**KARADENİZ TEKNİK ÜNİVERSİTESİ**

**OF TEKNOLOJİ FAKÜLTESİ**

**YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ**



**2019-2020 Bahar Dönemi**

**YZM-4032 Meta-Sezgisel Optimizasyon**

**NUMARA - AD SOYAD**

**NUMARA - AD SOYAD**

**BSA FdbPop, FdbPopArtnDu ve FdbPopDinamik Yöntemlerini Kullanarak Geliştirme**

**Doç. Dr. Hamdi Tolga KAHRAMAN**

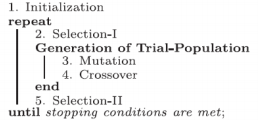
**Backtracking Search Optimization Algorithm**

BSA, gerçek değerli sayısal optimizasyon problemlerini çözmek için yeni bir evrimsel algoritma (EA)’dır. EA'lar, doğrusal olmayan, ayırt edilemeyen ve karmaşık sayısal optimizasyon problemlerini çözmek için yaygın olarak kullanılan popüler stokastik arama algoritmalarıdır. BSA, kontrol parametrelerinde aşırı duyarlılık, erken yakınsama ve yavaş hesaplama gibi EA'larda sık karşılaşılan sorunların etkilerini azaltmayı amaçlamaktadır. Birçok arama algoritmasından farklı olarak BSA'nın tek bir kontrol parametresi vardır. Dahası, BSA’nın problem çözme performansı bu parametrenin başlangıç değerine fazla duyarlı değildir. BSA, etkili, hızlı ve çok modlu problemleri çözebilen ve farklı sayısal optimizasyon problemlerine kolayca adapte olmasını sağlayan basit bir yapıya sahiptir. BSA’nın bir deneme popülasyonu oluşturma stratejisi iki yeni geçit ve mutasyon operatörü içerir. BSA’nın deneme popülasyonları oluşturma ve arama yönü matrisinin ve arama alanı sınırlarının genliğini kontrol etme stratejileri, ona çok güçlü keşif ve arama yetenekleri sağlar. Özellikle, BSA, arama yönü matrisinin üretilmesinde kullanılmak üzere, rastgele seçilen bir önceki nesilden bir popülasyonu depoladığı bir belleğe sahiptir. Böylece BSA’nın hafızası, bir deneme hazırlığı oluştururken önceki nesillerden elde edilen deneyimlerden faydalanmasına izin verir.

BSA, küresel bir küçültücü olarak tasarlanmış, popülasyona dayalı yinelemeli bir EA'dır. BSA, işlevlerini diğer EA'larda olduğu gibi beş sürece bölerek açıklanabilir:

* Başlatma
* Seçim-I
* Mutasyon
* CrossOver
* Seçim-II

Algoritmanın genel işleyişi şekil 1 de verildiği gibidir.



Şekil 1

**Başlatma (Initialization) :** BSA, P popülasyonunu Denklem 1 ile başlatır



Denklem 1

i = 1,2,3 için. . N ve j = 1,2,3;. . ., D, burada N ve D sırasıyla popülasyon büyüklüğü ve problem boyutu, U eşit dağılım ve her Pi, P popülasyonunda hedef bireydir.

**Seçim-I (Selection-I) :** BSA’nın Selection-I aşaması, arama yönünün hesaplanmasında kullanılacak geçmiş "oldP" popülasyonunu belirler. Bu işlem için denklem 2 kullanılır.



Denklem 2

BSA, her yinelemenin başında Denklem 3’te verilen 'if-then' kuralı aracılığıyla oldP'yi yeniden tanımlama seçeneğine sahiptir.

Denklem 3

Burada “=:” güncelleme işlemidir. Denklem 3 BSA'nın rasgele seçilen bir önceki nesle ait bir popülasyonu tarihsel popülasyon olarak atamasını sağlar ve değiştirilene kadar bu tarihsel popülasyonu hatırlar. Böylece BSA'nın bir hafızası vardır. OldP belirlendikten sonra, Denklem 4 eski P'deki bireylerin sırasını rastgele değiştirmek için kullanılır.



Denklem 4

Denklemde kullanılan “permuting” işlevi, rastgele bir karıştırma işlevidir.

**Mutasyon(Mutation) :** BSA'nın mutasyon süreci, Denklem 5 kullanarak mutant deneme popülasyonunun başlangıç formunu oluşturur.



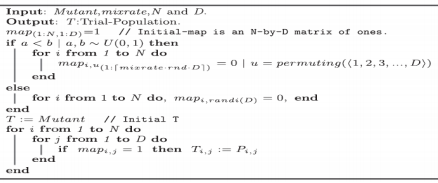
Denklem 5

Denklem 5, F, arama yönü matrisinin genliğini kontrol eder (oldP-P). Çünkü tarihsel popülasyon arama-yön matrisinin hesaplanmasında kullanıldığından BSA, önceki nesillere ait deneyimlerinden kısmen yararlanan bir deneme popülasyonu üretir.

**Çaprazlama (Crossover) :** BSA’nın çaprazlama süreci, deneme popülasyonu T'nin nihai formunu oluşturur. Deneme popülasyonunun başlangıç değeri, mutasyon işleminde ayarlandığı gibi Mutant'tır. Optimizasyon problemi için daha iyi kondisyon değerleri olan deneme bireyleri, hedef popülasyon bireylerini geliştirmek için kullanılır. BSA’nın çaprazlama sürecinin iki adımı var. İlk adım, T'nin bireylerini P'nin ilgili bireyleri kullanılarak manipüle edilecek olan N \* D boyutundaki bir ikili tamsayı değerli matrisi(map-Denklem 6) hesaplar.



Denklem 6

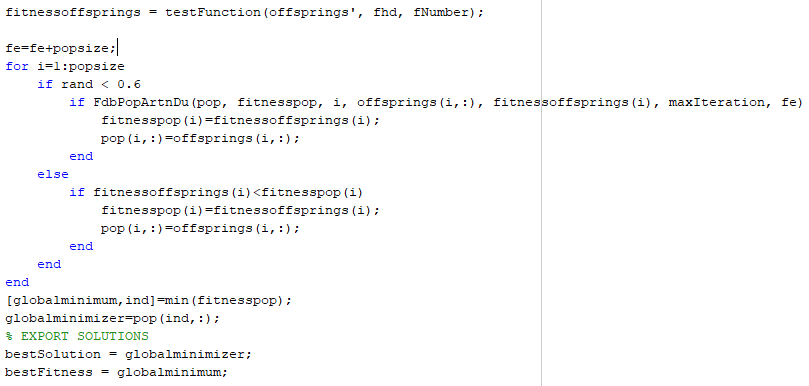


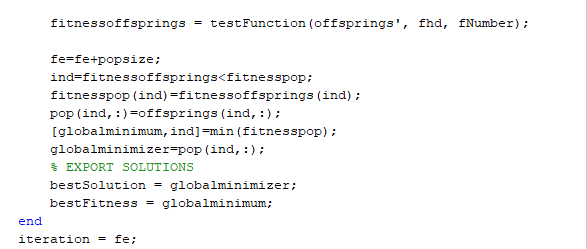
Yukarıdaki algoritma BSA’nın benzersiz çaprazlama stratejisini gösterilmektedir

**Seçim-II (Selection-II) :** BSA’nın Seçim-II aşamasında, uygun "Pis" değerinden daha iyi kondisyon değerlerine sahip "Tis", açgözlü bir seçime dayanarak "Pis" i güncellemek için kullanılır. En iyi P (Pbest) bireyinin BSA tarafından şimdiye kadar elde edilen küresel minimum değerden daha iyi bir uygunluk değeri varsa, küresel minimizer Pbest olarak güncellenir ve küresel minimum değer Pbest'in uygunluk değeri olarak güncellenir. BSA'nın yapısı oldukça basittir; böylece farklı sayısal optimizasyon problemlerine kolayca adapte olur.

**Backtracking Search Optimization Algorithm Geliştirme**

Algoritma içerisinde seçim-I ve seçim-II adımı tespit edilmiştir. Bu adımlar popülasyonun güncellendiği adımlardır. Bu seçimler için daha önceden belirlenmiş fdbpop,fdbpopartndu ve fdbpopdinamik yöntemleri ile rasgele seçimler yapılması sağlanmıştır. Aşağıda bu adımlara uygulanma şekilleri verilmiştir. Sonuçlar excel dosyası şeklinde ek yapılmıştır. Burada dikkat edilmesi gereken husus algoritmanın ham kodunda her iterasyonda çözüm adayları topluca güncelleniyor. Biz ise burada yapıyı döngüye çevirip her çözüm adayının teker teker karşılaştırılmasını sağladık.





Her yöntemi yüzde 20, 40, 60 oranlarında uygulayarak 10 durum elde ettik. Sonuçları excel olarak paylaştık.

**Sonuçlar**

Algoritmanın kendi çalışması ve fdbpop, fdbpopartdu ve fdbpopdinamik yöntemleri kullanılarak sezgiselleştirilmiş algoritmanın çalışması incelenip belirli varyasyonlar altında sonuçlar karşılaştırılmıştır. Algoritma sezgiselleştirmeden önce belirli bir mutasyon matrisi üzerinde işlemler yapıp ekosistemin güncellenmesini sağlamaktadır. Fdb yöntemlerimiz ise tek değer almaktadır. Kod üzerinde küçük değişiklikler yaparak mutasyon matrisini tek tek gönderip yöntemler icra edilmiştir. Bu sezgiselleştirme yöntemlerinin sonuçları ve bsa’nın sonuçları exceller halinde paylaşmıştır.

**Fdbpop ile Sezgiselleştirilmiş BSA ve BSA Karşılaştırması**

Fbdpop ile %20 oranında sezgiselleştirme gerçekleştirdiğinde algoritma %5.1 oranında performans kaybı yaşamaktadır.

Fbdpop ile %40 oranında sezgiselleştirme gerçekleştirdiğinde algoritma %1.3 oranında performans kaybı yaşamaktadır.

Fbdpop ile %60 oranında sezgiselleştirme gerçekleştirdiğinde algoritma %3.2 oranında performans kaybı yaşamaktadır.

**Fdbpopartdu ile Sezgiselleştirilmiş BSA ve BSA Karşılaştırması**

Fbdpopartdu ile %20 oranında sezgiselleştirme gerçekleştirdiğinde algoritma %4.4 oranında performans kaybı yaşamaktadır.

Fbdpopartdu ile %40 oranında sezgiselleştirme gerçekleştirdiğinde algoritma %11.92 oranında performans kaybı yaşamaktadır.

Fbdpopartdu ile %60 oranında sezgiselleştirme gerçekleştirdiğinde algoritma %8.8 oranında performans kaybı yaşamaktadır.

**Fdbpopdinamik ile Sezgiselleştirilmiş BSA ve BSA Karşılaştırması**

Fdbpopdinamik ile %20 oranında sezgiselleştirme gerçekleştirdiğinde algoritma %9.58 oranında performans kaybı yaşamaktadır.

Fdbpopdinamik ile %40 oranında sezgiselleştirme gerçekleştirdiğinde algoritma %11.91 oranında performans kaybı yaşamaktadır.

Fdbpopdinamik ile %60 oranında sezgiselleştirme gerçekleştirdiğinde algoritma %7.7 oranında performans kaybı yaşamaktadır.

Sonuçlar göz önünen alındığında BSA’da önerilen yöntemler başarılı olmamıştır.

**Not:** Karşılaştırmalar excel çıktılarının 9.satırına göre yapılmıştır.