T.C.
BİLECİK ŞEYH EDEBALİ ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

KAOTİK MUTASYON STRATEJİLERİ TABANLI TEK ADAY OPTİMİZASYON ALGORİTMALARININ MÜHENDİSLİK TASARIM PROBLEMLERİNDE UYGULANMASI

YÜKSEK LİSANS TEZİ

HALİL İBRAHİM EMEK

TEZ DANIŞMANI
PROF. DR. UĞUR YÜZGEÇ

BİLECİK, 2024
10590787

T.C.
BİLECİK ŞEYH EDEBALİ ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

KAOTİK MUTASYON STRATEJİLERİ TABANLI TEK ADAY OPTİMİZASYON ALGORİTMALARININ MÜHENDİSLİK TASARIM PROBLEMLERİNDE UYGULANMASI

YÜKSEK LİSANS TEZİ

HALİL İBRAHİM EMEK

TEZ DANIŞMANI
PROF. DR. UĞUR YÜZGEÇ

BİLECİK, 2024
10590787

BEYAN

Kaotik Mutasyon Stratejileri Tabanlı Tek Aday Optimizasyon Algoritmalarının Mühendislik Tasarım Problemlerinde Uygulanması adlı yüksek lisans tezinin hazırlık ve yazımı sırasında bilimsel araştırma ve etik kurallarına uyduğumu, başkalarının eserlerinden yararlandığım bölümlerde bilimsel kurallara uygun olarak atıfta bulunduğumu, kullandığım verilerde herhangi bir tahrifat yapmadığımı, tezin herhangi bir kısmının Bilecik Şeyh Edebali Üniversitesi veya başka bir üniversitede başka bir tez çalışması olarak sunulmadığını, aksinin tespit edileceği muhtemel durumlarda doğabilecek her türlü hukuki sorumluluğu kabul ettiğimi ve vermiş olduğum bilgilerin doğru olduğunu beyan ederim.

<table>
<tr><td colspan="4">Bu çalışmanın,
Bilimsel Araştırma Projeleri (BAP), TÜBİTAK veya benzeri kuruluşlarca desteklenmesi durumunda; projenin ve destekleyen kurumun adı proje numarası ile birlikte, ETİK KURUL onayı alınması durumunda ise ETİK KURUL tarih karar ve sayı bilgilerinin beyan edilmesi gerekmektedir.</td></tr>
<tr><td>DESTEK ALINMIŞTIR</td><td></td><td>DESTEK ALINMAMIŞTIR</td><td>X</td></tr>
<tr><td colspan="4">Destek alındı ise;</td></tr>
<tr><td colspan="4">Destekleyen kurum;</td></tr>
<tr><td colspan="2">Desteğin Türü</td><td colspan="2">Proje Numarası</td></tr>
<tr><td colspan="2">1- BAP (Bilimsel Araştırma Projesi)</td><td colspan="2" rowspan="3"></td></tr>
<tr><td colspan="2">2- TÜBİTAK</td></tr>
<tr><td colspan="2">Diğer;……</td></tr>
<tr><td colspan="4">ETİK KURUL onayı var ise;</td></tr>
<tr><td colspan="2">ETİK KURUL karar tarih/sayı:</td><td colspan="2">……/……..</td></tr>
</table>

Halil İbrahim EMEK

Tarih

İmza

ÖN SÖZ

Bu tez çalışmasının yazılmasında, çalışmamı sahiplenerek takip eden danışmanım Sayın Prof. Dr. Uğur YÜZGEÇ' e değerli katkı ve emekleri için teşekkürlerimi ve saygılarımı sunarım.

Savunma sınavı sırasında değerli jüri üyeleri Sayın Prof. Dr. Cihan KARAKUZU'ya ve Sayın Doç. Dr. Serhat YILMAZ'a çalışmamın son haline gelmesindeki değerli katkıları adına teşekkürlerimi ve saygılarımı sunarım.

Son olarak tüm eğitim hayatım boyunca benden maddi ve manevi desteklerini esirgemeyen her zaman yanımda olan sevgili aileme, bu günlere ulaşmamdaki emekleri adına değerli eşim Ferzan KALAYCI EMEK'e ve değerli oğlum Oğuz Kağan EMEK'e teşekkürlerimi bir borç bilirim.

Halil İbrahim EMEK

2024

ÖZET

KAOTİK MUTASYON STRATEJİLERİ TABANLI TEK ADAY OPTİMİZASYON ALGORİTMALARININ MÜHENDİSLİK TASARIM PROBLEMLERİNDE UYGULANMASI

Sezgisel arama algoritmaları, karmaşık optimizasyon sorunlarını ele almak için yaygın olarak kullanılan bir metodolojidir. Tipik olarak bu algoritmalar, arama uzayı boyunca bir dizi potansiyel çözümde gezinerek optimum çözüme yakınsama hedefiyle sürü veya popülasyon tabanlı yaklaşımlara dayanmaktadır. Bununla birlikte, bu yaklaşımların çoğunluğunun önemli sayıda parametre gerekliliği, yüksek hesaplama maliyeti, erken yakınsama ve global çözümü garanti edememe gibi eksiklikleri vardır.

Bu tez çalışmasında, sürü tabanlı metodolojilerin aksine, optimizasyon süreci boyunca tek bir aday çözüm kullanan yeni bir sezgisel arama stratejisi olan Tek Aday Optimizasyon (SCO) algoritması ele alınmaktadır. SCO, aday çözümün konumunun güncellenmesi için iki aşamalı bir metodoloji kullanır. İlk aşamada aday, kendi içsel bilgisinden yararlanarak arama uzayının çeşitli bölgelerini araştırır. İkinci aşamada aday, yerel arama yöntemlerini kullanarak kendi bölgesi içinde en uygun çözümü arar. Bu şekilde, SCO algoritması keşif ve sömürü yetenekleri arasında bir denge kurar.

SCO algoritması, basitlik, sınırlı sayıda parametre, düşük hesaplama maliyeti ve yüksek başarım gibi çeşitli avantajlar sunar. Bununla birlikte, SCO algoritmasının potansiyel eksiklikleri de yok değildir. Bunlar arasında tek bir aday çözümün sınırlı keşif kabiliyeti, yerel minimum noktalarına yakalanma riski ve optimal olmayan bölgelerde takılıp kalma olasılığı yer almaktadır. SCO'nun keşif mekanizması aday çözümü hızla sıfıra götürebilirken, bu durum sıfır olmayan çözümlere sahip problemler için SCO'nun çözüme yakınsamasını engelleyebilir. Bu sınırlamaları ele almak ve SCO algoritmasının başarımını iyileştirmek için çeşitli yöntemler önerilebilir.

Bu tez çalışmasında, SCO algoritmasının etkinliğini artırmak ve bahsedilen handikaplarını yenebilmek için Chaucy, Gauss ve Levy gibi kaotik fonksiyonlara dayalı yeni bir mutasyon tekniği önerilmektedir (Kale ve Yüzgeç, 2022:3). Bu mutasyon operatörü, aday çözümün konumunu güncellerken arama uzayının farklı bölgelerini dolaşmasını sağlayarak keşif yeteneğini geliştirir (Parmaksız vd. 2023:5). Önerilen kaotik mutasyon stratejisi tabanlı tek aday optimizasyon algoritması (CSCO: Chaotic mutation based Single Candidate Optimizer), literatürden elde edilen 23 farklı kıyaslama fonksiyonu için orijinal SCO algoritması ile

karşılaştırmalı olarak değerlendirilmiştir. Bunun yanısıra, CSCO algoritmasının başarımı, literatürde tanımlanan çeşitli mühendislik tasarım problemlerinin incelenmesi yoluyla değerlendirilmiştir. Bunlar arasında Kaynaklı Kiriş Tasarımı (WBD), Sıkıştırma Yayı Tasarımı (CSD) ve Basınçlı Kap Tasarımı (PVD) yer almaktadır. Karşılaştırmalarda orijinal SCO algoritmasının yanı sıra literatürde sıklıkla kullanılan popüler sezgisel algoritmalar da kullanılmıştır.

Sonuçlar, farklı mutasyon tekniklerinin kullanıldığı CSCO'nun başarımında kayda değer bir iyileşme olduğunu göstermektedir. Öncelikle 23 adet kıyaslama fonksiyonunun karşılaştırılmasında, CSCO algoritması en iyi metrik değerlerine göre orjinal SCO algoritmasından %73,91'lik daha iyi bir başarım göstermiştir. Aynı şekilde, en kötü metrik değerlerin karşılaştırılmasında CSCO, kıyaslama fonksiyonlarında %56,52'lik bir başarı oranı ile dikkat çekmektedir. Ayrıca, CSCO 13 fonksiyonda SCO'dan %56,52 daha etkili bir medyan metrik başarımı ve 10 fonksiyonda %43,48 daha etkili bir ortalama metrik başarımı sergilemektedir. Ayrıca, SCO'ya kıyasla %69,57 daha düşük standart sapma değeriyle, CSCO'nun test fonksiyonları arasında daha düzgün ve istikrarlı bir davranış sergilemesi de dikkat çekicidir. Bu gözlemler, CSCO'nun test fonksiyonlarının optimizasyonunda SCO'ya kıyasla daha üstün bir yetkinliğe sahip olduğunu doğrulamaktadır.

Bunun yanı sıra SCO ve CSCO algoritmalarının 10 mühendislik tasarım problemleri yakınsama eğrileri analiz edildiğinde 10 problemden 9'unda CSCO algoritmasının daha az iterasyonda, yerel minimumlara takılmadan çözüm noktasına yakınsadığı görülmektedir. Mühendislik tasarım problemlerinde CSCO algoritması %90 başarı oranıyla yapılan geliştirmelerin verimliliğini ortaya koymaktadır.

Anahtar Kelimeler: Kaotik, Mutasyon, Mühendislik Tasarım Problemleri, Sezgisel, Tek Adaylı Optimizasyon.

ABSTRACT

APPLICATION OF CHAOTIC MUTATION STRATEGY BASED SINGLE CANDIDATE OPTIMIZATION ALGORITHM TO ENGINEERING DESIGN PROBLEMS

Heuristic search algorithms are a widely used methodology for addressing complex optimization problems. Typically, these algorithms are based on swarm or population-based approaches with the goal of converging to the optimal solution by navigating through a set of potential solutions across the search space. However, most of these approaches have shortcomings such as the requirement of a significant number of parameters, high computational cost, premature convergence and the inability to guarantee a global solution.

In this thesis, we consider a new heuristic search strategy, the Single Candidate Optimization (SCO) algorithm, which, in contrast to swarm-based methodologies, uses a single candidate solution throughout the optimization process. SCO uses a two-stage methodology for updating the position of the candidate solution. In the first stage, the candidate searches various regions of the search space utilizing its own internal knowledge. In the second stage, the candidate searches for the optimal solution within its region using local search methods. In this way, the SCO algorithm strikes a balance between exploration and exploitation capabilities.

The SCO algorithm offers several advantages such as simplicity, limited number of parameters, low computational cost and high performance. However, the SCO algorithm is not without potential shortcomings. These include the limited exploration capability of a single candidate solution, the risk of getting caught in local minima, and the possibility of getting stuck in suboptimal regions. While SCO's exploration mechanism can quickly lead a candidate solution to zero, this can prevent SCO from converging to a solution for problems with non-zero solutions. Various methods can be proposed to address these limitations and improve the performance of the SCO algorithm.

In this thesis, a new mutation technique based on chaotic functions such as Chaucy, Gauss and Levy is proposed to improve the efficiency of the SCO algorithm and overcome the mentioned handicaps. This mutation operator improves the exploration capability by making the candidate solution traverse different regions of the search space while updating its position. The proposed Chaotic mutation based Single Candidate Optimizer (CSCO) algorithm is evaluated against the original SCO algorithm for 23 different benchmark functions obtained from the literature. In addition, the performance of the CSCO algorithm is evaluated by analyzing various engineering

design problems identified in the literature. These include Welded Beam Design (WBD), Compression Spring Design (CSD) and Pressure Vessel Design (PVD). In addition to the original SCO algorithm, popular heuristic algorithms frequently used in the literature were also used in the comparisons.

The results show a significant improvement in the performance of CSCO using different mutation techniques. First of all, in the comparison of 23 benchmark functions, the CSCO algorithm outperformed the original SCO algorithm by 73.91% for the best metric values. Likewise, when comparing the worst metric values, CSCO outperforms the original SCO algorithm by 56.52% for the benchmark functions. Furthermore, CSCO shows a 56.52% more effective median metric performance than SCO in 13 functions and a 43.48% more effective mean metric performance in 10 functions. It is also noteworthy that CSCO exhibits a smoother and more stable behavior across test functions, with a standard deviation 69.57% lower than SCO. These observations confirm that CSCO has a superior capability in optimizing test functions compared to SCO.

In addition, when the convergence curves of SCO and CSCO algorithms for 10 engineering design problems are analyzed, it is seen that in 9 out of 10 problems, the CSCO algorithm converges to the solution point in fewer iterations without getting stuck in local minima. In engineering design problems, the CSCO algorithm demonstrates the efficiency of the improvements made with a 90% success rate.

Keywords: Chaotic, Engineering Design Problems, Heuristic, Mutation, Single Candidate Optimization.

İÇİNDEKİLER

Sayfa

TABLOLAR LİSTESİ

Sayfa

ŞEKİLLER LİSTESİ

Sayfa

ALGORİTMALAR LİSTESİ

Sayfa

KISALTMALAR VE SİMGELER LİSTESİ

ABC: Yapay Arı Kolonisi (Artificial Bee Colony)

ACO: Karınca Kolonisi Optimizasyonu (Ant Colony Optimization)

BA: Yarasa Algoritması (Bat Algorithm)

BBO: Biyocoğrafya Tabanlı Optimizasyon (Biogeography-Based Optimization)

CS: Guguk Kuşu Algoritması (Cuckoo Search Algorithm)

CSCO: Kaotik Mutasyon Stratejisi Tabanlı Tek Aday Optimizasyon Algoritması (Chaotic Mutation Strategy Based Single Candidate Optimization Algorithm)

DE: Farksal Gelişim (Differential Evolution)

EM: Elektromanyetizma Benzeri Arama (Electromagnetism-Like Search)

FA: Ateşböceği Algoritması (Firefly Algorithm)

GA: Genetik Algoritma (Genetic Algorithm)

GSA: Yerçekimsel Arama Algoritması (Gravitational Search Algorithm)

GWO: Gri Kurt Optimizasyonu (Grey Wolf Optimizer)

PSO: Parçacık Sürü Optimizasyonu (Particle Swarm Optimization)

SA: Benzetimli Tavlama (Simulated Annealing)

SCA: Sinüs Kosinüs Algoritması (Sine Cosine Algorithm)

SCO: Tek Aday Optimizasyonu (Single Candidate Optimizer)

SS: Standart sapma

SSA: Salp Sürüsü Algoritması (Salp Swarm Algorithm)

1.GİRİŞ

Son on yılda bilimsel ve teknolojik ilerlemenin hızlanması, giderek daha karmaşık hale gelen gerçek dünya problemlerinin ortaya çıkmasına yol açarak hızlı ve verimli optimizasyon algoritmalarının geliştirilmesi ihtiyacını ortaya çıkarmıştır. Optimizasyon sürecinin ilk aşaması, maksimize veya minimize edilebilecek bir amaç fonksiyonunun formüle edilmesidir. Optimizasyon problemi formüle edildikten sonra, en uygun çözümle sonuçlanacak en uygun değişkenleri belirlemek için bir optimizasyon algoritması gereklidir (Pinciroli vd., 2023:4).

Genel olarak, optimizasyon problemleri deterministik veya stokastik yöntemlerle çözülebilir. Verilen problemlerin optimum çözümlerini belirlemeye yönelik iki önemli deterministik yöntem, gradyan bilgisi ile doğrusal ve doğrusal olmayan programlamadır. Ancak, bu geleneksel deterministik yöntemler yerel optimumlara yakınsayabilir. Geleneksel metodolojilerin eksikliklerini gidermek için, doğası gereği stokastik olan sezgisel algoritmalar, karmaşık gerçek dünya optimizasyon problemlerini ele almak için kullanılabilir. Sezgisel algoritmalar, kablosuz iletişim, yapay zekâ, robotik, lojistik, finans, sağlık, tarım, eğitim ve ulaşım da dahil olmak üzere çeşitli alanlardaki farklı optimizasyon problemlerini çözmek için adapte edilmişlerdir (Li ve Grossmann, 2021:3).

Sezgisel algoritmaların temel özellikleri basitlikleri, esneklikleri, yerel optimumları atlatma kapasiteleri ve türevsiz mekanizmalarıdır. Sezgisel algoritmaların arama süreci iki farklı aşamaya ayrılır: keşif ve sömürü. Keşif aşaması, potansiyel olarak en uygun çözümle sonuçlanabilecek umut verici bölgelerin belirlenmesi amacıyla arama uzayının kapsamlı bir şekilde incelenmesini kapsar. Yetersiz keşif, yerel optimumların birikmesine neden olabilir. Sömürü aşaması, keşif aşamasında belirlenen umut verici bölgelerin çevresinde arama yapmakla ilgilidir. Bu aşamada etkin bir uygulamanın sağlanamaması, çözüm doğruluğunda kayda değer bir azalmaya neden olabilir. Sezgisel algoritmaların karşılaştığı en önemli zorluklardan biri, optimum çözüme ulaşma garantisinin olmamasıdır. Bu algoritmalar, genellikle en iyiye yakın çözümler sunar, ancak her zaman en iyi çözümü bulacaklarını garanti edemezler (Chen vd., 2009:680).

Sezgisel algoritmalar, karmaşık problemlerin çözümünde kritik öneme sahiptir ve optimum sonuçları kolaylaştıran verimli ve esnek yaklaşımlar sunar. Bu algoritmalar doğal süreçlerden veya sosyal etkileşimlerden esinlenmekte ve bir amaç fonksiyonunu optimize etmek ve optimal bir çözüm aramak için tasarlanmaktadır. Sezgisel algoritmaların uygulamaları

mühendislik, işletme, yapay zekâ ve bilgisayar bilimi gibi alanları kapsayacak şekilde çeşitlilik göstermektedir. Sezgisel algoritmalar evrimsel, sürü dinamikleri ve fiziksel ilkelere dayalı olmak üzere üç ana kategoriye ayrılabilir. Evrimsel algoritmalar Darwin'in doğal seçilim teorilerine dayanır ve Genetik Algoritma (GA: Genetic Algorithm) (Mirjalili ve Mirjalili, 2019:43), Farksal Gelişim (DE: Differential Evolution) (Price, 2013:188) ve benzeri yöntemleri içerir. Sürü tabanlı algoritmalar, hayvan sürülerinin kolektif eylemlerinden ilham alır ve Parçacık Sürü Optimizasyonu (PSO: Particle Swarm Optimization) (Wang vd., 2018:1), Yapay Arı Kolonisi (ABC: Artificial Bee Colony) (Karaboğa vd., 2014:22), Guguk Kuşu Arama Algoritması (CS: Cuckoo Search Algorithm) (Song vd., 2020:2), Yarasa Algoritması (BA: Bat Algorithm) (Yang ve Slowik, 2013:141), Karınca Kolonisi Optimizasyonu (ACO: Ant Colony Optimization) (Dorigo ve Stützle, 2019:311), Ateşböceği Algoritması (FA: Firefly Algorithm) (Yang, 2010a:3), Salp Sürüsü Algoritması (SSA: Salp Swarm Algorithm) (Faris vd., 2020:188), Gri Kurt Optimizasyonu (GWO: Grey Wolf Optimizer) (Mirjalili vd., 2014:48), algoritmalarını içerir (Kılıç vd. 2020:3803). Fiziksel tabanlı algoritmalar, evrensel fizik yasalarını modelleyerek çalışır ve Benzetimli Tavlama (SA: Simulated Annealing) (Delahaye vd., 2019:5), Elektromanyetizma Benzeri Arama (EM: Electromagnetism-Like Search) (Rocha ve Fernandes, 2009:1934), Yerçekimsel Arama Algoritması (GSA: Gravitational Search Algorithm) (Rashedi vd., 2018:142) gibi öne çıkan örnekleri içerir.

Tek Aday Optimizasyonu (SCO) algoritması, geleneksel sürü tabanlı sezgisel arama algoritmalarına alternatif olarak yeni bir strateji önermektedir (Shami vd., 2022:5). Sürü veya popülasyon tabanlı sezgisel algoritmaların aksine SCO, optimizasyon sürecinin tamamı boyunca tek bir aday çözüme odaklanarak daha üstün çözümler elde etmek için tasarlanmıştır. Optimizasyon süreci, her biri aday çözümün konumunu güncellemek için farklı bir metodoloji kullanan iki farklı aşamadan oluşur. Tek çözüm odaklı algoritmalar ve iki aşamalı yaklaşımlar bağımsız sezgisel optimizasyon teknikleri olarak geliştirilmiş olsa da SCO bu iki kavramı birleşik ve sağlam bir algoritmaya entegre etmektedir. Algoritmanın dikkate değer bir yönü, aday çözümün konumunu güncellemek için algoritmanın mevcut durumundan türetilen kendine özgü bir denklem kümesini kullanmasıdır. Bu bileşenlerin entegrasyonu SCO'nun optimizasyon problemlerine yenilikçi ve verimli bir çözüm sunmasını sağlar.

Tek Aday Optimizasyonu (SCO) algoritması belirli avantajlar sunarken, potansiyel sınırlamaları da dikkate alınmalıdır. SCO algoritmasının doğasında bulunan tek bir aday çözüme özel odaklanma, keşif kapasitesi açısından sınırlayıcı olabilir. Birden fazla aday çözümün sağladığı çeşitliliğin yokluğunda, SCO çözüm uzayını verimli bir şekilde taramakta

zorluklarla karşılaşabilir ve potansiyel olarak yerel optimumlarda sıkışıp kalabilir. Ayrıca, arama uzayının optimal olmayan bölgelerinde takılabilir ve küresel optimumu tespit edemeyebilir. SCO'nun etkinliği, ilk aday çözümün başlangıçtaki konumuna bağlıdır. Ayrıca, keşif aşamasında uygulanan SCO mekanizması, aday çözümün hızla sıfır konumuna ulaşmasına neden olabilir ve bu da sıfır olmayan optimum çözüm noktalarına sahip problemlerde algoritmanın yakınsama sürecini engelleyebilir.

SCO algoritmasının sınırlamalarından biri olan yerel optimumdan kaçınmak ve keşif aşamasında SCO algoritmasının iç kısıtlamalarından etkilenmemek için geliştirilen CSCO algoritmasında kaos teorisinden yararlanılmıştır. Kaos teorisi yaklaşımı SCO algoritması ile ilişkili eksiklikleri gidermek için tercih edilmiştir. Bu eksiklikler giderildikten sonra, geliştirilen CSCO algoritmasının başarımı SCO algoritmasına göre belirgin bir şekilde üstün olacaktır.

SCO algoritmasının uygulanmasında, başlangıçta sınır kısıtlamaları, boyut, fonksiyon, maksimum iterasyon sayısı, başarısız yakınsama sayısını kontrol eden m değişkeni, keşif aşaması iterasyon sayısını belirleyen alpha değişkeni, uygunluk durumunu kontrol eden FitImp değişkeni ve sabit b değeri belirlenir. Belirlenen sınır kısıtlamaları içerisinde rastgele bir aday çözüm seçilir ve bu çözüm "en iyi çözüm" olarak adlandırılan değişkene atanır. Başarısız uygunluk deneme sayacı ise sürece dahil edilir.

Maksimum iterasyon sayısına kadar, ağırlık değeri (w) hesaplanır ve uygunluk iyileşmesi gözlemlenmediğinde uygunluk deneme sayacı artırılır. Alpha değişkeni sayısı kadar keşif aşaması, Denklem 2.1'de verilen formül ile rastgele değerler üretilerek devam ettirilir. Başarısız uygunluk deneme sayacının m değişkeni ile belirlenen değere ulaşması ve keşif aşaması iterasyon sayısının aşılması durumunda, sömürü aşamasına geçilir ve Denklem 2.4'te belirtilen formül kullanılarak yakınsama sağlanır.

Her aday çözüm için sınır kısıtlamaları kontrol edilir ve eğer bu kısıtlamalar aşılmış ise, sınır kısıtlamalarına uyan son aday çözüme geri dönülür. Her aday çözümün uygunluk durumu değerlendirildikten sonra, uygunluk sağlanıyorsa FitImp değişkeni artırılır ve yeni bir aday çözüm belirlenir.

Doğrusal olmayan dinamik sistemlerin incelenmesi, sistemlerin başlangıç koşullarındaki küçük değişikliklere karşı aşırı hassasiyeti göz önüne alındığında, kaos teorisi ile ayrılmaz bir şekilde bağlantılıdır. Liu' nun (Liu vd., 2005:1) gözlemlediği gibi, bu tür sistemler genellikle deterministik yapılarına rağmen öngörülemeyen sonuçlara yol açan kararsız periyodik hareketler sergiler. Optimizasyon algoritmalarının ilerlemesi ve kaotik

sistemlerin kapsamlı bir şekilde incelenmesi, doğrusal olmayan olayların daha derinlemesine anlaşılmasını kolaylaştırmış ve bu alandaki araştırmalara olan ilgi artmıştır. Kaotik sistemlerin temel bir özelliği, parametrelerdeki veya başlangıç durumlarındaki küçük değişikliklerin sistemlerin uzun vadeli yörüngeleri üzerinde önemli ve derin etkilere sahip olabilmesidir (Yüzgeç ve Eser 2018:12). Bu hassasiyet, kaotik sistemlerin analizini ve kontrolünü çok önemli bir çaba haline getirmekte ve doğrusal olmayan disiplinlerdeki uygulamaların çeşitlenmesine katkıda bulunmaktadır.

Bu çalışmada, kaotik fonksiyonlara dayalı olarak geliştirilen Kaotik Mutasyon Stratejisi Tabanlı Tek Aday Optimizasyon (CSCO) algoritmasının etkinliği hem orijinal SCO algoritması hem de mevcut popüler sezgisel algoritmalar ile karşılaştırmalı olarak değerlendirilmiştir. Bu, Tek Adaylı Optimizasyon (SCO) algoritmasının bilinen kısıtlamalarının üstesinden gelmek ve arama başarımını iyileştirmek için yapılmıştır. Literatürde bulunan kıyaslama fonksiyonları ile mühendislik tasarım problemlerinin karşılaştırmalı analizi, CSCO algoritmasının orijinal SCO algoritmasından daha iyi başarım gösterdiğini ortaya koymaktadır. Ayrıca, CSCO algoritması karşılaştırmada diğer popüler sezgisel algoritmalarla rekabet edebilir bir başarım göstermiştir.

2.METODOLOJİ

Bu bölümde, optimizasyon kavramının tanımı, optimizasyon problemleri ve bu proble mlerin sınıflandırması ayrıntılı olarak ele alınmıştır. Bunun yanı sıra, sezgisel algoritmalar ile bu çalışmada kullanılan SCO algoritması incelenmiş, ayrıca geliştirilen CSCO algoritması hak kında kapsamlı bilgiler sunulmuştur. Geliştirilen algoritmanın performansının değerlendirilme sinde kullanılan kıyaslama fonksiyonları ve mühendislik tasarım problemleri de detaylandırıl mıştır.

2.1. Optimizasyon

Optimizasyon, belirli koşullar altında belirli bir problem için en uygun çözümü veya çözümleri arama ve belirleme süreci olarak tanımlanmaktadır. Modern çağda, optimizasyonun birçok alanda bir gereklilik olduğu giderek daha açık hale gelmektedir. Bunlar arasında harcamaların azaltılması ve kârlılığın artırılması, zaman yönetiminin optimizasyonu, enerji verimliliğinin artırılması, seri üretilen ürünlerin üretiminin artırılması ve elektronik olarak çalıştırılan cihazların boyutlarının küçültülmesi yer almaktadır. Bu nedenle optimizasyon algoritmaları üzerine araştırmalar uzun yıllar önce başlamış ve günümüzde daha da önem kazanmaya devam etmektedir (Zhang vd., 2022:7).

Belirli bir probleme en uygun çözümü bulmayı amaçlayan bir optimizasyon sürecinde, öncelikle karar değişkenlerinin (karar parametreleri veya tasarım parametreleri olarak da bilinir) tanımlanması gerekir. Optimizasyon algoritmaları bağlamında karar değişkenleri, optimizasyon algoritmasının uygulanması yoluyla değeri belirlenmeye çalışılan parametreler olarak tanımlanmaktadır. Daha sonra, yukarıda bahsedilen karar değişkenleri kullanılarak amaç fonksiyonu tanımlanmalıdır. Bu fonksiyon, alternatif çözümlerin karşılaştırılması, çözüm kalitesinin veya uygunluğunun ölçülmesi ve optimizasyon algoritmasının uzayda yönlendirilmesi gibi çeşitli amaçlara hizmet eder (Varol Altay, 2022:70).

Optimizasyon problemlerinde, amaç fonksiyonu eldeki problemin türüne uygun olarak seçilir ve iki kategoride sınıflandırılabilir: minimize edilecek bir maliyet fonksiyonu veya maksimize edilecek bir kâr fonksiyonu. Amaç fonksiyonunun bir maliyet fonksiyonu olması durumunda, alternatif çözümler arasından en düşük sonucu veren çözüm en uygun çözüm olarak kabul edilir. Tersine, eğer amaç fonksiyonu bir kâr fonksiyonu ise, alternatif çözümlerden en yüksek sonucu üreten çözüm en uygun çözüm olarak kabul edilir (Yang, 2010b:15).

Ek olarak, optimizasyon problemiyle ilişkili kısıtlama fonksiyonları tanımlanmalıdır. Optimizasyon problemlerinde kısıtlama fonksiyonları, çözümün belirli koşulları sağlamasını

gerektiren matematiksel ifadeler olarak tanımlanır. Bu fonksiyonlar, optimizasyon probleminin çözüm alanını sınırlar ve genellikle eşitsizlik veya eşitlik şeklinde ifade edilirler (Garip vd., 2021:78).

2.1.1. Optimizasyon problemlerinin sınıflandırılması

Optimizasyonun sınıflandırılmasının henüz kesin bir şekilde belirlenmediği ve literatürde özellikle bazı terminolojilerin kullanımı konusunda belirsizliklerin bulunduğu gözlemlenmiştir. Bu çalışmada, en yaygın terminolojilerin kullanılmasına karar verilmiştir. Bununla birlikte, sınıflandırmalarda titiz olunması amaçlanmamış; daha ziyade ilgili tüm kavramların özlü bir şekilde tanıtılması hedeflenmiştir. Daha açık bir ifadeyle, sınıflandırmanın amaç sayısı, karar değişkenlerinin sayısı, doğrusal olup olmadığı, kısıt sayısı, karar değişkenlerinin değeri, amaç ve kısıt fonksiyonlarının ayrılabilirliği, optimizasyon problemlerinin ayrık veya sürekli olabilmeleri açısından yapılabileceği belirtilmiştir.

Optimizasyon problemleri amaç sayısına göre tek amaçlı ve çok amaçlı olmak üzere iki kategoride sınıflandırılabilir. Tek amaçlı optimizasyon problemi terimi, tek bir amaç fonksiyonunun olduğu bir problemi tanımlamak için kullanılır. Buna karşılık, çok amaçlı optimizasyon problemi birden fazla amaç fonksiyonunun bulunduğu bir problem olarak tanımlanır (Yang, 2010b:17).

Gerçek dünyadaki optimizasyon problemlerinin çoğunun çok amaçlı olduğu gözlemlenmiştir. Örneğin, bir araba motoru tasarlanırken, yakıt verimliliğinin en üst düzeye çıkarılması, karbondioksit emisyonunun en aza indirilmesi ve gürültü seviyesinin düşürülmesi hedeflenmektedir. Burada üç amaç bulunmaktadır. Bazen bu hedeflerin birbiriyle çeliştiği ve bazı uzlaşmalara ihtiyaç duyulduğu görülmektedir. Örneğin, bir ev kiralanırken veya satın alınırken, mümkün olan en iyi konumda ve geniş bir alana sahip olunması istenirken, mümkün olduğunca az harcama yapılması amaçlanmaktadır.

Bazı optimizasyon problemlerinde, çözümün kalitesini değerlendirmek için birden fazla kriter olabilir ve bu kriterleri tek bir amaç fonksiyonunda toplamak mümkün olmayabilir ve birden fazla amaç fonksiyonu oluşturulur. Bu gibi durumlarda, çok amaçlı optimizasyon problemlerinin çözümü tek amaçlı optimizasyon problemlerine göre daha zordur, çünkü tanımlanan amaç fonksiyonlarından birinin minimize edilmesi gerekirken diğerinin maksimize edilmesi gerekebilir (Karaboğa, 2014:5).

Bir optimizasyon problemindeki karar değişkenlerinin sayısı, problemin karmaşıklığını ölçmeye yarar. Tek bir karar değişkeni tek boyutlu bir optimizasyon problemini oluştururken,

çok boyutlu bir optimizasyon problemi birden fazla karar değişkeninin varlığıyla tanımlanır (Ansay ve Köse, 2023:85).

Optimizasyon problemleri bağlamında, doğrusal bir optimizasyon problemi, amaç ve kısıtlama fonksiyonlarının doğrusal olduğu bir problem olarak tanımlanır. Buna karşılık, doğrusal olmayan bir optimizasyon problemi, amaç veya kısıtlama fonksiyonlarından en az birinin doğrusal olmadığı bir problemdir (Ertürk vd., 2021:144).

Ayrıca, optimizasyon problemleri kısıtsız optimizasyon problemleri ve kısıtlı optimizasyon problemleri olarak da ayrılabilirler. Kısıtsız optimizasyon problemleri, amaç fonksiyonunun optimize edilmesi gereken herhangi bir kısıtın bulunmadığı problemlerdir. Bu tür problemler genellikle daha basit ve çözümü daha kolaydır. Kısıtlı optimizasyon problemleri, amaç fonksiyonunun optimize edilmesi gereken ve belirli kısıtların bulunduğu problemlerdir. Bu tür problemler, gerçek dünya uygulamalarında daha yaygındır çünkü çoğu zaman belirli sınırlamalar altında en iyi çözümü bulmak gerekir (Yang, 2010b:68).

Bir optimizasyon probleminde amaç fonksiyonunu oluşturan karar değişkenlerinin tamamı veya bir kısmı tamsayı değerler alacak şekilde tanımlanmışsa, bu tür problemler tamsayı programlama problemleri olarak adlandırılır. Bir optimizasyon problemindeki karar değişkenleri gerçek değerler alabiliyorsa, bu problem gerçek değerli programlama problemi olarak adlandırılır (Lokman, 2017:21).

Ayrıca, optimizasyon problemleri amaç ve kısıt fonksiyonlarının ayrılabilirliğine göre sınıflandırılabilir. Fonksiyonlar alt fonksiyonlara ayrıştırılabiliyorsa, ayrıştırılabilir optimizasyon problemleri olarak sınıflandırılırlar; tersine, fonksiyonların alt fonksiyonlara ayrıştırılamadığı problemler ayrıştırılamaz optimizasyon problemleri olarak adlandırılır (Jia vd., 2021:2).

Ayrıca, optimizasyon problemleri ayrık veya sürekli olarak sınıflandırılabilir. Ayrık optimizasyon problemleri, ayrık niceliklerin (nesnelerin) belirli bir kritere göre en uygun olacak şekilde düzenlenmesi, gruplandırılması, sıralanması ve seçilmesi ile ilgilidir. Ayrık optimizasyon problemlerine bir örnek, gerçekleştirilecek bir görev listesi için en uygun sıralamanın belirlenmesidir. Karar değişkenlerinin değerlerinin sürekli olduğu optimizasyon problemleri sürekli optimizasyon problemleri olarak adlandırılır (Baş, 2020:11).

Genel olarak, optimizasyon algoritmaları iki ana kategoride sınıflandırılabilir: Deterministik ve Stokastik. Deterministik algoritmalar, aynı başlangıç değerlerini kullanarak

her iterasyonda aynı sonuçları verir ve problemin kesin çözümünü sağlar. Buna karşılık, stokastik algoritmalar, doğasında var olan rastgelelik nedeniyle, aynı başlangıç değerleriyle yürütüldüklerinde bile tipik olarak benzer ancak farklı sonuçlar verir. Kesin bir çözümü garanti etmemekle birlikte, neredeyse kesin olan bir çözüm sağlarlar (Yang, 2010b:80).

Deterministik algoritmalar, basit problemlerin ele alınmasında ve yerel optimumların belirlenmesinde oldukça etkilidir. Bununla birlikte, bu algoritmaların küresel bir optimuma ulaşmadan yerel bir optimuma yakınsama olasılığı vardır. Deterministik algoritmaların yerel optimuma ulaşmasını önlemek için, algoritmik sürece rastgelelik eklenerek stokastik yetenekler ortaya çıkarılır (Yang, 2010b:80).

Stokastik algoritmalar doğası gereği bir dereceye kadar rastgelelik içerir (Yang, 2010b:20). Bu algoritmalar özellikle geleneksel deterministik yöntemlerin zorlanabileceği karmaşık problemleri çözmek için kullanışlıdır.

2.2. Sezgisel Algoritmalar

Sezgisel algoritmalar, karmaşık problemlerin çözümünde büyük öneme sahiptir ve çözüm bulma sürecini hızlandırmak ve çözümlerin keşfi için etkili ve uyarlanabilir metodolojiler sunar (Karakuzu, 2017:4706). Bu algoritmalar en az bir amaç fonksiyonunu optimize eder ve doğal süreçlerden veya sosyal etkileşimlerden ilham alarak optimum çözümleri belirlemek için tasarlanmıştır. Önemli sezgisel algoritmalar arasında Parçacık Sürü Optimizasyonu (PSO) (Kennedy ve Eberhart, 1995:1943), Genetik Algoritma (GA) (Mirjalili ve Mirjalili, 2019:43), Farksal Gelişim (DE) (Kukkonen ve Lampinen, 2005:444), Yapay Arı Kolonisi (ABC) (Karaboğa vd., 2014:22), Gri Kurt Optimizasyon algoritması (GWO) (Mirjalili vd., 2014:47) ve Karınca Kolonisi Optimizasyonu (ACO) (Dorigo ve Stützle, 2019:312) sayılabilir. Bununla birlikte, PSO, GA, DE, ABC, GWO ve ACO gibi popülasyon tabanlı sezgisel algoritmalar belirli sınırlamalar sergileyebilir. Geniş bir aday çözüm kümesinin bakımı ve güncellenmesi, bu algoritmalar için önemli hesaplama maliyetlerine neden olabilir. Ayrıca, bu algoritmaların etkinliği parametrelerin titizlikle kalibre edilmesine bağlıdır ki bu da zorlu bir süreç olabilir. Ayrıca, bu algoritmalar yerel optimumlardan kaçınmada zorluklarla karşılaşabilir ve karmaşık arama uzaylarında küresel optimumun keşfini garanti edemeyebilir.

2.2.1. Tek aday optimizasyon algoritması (SCO)

Tek Aday Optimizasyon Algoritması (SCO), optimizasyon sürecinde yalnızca bir arama aday çözümüne odaklanarak geleneksel çoklu aday çözüm yaklaşımlarından farklı olan yenilikçi bir metodolojidir. Popülasyon tabanlı sezgisel algoritmaların aksine, SCO algoritması

tek bir çözüme odaklanarak optimizasyon başarımını artırmayı ve zaman maliyetini azaltmayı amaçlamaktadır (Doğan ve Yüzgeç, 2023a:3). SCO, optimizasyon sürecini iki aşamaya ayırır: keşif ve sömürü. İki aşamalı stratejinin amacı, keşif ve sömürü arasında çeşitlilik ve dengeyi sağlamaktır. Her iki aşamada da sinerjik etkiler yaratmak için çeşitli stratejiler ve mekanizmalar uygulanır (Shami vd., 2022:5). Tek aday yaklaşımı ve iki aşamalı süreç, SCO'nun her bir metodolojinin avantajlarından verimli bir şekilde yararlanmasını sağlayarak esnek ve güçlü bir optimizasyon aracı yaratır (Doğan ve Yüzgeç, 2023b:3). Tek çözüme dayalı algoritmalar ve iki aşamalı yaklaşımlar iki yerleşik sezgisel optimizasyon yöntemi olmasına rağmen, ayrı ayrı uygulanmışlardır. Geliştirilen yaklaşım, tek aday yaklaşımını iki aşamalı strateji ile bütünleştirerek tek bir sağlam algoritma oluşturmaktadır. En önemlisi, önerilen algoritma, aday çözümün konumunu yalnızca mevcut konumuna, yani bilgisine dayanarak güncellemek için benzersiz bir denklem seti sunmaktadır. SCO'nun ilk aşamasında, aday çözümün konumu önceden tanımlanmış bir formülün uygulanması yoluyla güncellenir. Denklem 2.1'de verilen bu formül, algoritmanın keşfetme aşamasında çalışmakta ve genel optimizasyon sürecine katkıda bulunmaktadır. Denklem 2.2'de verilen formül ise SCO algoritmasının ağırlık parametresinin hesaplanmasını göstermektedir.

$$x(j) = \begin{cases} S(j) + (w|S(j)|), eğer\ r_1 < 0{,}5 \\ S(j) - (w|S(j)|),\ diğer \end{cases} \quad (2.1)$$

$$w(i) = e^{-(\frac{bi}{i_{max}})^b} \quad (2.2)$$

Bu bağlamda, x(j) aday çözümün j. boyuttaki konumunu, w ağırlık parametresini, S(j) her iterasyon sonunda elde edilen en iyi çözümü, b sabit bir parametreyi, i mevcut iterasyonu, i_{max} maksimum iterasyon sayısını ve r_1 0 ile 1 arasında rastgele bir sayıyı temsil etmektedir. SCO'nun 2. aşamasında, ilk aşamada belirlenen en iyi konumun hemen yakınında bir arama gerçekleştirilir. Bu derinlemesine keşif süreci, arama uzayının büyük bir bölümünü kapsar ve böylece potansiyel çözüm uzayının geniş bir kesitini tarar. Daha sonraki aşamalarda, arama süreci daraltılarak en verimli sonuçların beklendiği bölgelere odaklanılır ve bu bölgeler daha ayrıntılı olarak incelenir. Denklem 2.3'te sömürü aşamasında aday çözümün konum güncellemesinin adım adım nasıl yapıldığını açıklamaktadır.

$$x(j) = \begin{cases} S(j) + (wr_2(UB(j) - LB(J)), eğer\ r_2 < 0{,}5 \\ S(j) - (wr_2(UB(j) - LB(J)),\ diğer \end{cases} \quad (2.3)$$

Bu denklemde, UB ve LB sırasıyla üst ve alt sınırları temsil eder ve r_2 0 ile 1 arasında bir değer alabilen rastgele bir sayıdır. SCO algoritması, artan fonksiyon değerlendirme sayısı

ile üstel olarak azalan ağırlık parametresinin (w) uyarlanabilir bir davranışına sahiptir. Burada bu mekanizma, optimizasyon sürecinde sömürü ve keşif faaliyetleri arasında dengeli bir geçişin sağlanmasında kritik bir rol oynamaktadır. Başlangıçta, w parametresi yüksek bir değere ayarlanarak SCO'nun arama uzayında kapsamlı bir keşif yapmasına izin verilir. Optimizasyon süreci ilerledikçe, w değerinin düşürülmesi çözümün iyileştirilmesine ve sömürü aşamasına odaklanılmasına olanak tanır. İkinci aşamada SCO, ilk aşamada elde edilen en iyi konumun etrafındaki uzayı kapsamlı bir şekilde keşfederek başlayan derin bir arama gerçekleştirir ve yerel optimumlarda sıkışıp kalma riskini en aza indirmek için konum güncelleme mekanizmasını değiştirir. Güncelleme denklemi (Denklem 2.3), optimizasyon sırasında aday çözümlerin keşif aşamasından sömürü aşamasına geçişine yardımcı olduğu için SCO çerçevesinde çok önemlidir ve aranacak uzayı azaltarak yalnızca gelecek vaat eden bölgelere odaklanılmasına yardımcı olur. Bu geçiş, algoritmanın aday çözümlerin arama uzayının optimal olmayan bölgelerinde sıkışıp kaldığı yerel optimumlardan kaçınma becerisini geliştirir. Konumları güncelleme mekanizmasını değiştirerek SCO, arama uzayının çeşitli bölgelerini verimli bir şekilde keşfedebilir. Bu, yerel optimal çözümlerde takılıp kalma riskini azaltır ve sonuçta daha iyi optimizasyon sonuçlarına yol açar. Ardışık m fonksiyon değerlendirmesinden sonra herhangi bir gelişme olmazsa, aday çözümün konumu Denklem 2.4'te belirtildiği gibi güncellenir.

$$x(j) = \begin{cases} S(j) + (r_4(UB(j) - LB(J)), eğer\ r_3 < 0{,}5 \\ S(j) - (r_5(UB(j) - LB(J)),\ diğer \end{cases} \quad (2.4)$$

Bu durumda, $r_{3\text{-}5}$ rastgele üretilen sayıları temsil eder.

Fonksiyon değerlendirme sayısı arttıkça w üstel olarak azalır. Bu davranış, arama sürecinin başlangıcında nispeten yüksek bir w değeri arama uzayının etkin bir şekilde keşfedilmesine yardımcı olurken, küçük bir w değeri optimizasyon sürecinin sonraki aşamalarında sömürü yeteneklerini güçlendirdiği için çok önemlidir. Sezgisel algoritmaların temel kısıtlamalarından biri, özellikle arama sürecinin son aşamalarında yerel optimumlarda sıkışıp kalmaktır. Başka bir deyişle, aday çözümlerin konumlarının sürekli güncellenmesi her zaman maliyet değerinde iyileştirme sağlamaz. SCO bu sorunu, m ardışık fonksiyon değerlendirmesinde uygunluk iyileştirmesi sağlanamazsa ikinci aşamada aday çözümün konumunu farklı şekilde güncelleyerek çözmektedir. Ardışık olarak uygunluk iyileştirmesi elde edilemeyen m fonksiyon değerlendirmesinin sayısını saymak için bir c sayacı kullanılır. Güncellenen adayın başarılı bir uygunluk elde edip edemeyeceğini belirlemek için ikili bir p parametresi kullanılır; burada p = 1 başarılı uygunluk iyileştirmesini gösterirken p = 0 uygunluk

iyileştirme başarısızlığını gösterir. SCO algoritmasının sözde kodu Algoritma 2.1'de sunulmuştur.

Algoritma 2.1. SCO'nun Sözde Kodu

```
1.lb,ub,dim,fc,Max_iter,m,a,b değerlerini gir
2.FitImp=0                                  //FitImp=0 uygunluk iyileştirmesinin olmadığını gösterir
3.for i=1'den Boyut'a (dim) kadar
4.      S(i)=lb(i)+rand*(ub(i) – lb(i))                                          //En iyi konum
5.end for
6.BF=fc(S)                                                                  //BF en iyi çözüm (Best Fitness)
7.Count=0                                                                // Başarısız uygunluk denemesi sayacı
8.for t=1'den Max_iter'e kadar
9.      w(t)=exp(-(bt/(Max_iter ))^b )
10.     if t>α ve FitImp=0 ise
11.             Count=1+ Count                                          //Başarısız uygunluk denemesi sayımı
12.     end if
13.     K=rand
14.     for j=1'den Boyut'a (dim) kadar
15.             if t<α ise
16.                     if rand<0,5 ise
17.                             X(j)=S(j)+w(t)*|S(j)|
18.                     else
19.                             X(j)=S(j)-w(t)*|S(j)|
20.                     end if
21.             else
22.             if Count=m ise
23.                     Count=0
24.                     if rand<0,5 ise
25.                             X(j)=S(j)+rand*(ub(j)-lb(j))
26.                     else
27.                             X(j)=S(j)-rand*(ub(j)-lb(j))
28.                     end if
29.             else
30.                     if rand<0,5 ise
31.                             X(j)=S(j)+w(t)*K*(ub(j)-lb(j))
32.                     else
33.                             X(j)=S(j)-w(t)*K*(ub(j)-lb(j))
34.                     end if
35.             end if
36.     end if
37.     if X(j)>ub(j) ise
38.             X(j)=S(j)
39.     end if
40.     if X(j)<lb(j) ise
41.             X(j)=S(j)
42.     end if
43.     end for
44.     if fc(x)<BF ise
45.             BF=fc(x), S=X, FitImp=1
46.     else
47.             FitImp=0
48.     end if
49.     gbest=S
50.end for
```

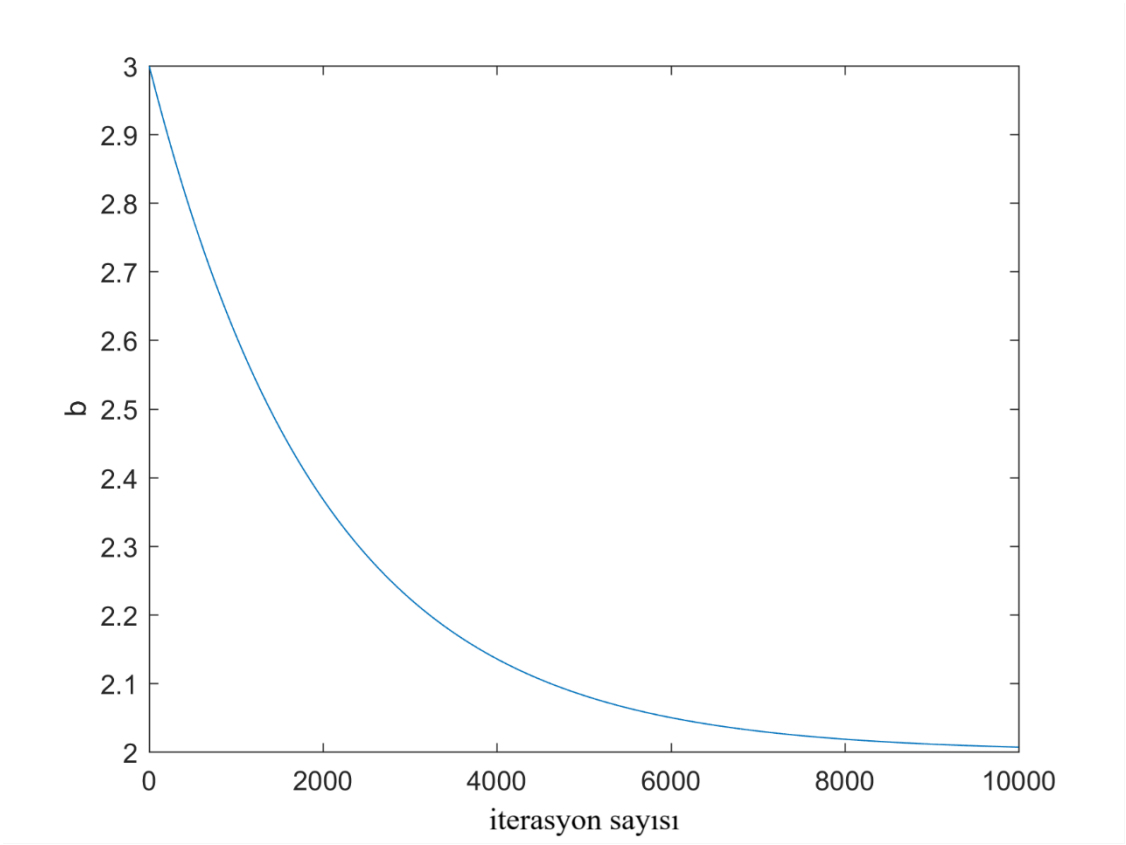
2.2.2. Kaotik mutasyon stratejisi tabanlı tek aday optimize edici (CSCO)

SCO algoritmasının bazı kısıtlamalarının üstesinden gelmek ve genel başarımını iyileştirmek için ilk olarak bu tez kapsamında kaotik davranış sergileyen Chaucy, Gauss ve Levy dağılımlarına dayanan yeni bir mutasyon operatörü önerilmiştir (Dokur vd., 2022:9). Bu yenilikçi mutasyon tekniği, algoritmanın keşif kapasitesini artırarak küresel optimizasyon sürecine katkıda bulunmaktadır (İnaç vd., 2022:14632). Önerilen mutasyon operatörü, aday çözümlerin konumlarını güncellerken arama uzayının daha geniş bölgelerine erişerek yerel minimumlardan kaçınmayı mümkün kılar, böylece çözüm uzayının daha verimli bir şekilde taranmasını sağlar. Bu yaklaşım, algoritmanın çeşitliliği korumasına ve arama uzayının farklı bölgelerindeki potansiyel çözümleri keşfetmesine olanak tanıyarak küresel optimuma ulaşma olasılığını artırmaktadır.

Fonksiyon değerlendirmelerinin m iterasyonu boyunca herhangi bir gelişme gözlenmediğinde, orijinal SCO algoritmasında yer alan mekanizma yerine kaotik mutasyon mekanizması devreye girerek algoritmanın yerel optimumlarda takılıp kalma gibi dezavantajlardan kaçınmasını sağlar. Bu yöntem, özellikle algoritmanın sömürü aşamasında kaotik fonksiyonların rastgele ve deterministik olmayan doğasından yararlanarak orijinal SCO algoritmasının sınırlamalarının üstesinden gelmeyi amaçlamaktadır. Bu, algoritmanın genel başarımı üzerinde olumlu bir etkiye sahiptir ve optimizasyon sürecinin verimliliğini artırır. SCO algoritmasında w ağırlık değerinin hesaplanmasında kullanılan b değeri sabit olup 2,4 değerine sahiptir. CSCO algoritmasında ise b değeri Denklem 2.5'te sunulduğu gibi sabit olarak kullanılmayarak, uyarlanabilir şekilde güncellenmiştir. b değeri kontrol edilerek 3 değerinden başlayıp iterasyon sayısı arttıkça 2 değerine doğru azalan şekilde kullanılmıştır.

$$b = 2 + e^{-\frac{5*i}{i_{max}}} \tag{2.5}$$

Bu denklemde i mevcut iterasyonu, i_{max} maksimum iterasyon sayısını temsil etmektedir. Şekil 2.1 b değerinin iterasyon sayısına göre grafiğini göstermektedir.



Şekil 2.1. İterasyon Sayısına Göre b Değişkeni Eğrisi

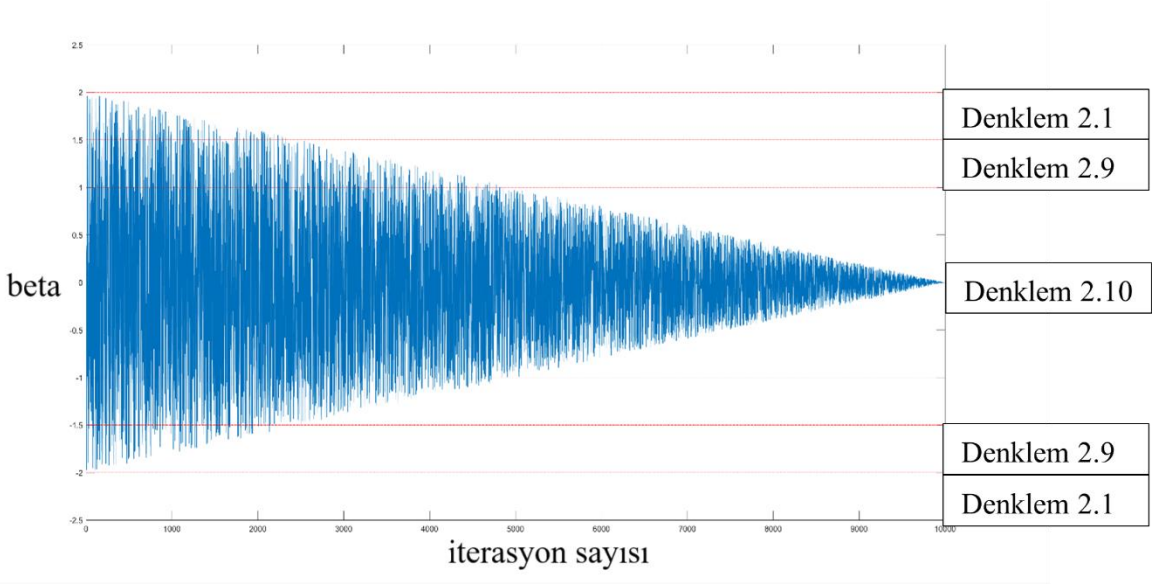
SCO algoritmasında keşif aşamasının iterasyon sayısı için kullanılan α değişkeni 1000 değerinde sabit olarak kullanılıyorken CSCO algoritmasında keşif aşamasından sömürü aşamasına geçişlerde p1 parametresi kullanılır. SCO algoritmasında başarısız yakınsama deneme sayısının sayaç ile sayılıp 5 değerine ulaştıktan sonra daha iyi yakınsama yapılması için yapılacak işlemler sadece sömürü aşamasında yapıldığı görülmüştür. CSCO algoritmasında ise başarısız deneme sayısı keşif aşamasının başından itibaren kontrol edilerek algoritmanın çözüm noktasına daha erken yakınsama yapması planlanmıştır. CSCO algoritmasında Denklem 2.6'deki formül ile yeni bir p_1 isimli parametre kullanılmaktadır. Denklem 2.7'de p_0 isimli rastgele değer üreten parametre sunulmaktadır. Denklem 2.8'de beta değişkeninin formülü sunulmaktadır.

$$p_1 = 2\left(1 - \left(\frac{i}{i_{max}}\right)\right) \tag{2.6}$$

$$p_0 = 2r_2 - 1 \tag{2.7}$$

$$\beta = p_0 . p_1 \tag{2.8}$$

Denklem 2.7'de bulunan r_2 rastgele üretilen değeri temsil eder. Denklem 2.8'de β, beta değerini temsil etmektedir. β değişkeni sayesinde Cauchy, Gauss ve Levy kaotik stratejileri kullanılarak CSCO algoritmasının özellikle keşif aşamasında SCO algoritmasından daha fazla bölge taranarak yerel minimum noktalarından kaçınması sağlanmıştır. Şekil 2.2 β değişkeninin grafiğini göstermektedir. β değişkeninin mutlak değeri 1,5 ile 2 arasında ise SCO algoritmasının yakınsama formülleri kullanılır ve eğer β değerinin mutlak değeri 1 ile 1,5 arasında ise CSCO algoritmasında geliştirilmiş yakınsama formülleri kullanılır.



Şekil 2.2. İterasyon Sayısına Göre β Değişkeni Eğrisi

Algoritma 2.2'de CSCO algoritmasındaki mutasyon yapısı verilmiştir. β değişkeninin kontrolü ile Cauchy, Gauss veya Levy kaotik stratejilerinden birinin kullanılarak CSCO algoritmasına kaotik mutasyon yapısı kazandırılmaktadır.

Algoritma 2.2. CSCO Algoritmasındaki Mutasyon Yapısı

```
r1=rand;
    if r1>0.7
        x=beta*cauchy(S); % cauchy mutasyonu
    elseif r1<0.7 && r1>0.3
        x=beta*gauss(S); % gauss mutasyonu
    else
        x=beta*levy(S); % levy mutasyonu
    end
```

Algoritma 2.3'de sunulduğu gibi SCO algoritmasından farklı olarak β değerinin mutlak değeri kontrol edilir. Eğer β değerinin mutlak değeri 1,5 ile 2 arasında ise SCO algoritmasındaki yakınsama formülü kullanılır. β değerinin mutlak değeri 1 ile 1,5 arasında ise SCO algoritmasından farklı olarak Denklem 2.9 ve Denklem 2.10'da sunulan yakınsama formülleri kullanılır.

Algoritma 2.3. CSCO Algoritmasındaki Yakınsama Yapısı

```
if  abs(beta)>1.5 && abs(beta)<=2
        if rand<0.5
                x(j) = S(j)+(w(t)*abs(S(j)));
        else
                x(j) = S(j)-(w(t)*abs(S(j)));
        end
else if abs(beta)>=1 && abs(beta)<1.5
        if rand<0.5
                x(j) = S(j)+(w(t)*sin((2*pi)*rand));
        else
                x(j) = S(j)+(w(t)*cos((2*pi)*rand));
        end
else
        if rand<0.5
                x(j) = S(j)+ w(t)*K*(ub(j)-lb(j))*sin((2*pi)*rand);
        else
                x(j) = S(j)+ w(t)*K*(ub(j)-lb(j))*cos((2*pi)*rand);
        end
end
```

$$x(j) = \begin{cases} S(j) + (w\sin(2\pi r_3)), & r_3 < 0 \\ S(j) - (w\sin(2\pi r_3)), & r_3 \geq 0 \end{cases} \tag{2.9}$$

$$x(j) = \begin{cases} S(j) + (wr_5(UB(j) - LB(J)\sin(2\pi r_4)), & r_4 < 0 \\ S(j) - (wr_5(UB(j) - LB(J)\sin(2\pi r_4)), & r_4 \geq 0 \end{cases} \tag{2.10}$$

Son olarak Algoritma 2.4'te sunulduğu gibi aday çözümün bir boyutunun sınırların dışına çıkıp çıkmadığını kontrol eden mekanizma CSCO algoritmasında oldukça farklıdır. SCO algoritmasında aday çözüm sınırların dışına çıktığında, yeni aday çözüm olarak sınırlar içinde yer alan son aday çözüm kullanılır. Bu yapının kullanılması aday çözümün tek bir noktaya takılmasına neden olabilir. CSCO algoritmasında ise sınır kısıtları aşıldığında sınırlar içerisinde kalan aday çözüm ile sınır kısıtlarının ortalaması alınır. Bu sayede SCO algoritmasında gözlemlenen tek çözüme odaklanma engellenmiş olur. Algoritma 2.5 CSCO algoritmasının sözde kodunu göstermektedir.

Algoritma 2.4. CSCO Algoritmasındaki Sınır Kısıtlaması

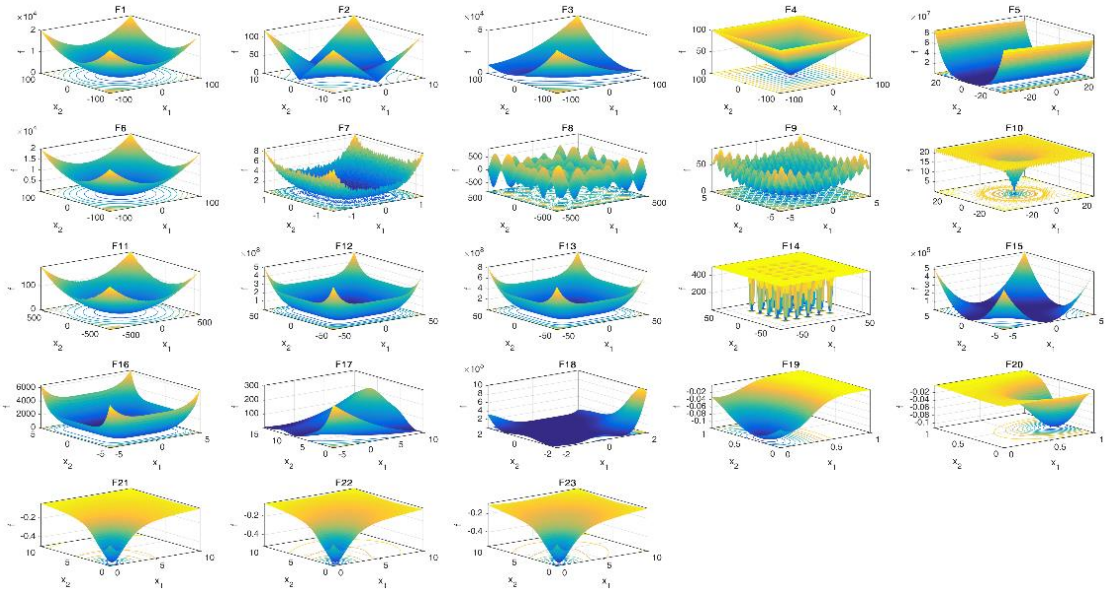
```
function vi = boundConstraint (vi, pop, lu)
% sınır kısıtı ihlal edilirse, değeri önceki değer ile sınırın ortası olacak şekilde ayarlayın
%
[NP, D] = size(pop);  % NP popülasyon büyüklüğü, D problem boyutu
%% alt sınırı kontrol et
xl = repmat(lu(1, :), NP, 1);
pos = vi < xl;
vi(pos) = (pop(pos) + xl(pos)) / 2;
%% üst sınırı kontrol et
xu = repmat(lu(2, :), NP, 1);
pos = vi > xu;
vi(pos) = (pop(pos) + xu(pos)) / 2;
end
```

Algoritma 2.5. CSCO'nun Sözde Kodu

```
1.lb,ub,dim,fc,Max_iter,m değerlerini gir
2.FitImp=0                          //FitImp=0 uygunluk iyileştirmesinin olmadığını gösterir
3.for i=1'den Boyut'a (dim) kadar
4.    S(i)=lb(i)+rand*(ub(i) – lb(i))                                    //En iyi konum
5.end for
6.BF=fc(S)                                                    //BF en iyi çözüm (Best Fitness)
7.Count=0                                                  // Başarısız uygunluk denemesi sayacı
8. for t=1'den Max_iter'e kadar
9.        b = 2+exp(-(5*t/Max_iter))                                  //Uyarlanabilir b değeri
10.       w(t)=exp(-(bt/(Max_iter ))^b )
11.       if FitImp=0 ise
12.               Count=1+ Count                                  //Başarısız uygunluk denemesi sayımı
13.       end if
14.       K=rand
15.       p1=2*(1-(t/Max_iter))     //keşif aşamasından sömürü aşamasına kadar kullanılan p1 parametresi
16.       for j=1'den Boyut'a (dim) kadar
17.               p0=2*rand()-1                                   //p0 parametresi
18.               beta=p0*p1       //kaotik stratejileri ve yeni yakınsama mekanizmaları için beta değişkeni
19.               if Count=m ise
20.                       Count=0
21.                       r1=rand
22.                       if r1>0.7 ise                          //kaotik stratejileri
23.                               x=beta*cauchy(S)
24.                       else if r1<0.7 ve r1>0.3 ise
25.                               x=beta*gauss(s)
26.                       else
27.                               x=beta*levy(S)
28.                       end if
29.                       break
30.               else
31.                       if abs(beta)>1.5 ve abs(beta)<=2 ise
32.                               if rand<0.5 ise
33.                                       x(j)=S(j)+(w(t)*abs(S(j)))
34.                               else
35.                                       x(j) = S(j)-(w(t)*abs(S(j)))
36.                               end if
37.                       else if abs(beta)>=1 ve abs(beta)<1.5 ise
38.                               if rand<0.5 ise
39.                                       x(j) = S(j)+(w(t)*sin((2*pi)*rand))
40.                               else                                  //yeni yakınsama mekanizmaları
41.                                       x(j) = S(j)+(w(t)*cos((2*pi)*rand))
42.                               end if
43.                       else
44.                               if rand<0.5 ise
45.                                       x(j) = S(j)+w(t)*K*(ub(j)-lb(j))*sin((2*pi)*rand)
46.                               else                                  //yeni yakınsama mekanizmaları
47.                                       x(j) = S(j)+w(t)*K*(ub(j)-lb(j))*cos((2*pi)*rand)
48.                               end if
49.                       end if
50.               end if
51.       end for
52.       x = boundConstraint(x,S,lu)                            //yeni sınır kontrol mekanizması
53.       if fc(x)<BF ise
54.               BF=fc(x), S=X, FitImp=1
55.       else
56.               FitImp=0
57.       end if
58.       gbest=S
59.end for
```

2.3. Kıyaslama Fonksiyonları

Bu çalışma, önerilen CSCO algoritmasının etkinliğini, literatürde yaygın olarak kabul gören 23 kıyaslama fonksiyonları üzerinde orijinal SCO algoritması ile karşılaştırmalı olarak değerlendirmektedir. Seçilen fonksiyonların tamamı tekli minimizasyon problemleridir. Şekil 2.3, seçilen fonksiyonların iki boyutlu uzayda görsel bir temsilini sunmaktadır.



Şekil 2.3. Kıyaslama Fonksiyonları

Bu fonksiyonlardan 1-7 arasındaki fonksiyonlar tek modlu test fonksiyonlarıdır (Kılıç ve Yüzgeç, 2019a:677). Tablo 2.1'de fonksiyonların formülleri, sınır değerleri ve minimum değerleri bulunmaktadır. 8-13 arasındaki fonksiyonlar ise çok modlu test fonksiyonlarıdır(Kılıç ve Yüzgeç, 2019b:170). Tablo 2.2'de fonksiyonların formülleri, sınır değerleri ve minimum değerleri bulunmaktadır. 14-23 arasındaki fonksiyonlar ise sabit boyutlu çok modlu test fonksiyonlarıdır. Tablo 2.3'de fonksiyonların formülleri, boyutları, sınır değerleri ve minimum değerleri bulunmaktadır.

Tablo 2.1. Tek Modlu Test Fonksiyonları

Fonksiyon	**Sınır Değerleri**	$\mathbf{f_{min}}$
$f_1(x) = \sum_{i=1}^{n} x_i^2$	$[-100,100]$	0
$f_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	$[-10,10]$	0

Tablo 2.1. Tablonun Devamı

$f_3(x) = \sum_{i=1}^{n} \left(\sum_{j-1}^{i} x_j \right)^2$	[−100,100]	0
$f_4(x) = max_i\{\|x_i\|, 1 \le i \le n\}$	[−100,100]	0
$f_5(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	[−30,30]	0
$f_6(x) = \sum_{i=1}^{n} ([x_i + 0{,}5])^2$	[−100,100]	0
$f_7(x) = \sum_{i=1}^{n} ix_i^4 + rastgele[0,1)$	[−1.28, 1.28]	0

Kaynak: (Yüzgeç ve İnaç, 2016:12)

Tablo 2.2. Çok Modlu Test Fonksiyonları

Fonksiyon	**Sınır Değerleri**	**f_{min}**
$f_8(x) = \sum_{i=1}^{n} -x_i \sin\left(\sqrt{\|x_i\|}\right)$	[−500,500]	−418.9829 × Boyut
$f_9(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10$	[−5.12,5.12]	0
$f_{10}(x) = -20\exp\left(-0{,}2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	[−32,30]	0
$f_{11}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	[−600,600]	0

Tablo 2.2. Tablonun Devamı

$f_{12}(x) = \frac{\pi}{n}\left\{10\sin(\pi y_1) + \sum_{i=1}^{n-1}(y_i-1)^2[1+10sin^2(\pi y_i+1)] + (y_n-1)^2 + \sum_{i=1}^{n} u(x_i,10,100,4)\right\}$ $y_i = 1+\frac{x_i+1}{4}$ $u(x_i,a,k,m) = \begin{cases} k(x_i-a)^m, & x_i > a \\ 0, & -a < x_i < a \\ k(-x_i-a)^m, & x_i < -a \end{cases}$	[−50,50]	0
$f_{13}(x) = 0{,}1\left\{sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i-1)^2[1+sin^2(3\pi x_i+1)] + (x_n-1)^2[1+sin^2(2\pi x_n)]\right\} + \sum_{i=1}^{n} u(x_i,5,100,4)$	[−50,50]	0

Kaynak: (Kılıç ve Yüzgeç, 2021:39)

Tablo 2.3. Sabit Boyutlu Çok Modlu Test Fonksiyonları

Fonksiyon	**Boyut**	**Sınır Değerleri**	**f_{min}**
$f_{14}(x) = \left(\frac{1}{500} + \sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i-a_{ij})^6}\right)^{-1}$	2	[−65,65]	1
$f_{15}(x) = \sum_{i=1}^{11}\left[a_i - \frac{x_1(b_i^2+b_i x_2)}{b_i^2+b_i x_3+x_4}\right]^2$	4	[−5,5]	0.00030
$f_{16}(x) = 4x_1^2 - 2{,}1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	[−5,5]	−1.0316
$f_{17}(x) = \left(x_2 - \frac{5{,}1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1-\frac{1}{8\pi}\right)cosx_1 + 10$	2	[−5,5]	0.398
$f_{18}(x) = [1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)] * [30+(2x_1-3x_2)^2 * (18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)]$	2	[−2,2]	3

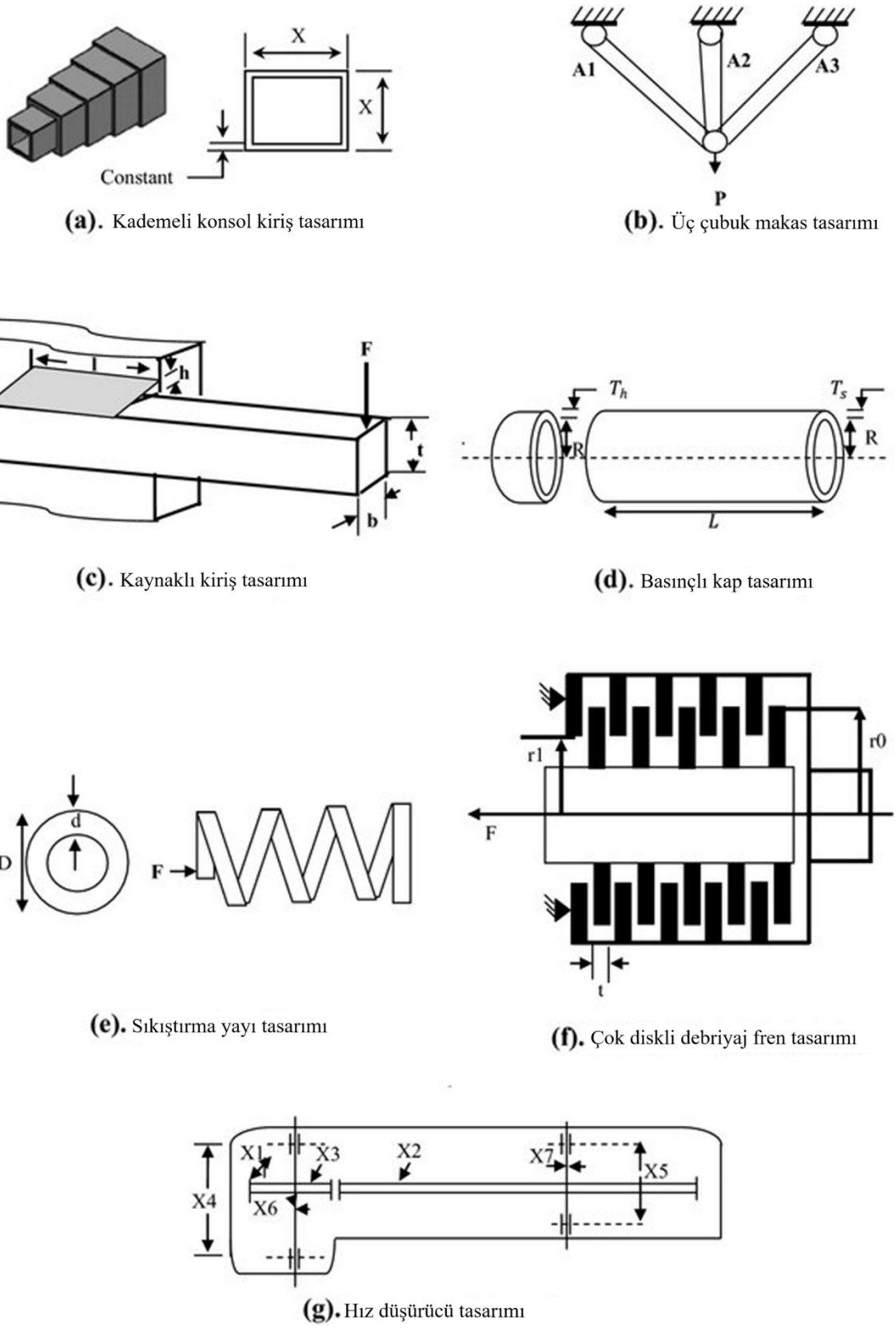
Tablo 2.3. Tablonun Devamı

$f_{19}(x) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2\right)$	3	[1,3]	−3.86
$f_{20}(x) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2\right)$	6	[0,1]	−3.32
$f_{21}(x) = -\sum_{i=1}^{5} [(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	[0,10]	−10.1532
$f_{22}(x) = -\sum_{i=1}^{7} [(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	[0,10]	−10.4028
$f_{23}(x) = -\sum_{i=1}^{10} [(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	[0,10]	−10.5363

Kaynak: (Shami vd., 2022:869)

2.4. Mühendislik Tasarım Problemleri

Bu bölümde, en yaygın mühendislik tasarım problemleri sunulmaktadır. Problemleri daha net açıklamak için matematiksel gösterim ve tanım verilmiştir. Bu problemler gerçek dünyada en çok bilinen mühendislik tasarım problemleridir. Genellikle, optimum tasarımın elde edilmesi için optimum parametrelerin belirlenmesinde etkin bir yönteme ihtiyaç duyulmaktadır. Her problemin ayarlanması gereken bazı değişkenleri vardır. Ayrıca, değişkenlerin verilen aralıktaki değerlerini sağlamak için bazı kısıtlamalar verilmiştir. Bu tasarım problemlerinden bazıları Şekil 2.4'te gösterilmiştir.



(a). Kademeli konsol kiriş tasarımı

(b). Üç çubuk makas tasarımı

(c). Kaynaklı kiriş tasarımı

(d). Basınçlı kap tasarımı

(e). Sıkıştırma yayı tasarımı

(f). Çok diskli debriyaj fren tasarımı

(g). Hız düşürücü tasarımı

Şekil 2.4. Mühendislik Tasarım Problemleri

Kaynak: (Samma, 2020:5)

2.4.1. Sıkıştırma yayı tasarımı

Mühendislik optimizasyonu alanında en sık karşılaşılan problemlerden biri, sıkıştırma yayı tasarımı olarak da bilinen helezon yayların optimum tasarımıdır (Arora, 2004:36). Optimizasyon görevi, belirli başarım kriterlerini karşılayan ve malzeme arızası olmadan belirli bir eksenel yükü destekleyebilen mümkün olan en düşük kütleye sahip bir yay tasarlamayı amaçlamaktadır (Tzanetos, 2023:2). Minimizasyon prosedürü, izin verilen kayma gerilmesi, dalgalanma frekansı eşiği ve gerekli minimum sapma dahil olmak üzere çeşitli parametrelerle kısıtlanır. Eldeki problem üç değişken içermektedir: tel çapı (d), ortalama bobin çapı (D) ve aktif bobin sayısı (N) (Mirjalili, 2014:55). Bu problemin formülleri Denklem 2.11 ile Denklem 2.19 arasında verilmiştir.

$$\vec{x} = [x_1, x_2, x_3] = [dDN], \tag{2.11}$$

Minimize edilecek fonksiyon,

$$f(\vec{x}) = (x_3 + 2)x_2 x_1^2, \tag{2.12}$$

Kısıtlayıcılar,

$$g_1(\vec{x}) = 1 - \frac{x_2^3 x_3}{71785 x_1^4} \leq 0, \tag{2.13}$$

$$g_2(\vec{x}) = \frac{4x_2^2 - x_1 x_2}{12566(x_2 x_1^3 - x_1^4)} + \frac{1}{5108 x_1^2} \leq 0, \tag{2.14}$$

$$g_3(\vec{x}) = 1 - \frac{140{,}45 x_1}{x_2^2 x_3} \leq 0, \tag{2.15}$$

$$g_4(\vec{x}) = \frac{x_1 + x_2}{1{,}5} - 1 \leq 0, \tag{2.16}$$

Değişken aralıkları,

$$0{,}05 \leq x_1 \leq 2{,}00, \tag{2.17}$$

$$0{,}25 \leq x_2 \leq 1{,}30, \tag{2.18}$$

$$2{,}00 \leq x_3 \leq 15{,}0 \tag{2.19}$$

2.4.2. Kaynaklı kiriş tasarımı

Bu problemin amacı, kaynaklı bir kirişin imalat maliyetini en aza indirmektir. Problem, kesme gerilmesi, eğilme gerilmesi, burkulma yükü, uç sapması ve yan kısıtlamalar gibi kısıtlamaları içerir. Dikkate alınması gereken dört değişken vardır: kaynak kalınlığı (h),

çubuğun bağlı kısmının uzunluğu (l), çubuk yüksekliği (t) ve çubuk kalınlığı (b) (Mirjalili, 2014:55). Bu problemin formülleri Denklem 2.20 ile Denklem 2.32 arasında verilmiştir.

$$\vec{x} = [x_1, x_2, x_3, x_4] = [hltb], \tag{2.20}$$

Minimize edilecek fonksiyon,

$$f(\vec{x}) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14.0 + x_2), \tag{2.21}$$

Kısıtlayıcılar,

$$g_1(\vec{x}) = \tau(\vec{x}) - \tau max \leq 0, \tag{2.22}$$

$$g_2(\vec{x}) = \sigma(\vec{x}) - \sigma max \leq 0, \tag{2.23}$$

$$g_3(\vec{x}) = \delta(\vec{x}) - \delta max \leq 0, \tag{2.24}$$

$$g_4(\vec{x}) = x_1 - x_4 \leq 0, \tag{2.25}$$

$$g_5(\vec{x}) = P - Pc(\vec{x}) \leq 0, \tag{2.26}$$

$$g_6(\vec{x}) = 0.125 - x_1 \leq 0, \tag{2.27}$$

$$g_7(\vec{x}) = 1.10471x_1^2 + 0.04811x_3x_4(14.0 + x_2) - 5.0 \leq 0, \tag{2.28}$$

Değişken aralıkları,

$$0.1 \leq x_1 \leq 2, \tag{2.29}$$

$$0.1 \leq x_2 \leq 10, \tag{2.30}$$

$$0.1 \leq x_3 \leq 10, \tag{2.31}$$

$$0.1 \leq x_4 \leq 2 \tag{2.32}$$

2.4.3. Basınçlı kap tasarımı

Bu problemin amacı, malzeme, şekillendirme ve kaynak masraflarını içeren silindirik bir kabın toplam maliyetini en aza indirmektir. Kabın her iki ucu da kapaklıdır ve baş kısmı yarım küre şeklindedir. Problem dört değişken içermektedir: kabuğun kalınlığı (T_s), başlığın kalınlığı (T_h), iç yarıçap (R) ve başlığın dikkate alınmadığı silindirik bölümün uzunluğu (L). Bu problem dört kısıtlama ile sınırlandırılmıştır (Mirjalili, 2014:56). Bu problemin formülleri Denklem 2.33 ile Denklem 2.42 arasında verilmiştir.

$$\vec{x} = [x_1, x_2, x_3, x_4] = [T_s, T_h, R, L], \tag{2.33}$$

Minimize edilecek fonksiyon,

$$f(\vec{x}) = -0.6224x_2x_3x_4 + 1.7781x_1x_2^3 + 3.1661x_2^2x_4 + 19.84x_2^3, \quad (2.34)$$

Kısıtlayıcılar,

$$g_1(\vec{x}) = -x_1 + 0.0193x_3 \leq 0, \quad (2.35)$$

$$g_2(\vec{x}) = -x_3 + 0.00954x_3 \leq 0, \quad (2.36)$$

$$g_3(\vec{x}) = -\pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 + 1296000 \leq 0, \quad (2.37)$$

$$g_4(\vec{x}) = x_4 - 240 \leq 0, \quad (2.38)$$

Değişken aralıkları,

$$0 \leq x_1 \leq 99, \quad (2.39)$$

$$0 \leq x_2 \leq 99, \quad (2.40)$$

$$10 \leq x_3 \leq 200, \quad (2.41)$$

$$10 \leq x_4 \leq 200 \quad (2.42)$$

2.4.4. Hız düşürücü tasarımı

Hız düşürücü, mekanik sistemin dişli kutusunun bir parçasıdır ve diğer birçok uygulama türünde kullanılır. Hız düşürücünün tasarımı, yedi tasarım değişkeni içerdiğinden daha zorlu bir kriterdir. Hız düşürücünün tasarımı, yüzey genişliği (x_1), dişlerin modülü (x_2), pinyondaki diş sayısı (x_3), rulmanlar arasındaki birinci milin uzunluğu (x_4), rulmanlar arasındaki ikinci milin uzunluğu (x_5), birinci milin çapı (x_6) ve ikinci milin çapı (x_7) ile birlikte ele alınır (Lin, 2013:2). Bu problemin formülleri Denklem 2.43 ile Denklem 2.61 arasında verilmiştir.

Minimize edilecek fonksiyon,

$$f(\vec{x}) = 0{,}7854x_1x_2^2(3{,}3333x_3^2 + 14{,}9334x_3) - 43{,}0934 - 1{,}508x_1(x_6^2 + x_7^2) + 7{,}4777(x_6^3 + x_7^3) + 0{,}7854(x_4x_6^2 + x_5x_7^2), \quad (2.43)$$

Kısıtlayıcılar,

$$g_1(\vec{x}) = \frac{27}{x_1x_2^2x_3} - 1 \leq 0, \quad (2.44)$$

$$g_2(\vec{x}) = \frac{397{,}5}{x_1x_2^2x_3} - 1 \leq 0, \quad (2.45)$$

$$g_3(\vec{x}) = \frac{1{,}93x_4^3}{x_2x_6^4x_3} - 1 \leq 0, \quad (2.46)$$

$$g_4(\vec{x}) = \frac{1{,}93x_5^3}{x_2x_7^4x_3} - 1 \leq 0, \quad (2.47)$$

$$g_5(\vec{x}) = \frac{\sqrt{\left(\frac{745x_4}{x_2x_3}\right)^2 + 16{,}9*10^6}}{110x_6^3} - 1 \leq 0, \quad (2.48)$$

$$g_6(\vec{x}) = \frac{\sqrt{\left(\frac{745x_5}{x_2x_3}\right)^2 + 157{,}5*10^6}}{85x_7^3} - 1 \leq 0, \quad (2.49)$$

$$g_7(\vec{x}) = \frac{x_2x_3}{40} - 1 \leq 0, \quad (2.50)$$

$$g_8(\vec{x}) = \frac{5x_2}{x_1} - 1 \leq 0; \quad (2.51)$$

$$g_9(\vec{x}) = \frac{x_1}{12x_2} - 1 \leq 0, \quad (2.52)$$

$$g_{10}(\vec{x}) = \frac{1{,}5x_6+1{,}9}{x_4} - 1 \leq 0, \quad (2.53)$$

$$g_{11}(\vec{x}) = \frac{1{,}1x_7+1{,}9}{x_5} - 1 \leq 0, \quad (2.54)$$

Değişken aralıkları,

$$2{,}6 \leq x_1 \leq 3{,}6, \quad (2.55)$$

$$0{,}7 \leq x_2 \leq 0{,}8, \quad (2.56)$$

$$17 \leq x_3 \leq 28, \quad (2.57)$$

$$7{,}3 \leq x_4 \leq 8{,}3, \quad (2.58)$$

$$7{,}3 \leq x_5 \leq 8{,}3, \quad (2.59)$$

$$2{,}9 \leq x_6 \leq 3{,}9, \quad (2.60)$$

$$5 \leq x_7 \leq 5{,}5 \quad (2.61)$$

2.4.5. Dişli katarı tasarımı

Matematiksel olarak, dişli dizisi tasarım problemi kısıtlamasız bir optimizasyon problemidir. Bu tasarım probleminin dört tamsayı değişkeni (T_a, T_b, T_d, T_f) vardır. Sürücü ve tahrik edilen miller arasında belirli bir dişli oranı elde etmek için bir dişli takımı tasarlanmalıdır.

Dişli çark tasarımının amacı, elde edilen dişli oranı ile gerekli dişli oranı olan 1/6.931 arasındaki hatayı en aza indirecek şekilde dört dişlinin her birindeki diş sayısını bulmaktır; burada her bir dişli çarkın diş sayısı 12-60 aralığında değişen tam sayıdır (Madic, 2021:3). Bu problemin formülleri Denklem 2.62 ve Denklem 2.63'de verilmiştir.

Minimize edilecek fonksiyon,

$$f(\vec{x}) = \left(\frac{1}{6{,}931} - \frac{T_d T_b}{T_a T_f}\right), \tag{2.62}$$

Değişken aralıkları,

$$12 \leq T_a, T_b, T_d, T_f \leq 60 \tag{2.63}$$

2.4.6. Himmelblau'nun problemi

Himmelblau bu konudaki fikri ortaya atan kişidir. Daha önce, problemin amaç fonksiyonunun minimize edilmesine ihtiyaç duyan çok sayıda evrimsel algoritma tabanlı algoritma için bir ölçüt olarak kullanılmıştır. Bu problemde beş tasarım değişkeni (x_1, x_2, x_3, x_4, x_5), 3 doğrusal olmayan eşitsizlik kısıtı ve 8 sınır koşulu bulunmaktadır (Abouhabaga, 2021:157). Bu problemin formülleri Denklem 2.64 ile Denklem 2.73 arasında verilmiştir.

Minimize edilecek fonksiyon,

$$f(\vec{x}) = 5{,}3578547x_3^2 + 0{,}8356891x_1x_5 + 37{,}2932239x_1 - 40792{,}141, \tag{2.64}$$

Kısıtlayıcılar,

$$g_1(\vec{x}) = 85{,}334407 + 0{,}0056858x_2x_5 + 0{,}00026x_1x_4 - 0{,}0022053x_3x_5, \tag{2.65}$$

$$g_2(\vec{x}) = 80{,}51249 + 0{,}0071317x_2x_5 + 0{,}0029955x_1x_2 - 0{,}0021813x_3^2, \tag{2.66}$$

$$g_3(\vec{x}) = 9{,}300961 + 0{,}0047026x_3x_5 + 0{,}0012547x_1x_3 + 0{,}0019085x_3x_4, \tag{2.67}$$

Kısıtlayıcı aralıkları,

$$0 \leq g_1(\vec{x}) \leq 92, \tag{2.68}$$

$$90 \leq g_2(\vec{x}) \leq 110, \tag{2.69}$$

$$20 \leq g_3(\vec{x}) \leq 25, \tag{2.70}$$

Değişken aralıkları,

$$78 \leq x_1 \leq 102, \tag{2.71}$$

$$33 \leq x_2 \leq 45, \tag{2.72}$$

$$27 \leq x_3, x_4, x_5 \leq 45 \tag{2.73}$$

2.4.7. Üç çubuk makas tasarımı

Üç çubuk kafes kiriş tasarım optimizasyon problemi ilk olarak Ray ve Saini (Ray, 2001:742) tarafından ortaya konulmuştur. Buna göre üç çubuğun Şekil 2.4.b'deki gibi yerleştirilmesi istenmektedir. Bu konumdaki çubukların ağırlığının minimize edilmesi amaçlanmaktadır.

Bu bir kısıtlayıcı optimizasyon problemidir. Bu problemde iki tasarım parametresi (x_1, x_2), çubuklar arası uzaklık (L) ve üç kısıtlayıcı fonksiyon vardır (Yıldırım, 2018:1). Bu problemin formülleri Denklem 2.74 ile Denklem 2.79 arasında verilmiştir.

Minimize edilecek fonksiyon,

$$f(\vec{x}) = \left(2\sqrt{2x_1} + x_2\right) * L, \tag{2.74}$$

Kısıtlayıcılar,

$$g_1(\vec{x}) = \frac{\sqrt{2}x_1 + x_2}{\sqrt{2}x_1^2 + 2x_1x_2} P - \sigma \leq 0, \tag{2.75}$$

$$g_2(\vec{x}) = \frac{x_2}{\sqrt{2}x_1^2 + 2x_1x_2} P - \sigma \leq 0, \tag{2.76}$$

$$g_3(\vec{x}) = \frac{1}{x_1 + \sqrt{2}x_2} P - \sigma \leq 0, \tag{2.77}$$

Değişken aralıkları,

$$0 \leq x1, x2 \leq 1, \tag{2.78}$$

Sabitler,

$$L = 100\ cm, P = 2\frac{KN}{cm2}\ ve\ \sigma = 2\frac{KN}{cm2} \tag{2.79}$$

2.4.8. Kademeli konsol kiriş tasarımı

Konsol kiriş tasarım problemi bağlamında amaç, dikey uç kuvveti altında kademeli konsol kirişin hacmini en aza indirmektir (Dhadwal, 2013:344). Her bir basamağın dikdörtgen kesitinin yüksekliği ve genişliği tasarım değişkenlerini oluştururken, problemin kendisi 10 tasarım değişkeni ve 11 kısıttan oluşmaktadır (Gandomi, 2011:8). Bu problemin formülleri Denklem 2.80 ile Denklem 2.91 arasında verilmiştir.

Minimize edilecek fonksiyon,

$$f(\vec{x}) = (b_1h_1 + b_2h_2 + b_3h_3 + b_4h_4 + b_5h_5) * L, \quad (2.80)$$

Kısıtlayıcılar,

$$g_1(\vec{x}) = \frac{6PL}{b_5h_5^2} - \sigma_d \leq 0, \quad (2.81)$$

$$g_2(\vec{x}) = \frac{6P*2L}{b_4h_4^2} - \sigma_d \leq 0, \quad (2.82)$$

$$g_3(\vec{x}) = \frac{6P*3L}{b_3h_3^2} - \sigma_d \leq 0, \quad (2.83)$$

$$g_4(\vec{x}) = \frac{6P*4L}{b_2h_2^2} - \sigma_d \leq 0, \quad (2.84)$$

$$g_5(\vec{x}) = \frac{6P*5L}{b_1h_1^2} - \sigma_d \leq 0, \quad (2.85)$$

Konsol kiriş ucunun sapması, limit sapmadan (Δ_{max}) daha küçük olmalıdır:

$$g_6(\vec{x}) = \frac{PL^3}{3E}\left(\frac{125}{L}\right) - \Delta_{max} \leq 0, \quad (2.86)$$

Her bir segmentin enine kesitinin yüksekliği ve genişliği arasındaki oran 20'den fazla olmamalıdır:

$$g_7(\vec{x}) = \frac{h_5}{b_5} - 20 \leq 0, \quad (2.87)$$

$$g_8(\vec{x}) = \frac{h_4}{b_4} - 20 \leq 0, \quad (2.88)$$

$$g_9(\vec{x}) = \frac{h_3}{b_3} - 20 \leq 0, \quad (2.89)$$

$$g_{10}(\vec{x}) = \frac{h_2}{b_2} - 20 \leq 0, \quad (2.90)$$

$$g_{11}(\vec{x}) = \frac{h_1}{b_1} - 20 \leq 0 \quad (2.91)$$

σ_d=14000, Δ_{max}=2,7, E=2*107 değerlerine sahiptir. Tasarım uzayı altı ayrık değişken içerir (konsol kirişin ilk üç bölümünün kesit boyutları) ve aşağıdaki gibi tanımlanır: b1: {1,2,3,4,5}, b2, b3: {2.4,2.6,2.8,3.1}, h1, h2: {45.0,50.0,55.0,60.0}, h3: {30, 31, ...,65}, 1≤b4, b5≤5 ve 30≤h4, h5≤65. Tüm boyutlar cm cinsinden ifade edilmiştir (Gandomi, 2011:9).

2.4.9. Çok diskli debriyaj fren tasarımı

Çok diskli bir debriyaj freni, enerjiyi bir noktadan diğerine aktarmak için bir dizi disk kullanan bir cihazdır. Bu problemin amacı, sekiz doğrusal olmayan kısıtı yerine getirirken debriyaj freninin toplam kütlesini en aza indirmektir (Rao, 2020:4). Bu problemde beş karar değişkeni vardır: x = (ri, ro, t, F, Z), burada ri ∈ [60,80] (bir adımda) mm cinsinden iç yarıçap, ro ∈ [91,110] (bir adımda) mm cinsinden dış yarıçap, t ∈ [1,3] (0. 5) mm cinsinden disk kalınlığı, F ∈ [600,1000] (10'luk adımlarla) N cinsinden harekete geçirici kuvvet ve Z ∈ [2,10] (1'lik adımlarla) sürtünme yüzeylerinin (veya disklerin) sayısıdır (Deb, 2006:1631). Optimizasyon problemi Denklem 2.92 ile Denklem 2.119 arasında formüle edilmiştir:

Minimize edilecek fonksiyonlar,

$$f_1(\vec{x}) = \pi(x_2^2 - x_1^2)x_3(x_5 + 1)\rho, \tag{2.92}$$

$$f_2(\vec{x}) = T = \frac{I_{Zw}}{M_h + M_f}, \tag{2.93}$$

Kısıtlayıcılar,

$$g_1(\vec{x}) = x_2 - x_1 - \Delta R \geq 0, \tag{2.94}$$

$$g_2(\vec{x}) = L_{max} - (x_5 + 1)(x_3 + \delta) \geq 0, \tag{2.95}$$

$$g_3(\vec{x}) = p_{max} - p_{rz} \geq 0, \tag{2.96}$$

$$g_4(\vec{x}) = p_{max}V_{sr,max} - p_{rz}V_{sr} \geq 0, \tag{2.97}$$

$$g_5(\vec{x}) = V_{sr,max} - V_{sr} \geq 0, \tag{2.98}$$

$$g_6(\vec{x}) = M_h - sM_s \geq 0, \tag{2.99}$$

$$g_7(\vec{x}) = T \geq 0, \tag{2.100}$$

$$g_8(\vec{x}) = T_{max} - T \geq 0, \tag{2.101}$$

Değişken aralıkları,

$$r_{i,min} \leq x_1 \leq r_{i,max}, \tag{2.102}$$

$$r_{o,min} \leq x_2 \leq r_{o,max}, \tag{2.103}$$

$$t_{min} \leq x_3 \leq t_{max}, \tag{2.104}$$

$$0 \leq x_4 \leq F_{max}, \tag{2.105}$$

$$2 \leq x_5 \leq Z_{max} \tag{2.106}$$

Parametreler,

$$M_h = \frac{2}{3}\mu x_4 x_5 \frac{x_2^3 - x_1^3}{x_2^2 - x_1^2}\ N \cdot mm, \tag{2.107}$$

$$\omega = \frac{\pi n}{30}\frac{rad}{s}, \tag{2.108}$$

$$A = \pi(x_2^2 - x_1^2)mm2, \tag{2.109}$$

$$p_{rz} = \frac{x_4}{A}\frac{N}{mm2}, \tag{2.110}$$

$$V_{sr} = \frac{\pi R_{sr} n}{30}\frac{mm}{s}, \tag{2.111}$$

$$R_{sr} = \frac{2}{3}\frac{x_2^3 - x_1^3}{x_2^2 - x_1^2}\ mm, \tag{2.112}$$

$$\Delta R = 20mm, \tag{2.113}$$

$$L_{max} = 30mm,\ \mu = 0.5, \tag{2.114}$$

$$p_{max} = 1MPa,\ \rho = 0.0000078\frac{kg}{mm3},\ V_{sr,max} = \frac{10m}{s}, \tag{2.115}$$

$$s = 1.5,\ T_{max} = 15s,\ n = 250rpm,\ M_s = 40Nm, \tag{2.116}$$

$$M_f = 3Nm,\ I_z = 55kg \cdot m2,\ \delta = 0.5mm,\ r_{i,min} = 60mm, \tag{2.117}$$

$$r_{i,max} = 80mm,\ r_{o,min} = 90mm,\ r_{o,max} = 110\ mm, \tag{2.118}$$

$$t_{min} = 1.5mm,\ t_{max} = 3mm,\ F_{max} = 1{,}000\ N,\ Z_{max} = 9 \tag{2.119}$$

2.4.10. Hidrodinamik baskı yatağı tasarımı

Hidrodinamik Baskı Yatağı Tasarımı problemi, en önemli ve köklü klasik makine mühendisliği tasarım problemlerinden biri olarak kabul edilir. Amacı, çerçevenin güç kaybını en aza indirmektir. Tasarım dört değişkenden oluşmaktadır: basamak yarıçapı (R), girinti yarıçapı (R_0), viskozite (μ) ve akış hızı (Q) (Rather, 2021:524). Bu problemin formülleri Denklem 2.120 ile Denklem 2.132 arasında verilmiştir.

$$\vec{x} = [\ RRo\mu Q\ ], \tag{2.120}$$

Minimize edilecek fonksiyon,

$$f(\vec{x}) = \frac{QP_0}{0{,}7} + E_f, \tag{2.121}$$

Kısıtlamalar,

$$s_1(\vec{x}) = W - W_s \geq 0, \quad (2.122)$$

$$s_2(\vec{x}) = P_{max} - P_0 \geq 0, \quad (2.123)$$

$$s_3(\vec{x}) = \Delta T_{max} - \Delta T \geq 0, \quad (2.124)$$

$$s_4(\vec{x}) = h - h_{min} \geq 0, \quad (2.125)$$

$$s_5(\vec{x}) = R - R_0 \geq 0, \quad (2.126)$$

$$s_6(\vec{x}) = 0{,}001 - \frac{y}{gP_0}\left(\frac{Q}{2\pi R_h}\right) \geq 0, \quad (2.127)$$

$$s_7(\vec{x}) = 5{,}000 - \frac{W}{\pi\left(R^2 R_0^2\right)} \geq 0 \quad (2.128)$$

Karar değişkeni aralık değerleri,

$$1 \leq R \leq 16, \quad (2.129)$$

$$1 \leq R_0 \leq 16, \quad (2.130)$$

$$1 * 10^{-6} \leq \mu \leq 16 * 10^{-6}, \quad (2.131)$$

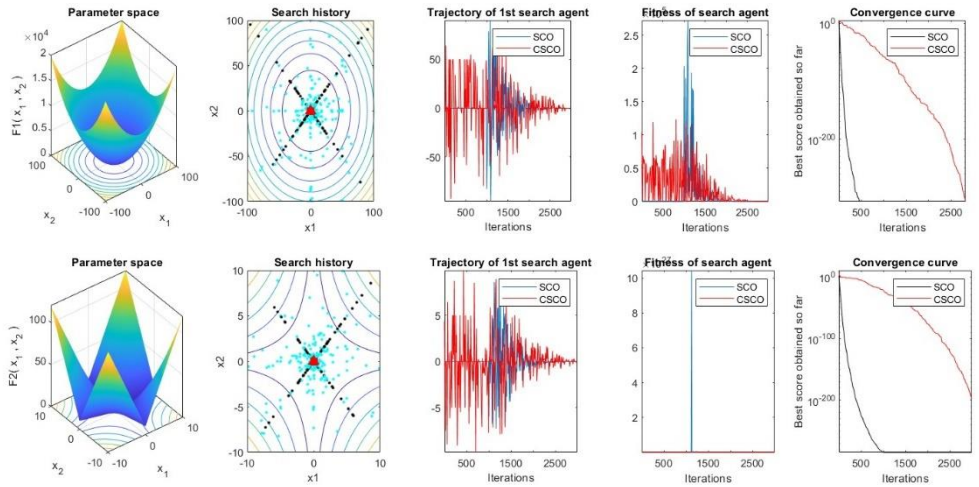
$$1 \leq Q \leq 16 \quad (2.132)$$

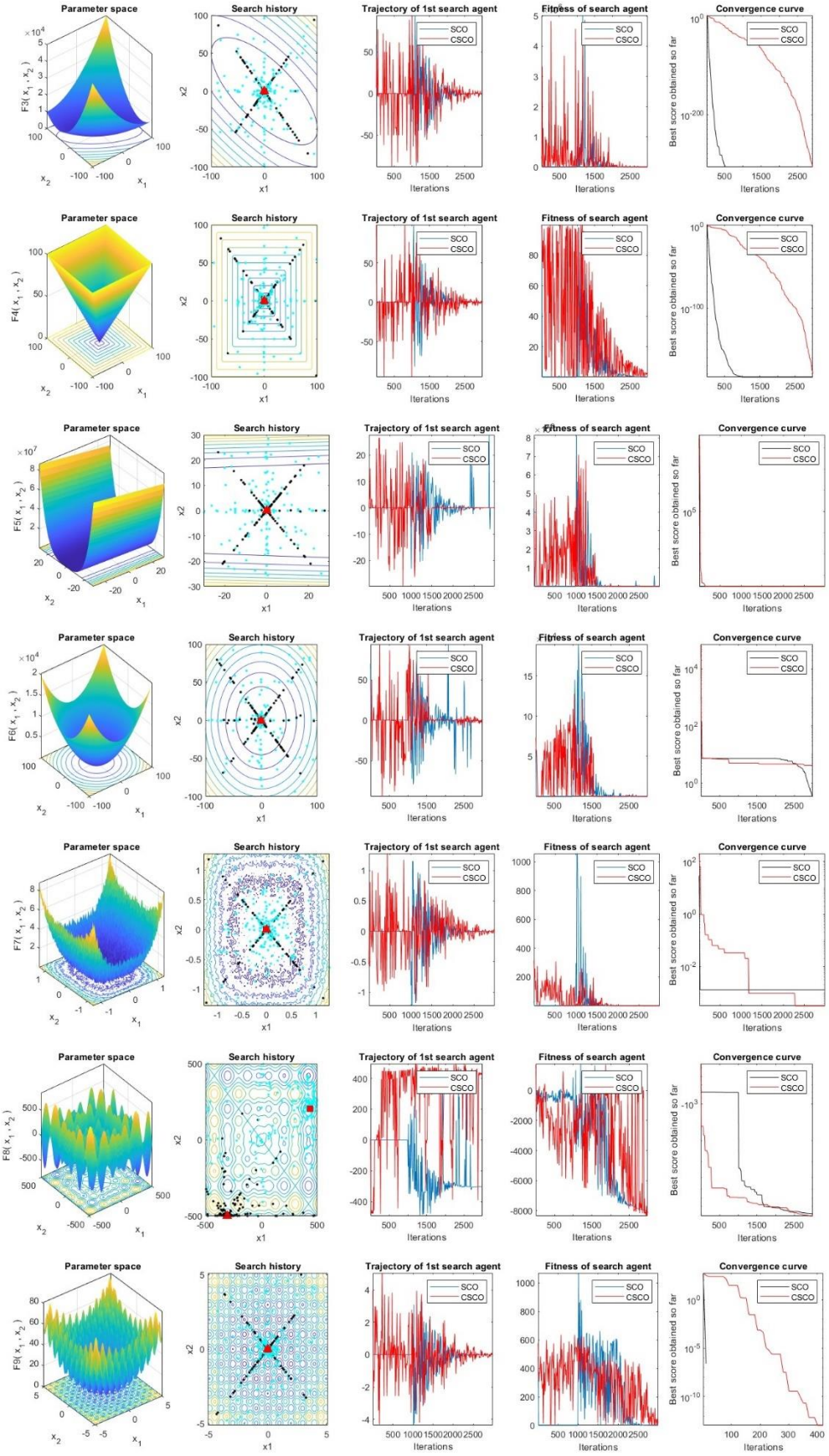
3.DENEYSEL SONUÇLAR

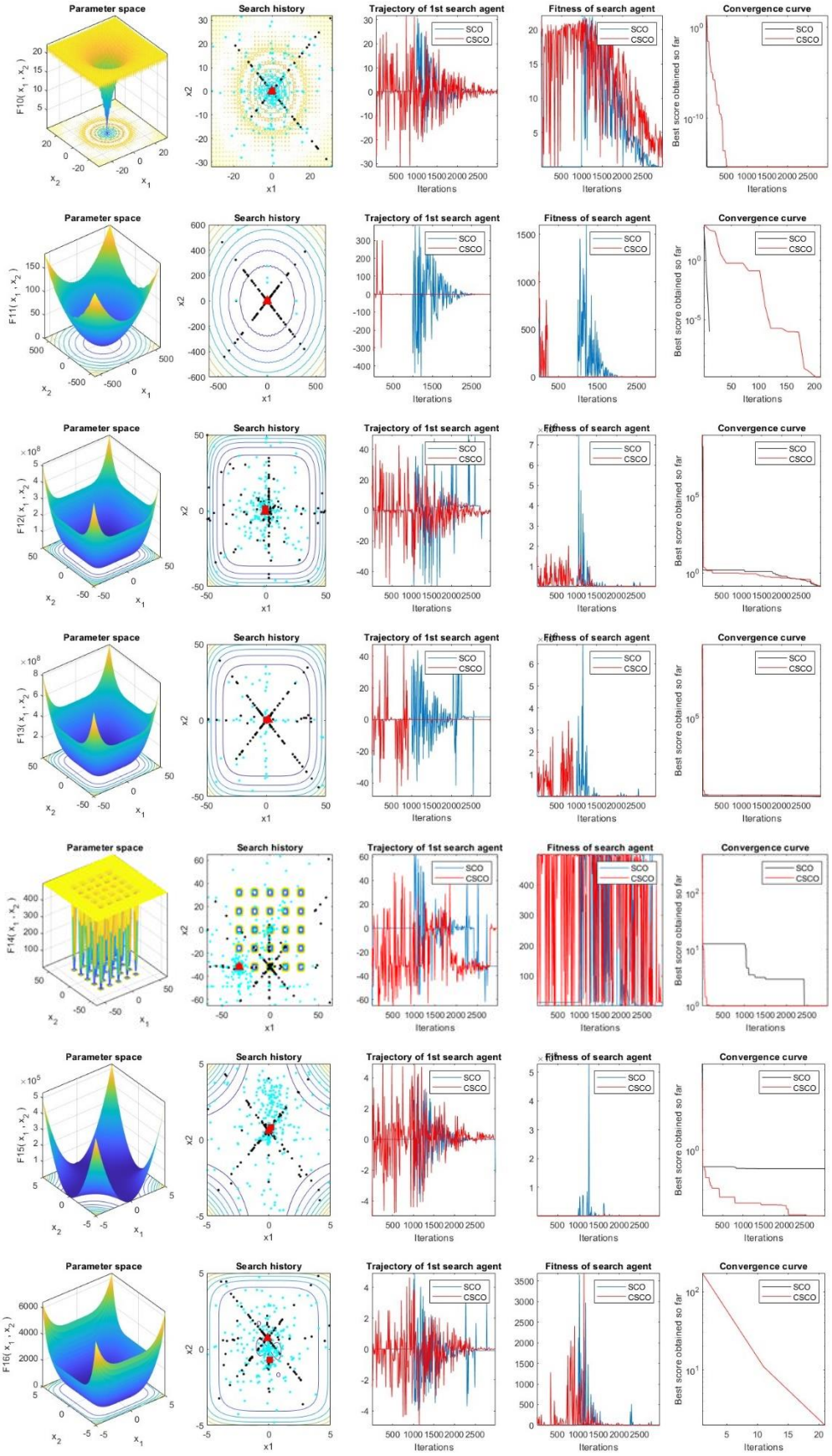
3.1. Kıyaslama Fonksiyonları Test Sonuçları

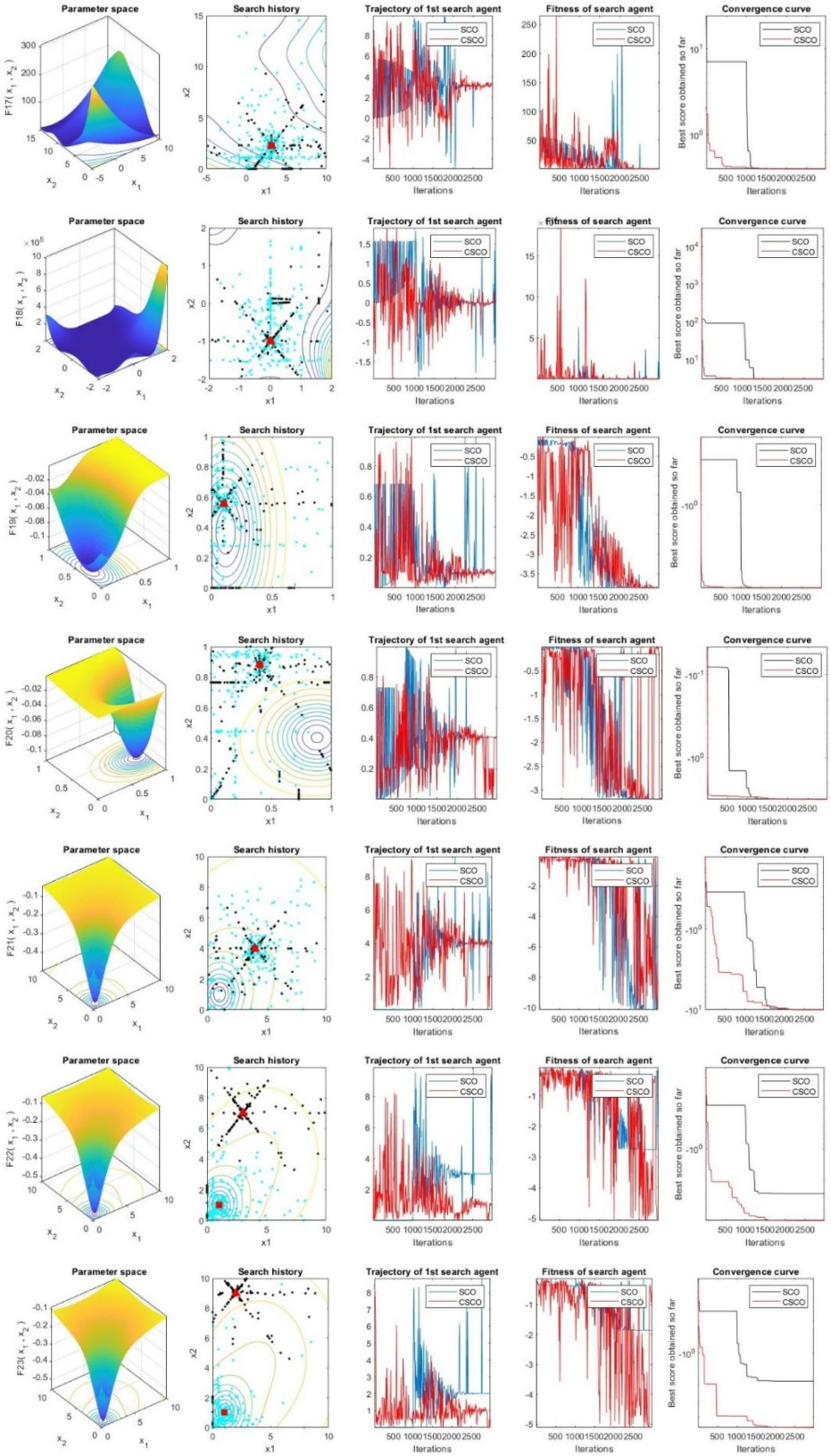
Analizin ilk aşamasında, her iki algoritmanın başarımını değerlendirmek için kıyaslama fonksiyonlarının iki boyutlu değerlendirmeleri yapılmıştır.

Analiz süreçleri, arama geçmişi analizi, yörünge analizi, arama ajanı uygunluk değeri analizi ve yakınsama analizi dahil olmak üzere çok sayıda metodolojinin kullanılmasını gerektirmektedir. Arama geçmişi analizi, optimizasyon aşaması boyunca arama uzayı içindeki potansiyel çözümlerin ilerleyişini izler. Yörünge analizi, adayın konum vektörünün başlangıç bileşeninin zamansal evrimini tanımlar. Arama ajanı uygunluk değeri analizi, optimizasyon iterasyonları boyunca adayın uygunluk değerindeki değişimleri izler. Yakınsama analizi, aday çözümlerin optimum çözüme doğru ilerlemesini değerlendirir. Şekil 3.1, kıyaslama fonksiyonları için karşılaştırmalı analiz sonuçlarına kapsamlı bir genel bakış sunmaktadır. Her bir fonksiyon için 5 adet grafik bulunmaktadır. İlk sıradaki grafik her fonksiyon için parametre uzayını göstermektedir. İkinci sıradaki şekilde ise arama geçmişi bulunmaktadır. Bu grafikte siyah renkli noktalar SCO algoritmasının arama geçmişini, turkuaz renkli noktalar ise CSCO algoritmasının arama geçmişini göstermektedir. Kırmızı renkli üçgen şekil SCO algoritmasının çözüm noktasına en yakın bulduğu sonucu, kırmızı renkli kare şekil ise CSCO algoritmasının çözüm noktasına en yakın bulduğu sonuç noktasını göstermektedir. 3. Sıradaki grafik ise aday çözüm noktalarının iterasyon sayısına bağlı izlediği yörüngeleri göstermektedir. Bu grafikte mavi renk SCO algoritmasını, kırmızı renk ise CSCO algoritmasını tanımlamaktadır. 4. Sıradaki grafik ise aday çözümün iterasyon sayısına bağlı uygunluk verilerini içermektedir. 5. Sıradaki grafik ise aday çözümlerin iterasyon sayısına bağlı yakınsama eğrilerini göstermektedir.




Parameter space
Search history
Trajectory of 1st search agent
Fitness of search agent
Convergence curve
SCO
CSCO
Iterations
Best score obtained so far
x1
x2


Parameter space
Search history
Trajectory of 1st search agent
Fitness of search agent
Convergence curve
SCO
CSCO
Iterations
Best score obtained so far
x1
x2



Şekil 3.1. SCO ve CSCO Algoritmalarının Analiz Sonuçları: Parameter Space: Parametre Uzayı, Search history: Arama Geçmişi, Trajectory of 1st Agent: Aday Çözümlerin Yörüngesi, Fitness of Search Agent: Aday Çözümlerin Uygunluğu, Convergence Curve: Yakınsama Eğrisi

23 farklı kıyaslama fonksiyonu için analiz edildikten sonra, CSCO algoritmasının orijinal SCO algoritmasından daha iyi başarım gösterdiği gözlemlenmiştir. İlk 4 fonksiyonun yakınsama eğrilerine bakıldığında SCO algoritmasının CSCO algoritmasına göre başarımının daha iyi olduğu gözlemlenmiştir. SCO algoritmasının bu başarımının arkasında çözüm noktasının (0,0) noktası olduğunda bu çözüme çok hızlı yakınsama yatkınlığıdır.

5. ve 6. fonksiyonda ise çözüm noktasının (0 ,0) olmasına rağmen CSCO algoritmasının başarımının SCO algoritmasının başarımına eşitlendiği gözlemlenmektedir. Bu sonuç SCO algoritmasının hızlı yakınsama eğilimine rağmen CSCO algoritmasının SCO algoritmasının başarımına yetişebileceğini göstermiştir.

7. fonksiyonda ise (0,0) noktasına hızlı yaklaşımın negatif etkileri gözlemlenmektedir. SCO algoritması yerel optimum noktası olan (0,0) noktasına çok hızlı yaklaşmış ama yerel optimum noktasında kalmıştır. CSCO algoritması ise çözüm noktasına ulaşmayı başarmıştır.

8. fonksiyonda SCO algoritmasının en büyük dezavantajlarından biri olan ilk 1000 iterasyonda keşif evresindeki iç kısıtlamaları açıkça gözükmektedir. CSCO algoritmasının bu kısıtlamalara takılmadan yakınsamaya devam ettiği gözlemlenmektedir.

9. fonksiyonda SCO algoritmasının iterasyonlarını tamamlayamadığı, CSCO algoritmasının 400. İterasyonda sonuca ulaştığı gözlemlenmektedir.

10. fonksiyonda beklenildiği gibi SCO algoritmasının (0,0) çözüm noktasına hızla ulaştığı gözlemlenmektedir.

11. fonksiyonda SCO algoritmasının sonuca ulaşamadığı, CSCO algoritmasının ise 200 iterasyonda sonuca ulaştığı gözlemlenmiştir.

12. ve 13. fonksiyonlarda SCO ve CSCO algoritmalarının başarımlarının aynı olduğu söylenebilir.

Özellikle, 14. fonksiyonun yakınsama eğrisi incelendiğinde, SCO algoritmasının iç kısıtlamalar nedeniyle keşif aşamasının ilk 1000 iterasyonu boyunca ilerleme kaydedemediği görülmüştür. İlk 1000 iterasyondan sonra, algoritma yerel optimumda sıkışıp kalmış ve 2400. iterasyona kadar bu şekilde kalmıştır. Ancak bu noktadan sonra küresel çözüme doğru ilerleyebilmiştir. Buna karşılık, CSCO algoritması SCO'nun bu sınırlamalarının üstesinden gelmekte ve yerel optimumlardan kaçınarak ve küresel çözüme yakınsayarak daha verimli bir başarım sergilemektedir.

15. Fonksiyonda SCO algoritmasının ilk 1000 iterasyonda iç kısıtlamalara takıldığı ve daha sonra sömürme aşamasında ise yerel minimum noktasında takılı kalıp çözüm noktasına ulaşamadığı görülmektedir. CSCO algoritması keşif aşaması ve sömürme aşamasında sürekli çözüm noktasına yaklaşıp çözüm noktasına ulaştığı gözlemlenmiştir.

16. fonksiyonda CSCO algoritmasının yaklaşık 20 iterasyonda çözüm noktasına ulaştığı gözlemlenmektedir.

17. fonksiyonda SCO algoritmasının ilk 1000 iterasyon olan keşif aşamasında iç kısıtlamalarına takıldığı ve 1000 iterasyon boyunca çözüm noktasına yaklaşımda bulunamadığı gözlemlenmektedir. CSCO algoritmasının başarımının oldukça yüksek olduğu gözlemlenmektedir.

18'inci fonksiyon için yakınsama eğrisi, SCO algoritmasının içsel olarak sınırlı olduğunu ve bunun da başarımını olumsuz etkilediğini göstermektedir. SCO algoritmasının 19-23 arasındaki fonksiyonlar için çözüm noktası orijinde olan problemlerde orijinal SCO algoritmasının hızlı sonuçlar ürettiği bilinmektedir. Ancak bu durumlarda bile CSCO algoritması daha iyi başarım göstermektedir. Analiz, CSCO algoritmasının özellikle keşif aşamasında ve yerel optimumlardan kaçınmada orijinal SCO algoritmasına göre önemli bir avantaja sahip olduğunu göstermektedir. Bu sonuçlar, CSCO algoritmasının genel optimizasyonda daha verimli ve etkili olduğunu göstermektedir. SCO ve CSCO algoritmalarının 23 kıyaslama fonksiyonu üzerinde 30 kez çalıştırılarak yapılan karşılaştırmalı analizi Tablo 3.1'de özetlenmiştir.

Deneysel sonuçlar, SCO'ya kıyasla CSCO'nun başarımında kayda değer bir iyileşme olduğunu göstermektedir. Özellikle, CSCO algoritması en iyi metrik değerlerde SCO'dan daha iyi başarım göstermekte ve bu da %73,91'lik bir iyileşmeyi yansıtmaktadır. Benzer şekilde, en kötü metrik değerlerle ilgili olarak CSCO, SCO algoritmasına kıyasla %56,52'lik bir iyileşme sergilemektedir. Ayrıca, CSCO 13 fonksiyonda SCO'dan %56,52 daha etkili bir medyan metrik başarımı ve 10 işlevde %43,48 daha etkili bir ortalama metrik başarımı sergilemektedir. Ayrıca, SCO'ya kıyasla %69,57 daha düşük standart sapma ile kanıtlandığı üzere, CSCO'nun test fonksiyonları arasında daha düzgün ve istikrarlı bir davranış sergilemesi de dikkat çekicidir. Bu gözlemler toplu olarak CSCO'nun test fonksiyonlarının optimizasyonunda SCO'dan daha yetkin olduğunu teyit etmektedir.

Tablo 3.1. SCO ve CSCO Algoritmalarının 30 Çalıştırmada İstatiksel Sonuçları

Fonk No	Orijinal SCO					CSCO				
	En İyi	En Kötü	Medyan	Ortalama	SS	En İyi	En Kötü	Medyan	Ortalama	SS
1	0,0E+00	0,0E+00	0,0E+00	0,0E+00	0,0E+00	0,0E+00	0,0E+00	0,0E+00	0,0E+00	0,0E+00
2	1,9E-307	2,0E-250	9,7E-276	1,3E-251	0,0E+00	0,0E+00	3,0E-181	1,8E-202	1,0E-182	0,0E+00
3	0,0E+00	3,5E-242	0,0E+00	1,2E-243	0,0E+00	0,0E+00	0,0E+00	0,0E+00	0,0E+00	0,0E+00
4	5,5E-184	1,5E-35	9,4E-137	4,8E-37	2,7E-36	0,0E+00	2,9E-171	2,2E-196	1,1E-172	0,0E+00
5	2,9E+01	2,9E+01	2,9E+01	2,9E+01	1,1E-01	2,9E+01	2,9E+01	2,9E+01	2,9E+01	9,7E-02
6	1,5E-01	6,5E-01	2,1E-01	2,5E-01	1,1E-01	2,4E+00	5,9E+00	4,2E+00	4,2E+00	8,6E-01
7	2,0E-05	3,0E-03	3,1E-04	4,6E-04	5,4E-04	7,6E-05	1,0E-02	1,3E-03	1,9E-03	2,0E-03
8	-1,0E+04	-7,3E+03	-8,2E+03	-8,3E+03	7,0E+02	-8,6E+03	-5,8E+03	-7,4E+03	-7,3E+03	7,4E+02
9	0,0E+00	0,0E+00	0,0E+00	0,0E+00	0,0E+00	0,0E+00	0,0E+00	0,0E+00	0,0E+00	0,0E+00
10	8,9E-16	8,9E-16	8,9E-16	8,9E-16	4,0E-31	8,9E-16	8,9E-16	8,9E-16	8,9E-16	4,0E-31
11	0,0E+00	0,0E+00	0,0E+00	0,0E+00	0,0E+00	0,0E+00	0,0E+00	0,0E+00	0,0E+00	0,0E+00
12	2,0E-02	2,6E-01	5,4E-02	7,3E-02	5,1E-02	9,6E-02	3,5E-01	2,1E-01	2,1E-01	6,9E-02
13	3,3E-01	3,0E+00	1,5E+00	1,5E+00	7,0E-01	1,8E+00	3,0E+00	2,9E+00	2,8E+00	2,4E-01
14	1,0E+00	2,0E+00	1,0E+00	1,2E+00	4,0E-01	1,0E+00	1,0E+00	1,0E+00	1,0E+00	4,5E-16
15	3,5E-04	1,2E-01	1,4E-03	7,1E-03	2,3E-02	3,5E-04	2,5E-03	8,7E-04	1,0E-03	5,6E-04
16	-1,0E+00	-1,0E+00	-1,0E+00	-1,0E+00	0,0E+00	-1,0E+00	-1,0E+00	-1,0E+00	-1,0E+00	0,0E+00
17	4,0E-01	4,0E-01	4,0E-01	4,0E-01	0,0E+00	4,0E-01	4,0E-01	4,0E-01	4,0E-01	1,8E-05
18	3,0E+00	3,0E+00	3,0E+00	3,0E+00	0,0E+00	3,0E+00	3,0E+00	3,0E+00	3,0E+00	0,0E+00
19	-3,9E+00	-3,9E+00	-3,9E+00	-3,9E+00	0,0E+00	-3,9E+00	-3,9E+00	-3,9E+00	-3,9E+00	4,3E-04
20	-3,3E+00	-3,1E+00	-3,3E+00	-3,3E+00	7,7E-02	-3,3E+00	-3,1E+00	-3,3E+00	-3,3E+00	6,7E-02
21	-1,0E+01	-2,6E+00	-1,0E+01	-8,6E+00	2,6E+00	-1,0E+01	-2,6E+00	-5,1E+00	-6,2E+00	2,5E+00
22	-1,0E+01	-2,8E+00	-1,0E+01	-8,3E+00	2,9E+00	-1,0E+01	-2,8E+00	-5,1E+00	-6,8E+00	2,6E+00
23	-1,1E+01	-5,1E+00	-1,1E+01	-9,3E+00	2,3E+00	-1,1E+01	-1,7E+00	-5,1E+00	-6,4E+00	2,9E+00

3.2. Mühendislik Tasarım Problemleri Test Sonuçları

Bu bölümde CSCO algoritmasının mühendislik tasarım problemlerini ele almadaki başarımının bir değerlendirmesi sunulmaktadır. Mevcut literatürden sıkıştırma yayı tasarımı, kaynaklı kiriş tasarımı, basınçlı kap tasarımı, hız düşürücü tasarımı, dişli katarı tasarımı, Himmelblau'nun problemi, üç çubuk makas tasarımı, kademeli konsol kiriş tasarımı, çok diskli debriyaj fren tasarımı, hidrodinamik baskı yatağı tasarımı olmak üzere on test problemi seçilmiştir. Her bir tasarım problemi için öncelikle CSCO algoritmasının orijinal SCO ile bir karşılaştırması sunulmuş, ardından CSCO algoritmasının literatürdeki popüler sezgisel yöntemlerle bir karşılaştırması yapılmıştır.

3.2.1. Orijinal SCO algoritması ile CSCO algoritmasının karşılaştırılması

CSCO algoritmasının etkinliği, literatürden on mühendislik tasarım problemi analiz edilerek değerlendirilmiştir. Bu analizler yapılırken CSCO algoritmasının başarımı orijinal SCO algoritması ile karşılaştırılmıştır. Ayrıca literatürde yaygın olarak kullanılan GA, PSO,

DE, ABC, GWO, GSA, BBO, SCA ve SSA algoritmalarının analiz sonuçları da göz önünde bulundurularak geliştirilen CSCO algoritmasının mühendislik tasarım problemlerinin çözümündeki başarımı değerlendirilmiştir. Bu algoritmalar yaygın olarak kullanıldıklarından, kıyaslama fonksiyonlarında ve mühendislik tasarım problemlerinde iyi bir başarım sergiledikleri bilindiğinden dolayı seçilmişlerdir. Karşılaştırma yapılırken SCO algoritmasının alpha değeri 1000, b değeri 2,4 alınmıştır. GWO algoritmasının a değeri 2'den 0'a giderek azalacak şekilde verilmiştir. PSO algoritmasının C_1 değeri 2, C_2 değeri 2 ve w ağırlık değerleri [0,9-0,4] şeklinde ayarlanmıştır. SSA algoritmasının pozisyon güncelleme olasılığı 0,5 olacak şekilde verilmiştir. GSA algoritmasının Alpha değeri 20, G_0 değeri 100, Rnorm değeri 2 ve Rpower değeri 1 olarak ayarlanmıştır. Her algoritma, çalışma başına maksimum 100 iterasyon olmak üzere 10 bağımsız çalışmaya tabi tutulmuştur.

Daha sonra, bu çalıştırmalarla elde edilen optimum çözümler karşılaştırmalı bir analize tabi tutulmuştur. Şekil 3.2.a, her iki algoritma kullanılarak elde edilen baskı yayı tasarım probleminin çözümleri için elde edilen yakınsama eğrilerini göstermektedir. CSCO algoritmasının başlangıçta karşılaştığı dezavantaja rağmen, her iki algoritma da birbirine yakınsayan yakınsama eğrilerinden de anlaşılacağı üzere karşılaştırılabilir bir başarım göstermiştir.

Şekil 3.2.b, SCO ve CSCO'nun kaynaklı kiriş tasarımını çözmedeki başarımlarının karşılaştırmasını göstermektedir; CSCO, SCO'ya göre daha üstün başarım sergilemektedir. Orijinal SCO'nun keşif aşamasındaki eksiklikleri, başarımı üzerinde zararlı bir etkiye sahiptir ve yakınsamada önemli bir gecikmeye neden olur.

Şekil 3.2.c'de basınçlı kap tasarım probleminin çözümünde SCO ve CSCO'nun başarımlarının karşılaştırılması gösterilmektedir. Sonuçlar, orijinal SCO'nun doğal sınırlamalarının başarımını olumsuz etkilediğini göstermektedir.

Şekil 3.2.d'de hız düşürücü tasarım probleminin çözümünde SCO ve CSCO'nun başarımlarının karşılaştırılması gösterilmektedir. Sonuçlar, SCO algoritmasının keşif bölümünde iç kısıtlamalarına takılı kalmasından dolayı çözüm noktasına ulaşmakta geç kaldığını göstermektedir.

Şekil 3.2.e'de dişli katarı tasarım probleminin çözümünde SCO ve CSCO'nun başarımlarının karşılaştırılması gösterilmektedir. Sonuçlar, CSCO algoritmasının başarımının SCO algoritmasına göre daha üstün olduğunu göstermektedir.

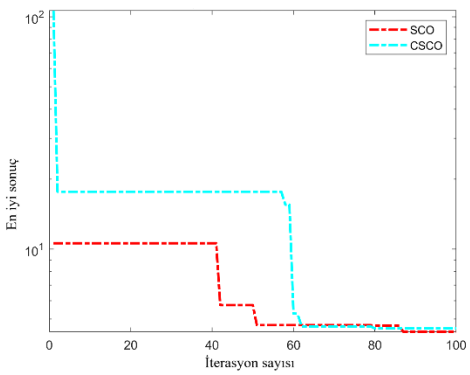
Şekil 3.2.f'de Himmelblau'nun probleminin çözümünde SCO ve CSCO'nun başarımlarının karşılaştırılması gösterilmektedir. Sonuçlar, SCO algoritmasının neredeyse 35 iterasyon boyunca iç kısıtlamalarına takılarak yakınsama gerçekleştiremediği ve bu yüzden çözüm noktasına ulaşmakta başarımının CSCO algoritmasının çok gerisinde kaldığı gözlemlenmektedir.

Şekil 3.2.g'de üç çubuk makas tasarım probleminin çözümünde SCO ve CSCO'nun başarımlarının karşılaştırılması gösterilmektedir. Sonuçlar, CSCO algoritmasının başarımının SCO algoritmasına göre daha üstün olduğu gözlemlenmektedir.

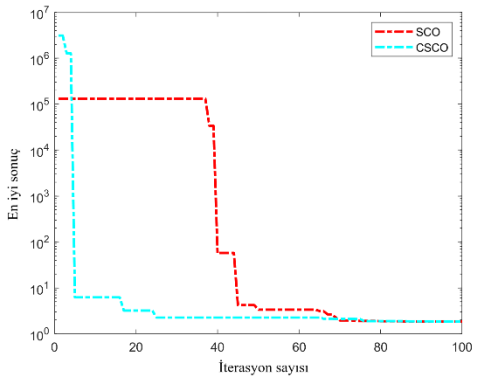
Şekil 3.2.h'de kademeli konsol kiriş tasarım probleminin çözümünde SCO ve CSCO'nun başarımlarının karşılaştırılması gösterilmektedir. Sonuçlar, SCO algoritmasının başarımının kötü olmadığını söyleyebilmemize rağmen CSCO algoritmasının başarımının daha üstün olduğu net bir şekilde gözükmektedir.

Şekil 3.2.i'de çok diskli debriyaj fren tasarım probleminin çözümünde SCO ve CSCO'nun başarımlarının karşılaştırılması gösterilmektedir. Sonuçlar, SCO algoritmasının keşif aşamasındaki iç kısıtlara takılması başarımını önemli ölçüde etkileyerek çözüm noktasına ulaşmasında gecikmesine neden olmaktadır.

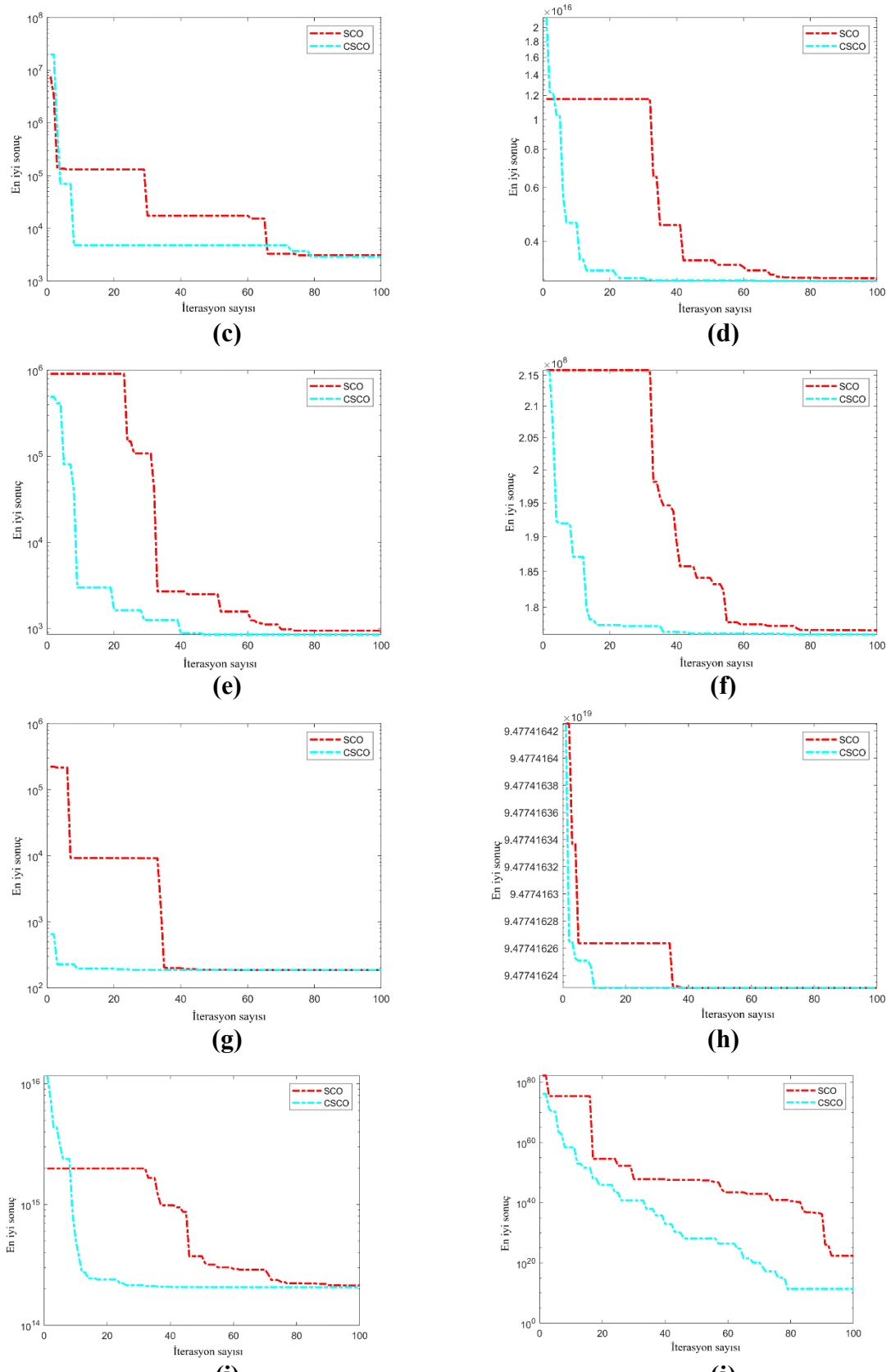
Şekil 3.2.j'de hidrodinamik baskı yatağı tasarım probleminin çözümünde SCO ve CSCO'nun başarımlarının karşılaştırılması gösterilmektedir. Sonuçlar, CSCO algoritmasının hem keşif hem sömürü aşamasında SCO algoritmasının başarımından daha üstün olduğu gözlemlenmektedir.



(a)



(b)



Şekil 3.2. Mühendislik Tasarım Problemleri İçin Orijinal SCO ve CSCO'nun Yakınsama Eğrileri: (a) Sıkıştırma Yayı, (b) Kaynaklı Kiriş, (c) Basınçlı Kap, (d) Hız Düşürücü, (e) Dişli Katarı, (f) Himmelblau'nun Problemi, (g) Üç Çubuk Makas, (h) Kademeli Konsol Kiriş, (i) Çok Diskli Debriyaj Fren, (j) Hidrodinamik Baskı Yatağı

Sonuç olarak 10 mühendislik tasarım problemlerinin yakınsama eğrileri analiz edildiğinde sıkıştırma yayı tasarım probleminde SCO ve CSCO algoritmalarının başarımlarının birbirine çok yakın olduğu, diğer 9 mühendislik tasarım problemlerinde ise CSCO algoritmasının başarımının daha üstün olduğu söylenebilir. CSCO algoritmasında uygulanan kaotik mutasyon stratejileri ve diğer geliştirmelerin CSCO algoritmasının başarımının artmasında önemli bir rol oynadığı yorumu yapılabilir.

3.2.2. CSCO algoritması ile sık kullanılan sezgisel algoritmaların karşılaştırılması

Önerilen CSCO algoritmasını on mühendislik tasarım problemi için popüler sezgisel algoritmalarla karşılaştırılmıştır. Bunlar Genetik Algoritma (GA) (Mirjalili ve Mirjalili, 2019), Parçacık Sürü Optimizasyonu (PSO) (Eberhart ve Kennedy, 1995), Farksal Gelişim (DE) (Kukkonen ve Lampinen, 2005) Algoritması, Yapay Arı Kolonisi (ABC) (Karaboğa vd., 2014), Gri Kurt Optimizasyonu (GWO) (Mirjalili vd., 2014), Yerçekimsel Arama Algoritması (GSA) (Rashedi vd., 2009), Biyocoğrafya Tabanlı Optimizasyon (BBO) (Simon, 2008) Algoritması, Sinüs Kosinüs Algoritması (SCA) (Mirjalili, 2016) ve Salp Sürü Algoritması (SSA)'dır (Mirjalili vd., 2017) .

Tablo 3.2, sıkıştırma yayı tasarım problemi için on çalıştırma boyunca her algoritma tarafından elde edilen optimum sonuçları göstermektedir. PSO, DE ve GWO algoritmaları, en iyi metrik değerlere göre değerlendirildiğinde benzerlerinden daha etkili olarak ayırt edilmektedir. En kötü metrik değerler için GA algoritması en yetkin algoritma olarak ortaya çıkmaktadır. PSO algoritması, medyan metrik değerlerine ilişkin güçlü başarımıyla dikkat çekmektedir. Ortalama metrik değerler ve standart sapma bağlamında, GWO algoritmasının başarımı benzersizdir ve bu yönlerden diğer algoritmaları geride bırakmaktadır. Tablo 3.3, CSCO ve sezgisel yöntemlerin koşma süresi sonuçlarını özetlemektedir.

Tablo 3.2. Sıkıştırma Yayı Tasarım Problemi İçin CSCO ve Diğer Sezgisel Yöntemlerin İstatistiksel Sonuçları

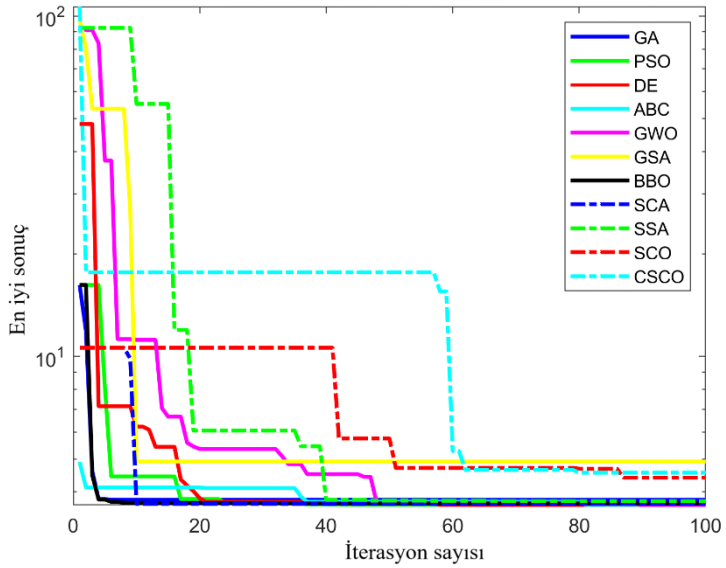
Algoritma	En İyi	En Kötü	Medyan	Ortalama	SS
GA	3.79	20.14	5.91	7.48	4.96
PSO	**3.66**	6.83	3.68	4.00	1.00
DE	**3.66**	4.94	3.70	3.82	0.40
ABC	3.67	4.03	3.77	3.79	0.10
GWO	**3.66**	**3.74**	**3.67**	**3.69**	**0.03**
GSA	4.91	23.75	10.36	10.81	5.65
BBO	3.70	22.14	5.85	7.50	5.54
SCA	3.69	4.96	4.09	4.15	0.36
SSA	3.76	33.40	5.94	9.87	9.34
SCO	4.41	30.61	8.96	12.82	9.30
CSCO	4.56	41.20	14.38	15.66	10.84

Tablo 3.3. Sıkıştırma Yayı Tasarım Problemi İçin CSCO ve Sezgisel Yöntemlerin Koşma Süresi Sonuçları

Algoritma	En İyi	En Kötü	Medyan	Ortalama	SS
GA	0.0616750	0.1359000	0.0918690	0.0927340	0.0249560
PSO	0.0499910	0.1049700	0.0527880	0.0596920	0.0170510
DE	0.0170310	0.0230530	0.0177430	0.0185030	0.0019633
ABC	0.0594420	0.0731790	0.0629050	0.0643450	0.0040185
GWO	0.0052232	0.0088239	0.0055446	0.0059139	0.0010696
GSA	0.0112360	0.0183640	0.0114550	0.0123780	0.0022117
BBO	0.0231790	0.0379550	0.0238750	0.0259630	0.0046562
SCA	0.0052480	0.0079362	0.0053892	0.0059739	0.0010726
SSA	0.0057431	0.0092176	0.0058695	0.0064807	0.0011114
SCO	**0.0002431**	**0.0003984**	**0.0002483**	**0.0002654**	**0.0000472**
CSCO	0.0016326	0.0026542	0.0017230	0.0018843	0.0003620

Tablo 3.3'te sunulan karşılaştırmalı analiz, çeşitli algoritmaların optimum çözümü belirlemedeki zamansal verimliliğini açıklamaktadır. SCO algoritmasının, değerlendirilen tüm ölçütlerdeki üstün başarımının da gösterdiği gibi, hız açısından benzerlerinden daha iyi başarım gösterdiği açıktır. CSCO algoritması SCO algoritmasının hızına erişemese de hız açısından diğer iyi bilinen sezgisel algoritmaları geride bırakmaktadır. CSCO ve SCO algoritmaları arasındaki fark hem keşif hem de sömürü aşamalarında CSCO algoritması tarafından kullanılan ve SCO algoritmasına göre yakınsamasını doğal olarak yavaşlatan kaotik mutasyon mekanizmasına bağlanabilir.

Şekil 3.3, sıkıştırma yayı tasarım probleminin çözümü için CSCO algoritması ile popüler sezgisel algoritmalar arasındaki bir karşılaştırmayı göstermektedir. Bu özel problemde, popüler sezgisel algoritmalar önerilen CSCO algoritmasından daha iyi başarım göstermiş olsa da, genel olarak CSCO algoritmasının başarımı diğer popüler sezgisel algoritmalardan daha iyidir.



Şekil 3.3. Sıkıştırma Yayı Tasarım Problemi İçin CSCO ve Sezgisel Yöntemlerin Karşılaştırılması

Kaynaklı kiriş tasarım problemine ilişkin Tablo 3.4'teki veriler incelendiğinde, PSO ve GWO algoritmalarının en iyi metrik değerlerinden de anlaşılacağı üzere üstün başarım gösterdiği görülmektedir. SCO ve CSCO algoritmaları da yukarıda bahsedilen algoritmaları yakından takip ederek iyi bir başarım sergilemektedir. Özellikle, GWO algoritması bu problemi ele almada en yetkin algoritma olarak öne çıkmaktadır. Ayrıca, CSCO algoritmasının etkinliği, karşılaştırılan diğer algoritmaların çoğundan daha üstündür. CSCO algoritması en iyi metrik değerler incelendiğinde başarım sıralamasında üçüncü sırada, en kötü metrik değerler incelendiğinde başarım sıralamasında altıncı sırada, medyan metrik değerler incelendiğinde başarım sıralamasında altıncı sırada, ortalama metrik değerler incelendiğinde başarım sıralamasında yedinci sırada ve standart sapma metrik değerler incelendiğinde başarım sıralamasında dokuzuncu sırada yer almaktadır.

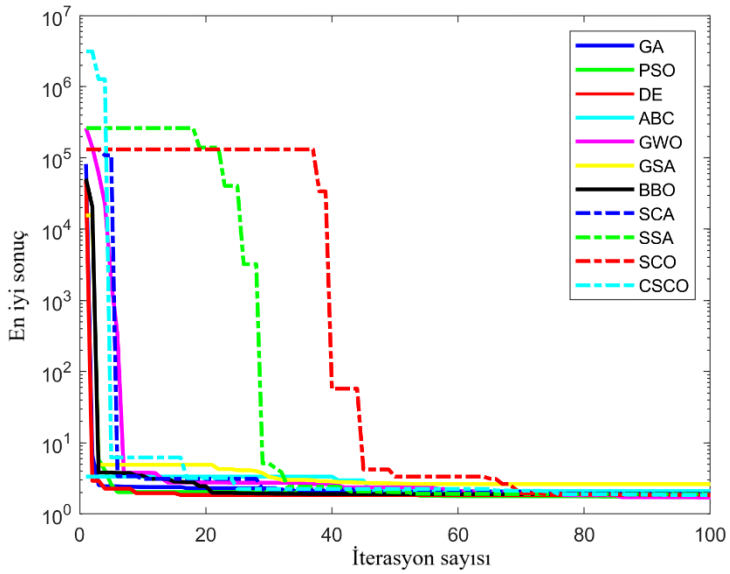
Kaynaklı kiriş tasarım probleminin çözümüne odaklanan Tablo 3.5'in analizi, algoritmaların zamansal verimliliğini ortaya koymaktadır. SCO algoritması, tüm metrik değerlerde en hızlı algoritma olarak ortaya çıkmakta ve benzerlerini önemli ölçüde geride bırakmaktadır. Daha sonra, CSCO algoritması, SCO kadar hızlı olmasa da sık kullanılan diğer sezgisel algoritmalara göre hız avantajını korumaktadır. Şekil 3.4, CSCO algoritmasının başarımını kaynaklı kiriş tasarım problemini çözmek için popüler sezgisel algoritmalarla karşılaştırmaktadır. CSCO algoritması, popüler sezgisel algoritmaların çoğundan daha iyi başarım göstermektedir.

Tablo 3.4. Kaynaklı Kiriş Tasarım Problemi Için CSCO ve Sezgisel Yöntemlerin İstatistiksel Sonuçları

Algoritma	En İyi	En Kötü	Medyan	Ortalama	SS
GA	2.07	3.70	3.19	3.09	0.54
PSO	**1.71**	2.52	1.82	1.88	0.24
DE	1.84	2.49	2.37	2.26	0.26
ABC	2.09	2.78	2.37	2.43	0.20
GWO	**1.71**	**1.75**	**1.72**	**1.73**	**0.02**
GSA	2.63	4.31	3.15	3.26	0.50
BBO	1.89	4.24	3.32	3.16	0.81
SCA	1.94	3.15	2.06	2.25	0.44
SSA	1.91	3.26	2.18	2.34	0.48
SCO	1.85	2818.40	2.84	284.86	890.21
CSCO	1.85	3.64	2.50	2.57	0.57

Tablo 3.5. Kaynaklı Kiriş Tasarım Problemi İçin CSCO ve Sezgisel Yöntemlerin Koşma Süresi Sonuçları

Algoritma	En İyi	En Kötü	Medyan	Ortalama	SS
GA	0.0618960	0.1143400	0.0766410	0.0788180	0.0148760
PSO	0.0516930	0.0877810	0.0689900	0.0686000	0.0111100
DE	0.0173060	0.0297730	0.0179450	0.0198130	0.0040111
ABC	0.0624060	0.0794010	0.0682260	0.0687020	0.0048719
GWO	0.0059972	0.0103220	0.0062223	0.0072458	0.0017289
GSA	0.0118230	0.0198590	0.0127160	0.0135620	0.0024517
BBO	0.0265100	0.0480490	0.0303410	0.0341520	0.0078402
SCA	0.0055226	0.0099640	0.0063057	0.0070072	0.0016557
SSA	0.0060111	0.0097634	0.0065783	0.0071368	0.0013583
SCO	**0.0002832**	**0.0038958**	**0.0003546**	**0.0007149**	**0.0011198**
CSCO	0.0018357	0.0086371	0.0023083	0.0031494	0.0020645



Şekil 3.4. Kaynaklı Kiriş Tasarım Problemi İçin CSCO ve Sezgisel Yöntemlerin Karşılaştırılması

Tablo 3.6, DE algoritmasının en düşük En İyi ve Ortalama metrik değerleri ve mütevazı bir standart sapma ile değerlendirilen algoritmalar arasında en verimli ve güvenilir olduğunu göstermektedir. GSA ve BBO algoritmaları, iyi çözümler bulma yeteneğine sahip olmakla birlikte, yüksek derecede değişkenlik ve düşük başarım riski sergilemektedir. CSCO'nun sonuçları makul bir aralıkta yer almakta, bu da onu eldeki optimizasyon probleminin özel gereksinimlerine ve kısıtlamalarına bağlı olarak uygulanabilir bir seçenek haline getirmektedir.

Tablo 3.7'deki diğer algoritmalarla karşılaştırıldığında, CSCO algoritması yürütme süresi açısından dengeli bir başarım göstermektedir. SCO'nun sahip olduğu en iyi metrik zamana ulaşamamaktadır. Ancak CSCO yine de daha hızlı algoritmalar arasında yer almaktadır. CSCO'nun başarımı, bazı değişkenliklere rağmen genel olarak güvenilirdir. CSCO, lider algoritma SCO'dan biraz daha yüksek değişkenliğe sahip olsa da yürütme süresinin kritik olduğu basınçlı kap tasarım problemi için rekabetçi bir seçim olmaya devam etmektedir.

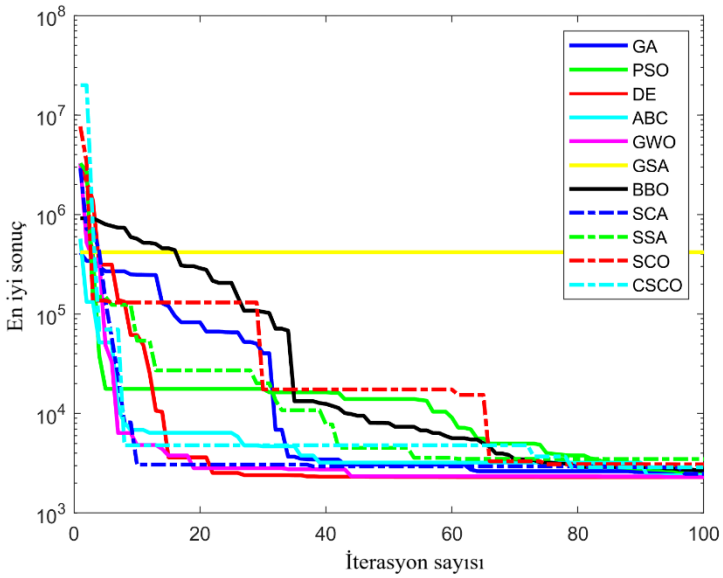
Tablo 3.6. Basınçlı Kap Tasarım Problemi İçin CSCO ve Sezgisel Yöntemlerin İstatistiksel Sonuçları

Algoritma	En İyi	En Kötü	Medyan	Ortalma	SS
GA	2584.8	4905.5	3076.5	3327.5	700.7
PSO	2408.8	3638.7	3616.9	3426.8	395.6
DE	**2302.5**	**3349.7**	**2309.4**	**2563.0**	401.9
ABC	2403.0	3733.0	2552.7	2821.2	492.3
GWO	2309.1	3685.0	2372.6	2629.7	493.2
GSA	418280.0	1341700.0	819320.0	849760.0	344710.0
BBO	2697.8	168830.0	3533.0	19872.0	52338.0
SCA	2482.6	6335.9	4494.2	4424.8	1849.4
SSA	3494.0	4600.0	3923.9	3988.8	**349.2**
SCO	3086.5	29122.0	3944.2	9262.7	9207.7
CSCO	2887.8	7586.3	4023.1	4665.8	1583.5

Tablo 3.7. Basınçlı Kap Tasarım Problemi İçin CSCO ve Sezgisel Yöntemlerin Koşma Süresi Sonuçları

Algoritma	En İyi	En Kötü	Medyan	Ortalama	SS
GA	0.0616680	0.1252100	0.0812410	0.0857710	0.0216260
PSO	0.0505280	0.0592190	0.0519090	0.0531200	0.0028523
DE	0.0171420	0.0290100	0.0176450	0.0192780	0.0037015
ABC	0.0613900	0.0978440	0.0626500	0.0685590	0.0131080
GWO	0.0054781	0.0064982	0.0055442	0.0057175	0.0004056
GSA	0.0113780	0.0134050	0.0115310	0.0119310	0.0007819
BBO	0.0261480	0.0386830	0.0264590	0.0289530	0.0041730
SCA	0.0051390	0.0084059	0.0051807	0.0056852	0.0010373
SSA	0.0054838	0.0088132	0.0055446	0.0059871	0.0010466
SCO	**0.0002098**	**0.0003442**	**0.0002180**	**0.0002398**	**0.0000437**
CSCO	0.0016343	0.0026480	0.0016802	0.0018294	0.0003196

Şekil 3.5, CSCO algoritmasının başarımını basınçlı kap tasarım problemini çözmek için kullanılan popüler sezgisel algoritmalarla karşılaştırmaktadır. CSCO algoritmasının başarımı, verilen sezgisel algoritmalar arasında en iyilerden biridir.



Şekil 3.5. Basınçlı Kap Tasarım Problemi İçin CSCO ve Sezgisel Yöntemlerin Karşılaştırılması

Tablo 3.8'de sunulan veriler incelendiğinde, hız düşürücü tasarım problemine ilişkin olarak CSCO algoritmasının, en iyi metrik ölçüsünde GA, PSO, DE, ABC, GWO, BBO, SCA ve SSA algoritmaları ile zirveyi paylaştığı gözlemlenmektedir. CSCO algoritması, en kötü, medyan ve ortalama metrik değerlerinde ise en iyi başarım gösteren algoritmaların hemen altında yer almaktadır. Standart sapma metrik ölçüsünde ise ortalama bir değer elde eden CSCO algoritmasının başarımı, hız düşürücü tasarım problemi için dikkate değer bir nitelik taşımaktadır.

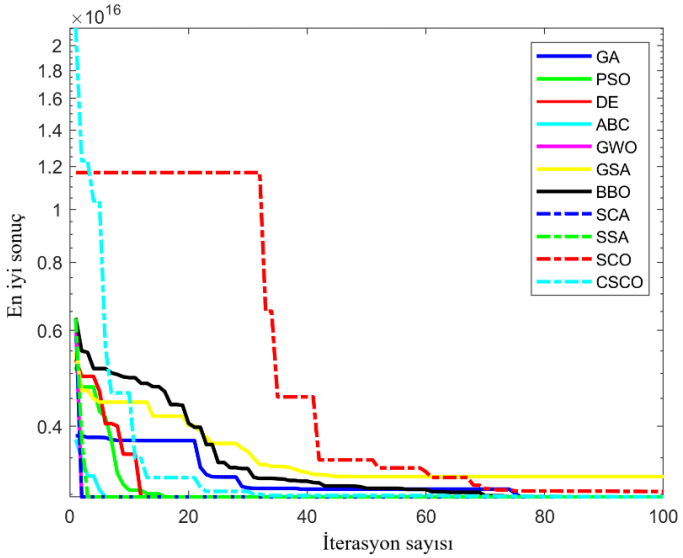
Tablo 3.9'da sunulan zamansal verimlilik değerleri karşılaştırıldığında, CSCO algoritmasının her metrik değerde SCO algoritmasının hemen arkasında yer aldığı ve diğer algoritmalardan daha hızlı çalıştığı ortaya konulmuştur. Şekil 3.6, CSCO algoritmasının başarımını hız düşürücü tasarım problemini çözmek için popüler sezgisel algoritmalarla karşılaştırmaktadır. Bu bağlamda, CSCO algoritması, en iyi başarım gösteren algoritmalardan biri olarak öne çıkmaktadır.

Tablo 3.8. Hız Düşürücü Tasarım Problemi İçin CSCO ve Sezgisel Yöntemlerin İstatistiksel Sonuçları

Algoritma	En İyi	En Kötü	Medyan	Ortalama	SS
GA	**2,97E+15**	3,25E+15	**2,97E+15**	3,01E+15	9,21E+13
PSO	**2,97E+15**	**2,97E+15**	**2,97E+15**	**2,97E+15**	1,28E+06
DE	**2,97E+15**	**2,97E+15**	**2,97E+15**	**2,97E+15**	**5,27E-01**
ABC	**2,97E+15**	**2,97E+15**	**2,97E+15**	**2,97E+15**	2,00E+03
GWO	**2,97E+15**	**2,97E+15**	**2,97E+15**	**2,97E+15**	1,19E+03
GSA	3,23E+15	4,94E+15	3,72E+15	3,79E+15	4,77E+14
BBO	**2,97E+15**	3,33E+15	3,04E+15	3,08E+15	1,29E+14
SCA	**2,97E+15**	**2,97E+15**	**2,97E+15**	**2,97E+15**	1,32E+05
SSA	**2,97E+15**	**2,97E+15**	**2,97E+15**	**2,97E+15**	**5,27E-01**
SCO	3,04E+15	5,46E+15	3,47E+15	3,67E+15	7,50E+14
CSCO	**2,97E+15**	3,53E+15	2,98E+15	3,06E+15	1,78E+14

Tablo 3.9. Hız Düşürücü Tasarım Problemi İçin CSCO ve Sezgisel Yöntemlerin Koşma Süresi Sonuçları

Algoritma	En İyi	En Kötü	Medyan	Ortalama	SS
GA	0,0653480	0,1347600	0,1017700	0,1006100	0,0230280
PSO	0,0492860	0,0939290	0,0614620	0,0658120	0,0157160
DE	0,0172840	0,0353010	0,0191220	0,0242190	0,0076381
ABC	0,0602320	0,1019200	0,0668050	0,0711620	0,0127610
GWO	0,0070289	0,0099284	0,0073818	0,0078148	0,0009363
GSA	0,0122760	0,0161330	0,0137500	0,0140280	0,0013693
BBO	0,0336230	0,0591920	0,0364260	0,0388580	0,0078391
SCA	0,0061876	0,0107990	0,0068527	0,0073503	0,0013850
SSA	0,0061929	0,0108540	0,0068966	0,0075922	0,0015927
SCO	**0,0003116**	**0,0005393**	**0,0003408**	**0,0003781**	**0,0000851**
CSCO	0,0016682	0,0028829	0,0019860	0,0021003	0,0004421



Şekil 3.6. Hız Düşürücü Tasarım Problemi İçin CSCO ve Sezgisel Yöntemlerin Karşılaştırılması

Tablo 3.10'da sunulan başarım verileri incelendiğinde, dişli katarı tasarım problemine yönelik olarak, her metrik değerde en yüksek başarımı sergileyen algoritmaların hemen arkasında yer alan CSCO algoritmasının, bu problem için güvenilir bir başarım sunduğu gözlemlenmiştir. SCO algoritmasının en kötü ve standart sapma metrik değerlerinde gösterdiği başarım düşüklüğünden etkilenmeyen CSCO algoritması, bu bağlamda üstün bir başarım sergilemiştir.

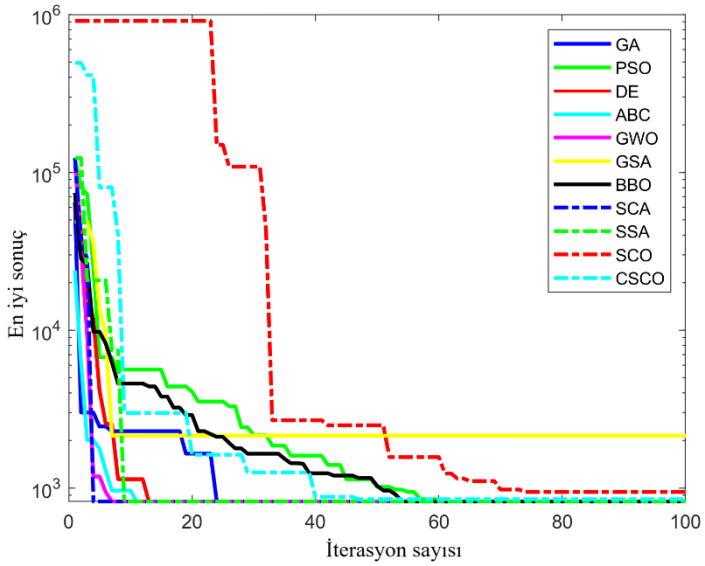
Dişli katarı tasarım problemi için zamansal verimlilik değerlerinin yer aldığı Tablo 3.11 incelendiğinde ise, CSCO algoritmasının, SCO algoritmasının hemen arkasında yer alarak diğer algoritmalardan daha hızlı olduğu belirlenmiştir. Şekil 3.7'de CSCO algoritmasının başarımı, dişli katarı tasarım problemini çözmek için kullanılan popüler sezgisel algoritmalarla karşılaştırılmaktadır. Bu karşılaştırma, CSCO algoritmasının dişli katarı tasarım problemi çözümünde güvenilir ve hızlı bir algoritma olarak iyi bir seçim olduğunu ortaya koymaktadır.

Tablo 3.10. Dişli Katarı Tasarım Problemi İçin CSCO ve Sezgisel Yöntemlerin İstatistiksel Sonuçları

Algoritma	En İyi	En Kötü	Medyan	Ortalama	SS
GA	**821,15**	1853,80	**821,15**	924,41	326,55
PSO	**821,15**	**821,15**	**821,15**	**821,15**	**0,00**
DE	**821,15**	**821,15**	**821,15**	**821,15**	**0,00**
ABC	**821,15**	**821,15**	**821,15**	**821,15**	**0,00**
GWO	**821,15**	**821,15**	**821,15**	**821,15**	**0,00**
GSA	2151,30	11699,00	4619,00	5875,30	3285,40
BBO	**821,15**	1119,30	**821,15**	878,32	120,66
SCA	**821,15**	**821,15**	**821,15**	**821,15**	**0,00**
SSA	**821,15**	**821,15**	**821,15**	**821,15**	**0,00**
SCO	943,28	25856,00	1329,80	5317,50	8220,80
CSCO	849,36	943,28	849,36	861,72	30,14

Tablo 3.11. Dişli Katarı Tasarım Problemi İçin CSCO ve Sezgisel Yöntemlerin Koşma Süresi Sonuçları

Algoritma	En İyi	En Kötü	Medyan	Ortalama	SS
GA	0,0611580	0,1294600	0,0999090	0,0933160	0,0220190
PSO	0,0493470	0,0771780	0,0617720	0,0603450	0,0096916
DE	0,0152270	0,0264980	0,0195670	0,0202630	0,0038325
ABC	0,0612740	0,0889480	0,0793070	0,0767610	0,0113380
GWO	0,0052248	0,0073312	0,0053146	0,0058811	0,0009061
GSA	0,0112380	0,0160320	0,0115380	0,0129170	0,0020070
BBO	0,0257500	0,0366840	0,0261130	0,0293990	0,0045834
SCA	0,0049177	0,0072186	0,0055672	0,0056599	0,0007016
SSA	0,0051733	0,0071457	0,0056521	0,0058962	0,0007569
SCO	**0,0001953**	**0,0002785**	**0,0002120**	**0,0002190**	**0,0000233**
CSCO	0,0016022	0,0022816	0,0017068	0,0017920	0,0002134



Şekil 3.7. Dişli Katarı Tasarım Problemi İçin CSCO ve Sezgisel Yöntemlerin Karşılaştırılması

Himmelblau'nun tasarım problemine ilişkin Tablo 3.12'de sunulan veriler incelendiğinde, CSCO algoritmasının en iyi metrik değerlerinde üstün başarım gösteren algoritmalardan biri olduğu görülmektedir. Bu algoritma, en kötü, medyan ve ortalama metrik değerlerinde zirvede yer alan algoritmalardan sadece küçük bir farkla geride kalmaktadır. En iyi, en kötü ve ortalama metrik değerleri karşılaştırıldığında, CSCO algoritmasının SCO algoritmasından daha iyi bir başarım sergilediği anlaşılmaktadır.

Tablo 3.13'te sunulan zamansal verimlilik değerleri incelendiğinde ise, SCO algoritmasının en hızlı algoritma olduğu belirlenmiştir. CSCO algoritmasının ise diğer sezgisel algoritmalardan daha hızlı bir şekilde çözüm noktasına ulaşarak zaman verimliliği açısından en iyi başarımı gösteren algoritmalardan biri olduğu gözlemlenmiştir.

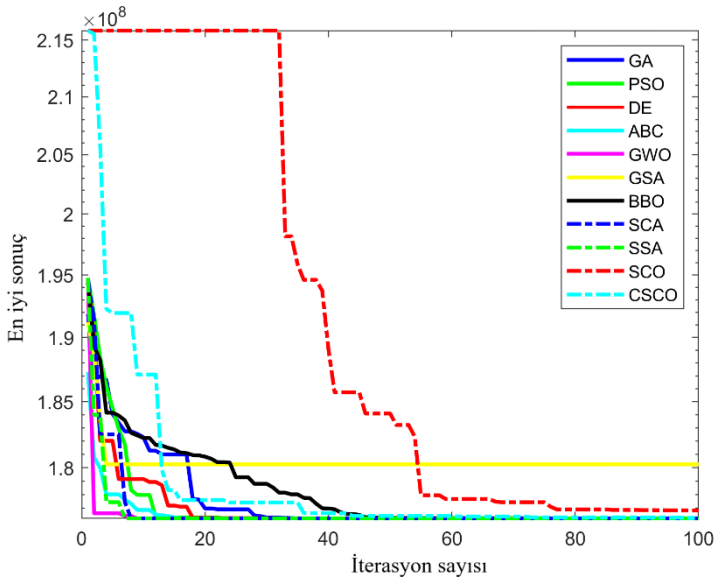
Tablo 3.12. Himmelblau'nun Tasarım Problemi İçin CSCO ve Sezgisel Yöntemlerin Istatistiksel Sonuçları

Algoritma	En İyi	En Kötü	Medyan	Ortalama	SS
GA	**1,76E+08**	1,81E+08	**1,76E+08**	1,77E+08	1,52E+06
PSO	**1,76E+08**	**1,76E+08**	**1,76E+08**	**1,76E+08**	1,10E+03
DE	**1,76E+08**	**1,76E+08**	**1,76E+08**	**1,76E+08**	1,85E+03
ABC	**1,76E+08**	**1,76E+08**	**1,76E+08**	**1,76E+08**	**0,00E+00**
GWO	**1,76E+08**	**1,76E+08**	**1,76E+08**	**1,76E+08**	**0,00E+00**
GSA	1,80E+08	1,83E+08	1,82E+08	1,82E+08	9,14E+05
BBO	**1,76E+08**	1,79E+08	1,77E+08	1,77E+08	1,01E+06
SCA	**1,76E+08**	**1,76E+08**	**1,76E+08**	**1,76E+08**	**0,00E+00**
SSA	**1,76E+08**	**1,76E+08**	**1,76E+08**	**1,76E+08**	4,74E+04
SCO	1,77E+08	1,83E+08	1,80E+08	1,80E+08	1,98E+06
CSCO	**1,76E+08**	1,79E+08	1,77E+08	1,77E+08	7,57E+05

Tablo 3.13. Himmelblau'nun Tasarım Problemi İçin CSCO ve Sezgisel Yöntemlerin Koşma Süresi Sonuçları

Algoritma	En İyi	En Kötü	Medyan	Ortalama	SS
GA	0,0708440	0,1310600	0,0987120	0,0991780	0,0179350
PSO	0,0499400	0,0816470	0,0532780	0,0557580	0,0094692
DE	0,0162870	0,0227670	0,0171660	0,0183530	0,0025517
ABC	0,0586100	0,1041100	0,0616070	0,0710130	0,0167340
GWO	0,0052803	0,0102780	0,0055552	0,0064828	0,0018075
GSA	0,0110600	0,0219000	0,0115780	0,0139740	0,0040692
BBO	0,0276330	0,0461520	0,0290810	0,0323250	0,0062867
SCA	0,0049160	0,0082324	0,0051367	0,0054757	0,0009963
SSA	0,0051357	0,0087321	0,0053801	0,0057643	0,0010712
SCO	**0,0001979**	**0,0003216**	**0,0002148**	**0,0002242**	**0,0000356**
CSCO	0,0015740	0,0026307	0,0016504	0,0017410	0,0003145

Şekil 3.8'de sunulan veriler incelendiğinde, Himmelblau'nun tasarım problemine ilişkin CSCO ve diğer algoritmaların karşılaştırmalı başarımları değerlendirildiğinde, CSCO algoritmasının başarımının en iyi algoritmalara çok yakın olduğu ve zaman verimliliği de dikkate alındığında seçim konusunda rekabetçi bir algoritma olduğu gözlemlenmektedir.



Şekil 3.8. Himmelblau'nun Tasarım Problemi İçin CSCO ve Sezgisel Yöntemlerin Karşılaştırılması

Üç çubuk makas tasarımı problemine ilişkin Tablo 3.14'te sunulan veriler incelendiğinde, tüm metrik değerlerde algoritmaların başarımlarının birbirine oldukça yakın olduğu gözlemlenmiştir. Bu nedenle, herhangi bir algoritmanın başarımının diğerlerinden

belirgin şekilde üstün olduğu sonucuna varılamamaktadır. Ancak, Tablo 3.15'teki zaman verimliliği başarım tablosu incelendiğinde, SCO algoritmasının en iyi başarımı sergilediği, hemen ardından ise CSCO algoritmasının zaman verimliliği açısından diğer algoritmalara kıyasla çok daha üstün bir başarım gösterdiği görülmektedir.

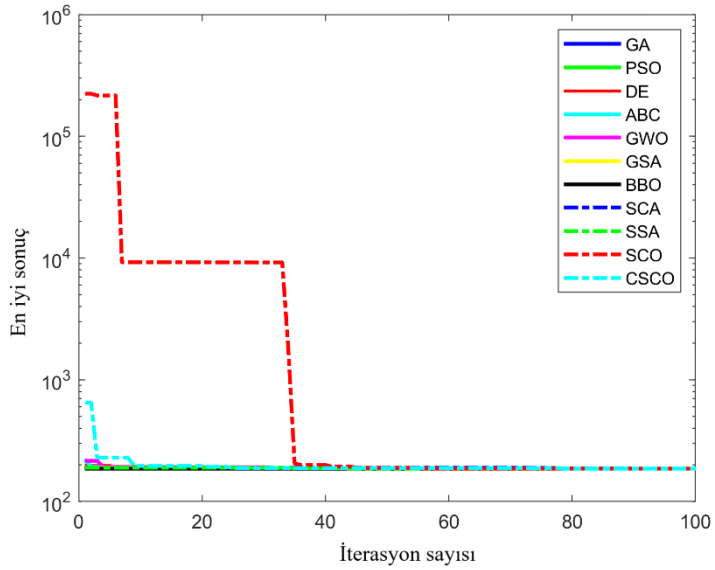
Şekil 3.9'daki karşılaştırma verileri incelendiğinde ise, SCO algoritmasının keşif aşamasında iç kısıtlamalar nedeniyle takıldığı, buna karşın CSCO algoritmasının başarımının en iyilerden biri olduğu tespit edilmiştir. Başarım ve zaman verimliliği birlikte değerlendirildiğinde, CSCO algoritmasının bu problemin çözümü için tercih edilebilecek en uygun algoritma olduğu sonucuna varılabilir.

Tablo 3.14. Üç Çubuk Makas Tasarım Problemi İçin CSCO ve Sezgisel Yöntemlerin İstatistiksel Sonuçları

Algoritma	En İyi	En Kötü	Medyan	Ortalama	SS
GA	186,43	201,69	189,04	190,36	4,75
PSO	**186,39**	186,40	**186,39**	**186,39**	**0,00**
DE	**186,39**	**186,39**	**186,39**	**186,39**	**0,00**
ABC	186,40	186,75	186,46	186,49	0,11
GWO	**186,39**	191,45	186,41	186,91	1,60
GSA	**186,39**	189,00	187,18	187,26	0,78
BBO	**186,39**	193,23	187,25	188,19	2,37
SCA	186,41	199,98	188,11	192,24	6,65
SSA	**186,39**	186,64	186,40	186,44	0,08
SCO	186,41	193,60	187,83	188,75	2,56
CSCO	186,41	197,96	186,97	188,81	3,80

Tablo 3.15. Üç Çubuk Makas Tasarım Problemi İçin CSCO ve Sezgisel Yöntemlerin Koşma Süresi Sonuçları

Algoritma	En İyi	En Kötü	Medyan	Ortalama	SS
GA	0,0660030	0,1371800	0,1007100	0,1008200	0,0256870
PSO	0,0482060	0,0739780	0,0563290	0,0585480	0,0077763
DE	0,0156890	0,0216600	0,0163060	0,0179000	0,0025746
ABC	0,0577690	0,0723350	0,0611910	0,0632050	0,0051144
GWO	0,0044724	0,0057492	0,0052297	0,0050642	0,0004976
GSA	0,0100740	0,0146500	0,0112170	0,0114740	0,0015187
BBO	0,0187570	0,0317770	0,0209830	0,0220960	0,0042526
SCA	0,0043178	0,0073346	0,0044671	0,0050022	0,0009623
SSA	0,0047994	0,0088764	0,0050825	0,0056360	0,0012500
SCO	**0,0001697**	**0,0003045**	**0,0001814**	**0,0002024**	**0,0000433**
CSCO	0,0015244	0,0030202	0,0015619	0,0018127	0,0004695



Şekil 3.9. Üç Çubuk Makas Tasarım Problemi İçin CSCO ve Sezgisel Yöntemlerin Karşılaştırılması

Kademeli konsol kiriş tasarım problemi bağlamında, Tablo 3.16'da sunulan istatistiksel sonuçlar incelendiğinde, en iyi, en kötü ve ortalama gibi başarım metriklerinde tüm algoritmaların benzer sonuçlar verdiği gözlemlenmiştir. Bu durum, algoritmaların genel başarımlarının birbirine yakın olduğunu göstermektedir.

Öte yandan, kademeli konsol kiriş tasarım problemi için zamansal verimlilik verilerinin yer aldığı Tablo 3.17 incelendiğinde, SCO algoritmasının en hızlı sonuçları ürettiği, CSCO algoritmasının ise ikinci sırada yer aldığı belirlenmiştir. Diğer algoritmalarla karşılaştırıldığında, SCO ve CSCO algoritmalarının zaman verimliliği açısından üstün başarım sergilediği açıkça görülmektedir.

Şekil 3.10'da sunulan kademeli konsol kiriş tasarım problemi için karşılaştırma verileri incelendiğinde, SCO algoritmasının 40 iterasyon sonrasında çözüm noktasına ulaştığı, diğer algoritmaların ise 20 iterasyondan daha önce çözüm noktasına ulaştığı tespit edilmiştir. Bu bulgular, SCO algoritmasının daha fazla iterasyon gerektirdiğini ancak yine de etkili bir çözüm sunduğunu göstermektedir.

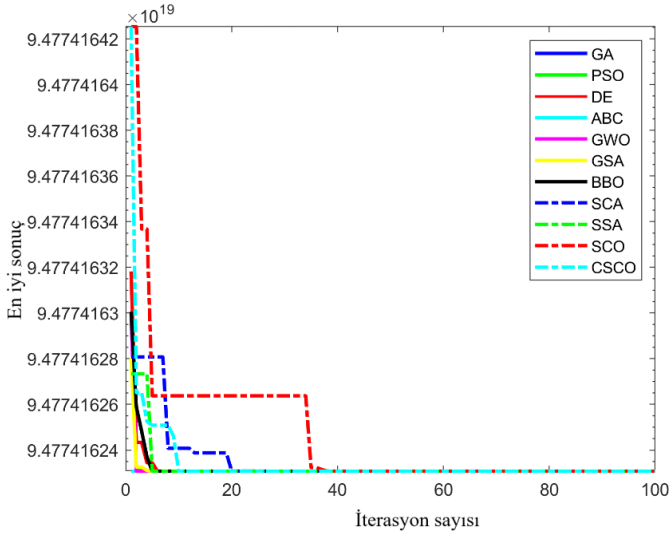
İstatistiksel veriler, zamansal verimlilik ve Şekil 3.10'deki karşılaştırmalar dikkate alındığında, kademeli konsol kiriş tasarım probleminin çözümü için CSCO algoritmasının en uygun seçenek olduğu sonucuna varılmıştır.

Tablo 3.16. Kademeli Konsol Kiriş Tasarım Problemi İçin CSCO ve Sezgisel Yöntemlerin İstatistiksel Sonuçları

Algoritma	En İyi	En Kötü	Medyan	Ortalama	SS
GA	**9,48E+19**	**9,48E+19**	**9,48E+19**	**9,48E+19**	4,63E+04
PSO	**9,48E+19**	**9,48E+19**	**9,48E+19**	**9,48E+19**	1,34E+04
DE	**9,48E+19**	**9,48E+19**	**9,48E+19**	**9,48E+19**	2,18E+04
ABC	**9,48E+19**	**9,48E+19**	**9,48E+19**	**9,48E+19**	7,72E+03
GWO	**9,48E+19**	**9,48E+19**	**9,48E+19**	**9,48E+19**	9,46E+03
GSA	**9,48E+19**	**9,48E+19**	**9,48E+19**	**9,48E+19**	1,73E+04
BBO	**9,48E+19**	**9,48E+19**	**9,48E+19**	**9,48E+19**	1,22E+04
SCA	**9,48E+19**	**9,48E+19**	**9,48E+19**	**9,48E+19**	1,64E+04
SSA	**9,48E+19**	**9,48E+19**	**9,48E+19**	**9,48E+19**	1,34E+04
SCO	**9,48E+19**	**9,48E+19**	**9,48E+19**	**9,48E+19**	**1,09E+04**
CSCO	**9,48E+19**	**9,48E+19**	**9,48E+19**	**9,48E+19**	1,22E+04

Tablo 3.17. Kademeli Konsol Kiriş Tasarım Problemi İçin CSCO ve Sezgisel Yöntemlerin Koşma Süresi Sonuçları

Algoritma	En İyi	En Kötü	Medyan	Ortalama	SS
GA	0,0622890	0,1227500	0,0897620	0,0914070	0,0181290
PSO	0,0483640	0,1034200	0,0529000	0,0655070	0,0228260
DE	0,0155550	0,0313650	0,0173500	0,0194490	0,0054333
ABC	0,0557680	0,0936470	0,0610040	0,0657050	0,0118000
GWO	0,0065104	0,0131920	0,0070214	0,0075203	0,0020254
GSA	0,0121310	0,0250290	0,0131690	0,0145730	0,0038901
BBO	0,0400370	0,0683420	0,0447890	0,0503990	0,0109830
SCA	0,0057229	0,0101150	0,0063333	0,0067879	0,0013089
SSA	0,0054644	0,0098976	0,0061379	0,0065126	0,0013260
SCO	**0,0002361**	**0,0004096**	**0,0002692**	**0,0002811**	**0,0000535**
CSCO	0,0015948	0,0028007	0,0017760	0,0018787	0,0003628



Şekil 3.10. Kademeli Konsol Kiriş Tasarım Problemi İçin CSCO ve Sezgisel Yöntemlerin Karşılaştırılması

Çok diskli debriyaj fren tasarımı problemi bağlamında, Tablo 3.18'de sunulan istatistiksel sonuçlar incelendiğinde, CSCO algoritmasının en iyi, en kötü, medyan ve ortalama metrik değerlerinde en yüksek başarım gösteren algoritmalardan biri olduğu tespit edilmiştir. Tablo 3.19'da yer alan zamansal verimlilik değerleri analiz edildiğinde ise, CSCO algoritmasının, SCO algoritması hariç diğer tüm algoritmalardan daha hızlı olduğu gözlemlenmiştir.

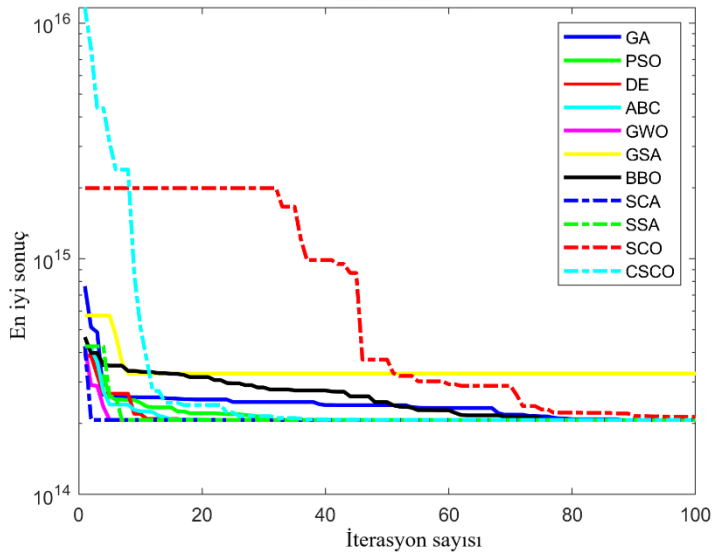
Şekil 3.11'de sunulan çok diskli debriyaj fren tasarımı problemi için algoritmaların karşılaştırılması sonucunda, CSCO algoritmasının en iyi başarıma sahip algoritmalardan biri olduğu bir kez daha doğrulanmıştır. Bu bağlamda, CSCO algoritması, söz konusu problem için hem hız hem de üstün başarım açısından öne çıkmaktadır.

Tablo 3.18. Çok Diskli Debriyaj Fren Tasarım Problemi İçin CSCO ve Sezgisel Yöntemlerin İstatistiksel Sonuçları

Algoritma	En İyi	En Kötü	Medyan	Ortalama	SS
GA	**2,07E+14**	2,31E+14	**2,07E+14**	2,12E+14	8,50E+12
PSO	**2,07E+14**	**2,07E+14**	**2,07E+14**	**2,07E+14**	3,40E+08
DE	**2,07E+14**	**2,07E+14**	**2,07E+14**	**2,07E+14**	**3,29E-02**
ABC	**2,07E+14**	**2,07E+14**	**2,07E+14**	**2,07E+14**	7,26E+05
GWO	**2,07E+14**	**2,07E+14**	**2,07E+14**	**2,07E+14**	**3,29E-02**
GSA	3,26E+14	5,10E+14	4,34E+14	4,30E+14	5,81E+13
BBO	**2,07E+14**	2,51E+14	2,25E+14	2,24E+14	1,53E+13
SCA	**2,07E+14**	**2,07E+14**	**2,07E+14**	**2,07E+14**	**3,29E-02**
SSA	**2,07E+14**	**2,07E+14**	**2,07E+14**	**2,07E+14**	**3,29E-02**
SCO	2,13E+14	3,01E+14	2,40E+14	2,47E+14	2,85E+13
CSCO	**2,07E+14**	**2,07E+14**	**2,07E+14**	**2,07E+14**	2,59E+09

Tablo 3.19. Çok Diskli Debriyaj Fren Tasarım Problemi İçin CSCO ve Sezgisel Yöntemlerin Koşma Süresi Sonuçları

Algoritma	En İyi	En Kötü	Medyan	Ortalama	SS
GA	0,0711250	0,1224500	0,1056700	0,1020500	0,0177000
PSO	0,0504050	0,1082600	0,0533390	0,0631790	0,0184020
DE	0,0164200	0,0477340	0,0178280	0,0219930	0,0097339
ABC	0,0619490	0,0998250	0,0689010	0,0731630	0,0125990
GWO	0,0056733	0,0094929	0,0060266	0,0066882	0,0012109
GSA	0,0115600	0,0191150	0,0123270	0,0132880	0,0024206
BBO	0,0289520	0,0526390	0,0298630	0,0345630	0,0085976
SCA	0,0053410	0,0090225	0,0056867	0,0064212	0,0013983
SSA	0,0055750	0,0101520	0,0058685	0,0067313	0,0018001
SCO	**0,0002367**	**0,0004139**	**0,0002645**	**0,0002902**	**0,0000640**
CSCO	0,0016141	0,0032972	0,0017806	0,0020311	0,0005635



Şekil 3.11. Çok Diskli Debriyaj Fren Tasarım Problemi İçin CSCO ve Sezgisel Yöntemlerin Karşılaştırılması

Hidrodinamik baskı yatağı tasarım problemi bağlamında, Tablo 3.20'de sunulan istatistiksel sonuçlar incelendiğinde, CSCO algoritmasının en iyi metrik değerlerinde üstün bir başarım sergileyen algoritmalardan biri olduğu tespit edilmiştir. Algoritmanın en kötü, medyan ve ortalama metrik değerlerinde ise zirvede yer alan diğer algoritmaların başarımına oldukça yakın bir başarım gösterdiği gözlemlenmiştir. Tablo 3.21'de sunulan zamansal verimlilik sonuçlarına göre, CSCO algoritması, SCO algoritmasının başarımına çok yakın bir verimlilik sergileyerek diğer algoritmalara kıyasla daha üstün bir başarım ortaya koymuştur.

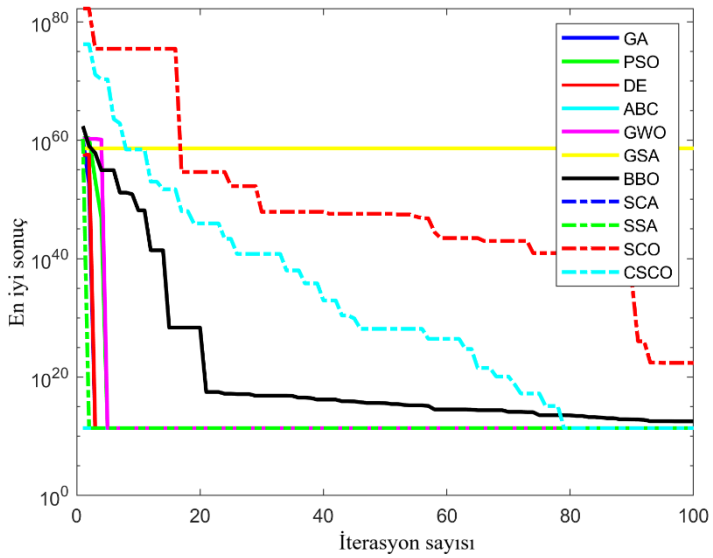
Tablo 3.20. Hidrodinamik Baskı Yatağı Tasarım Problemi İçin CSCO Ve Sezgisel Yöntemlerin İstatistiksel Sonuçları

Algoritma	En İyi	En Kötü	Medyan	Ortalama	SS
GA	**2,37E+11**	1,01E+54	1,13E+42	1,01E+53	3,20E+53
PSO	**2,37E+11**	**2,37E+11**	**2,37E+11**	**2,37E+11**	4,16E+05
DE	**2,37E+11**	**2,37E+11**	**2,37E+11**	**2,37E+11**	**6,43E-05**
ABC	**2,37E+11**	**2,37E+11**	**2,37E+11**	**2,37E+11**	**6,43E-05**
GWO	**2,37E+11**	**2,37E+11**	**2,37E+11**	**2,37E+11**	**6,43E-05**
GSA	4,15E+58	4,25E+70	7,87E+61	4,69E+69	1,34E+70
BBO	3,29E+12	2,88E+45	8,20E+28	2,96E+44	9,08E+44
SCA	**2,37E+11**	**2,37E+11**	**2,37E+11**	**2,37E+11**	**6,43E-05**
SSA	**2,37E+11**	**2,37E+11**	**2,37E+11**	**2,37E+11**	**6,43E-05**
SCO	2,44E+22	1,07E+40	8,69E+29	1,07E+39	3,38E+39
CSCO	**2,37E+11**	3,28E+12	2,74E+12	1,80E+12	1,34E+12

Tablo 3.21. Hidrodinamik Baskı Yatağı Tasarım Problemi İçin CSCO ve Sezgisel Yöntemlerin Koşma Süresi Sonuçları

Algoritma	En İyi	En Kötü	Medyan	Ortalama	SS
GA	0,0636540	0,1341100	0,0807730	0,0885290	0,0238620
PSO	0,0491940	0,0673630	0,0511840	0,0543180	0,0062480
DE	0,0163480	0,0214780	0,0173900	0,0180230	0,0018649
ABC	0,0605580	0,0870480	0,0633660	0,0663310	0,0080664
GWO	0,0056842	0,0115050	0,0061554	0,0066733	0,0017338
GSA	0,0117320	0,0209090	0,0118280	0,0129620	0,0028510
BBO	0,0260750	0,0345280	0,0280450	0,0286840	0,0028316
SCA	0,0057499	0,0092241	0,0060271	0,0063876	0,0010618
SSA	0,0060190	0,0118290	0,0062530	0,0069136	0,0017733
SCO	**0,0002794**	**0,0004990**	**0,0003044**	**0,0003210**	**0,0000645**
CSCO	0,0016987	0,0033756	0,0017979	0,0019689	0,0005103

Şekil 3.12, CSCO algoritmasının hidrodinamik baskı yatağı tasarım problemini çözme konusundaki başarımını popüler sezgisel algoritmalarla karşılaştırmaktadır. Bu karşılaştırma sonucunda, CSCO algoritmasının en iyi başarıma sahip algoritmalardan biri olduğu bir kez daha doğrulanmıştır.



Şekil 3.12. Hidrodinamik Baskı Yatağı Tasarım Problemi İçin CSCO ve Sezgisel Yöntemlerin Karşılaştırılması

Tablo 3.22'de verilen CSCO algoritmasının farklı problemler üzerindeki başarımı incelendiğinde, bazı problemler için oldukça tutarlı ve sabit bir başarım sergilediği, bazı problemler için ise ortalama değer gösterdiği görülmektedir. Örneğin, Sıkıştırma yayı tasarım probleminde algoritmanın başarımının diğer sezgisel yöntemlere göre düşük olduğu

gözlemlenebilir. Kaynaklı kiriş ve basınçlı kap tasarım problemlerinde ise CSCO algoritmasının başarımının ortalamada kaldığı gözlemlenmiştir. Özellikle CSCO algoritmasının hız düşürücü, Himmelblau'nun problemi, kademeli konsol kiriş, çok diskli debriyaj fren, hidrodinamik baskı yatağı tasarım problemlerinde ise başarım sıralamalarında ilk sırada yer alarak bu problemlerin çözümünde en uygun seçenek olduğu gözlemlenmektedir.

Tablo 3.22. 10 Mühendislik Problemi İçin CSCO Algoritmasının Başarım Sıralamaları

Problem	En İyi	En Kötü	Medyan	Ortalama	SS
Sıkıştırma Yayı	8	11	11	11	11
Kaynaklı Kiriş	3	7	6	7	9
Basınçlı Kap	8	8	9	8	7
Hız Düşürücü	1	4	2	3	7
Dişli Katarı	2	2	2	2	2
Himmelblau'nun Problemi	1	2	2	2	3
Üç Çubuk Makas	3	7	5	7	8
Kademeli Konsol Kiriş	1	1	1	1	2
Çok Diskli Debriyaj Fren	1	1	1	1	3
Hidrodinamik Baskı Yatağı	1	2	2	2	2

4.SONUÇLAR ve ÖNERİLER

Bu tez çalışması, optimizasyon problemlerinin çözümü için geleneksel sürü tabanlı yöntemlerden farklı olarak tek bir aday çözüme odaklanan Tek Aday Optimizasyon Algoritmasının (SCO) kaotik mutasyon stratejileri kullanılarak geliştirilmiş Kaotik Mutasyon Stratejisi Tabanlı Tek Aday Optimizasyon (CSCO) algoritmasının başarımının analizini yapmaktadır. SCO, keşif ve sömürü arasında denge kuran iki aşamalı bir yöntemle konum güncelleme stratejisi kullanır. Tasarımda minimalizm, az sayıda parametre, azaltılmış hesaplama gereksinimleri ve sağlam başarım gibi avantajlarına rağmen, SCO'nun dezavantajları da yok değildir. Bunlar arasında sınırlı keşif eğilimi, yerel optimumlarda sıkışıp kalmaya yatkınlık ve optimal olmayan bölgelere karşı artan hassasiyet sayılabilir. Bu zorlukların üstesinden gelmek amacıyla araştırma, algoritmanın keşif kapasitesini önemli ölçüde artırmak için kaotik fonksiyonlarla desteklenen yenilikçi bir mutasyon stratejisi sunmaktadır.

Bu çalışmada, Kaotik Mutasyon Stratejisi Tabanlı Tek Aday Optimizasyon (CSCO) algoritmasının ve selefi SCO'nun, ilk olarak 23 benchmark test seti için karşılaştırılması yapılmıştır. Karşılaştırmalı çalışmanın sonuçları, CSCO algoritmasının orijinal SCO'ya kıyasla sağlam bir optimizasyon aracı olarak etkinliğini göstermektedir. CSCO algoritmasında tanımlanan p_0 ve p_1 parametreli ile bu parametrelerden elde edilen β (beta) değişkeni sayesinde, keşif ve sömürü arasında bir denge kurulmasına ve üstün optimizasyon sonuçları elde edilmesine olanak sağlanmıştır. Elde edilen sonuçlar, CSCO'nun En İyi, En Kötü, Medyan ve standart sapma ölçümleri dahil olmak üzere çeşitli değerlendirme ölçütlerinde SCO'dan daha iyi başarım gösterdiğini göstermektedir. Ek olarak, SCO algoritmasının çözüm noktasına yakınsamada karşılaştığı zorlukları aşmak amacıyla, CSCO algoritmasında β değişkeninin kontrol edilmesiyle geliştirilen yakınsama formülleri, bu kısıtlamaların üstesinden gelinmesinde yardımcı olarak CSCO algoritmasının, SCO algoritmasına kıyasla daha kararlı ve güvenilir bir yakınsama gerçekleştirmesini sağlamaktadır. Bu fonksiyonların sayesinde CSCO algoritmasının karmaşık optimizasyon ortamlarında gezinme konusundaki yeterliliği SCO algoritmasına kıyasla daha yüksektir. Algoritmanın erken yakınsamadan kaçınırken arama uzayını sistematik olarak keşfetme yeteneği, CSCO'yu çok çeşitli gerçek dünya senaryolarında ve karmaşık optimizasyon problemlerinde kullanım için umut verici bir aday haline getirmektedir. Daha sonra bu çalışma, SCO ve CSCO algoritmalarının on mühendislik tasarım problemi bağlamında yerleşik sezgisel algoritmalarla birlikte karşılaştırmalı bir analizini sunmaktadır. Deneysel sonuçlar, farklı mutasyon metodolojilerinin entegrasyonunun SCO

algoritmasının optimizasyon yeterliliğini önemli ölçüde artırdığını göstermektedir. CSCO algoritmasının başarımı, popüler sezgisel algoritmalara kıyasla yüksektir. CSCO algoritmasının etki alanını sistematik olarak arama ve erken yakınsamadan kaçınma yeteneği, onu çeşitli pratik ortamlarda ve karmaşık optimizasyon problemlerinde kullanım için güçlü bir aday haline getirmektedir. Devam eden araştırma ve iyileştirmelerin CSCO'nun farklı sektörlerde önemli bir optimizasyon aracı olarak etkinliğini daha da artırması beklenmektedir.

Gelecek çalışma olarak CSCO algoritmasının ikili (binary) problemlere uyarlanması potansiyel olarak değerlendirilebilir. Ayrıca, CSCO'nun çok amaçlı optimizasyon problemlerine entegrasyonu da mümkündür. Bunun yanı sıra, CSCO algoritmasının diğer yapılarla hibrit olarak çalıştırılması da araştırılabilir. CSCO algoritması farklı alanlardaki optimizasyon problemlerini çözmek için de ayarlanabilir ve kullanılabilir.

KAYNAKÇA

Abouhabaga, O., Gadallah, M. H., Kouta, H. K., & Zaghloul, M. A. (2021). A novel approach for optimizing real life problems using hybrid genetic algorithm and inner–outer array. *Port-Said Engineering Research Journal, 25*(2), 155-164.

Alhijawi, B., & Awajan, A. (2023). Genetic algorithms: Theory, genetic operators, solutions, and applications. *Evolutionary Intelligence,* 1-12.

Ansay, S., & Köse, B. (2023). Sınırsız optimizasyon. In *Teknobilim-2023 Optimizasyon Modelleme ve Yapay Zekâ Optimizasyon Algoritmaları* (p. 83).

Arora, J. S. (2004). *Introduction to optimum design*. Elsevier.

Baş, E. (2020). *Sosyal örümcek algoritmasının sürekli ve ayrık optimizasyon problemlerinde başarım iyileştirmeleri.*

Chen, J., Xin, B., Peng, Z., Dou, L., & Zhang, J. (2009). Optimal contraction theorem for exploration–exploitation tradeoff in search and optimization. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, 39*(3), 680-691.

Deb, K., & Srinivasan, A. (2006, July). Innovization: Innovating design principles through optimization. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation* (pp. 1629-1636).

Delahaye, D., Chaimatanan, S., & Mongeau, M. (2019). Simulated annealing: From basics to applications. In *Handbook of Metaheuristics* (pp. 1-35).

Dhadwal, M. K., Lim, K. B., Jung, S. N., & Kim, T. J. (2013). Particle swarm assisted genetic algorithm for the optimal design of flexbeam sections. *International Journal of Aeronautical and Space Sciences, 14*(4), 341-349.

Doğan, C., & Yüzgeç, U. (2023a). Accelerated Opposition Learning based Single Candidate Optimization Algorithm. 11th International Congress of Academic Studies, ICAR23.

Doğan, C., & Yüzgeç, U. (2023b). Opposition Learning based Single Candidate Optimizer for Clustering Problems. 11th International Congress of Academic Studies, ICAR23.

Dokur, E., Yüzgeç, U., & Kurban, M. (2021). Performance comparison of hybrid neuro-fuzzy models using meta-heuristic algorithms for short-term wind speed forecasting. Electrica, 21(3), 305-321.

Dorigo, M., & Stützle, T. (2019). Ant colony optimization: Overview and recent advances. In *Springer International Publishing* (pp. 311-351).

Eberhart, R., & Kennedy, J. (1995, November). Particle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks* (Vol. 4, pp. 1942-1948).

Ertürk, E., Aydın, E., & Şıldır, H. (2021). Tam sayılı ve sürekli optimizasyon problemi ile reaksiyon ağ modellerinin küçültülmesi. *Konya Journal of Engineering Sciences, 9*, 142-156.

Faris, H., Mirjalili, S., Aljarah, I., Mafarja, M., & Heidari, A. A. (2020). Salp swarm algorithm: Theory, literature review, and application in extreme learning machines. In *Nature-Inspired Optimizers: Theories, Literature Reviews and Applications* (pp. 185-199).

Gandomi, A. H., Yang, X. S., & Alavi, A. H. (2011). Mixed variable structural optimization using firefly algorithm. *Computers & Structures, 89*(23-24), 2325-2336.

Garip, Z., Çimen, M. E., & Boz, A. F. (2021). Harris şahinleri ve balina optimizasyon algoritmalarının kısıt işleme teknikleriyle uygulaması: Karşılaştırmalı bir çalışma. *Journal of Intelligent Systems: Theory and Applications, 4*(2), 76-85.

İnaç, T., Dokur, E., & Yüzgeç, U. (2022) A multi-strategy random weighted gray wolf optimizer-based multi-layer perceptron model for short-term wind speed forecasting. Neural Comput & Applic 34, 14627–14657.

Jia, Z., Gao, X., Cai, X., & Han, D. (2021). Local linear convergence of the alternating direction method of multipliers for nonconvex separable optimization problems. *Journal of Optimization Theory and Applications, 188*, 1-25.

Kale G. A., & Yüzgeç U., (2022) Advanced strategies on update mechanism of Sine Cosine Optimization Algorithm for feature selection in classification problems, Engineering Applications of Artificial Intelligence, Volume 107.

Karaboga, D., Gorkemli, B., Ozturk, C., & Karaboga, N. (2014). A comprehensive survey: Artificial bee colony (ABC) algorithm and applications. *Artificial Intelligence Review, 42*, 21-57.

Karaboğa, D. (2014). *Yapay zeka optimizasyon algoritmaları*. Nobel Akademik Yayıncılık.

Karakuzu, C. (2017) "On the performance of newsworthy meta-heuristic algorithms based on point of view fuzzy modelling," Turkish Journal of Electrical Engineering and Computer Sciences: Vol. 25: No. 6, Article 22.

Kennedy, J., & Eberhart, R. (1995, November). Particle swarm optimization. In *Proceedings of ICNN'95-International Conference on Neural Networks* (Vol. 4, pp. 1942-1948). IEEE.

Kılıç, H., & Yuzgec, U. (2021). Improved Antlion Optimization Algorithm For Quadratic Assignment Problem. Malaysian Journal of Computer Science, 34(1), 34–60.

Kılıç, H., & Yüzgeç, U. (2019a) Tournament selection based antlion optimization algorithm for solving quadratic assignment problem, Engineering Science and Technology, an International Journal, Volume 22, Issue 2.

Kılıç, H., & Yüzgeç, U. (2019b) Improved antlion optimization algorithm via tournament selection and its application to parallel machine scheduling, Computers & Industrial Engineering, Volume 132.

Kilic, H., Yüzgeç, U., & Karakuzu, C. (2020) A novel improved antlion optimizer algorithm and its comparative performance. Neural Comput & Applic 32, 3803–3824.

Konca, Ş., & Atalan, S. (2023). Optimizasyon problemlerinin. In *Teknobilim-2023 Optimizasyon Modelleme ve Yapay Zekâ Optimizasyon Algoritmaları* (p. 49).

Kukkonen, S., & Lampinen, J. (2005, September). GDE3: The third evolution step of generalized differential evolution. In *2005 IEEE Congress on Evolutionary Computation* (Vol. 1, pp. 443-450). IEEE.

Li, C., & Grossmann, I. E. (2021). A review of stochastic programming methods for optimization of process systems under uncertainty. *Frontiers in Chemical Engineering, 2*, 622241.

Lin, M. H., Tsai, J. F., Hu, N. Z., & Chang, S. C. (2013). Design optimization of a speed reducer using deterministic techniques. *Mathematical Problems in Engineering, 2013*(1), 419043.

Liu, Z., Yang, H., & Lai, M. (2005, November). Electricity price forecasting model based on chaos theory. In *2005 International Power Engineering Conference* (pp. 1-449). IEEE.

Lokman, B. (2017). Çok amaçlı tamsayı programlama problemleri için temsili çözüm üreten yaklaşımların ve kalite ölçülerinin incelenmesi. *Endüstri Mühendisliği, 28*(1), 19-39.

Madic, M., Janković, P., Trifunović, M., & Kovacević, M. (2021). Application of software solution for solving engineering design optimization problems. In *IOP Conference Series: Materials Science and Engineering* (Vol. 1018, No. 1, p. 012025). IOP Publishing.

Mirjalili, S. (2016). SCA: A sine cosine algorithm for solving optimization problems. *Knowledge-Based Systems, 96*, 120-133.

Mirjalili, S., & Mirjalili, S. (2019). Genetic algorithm. In *Evolutionary Algorithms and Neural Networks: Theory and Applications* (pp. 43-55).

Mirjalili, S., Gandomi, A. H., Mirjalili, S. Z., Saremi, S., Faris, H., & Mirjalili, S. M. (2017). Salp swarm algorithm: A bio-inspired optimizer for engineering design problems. *Advances in Engineering Software, 114*, 163-191.

Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. *Advances in Engineering Software, 69*, 46-61.

Parmaksiz H., Yüzgeç U.,Dokur E., & Erdogan N., (2023) Mutation based improved dragonfly optimization algorithm for a neuro-fuzzy system in short term wind speed forecasting, Knowledge-Based Systems, Volume 268.

Pinciroli, L., Baraldi, P., & Zio, E. (2023). Maintenance optimization in Industry 4.0. *Reliability Engineering & System Safety, 234*, 109204.

Price, K. V. (2013). Differential evolution. In *Handbook of Optimization: From Classical to Modern Approach* (pp. 187-214). Berlin, Heidelberg: Springer Berlin Heidelberg.

Rao, R. V., & Pawar, R. B. (2020). Constrained design optimization of selected mechanical system components using Rao algorithms. *Applied Soft Computing, 89*, 106141.

Rashedi, E., & Nezamabadi-Pour, H. (2018). A comprehensive survey on gravitational search algorithm. *Swarm and Evolutionary Computation, 41*, 141-158.

Rashedi, E., Nezamabadi-Pour, H., & Saryazdi, S. (2009). GSA: A gravitational search algorithm. *Information Sciences, 179*(13), 2232-2248.

Rather, S. A., & Bala, P. S. (2021). Lévy flight and chaos theory-based gravitational search algorithm for mechanical and structural engineering design optimization. *Open Computer Science, 11*(1), 509-529.

Ray, T., & Saini, P. (2007). Engineering design optimization using a swarm with an intelligent information sharing among individuals. *Journal of Engineering Optimization, 33*(1), 735-748.

Rocha, A. M. A., & Fernandes, E. M. (2009). Hybridizing the electromagnetism-like algorithm with descent search for solving engineering design problems. *International Journal of Computer Mathematics, 86*(10-11), 1932-1946.

Samma, H., Mohamad-Saleh, J., Suandi, S. A., & Lahasan, B. (2020). Q-learning-based simulated annealing algorithm for constrained engineering design problems. *Neural Computing and Applications, 32*. https://doi.org/10.1007/s00521-019-04008-z

Shami, T. M., Grace, D., Burr, A., & Mitchell, P. D. (2022). Single candidate optimizer: A novel optimization algorithm. *Evolutionary Intelligence,* 1-25.

Simon, D. (2008). Biogeography-based optimization. *IEEE Transactions on Evolutionary Computation, 12*(6), 702-713.

Song, P. C., Pan, J. S., & Chu, S. C. (2020). A parallel compact cuckoo search algorithm for three-dimensional path planning. *Applied Soft Computing, 94*, 106443.

Tzanetos, A., & Blondin, M. (2023). A qualitative systematic review of metaheuristics applied to tension/compression spring design problem: Current situation, recommendations, and research direction. *Engineering Applications of Artificial Intelligence, 118*, 105521.

Varol Altay, E. (2022). Gerçek dünya mühendislik tasarım problemlerinin çözümünde kullanılan metasezgisel optimizasyon algoritmalarının başarımlarının incelenmesi. *International Journal of Innovative Engineering Applications, 6*(1), 65-74.

Wang, D., Tan, D., & Liu, L. (2018). Particle swarm optimization algorithm: An overview. *Soft Computing, 22*, 387-408.

Yang, X. S. (2010a). Firefly algorithm, stochastic test functions and design optimization. *International Journal of Bio-Inspired Computation, 2*(2), 78-84.

Yang, X. S. (2010b). *Engineering optimization: An introduction with metaheuristic applications*. John Wiley & Sons.

Yang, X. S., & He, X. (2013). Bat algorithm: Literature review and applications. *International Journal of Bio-Inspired Computation, 5*(3), 141-149.

Yang, X. S., & Slowik, A. (2020). Firefly algorithm. In *Swarm Intelligence Algorithms* (pp. 163-174). CRC Press.

Yıldırım, A. E., & Karcı, A. (2018, September). Application of three bar truss problem among engineering design optimization problems using artificial atom algorithm. In *2018 International Conference on Artificial Intelligence and Data Processing (IDAP)* (pp. 1-5). IEEE.

Yüzgeç U., & Eser M., (2018) Chaotic based differential evolution algorithm for optimization of baker's yeast drying process, Egyptian Informatics Journal, Volume 19, Issue 3.

Yüzgeç, U., & İnaç, T. (2016). Adaptive spiral optimization algorithm for benchmark problems. Bilecik Şeyh Edebali Üniversitesi Fen Bilimleri Dergisi, 3(1), 8-15.

Zhang, W., Gu, X., Tang, L., Yin, Y., Liu, D., & Zhang, Y. (2022). Application of machine learning, deep learning and optimization algorithms in geoengineering and geoscience: Comprehensive review and future challenge. *Gondwana Research, 109*, 1-17.