelpazede optimizasyon problemlerinde etkili bir şekilde kullanılmasını mümkün kılar (reference needed). Algoritmanın esnek yapısı ve güçlü performansı, diferansiyel evrim algoritmasını evrimsel hesaplama alanında önemli bir araç haline getirmektedir.

IV. Historical Development and Application Areas of Differential Evolution Algorithm

Diferansiyel evrim (DE) algoritmasının tarihsel sürecine baktığımızda, evrimsel algoritmalar ailesinin güçlü ve etkin bir üyesi olarak, 1995 yılında Rainer Storn ve Kenneth Price tarafından tanıtıldığı görülebilir (Zhu, 2018). Storn ve Price, DE algoritmasını geliştirirken evrimsel hesaplamanın temel prensiplerinden esinlenmiş ve genetik operatörlerin etkinliğini artırmak amacıyla yeni bir mutasyon mekanizması önermiştir. DE algoritması, popülasyon tabanlı doğası ve diferansiyel mutasyon yöntemi ile diğer evrimsel algoritmalardan ayrılır (Feoktistov, 2006). Bu mekanizma, mevcut popülasyon üyeleri arasındaki farkları kullanarak yeni çözümler üretir ve böylece çözüm uzayında daha etkin bir arama yapılmasını sağlar (Wong & Dong, 2015).

Diferansiyel evrim algoritmasının tarihi, evrimsel hesaplamanın geniş bağlamında yer almaktadır. Evrimsel hesaplamanın tarihsel kökenleri, 1950'lere kadar uzanır ve bu dönemde genetik algoritmalar (GA), evrim stratejileri (ES) ve genetik programlama (GP) gibi çeşitli yöntemler geliştirilmiştir. Bu yöntemler, biyolojik evrim süreçlerini taklit ederek, optimizasyon problemlerine yenilikçi çözümler sunmuştur (Vikhar, 2016). Diferansiyel evrim algoritması, bu erken çalışmaların bir devamı niteliğindedir ve özellikle sürekli optimizasyon problemlerine odaklanmıştır. DE algoritması, tanıtıldığı dönemde diğer evrimsel algoritmalara kıyasla daha az parametre ayarına ihtiyaç duyması ve yüksek performans göstermesi ile büyük ilgi görmüştür (Mayer et.al, 2005).

DE algoritmasının ayırt edici özelliklerinden biri, genetik operatörlerin kullanımında basitlik ve etkinliğin bir arada bulunmasıdır. Mutasyon, çaprazlama ve seçilim operatörleri, diferansiyel evrim algoritmasında özel bir biçimde uygulanır ve bu sayede global optimuma ulaşma sürecinde yüksek performans elde edilir. Diğer evrimsel algoritmalardan farklı olarak, DE algoritması parametrik optimizasyon problemlerinde daha az parametre ayarına ihtiyaç duyar (Karaboğa & Ökdem, 2004). Genetik algoritmalar (GA) ve evrim stratejileri (ES), genellikle popülasyon büyüklüğü, mutasyon oranı ve çaprazlama oranı gibi bir dizi parametrenin dikkatli bir şekilde ayarlanmasını gerektirirken, DE algoritmasında sadece skalalama faktörü (F) ve çaprazlama oranı (CR) gibi birkaç temel parametre bulunur (Hegerty et.al, 2009). Bu sadelik, DE algoritmasının kullanımını ve uyarlamasını kolaylaştırır.

Diferansiyel evrim algoritmasının uygulama alanları oldukça geniştir ve mühendislik optimizasyonu, veri madenciliği, makine öğrenimi, sistem tanılama ve görüntü işleme gibi çeşitli alanlarda başarıyla uygulanmıştır. Örneğin, yapay sinir ağlarının eğitimi, genetik programlama, kontrol sistemlerinin tasarımı ve kimyasal proses optimizasyonu gibi spesifik uygulamalarda DE algoritmasının üstün performansını ortaya koyan yaklaşımlar bulunmaktadır (Plagianakos, 2008). DE algoritması, parametrik optimizasyon problemlerinin çözümünde yaygın olarak kullanılmaktadır ve bu alanlarda diğer optimizasyon tekniklerine kıyasla daha hızlı ve güvenilir sonuçlar elde etme kapasitesine sahiptir (Tanabe & Fukunaga, 2013).

Mühendislik alanında, DE algoritması özellikle mekanik ve elektrik mühendisliği problemlerinde kullanılmaktadır. Bu problemler genellikle çok boyutlu ve karmaşık yapıda olup, DE algoritmasının esnek ve güçlü optimizasyon yeteneklerinden yararlanır (Gong et.al, 2009). Veri madenciliği ve makine öğrenimi alanlarında, DE algoritması büyük veri setlerinin analizi ve model optimizasyonu gibi görevlerde etkin bir araç olarak kullanılmaktadır. Ayrıca, biyoinformatik ve finansal modelleme gibi hızla gelişen alanlarda da DE algoritmasının uygulanabilirliği üzerine yapılan çalışmalar, algoritmanın önemini ve etkisini daha da pekiştirmektedir (Pant et.al, 2020).

Diferansiyel evrim algoritması, evrimsel algoritmalar ailesinin güçlü ve etkin bir üyesi olarak, sürekli ve yüksek boyutlu optimizasyon problemlerinin çözümünde önemli bir araç olarak kabul edilmektedir. Algoritmanın tarihsel gelişimi ve geniş uygulama alanları, onun optimizasyon literatüründe önemli bir yer edinmesini sağlamıştır. DE algoritmasının basit yapısı, geniş bir uygulama yelpazesinde kolayca adapte edilebilmesini sağlamış ve bu da DE algoritmasının popülerliğini artırmıştır. Bu bağlamda, diferansiyel evrim algoritmasının gelecekteki araştırmalar ve uygulamalar için sunduğu potansiyel, bilimsel ve endüstriyel çevrelerde dikkat çekmeye devam edecektir.

V. Differences of Differential Evolution Algorithm from Other Evolutionary Algorithms

Diferansiyel evrim (DE) algoritması, evrimsel algoritmalar ailesinin önemli bir üyesi olarak, diğer evrimsel algoritmalardan belirgin farklılıklar göstermektedir. Bu farklılıklar, algoritmanın temel prensiplerinde ve genetik operatörlerin kullanımında ortaya çıkar. DE algoritmasının genetik algoritmalar (GA), evrim stratejileri (ES) ve genetik programlama (GP) ile karşılaştırılması, algoritmanın özgün yönlerini ve etkinliğini ortaya koymak açısından önemlidir (Kachitvichyanukul, 2012).

Genetik algoritmalar (GA), evrimsel algoritmaların en bilinen üyelerinden biridir ve biyolojik evrim süreçlerini taklit eder. GA, popülasyon tabanlı bir yöntem olup, kromozom olarak adlandırılan aday çözümlerden oluşan bir popülasyon kullanır. Her bir kromozom, çözüm uzayında bir noktayı temsil eder ve genetik operatörler (mutasyon, çaprazlama ve seçilim) aracılığıyla yeni çözümler üretilir (Kramer & Kramer, 2017). Diferansiyel evrim algoritması ile genetik algoritmalar arasındaki en temel fark, mutasyon mekanizmasında yatmaktadır. GA'da mutasyon, bireysel kromozomların genlerinde rastgele değişiklikler yaparak çeşitliliği artırır. Buna karşılık, DE algoritmasında mutasyon, mevcut popülasyon üyeleri arasındaki vektör farklarını kullanarak yeni bireyler üretir. DE'nin mutasyon formülü, vektör farklarının kullanımıyla daha yönlendirilmiş ve etkin bir arama sağlar, bu da genetik algoritmaların rastgele mutasyon yöntemine göre daha hızlı bir şekilde global optimuma ulaşmayı mümkün kılar (Hegerty et.al, 2009).

Çaprazlama operatöründe de farklar mevcuttur. GA'da, iki ebeveyn kromozomunun genlerinin belirli bir kısmı değiştirilerek yeni bir kromozom oluşturulur. DE'de ise çaprazlama, mutasyon sonucu elde edilen birey ile mevcut birey arasında yapılır. DE'nin çaprazlama mekanizması, belirli bir çaprazlama oranı (CR) kullanarak mutasyon ve mevcut bireylerin bileşenlerini birleştirir. Bu süreç, çözüm kalitesini artırmak ve algoritmanın etkinliğini korumak için kritik bir rol oynar (Tušar & Filipič, 2007). Seçilim aşamasında, hem GA hem de DE, daha iyi uygunluk değerine sahip bireyleri seçer. Ancak, DE algoritmasında seçilim, mutasyon ve çaprazlama sonucu elde edilen bireylerin mevcut popülasyonla karşılaştırılması yoluyla yapılır. GA'da ise genellikle turnuva seçilimi veya rulet seçilimi gibi yöntemler kullanılır. Bu seçilim süreci, DE'nin daha etkin bir şekilde en iyi çözümleri bulmasını sağlar (Wong & Dong, 2005).

Evrim algorithms, genetik algoritmalara benzer şekilde biyolojik evrim süreçlerinden ilham alır ancak daha çok sürekli optimizasyon problemlerine odaklanır. Bunlar, özellikle mutasyon operatörünün önemine vurgu yapar ve genellikle normal dağılım kullanarak mutasyon yapar (Yu & Gen, 2019). DE algoritması ile evrim stratejileri arasındaki en belirgin fark, mutasyon mekanizması ve parametre ayarlarında görülür. Bu algoritmalar, mutasyon için genellikle normal dağılım kullanarak, her bir bireyin genlerini küçük rastgele değişikliklerle değiştirmeyi tercih eder. Buna karşılık, DE algoritması diferansiyel mutasyon kullanır ve mevcut popülasyon üyeleri arasındaki vektör farklarını kullanarak yeni bireyler üretir. Bu yöntem, evrimsel algoritmaların rastgele mutasyonuna kıyasla daha yönlendirilmiş bir arama sağlayabilir (Hrstka et.al, 2003).

Evrimsel algoritmik yapılar, genellikle (μ, λ) veya $(\mu + \lambda)$ seçilim stratejileri kullanılır. Bu stratejiler, bir popülasyondan μ ebeveyn seçer ve λ çocuk üretir. DE algoritmasında ise seçilim, her bir aday çözümün mutasyon ve çaprazlama sonucu elde edilen yeni çözümlerle karşılaştırılması yoluyla yapılır. Bu seçilim süreci, DE'nin daha etkin bir şekilde en iyi çözümleri bulmasını sağlar (Zeng et.al, 2021). DE algoritmasının seçilim mekanizması, çözüm kalitesini sürekli olarak iyileştirir ve algoritmanın global optimuma ulaşma yeteneğini artırır.

Genetik programlama (GP), genetik algoritmalara benzer prensiplere dayanır ancak aday çözümler olarak bilgisayar programlarını kullanır. GP, genellikle ağaç yapısında temsil edilen programları evrimleştirir ve bu programlar üzerinde genetik operatörler uygular (Moraglio et.al, 2012). DE algoritması ile genetik programlama arasındaki en büyük fark, temsil ve arama mekanizmalarındadır. GP'de, aday çözümler ağaç yapısında temsil edilen programlar iken, DE'de sürekli alanlarda temsil edilen vektörler kullanılır. GP, program ağaçları üzerinde çaprazlama ve mutasyon yaparak yeni programlar üretir. DE ise diferansiyel mutasyon ve vektör farkları kullanarak yeni çözümler üretir (Feoktistov, 2006).

Çaprazlama mekanizması da farklılık gösterir. GP'de, iki program ağacı arasında gen değiş tokuşu yapılarak yeni programlar oluşturulur. DE'de ise çaprazlama, mutasyon sonucu elde edilen vektörler ile mevcut vektörler arasında yapılır. Ayrıca, seçilim sürecinde GP, programların uygunluk değerlerine dayanarak en iyi programları seçerken, DE algoritması vektörlerin uygunluk değerlerini kullanarak en iyi çözümleri belirler (Zaharie, 2009). Bu farklılıklar, DE algoritmasının sürekli ve yüksek boyutlu optimizasyon problemlerinde neden bu kadar etkili olduğunu açıklamaktadır.

Sonuç olarak, diferansiyel evrim algoritması, genetik algoritmalar, evrim stratejileri ve genetik programlama gibi diğer evrimsel algoritmalardan önemli farklılıklar göstermektedir. DE algoritmasının diferansiyel mutasyon mekanizması, yönlendirilmiş arama kabiliyeti ve etkin parametre ayarları, onu sürekli ve yüksek boyutlu optimizasyon problemlerinde güçlü bir araç haline getirmektedir (Yang et.al, 2008). Bu farklılıkların derinlemesine anlaşılması, DE algoritmasının diğer evrimsel algoritmalarla karşılaştırıldığında neden bu kadar etkili olduğunu ve geniş bir uygulama yelpazesinde başarıyla kullanılmasını açıklamaktadır.

VI. Application Areas of Differential Evolution Algorithm

Diferansiyel evrim (DE) algoritması, geniş bir uygulama yelpazesine sahip olup, çeşitli bilimsel ve mühendislik alanlarında etkin bir optimizasyon aracı olarak kullanılmaktadır. Bu bölümde, DE algoritmasının mühendislik optimizasyonu, veri madenciliği, makine öğrenimi, sistem tanılama, görüntü işleme, biyoinformatik ve finansal modelleme gibi alanlardaki

spesifik uygulamaları teknik ve akademik bir üslupla detaylandırılacaktır.

Mühendislik optimizasyonu, DE algoritmasının en yaygın kullanım alanlarından biridir. Mekanik mühendisliği problemlerinde, DE algoritması tasarım optimizasyonu, yapısal analiz ve dinamik sistemlerin kontrolü gibi çeşitli alanlarda uygulanmaktadır. Örneğin, karmaşık mekanik sistemlerin optimal tasarımı, genellikle çok sayıda değişken ve kısıtlama içeren zor bir problemdir (Liao, 2010). DE algoritması, diferansiyel mutasyon ve çaprazlama mekanizmaları sayesinde geniş bir çözüm uzayını hızlı ve etkin bir şekilde tarayarak bu tür problemlerde başarılı sonuçlar elde etmektedir (Gong et.al, 2014). DE algoritması, aerodinamik şekil optimizasyonu, ısı değiştirici tasarımı ve robotik kol tasarımı gibi spesifik mekanik mühendisliği uygulamalarında da etkili bir şekilde kullanılmaktadır (Masike et.al, 2024).

Elektrik mühendisliği problemlerinde, DE algoritması enerji sistemleri optimizasyonu, elektrik devre tasarımı ve kontrol sistemlerinin iyileştirilmesi gibi alanlarda yaygın olarak kullanılmaktadır. DE algoritması, özellikle büyük ölçekli ve karmaşık optimizasyon problemlerinde üstün performans göstermektedir. Örneğin, enerji üretim sistemlerinde maliyet minimizasyonu ve verimlilik maksimizasyonu gibi çoklu hedefli optimizasyon problemleri, DE algoritması ile etkili bir şekilde çözülebilir (Tiwari & Srivastava, 2016). Ayrıca, yenilenebilir enerji sistemlerinin optimizasyonu, güç dağıtım ağlarının güvenilirliği ve elektrikli araçların şarj altyapısının optimizasyonu gibi uygulamalarda da DE algoritması başarılı sonuçlar vermektedir (Moradi et.al, 2015).

Veri madenciliği ve makine öğrenimi alanlarında, DE algoritması büyük veri setlerinin analizi ve model optimizasyonu gibi görevlerde etkin bir araç olarak kullanılmaktadır. Büyük veri setlerinin analizi, genellikle çok sayıda değişkenin ve ilişkilerin olduğu karmaşık bir süreçtir. DE algoritması, bu tür problemleri çözmek için uygun bir araçtır, çünkü diferansiyel mutasyon mekanizması, geniş veri uzayında etkili bir keşif yapmayı sağlar (Karaseva, 2022). Makine öğreniminde, DE algoritması yapay sinir ağlarının eğitimi, hiperparametre optimizasyonu ve destek vektör makineleri gibi modellerin iyileştirilmesinde kullanılmaktadır. Örneğin, DE algoritması, derin öğrenme modellerinin eğitimi için hiperparametre ayarlarının optimize edilmesinde başarılı bir şekilde kullanılmaktadır (Da Silva, 2010).

Sistem tanılama ve görüntü işleme alanlarında, DE algoritması sistem parametrelerinin optimizasyonu ve görüntü iyileştirme gibi görevlerde başarılı bir şekilde uygulanmaktadır. Sistem tanılamada, DE algoritması karmaşık sistemlerin dinamik davranışlarını modellemek için kullanılabilir. Örneğin, endüstriyel süreçlerin kontrolü ve izlenmesi için DE algoritması ile optimizasyon yapılabilir (Zaher & Pant, 2017). Görüntü işleme alanında ise, DE algoritması görüntü segmentasyonu, özellik çıkarımı ve desen tanıma gibi görevlerde etkili sonuçlar vermektedir. Örneğin, tıbbi görüntüleme alanında DE algoritması, MRI ve CT görüntülerinin segmentasyonu ve tümör tespiti gibi uygulamalarda kullanılmaktadır (Rajesh & Kumar, 2022).

Biyoinformatik ve finansal modelleme alanlarında, DE algoritması biyolojik verilerin analizi ve finansal piyasa modellerinin optimizasyonu gibi görevlerde kullanılmaktadır. Biyoinformatikte, genetik dizilim analizi, protein yapı tahmini ve biyolojik ağların modellenmesi gibi problemler, DE algoritması ile etkili bir şekilde çözülebilir. Örneğin, DE algoritması, genetik dizilimlerin hizalanması ve filogenetik ağaçların oluşturulması gibi görevlerde kullanılmaktadır (Salihov et.al, 2021). Finansal modellemede ise, portföy optimizasyonu, risk analizi ve fiyatlandırma modellerinin iyileştirilmesi gibi görevlerde DE algoritması başarılı sonuçlar vermektedir. Örneğin, DE algoritması, finansal portföylerin risk-getiri dengesini optimize etmek için kullanılmaktadır (Krink & Paterlini, 2008).

Sonuç olarak, diferansiyel evrim algoritması, çok çeşitli uygulama alanlarında etkili ve esnek bir optimizasyon aracı olarak öne çıkmaktadır. Algoritmanın geniş bir yelpazede başarılı bir şekilde uygulanabilmesi, diferansiyel mutasyon ve çaprazlama mekanizmalarının gücünü yansıtmaktadır (Suganthan, 2012). DE algoritmasının mühendislikten biyoinformatiğe, veri madenciliğinden finansal modellemeye kadar geniş bir spektrumda sunduğu çözümler, onun optimizasyon problemlerindeki üstün performansını ve önemini ortaya koymaktadır.

VII. Use of Differential Evolution Algorithm in Software Development Processes

Diferansiyel evrim (DE) algoritması, optimizasyon problemlerine sunduğu etkin ve esnek çözümlerle yazılım geliştirme süreçlerinde önemli bir yer edinmiştir. Yazılım geliştirme çabası tahmini, test veri üretimi, hata ayıklama ve yazılım gereksinim seçimi gibi kritik alanlarda DE algoritmasının sağladığı yüksek performans, bu alanlarda başarı elde edilmesini sağlamaktadır. DE algoritmasının yazılım geliştirme süreçlerindeki kullanımını detaylı bir şekilde inceleyen çalışmalar, algoritmanın geniş uygulama yelpazesi ve üstün performansını vurgulamaktadır.

Yazılım geliştirme projelerinde maliyet ve zaman yönetimi açısından doğru çaba tahmini yapmak büyük önem taşır. Bu alanda yapılan araştırmalar, DE algoritmasının çaba tahmini süreçlerinde sağladığı üstün performansı ortaya koymaktadır. Örneğin, Singal, Kumari ve Sharma (2020) tarafından yapılan çalışmada, DE algoritmasının mutasyon stratejilerinin yazılım geliştirme çabası tahmininde yüksek doğruluk sağladığı belirtilmiştir. Benzer şekilde, Benala ve Mall (2018), analoji tabanlı yazılım geliştirme çabası tahmininde DE algoritmasının etkinliğini incelemiş ve bu yöntemle optimize edilmiş sonuçlar elde edilmiştir. Khatibi Bardsiri ve Hashemi (2016) tarafından yapılan çalışma ise, DE algoritmasının yazılım hizmetleri geliştirme çabası tahmininde etkili olduğunu göstermektedir. Bu çalışmalar, DE algoritmasının çaba tahmininde geleneksel yöntemlere kıyasla üstün performans sergilediğini açıkça ortaya koymaktadır.

Yazılım test süreçlerinde, yazılımın farklı senaryolar altında nasıl davrandığını değerlendirmek için yeterli ve çeşitli test verilerine ihtiyaç duyulur. Becerra, Sagarna ve Yao (2009),

DE algoritmasının yazılım test verisi üretiminde etkili bir araç olarak kullanılabileceğini göstermiştir. DE algoritması ile otomatik olarak üretilen test veri setleri, yazılımın daha kapsamlı ve çeşitli senaryolar altında test edilmesini sağlar. Ayrıca, Yang ve diğerleri (2021) tarafından yapılan çalışmada, DE algoritması kullanılarak yazılım hatalarının tespit edilmesi ve düzeltilmesi için optimize edilmiş çözümler sunulmuştur. Bu, hata ayıklama süreçlerinin otomasyonunu ve etkinliğini artırarak yazılım kalitesini yükseltir.

Yazılım gereksinim mühendisliği süreci, projenin başarısı için kritik bir bileşendir. Kumari, Srinivas ve Gupta (2012), kuantum esinli çok amaçlı diferansiyel evrim algoritmasını (QMDEA) kullanarak yazılım gereksinim seçimi üzerine bir çalışma gerçekleştirmiştir. Bu algoritma, gereksinim mühendisliği sorunlarının çözümünde DE algoritmasının potansiyelini göstermektedir. Gereksinimlerin optimal seçimi, projenin hedeflerine ulaşmasında önemli bir rol oynar ve DE algoritması bu süreçte etkili bir araç olarak kullanılabilir.

DE algoritması, yazılım proje planlaması ve yönetiminde de önemli bir rol oynar. Amiri ve Barbin (2015), DE algoritmasını kullanarak yazılım proje zamanlama problemini optimize etmiş ve bu yaklaşımın, proje yöneticileri için en iyi zamanlama seçeneklerini belirlemede etkili olduğunu göstermiştir. Biju ve Victoire (2015), DE algoritmasını yazılım proje zamanlama problemlerinde kullanarak proje yönetimi süreçlerinde önemli iyileştirmeler sağlamıştır. Bu çalışmalar, DE algoritmasının proje planlaması ve yönetimi süreçlerinde sağladığı avantajları ortaya koymaktadır. DE algoritmasının yazılım geliştirme süreçlerinde sunduğu diğer bir önemli katkı, performans optimizasyonudur. Yazılımın hızını, verimliliğini ve kaynak kullanımını iyileştirmek amacıyla kullanılan DE algoritması, bellek yönetimi, işlemci kullanımı ve veri akışlarının optimize edilmesi gibi görevlerde başarılı sonuçlar vermektedir. Özellikle yüksek performans gerektiren uygulamalarda, paralel işlem ve dağıtık sistemlerin performans ayarlamasında DE algoritması etkili bir şekilde kullanılmaktadır. Yaghoobi (2020) tarafından yapılan çalışmada, yazılım güvenilirliği modellerinin parametre optimizasyonunda DE algoritmasının kullanımı incelenmiş ve başarılı sonuçlar elde edilmiştir.

Sonuç olarak, diferansiyel evrim algoritması, yazılım geliştirme süreçlerinde geniş bir uygulama yelpazesine sahip etkili bir optimizasyon aracıdır. Yazılım geliştirme çabası tahmini, test veri üretimi, hata ayıklama ve yazılım gereksinim seçimi gibi alanlarda başarılı bir şekilde kullanılmaktadır. Literatürdeki çalışmalar, DE algoritmasının yazılım geliştirme süreçlerindeki esnekliği ve etkinliğini vurgulamaktadır. Bu bağlamda, DE algoritmasının yazılım mühendisliği ve ilgili alanlarda kullanımının artması ve daha sofistike uygulamalarla entegrasyonu beklenmektedir. DE algoritmasının sunduğu bu güçlü optimizasyon yetenekleri, yazılım geliştirme süreçlerinde onun vazgeçilmez bir araç olmasını sağlamaktadır. Gelecekte, DE algoritmasının yazılım mühendisliği uygulamalarında daha geniş bir yelpazede ve daha karmaşık problemler için kullanılması öngörülmektedir.

VIII. Use of Differential Evolution Algorithm in Software Development Processes

Diferansiyel evrim (DE) algoritması, son birkaç on yılda evrimsel hesaplamanın önemli bir parçası olarak geniş bir kabul görmüştür. Algoritmanın temel prensipleri, basitlik, güçlü performans ve esneklik üzerine kuruludur. Ancak, DE algoritmasının gelecekteki potansiyelini tam olarak değerlendirebilmek için yeni uygulama alanları, algoritmik iyileştirmeler ve çoklu hedefli optimizasyon problemleri gibi konulara odaklanmak gerekmektedir (Ahmad et.al, 2022). Diferansiyel evrim algoritmasının yeni uygulama alanları, algoritmanın esnekliğini ve geniş kullanım potansiyelini göstermektedir. DE algoritmasının başarılı olduğu mevcut alanlar dışında, yeni ve gelişmekte olan teknolojilerde de potansiyel uygulama alanları bulunmaktadır. Örneğin, kuantum hesaplama ve kuantum algoritmaların optimizasyonu, DE algoritmasının yenilikçi uygulama alanlarından biridir (Deng et al., 2020). Kuantum bilgisayarlarının performansını optimize etmek için DE algoritmasının kullanılması, bu alandaki araştırmalara önemli katkılar sağlayabilir. Ayrıca, biyoteknoloji ve genetik mühendisliği gibi alanlarda da DE algoritması kullanılabilir. Genetik dizilimlerin analizi, protein yapı tahmini ve biyolojik ağların modellenmesi gibi biyoinformatik problemlerinde DE algoritması, mevcut yöntemlere kıyasla daha etkin çözümler sunabilir (Salihov et al., 2021).

Algoritmik iyileştirmeler, DE algoritmasının performansını ve verimliliğini artırmak için önemli bir araştırma yönüdür. DE algoritmasının temel prensipleri, mutasyon, çaprazlama ve seçilim operatörlerinin etkin kullanımı üzerine kuruludur. Bu operatörlerin optimize edilmesi ve adaptif mekanizmaların eklenmesi, algoritmanın performansını daha da iyileştirebilir. Örneğin, adaptif diferansiyel evrim (ADE) algoritmaları, kontrol parametrelerini dinamik olarak ayarlayarak performansı artırmayı hedefler (Zhan et.al, 2019). Ayrıca, algoritmanın mutasyon ve çaprazlama stratejilerinin iyileştirilmesi, daha hızlı ve güvenilir çözümler elde edilmesini sağlayabilir. Özellikle, karmaşık ve yüksek boyutlu problemler için geliştirilmiş yeni stratejiler, DE algoritmasının optimizasyon kabiliyetini artırabilir (Ahmad et al., 2022).

Çoklu hedefli optimizasyon problemleri, DE algoritmasının gelecekteki araştırma yönlerinde önemli bir yer tutmaktadır. Çoklu hedefli optimizasyon, birden fazla amaç fonksiyonunu aynı anda optimize etmeyi gerektirir ve bu tür problemler, genellikle birbirleriyle çelişen hedefler içerir. DE algoritmasının bu tür problemler için uyarlanması, Pareto optimal çözümler kümesinin bulunmasını sağlayabilir. Çoklu hedefli diferansiyel evrim (MODE) algoritmaları, bu amaçla geliştirilmiş ve başarılı sonuçlar elde etmiştir (Wang et.al, 2017). MODE algoritmaları, farklı hedefler arasındaki dengeyi sağlayarak, optimal çözümler kümesini daha etkili bir şekilde keşfetmeyi mümkün kılar. Ayrıca, DE algoritmasının çoklu hedefli optimizasyon problemlerinde uygulanabilirliğini artırmak için yeni ve yenilikçi yöntemlerin geliştirilmesi, gelecekteki araştırmalarda önemli bir rol oynayacaktır (Chakraborty, 2008).

Diferansiyel evrim algoritmasının gelecekteki araştırma yönleri, sadece mevcut uygulama alanlarını genişletmekle kalmaz, aynı zamanda algoritmanın temel yapısını iyileştirerek daha sofistike ve karmaşık problemleri çözme kapasitesini artırır. Algoritmanın performansını artırmak için önerilen iyileştirmeler, DE algoritmasının daha geniş bir optimizasyon problemleri yelpazesinde etkili bir şekilde kullanılmasını sağlayacaktır. Ayrıca, DE algoritmasının farklı alanlarda entegrasyonu ve adaptasyonu, yeni teknolojik gelişmelere uyum sağlama kapasitesini de gösterecektir (Mininno et al., 2010). Sonuç olarak, diferansiyel evrim algoritmasının gelecekteki araştırma yönleri, yeni uygulama alanları, algoritmik iyileştirmeler ve çoklu hedefli optimizasyon problemleri gibi konulara odaklanmaktadır. Bu araştırma yönleri, DE algoritmasının performansını ve uygulama potansiyelini artırarak, optimizasyon alanında daha geniş bir kullanım yelpazesine sahip olmasını sağlayacaktır. Diferansiyel evrim algoritmasının esnekliği ve geniş uygulama yelpazesi, onu gelecekteki optimizasyon problemlerinin çözümünde önemli bir araç haline getirecektir (Padhye et al., 2013). Algoritmanın sürekli olarak geliştirilmesi ve yeni alanlara uygulanması, DE algoritmasının optimizasyon literatüründe ve pratik uygulamalarda önemli bir yer edinmesini sağlamaya devam edecektir.

IX. Sonuç

Diferansiyel Evrim (DE) algoritması, evrimsel hesaplama yöntemleri arasında basitliği, esnekliği ve güçlü performansıyla dikkat çeken bir optimizasyon tekniğidir. Bu çalışma, DE algoritmasının temel prensiplerini, mutasyon, çaprazlama ve seçilim gibi genetik operatörlerin etkin kullanımı üzerinden detaylandırmıştır. Ayrıca, algoritmanın çeşitli yazılım mühendisliği problemlerine olan uygulanabilirliğini incelemiştir.

Araştırmamız, DE algoritmasının yazılım geliştirme süreçlerinde test optimizasyonu, performans ayarlaması, yazılım hata ayıklama ve bileşen optimizasyonu gibi kritik görevlerde başarılı sonuçlar verdiğini göstermektedir. Algoritmanın düşük parametre gereksinimi ve bu parametrelerin kolayca ayarlanabilmesi, yüksek boyutlu ve karmaşık problemlerde dahi üstün performans sergilemesini sağlamaktadır.

DE algoritmasının diferansiyel mutasyon mekanizması, çözüm uzayında daha yönlendirilmiş ve etkin bir arama süreci sağlayarak global optimuma ulaşma olasılığını artırır. Bu çalışma, DE algoritmasının geniş bir yelpazede optimizasyon problemlerinde etkili bir çözüm aracı olduğunu ve bu alandaki potansiyelini ortaya koymuştur.

Sonuç olarak, Diferansiyel Evrim algoritması, hem akademik araştırmalarda hem de endüstriyel uygulamalarda geniş bir kullanım alanına sahip olup, optimizasyon problemlerinin çözümünde önemli bir rol oynamaktadır. Gelecekte, DE algoritmasının performansını daha da artıracak yenilikçi yöntemlerin ve uygulamaların geliştirilmesi, bu alandaki araştırmalara değerli katkılar sağlayacaktır.

References

[1] Pétrowski, A., Ben-Hamida, S. (2017). Evolutionary algorithms. John Wiley Sons.

[2] Price, K. V., Storn, R. M., Lampinen, J. A. (2005). The differential evolution algorithm. Differential evolution: a practical approach to global optimization, 37-134.

[3] Plagianakos, V. P., Tasoulis, D. K., Vrahatis, M. N. (2008). A review of major application areas of differential evolution. Advances in differential evolution, 197-238.

[4] Amiri, M., Barbin, J. P. (2015). New approach for solving software project scheduling problem using differential evolution algorithm. International journal in foundations of computer science technology (IJFCST), 5(1), 1-9.

[5] Karimi, A., Gandomani, T. J. (2021). Software development effort estimation modeling using a combination of fuzzy-neural network and differential evolution algorithm. International Journal of Electrical and Computer Engineering, 11(1), 707.

[6] Yang, X. S. (2010). Nature-inspired metaheuristic algorithms. Luniver press.

[7] Bozorg-Haddad, O., Solgi, M., Loáiciga, H. A. (2017). Meta-heuristic and evolutionary algorithms for engineering optimization. John Wiley Sons.

[8] Malik, H., Iqbal, A., Joshi, P., Agrawal, S., Bakhsh, F. I. (Eds.). (2021). Metaheuristic and evolutionary computation: algorithms and applications (Vol. 916). Berlin/Heidelberg, Germany: Springer.

[9] Burke, E. K., Kendall, G. (2014). Search methodologies: introductory tutorials in optimization and decision support techniques. Springer.

[10] Poli, R., Koza, J. (2013). Genetic programming. In Search methodologies: Introductory tutorials in optimization and decision support techniques (pp. 143-185). Boston, MA: Springer US.

[11] Mayer, D. G., Kinghorn, B. P., Archer, A. A. (2005). Differential evolution–an easy and efficient evolutionary algorithm for model optimisation. Agricultural Systems, 83(3), 315-328.

[12] Qing, A. (2009). Differential evolution: fundamentals and applications in electrical engineering. John Wiley Sons.

[13] Feoktistov, V. (2006). Differential evolution (pp. 1-24). Springer US.

[14] Zhu, Z., Chen, L., Xia, C., Yuan, C. (2018). A history-driven differential evolution algorithm for optimization in dynamic environments. International Journal on Artificial Intelligence Tools, 27(06), 1850028.

[15] Wong, K. P., Dong, Z. Y. (2005, November). Differential evolution, an alternative approach to evolutionary algorithm. In Proceedings of the 13th International Conference on, Intelligent Systems Application to Power Systems (pp. 73-83). IEEE.

[16] Vikhar, P. A. (2016, December). Evolutionary algorithms: A critical review and its future prospects. In 2016 International conference on global trends in signal processing, information computing and communication (ICGTSPICC) (pp. 261-265). IEEE.

[17] Karaboğa, D., Ökdem, S. (2004). A simple and global optimization algorithm for engineering problems: differential evolution algorithm. Turkish Journal of Electrical Engineering and Computer Sciences, 12(1), 53-60.

[18] Hegerty, B., Hung, C. C., Kasprak, K. (2009, November). A comparative study on differential evolution and genetic algorithms for some combinatorial problems. In Proceedings of 8th Mexican international conference on artificial intelligence (Vol. 9, p. 13).

[19] Tanabe, R., Fukunaga, A. (2013, June). Success-history based parameter adaptation for differential evolution. In 2013 IEEE congress on evolutionary computation (pp. 71-78). IEEE.

[20] Gong, W., Cai, Z., Zhu, L. (2009). An efficient multiobjective differential evolution algorithm for engineering design. Structural and Multidisciplinary Optimization, 38, 137-157.

[21] Pant, M., Zaheer, H., Garcia-Hernandez, L., Abraham, A. (2020). Differential Evolution: A review of more than two decades of research. Engineering Applications of Artificial Intelligence, 90, 103479.

[22] Kachitvichyanukul, V. (2012). Comparison of three evolutionary algorithms: GA, PSO, and DE. Industrial Engineering and Management Systems, 11(3), 215-223.

[23] Kramer, O. (2017). Genetic algorithms (pp. 11-19). Springer International Publishing.

[24] Tušar, T., Filipič, B. (2007, March). Differential evolution versus genetic algorithms in multiobjective optimization. In International Conference on Evolutionary Multi-Criterion Optimization (pp. 257-271). Berlin, Heidelberg: Springer Berlin Heidelberg.

[25] Yu, X., Gen, M. (2010). Introduction to evolutionary algorithms. Springer Science Business Media.

[26] Hrstka, O., Kučerová, A., Lepš, M., Zeman, J. (2003). A competitive comparison of different types of evolutionary algorithms. Computers Structures, 81(18-19), 1979-1990.

[27] Zeng, Z., Zhang, M., Chen, T., Hong, Z. (2021). A new selection operator for differential evolution algorithm. Knowledge-Based Systems, 226, 107150.

[28] Moraglio, A., Krawiec, K., Johnson, C. G. (2012). Geometric semantic genetic programming. In Parallel Problem Solving from Nature-PPSN XII: 12th International Conference, Taormina, Italy, September 1-5, 2012, Proceedings, Part I 12 (pp. 21-31). Springer Berlin Heidelberg.

[29] Zaharie, D. (2009). Influence of crossover on the behavior of differential evolution algorithms. Applied soft computing, 9(3), 1126-1138.

[30] Yang, Z., Yao, X., He, J. (2008). Making a difference to differential evolution. Advances in metaheuristics for hard optimization, 397-414.

[31] Chakraborty, U. K. (Ed.). (2008). Advances in differential evolution (Vol. 143). Springer Science Business Media.

[32] Liao, T. W. (2010). Two hybrid differential evolution algorithms for engineering design optimization. Applied Soft Computing, 10(4), 1188-1199.

[33] Gong, W., Cai, Z., Liang, D. (2014). Engineering optimization by means of an improved constrained differential evolution. Computer Methods in Applied Mechanics and Engineering, 268, 884-904.

[34] Masike, R., Kaur, K., Arora, R., Shridhara, S. N. (2024). Optimization of Holding Force for a Climbing Robot Based on a Differential Evolutionary Algorithm. Recent Advances in Electrical Electronic Engineering (Formerly Recent Patents on Electrical Electronic Engineering), 17(1), 54-59.

[35] Tiwari, N., Srivastava, L. (2016, March). Generation scheduling and micro-grid energy management using differential evolution algorithm. In 2016 International Conference on Circuit, Power and Computing Technologies (ICCPCT) (pp. 1-7). IEEE.

[36] Karaseva, T. S. (2022). Hybridization of Machine Learning Models and Differential Evolution in Data Mining. European Proceedings of Computers and Technology.

[37] Da Silva, A. J., Mineu, N. L., Ludermir, T. B. (2010). Evolving artificial neural networks using adaptive differential evolution. In Advances in Artificial Intelligence–IBERAMIA 2010: 12th Ibero-American Conference on AI, Bahía Blanca, Argentina, November 1-5, 2010. Proceedings 12 (pp. 396-405). Springer Berlin Heidelberg.

[38] Zaheer, H., Pant, M. (2017). Solution to industrial optimization problems through differential evolution variants. Materials and Manufacturing Processes, 32(10), 1131-1143.

[39] Rajesh, C., Kumar, S. (2022). An evolutionary block based network for medical image denoising using Differential Evolution. Applied Soft Computing, 121, 108776.

[40] Salihov, S., Maltsov, D., Samsonova, M., Kozlov, K. (2021). Solution of mixed-integer optimization problems in bioinformatics with differential evolution method. Mathematics, 9(24), 3329.

[41] Suganthan, P. N. (2012). Differential evolution algorithm: recent advances. In Theory and Practice of Natural Computing: First International Conference, TPNC 2012, Tarragona, Spain, October 2-4, 2012. Proceedings 1 (pp. 30-46). Springer Berlin Heidelberg.

[42] Singal, P., Kumari, A. C., Sharma, P. (2020). Estimation of software development effort: A Differential Evolution Approach. Procedia Computer Science, 167, 2643-2652.

[43] Benala, T. R., Mall, R. (2018). DABE: Differential evolution in analogy-based software development effort estimation. Swarm and Evolutionary Computation, 38, 158-172.

[44] Khatibi Bardsiri, A., Hashemi, S. M. (2016). A differential evolution-based model to estimate the software services development effort. Journal of Software: Evolution and Process, 28(1), 57-77.

[45] Becerra, R. L., Sagarna, R., Yao, X. (2009, May). An evaluation of differential evolution in software test data generation. In 2009 IEEE Congress on Evolutionary Computation (pp. 2850-2857). IEEE.

[46] Yang, X., Yu, H., Fan, G., Yang, K. (2021). DEJIT: a differential evolution algorithm for effort-aware just-in-time software defect prediction. International Journal of Software Engineering and Knowledge Engineering, 31(03), 289-310.

[47] Kumari, A. C., Srinivas, K., Gupta, M. P. (2012, December). Multi-objective test suite minimisation using quantum-inspired multi-objective differential evolution algorithm. In 2012 IEEE International Conference on Computational Intelligence and Computing Research (pp. 1-7). IEEE.

[48] Biju, A. C., Victoire, T. A. A., Mohanasundaram, K. (2015). An improved differential evolution solution for software project scheduling problem. The Scientific World Journal, 2015.

[49] Yaghoobi, T. (2020). Parameter optimization of software reliability models using improved differential evolution algorithm. Mathematics and Computers in Simulation, 177, 46-62.

[50] Das, S., Mullick, S. S., Suganthan, P. N. (2016). Recent advances in differential evolution–an updated survey. Swarm and evolutionary computation, 27, 1-30.

[51] Zhan, Z. H., Wang, Z. J., Jin, H., Zhang, J. (2019). Adaptive distributed differential evolution. IEEE transactions on cybernetics, 50(11), 4633-4647.

[52] Deng, W., Liu, H., Xu, J., Zhao, H., Song, Y. (2020). An improved quantum-inspired differential evolution algorithm for deep belief network. IEEE Transactions on Instrumentation and Measurement, 69(10), 7319-7327.

[53] Ahmad, M. F., Isa, N. A. M., Lim, W. H., Ang, K. M. (2022). Differential evolution: A recent review based on state-of-the-art works. Alexandria Engineering Journal, 61(5), 3831-3872.

[54] Wang, S., Li, Y., Yang, H. (2017). Self-adaptive differential evolution algorithm with improved mutation mode. Applied Intelligence, 47, 644-658.

[55] Salihov, S., Maltsov, D., Samsonova, M., Kozlov, K. (2021). Solution of mixed-integer optimization problems in bioinformatics with differential evolution method. Mathematics, 9(24), 3329.