**YZM 4032**

**Meta-Sezgisel Optimizasyon Dersi**

**Dönem Raporu**

**Algoritma Adı:** Dynamic differential annealed optimization

**Öğrenci Adı ve Soyadı:** Elif Şahinkaya

**Öğrenci No:**334341

**Öğrenci İletişim Bilgileri**

**Cep tel: 0**5537714333

**\*e-mail adresi:** elifsahinkaya@outlook.com

**Dinamik Diferansiyel Tavlı Optimizasyon Algoritmasının FDB Yöntemi ile İyileştirilmesi: yeni metasezgisel optimizasyon mühendislik uygulama algoritmasının geliştirilmesi**

**ÖZET**

Bu makale çalışmasında son zamanlarda geliştirilmiş güncel bir meta-sezgisel arama (MSA) yöntemi olan dinamik diferansiyel tavlı (dynamic differential annealed Optimization, DDAO) algoritmasının iyileştirilmiş bir versiyonu geliştirilmektedir. DDAO’da minimum veya maksimumun gerekli olduğu matematiksel optimizasyon problemlerinin süreçlerini daha etkili bir şekilde modelleyebilmek amacıyla arama sürecine rehberlik eden çözüm adayları uzaklık-uygunluk dengesi (fitness distance balance, FDB) yöntemi kullanılarak belirlenmiştir. Geliştirilen FDB-tabanlı DDAO algoritmasının performansını test etmek ve doğrulamak amacıyla güncel bir karşılaştırma problemleri havuzu olan CEC 2020 kullanılmıştır. Önerilen algoritmanın farklı tiplerdeki ve farklı boyutlardaki arama uzaylarındaki performansını test etmek ve doğrulamak için test problemleri 30/50/100 boyutta tasarlanmıştır. Her deneysel çalışmadan elde edilen veriler parametrik olmayan istatistiksel test yöntemleri kullanılarak analiz edilmiştir. Analiz sonuçlarına göre ve farklı boyutlarda olmak üzere tüm deneysel çalışmalarda FDB-DDAO varyasyonlarının baz algoritmaya kıyasla bazıları üstün bir performans sergilemişlerdir. Önerilen FDB-DDAO algoritması baz algoritmadan daha iyi sonuçlara sahiptir.

# GİRİŞ

Bu yaklaşım geniş bir yelpazeyi küresel minimum veya maksimumun gerekli olduğu matematiksel optimizasyon problemleri çözmek için kullanılabilecek yeni bir optimizasyon algoritması önermektedir.Yeni algoritma, rastgele aramaya ve klasik benzetilmiş tavlama algoritmasına dayanmaktadır (Modern yüksek kaliteli çelik üretim süreci) ve dinamik diferansiyel tavlı optimizasyon (DDAO) olarak belirlenmiştir.Dinamik diferansiyel tavlı optimizasyon algoritması, çok sayıda yüksek alıntı yapılan optimizasyon algoritmaları ile karşılaştırılmıştır.Sayısal testlere göre, DDAO, bu algoritmalardan bazılarından daha iyi performans göstermiştir. Birçok durumda yüksek performans göstermiştir.Kısıtlı yol planlaması ve yay tasarımı problemi pratik bir mühendislik optimizasyon problemi olarak seçilmiştir. DDAO, küresel minimum olan sorunlarına ve yay tasarım problemi için birçok algoritmadan daha verimli çözümler üretmiştir.

Mekanik iyileştirme için birçok çalışma yapılmıştır.Çeliğin özellikleri gerekli kütleyi azaltmak için üretim süreci için maliyeti etkiler.Bu çalışmalar, yüksek mukavemetli çeliklerin geliştirilmesi ile ilgilidir,gelişmiş yüksek olarak belirlenen önemli oluşum oranlarıyla dayanıklı çelikler . Çift fazlı çelik bir tür gelişmiş benzersiz bir mikro yapıya sahip yüksek mukavemetli çelikler süper mekanik özelliklerdir. Genel olarak, yüksek bir sıcaklıkta demir erir sıvı hale ve sürekli soğutma ile katı hale oda sıcaklığında duruma gelir. Metalin özellikleri bağlı olduğu katı haldeki mevcut fazlar hakkında: örneğin, ferritik faz esneklik verir, martensit fazı sertlikten sorumludur ve bainit ve östenit gibi başka aşamalar da vardır. Genellikle, birçok demir fazı aynı katıda bir arada bulunur ve nihai mekanik özelliğe bir istenen kombinasyonu verirler. Çift fazlı çelik (DP) üretmek için bir parça demir mikro yapının yaklaşık olarak homojen olduğu yeniden kristalleşme noktasının üzerinde yüksek bir sıcaklığa ısıtılır.Bir süre sonra demir parçası daha ince bir kalınlıkta haddelenmelidir ve haddeleme sırasında metal soğuyacaktır. Sonra mekanik deformasyon uygulanarak metal müteakip diferansiyel soğutmaya tabi tutulur; hızlandırılmış soğutma, hava soğutma, başka bir hızlandırılmış soğutma ve son olarak oda sıcaklığına yavaş soğutma. Bu süreç, bir mikro yapıya sahip çeliğe yol açar. Adalar şeklinde sert martensitik faz içeren ferritik matris. Yumuşak ferrit ve sertin birleşik etkilerimartenzit, çift fazlı çeliğe kabul edilebilir bir oluşum hızı ile yüksek mukavemet verir ,bu özellikle otomotiv endüstrisinde çok önemlidir. Daha önce açıklanan sistematik prosedür, bir yüksek kaliteli çelik tipi (çift fazlı) net bir optimization. Metal, düşük kaliteden yüksek kaliteye doğru geliştirilir. Bu davranış, dinamik farklılığa ilham verdi. Bu çalışmada geliştirilen tavlı optimizasyon algoritmasıdır.

# YÖNTEM

MSA algoritmaları optimizasyon sürecinde esas olarak iki gereksinimi yerine getirmektedirler. Bunlar; arama uzayındaki referans bir konumun yakın komşuluğunda hassas bir şekilde araştırma yapabilmek ve ihtiyaç duyulduğunda arama uzayındaki umut vadeden konumları etkili bir şekilde keşfedebilmektir. MSA algoritmaları bu gereksinimleri yerine getirirken benzer adımlardan oluşan bir arama süreci yaşam döngüsünü takip ederler. Buna göre MSA algoritmalarında arama sürecinin genel adımları Algoritma 1’de verilmektedir.

|  |
| --- |
| Algoritma 1. MSA algoritmalarında arama sürecinin genel adımları [22] |
| 1. Optimizasyon probleminin tanımlanması (amaç fonksiyonu, tasarım parametreleri vd. öğeler) 2. Algoritma parametrelerinin tanımlanması ve çözüm adayları topluluğunun oluşturulması 3. Aday çözümlerin uygunluk değerlerinin hesaplanması 4. Arama Süreci Yaşam Döngüsü    1. Seçim süreci    2. Komşuluk araması ve çeşitliliğin sağlanması    3. Çözüm adayı topluluğunun güncellenmesi 5. Sonlandırma kriteri sağlandı mı?    1. Hayır (Adım 4’e dön)    2. Evet (Arama sürecini sonlandır ve en iyi çözüm adayını kaydet) |

Algoritma 1’de verilen adımlara göre MSA algoritmalarında ilk üç adım ortaktır. Algoritmaların performansları arasındaki farklılıklar ise dördüncü adımdaki işlemlere bağlıdır. Dördüncü adımda her algoritmanın kendine özgü arama süreci yaşam döngüsü işletilmektedir. Arama süreci yaşam döngüsünün ilk adımında arama sürecine rehberlik edecek olan adaylar seçilir. Seçim süreci popülasyon üyeleri arasından rastgele,aç gözlü,olasılıksal yöntemlerden biri kullanılarak gerçekleştirilebilir. Rastgele seçim yöntemi arama sürecinde çeşitliliğe katkı sunmak amacıyla kullanılır. Ancak özellikle büyük boyutlu ve karmaşıklık düzeyi yüksek olan arama uzaylarında etkisi düşüktür.

## DDAO Algoritması

**1.** Çeliğin kütlesi başlangıçta molekül gruplarından oluşur iyileştirilmesi gereken (aday çözümlerin nüfusu) bir ferrit ve martensit karışımı (küresel çözüm) olmalıdır.

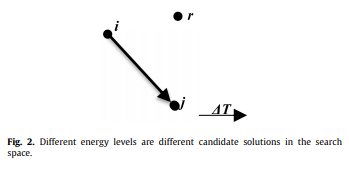
**2**. Çift fazlı çelik üretiminde sıcaklık düşer ve sıcaklıktaki her düşüş için farklı bir çelik fazı oluşturma şansı vardır. Bu, küresel bir çözüm ararken matematiksel optimizasyondaki yineleme sürecine eşdeğerdir.

**3.** Metaldeki her fazın kendi iç enerjisi vardır ve bu matematiksel optimizasyonun amaç fonksiyonunun değerine eşdeğer olur.

**4.** Şekil 1 farklı soğutma hızlarını, havayla soğutmayı, hızlandırılmış soğutmayı ve yavaş soğutmayı gösterir; soğutma işlemini temsil etmek için aşağıdaki denklemi önerilir:

 (1)

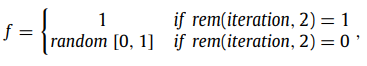
burada S k yineleme numarası ( k ) için önerilen yeni bir çözümdür , k = 1 ... n burada n yineleme sayısıdır ve Sc i ve Sc j , rastgele seçilen popülasyondan rastgele seçilen çözümlerdir. ( i ) ve ( j ) endeksleri içinde , Sr rastgele oluşturulmuş bir çözümdür. Daha açıklamak gerekirse, şek. 2, sıcaklık değiştiğinde I noktasından j noktasına √ T ile varır.

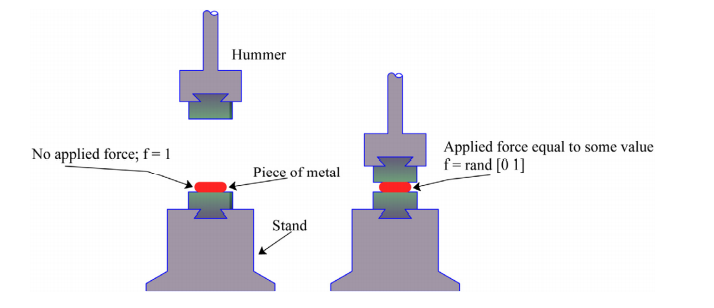


Her birinin enerji seviyesi, nesnel değere sahip bir çözüme eşdeğerdir başka bir deyişle, iki rastgele enerji seviyesi arasındaki fark bir çözüm de vardır. Ayrıca, bu farkı rastgele ekleyerek uzayda enerji, şek. 2, yeni bir değer dönecektir farklı enerjiye sahip bir çözüm olan enerji (nesnel değer).

Bu ekolayzır için hayali bir temeldir. (1), omurgası olan önerilen algoritmanın arama motoru için ve ana yakınsama için optimizasyon problemleri sorumludur. Buna değer, Ekolayzırdan bahsetmişken. (1) ve tamamlayıcı denklemi (3) diferansiyel evrimde belirtilen mutasyon sürecine benzer , ancak bizim durumumuzda evrim yoktur, random vardır seçim ve hatta SR popülasyondan rastgele seçilir başka bir alt popülasyondan, daha sonra açıklayacağımız gibi.

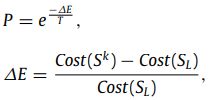
**5.** Isının diferansiyel indirgenmesi sırasında metal yuvarlanır ve bu mekanik işlem matematiksel olarak modellenmelidir. Programlama nedenleriyle, metalin haddeleme yerine dövme işlemine tabi tutulduğunu varsaymalıyız. Dövme sırasında çekicin dinamik davranışı, 1 ile rastgele bir sayı arasında dalgalanan bir parametre olarak temsil edilebilir.

(2)

burada f, dövme parametresidir ve rem, 2'ye bölme işleminden sonra kalandır. Yinelemeler sırasında, denklem (2), geçerli yineleme sayısı tek olduğunda, yani 1, 3, 5..vb. f bir olacak, mevcut iterasyon sayısı çift ise (2,4,6…vb.) f sıfır ile bir arasında rastgele bir sayı olacaktır. Bu, hummer tarafından uygulanan kuvvetin değişiklik olmaması ile rastgele bir değer arasında dalgalandığı Şekil 2’de gösterilen gerçek hayatta dövme ile tutarlıdır. 

Dövme, diferansiyel soğutma ile çevrim içi olarak yapıldığından, denklem (1) şu şekilde değiştirilir:(3)

**6.** Gerçek tavlama işleminde, düşük sıcaklıklarda olduğundan daha yüksek sıcaklıklarda yeni fazların oluşumunu kabul etme olasılığı daha yüksektir. Optimizasyon sürecinde, açıklanan olasılık formülüne bağlı olarak aynı prosedürü kullanıyoruz. SA algoritmasıyla:

(4) (5) 

P, yeni bir çözümü kabul etme olasılığı olduğunda, ΔE denklem (3)'ten önerilen çözümün amaç değeri ile popülasyondaki L indeksinin bir çözümü olan SL çözümünün amaç değeri arasındaki farktır, L =1,…, popülasyon büyüklüğü. T, yüksek değerle başlaması ve yinelemeler sırasında sürekli olarak daha düşük bir değere güncellenmesi gereken sıcaklık değişkenidir.

Önerilen çözüm, P > rasgele ∈ [0,1] ise kabul edilebilir. Aramanın başında T, yüksek değerle başlar; sonuç olarak, denklem (4)'e göre P bire yakın olacaktır. Bu, geniş bir rasgele sayı aralığının birden az olabileceği ve çözümün seçileceği anlamına gelir. T'nin düşük değerinde, P olasılığı sıfıra yakın olacaktır; denklem (4)'e göre,bu, çok dar bir rastgele sayı aralığının P'den daha az olabileceği ve çözümün seçilme olasılığının daha düşük olduğu anlamına gelir. Örneğin, ,  iken, bu, sıcaklık düştükçe yeni bir çözüm seçme olasılığını azaltmak için basit bir mekanizmadır.

**7.** İşlem 4. adımdan itibaren tekrarlanır ve her iterasyon için en iyi çözüm saklanır.

Ana arama motoru rastgele arama iken, dinamik tavlama işlemi arama üzerinde bir düzeltmedir; bu yüzden dinamik diferansiyel tavlanmış optimizasyon olarak adlandırılmıştır. Dinamik diferansiyel uygulamak için bir MATLAB platformu kullanılmıştır.

DDAO algoritmasının sözde kodu:

***X i popülasyonunu başlat (i=1,2,…,n)***

***T parametresini başlat, soğutma hızı***

***Her çözümün maliyetini hesaplayın***

***X b = En iyi çözüm***

***while (t<Max yineleme)***

***Alt popülasyon S'yi başlat***

***Alt popülasyonun maliyetini hesaplayın***

***Alt popülasyonu sırala***

***S r = Alt popülasyondaki en iyi çözüm***

***Popülasyondan iki rastgele çözüm X m ve X n seçin***

***(3) denkleminden S k hesaplayın***

***Popülasyon X'i sırala***

***X popülasyonunda foreach çözümü***

***bir gelişme varsa***

***X ben = Sk***

***aksi takdirde, denklem (4) ve (5)'i kullanarak X popülasyonundaki en kötü çözümü değiştirin***

***endif***

***son***

***X b'yi güncelle***

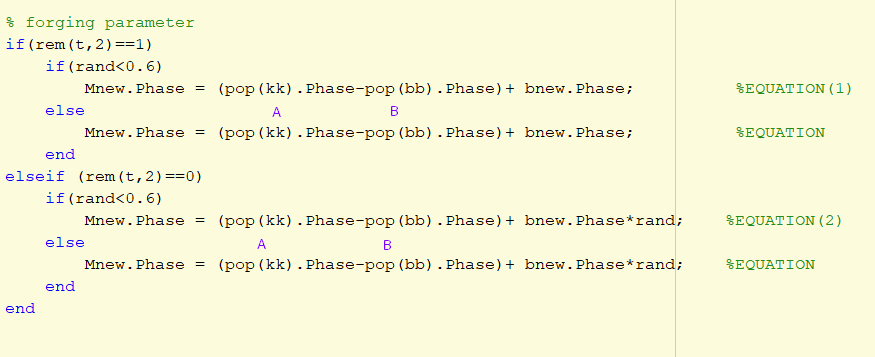
***T=T\*soğutma hızı***

***t=t+1***

***son zaman***

***dönüş X b***

**DDAO-FDB Uygulanması**

****

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Equation** | **Uygulanan yer** |  |  |  |  |
| 1 | A | CASE1 | CASE2\_1 | CASE3\_1 |  |
| 1 | B | CASE2 | CASE2\_2 | CASE3\_2 |  |
| 2 | A | CASE3 | CASE2\_3 | CASE3\_3 |  |
| 2 | B | CASE4 | CASE2\_4 | CASE3\_4 |  |
| 1 | A,B | CASE5 | CASE2\_5 | CASE3\_5 |  |
| 1,2 | A,A | CASE6 | CASE2\_6 | CASE3\_6 |  |
| 1,2 | A,B | CASE7 | CASE2\_7 | CASE3\_7 |  |
| 1,2 | B,A | CASE8 | CASE2\_8 | CASE3\_8 |  |
| 1,2 | B,B | CASE9 | CASE2\_9 | CASE3\_9 |  |
| 2 | A,B | CASE10 | CASE2\_10 | CASE3\_10 |  |
| 1,1,2 | A,B,A | CASE11 | CASE2\_11 | CASE3\_11 |  |
| 1,1,2 | A,B,B | CASE12 | CASE2\_12 | CASE3\_12 |  |
| 1,2,2 | A,A,B | CASE13 | CASE2\_13 | CASE3\_13 |  |
| 1,2,2 | B,A,B | CASE14 | CASE2\_14 | CASE3\_14 |  |
| 1,2,1,2 | A,B,A,B | CASE15 | CASE2\_15 | CASE3\_15 |  |
|  |  | 0.6 | 0.3 | 0.9 | Oranlar |

# DENEYSEL ÇALIŞMA AYARLARI

* Algoritmalar arasında fırsat eşitliğini sağlamak için, amaç fonksiyonunu azami değerlendirme sayısı üzerinden sonlandırma kriteri tanımlanmıştır. Bu değer 10.000\*d (d:problem boyutu) dir.
* Önerilen yöntemin düşük, orta ve yüksek boyutlu arama alanlarındaki performansını ortaya çıkarmak için dinamik olarak boyutlandırılabilen CEC2020 karşılaştırma fonksiyonları kullanılmıştır.
* Deneysel çalışmalar MATLAB®R2016B'da, Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz ve 16 GB RAM ve x64 tabanlı işlemci üzerinde gerçekleştirildi.

# Analiz Sonuçları

Bu bölümde deneysel çalışmalardan elde edilen veriler kullanılarak DDAO’nun ve FDB-DDAO varyasyonlarının karşılaştırmalı olarak performansları analiz edilmektedir. Algoritmaların birbirlerine göre performanslarını karşılaştırmak ve onları sıralamak için Friedman Testi, SDO algoritmasını baz modeli ile FDB-DDAO varyasyonlarının her birini ayrı-ayrı ikili olarak karşılaştırmak için Wilcoxon testi uygulanmıştır. Buna göre analiz sonuçları takip eden alt bölümlerde sırasıyla sunulmaktadır.

## İstatistiksel Analiz Sonuçları

Aşağıdaki tabloda bu makalede geliştirilen farklı FDB-DDAO varyasyonunun ve DDAO algoritmasının baz modelinin Friedman test yöntemine göre sıralamaları verilmektedir.Sıralamalar CEC 2020 karşılaştırma havuzundaki problemlerin 30/50/100 boyutları için gerçekleştirilen çalışmadan elde edilen veriler kullanılarak elde edilmiştir. Sonradan aldığımız sonuçlarda daha iyi olan caselerden 3 adet CASE3\_16 ,CASE3\_17 ,CASE3\_18 isimli caseleride ekleyerek daha iyi sonuçlar elde etmeye çalıştık.Sonuç olarak en iyi ilk 3 case’in arasında CASE3\_18’de yer aldı.

Friedman analiz yöntemine göre:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Algoritmalar | Boyut = 30 | Boyut = 50 | Boyut = 100 | Ortalama |
| CASE1 | 24,536 | 25,24 | 25,704 | 25,16 |
| CASE2 | 25,268 | 25,066 | 24,176 | 24,83667 |
| CASE3 | 26,356 | 25,352 | 26,132 | 25,94667 |
| CASE4 | 25,616 | 26,312 | 26,206 | 26,04467 |
| CASE5 | 25,836 | 26,664 | 27,216 | 26,572 |
| CASE6 | 24,916 | 24,306 | 26,844 | 25,35533 |
| CASE7 | 25,874 | 24,218 | 26,136 | 25,40933 |
| CASE8 | 25,196 | 23,508 | 26,108 | 24,93733 |
| CASE9 | 24,976 | 25,296 | 25,886 | 25,386 |
| CASE10 | 24,34 | 24,26 | 22,904 | 23,83467 |
| CASE11 | 26,426 | 24,776 | 25,13 | 25,444 |
| CASE12 | 25,356 | 26,462 | 24,518 | 25,44533 |
| CASE13 | 24,236 | 22,968 | 23,796 | 23,66667 |
| CASE14 | 23,368 | 24,258 | 24,506 | 24,044 |
| CASE15 | 25,42 | 25,768 | 24,066 | 25,08467 |
| CASE2\_1 | 25,24 | 26,36 | 25,208 | 25,60267 |
| CASE2\_2 | 24,342 | 24,26 | 25,328 | 24,64333 |
| CASE2\_3 | 24,644 | 24,484 | 26,228 | 25,11867 |
| CASE2\_4 | 24,044 | 25,926 | 25,448 | 25,13933 |
| CASE2\_5 | 23,952 | 26,084 | 25,536 | 25,19067 |
| CASE2\_6 | 26,282 | 24,804 | 25,89 | 25,65867 |
| CASE2\_7 | 25,964 | 25,854 | 24,564 | 25,46067 |
| CASE2\_8 | 27,244 | 25,38 | 25,78 | 26,13467 |
| CASE2\_9 | 25,196 | 24,088 | 27,48 | 25,588 |
| CASE2\_10 | 23,34 | 24,432 | 24,568 | 24,11333 |
| CASE2\_11 | 25,556 | 26,396 | 24,254 | 25,402 |
| CASE2\_12 | 25,56 | 24,698 | 24,95 | 25,06933 |
| CASE2\_13 | 23,076 | 25 | 25,148 | 24,408 |
| CASE2\_14 | 24,512 | 23,684 | 26,26 | 24,81867 |
| CASE2\_15 | 24,324 | 24,396 | 24,892 | 24,53733 |
| CASE3\_1 | 26,808 | 25,372 | 25,632 | 25,93733 |
| CASE3\_2 | 26,05 | 24,376 | 23,662 | 24,696 |
| CASE3\_3 | 24,89 | 24,524 | 24,768 | 24,72733 |
| CASE3\_4 | 26,91 | 24,044 | 24,188 | 25,04733 |
| CASE3\_5 | 27,468 | 27,22 | 22,918 | 25,86867 |
| CASE3\_6 | 24,46 | 22,736 | 25,456 | 24,21733 |
| CASE3\_7 | 24,212 | 25,96 | 26,608 | 25,59333 |
| CASE3\_8 | 25,296 | 24,436 | 24,752 | 24,828 |
| CASE3\_9 | 25,964 | 26,18 | 24,59 | 25,578 |
| CASE3\_10 | 22,556 | 23,596 | 23,07 | 23,074 |
| CASE3\_11 | 27,24 | 26,848 | 26,162 | 26,75 |
| CASE3\_12 | 27,946 | 27,48 | 24,784 | 26,73667 |
| CASE3\_13 | 24,002 | 24,952 | 24,1 | 24,35133 |
| CASE3\_14 | 22,776 | 23,352 | 22,684 | 22,93733 |
| CASE3\_15 | 23,572 | 24,384 | 25,176 | 24,37733 |
| CASE3\_16 | 23,308 | 23,88 | 22,944 | 23,37733 |
| CASE3\_17 | 22,826 | 24,324 | 23,696 | 23,61533 |
| CASE3\_18 | 21,936 | 25,82 | 24,328 | 24,028 |
| DDAO | 25,784 | 25,216 | 24,62 | 25,20667 |

deneylerin tümünde FDB-DDAO varyasyonlarının 21 tanesinde DDAO algoritmasını geride bırakmışlardır. Bu sonuç, FDB tabanlı tasarımsal değişikliğin DDAO’nun doğal sürecine daha uygun bir yapıya kavuşmasını sağladığının güçlü bir işaretidir.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Algoritmalar | Boyut = 30 | Boyut = 50 | Boyut = 100 | Ortalama Sıra |
| Case2 | 2,404 | 2,416 | 2,444 | 2,421333 |
| Case3 | 2,472 | 2,412 | 2,476 | 2,453333 |
| Case1 | 2,38 | 2,6 | 2,604 | 2,528 |
| DDAO | 2,744 | 2,572 | 2,476 | 2,597333 |

Wilcoxon analiz yöntemine göre:

|  |  |  |  |
| --- | --- | --- | --- |
| **vs. DDAO +/=/-** | **Boyut = 30** | **Boyut = 50** | **Boyut = 100** |
| CASE1 | 0\_10\_0 | 0\_10\_0 | 0\_9\_1 |
| CASE2 | 0\_10\_0 | 0\_10\_0 | 0\_10\_0 |
| CASE3 | 0\_9\_1 | 0\_10\_0 | 0\_9\_1 |
| CASE4 | 1\_9\_0 | 0\_10\_0 | 0\_9\_1 |
| CASE5 | 1\_8\_1 | 1\_8\_1 | 0\_9\_1 |
| CASE6 | 1\_8\_1 | 0\_10\_0 | 0\_9\_1 |
| CASE7 | 0\_10\_0 | 1\_9\_0 | 0\_10\_0 |
| CASE8 | 1\_8\_1 | 1\_9\_0 | 0\_9\_1 |
| CASE9 | 0\_10\_0 | 0\_10\_0 | 0\_10\_0 |
| CASE10 | 0\_9\_1 | 1\_9\_0 | 0\_10\_0 |
| CASE11 | 0\_9\_1 | 0\_9\_1 | 1\_7\_2 |
| CASE12 | 0\_9\_1 | 0\_9\_1 | 0\_10\_0 |
| CASE13 | 0\_9\_1 | 0\_10\_0 | 0\_10\_0 |
| CASE14 | 1\_9\_0 | 0\_9\_1 | 0\_10\_0 |
| CASE15 | 1\_8\_1 | 0\_8\_2 | 2\_7\_1 |
| CASE2\_1 | 0\_10\_0 | 0\_10\_0 | 0\_10\_0 |
| CASE2\_2 | 2\_8\_0 | 0\_10\_0 | 0\_10\_0 |
| CASE2\_3 | 0\_10\_0 | 0\_10\_0 | 0\_9\_1 |
| CASE2\_4 | 0\_10\_0 | 0\_10\_0 | 0\_10\_0 |
| CASE2\_5 | 1\_9\_0 | 0\_10\_0 | 0\_9\_1 |
| CASE2\_6 | 0\_10\_0 | 1\_9\_0 | 0\_10\_0 |
| CASE2\_7 | 0\_9\_1 | 0\_10\_0 | 1\_9\_0 |
| CASE2\_8 | 0\_9\_1 | 0\_10\_0 | 0\_10\_0 |
| CASE2\_9 | 0\_10\_0 | 0\_9\_1 | 0\_9\_1 |
| CASE2\_10 | 1\_9\_0 | 0\_10\_0 | 0\_10\_0 |
| CASE2\_11 | 0\_9\_1 | 0\_10\_0 | 0\_10\_0 |
| CASE2\_12 | 0\_10\_0 | 0\_10\_0 | 0\_10\_0 |
| CASE2\_13 | 2\_8\_0 | 0\_10\_0 | 0\_10\_0 |
| CASE2\_14 | 1\_9\_0 | 1\_9\_0 | 0\_9\_1 |
| CASE2\_15 | 1\_8\_1 | 0\_10\_0 | 1\_9\_0 |
| CASE3\_1 | 0\_10\_0 | 0\_10\_0 | 0\_10\_0 |
| CASE3\_2 | 1\_9\_0 | 0\_10\_0 | 0\_10\_0 |
| CASE3\_3 | 0\_10\_0 | 0\_10\_0 | 0\_10\_0 |
| CASE3\_4 | 0\_9\_1 | 0\_10\_0 | 0\_10\_0 |
| CASE3\_5 | 0\_8\_2 | 0\_9\_1 | 2\_7\_1 |
| CASE3\_6 | 1\_8\_1 | 0\_10\_0 | 0\_10\_0 |
| CASE3\_7 | 1\_9\_0 | 0\_10\_0 | 0\_10\_0 |
| CASE3\_8 | 0\_10\_0 | 1\_9\_0 | 0\_10\_0 |
| CASE3\_9 | 0\_9\_1 | 0\_10\_0 | 0\_10\_0 |
| CASE3\_10 | 2\_8\_0 | 1\_9\_0 | 0\_10\_0 |
| CASE3\_11 | 0\_8\_2 | 0\_9\_1 | 0\_8\_2 |
| CASE3\_12 | 0\_8\_2 | 0\_9\_1 | 1\_8\_1 |
| CASE3\_13 | 2\_7\_1 | 1\_9\_0 | 0\_10\_0 |
| CASE3\_14 | 3\_7\_0 | 1\_9\_0 | 0\_10\_0 |
| CASE3\_15 | 1\_8\_1 | 2\_7\_1 | 2\_7\_1 |
| CASE3\_16 | 1\_8\_1 | 0\_10\_0 | 0\_10\_0 |
| CASE3\_17 | 0\_10\_0 | 0\_10\_0 | 0\_10\_0 |
| CASE3\_18 | 0\_10\_0 | 1\_7\_2 | 1\_8\_1 |

DDAO algoritmasının baz modeli ile FDB-DDAO varyasyonları arasındaki ikili karşılaştırmalara ait analiz sonuçları aşağıda verilmektedir. Buna göre FDB-DDAO varyasyonları ikili karşılaştırmalarda DDAO algoritmasına üstünlük kurmuşlardır.Bu durumlar için 10.000\*d uygulanmıştır.

|  |  |  |  |
| --- | --- | --- | --- |
| vs. DDAO +/=/- | Boyut = 30 | Boyut = 50 | Boyut = 100 |
| Case2 | 3\_7\_0 | 1\_9\_0 | 0\_10\_0 |
| Case3 | 2\_8\_0 | 1\_9\_0 | 0\_10\_0 |
| Case1 | 0\_10\_0 | 1\_7\_2 | 1\_8\_1 |

## Yakınsama Analiz Sonuçları

Bu bölümde DDAO ve FDB-DDAO varyasyonlarının yakınsama eğrileri sunulmaktadır. Algoritmaların yakınsama yeteneklerini incelemek için CEC 2020 problem setinde dört farklı tipteki problemlerden birer adet seçilmiştir. Seçilen problemlerden 30/50/100 boyutlarında arama uzayları tasarlanmış ve algoritmaların yakınsama davranışları gözlemlenmiştir. Seçilen problemler F1 (Tek Modlu), F3 (Multimodal), F7(Hibrit) ve F10(Kompozisyon)’dir.

|  |  |  |
| --- | --- | --- |
| 30D | 50D | 100D |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Gördüğümüz tabloda dört satır ve üç sütun olmak üzere, algoritmaların on iki farklı durum için yakınsama eğrileri verilmektedir. Buna göre satırlar sırasıyla tekmodlu/ çokmodlu/ melez/ komposizyon tiplerdeki problemleri ve sütunlar da bu problemlerin 30/50/100 boyutlu tasarımlarının olduğu durumları göstermektedir. Bu şartlar altında verilen yakınsama eğrileri incelendiğinde, FDB-DDAO varyasyonlarının DDAO algoritmasının baz modeline kıyasla daha hızlı yakınsadığı söylenemez. Hatta yakınsama hızları arasında önemli bir farklılık olmadığı söylenebilir.Sonuç olarak FDB-DDAO varyasyonuyla bu caseler ile baz algoritmada iyileştirmeye giderek daha iyi sonuçlar alarak verim elde ettik.