DÜELLO OPTİMİZASYON ALGORİTMASININ İKİLİ OPTİMİZASYON PROBLEMLERİNE UYGULANMASI

Hacer DÖNMEZ

2020
YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ

Tez Danışmanı
Dr. Öğr. Üyesi Emrullah SONUÇ

DÜELLO OPTİMİZASYON ALGORİTMASININ İKİLİ OPTİMİZASYON PROBLEMLERİNE UYGULANMASI

Hacer DÖNMEZ

T.C.
Karabük Üniversitesi
Lisansüstü Eğitim Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalında
Yüksek Lisans Tezi
Olarak Hazırlanmıştır

Tez Danışmanı
Dr. Öğr. Üyesi Emrullah SONUÇ

KARABÜK
Kasım 2020

Hacer DÖNMEZ tarafından hazırlanan "DÜELLO OPTİMİZASYON ALGORTİMASININ İKİLİ PROBLEMLERİNE UYGULANMASI" başlıklı bu tezin Yüksek Lisans Tezi olarak uygun olduğunu onaylarım.

Dr. Öğr. Üyesi Emrullah SONUÇ .....
Tez Danışmanı, Bilgisayar Mühendisliği Anabilim Dalı

Bu çalışma, jürimiz tarafından oy birliği ile Bilgisayar Mühendisliği Anabilim Dalında Yüksek Lisans tezi olarak kabul edilmiştir. 13/11/2020

Ünvanı, Adı SOYADI (Kurumu)	İmzası
Başkan : Dr. Öğr. Üyesi Emrullah SONUÇ (KBÜ)	.....
Üye : Dr. Öğr. Üyesi Rafet DURGUT (KBÜ)	.....
Üye : Dr. Öğr. Üyesi Kemal AKYOL (KÜ)	.....

KBÜ Lisansüstü Eğitim Enstitüsü Yönetim Kurulu, bu tez ile, Yüksek Lisans derecesini onamıştır.

Prof. Dr. Hasan SOLMAZ .....
Lisansüstü Eğitim Enstitüsü Müdürü

"Bu tezdeki tüm bilgilerin akademik kurallara ve etik ilkelere uygun olarak elde edildiğini ve sunulduğunu; ayrıca bu kuralların ve ilkelerin gerektirdiği şekilde, bu çalışmadan kaynaklanmayan bütün atıfları yaptığımı beyan ederim."

Hacer DÖNMEZ

ÖZET

Yüksek Lisans Tezi

DÜELLO OPTİMİZASYON ALGORİTMASININ İKİLİ OPTİMİZASYON PROBLEMLERİNE UYGULANMASI

Hacer DÖNMEZ

Karabük Üniversitesi
Lisansüstü Eğitim Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Tez Danışmanı:
Dr. Öğr. Üyesi Emrullah SONUÇ
Kasım 2020, 51 sayfa

Optimizasyon problemleri bilgisayar mühendisliği ve matematik başta olmak üzere birçok disiplin tarafından çalışılmaktadır. Bu problemlerin çözümünde sağlanan gelişmeler mühendislik, sağlık, işletme gibi birçok alanda zaman ve maliyet kazancı sağlamaktadır. Optimizasyon problemleri kullandıkları karar değişkenlerinin yapısına ve arama uzayına göre basit olarak sürekli veya ayrık olarak iki sınıfa ayrılabilir. Düello optimizasyon algoritması, sürekli optimizasyon problemleri için geliştirilmiş olup, ayrık problemlere uygulanabilmesi için bazı düzenlemeler gerekmektedir. Bu tez çalışmasında, düello optimizasyon algoritmasının ayrık problemlerin özel bir şekli olan ikili yapıdaki optimizasyon problemlerine uygulanabilmesi için bir yöntem önerilmiştir. Önerilen yöntemin karar değişkenleri ikili formda olup, komşuluk operatörü olarak mantıksal işlemler kullanılmaktadır. Geliştirilen yöntem one-max ve kapasitesiz tesis yerleşim problemleri üzerinde test edilmiştir. Yöntemin başarısı literatürdeki diğer yöntemlerle karşılaştırmalı olarak verilmiştir. Yapılan çalışmaların

çıkarımları ele alındığında, önerilen yöntemlerin, standart sapma ve yakınsama özellikleri ile, ikili optimizasyon problemleri çözümünde alternatif, rekabetçi ve güçlü oldukları görülmektedir. Böylece bu tez ikili optimizasyon alanında literatüre katkı sağlamaktadır.

Anahtar Sözcükler : Düello Optimizasyon Algoritması, İkili Optimizasyon, Metasezgisel Algoritmalar.

Bilim Kodu : 92402

ABSTRACT

M. Sc. Thesis

APPLICATION OF DUELIST OPTIMIZATION ALGORITHM TO BINARY OPTIMIZATION PROBLEMS

HACER DÖNMEZ

Karabük University
Institute of Graduate Programs
Department of Computer Engineering

Thesis Advisor:
Assist. Prof. Dr. Emrullah SONUÇ
November 2020, 51 pages

Optimization problems have been applied in many dissiplinary fields, notably computer engineering and mathematics. One can achieve a satisfactory solution reducing total cost in a reasonable time through many different kinds of optimization problems and those problems can be applied in fields such as engineering, business and healthcare. Optimization problems are basically subdivied into two parts in terms of the typr of decision variables being used and the space; continuous and discrete. Duelist optimization were developed for continuous optimization problems and some measurements should be done before applying to discrete optimization problems. In this thesis, a method has been proposed to apply duelist optimization into 0-1 binary optimization problems; a special kind of discrete optimization. Decision variables in this method are in binary form and logic operators have been used as neighborhood operators. This proposed method were tested for one-max and resulted in uncapasited facilty location problems success. The success of the method has been compared with other methods in the literature. Considering the implications of the studies, it is seen

that the proposed methods are alternative, competitive and robust in solving binary optimization problems with their standard deviation and convergence properties. Thus, this thesis contributes to the literature in the field of binary optimization.

Keywords: Duelist Optimization Algorithm, Binary Optimization, Metaheuristics Algorithms.
Science Code : 92402

TEŞEKKÜR

Bu tez çalışmasının planlanmasında, araştırılmasında, yürütülmesinde ve oluşumunda ilgi ve desteğini esirgemeyen, engin bilgi ve tecrübelerinden yararlandığım, yönlendirme ve bilgilendirmeleriyle çalışmamı bilimsel temeller ışığında şekillendiren sayın hocam Dr. Öğr. Üyesi Emrullah SONUÇ'a sonsuz teşekkürlerimi sunarım.

Sevgili aileme maddi ve manevi hiçbir yardımı esirgemeden yanımda oldukları için, ailemizin en küçük üyesi olan Yusuf Dönmez'e varlığıyla hayatıma neşe kattığı için tüm kalbimle teşekkür ederim.

İÇİNDEKİLER

Sayfa

ŞEKİLLER DİZİNİ

Sayfa

ÇİZELGELER DİZİNİ

Sayfa

KISALTMALAR DİZİNİ

ABA	:	Ateş Böceği Algoritması
BT	:	Benzetimli Tavlama Algoritması
BYP	:	Birim Yüklenme Problemi
DG	:	Diferansiyel Gelişim
DEBT	:	Dağıtılmış Evrimsel Benzetimli Tavlama
DOA	:	Düello Optimizasyon Algoritması
EBT	:	Evrimsel Benzetimli Tavlama
GA	:	Genetik Algoritma
İASÇA	:	İkili Ağırlıklı Süperpozisyon Çekim Algoritması
İGA	:	İkili Genetik Algoritma
İGKO	:	İkili Gri Kurt Algoritma
İPSO	:	İkili Parçacık Sürü Optimizasyonu
İYA	:	İkili Yarasa Algoritması
İYAK	:	İkili Yapay Arı Kolonisi Algoritması
KKO	:	Karınca Koloni Algoritması
KTYP	:	Kapasitesiz Tesis Yerleşim Problemi
N	:	Boyut
PSO	:	Parçacık Sürü Optimizasyonu
SÇP	:	Sırt Çantası Problemi
TYP	:	Tesis Yerleşim Problemi
YAK	:	Yapay Arı Kolonisi Algoritması

BÖLÜM 1

GİRİŞ

1.1. OPTİMİZASYON KAVRAMI

Temel olarak belirlenen bir amaç veya amaçlar doğrultusunda, girdilerin maksimum veya minimum çıktısını belirlemeye yönelik matematiksel bir süreç olarak tanımlanan [1] optimizasyon, insanlar ve doğa tarafından günlük yaşamlarında yaygın olarak kullanılan fiziksel sistemlerin ve karar biliminin analizi için önemli bir araçtır [2]. Matematiksel fonksiyonlarla tanımlanan problemlere en iyi çözümü bulmayı amaçlayan [3] işlemler bütünü olan optimizasyon teknikleri, en iyisinin veya en kötüsünün bulunması istenen bir çalışmada, optimal değerin hesaplanabileceği bir dizi teknik, yöntem, prosedür ve algoritmadan oluşmaktadır [4]. Bu tanımlamalar paralelinde, ele alınan bir soruna en uygun çözümün elde edilmesini amaçlayan, mühendislik ve işletme bilimleri gibi pek çok farklı disiplindeki çalışmalarda, optimizasyon şimdi olduğu gibi gelecekte de çok önemli bir yere sahip olacaktır.

Optimizasyon kavramı birçok disiplinin (mühendislik, işletme ve bilim) ilgisini çekmiştir ve daha önce çözülmesi zor kabul edilen problemlerin çözülmesine yardımcı olmuştur [4] .

Optimizasyon yaklaşımları; analitik yöntemler, grafiksel yöntemler, deneysel yöntemler ve sayısal yöntemler olarak dört farklı şekilde sınıflandırılabilir.

Analitik yöntemlerle problem matematiksel olarak ifade edilebilmektedir. Problemin çözümünde matematiksel fonksiyonun türetilmesiyle ilgilenmektedir. Yüksek kaliteli doğrusal olmayan problem çözmede ve bağımsız değişkenlerin sayısının üçten fazla olduğu durumlarda analitik yöntemler kullanılmamaktadır. Ele alınan problemi grafik yöntemi ile çözerken, maksimum veya minimum noktayı bulmak için amaç

fonksiyonunun grafiğini kullanılmaya çalışılmaktadır. Grafik yöntemler sınırlı bir alanda kullanılmaktadır, çünkü fonksiyon sayısının ikiyi aştığı durumlarda kullanımı zor olmaktadır. Deneysel yöntem ile problemlerin çözümleri, deney şartları uygun hazırlanarak bu şartlar altında deneme yanılma yoluyla sorunun maksimum ve minimum değerleri araştırılmaktadır. Deneysel ortamda, maksimum ve minimum noktanın kesin olarak belirlenmesi zor olmaktadır. Çok değişkenli problemler için doğru sonuçlar vermeyebilir ve çevresel koşullardan da etkilenebilmektedir. Sayısal yöntem ile problemi çözmek ise, sayısal yöntemlerin kademeli olarak uygulanmasıyla bir çözüm bulma girişimidir. Problemin çözülmesinin zor olduğu ve bağımsız değişkenlerin sayısının fazla olduğu problemler için kullanılmaktadır. Sayısal optimizasyon matematiksel programlama olarak da bilinmektedir. Son elli yılda geliştirilen matematiksel programlama modelleri; doğrusal, doğrusal olmayan, tam sayı, ikinci dereceden ve dinamik programlamadır [4].

Optimizasyon modelinde maksimuma veya minimuma getirilmesi gereken fonksiyona amaç fonksiyonu denir. Karar değişkenleri, kontrol altında tutulması gereken ve sistem performansını etkileyen değişkenlerdir. Kısıtlamalar, karar değerlerinin alacabileceği aralığını belirleyen değişkenlerdir [5].

Zamanla değişen optimizasyon problemleri dinamik optimizasyon, zamanla sabit kalan optimizasyon problemleri ise, statik optimizasyon olarak adlandırılır. Statik optimizasyon problemleri söz konusu olduğunda, matematiksel ifadeler içeren modeller cebirsel denklemlerden oluşmaktadır. Optimizasyon fonksiyonundaki parametre değişkenlerin alabileceği değerlere göre optimizasyon problemleri sürekli(continuous), ayrık(discrete) ve karışık(mixed) olarak sınıflandırılabilir. Girdi ve çıktı değerleri gerçek sayı olmayan problemlere ayrık problemler denilmektedir. Girdi ve çıktı değerleri gerçek değerler alan problemlere sürekli optimizasyon problemleri denilmektedir [6].

Sürekli optimizasyon problemlerinde, fonksiyonları minimize veya maksimize etmek için çeşitli sürekli değişken değerleri elde edilmelidir. Fonksiyonlar yapılarına göre tek ve çok modlu olarak adlandırılmaktadır. Tek modlu yapılar bir noktada yerel bir

optimal noktaya sahiptir. Yerel ve küresel optimum noktaların bulunmasında kullanılan optimizasyon yöntemleri aşağıdaki örnekler verilebilir [7].

1. Tek boyutta minimizasyon yöntemleri (Golden Section)
2. Türeve dayalı olmadan çok boyutlu arama yöntemleri (Conjugate Gradient)
3. Birinci türeve dayalı yöntemler (Quasi Newton)
4. İkinci türeve dayalı työntemler (Newton, Levenberg, Mardquardt)
5. Metasezgisel algoritmalar (Karınca Kolonisi Algoritması (KKO), Genetik Algoritma (GA), Yapay Arı Kolonisi (YAK) vb.)

Optimum tasarım problemlerinin modellenmesi oldukça önemlidir ve aşağıdaki beş adımdan oluşmaktadır [8].

1. Veri ve Bilgi Toplama
2. Problemin Tanımı
3. Tasarım Parametrelerinin Tanımı
4. Optimizasyon Kriteri
5. Sınırlamaların Formülasyonu

Veri ve Bilgi Toplama: Modelleme süreci, sorunlar ve gereksinimler hakkında bilgi toplamakla başlamaktadır. Problem özellikleri, kaynak sınırları, problem maliyetleri ve performans gereksinimleri gibi başka bilgiler gerekli olabilmektedir. Bu nedenle, sorunla ilgili tüm bilgiler toplanmalıdır.

Problem tanımı: Süreç, problemin tanımlayıcı ifadesinin oluşturulmasıyla devam etmektedir. Hazırlanan ifade sorunun tüm hedeflerini ve yerine getirmesi gereken şartları tanımlamaktadır. Bazı problemler için sorunun tanımı belirsiz olabilir. Bu tür problemleri formüle etmek ve çözmek için, sorunun modeli hakkında varsayımlar yapılır.

Tasarım parametrelerinin tanımı: Formülasyon sürecinde, bu adım, sistemi tanımlayan, tasarım değişkenleri olarak bilinen değişkenler kümesini tanımlar. Genellikle bu değişkenlere optimizasyon değişkenleri de denir. Değişkenler bağımsız

olmalıdır ve istenen herhangi bir değer atanabilir. Atanan her değer farklı tasarımlara sebep olmaktadır ve bağımsız parametrelerin sayısı problemin serbestlik derecesini göstermektedir.

Problem için uygun tasarım parametreleri seçilmezse, ifade ya yanlıştır ya da oluşturulması imkansızdır. Bu nedenle, ilk aşamada tasarım parametrelerini belirlemek için tüm seçenekler araştırılmalıdır. Bazı durumlarda, belirtilen serbestlik derecelerinden daha fazla tasarım parametresi atamak istenebilir. Bu formülasyondaki esnekliği arttırır. Ek değişkenler için sabit değerler belirleyerek, bu değişken formülasyondan çıkarılabilir. Tasarım parametrelerini belirlemenin zor olduğu sorunlar da olabilmektedir. Böyle bir durumda, tüm değişkenlerin bir listesi yapılır, her değişken ayrı ayrı ele alınır ve bir tasarım parametresi olarak uygun olup olmadığına karar verilir. Geçerli bir tasarım parametresi olduğuna karar verilirse, bir başlangıç çözümü seçmek için bu değişkene sayısal bir değer atanabilir.

Özet olarak; bir problem için tasarım parametrelerini belirlerken aşağıdakiler göz önüne alınmalıdır:

1. Optimizasyon problemini uygun şekilde formülize etmek için gereken tasarım parametreleri minimum sayıda olmalıdır. Yani serbestlik derecesi küçük olmalıdır.
2. Tasarım parametreleri mümkün olduğu kadar birbirlerinden bağımsız olmalıdır.
3. Değilse bile aralarında bazı eşitlik sınırlamaları olmalıdır.
4. Çoğu bağımsız parametrede olduğu gibi problemin formülizasyon aşamasında tasarım parametrelerine mümkün olduğunca değer atanabilir olmalıdır.
5. Sistemin başlangıç çözümü belirlenirken tanımlanan her tasarım parametresine nümerik deger verilmelidir.

Optimizasyon Kriterleri: Sistem için kabul edilebilir birçok tasarım olabilir. Ancak bazıları diğerlerinden daha iyidir. Bu durumda, tasarımların nasıl karşılaştırıldığı ve hangisinin daha iyi olduğu belirlenmelidir. Bunun için tasarımın başarısını belirleyen kriterler olmalıdır. Optimizasyon kriteri, sayısal bir değerin elde edilebileceği skaler

bir fonksiyon olmalıdır. Dolayısıyla skaler vektörü, tasarım parametresi vektörünün bir fonksiyonu olmalıdır. Optimum tasarım problemlerinde bu kritere objektif fonksiyon denir. Bu işlev gerektiğinde büyütülebilir veya küçültülebilir. En aza indirilmeye çalışılan optimizasyon kriteri literatürde genellikle maliyet fonksiyonu olarak adlandırılmaktadır. Konu tasarım problemleri olduğunda, uygun amaç fonksiyonunu seçmek önemli bir karardır. Net bir amaç fonksiyonu tanımlanmalıdır. Bazı durumlarda iki veya daha fazla nesnel fonksiyon tanımlanabilir. Bunlar genel amaçlı optimizasyon problemleri olarak tanımlanmaktadır.

Sınırlamaları Formülasyon: Modelleme sürecindeki son adım, tüm kısıtlamaları tanımlamak ve bunlar için talimatlar oluşturmaktır. Tasarımın keşfedilebileceği alanı sınırlayan herhangi bir terime kısıtlama denilmektedir. Uygun kısıtlamaların tanımlanmasıyla, mevcut kaynakları kullanabilen ve performans gereksinimlerini karşılayabilen gerçeğe en yakın sistem tasarlanmalıdır. Sınırlamalar tasarım parametrelerine bağlı olmalıdır. Çünkü tasarım parametrelerinin farklı değerleri için sınır değerleri de değişmektedir. Bu nedenle, anlamlı bir kısıtlama fonksiyonu, tasarım parametrelerinden en az birinin bir fonksiyonu olmalıdır. Modelleme süreci tamamlandıktan sonra sorunun çözümüne geçilir. Herhangi bir kısıtlama olup olmadığı, tasarım parametrelerinin yapısı, problemin yapısı, problemin içerdiği denklemlerin yapısı gibi değerlendirmelerle, soruna uygun problemi çözme yöntemi kullanılır. Bu problem çözme yöntemleri optimizasyon yöntemleri olarak tanımlanır.

1.2. OPTİMİZASYON PROBLEMLERİNİN SINIFLANDIRILMASI

Literatürde çok sayıda ve farklı türde optimizasyon problemi bulunmaktadır. Bu problemler pek çok farklı kritere göre sınıflandırılabilir. Bu sınıflandırma kriterleri aşağıda 7 ana başlıkta incelenmiştir.

1.2.1. Sınırlamaların Varlığına Göre Sınıflandırma

Optimizasyon problemleri karar değişkenlerinin kısıtlamaları olup olmamasına göre sınırlamalı ya da sınırlamasız optimizasyon problemi olarak adlandırılmaktadır.

Kısıtlı optimizasyon problemlerinde, arama uzayında olanaklı ve olanaksız alanlar bulunabilmektedir. Bu yüzden optimizasyon algoritmasının olanaksız alanları çözüme dahil edilmeyip olanaklı alanda bulunan yeni çözüm değerleri dikkate alınmalıdır [9].

1.2.2. Tasarım Parametrelerinin Yapısına Göre Sınıfandırma

Tasarım parametrelerinin yapısına bağlı olarak optimizasyon problemleri iki alt kategoriye ayrılabilir. İlk kategoride tasarım parametrelerinin zamana bağlı olarak değişmediği ya da belirsizlik içermediği problem türleri vardır. Bu problemlere statik optimizasyon problemleri denir. İkinci kategoride ise tasarım parametreleri zamana bağlı degişkenlik göstermektedir. Bu problem türünde ise amaç, zamana ya da olaylara bağlı degişiklik gösteren optimal çözümü bulmaktır. Bu tarz problemlere dinamik optimizasyon problemi denilmektedir.

1.2.3. İçerdiği Denklemlerin Yapısına Göre Sınıflandırma

Optimizasyon problemleri için başka önemli bir sınıflandırma amaç fonksiyonu ve sınırlamalar için kullanılan ifadelerin yapısıdır. Bu sınıflandırmaya göre optimizasyon problemi doğrusal (lineer), doğrusal olmayan (nonlineer) olabilir. Amaç fonksiyonu ve tüm sınırlamalar doğrusal ise Doğrusal Programlama, amaç ve sınırlama fonksiyonlarından herhangi biri doğrusal değil ise Doğrusal Olmayan Programlama adı verilmektedir. Amaç fonksiyonu ve sınırlama fonksiyonları, c_k ve x_i pozitif reel sayılar, $a_i k$ reel sayı olmak üzere;

$$f(x) = \sum_{k=1}^{K} (c_k\, x_1\, a_1 k\, x_2\, a_2 k \ldots x_n\, a_n k) \qquad (1.1)$$

Şeklinde, yani polinom ise Geometrik Programlama adını alır. Kuadratik amaç fonksiyonu ve doğrusal sınırlamalı doğrusal olmayan programlama problemi ise "Kuadratik Programlama" olarak tanımlanmaktadır.

1.2.4. Karar Değişkenlerinin Alacabileceği Değerlere Göre Sınıflandırma

Optimizasyon problemleri karar değişkenleri tipi tamsayı değeri veya reel sayı değeri alabilmektedir. Bir optimizasyon probleminin tüm ya da bazı karar değişkenlerinin sadece tamsayı (integer) değeri almakla sınırlandırılırsa, problem tamsayı programlama problemi olmaktadır. Diğer yandan, tüm karar değişkenleri reel değer almasına izin verilirse, optimizasyon problemi reel değerli programlama problemi olmaktadır. Başka bir grup ise sürekli ve ayrık optimizasyon problemidir. Sürekli optimizasyon problemleri karar değişkenlerinin alacağı değerlerin sürekli olması durumunu; ayrık optimizasyon problemleri ise, ayrık ifadelerin optimal olarak değiştirilmesi, kümelenmesi, sıralanması veya seçilmesi durumlarını ifade etmektedir. Bazı problemlerin karar değişkenleri sürekli ve ayrık değerler alabilir. Bu tür optimizasyon problemleri karışık (mixed) sınıfına dahil edilmektedir.

1.2.5. Değişkenlerin Deterministikliğine Göre Sınıflandırma

İçerdiği değişkenlerin deterministik olmasına göre, optimizasyon problemleri deterministik ve stokastik programlama problemleri olarak sınıflandırılmaktadır. Problemdeki parametrelerin tümü ya da bazıları olasılıksal (deterministik olmayan veya stokastik) olduğunda stokastik programlama problemi olarak kabul edilir. Aksi durumda ise deterministik programlama problemi olarak isimlendirilir.

1.2.6. Fonksiyonların Ayrılabilirliğine Göre Sınıflandırma

Optimizasyon problemleri amaç ve sınırlama fonksiyonlarının ayrılabilirligine dayanarak sınıflandırılabilir. Eğer $f(x)$ fonksiyonu n tek-degişkenli fonksiyonun, $f1(x1), f2(x2), \dots\ fn(xn)$ toplamı yani,

$$f(x) = \sum_{i=1}^{n} fi(xi) \tag{1.2}$$

Şeklinde ifade edilebilirse ayrılabilir (separable) oldugu söylenir. Bu formda ifade edilemiyorsa ayrılamayan (non-separable) olarak isimlendirilmektedir.

1.2.7. Amaçların Fonksiyonun Sayısına Göre Sınıflandırma

Optime edilmesi gereken amaç fonksiyonlarının sayısına göre, optimizasyon problemleri tek(single) ve çok(multi) amaçlı programlama problemleri olarak gruplandırılabilir.

1.3. OPTİMİZASYON ALGORİTMALARI

Optimizasyon yöntemlerinin kökeni Newton, Lagrange ve Cauchy'nin çalışmalarına dayanmaktadır. Newton ve Leibnitz'in hesaplamaya katkıları sayesinde diferansiyel hesaplama yöntemleri geliştirilmiştir. Fonksiyonların minimum değerleri Bernoulli, Euler, Lagrange ve Weirstrass tarafından varyasyon hesaplamaları bulunarak belirlenmiştir. Bilinmeyen yeni faktörlere sahip kısıtlı optimizasyon problemleri için optimizasyon yöntemine Lagrange adı verilmiştir. Cauchy, sınırlandırılmamış minimizasyon problemleri için en dik iniş yöntemini ilk kez kullanmıştır. Analitik optimizasyon yöntemlerinde, çok değişkenli bir fonksiyonun minimum veya maksimum değerlerini bulmak için diferansiyel hesaplamalar kullanılabilir. Bu yöntemler, tasarım parametrelerinin iki kere farklılaşabileceğini ve türetmenin sürekli olduğunu varsayar. Lagrange çarpan yöntemi, kısıtlı eşitliğe sahip optimizasyon problemleri için kullanılabilir. Problemin eşitsizlik kısıtlamaları varsa, Kuhn-Tucker koşulları kullanılabilir. Bununla birlikte, bu yöntemler çözülmesi zor olabilen doğrusal olmayan eşzamanlı denklemler oluşturabilir. Bu nedenle, çeşitli problem türlerini çözmek için analitik yöntemlere yakınsayan sayısal yöntemler ortaya çıkmıştır. Doğrusal olmayan, doğrusal, geometrik, kuadratik ve tamsayı programlama yöntemleri, yöntemin adında belirtilen belirli problem gruplarını çözmek için kullanılabilir. Bu yöntemlerin çoğu, ilk çözümden başlayıp yinelemeli olarak devam eden sayısal yöntemlerdir. Bunlardan doğrusal olmayan programlama yöntemi, herhangi bir optimizasyon problemini çözmek için kullanılabilecek en genel yöntemdir.

Geleneksel veya klasik algoritmaların çoğu deterministiktir. Örneğin, doğrusal programlamadaki simpleks yöntemi deterministiktir. Bazı deterministik optimizasyon algoritmaları gradyan bilgisini kullanır, bunlara gradyan tabanlı algoritmalar denir.

Sezgisel algoritmalar, klasik algoritmalar ile çözümü mümkün olmayan problemler için tasarlanmıştır. Genel olarak, sezgisel, "bulmak" veya "deneme yanılma yoluyla keşfetmek" anlamına gelir. Zor bir optimizasyon problemine kaliteli çözümler makul bir süre içinde bulunabilir, ancak optimum çözümlere ulaşılacağının garantisi yoktur. Bu algoritmaların çoğu zaman çalıştığı, ancak her zaman çalışmayacağı beklenmektedir. Bu, mutlaka en iyi çözümleri değil, kolayca erişilebilen iyi çözümler istenildiğinde uygun olmaktadır [10].

Sezgisel algoritmalar geliştirilip üzerinde daha fazla çalışma yapılarak, yeni tür olan metasezgisel algoritmalar ortaya çıkmıştır. Burada meta, " daha yüksek seviye " ve heuristic "bulan" anlamına gelir ve genellikle basit sezgisel yöntemlerden daha iyi performans göstermektedir. Tüm metasezgisel algoritmalar, belirli bir rasgtele seçim ödünleşme (trade-off) ve yerel arama kullanır. Literatürde sezgisel ve metasezgisel kelimelerinin kararlaştırılmış tanımları bulunmamaktadır. ‘Sezgisel’ ve 'metasezgisel' ifadeleri birbirinin yerine kullanılabilmektedir. Bununla birlikte, yaygın olan rastgele ve yerel arama ile tüm stokastik algoritmaların metasezgisel olarak adlandırıldığı görülmektedir. Rastgelelik, yerel aramadan küresel ölçekte aramaya geçmek için imkân sağlamaktadır. Bu nedenle, neredeyse tüm metasezgisel algoritmalar küresel optimizasyona uygun olmayı amaçlamaktadır [10].

Metasezgisel algoritmalar genellikle canlıların genetik ve nörobiyolojik davranışlarından ve karakteristiklerinden ilham alarak geliştirilmiştir. GA, benzetimli tavlama (BT), parçacık sürüsü optimizasyonu (PSO), KKO, diferansiyel gelişim algoritması (DG) ve YAK metasezgisel yöntemlerdir. Bu yöntemler, son yıllarda karmaşık teknik sorunları çözmek için daha sık kullanılmaktadır. GA, doğal genetik ve seçim ilkelerine dayanmaktadır. BT, ısıtılan katıların, çıkabileceği maksimum sıcaklığa kadar çıktıktan sonra belirli bir hızda soğutulmasını simüle etmektedir. Hem GA hem de BT, küresel bir minimum veya optimal değere yakın bir değer bulması muhtemel olan stokastik yöntemlerdir. Bu nedenle, ayrık optimizasyon problemlerini çözmek için uygundur. PSO, böcekler, kuşlar veya balıklar gibi canlıların koloni davranışlarına dayanmaktadır. KKA, karıncaların yuvalarından yiyecek kaynağına doğru yol alan genel davranışlarına dayanmaktadır. DE, sürekli uzayda optimizasyon problemlerini çözmek için geliştirilmiş verimli ve güçlü bir stokastik araştırma

yöntemidir. YAK, bal arılarının akıllı yiyecek arama davranışını simüle etmektedir. Optimizasyon yöntemleri genel olarak Çizelge 1. 1 ‘de verilmiştir.

Çizelge 1. 2. Optimizasyon Algoritmaları.

Analitik Metotlar	**Metasezgisel Metotlar**
Hesaplama Metotları	Genetik Algoritmalar
Varyasyon Hesaplamaları	Benzetimli Tavlama
Nonlineer Programlama	Karınca Koloni Optimizasyonu
Geometrik Programlama	Parçacık Sürü Optimizasyonu
Kuadratik Programlama	Diferansiyel Gelişim
Lineer Programlama	Yapay Arı Kolonisi Algoritması
Dinamik Programlama	
Tamsayı Programlama	
Stokastik Programlama	
Çok Amaçlı Programlama	

BÖLÜM 2

İKİLİ OPTİMİZASYON

2.1. İKİLİ OPTİMİZASYON TANIMI

İkili optimizasyon, ayrık optimizasyon araştırma alanının bir parçasıdır. İkili optimizasyonun arama alanı "0" ve "1" değerlerinden oluşur. Birçok ayrık optimizasyon problemi ikili optimizasyon problemi olarak modellenebilir ve ikili optimizasyon algoritmaları ile çözülebilir [11].

Birim Yüklenme Problemi (BYP), sırt çantası problemi (SÇP) ve özellik seçimi (ÖS) gibi bazı ayrık problemler, modifikasyonlar olmadan çözülemez. Bu nedenle, birçok araştırmacı ikili problemleri çözmek için İkili Genetik Algoritma(İGA), İkili Parçacık Sürüsü Optimizasyonu (İPSO) ,İkili Yarasa Algoritması (İYA) ve İkili Gri Kurt Optimizasyon (İGKO) gibi yeni algoritmalar önermiş veya mevcut olanlar yeniden düzenlenmiştir [12].

İkili optimizasyonlar, grafik ikiye bölme (graph bisection) [13,14],markov rastgele alanlar (Markov random fields) [15],permütasyon problemleri [16,17], grafik teorisi (graph matching)[18–20], görüntü bölütleme (image segmentation) [21,22], görüntü kaydı (image registration) [23] , sosyal ağ analizi (social network analysis) [24,25], kümeleme (clustering) [26] vb. dahil olmak üzere hem bilgisayar görüyü hem de makine öğrenimi ile ilgilenen birçok uygulamada kendini göstermekdir [27].

2.2. İKİLİ OPTİMİZASYON YÖNTEMLERİ

Optimizasyon yöntemleri başlangıçta arama uzayının sürekli olduğu varsayımı altında geliştirildiğinden, orijinal yöntemler ikili problemler üzerinde etkili değildir. Bu nedenle, literatürde aşağıdaki gibi birkaç modifikasyon tekniği önerilmiştir [28].

2.2.1. Transfer Fonksiyonu

Kennedy ve Eberhart [29], sürekli değişkenleri ikili değerlerle adapte etmek için bir sigmoid fonksiyonu geliştirmişlerdir. Bu yöntem literatürde ilk olarak, bir sigmoid fonksiyonu kullanılarak gerçek hız vektörü değerini 0 veya 1 değerine dönüştürdüğü PSO algoritmasına uygulanmıştır. Mirjalili ve Lewis[30] tarafından önerilen sekiz transfer fonksiyonu kullanılır. S-biçimli fonksiyonlar Çizelge 2.1.'de, V-biçimli fonksiyonlar Çizelge 2.2.'de gösterilmiştir. Ayrıca, S-biçimli ve V-biçimli fonksiyonların grafiksel gösterimi Şekil 2.1.'de gösterilmiştir.

Çizelge 2. 1. S-biçimli transfer fonksiyonları [30].

İsmi	Fonksiyonu
S1	$T(x) = \frac{1}{1+e^{-2x}}$
S2	$T(x) = \frac{1}{1+e^{-x}}$
S3	$T(x) = \frac{1}{1+e^{(-x/2)}}$
S4	$T(x) = \frac{1}{1+e^{(-x/3)}}$

2.2.2. Açı Modülasyonu

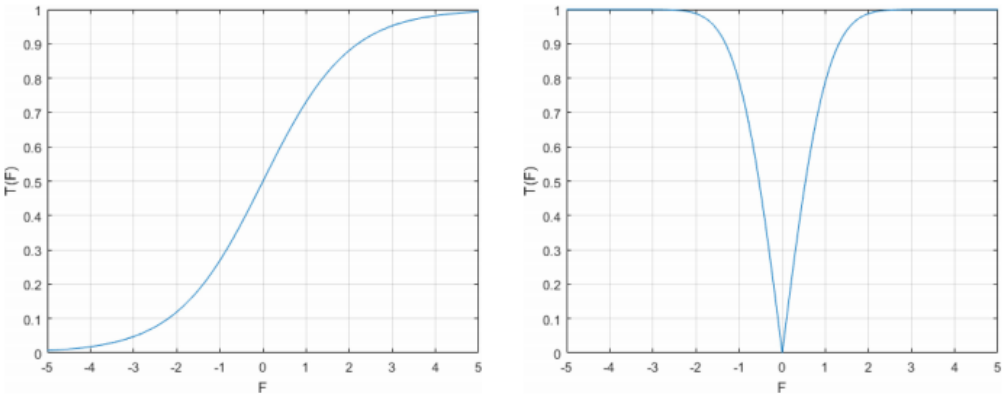
Bu teknik telekomünikasyon endüstrisinden türetilmiş olup, öncelikle sinyal işleme alanında kullanılmaktadır**.** İkili dizileri oluşturmak amacıyla birleşik sinüs ve kosinüs fonksiyonları kullanılmaktır. Fonksiyonun dört sürekli değerli parametresi vardır ve her bir parametre seti yalnızca bir ikili diziyi temsil etmektedir [28] .

2.2.3. Kuantum İlhamlı Bitler

Kuantum hesaplama bilim dalına dayanarak geliştirilen tekniğe göre, her bir ajan bir çift sayıyla tanımlanır ve aday çözümler üretmek amacıyla, dönme açısının devamlı olarak yenilendiği bir rotasyon matrisi kullanılmaktadır [28].

Çizelge 2. 2. V-biçimli transfer fonksiyonları [30].

İsmi	Fonksiyonu
V1	$T(x) = \left\lvert \operatorname{erf}\left(\frac{\sqrt{\pi}}{2}x\right)\right\rvert$
V2	$T(x) = \lvert\tanh(x)\rvert$
V3	$T(x) = \left\lvert \frac{x}{\sqrt{1+x^2}}\right\rvert$
V4	$T(x) = \left\lvert \frac{2}{\pi}\arctan\left(\frac{\pi}{2}x\right)\right\rvert$



Şekil 2. 1. Transfer fonksiyonlarının grafiksel gösterimleri.

2.2.4. Genetik Operatörler

İkili optimizasyon problemleri arama uzayının ikili yapıda olması, evrimsel algoritmalarda ikili çaprazlama ve takas operatörü gibi genetik operatörlerin uygulanabilmesine imkan vermektedir. Öztürk ve ark. tarafından, özel arama stratejileri tasarlamak amacıyla genetik operatörler kullanılmış ve ABC algoritmasının etkin bir ikili versiyonu tavsiye edilmiştir [28].

2.2.5. İkili Operatörler

İkili optimizasyon problemlerinin çözümünde ikili değerler ile çalışan ‘’_and_’’ (ve), ‘’_or_’’ (veya), ‘’_not_’’ (değil) ve ‘’_xor_’’ operatörleri kullanılabilmektedir [28].

2.2.6. Benzerlik Tabanlı Yaklaşımlar

Bu teknik ikili dizilerin benzerlik oranlarını hesaplayıp bu değerlere göre yeni aday çözüm üretilmektedir.

2.3. İKİLİ OPTİMİZASYON LİTERATÜR ARAŞTIRMASI

İkili optimizasyon problemleri için kesin, yaklaşık ve metasezgisel algoritmalar gibi birçok farklı yaklaşım vardır. Literatürde farklı ikili optimizasyon problemleri çeşitli algoritmalarla çözülmeye çalışılmıştır. Bu çalışmalardan bazıları aşağıda gruplandırılarak özetlenmiştir.

İkili optimizasyon problemleri, kombinatoryal optimizasyon problemlerinin önemli bir sınıfını temsil eden ikili tabanlı bir problem uzayı olarak gösterilir [31]. Bazı optimizasyon problemleri sürekli arama uzayları ile çözebilirken, bazı problemler ayrık arama uzaya sahiptir [29], [30], [32]. Sürekli optimizasyonda, arama uzayı sürekli değerler alırken, ikili optimizasyonda arama uzayı {0,1} değerleri alır. "0" yokluğu ve "1" varlığı temsil etmektedir. Arama uzayında bu iki değeri kullanarak birçok problem ikili uzayda çözülebilmektedir. İkili optimizasyon problemleri, tesis konumu (acil durum araçları, sağlık merkezleri ve ticari banka şubeleri dahil) ve planlama görevlerini (bütçeleme, esnek üretim sistemleri, telekomünikasyon, toplu taşıma hizmetleri, rüzgar türbini yerleşimi vb.) içermektedir [33-35] . Ayrıca, iyi bilinen ikili NP-zor optimizasyon problemleri (SÇP, kaynak tahsis problemi, boyutsal küçültme, ÖS, ağ optimizasyonu, BYP ve hücre oluşumu dahil) çözümünde kullanılmaktadır [36-40].

Literatürde birçok geleneksel yöntem ikili optimizasyon problemlerini çözmek için önerilmiştir.Bu geleneksel yöntemler, daha az boyutlu problemler için kaliteli çözümler sağlamaktadır. Boyut sayısı artarsa, hesaplama zorlaşabilmektedir. Genel olarak, büyük ölçekli ikili optimizasyon problemleri için metasezgisel yöntemler gibi makul bir sürede kabul edilebilir düzeyde çözümler bulan bazı etkili algoritmalara ihtiyaç vardır. Bu problemler için Ghosh [41] tarafından TA algoritmasına dayalı sezgisel komşuluk arama bir yöntem geliştirmiştir. Bu yöntemde, komşuluk mevcut olan en iyi çözümü bulmak için tüm çözümlerin komşuluğu arama geçmişine bağlı olarak sürekli olarak değişmektedir. Başka bir çalışmada, Zhuang ve Galiana[42], 100 termal birim yüklenme problemine kadar olan sistemlerde BT algoritması uygulamışlardır. Önerilen algoritmada, rastgele uygun bir çözüm üreteci ve temel optimizasyon algoritması kullanır.Aydın ve Fogarty [43], evrimsel benzetilmiş tavlama (EBT) algoritması ve dağıtılmış evrimsel benzetilmiş tavlama (dEBT) önermişler ve algoritmaların performansı, klasik atölye çizelgeleme problemi ve KTYP olmak üzere iki yaygın kıyaslama problemi üzerinde incelenmiştir. Falkenauer ve Bouffouix[44], atölye çizelgeleme problemini çözmek için GA'ya dayalı bir çalışma yapmışlardır. Khuri vd. [45], GENEsYs adlı GA tabanlı bir algoritma önermiş ve 0-1 çoklu sırt çantası problemine uygulanmıştır.Alana özgü bilgilerle geliştirilen diğer birçok GA tabanlı algoritmanın aksine, GENEsYs, uygulanabilir olmayan çözümleri cezalandırmak için basit bir uygunluk fonksiyonu kullanmıştır [46].

Tesis yerleşim problemi (TYP) için literatürde farklı çalışmalar bulunmaktadır. TYP, operasyon yönetimi ve bilgisayar bilimindeki en temel problemlerden biridir [47–52]. Kuehn ve Hamburger, problem için iki aşamalı ilk sezgisel yöntemi geliştirmiştir [47]. Barcelo ve ark. [53] kapasiteli tesis yerleşim problemi için lagrange gevşeme sezgisel algoritması geliştirmiştir. TYP'in diğer bir versiyonu olan kapasitesiz tesis yerleşim problemi (KTYP) kombinatoryal optimizasyon problemi olup Np-zor yapıdadır [54]. Problemi çözmek için farklı yaklaşımlar önerilmiştir. KTYP'yi çözmek için dal sınır [55,56] , doğrusal programlama [57], sabit faktör yaklaşım algoritması [58], yarı Lagrange gevşemesi [59] , vekil yarı Lagrange dual [60], ikili tabanlı yöntem [60], temel ikili tabanlı yaklaşım [61], gibi farklı deterministik yöntemler geliştirilmiştir. Ayrıca, TA [62,63], bilinçsiz arama [64] , ayrık PSO [65], ABC[66,67], GA [68], BT

[68], KKO [69], ateş böceği algoritması (ABA) [70] gibi birçok metasezgisel yöntem KTYP'yi çözmek için geliştirilmiştir [71].

Mantık operatörleri ve benzerlik ölçme teknikleri, literatürdeki aday çözümlerin üretim aşamasında sıklıkla ikili algoritmalar tarafından kullanılmaktadır. Mantık operatörleri, çözüm uzayında ikili yapılar üzerinde çalışmak için kullanılabilmektedir. Bu operatörlerin giriş ve çıkış değerleri {0,1} ikili değerlerden oluşmaktadır. Son yıllarda optimizasyon problemlerinde kullanımı popüler hale gelmiştir. Mantık operatörlerin, arama alanındaki ikili optimizasyon algoritmalarının arama kapasitesini geliştirmiştir. Benzerlik ölçüm tekniği, iki farklı ikili yapı arasındaki benzerliği ölçmek için geliştirilmiştir. Yapay Alg Algoritması (YAA) [34], ikili optimizasyon problemleri çözümünde lojik özel veya "xor" operatörü yardımıyla aday çözümler elde ettikten sonra, ikinci mekanizma olarak, ilk mekanizmadan elde edilen çıkarım yardımıyla stigmerjik davranış temeli kullanarak ikili hale getirilmiş olmaktadır. BinAAA olarak adlandırılan ikili YAA, sonuçların çözüm kalitesini, yakınsamasını ve sağlamlığını göstermek için ikili kıyaslama problemleri üzerinde test edilmiştir. Çınar ve Kiran, mantık kapıları (LogicTSA) ve benzerlik ölçüm teknikleri (SimTSA) kullanarak çeşitli ikili optimizasyon problemlerini çözmek için ikili ağaç tohum algoritmaları önermiştir. Yerel arama performansını iyileştirip KTYP'yi çözmek için çaprazlama teknikleri ve mutasyonları kullanarak İyileştirilmiş Dağılım Arama Metodu bir sürümü önerilmiştir [72]. Yöntemin iyileştirme tekniklerinin etkileri, farklı KTYP veri seti üzerinde test edilmiştir. Ayrıca, sağlamlığı ve etkinliği değerlendirmek için, geliştirilmiş yöntem diğer yöntemlerle karşılaştırılmıştır. Aslan vd. Jaya algoritmasını xor operatörü (JayaX) ile geliştirilmiştir. Aslan vd. (2019), "xor" mantık kapısını yerel arama stratejisiyle (JayaX ve JayaX-LSM) entegre ederek Jaya'nın yeni bir ikili sürümünü önermektedir. "xor", sömürü ve keşif arasında bir denge sağlar ve önerilen algoritmada kullanılan yerel arama modülü, küresel en iyi çözüm etrafında daha iyi bir çözüm bulmaya çalışmıştır [46].

Düello Optimizasyon Algoritması (DOA), Biyanto ve ark. [73] tarafından insan mücadelesine dayanan ve her düellocudan öğrenen bir optimizasyon algoritması modellenmiştir. Algoritmayı kontrol sistemleri [74], damıtma kolonosi tasarımı [75], petrol sondajı ve yeşil bina tasarımı [76] verimliliğini arttırmak için kullanmıştır.

Önerilen algoritma, ilk düellocularla başlar. Düello, kazanan ve kaybedenleri belirlemektir. Kazanan, dövüş yeteneklerini geliştirebilecek yeni becerilerini veya tekniklerini denerken, kaybeden kazanandan öğrenir. En yüksek dövüş yeteneklerine sahip birkaç düellocu şampiyon olur. Şampiyon, yetenekleri gibi yeni bir düellocu yetiştirir. Yeni düellocu, her şampiyonun temsilcisi olarak turnuvaya katılır. Tüm düellocular yeniden değerlendirilir ve en kötü dövüş yeteneklerine sahip düellocular, düellocu sayısını korumak için elenir. Çalışmada benchmark (kıyaslama) fonksiyonu kullanılmıştır. Sonuçlar, DOA çeşitli fonksiyonlar diğer algoritmalardan daha iyi performans gösterdiğini göstermektedir [73].

BÖLÜM 3

YÖNTEM

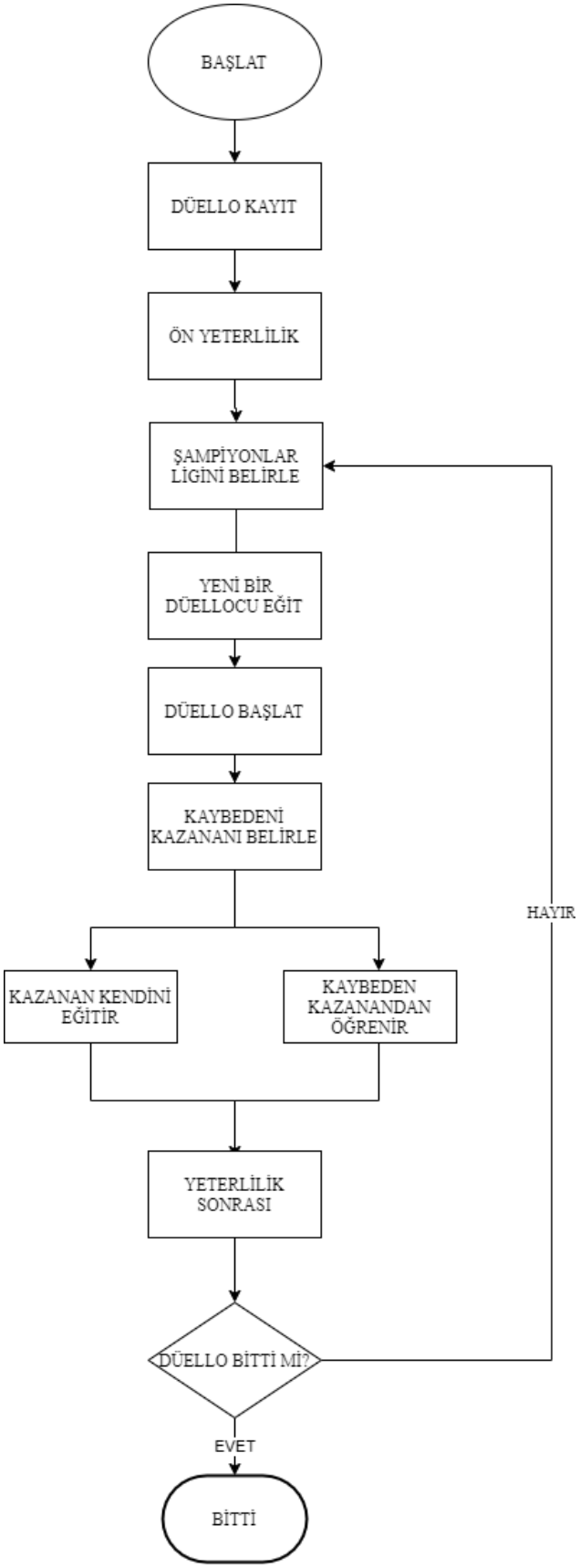
3.1. DÜELLO OPTİMİZASYON ALGORİTMASI

DOA, insan dövüşü ve öğrenme yeteneklerinden esinlenen GA'ya dayalı yeni bir algoritmadır. Genel olarak, GA'da bir bireyi yeni bireye dönüştürmenin iki yolu vardır. Birincisi çaprazlama (crossover) operatörü, yeni bir birey üretmek için farklı bireyler arasında bilgi değişimini sağlayan işleçtir. İkincisi mutasyon (mutation), yeni bireylerin ebeveynlerinin birebir olması olasılığını önlemek ve türdeki çeşitliliği artırmak amacıyla, bireyin yenisine dönüştüğü evredir.

DOA, popülasyon içindeki tüm bireylere düellocu denir. Tüm bu popülasyon içindeki düellocular şampiyonları, kazananları ve kaybedenleri belirlemek için birebir savaşırlar. Gerçek hayatta olduğu gibi mücadelenin sonunda en güçlünün kaybetme olasılığı vardır. Zayıf olanın ise kazanacak kadar şanslı olma olasılığı vardır. Her düellocunun kendini geliştirmesi için iki yol vardır. Bunlardan biri olan inovasyon, sadece kazananların daha iyiye ulaşmak için çözümleri yeniliklerle geliştirilmesi evresidir. Diğer bir çözümü geliştirme yöntemi, kaybedenlerin kazananların deneyimlerinden öğrenerek kendini dönüştürme işlemidir. GA'da hem mutasyon hem de çaprazlama en iyi çözümü bulmak için herhangi bir çözüm üretmede kör (yetersiz) gibi görünmektedir. Bu kör nokta, GA'da üretilen her çözümün veya bireyin daha optimum bir çözüme sahip olamayacağı anlamına gelir. Aslında, GA optimum çözümü üretmeye çalışırken kötü çözüme düşebilir. DOA, düelloculara sınıflandırmalarına göre farklı işlemler uygulayarak bu kör etkiyi en aza indirmeye çalışır.

Düello, bir veya daha fazla kişi ile diğer kişi(ler) arasında bir maç olarak tanımlanabilir. Düello, örneğin satranç ve köprü oyunlarında fiziksel güç, beceri ve entellektüel yetenekler gerektirir. Fiziksel gücü içeren yaygın düello tipilerinden biri bokstur. Boks, iki kişinin her birini belirli kurallar altında nakavt etmesi gereken dünyanın en popüler sporlarından biridir. Her düelloda kazananlar, kaybedenler ve kurallar vardır. Bir maçta kazanan olma olasılığı güç, beceri ve şansa bağlıdır. Düellodan sonra, kazananın ve kaybedenlerin yeteneklerini bilmek çok önemlidir. Kaybeden kazanandan öğrenebilir ve kazanan inovatif girişimlerle yetenek ve becerilerini geliştirebilir. Önerilen algoritmada, her düellocu eşsiz bir zafer için rakibinden yeni teknik veya beceriler öğrenerek kendini geliştirir.

Algoritmanın akış şeması Şekil. 3.1'de gösterilmektedir [73]. İlk olarak, düellocu popülasyonu kaydedilir. Düellocuların özellikleri ikili diziye kodlanmıştır. Bütün düellocular, dövüş yeteneklerini belirlemek için değerlendirilir. Düello planı, bir dizi düello katılımcısını içeren her düellocu için ayarlanır. Düelloda, her düellocu diğer düellocu ile bire bir savaşacaktır. Bu bire bir mücadele, yerel optimumdan kaçınmak için gladyatör savaşı yerine kullanılır. Her düelloda, dövüş yeteneklerine ve şanslarına bağlı olarak bir kazanan ve kaybeden üretilir. Düellodan sonra şampiyon da belirlenir. Bu şampiyonlar en iyi savaş yeteneklerine sahip düellocular arasından seçilir. Daha sonra, her kazanan ve kaybeden dövüş yeteneklerini yükseltme fırsatına sahip olurlar, bu arada her şampiyonun yeni düellocu yetenekleriyle eğitilmesini sağlanır. Yeni düellocu bir sonraki maça katılır. Her kaybeden, yetenek setinin belirli bir bölümünü kazananın yetenek setiyle değiştirerek rakiplerinden nasıl daha iyi bir düellocu olacağını öğrenecektir. Öte yandan kazanan, yetenek seti değerlerini değiştirerek yeni bir yetenek geliştirmeye çalışacaktır. Her düellocu savaş yeteneği bir sonraki düello için yeniden değerlendirilir. Tüm düellocular daha sonra eleme sonrası yeniden değerlendirilir ve kimin şampiyon olacağı belirlemeye başlanılır. Şampiyonlar tarafından eğitilen yeni düellocular olduğundan, en kötü düellocuların tümü düellodaki düellocuların sayısını korumak için elenir. Bu işlem düello bitene kadar devam edecektir. DOA'daki aşamalar aşağıdaki şekilde verilmiştir.



Şekil 3. 1. Düello algoritması akış şeması [73].

1. Bir düellocu setindeki her düellocu ikili dizi kullanılarak kaydedilir. Düellocu algoritmasında ikili dizi yetenek seti olarak adlandırılır. N_{var} boyutlu optimizasyon probleminde, düellocu sayısı N_{var} boyutunun iki katı boyuta sahip olacaktır.
2. Ön yeterlilik, her düellocuya, yeteneklerini temel alarak dövüş yeteneklerini ölçmek veya değerlendirmek için verilen bir testtir.
3. En iyi düellocuların bulunduğu şampiyonlar dizisi belirlenir. Her şampiyon, düello yetenekleriyle aynı şekilde yeni bir düellocu eğitilir. Bu yeni düellocu, şampiyonun düellodaki yerini alır ve bir sonraki savaşa katılır.
4. Her düellocu arasındaki düello planı rastgele belirlenir. Her düellocu, savaş yeteneklerini ve şansını kullanarak kazananı ve kaybedeni belirlemek için savaşır. Düello basit bir mantık kullanarak hareket eder. Düellocu A'nın şansı düellocu B'den daha yüksekse, düellocu A kazanır bunun tam terside mümkündür. Düellocunun şansı tamamen rastgeledir. Şans katsayılarını belirlemek için sözde kod algoritması Çizelge 3.1'de gösterilmiştir [73].

Çizelge 3. 1. Kazanan ve kaybedenlerin belirlenmesi sözde kodu [73].

```
1.  FC  =  Dövüş Yetenekleri  LC = Şans Katsayısı
2.  A (Şans) = A(FC) * (LC + random(0-1) * LC)
3.  B (Şans) = B(FC) * (LC + random(0-1) * LC)
4.              if (A (Şans) + A(FC)) > (B (Şans) + B(FC))
5.                  A(Kazanan) = 1;
6.                  B(Kazanan) = 0;
7.              else
8.                  A(Kazanan) = 0;
9.                  B(Kazanan) = 1;
```

5. Düellodan sonra, her düellocu şampiyon, kazanan ve kaybeden ayrılır. Her düellocunun dövüş yeteneklerini geliştirmek için, her kategori için üç tür iyileştirme vardır. Kaybedenler için ilk iyileştirme, her kaybeden kazanandan öğrenir. Eğitim, kaybedenin kazananın yetenek setinin bir kısmını kopyalayabileceği anlamına gelir. İkinci iyileştirme kazananlar için

tasarlanmıştır, her kazanan yeteneklerini geliştirir, yeni deneme yapar. Bu iyileştirme, kazananın yetenek seti ile rastgele manipülasyonlardan oluşur. Sonunda, her şampiyon yeni bir düellocu oluşturur. Kazanan ve kaybedeni belirlemek için sözde kod algoritması Çizelge 3.2 ’de gösterilmiştir [73].

Çizelge 3. 2. Kazanan ve kaybeden iyileştirme sözde kodu [73].

```
1.  if A (kazanan) = 1;
2.      for i = 1 : (yset_uzunlugu)
3.          D = random(0...1);
4.          if D < inovasyon
5.              A (yset) = rand(0...9);
6.          end
7.      end
8.  else
9.      for i = 1 : (yset_uzunlugu)
10.         E = random(0...1);
11.         if E < ogrenme_katsayisi
12.             A (yset) = B (yset);
13.         end
14.     end
15. end
```

Birkaç yeni düellocu devreye girdiğinde, düellocu sayısının aynı kalması için bir istisna olmalıdır. İstisna, her düellocunun düello yeteneklerine dayanır. En kötü düello yeteneklerine sahip düellocular hariç tutulur.

3.2. ÖNERİLEN İKİLİ DÜELLO OPTİMİZASYON ALGORİTMASI

İkili optimizasyonda, olası çözüm kümesindeki her bir değer “1” veya “0” olmalıdır. DOA’nın ikili optimizasyon problemlerini ele alabilmesi için, çözüm uzayı ilk olarak ikili değerlere {0,1} dönüştürülmelidir. Bu işlem ilk adımda rastgele olarak gerçekleştirilmektedir. [0,1] aralığında seçilen rastgele bir sayı 0.5’ten küçükse 0, büyükse 1 olarak kabul edilir. Bu işlemler DOA’daki her bir birey için tekrarlanır ve olası ilk çözümler bu adımın sonunda elde edilir. Bu çözümler kullanılarak kazananlar, kaybedenler ve şampiyonlar belirlenir.

Bu çalışmada mutasyon ve mantıksal operatörler yeni bir çözüm elde etmek için kullanılmaktadır. Mantık kapıları ‘ve’, ‘veya’, ‘değil’ ve ‘xor’ operatörleri olarak adlandırılır. Bu nedenle, algoritmalarda kullanılan mantık operatörleri oldukça önemlidir. Çünkü güncelleme işlemi için ‘veya’ operatörü kullanılıyorsa, yalnızca iki aday çözüm karşılık gelen boyut 0 alırsa, yeni çözüm karşılık gelen boyut 0 olacaktır. Dolayısıyla, güncelleme mekanizması için "veya" operatörü kullanılırsa, yeni çözüme karşılık gelen boyut %75 olasılıkla "1" olacaktır. Güncelleme mekanizması için "ve" operatörü kullanılırsa "1" olasılığı % 25 olacaktır [77]. Önerilen yaklaşımda, denklem 3.1’deki bitsel işlemler kullanılmıştır.

$$X = X \oplus (Rand\{0,1\} \,\&\, (X \oplus Y)) \tag{3.1}$$

Sonuç olarak, Çizelge 3.3'te verilen "xor" kapısının doğruluk tablosu kontrol edildiğinde, "0" ve "1" seçim olasılığının eşit olduğu görülecektir. $\oplus$ sembolü "xor" mantık operatörünü temsil eder.

Çizelge 3. 3. XOR mantık kapısı için doğruluk tablosu.

Giriş		**Çıkış**
A	**B**	
0	0	0
0	1	1
1	0	1
1	1	0

İlk çözümler elde edildikten sonra önerilen yöntemin adımları aşağıdaki gibidir:

1. Tüm düellocular arasında en iyi çözüme sahip düellocuların %10'u şampiyon olarak seçilir.
2. Geri kalan düellocular ikili olarak gruplanır ve aralarında düello yaparlar. Bu dülleoda şans kat sayısı düelloya giren kötü çözüme sahip bireylerin kazanma şansını artırabilir. Düello sonucunda kazananlar ve kaybedenler belirlenir.
3. Kazananlar inovasyon katsayısı kullanılarak belli bir oranda inovasyona (mutasyon) tabi tutulur.
4. Kaybedenler ise öğrenme katsayısı kullanılarak kazanan düelloculardan çözüme ait değerleri kopyalabilir (yeteneklerini geliştirirler).
5. Şampiyonlar ise kendi aralarında bir şampiyonlar ligi düzenler. Her bir şampiyon eşleştiği şampiyon ile bitsel operatörler kullanılarak yeni bir çözüm elde eder.
6. Düellocular arasından en kötü çözüme sahip düellocuların %10'u listeden çıkarılır ve yeni düellocular davet edilir. Bununla yeni yeteneklerin kazandırılması amaçlanmaktadır.
7. 1-6 arası işlemler maksimum iterasyon sayısı elde edilinceye kadar tekrar eder. Bu adımların sözde kodu Çizelge 3.2'de verilmiştir.

Çizelge 3. 4. İkili Düello Optimizasyon Algoritması.

1. Parametreleri (İnovasyon, Şans, Öğrenme…) ayarla.
2. Düellocunun ilk çözümünü rastgele üret.
3. Düellocunun çözümüne ait yetenek değerini bul.
4. for i = 1 : (maks_iter)
5. Şampiyonları belirle
6. for j = 1 : $N_{düello}$
7. ŞansA için katsayısını belirle
8. ŞansB için katsayısını belirle
9. if ŞansA > ŞansB:
10. Kazanan.ekle(A)
11. Kaybeden.ekle(B)
12. else:
13. Kazanan.ekle(B)
14. Kaybeden.ekle (A)
15. end
16. end
17. for k = 1 :
18. if rand(0,1) <inovasyon_ksayısı
19. inovasyon (kazanan ^{k})
20. end
21. for k = kaybeden sayısı:
22. if rand(0,1) < ögrenme_ksayısı
23. öğrenme (kaybeden ^{k})
24. end
25. for k = 1 : samp_sayısı
26. if rand(0,1) < ögrenme_ksayısı
27. $X^{k} = X^{k} \oplus (\text{Rand}\{0,1\} \& (X^{k} \oplus X^{k+1})$
28. k = k + 1
29. end
30. end
31. end

BÖLÜM 4

DENEYSEL SONUÇLAR

4.1. ONE-MAX (BIR-ENB) PROBLEMİ

One-max problemi, uyarlanabilir operatör seçim algoritmalarının performansını değerlendirmek için yaygın olarak kullanılan iyi bilinen bir problemdir. One-max problemi, n-uzunluğundaki bit dizelerini dikkate alır; 0^n bireylerden başlayarak (yani, n sıfırlardan oluşan dizeler), amaç, birimlerin sayısını en üst düzeye çıkarmak, yani bir 1^n -bit dizesine ulaşmaktır. $|x|_1$ ile işaretlenmiş x bit dizesinin tahmini, birim sayısına karşılık gelir [78].

One-max problemi, genetik algoritmayı temsil eden en temel problemdir. Bu problem, bir ikili dizinin maksimum değerini hesaplamaya odaklanır. Bir dizi ikili dizeler oluşturmak için, madeni paraların istenen ikili miktarı kadar atılması manuel olarak yapılabilir. Bilgisayarlı ise, bir ikili dize oluşumu, gerektiği kadar 0 ve 1 rastgele sayı üreterek yapmak çok kolay olacaktır. Belirtilen ikili dizenin uygunluk (fitness) değerini hesaplamak için aşağıdaki fonksiyon kullanılmaktadır [79]:

$$f(x) = \sum_{i=1}^{L} x_i \tag{4.1}$$

burada L düellocu popülasyonu, x_i düellodaki bireyleri ifade eder.

4.1.1. Deneysel Sonuçlar ve Karşılaştırmalar

Bu tez çalışmasında, farklı metasezgisel optimizasyon algoritması kullanılarak ikili optimizasyon problemleri çözülmüştür. Çalışmada kullanılan İDOA ile GA kıyaslanmıştır. Bu çalışmada kullanılacak algoritmalar Intel (R) Core (TM) i7-4510U işlemci, 2.00 GHz hız ve 8 GB RAM, 64 bit işletim sistemine sahip bilgisayarda Python 3.8 ortamında kodlanarak geliştirilmiş ve analiz edilmiştir.

Çizelge 4. 1. Deneyler için parametre ayarı.

Algoritma	Parametre	Değer
Genel Ayarlar	Populasyon	80
	İterasyon	500
	Boyut	[50,100,500,1000]
İDOA	Şans Katsayısı	0.1
	İnovasyon Katsayısı	0.01
	Öğrenme Katsayısı	0.8
GA	Mutasyon Katsayısı	0.01
	Çaprazlama	0.5

GA için 3 farklı çaprazlama operatörü kullanılmış ve her birinin sonuçları İDOA ile ayrı ayrı karşılaştırılmıştır. Yöntemlerin kıyaslanması için, ortak kontrol parametreleri Çizelge 4.1'de görüldüğü üzere birbirlerine eşit olarak seçilmiştir. Böylece bütün metotlarda, popülasyon büyüklüğü 60,80,100 için test edilmiştir. GA'nın tek noktadan, iki noktadan ve çok noktadan çaprazlama versiyonlarında [80] tarafından önerildiği üzere, çaprazlama oranı (crossover rate) 0.8 ve mutasyon oranı (mutation rate) 0.01 olarak tercih edilmiştir. İDOA parametre değerleri Çizelge 4.1'de GA değerlerine denk seçilip öğrenme değeri 0.8, inovasyon değeri 0.01 ve şans katsayısı 0.1 olarak ayarlanmıştır.

Her bir problem için tüm algoritmalar 10 kez birbirinden bağımsız olarak koşturulmuş ve optimum değeri bulma sayılarının ortalama değeri ve standart sapması, Çizelge 4.2'de sunulmuştur. İDOA uygulanan yöntemde, problemin maksimum değerini en kısa sürede bulmak için yapılan bu çalışmada 50, 100, 500 ve 1000 iterasyon sayıları ile 80 popülasyon sayıları kullanılmıştır. İDOA'nın ikili algoritmaya çevirmede

kullanılan Exclusive-or (xor) yöntemlerinin sonuçları verilmiştir. Bu yöntemler ile ulaşılan maksimum değerler Çizelge 4.2, Çizelge 4.3, Çizelge 4.4, Çizelge 4.5'de yer almaktadır.

Çizelge 4. 2. Popülasyon değeri 50 olan problemin İDOA ile GA karşılaştırma sonuçları.

	İDOA	**GA-SP**	**GA-TP**	**GA-UP**
1	50	50	50	49
2	50	50	50	50
3	50	50	50	49
4	50	50	50	50
5	50	50	50	50
6	50	50	50	47
7	50	50	50	47
8	50	50	50	50
9	50	50	50	46
10	50	50	50	49
En İyi	**50**	**50**	**50.00**	**50**
En Kötü	**50**	**50**	**50**	**46**
Ortalama	**50.00**	**50.00**	**50.00**	**48.70**
Std.Sap	**0.00**	**0.00**	**0.00**	**1.42**

Çizelge 4.2'de yer alan İDOA sonuçları ilk önce 50 iterasyon ile 50 popülasyon sayısı kullanılarak test edilmiştir. Bu popülasyon sayılarına göre optimum değerlere ulaşılmıştır. Popülasyonun boyutu 100 yapıldığında algoritmanın sonuçları Çizelge 4.3'te vermiştir. İDOA yöntemi 10 çalıştırmanın tümünde optimum sonuca ulaşmıştır. GA yöntemleri kullanıldığında ise 10 denemenin hepsinde en uygun çözüme ulaşılamamıştır. Tek noktadan çaprazlamalı (GA-SP) ve iki noktadan çaprazlamalı (GA-TP) yönteminlerinde sonuç değerleri birbirine yakın bütün noktalardan çaprazlamalı (GA-UP) yönteme göre güven aralıkları daha güçlü performans sergilemektedir.

Çizelge 4. 3. Popülasyon değeri 100 olan problemin İDOA ile GA karşılaştırma sonuçları.

	İDOA	**GA-SP**	**GA-TP**	**GA-UP**
1	100	91	95	84
2	100	95	93	82
3	100	93	96	80
4	100	91	96	78
5	100	96	97	79
6	100	92	96	79
7	100	94	94	82
8	100	92	92	81
9	100	92	96	79
10	100	96	95	84
En İyi	**100**	**96**	**97.00**	**84**
En Kötü	**100**	**91**	**92**	**78**
Ortalama	**100**	**93.20**	**95**	**80.80**
Std.Sap	**0.00**	**1.83**	**1.48**	**2.04**

Çizelge 4. 4. Popülasyon değeri 500 olan problemin İDOA ile GA karşılaştırma sonuçları.

	İDOA	**GA-SP**	**GA-TP**	**GA-UP**
1	500	344	335	306
2	499	337	366	310
3	500	340	348	298
4	500	350	356	307
5	500	329	353	305
6	500	339	350	306
7	499	339	337	302
8	500	336	361	305
9	500	337	340	309
10	500	341	337	295
En İyi	**500**	**350**	**366.00**	**310**
En Kötü	**499**	**329**	**335**	**295**
Ortalama	**499.80**	**339.20**	**348.30**	**304.30**
Std.Sap	**0.40**	**5.17**	**10.30**	**4.47**

İDOA performansını daha net değerlendirilmesi için popülasyon boyutu artılıp test edilmiştir. Popülasyon boyutu 500 yapıldığında sonuçlar Çizelge 4.4'te verilmiştir. Boyut 500'e çıkarılıp test edildiğinde maksimum değere İDOA yöntemi ulaşabilmiştir.

GA'nın kullanılan 3 farklı yöntemde çözümün maksimum değeri 366 olarak belirlenmiştir.

İDOA performansını daha net değerlendirilmesi için popülasyon boyutu artılıp test edilmiştir. Popülasyon boyutu 1000 yapıldığında sonuçlar Çizelge 4.5'de verilmiştir. Boyut artırılarak 1000'e çıkarılıp algoritma test edildiğinde optimum sonuç açısından daha üstün bir performans sergilemektedir. Önerilen algoritma, tüm boyutlarda GA'lardan daha iyidir.

Çizelge 4. 5. Popülasyon değeri 1000 olan problemin İDOA ile GA karşılaştırma sonuçları.

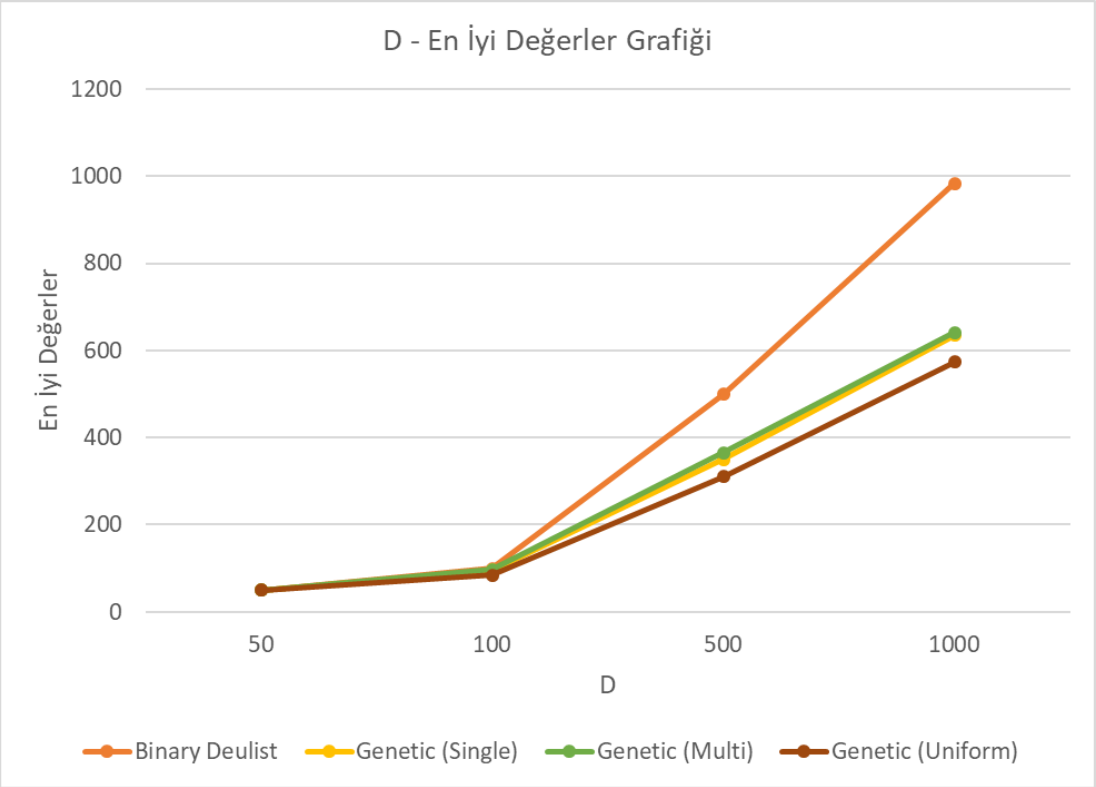
	İDOA	**GA-SP**	**GA-TP**	**GA-UP**
1	977	600	612	564
2	979	616	627	563
3	970	616	618	563
4	980	623	625	574
5	979	618	641	557
6	972	635	633	560
7	982	604	642	567
8	978	609	614	559
9	984	613	627	561
10	980	618	624	564
En İyi	**984**	**635**	**642.00**	**574**
En Kötü	**970**	**600**	**612**	**557**
Ortalama	**978.10**	**615.20**	**626.30**	**563.20**
Std.Sap	**4.04**	**9.33**	**9.70**	**4.51**

Exclusive-or (xor) operatörü problem çözümü kullanılarak yapılan ilk karşılaştırmada, önerilen yöntem GA çarpazlama operatörleriyle kıyaslanmıştır. Bu sonuçlara dayanılarak yapılan kıyaslama Çizelge 4.5'de görülmektedir. İDOA yöntemi ile problemlerin çözümü, standart sapma değerleri açısından, diğer algoritmalardan daha güçlü bir performans ortaya koymaktadır.

Çizelge 4. 6. İDOA ile GA yöntemleri sonuçlarının karşılaştırılması.

Yöntem		**50**	**100**	**500**	**1000**
İDOA	En İyi	50	100	500	984
	En Kötü	50	100	499	970
	Ortalama	50	100	499.8	978.1
	Std.Sap.	0	0	0.40	4.04
GA-SP	En İyi	50	96	350	635
	En Kötü	50	91	329	600
	Ortalama	50	93.2	339	615.2
	Std.Sap.	0	1.83	5.17	9.33
GA-TP	En İyi	50	97	366	642
	En Kötü	50	92	335	612
	Ortalama	50	95	348.3	626.3
	Std.Sap.	0	1.48	10.30	9.70
GA-UP	En İyi	50	84	310	574
	En Kötü	46	78	295	557
	Ortalama	48.7	80.8	304.3	563.2
	Std.Sap.	1.42	2.04	4.47	4.51

Yapılan karşılaşlaştırmanın daha anlaşılır ve daha görünür hale gelmesi amacıyla Çizelge 4.6. hazırlanmıştır. Grafikteki popülasyon büyüklüğü değeri 50 olduğunda algoritmaların performansları birbirine yakın olmaktadır. Aynı parametrelerle popülasyon boyutu 100 olduğunda İDOA'sı 0 (sıfır) standart sapma değeriyle sonucu bulmuştur. Önerilen algoritma, problem boyut sayısı artırıldığında, Şekil 4.2'deki grafikten de anlaşılacağı üzere, daha güvenli performans göstermektedir.

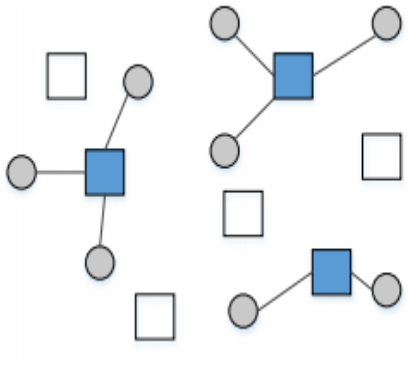


Şekil 4. 1. İDOA ile GA yakınsama grafikleri.

Deneysel çalışma sonuçları ele alındığında, geliştirilen yöntemlerin çözüm kalitesinin, standart sapma ve yakınsama karakteristikleri açısından ikili optimizasyon problemlerini çözmede rekabetçi, alternatif ve güçlü oldukları anlaşılmaktadır.

4.2. KAPASİTESİZ TESİS YERLEŞİM PROBLEMİ (KTYP)

KTYP, çeşitli uygulamalara sahip olan en popüler ikili optimizasyon problemlerinden biridir. KTYP, potansiyel tesisler içerisinden müşterilere hizmet verecek olan tesisin konumunu belirlenmeye çalışmaktadır. Amaç, açılacak her bir tesis ile tüm müşterilerin taleplerini karşılamak amacıyla, açılış maliyeti ve müşterilerin tesise ulaşım maliyeti toplamını en aza indirmektir [71] . KTYP'nin problemleri ele alış şekli Şekil 4. 2'de görülmektedir.



Şekil 4. 2. KTYP setindeki problemlerin gösterimi [81].

: Açık aday tesis yerleri

: Boş tesis yerleri

: Müşteriler

____ : Yerine getirme bağlantısı

KTYP matematiksel olarak şu şekilde ifade edilebilir;

n = potansiyel tesis sayısı ve m = müşteri sayısı

$$min \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij}\, x_{ij} + \sum_{j=1}^{m} f_i\, y_i \tag{4.2}$$

$$s.t \sum_{i=1}^{n} x_{ij} = 1, \qquad j = 1, \dots, m \tag{4.3}$$

$$x_{ij} - y_i \leq 0, i = 1, \dots, n, j = 1, \dots, m, \tag{4.4}$$

$$x_{ij} \in \{0,1\}, i = 1, \dots, n, j = 1, \dots, m, \tag{4.5}$$

$$y_i \in \{0,1\}, \qquad i = 1, \dots, n, \tag{4.6}$$

$$c_{ij} = i\ tesisinden\ j\ müşterisine\ taşıma\ maliyeti \tag{4.7}$$

$$f_i = i\ tesisi\ için\ sabit\ maliyet \tag{4.8}$$

$$x_{ij} = \begin{cases} 1 \ \ i\ tesisinden\ j\ müşteri\ hizmet\ alıyorsa \\ 0 \ \ aksine \end{cases} \quad (4.9)$$

$$y_i = \begin{cases} 1 \ \ i\ tesisi\ \ açıksa \\ 0 \ \ aksine \end{cases} \quad (4.10)$$

İlk deneyler için OR-Lib 'den alınan 12 KTYP analiz ve deneylerde kullanılmıştır. Çizelge 4.7'de problemin adı, boyutluluk (Tesisler X Müşteriler) ve optimum maliyeti verilmiştir.

Çizelge 4. 7. İDOA ile elde edilen KTYP optimum maliyet değerleri.

Problemin Adı	Boyutu (Tesisler X Müşteriler)	Optimum Maliyet
Cap71	16 X 50	932.615,750
Cap72	16 X 50	977.799,400
Cap73	16 X 50	1.010.641,450
Cap74	16 X 50	1.034.976,975
Cap101	25 X 50	796.648,437
Cap102	25 X 50	854.704,200
Cap103	25 X 50	893.782,112
Cap104	25 X 50	928.941,750
Cap131	50 X 50	793.439,562
Cap132	50 X 50	851.495,325
Cap133	50 X 50	893.076,712
Cap134	50 X 50	928.941,750

Test problemleri dört gruba ayrılmıştır. İlk grup Cap71, 72, 73 ve 74'ten oluşmaktadır. Bu problemler karar değişken (tesis) sayısı 16 olan küçük boyutlu problemler olarak adlandırılır. Cap101, 102, 103 ve 104 problemlerinde 25 karar değişkeni olup orta boyutlu problemlerdir. Büyük boyutlu problemler Cap131, 132, 133 ve 134'tür. Bu problemlerdeki çözüm uzayı, tesis boyutuna göre üstel arttığından problemi en iyi şekilde çözmek için etkili araştırma yapılması gerekir.

4.2.1 KTYP TEST SETİ KULLANILARAK YAPILAN KARŞILAŞTIRMALAR

Önerilen yöntemin performansını diğer son teknoloji algoritmalarla daha iyi anlamak için OR-Library'den alınan kapasitesiz KTYP test seti üzerinde değerlendirilir. Bu çalışmada kullanılacak algoritmalar Intel (R) Core (TM) i7-4510U işlemci, 2.00 GHz

hız ve 8 GB RAM, 64 bit işletim sistemine sahip bilgisayarda Python 3.8 ortamında kodlanarak geliştirilmiş ve analiz edilmiştir.

Çizelge 4. 8. KTYP için İDOA parametre ayarları.

Algoritma	Parametre	Değer
İDOA	Popülasyon	80
	İterasyon	500
	Şans Katsayısı	0.1
	İnovasyon Katsayısı	0.1
	Öğrenme Katsayısı	0.5

Algoritmaların adil karşılaştırmasını gerçekleştirmek amacıyla, algoritmaların ortak kontrol parametreleri birbirine eşit olarak seçilir. Popülasyon boyutu 80 ve sonlandırma koşulu maxFEs (maksimum fonksiyon çağırma) değeri 80.000 olarak ayarlandı. Önerilen yöntem için yapılan deneysel çalışmalar KTYP test seti üzerinde 30 kez çalıştırılıp elde edilen sonuçlar yakın zamanda önerilmiş algoritmaların sonuçlarıyla karşılaştırılmıştır. İDOA parametre değerleri, Çizelge 4.8'de de görüleceği gibi, öğrenme değeri 0.5, inovasyon değeri 0.1 ve şans katsayısı 0.1 olarak ayarlanmıştır.

Her bir problem için İDOA bağımsız olarak 30 kez çalıştırılmış ve elde edilen sonuçlar Çizelge 4.9'da sunulmuştur. Bu çizelge ile en iyi, en kötü, standart sapma, Gap değeri ve optimum değeri bulma sayısı (İsabet) sunulmuştur. Çizelgedeki Gap değerleri aşağıdaki gibi hesaplanmaktadır:

$$\text{Gap} = \frac{f_{ort} - f_{opt}}{f_{opt}} \text{ x } 100 \tag{4.10}$$

Denklemde, Gap değeri yöntemin 30 defa çalıştırma sonucunda elde ettiği ortalama değer ile problemin optimum değeri arasındaki oransal fazlalığı, f_{ort} yöntemin 30 defa

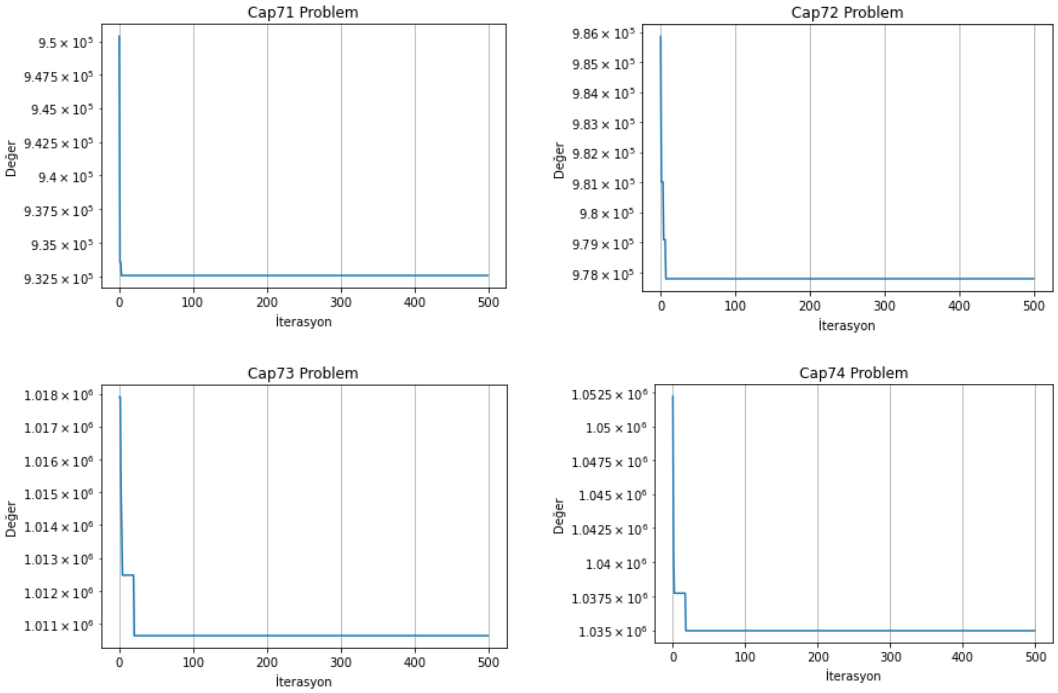
çalıştırma sonucunda elde ettiği ortalama değeri, f_{opt} ise probleminin optimum değerini ifade etmektedir.

Çizelge 4. 9. KTYP için İDOA'nın sonuçları.

Problem	**İDOA**					
	En İyi	**Ortalama**	**En Kötü**	**Std.Sap.**	**Gap**	**İsabet**
cap71	932.615,750	932.615,750	932.615,750	0,000	0,000	30
cap72	977.799,400	977.799,400	977.799,400	0,000	0,000	30
cap73	1.010.641,450	1.010.641,450	1.010.41,450	0,000	0,000	30
cap74	1.034.976,975	1.034.976.975	1.034.976,975	0,000	0,000	30
cap101	796.648,437	796.763,143	797.508,725	292,442	0,014	26
cap102	854.704,200	854.704,200	854.704,200	0,000	0,000	30
cap103	893.782,112	893.985,828	894.801,163	377,019	0,023	22
cap104	928.941,750	928.941,750	928.941,750	0,000	0,000	30
cap131	793.439,562	793.865,023	794.299,850	426,182	0.054	15
cap132	851.495,325	851.506,978	851.670,125	43,603	0.001	28
cap133	893.076,712	893.680,197	893.680,197	578,599	0.068	12
cap134	928.941,750	928.977,471	929.477,563	133,655	0.004	28

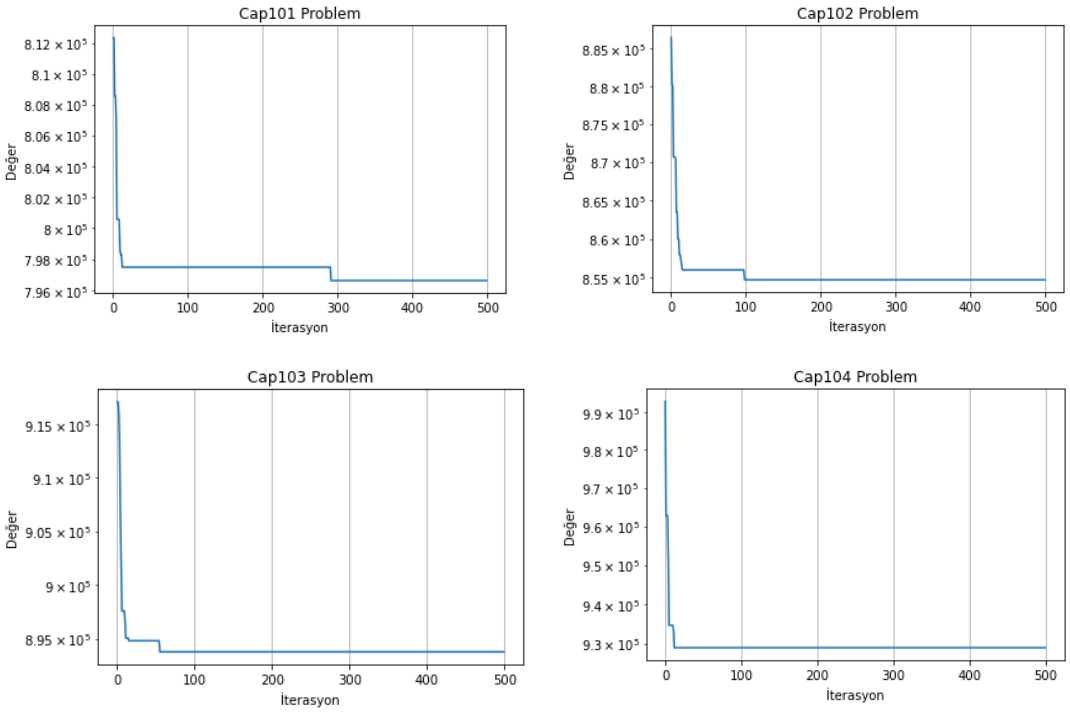
Sonuçların daha iyi ele alınabilmesi amacıyla, problemler boyutlarına göre, az, orta, ve büyük boyutlu olmak üzere 3 farklı grupta incelenmiş ve Şekil 4. 3- 4.6'de gösterilmiştir.

İDOA yöntemi kullanılarak az boyutlu Cap71, Cap72, Cap73 ve Cap74 problemleri için, ikili olarak yapılandırılmış optimizasyon algoritmalarına ait elde edilen yakınsama eğrileri Şekil 4.3'de verilmiştir.



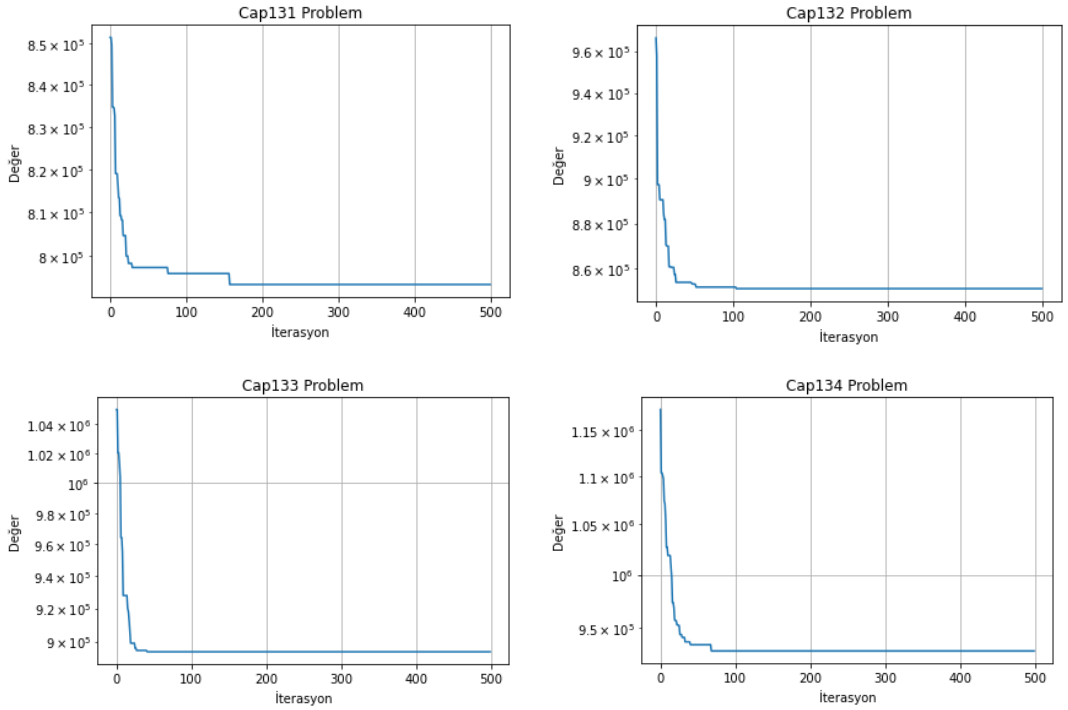
Şekil 4. 3. İDOA için Cap71, Cap72, Cap73 ve Cap74 yakınsama grafikleri.

İDOA yöntemi ile orta boyutlu Cap101, Cap102, Cap103 ve Cap104 problemleri için, ikili olarak yapılandırılmış optimizasyon algoritmalarına ait elde edilen yakınsama eğrileri Şekil 4. 4'de verilmiştir.



Şekil 4. 4**.** İDOA için Cap101, Cap102,Cap103 ve Cap104 yakınsama grafikleri.

İDOA yöntemi ile büyük boyutlu Cap131, Cap132, Cap133 ve Cap134 problemleri için, ikili olarak yapılandırılmış optimizasyon algoritmalarına ait elde edilen yakınsama eğrileri Şekil 4.5'de verilmiştir.



Şekil 4. 5. İDOA için Cap131, Cap132, Cap133 ve Cap134 yakınsama grafikleri.

Tüm algoritmalar, her bir problem için 30 kez birbirinden bağımsızca koşturulmuş ve elde edilen sonuçlar; en iyi, en kötü, standart sapma ve optimum değeri bulma sayısı (isabet) Çizelge 4.10, Çizelge 4.11, Çizelge 4.12'de gösterilmiştir. İDOA'nın sonuçları, ikili ağırlıklı süperpozisyon çekim algoritması (İASÇA) [81], ikili parçacık sürü optimizasyon algoritması (İPSO) [82] ve ikili yapay arı kolonisi algoritması (İYAK) [83]'ye dayalı algoritmaların daha önce yayınlanmış sonuçlarıyla kıyaslanmıştır.

Çizelge 4. 10. Cap71, Cap72, Cap73, Cap74 problemlerinin İDOA ve diğer ikili optimizasyon algoritmalar ile karşılaştırması.

Yöntem		**Cap71**	**Cap72**	**Cap73**	**Cap74**
İPSO	En İyi	932.615,750	977.799,400	1.010.641,450	1.034.976,980
	En Kötü	934.199,140	983.713,810	1.012.643,690	1.045.342,230
	Std.Sap.	562.230	1.324,300	702,130	2.124,540
	İsabet	25	25	22	0
İYAK	En İyi	932.615,750	977.799,400	1.010.641,450	1.034.976,975
	En Kötü	932.615,750	977.799,400	1.010.641,450	1.034.976,975
	Std.Sap.	0.000	0.000	0.000	0.000
	İsabet	30	30	30	30
İASÇA	En İyi	932.615,750	977.799,400	1.010.641,450	1.034.976,975
	En Kötü	932.615,750	977.799,400	1.010.641,450	1.034.976,975
	Std.Sap.	0.000	0.000	0.000	0.000
	İsabet	30	30	30	30
İDOA	En İyi	932.615,750	977.799,400	1.010.641,450	1.034.976,975
	En Kötü	932.615,750	977.799,400	1.010.641,450	1.034.976,975
	Std.Sap.	0.000	0.000	0.000	0.000
	İsabet	30	30	30	30

Çizelge 4. 11. Cap101, Cap102, Cap103, Cap104 problemlerinin İDOA ve diğer ikili optimizasyon algoritmalar ile karşılaştırması.

Yöntem		**Cap101**	**Cap102**	**Cap103**	**Cap104**
İPSO	En İyi	796.648,440	854.704,200	893.782,110	928.941,750
	En Kötü	802.457,230	857.380,850	899.424,910	944.394,830
	Std.Sap.	1.480,720	1.015,640	1.695,790	3.842,640
	İsabet	0	10	0	18
İYAK	En İyi	796.648,440	854.704,200	893.782,110	928.941,750
	En Kötü	796.648,440	854.704,200	894.008,140	928.941,750
	Std.Sap.	0.000	0.000	85.670	0.000
	İsabet	30	30	25	30
İASÇA	En İyi	796.648,437	854.704,200	893.782,112	928.941,750
	En Kötü	797.508,725	854.704,200	895.27,188	928.941,750
	Std.Sap.	0.000	0.000	0.000	0.000
	İsabet	30	30	30	30
İDOA	En İyi	796.648,437	854.704,200	893.782,112	928.941,750
	En Kötü	797.508,725	854.704,200	895.027,188	928.941,750
	Std.Sap.	292.442	0.000	377.019	0.000
	İsabet	26	30	22	30

Çizelge 4. 12. Cap131, Cap132, Cap133, Cap134 problemlerinin İDOA ve diğer ikili optimizasyon algoritmalar ile karşılaştırması.

Yöntem		**cap131**	**cap132**	**cap133**	**cap134**
İPSO	En İyi	795.291,860	851.495,330	893.076,710	928.941,750
	En Kötü	804.549,640	865.667,160	909.908,700	951.803,250
	Std.Sap.	2.429,540	4.297,070	4.210,930	6.619,050
	İsabet	0	0	0	7
İYAK	En İyi	793.439,560	851.495,330	893.076,710	928.941,750
	En Kötü	794.910,640	851.636,700	895.407,930	928.941,750
	Std.Sap.	1.065,730	213,280	561,340	0.000
	İsabet	6	14	5	30
İASÇA	En İyi	793.439,562	851.495,325	893.076,712	928.941,750
	En Kötü	798.449,038	852.257,975	894.801,163	934.586,975
	Std.Sap.	1.025,786	251,654	501,912	1.016,144
	İsabet	6	23	7	26
İDOA	En İyi	793.439,562	851.495,325	893.076,712	928.941,750
	En Kötü	794.299,850	851,670,125	893.680,197	929.477,563
	Std.Sap.	426,182	43,603	578,599	133,655
	İsabet	15	28	12	28

Çizelge 4.12'de görüldüğü gibi, önerilen algoritma birinci sıradadır ve 12 problemden 9'ünün optimal çözümlerine ulaşmaktadır optimum çözüme ulaşma sayılarının diğer algoritma değerlerinden büyük veya eşit olduğu görülmektedir.

Çizelge 4. 13. KTYP'nin İDOA ve diğer ikili optimizasyon algoritmalar ile karşılaştırması.

Problem	İPSO		İYAK		İASÇA		İDOA	
	Std.Sap.	**İsabet**	**Std.Sap.**	**İsabet**	**Std.Sap.**	**İsabet**	**Std.Sap.**	**İsabet**
cap71	562,230	25	0.000	30	0.000	30	0.000	**30**
cap72	1.324,300	25	0.000	30	0.000	30	0.000	**30**
cap73	702,130	22	0.000	30	0.000	30	0.000	**30**
cap74	2.124,540	0	0.000	30	0.000	30	0.000	**30**
cap101	1.480,720	0	0.000	**30**	380,434	22	292,442	26
cap102	1.015,640	10	0.000	30	0.000	30	0.000	**30**
cap103	1.695,790	0	85.670	**25**	470,951	10	377,019	22
cap104	3.842,640	18	0.000	30	0.000	30	0.000	**30**
cap131	2.429,540	0	1.065,730	6	1.025,786	6	426,182	**15**
cap132	4.297,070	0	213,280	14	251,654	23	43,603	**28**
cap133	4.210,930	0	561,340	5	501.912	7	578,599	**12**
cap134	6.619,050	7	0.000	**30**	1.016,144	26	133,655	28

Elde edilen sonuçlar standart sapma ve isabet değerleri açısından Çizelge 4.13'de görülmektedir. Çizelgede her bir problem için birinci olan yöntemin isabet değerleri kalın yazı tipinde verilmiştir.

İDOA ve diğer 3 farklı yöntem ile 12 KTYP problemleri üzerinde gerçekleştirilen karşılaştırmalar, İDOA yönteminin rekabetçi ve alternatif bir ikili optimizasyon algoritması olduğunu göstermiştir.

BÖLÜM 5

SONUÇLAR VE ÖNERİLER

Bu tez kapsamında ikili optimizasyon yöntemleri ve test problemleri araştırılıp incelenmiştir. İkili optimizasyon yöntemi olarak DOA üzerinde çalışılmıştır. Önerilen yöntemler kullanılarak DOA'sı için 0-1 sonuçlar elde edebilmek için bitsel operatörlerden exclusive-or (xor) fonksiyonu kullanılmıştır. Farklı iterasyon ve popülasyon sayıları kullanılarak bu önerilen ikili yöntemlerin one-max problemleri üzerinde test edilmiştir. Ayrıca, geliştirilen İDOA yöntemi KTYP test seti üzerinde çalıştırılmış elde edilen sonuçlar İPSO, İYAK ve İASÇA algoritma sonuçlarıyla kıyaslanmıştır. İDOA yöntemi ikili optimizasyon problem çözümünde diğer algoritmalarla kıyaslanabilecek derecede bir performans ortaya koymuştur.

Tez kapsamında aşağıda belirtilen akademik çalışma gerçekleştirilmiştir:

- Uluslararası Bildiri: "Binary optimization using duelist optimization algorithm for One Max Problem", 3rd International Conference on Advanced Technologies, Computer Engineering and Science (ICATCES), 2020.

İlerideki çalışmalarda, optimizasyon parametrelerinin performans üzerindeki etkileri, sırt çantası ve diğer ikili optimizasyon problemleri için uygulanarak sonuçların irdelenmesi amaçlanmaktadır. Buna ek olarak farklı metasezgisel algoritmaları ikili yöntemler ve ikili algoritmalara çevirerek performans karşılaştırılması planlanmaktadır.

KAYNAKLAR

1. Haupt, R. L. and Haupt, S. E., "Practical Genetic Algorithms Second Edition", ***John Wiley & Sons***, Hoboken, 18-22, 27-30, 189-190 (2003).

2. Zhu, C., Byrd, R. H., Lu, P., and Nocedal, J., "Algorithm 778: L-BFGS-B: Fortran Subroutines for Large-Scale Bound-Constrained Optimization", ***ACM Transactions On Mathematical Software***, 23 (4): 550–560 (1997).

3. Fletcher, R., "Methods for the solution of optimization problems", ***Computer Physics Communications***, 3 (3): 159–172 (1972).

4. Antoniou, A. and Lu, W. S., "Practical Optimization: Algorithms and Engineering Applications", Practical Optimization: Algorithms and Engineering Applications, ***Springer US***, 1–669 (2007).

5. Winston, W. L., "Operations Research: Applications and Algorithms", ***Brooks\Cole***, Belmont, 380-384 (2004).

6. Lozano, L., Smith, J. C., and Kurz, M. E., "Solving the traveling salesman problem with interdiction and fortification", ***Operations Research Letters***, 210-216 (2017).

7. Karaboga, D. and Akay, B., "A comparative study of Artificial Bee Colony algorithm", ***Applied Mathematics And Computation***, 108-132 (2009).

8. Arora, J. S., "Introduction to Optimum Design 4nd ed.", ***Elsevier Academic Press***, İowa, 17-25 (2004).

9. Yaman, F., "Optimizasyon Problemlerinin Çözümünde Hesaplama Maliyetinin Azaltılması", Doktora Tezi, ***Ankara Üniversitesi Fen Bilimleri Enstitüsü***, Ankara, 12-20 : (2014).

10. Yang, X.-S., "Nature-Inspired Metaheuristic Algorithm", Second Edi. Ed., ***Luniver Press***, United Kingdom, (2010).

11. Djelloul, H., Sabba, S., and Chikhi, S., "Binary bat algorithm for graph coloring problem" ***Second World Conference on Complex Systems (WCCS)***, IEEE, 481-486 (2014).

12. Al-Madi, N., Faris, H., and Mirjalili, S., "Binary multi-verse optimization algorithm for global optimization and discrete problems", ***International Journal Of Machine Learning And Cybernetics***, 10 (12): 3445–3465 (2019).

13. Goemans, M. X. and Williamson, D. P., "Improved Approximation Algorithms for Maximum Cut and Satisflability Problems Using Semidefinite Programming", ***Journal Of The ACM (JACM)***, (1995).

14. Keuchel, J., Schnörr, C., Schellewald, C., and Cremers, D., "Binary partitioning, perceptual grouping, and restoration with semidefinite programming", ***IEEE Transactions On Pattern Analysis And Machine Intelligence***, 25 (11): 1364–1379 (2003).

15. Boykov, Y., Veksler, O., and Zabih, R., "Fast approximate energy minimization via graph cuts", ***IEEE Transactions On Pattern Analysis And Machine Intelligence***, 23(11), 1222-1239 (2001).

16. Jiang, B., Liu, Y. F., and Wen, Z., "Lp-Norm regularization algorithms for optimization over permutation matrices", ***SIAM Journal On Optimization***,26(4): 2284-2313 (2016).

17. Fogel, F., Jenatton, R., Bach, F., and D'aspremont, A., "Convex relaxations for permutation problems", ***Advances in Neural Information Processing Systems***, 1016-1024 (2013).

18. Cour, T. and Shi, J., "Solving Markov Random Fields with Spectral relaxation", ***Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics***, 2: 75-82 (2007).

19. Toshev, A., Shi, J., and Daniilidis, K., "Image matching via saliency region correspondences", ***IEEE Conference on Computer Vision and Pattern Recognition***, 1-8 (2007).

20. Zaslavskiy, M., Bach, F., and Vert, J. P., "A path following algorithm for the graph matching problem", ***IEEE Transactions On Pattern Analysis And Machnine İntelligence,*** 31(10) : 2227-2243 (2009).

21. Shi, J. and Malik, J., "Normalized cuts and image segmentation", ***IEEE Transactions On Pattern Analysis And Machine Intelligence***, 22(8): 888-905 (2000).

22. Joulin, A., Bach, F., and Ponce, J., "Discriminative clustering for image co-segmentation", ***IEEE Computer Society Conference on Computer Vision and Pattern Recognition***, 1943-1950 (2010).

23. Wang, P., Shen, C., Van Den Hengel, A., and Torr, P. H. S., "Large-scale binary quadratic optimization using semidefinite relaxation and applications", ***IEEE Transactions On Pattern Analysis And Machine Intelligence***, 39(3): 470-485 (2017).

24. Ames, B. P. W., "Guaranteed Recovery of Planted Cliques and Dense Subgraphs by Convex Relaxation", ***Journal Of Optimization Theory And Applications***, 167(2): 653-675 (2015).

25. Zhang, Z., Li, T., Ding, C., and Zhang, X., "Binary matrix factorization with applications", ***Seventh IEEE international conference on data mining (ICDM 2007)*** ,391-440 (2007).

26. Ames, B. P. W., "Guaranteed clustering and biclustering via semidefinite

programming", ***Mathematical Programming***, 147(1-2): 429-465 (2013).

27. Yuan, G. and Ghanem, B., "An exact penalty method for binary optimization based on MPEC formulation", ***AAAI***, 2867-2875 (2017).

28. Banitalebi, A., Aziz, M. I. A., and Aziz, Z. A., "A self-adaptive binary differential evolution algorithm for large scale binary optimization problems", ***Information Sciences***, 367–368: 487–511 (2016).

29. Kennedy, J. and Eberhart, R. C., "Discrete binary version of the particle swarm algorithm", ***IEEE International Conference on***, 4104-4108 (1997).

30. Mirjalili, S. and Lewis, A., "S-shaped versus V-shaped transfer functions for binary Particle Swarm Optimization", ***Swarm And Evolutionary Computation***, 9: 1–14 (2013).

31. Rizk-Allah, R. M. and Hassanien, A. E., "New binary bat algorithm for solving 0–1 knapsack problem", ***Complex & Intelligent Systems***, 4: 31-53 (2018).

32. Rashedi, E., Nezamabadi-pour, H., and Saryazdi, S., "GSA: A Gravitational Search Algorithm", ***Information Sciences***, 9(3): 727-745 (2009).

33. Beşkirli, M., Koç, İ., Haklı, H., and Kodaz, H., "A new optimization algorithm for solving wind turbine placement problem: Binary artificial algae algorithm", ***Renewable Energy***, 121: 301-308 (2018).

34. Korkmaz, S., Babalik, A., Mustafa, ·, and Kiran, S., "An artificial algae algorithm for solving binary optimization problems", ***International Journal Of Machine Learning And Cybernetics***, 9 (3): 1233–1247 (2018).

35. Prescilla, K. and Immanuel Selvakumar, A., "Modified Binary Particle Swarm optimization algorithm application to real-time task assignment in heterogeneous multiprocessor", ***Microprocessors And Microsystems***, 37: 583-589 (2013).

36. Babaoglu, I., Findik, O., and Ülker, E., "A comparison of feature selection models utilizing binary particle swarm optimization and genetic algorithm in determining coronary artery disease using support vector machine", ***Expert Systems With Applications***, 37 (4): 3177–3183 (2010).

37. Emary, E., Zawbaa, H. M., and Hassanien, A. E., "Binary grey wolf optimization approaches for feature selection", ***Neurocomputing***, 172: 371-381 (2016).

38. Fan, K., You, W., and Li, Y., "An effective modified binary particle swarm optimization (mBPSO) algorithm for multi-objective resource allocation problem (MORAP)", ***Applied Mathematics And Computation***, (2013).

39. Pal, A. and Maiti, J., "Development of a hybrid methodology for dimensionality reduction in Mahalanobis-Taguchi system using Mahalanobis distance and binary particle swarm optimization", ***Expert Systems With Applications***, 37 (2): 1286–1293 (2010).

40. Baş, E. and Ülker, E., "A binary social spider algorithm for uncapacitated facility location problem", ***Expert Systems With Applications***, 161: 113618 (2020).

41. Ghosh, D., "Neighborhood search heuristics for the uncapacitated facility location problem", ***European Journal of Operational Research***, 150 (1): 150-162 (2003).

42. Zhuang, F. and Galiana, F. D., "Unit commitment by simulated annealing", ***IEEE Transactions On Power Systems***, 5 (1): 311–318 (1990).

43. Aydin, M. E. and Fogarty, T. C., "A distributed evolutionary simulated annealing algorithm for combinatorial optimisation problems", ***Journal Of Heuristics***, 10(3): 269–292 (2004).

44. Ye, L. I., & Yan, Chen, "A genetic algorithm for job-shop scheduling", ***Journal of software,*** 5(3): 269-274 (2010).

45. Khuri, S., Bäck, T., and Heitkotter, J., "The zero/one multiple knapsack problem and genetic algorithms", ***SAC'94: Proceedings of the 1994 ACM sysmposium on Applied computing,*** 188-193 (1994).

46. Aslan, M., Gunduz, M., and Kiran, M. S., "JayaX: Jaya algorithm with xor operator for binary optimization", ***Applied Soft Computing Journal***, 82: 105576 (2019).

47. Kuehn, A. A. and Hamburger, M. J., "A Heuristic Program for Locating Warehouses", ***Management Science***, 9(4): 643-666 (1963).

48. Manne, A. S., "Plant Location Under Economies-of-Scale—Decentralization and Computation", ***Management Science***, 11(2): 213-235 (1964).

49. Shu, J., Teo, C. P., and Shen, Z. J. M., "Stochastic transportation-inventory network design problem", ***Operations Research***, 53 (1): 48-60 (2005).

50. Stollsteimer, J. F., "A Working Model for Plant Numbers and Locations", ***Journal Of Farm Economics***, 45(3): 631-645 (1963).

51. Teo, C. P. and Shu, J., "Warehouse-retailer network design problem", ***Operations Research***, 52: 396-408 (2004).

52. Claveria, O., Monte, E., and Torra, S., "Evolutionary Computation for Macroeconomic Forecasting", ***Computational Economics***, 53 (2): 833–849 (2019).

53. Barcelo, J., Hallefjord, Å., Fernandez, E., and Jörnsten, K., "Lagrangean relaxation and constraint generation procedures for capacitated plant location problems with single sourcing", ***OR Spektrum***, 12 (2): 79–88 (1990).

54. Holmberg, K., "Exact solution methods for uncapacitated location problems with convex transportation costs", ***European Journal Of Operational Research***, 114 (1): 127–140 (1999).

55. Akinc, U. and Khumawala, B. M., "Efficient Branch And Bound Algorithm For The Capacitated Warehouse Location Problem", ***Management Science***, 23(6): 585-

594 (1977).

56. Bilde, O. and Krarup, J., "Sharp Lower Bounds and Efficient Algorithms for the Simple Plant Location Problem", ***Annals Of Discrete Mathematics***, 79-97 (1977).

57. Van Roy, T. J., "Cross Decomposition Algorithm For Capacitated Facility Location", ***Operations Research***, 34 (1): 145-163 (1986).

58. Shmoys, D. B., Tardos, E., and Aardal, K., "Approximation algorithms for facility location problems", ***Proceedings of the twenty-ninth annual ACM symposium on Theory of computing***, 265-274 (1997).

59. Monabbati, E., "An application of a Lagrangian-type relaxation for the uncapacitated facility location problem", ***Japan Journal Of Industrial And Applied Mathematics***, 31(3): 483-499 (2014).

60. Erlenkotter, D., "A dual-based procedure for uncapacitated facility location.", ***Operations Research***, 26(6): 992-1009 (1978).

61. Körkel, M., "On the exact solution of large-scale simple plant location problems", ***European Journal Of Operational Research***, (1989).

62. Al-Sultan, K. S. and Al-Fawzan, M. A., "A tabu search approach to the uncapacitated facility location problem", ***Annals Of Operations Research***, 86: 91-103 (1999).

63. Sun, M., "Solving the uncapacitated facility location problem using tabu search", ***Computers And Operations Research***, 33(9): 2563-2589 (2006).

64. Ardjmand, E., Park, N., Weckman, G., and Amin-Naseri, M. R., "The discrete Unconscious search and its application to uncapacitated facility location problem", ***Computers And Industrial Engineering***, (2014).

65. Guner, A. R. and Sevkli, M., "A Discrete Particle Swarm Optimization Algorithm for Uncapacitated Facility Location Problem", ***Journal Of Artificial Evolution And Applications***, 10 (2008).

66. Watanabe, Y., Takaya, M., and Yamamura, A., "Fitness function in ABC algorithm for uncapacitated facility location problem", ***Information and Communication Technology-EurAsia Conference. Springer,*** 129-138 (2015).

67. Tunçbilek, N., Tasgetiren, F., and Esnaf, S., "Artificial Bee Colony Optimization Algorithm for Uncapacitated Facility Location Problems", ***Journal of Economic & Social Research***,, 14(1): 1-24 (2012).

68. Kratica, J., Tošic, D., Filipović, V., and Ljubić, I., "Solving the simple plant location problem by genetic algorithm", ***RAIRO - Operations Research***, 35 (1): 127-142 (2001).

69. Kole, A., Chakrabarti, P., and Bhattacharyya, S., "An Ant Colony Optimization Algorithm for Uncapacitated Facility Location Problem", ***Artificial Intelligence***

And Applications, 2014 (1): 55–61 (2014).

70. Tsuya, K., Takaya, M., and Yamamura, A., "Application of the firefly algorithm to the uncapacitated facility location problem", ***Journal of Intelligent & Fuzzy Systems,*** 32(4), 3201-3208 (2017).

71. Atta, S., Mahapatra, P. R. S., and Mukhopadhyay, A., "Solving uncapacitated facility location problem using monkey algorithm", ***Intelligent Engineering Informatics. Springer***, Singapore,.71-78. (2018).

72. Hakli, H. and Ortacay, Z., "An improved scatter search algorithm for the uncapacitated facility location problem", ***Computers And Industrial Engineering***, 135: 127-140 (2019).

73. Biyanto, T. R., Fibrianto, H. Y., Nugroho, G., Hatta, A. M., Listijorini, E., Budiati, T., and Huda, H., "Duelist algorithm: An algorithm inspired by how duelist improve their capabilities in a duel", Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), ***Springer Verlag***, 39–47 (2016).

74. Biyanto, T. R., Alfarisi, M. S., Afdanny, N., Setiawan, H., and Hasan, A., "Simultaneous optimization of tuning PID cascade control system using Duelist Algorithms", ***IOP Conference Series: Materials Science and Engineering***, 458 (2018).

75. Da Costa, S., Sasanti, G. F., Musyafa, A., Soeprijanto, A., and Biyanto, T. R., "Duelist algorithm for optimisation of safety instrumented system at distillation column based on RAMS + C", ***Safety And Reliability***, 177-193 (2017).

76. Biyanto, T. R., Matradji, Syamsi, M. N., Fibrianto, H. Y., Afdanny, N., Rahman, A. H., Gunawan, K. S., Pratama, J. A. D., Malwindasari, A., Abdillah, A. I., Bethiana, T. N., and Putra, Y. A., "Optimization of energy efficiency and conservation in green building design using duelist, Killer-Whale and Rain-Water Algorithms", ***IOP Conference Series: Materials Science and Engineering***, 267 (2017).

77. Kiran, M. S. and Gündüz, M., "XOR-based artificial bee colony algorithm for binary optimization", ***Turkish Journal Of Electrical Engineering And Computer Sciences***, 2307-2328 (2013).

78. Goëffon, A. and Lardeux, F., "Optimal one-max strategy with dynamic island models" ***Proceedings - International Conference on Tools with Artificial Intelligence, ICTAI***,485-488 (2011).

79. Khair, U., Lestari, Y. D., Perdana, A., Hidayat, D., and Budiman, A., "Genetic algorithm modification analysis of mutation operators in max one problem", T***hird International Conference on Informatics and Computing (ICIC)***, 1-6 (2018).

80. John, H., "Holland, Adaptation in natural and artificial systems", ***Ann Arbor MI University Of Michigan Press***, 1-200 (1992).

81. Baykasoğlu, A., Ozsoydan, F. B., and Senol, M. E., "Weighted superposition attraction algorithm for binary optimization problems", ***Operational Research***, 20:2555–2581 (2020).

82. Sevkli, M. and Guner, A. R., "A continuous particle swarm optimization algorithm for uncapacitated facility location problem", ***International Workshop on Ant Colony Optimization and Swarm Intelligence*,** 316-323 (2006).

83. Kiran, M. S., "The continuous artificial bee colony algorithm for binary optimization", ***Applied Soft Computing Journal***, 33: 15-23 (2015).

ÖZGEÇMİŞ

Hacer DÖNMEZ 1994 yılında Malatya'da doğdu; ilk ve orta öğrenimini aynı şehirde tamamladı. Gümüşhane Üniversitesi Mühendislik ve Doğa Bilimleri Fakültesi Matematik Mühendisliği Bölümü'nde 2017 yılında mezun oldu. Yüksek lisans eğitimini Karabük Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı'nda 2020 yılında tamamladı.

ADRES BİLGİLERİ

Adres : Merkez Mah. Karabük Cad.
No : 24/11
Safranbolu / KARABÜK
Tel : (544) 201 3095
E-posta : hacerdonmezz@outlook.com