ÇOK MODLU OPTİMİZASYON PROBLEMLERİ İÇİN BİR YAPAY ARI KOLONİSİ ALGORİTMASI GERÇEKLEŞTİRİMİ

Yunus ÖZCAN

Kütahya Dumlupınar Üniversitesi
Lisansüstü Eğitim Öğretim ve Sınav Yönetmeliği Uyarınca
Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalında
YÜKSEK LİSANS TEZİ
Olarak Hazırlanmıştır.

Danışman: Doç. Dr. Doğan AYDIN

Aralık -2019

KABUL VE ONAY SAYFASI

Yunus ÖZCAN tarafından hazırlanan "ÇOK MODLU OPTİMİZASYON PROBLEMLERİ İÇİN BİR YAPAY ARI KOLONİSİ ALGORİTMASI GERÇEKLEŞTİRİMİ" adlı tez çalışması, aşağıda belirtilen jüri tarafından Kütahya Dumlupınar Üniversitesi Lisansüstü Eğitim Öğretim ve Sınav Yönetmeliğinin ilgili maddeleri uyarınca değerlendirilerek OY BİRLİĞİ ile Kütahya Dumlupınar Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalında YÜKSEK LİSANS TEZİ olarak kabul edilmiştir.

03/12/2019

Prof. Dr. Önder UYSAL
Enstitü Müdürü, Fen Bilimleri Enstitüsü _____

Doç. Dr. Doğan AYDIN
Anabilim Dalı Başkanı, Bilgisayar Mühendisliği Anabilim Dalı

Doç. Dr. Doğan AYDIN
Danışman, Bilgisayar Mühendisliği Anabilim Dalı

Sınav Komitesi Üyeleri

Doç. Dr. Doğan AYDIN
Bilgisayar Mühendisliği, Kütahya Dumlupınar Üniversitesi

Dr. Öğr. Üyesi Durmuş ÖZDEMİR
Bilgisayar Mühendisliği, Kütahya Dumlupınar Üniversitesi

Prof. Dr. Eyyüp GÜLBANDILAR
Bilgisayar Mühendisliği, Eskişehir Osmangazi Üniversitesi

ETİK İLKE VE KURALLARA UYGUNLUK BEYANI

Bu tezin hazırlanmasında Akademik kurallara riayet ettiğimizi, özgün bir çalışma olduğunu ve yapılan tez çalışmasının bilimsel etik ilke ve kurallara uygun olduğunu, çalışma kapsamında teze ait olmayan veriler için kaynak gösterildiğini ve kaynaklar dizininde belirtildiğini, Yüksek Öğretim Kurulu tarafından kullanılmak üzere önerilen ve Dumlupınar Üniversitesi tarafından kullanılan İntihal Programı ile tarandığını ve benzerlik oranının %14 çıktığını beyan ederiz. Aykırı bir durum ortaya çıktığı takdirde tüm hukuki sonuçlara razı olduğumuzu taahhüt ederiz.

Doç. Dr. Doğan AYDIN

Yunus ÖZCAN

ÇOK MODLU OPTİMİZASYON PROBLEMLERİ İÇİN BİR YAPAY ARI KOLONİSİ ALGORİTMASI GERÇEKLEŞTİRİMİ

Yunus ÖZCAN
Bilgisayar Mühendisliği, Yüksek Lisans Tezi, 2019
Tez Danışmanı: Doç. Dr. Doğan AYDIN

ÖZET

Optimizasyon problemleri günümüzde çözülmesi zor olan problemlerden biridir. Birden fazla çözüme sahip olan optimizasyon problemlerine çok modlu (multimodal) optimizasyon problemleri denir. Optimizasyon problemlerinde sürü zekâsı algoritmaları çoğunlukla kullanılmaktadır. Sürü zekâsı algoritmalarından biri olan Yapay Arı Kolonisi (Artificial Bee Colony, ABC) algoritması doğadaki arıların yiyecek aramasından esinlenerek oluşturulmuştur. ABC algoritması bir çözüme sahip optimizasyon problemleri için kurgulandığı içi multimodal optimizasyon problemlerinde başarılı olamamaktadır.

Multimodal optimizasyon probleminde yapay arı kolonisi algoritmasını kullanabilmek için algoritmaya çeşitli stratejilerin eklenmesi gerekmektedir. Bu tezde Yapay Arı Kolonisi algoritmasına üç farklı yeni strateji eklenerek multimodal optimizasyon problemlerinde kullanılabilecek "Elit Yiyecek Kaynaklı Adaptif Yapay Arı Kolonisi Algoritması (SABCElit)" geliştirilmiştir. Burada algoritmada gerçekleştirilen stratejiler, "Elit Yiyecek Kaynakları ile Çalışma Stratejisi", "Elastik Popülasyon Güncelleme Statejisi" ve "Kendinden-Uyarlamalı Arama Denklemi Seçme" stratejileridir.

Algoritmaya eklenen parametreler algoritmanın başarısında büyük öneme sahip olduğu için parametre değeri iyi ayarlanmalıdır. irace aracı algoritmaların uygun parametre değerlerinin belirlenmesinde kullanılan bir yöntemdir. Bu tezde, SABCElit algoritmasının parametreleri de irace aracı kullanılarak ayarlanmıştır.

Geliştirilen SABCElit Algoritması multimodal fonksiyonları içeren CEC 2013 fonksiyon ölçüt kümesi üzerinde test edilmiştir. Elde edilen sonuçlar literatürdeki diğer algoritmaların sonuçları ile karşılaştırılmıştır. Karşılaştırma sonucunda algoritmamız toplam 86 noktada iyi sonuç verirken algoritmamıza en yakın algoritma 83 noktada iyi sonuç vermiştir.

Anahtar Kelimeler: Multimodal optimizasyon, Sürekli Optimizasyon. Sürü Zekâsı, Yapay Arı Kolonisi.

ARTIFICIAL BEE COLONY IMPLEMENTATION FOR MULTIMODAL OPTIMIZATION PROBLEMS

Yunus ÖZCAN
Computer Engineering, M.S. Thesis, 2019
Thesis Supervisor: Assoc. Prof. Dr. Doğan AYDIN

SUMMARY

Optimization problems are one of the most difficult problems to solve today. Optimization problems with multiple solutions are called multimodal optimization problems. Swarm intelligence algorithms are mostly used in optimization problems. Artificial Bee Colony (ABC) algorithm, which is one of the swarm intelligence algorithms, was inspired by the foraging behavior of the nature bees. ABC is generally not designed for optimization problems that have sigle solution and cannot be successful in multimodal optimization problems.

In order to use the Artificial Bee Colony algorithm in multimodal optimization problem, various strategies must be added to the algorithm. In this thesis, “Self-Adaptive Artificial Bee Colony Algorithm with Elite Food Sources (SABCElit)” which can be used in multimodal optimization problems has been developed by adding three new strategies to Artificial Bee Colony algorithm. The strategies implemented in this algorithm are “Working with Elite Food Sources”, “Elastic Population Update Strategy” and “Self-Adaptive Search Equation Selection” strategies.

Since the parameters added to the algorithm are of great importance for the success of the algorithm, the parameter value should be tuned properly. The Irace tool is a method that used to determine the appropriate parameter values of the algorithms. The parameters of SABCElit algorithm were set using irace tool as well.

The developed SABCElit algorithm was tested on a CEC 2013 benchmark function set containing multimodal functions. The results were compared with the results of other algorithms in the literature. As a result of the comparison, our algorithm gave good results in 86 points while the algorithm closest to our algorithm gave good results in 83 points.

Keywords: Multimodal optimization, Artificial Bee Colony, Swarm Intelligene, Continuous Optimizastion

TEŞEKKÜR

Çalışmalarım esnasında verdiği destek için ve göstermiş olduğu anlayışından dolayı saygıdeğer danışmanım Doç. Dr. Doğan AYDIN'a, analiz sonuçlarımın yorumlamasında yardımlarını esirgemeyen Arş.Gör.Dr. Gürcan YAVUZ'a, teşekkürlerimi sunarım.

Bu çalışmada maddi ve manevi olarak destek veren eşime, anneme, babama, kardeşime ve çalışma arkadaşlarıma sonsuz teşekkür ederim.

Yunus ÖZCAN

İÇİNDEKİLER

Sayfa

İÇİNDEKİLER (Devam)

Sayfa

İÇİNDEKİLER (Devam)

Sayfa

ŞEKİLLER DİZİNİ

Şekil **Sayfa**

ŞEKİLLER DİZİNİ(Devam)

Şekil **Sayfa**

ÇİZELGELER DİZİNİ

Çizelge **Sayfa**

SİMGELER VE KISALTMALAR DİZİNİ

Simge	Açıklama
$\phi_{i,j}$ $[-a,+a]$	Arasında Rastgele Bir Reel Değer
$nbest$	En İyi Komşu
gbd_j	Global En Yakın Komşunun j. Boyuttaki Değeri
itr_{MAX}	Maksimum İterasyon Sayısı
se_s	s. Sıradaki Arama Denkleminin Başarı Değeri
c_s	s. Sıradaki Arama Denkleminin İlgili İterasyondaki İyileştirdiği Yiyecek Kaynağı Sayısı
$nbest_j$	Yiyecek Kaynağının En İyi Komşusunun j. Boyuttaki Değeri
$dist(x_i - x_n)$	Seçilen Yiyecek Kaynağı İle Diğer Yiyecek Kaynağı Arasındaki Öklid Mesafesi
V_{ij}	Yeni Besin Kaynağı
X_{ij}	Mevcut Besin Kaynağı
p_i	i. Besin Kaynağının Rulet Tekerleğindeki Genişliği
uf_i	i.Besin Kaynağının Uygunluk Değeri
x_j^{max}	Parametrenin Üst Sınırı
x_j^{min}	Parametrenin Alt Sınırı
a	Her Bir Denklem Örneği İçin Rastgele Üretilen Bir Pozitif Reel Sayı
i	0 ile Besin Kaynağı Sayısı Arasında Bir Sayı
j	0 ile Parametre Sayısı Arasında Bir Sayı
k	Sıfırla Yiyecek Kaynağı Değeri Arasında Rastgele Sayı
n	0 İle UD Arasında Bir Sayı
Φ	(-1 1) Arasında Rastgele Sayı
$abs(f_i)$ fi	Değerinin Mutlak Değeri
fi	i. Besin Kaynağının Amaç Fonksiyon Değeri
$raund(0,1)$	0 İle 1 Arasında Rastgele Sayı

SİMGELER VE KISALTMALAR DİZİNİ (Devam)

Kısaltma	Açıklama
ABC	Artificial Bee Colony
ACO	Ant Colony Optimization
AIS	Artificial Immune Systems
ANOF	Adaptive Niching-Based Evolutionary Algorithm
ANPSO	Adaptive Niching PSO
AR	Average Number Of Function Evaluations
CA	Cultural Algorithms
CAMES	Covariance Matrix Adaptation Evolution Strategy
CDE	Clustering Based Differential Evolution
CF	Crowd Factor
CFES	Fonksiyon Çağırma Sayısı
DADE	Dynamic Archive DE
DC	Deterministic Crowding
DE	Differential Evolution
EA	Evolutionary Algorithms
EDA	Estimated Distributed Algorithm
Fe	Algoritmanın Döngü Sayısı
FER-PSO	Fitness-Euclidean Distance Ratio PSO
FIPS	Fully Informed PSO
FN	Fonksiyon Numarası
GA	Genetic Algorithms
gbd	Global En Yakın Komşu
Maxfes	Maximum Çalıştırma Sayısı
MMO	Multi-Model Optimization
NCSDE	Niching Community Species Based DE
NichePSO	Niching Particle Swarm Optimization
NKP	Toplam Optima Sayısı
NPFi	i. Çalışmadaki Bulunan Optima Sayısı
NR	Toplam Çalıştırma Sayısını
NSF	Niching Swarm Filtering
NSR	Başarılı Çalıştırma Sayısı
PR	Peak Ratio

SİMGELER VE KISALTMALAR DİZİNİ (devam)

Kısaltma	Açıklama
Ps	Havuz Boyutu
PSO	Particle Swarm Optimization
SABCElit	Elit Yiyecek Kaynaklı Adaptif Yapay Arı Kolonisi Algoritması
SPSO	Speciation-Based PSO
SR	Success Rate
UD	Uygunluk Değeri Sayısı
VPSO	Vector-Based PSO

1. GİRİŞ

Günlük yaşantımızda karşılaştığımız problemlerin bazıları bilgisayar biliminde çözümü zor olan NP (Non-polinomial) türünde problemlerdir. Bu problemlerin önemli bir kısmını optimizasyon (eniyileme) problemleri oluşturmaktadır. Optimizasyon probleminde amaç, bir amaç fonksiyonunu minimize veya maksimize etmektir. Bazen bir amaç fonksiyonunda birden çok mimimum veya maksimum çözüm, çözüme çok yakın yerel optimum çözümler bulunabilir. Bu tür optimizasyon problemleri çok modlu (multimodal)[1] optimizasyon problemleri olarak adlandırılır (Wong vd., 2010). Multimodal problemlerin çözümü güç olduğu için çözümünde genellikle yaklaşım yöntemleri kullanılır. Bunlar içerisinde metasezgisel yöntemler son yıllarda başarılı sonuçlar elde edildiği için yoğunlukla kullanılmaktadır.

Metasezgisel yaklaşımların bir kısmı doğadan esinlenen sürü zekasına dayalı algoritmalardır. Bu yöntemlerden bazıları; Yapay Arı Kolonisi Algoritması (ABC), Karınca Kolonisi Algoritması (ACO) ve Parçacık Sürü Algoritmasıdır (PSO). ACO karıncaların yem arama özelliklerinden, PSO kuş sürülerinin hareketlerinden, ABC ise doğadaki arıların yiyecek kaynağı aramasında esinlenerek geliştirilmiştir. Bu algoritmalar ilk olarak tek optimum çözüme sahip olan optimizasyon problemlerinin çözümü için tasarlanmıştır ve bu problem türlerinde algoritmalarla başarılı sonuçlar elde edilmiştir.

Metasezgisel yöntemler tek bir optimum çözüme odaklı çalıştıkları için multimodal problemlerin çözümünde başarılı olamamaktadır. Bu tür problemlerin çözümünde bu algoritmaların kullanılabilmesi için uygun değişikliklerin yapılması gerekmektedir. Bu tezde, ABC algoritmasının multimodal problemlerin çözümünde kullanılabilmesi amaçlanmıştır. Başarılı sonuçlar elde edilebilmesi için gerekli olabilecek değişiklikler gerçekleştirilerek "Elit Yiyecek Kaynaklı Adaptif Yapay Arı Kolonisi Algoritması" adında yeni bir ABC varyantı geliştirilmiştir. Geliştirilen ABC varyantının başarısı olabilmesi için parametre ayar yöntemleri etkin olarak kullanılmıştır. Başarım testleri için iyi bilinen ve yaygın olarak bir ölçüt fonksiyon kümesi CEC 2013 üzerinde yapılmıştır. Elde edilen sonuçlar ile literatürdeki bilinen iyi sonuçlar karşılaştırılmıştır. Tezin içeriği aşağıdaki gibi organize edilmiştir;

Birinci bölümde, optimizasyon teriminin tanımı ve çeşitleri anlatıldıktan sonra multimodal optimizasyon problemlerinin tanımı, çözüm yolları ve gerçek dünya problem örnekleri anlatılmıştır.

[1] Tezin diğer bölümlerinde "çok modlu" kelimesi yerine "multimodal" kelimesi tercih edilmiştir.

İkinci bölümde algoritmamızın temelini oluşturan yapay arı kolonisi algoritması anlatılmıştır.

Üçüncü bölümde multimodal optimizasyon problem seti için geliştirdiğimiz elit yiyecek kaynaklı adaptif yapay arı kolonisi algoritması anlatılmıştır.

Dördüncü bölümde problem seti, algoritmamızın parametre ayarlarının nasıl yapıldığı, algoritma parametre değerleri ve problem setinin çözümünde kullanılan donanım anlatılmıştır.

Beşinci bölümde algoritmamızın, multimodal optimizasyon problem setindeki başarısı ve diğer algoritmalarla karşılaştırılması yer almaktadır.

2. PROBLEM TANIMI

2.1. Optimizasyon

Optimizasyon, birçok disiplinin (mühendislik, işletme ve fen bilimleri) ilgi alanına girmiş ve daha önceleri çözümü zor olarak kabul edilen problemlerin çözümlerinde yardımcı olmuştur. (Antoniou ve Lu, 2007).

Optimizasyon günlük yaşantılarında insanların ve doğanın, sıklıkla uyguladığı bir kavram olup fiziksel sistem analizinde ve karar biliminde kullanılan önemli bir araçtır (Nocedal vd., 1999). Matematiksel fonksiyonlarla tanımlanan problemlerde en iyi çözümü bulmayı amaçlayan bilim dalıdır (Fletcher, 2013). Bir başka ifade ile girdilerin maksimum ya da minimum çıktısını bulmaya yarayan matematiksel bir süreç olarak tanımlanmaktadır (Randy ve Sue, 2004).

Optimizasyon sürecinde en iyi ve en kötü bulunmaya çalışılır. Problem için en uygun çözüm kümesini bulmayı amaçlamaktadır. Ayrıca optimizasyon teorisi, en uygun değerin hesaplanmasında kullanılabilecek teknikler, metotlar, prosedürler ve algoritmalar bütünüdür (Antoniou ve Lu, 2007).

Optimizasyon konusundaki yaklaşımlar; analitik metotlar, grafik metotlar, deneysel metotlar ve nümerik metotlar olmak üzere dört farklı şekilde sınıflandırabilir (Antoniou ve Lu, 2007). Analitik metotlarda problem matematiksel olarak ifade edilmek zorundadır. Problem çözülürken matematiksel fonksiyonun türeviyle ilgilenir. Yüksek dereceli doğrusal olmayan problem çözümlerinde ve bağımsız değişken sayısının üçten fazla olduğu durumlarda analitik metotlar uygulanamaz. Grafik metotla problem çözülürken amaç fonksiyonun grafiği çizilerek maksimum veya minimum noktası bulunmaya çalışılır. Grafik metot fonksiyon sayısının ikiyi geçtiği durumlarda kullanılması zor olduğu için kısıtlı bir alanda kullanılmaktadır. Deneysel metotla problem çözülürken probleme uygun deneye ortamı hazırlanır ve deney ortamında deneme yanılma yöntemiyle problemin maksimum ve minimum noktası bulunmaya çalışılır. Deney ortamında olduğu için tam olarak maksimum ve minimum nokta bulunamayabilir. Değişken sayısının çok olduğu ve çevre koşulundan etkilenebilir problemlerde deneme yanılma yöntemi doğru sonuç vermeyebilir. Nümerik metotla problem çözümünde aşamalı olarak nümerik metotlar uygulanarak çözüme ulaşılmaya çalışır. Problemin çözümünün zor olduğu ve bağımsız değişken sayısının çok olduğu problemlerde kullanılır. Matematiksel olarak zor problemler olduğu için bilgisayar ortamında çözülür. Nümerik optimizasyon matematiksel programlama olarak da adlandırılmaktadır. Son kırk yıl içerisinde gelişen matematiksel programlama modelleri;

doğrusal, doğrusal olmayan, tam sayılı, kuadratik ve dinamik programlamadır (Antoniou ve Lu, 2007).

Optimizasyon modelinde maksimum veya minimum yapılmaya çalışılan fonksiyona amaç fonksiyonu denir. Karar değişkeni, sistem performansını etkileyen kontrol altında tutulması gereken değişkenlerdir. Kısıtlar, karar değişkenlerinin alabileceği aralığı belirleyen parametrelerdir (Winston, 2004).

Zamana bağlı değişen optimizasyon problemlerine dinamik optimizasyon zamana göre değişmeyen optimizasyon problemlerine statik optimizasyon denir. Statik optimizasyon problemlerinde matematiksel model cebirsel ve soyut denklemlerden oluşur.

Optimizasyon problemleri, optimizasyon fonksiyonundaki değişkenlerin niteliğine bağlı olarak sürekli, ayrık ve karma olarak sınıflandırılabilir. Giriş ve çıkış değerleri gerçek sayılardan oluşmayan problemlere kesikli problemler denir. Giriş ve çıkış değerleri gerçek sayılardan oluşan problemlere sürekli optimizasyon problemleri denir (Lozano vd., 2011). Tezde üzerinde çalışılacak optimizasyon problemleri sürekli optimizasyon problemleri olduğu için aşağıdaki bölümde bu konu ele alınmıştır.

Literatürde çeşitli sürekli optimizasyon problemleri vardır (Jamil ve Yang, 2013). Bu problemlerde fonksiyonları minimize veya maksimize edilebilmesi için çeşitli sürekli değişken değerlerinin elde edilmesi gerekir. Fonksiyonlar yapısına göre unimodal ve multimodal olarak yapılandırılır. Unimodal yapılar bir noktada local optimum noktaya sahiptir.

Lokal ve global optimumların bulunmasında kullanılan optimizasyon tekniklerine aşağıdaki örnekler verilebilir (Akay, 2009a).

- Tek boyutta minimizasyon teknikleri (Golden Section)
- Türeve dayalı olmadan çok boyutlu arama teknikleri (Conjugate Gradient)
- Birinci türeve dayalı teknikler (Quasi Newton)
- İkinci türeve dayalı teknikler (Newton, Levenberg, Mardquardt)
- Modern sezgisel algoritmalar (Ant Colony Optimization(ACO) , Genetic Algorithms (GA), Artificial Bee Colony(ABC))

2.2. Multimodal Optimizasyon

Gerçek dünya optimizasyon problemleri arama alanında birden fazla optimal çözüme sahip olabilir (Wong vd., 2010; Wong vd., 2015; Wang vd., 2012). Bir problemin sonucuna karar vermeden önce problemle ilgili birden fazla optimum çözüm olmasını istenir. Problem için karar

verilen çözüm uygun değilse zaman kaybetmeden alternatif bir çözüm üzerinden devam etmek istenir.

Algoritmanın, bir kez çalıştırılıp birden fazla çözüm bulunması aşağıdaki faydaları sağlamaktadır.

- Problem hakkında daha detaylı bilginin oluşmasını sağlar.
- Problemin gizli özelliklerinin ortaya çıkmasını sağlar.
- Problemin çözüm uzayında birden fazla çözüm aramak çözüm uzay çeşitliliğin korunmasını sağlar.
- Birden fazla çözümün olması çözüm için esneklik sağlar. Esneklik sayesinde kullanılan çözümde yaşanacak herhangi bir sıkıntıda verim, bakım ve kalite sonuçları negatif olarak etkilenmez.
- Birden fazla çözümün olması problemdeki kullanılan çözümün doğrulamasını yapmada kullanılabilir.

Multimodel yöntemlerde optimum çözümün bulunduğu tepe noktasından belirli uzaklıkta bulunan çözümler yeni aday optimum çözüm olabilirler. Aday optimum çözümün optimum noktadan uzaklığı probleme göre değişiklik göstermektedir.

Multimodel optimizasyon probleminin matematiksel tanımı aşağıdaki gibidir.

$$f(X) : Rn \rightarrow \mathrm{R}, \tag{2.1}$$

X, karar değişkenlerden oluşan bir vektördür.

$$X = \{x1, x2, \ldots, xn\} \in \mathrm{R}. \tag{2.2}$$

F(x) fonksiyonlarının çözümü f(x) değerinden küçük olan problemler($f(x*) \leq f(x)$) *global* minimizasyon problemleridir. F(x) fonksiyonunun çözümü f(x) değerinden büyük problemler($f(x*) \geq f(x)$) global maksimizasyon problemleridir. X değerleri fonksiyonda istediğimiz sonuçları verdiği için çözüm kümemizi oluşturur (Wedyan, 2017).

2.3. Multimodal Optimizasyon Probleminde Kullanılan Yöntemler

Niş kavramı genel olarak çevrede bulunan kaynakların bir alt kümesi olarak tanımlanabilir. Farklı tür ve organizmalar farklı yaşam alanlarında yaşayabilmek için farklı nişlere ihtiyaç duyarlar. Tür belirli bir nişten yararlanan bireyler sınıfıdır. Her niş sınırlı sayıda kaynağa sahip olduğundan çevrede zamanla farklı türler ortaya çıkabilir (Je, 1997).

Optimizasyonda tepe noktasının bulunduğu alan için niş terimi kullanılır. Türler, tepe çevresinde tutulan alt popülasyonlardır.

Niş yöntemleri daha çok, multimodal optimizasyon problemlerini çözmek için tasarlanmıştır. Meta sezgisel yöntemlerde nüfus çeşitliliği meta sezgisel algoritmanın arama alanının belirlenmesinde önemli bir etkiye sahiptir.

Nüfus çeşitliliği arttıkça algoritmanın ilerlemesi azalmaktadır. Nüfus çeşitliliğin korunması ve mevcut çözümün iyi bir doğrulukta olması arasındaki dengenin kurulması, tek bir optimal çözüm bulmak istenen algoritmalar için ortak amaçtır. Niş algoritmada nüfus çeşitliliğinin artmasının yanı sıra birden fazla optimal çözüm bulunmasına yardımcı olur. Nüfus çeşitliliğinin tek başına fazla olması bir den fazla iyi çözümün bulunabileceği anlamına gelmemektedir. Algoritmada nüfus çeşitliliği belirlenirken istenilen çözümlere yakınsamasına izin verilmelidir.

2.3.1. Niş yapma yönteminde kullanılan klasik metotlar

Fitness paylaşımı ve kalabalıklaştırma (Fitness sharing and crowding)

Klasik niş yapma yöntemlerinden en fazla kullanılan yöntem fitness paylaşım yöntemidir. Yöntem, popülasyondaki bireylerin benzerliğine dayanarak birkaç alt popülasyona bölme mekanizması olarak tanımlanabilir (Goldberg ve Richardson, 1987). Fitness paylaşımı, sınırlı kaynağa sahip bir bölgede yaşayan aynı nişi işgal eden farklı bireylerin kaynağı paylaşımından ilham alınarak oluşturulmuştur. Fitness paylaşımı bir bireyin fitness değerini komşu bireylere göre düşürerek farklı bir popülasyon sağlamayı amaçlar. Yakın noktalarda bulunan bireylerin fitness değerleri düşürülerek farklı alanlara yoğunlaşmayı sağlamaktadır.

Fitness paylaşım metodunun en zor kısmı niş yarıçapını ve ölçeklendirme faktörünün (Goldberg ve Deb, 1992; Darwen ve Yao, 1995) uygun değerin belirlenmesidir.

Fitness paylaşımını geliştirmek için dinamik fitness paylaşımı (Della Cioppa vd., 2007), dinamik niş paylaşımı (Miller ve Shaw, 1996) ve temizlik (Petrowski, 1996) teknikleri gibi birçok teknik oluşturulmuştur.

Fitness yaklaşımı kalabalık bireylerin fitness değerini düşürmeyi amaçlarken kalabalık bölgeden uzak ve iyi sonuçların fitness değerini arttırır. Kalabalık bölgede bir yavru rastgele örneklerle kıyaslanır ve içerisindeki en benzerle değiştirilir. Örnek büyüklüğü belirlemek için Crowd Factor (CF) kullanılır.

Kalabalık üzerinde değişiklikler yapılarak Deterministic Crowding (DC) algoritma geliştirilmiştir. Algoritma sonucunda çok sayıda tepe noktası tespit edilip korunabilmiştir.

Diğer metotlar

Bilinen diğer metotlar sınırlı turnuva seçimi (Harik, 1995), çok bireyli GA, temizlemeyi içeren (Petrowski, 1996) birçok başka niş yöntemi geliştirilmiştir. Standart evrimsel algoritmada niş yöntemlerinin temel amacı nüfus çeşitliliğinin korunmasıdır. Birden fazla optimal çözüm bulabilmek için nüfus çeşitliğinin olması çok önemlidir.

2.3.2. Niş yapma yönteminde kullanılan yeni metotlar

Meta sezgisel algoritmalar niş oluşumunu teşvik etmede popüler hale gelmişlerdir. Bu amaçla kullanılan meta sezgisel algoritmalar arasında Parçacık sürü optimizasyonu (PSO) ve Diferansiyel Evrim (DE) algoritması vardır. Niş oluşturma yöntemleri yerel arama yöntemleriyle beraber kullanılmıştır.

Parçacık sürü optimizasyonu

PSO, kuş sürülerinden ilham alınarak oluşturulmuştur. Algoritmada her bir parçacık en iyi konumun bilgisini hafızasında tutar ve bilgiyi sürünün diğer parçacıklarıyla paylaşabilir. Her yenilemede her bir parçacık bilenen en iyi pozisyonun skolastik ortalamasının bulunduğu konuma doğru itilir. PSO'nun temel algoritması çoklu çözüm bulmak için yeterli değildir (Engelbrecht vd., 2005). Yeterli olmamasının temel nedeni parçacıkların birleştikçe sürü çeşitliliğinin kaybolmasıdır.

Her parçacığın hafızasının olması ve o anki en iyi konumun tutulması niş yöntemi için kullanılabilir. Bir küme kâşif ve en iyi konumların tutulduğu iki kısma ayrılabilir. Kâşif küme arama alanını daha geniş bir şekilde keşfetmeye eğimliyken diğer küme en iyi konumların arşivini tutma eğilimindedir.

Sınırlandırılmış bir iletişim topolojisi sürünün parçacıkları üzerinde haritalanırsa bu parçacık sadece topolojik alanda en iyi yerel konumlarına çekilecektir. Arama ilerledikçe farklı nişler farklı alanlarda bulunan optimalar üzerinde bulunduğu için çoklu optimaların bulunması sağlanmış olur. Örnek olarak Fitness-Euclidean Distance Ratio (FER-PSO) (Li, 2007), Ring Topology Based Niching PSO (Li, 2010) ve Fully Informed PSO (FIPS) (Mendes vd., 2004) algoritmaları verilebilir.

Klasik niş davranışını teşvik etmek için faydalı olduğu düşünülen klasik niş metotlarıyla beraber çalışan PSO algoritmaları oluşturulmuştur. Türü tanımlamak için niş yarıçapının önceden belirtilmesi gerekmektedir. Her yenilemede türlerin birleşmesine veya ayrılmasına izin verilir.

Farklı türlerin farklı optimalar etrafında adaptif olarak oluşumu Speciation-Based PSO (SPSO) (Li, 2004; Parrott ve Li, 2006) makalesinde incelenmiştir.

Diğer benzer yöntemler arasında Niching Particle Swarm Optimization(Niche PSO) (Brits vd., 2002), nbest PSO (Brits vd., 2002) ve Multiswarms (Blackwell ve Branke 2006) bulunmaktadır.

Her bir yenilemede istatiksel olarak hesaplama yapılarak niş yarıçapını önceden belirleme ihtiyacını ortadan kaldırmak için Adaptive Niching PSO (ANPSO) (Li, 2011) algoritması geliştirilmiştir. Niş yarıçapının önceden belirlenmesini önlemek için Vector-Based PSO (VPSO) algoritması geliştirilmiştir (Schoeman ve Engelbrecht, 2004).

Differential evolution (DE)

Differential Evolution (DE), popülasyon tabanlı sezgisel bir optimizasyon tekniğidir. DE algoritmasının PSO algoritmasından farkı yeni yavrular üretilirken bireylerin konumları rastgele örneklenmiş çiftler arasındaki fark kullanılarak belirlenir. DE, arama alanına belirli aşamaya kadar kendiliğinden adapte olur.

Rökünen Multi-Model Optimization (MMO) problemleri için farklı local ve global seçiminin nasıl kullanılacağı hakkında fikirler sunmuştur (Rönkkönen ve Lampinen 2007). Bu yerel seçim yönteminde yavru sadece kendi ebeveyniyle rekabet ettiğinden Evolutionary Algorithms (EA) da kullanılan DC ye benzemektedir. EA nın DC ye göre avantajı ek bir niş parametresi belirtme zorunluluğu olmamasıdır.

DE algoritması hakkındaki çalışmalarda, bireylerin bazı yenilemelerden sonra local ve global küresel optima etrafında toplanma eğiliminde olduğu ortaya çıkmıştır (Epitropakis vd., 2011; Epitropakis vd., 2012). DE değişkenleri için oluşturulabilecek değişen kümelenme eğilimi dereceleri ölçmek için H ölçüm metodu önerilmiştir (Epitropakis vd., 2011). Bu gözlemden esinlenerek DE/rand/1mutasyon operatörü ilave kontrol parametresi eklemek zorunda kalmadan niş yapma davranışını uyarmak üzere geliştirildi (Epitropakis, vd., 2011). Standart DE koduna birkaç satır eklenerek DE/rand/1 algoritması üretilmiştir. Algoritmada dağıtılmış bireyler küresel optimum çevresine kümelenmektedir.

Clustering Based Differential Evolution (CDE) algoritması, seçim operatöründe yavruların, popülasyondaki en yakın rakipleriyle karşılaştırılmasını kullanarak geliştirilmiştir (Thomsen, 2004).

Bireylerin türlerine göre topluluk stratejisi veya açgözlü seçim yaklaşımlarını kullanarak Niching Community Species Based DE (NCSDE) algoritması geliştirilmiştir. NCSDE algoritmasında niching parametreleri belirlerken önceki bilgilere ihtiyaç yoktur ve birebir açgözlü yaklaşımıyla birey çeşitliliği korunmaya çalışılır (Huang vd., 2018).

Seçilen popülasyon büyüklüğüne göre niş yönteminin bulacağı optimal sonuç değişir. Popülasyon büyüklüğü bağımlılığından kurtulmak için Dynamic Archive DE (DADE) algoritması geliştirildi (Epitropakis vd., 2013).

Uygunluk ve yakınlık bilgisine dayanan olasılıklı ebeveyn seçim mekanizmasına sahip DE algoritmaları geliştirilmiştir. Mutasyon aşamasında hedef vektöre yakın daha zeki bireylerin seçilme olasılığını arttırmak için ebeveyn seçimi olasılıksal bir şema ile değiştirilir. Bu takas, algoritmanın kesif kabiliyetini engellemeden iyi sonuç alınabilecek bölgelerin değişmesini en aza indirmek için kullanılmıştır.

Niş yöntemlerinin ortak özelliği gelişen nüfusun yakınlık bilgilerinin arama işlemine dâhil edilmesidir. Bu niş yapma yönteminin karmaşıklığını arttırmaktadır. Bu problemi hafifletmek için Zhang ve arkadaşları çift yönlü mesafe yerine toplama mekanizmasını içeren algoritma geliştirdiler (Zhang vd., 2017).

Other meta-heuristics

Niş yöntemleri multimodel problemler için etkili bir çözüm sunmaktadır. Niş topoloji metotları kullanılarak parametrelerde karşılaşılan sorunlar aşılmaya çalışılmıştır. Xiaodong Li ve arkadaşları tarafından halka topolojisi kullanılarak R2PSO ve R3SPO algoritması geliştirilmiştir (Li, 2010). R2PSO veR3PSO algoritmaları, indeks tabanlı halka topolojisine sahip PSO çeşididir. R2PSO algoritmasında her birey sağındaki komşunun local hafızasında depolanan bilgileri kullanarak gelişirken, R3PSO algoritmasında her birey sol ve sağındaki komşunun local hafızasında depolanan bilgileri kullanarak gelişir.

Multimodal problemlerde optimalar arasındaki mesafeyi belirleyen niş parametre değerlerini belirlemek zordur. Xiaodong Li ve arkadaşları, multimodal problemler için belirlenmesi zor olan parametreleri ortadan kaldıran FER-PSO algoritmasını önermiştir. Algoritma popülasyon yeterince büyük olduğu problemlerde optimalar arasındaki mesafeyi uygun öklid mesafe oranını göre belirler (Li, 2007).

Multimodal problemlerini optimize etmek için niş temelli evrimsel algoritma, Adaptive Niching-based evolutionary algorithm (ANOF) geliştirilmiştir. Potansiyel optimalardan daha

küçük objektif değere sahip ve daha uzak noktada optimaların olduğu varsayımına dayanarak geliştirilmiştir (Luo ve F. Gu, 2016).

Multimodal problemlerin çözümünde kullanılan diğer algoritmalar arasında Cultural Algorithms (CA) (Ali ve Awad, 2014). Artificial Immune Systems (AIS) (Forrest ve Javornik 2013; Castro ve Timmis, 2002), Estimated Distributed Algorithm (EDA) (Yang vd., 2016) algoritmaları vardır.

2.4. Multimodal Gerçek Dünya Problemleri

Gerçek dünya mutlimodel optimizasyon problemlerine makas yapma optimizasyonu (Truss-structure optimization), kısıtlı kaynaklı proje çizelgeleme optimizasyonu(Resource constrained multi-project scheduling problems), vücut hareketinin gerçek zamanlı takibi (Real-time tracking of body motion), çeşitli hat aralıklı holografik ızgara tasarımı (Holographic grating design), görüntü segmentasyonu (Image segmentation) (Birattari ve Kacprzyk, 2009), protein yapısı tahmini (Protein structure prediction), denklem sistemlerinin çözümü (Solving systems of equations) örnek verilebilir.

2.4.1. Makas yapma problemi (Truss-structure optimization)

Minimum ağırlığa sahip en uygun kesit, topoloji ve yapıyı oluşturma problemidir. Konsantre yük ve düğüm noktalarını dikkate alarak, elemanların boyutları ve düğüm noktaları arasındaki bağlantıları en az ağırlığa sahip olması gerekir. Algoritmanın, problemin tüm tasarım kısıntıları dikkate alarak sonucu bulması gerekir. Optimum sonuca sahip birden çok topoloji bulunabilir. Problem iki aşamalı olarak Luh ve Lin (López-Ibáñez ve Stützle, 2014) makalesinde incelenmiştir. İlk aşamada optimum topolojiler aranmış, ikinci aşamadaysa ilk aşamada bulunan optimumum topolojiye uygun optimum boyut ve şekil aranmıştır. Problemin çözümü için fitness paylaşımına sahip ikili PSO yapısı kullanılabilir.

2.4.2. Kısıtlı kaynak problemi (Resource constrained multi-project scheduling problems)

Ortak kaynak havuzu kullanarak birden çok proje tanımlanması problemidir. Proje için öncelik sınırlarını ve kaynak kısıtlarını dikkate alarak projelerin görevlerine öncelik verilmesi gerekir. Projelerin tekrar oluşturulması ve proje görevlerinin sınırlandırılması sırasındaki zaman kaybını azaltmayı amaçlamaktadır. Deterministik kalabalıklaştırma ve temizleme yöntemleri problemde çoklu optimal çözümlerin bulunması için kullanılmıştır.

2.4.3. Vücut hareketinin gerçek zamanlı takibi problemi (Real-time tracking of body motion)

Vücut hareketlerinin gerçek zamanlı izlenmesi problemidir. Problemin çözümü için Niching Swarm Filtering (NFS) algoritması geliştirilmiştir.

2.4.4. Holografik ızgara tasarım problemi (Holographic grating design)

Holografik ızgaralar, vakum ultraviyole ve yumuşak X-ışını spektrumundaki ana kırınım elementleridir. Son yıllarda, çeşitli hat aralıklı holografik ızgaraları, kendi kendine odaklanma yapıp, sapmaları giderici özellikleri nedeniyle yüksek çözünürlüklü spektrometreler ve monokromatörlerde yaygın olarak kullanılmaktadır. Varied-Line-Spacing holografik ızgaraların uygulanması yardımcı optik elemanların sayısını azaltır, difraktif aletlerin yapısını basitleştirir, ışık verimini ve optik çözünürlüğü verimli bir şekilde arttırır. Şematik diyagramında ışın ayırıcı, ışık filtresi vb. gibi birçok yardımcı optik eleman vardır. Kayıt parametreleri uygun şekilde seçilmezse, bu yardımcı elemanlar birbirlerine müdahale eder. Yardımcı elemanları matematiksel olarak ifade etmek zor olduğundan birden fazla çözüm bulunup denemeler yapılarak yardımcı elemanların durumu gözlenmesi gerekir (Qing vd., 2008).

2.4.5. Metabolik ağ modellemesi problemi (Metabolic network modelling)

Metabolik ağ, genelleştirilmiş Kütle Eylem Kinetiği formülü kullanılarak modellenmiştir. Klasik optimizasyon probleminin verdiği tek optimum sonuç biyolojik olarak mantıklı olmayabilir. Kronfeld makalesinde, sorunun çok yönlü bir model olduğunu ve hassasiyet analizi için birlikte kullanılabilecek çok sayıda yüksek kaliteli çözüm bulunduğunu anlatmıştır. Bazı parametreler diğer parametrelere göre daha hassastır. Hassaslığı az olan parametre değerlerinin belirlenerek daha iyi çözüm bulunması sağlanmış olur (Kronfeld vd., 2009).

2.4.6. İlaç Molekül tasarım problemi (Drug molecule design)

Birbirinden farklı çözeltilerde molekül seçimi yapılması çok amaçlı kısıtlı optimizasyon problemidir. Çözelti elde edildikten sonra uzmanlar tarafından çözeltinin fiziksel deneylerden geçmesi gerekmektedir. Bu nedenle modelin hedefler ve kısıtlar konusunda birden fazla çözüm çeşitliliği sağlaması gerekmektedir. Problem J. W. Kruisselbrink ve arkadaşları tarafından geliştirilen Non-Dominated Sorting Genetic Algoritlim algoritmasıyla optimize edilerek tasarım alanında çözüm çeşitliliği arttırılmaya çalışılmıştır (Kruisselbrink vd., 2009).

2.4.7. Femtosecond lazer dalga şekillendirme problemi (Femtosecond laser pulse shaping problem)

Yüksek kalitede birden fazla benzersiz nabız profili bulmak için uygulanabilir ilk çözüm arasındaki mesafe tanımlanmıştır. Bu durumda farklı nişlerin aynı kavramsal tasarımları temsil etmesi sağlanmış olur. Femtosaniye Lazer Nabız Şekillendirme problemini çözmek için Covariance Matrix Adaptation Evolution Strategy (CMAES) tabanlı bir niching metodu kullanılmıştır (Aydın vd., 2017).

2.4.8. Otomatik nokta belirleme problemi (Automatic point correspondence)

Görüntü özellik noktaları arasındaki uyuşmazlıkların tespiti, iki veya daha fazla veri setinin geometrik olarak hizalanması, 2D görüntülerin 3B şeklinin geri kazanılması, nesne parçalanması gibi bir dizi görüntü işleme uygulamasında gereklidir. İki görüntüdeki homolog noktaları bulmak için en popüler yöntemlerden biri, iki görüntünün ilgi noktalarında ortalanmış görüntü yamalarının karşılaştırılmasına karşılık gelen şablon eşleşmesidir. DC ve Reactive Tabu Search olarak adlandırılan iki yavru yerleştirme operatörü, rulet tekerleği seçimi ve sınırlı çiftleşme olarak adlandırılan iki seçim operatörü ile test edilmiştir. En iyi performans gösteren çok modlu GA varyasyonunun seçimi ve ince ayarlanması, 2B matematiksel fonksiyon kullanılarak elde edilmiştir. Tanımlanması ve ince ayarlanması yapılan multimodal GA tabanlı algoritması kullanılarak, bir çift 2D görüntünün otomatik nokta yazışmaları için doğru nesnel işlevi optimize edilmiştir. GA tabanlı optimizasyon algoritması geliştikçe, birinci görüntü üzerinde birden fazla aday nokta pozisyonu tespit ederken, local dönüşüm parametreleri ikinci görüntüde homolog nokta pozisyonlarını tanımlar. Önerilen multimodal GA esaslı algoritma, bilinen dönüşüm altında 2D gerçek tıbbi görüntülere ve bilinmeyen dönüşüm bilgisayar tomografisi dilimlerine uygulanmıştır (Delibasis vd., 2010).

2.4.9. Atölye çizelgeleme problemi (Job shop scheduling problem)

Atölye çizelgeleme problemi, literatürde sıkça rastlanan optimizasyon problemlerindendir. Perez ve arkadaşları problemin çözümü için GA tabanlı MMO metodunu geliştirmiştir (Pérez, 2003; Pérez vd., 2012).

2.4.10. Sismolojik ters problemi (Seismological inverse problem)

Deprem kaynak parametrelerinden telesismik cisim dalgalarının problemi için Niching GA uygulanmıştır (Koper vd., 1999). Dalga biçim inversiyon probleminde bulunan çözümler arasında benzerliği ölçmek için mesafe ölçüm metodu uygulanmıştır.

2.4.11. Monte carlo doğrusal olmayan filtreleme problemi (Monte carlo nonlinear filtering)

Standart Monte Carlo filtrelerinde, tekrar eden örneklemeler rastgele dağıldığı için çeşitli kayıplar yaşanır. Monte Carlo filtreleme algoritmasını geliştirmek için Niching yöntemleri kullanılmıştır (Bienvenüe vd., 2002). Kolmogorov-Smirnov metriği iki olası dağılım arasındaki mesafenin ölçülmesinde kullanılmıştır (Bienvenüe vd., 2002).

2.4.12. Rekabetçi tesislerin konumu ve tasarım problemi (Competitive facilities location and design)

Tesis konum probleminde çoklu global çözümler elde edilmesi gerekir. Universal Evolutionary Global Optimizer, benzetilmiş tavlama ve yeniden başlamalı yöntemlerden daha iyi performans göstermiştir (Juana ve Fernandez ,2009).

2.4.13. Denklem sistemlerinin çözümü (Solving systems of equations)

Lineer denklem sistemlerini çözmek için PSO algoritması geliştirilmiştir (Brits vd., 2002). Niş algoritmaları, çoklu çözümleri olan denklemleri çözmek için uygun bir metottur. Son yıllarda doğrusal olmayan denklemlerin çözümünde niş metotları kullanılmaya başlanmıştır (Song, vd., 2015).

2.4.14. Protein yapısı tahmini (Protein structure prediction)

3D Hidrofobik Polar kafes modelindeki bir protein yapısının tahmin problemi, MMO problemi olarak formüle edilmiştir (Wong vd., 2010). Makalede basit bir niş yapısı bile mevcut durumdan daha iyi sonuçlar verdiği anlatılmıştır.

2.4.15. Elektrikli taşıtlar için asenkron motor tasarım problemi (Induction motor design for electric vehicle)

Asenkron motorların şekil veya yapı tasarımında, çok sayıda optimum profil tanımlamaya ihtiyaç vardır. Sadece geometrik sınırlamalar ve stator bobin sargısı gibi imalat hususlarını dikkate alarak optimizasyon yaptığımızda doğru sonuç alamayabiliriz. Bunu sebebi optimizasyon sırasında matematiksel olarak belirlenemeyen parametrelerdir. Birden fazla optimum sonuç bulup denemeler yapılarak problem çözülmeye çalışılır. Problemin çözümü için sınırlı turnuva seçimi olan bir niş metodu kullanılmıştır (Cho vd., 2001).

3. YAPAY ARI KOLONİSİ (ARTIFICIAL BEE COLONY)

3.1. Optimizasyon Metotlarının Sınıflandırılması

Sezgisel yöntemlerle çözülebilen optimizasyon metotları, klasik ve modern sezgisel metotlar olarak sınıflandırılır (Kartal ve Cura, 2015). Arama uzayını tamamen araştırarak çözüm bulmak için klasik metotlar kullanılabilir.

Modern sezgisel metotlar deterministtik ve olasılıksal olarak ikiye ayrılmaktadır (Kartal ve Cura, 2015). Olasılık temelli metotlar tek ve birden çok iteratif olarak geliştirilen metotlar olarak ikiye ayrılır (Karaboğa ve Akay, 2009). Sezgisel tekniklerde bir sonraki adım bir önceki adıma göre şekillenir. Bu yöntemler zaman olarak tasarruf sağlasa da bize çözümü garanti etmez (Weise, 2009).

Optimizasyon probleminde kullanılan popülasyon tabanlı sezgisel yöntemler, gelişime dayalı ve sürü zekâsı algoritmalar olarak iki sınıfta incelenebilir (Kartal ve Cura, 2015). Gelişime dayalı algoritmalara, evrime ve genetiği dayalı algoritmalar örnek verilebilir. Sürü zekâsı algoritmalarına canlıların davranışından esinlenerek oluşturulan algoritmalar örnek verilebilir.

Sürü zekâsı, sosyal canlıların besin arama süreçlerinin incelenmesiyle ortaya çıkan davranışsal ve sayısal kavramdır (Belal, 2005). Sosyal canlılara; karıncalar, arılar, kuşlar örnek olarak verilebilir (Karaboğa, 2011).

Sürü zekâsı yapay zekânın bir parçasıdır. Sosyal hayvanların kolektif davranışları incelenerek ortaya çıkarılmış metotlardır. Bir yönetici olmadan kendi kendine organize olan davranışlara kolektif davranış denir. Karıncaların kendilerinden çok büyük bir yaprağı birlikte yuvalarına taşımaları bir kolektif davranış örneğidir (Blum ve Merkle, 2008).

Kolektif davranışın temel prensipleri homojenlik, mekân, çarpışmayı önleme, hız eşleme, sürü merkeziyetidir (Grosan vd., 2006).

Sürü zekâsı terimini, sosyal hayvanların kolektif davranışlarından ilham alınarak geliştirildiğinden bu tip metotlarda isim olarak kullanmıştır (Bonabeau ve Dorigo, 1999).

Sürünün zeki olarak tanımlanabilmesi için yakınlık, kalite, dağılım cevabı, kararlılık, uyarnabilirlik prensibine sahip olması gerekir (Kennedy, 2006).

Sosyal hayvanlar anatomik farklılıkları iş bölümünü ortaya çıkarmıştır. Sosyal hayvanların kolektif davranışlarının birçoğu kendi kendine organize olmasından kaynaklamadır.

Kolektif davranışın en önemli parçası kendi kendine organize olmaktır (Bonabeau ve Dorigo, 1999).

Kendi kendine organize olabilmenin kuralları; pozitif ve negatif geri besleme, dalgalanma, çoklu etkileşimdir (Bonabeau ve Dorigo, 1999).

Sürü zekâsında sürü arasındaki iletişim dolaylı ve direk olmak üzere ikiye ayrılır. Direk iletişim sürüdeki canlıların kendi aralarında yaptıkları iletişimdir. Bal arılarının sallanma dansı örnek olarak verilebilir. Dolaylı iletişim sürüdeki canlıların çevre yardımıyla yaptıkları iletişimdir. Dolaylı iletişimde bir uyarıcı vardır. Sürü içerisindeki canlılar uyarıcı sayesinde kendi davranışlarıyla diğer canlıların davranışları arasındaki farkı ayırt ederler. Çevresindeki değişime kolaylıkla adapte olabilirler. Dolaylı iletişime en iyi örnek karıncaların besin kaynağına giderken feromen maddesi salgılamalarıdır (Belal, 2005).

Sürü zekâları birçok optimizasyon probleminin çözümü olan algoritmalar gelişmesine sebep olmuştur. Kuşların göç hareketleri PSO algoritmasının tasarlanmasını sağlarken, karıncaların ve arıların besin arama davranışları ACO ve ABC algoritmasını ortaya çıkarmasına sebep olmuştur (Engelbrecht, 2007).

3.2. Arıların Gerçek Davranışı

Doğada sürü şeklinde yaşayan canlılardan kendi kendine örgütlenme ve iş bölümüne sahip canlılar incelenerek modellenmeye çalışılmıştır. Modeller algoritmaya dönüştürülerek sürü zekâsı algoritmaları oluşturulmuştur. Sürü zekâsı algoritmalından biri de ABC algoritmasıdır.

ABC algoritması, arıların besin arama davranışlarının incelenmesi sonucunda ortaya çıkmıştır.

3.2.1. Gerçek arıların davranışları

Sosyal çevrelerinde yaşayan arılar kendi aralarında bir düzen kurar ve bu düzene göre hareket ederler. Bal arıları yiyeceğin bulunup yuvaya getirilmesini, yiyeceğin saklanmasını ve dağıtılmasını kendi aralarındaki hiyerarşi sayesinde sağlarlar. Arı kolonisindeki işler o iş ile özdeşleşmiş arılar tarafından yapılmaktadır. Arılar kendi arasında iş bölümü yapmış ve kendi kendine organize olabilmektedirler. Arılar minimum enerji harcayarak yiyecek bulmaya hedeflemektedir. Kovan içerisindeki arılar kraliçe arı erkek arı ve işçi arı olmak üzere üç sınıfa ayrılırlar (Karaboğa ve Akay, 2009).

Kraliçe arı

Her kovanda sadece bir adet bulunan kraliçe arı işçi arı ve erkek arıların işleyişi ve koloninin bütünlüğünden sorumludur. Kraliçe arı salgıladığı madde sayesinde kovana giren yabancı arıları fark ederek koloninin korunmasını sağlar. Kraliçe arı kolonideki yumurta üreterek tek arıdır. Koloninin besin kaynağına göre yumurta üretimi yapar. Bu sayede koloninin büyüklüğünü kontrol altında tutar.

Erkek arı

Bir kolonide birden çok erkek arı bulunur. Savunmaya yarar iğneleri bulunmaz. Besin toplayacak organları da bulunmayan erkek arıların görevi kraliçe arı ile çiftleşmektir.

İşçi arı

Kolonide sayıca en fazla olan gruptur. Kolonide, besinlerin toplanmasından, depolanmasından, koloninin güvenliğinden ve ölü arıların temizlenmesinden sorumludur. İşçi arılar da dişi arılardır. Kraliçe arıdan tek farkı üreme yetenekleri yoktur. Bu arılar yaşamlarının ikinci yarısında yiyeceklerin aranmasından sorumludur. Çevreyi gözlemleyerek besinlerin konumunu belirler. Yaz aylarında 6 hafta, kış aylarında 4-9 ay kadar yaşayabilmektedirler.

Kovandaki en önemli görev besin aranmasıdır. Arının kovandan ayrılmasıyla besin arama süreci başlar. Arı yiyecek kaynağını bulduğunda besini karın bölgesine depolar. Bu süreç besinin kovana uzaklığına ve besinin miktarına bağlıdır. Arı depoladığı besini bal yapmaya başlar. Kovanına döndükten sonra oluşturduğu balı boş peteğe aktarır. Petekleri, bakteri ve mikroplardan korumak için bal mumu ile kapatır.

Besin arayıcı arı, balı peteğe bıraktıktan sonra besin kaynağının kovana uzaklığını, besinin kalitesini ve besinin miktarını yaptığı özel dans ile diğer arılara bildirir. Dans sırasında, kovandaki diğer arılar besin arayıcı arının antenine dokunarak bu bilgileri alırlar. Arıların dairesel, kuyruk ve titreme dansı olmak üzere üç çeşit dansı vardır.

Dairesel dans

Kaynak yuvaya 100 metreden daha az uzaklıkta ise yani uzaklığın önemi yoksa yaparlar. Herhangi bir yön bilgisi içermez.

Titreme dansı

Kovanın kapasitesi ve besin getirme görevi arasında ki dengeyi belirlemektir. Arı yavaş tempoda ve düzensiz tarzda bacaklarını titreterek ileri geri sağa ve sola hareket eder. Arı, besin kaynağının çok zengin olduğunu fakat işleyebileceklerinin çok üstünde bir besinin olduğunu ve bir an önce besini taşımaları gerektiğini anlatmaya çalışır.

Kuyruk dansı

Besin toplayıcı arılar, 100 metreden daha uzakta olan besin kaynağı hakkındaki bilgiyi aktarmak için yaparlar. Besin kaynağının yönünü güneşe olan açıyla belirlerler. Her 15 saniyede dansın tekrarlanma sayısı besin kaynağının uzaklığını ifade eder. Kuyruk dansı yapan arı diğer arıların titreşim hareketi yapması sonucu dansına son verir.

Kovan içinde ki işlerin yürütülmesi için belirli sayıdaki arıların belirli görevlere getirilmesine ihtiyaç vardır (Beekman vd., 2007).

Yiyecek arama modellenmesi yiyecek kaynaklarını belirleme, görevli işçi arı ve görevsiz işçi arı adımından oluşur. Sosyal bir canlı olan arılar çevresindeki değişimlere hemen adapte olabilirler. Arılar yaptıkları işlerde uzmanlaşmıştır. Arılar değişime ayak uydurarak bir başka arının yaptığı işi yapabilmektedir.

Bir arının yetişken bir arı olana kadar besleme, depolama, bal ve polenlerin elde edilmesi ve dağıtılması, iletişim ve besin arama gibi görevleri vardır (Kartal ve Cura, 2015).

Bal arıları yiyecek seçerken belirli parametreleri değerlendirir ve yuvaya en uygun yiyecek kaynağını seçmeye çalışırlar. Arıların besin arama davranışlarını Tereshko, reaktif difüzyon denklemleri temel alınarak modellemiştir (Tereshko, 2000). Model besin kaynağı, görevli arılar ve görevsiz arılar olmak üzere üç ana bileşenden oluşur. Besin kaynağına yönelme ve besin kaynağından uzaklaşma olmak üzere iki tane davranış modu vardır (Tereshko ve Loengarov, 2005).

Yiyecek kaynakları

Arıların besin kaynaklardır. Arı besin kaynağı değerlendirilirken besinin çeşidi, besinin yuvaya uzaklığı, besinin miktarı ve besinin çıkarılma kolaylığı gibi birçok parametreyi inceler. Besin kaynağının zenginliği tek bir unsur olarak alınabilir (Seeley, 1995). Arılar besin kaynaklarından en uygun olanı besin kaynağı olarak seçerler.

Görevli işçi arılar

Keşfedilen yiyecek kaynaklarını kovana getirilmesinden sorumludur. Görevli arılar gittikleri yiyecek kaynaklarıyla ilişkilendirilir. Görevli arı, yiyeceğin yuvaya uzaklığı, yönü ve karlılığı hakkındaki bilgiyi taşır ve arılarla paylaşır (Karaboğa, 2005). Besin kaynağının zenginliğine göre görevli arı dans bölgesinde kuyruk dansını yapar. Görev süresi biten arılar görevsiz arı olarak çalışmaya devam ederler.

Görevsiz İşçi Arılar

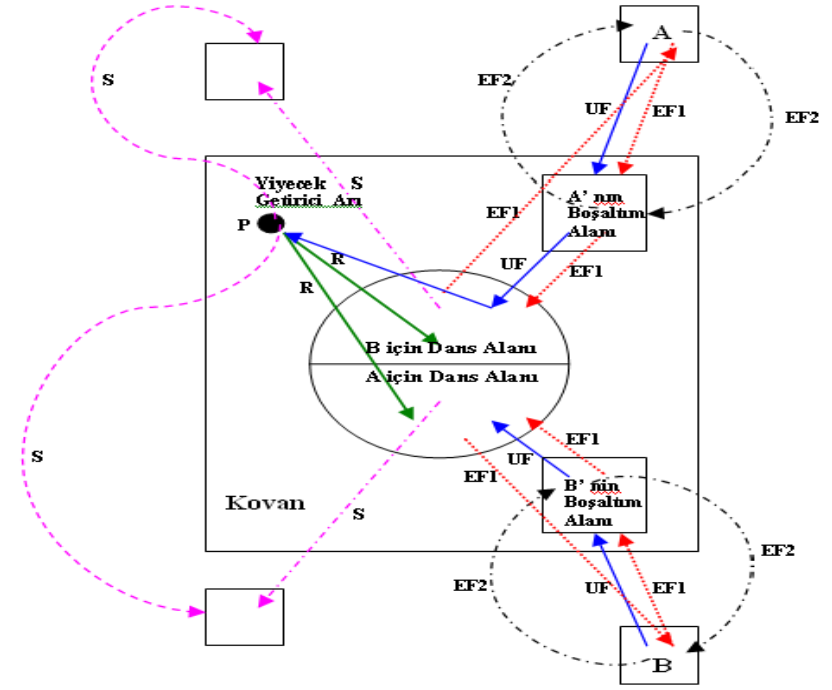
Bu arılar sürekli yeni bir yiyecek kaynağı arayışındadır. Kâşif arı ve gözcü arı olmak üzere iki tip görevsiz işçi arı vardır. Kâşif arı rastgele yiyecek kaymalarına giderek yiyecek arayışındadır. Gözcü arılar kovanda bekleyerek işçi arıların danslarını izler ve işçi arının hareketlerine göre yeni yiyecek kaynağına giderler. Kâşif arı herhangi bir bilgi olmadan besin kaynağı aramaya çıkar. Kâşif arı sayısı ortalama arıların %5-8 ini oluşturur (Karaboğa, 2005).

Bilgi alışverişi arıların yiyecek bulması için çok önemlidir. Dans alanı, kovandaki bilgi alışverişin en önemli kısımdır. Dans alanında arılar dans ederek, besin kaynağının kalitesi hakkında diğer arılara bilgi verirler. Arılar, daire dansı kuyruk dansı ve titreme dansı yaparlar. Dans süresiyle besin kaynağının uzaklığı doğru orantılıdır. Mevcut yiyecek kaynakları izlenerek en uygun yiyecek kaynağı belirlenir. En uygun yiyecek kaynağı en yüksek olasılık değerine sahiptir (Karaboğa, 2005).

Başlangıçta potansiyel yiyecek arayıcı arı işsiz arı olarak araştırmaya başlar. Arı yiyecek kaynaklarının yerinden habersizdir (Karaboğa, 2005). Bu durumda arı için iki durum ortaya çıkar.

1- Arı kâşif arı olarak iç ve dış etkenlere bakarak yiyecek aramaya başlayabilir.
2- Arı kuyruk dansını izleyen gözlemci arı olarak dans kaynağına gidebilir. Arı kaynağı hafızasında tutarak yiyecek kaynaklarını toplamaya başlar. Bu arı yiyeceğin zenginliğine göre görevli arı haline gelmiş olur. Belirli bir miktar besinle yuvaya dönen arı besini boşaltır. Besinleri boşalttıktan sonra 3 ihtimal ortaya çıkar.

a) Kaynağı bırakarak görevsiz işçi arı olur.
b) Besin kaynağına dönmeden dans eder ve kovandaki diğer arıları kaynağa yönlendirir.
c) Arılara haber vermeden kaynağa gider.



Şekil 3.1. Arıların hareketi (Karaboğa, 2011).

3.3. ABC Algoritması

Sosyal davranış sergileyen canlıların besin ararken kurdukları zeki davranışlar problem çözmede ilham kaynağı olmuştur. Derviş Karaboğa, doğadaki arıların besin arama davranışlarını modelleyerek ABC geliştirilmiştir (Karaboğa, 2005). Algoritma bazı varsayımlar içermektedir. Varsayımlardan biri her bir yiyecek kaynağı için sadece bir görevli arı bulunmasıdır. Bu varsayımdan dolayı toplam yiyecek kaynağı sayısı görevli arıların sayısına eşittir (Karaboğa ve Akay, 2007). Diğer varsayım da gözcü arıların sayısı görevli arıların sayısına eşit olmasıdır. Besin kaynaklarının yerleri optimizasyon problemindeki olası çözümlere denk gelirken kaynaktaki besin zenginliği, çözümlerin kalitesine karşılık gelmektedir (Karaboğa, 2011). ABC algoritmasında, koloni de üç farklı tipte arı bulunmaktadır. Bunlar işçi arılar, gözcü arılar ve kâşif arılardır. ABC algoritmasında görevli ve görevsiz arıların sayısı eşittir. İşçi arının sayısı gözcü arıya eşittir. Görevli arının besin kaynağı biterse arı kâşif arı olur (Karaboğa, 2005).

ABC algoritmasının adımları aşağıdaki gibidir.

Adım 1: ABC algoritmasında besin kaynakları rastgele oluşturulur. İşçi ve gözcü arı sayısı, besin kaynağı sayısına göre belirlenir. Limit değişkeni belirlenir. Algoritmanın çalışma sayısını tutan çalışma sayısı değişkeni tanımlanır.

Adım 2: Besin kaynaklarındaki besinlerin değeri problemin amaç fonksiyonuna göre hesaplanır.

Adım 3: İşçi arılar rastgele besin kaynaklara yönelerek besini işler ve besinin yeni değerini hesaplanır. Yeni değer kayıtlı olan diğer değerlerden daha iyiyse hafızaya alınır. Hafızaya alınan çözüm değeri değiştiyse limit değeri sıfırlanır değişmediyse limit bir arttırılır.

Adım 4: Gözcü arı aşaması işçi arı aşamasından sonra başlar. Besin uygunluk değerine göre besin seçilir değerlendirilir ve daha önceki değerlerden iyiyse hafızadakiyle değiştirilir. Hafızaya alınan çözüm değerinde değişim olduysa limit değeri sıfırlanır değişim olmadıysa limit bir arttırılır. Gözcü arı aşamasının işçi arı aşamasından farkı seçim işleminin uygunluk değerine göre yapılmasıdır.

Adım 5: Gözcü arılardan sonra kâşif arı aşaması başlar. Kâşif arı aşaması algoritmanın local minimum veya maksimumda takılmasına engel olması içindir. Çözüm ve limit sıfırlanıp tekrar çözüm elde edilir. Çözümler karşılaştırılır iyi olan hafızada tutulur.

Adım 6: Algoritma başında belirlenen döngü sayısına ulaşılıncaya kadar işçi arı, gözcü arı ve kâşif arı aşaması devam eder. Döngü sayısı bitiminde önce durdurma koşulu sağlanırsa algoritma sonlandırılır.

3.3.1. Yiyecek kaynağı bölgelerinin üretilmesi

Algoritma, arama uzayında çözümlere karşılık gelebilecek yiyecek kaynağı yerlerini rastgele üreterek başlar. Rastgele yiyecek kaynağı üretilirken aşağıdaki formülden yararlanılır.

$$x_{ij} = x_j^{min} + raund(0,1)(x_j^{max} - x_j^{min}) \tag{3.1}$$

Burada i besin kaynağı sayısını j de parametre sayısını belirtmektedir. x_j^{min} parametrelerin alt sınırı x_j^{max}parametrenin üst sınırıdır. Belirlenen alt ve üst sınır aralığında besin kaynakları üretiliş olur.

3.3.2. İşçi arıların yiyecek kaynağı bölgelerine gönderilmesi

İşçi arı yiyecek kaynağına yakın yeni bir yiyecek kaynağı belirleyerek kalitesini değerlendirilir ve hafızaya alır. İşçi arılar besin kaynaklarına yönelip problemin amacına göre sonucu değerlendirir ve bilgilerini güncelleyip hafızaya alır. Çözüm değerini iyileştiren değerler hafıza da tutulur. Yeni kaynağın belirlenmesinde aşağıdaki formülden yararlanılır.

$$V_{ij} = X_{ij} + \Phi_{ij}(X_{ij} - X_{kj}) \tag{3.2}$$

X_{ij} mevcut kaynaktır. V_{ij} yeni besin kaynağı, X_{ij} kaynağının parametresinin değiştirilmesi sonucunda bulunur. J değeri sıfırla parametre değeri arasında rastgele bir sayıdır. K değeri sıfırla yiyecek kaynağı değeri arasında rastgele bir sayıdır. Φ değeri (-1 1) arasında rastgele sayıdır.

X_{ij} ve X_{kj} arasındaki fark azaldıkça X_{ij} parametresi değerinin değişimi azalacaktır. V_{ij} değeri daha önceden belirlenen değerler arasında yer alması için aşağıdaki denklemden yararlanılır.

$$V_{ij} = \begin{cases} X_j^{min}, & V_{ij} < X_j^{min} \\ V_{ij}, & X_j^{min} \leq V_{ij} \leq X_j^{max} \\ X_j^{max}, & V_{ij} > X_j^{max} \end{cases} \quad (3.3)$$

V_{ij} yeni bir besin kaynağını temsil etmektedir. Yeni besin kaynağının kalitesi hesaplanarak uygunluk değeri atanır. Bir besin kaynağının uygunluk değeri (uf_i) aşağıdaki denkleme göre belirlenir.

$$uf_i = \begin{cases} \frac{1}{1+f_i}, & fi \geq 0 \\ 1 + abs(f_i), & fi < 0 \end{cases} \quad (3.4)$$

fi besin kaynağının amaç fonksiyon değeridir. Problem türüne göre uygunluk hesap yöntemi farklılık gösterebilir. Çözüm değeri bulunan uygunluk değerine göre hesaplanır. Uygunluk değeri daha önce bulunan uygunluk değerlerinden daha iyiyse eski değerle değiştirilir. Uygunluk değeri değişmediğinde sayaç bir arttırılır.

3.3.3. Gözcü arıların kullanacakları olasılık değerinin hesaplanması

İşçi arılar çevredeki araştırmalarını tamamladıktan sonra kovana dönerek dans alanında dansına başlar. Gözcü arılar dans bilgisinden yararlanarak yiyecek miktarıyla orantılı olarak yiyecek kaynağı seçerler. Olasılıksal yiyecek kaynağı seçim yöntemi yiyecek miktarına karşılık gelen uygunluk değerlerine göre yapılmaktadır. Uygunluk değerine bağlı yiyecek kaynağının seçilmesinde çeşitli metotlar vardır. Bunlardan bazıları; rulet tekerleği, sıralamaya dayalı seçim, skolastik örnekleme ve turnuva yöntemleridir (Akay, 2009b). ABC rulet sistemini kullanarak hesaplamıştır.

Rulet sisteminin denklemi aşağıdaki gibidir.

$$p_i = \frac{uf_i}{\sum_{n=1}^{UD} uf_n} \tag{3.5}$$

Hesaplanan uygunluk değerinin toplam uygunluk değerine oranlanmasıyla rulet tekerleğindeki genişlikler elde edilmiş olur. Kaynağın yiyecek miktarı artıkça kaynak bölgesine seçilecek gözcü arı sayısı da artacaktır.

3.3.4. Gözcü arıların yiyecek kaynağı bölgesini seçmesi

Rulet tekerleğinden seçilen yiyecek kaynağına gözcü arı giderek yeni yiyecek kaynakları bulmaya çalışır. Gözcü arılar yiyecek kaynaklarını ararken işçi arı aşamasında kullanılan yiyecek kaynağı formülünden yararlanır (Formül 3.2). Yiyecek kaynağı gözcü arı tarafından değerlendirilir ve değeri hesaplanır. Hesaplanan yeni değer eski değerden daha iyiyse hafızaya alınır ve gelişim sayaç sıfırlanır, daha kötüyse sayaç arttırılır. Gözcü arılar, besin kaynaklarına gidene kadar süreç devam eder.

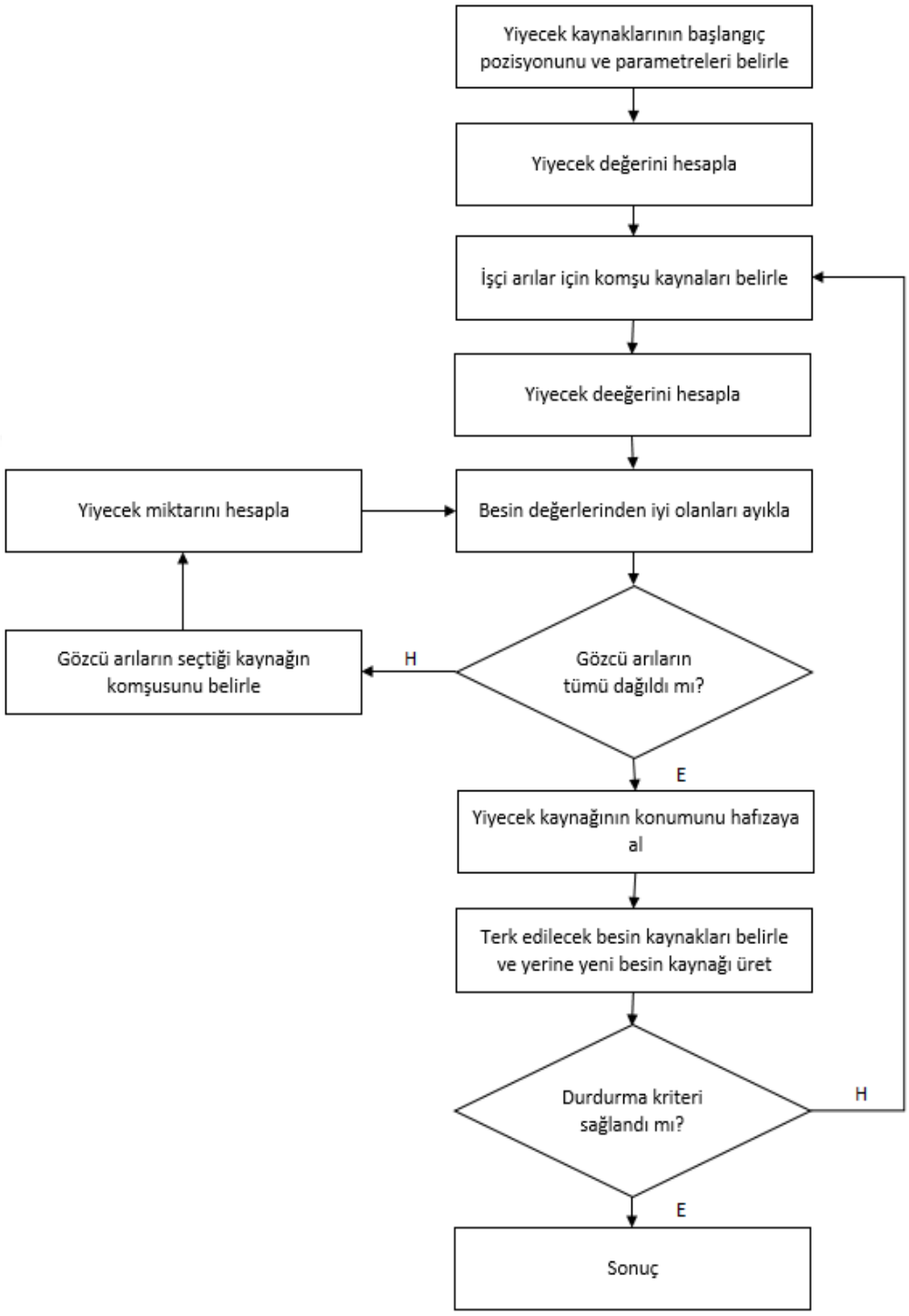
3.3.5. Yiyecek kaynağının terk edilmesi ve kâşif arı aşaması

İşçi ve gözcü arıların besin kaynaklarına gönderilmesi tamamlandıktan sonra gelişim sayacı kontrol edilir. Gelişim sayacı kaynaktan faydalanıp faydalanamadığı hakkında bize bilgi verir. Gelişim değeri parametre değerinden daha büyükse çözümün daha fazla iyileşemediğini gösterir. Gelişim değeri parametre değerini geçtiği durumda kâşif arılar yeni besin kaynağı oluşturup kaynağın besin değerini hesaplar. Yeni kaynağın besin değeri daha önceki besin değerinden iyiyse hafızaya alınır kötüyse ihmal edilir. Algoritmanın adımları belirtilen durma koşulları sağlanıncaya kadar devam eder.

3.3.6. Algoritmanın çalışmasına son verilmesi

Algoritmanın çalıştırılmasının sonlandırılmasında çeşitli parametreler etken olabilir. Algoritma, önceden belirlenen belirli bir iterasyon sayısına kadar veya fonksiyonun çağırma sayısına kadar çalışması sonucunda sonlandırılır. Eğer algoritma local ve global optimum nokta sayısını bulursa veya belirli bir hata eşik değerine ulaşırsa ilk iki durum gerçekleşmesi beklenmeden sonlandırılabilir.

Algoritma akış şeması şekil 3.2 de gösterilmiştir.



Şekil 3.2. ABC algoritmasının akış şeması.

4. ELİT YİYECEK KAYNAKLI ADAPTİF YAPAY ARI KOLONİSİ ALGORİTMASI

Multimodal problem örneklerini çözerken popülasyon bireylerinin tüm yerel ve küresel (global) optimum noktalarda konumlanması gerekmektedir. Klasik ABC algoritması multimodal problemlerin çözümünde gerekli başarıyı gösterememektedir. Bunun en önemli nedenlerinden nedeni multimodal problemlerinin çözümünde algoritmanın hem birçok global ve yerel optimum noktayı tarayabilme kabiliyetinin olması hem de bulunan değerli noktalarda odaklanmayı sağlayacak güçlü stratejilerinin olması gerekmektedir. Klasik ABC algoritması içerisinde önerilen arama denklemi ve açgözlü seçme mekanizması bu iki gereksinimi yerine getirememektedir. Çünkü klasik ABC algoritması için önerilen arama denklemi her ne kadar keşif gücü yüksek olsa da odaklanma gücü zayıftır. Öte yandan açgözlü seçme mekanizması yeni bireylerin eski bireylerden iyi olduğu durumlarda eski bireyi popülasyondan silmektedir. Multimodal problemlerde birden çok optimum ve yerel optimum nokta bulunduğundan eski bireyin bir yerel optimum noktada olabileceği göz ardı edilmektedir. Dolayısıyla gerçekte bulunmuş olma ihtimali olan optimum noktalar algoritmanın kendi mekanizması nedeniyle unutulabilmektedir.

Bu tezde algoritmanın dezavantajlarını ortadan kaldırmak için iki yeni strateji geliştirilmiştir. Bunlardan birincisi üç arama denklemi içeren yeni bir adaptif arama denklemi seçim stratejisidir. Diğeri ise yerel ve global optimum bilgilerin kaydedilmesine yardımcı olacak elit yiyecek kaynakları ile çalışma stratejisidir. Bu yüzden geliştirmiş olduğumuz yeni ABC varyantına "Elit Yiyecek Kaynaklı Adaptif Yapay Arı Kolonisi Algoritması" (SABCElit) ismini verdik. Aşağıdaki alt bölümlerde bu tezde gerçekleştirmiş olduğumuz yeni ABC algoritması için önerdiğimiz stratejiler değerlendirilmiştir.

4.1. Adaptif Arama Denklemi Stratejisi

Klasik ABC algoritmasında Denklem 3.1 de gösterilen arama denklemi kullanılmakta olup algoritma süresince bu denklem değişmemektedir. Fakat her bir problem için birbirinden farklı arama denklemleri gereklidir. Bu yüzden klasik ABC algoritmasında kullanılan arama denklemi ile birlikte multimodal problemlerin çözümünde etkin kullanılabilecek iki yeni arama denklemi bu tezde önerilmiştir. Bu denklemler aşağıdaki gibidir:

$$v_{i,j} = x_{i,j} + \phi_{i,j}(x_{i,j} - gbd_j) \tag{4.1}$$

$$v_{i,j} = x_{i,j} + \phi_{i,j}(nbest_{i,j} - gbd_j) \tag{4.2}$$

Burada gbd_j global en yakın komşunun j. boyuttaki değeri, $nbest_j$ yiyecek kaynağının en iyi komşusunun j. boyuttaki değeri ve $\phi_{i,j}$ rastgele bir reel değer olup $[-a,+a]$ arasında değer almaktadır ve a değeri her bir denklem örneği için rastgele üretilen bir pozitif reel sayıdır.

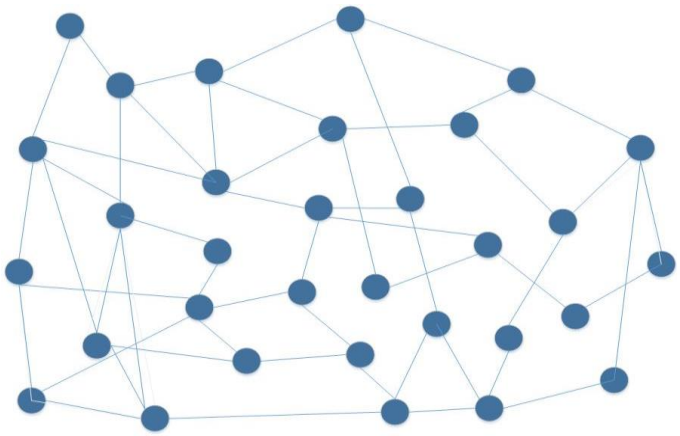
Global en yakın komşunun (gbd) seçilmesi olasılığı işlemi aşağıdaki denklem kullanılarak gerçekleşmektedir.

$$p_{gbd} = \frac{\frac{1}{dist(x_i - gbd)}}{\sum_{n=1,n\neq i}^{SN} \frac{1}{dist(x_i - x_n)}} \tag{4.3}$$

Burada $dist(x_i - x_n)$ seçilen yiyecek kaynağı ile diğer bir yiyecek kaynağı arasındaki öklit uzaklığını ifade etmektedir.

En iyi komşunun ($nbest$) seçilmesinde ise seçilen yiyecek kaynağının komşuluk topolojisi ön plana çıkmaktadır. Klasik ABC algoritmasında komşuluk topolojisi bulunmadığından tüm yiyecek kaynakları komşu olarak kabul edilir ve bu durumda $nbest$ global en iyi çözümü ifade eder. Fakat bu çalışmada, klasik ABC yönteminden farklı olarak her iterasyonda kendini güncelleyen bir rasgele komşuluk topolojisi kullanılmıştır. Bu komşuluk topolojisinde her bir yiyecek kaynağının maksimum 4/SN kadar komşusu bulunmaktadır ve $nbest$ bu topolojideki en iyi bireyi ifade eder. Örnek bir topoloji görünümü şekil 4.1 de gösterilmiştir. Her bir iterasyonda her bir yiyecek kaynağı için komşuluk topolojisi değiştiğinden bir iterasyon denklem için kullanılan $nbest$ yiyecek kaynağı güncellenmiş olur. Böylece yiyecek kaynakları her adımda iyi bir yiyecek kaynağına yaklaşarak odaklanmayı sağlamaktadır. Öte yandan komşu sayısının kısıtlı olması keşfetme gücüne katkı sağlamaktadır.

Yine klasik ABC algoritmasındaki arama denkleminde yiyecek kaynağının rasgele seçilen bir boyutu üzerinde işlem yapılmaktadır. Fakat yapılan çalışmalarda (Akay ve Karaboğa, 2012; Aydın vd., 2017) multimodal problem türleri için bir defada birden çok boyutun değiştirilmesi gerektiği görülmüştür. Dolayısıyla bu tezde önerilen üç arama denklemi bir yiyecek kaynağının birden çok boyutunu değiştirebilecek şekilde tasarlanmıştır. Her bir denklemin kaç boyutta değişiklik yapacağı m ($1 \leq m \leq D$) parametresi ile belirlenmiş olup bu değer her bir denklem için rasgele verilmektedir.



Şekil 4.1. Bir rasgele komşuluk topolojisi örneği. Noktalar yiyecek kaynaklarını çizgiler ise komşulukları ifade etmektedir.

Çizelge 4.1. 10 büyüklüklü bir arama denklemi havuzu örneği.

Arama Denklemi	m	se
$v_{i,j} = x_{i,j} + \phi_{i,j}(x_{i,j} - gbd_j)$	*1*	*0*
$v_{i,j} = x_{i,j} + \phi_{i,j}(nbest_{i,j} - gbd_j)$	*D*	*0*
$v_{i,j} = x_{i,j} + \phi_{i,j}(nbest_{i,j} - gbd_j)$	*7*	*0*
$v_{i,j} = x_{i,j} + \phi_{i,j}(x_{i,j} - gbd_j)$	*3*	*0*
$v_{i,j} = x_{i,j} + \phi_{i,j}(x_{i,j} - x_{r,j})$	*1*	*0*
$v_{i,j} = x_{i,j} + \phi_{i,j}(x_{i,j} - gbd_j)$	*D*	*0*
$v_{i,j} = x_{i,j} + \phi_{i,j}(x_{i,j} - x_{r,j})$	*D*	*0*
$v_{i,j} = x_{i,j} + \phi_{i,j}(x_{i,j} - gbd_j)$	*2*	*0*
$v_{i,j} = x_{i,j} + \phi_{i,j}(nbest_{i,j} - gbd_j)$	*1*	*0*
$v_{i,j} = x_{i,j} + \phi_{i,j}(x_{i,j} - gbd_j)$	*5*	*0*

Bu tezde önerilmiş olan her biri farklı *m* değerlerine sahip olacak şekilde rasgele sırada bir arama denklemi havuzuna atılmaktadır. Ayrıca her arama denklemine ait başarı değeri sıfır olarak ilklenir. Havuzdaki arama denklemlerinden her biri sırasıyla algoritmanın bir iterasyonunda kullanılır. Eğer seçilen arama denklemi ile bir yiyecek kaynağının konumu iyileştirilirse bu kaydedilerek iterasyon sonunda toplam iyileştirmelerin miktarı hesaplanır. Daha sonra o arama denklemine ait başarı değeri aşağıdaki formül kullanılarak güncellenir;

$$se_s = se_s \cdot \frac{(CFES - MAXFES)}{MAXFES} + c_s \tag{4.4}$$

Burada se_s s. sıradaki arama denkleminin başarı değeri, c_s s. sıradaki arama denkleminin ilgili iterasyondaki iyileştirdiği yiyecek kaynağı sayısı, *CFES* ise o anki fonksiyon çağırma sayısıdır. Böylece algoritma sonlara yaklaştıkça önceki başarı değerini daha çok hesaba katmaktadır. Havuzdaki tüm arama denklemleri kullanıldığında havuzdaki denklemler başarı değerlerine göre sıralanır ve havuzun boyutu (*ps*) aşağıdaki denklem kullanılarak küçültülür.

$$ps = \frac{ps^2}{itr_{MAX}} \tag{4.5}$$

Burada itr_{MAX} maksimum iterasyon sayısını ifade etmektedir. Bu yaklaşım sadece probleme uygun olmayan arama denklemleri elimine edilmiş olur. Böylece adaptif olarak probleme uygun arama denklemleri iterasyonlar ilerledikçe belirlenmektedir.

4.2. Elit Yiyecek Kaynakları ile Çalışma Stratejisi

Multimodal problemlerde birden çok yerel ve global optimum noktanın bulunmaktadır. Algoritma bu noktalara yakın noktalara erişim sağlasa da aç gözlü seçim kuralı nedeniyle bu noktalar unutulabilmektedir. Örneğin bir yerel minimuma erişmiş bir yiyecek kaynağı sonraki iterasyonlarda belli oranda uzak bir konumdaki bir global optimum noktasını bulduğunda konumunu bu global noktaya taşımakta ve önceden keşfettiği yerel minimum noktayı unutmaktadır. Fakat multimodal problemlerde birbirine belli oranda uzak olan tüm yerel ve optimum noktalarının korunması gereklidir. Bu tezdeki çalışmada, algoritmanın çalıştırılması esnasında belli uzaklıktaki iyi çözümlerin saklanarak bu çözümlere "elit yiyecek kaynağı" olarak adlandırıldığı bir strateji geliştirilmiştir. Bu stratejinin temel hedefi elit yiyecek kaynaklarının nasıl belirleneceği ve listenin nasıl güncelleneceği ve saklanan elit yiyecek kaynaklarının nasıl kullanıldığıdır. Aşağıda bu stratejinin ayrıntılarına yer verilmiştir.

4.3. Elit Yiyecek Kaynaklarının Belirlenmesi

Multimodal problemlerin çözümünde tüm yerel ve global optimum noktaların belirlenmesi gerektiğinden birbirinden farklı uzaklıklardaki iyi çözümlerin kaydedilmesi gerekmektedir. Bu tezde bu tür çözümlere elit çözümler adı verilmiştir. Elit çözümlerin saklandığı listeye de elit listesi denilmiştir. Elit listesi her iterasyon sonunda popülasyon elemanları değerlendirilerek güncellenir. Optimum nokta bilinmediği için elit çözümlerin seçiminde mevcut bulunan en iyi çözüm referans alınır. Ayrıca her bir elit çözüm adayı mevcut elit listesindeki

çözümlerden daha kötü olmamalı ve bu çözümlerden belirli bir mesafe uzakta olması gerekmektedir. Dolayısıyla bir elit çözümün belirlenmesinde iki parametre önemli rol oynar. Bunlardan bir tanesi popülasyondaki aday yiyecek kaynaklarının en iyi çözüme ne kadar yaklaştıklarında elit olarak kabul edilebileceğini gösteren epsilon (ε) parametresi, diğeri de elit listesindeki tüm elit yiyecek kaynaklarından ne kadar uzak olması gerektiğini gösteren yarıçap (*r*) parametresidir. Bu parametre değerleri aynı zamanda multimodal problem kümesinde yerel optimum noktaların belirlenmesinde kriter olarak verildiğinden bu parametrelerin değerleri problemde belirtilen değerlere göre atanır.

Buna göre, her bir iterasyon sonunda yiyecek kaynaklarının amaç fonksiyonları, $f(x_i)$, değerlendirilir. Her bir yiyecek kaynağının amaç fonksiyon değeri, en iyi yiyecek kaynağı x_{best}'in amaç fonksiyon değerine epsilon'dan daha az olacak şekilde yakın bir değerse ($|(f(x_{best}) - f(x_i)| < \varepsilon$) aday elit yiyecek kaynakları içerisine alınır. Daha sonra aday elit yiyecek kaynakları elit listesindeki mevcut elit yiyecek kaynakları ile karşılaştırılır. Eğer bir aday elit yiyecek kaynağı mevcut listedeki tüm yiyecek kaynaklarından en az *r* kadar öklit uzaklığında ise bu yiyecek kaynağı elit yiyecek kaynaklarına eklenir.

4.4. Elit Yiyecek Kaynaklarının Güncellenmesi

Her iterasyon sonunda elit yiyecek kaynaklarına yeni elit çözümler eklenebilir veya algoritmanın o ana kadar bulmuş olduğu en iyi çözüm, x_{best}, değişebilir. Bu durumda mevcut listedeki elit çözümlerin amaç fonksiyon değeri o anki en iyi çözümün, x_{best}, amaç fonksiyon değerinden farkı epsilondan fazla olabilir. Bu durumda bu yiyecek kaynakları artık elit olamayacaklardır. Elit yiyecek kaynakları listesi güncellenirken bu durum tüm elit yiyecek kaynakları için kontrol edilir. $|(f(x_{best}) - f(x_i)| < \varepsilon$ koşulunu sağlayamayan yiyecek kaynakları listeden çıkartılarak elenir.

4.5. Elit Yiyecek Kaynaklarının Kullanılması

Elit yiyecek kaynakları listesi hem en iyi aday çözümleri tutarken aynı zamanda algoritma başarısının artmasına yardımcı olmaktadır. Çünkü elit yiyecek kaynakları ayrıca sistemde o ana kadar öğrenilmiş en iyi bilgileri taşıdığından sürüdeki uzman bireylerdir. SABCElit algoritmasında algoritma durağanlığa girdiğinde belli bir sayıda yeni yiyecek kaynağı popülasyona eklenerek popülasyon boyutunun büyütülmesi sağlanır. Her bir yeni eleman (tecrübesiz) üretilirken elit bireylerin (uzmanların) bilgilerinden yararlanılır. Bunun için arama

uzayında rasgele üretilen yeni bireyin konumu rasgele seçilen elit bireylerden birinin konuma aşağıdaki formül yardımıyla ötelenir:

$$x_{i,j} = x'_{i,j} + \phi_{i,j}(x'_{i,j} - x_{elit,j}) \tag{4.6}$$

Burada $x'_{i,j}$ rasgele üretilen bir yiyecek kaynağının j. boyuttaki değerini, $x_{i,j}$ yiyecek kaynağının yeni j. boyuttaki konumu, x_{elit} ise elit listesinden rasgele seçilen bir elit yiyecek kaynağını ifade etmektedir. Bu şekilde yeni yiyecek kaynaklarının üretilmesi, iyi çözümlere hızlı bir şekilde yakınlaşma sağlamaktadır.

Algoritmanın durağanlığa girmesi durumu maksimum durağanlık limiti ($stagnationLimit$) parametresi ile kontrol edilir. Eğer algoritma $stagnationLimit$ iterasyon sayısında en iyi çözümde iyileşme gösteremezse bu durumda algoritmanın durağan olduğu kabul edilir ve popülasyona yeni bireyler eklenir.

Elit çözümler ile yiyecek kaynakları eklendikçe popülasyon boyutu önemli bir şekilde artmaktadır. Bu durumda algoritma her iterasyonda önemli oranda amaç fonksiyonu deneme sayısı bütçesini ciddi oranda tüketmektedir. Dolayısıyla bireyler belli bir iterasyon kadar görevlerini yaptıkça bazı bireyler popülasyondan çıkartılır. Bunun için, her popülasyon tekrar ilkleme periyoduna ulaşıldığında ($RPop_{itr}$) popülasyon boyutu mevcut popülasyon sayısının dörtte birine indirilir. Popülasyonda kalan bireyler en iyi bireylerdir. Popülasyon boyutu düşürülürken popülasyonun en az boyutu 10 kabul edilir ve popülasyon boyutunun 10 değerinin altına düşmesine izin verilmez. Bu algoritmamız için bizim koyduğumuz bir kuraldır. Ayrıca $stagnationLimit$ değeri her zaman için $RPop_{itr}$ değerinden küçük tutulur. Böylece popülasyona yeni birey ekleme oranı genellikle popülasyondan eleman çıkarma oranından yüksek tutulmuş olur. Sonuç olarak algoritmada popülasyon boyutu elastik bir yapıda olup çalıştırılma süreci boyunda değişkenlik gösterir.

5. DENEY TASARIMI

5.1. Problem Seti

Popülasyon tabanlı algoritmalar genel olarak tek bir optimum nokta bulmak için tasarlanmıştır. Gerçek dünya problemlerinde birden fazla optimum çözüm olabilir. Çoklu optimum sonucu olan problemleri çözmek için birçok metot geliştirilmiştir. Geliştirilen metotlarla çözülmeye çalışılan problem setleri iki veya üç boyutlu oldukları için metotların sonuçlarını değerlendirmek zordur. Bazı niş yöntemleri ayarlanması zor olan parametreler sundukları için problemlerin bu metotlarla kullanılması zordur. Multimodal problemlerin çözüm metotları kıyaslanırken farklı test fonksiyonları veya aynı test fonksiyonlarının farklı parametreleri kullanıldığı için doğru kıyaslama sonucu vermemektedir.

Multimodal metotların kıyaslanabilmesi için 20 adet test fonksiyonu bir araya getirilmiştir. Test fonksiyonları, yeni çalışmalara dayanan, iyi bilinen ve yaygın olarak kullanılan fonksiyonlardan ve IEEE CEC 2005 karar verilen fonksiyonlardan oluşur.

Algoritmada kullanılan test fonksiyonları aşağıdaki gibidir (Li vd., 2013).

F1 Five-Uneven-Peak Trap (1D)

F2 Equal Maxima (1D)

F3 Uneven Decreasing Maxima (1D)

F4 Himmelblau (2D)

F5 Six-Hump Camel Back (2D)

F6 Shubert (2D, 3D)

F7 Vincent (2D, 3D)

F8 Modified Rastrigin - All Global Optima (2D)

F9 Composition Function 1 (2D)

F10 Composition Function 2 (2D)

F11 Composition Function 3 (2D, 3D, 5D, 10D)

F12 Composition Function 4 (3D, 5D, 10D, 20D)

Tüm test fonksiyonları aşağıdaki özelliklere sahiptir.

- Tüm test fonksiyonları maksimizasyon problemini içerir.
- F1, F2 ve F3 bir boyutlu multimodal ölçeklendirilebilir fonksiyondur.
- F4 ve F5 iki boyutlu multimodal ölçeklenemeyen fonksiyondur.
- F6 fonksiyonundan F8 fonksiyonuna kadar olan fonksiyonlar ölçeklenebilir multimodal fonksiyondur. F6 ve F7 fonksiyonun optima sayısı D boyutu ile belirlenir. Kullanıcı tarafında kontrol edilmesi gerekir.
- F9 ila F12 farklı özelliklere sahip birkaç temel fonksiyon tarafından oluşturulan ölçeklenebilir multimodal fonksiyonlardır. F9 ve F10 simetrik olmayan ve ayrılabilen fonksiyonlarken, F11 ve F12 simetrik olmayan ve ayrılamayan fonksiyonlardır. Tüm kompozisyon fonksiyonlarındaki global optimal sayısı D den bağımsızdır.

5.1.1. F1: Five-uneven-peak trap (1D)

Probleme ait fonksiyon, özellik ve grafik bilgisi aşağıdaki gibidir.

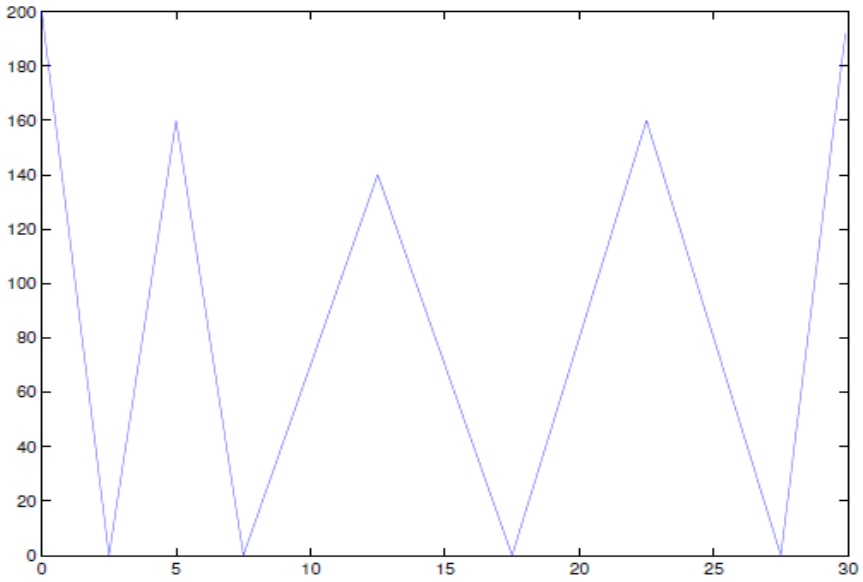
Problemin fonksiyonu

$$F1(x) = \begin{cases} 80(2.5 - x) & for\ 0 \le x \le 2.5, \\ 64(x - 2.5) & for\ 2.5 \le x \le 5.0, \\ 64(7.5 - x) & for\ 5 \le x \le 7.5, \\ 28(x - 7.5) & for\ 7.5 \le x \le 12.5, \\ 28(17.5 - x) & for\ 12.5 \le x \le 17.5, \\ 32(x - 17.5) & for\ 17.5 \le x \le 22.5, \\ 32(27.5 - x) & for\ 22.5 \le x \le 27.5, \\ 80(x - 27.5) & for\ 27.5 \le x \le 30, \end{cases} \quad (5.1)$$

Problemin özellikleri

- X değeri 0 ile 30 arasında olmalıdır. x ϵ (0,30);
- Global optima sayısı:2
- Local optima sayısı :3

Problemin grafiği



Şekil 5.1. Five-uneven-peak trap grafiği (Li vd., 2013).

5.1.2. F2: Equal maxima (1D)

Probleme ait fonksiyon, özellik ve grafik bilgisi aşağıdaki gibidir.

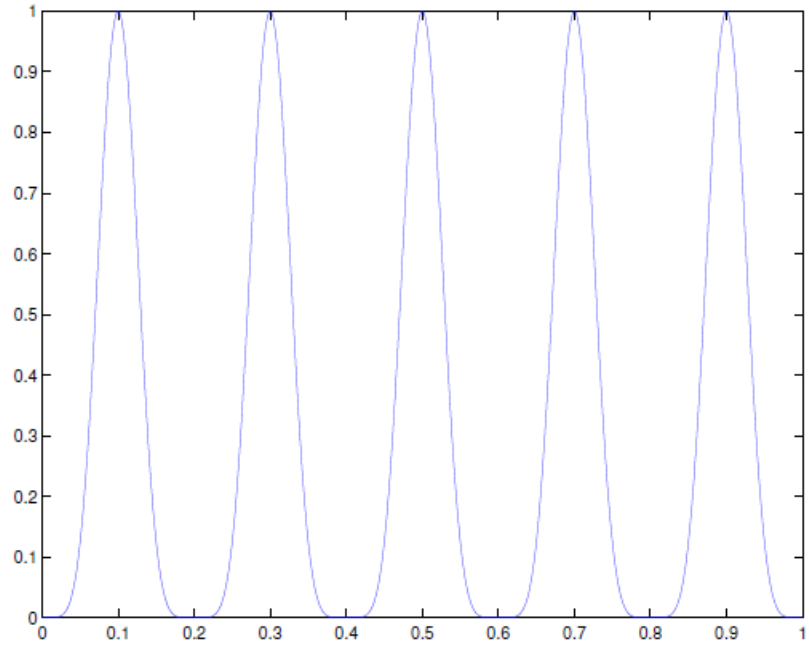
Problemin fonksiyonu

$$F2(x) = sin^6(5\pi x) \tag{5.2}$$

Problemin özellikleri

- X değeri 0 ile 1 arasında olmalıdır. x ∈ (0,1);
- Global optima sayısı:5
- Local optima sayısı :0

Problemin grafiği



Şekil 5.2. Equal maxima grafiği (Li vd., 2013).

5.1.3. F3: Uneven decreasing maxima (1D)

Probleme ait fonksiyon, özellik ve grafik bilgisi aşağıdaki gibidir.

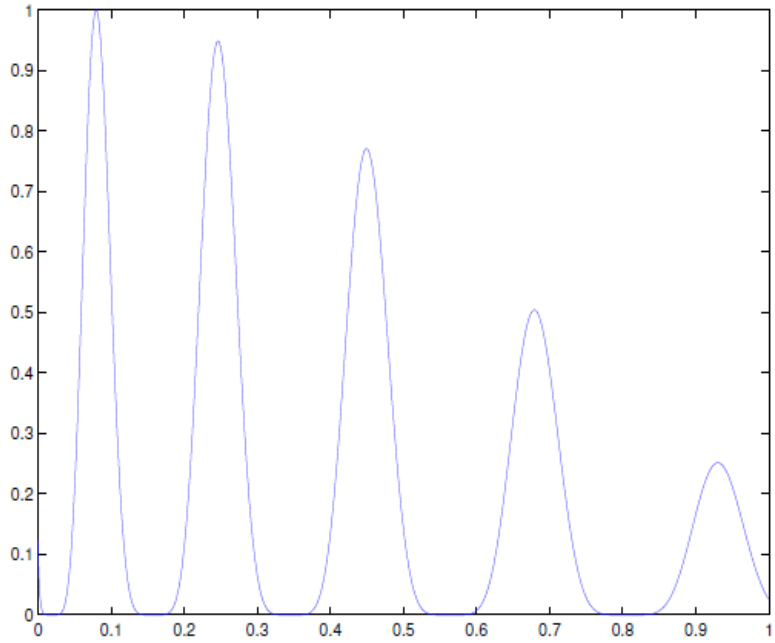
Problemin fonksiyonu

$$F(x) = \exp\left(-2\log\left(2\frac{x-0.08}{0.854}\right)^2\right)sin^6(5\pi(x^{\frac{3}{4}} - 0.05)) \quad (5.3)$$

Problemin özelliği

- X değeri 0 ile 1 arasında olmalıdır. x ∈ (0,1);
- Global optima sayısı:1
- Local optima sayısı :4

Problemin grafiği



Şekil 5.3. Uneven decreasing maxima grafiği (Li vd., 2013).

5.1.4. F4: Himmelblau (2D)

Probleme ait fonksiyon, özellik ve grafik bilgisi aşağıdaki gibidir.

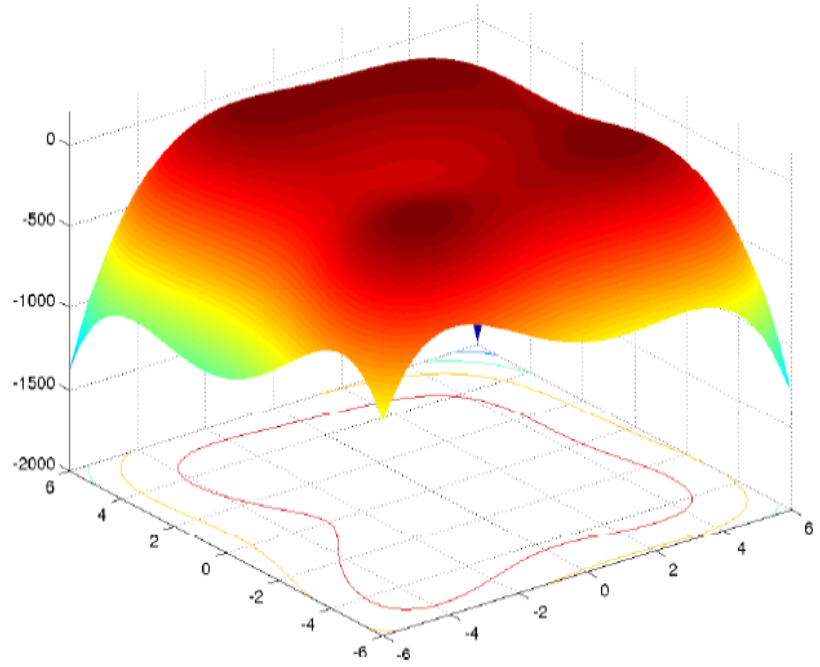
Problemin fonksiyonu

$$F(x) = 200 - (x^2 + y - 11)^2 - (y^2 + x - 7)^2 \quad (5.4)$$

Problemin fonksiyonu

- X değeri -6 ile 6 arasında olmalıdır. x, y ∈ (-6,6);
- Global optima sayısı:4
- Local optima sayısı :0

Problemin grafiği



Şekil 5.4. Himmelblau grafiği (Li vd., 2013).

5.1.5. F5: Six-hump camel back (2D)

Probleme ait fonksiyon, özellik ve grafik bilgisi aşağıdaki gibidir.

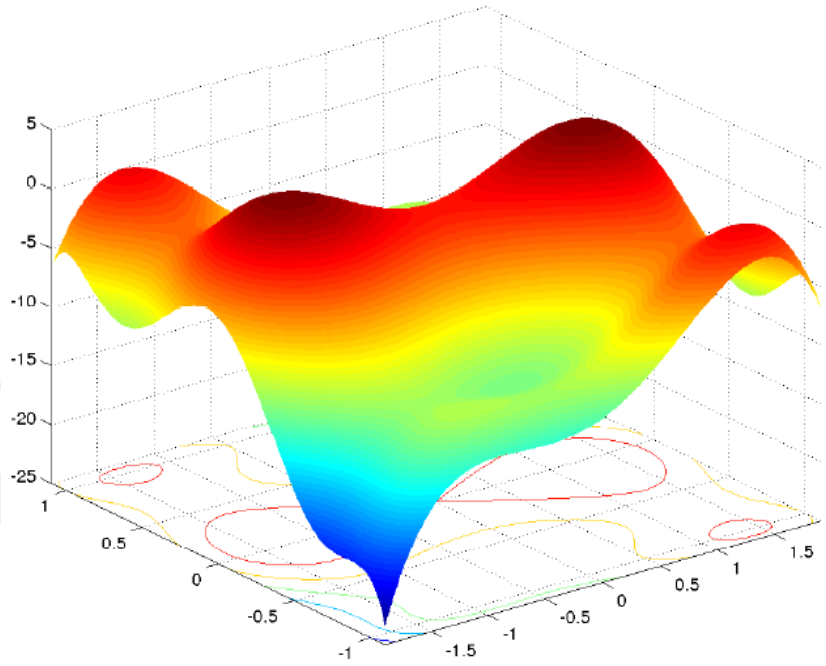
Problemin fonksiyonu

$$F(x) = -4((4 - 2.1x^2 + \frac{x^4}{3})x^2 + xy + (4y^2 - 4)y^2) \quad (5.5)$$

Problemin özelliği

- X değeri-1.9 ile 1,9 arasında olmalıdır. Y değeri-1.1 ile 1,1 arasında olmalıdır. X ∈ (-6,6); y ∈ (-1.1,1.1);
- Global optima sayısı:2
- Local optima sayısı :2

Problemin grafiği



Şekil 5.5. Six-hump camel back grafiği (Li vd., 2013).

5.1.6. F6: Shubert

Probleme ait fonksiyon, özellik ve grafik bilgisi aşağıdaki gibidir.

Problemin fonksiyonu

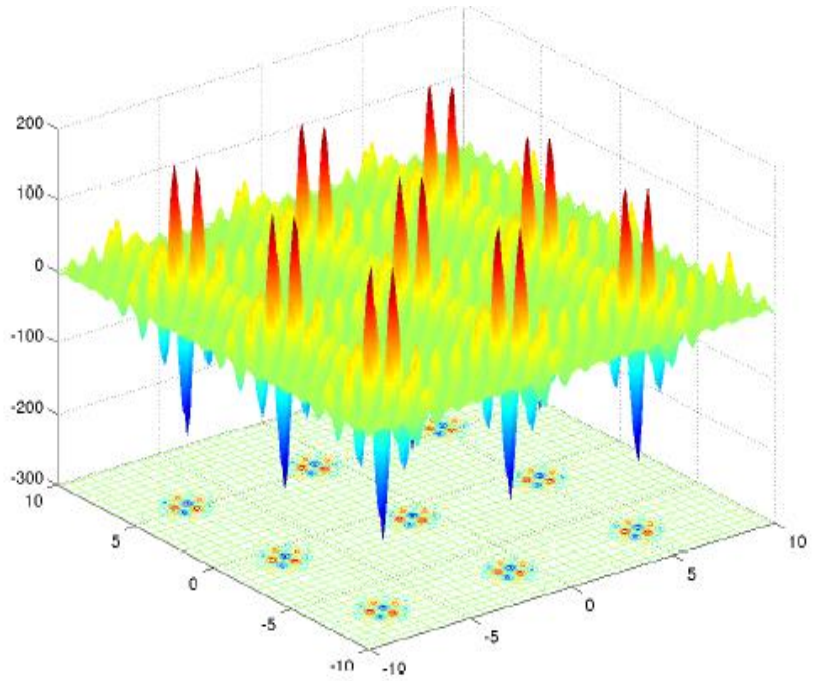
$$F(x) = -\prod_{i=1}^{D} \sum_{j=1}^{5} jcos((j+1)x_i + j) \tag{5.6}$$

Problemin özelliği

- $x_i \epsilon (-10,10)^D, i = 1,2 \dots D$
- Global optima sayısı: $D.3^D$
- Local optima sayısı :çok

Fonksiyon Shubert fonksiyonunun ters çevrilmiş halidir. Eşit olmayacak şekilde dağılmış $D.3^D$ global optima vardır. 3^Dglobal optima D gruplarına ayrılır her gruptaki çözümler birbirine yakındır. Aşağıdaki şekilde her biri birbirine çok yakın 9 çiftle 18 global optimum noktanın olduğu Shubert 2D fonksiyonu çizilmiştir. Toplam 760 tane global ve local optima vardır.

Problemin grafiği



Şekil 5.6. Shubert grafiği (Li vd., 2013).

5.1.7. F7: Vincent

Probleme ait fonksiyon, özellik ve grafik bilgisi aşağıdaki gibidir.

Problemin fonksiyonu

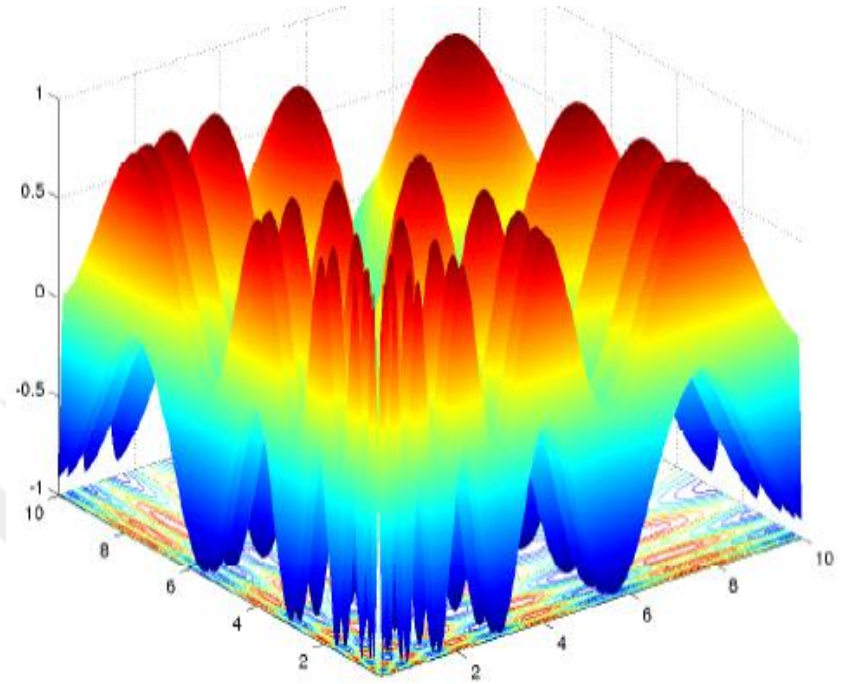
$$F(x) = \frac{1}{D}\sum_{i=1}^{D} \sin\left(10\log\left(x_i\right)\right) \tag{5.7}$$

Problemin özelliği

- $x_i \epsilon (0.25,10)^D, i = 1,2 \ldots D$
- Global optima sayısı : 6^D
- Local optima sayısı :0

Fonksiyon Vincent fonksiyonunun ters çevrilmiş halidir. Vincent fonksiyonunda global optimalar arasında çok aralık vardır. Ayrıca local fonksiyonu yoktur.

Problemin grafiği



Şekil 5.7. Vincent grafiği (Li vd. 2013).

5.1.8. F8: Modified rastrigin - all global optima

Probleme ait fonksiyon, özellik ve grafik bilgisi aşağıdaki gibidir.

Problemin fonksiyonu

$$F(x) = -\sum_{i=1}^{D} (10 + 9\cos(2\pi k_i x_i)) \tag{5.8}$$

Problemin özelliği

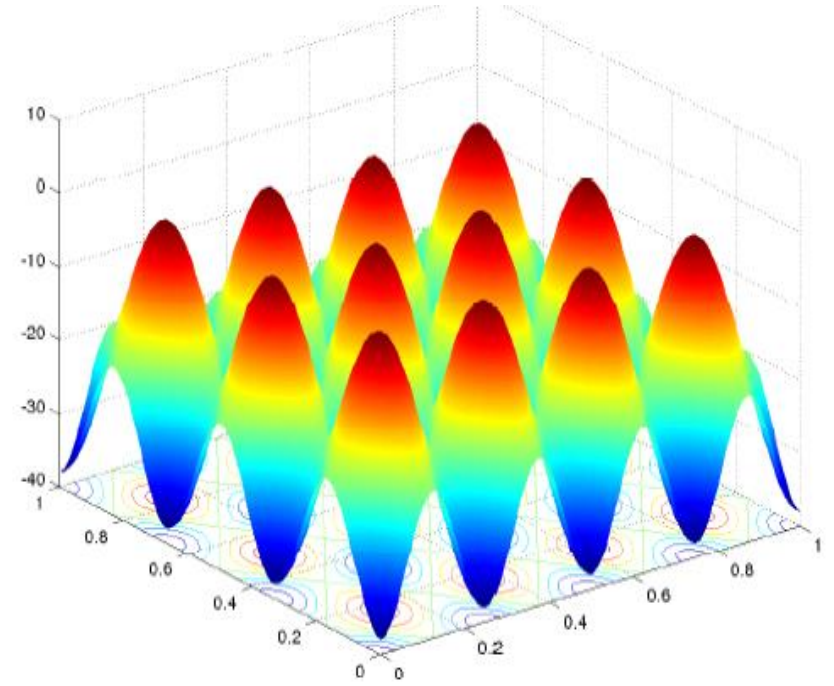
- $x_i \epsilon (0,1)^D, i = 1,2 \ldots D$
- Global optima sayısı: $\prod_{i=1}^{D} k_i$
- Local optima sayısı :0

Değiştirilmiş Rastrigin işlemidir. Problem örneği aşağıdaki ayarlarla kullanılmıştır.

$k_i = 1$ for i=1-3, 5-7, 9-11, 13-15, and $k_4 = 2, k_8 = 2, k_{12} = 3, k_{16} = 4$. Bu durumda f=16 da özdeş değerlere sahip 48 eşit dağıtılmış global optima vardır.

Problemin grafiği

Fonksiyonun grafiği çizdirilirken D=2; $k_1 = 3; k_2 = 4$ değerleri kullanılmıştır.



Şekil 5.8. Modified rastrigin- all global optima grafiği (Liv d., 2013).

5.1.9. Kompozisyon fonksiyonları

D boyutlu kompozisyon fonksiyonu genellikle $CF_J: A_D \subset R^D \rightarrow R$ şeklindeyken, n temel fonksiyonu $F_i: A_D \subset R^D \rightarrow R$ şeklindedir. Her bir temel işlev A_D optimizasyon alanı içinde yeni bir konuma kaydırılır. Kaydırıldıktan sonra doğrusal dönüşüm matrisine döndürülebilir veya olduğu gibi kullanılabilir. Kompozisyon fonksiyonu CF_J aşağıdaki formüle göre hesaplanır.

$$CF_J(x) = \sum_{i=1}^{n} wi(f_i((x - o_i)/ \lambda_i . M_i) + bias_i) + f_{bias}^{j} \qquad (5.9)$$

n işlev sayısıdır. f_i i. temel fonksiyonun normalleşmesini gösterir. wi ağırlığa karşılık gelir. o_i her f_i den en uygun olanı değiştirir. M_i her f_i nin liner dönüşüm matrisidir ve λ_i her bir f_i fonksiyonunu germek veya sıkıştırmak için kullanılan bir parametredir. Kompozisyon işlevi $bias_i$ ve f_{bias}^{j} olmak üzere iki adet sapma parametresi içerir. İlk parametre her bir temel işlev için işlev sapmasını ve optimum noktalar arasından hangisinin global olduğunu tanımlarken ikinci parametre oluşturulan birleştirme işlevi için işlev değeri sapmasını tanımlar.

5.2. Algoritmanın Parametre Ayarları

Optimizasyon problemlerini çözmek için kullanılan algoritmaların en iyi performansına ulaşması için ayarlanması gereken birçok parametre ve tasarım seçenekleri vardır. Problemin çözümünde kullanılan birçok algoritma türü için bu geçerlidir. Algoritmanın performansını en iyi değere çıkarmak için algoritmanın onlarca parametresine ince ayar yapılması gerekebilir. Algoritmanın parametreleri probleme özgü değişiklik gösterebilir. Bu yüzden problem değiştiğinde algoritma parametrelerinde tekrar ayarlanması gerekebilir. Örneğin hat dengeleme için tasarlanmış algoritmanın parametre ayarlarıyla çizelgeleme için tasarlanmış algoritmanın parametreleri aynı olmayabilir.

Uzun yıllar algoritmaların parametreleri, algoritmayı oluşturan kişiler tarafından deney ortamında farklı parametre ve metotlar denenerek en iyi parametreler bulunmaya çalışılmıştır. Manuel parametre ayarlama, hiç parametre ayarı yapılmamasından daha iyi olsa ve daha yüksek performans sağlasa da bazı dezavantajlara sahiptir. Bu dezavantajlar;

- Manuel parametre ayarlamasında insan çabası yüksektir.
- Kişisel tecrübeler ve sezgiler kullanarak parametre ayarlanmaya çalışıldığı için bazı ön yargılardan dolayı iyi sonuç vermez.
- Sınırlı sayıda örnek tarafından incelenir.
- Tasarım ve parametre ayarlamasında kullanılan parametreler son algoritmada da kullanılmasında dolayı taraflı bir algoritma değerlendirilmesi yapılmış olur.

Algoritmalın parametreleri bu sebeplerden dolayı otomatik yapılarla belirlenmeye çalışılmıştır. Kullanılan yöntemler; deneysel tasarım teknikleri (Adenso-Díaz ve Laguna 2006; Coy vd., 2001), yarış yaklaşımları (Birattari vd., 2002) ve sezgisel arama teknikleridir (Ansótegui vd., 2009; Balaprakash vd., 2007; Hutter vd., 2009; Nannen ve Eiben 2006; Riff ve Montero, 2013).

Otomatik algoritma konfigürasyonu, makine öğrenme bakış açısından, bir takım eğitim problemi örnekleri üzerinde öğrenerek bilinmeyen problem örneklerini çözmek için iyi parametreler bulma problemidir (Birattari ve Kacprzyk, 2009). Otomatik algoritma konfigürasyonundaki amaç algoritmada kullanılacak parametreleri daha az maliyet harcayarak eğitim aşamasında bulunmasıdır. Otomatik algoritma konfigürasyonu, parametre ayarlama ve üretim aşamaları arasındaki fark nedeniyle, bir örneği çözerken parametre ayarlarını uyarlayan veya kontrol eden çevrimiçi yaklaşımlardan farklı olarak çevrimdışı ayarlama olarak da adlandırılır (Battiti vd., 2008).

5.2.1. Otomatik algoritma parametre yapılandırılması

Konfigürasyon algoritmaları

Optimizasyon problemlerinin sonucunu etkileyen, ayarlanabilir birden fazla parametrelerden oluşabilir. Örnek olarak genelleştirilmiş yapılandırılabilir ABC algoritmasında (Aydın vd., 2017) birden fazla sonucu etkileyen parametrelerden oluşur. Başka bir örnek verecek olursak valf noktası etkisiyle dış bükey olmayan ekonomik sevkiyat probleminde kullanılan ABC algoritmasında (Aydin vd., 2017) ABC metodunun parametreleri ve problemin kendi parametrelerinin ayarlanması algoritmanın daha iyi sonuçlar oluşturmasını sağlamıştır. Parametrelerin yapılandırılabilir olmasının sebebi algoritmaların her problem için tek bir optimal parametre ayarlarının bulunmamasına bağlıdır (Birattari, 2009; Adenso-Díaz ve Laguna, 2006; Hutter vd., 2010). Algoritma parametreleri üç ayrı sınıfta incelenebilir. Bunlar;

Kategorik parametreler: Kesin bir sıra veya hassas değerleri olmayan ayrık değerleri temsil eden parametrelerdir.

Sıralı parametreler: Sıralı değişkenlere karşılık gelen parametrelerdir

Sayısal parametreler: Gerçek bir sayıya karşılık gelen parametrelerdir.

Parametre değerlerinden biri değiştiğinde diğer parametrelerin değeri değişimden etkilenebilir.

Algoritma yapılandırma sorunu

Bir problem için birden fazla parametre ayarı olabilir. Algoritmanın tüm parametreleri içeren bir dizi kullanılarak parametre dizisi oluşturulur. Parametre dizisinden i. parametre probleme uygulandığında C (θ, i) maliyet hesabı oluşmuş olur. Maliyet değeri sorunumuza göre değişiklik gösterir. Örneğin minimum değerlerin bulunduğu bir problemde maliyet değeri problem sonucunda çıkan değerler olup en iyi maliyet en düşük değere karşılık gelmektedir. Otomatik parametre ayarında amaç en uygun parametre değerlerini bularak algoritmanın en düşük maliyetini bulmaktır.

Problem için bir algoritma yapılandırırken, optimize etmek istediğimiz ölçüt, F (θ) fonksiyonudur. F (θ), i. parametrenin beklenen maliyetidir. F (θ), konfigürasyonların örnekler üzerinde sıralanmasını belirler. F (θ) fonksiyonunun kesin değeri bilinemez sadece örnekler üzerinden bir değer tahmin edilir. Pratik olarak hesaplanmasının zor olduğu problemlerde her bir

deney belirli bir algoritmanın belirli deney ortamında gerçekleştirildiği bir çalışmadır (Bartz-Beielstein , 2007).

Otomatik algoritma konfigürasyon metodu

Algoritma konfigürasyon problemi önemli bir sorun olmasına rağmen uzun yıllar manuel olarak yapılmıştır. Birkaç makalede daha sistematik tekniklerden yararlanarak az sayıda parametre içeren algoritmaların parametreleri ayarlanmıştır. Kullanılan yöntemler arasında local arama yöntemleri, Taguchi kısmi tasarımları, tepki yüzey metodolojisine dayanan yöntemler (Coy vd., 2001) ve ANOVA tekniklerinin daha sistematik olarak kullanılmasıyla (Ridge ve Kudenko, 2007; Ruiz ve Maroto, 2005) oluşan yöntemler yer almaktadır.

Parametreleri sayısal veri olan algoritmalın, parametreleri ayarlanırken CMA-ES (Hansen ve Ostermeier 2001), BOBYQA (Powell, 2009) veya MADS (Audet ve Orban, 2006) kullanılabilir. Bu yöntemlerde karar değişkenleri yuvarlanarak tam sayı parametreleri optimize edilebilir.

Algoritma parametre denemeleri yapıldıktan sonra en önemli işlem algoritma parametrelerinin puanlanmasıdır. Algoritma parametrelerinin maliyet değeri hesaplanırken kullanılan yöntemlerden biri de vekil modeldir. Tahminlere dayanarak seçilen parametreler fiili olarak kullanılarak tahmin modeli güncellenir.

Algoritmanın parametreleri ayarlanırken, sıralı istatiksel test kullanarak birkaç aday arasından seçim yapmak için yarış (Maron ve W. Moore, 1997) metodu kullanılabilir. Yarış için seçilen ilk aday parametre çözümü teknikleriyle rastgele veya probleme özgü bilgiye dayanarak seçilebilir. Yenilenen yarışlarda bir sonraki test edilecek parametre değeri geçmişteki parametre değeri sonuçlarına göre belirlenir.

Yenilemeli Yarış: Problem seti için oluşturulan algoritmanın parametrelerini ayarlamak otomatik konfigirasyon paketi irace kullanılmıştır. İrace algoritmanın parametrelerini ayarlamak için F yarışından yararlanmaktadır. F yarışı Friedman'ın parametrik olmayan iki yönlü varyant analizini kullanan yenilemeli yarış uygulamasıdır (Balaprakash vd., 2007; Birattari, vd., 2010).

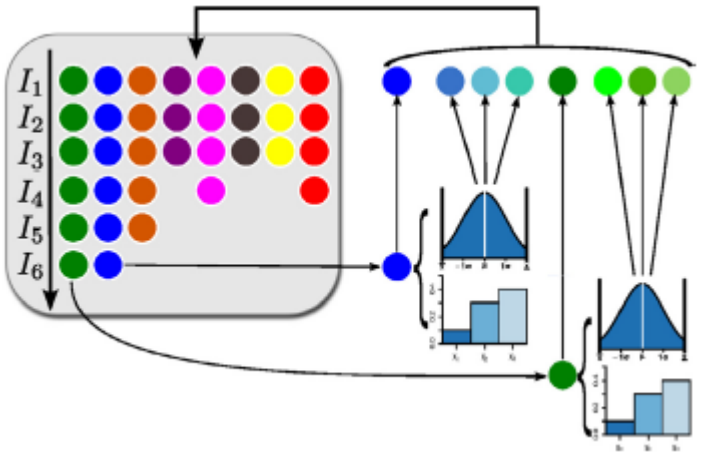
Yenilemeli yarış 3 adımdan oluşan otomatik yapılandırma metodudur. Bunlar;

- Belirli bir örnekleme modelinden aday konfigürasyonlar oluşturur.
- Yeni örneklenmiş konfigürasyonlardan en iyi konfigürasyonu bulmak için yarışa tabi tutulur.
- Örnekleme modeli seçkin bir aday konfigürasyon seti ile güncellenir.

Yukarıdaki üç adım sonlandırma kriterleri sağlanıncaya kadar devam eder ve sonuca ulaşmaya çalışır.

Yenilenen yarışta her parametre parametreler arasındaki kısıtlamalar ve koşullar dışında diğer parametrelerinin örnekleme dağılımından bağımsız bir örnekleme dağılımıyla ilişkilendirilmiştir. Örnekleme dağılımı sayısal parametreler için kesik normal bir dağılımken, kategorik parametreler için ayrık bir dağılımdır. Dağılımın güncellenmesi normal dağılım durumunda ortalama veya standart sapmayla olurken ayrık dağılımda ayrık olasılık değerlerinin değişmesiyle olur. Gelecek yenileme değerine karar verirken şimdiye kadar bulunan en iyi parametre değerini kullanır.

İlk kez makine öğreniminin model seçimini yapmak için geliştirilen yarış daha sonra optimizasyon algoritmalarının en iyi konfigürasyonunu bulmak için uyarlanmıştır. Yarış sınırlı sayıda aday çözümle başlar. Aday çözümler değerlendirilerek istatiksel olarak en kötü olan konfigürasyon atılır. Test başlangıcındaki örnekler sonucu bulmada çok etkili olduğu için başlangıçta çok fazla aday çözüm değerlendirilir. Değerlendirme işlemi devam ettikçe aday çözüm azalır bu işlem en iyi maliyetli aday çözüm bulununcaya kadar veya bitirme koşulu oluşuncaya kadar devam eder. Algoritmanın adımlarını Şekil 5.9'dadır.



Şekil 5.9. Yenilemeli yarış adımları (López-Ibáñez ve Dubois-Lacoste, 2016).

Yenilemeli yarış gerçekleştireceği yenileme sayısını tahmin ederek başlar. Yenileme sayısı $2 + log_2 N^{param}$ formülüyle bulunur. N^{param} Parametre sayısını ifade eder. Her yenilemede sınırlı bir hesaplama bütçesi olan yarış gerçekleştirilir.

İlk yenilemede ilk aday konfigürasyon kümesi (X parametre uzayı) eşit şekilde örneklenerek üretilir (adım 1) ve en iyi konfigürasyonlar yarışla (adım 2) belirlenir. Parametre alanı örneklendirilirken parametreler koşulların bağımlılık grafiği tarafından belirlenen sıraya göre değerlendirilir. Koşulu olmayan parametre önce değerlendirilirken koşulu olan parametre daha sonra değerlendirilir. Yarış başladığında konfigürasyonlardan her birinin maliyeti ilk kez C (adım 3) arayıcılığıyla hesaplanır. Yapılandırmamalar belirli bir örnek sayısına ulaşıncaya kadar tekrarlanır ve sonuçlar üzerinden istatiksel testler yapılır. Testler sonucunda durumu kötü olan parametreler listeden kaldırılır.

Listeden kaldırılacak parametre konfigürasyonu seçmek için çeşitli alternatifler vardır. F-Race algoritması (Birattari ve Kacprzyk, 2009; Birattari, vd., 2002) parametrik olmayan Friedman'ın iki yönlü varyans analizine (Friedman testi) ve Conover tarafından açıklanan geçici görev ilişki testine dayanmaktadır. Belirli bir konfigürasyon senaryosu için en uygun test, ayar hedefine ve maliyet fonksiyonunun özelliklerine bağlıdır. Her yarıştan sonra ayakta kalan konfigürasyonlara yarış sırasında hangi istatiksel testin kullanıldığına bağlı olarak toplamların toplamına veya ortalama maliyetine göre bir dereceye atılır.

İrace'de uygulanan yinelenen yarış algoritması, erken birleşmeyi önlemek için "yumuşak yeniden başlatma" mekanizması içeriyor. Klasik F-yarışında çeşitlilik örneklenmiş konfigürasyonların değişkenliğiyle ortaya çıkar. Örnekleme dağılımı birbirine benzer birkaç konfigürasyondan oluşursa çeşitlilik kaybolur ve yeni oluşturulan aday konfigürasyonlar test edilen konfigürasyonlara çok benzer olur. Bu benzerlik yüzünden algoritma, parametre varyasyonlarını tekrar tekrar değerlendirerek kalan bütçeyi boşa harcar.

Yumuşak yeniden başlatma mekanizması, her yeni aday kümesini oluşturduktan sonra erken yakınlaşmayı kontrol eder. İki aday yapılandırma arasında mesafe sıfır olduğunda erken yaklaşma olduğu düşünülür. İki yapılandırma arasındaki mesafe parametre ayarları arasındaki maksimum mesafedir. Erken yakınsama tespit edildiğinde, örnekleme dağılımını kısmen yeniden başlatarak yumuşak yeniden başlatma uygulanır. Yeniden başlatmada, elit yapılandırma kullanılarak sıfır mesafeli adaylar yapılandırılır. Diğer mesafeli aday yapılandırmalar yeni konfigürasyonlara yol açabileceğinden değiştirilmez.

İrace de ki parametrelerin amacı İrace'yi algoritmalar karşısında daha esnek bir hale getirmek olsa da İrace algoritmasında kullanılan F yarış prosedürü arama davranışını doğrudan etkileyecek birkaç parametreye sahiptir. İrace parametreleri belirlenirken algoritmanın birçok senaryosunu incelenmeli ve ondan sonra parametreler belirlenmelidir. İracenin çalışma prensibi Şekil 5.10 da verilmiştir.

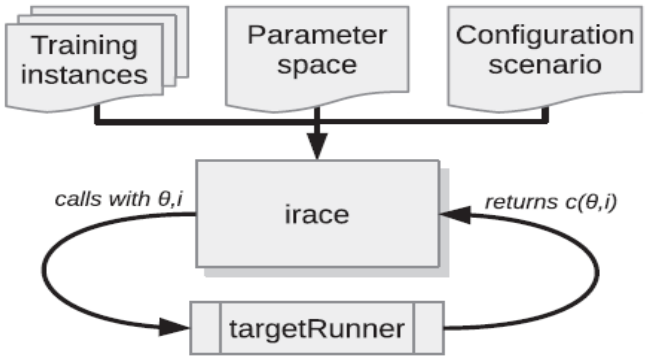


Fig. A.11. Scheme of the user-provided components required by irace.

Şekil 5.10. İrace çalışma prensibi (Birattari ve Kacprzyk, 2009).

5.3. İracenin Kullanımı

Problem setimizin algoritması oluşturulduktan sonra algoritma parametrelerinin ayarlanması gerekir. Parametre ayarlarını, irace programında yapabilmek için parametrelerin özellikleriyle belirtilmesi gerekir. İndirilen irace programındaki parametre.txt dosyasına parametreler tanımlanırken aşağıdaki tanım metotları kullanılır.

<name>: Parametrenin adının tanımlandığı kısımdır. Parametre tanımlanırken parametreye karşılık gelecek en kısa biçimde parametre tanımın yapılması çoklu parametre ayarlarında parametre karışıklığının önüne geçer.

<label>: Parametrenin algoritmadaki etiketini ifade eder. Problem seti için yazılmış olan algoritma çalıştırılırken tanımlanmış parametrelerin dışarıdan girilmesi gerekir. Dışarıdan girilen parametreler label vasıtasıyla algoritma tarafından tanınır.

<type>: Parametrenin tipini karşılık gelir. Algoritmaya uygun parametreler irace tarafından bulunurken irace algoritmanın tipine göre parametre belirlemeye çalışır. Tek bir harf ile tanımlanır. Parametre tipleri; tam sayı(i), gerçek sayı(r), sıralı(o) ve kategorik(c) dir. Parametre için en uygun parametre tipinin girilmesi İrace'nin daha iyi bir sonuç vermesi için önemlidir. Gerçek parametreler belirli bir aralıkta herhangi bir gerçek değeri alabilen parametrelerdir. Gerçek değerin hane sayısı kaç belirtildiyse sonuç o haneye yuvarlanır. Örnek olarak hane sayısı 3 olan bir parametre sonuç 0.1114 veya 0.1116 ise gerçek değer 0.111 olur. Tamsayı parametreleri belirtilen aralık içindeki sadece tam sayı değerlerini alabilir. Kategorik parametreler bir dizi şeklinde tanımlanır. İrace sonucu bulurken bu dizi içerisinden bir değeri sonuç olarak döndürür. Sıralı parametreler kategori parametreler gibidir farkı sıralı bir listeye sahiptir.

<domain>: Tam ve gerçek sayılardan oluşan parametrenin aralığını veya kategorik ve sıralı değerlerden oluşan parametrenin değer kümesinin belirtildiği kısımdır. Aralık alt değer ve üst değer şeklinde tanımlanır. Değer kümesinde küme (değer1, değer2 ,… değern) şeklinde tanımlanır. Parametre değer aralığının doğru belirlenmesi iracenin daha hızı ve iyi sonuçlar vermesini sağlar.

<condition>: Parametrenin koşullu olarak etkin veya devre dışı olup olmadığının belirtildiği kısımdır. Parametre tanımlanırken false değeri atanırsa uygun parametre aralığı bulurken parametreye herhangi bir değer atanmaz. Parametre koşulu geçerli bir R mantıksal ifade olmalıdır. Koşul false durumundayken diğer parametre adlarından birini içerebilirken false durumunda değilse irace hata verir.

Hook-run dosyasında parametre ayarı yapılacak algoritmanın uzantısı ve irace programının nasıl çalışacağı belirtilir. İrace programı çakıştırıldıktan sonra parametre sonuçlarının bulunduğu stdout ve algoritma çalışma hatalarının bulunduğu stderr dosyasını üretir. Algoritma çalıştırılmadan önce sabit parametreler ve ayarlanmak istenen parametreler belirtilir. Ayarlanmak istenen parametreler parametre.txt dosyasından alınacaktır.

Tun-conf dosyasında algoritma çalışma sayısını belirtmek için maxExperiments parametresinin ayarlanması gerekir. Parametrenin çok yüksek ayarlanması algoritmanın uzun süre boyunca çalışmasına sebep olur, çok düşük ayarlanmasında istenilen parametre ayarının yapılamamasına sebep olur. Algoritmamızın parametrelerini bulurken kullandığımızı maksimum çalışma sayısı 10000 olarak belirtilmiştir.

Algoritmamız birden fazla soruna tek bir optimum parametre üretmesini istediğimiz için parametreleri ayarlayacağımız problem listenin iyi ayarlanması gerekmektedir. İnstance file dosyası algoritmanın hangi problemleri dikkate alarak parametreleri ayarlayacağının belirtildiği kısımdır. Basit problemlerden oluşan bir liste daha zor problemlerde kötü sonuç verirken zor problemlerden oluşan bir liste basit problemlerde kötü sonuç verebilir bu yüzden problem setini doğru seçmeliyiz.

Çalıştır.sh dosyasında paralel olarak çalıştırılacak süreç sayısı belirtilir. Süreç sayısı belirtilirken bilgisayarın özellikleri dikkate alınmalıdır aksi takdirde bilgisayarda aşırı ısınma kapanma gibi sorunlar ortaya çıkabilir. Paralel çalışma sayısı belirtildikten sonra sonuçların yazıldığı dosya uzantısı ve dosya ismi belirtilir. Aşağıdaki örnekte 6 paralel süreç kullanan ve istenilen dosyaya sonuçları yazan çalıştır.sh dosya içeriği verilmiştir.

```
echo "... new_se_abc_ls_cec2017 sonuçları hesaplanıyor ... "
```

irace --parallel 6 >> ../niching_se_abc_ls_cec2015_topoly_03022019_2;

İrace konfigürasyon işlemini yaparken her konfigürasyondan sonra ayrıntılı bir rapor oluşturur. Rapor içerisinde elit yapılandırmalar bulunan en iyi yapılandırma tablosu ve komut satırı vardır.

İrace parametre ayarı yaparken R veri küme dosyası olan irace.Rdata dosyasını oluşturur ve her konfigürasyon işleminden sonra bu dosyayı günceller. Dosya aşağıda belirtilen parametreleri içerir.

Scenario: İrace konfigürasyon senaryosu

Parametreler: Algoritmanın o ana kadar bulduğu parametreleri içerir.

Seeds: Örnek ve tohumdan oluşan iki sütunlu matrisi içerir.

allConfigurations: İrace işlemi sırasında oluşturulan tüm konfigürasyonları içerir.

Experiments: Yenileme sırasında deneyimleri içerir.

5.4. Algoritmanın Parametre Sonuçları

Multi model problemlerin çözümünde kullanılan algoritmaların iyi bir sonuç verebilmesi için algoritma parametrelerinin doğru seçilmesi gerekmektedir. Parametreler belirlenirken doğru aralık ve tip belirlenmelidir. Aralığı büyük tutmak İrace programının gereksiz aralıklarda arama yapmasına ve işlemin uzun sürmesine sebep olmaktadır. Doğru tip belirlenmediğinde algoritma için gerekli parametre tam olarak ayarlanamadığından tam olarak istenilen sonucu veremeyecektir. Algoritmanın en iyi sonucu, a parametresinin 1,1 değerinde verdiği bir durumda a parametresinin tipi sayısal ayarlanırsa parametre değeri 1,0 olacağından en iyi sonuca ulaşamamış oluruz. Algoritmamızın parametre aralığı ve tipi belirlenirken yukarıdaki unsurlar dikkate alınmıştır.

Parametre havuzu irace de çalıştırıldıktan sonra değerler eldi edilmiştir. Algoritmamızda kullanılan parametre, parametre tipi, aralığı ve sonucu aşağıdaki tabloda verilmiştir.

Çizelge 5.1. Algoritma parametre değerleri.

PARAMETRE ADI	**TİPİ**	**ARALIĞI**	**DEĞER**
limitF	float	1-5	2,1685
ps(pool size)	int	50;100;150;200	50
Incamount	int	20;30;40;50;60;70;80;90;100;120	100

Çizelge 5.1. (Devam) Algoritma parametre değerleri.

PARAMETRE ADI	TİPİ	ARALIĞI	DEĞER
SN_{init}	int	50;60;70;80;90;100;120;140;150;160;180;200	50
Incamount	int	20;30;40;50;60;70;80;90;100;120	100
SN_{max}	int	300;350;400;500;600;700;750;800;900;1000	600
StagnationLimit	int	50;100;150;200	200
maxfailures	int	25;50;100;150;200	200
topology	int	1;2;5	2
r	float	0,01	0,01
seed	float	random	random
ε	float	0,1;0,01;0,001;0,0001;0,00001	0,1
problemID	İnt	1-20	1

5.5. Donanım ve Yazılım

SABCElit algoritmasının programlanmasında C++ programlama dili kullanılmıştır. Algoritma Linux tabanlı Ubuntu işletim sisteminde yazılmıştır. SABCElit algoritmasının kodlanması ve test fonksiyonlarının sonuçları; İntel i7 7700HQ işlemci,16 GB RAM özelliklerine sahip bilgisayarda gerçekleştirilmiştir.

6. DENEY SONUÇLARI

6.1. Sonuçların Hesaplanması

Multimodal problemler birden fazla optimum noktaya sahip olabilirler. Problem için algoritma çalıştırıldığında, problem için belirlenen bütün optimum sonuçları vermesi beklenir. Çözüm kümesi belirlenirken çözümlerin birbirinden belirli bir uzaklıkta olması gerekir. Problem setinde iki çözüm arasındaki belirlenen uzaklık değerine göre sonuçların bulunması istenmiştir. Belirlenen optimalar arası uzaklık değerleri; 0.1, 0.01, 0.001, 0.0001, 0.00001 dir.

Algoritmanın başarısı ölçülürken problem setinde belirlenen optimalar arası uzaklık ve niş yarıçapı değerleri dikkate alınarak istenilen çalıştırılma sayısı kadar çalıştırılır. Problem setimizde algoritmanın çalıştırılma sayısı 51 olmalıdır. Algoritmanın başlangıçta arama uzayı rastgele sayılardan oluşmalıdır. Algoritma döngü sayısı maksimum döngü sayısından fazla olamaz. Maksimum çalıştırma sayısı Çizelge 6.1'de, niş yarıçap ve çalıştırma sayısı tablosu Çizelge 6.2 de verilmiştir.

Çizelge 6.1. Maksimum çalıştırma sayısı ve niş yarıçapı.

Fonksiyon Numarası (FN)	Maximum Çalıştırma Sayısı (Maxfes)
F1 den F5 e kadar (1D veya 2D)	5,0E+04
F6 dan F11(2D)	2,0E+05
F6dan F12(3D ve üzeri)	4,0E+05

Çizelge 6.2. Niş yarıçapı ve optima sayısı.

FN	r	Peak height	Global optima
F1 (1D)	0,01	200	2
F2 (1D)	0,01	1.0	5
F3 (1D)	0,01	1,0	1
F4 (2D)	0,01	200	4
F5 (2D)	0,5	1,031628453	2
F6 (2D)	0,5	186,7309088	18
F7 (2D)	0,2	1,0	36
F6 (3D)	0,5	2709,093505	81
F7 (3D)	0,2	1,0	216
F8 (2D)	0,01	-2,0	12
F9 (2D)	0,01	0	6
F10 (2D)	0,01	0	8
F11 (2D)	0,01	0	6
F11 (3D)	0,01	0	6
F12 (3D)	0,01	0	8
F11 (5D)	0,01	0	6
F12 (5D)	0,01	0	8
F11 (10D)	0,01	0	6
F12 (10D)	0,01	0	8
F12 (20D)	0,01	0	8

Problem setinde local ve global optima noktaları mevcuttur. Algoritmanın başarısı ölçülürken, algoritma optimalar arası uzaklık mesafesi parametresi dikkate alınarak çalışması sonucunda bulduğu local ve global optima sayısı toplanarak toplam optima sayısı bulunur. Algoritmanın performansını ölçmek için Success Rate (SR), Peak Ratio(PR) ve Average Number Of Function Evaluations (AR) değerlerinin hesaplanması gerekmektedir.

$$PR = \frac{\sum_{run=1}^{NR} NPF_i}{NKP * NR} \tag{6.1}$$

NR toplam çalıştırma sayısını ifade eder. NPF_i i. çalışmadaki bulunan optima sayısını ifade eder. NKP toplam optima sayısını ifade eder.

$$SR = \frac{NSR}{NR} \tag{6.2}$$

NSR başarılı çalıştırma sayısıdır. Başarılı çalıştırmada fonksiyonun tüm optima değerleri bulunur.

$$AR = \frac{\sum_{run=1}^{NR} FE_i}{NR} \tag{6.3}$$

FE algoritmanın döngü sayısıdır. Algoritmanın sonucu bulamadığındaki son FE değeri maksimum çalıştırma sayısına eşittir.

6.2. Algoritma Çalışma Sonuç Tablosu

Algoritma test fonksiyonlarıyla çalıştırıldıktan sonra çıkan değerler aşağıdaki gibidir.

Çizelge 6.3. 0.1 optima arası uzaklıkta PR, SR ve AR sonuçları.

FN	PR	SR	AR
F1 (1D)	1,000	1,000	0,003
F2 (1D)	1,000	1,000	0,003
F3 (1D)	1,000	1,000	0,003
F4 (2D)	1,000	1,000	0,021
F5 (2D)	1,000	1,000	0,005
F6 (2D)	1,000	1,000	0,235
F7 (2D)	1,000	1,000	0,029
F6 (3D)	0,879	0,039	1,000
F7 (3D)	1,000	1,000	0,100
F8 (2D)	1,000	1,000	0,059
F9 (2D)	1,000	1,000	0,017

Çizelge 6.3. (Devam) 0.1 optima arası uzaklıkta PR, SR ve AR sonuçları.

FN	PR	SR	AR
F10 (2D)	0,983	0,863	0,464
F11 (2D)	0,951	0,784	0,362
F11 (3D)	1,000	1,000	0,017
F12 (3D)	1,000	1,000	0,035
F11 (5D)	1,000	1,000	0,027
F12 (5D)	1,000	1,000	0,042
F11 (10D)	1,000	1,000	0,078
F12 (10D)	0,350	0,157	0,923
F12 (20D)	0,703	0,294	0,842

Optimalar arası uzaklık 0.1 ayarlandığında 1 boyutlu problemlerin hepsinde 2 boyutlu problemlerin büyük bir kısmında PR ve SR sonuçlarında yüzde yüz başarı sağlanmıştır.

Çizelge 6.4. 0.01 optima arası uzaklıkta PR, SR ve AR sonuçları.

FN	PR	SR	AR
F1 (1D)	1,000	1,000	0,003
F2 (1D)	1,000	1,000	0,013
F3 (1D)	1,000	1,000	0,006
F4 (2D)	1,000	1,000	0,037
F5 (2D)	1,000	1,000	0,012
F6 (2D)	1,000	1,000	0,349
F7 (2D)	1,000	1,000	0,345
F6 (3D)	0,830	0,039	1,000
F7 (3D)	0,988	0,373	1,000
F8 (2D)	1,000	1,000	0,095
F9 (2D)	0,912	0,569	0,903
F10 (2D)	0,968	0,765	0,604
F11 (2D)	0,712	0,137	1,000
F11 (3D)	0,680	0,039	1,000
F12 (3D)	0,419	0,000	1,000
F11 (5D)	0,657	0,000	1,000
F12 (5D)	0,262	0,000	1,000
F11 (10D)	0,386	0,098	1,000
F12 (10D)	0,157	0,000	1,000
F12 (20D)	0,110	0,000	1,000

Optimalar arası uzaklık 0.01 ayarlandığında 1 boyutlu problemlerin hepsinde 2 boyutlu problemlerin büyük bir kısmında PR ve SR sonuçlarında yüzde yüz başarı sağlanmıştır.

Çizelge 6.5. 0,001 optima arası uzaklıkta PR, SR ve AR sonuçları.

FN	PR	SR	AR
F1 (1D)	1,000	1,000	0,003
F2 (1D)	1,000	1,000	0,044
F3 (1D)	1,000	1,000	0,009
F4 (2D)	1,000	1,000	0,084
F5 (2D)	1,000	1,000	0,023
F6 (2D)	0,999	0,980	0,462
F7 (2D)	1,000	1,000	0,467
F6 (3D)	0,741	0,098	1,000
F7 (3D)	0,928	0,275	1,000
F8 (2D)	1,000	1,000	0,126
F9 (2D)	0,830	0,353	0,926
F10 (2D)	0,926	0,510	0,796
F11 (2D)	0,712	0,137	1,000
F11 (3D)	0,673	0,020	1,000
F12 (3D)	0,336	0,000	1,000
F11 (5D)	0,657	0,000	1,000
F12 (5D)	0,255	0,000	1,000
F11 (10D)	0,324	0,059	1,000
F12 (10D)	0,159	0,000	1,000
F12 (20D)	0,096	0,000	1,000

Optimalar arası uzaklık 0.001 ayarlandığında 1 boyutlu problemlerin hepsinde 2 boyutlu problemlerin büyük bir kısmında PR ve SR sonuçlarında yüzde yüz başarı sağlanmıştır.

Çizelge 6.6. 0,0001 optima arası uzaklıkta PR, SR ve AR sonuçları.

FN	PR	SR	AR
F1 (1D)	1,000	1,000	0,003
F2 (1D)	1,000	1,000	0,033
F3 (1D)	1,000	1,000	0,013
F4 (2D)	1,000	1,000	0,107
F5 (2D)	1,000	1,000	0,034
F6 (2D)	0,996	0,922	0,566
F7 (2D)	0,997	0,882	0,650
F6 (3D)	0,602	0,020	1,000
F7 (3D)	0,778	0,118	1,000
F8 (2D)	0,998	0,980	0,165

Çizelge 6.6. (Devam) 0,0001 optima arası uzaklıkta PR, SR ve AR sonuçları.

FN	PR	SR	AR
F9 (2D)	0,804	0,176	0,973
F10 (2D)	0,914	0,431	0,827
F11 (2D)	0,706	0,118	1,000
F11 (3D)	0,673	0,020	1,000
F12 (3D)	0,326	0,000	1,000
F11 (5D)	0,650	0,000	1,000
F12 (5D)	0,250	0,000	1,000
F11 (10D)	0,350	0,098	1,000
F12 (10D)	0,132	0,000	1,000

Optimalar arası uzaklık 0.0001 ayarlandığında 1 boyutlu problemlerin hepsinde 2 boyutlu problemlerin büyük bir kısmında PR ve SR sonuçlarında yüzde yüz başarı sağlanmıştır.

Çizelge 6.7 0,00001 optima arası uzaklıkta PR, SR ve AR sonuçları.

FN	PR	SR	AR
F1 (1D)	1,000	1,000	0,003
F2 (1D)	1,000	1,000	0,058
F3 (1D)	1,000	1,000	0,015
F4 (2D)	1,000	1,000	0,122
F5 (2D)	1,000	1,000	0,044
F6 (2D)	0,992	0,863	0,615
F7 (2D)	0,985	0,588	0,830
F6 (3D)	0,537	0,020	1,000
F7 (3D)	0,613	0,039	1,000
F8 (2D)	0,997	0,961	0,277
F9 (2D)	0,820	0,294	0,949
F10 (2D)	0,870	0,255	0,893
F11 (2D)	0,706	0,118	1,000
F11 (3D)	0,673	0,020	1,000
F12 (3D)	0,311	0,000	1,000
F11 (5D)	0,650	0,000	1,000
F12 (5D)	0,255	0,000	1,000
F11 (10D)	0,288	0,020	1,000
F12 (10D)	0,127	0,000	1,000
F12 (20D)	0,071	0,000	1,000

Optimalar arası uzaklık 0.00001 ayarlandığında 1 boyutlu problemlerin hepsinde 2 boyutlu problemlerin bir kısmında PR ve SR sonuçlarında yüzde yüz başarı sağlanmıştır.

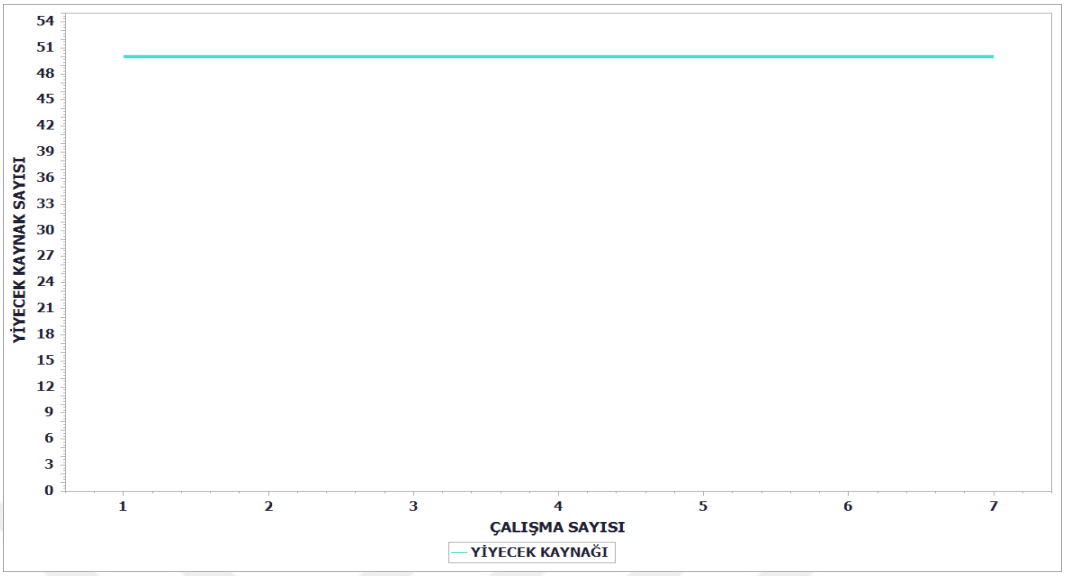
F1 fonksiyonundan F5 fonksiyona kadar tüm optima arası mesafede PR ve SR değeri en yüksek seviyede olup F1 fonksiyonda AR değeri en düşük seviyededir. Optimalar arası uzaklık azaldıkça bulunan optima sayısı azalmakta ve PR ve SR değerleri düşmekte AR değeri artmaktadır. Optima arası mesafe azaldıkça hassaslık artarken çevrelediği alan azaldığından bulunan optima sayısı azalır. Optima sayısındaki düşüş PR ve SR değerini düşmesine AR değerinin artmasına sebep olur.

6.3. Algoritmanın Çalışma Grafikleri

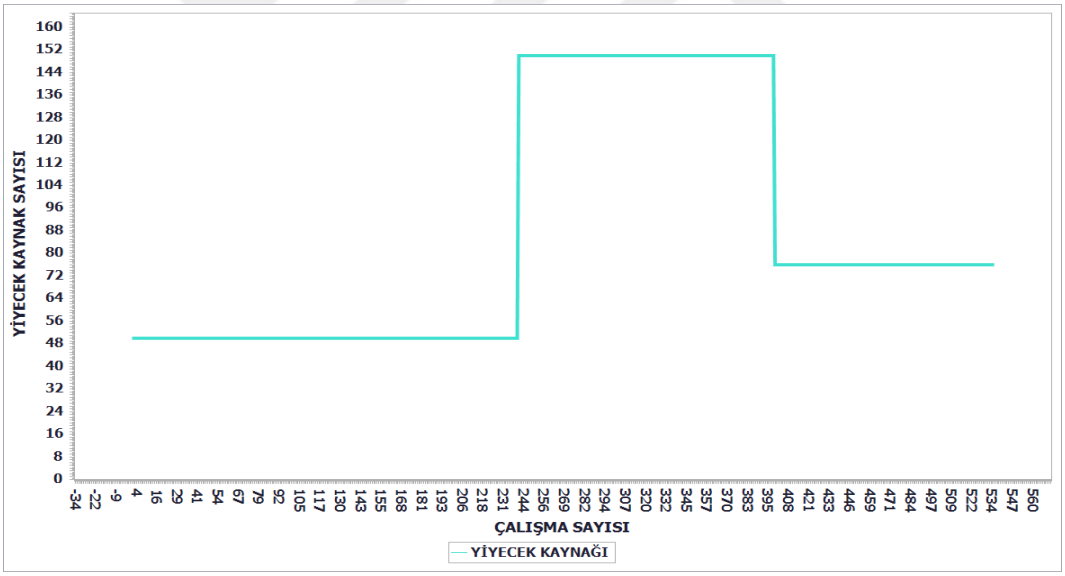
Algoritmanın çalışma esnasındaki durumunu incelemek için parametre ve çözümlerin zamanla değişim grafikleri çizdirilmiştir. Grafikler, her bir fonksiyon için optima arası uzaklık 0.0001 ayarlanarak çizdirilmiştir. Dört farklı grafik çizdirilmiş olup grafik türleri ve grafiklerle ilgili açıklamalar aşağıdaki gibidir.

6.3.1. Yiyecek kaynağının çalıştırma sayısıyla değişimi

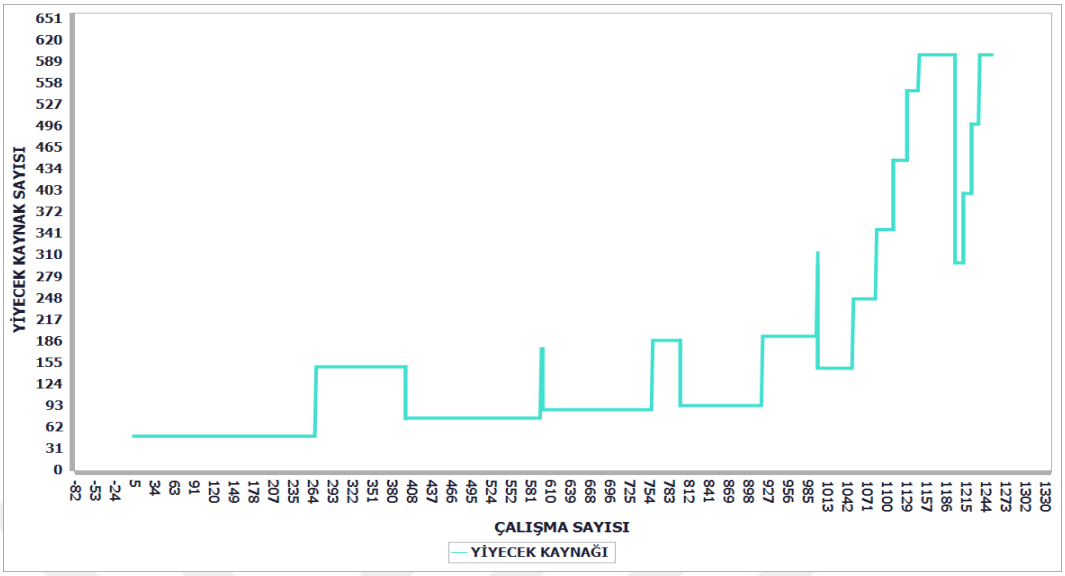
Algoritmamızda yiyecek kaynağı sayısı dinamik bir yapıya sahiptir. Belirli bir değerle başlayan yiyecek kaynağı sayısı iyileştirme durumuna göre değişiklik göstermektedir. Algoritmada yiyecek kaynağı belirli seviyeye geldiğinde kötü olan yiyecek kaynağı elenerek yiyecek kaynağı sayısı düşürülmektedir. Bu sayede algoritma daha az sürede yeni yiyecek kaynaklarını tarar ve iyileştirme vermeyen yiyecek kaynaklarından kurtulmuş olur. Aşağıda problem setindeki bazı fonksiyonların algoritmanın çalıştırılma sayısına göre yiyecek kaynağı sayısını gösteren grafikler yer almaktadır.



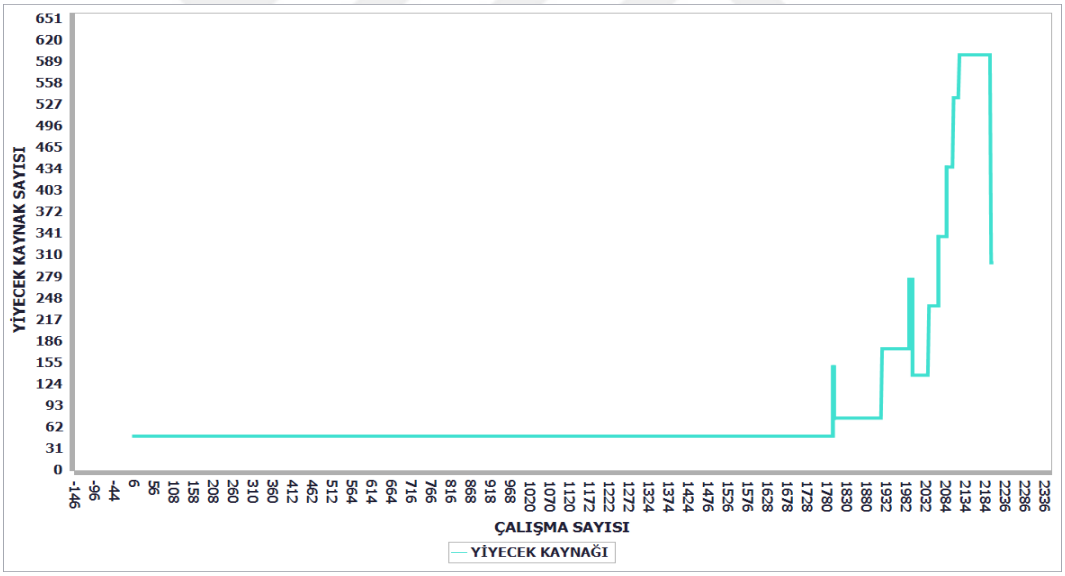
Şekil 6.1. F2 fonksiyonu çalıştırma sayısının yiyecek kaynağıyla değişimi.



Şekil 6.2. F6 fonksiyonu çalıştırma sayısının yiyecek kaynağıyla değişimi.



Şekil 6.3. F14 fonksiyonu çalıştırma sayısının yiyecek kaynağıyla değişimi.



Şekil 6.4. F20 fonksiyonu çalıştırma sayısının yiyecek kaynağıyla değişimi.

F2 fonksiyonunda yiyecek kaynağı sayısı değişimine gerek kalmadan optimaları bulmuştur.

F6 fonksiyonunda optimaları bulabilmek için yiyecek kaynağında değişimler olmuştur. Algoritmamızda 200. çalışmaya kadar istenilen sonucu elde edemediğimiz için algoritmaya yeni yiyecek kaynakları eklenip aramaya devam edilmiştir. 400. çalışmaya gelindiğinde algoritmadaki yiyecek kaynaklarından kötü olanlar elenerek eski değerine düşürülmüştür. Bu sayede

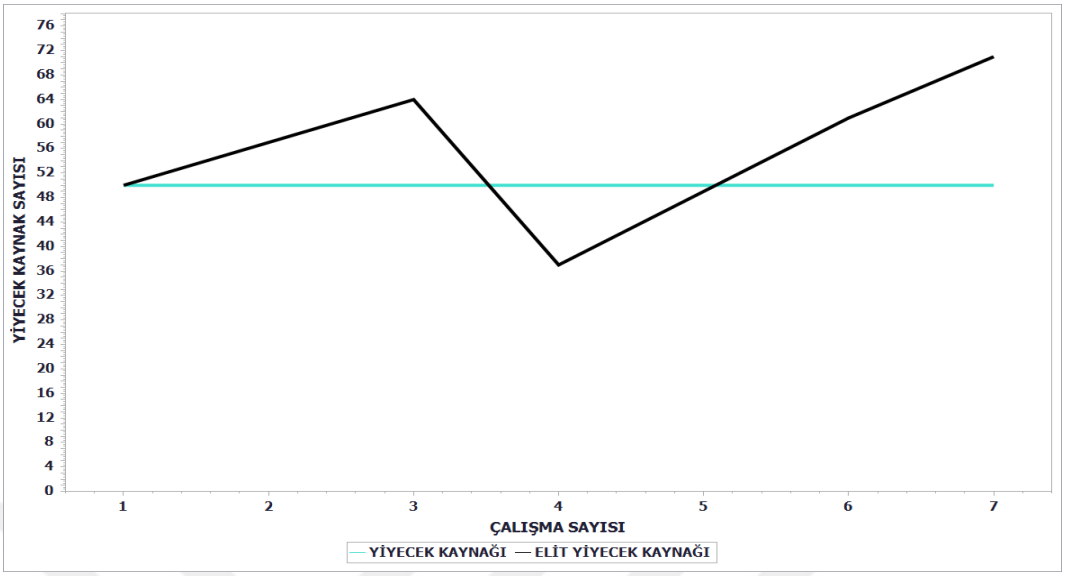
algoritmada daha verimli yiyecek kaynakları kullanılarak çözüme daha hızlı ulaşılmaya çalışılmıştır.

F14 fonksiyonunda optimaları bulabilmek için yiyecek kaynağında değişimler olmuştur. Algoritmamızda 200. çalışmaya kadar istenilen sonuç elde edemediğimiz için algoritmaya yeni yiyecek kaynakları eklenip aramaya devam edilmiştir. 400. Çalışmaya gelindiğinde algoritmadaki yiyecek kaynaklarından kötü olanlar elenerek eski değerine düşürülmüştür. 400. çalışmadan 600. çalışmaya kadar yiyecek kaynağı sabit kalmış, 600. çalışmada yiyecek kaynağı tekrar arttırılmış ve kötü yiyecek kaynakları elenmiştir. 750. çalışmada yiyecek kaynağı tekrar arttırılmış 800 çalışmada eleme yapılmıştır. 900 çalışmadan 1200 çalışmaya kadar yiyecek kaynağındaki artış devam etmiş 1200 çalışmada eleme yapılmıştır 1250 çalışmadan sonra artış devam etmiştir.

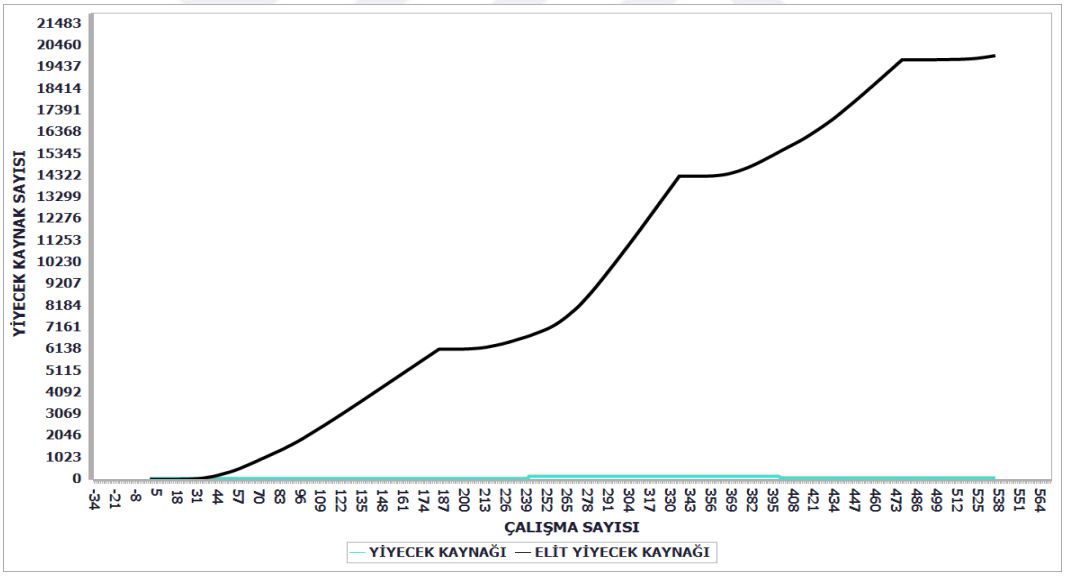
F20 fonksiyonunda optimaları bulabilmek için yiyecek kaynağında değişimler olmuştur. 1800. çalışmaya kadar yiyecek kaynağı aynı kalmış 1800. çalışmada artmaya başlamıştır. 2200. çalışmaya kadar artış devam etmiş olup bu kısımda kötü yiyecek kaynakları elenmiştir.

6.3.2. Yiyecek ve elit yiyecek kaynağının çalıştırma sayısıyla değişimi

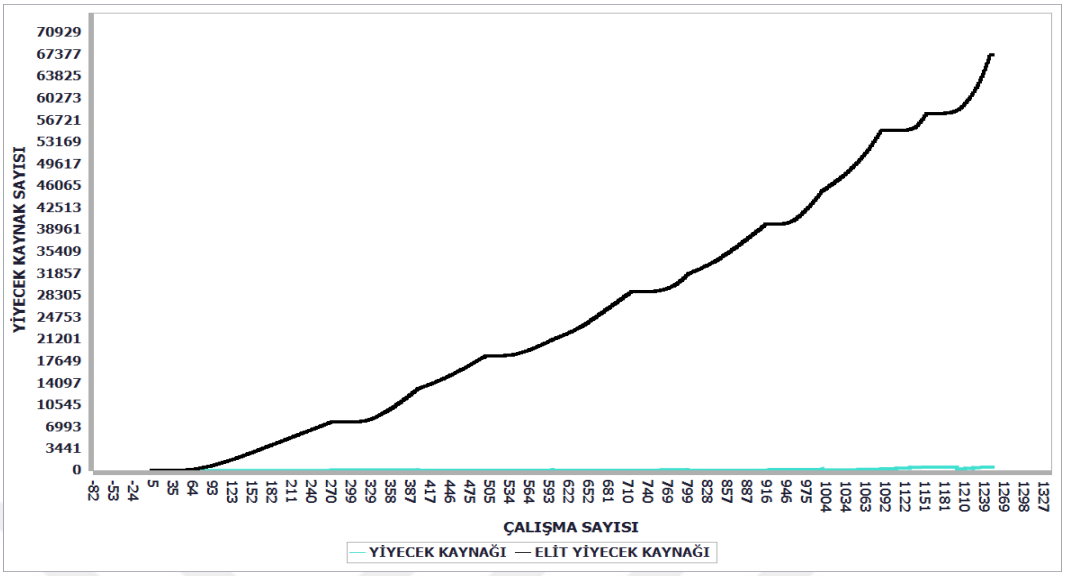
Algoritmamızın yiyecek kaynağının içerisinde bulunup çözüme daha yakın olabileceği düşünülen kaynaklara elit yiyecek kaynağı denir. Elit yiyecek kaynağı sayısı dinamik bir yapıya sahiptir. Belirli bir değerle başlayan elit yiyecek kaynağı sayısı iyileştirme durumuna göre değişiklik göstermektedir. Algoritmada elit yiyecek kaynağı belirli seviyeye geldiğinde kötü olan elit yiyecek kaynağı elenerek elit yiyecek kaynağı sayısı düşürülmektedir. Aşağıda problem setindeki bazı fonksiyonların algoritmanın çalıştırılma sayısına göre yiyecek ve elit yiyecek kaynağı sayısının değişim grafikleri yer almaktadır.



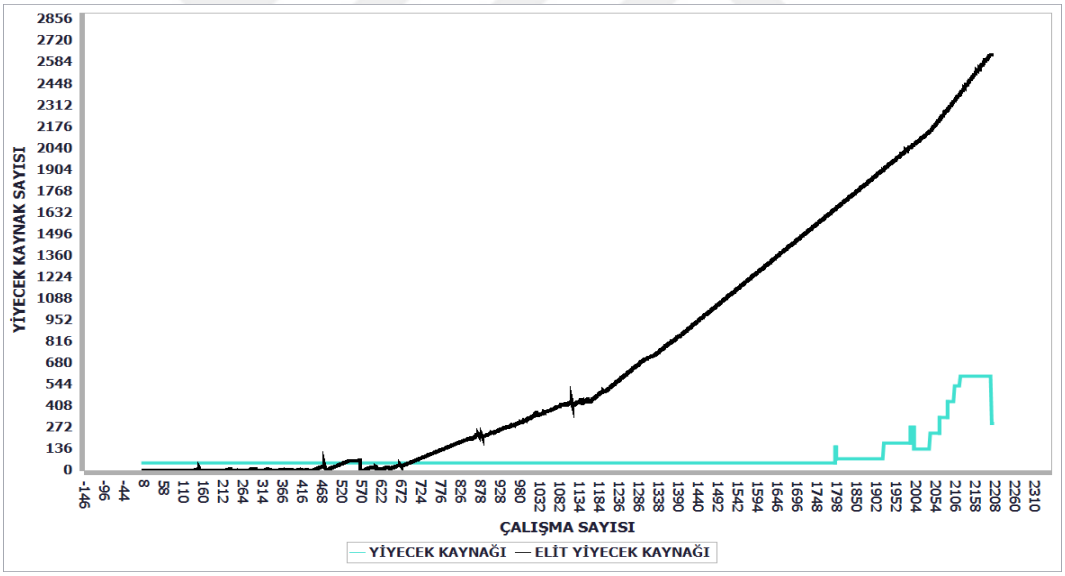
Şekil 6.5. F2 fonksiyonu yiyecek ve elit yiyecek kaynağının değişimi.



Şekil 6.6. F6 fonksiyonu yiyecek ve elit yiyecek kaynağının değişimi.



Şekil 6.7. F14 fonksiyonu yiyecek ve elit yiyecek kaynağının değişimi.



Şekil 6.8. F20 fonksiyonu yiyecek ve elit yiyecek kaynağının değişimi.

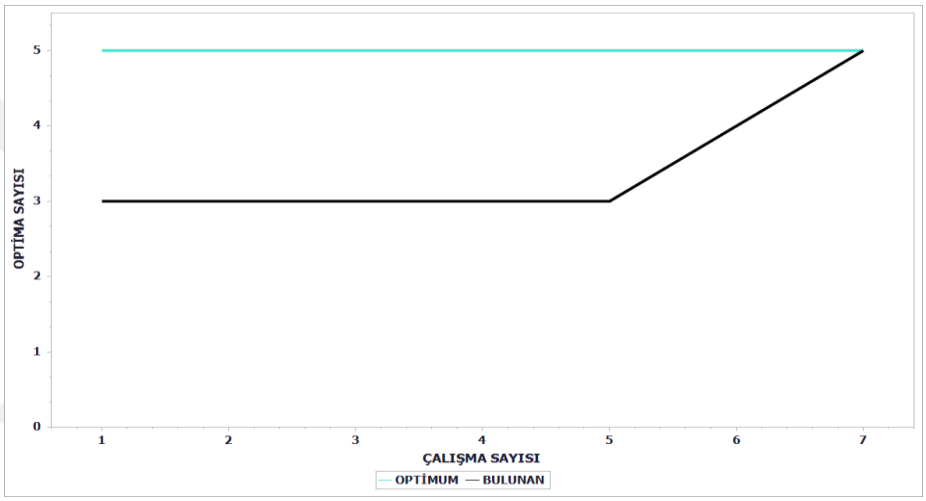
F2 fonksiyonunda elit yiyecek kaynağı değişime uğrarken yiyecek kaynağı sabit kalmıştır. F6, F14 ve F20 fonksiyonlarında elit yiyecek kaynağı sürekli artış halindedir.

Yiyecek kaynaklarının artışı elit yiyecek kaynaklarının artışına göre daha yavaştır. Bunun sebebi yiyecek kaynağının çevresi tam olarak taranmadan arttırılmamaktadır. Çözümü oluşturabilecek veya çözüme yakın sonuçlar içeren tüm yiyecek kaynakları elit yiyecek kaynağına alındığından elit yiyecek kaynağı daha fazla artış göstermektedir.

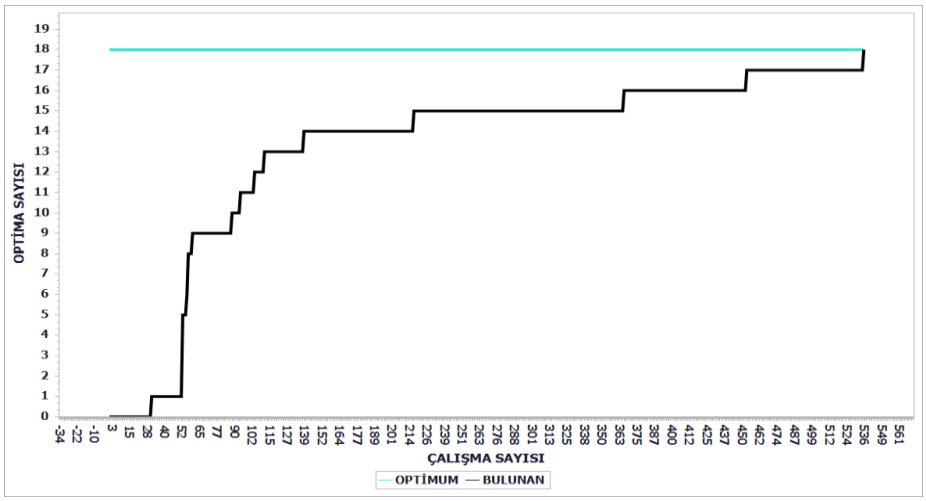
6.3.3. Toplam ve bulunan optima sayısının çalıştırma sayısıyla değişimi

Multimodal problemlerde birden fazla çözüm olabilir. Problem setinde belirtilen çalışma sayısına göre problemin çözüm kümesi aranmaya başlanmaktadır. Belirtilen çalışma sayısından önce tüm optimalar bulunduysa algoritma çalışması durdurulmaktadır. Eğer çalışma sayısında tüm optimalar bulunamazsa en son bulunan optimalar değerlendirmeye alınır.

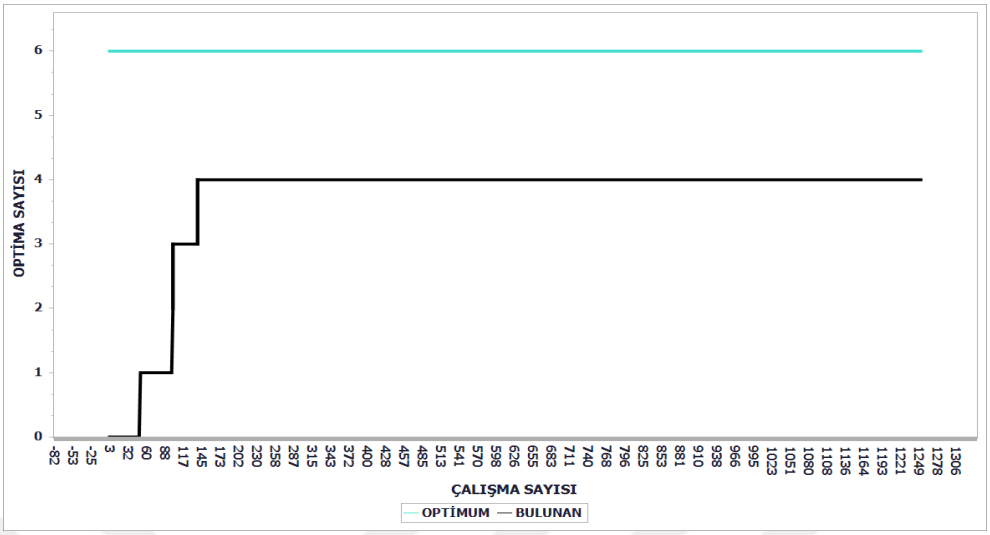
Aşağıda problem setindeki bazı fonksiyonların algoritmanın çalıştırılma sayısına göre toplam optima sayısını gösteren grafikler yer almaktadır.



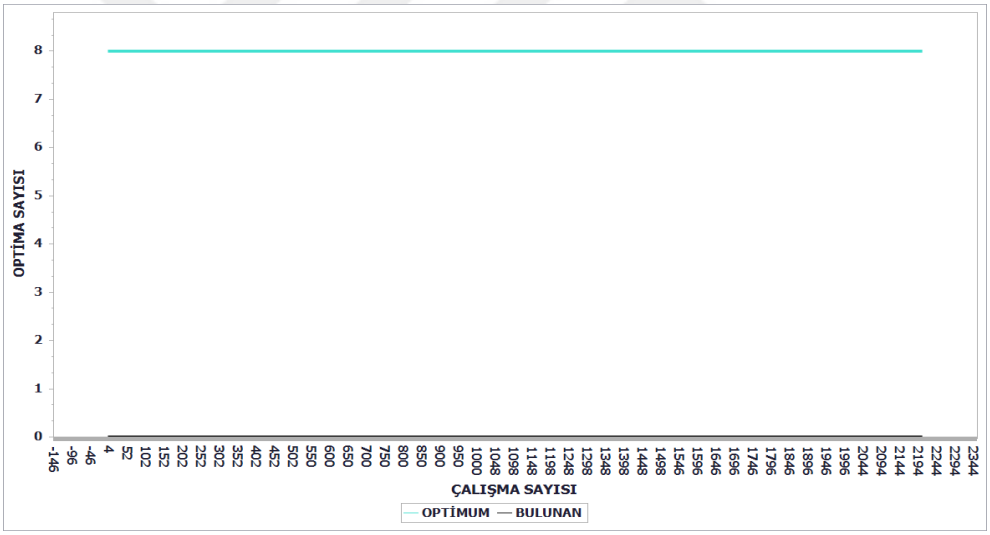
Şekil 6.9. F2 fonksiyonunun toplam ve bulunan optima sayısının çalıştırma sayısıyla değişimi.



Şekil 6.10. F6 fonksiyonunun toplam ve bulunan optima sayısının çalıştırma sayısıyla değişimi.



Şekil 6.11. F14 fonksiyonunun toplam ve bulunan optima sayısının çalıştırma sayısıyla değişimi.



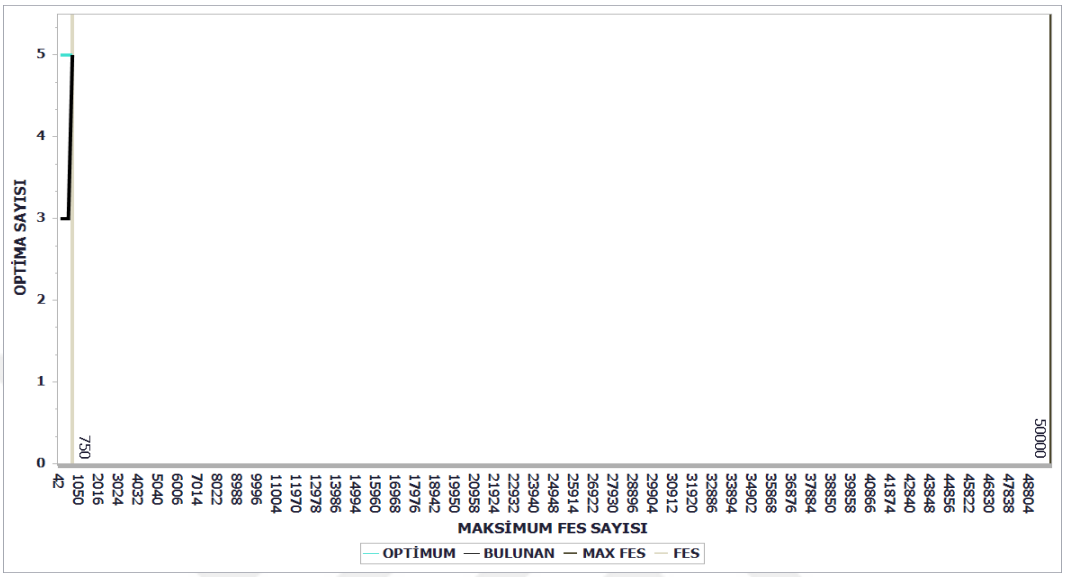
Şekil 6.12. F20 fonksiyonunun toplam ve bulunan optima sayısının çalıştırma sayısıyla değişimi.

Algoritmamız problem setindeki F2 ve F6 problemlerinde optimaların hepsini bulmuştur. F14 probleminde 6 optima noktadan 4 tanesini bulmuş olup, F20 probleminde 8 optima noktadan hiçbirini bulamamıştır.

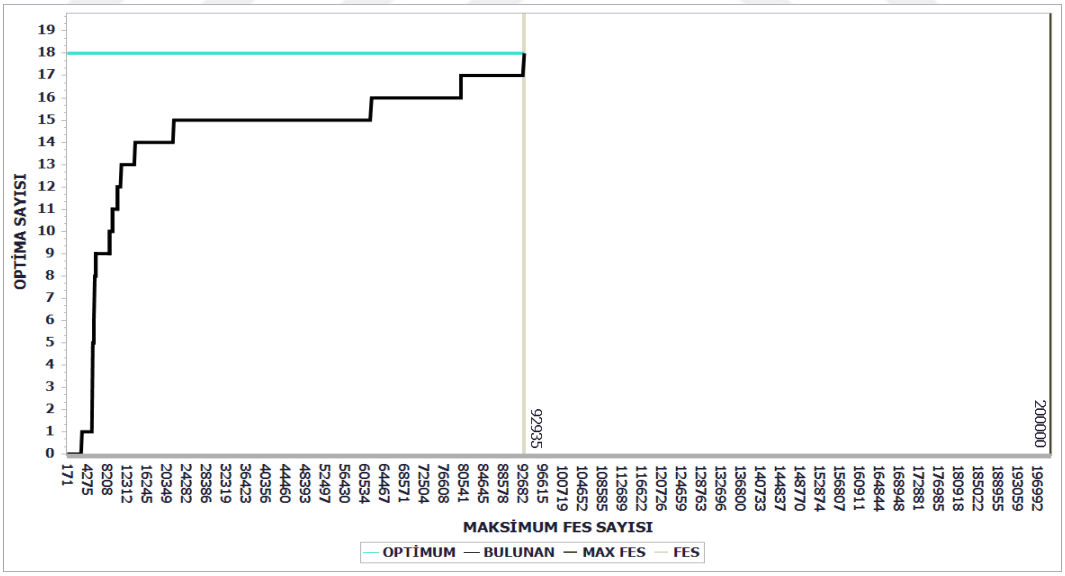
6.3.4. Toplam ve bulunan optima sayısının fes değerine göre değişimi

Problemleri çözmek için kullanılan algoritmaların birbiriyle kıyaslanabilmesi için belirli koşullar gereklidir. Fes koşulu algoritmaların çalışma periyodu veren önemli parametrelerdendir. Algoritma maksimum fes değerinden önce optimaları bulması, algoritmanın hızlı bir şekilde

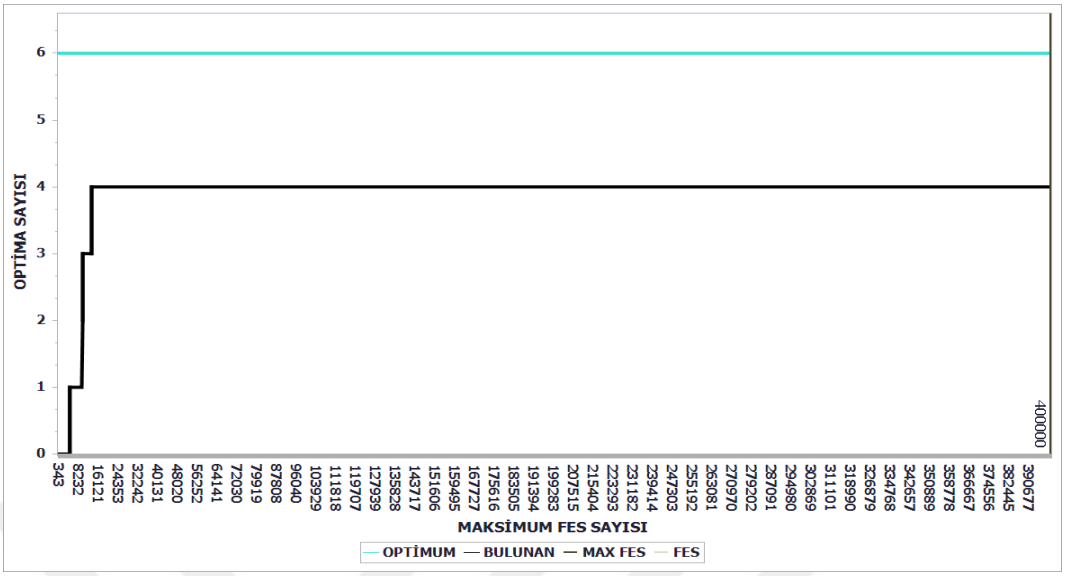
sonuca ulaştığını göstermektedir. Problem setindeki bazı fonksiyonların fes değerlerine göre optimum bulma grafikleri aşağıdaki gibidir.



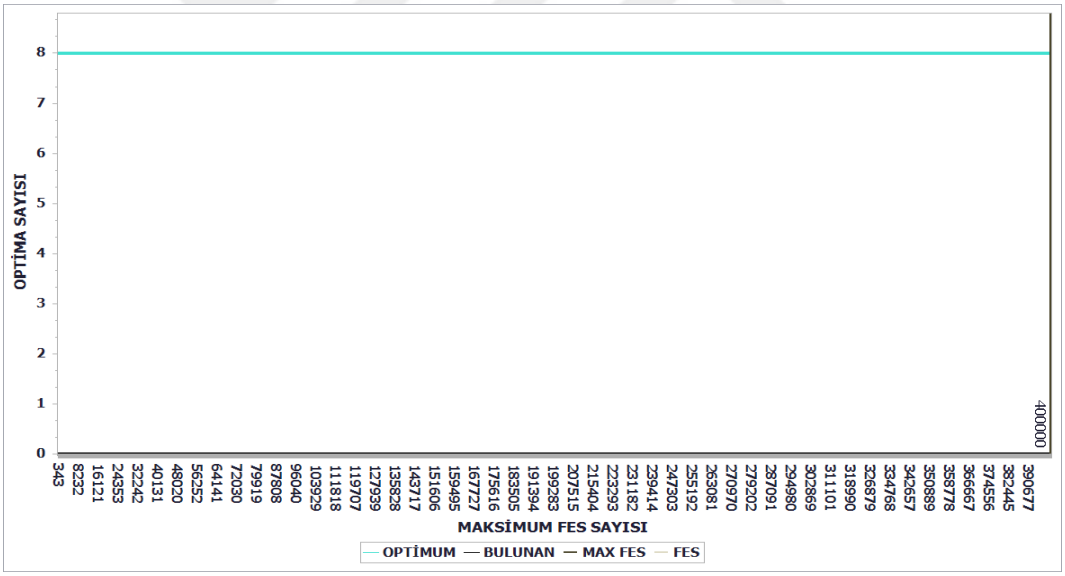
Şekil 6.13. F2 fonksiyonunun toplam ve bulunan optima sayısının fes değeriyle değişimi.



Şekil 6.14. F6 fonksiyonunun toplam ve bulunan optima sayısının fes değeriyle değişimi.



Şekil 6.15. F14 fonksiyonunun toplam ve bulunan optima sayısının fes değeriyle değişimi.



Şekil 6.16. F20 fonksiyonunun toplam ve bulunan optima sayısının fes değeriyle değişimi.

Algoritmamızın 2 numaralı fonksiyonda çok hızlı bir sonuç aldığını görülmektedir. 2 numaralı fonksiyonun maksimum fes değeri 50000 iken algoritmamız 250 fes değerinde 2 optimanın 2 sini bulmuştur. Algoritmamız 6 numaralı fonksiyonda maksimum fes değerine ulaşmadan tüm optimaları bulmuştur. 6 numaralı fonksiyonun maksimum fes değeri 200000 iken algoritmamız 92935 fes değerinde 18 optimanın 18 ini bulmuştur. Algoritmamız 14 numaralı fonksiyonda tüm optimaları bulamamıştır. 14 numaralı fonksiyonunda 400000 fes değerinde 6

optimanın 4 ünü bulmuştur. Algoritmamız 20 numaralı fonksiyonda hiçbir optimayı bulamamıştır. 20 numaralı fonksiyonunda 400000 fes değerinde 8 optimanın 0 ını bulmuştur.

6.4. Algoritmanın Sonucunun Diğer Algoritmalarla Karşılaştırılması

Algoritmanın diğer algoritmalar karşısındaki durumunu belirlemek için çeşitli karşılaştırmaların yapılması gerekmektedir. Aynı problem seti ve aynı kısıtlar kullanılarak bulunan algoritma sonuçları daha önceden belirlenmiş metotlar kullanılarak diğer algoritma sonuçlarıyla karşılaştırılır. Problem setinde belirlenen karşılaştırma metotları PR, SR ve AR değeridir. Aşağıdaki tablolarda PR ve SR değerine göre algoritmamızın diğer algoritmalar karşısında durumunu gösteren tablo ve sonuçlar yer almaktadır.

6.4.1. SABCElit ve diğer algoritmaların PR sonuçları

Çizelge 6.8. SABCElit ve diğer algoritmaların 0.1 uzaklıktaki PR sonuçları.

FN	SABCElit	R3PSO	R2PSO	NCSDE	CDE	ANOF	NRAND	RAND	FERPSO
F1	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
F2	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
F3	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
F4	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
F5	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
F6	1,000	0,653	0,775	0,953	1,000	0,987	0,450	1,000	0,586
F7	1,000	0,264	0,388	0,476	1,000	0,941	0,347	0,703	0,293
F8	0,879	0,177	0,103	0,255	0,649	0,584	0,108	0,847	0,102
F9	1,000	0,061	0,090	0,120	0,275	0,341	0,097	0,271	0,056
F10	1,000	0,915	0,974	1,000	1,000	1,000	1,000	1,000	0,823
F11	1,000	1,000	1,000	1,000	1,000	0,667	0,683	0,937	1,000
F12	0,983	0,380	0,436	1,000	0,353	0,751	0,855	0,380	0,660
F13	0,951	0,788	0,807	0,993	1,000	0,677	0,667	0,837	0,827
F14	1,000	1,000	0,993	1,000	1,000	0,680	0,667	0,683	1,000
F15	1,000	0,625	0,522	0,875	1,000	0,725	0,522	0,730	0,793
F16	1,000	0,922	0,467	1,000	1,000	0,677	0,677	0,697	0,963
F17	1,000	0,154	0,044	0,700	0,907	0,443	0,345	0,567	0,165
F18	1,000	0,350	0,412	1,000	1,000	0,747	0,403	0,517	0,060
F19	0,350	0,015	0,007	0,405	0,000	0,285	0,227	0,000	0,000
F20	0,703	0,051	0,027	0,828	0,064	0,275	0,130	0,502	0,000

Algoritmamız, 15 fonksiyonda 0,99 değerinden daha büyük sonuç vererek en iyi algoritma olmuştur. CDE algoritması 14 fonksiyonda 0,99 değerinden büyük sonuç vererek en iyi ikinci algoritma olmuştur.

Çizelge 6.9. SABCElit ve diğer algoritmaların 0.01 uzaklıktaki PR sonuçları.

FN	SABCElit	R3PSO	R2PSO	NCSDE	CDE	ANOF	NRAND	RAND	FERPSO
F1	1,000	1,000	1,000	1,000	0,710	1,000	1,000	0,710	1,000
F2	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
F3	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
F4	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
F5	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
F6	1,000	0,649	0,706	0,938	0,999	0,982	0,438	0,999	0,582
F7	1,000	0,226	0,308	0,382	0,724	0,961	0,346	0,724	0,293
F8	0,830	0,175	0,056	0,253	0,835	0,564	0,105	0,835	0,098
F9	0,988	0,056	0,085	0,097	0,272	0,338	0,095	0,272	0,056
F10	1,000	0,869	0,940	1,000	1,000	1,000	1,000	1,000	0,807
F11	0,912	0,595	0,631	0,920	0,690	0,667	0,673	0,690	0,763
F12	0,968	0,373	0,395	1,000	0,055	0,745	0,837	0,055	0,660
F13	0,712	0,582	0,601	0,707	0,683	0,667	0,667	0,683	0,643
F14	0,680	0,572	0,490	0,670	0,667	0,667	0,667	0,667	0,590
F15	0,419	0,159	0,147	0,625	0,690	0,725	0,535	0,690	0,260
F16	0,657	0,369	0,134	0,660	0,667	0,667	0,663	0,667	0,303
F17	0,262	0,076	0,015	0,375	0,425	0,463	0,325	0,425	0,060
F18	0,386	0,075	0,065	0,390	0,250	0,580	0,343	0,250	0,010
F19	0,157	0,015	0,002	0,338	0,000	0,295	0,167	0,000	0,000
F20	0,110	0,034	0,015	0,210	0,013	0,203	0,127	0,013	0,000

Algoritmamız, 8 fonksiyonda 0,99 değerinden daha büyük sonuç vererek en iyi algoritma olmuştur. NCDSE algoritması 7 fonksiyonda 0,99 değerinden büyük sonuç vererek en iyi ikinci algoritma olmuştur.

Çizelge 6.10. SABCElit ve diğer algoritmaların 0.001 uzaklıktaki PR sonuçları.

FN	SABCElit	R3PSO	R2PSO	NCSDE	CDE	ANOF	NRAND	RAND	FERPSO
F1	1,000	1,000	1,000	1,000	0,090	1,000	1,000	0,090	1,000
F2	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
F3	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
F4	1,000	0,985	0,990	1,000	1,000	1,000	1,000	1,000	1,000
F5	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
F6	0,999	0,646	0,629	0,937	0,972	0,981	0,440	0,972	0,577
F7	1,000	0,224	0,303	0,364	0,715	0,944	0,349	0,715	0,292
F8	0,741	0,172	0,030	0,251	0,716	0,562	0,113	0,716	0,093
F9	0,928	0,056	0,078	0,090	0,274	0,340	0,099	0,274	0,056
F10	1,000	0,848	0,912	1,000	1,000	1,000	0,998	1,000	0,807
F11	0,830	0,595	0,605	0,920	0,667	0,667	0,683	0,667	0,763
F12	0,926	0,373	0,380	1,000	0,007	0,745	0,815	0,007	0,653
F13	0,712	0,582	0,582	0,707	0,667	0,667	0,667	0,667	0,643
F14	0,673	0,572	0,438	0,670	0,667	0,667	0,667	0,667	0,587
F15	0,336	0,157	0,137	0,613	0,627	0,730	0,507	0,627	0,260
F16	0,657	0,369	0,108	0,660	0,667	0,667	0,663	0,667	0,300
F17	0,255	0,076	0,002	0,353	0,280	0,443	0,295	0,280	0,060
F18	0,324	0,075	0,052	0,377	0,200	0,593	0,323	0,200	0,010
F19	0,159	0,015	0,000	0,333	0,000	0,300	0,152	0,000	0,000
F20	0,096	0,034	0,007	0,208	0,000	0,158	0,130	0,000	0,000

Algoritmamız, 8 fonksiyonda 0,99 değerinden daha büyük sonuç vererek en iyi algoritma olmuştur. NCDSE algoritması 7 fonksiyonda 0,99 değerinden büyük sonuç vererek en iyi ikinci algoritma olmuştur.

Çizelge 6.11. SABCElit ve diğer algoritmaların 0.0001 uzaklıktaki PR sonuçları.

FN	SABCElit	R3PSO	R2PSO	NCSDE	CDE	ANOF	NRAND	RAND	FERPSO
F1	1,000	1,000	1,000	1,000	1,000	1,000	1,000	0,020	1,000
F2	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
F3	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
F4	1,000	0,971	0,956	1,000	0,975	1,000	1,000	0,995	1,000
F5	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
F6	0,996	0,642	0,540	0,933	0,656	0,980	0,434	0,107	0,571
F7	0,997	0,221	0,290	0,352	0,712	0,951	0,337	0,709	0,290
F8	0,602	0,171	0,016	0,250	0,567	0,558	0,112	0,290	0,091
F9	0,778	0,056	0,062	0,087	0,273	0,341	0,095	0,274	0,055
F10	0,998	0,833	0,877	1,000	1,000	1,000	1,000	1,000	0,805
F11	0,804	0,595	0,569	0,920	0,667	0,667	0,673	0,667	0,763
F12	0,914	0,373	0,350	1,000	0,000	0,745	0,815	0,007	0,645
F13	0,706	0,582	0,565	0,703	0,667	0,667	0,667	0,667	0,643
F14	0,673	0,569	0,350	0,670	0,667	0,667	0,667	0,667	0,580
F15	0,326	0,157	0,132	0,613	0,561	0,713	0,502	0,490	0,260
F16	0,650	0,369	0,065	0,660	0,667	0,667	0,663	0,667	0,300
F17	0,250	0,076	0,000	0,340	0,005	0,465	0,290	0,115	0,060
F18	0,350	0,075	0,042	0,373	0,193	0,617	0,270	0,173	0,010
F19	0,132	0,015	0,000	0,325	0,000	0,298	0,125	0,000	0,000
F20	0,086	0,034	0,000	0,205	0,000	0,168	0,125	0,000	0,000

Algoritmamız, 8 fonksiyonda 0,99 değerinden daha büyük sonuç vererek en iyi algoritma olmuştur. NCDSE algoritması 7 fonksiyonda 0,99 değerinden büyük sonuç vererek en iyi ikinci algoritma olmuştur.

Çizelge 6.12. SABCElit ve diğer algoritmaların 0.00001 uzaklıktaki PR sonuçları.

FN	SABCElit	R3PSO	R2PSO	NCSDE	CDE	ANOF	NRAND	RAND	FERPSO
F1	1,000	1,000	1,000	1,000	1,000	1,000	1,000	0,000	1,000
F2	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
F3	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
F4	1,000	0,971	0,951	1,000	0,299	1,000	1,000	0,420	1,000
F5	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
F6	0,992	0,641	0,466	0,929	0,168	0,000	0,000	0,000	0,569
F7	0,985	0,221	0,271	0,343	0,712	0,944	0,333	0,716	0,289
F8	0,537	0,169	0,010	0,249	0,425	0,554	0,113	0,038	0,087
F9	0,613	0,055	0,048	0,086	0,273	0,341	0,094	0,270	0,055
F10	0,997	0,827	0,824	1,000	1,000	1,000	1,000	1,000	0,793
F11	0,820	0,595	0,526	0,920	0,667	0,667	0,670	0,667	0,763
F12	0,870	0,373	0,316	1,000	0,000	0,743	0,777	0,002	0,643
F13	0,706	0,582	0,526	0,700	0,667	0,667	0,667	0,667	0,640
F14	0,673	0,565	0,291	0,670	0,667	0,667	0,667	0,667	0,580
F15	0,311	0,157	0,125	0,605	0,453	0,725	0,507	0,375	0,260
F16	0,650	0,369	0,052	0,660	0,667	0,667	0,657	0,667	0,297
F17	0,255	0,076	0,000	0,338	0,000	0,475	0,287	0,047	0,060
F18	0,288	0,075	0,039	0,360	0,176	0,593	0,250	0,170	0,010
F19	0,127	0,015	0,000	0,310	0,000	0,300	0,127	0,000	0,000
F20	0,071	0,034	0,000	0,205	0,000	0,178	0,123	0,000	0,000

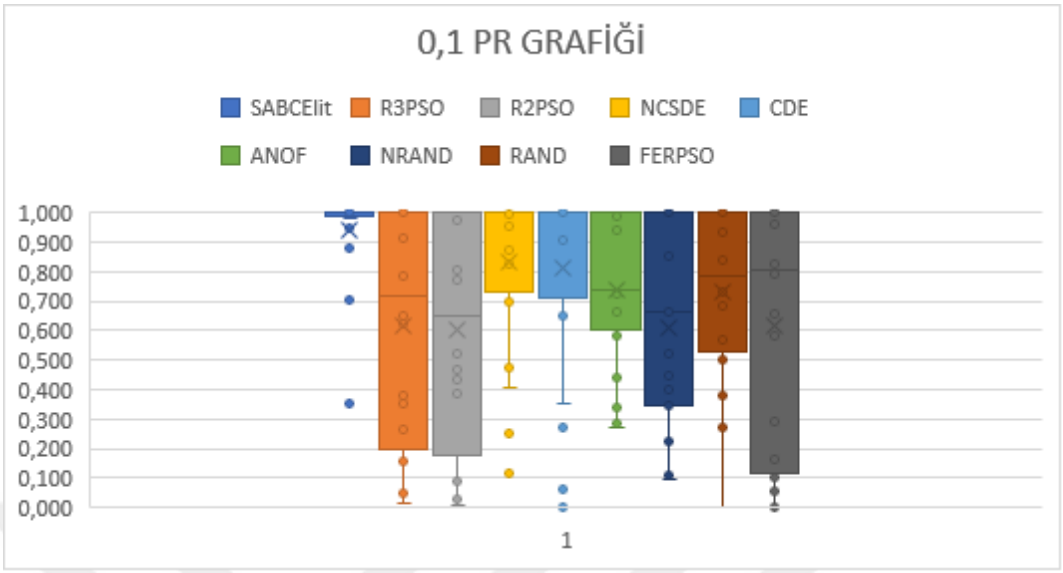
Algoritmamız ve NCDSE algoritması, 7 fonksiyonda 0,99 değerinden daha büyük sonuç vererek en iyi algoritmalar olmuştur.

PR sonuçlarında algoritmamız toplam 46 noktada 0,99 değerinden daha büyük sonuca ulaşarak en iyi algoritma olmuştur. NCDSE algoritması, PR sonuçlarında toplam 44 noktada 0,99 değerinden daha büyük sonuca ulaşarak en iyi ikinci algoritma olmuştur.

6.4.2. SABCElit ve diğer algoritmaların PR grafikleri

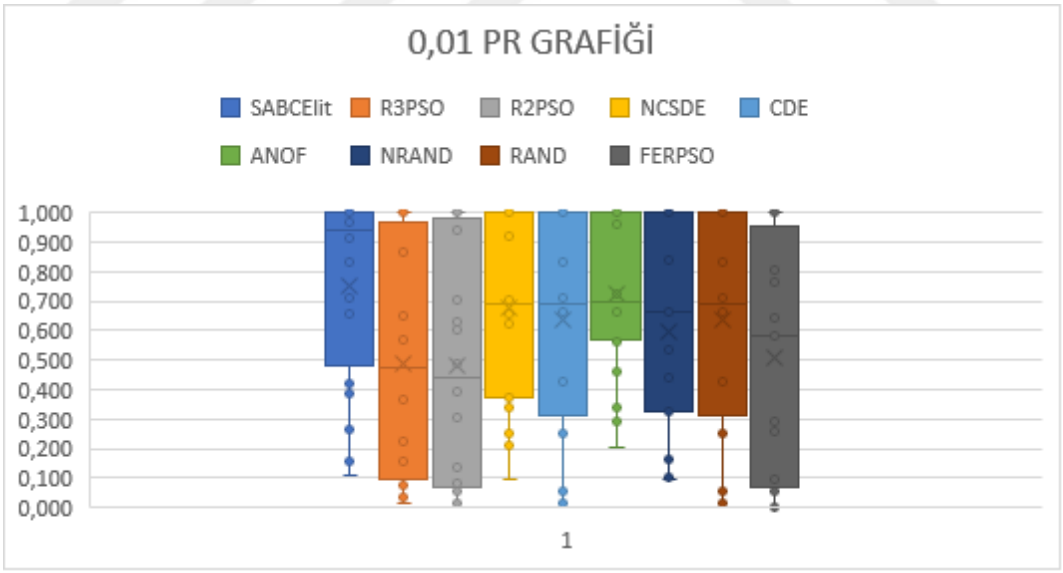
Kutu grafiği

Bir değişkenin sıklık dağılımını göstermek için kutu grafiği kullanılır. Kutu grafikleri değerlerin hangi noktada kümelendiği konusunda bize bilgi vermektedir. Aşağıda algoritmamızın ve diğer algoritmaların kutu grafikleri gösterilmiştir.



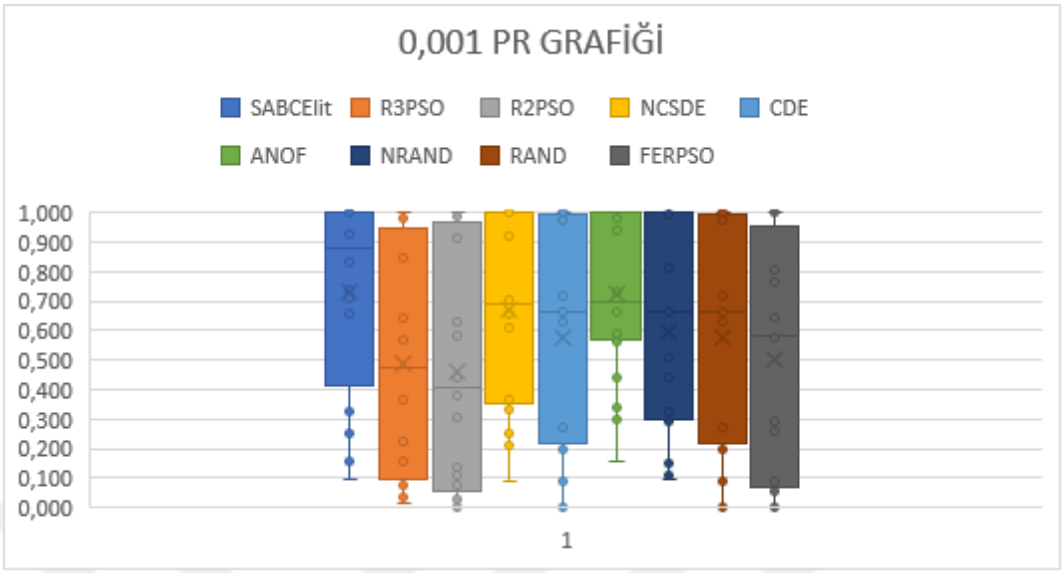
Şekil 6.17. SABCElit ve diğer algoritmaların 0.1 uzaklıktaki PR kutu grafiği.

Algoritmamız 0.1 uzaklıkta daha çok 0.9 ile 1.00 arasında sonuçlara sahiptir. Algoritmamıza en yakın sonuç 0.75 ile 1 arasında sonuçlara sahip olan NCDSE algoritmasıdır.



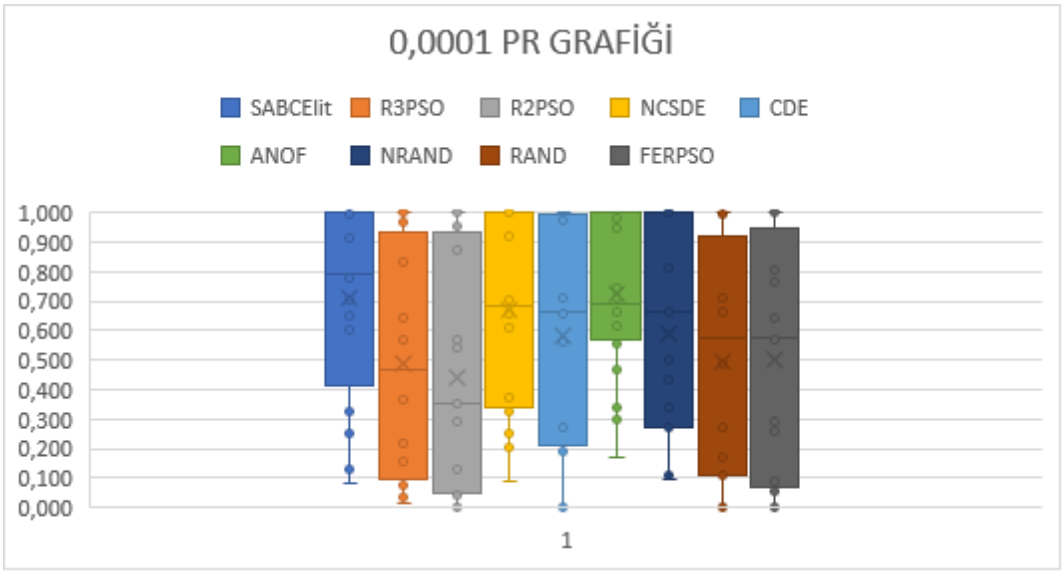
Şekil 6.18. SABCElit ve diğer algoritmaların 0.01 uzaklıktaki PR kutu grafiği.

Algoritmamız 0.01 uzaklıkta daha çok 0.4 ile 1 arasında sonuçlara sahiptir. Sonuçların dağılıma göre en iyi algoritma 0.6 ile 1 arasında dağılım gösteren ANOF algoritmasıdır. Algoritmamız ANOF algoritmasından sonra en iyi ikinci dağılama sahiptir.



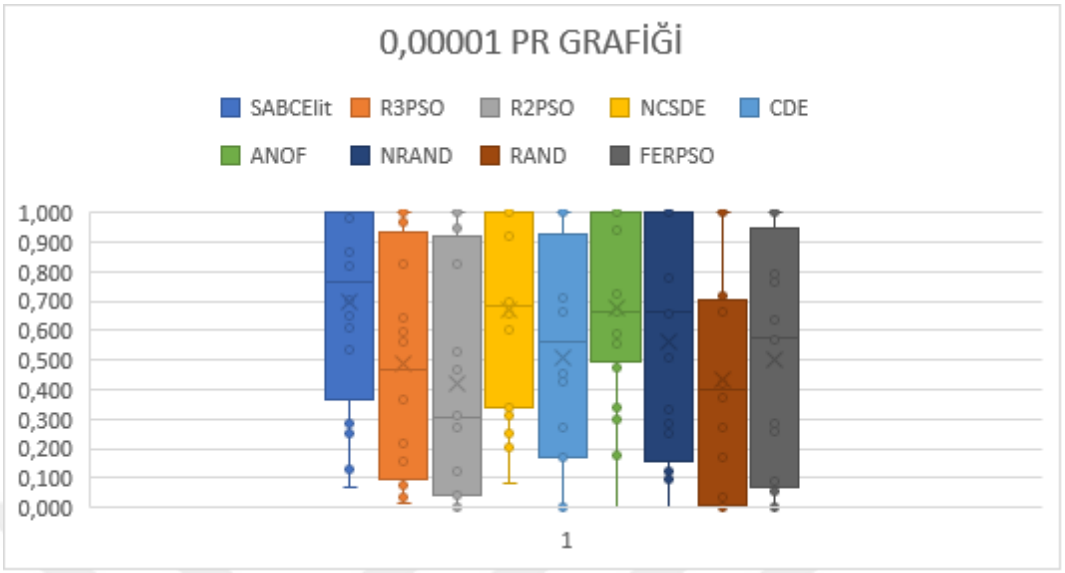
Şekil 6.19. SABCElit ve diğer algoritmaların 0.001 uzaklıktaki PR kutu grafiği.

Algoritmamız 0.001 uzaklıkta daha çok 0.4 ile 1 arasında sonuçlara sahiptir. Sonuçların dağılıma göre en iyi algoritma 0.6 ile 1 arasında dağılım gösteren ANOF algoritmasıdır. Algoritmamız ANOF algoritmasından sonra en iyi ikinci dağılama sahiptir.



Şekil 6.20. SABCElit ve diğer algoritmaların 0,0001 uzaklıktaki pr kutu grafiği.

Algoritmamız 0.0001 uzaklıkta daha çok 0.4 ile 1 arasında sonuçlara sahiptir. Sonuçların dağılıma göre en iyi algoritma 0.55 ile 1 arasında dağılım gösteren ANOF algoritmasıdır. Algoritmamız ANOF algoritmasından sonra en iyi ikinci dağılama sahiptir.

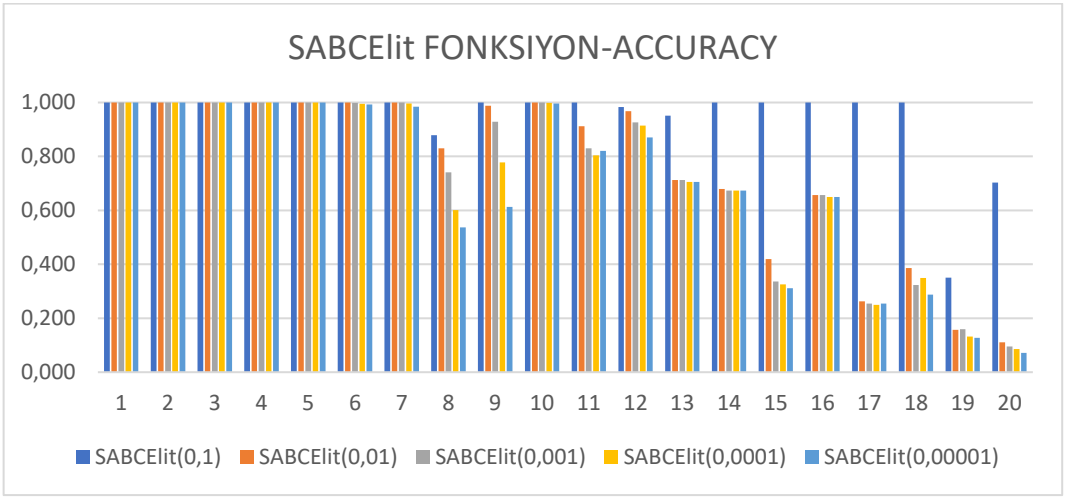


Şekil 6.21. SABCElit ve diğer algoritmaların 0,00001 uzaklıktaki pr kutu grafiği.

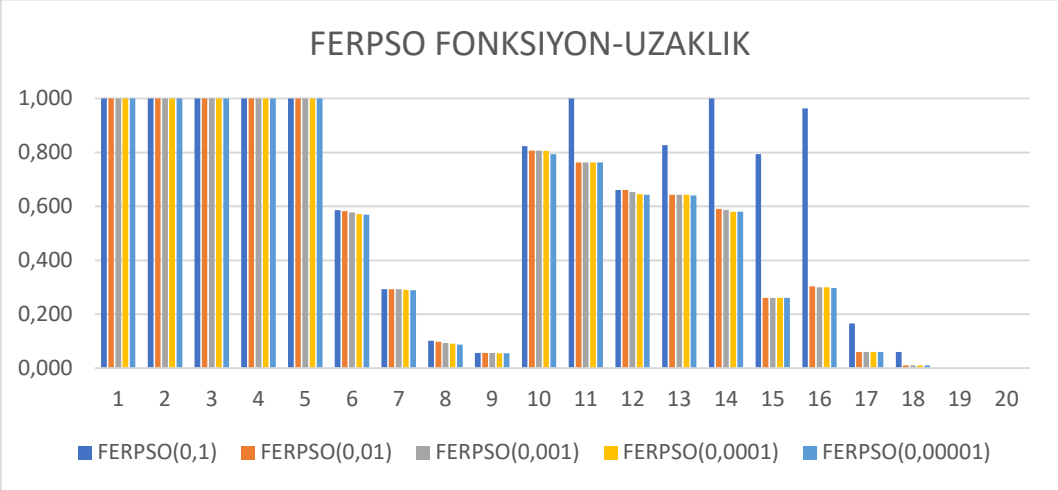
Algoritmamız 0.00001 uzaklıkta daha çok 0.38 ile 1 arasında sonuçlara sahiptir. Sonuçların dağılıma göre en iyi algoritma 0.48 ile 1 arasında dağılım gösteren ANOF algoritmasıdır. Algoritmamız ANOF algoritmasından sonra en iyi ikinci dağılama sahiptir.

Bar grafiği

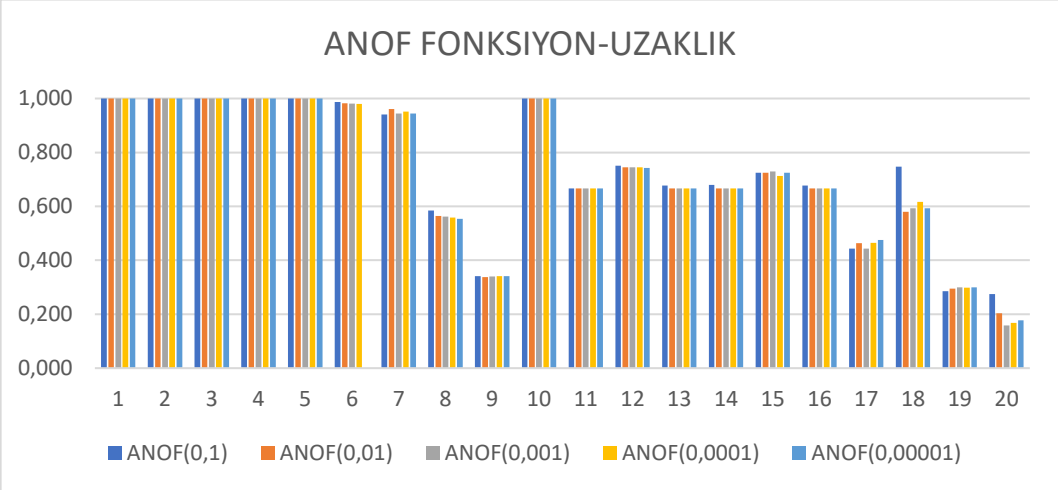
Uzaklık değerine göre algoritmalın sonuçları değişmektedir. Bar grafiğiyle algoritmalın uzaklık PR değeri değişimi aşağıdaki grafikte verilmiştir.



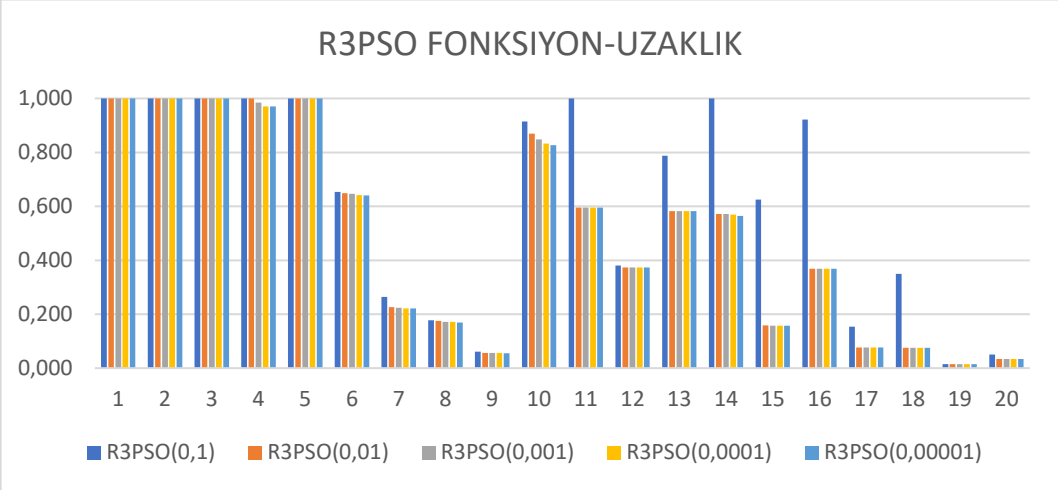
Şekil 6.22. SABCElit fonksiyon- uzaklık grafiği.



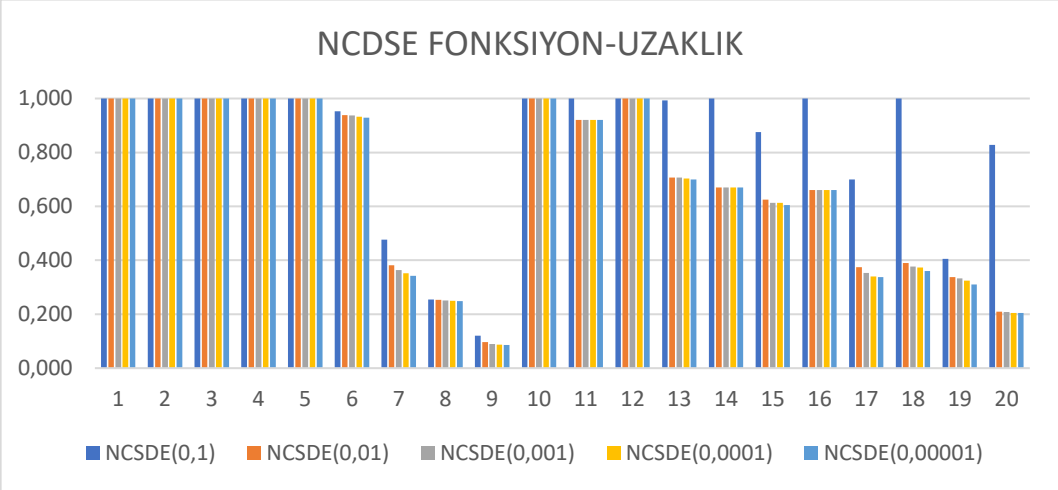
Şekil 6.23. FERPSO fonksiyon- uzaklık grafiği.



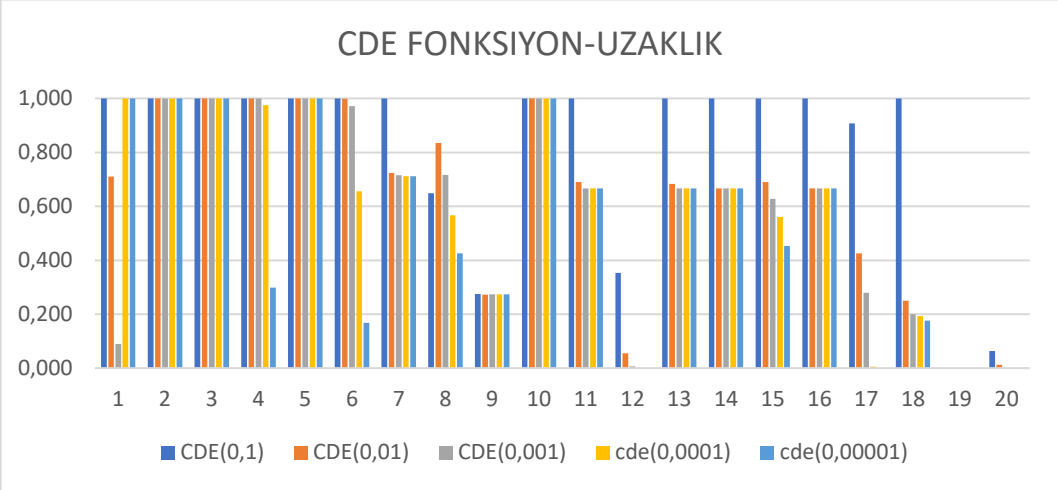
Şekil 6.24. ANOF fonksiyon- uzaklık grafiği.



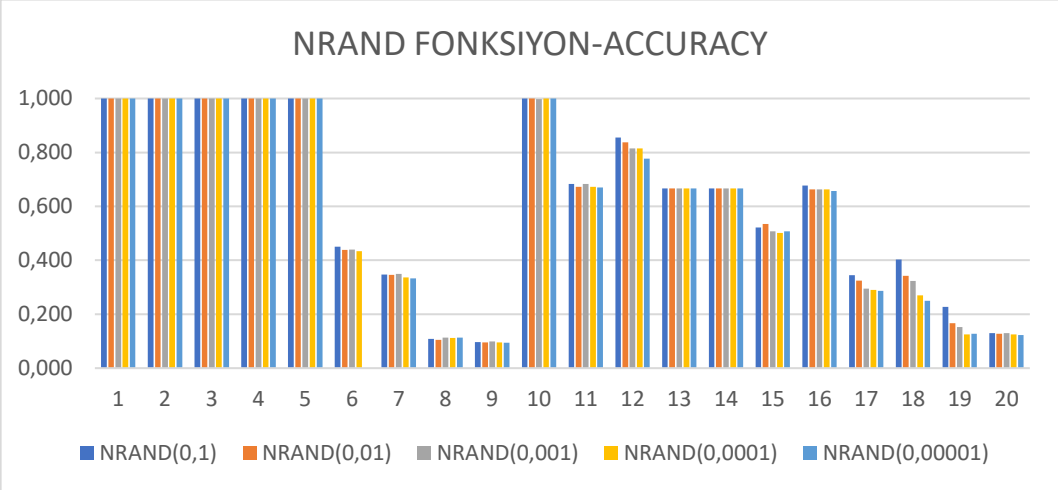
Şekil 6.25. R3PSO fonksiyon- uzaklık grafiği.



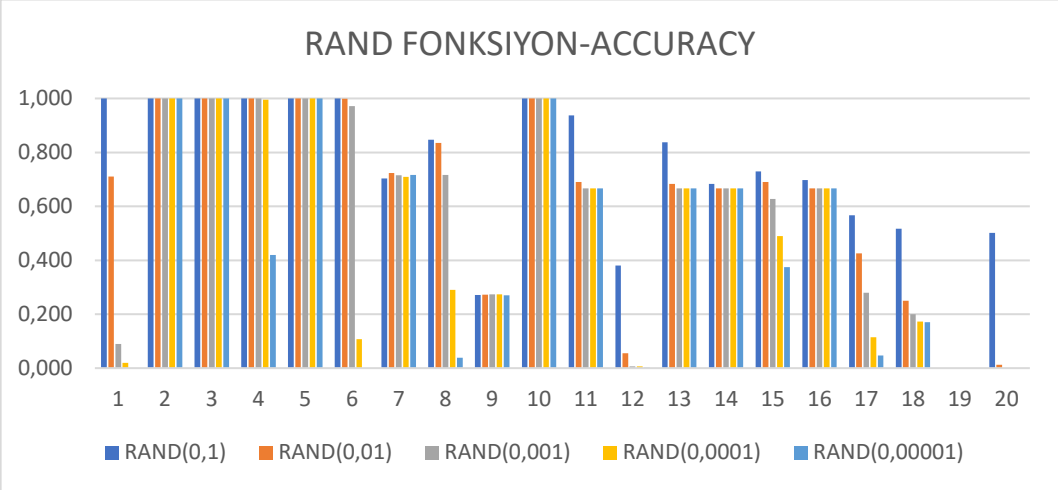
Şekil 6.26. NCDSE fonksiyon- uzaklık grafiği.



Şekil 6.27. CDE fonksiyon- uzaklık grafiği.



Şekil 6.28. NRAND fonksiyon-uzaklık grafiği.



Şekil 6.29. RAND fonksiyon- uzaklık grafiği.

Algoritmamız 1, 2, 3, 4, 5, 6, 7 ve 12 numaralı fonksiyonlarda uzaklık parametresi değişse bile bire yakın değerlerde sonuç vermiştir. NCDSE algoritması 1, 2, 3, 4, 5, 10 ve 12 numaralı fonksiyonlarda uzaklık parametresi değişse bile bire yakın değerlerde sonuç vermiştir.7 fonksiyonda iki algoritmanın uzaklık parametresinin değişmesi sonuca etki etmemiştir. 10. fonksiyonda algoritmamız NCDSE algoritmasına çok yakın sonuç vermiştir.

6.4.3. SABCElit ve diğer algoritmaların SR sonuçları

Çizelge 6.13. SABCElit ve diğer algoritmaların 0.1 uzaklıktaki SR sonuçları.

FN	SABCElit	R3PSO	R2PSO	NCSDE	CDE	ANOF	NRAND	RAND	FERPSO
F1	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
F2	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
F3	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
F4	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
F5	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
F6	1,000	0,000	0,000	0,560	1,000	0,780	0,000	1,000	0,000
F7	1,000	0,020	0,078	0,000	1,000	0,160	0,000	0,000	0,000
F8	0,039	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
F9	1,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
F10	1,000	0,627	0,843	1,000	1,000	1,000	1,000	1,000	0,240
F11	1,000	1,000	1,000	1,000	1,000	0,000	0,000	0,720	1,000
F12	0,863	0,000	0,000	1,000	0,000	0,000	0,240	0,000	0,000
F13	0,784	0,529	0,510	0,980	1,000	0,000	0,000	0,400	0,540
F14	1,000	1,000	0,980	1,000	1,000	0,040	0,000	0,000	1,000
F15	1,000	0,569	0,373	0,700	1,000	0,000	0,000	0,000	0,740
F16	1,000	0,902	0,275	1,000	1,000	0,000	0,000	0,000	0,960
F17	1,000	0,098	0,000	0,480	0,804	0,000	0,000	0,080	0,120
F18	1,000	0,333	0,392	1,000	1,000	0,400	0,000	0,080	0,060
F19	0,157	0,000	0,000	0,100	0,000	0,000	0,000	0,000	0,000
F20	0,294	0,020	0,000	0,800	0,059	0,021	0,000	0,380	0,000

Algoritmamız, 15 fonksiyonda 0,99 değerinden daha büyük sonuç vererek en iyi algoritma olmuştur. CDE algoritması 14 fonksiyonda 0,99 değerinden büyük sonuç vererek en iyi ikinci algoritma olmuştur.

Çizelge 6.14. SABCElit ve diğer algoritmaların 0.01 uzaklıktaki SR sonuçları.

FN	SABCElit	R3PSO	R2PSO	NCSDE	CDE	ANOF	NRAND	RAND	FERPSO
F1	1,000	1,000	1,000	1,000	1,000	1,000	1,000	0,500	1,000
F2	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
F3	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
F4	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
F5	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
F6	1,000	0,000	0,000	0,440	1,000	0,720	0,000	0,980	0,000
F7	1,000	0,000	0,000	0,000	0,000	0,220	0,000	0,000	0,000
F8	0,039	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
F9	0,373	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
F10	1,000	0,392	0,588	1,000	1,000	1,000	1,000	1,000	0,100
F11	0,569	0,000	0,000	0,540	0,000	0,000	0,000	0,040	0,040
F12	0,765	0,000	0,000	1,000	0,000	0,000	0,220	0,000	0,000
F13	0,137	0,000	0,000	0,000	0,000	0,000	0,000	0,020	0,000
F14	0,039	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
F15	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
F16	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
F17	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
F18	0,098	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
F19	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
F20	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000

Algoritmamız, 8 fonksiyonda 0,99 değerinden daha büyük sonuç vererek en iyi algoritma olmuştur. CDE algoritması 7 fonksiyonda 0,99 değerinden büyük sonuç vererek en iyi ikinci algoritma olmuştur.

Çizelge 6.15. SABCElit ve diğer algoritmaların 0.001 uzaklıktaki SR sonuçları.

FN	SABCElit	R3PSO	R2PSO	NCSDE	CDE	ANOF	NRAND	RAND	FERPSO
F1	1,000	1,000	1,000	1,000	1,000	1,000	1,000	0,000	1,000
F2	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
F3	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
F4	1,000	0,941	0,961	1,000	1,000	1,000	1,000	1,000	1,000
F5	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
F6	0,980	0,000	0,000	0,440	0,647	0,720	0,000	0,740	0,000
F7	1,000	0,000	0,000	0,000	0,000	0,060	0,000	0,000	0,000
F8	0,098	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
F9	0,275	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
F10	1,000	0,255	0,392	1,000	1,000	1,000	0,980	1,000	0,100
F11	0,353	0,000	0,000	0,540	0,000	0,000	0,000	0,000	0,040
F12	0,510	0,000	0,000	1,000	0,000	0,000	0,140	0,000	0,000
F13	0,137	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
F14	0,020	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
F15	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
F16	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
F17	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
F18	0,059	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
F19	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
F20	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000

Algoritmamız ve NCDSE algoritması, 7 fonksiyonda 0,99 değerinden daha büyük sonuç vererek en iyi algoritmalar olmuştur.

Çizelge 6.16. SABCElit ve diğer algoritmaların 0.0001 uzaklıktaki SR sonuçları.

FN	SABCElit	R3PSO	R2PSO	NCSDE	CDE	ANOF	NRAND	RAND	FERPSO
F1	1,000	1,000	1,000	1,000	1,000	1,000	1,000	0,000	1,000
F2	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
F3	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
F4	1,000	0,882	0,824	1,000	0,902	1,000	1,000	0,980	1,000
F5	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
F6	0,922	0,000	0,000	0,420	0,020	0,700	0,000	0,000	0,000
F7	0,882	0,000	0,000	0,000	0,000	0,260	0,000	0,000	0,000
F8	0,020	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
F9	0,118	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
F10	0,980	0,176	0,255	1,000	1,000	1,000	1,000	1,000	0,080
F11	0,176	0,000	0,000	0,540	0,000	0,000	0,000	0,000	0,040
F12	0,431	0,000	0,000	1,000	0,000	0,000	0,160	0,000	0,000
F13	0,118	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
F14	0,020	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
F15	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
F16	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
F17	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
F18	0,098	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
F19	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
F20	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000

Algoritmamız, 5 fonksiyonda 0,99 değerinden daha büyük sonuç vererek, en iyi dördüncü algoritma olmuştur. NCSDE algoritması 7 fonksiyonda 0,99 değerinden büyük sonuç vererek en iyi algoritma olmuştur.

Çizelge 6.17. SABCElit ve diğer algoritmaların 0.00001 uzaklıktaki SR sonuçları.

FN	SABCElit	R3PSO	R2PSO	NCSDE	CDE	ANOF	NRAND	RAND	FERPSO
F1	1,000	1,000	1,000	1,000	1,000	1,000	1,000	0,000	1,000
F2	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
F3	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
F4	1,000	0,882	0,804	1,000	0,000	1,000	1,000	0,040	1,000
F5	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
F6	0,863	0,000	0,000	0,400	0,000	0,000	0,000	0,000	0,000
F7	0,588	0,000	0,000	0,000	0,000	0,100	0,000	0,000	0,000
F8	0,020	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
F9	0,039	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
F10	0,961	0,157	0,078	1,000	1,000	1,000	1,000	1,000	0,060
F11	0,294	0,000	0,000	0,540	0,000	0,000	0,000	0,000	0,040
F12	0,255	0,000	0,000	1,000	0,000	0,000	0,100	0,000	0,000
F13	0,118	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
F14	0,020	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
F15	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
F16	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
F17	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
F18	0,020	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
F19	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
F20	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000

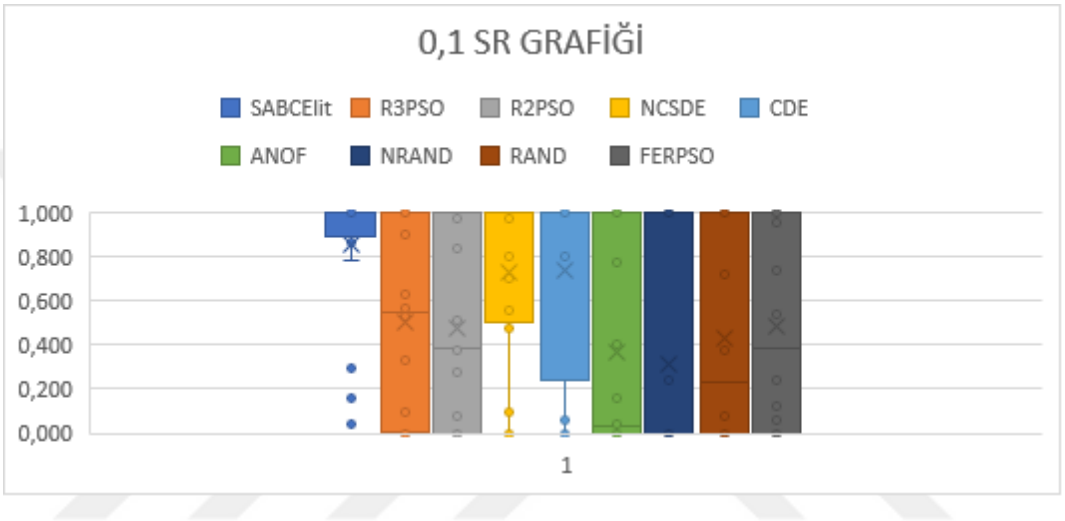
Algoritmamız, 5 fonksiyonda 0,99 değerinden daha büyük sonuç vererek, en iyi dördüncü algoritma olmuştur. NCSDE algoritması 7 fonksiyonda 0,99 değerinden büyük sonuç vererek en iyi algoritma olmuştur.

SR sonuçlarında algoritmamız toplam 40 noktada 0,99 değerinden daha büyük sonuca ulaşarak en iyi algoritma olmuştur. NCDSE algoritması, PR sonuçlarında toplam 39 noktada 0,99 değerinden daha büyük sonuca ulaşarak en iyi ikinci algoritma olmuştur.

6.4.4. SABCElit ve diğer algoritmaların SR grafikleri

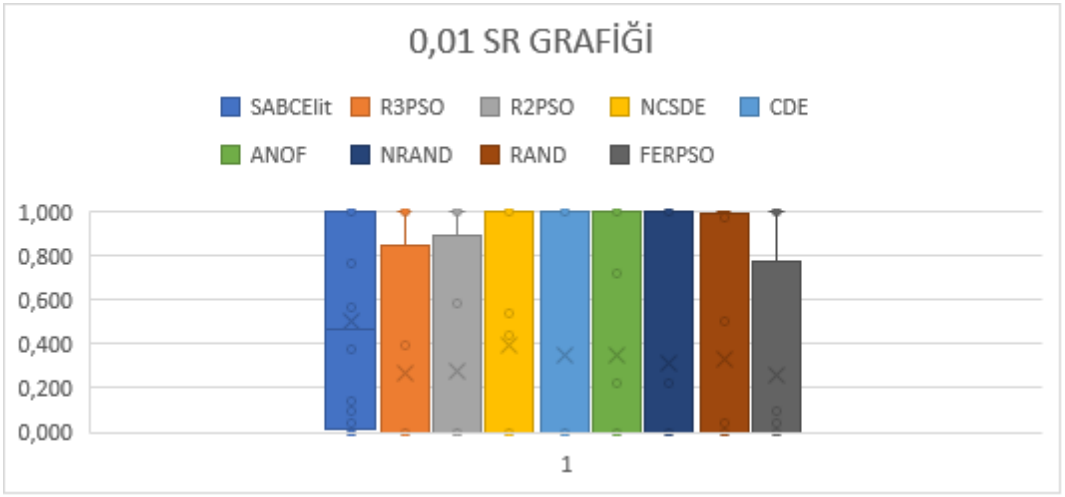
Kutu grafiği

Bir değişkenin sıklık dağılımını göstermek için kutu grafiği kullanılır. Kutu grafikleri değerlerin hangi noktada kümelendiği konusunda bize bilgi vermektedir. Aşağıda algoritmamızın ve diğer algoritmaların kutu grafikleri gösterilmiştir.



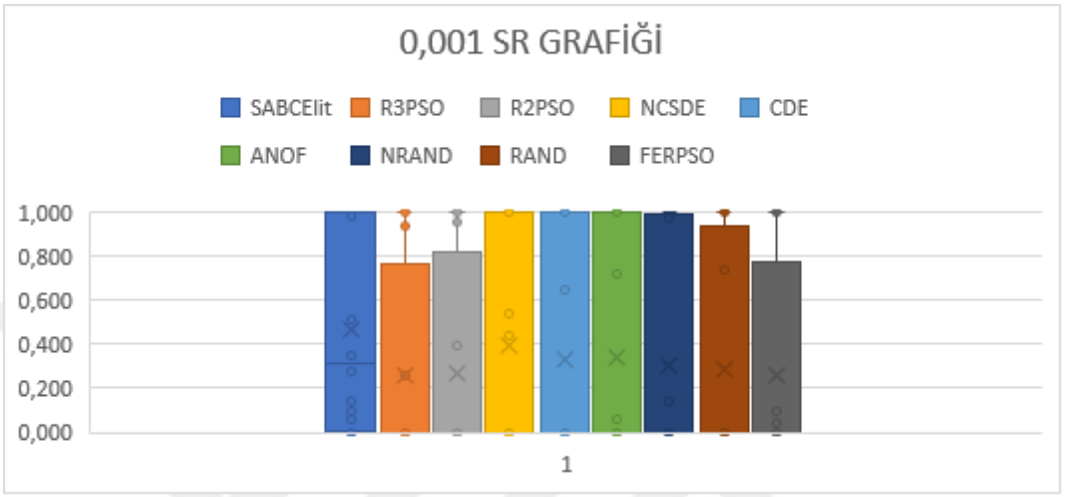
Şekil 6.30. SABCElit ve diğer algoritmaların 0.1 uzaklıktaki SR kutu grafiği.

Algoritmamız 0.1 uzaklıkta daha çok 0.9 ile 1 arasındadır. Algoritmamıza en yakın sonucu 0.75 ile 1 arasında olan NCDSE algoritması vermiştir.



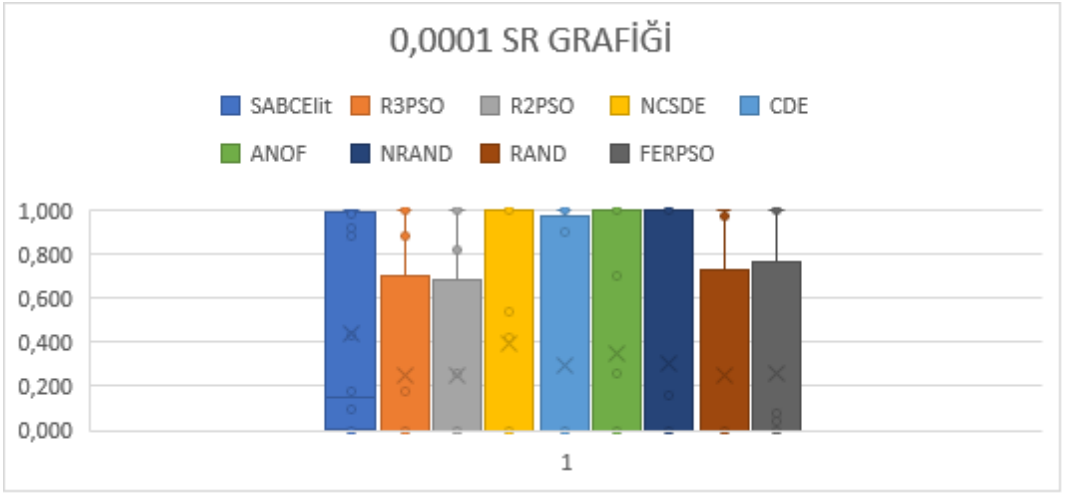
Şekil 6.31. SABCElit ve diğer algoritmaların 0.01 uzaklıktaki SR kutu grafiği.

Algoritmamız 0.01 uzaklıkta daha çok 0 ile 1 arasındadır. Sonuçların dağılıma göre YNABC, NCSDE, CDE, ANOF, CEC13 algoritmaları diğer algoritmalara göre daha iyi sonuç vermiştir.



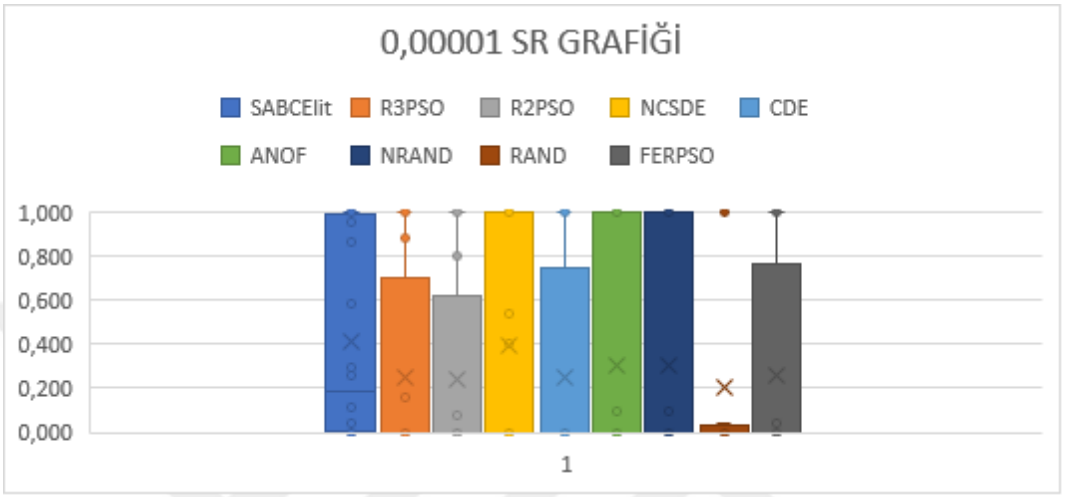
Şekil 6.32. SABCElit ve diğer algoritmaların 0.001 uzaklıktaki SR kutu grafiği.

Algoritmamız 0.001 uzaklıkta daha çok 0 ile 1 arasındadır. Sonuçların dağılıma göre YNABC, NCSDE, CDE, ANOF, CEC13(NRAND) algoritmaları diğer algoritmalara göre daha iyi sonuç vermiştir.



Şekil 6.33. SABCElit ve diğer algoritmaların 0.0001 uzaklıktaki SR kutu grafiği.

Algoritmamız 0.0001 uzaklıkta daha çok 0 ile 1 arasındadır. Sonuçların dağılıma göre YNABC, NCSDE, CDE, ANOF, CEC13(NRAND) algoritmaları diğer algoritmalara göre daha iyi sonuç vermiştir.

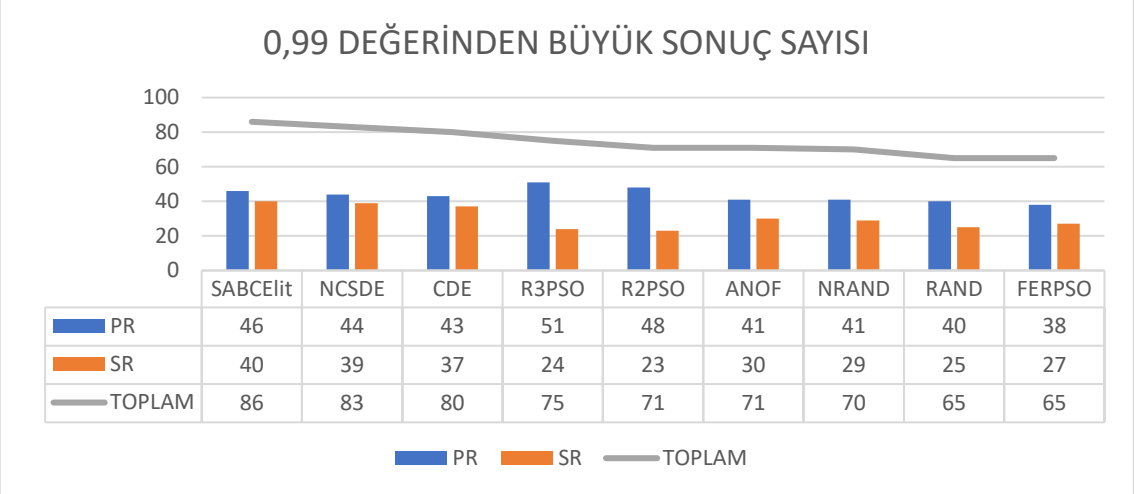


Şekil 6.34. SABCElit ve diğer algoritmaların 0.00001 uzaklıktaki SR kutu grafiği.

Algoritmamız 0.00001 uzaklıkta daha çok 0 ile 1 arasındadır. Sonuçların dağılıma göre YNABC, NCSDE, ANOF, CEC13(NRAND) algoritmaları diğer algoritmalara göre daha iyi sonuç vermiştir.

6.4.5. SABCElit ve diğer algoritmaların SR ve PR sonuçları

Algoritmalarının 0,99 değerinden büyük PR ve SR sonuç sayısını gösteren şekil aşağıdaki gibidir.



	SABCElit	NCSDE	CDE	R3PSO	R2PSO	ANOF	NRAND	RAND	FERPSO
PR	46	44	43	51	48	41	41	40	38
SR	40	39	37	24	23	30	29	25	27
TOPLAM	86	83	80	75	71	71	70	65	65

Şekil 6.35. Algoritmaların 0,99 değerinden büyük PR ve SR sonuç sayısı.

Algoritmamız, toplam 86 noktada 0,99 değerinden daha büyük sonuç vererek en iyi algoritma olmuştur. NCDSE algoritması, toplam 83 noktada 0,99 değerinden daha büyük sonuç vererek en iyi ikinci algoritma olmuştur.

7. SONUÇ

Bu çalışmada CEC 2013 multimodal problem setinin çözümü için ABC tabanlı “Elit Yiyecek Kaynaklı Adaptif Yapay Arı Kolonisi Algoritması” (SABCElit) algoritması geliştirilmiştir. Algoritma geliştirilirken, ABC algoritmasının klasik ABC yöntemlerinden farklı olarak birden çok çözüme odaklanabilecek şekilde tasarlanmıştır. Bunun için, adaptif arama denklemlerinin kullanımı ve elit çözümlerin saklanma ve yeni popülasyon üyelerinin üretilmesinde kullanımı stratejileri geliştirilmiştir.

Algoritmaya ait deneysel sonuçlar CEC 2013 multimodal problem seti üzerinden elde edilmiştir. Deneylerde CEC 2013 teknik raporunda belirtilen çalıştırma koşullarına dikkat edilmiş ve algoritmaya ait parametre değerleri otomatik parametre ayar yöntemleri ile belirlenmiştir. SABCElit algoritması aynı problem setinde daha önceden çözümü bulan algoritmalarla karşılaştırılmıştır. Algoritmamızın sonuç ve grafikleriyle diğer algoritmalarla kıyaslanması deney sonuçları bölümünde incelenmiştir. Deney sonuçları, geliştirilen algoritmanın CEC 2013 multimodal problem seti çözümü için karşılaştırılan algoritmalardan daha iyi sonucu verdiği görülmüştür.

KAYNAKLAR DİZİNİ

Antoniou, A. ve Lu, W. S. (2007), *Practical optimization: Algorithms and engineering applications*, s.1-3.

Adenso-Díaz, B. ve Laguna, M. (2006), "Fine-tuning of algorithms using fractional experimental designs ve local search," *Operations research,* 54(1), s.99–114.

Akay, B. (2009a), "Nümerik Optimizasyon Problemlerinde Yapay Arı Kolonisi (Artificial Bee Colony) Algoritmasının Performans Analizi," Erciyes Üniversitesi Fen Bilimleri Enstitüsü, s.9.

Akay, B. (2009b), "Nümerik Optimizasyon Problemlerinde Yapay Arı Kolonisi (Artificial Bee Colony) Algoritmasının Performans Analizi," Erciyes Üniversitesi Fen Bilimleri Enstitüsü, s.61.

Akay, B. ve Karaboğa, D. (2012), "A modified artificial bee colony algorithm for real-parameter optimization," *Information Sciences,* 192, s.120–142.

Ali, M. Z. ve Awad, N. H. (2014), "A novel class of niche hybrid Cultural Algorithms for continuous engineering optimization," *Information Sciences,* 267, s.158–190.

Ansótegui, C., Sellmann, M., Tierney, K. (2009), "A gender-based genetic algorithm for the automatic configuration of algorithms," In International Conference on Principles and Practice of Constraint Programming, s.142–157.

Audet, C. ve Orban, D. (2006), "Finding optimal algorithmic parameters using derivative-free optimization," *SIAM Journal on Optimization,* 17(3), s.642–664.

Aydın, D., Yavuz, G., Stützle, T. (2017), "ABC-X: a generalized, automatically configurable artificial bee colony framework, " *Swarm Intelligence,* 11(1), s.1–38.

Aydin, D., Yavuz, G., Ozyön, S., Yaşar, C., Stützle, T. (2017), "Artificial bee colony framework to non-convex economic dispatch problem with valve point effects: A case study," In Proceedings of the Genetic and Evolutionary Computation Conference Companion, s.1311–1318.

Balaprakash, P., Birattari, M., Stützle, T. (2007), "Improvement Strategies for the F-Race Algorithm: Sampling Design ve Iterative Refinement," In International workshop on hybrid metaheuristics, s.108–122.

Bartz-Beielstein, T. (2007), *Experimental Research in Evolutionary Computation*, s. 311–336.

Battiti, R., Brunato, M., Mascia, F. (2008), *Reactive search and intelligent optimization*.

Beekman, M., Gilchrist, A. L., Duncan, M., Sumpter, D. J. T. (2007), "What makes a honeybee scout?," *Behavioral Ecology and Sociobiology*, 61(7), s. 985–995.

Belal, M., Gabe, J., El-Sayed, H., Almojel, A. (2005), *Handbook of Bioinspired Algorithms ve Applications*, s.55.

KAYNAKLAR DİZİNİ (Devam)

Bienvenüe, A., Joannides, M., Bérard, J., Fontenas, É., François, O. (2002), "Niching in Monte Carlo Filtering Algorithms," In International Conference on Artificial Evolution (Evolution Artificielle), s.19–30.

Birattari, M. (2009), *Tuning metaheuristics: a machine learning perspective*, s.1-50.

Birattari, M., Stützle, T., Paquete, L., Varrentrapp, K. (2002), "A Racing Algorithm for Configuring Metaheuristics," In Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation, s.11–18.

Birattari, M., Yuan, Z., Balaprakash, P., Stützle, T. (2010), "F-Race ve Iterated F-Race: An Overview," In Experimental Methods for the Analysis of Optimization Algorithms, s.311–336.

Blackwell, T., Branke, J. (2006), "Multiswarms, exclusion, ve anti-convergence in dynamic environments," *IEEE transactions on evolutionary computation*, 10(4), s.459–472.

Blum, C. ve Merkle, D. (2008), *Swarm Intelligence: Introduction and Application*, s.43-44.

Bonabeau, E. ve Dorigo, M. (1999), *Swarm Intelligence: From Natural to Artificial Systems*, s.7.

Brits, R., Engelbrecht, A. P., Bergh, F. (2002), "Solving Systems of Unconstrained Equations using Particle Swarm Optimization," In IEEE International Conference on Systems, Man and Cybernetics, 3(6).

Brits, R., Engelbrecht, A. P., Bergh, F. (2002), "A niching particle swarm optimizer," In Proceedings of the 4th Asia-Pacific conference on simulated evolution and learning, 2, s.692–696.

Castro, L. N. ve Timmis, J. (2002), "An artificial immune network for multimodal function optimization," In Proceedings of the 2002 Congress on Evolutionary Computation, 1, s.699–704.

Cho, D. H., Jung, H. K., Lee, C. G. (2001), "Induction motor design for electric vehicle using a niching genetic algorithm," *IEEE Transactions on Industry Applications*, 37(4), s.994–999.

Coy, S. P., Golden, B. L., Runger, G. C., Wasil, E. A., (2001), "Using experimental design to find effective parameter settings for heuristics," *Journal of Heuristics*, 7(1), s.77–97.

Darwen, P. ve Yao, X. (1995), "Dilemma for fitness sharing with a scaling function," Proceedings of 1995 IEEE International Conference on Evolutionary Computation.

Delibasis, K., Asvestas, P. A., Matsopoulos, G. K. (2010), "Multimodal genetic algorithms-based algorithm for automatic point correspondence," *Pattern Recognition*, 43(12), s.4011–4027.

Della Cioppa, A., De Stefano, C., Marcelli, A. (2007), "Where are the niches? Dynamic fitness sharing," *IEEE Transactions on Evolutionary Computation*, 11(4), s.453–465.

Engelbrecht, A. (2007), *Computational intelligence*, s.8.

KAYNAKLAR DİZİNİ (Devam)

Engelbrecht, A. P., Masiye, B. S., Pampará, G. (2005), "Niching ability of basic particle swarm optimization algorithms," In Proceedings 2005 IEEE Swarm Intelligence Symposium, s.397–400.

Epitropakis, M. G., Li, X., Burke, E. K., (2013), "A dynamic archive niching differential evolution algorithm for multimodal optimization," In 2013 IEEE Congress on Evolutionary Computation, s.79–86.

Epitropakis, M. G., Plagianakos, V. P., Vrahatis, M. N. (2012), "Multimodal optimization using niching differential evolution with index-based neighborhoods," In 2012 IEEE Congress on Evolutionary Computation, s.1–8.

Epitropakis, M. G., Plagianakos, V. P., Vrahatis, M. N. (2011), "Finding multiple global optima exploiting differential evolution's niching capability," In 2011 IEEE Symposium on Differential Evolution (SDE), s.1–8.

Epitropakis, M. G., Tasoulis, D. K. Pavlidis, N. G., Plagianakos, V. P., Vrahatis, M. N. (2011), "Enhancing differential evolution utilizing proximity-based mutation operators," *IEEE Transactions on Evolutionary Computation*, 15(1), s.99–119.

Fletcher, R. (2013), *Practical Methods of Optimization*, s.3.

Forrest, S., Javornik, B. (2013), "Using genetic algorithms to explore pattern recognition in the immune system," *Evolutionary Computation*, 1(3), s.191–211.

Goldberg, D. E. ve Richardson, J. (1987), "Genetic algorithms with sharing for multimodal function optimization," In Genetic algorithms and their applications: Proceedings of the Second International Conference on Genetic Algorithms, s.41–49.

Goldberg, D. E. ve Deb, K. (1992), "Massive multimodality, deception, and genetic algorithms," *Parallel Problem Solving from Nature 2*, s.37–48.

Grosan, C., Abraham, A., Chis, M. (2006), *Swarm intelligence in data mining*, s.1–20.

Hansen, N. ve Ostermeier, A. (2001), "Completely derandomized self-adaptation in evolution strategies.," *Evolutionary Computation*, 9(2), s.159–195.

Harik, G. (1995), "Finding multimodal solutions using restricted tournament selection," *Proceedings of the Sixth International Conference on Genetic Algorithms*, s.24–31.

Huang, T., Zhan, Z. H., Jia, X. D., Yuan, H. Q., Jiang, J. Q., Zhang, J. (2018), "Niching community based differential evolution for multimodal optimization problems," In 2017 IEEE Symposium Series on Computational Intelligence, s.1–8.

Hutter, F., Hoos, H. H., Leyton-Brown, K. (2010), "Automated configuration of mixed integer programming solvers," In International Conference on Integration of Artificial Intelligence (AI) and Operations Research (OR) Techniques in Constraint Programming, s.186–202.

KAYNAKLAR DİZİNİ (Devam)

Hutter, F., Hoos, H. H., Ca, K.U., Be, S. A. (2009), "ParamILS: An Automatic Algorithm Configuration Framework," *Journal of Artificial Intelligence Research,* s.267–306.

Je, H. (1997), "The nature of niching: Genetic algorithms and the evolution of optimal, cooperative populations," *Cooperative Populations, University of Illinois.*

Jamil, M. ve Yang, X. S. (2013), "A literature survey of benchmark functions for global optimisation problems," International Journal of Mathematical Modelling and Numerical Optimisation, s.150–194.

Juana, L. ve Fern´andez, J. (2009) "Solving the Multiple Competitive Facilities Location ve Design Problem on the Plane," *Evolutionary Computation*, 17(1), s.21–53.

Karaboğa, D. ve Akay, B. (2009), "A survey: Algorithms simulating bee swarm intelligence," *Artificial Intelligence Review*, 31(1-4), s.61–85.

Karaboğa, D. ve Akay, B. (2007), "Artificial Bee Colony (ABC) Algorithm on Training Artificial Neural Networks," In 2007 IEEE 15th Signal Processing and Communications Applications.

Karaboğa, D. (2005), "An Idea Based On Honey Bee Swarm For Numerıcal Optımızatıon," *Journal of global optimization*, 39(3), 459-471.

Karaboğa, D. (2011), *Yapay Zeka Optimizasyon Algoritmaları*, s.109-205.

Kartal, B. ve Cura, T. (2015), "Yapay ari kolonisi algoritmasi ile finansal portföy optimizasyonu," Yayımlanmamış Doktora Tezi, İstanbul: İstanbul Üniversitesi Sosyal Bilimler Enstitüsü.

Kennedy, J. (2006), "Swarm Intelligence," *In Handbook of Nature-Inspired and Innovative Computing*, s.187–220.

Koper, K. D., Wysession, M. E., Wiens, D. A. (1999), "Multimodal function optimization with a niching genetic algorithm: A seismological example," *Bulletin of the Seismological Society of America*, 89(4), s. 978–988.

Kronfeld, M., Dräger, A., Aschoff, M. ve Zell, A. (2009), "On the benefits of multimodal optimization for metabolic network modeling," In German conference on bioinformatics 2009, s.191–200.

Kruisselbrink, J. W., Aleman, A., Emmerich, M., IJzerman, A. P., Bender, A., Baeck, T., Van Der Horst, E. (2009), "Enhancing search space diversity in multi-objective evolutionary drug molecule design using niching," In Proceedings of the 11th Annual conference on Genetic and evolutionary computation, s.217–224.

Li, X. (2004), "Adaptively choosing neighbourhood bests using species in a particle swarm optimizer for multimodal function optimization," In Genetic and Evolutionary Computation Conference, s.105–116.

KAYNAKLAR DİZİNİ (Devam)

Li, X. (2007), "A multimodal particle swarm optimizer based on fitness Euclidean-distance ratio," In Proceedings of the 9th annual conference on Genetic and evolutionary computation, s.78–85.

Li, X. (2011), "Developing Niching Algorithms in Particle Swarm Optimization," *In Handbook of Swarm Intelligence*, s.67–88.

Li, X. (2010), "Niching without niching parameters: Particle swarm optimization using a ring topology," *IEEE Transactions on Evolutionary Computation*, 14(1), s.150–169.

Li, X., Engelbrecht, A., Epitropakis, M. G. (2013), "Benchmark Functions for CEC'2013 Special Session ve Competition on Niching Methods for Multimodal Function Optimization," RMIT University, Evolutionary Computation and Machine Learning Group, s.1–7.

López-Ibáñez, M. ve Stützle, T. (2014), "Automatically improving the anytime behaviour of optimisation algorithms," *Automatically improving the anytime behaviour of optimisation algorithms*, 235(3), s.569–582.

López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L. P., Birattari, M., Stützle, T. (2016), "The irace package: Iterated racing for automatic algorithm configuration," *Operations Research Perspectives*, 3, s.43-58.

Lozano, M. Molina, D., Herrera, F. (2011), "Editorial scalability of evolutionary algorithms and other metaheuristics for large-scale continuous optimization problems," *Soft Computing*, 15(11), s.2085–2087.

Luo, J. ve Gu, F. (2016), "An Adaptive Niching-Based Evolutionary Algorithm for Optimizing Multi-Modal Function," International Journal of Pattern Recognition and Artificial Intelligence, 30(03), s.1–19.

Maron, O. ve Moore, A. W. (1997), "The Racing Algorithm: Model Selection for Lazy Learners," *Artificial Intelligence Review,* 11(1-5), s.193–225.

Mendes, R., Kennedy, J., Neves, J. (2004), "The fully informed particle swarm: Simpler, maybe better," *IEEE transactions on evolutionary computation*, 8(3), s.204–210.

Miller, L. ve Shaw, M. J. (1996), "Genetic algorithms with dynamic niche sharing for multimodal function optimization," In Proceedings of IEEE international conference on evolutionary computation, s.786–791.

Nannen, V. ve Eiben, A.E. (2006), "A Method for Parameter Calibration ve Relevance," In Proceedings of the 8th annual conference on Genetic and evolutionary computation, s.183–190.

Nocedal, J. ve Wright, J. S. (1999), *Numerical Optimization*, s.1.

Parrott, D. ve Li, X. (2006), "Locating ve tracking multiple dynamic optima by a particle swarm model using speciation," *IEEE Transactions on Evolutionary Computation*, 10(4), s.440–458.

KAYNAKLAR DİZİNİ (Devam)

Pérez, E., Herrera, F., Hernández, C. (2003), "Finding multiple solutions in job shop scheduling by niching genetic algorithms," *Journal of Intelligent manufacturing*, 14(3-4), s.323–339.

Pérez, E., Posada, M., Herrera, F. (2012), "Analysis of new niching genetic algorithms for finding multiple solutions in the job shop scheduling," *Journal of Intelligent manufacturing*, 23(3), s.341–356.

Petrowski, A. (1996), "Clearing procedure as a niching method for genetic algorithms," In Proceedings of IEEE international conference on evolutionary computation, s.798–803.

Powell, M. (2009), "The BOBYQA algorithm for bound constrained optimization without derivatives," *Cambridge NA Report NA2009/06*, s.39.

Qing, L. Gang, W., Zaiyue, Y., Qiuping, W. (2008), "Crowding clustering genetic algorithm for multimodal function optimization," *Applied Soft Computing*, 8(1), s.88–95.

Randy, L. H. ve Sue, E. H. (2004), *Practical genetic algorithms*, s. 2.

Ridge, E. ve Kudenko, D. (2007), "Tuning the performance of the MMAS heuristic," In International Workshop on Engineering Stochastic Local Search Algorithms, s.46–60.

Riff, M. C. ve Montero, E. (2013), "A new algorithm for reducing metaheuristic design effort," In 2013 IEEE Congress on Evolutionary Computation, s.3283–3290.

Rönkkönen, J. ve Lampinen, J. (2007), "On determining multiple global optima by differential evolution," In Proc. Eurogen Evol. Deterministic Methods Design Optimization Control, s.146–151.

Ruiz, R. ve Maroto, C. (2005), "A comprehensive review ve evaluation of permutation flowshop heuristics," *European Journal of Operational Research*, 165(2), s.479–494.

Schoeman, I. L. ve Engelbrecht, A. P. (2004), "Using vector operations to identify niches for particle swarm optimization," In IEEE Conference on Cybernetics and Intelligent Systems, 1, s.361–366.

Seeley, T. (1995), *The Wisdom of the Hive*, s.317.

Song, W., Wang, Y., Li, H. X., Cai, Z. (2015), "Locating multiple optimal solutions of nonlinear equation systems based on multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, 19(3), s.414–431.

Tereshko, V. (2000), "Reaction-diffusion model of a honeybee colony's foraging behaviour," In International Conference on Parallel Problem Solving from Nature, s. 807–816.

Tereshko, V. ve Loengarov, A. (2005), "Collective Decision-Making in Honey Bee Foraging Dynamics," *Computing and Information Systems*, 9(3), s.1–7.

KAYNAKLAR DİZİNİ (Devam)

Thomsen, R. (2004), "Multimodal optimization using crowding-based differential evolution," In Proceedings of the 2004 Congress on Evolutionary Computation, 2, s.1382–1389.

Wang, H., Moon, I., Yang, S., Wang, D. (2012), "A memetic particle swarm optimization algorithm for multimodal optimization problems," *Information Sciences*, 197, s.38–52.

Wedyan, A., Whalley, J., Narayanan, A. (2017), "Hydrological Cycle Algorithm for Continuous Optimization Problems," *Journal of Optimization*, s.1–25.

Weise, T. (2009), "Global Optimization Algorithm: Theory and Application," *Self-Published*, s.22.

Winston, W. L. (2004), *Operations Research: Applications and Algorithms*, s.2.

Wong, K. C., Leung, K. S., Wong, M. H. (2010), "Protein structure prediction on a lattice model via multimodal optimization techniques," In Proceedings of the 12th annual conference on Genetic and evolutionary computation, s.155–162.

Wong, K. C., Li, Y., Peng, C., Zhang, Z. (2015), "SignalSpider: Probabilistic pattern discovery on multiple normalized ChIP-Seq signal profiles," *Bioinformatics*, 31(1), s. 17–24.

Yang, Q., Chen, W. N., Li, Y, Chen, C. L. P., Xu, X. M., Zhang, J. (2016), "Multimodal estimation of distribution algorithms," *IEEE transactions on cybernetics*, 47(3), s.1–15.

Zhang, Y. H., Gong, Y. J., Zhang, H. X., Gu, T. L., Zhang, J. (2017), "Toward fast Niching evolutionary algorithms: a locality sensitive hashing-based approach," *IEEE Transactions on Evolutionary Computation*, 21(3), s.347-362.

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, adı : ÖZCAN, Yunus

Doğum tarihi ve yeri : 04.03.1993 - Çameli/DENİZLİ

e-mail : yyunusozcan@gmail.com

Eğitim

Derece	Eğitim Birimi	Mezuniyet Tarihi
Lise	: Hasan Tekin Ada Lisesi	2005
Lisans	: Dumlupınar Üniversitesi Bilgisayar Mühendisliği	2016
Lisans	: Dumlupınar Üniversitesi Elektrik Elektronik Mühendisliği	2016

İş Deneyimi

Yıl	Yer	Görev
2016-	Nursan Kablo Donanımları	Ar-Ge Mühendisi

Yabancı Dil

İngilizce