T.C.

SELÇUK ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

AYRIK OPTİMİZASYON PROBLEMLERİNİN ÇÖZÜMÜNDE GÖÇMEN KUŞLAR OPTİMİZASYON (MBO) ALGORİTMASININ İYİLEŞTİRİLMESİ

Vahit TONGUR

DOKTORA TEZİ

Bilgisayar Mühendisliği Anabilim Dalını

Kasım-2018
KONYA

TEZ KABUL VE ONAYI

Vahit TONGUR tarafından hazırlanan "Ayrık Optimizasyon Problemlerinin Çözümünde Göçmen Kuşlar Optimizasyon (MBO) Algoritmasının İyileştirilmesi" adlı tez çalışması 12/11/2018 tarihinde aşağıdaki jüri tarafından oy birliği ile Selçuk Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı'nda DOKTORA TEZİ olarak kabul edilmiştir.

Jüri Üyeleri	**İmza**
Başkan Prof. Dr. Harun UĞUZ	.....
Danışman Prof. Dr. Erkan ÜLKER	.....
Üye Doç. Dr. Mustafa Servet KIRAN	.....
Üye Dr. Öğr. Üyesi Mehmet HACIBEYOĞLU	.....
Üye Dr. Öğr. Üyesi Onur İNAN	.....

Yukarıdaki sonucu onaylarım.

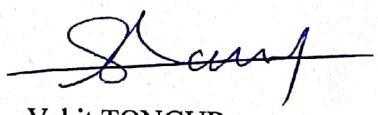
Prof. Dr. Mustafa YILMAZ
FBE Müdürü

TEZ BİLDİRİMİ

Bu tezdeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edildiğini ve tez yazım kurallarına uygun olarak hazırlanan bu çalışmada bana ait olmayan her türlü ifade ve bilginin kaynağına eksiksiz atıf yapıldığını bildiririm.

DECLARATION PAGE

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.



Vahit TONGUR

Tarih: 12.11.2018

ÖZET

DOKTORA TEZİ

AYRIK OPTİMİZASYON PROBLEMLERİNİN ÇÖZÜMÜNDE GÖÇMEN KUŞLAR OPTİMİZASYON (MBO) ALGORİTMASININ İYİLEŞTİRİLMESİ

Vahit TONGUR

Selçuk Üniversitesi Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Prof. Dr. Erkan ÜLKER

2018, 119 Sayfa

Jüri
Prof.Dr. Erkan ÜLKER
Prof.Dr. Harun UĞUZ
Doç.Dr. Mustafa Servet KIRAN
Dr.Öğr.Üyesi Mehmet HACIBEYOĞLU
Dr.Öğr.Üyesi Onur İNAN

Göçmen kuşlar optimizasyon (MBO) algoritması sürü zekasına dayanan meta-sezgisel bir optimziasyon algoritmasıdır. MBO algoritması tek sürüden oluşan bir popülasyona sahiptir. Popülasyon içerisindeki bireyler fayda mekanizması olarak adlandırılan bir iletişime sahiptir. Bu iletişim sayesinde sürüdeki bireyler en iyi sonuca sahip bireye benzemeye çalışırlar. Ancak bu durum algoritmanın yerel optimum noktalara takılmasına sebep olabilmektedir.

Bu tez çalışmasında ayrık problemlerin çözümü için tasarlanan MBO algoritması iyileştirilmiştir. Ayrıca farklı türden problemlerin çözümünde kullanılabilmesi için MBO'nun temel yapısını bozmadan yeni algoritmalar geliştirilmiştir. Yerel optimumlara takılmasını önlemek için MBO algoritmasına birden çok sürü dahil edilerek çok sürülü MBO (ÇS-MBO) algoritması geliştirilmiştir. Geliştirilen bu ÇS-MBO algoritmasında, her sürü birbirinden bağımsız olarak arama işlemi yaptırılmıştır. Global aramayı kuvvetlendirmek için ÇS-MBO algoritması yine bir optimizasyon algoritması olan parçacık sürü optimizasyon (PSO) algoritması ile birlikte kullanılarak melez bir ÇS-MBO (PSO-ÇS-MBO) algoritması geliştirilmiştir. Bu melez algoritma, ÇS-MBO'daki sürülerin PSO yardımıyla etkileşimde bulunmasını sağlamıştır. Bunun yanında ikili (binary) problemlerin çözümünde kullanılmak üzere MBO algoritmasının temel yapısı kullanarak ikili MBO (BMBO) algoritması geliştirilmiştir. BMBO algoritmasının performansını artırmak için çok sürülü BMBO (ÇS-BMBO) algoritması geliştirilmiştir.

Tüm iyileştirilen ve geliştirilen algoritmaların performansları literatürde bulunan problemler üzerinde test edilmiştir. Ayrık kombinatoryal problemlerden olan Gezgin satıcı problemi (GSP), çok boyutlu iki yönlü sayı bölümle problemi (ÇBİYSBP), graf boyama problemi (GBP) ve tek boyutlu kesim problemi (TBKP) MBO, ÇS-MBO ve PSO-ÇS-MBO algoritmaları ile çözülmüş ve performansları kıyaslanmıştır. Ayrıca ayrık ikili problemlerden olan ikili sırt çantası problemi (İSÇP) ve çok boyutlu ikili sırt çantası problemi (ÇBİSÇP) BMBO ve ÇS-BMBO algoritmaları ile çözülmüş ve sonuçları kıyaslanmıştır. Elde edilen sonuçlar, iyileştirilen ÇS-MBO ve PSO-ÇS-MBO algoritmalarının MBO algoritmasından daha başarılı olduğunu göstermiştir. Ayrıca geliştirilen BMBO ve ÇS-BMBO algoritmalarından elde edilen sonuçlara göre ÇS-BMBO algoritmasının BMBO algoritmasına göre daha başarılı olduğu görülmüştür.

Anahtar Kelimeler: Çok sürülü göçmen kuşlar optimizasyon algoritması, göçmen kuşlar optimizasyon algoritması, ikili göçmen kuşlar optimizasyon algoritması

ABSTRACT

Ph.D THESIS

IMPROVEMENT OF MIGRATING BIRDS OPTIMIZATION (MBO) ALGORITHM IN SOLUTION OF DISCRETE OPTIMIZATION PROBLEMS

Vahit TONGUR

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCE OF SELÇUK UNIVERSITY
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN COMPUTER ENGINEERING

Advisor: Prof. Dr. Erkan ÜLKER

2018, 119 Pages

Jury
Prof.Dr. Erkan ÜLKER
Prof.Dr. Harun UĞUZ
Assoc.Prof.Dr. Mustafa Servet KIRAN
Asst.Prof.Dr. Mehmet HACIBEYOĞLU
Asst.Prof.Dr. Onur İNAN

The migrating birds optimizaton (MBO) algorithm is a meta-heuristic algorithm based on swarm intelligence. The MBO algorithm has a population which consists of one swarm. Individuals in the population has a communication method called benefit mechanism. Owing to this communication method, individuals in the swarm try to converge to the individual which has the best solution. However, this may cause the algorithm to be stacked at local optimum.

In this thesis study, the MBO algorithm, which was developed to solve the discrete problems, was improved. Besides, without changing the fundamental structure of MBO, new algorithms were developed to be used in the solution of different types of problems. Multi flock MBO (MF-MBO) was developed by including more than one flock to the algorithm to avoid stacking of MBO at local optimum. In this developed MF-MBO, it was had the each flock searched individually. By using MF-MBO and Particle Swarm Optimization (PSO) together, a hybrid PSO-MF-MBO algorithm was developed to strengthen the global search. This hybrid algorithm provided that flocks in the MF-MBO interact with each other with the help of PSO. In addition, binary MBO algorithm (BMBO) was developed by using fundamental structure of MBO for the solution of binary problems. In order to improve the performance of the BMBO algorithm, a multi flock BMBO (MF-BMBO) algorithm was developed. The performances of all the algorithms developed and enhanced were tested with the problems existing in the literature. Some discrete combinatorial problems such as Traveling Salesman Problem, Multi-Dimensional Two-Way Number Partitioning Problem, Graf Coloring Problem, One-Dimensional Cuting Problem were solved and their performances were compared by using the algorithms of MBO, MF-MBO and PSO-MF-MBO. Additionally, binary problems such as Binary Knapsack Problem and Multi-Dimensional Knapsack Problem were solved and compared with the algorithms of BMBO and MF-BMBO.

The results showed that the enhanced MF-MBO and PSO-MF-MBO algorithms were much more successful than that of MBO. Besides, according to results obtained by the algorithm of the enhanced BMBO and MF-BMBO, it was observed that MF-BMBO was more successful than that of BMBO.

Keywords: Binary migrating birds optimization algorithm, migrating birds optimization algorithm, multi-flock migrating birds optimization algorithm

ÖNSÖZ

Bu çalışmada, bilgi ve tecrübeleriyle beni yönlendiren, desteğini ve anlayışını hiçbir zaman esirgemeyen ve değerli katkılarından dolayı danışmanım Sayın Prof. Dr. Erkan ÜLKER'e;

Tez izleme komitesinde yer alan ve tezin ilerlemesi ve geliştirilmesinde kıymetli görüşleriyle katkıda bulunan Sayın Doç. Dr. Mustafa Servet KIRAN ve Sayın Dr. Öğr. Üyesi Mehmet HACIBEYOĞLU'na;

Çalışmam boyunca misafirperverliğinden dolayı Selçuk Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümü öğretim elemanlarına;

Çalışmamın her anında yanımda olan ve manevi desteğini hiç esirmeyen sevgili eşim Evren'e ve aileme;

İçtenlikle teşekkürlerimi sunarım.

Vahit TONGUR
KONYA-2018

İÇİNDEKİLER

KISALTMALAR

AA	: Açgözlü arama
BMBO	: İkili göçmen kuşlar optimizasyon algoritması
ÇBİYSBP	: Çok boyutlu iyi yönlü sayı bölümleme problemi
ÇBÖE	: Çok büyük ölçekli entegrasyon
ÇBSÇP	: Çok boyutlu sırt çantası problemi
ÇS-MBO	: Çok sürülü göçmen kuşlar optimizasyon algoritması
DKA	: Değişken komşuluk arama
EM	: Elektromanyetizma
EPD	: En küçük pozisyon değeri
GA	: Genetik algoritma
GBP	: Graf boyama problemi
GSP	: Gezgin satıcı problemi
İÇBSÇP	: İkili Çok boyutlu sırt çantası problemi
İSÇP	: İkili sırt çantası problemi
İYSBP	: İki yönlü sayı bölümleme problemi
KAP	: Karesel atama problemi
MBO	: Göçmen kuşlar optimizasyon algoritması
MİB	: Merkezi işlemci birimi
PSO	: Parçacık sürü optimizasyonu
PSO-ÇS-MBO	: Parçacık sürü optimizasyonu tabanlı çok sürülü göçmen kuşlar optimizasyon algoritması
SBP	: Sayı bölümleme problemi
TBKP	: Tek boyutlu kesim problemi
TDP	: Tamsayı doğrusal programlama

1. GİRİŞ

Genel olarak optimizasyon, belirli kısıtlar altında en uygun çıktıyı bulmak için yapılan çalışmaların tamamı şeklinde tanımlanabilir. Başka bir deyişle belirli şartlar altında mümkün olan tüm çözümler arasından en iyi çözümü elde etmektir. Aranan çıktı veya en iyi çözüm, ele alınan probleme göre kimi zaman minimum kimi zaman da maksimum değer olabilir. Optimizasyon problemleri mühendislik, finans, endüstri ve daha birçok alanda karşımıza çıkabilmektedir. Karşımıza çıkan optimizasyon problemlerini çözebilmek için yıllar içerisinde birçok farklı yöntem ve yaklaşım kullanılmıştır. Problemler karşısında kesin çözüm elde eden istatistiksel ve matematiksel yöntemler her ne kadar başarılı sonuçlar elde etse de günümüz şartlarında çözülmesi gereken problemlerin boyutları ve karmaşıklığı giderek arttığından dolayı bu tür yöntemlerden bir sonucun elde edilmesi için gerekli zaman makul süreler dışına çıkabilmektedir. Bundan dolayı son zamanlarda optimizasyon problemlerinin çözümünde meta-sezgisel yöntemlerin de sıklıkla kullanıldığı görülmektedir. Meta-sezgisel yöntemler klasik matematiksel yöntemlerde olduğu gibi kesin sonucu garanti etmezler. Ancak optimum sonuca yakın değerler üretebilirler. Bunun yanında meta-sezgisel yöntemler, karmaşıklığı fazla olan problemleri makul süreler içerisinde çözebilirler.

Herhangi bir meta-sezgisel algoritmanın iki temel bileşeni şunlardır: Keşif ve çeşitlendirme. Keşif, mevcut iyi bir çözüme ait bilgilerinin kullanılarak yerel bir arama yapmak anlamına gelirken, çeşitlendirme, global ölçekte arama uzayını keşfetmek için çeşitli çözümlerin üretilmesidir. Bu iki bileşenin iyi bir şekilde yapılandırılması, genellikle global optimumun ulaşılabilir olmasını sağlar (Yang, 2010a). Meta-sezgisel algoritmalar birçok yönden sınıflandırılabilirler. Genellikle popülasyona ve yörüngeye dayalı sınıflandırma en genel olanıdır. Örneğin, parçacık sürü optimizasyon algoritması popülasyona dayalı bir algoritma iken benzetimli tavlama algoritmasında tek bir çözüm vardır.

Son yıllarda doğadan esinlenmiş birçok meta-sezgisel algoritma literatüre kazandırılmıştır. Özellikle doğadaki canlıların davranışları incelenerek geliştirilen birçok meta-sezgisel algoritma vardır. Parçacık sürü optimizasyonu, doğadan esinlenmiş ve popülasyon temelli bir algoritmadır. Bu algoritma, balık veya kuş sürülerinin yiyecek kaynağı arayışı esnasında sergiledikleri davranışlardan esinlenmiştir. Birbirlerinden etkilenerek hareket eden sürü bireyleri bu davranışlarıyla hem yiyecek kaynağına ulaşabilmektedirler hem de düşmanlarından korunabilmektedirler (Eberhart ve Kennedy,

1995). Karınca kolonisi algoritması da popülasyon temelli bir algoritma olup karıncaların yiyecek arayışı esnasında sergiledikleri davranışlardan esinlenilmiştir. Karıncalar yiyecek bulmak için yuvadan çıktıklarında geçtikleri yere feromon denilen özel bir kimyasal madde bırakırlar. Bu maddenin kokusu diğer karıncalar tarafından algılanır ve daha önce oradan başka bir karıncanın geçtiğini anlarlar. Geçtikleri yola bırakılan bu kimyasal madde zamanla buharlaşarak kaybolur. Eğer o yoldan çok sayıda karınca geçtiyse kimyasal maddenin kokusu da aynı oranda şiddetlidir ve yol güzergâhında bir yiyecek kaynağının olma ihtimali yüksektir. Yiyecek arayışına çıkmış bir karınca bu bilgileri yorumlayarak kokunun şiddetli olduğu yere doğru gitmeye başlar. Bu güzergâh aynı zamanda yiyecek kaynağı ile yuva arasındaki en kısa yolu oluşturur. Karıncaların bu karmaşık davranışlarından esinlenen karınca kolonisi algoritması ayrık optimizasyon problemlerinin çözümü için tasarlanmıştır (Dorigo ve Di Caro, 1999). Yapay arı kolonisi algoritması, bal arılarının nektar arayışını konu alan popülasyon temelli bir algoritmadır (Karaboga ve Basturk, 2007). Guguk kuşu algoritması doğa esinli bir başka optimizasyon algoritmasıdır. Bu algoritma da popülasyon tabanlıdır. Sürüler halinde gezen guguk kuşları, yumurtalarını başka kuşların yuvalarına bırakarak yavrularının bakımını başka kuş türlerine bırakırlar. Guguk kuşunun yumurtalarını bırakması için uygun bir yuva bulması gerekmektir. Bu yuva arayışı optimizasyon problemlerine uyarlanmıştır (Yang ve Deb, 2009). Son zamanlarda literatüre kazandırılan ağaç tohum algoritması da popülasyon tabanlı algoritmalardandır. Bu algoritma ağaçların tohumlarını etrafa saçarak yeni fidanların filizlenmesi olayından esinlenilmiştir. Doğada zamanla her canlı yaşlanır ve ölür. Bu ağaçlar için de geçerlidir. Ancak ormanlık alanların çok uzun süre ağaçlıklarla dolu olmasının sebebi yaşlanan ve ölen ağaçların yerini yenilerinin almasındandır. Ağaçlar belirli bir olgunluğa ulaştıklarında tohumlarını etrafına dökebilir veya dış etkenler vasıtasıyla bir ağacın tohumu çok uzak bölgelere taşınabilir. Tohumunu rastgele bir konuma düşüren bir ağacın etrafında yeni ağaçlar büyür. Bu doğa olayından esinlenen ağaç tohum algoritmasında, her ağaç bir çözümü temsil etmekte ve yeni filizlenen ağaçlar aday çözümleri temsil ederken, orman zeminine benzetilen arama uzayı en uygun çözümü bulmak için taranmaktadır (Kiran, 2015). Bu algoritmalara benzer şekilde; balina optimizasyon algoritması (Mirjalili ve Lewis, 2016), yarasa algoritması (Yang, 2010b), ateş böceği algoritması (Yang, 2010b), meyve sineği optimizasyon algoritması (Pan, 2012) ve buna benzer daha bir çok algoritma popülasyon tabanlı olup doğadan esinlenmiş algoritmalardır.

Benzetimli tavlama algoritması metallerin ısıtılıp soğutulmasından esinlenilmiştir. Metal yüksek sıcaklıklara maruz kaldıktan sonra yavaş yavaş soğutulması sonucunda metalin yapısında bir kristalleşme meydana gelir. Bu kristalleşme metalin moleküler yapısında değişikliğe neden olur. Benzetilmiş tavlama algoritması bu moleküler değişimi referans alarak geliştirilmiştir (Kirkpatrick ve ark., 1983).

Doğadan esinlenmeyen optimizasyon algoritmaları da görmek mümkündür. Bunlardan en iyi bilineni tabu arama algoritmasıdır. Ayrıca tabu arama algoritması popülasyon tabanlı bir algoritma da değildir. Benzetilmiş tavlama algoritması gibi tabu arama algoritması da yerel bir arama algoritmasıdır. Tabu arama algoritması, rastgele bir çözüm ile başlatılır ve her adımda bu çözüm iyileştirilmeye çalışılır. Bunu yaparken daha önce elde ettiği çözümleri tekrar değerlendirmemek için tabu isminde bir hafızaya atar. Hafızaya yerleşen hareket belirli bir süre boyunca tabu listesinde kalır. Mevcut çözümü geliştirirken hafızada daha önce yapılmış olan hareketlere de bakılır. Bu yasaklı hafıza ile algoritmanın hep aynı hareketi yapması önlenmiş olur. Ancak artık yeni çözümlerin üretilemeyeceği durumlar göz önüne alınarak, algoritmaya ismini veren bu listedeki yasak belirli bir süre sonra kaldırılır (Glover, 1989).

Günümüzde bir çok meta-sezgisel algoritma olmasına rağmen bunlardan çoğu optimizasyon probleminde aynı başarıyı elde edememektedirler. Wolpert ve Macready (1995) “No free lunch theorems for search” isimli çalışmada optimizasyon algoritmalarının bu durumunu incelemişlerdir. Bu yüzden çoğunlukla ele alınan problemi çözmek için kullanılan meta-sezgisel yöntemde sonucu iyileştirmeye yönelik değişikliklere ihtiyaç duyulmaktadır. Göçmen kuşlar optimizasyon algoritması (MBO) da optimizasyon problemlerinin çözümünde kullanılan popülasyon tabanlı meta-sezgisel algoritmalardan bir tanesidir. Bu tez çalışmasında MBO algoritmasının performansını artıracak yöntemler araştırılmış ve uygulanmıştır. Ayrıca orijinal MBO algoritmasının farklı türdeki problemleri çözebilecek kabiliyete sahip olması sağlanmıştır.

Yapılan bu tez çalışmasında, literatüre son yıllarda girmiş olan Göçmen Kuşlar Optimizasyon (MBO) Algoritması’nın performansının iyileştirilmesi için farklı yöntemler önerilmiştir. Bir çok meta-sezgisel algoritmada olduğu gibi, MBO algoritması da belirli optimizasyon problemlerinde çok başarılı sonuçlar verirken kimi optimizasyon problemlerinde beklenen performansı sergileyememektedir. Bu tez çalışması kapsamında, MBO algoritmasının her problemde başarılı olmasından ziyade başarı oranın düşük olduğu problemlerde de orijinal halinden daha iyi sonuçlar üretebilmesini sağlayacak yeni yöntemlerle donatılması sağlanmıştır. Böylelikle MBO algoritmasının

daha yaygın optimizasyon problemlerinde kullanılma imkanı sağlanmıştır. Bunun yanında, ayrık (discrete) problemler için tasarlanan MBO algoritmasının, ikili (binary) optimizasyon problemlerinde de kullanılabilmesi amacıyla BMBO ve BMBO'nun geliştirilmiş versiyonu olan ÇS-BMBO isminde iki yeni algoritma da üretilmiştir. Üretilen bu algoritmalar orijinal MBO algoritmasının temel yapısı kullanılarak geliştirilmiştir. Bu iki algoritmanın çözebileceği problemler de değerlendirildiğinde MBO algoritmasının farklı türdeki optimizasyon problemlerinde de rahatlıkla kullanılabilmesi sağlanmıştır.

Tezin organizasyonu şu şekildedir: kaynak araştırması başlığında MBO algoritması ile yapılmış çalışmalar anlatılmıştır, materyal ve yöntem başlığı altında orijinal MBO algoritması detaylı bir şekilde anlatılmıştır. Ayrıca tez çalışması kapsamında ele alınan problemler tanıtılmıştır. Aynı başlık altında geliştirilen yöntemlerden detaylı bir şekilde bahsedilmiştir. Araştırma sonuçları ve tartışma başlığında orijinal MBO ve geliştirilen yöntemlerin, ele alınan problemlere nasıl uygulandığı detaylandırılmış ve elde edilen sonuçlar tablolar halinde verilmiştir. Son olarak sonuçlar ve öneriler başlığında elde edilen sonuçların genel bir değerlendirmesi yapılmış ve ilerideki çalışmalara yön verecek önerilerde bulunulmuştur.

2. KAYNAK ARAŞTIRMASI

Bu bölümde ilk kez 2012 yılında literatüre giren MBO algoritması ile bugüne kadar yapılmış olan çalışmalar ele alınmıştır.

MBO algoritması ilk olarak Duman ve ark. (2012) tarafından literatüre kazandırılmıştır. MBO, göçmen kuşların göç ederken sergiledikleri "V" dizilimindeki uçuş düzenlerinden esinlenmiş doğa esinli bir optimizasyon algoritmasıdır. Ayrık problemleri çözmek için tasarlanmış bu algoritmanın performansı karesel atama problemleri (KAP) üzerinde test edilmiştir. Yazarlar literatüre kazandırdıkları MBO algoritmasının genetik algoritma, tabu arama, parçacık sürü optimizasyonu, diferansiyel evrim algoritması, dağıtık arama ve rehberli evrimsel benzetimli tavlama algoritmalarından daha iyi sonuçlar elde ettiğini raporlamışlardır.

Gao ve ark. (2013) MBO algoritmasındaki popülasyonu birden çok sürüye bölerek geliştirilmiş bir MBO algoritmasını beklemesiz iş akış çizelgeleme probleminin çözümü için önermişlerdir. Başlangıç popülasyonunu oluşturmak için NEH, SDH ve EDY'den oluşan üç sezgisel yöntemden faydalanmışlardır. Komşuluk yapısı olarak takas ve araya ekleme yöntemlerini kullanmışlardır.

Duman ve Elikucuk (2013) MBO algoritmasını kredi kartı dolandırıcılığı algılama problemine uygulamışlardır. Orijinal MBO'nun performansını artırmak için fayda mekanizması üzerinde bazı değişikliğe gitmişlerdir.

Makas ve Yumusak (2013) iki farklı meta-sezgisel algoritma olan MBO ve yapay arı kolinisi (ABC) algoritmalarını birlikte kullanarak melez bir algoritma ile MBO algoritmasının komşu üretme parametresini iterasyona bağlı olarak azaltarak değişken komşuluk yapısına sahip bir MBO algoritmasını birlikte önermişlerdir. Önerilen algoritmalara standart test fonksiyonları kullanılarak bir dizi performans testi uygulamışlardır. Ayrıca önerilen yöntemler UCI ve KEEL web sitelerinden dokuz farklı veri seti için uygulanan sinir ağlarını eğitmek için kullanmışlardır. Elde ettikleri sonuçlar, önerilen yöntemlerin, global optimumlara daha iyi yakınsamalar yaparak orijinal MBO'yu geride bıraktığını göstermiştir.

Alkaya ve ark. (2014) MBO'nun komşuluk yapısını değiştirerek çok boyutlu sürekli arama uzayında MBO'nun performansını incelemişlerdir.

Niroomand ve ark. (2015) genetik algoritma operatörlerini kullanarak değiştirilmiş bir MBO algoritmasını esnek üretim sistemlerinde kapalı döngü yerleşim problemine uygulamışlardır. Orijinal MBO üzerindeki değişikliği MBO'nun komşu

çözüm üretim aşamasında gerçekleştirilmişlerdir. Komşu çözümler üretirken genetik algoritmanın çaprazlama yöntemlerini kullanmışlardır. Çaprazlama yöntemi olarak kısmi eşlemeli çaprazlama (Partial Mapped Crossover-PMX) yöntemini kullanmışlardır. Bu yöntemle elde ettikleri yeni bireyler üzerinde genetik algoritmadaki mutasyon operatörünü uygulamışlardır. Bu yolla değiştirdikleri MBO algoritmasından elde ettikleri sonuçları orijinal MBO ve benzetimli tavlama algoritmaları ile kıyaslamışlar ve önerdikleri algoritmanın daha iyi sonuçlar verdiğini belirtmişlerdir.

Soto ve ark. (2015) makine parçası hücre formasyon problemini MBO algoritması ile çözmüşlerdir. Yaptıkları deneysel çalışmalarda, test problemlerinin çoğunda, önerilen yaklaşımdan elde edilen sonuçların, literatürde bilinen diğer yöntemlerden elde edilen sonuçlardan daha iyi olduğunu göstermişlerdir.

Soto ve ark. (2016) makine parçası hücre oluşum problemlerinin çözümünde MBO algoritmasını kullanmışlardır.

Gao ve Pan (2016) küçük sürülerden oluşan bir mikro-MBO algoritmasını dinamik karıştırma işlemi ve çeşitlilik kontrol yöntemleri ile birleştirerek çok kaynaklı-kısıtlı iş akış çizelgeleme problemini çözmek için kullanmışlardır.

Zhang ve ark. (2017) melez iş akış çizelgeleme probleminin çözümü için efektif bir MBO algoritması önermişlerdir. Önerdikleri algoritmada komşuluk operatörlerini birleştiren kombine bir komşuluk arama stratejisi geliştirmişlerdir. Ayrıca daha iyi çözümler bulma olasılığını artırmak için V formasyonunda sıralanmış sürünün her iki kolu arasında etkileşimi güçlendirmeye yönelik rekabetçi bir mekanizma kullanmışlardır. Glover operatörüne dayanan keşif aşaması ve iyi tasarlanmış bir yerel arama yöntemini, yerel optimumlara takılmış bireylere uygulamışlardır. Bu sayede algoritmanın potansiyel gelecek vaat eden alanları keşfetmesine yardımcı olmasını sağlamışlardır.

Benkalai ve ark. (2017) bir permütasyon iş akış ortamında bir grup makinede kurulum süreleriyle birlikte bir dizi bağımsız işin çizelgelenmesi problemini çözmek için geliştirilmiş bir MBO algoritması önermişlerdir. Geliştirilmiş MBO algoritmasında komşuluk yapısı olarak Or-opt komşuluk yapısını kullanmışlardır. Geliştirdikleri MBO algoritmasının performansını test edebilmek için ele aldıkları problemleri orijinal MBO algoritması ile de çözmüşlerdir. Geliştirilen yöntemin orijinal MBO algoritmasından daha iyi sonuçlar verdiğini raporlamışlardır.

(Oz, 2017) çok amaçlı görev paylaşımı problemi için geliştirilmiş bir MBO algoritması kullanmıştır. Çalışmasında orijinal MBO algoritmasının komşuluk yöntemi olarak probleme özgü bir komşu fonksiyon tasarlamıştır. Bir çözümün komşu fonksiyonu

olarak hem yerel arama hem de çeşitlendirme mekanizmalarına sahip açgözlü bir fonksiyon tanımlamıştır. Bu yöntemle algoritmanın yerel arama ile global arama arasındaki tercihini değiştirmek için dinamik bir mekanizma sağlamıştır.

Sioud ve Gagne (2018) diziye bağlı kurulum süreleriyle iş akış problemini geliştirilmiş bir MBO algoritması ile çözmüşlerdir. MBO'nun yerel aramasını kuvvetlendirmek için komşuluk yapısı içerisine bir tabu listesi ve yeniden başlatma mekanizması yerleştirmişlerdir. Bunun yanında yeni liderin seçim sürecinde orijinal bir metot ekleyerek MBO'nun performansını iyileştirmişlerdir.

Meng ve ark. (2018) iş bölümlü iş akış çizelgeleme problemini geliştirilmiş bir MBO algoritması ile çözülmüşlerdir. Önerdikleri algoritmada harmoni arama tabanlı bir komşuluk yöntemi geliştirmişlerdir. Buna ilaveten yerel optimumlardan kaçınmak için bir sıçrama mekanizması da geliştirmişlerdir.

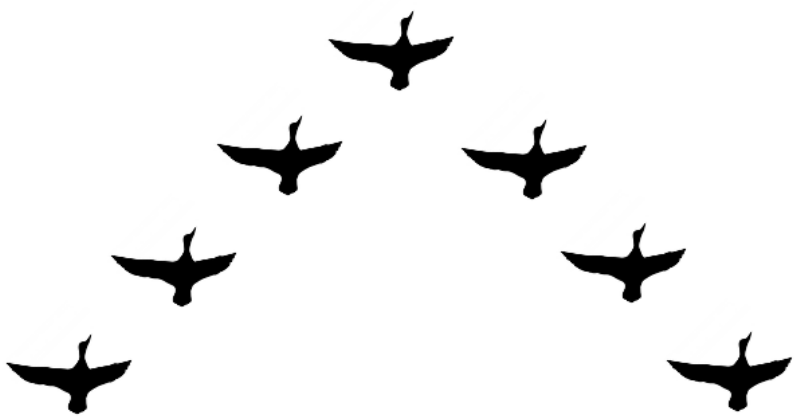
3. MATERYAL VE YÖNTEM

Tez çalışması kapsamında, göçmen kuşlar optimizasyon (MBO) algoritması üzerine iyileştirme çalışmaları yapılmıştır. Yapılan bu çalışmalarda MBO algoritmasının temel yapısı kullanılmıştır. Bunun yanında, ikili (binary) optimizasyon problemlerinin çözümünde kullanılmak üzere MBO algoritmasından türetilmiş yeni bir algoritma geliştirilmiştir.

İyileştirilen ve geliştirilen yöntemlerin başarısını görebilmek için orijinal MBO ile MBO'nun iyileştirilmiş ve geliştirilmiş versiyonları, literatürde iyi bilinen optimizasyon problemleri üzerine uygulanmıştır. Problemlerin çözümünden elde edilen sonuçlara göre algoritmaların performansları kıyaslanmıştır.

3.1. Göçmen Kuşlar Optimizasyon Algoritması (MBO)

Göçmen kuşlar optimizasyon algoritması (MBO) ilk olarak Duman ve ark. (2012) tarafından önerilmiştir. Algoritma, göçmen kuşların göç ederken sergiledikleri "V" diziliminden sağladıkları enerji tasarrufundan esinlenmiştir. MBO algoritması ayrık problemler için tasarlanmış ve gerçek yaşam problemlerinden doğan karesel atama problemleri (KAP) üzerinde test edilmiştir.

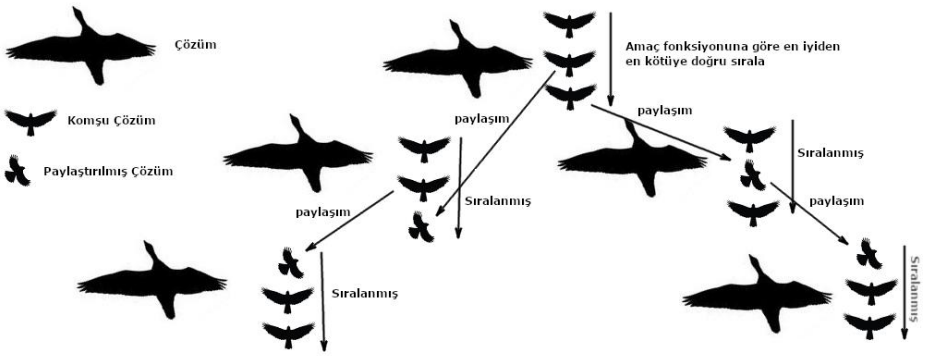


Şekil 3.1. MBO algoritmasında V formasyonundaki sürü

Göçmen kuşların havadayken daha uzun mesafeleri uçabilmeleri için kullandıkları en yaygın uçuş şekli V formasyonudur. Bu formasyonun temel dürtüsünün enerji tasarrufu olduğu bilinmektedir. V formasyonundaki lider kuş en çok enerji harcayan kuştur. Diğer pozisyondaki kuşlar, öndeki kuşların uçuş esnasındaki kanat çırpışından dolayı oluşturdukları girdaptan faydalanarak daha az enerji harcayarak daha uzun süre havada kalabilmektedirler. Farklı büyüklükteki göçmen kuşların V şeklindeki dizilimi de farklı olmaktadır. Bunun nedeni farklı kuşların farklı kanat uzunluklara sahip olmasıdır. Öndeki kuşun oluşturduğu girdaptan faydalanabilmek için onu takip eden kuşun yerleşimi belli bir mesafe ve açıyla gerçekleşmektedir. Bu yüzden farklı büyüklükteki kuşların oluşturdukları V formasyonları farklı büyüklükte olabilmektedir.

MBO algoritması mevcut çözümlerle başlatılır ve bu çözümler her adımda geliştirilmeye çalışılır. Başlangıç çözümler rastgele belirlenir. Daha sonra algoritma içerisindeki komşuluk yapısı ve paylaşım mekanizmaları kullanılarak mevcut çözümler optimum sonuca doğru yaklaştırılmaya çalışılır. Algoritma üzerinde bu işlemler gerçekleştirilirken gerçek hayattaki kuşların V dizilimindeki enerji tasarrufundan esinlenilir. V dizilimindeki her kuş arama uzayında bir çözümü temsil etmektedir. Bu çözümlerin bölgelerinde yerel arama yapabilmesi için mevcut çözümden komşu çözümler üretilir. Bundan dolayı MBO algoritmasında komşuluk arama yöntemi kullanılmaktadır. Duman ve ark. (2012) KAP problemleri için komşu çözüm üretimini mevcut çözüm permütasyonunun sadece bir çiftini yer değiştirerek (takas işlemi yaparak) gerçekleştirmişlerdir. Her çözüm için daha önceden belirlenmiş ve parametre olarak verilen değer kadar komşu çözüm üretilir. Daha sonra yeni üretilen bu komşu çözümler amaç fonksiyonuna göre iyiden kötüye doğru sıralanır. En iyi komşu çözüm mevcut çözümü geliştirmek için kendisine ayrılırken kalan diğer en iyi çözümler sırasıyla yine parametre olarak verilen ve önceden belirlenmiş değer kadar kendisinden sonra gelen kuşa aktarılır. Bu aktarıma komşu çözüm paylaşımı (fayda mekanizması) denir. Bu komşu çözüm paylaşımı, MBO algoritmasını diğer meta-sezgisel algoritmalardan ayıran en önemli özelliktir. Bu sayede sürüdeki her kuş birbirleriyle etkileşim halinde olmakta ve çözüme daha hızlı bir şekilde yakınsamaktadırlar. Komşu çözüm üretimi ve paylaşımı Şekil 3.2'de gösterilmiştir. Kuşların komşu çözüm üretme ve paylaşma süreci lider kuşun parametre olarak verilen kanat çırpma sayısı kadar tekrar ettirilir. Kanat çırpma değerine ulaşıldığında lider kuş dinlenmek için sürünün arkasına gönderilir. Yorulan lider kuş ilk olarak sürünün sol kanadının en sonuna gönderilirken sol koldaki lideri takip eden kuş liderin pozisyonuna atanır ve sol kanattaki diğer kuşlar da sırasıyla birbirlerinin yerine

gönderilirler. Bu aşamada kanat çırpma parametresi sıfırlanır ve yeni bir dizilim elde edilir. Elde edilen bu yeni dizilimle çözüm arama işlemine devam edilir. Yeni lider kuş tekrar yorulduğunda (kanat çırpma sayacı parametre olarak önceden belirlenen değere ulaştığında) bu defa sürünün sağ kanadının en sonuna gönderilir ve sağ kanatta liderin hemen arkasındaki kuş, lider pozisyonuna atanır ve sağ kanattaki diğer kuşlar da sırasıyla birbirlerinin yerine gönderilirler. Bu işlem algoritma sonlandırılıncaya kadar devam ettirilir.



Şekil 3.2. Komşu çözüm üretimi ve komşu paylaşımı (*k=3 ve x=1*)

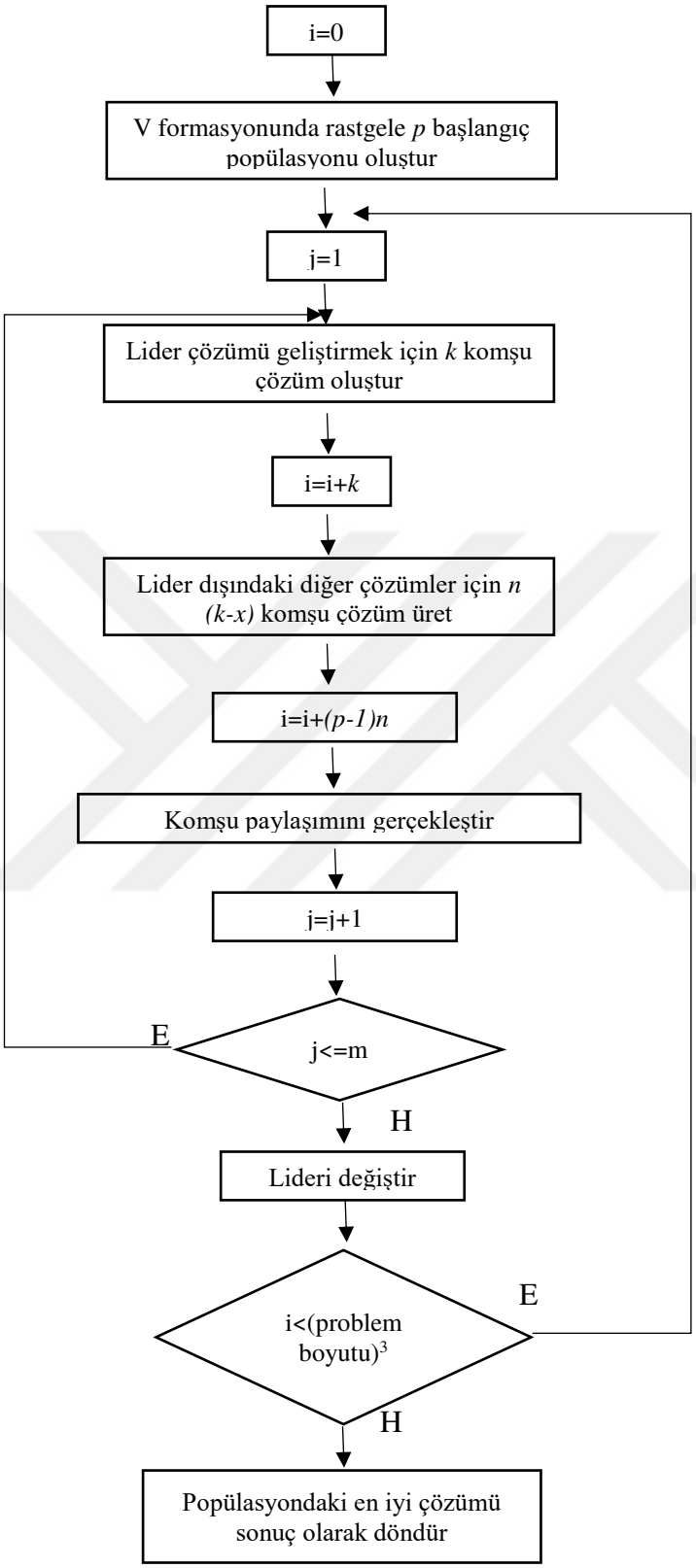
MBO algoritmasında durdurma kriteri genelde problem boyutuna bağlı olmaktadır. Duman ve ark. (2012) algoritmanın çalışma süresini, toplam üretilen komşu çözüm sayısının problem boyutunun küpüne eşit olacak şekilde belirlemişlerdir. Algoritmanın çalışma süresinin bu şekilde belirlenmesi sürüdeki her kuşun en az bir kez lider pozisyona geçebileceği kadar büyüklükte olmasını sağlamak amacıyla olduğu düşünülmektedir.

MBO algoritmasında kullanılan parametreler Çizelge 3.1'de verilmiştir. Bu parametreler daha sonra başlıklar halinde detaylı bir şekilde anlatılmıştır.

Çizelge 3.1. MBO algoritmasının parametreleri

Parametre	Açıklaması
p	Başlangıç çözüm sayısı (Popülasyon)
k	Lider çözüm için komşu çözüm sayısı
n	Kalan çözümler (lider dışında) için komşu çözüm sayısı
x	Komşu çözümlerin paylaşım sayısı
m	Kanat çırpma sayısı (tur)

MBO algoritmasına ait akış diyagramı Şekil 3.3'te verilmiştir.



Şekil 3.3. MBO algoritmasının akış diyagramı

3.1.1. Başlangıç Popülasyonu (kuş sayısı)

MBO algoritması belirli sayıda birey içeren başlangıç popülasyonu ile başlatılır. Popülasyon büyüklüğü başlangıçta sabit olarak verilir ve algoritma sonlanıncaya kadar aynı kalır. Bu değer en az üç olmalı ve tek sayılardan seçilmelidir. Bunun nedeni sürünün V formasyonunu oluşturmasından kaynaklanmaktadır. Popülasyon değeri üç seçildiğinde bireylerden bir tanesi lider atanır ve liderin sağına ve soluna birer birey yerleştirilir. Eğer sürü yirmi beş bireyden oluşturulursa bu durumda yine bireylerden birisi lider atanır ve geriye kalan yirmi dört bireyden on ikisi liderin soluna diğer on ikisi de liderin sağına yerleştirilir. Örnek bir popülasyon Şekil 3.1'de verilmiştir.

Şekil 3.1'de yedi bireyden oluşan bir popülasyon görülmektedir. Bireylerden birisi lider iken kalan altı birey üçerli iki grup halinde bireyin sol ve sağ tarafına yerleştirilmişlerdir.

3.1.2. Komşu çözüm üretimi ve paylaşımı

MBO algoritmasında çözümlerin geliştirilebilmesi için mevcut çözümlerin etrafında yeni çözümler aranır. Kendi çözümünden farklı ama yine kendi çözümünden çok uzak olmayan bu yeni çözümlere komşu çözümler denir. MBO'da komşu çözüm üretimi için birden çok yöntem kullanılabilir. Duman ve ark. (2012) KAP problemini MBO ile çözerlerken komşu çözüm üretme yöntemi olarak takas (swap) yöntemini kullanmışlardır. Ancak bu yöntemin başarısı probleme göre değişebilmektedir. Takas yönteminin dışında sık olarak kullanılan bir başka yöntem ise araya ekleme (insertion) yöntemidir. Her iki yöntem için ele alınan probleme göre yöntemler test edilip başarısına göre bu yöntemlerden birisi seçilebilir. Aynı problem için her iki yöntemin kullanıldığı melez yöntemler de mevcuttur. Bunlar ele alınan problemin çözümünde test edilebilir.

MBO'da komşu çözüm sayısı algoritmanın başlangıcında parametre olarak verilir ve algoritma sonlanıncaya kadar aynı kalır. Ancak komşu çözüm üretme sayısı, lider ve diğer bireyler için farklı değerdedir. Toplamda her birey aynı komşu çözüme sahip olmasına karşın üretilen komşu çözümlerin sayısındaki farklılık komşu paylaşımından kaynaklanmaktadır.

Lider ve diğer bireylerin komşu çözüm üretme sayıları Denklem 3.1, 3.2 ve 3.3'e göre belirlenir.

$$s \in N^{+}; k = 2s + 1 \tag{3.1}$$

$$1 \leq x \leq \left(\frac{k-1}{2}\right) \tag{3.2}$$

$$n = k - x \tag{3.3}$$

Burada, $s > 0$ ve bir doğal sayıdır, k lider birey için üretilecek komşu çözüm sayısını göstermektedir, x paylaşılan komşu çözüm sayısını göstermektedir ve n lider dışındaki diğer bireyler için üretilecek komşu çözüm sayısını göstermektedir. Denklem 3.1'e göre, lider üç komşu çözüm ürettiğinde *(k=3)* komşu paylaşım sayısı Denklem 3.2'ye göre 1 olmak zorundadır *(x=1)*. Bu durumda lider dışındaki diğer bireylerin komşu çözüm üretme sayıları Denklem 3.3'e göre iki olarak hesaplanır *(n=3-1=2)*. Buradan hareketle, komşu çözüm paylaşımı yapıldıktan sonra lider ve sürüdeki diğer bireylerin toplam komşu çözüm sayılarının eşit olduğu açıkça görülebilmektedir. Bu durum Şekil 3.2'de gösterilmiştir.

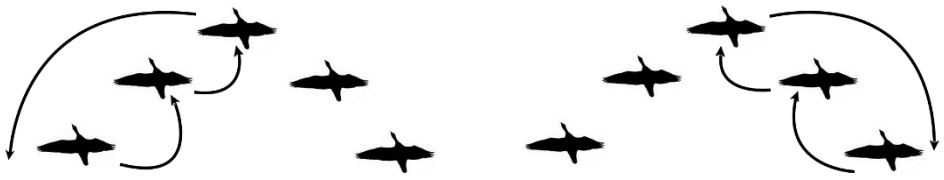
Şekil 3.2'ye göre, lider kuş için üç komşu çözüm üretilir ve bu çözümler paylaştırılmadan (diğer bireylere aktarılmadan) önce amaç fonksiyonuna göre iyiden kötüye doğru sıralanır. En iyi komşu çözüm mevcut çözümü geliştirmek için rezerve edilirken en iyi ikinci çözüm kendisinin solundakine ve en iyi üçüncü çözüm kendisinin sağındaki bireyin çözüm kümesine aktarılır. Şekil 3.2'den lider bireyin dışındaki bireyler için iki çözüm üretildiği görülmektedir. Lider dışındaki bireye kendi önündeki bireyden komşu çözüm paylaşımı yapıldığında ve paylaşılan komşu çözümün, kendisi için üretilen komşu çözümlerle birleştirildiğinde toplamda sürüdeki tüm bireyler eşit sayıda komşu çözümlere sahip olduğu görülebilmektedir. Paylaşılan komşu çözümü alan arkadaki birey, kendisi için üretilen komşu çözümlerle birlikte amaç fonksiyonuna göre yeniden değerlendirilir. Komşu çözümler iyiden kötüye doğru sıralanır. En iyi komşu çözüm kendi çözümünü geliştirmek için rezerve edilirken, en iyi ikinci çözüm kendisinden sonra gelen bireye aktarılır. Çözüm kümesindeki eleman sayısı sabit olduğundan dolayı paylaşımdan sonra mevcut çözümün kendisinde kalan en kötü komşu çözüm ya da çözümler atılır. Sürüde kendi komşu çözümlerini diğer bireylerle paylaşmayan sadece iki birey vardır. Bunlar sürünün hem sağ hem de sol tarafın en arkasında bulunan iki bireydir. Bu bireyler için önündeki bireyden gelen komşu çözüm kabul edilir ancak kendisinden sonra birey olmadığı için bunların çözümleri paylaştırılamazlar. Eğer *x=2* değerini alırsa bu durumda $k >= 5$ olmalıdır. Bu durumda lider için üretilen komşu çözümlerden en iyi komşu çözüm liderin geliştirilmesi için rezerve edilirken; ikinci en iyi liderin soluna, üçüncü en iyi liderin sağına, dördüncü en iyi yine liderin soluna ve beşinci en iyi (son

komşu çözüm) liderin yine sağına olacak şekilde ardışık paylaştırılır. Eğer $k=7$ ve $x=2$ ise bu durumda lider için üretilen komşu çözümler sıralandıktan ve gerekli paylaşımlar yapıldıktan sonra kalan en sondan iki komşu çözüm atılır.

3.1.3. Kanat çırpma ve lider değişimi

Kanat çırpma parametresi de diğer parametrelerde olduğu gibi algoritmanın başlangıcında belirlenir ve algoritma sonlandırılıncaya kadar aynı kalır. Kanat çırpma parametresi, sürüdeki lider değişiminin hangi sıklıkta yapılacağını belirler. MBO'da V formasyonundaki sürünün dizilimi belirli aralıklarla değiştirilir. Her değişim gerçekleştirildiğinde sürünün belirli bir süre boyunca aynı dizilimde kalması sağlanmalıdır. Bu süreyi kanat çırpma parametresi belirler. Sürüdeki ilk lider değişimi öncelikle sürünün sol kanadında gerçekleştirilir. Bu değişim gerçekleştikten sonra kanat çırpma parametresi sıfırlanır. İkinci değişim zamanı geldiğinde bu defa değişim sürünün sağ kanadı üzerinde gerçekleştirilir ve tekrar kanat çırpma parametresi sıfırlanır. Bu işlem algoritmanın çalışması sonlandırılıncaya kadar devam ettirilir ve değişim sırasıyla sürünün bir sol kanadında bir sağ kanadında gerçekleştirilir.

Sürünün sol kanadında gerçekleştirilen lider değişiminde, sürünün lideri sürünün sol kanadının en arkasına gönderilirken lideri takip eden birey liderin konumuna geçer. Bireyin yer değişiminden kaynaklanan aradaki boşluğu doldurmak için sürünün sol kanadındaki bireyler önlerindeki boşluğu dolduracak şekilde bir basamak lidere doğru yaklaştırılırlar. Sürünün sağ kanadındaki lider değişimi de sol kanadındaki değişimle aynı prosedürlere sahiptir. Şekil 3.4'te örnek bir lider değişimi gösterilmiştir.



Şekil 3.4. Lider değişimi

3.1.4. Durdurma kriteri (İterasyon)

Durdurma kriteri (iterasyon), algoritmanın ne kadar süreyle çalıştırılacağının belirlendiği parametredir. Duman ve ark. (2012) KAP probleminin çözümünde iterasyon parametresini problemin boyutuna bağlı olarak belirlemişlerdir. Algoritmanın başlangıcından itibaren üretilen toplam komşu çözüm sayısı, ele alınan problem boyutunun küpüne eşit olacak şekilde belirlenmiştir. Buradan da anlaşılabiliyor ki MBO algoritmasında ele alınan problemin boyutu büyüdükçe algoritmanın çalışma zamanı da artmaktadır.

MBO algoritmasının sözde kodu Algoritma 3.1'de verilmiştir.

p = Başlangıç çözüm sayısı (Popülasyon)
k = Lider çözüm için komşu çözüm sayısı
n = Kalan çözümler (lider dışında) için komşu çözüm sayısı
x = Komşu çözümlerin paylaşım sayısı
m = Kanat çırpma sayısı (tur)

p adet rastgele başlangıç çözüm üret ($S_1, S_2, \ldots, S_p$)
Çözümlerden birini lider olarak ata
Kalan $p - 1$ adet çözümden sol ($S_{L1}, S_{L2}, \ldots, S_{L(p-1)/2}$) ve sağ ($S_{R1}, S_{R2}, \ldots, S_{R(p-1)/2}$) kanadı oluşturacak iki liste oluştur
soltaraf = *true*
$i = 0$
while $i < (problemboyutu)^3$ **do**
  for $j = 0$ **to** m **do**
    Lider için k adet komşu çözüm üret ve amaç fonksiyonuna göre en iyiden en kötüye doğru sırala ($LB_1, LB_2, \ldots, LB_k$)
    $i = i + k$
    if $f(LB_1) < f(lider)$ **then** //minimizasyon problemi için
      lider = LB_1
    end if
    SL_1'in komşu çözümlerine paylaşımdan gelen $LB_2, LB_4, \ldots, LB_{k-1}$ çözümlerini ekle
    SR_1'in komşu çözümlerine paylaşımdan gelen $LB_3, LB_5, \ldots, LB_k$ çözümlerini ekle
    $t = 1$
    while $t < (p - 1) / 2$ **do**
      S_{Lt} çözümü için $k - x$ adet komşu çözüm üret ($NL_1, NL_2, \ldots, NL_{k-x}$) ve amaç fonksiyonuna göre en iyiden en kötüye doğru sırala
      S_{Rt} çözümü için $k-x$ adet komşu çözüm üret ($NR_1, NR_2, \ldots, NR_{k-x}$) ve amaç fonksiyonuna göre en iyiden en kötüye doğru sırala
      $i = i + 2(k - x)$
      if $f(NL_1) < f(S_{Lt})$ **then**
        $S_{Lt} = NL_1$
      end if
      if $f(NR_1) < f(S_{Rt})$ **then**
        $S_{Rt} = NR_1$
      end if
      $S_{L(t+1)}$'in komşu çözümlerine $NL_2, NL_3, \ldots, NL_{x+1}$ çözümlerini ekle
      $S_{R(t+1)}$'in komşu çözümlerine $NR_2, NR_3, \ldots, NR_{x+1}$ çözümlerini ekle
      $t = t + 1$
    end while
  end for
  if *soltaraf* = *true* **then**
    lider çözümü sürünün sol tarafının en arkasına gönder ve S_{L1}'i yeni lider olarak ata
  else
    lider çözümü sürünün sağ tarafının en arkasına gönder ve S_{R1}'i yeni lider olarak ata
  end if
  soltaraf = *!soltaraf*
end while
sürüdeki en iyi çözümü sonuç olarak ver

Algoritma 3.1. MBO Algoritması

3.2. Optimizasyon Problemleri

Optimizasyon problemlerini genel olarak ayrık ve sürekli optimizasyon problemleri şeklinde iki sınıfa ayırmak mümkündür.

Bu tez çalışmasında ele alınan orijinal MBO ile geliştirilen ve iyileştirilen MBO algoritmalarının performansını görmek için; Gezgin satıcı problemi (GSP), çok boyutlu iki yönlü sayı bölümleme problemi (ÇBİYSBP), ikili sırt çantası problemi (İSÇP), ikili çok boyutlu sırt çantası problemi (İÇBSÇP), tek boyutlu kesim problemi (TBKP) ve graf boyama problemi (GBP) gibi ayrık problemler çözülmüştür. Ele alınan problemler başlıklar halinde açıklanmıştır.

3.2.1. Gezgin satıcı problemi (GSP)

GSP yöneylem araştırması alanında incelenen kombinatoryal bir optimizasyon problemidir. GSP, ismini bir tam tur süresince tüm şehirlerdeki müşterilere seyahat etmesi gereken bir satıcının yaptığı davranıştan alır. GSP'nin amacı, belirli sayıda şehri, başlangıç noktasından itibaren her şehre yalnız bir kez uğramak kaydıyla gezen ve başlangıç noktasına geri dönen en kısa yolu bulma problemidir (Zhong ve ark., 2007; Ma ve ark., 2008). Bu problem türü araç rotalama, baskı devresi yerleşim problemleri, çizelgeleme tabloları, esnek üretim sistemleri gibi çeşitli alanlarda da uygulanmıştır. GSP, kombinatoryal optimizasyon problemleri için sağlam bir temel ve zengin deneyimler sağlayabilir. Bu yüzden, matematik, biyoloji, mühendislik ve diğer alanlardan birçok araştırmacının ilgisini çekmiştir.

GSP aşağıdaki gibi tanımlanabilir;

$G = (N, A)$ skaler bir graf olsun. $N = \{1, 2, 3, \ldots, t\}$ şehirleri ve $A = N \times N$ şehirler arasındaki yolları göstersin. Bu durumda şehir çiftleri arasındaki uzaklık matrisi $D = \left(d_{ij}\right)_{t \times t}$ olarak ifade edilebilir. Ayrıca bir çözümün permütasyonu $\pi = \{\pi_1, \pi_2, \ldots, \pi_t\}$ olarak gösterilebilir. Bu durumda GSP probleminde amaç fonksiyonu Denklem 3.4'teki gibi ifade edilebilir.

$$minf(\mu) = \sum_{i=1}^{t-1} d_{\pi_i \pi_{i+1}} + d_{\pi_t \pi_1} \tag{3.4}$$

Burada D tüm şehirlerarası çiftlerin matrisini temsil ederken d_{ij} i şehri ile j şehri arasındaki mesafeyi göstermektedir. Simetrik GSP problemlerinde graflar yönsüzdür ve her zaman $d_{ij} = d_{ji}$ eşitliği sağlanır.

GSP'nin çözümü kolay gibi görünmesine rağmen, hesaplama karmaşıklığı şehirlerin sayısı ile üssel olarak artar. Bu yüzden GSP bir NP-zor kombinatoryal optimizasyon problemidir ve makul süreler içerisinde geleneksel metotlarla çözülemeyebilir (Laporte, 1992).

GSP problemi zorluğundan dolayı birçok araştırmacının ilgisini çekmiş ve meta-sezgisel yöntemlerle çözülmeye çalışılmıştır. GSP problemi şimdiye kadar birçok yöntemle çözülmeye çalışılmıştır. Helsgaun (2000) Gezgin satıcı problemini Lin-Kernighan sezgiseyli çözmüştür. Çalışmasında, kullandığı yöntemin çok çeşitli boyutlardaki GSP probleminin çözümünde başarılı olduğunu belirtmiştir. Meer (2007) benzetilmiş tavlama algoritmasında 2-opt sezgiselini kullanarak GSP'yi çözmüştür. Zhong ve ark. (2007) GSP'nin çözümü için C3DPSO olarak isimlendirilen ayrık bir parçacık sürü optimizasyon algoritması önermişlerdir. Önerilen algoritmada güncelleme formülü değiştirilmiş ve yerel ve global arama arasındaki dengeyi korumaya yardımcı olmak için mutasyon faktörü olarak isimlendirilen yeni bir parametre eklemişlerdir. Ayrıca GSP'nin çözümünde kullanılan diğer birçok meta-sezgisel algoritmadan farklı olarak önerdikleri algoritmada parçacıklar, şehirlerin sıralamasını belirleyen permütasyonları değil şehirler arasındaki yolları temsil eden kenar kümelerinin setlerini tutmaktadır. TSPLIB kütüphanesinden seçtikleri test problemlerinden elde ettikleri sonuçlara göre 200 şehir içeren problemlerde dahi önerdikleri yöntemin başarılı sonuçlar verdiğini belirtmişlerdir. Shi ve ark. (2007) GSP için yeni bir parçacık sürüsü optimizasyon algoritması sunmuşlardır. Yakınsama hızını hızlandırmak için belirsiz bir arama stratejisi ve bir çaprazlama tekniği kullanmışlardır. GSP'nin çözümünde kullanılan sürü zekalı diğer mevcut algoritmalar ile karşılaştırdıklarında, çözülen problemlerin büyüklüğünün önerilen algoritma kullanılarak arttırılabileceğini belirtmişlerdir. Bunun yanında GSP'yi genelleştirilmiş kromozom yapısı kullanarak çözmek için başka bir PSO tabanlı algoritma önermişlerdir. Yakınsama hızını artırmak için iki yerel arama tekniği kullanmışlardır. Deneysel sonuçlar, önerilen algoritmaların etkinliğini göstermiştir. Zhong ve ark. (2008) GSP'nin çözümünde tabu arama algoritmasını kullanmışlardır. Tabu arama algoritmasında komşuluk yöntemi olarak ters çevirme (devrik) ve araya ekleme yöntemlerini kullanmışlardır. Ayrıca tabu aramada önceki hareketin bilgisini kullanarak sonraki hareketlerin yönlendirilmesi ve ilişkili komşuluk yöntemlerini

kullanmışlardır. Yaptıkları deneysel çalışmalar sonucunda, ilişkili komşuluk yapısının daha iyi performansa sahip olduğunu belirtmişlerdir. Wang ve ark. (2009) GSP için iki aşamalı bir benzetilmiş tavlama algoritması önermişlerdir. İlk aşamada, bazı uygun çözümler veya kapalı turlar elde etmek için basit bir benzetimli tavlama algoritması kullanmışlardır. İkinci aşamada, basit benzetimli tavlama algoritması ile elde edilen çözümler veya kapalı turlara dayanan iyi çözümler elde etmek için etkili bir benzetimli tavlama algoritması önermişlerdir. Yaptıkları deneysel çalışmalarda önerdikleri yöntemin başarılı sonuçlar verdiğini göstermişlerdir. Liu ve Zeng (2009) GSP'yi çözmek için RMGA adlı güçlendirilmiş mutasyon operatörü olan geliştirilmiş bir genetik algoritma önermişlerdir. Önerdikleri algoritma ile yaptıkları deneysel çalışmalarda tatmin edici sonuçlar elde ettiklerini belirtmişlerdir. Karapetyan ve Gutin (2011) GSP'yi Lin-Kernighan sezgiseli ile çözmüşlerdir. Çalışmalarında Lin-Kernighan sezgiselinin faklı varyasyonlarını GSP üzerinde test etmişleridir. Geng ve ark. (2011) çalışmalarında, GSP'yi çözmek için benzetimli tavlama ve açgözlü arama tekniklerini temel alan etkili bir yerel arama algoritması önermişlerdir. Daha iyi çözümler elde etmek için, standart benzetimli tavlama algoritmasına dayanan önerilen algoritma, araştırma sırasında farklı olasılıklara sahip üç çeşit mutasyonun kombinasyonundan oluşturmuşlardır. Daha sonra önerilen algoritmanın yakınsama hızını artırmak için açgözlü arama tekniğini kullanmışlardır. Deneysel sonuçlar, önerilen algoritmanın CPU zamanı açısından ve GSP için önerilen bazı yeni algoritmalar arasında daha iyi sonuçlar ürettiğini göstermişlerdir. Wang ve Xu (2011) GSP'yi çözmek için bir melez diferansiyel evrim algoritması önermişlerdir. Önerilen algoritma, tepe tırmanma algoritması ile diferansiyel evrim algoritmasının birleştirilmiş versiyonudur. Deneysel çalışmalarda önerilen yöntemin daha verimli sonuçlar ürettiği gösterilmiştir. Kıran ve ark. (2013) GSP'nin çözümünde ayrıklaştırılmış yapay arı algoritmasını kullanarak sekiz farklı komşuluk operatörünü incelemişlerdir. Tuba ve Jovanovic (2013) GSP'nin çözümü için geliştirilmiş bir karınca koloni algoritmasını önermişlerdir. Önerdikleri yöntem, karınca koloni algoritması ile bulunan mevcut en iyi çözümün bazı elementlerinin reddedilmesine dayanmaktadır. Deneysel çalışmalarında, önerdikleri yöntemden elde ettikleri sonuçları orijinal karınca koloni algoritması ve parçacık sürü optimizasyon algoritmasından elde ettikleri sonuçlarla değerlendirmişlerdir. Pragya ve ark. (2015) geliştirdikleri boyutlu karınca koloni algoritmasını GSP'nin çözümünde kullanmışlardır. Önerdikleri algoritmada GSP'nin arama alanını birden çok alana bölmüşlerdir. Deneysel çalışmalarında önerdikleri yöntemin daha başarılı sonuçlar elde ettiğini ifade etmişlerdir. Zhou ve ark.

(2015) ayrıklaştırılmış yabani ot algoritması ile GSP'yi çözmüşlerdir. Yerel arama yöntemi olarak da 3-opt ve geliştirilmiş bir 2-opt sezgisellerini yerel arama için kullanmışlardır. Deneysel çalışmalardan elde ettikleri sonuçlar, makul zamanlar içerisinde tatmin edici sonuçların alındığını göstermişlerdir. Marinakis ve ark. (2016) değişken komşuluk arama yöntemi ile yapay arı kolonisi algoritmasını birleştirerek oluşturdukları melez yöntemi çoklu rotalama problemi üzerinde kullanmışlardır. Önerdikleri yöntemden elde ettikleri sonuçları parçacık sürü optimizasyon algoritmasının çeşitli versiyonları ile karşılaştırmışlardır. Huang ve ark. (2017) GSP'yi geliştirilmiş bir meyve sineği optimizasyon algoritması ile çözmüşlerdir. Meyve sineği algoritmasında üç temel değişiklik yapmışlardır. İlk olarak, yakınsama oranını arttırmak için meyve sineklerinin yiyecek arama davranışında görme araştırma süreci güçlendirmişlerdir. İkincisi, çeşitliliği arttırmak için orijinal meyve sineği algoritmasına bir eleme mekanizması eklemişlerdir. Üçüncü olarak da bir ters operatör ile çarpma operatörü önerilmişlerdir. Önerdikleri yöntemden elde ettikleri sonuçlar, yakınsama oranı ile sonuçların kalitesi açısından kıyasladıkları diğer yöntemlere göre daha başarılı olduğunu belirtmişlerdir.

3.2.2. Çok boyutlu iki yönlü sayı bölümleme problemi (ÇBİYSBP)

Sayı bölümleme problemi (SBP), literatürde "En zor problem" olarak tanımlanan klasik, zorlayıcı ve çekici bir optimizasyon problemidir (Mertens, 2006). Bilgisayar bilimi ve sayı teorisinde, İki Yönlü Sayı Bölümleme Problemi (İYSBP), verilen bir sonlu elemanlar kümesinin (*SEK*) AK_1'deki koordinatların toplamının AK_2'deki koordinatların toplamına eşit olduğu iki alt kümeye (AK_1 ve AK_2) bölümlenmesiyle ilgilidir (Mertens, 1998; Hacibeyoglu ve ark., 2014). Çok boyutlu iki yönlü sayı bölümleme problemi (ÇBİYSBP), İYSBP'nin özel bir formu olan kombinatoryal NP-zor optimizasyon problemidir (Kojic, 2010). ÇBİYSBP ve İYSBP arasındaki temel fark, ÇBİYSBP'de problemi daha da zorlaştıran çoklu koordinatların dikkate alınmasıdır.

ÇBİYSBP'nin matematiksel modeli Denklem 3.5'deki gibidir.

$$f = max \left| \sum_{j \in AK_1} c_{j,kr} - \sum_{j \in AK_2} c_{j,kr} \right| \tag{3.5}$$

Burada, $c_{j,kr}$ *j.* elemanın *kr* koordinatındaki değeri ifade etmektedir.

ÇBİYSBP'nin hem teorik hem de pratik açıdan önemi vardır. ÇBİYSBP pratik olarak çoklu işlemcilerin zamanlanması, genel anahtar şifreleme, veritabanı işleme ve görev zamanlaması, çok büyük ölçekli entegrasyon (ÇBÖE), devre boyutunun en aza indirilmesi ve adil takım seçilmesi gibi birçok önemli gerçek yaşam problemlerinde kullanılmaktadır (Mertens, 2006; Rodriguez ve ark., 2017). Çoklu işlemcilerin zamanlanmasında, farklı çalışma süreleri ve paralel işlemlerin sayısı ile birlikte işlerin sayısı mevcuttur. Bu sorunu çözmek için, bu işler tüm görevlerin tamamlanma süresini en aza indirgemek için bir işlemciye atanır (Bauke ve ark., 2004; Dell'Amico ve ark., 2008). Genel anahtar şifrelemede, Merkle-Hellmann sistemi İYSBP'nin benzer bir versiyonuna dayanmaktadır (Merkle ve Hellman, 1978). Adil takımları seçerken, farklı yetkinliklere sahip bir dizi kişi göz önünde bulundurulur ve eşit güçlere sahip takımların atanması amaçlanır (Hayes, 2002). Problemin bu tanımı ÇBİYSBP ile tamamen aynıdır, çünkü her yetkinlik ÇBİYSBP'nin bir koordinatı olarak düşünülebilir ve adil bir seçim için bu yetkinlikler arasındaki fark en aza indirilmelidir ki bu da ÇBİYSBP ile aynı amacı taşır.

Sayı bölümleme problemi, literatürde kesin çözüm bulan algoritmalar (Horowitz ve Sahni, 1974), açgözlü sezgisel yöntemler (Karmarkar ve Karp, 1982; Korf, 1998), benzetilmiş tavlama (Johnson ve ark., 1991), GRASP (Arguello ve ark., 1996), memetik algoritma (Berretta ve ark., 2004), tabu arama (Alidaee ve ark., 2005), tamsayı doğrusal programlama modeli (Koyutürk, 2000) ve yayılım öncelikli arama, arama ağacının melez bir yapısı ve ışın arama (Pedroso ve Kubo, 2010) gibi farklı yaklaşımlarla çözülmüştür.

Son zamanlarda, iki boyutlu iki yönlü sayı bölümleme problemini çözmek için üç meta-sezgisel algoritma, Açgözlü Algoritması (AA), melez AA ve Değişken Komşuluk Arama (DKA) geliştirmiştir (Fuksz ve ark., 2013). Yerel optimuma takılmaktan kaçınmak ve iyi sonuçlar elde etmek için DKA, AA içerisinde yerel arama operasyonları ile uygulanmıştır. Melez AA-DKA algoritması, AA ve DKA algoritmaları ile karşılaştırıldığında daha iyi sonuçlar elde etmiştir. Pop ve Matei (2013), Genetik Algoritma (GA)'nın güçlü bir yerel arama prosedürü ile birleştirilmesiyle elde edilen çok boyutlu çok yönlü sayı bölümleme problemini çözmek için etkili bir memetik algoritma sunmuşlardır. Memetik algoritma, GA'nın başlangıç popülasyonu üretimi, çaprazlama ve mutasyon operasyonlarını kullanarak melezleştirilmiştir. Başlangıç popülasyonu kısmen rastgele kısmen de ÇBİYSBP'ye bağlı olarak oluşturulmuştur. Çaprazlama işleminde, tek nokta çaprazlama seçilmiş ve yeni neslin %85'i çaprazlama işlemi ile oluşturulmuştur.

Kalan %15'i doğrudan önceki nesilden kopyalanmıştır. Ek olarak, mutasyon işleminde, %10 olasılık değeri ile yeni çocuk kromozomlar oluşturulmuştur. Önerilen memetik algoritmanın sonuçları Hacibeyoglu ve ark. (2014)'nın çalışmasında önerilen Tamsayılı Doğrusal Programlama (TDP) modelinden elde edilen sonuçlarla karşılaştırılmıştır. Kratica ve ark. (2014) ÇBİYSBP'yi DKA ve Elektromanyetizma (EM) benzeri iki meta-sezgisel yaklaşım kullanarak çözmüşlerdir. DKA rastgele bir çözümle başlatılmış ve yerel arama prosedüründe 1-swap yöntemi kullanılmıştır. EM yaklaşımında, çözümlerin geliştirilmesinde yerel arama için ölçeklendirme ve taşıma (moving) yöntemleri kullanılmıştır. DKA ve EM'den alınan sonuçlar (Pop ve Matei, 2013)'deki memetik algoritma ve (Hacibeyoglu ve ark., 2014)'teki TDP modeli sonuçları ile karşılaştırılmıştır.

ÇBİYSBP'de verilen bir *SEK* kümesi içerisinde her bir eleman kümesinin (e_i = $_{1,\dots,j}$∈*SEK*) *kr* adet *(kr>1)* elemanı vardır. $SEK = \{e_1(c_1, c_2, \dots, c_{kr}), e_2(c_1, c_2, \dots, c_{kr}), \dots, e_j(c_1, c_2, \dots, c_{kr})\}$ örnek bir problemdir ve burada *j, SEK* içindeki eleman sayısını gösterirken *kr* koordinat sayısını göstermektedir. Problemin temel amacı, verilen *SEK* kümesini AK_1 ve AK_2 şeklinde iki alt kümeye ayırmak ($AK_1 \cup AK_2 = SEK$, $AK_1 \cap AK_2 = \emptyset$), daha sonra AK_1 ve AK_2 alt kümelerinin her bir koordinatlarının toplamlarının farkını minimum yapmaktır. Her bir problem için mümkün çözümlerin toplam sayısı 2^t iken, her bir çözüm iki aynı amaç fonksiyon değerine sahiptir, böylece aynı olmayan toplam çözüm sayısı 2^t / 2'dir. $j = 8$ ve $kr = 3$ değerlerine sahip örnek bir problemin çözüm adımları aşağıda verilmiştir.

Adım 1: $j = 8$ ve $kr = 3$ değerleriyle örnek bir problem oluşturulsun.

SEK = {(1 4 2), (7 3 1), (6 5 9), (4 8 6), (2 2 5), (9 1 3), (2 4 5), (8 1 6)}

Adım 2: Verilen *SEK* kümesi AK_1 ve AK_2 şeklinde iki alt kümeye bölünür, Örneğin AK_1'in üç elemanı ve AK_2'nin beş elemanı olsun:

AK_1 = {(1 4 2), (7 3 1), (6 5 9)}

AK_2 = {(4 8 6), (2 2 5), (9 1 3), (2 4 5), (8 1 6)}

Adım 3: AK_1 ve AK_2 alt kümeleri için koordinatlarının toplamı hesaplanır.

Toplam (AK_1) = {((1 + 7 + 6) (4 + 3 +5) (2 + 1 + 9))} = {14 12 12}

Toplam (AK_2) = {((4 + 2 + 9 + 2 + 8) (8 + 2 + 1 + 4 +1) (6 + 5 + 3 + 5 + 6))} = {25 16 25}

Adım 4: Tüm koordinatların mutlak farkları ayrı ayrı hesaplanır:

Mutlak farklar: {|14 − 25| |12 − 16| |12 − 25|} = {11 4 13}

Adım 5: Amaç fonksiyonunun değeri, koordinatlar arasındaki maksimum fark ile belirlenir:

$f(SEK)$ = max(11, 4, 13) = 13

Yukarıdaki örnekte görüldüğü gibi amaç fonksiyonu beşinci adımda hesaplanmıştır. Problemin esas amacı, AK_1 ve AK_2'deki koordinatların toplamı mümkün olduğunca eşit olacak şekilde amaç fonksiyonunu en aza indirecek AK_1 ve AK_2 alt kümelerini belirlemektir.

3.2.3. İkili (0-1) sırt çantası problemi (İSÇP)

Sırt çantası problemi ayrık bir optimizasyon problemidir. Bu problem genel olarak şu şekilde ifade edilebilir. Bir hırsız, çaldığı nesnelerin pahaca en değerli ve ağırlık bakımından en hafif olmasını ister. Yani bu problem, kapasite sınırı olan bir çanta veya torbaya μ adet nesne içerisinden hangilerinin yerleşmesi gerektiği ile ilgilidir. Çantaya yerleştirilecek her bir nesnenin bir değeri (dg) ve bir de ağırlığı (ag) vardır. Bu problemdeki amaç, kapasite sınırı K olan bir çanta içerisine değerler toplamı maksimum olacak nesneleri yerleştirmektir. İkili sırt çantası probleminin matematiksel modeli Denklem 3.6 ve 3.7'deki gibidir.

$$dg_{max} = \sum_{i=1}^{\mu} dg_i h_i \tag{3.6}$$

Kısıtlar altında;

$$\sum_{i=1}^{\mu} ag_i h_i \leq K; \tag{3.7}$$

$$h_i \in \{0,1\}, i = 1, 2, \dots, \mu$$

burada, μ nesne sayısını, dg_i i. nesnesin değerini, ag_i i. nesnenin ağırlığını ve h_i i. nesnenin çanta içerisine yerleştirilmesi için seçilip seçilmediğini göstermektedir. Eğer i. nesne çanta içerisine yerleştirilirse $h_i = 1$ yerleştirilmezse $h_i = 0$ olur. Bu nedenle bu tez çalışmasında ele alınan sırt çantası problemi ikili formda (0-1) sırt çantası problemidir. Bu problemin çözümü basit gibi görülebilir. Ancak problemin hesaplama karmaşıklığı problemin boyutuyla birlikte üssel olarak artar. Bu yüzden sırt çantası problemi NP-zor kombinatoryal optimizasyon problemidir ve geleneksel metotlar ile makul süreler içerisinde çözülemeyebilir (Feng ve ark., 2017).

Sırt çantası problemi, gerçek yaşam problemlerinde yaygın bir şekilde kullanılır. Granmo ve ark. (2007) lineer olmayan kesirli sırt çantası problemini ele almışlar ve World Wide Web ile ilgili iki kaynak tahsis problemine nasıl etkili bir şekilde uygulanabileceğini göstermişlerdir. Nawrocki ve ark. (2009) gerçek-zamanlı zor problemlerden periyodik yükleme problemini "Knapsack-Lightening" olarak isimlendirilen sırt çantası probleminin yeni bir versiyonunu, açgözlü sezgiseller ve kesin çözüm bulan algoritmalar ile çözmüşlerdir. Vanderster ve ark. (2009) çalışmalarında hesaplama ağları üzerinde kaynak tahsisi için bir yardımcı model sunmuşlar ve tahsis problemini, 0-1 çok-noktalı çok boyutlu sırt çantası probleminin farklı bir biçimi olarak formüle etmişleridir.

Gerçek yaşam problemlerinin modellenmesinde örnek alınan sırt çantası problemi araştırmacıların ilgisini çekmiş ve sırt çantası probleminin çözümü için farklı yöntemler önerilmiştir. Shi (2006) klasik 0-1 sırt çantası problemini karınca koloni optimizasyon algoritması ile çözmüştür. Kong ve ark. (2008) yeni bir ikili (binary) karınca koloni optimizasyon algoritması ile çok boyutlu sırt çantası problemini çözmüşlerdir. Liu ve Liu (2009) literatüre son zamanlarda giren şema yönlendirici evrimsel algoritmanın performansını 0-1 sırt çantası problemi üzerinde test etmiş ve önerdiği yöntem, açgözlü algoritma ve basit genetik algoritma gibi yöntemleri geride bırakmıştır. Balev ve ark. (2008) dinamik programlama tabanlı bir azaltma prosedürü ile çok boyutlu 0-1 sırt çantası problemini çözmüşlerdir. Zou ve ark. (2011) 0-1 sırt çantası probleminin çözümü için yeni bir global harmoni arama algoritmasını önermişlerdir. Zhang ve ark. (2013) biyolojiden esinlenen amoeboid organizma algoritması ile 0-1 sırt çantası problemini çözmüşlerdir. Wang ve ark. (2013) yeni bir ikili meyve sineği optimizasyon algoritması ile çok boyutlu sırt çantası problemini çözmüşler ve yöntemlerinin özellikle geniş ölçekli çok boyutlu sırt çantası problemlerinde daha verimli olduğu sonucuna varmışlardır. Truong ve ark. (2013) kimyasal reaksiyonlardan esinlenen ve literatürde yeni olan kimyasal reaksiyon optimizasyon algoritmasını greedy stratejisi kullanarak 0-1 sırt çantası problemini çözmüşlerdir. Elde ettikleri sonuçları literatürdeki diğer meta-sezgisel algoritmalardan olan genetik algoritma, karınca koloni optimizasyon algoritması ve evrimsel algoritma ile kıyaslamışlardır. Önerdikleri yöntemin diğer algoritmalardan daha iyi sonuç verdiğini göstermişlerdir.

3.2.4. İkili (0-1) çok boyutlu sırt çantası problemi (İÇBSÇP)

İkili çok boyutlu sırt çantası problemi (İÇBSÇP) sırt çantası probleminin bir alt problemidir. İkili sırt çantası problemi başlığında verilen örnek tek boyutlu bir sırt çantasına örnek olarak verilebilir. Yani çantanın tek bir kapasite sınırı vardır (çantanın taşıyabileceği maksimum ağırlık). Eğer çantaya yerleştirilen nesnelerin birden çok özelliği dikkate alınarak bir kapasite sınırı kontrol edilecekse bu durumda sırt çantası çok boyutlu sırt çantası problemine dönüşmektedir. Yani hem ağırlığının hem de hacminin hesaba katıldığı μ adet nesnenin var olduğu ve bu nesnelerin bir sırt çantasına yerleştirilmesi gerektiği düşünülsün. Bu durumda sırt çantasının taşıyabileceği maksimum ağırlık ve alabileceği maksimum hacimden dolayı iki farklı kapasite sınırı (kısıtlar) ortaya çıkacaktır. Verilen örneğe göre bu sırt çantası problemi iki boyutlu sırt çantası problemi olarak adlandırılır. Her μ nesnesinin dg değere sahip olduğu düşünüldüğünde İÇBSÇP'nin amacı bu kısıtlar altında çantanın alabileceği toplam faydayı dg_{max} bulmaktır. ks kısıtının olduğu bir sırt çantası problemi Denklem 3.8'deki gibi modellenebilir.

$$dg_{max} = \sum_{i=1}^{\mu} dg_i h_i \tag{3.8}$$

Kısıtlar altında;

$$\sum_{i=1}^{\mu} ag_{ji} h_i \leq K_j \tag{3.9}$$

$$i = 1, \dots, \mu; j = 1, \dots, ks;\ h_i \in \{0,1\}$$

Burada μ nesne sayısını, ks problemin kısıt sayısını (problem boyutu), dg_i i. nesnenin değerini (faydası), ag_{ji} j. kısıta göre i. nesnenin maliyetini, K_j j. kısıtın kapasitesini ve h_i i. nesnenin seçilip seçilmediğini göstermektedir. Eğer i. nesne seçilmişse h_i 1, değilse 0 değerine sahiptir. Bu modele göre, çok boyutlu sırt çantası problemi (ÇBSÇP) ikili (0-1) çok boyutlu sırt çantası problemi (İÇBSÇP) olarak adlandırılır.

ÇBSÇP, kargo yükleme problemlerinde, sermaye harcamalarının bütçeleme problemlerinde, portföy seçiminde, bagaj yükleme gibi bir çok gerçek yaşam problemlerinin çözümünde etkin rol oynamıştır. Literatürde ÇBSÇP ile ilgili birçok çalışma mevcuttur. Chu ve Beasley (1998) ÇBSÇP'yi çok yaygın kullanılan meta-

sezgisel algoritmalardan birisi olan genetik algoritma ile çözmüşlerdir. Vasquez ve Hao (2001) İÇBSÇP'yi doğrusal programlama ve tabu arama algoritmalarının birleştirilmesiyle ortaya çıkan melez bir yöntemle çözmüşlerdir. Parra-Hernandez ve Dimopoulos (2005) İÇBSÇP'nin farklı bir türü olan çok seçimli çok boyutlu sırt çantası problemini ele almışlardır. Problemi öncelikle ÇBSÇP'ye indirgemişler. Ortaya çıkan indirgenmiş problemi doğrusal programlama yöntemiyle çözmüşler ve değişkenler için bir dizi yeni değerler hesaplamışlardır. Alonso ve ark. (2006) genetik algoritmaya flipping yerel arama modeli ekleyerek İÇBSÇP'yi çözmüşlerdir. Alves ve Almeida (2007) İÇBSÇP'yi Tchebycheff skalarizasyon fonksiyonuna dayanan yeni bir çok amaçlı genetik algoritma ile çözmüşlerdir. Akcay ve ark. (2007) İÇBSÇP'yi açgözlü algoritma ile çözerken Sbihi (2007) çoktan seçmeli ÇBSÇP'yi iki aşamalı yeni bir algoritma ile çözmüştür. Elde ettiği sonuçlara göre önerdiği yöntem çoktan seçmeli ÇBSÇP'yi en iyi şekilde çözmeyi başardığını belirtmiştir. Balev ve ark. (2007) çalışmalarında İÇBSÇP'nin merkezi işlemci birimi (MİB) zamanını doğrusal programlama-yumuşatma ve dinamik programlama kullanarak iyileştirmeye çalışmışlardır. Vimont ve ark. (2008) azaltılmış maliyet analizine dayalı implicit enumeration yöntemi ile İÇBSÇP'yi çözmüşlerdir. Kong ve ark. (2008) ÇBSÇP için ikili bir karınca koloni algoritması kullanmışlardır. Yerel aramayı güçlendirmek ve yerel optimumlardan kaçınmak için de farklı bir feromon güncelleme stratejisi kullanmışlardır. Hanafi ve Wilbaut (2008) dağıtık arama algoritması ile İÇBSÇP'yi çözmüşlerdir. Wilbaut ve ark. (2009) İÇBSÇP için değişkenlerin dinamik sabitlenmesine dayanan yinelemeli bir plan geliştirmişlerdir. Boyer ve ark. (2009) İÇBSÇP'yi çözmek için surrogate relaxation ve değiştirilmiş dinamik programlama algoritmalarını kullanmışlardır. Wilbaut ve ark. (2009) iteratif değişken tabanlı bir sezgisel algoritma ile İÇBSÇP'yi çözmüşlerdir. Boussier ve ark. (2010) İÇBSÇP'yi çözmek için çok seviyeli bir arama stratejisine dayalı kesin bir yöntem önermişlerdir. Ke ve ark. (2010) karınca kolonisi algoritmasını İÇBSÇP üzerine uygulamışlardır. Sipahioğlu ve Saraç (2010) çok amaçlı sırt çantası probleminin çözümüne yeni bir yaklaşım olan konik skalerleştirme yöntemini kullanmışlardır. Hanafi ve Wilbaut (2011) İÇBSÇP için geliştirilmiş bir yakınsak meta-sezgiel yöntem önermişlerdir. Hanafi ve Wilbaut (2011) İÇBSÇP'nin çözümünde yeni bir yakınsak algoritma kullanmışlardır. Hill ve ark. (2012) İÇBSÇP çözerken önce problem boyutunu azaltan bir yöntem kullanmışlar ardından problemi sezgisel bir yöntem ile çözmüşlerdir. Wang ve ark. (2013) ikili meyve sineği optimizasyon algoritmasını kullanarak ÇBSÇP'yi çözmüşlerdir. Yoon ve Kim (2013) İÇBSÇP'nin çözümünde Lagrangian relaxation metodunu kullanmışlardır. Ayrıca

çalışmalarında Lagrange çarpanlarını bulmak için bir memetik algoritmasından faydalanmışlardır.

3.2.5. Graf boyama problemi (GBP)

Graf boyama problemi (GBP) tipik bir kombinatoryal optimizasyon problemidir. GBP veri madenciliği, resim bölütleme, network gibi bilgisayar alanlarının yanında harita renklendirme, ders ve sınav çizelgelerinin hazırlanması, GSM ağ yapıları ve radyo frekans atamaları gibi birçok gerçek dünya problemlerinin çözümünde model alınmış bir problemdir.

GBP aşağıdaki gibi tanımlanabilir.

Grf = (Dgm, Knr) yönsüz bir graftır.

$Dgm = \{dgm_1, dgm_2, \ldots, dgm_t\}$ graftaki düğüm kümelerini ve

$Knr = \{knr_1, knr_2, \ldots, knr_t\}$ ise kenarlar kümlerini temsil etmektedir.

Bu problemdeki amaç her iki bitişik (komşu) düğümün aynı renge sahip olmayacak şekilde tüm grafı en az *rnk* adet renge boyamaktır. Bu *rnk* kromatik entropi olarak isimlendirilir ve *χ(Grf)* şeklinde gösterilir. *Grf* grafının komşu matrisi *Kmş(Grf)* $t \times t$ boyutunda simetrik bir ikili (binary) matristir. Eğer dgm_i ve dgm_j düğümleri arasında bir kenar varsa dgm_i ve dgm_j komşu düğümler olarak isimlendirilir ve $Kmş_{ij} = 1$ olur. Aksi halde $Kmş_{ij} = 0$'dır.

GBP'nin matematiksel modeli Denklem 3.10'daki gibi ifade edilebilir.

$$minf(C) = \sum_{i=1}^{t} \sum_{j=1}^{t} Kmş_{ij} b \tag{3.10}$$

$$b = \begin{cases} eğer\ \left(Kmş_{ij} = 1\right)\ ve\ \left(rnk_i = rnk_j\right) ise\ 1 \\ eğer\ \left(Kmş_{ij} = 0\right)\ veya\ \left(rnk_i \neq rnk_j\right) ise\ 0 \end{cases} \tag{3.11}$$

$$i \in \{1, \ldots, t\}, j \in \{1, \ldots, t\}, Kmş_{ij} \in \{0,1\}, b \in \{0,1\}, f(C) \geq 0$$

Burada *t* düğüm sayısı, $Kmş_{ij}$ *Kmş(Grf)* komşu matrisinin *i.*satırının *j.* sütunundaki hücre değeri, *b* "0" veya "1" değeri alan bir tamsayı, rnk_i *i.* düğümün rengi ve rnk_j *j.* düğümün rengidir. Denklem 3.10'a göre eğer *f(C) = 0* ise *Grf* grafı, komşu düğümleri aynı renkte olmayacak şekilde *rnk* renge boyanmıştır. Eğer *f(C) = 1* ise sadece bir çift komşu düğüm aynı renge boyanmış demektir.

GBP polinom zamanda çözülemeyen NP-tam (complete) problem olduğu için (Garey ve Johnson, 1979) literatürde birçok meta-sezgisel yöntemle çözülmeye çalışılmıştır. Avanthay ve ark. (2003) GBP için değişken komşuluk arama yönteminin bir adaptasyonunu önermişlerdir. Durrett ve ark. (2010) graf boyama ile ilgili kombinatoryal bir optimizasyon problemi olan kromatik entropy probleminin çözümünde genetik algoritmayı kullanmışlardır. Eiben ve ark. (1998) GBP'nin çözümünde adaptif evrimsel algoritmayı ele almışlardır. GBP, melezleştirilmiş genetik algoritmalar ile de çözülmüştür (Fleurent ve Ferland, 1996; Kong ve ark., 2003). Bir başka melezleştirilmiş algoritma olan evrimsel algoritma ile de GBP'nin çözümü incelenmiştir (Galinier ve Hao, 1999; Mahajan ve Ali, 2008). Marappan ve ark. (2016) çalışmalarında GBP için iki yöntem önermişlerdir. İlk yöntemde, düğümler kümesinde komşu kenarları en aza indirgemek için açgözlü ve sezgisel yöntemleri kullanmışlardır. İkinci yöntemde ise GBP'yi parçala ve fethet yöntemiyle çözmüşlerdir. Porumbel ve ark. (2008) GBP için yeni bir değerlendirme fonksiyonu önermişlerdir. Bu fonksiyon sadece komşu düğümlerin sayısını değil aynı zamanda graf yapısına ilişkin bilgileri de hesaba katarak işlem yapmıştır. Porumbel ve ark. (2008) genetik programlama ile GBP'nin çözümünü incelemişlerdir. Seçtikleri on test probleminin altısında optimum sonuca ulaşmışlardır. Ayrıca, çizelgeleme (Marx, 2004), gömülü sistemlerde kaydedici paylaşımı (Wu ve Li, 2007), radyo frekans ataması (Singha ve ark., 2008), harita renklendirme (Gwee ve ark., 1993) ve elektronik devrelerde gürültü azaltma (Maitra ve ark., 2010) gibi birçok gerçek yaşam problemlerinin çözümünde GBP modelinden faydalanılmıştır.

3.2.6. Tek boyutlu kesim problemi (TBKP)

Endüstriyel sanayide üreticiler için hammadde kesimindeki fire oldukça önemlidir. Birçok alanda (kâğıt, cam, plastik vb.) ürün elde edebilmek için müşterilerin talepleri doğrultusunda belirli ölçülerde gelen hammaddenin kesilmesi gereklidir. Bu kesim işlemi esnasında hem müşteri taleplerini karşılamak hem de hammadde firesini en aza indirgemek için, müşteri taleplerinin hammadde üzerine optimum şekilde nasıl yerleştirileceği oldukça önemli bir problemdir. Bundan dolayı bu tür problemlerin çözümü uzunca bir zamandır bilim insanlarının ilgisini çekmektedir.

Tek Boyutlu Kesim Problemi (TBKP) NP-zor problemdir (Tanir ve ark., 2016) ve stok kesim probleminin bir alt sınıfıdır. Bu problem genel olarak boru, kereste veya rulo şeklindeki hammaddenin sadece tek bir eksendeki mesafesi baz alınarak parçalara

ayrılmasıyla ilgilidir. Problemin gerçek hayattaki uygulamasında birçok kısıt ve parametre mevcuttur. Müşteri taleplerine karşın stokta yeterli hammaddenin olması, kullanılan hammaddelerin farklı boyutlarda olması, kesim sonrası oluşan artık hammaddenin sonraki siparişlerde yeniden kullanılması, kesim esnasında kullanılan bıçakların kalınlıkları bunlardan bazılarıdır. Ancak deneysel çalışmaların birçoğunda bu gibi durumlar ihmal edilmektedir.

TBKP matematiksel olarak Denklem 3.12'deki gibi ifade edilebilir.

$$minf(F) = \sum_{j=1}^{t} fr_j \tag{3.12}$$

$$fr_j = L_j - \sum_{i=1}^{u} tp_{ij} l_i \tag{3.13}$$

$$j = 1, 2, \dots, t; i = 1, 2, \dots, u$$

Burada *t* kesilecek olan stoktaki hammadde sayısı, *u* talep edilen parça sayısı, fr_j *j*. stok üzerinde oluşan fire, L_j *j*. stoğun uzunluğu, tp_{ij} *j*. stoktaki talep edilen *i*. parçanın sayısı, l_i talep edilen *i*. parçanın uzunluğudur.

Denklem 3.12'den de anlaşılacağı üzere kesim işleminde kullanılan tüm hammaddelerde oluşan firelerin minimum toplamı problemin amaç fonksiyonu (*f(F)*) olarak belirlenmiştir.

Literatürde TBKP'nin çözümü için birçok çalışma yapılmıştır. Scheithauer ve Terno (1995) doğrusal tamsayı minimizasyon problemi için kullanılan integer round-up property yöntemini TBKP için kullanmışlardır. Vahrenkamp (1996) random arama tekniğinin başarısını TBKP üzerinde denemiştir. Vance (1998) dal sınır ve sütun üretme yöntemlerinin melezleştirilmiş hali olan brunch-and-price yöntemini TBKP üzerinde test etmiştir. Wagner (1999) genetik algoritmayı TBKP'ye uygulamıştır. Liang ve ark. (2002) TBKP'ye evrimsel programlama ve genetik algoritmayı uygulamışlar ve başarılı sonuçlar elde etmişlerdir. Belov ve Scheithauer (2002) çok boyutlu stok uzunluğu ile TBKP için rounding sezgisel algoritmasını önermişlerdir. Umetani ve ark. (2003) TBKP için uyarlamalı model oluşturma ile yinelenen bir yerel arama algoritmasını (ILS-APG) önermişlerdir. Gradisar ve Trkman (2005) sıralı sezgisel prosedür ile dal sınır yöntemlerini birleştirerek melez bir algoritma ile TBKP'yi çözmüşlerdir. Yang ve ark. (2006) ise geliştirilmiş bir tabu arama algoritmasını TBKP'ye uygulamışlardır. Berberler

ve ark. (2011) yeni bir dinamik programlama algoritmasını TBKP çözümünde kullanmışlardır. Cherri ve ark. (2014) ise TKBP için literatürde yapılan çalışmaları kapsamlı bir şekilde ele alan bir gözden geçirme çalışması yapmışlardır. Geçmişte ve yakın zamanda kullanılan yöntemleri ve bu yöntemlerden elde edilen sonuçları toplu bir şekilde okuyucularına sunmuşlardır. KESKİN (2015) çoklu stok büyüklüklerinin TBKP için sezgisel yöntemle alternatif kesim şekillerini bulmuş, ardından tam sayılı doğrusal programlama ile iki aşamadan oluşan bir yöntem önermiştir.

3.3. Ayrık Problemlerinin Çözümünde Göçmen Kuşlar Optimizasyon (MBO) Algoritmasının İyileştirilmesi

Orijinal MBO algoritması tek sürü içerisinde bulunan birden çok birey ile arama uzayında optimum değeri arayan bir algoritmadır. İdeal çözümü arama işlemi esnasında, algoritma birtakım adımları çalıştırır. MBO algoritmasının yapısı ve adımları bir önceki bölümde bahsedilmiştir. MBO algoritmasının bazı optimizasyon problemlerinde algoritmanın yerel optimum noktalara takıldığı tespit edilmiştir. Bu, aynı zamanda birçok meta-sezgisel algoritmanın da ortak sorunudur. Literatürde, yerel çözümlere takılmaktan kaçınmak için genellikle algoritmaların temel yapısını bozmadan algoritmayı global çözüme taşıyabilecek bir takım değişikliklerin yapıldığı görülmektedir. MBO algoritmasının da global optimum çözüme ulaşmasının yolları aranmış ve farklı teknikler uygulanmıştır. Bu yöntemler bu bölümde başlıklar halinde açıklanmıştır.

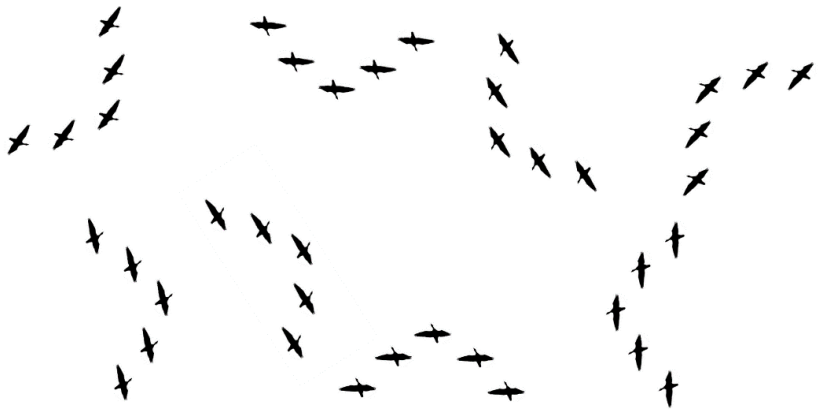
3.3.1. Çok-Sürülü MBO algoritması (ÇS-MBO)

Orijinal MBO algoritmasında bütün bireyler (çözümler) tek bir sürü içerisindedir ve bu bireyler komşu paylaşımı mekanizması ile etkileşimde bulunmaktadırlar. Bu yaklaşım, orijinal MBO algoritmasının sürü içerisindeki tüm bireyleri kaliteli çözümlere doğru yakınsamasını sağlamaktadır. Ancak yakınsanan nokta eğer bir yerel optimum çözüm ise bu durumda tüm bireyler yerel optimum çözüm etrafında toplanmış olacaklardır ve sürü içerisindeki tüm bireyler zamanla birbirine benzeyecektir. Bundan dolayı, orijinal MBO algoritması farklı bir konumda olan global optimum çözüme yaklaşamayacaktır. Bu olumsuzluğu gidermek için orijinal MBO algoritmasından farklı olarak çok-sürülü MBO (ÇS-MBO) algoritması önerilmiştir. Bu iyileştirilmiş algoritmada tek bir sürü yerine birden çok sürü kullanılmıştır. Her sürü içerisindeki

bireyler için orijinal MBO algoritmasında olduğu gibi mevcut çözümlerin geliştirilebilmesi için komşu çözümler üretilmiş ve komşu paylaşımı yardımıyla birbirleriyle etkileşim halinde olmaları sağlanmıştır. Buna karşın sürülerin birbirlerinden bağımsız bir şekilde arama işlemi gerçekleştirilmiştir. Bu yöntemle arama uzayında herhangi bir yerel çözüm bulunduğunda arama yapan tüm bireylerin bu yerel çözüm etrafında toplanması engellenmeye çalışılmıştır.

Orijinal MBO algoritmasına ek bir parametre ile arama işleminin kaç sürü ile yapılacağı belirlenmiştir.

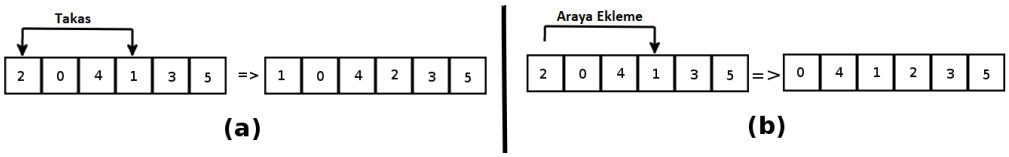
ÇS-MBO algoritmasının performansı GSP, ÇBİYSBP, GBP ve TBKP problemleri üzerinde test edilmiştir. Ayrıca, orijinal MBO algoritması çok-sürülü olarak değiştirilirken komşu çözüm üretme yöntemi de çözülen probleme göre değiştirilmiştir. Orijinal MBO algoritmasında komşu çözüm, permutasyon dizisindeki rastgele seçilen iki elemanın karşılıklı olarak değiştirilmesi (takas) suretiyle gerçekleştirilmektedir. Çok-sürülü MBO algoritmasında ise komşu çözüm üretme yöntemi olarak hem takas yöntemi hem de araya ekleme (insertion) yöntemleri kullanılmıştır. Çok-sürülü MBO algoritmasının başlangıç çözümlerinin yerleşimi Şekil 3.5'te gösterilmiştir.



Şekil 3.5. ÇS-MBO algoritması

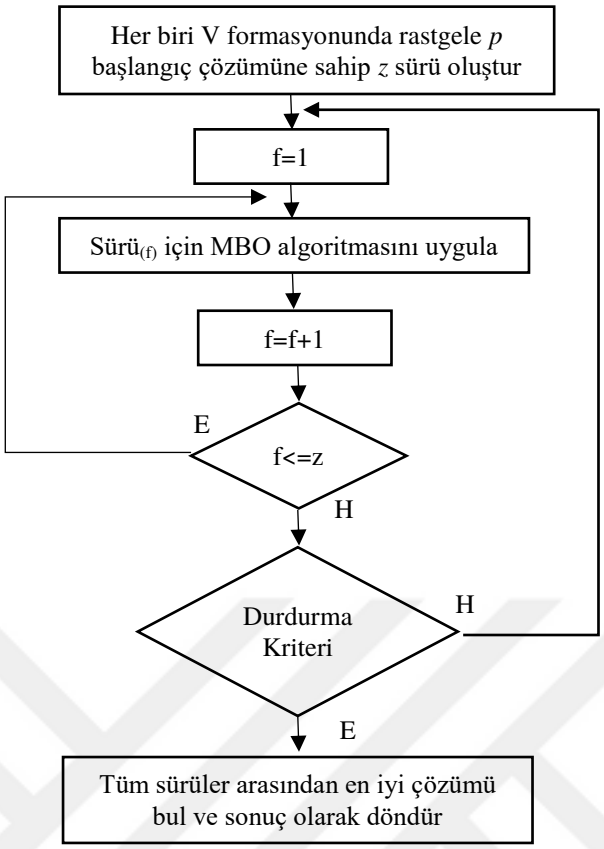
Şekil 3.5'ten de anlaşılacağı gibi ÇS-MBO algoritmasında bireyler birden çok sürü içerisine yerleştirilir ve sürüler birbirinden bağımsız olarak hareket ettirilir.

Ayrıca ÇS-MBO algoritmasında kullanılan komşu çözüm üretme yöntemleri Şekil 3.6'da gösterilmiştir.



Şekil 3.6. Komşu çözüm üretme yöntemleri

Şekil 3.6'da iki farklı komşu çözüm üretme yöntemi gösterilmektedir. Orijinal MBO algoritmasında Şekil 3.6 (a) gösterildiği gibi takas yöntemini kullanarak komşu çözüm üretilmektedir. Takas yapılacak konumlar rastgele belirlenir ve konumlar içerisindeki değerler karşılıklı olarak yer değiştirilir. Araya ekleme yönteminde ise yine rastgele iki konum belirlenir ve konumlardan birisi içerisindeki değer rastgele belirlenen diğer konumun arasına eklenir. Eğer eklenecek değerin konumu araya eklenecek konumun solunda ise araya eklenecek konumdan eklenecek değerin boşaltıldığı konuma doğru her bir eleman sola doğru kaydırılır. Eğer eklenecek değerin konumu araya eklenecek konumun sağında ise bu defa da kaydırma işlemi sağa doğru gerçekleştirilir. Şekil 3.6 (b)'de ekleme yerine kadar her bir elemanın sola doğru kaydırıldığı gösterilmektedir. ÇS-MBO algoritmasında Şekil 3.6'da gösterilen her iki komşuluk yöntemi ele alınan probleme göre tercih edilmiştir. Çözülen bazı problemlerde ise her ikisi birden kullanılmıştır ÇS-MBO algoritmasına ait akış diyagramı Şekil 3.7'de verilmiştir.



Şekil 3.7. ÇS-MBO algoritmasının akış diyagramı

3.3.2. PSO Tabanlı Çok-Sürülü MBO algoritması (PSO-ÇS-MBO)

Birçok meta-sezgisel algoritma, yerel optimuma kolayca takılabilmektedir. Bunu önlemek için, kullanılan algoritmalar üzerinde temel yapıları değiştirmeyecek şekilde değişikliğe gidilir. Bu değişikliklerin genellikle başlangıç populasyonu ve komşuluk yapısı üzerinde olduğu görülür. Bunun yanında bazı meta-sezgisel algoritmaların iyi yanlarının alınarak geliştirilmek istenen algoritmaya uyarlandığı da görülmektedir. Bu şekilde geliştirilmiş algoritmalara melez (hibrit) algoritmalar da denmektedir. Literatürde bu tür melez algoritma örneklerine rastlamamız mümkündür. Debels ve ark. (2006) kısıtlı kaynak proje çizelgeleme probleminin çözümü için genel popülasyon tabanlı evrimsel arama yöntemi ile dağıtık arama algoritmasından seçilen yapıların birleşiminden oluşan melez bir yöntem önermişlerdir. Salcedo-Sanz ve ark. (2006) genetik algoritma ile benzetimli tavlama algoritmalarından oluşan melez bir algoritmayı hetorojen bilgisayar sistemlerinde görev atama problemi için önermişlerdir. Poorzahedy ve Rouhani (2007), genetik algoritma, benzetimli tavlama ve tabu arama algoritmalarını kullanarak yedi

farklı melez yapı geliştirmişler ve Sioux Falls ağında test etmişlerdir. Rao ve Shyju (2008) çok katlı kompozit silindirik kenarın optimal dizaynı için benzetimli tavlama ile tabu arama algoritmalarını birlikte kullanmışlardır. Duan ve ark. (2010) insansız savaş uçağının üç boyutlu rota planlaması için karınca koloni algoritması ile diferansiyel evrim algoritmalarını birlikte kullanmışlardır. Jolai ve ark. (2012), benzetimli tavlama algoritması ile uyarlanmış emperyalist rekabetçi algoritmayı birleştirerek geliştirdiği melez algoritmayı beklemesiz seri üretim planlama probleminin çözümünde kullanmışlardır. Banos ve ark. (2013) zaman kısıtlı araç rotalama probleminin çözümünde evrimsel hesaplama ve benzetimli tavlama algoritmalarını birleştirerek pareto tabanlı melez bir algoritma kullanmışlardır. Azadeh ve ark. (2013) ekip çizelgeleme problemini çözmek için yerel bir arama sezgiseliyle senkronize bir parçacık sürü optimizasyon algoritmasını kullanmışlardır. Wang ve ark. (2015) zaman serileri tahmini için uyarlamalı diferansiyel evrim algoritması ile geri yayılımlı sinir ağını birlikte kullanmışlardır. Zeng ve ark. (2016) diferansiyel gelişim algoritması ve benzetimli tavlama algoritmalarından oluşan melez bir algoritma ile ticaret kredisiyle ortak ikmal ve dağıtım problemini çözmüşlerdir. Gulcu ve ark. (2018) gezgin satıcı probleminin çözümünde karınca koloni optimizasyon algoritması ile 3-opt algoritmalarından oluşan melez bir algoritma önermişlerdir.

Bu tezde ele alınan orijinal MBO algoritması üzerine yapılan bir diğer değişiklik PSO algoritması ile birleştirilmiş melez bir MBO algoritma örneğidir. Bir önceki başlıkta anlatılan ÇS-MBO algoritmasında çoklu sürüler birbirlerinden bağımsız olarak hareket ettirilmekteydiler. Her ne kadar tek sürü kullanan orijinal MBO algoritmasına göre çok-sürülü MBO algoritması arama uzayını daha detaylı tarayabilse de sürüler birbirinden bağımsız hareket ettiğinden dolayı bireyler veya sürüler kimi zaman global optimuma yeteri kadar yaklaşamamakta veya farklı yerel optimum noktalara yakalanabilmektedir. Buradan hareketle, hem arama uzayını daha iyi tarayabilecek hem de o an için bulunan en iyi sonuca doğru sürüyü yavaş yavaş yaklaştırabilecek bir yöntemin geliştirilmesi tez kapsamına alınmıştır. ÇS-MBO algoritmasına yeni bir modül entegre edilmiştir. Bu modül birden çok sürünün birbiriyle etkileşime geçmesini sağlayacak bir yapının eklenmesi ile gerçekleştirilmiştir. Eklenen bu modül sürü temelli meta-sezgisel algoritmalardan olan parçacık sürü optimizasyon (PSO) algoritmasıdır. PSO algoritması, ÇS-MBO algoritmasında sürülerin haberleşmesi için kullanılmıştır. Oluşturulan bu melez yöntemin daha iyi anlaşılabilmesi için PSO algoritmasının nasıl çalıştığının bilinmesi gereklidir.

3.3.2.1. Parçacık Sürü Optimizasyon (PSO) Algoritması:

PSO algoritması, Eberhart ve Kennedy (1995) tarafından balık ve kuş sürülerinin yiyecek arayışı esnasında sergiledikleri davranışlardan esinlenerek literatüre kazandırılmış sürü temelli meta-sezgisel bir algoritmadır. Balık veya kuş sürülerindeki her bir birey sürüdeki diğer bireylerden etkilenerek yiyecek kaynağına doğru yön ve hız bilgisini güncelleyerek hareket ederler. PSO algoritmasında da bu davranıştan esinlenilmiştir. PSO algoritması popülasyon tabanlı bir meta-sezgisel algoritmadır. Popülasyondaki her bir birey bir çözümü temsil etmektedir ve bireylerin konum değiştirmesi sürünün hız ve önceki konum bilgisine bağlı olarak gerçekleştirilir. PSO algoritmasında her bir birey bir hız ve konum vektörüne sahiptir. Başlangıçta bu vektörlerden konum vektörü rastgele değerlerden oluşturulurken hız vektörü genellikle sıfır değeri ile doldurulur. Bunun yanında bir de sürünün konum ve hız vektörü mevcuttur. Sürüye ait bu konum ve hız vektörlerine global en iyi (*gBest*) ismi verilmektedir. *gBest* her iterasyonda sürüde bulunan tüm bireyler arasından en iyi çözüme sahip bireyin çözümü ile kıyaslanır, eğer *gBest*'ten daha iyi bir çözüm bulunmuşsa *gBest* güncellenir. Ayrıca her bir bireyin bir hafızası vardır ve mevcut iterasyona kadar elde ettikleri en iyi çözümü bu hafızada saklarlar. Bu hafızaya da *pBest* ismi verilmektedir. Sürü içerisindeki bireylerin yeni konumları kendilerine ait hız vektörü, *pBest* ve *gBest* vektörlerine göre güncellenir. PSO algoritmasındaki her bir bireyin konum ve hız güncelleme denklemleri Denklem 3.14 ve 3.15'te verilmiştir.

$$v^{t+1} = wv^t + c_1 r_1 (pBest - pos^t) + c_2 r_2 (gBest - pos^t) \tag{3.14}$$

$$pos^{t+1} = pos^t + v^{t+1} \tag{3.15}$$

Burada, c_1 ve c_2 öğrenme faktörüdür ve genellikle sabit 2 değerini alırlar. r_1 ve r_2 [0, 1] aralığında üretilmiş rastgele değerlerdir. *w* intertia faktörü olarak isimlendir. *w* parametresi, her iterasyonda hız değerini bir önceki iterasyona göre düşürmeye yarar. *v* bireye ait hız vektörüdür. *pos* ise bireye ait konum vektörüdür. *t* ise iterasyon sayısını ifade eder.

3.3.2.2. PSO'nun ÇS-MBO'ya adapte edilmesi

PSO algoritması ÇS-MBO algoritmasına uygulanırken her sürü, PSO algoritmasındaki bir bireye benzetilir, her sürüdeki her bir kuş kendi konum ve hız vektörüne sahiptir ve sürüler arası etkileşim sadece sürü liderleri arasında gerçekleştirilir. Sürüler arası etkileşim, lider bireyler üzerinden gerçekleştirildiği için lider değişimi de orijinal ve ÇS-MBO algoritmasından farklıdır. PSO tabanlı çok-sürülü MBO algoritmasında lider değişimi ilk olarak orijinal MBO'da olduğu gibi yapılır. Yani lideri takip eden birey, liderin yerini alırken lider, sürünün sağ veya sol kanadının en arkasına gönderilir. Farklı olan yönü ise liderin arkasında olup lider pozisyonuna güncellenen yeni liderin pozisyonu diğer sürülerdeki liderlerin hız ve konum bilgilerine bağlı olarak güncellenmesidir. Bu işlem, PSO algoritmasında bireylerin hıza göre pozisyonlarının değişimine benzemektedir. Yeni liderin pozisyonu Denklem 3.19'a göre hesaplanır. Ancak, PSO algoritması sürekli optimizasyon problemleri için tasarlanmıştır. Bu nedenle PSO algoritmasındaki aşamalar ayrık problemlere direkt olarak uygulanamaz. Öncelikle PSO'nun ayrık problemleri çözecek şekilde değiştirilmesi gerekmektedir. Bu tezde PSO'nun ayrık problemlere uygun hale getirilme işlemleri GSP problemi üzerinden örneklendirilmiştir. PSO algoritmasına göre *i.* parçacık *t*. iterasyonda ise bu parçacık X_i^t şeklinde gösterilir ve d-boyutlu GSP'nin çözümü için parçacığın çözüm kümesi $X_i^t = [x_{i1}^t, x_{i2}^t, x_{i3}^t, \dots, x_{id}^t]$ şeklinde gösterilir. Bu çözüm setindeki her bir pozisyon bir gerçel sayıdır. Bu nedenle parçacığın çözüm seti bir permütasyon olarak gösterilemez. Buna karşın GSP ayrık bir problemdir ve çözülürken genellikle permütasyon kodlama kullanılmaktadır. PSO algoritması, GSP'yi çözmek için kullanılacaksa çözüm seti permütasyon kodlama şeklinde olmalıdır. Bu değişimi sağlayabilmek için en küçük pozisyon değeri (EPD) kuralı uygulanmıştır. EPD kuralını daha iyi anlamak için;

X_{if}^t çözümünün posizyon $\left(x_{ijf}^t\right)$, hız $\left(v_{ijf}^t\right)$ ve permütasyon $\left(\pi_{ijf}^t\right)$ bilgileri Çizelge 3.2'de verilmiştir. Burada *t, f, i* ve *j* sırasıyla lider değişim sayısı, sürü numarası, parçacık ve boyutu ifade etmektedir.

Çizelge 3.2. X_{if}^t parçacığına ait çözüm permütasyonunun elde edilmesi

Boyut, j	1	2	3	4	**5**	6
x_{ijf}^t	2,63	1,09	2,20	2,53	**0,89**	2,93
v_{ijf}^t	-0,13	0,19	2,23	-3,5	-3,81	1,30
π_{ijf}^t	**5**	2	3	4	1	6

Çizelge 3.2'ye göre X_{if}^{t} parçacığının pozisyonları unifrom dağılımlı ve önceden belirlenmiş minimum ve maksimum değerler arasındaki rastgele sayılardan oluşturulur. EPD kuralına göre, daha sonra bu değerler küçükten büyüğe doğru sıralanırlar. Sırlama esnasında değerler yer değiştirirken değerin tutulduğu dizinin indis numarası da değerle birlikte yer değiştirir. Böylece rastgele oluşturulmuş bir permütasyon sıralaması elde edilmiş olur. Çizelge 3.2'de boyutu altı olan (şehir sayısı=6) bir problem için başlangıç konum bilgisi $\left(x_{ijf}^{t}\right)$ [0,4] arasında rastgele uniform değerlerle doldurulmuştur. EPD kuralına göre, en küçük konum bilgisi $x_{i5f}^{t} = 0{,}89$'dur. Bu değer x_{ijf}^{t} konum dizisinin 5. elemanıdır. Bu nedenle $\left(\pi_{ijf}^{t}\right)$ permütasyonuna ilk şehir olarak $\left(\pi_{i1f}^{t} = 5\right)$ atanacaktır. İkinci en küçük değer $x_{i2f}^{t} = 1{,}09$'dur. Bu değer de $\left(x_{ijf}^{t}\right)$ konum dizisinin 2. elemanıdır. Bu nedenle $\left(\pi_{ijf}^{t}\right)$ permütasyonuna ikinci şehir olarak $\left(\pi_{i2f}^{t} = 2\right)$ atanacaktır. x_{ijf}^{t} konum dizisindeki diğer elemanlar da aynı şekilde değerlendirildiğinde X_{if}^{t} parçacığının çözüm permütasyonu $\pi_{if}^{t} = \{5,2,3,4,1,6\}$ şeklinde oluşacaktır.

3.3.2.3. PSO-ÇS-MBO Algoritmasının Başlangıç Popülasyonu

PSO tabanlı ÇS-MBO algoritmasında (PSO-ÇS-MBO), PSO algoritması sadece sürü liderleri arasında uygulanmıştır. Diğer bir deyişle, PSO algoritması, sürüler arasındaki etkileşimi sağlamak için kullanılmıştır. Bu sayede tüm sürülerin global optimum bölgeye doğru toplanması hedeflenmiştir. PSO, sadece sürü liderlerine uygulanmış olmasına rağmen, sürü liderleri dışındaki diğer bireyler de bir PSO parçacığı gibi konum, hız ve permütasyon bilgilerine sahiptir. Çünkü lider değişiminden dolayı her bireyin lider olma potansiyeli vardır. PSO-ÇS-MBO algoritmasında popülasyon, her sürüde aynıdır ve *p* kadardır. Her sürüdeki bireyin başlangıç çözüm permütasyonunu belirleyen konum ve hız değerleri sırasıyla Denklem 3.16 ve 3.17'ye göre belirlenmiştir.

$$x_{ijf}^{0} = x_{min} + (x_{max} - x_{min}) * r_1 \qquad (3.16)$$

$$i = 1, 2, ..., p;\ j = 1, 2, ..., d;\ f = 1, 2, ..., z$$

Burada *p* bir sürüdeki birey sayısıdır. *d* problemin boyutu ve *z* ise sürü sayısını ifade etmektedir.

x_{ijf}^0, *f.* sürünün ilk liderindeki *i.* bireyin *j.* boyutuna ait konum bilgisini göstermektedir. $x_{min} = 0$, $x_{max} = 4$ ve r_1 ise [0, 1] arasında üniform rastgele bir değerdir.

$$v_{ijf}^0 = v_{min} + (v_{max} - v_{min}) * r_2 \tag{3.17}$$

Burada v_{ijf}^0 *f.* sürünün ilk liderindeki *i.* bireyin *j.* boyutuna ait hız bilgisini göstermektedir. $v_{min} = -4$, $v_{max} = 4$ ve r_2 ise [0, 1] arasında üniform rastgele bir değerdir.

Denklem 3.16 ve 3.17'den elde edilen başlangıç konum ve hız bilgilerine sahip bireylerin çözüm permütasyonları EPD kuralına göre belirlenir. PSO-ÇS-MBO algoritmasında *pBest*'in sayısı PSO algoritmasından farklıdır ve bu sayı sürü sayısına eşittir. Çünkü PSO algoritması sadece sürü liderlerine uygulanmıştır. Bu nedenle her sürü bir *pBest*'e sahiptir ve *pBest*'in başlangıç değeri, ilgili sürünün sürü liderine ait konum bilgisidir. Yani $p_{Best}^f = X_{1f}^0$'dır. Global best (*gBest*) ise seçilen en iyi *pBets* olarak atanır.

Orijinal MBO algoritmasında sürü liderlerinin etkisi V dizilimindeki sürünün her iki tarafına da komşu çözüm paylaşımı yapabildiğinden dolayı çok önemlidir. Ayrıca Denklem 3.18 ve 3.19'a göre, bu liderlerin bir hız vektörüne sahip olması gerekmektedir. Diğer bir deyişle $\delta_f^0 = v_{1f}^0, f = 1, 2, ..., z$. Burada δ_f^0 *f.* sürünün başlangıç hız vektörüdür. Daha sonra bu hız vektörü her lider değişiminde Denklem 3.18'e göre güncellenir.

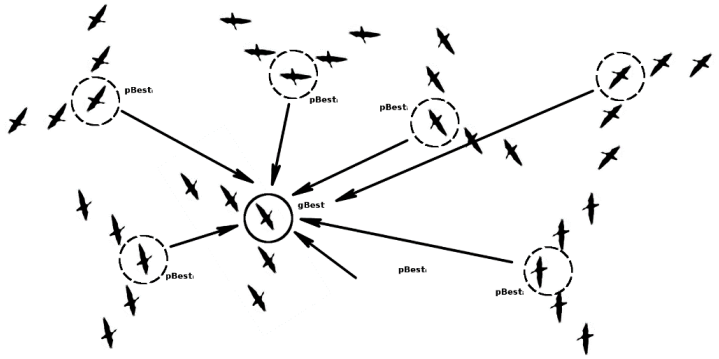
$$\delta_f^{t+1} = w\delta_f^t + c_1 r_1 (p_{Best}^f - X_{0f}^t) + c_2 r_2 (g_{Best}^f - X_{0f}^t) \tag{3.18}$$

$$X_{0f}^{t+1} = X_{0f}^t + \delta_f^{t+1} \tag{3.19}$$

Burada, δ_f^{t+1}, *(t+1).* lider değişimindeki *f.* sürünün hız vektörünü, *w* intertia faktörü, δ_f^t, *t.* lider değişimindeki *f.* sürünün hız vektörünü temsil etmektedir. c_1 ve c_2 öğrenme katsayılarını temsil eder. Bu değişkenlere sabit bir değer atanır ve genellikle bu değer 2'ye eşittir, r_1 ve r_2 [0, 1] aralığında üniform rastgele bir sayıdır. p_{Best}^f, *f.* sürü tarafından o ana kadar bulunmuş en iyi çözümün konum vektörünü temsil eder. g_{Best}^f ise o ana kadar lider pozisyonunda olan tüm bireyler tarafından bulunan en iyi çözümün konum vektörünü temsil etmektedir.

f. sürünün hız vektörü, yeni liderin konum vektörünün $\left(X_{0f}^{t+1}\right)$ hesaplanmasında kullanılır ve Denklem 3.18'e göre güncellenir. Yeni liderin konumu, bir önceki liderin konum ve hız vektöründen faydalanarak Denklem 3.19'a göre güncellenir. Liderin yeni konum bilgisine bağlı olarak yeni bir çözüm permütasyonu elde edilir.

PSO-ÇS-MBO algoritması Şekil 3.8'deki gibi gösterilebilir. Şekil 3.8'den de görülebileceği gibi, sürüdeki her lider PSO'da bir birey olarak düşünülebilir. Sürü liderinin yeni konumu, sürülere ait hız, konum ve mevcut en iyi çözüm bilgileri kullanılarak belirlenir. Bunun amacı, sürüleri PSO algoritmasında olduğu gibi global çözüme doğru yaklaştırmaktır. Böylece, orijinal MBO algoritmasının yapısı korunmuş ve her sürü, lider değişim gerçekleştirilene kadar kendi konumunda yerel aramaya devam ettirilmiştir.



Şekil 3.8. PSO-ÇS-MBO algoritmasında sürülerin haberleşmesi

3.3.2.4. PSO-ÇS-MBO Algoritmasının Komşuluk Yöntemi

PSO-ÇS-MBO algoritmasında komşuluk üretimi, doğrudan bireylerin permütasyonlarına uygulanır. Çizelge 3.3'te komşuluk yöntemi olarak araya ekleme yöntemi seçilmiştir. Ancak, bireylerin permütasyonları EPD kuralına göre oluşturulduğundan dolayı bir onarım sürecine ihtiyaç duyulmuştur. Bu onarım süreci Çizelge 3.3 ve 3.4'te gösterilmiştir.

Çizelge 3.3. Onarım sürecinden önce gerçekleştirilen komşu çözüm üretimi

Boyut, j	1	2	3	4	5	6
x_{ijf}^{t}	2,63	1,09	2,20	2,53	0,89	2,93
π_{ijf}^{t}	**5**	2	3	**4**	1	6
x_{ijf}^{t}	2,63	1,09	2,20	2,53	0,89	2,93
π_{ijf}^{t}	2	3	4	**5**	1	6

Çizelge 3.3'te çözüm permütasyonun ilk elemanı olan $\pi_{i1f}^{t} = 5$ permutasyonun dördüncü sırasına $\pi_{ijf}^{t} = 4$ eklenmiştir. Eğer komşu çözüm üretimi doğrudan permütasyon sırasına uygulanırsa EPD kuralı bozulur. EPD kuralının daima

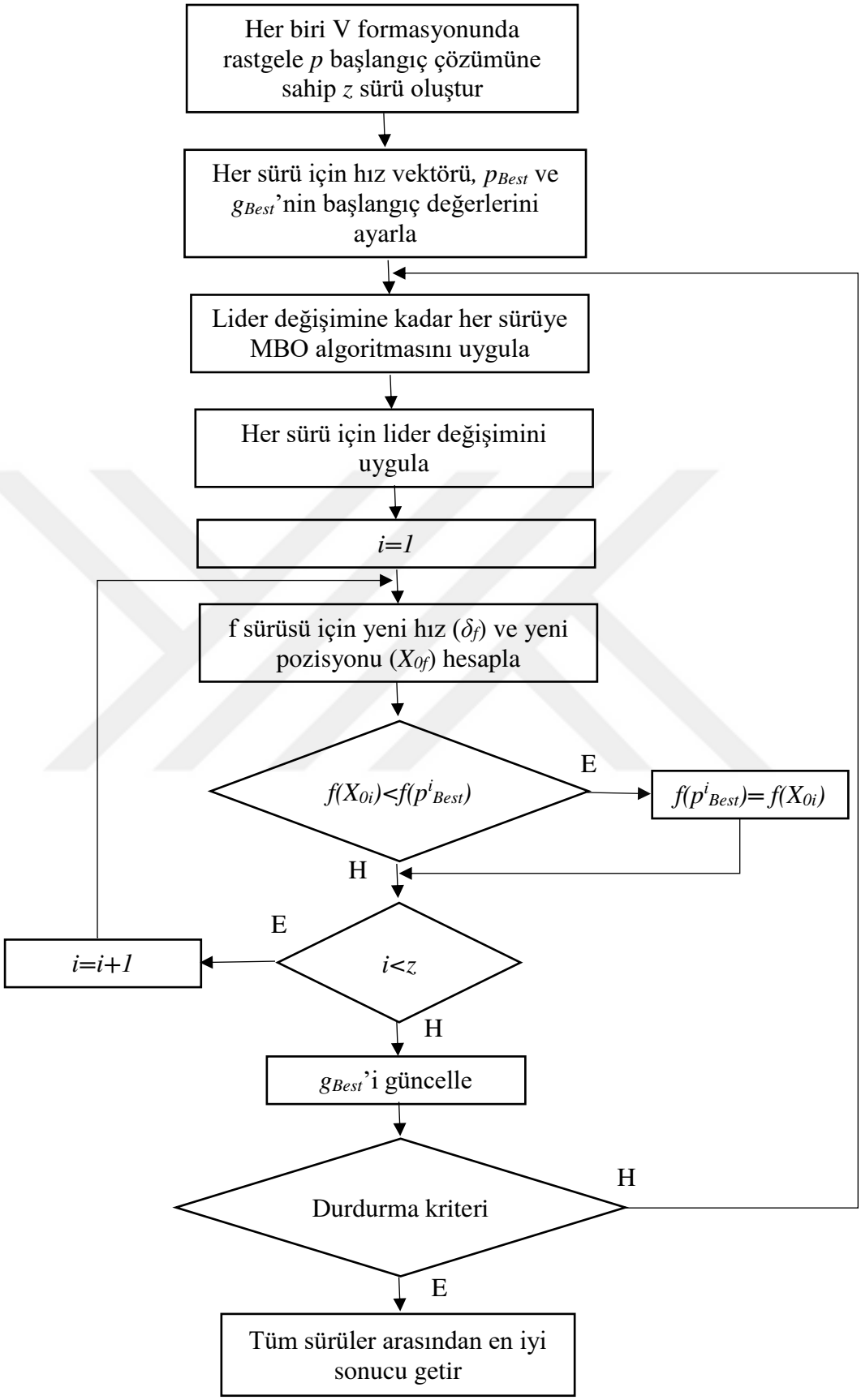
sağlanabilmesi için oluşturulan komşu çözümün onarımdan geçmesi gerekmektedir. Buna göre Çizelge 3.3'teki ilk konum bilgisi (π_{i1f}^{t}'ten dolayı) $x_{i1f}^{t} = 2{,}63, j = 4$ konumu için araya eklenir. Böylelikle EPD kuralı korunmuş olur. Çözümün onarım sürecinden sonraki hali Çizelge 3.4'te verilmiştir.

Çizelge 3.4. Onarım sürecinden sonraki komşu çözüm

Boyut, j	1	2	3	4	5	6
x_{ijf}^{t}	2,63	1,09	2,20	2,53	0,89	2,93
π_{ijf}^{t}	5	2	3	4	1	6
x_{ijf}^{t}	1,09	2,20	2,53	**2,63**	0,89	2,93
π_{ijf}^{t}	2	3	4	5	1	6

Çizelge 3.4'e göre üretilen yeni komşu çözüme ait konum bilgileri de bireyin permütasyonuna uygun hale getirilmiş olur.

PSO-ÇS-MBO algoritmasına ait akış diyagramı Şekil 3.9'da verilmiştir.



Şekil 3.9. PSO-ÇS-MBO algoritmasının akış diyagramı

3.3.3. İkili (Binary) MBO algoritması (BMBO)

Orijinal MBO algoritmasında sürüdeki her birey bir çözüm permütasyonuna sahiptir. Ancak bazı optimizasyon problemlerini çözebilmek için çözüm permütasyonunun ikili (0/1) olarak ifade edilmesi gerekmektedir. Bir önceki bölümde bahsedilen tek boyutlu ve çok boyutlu sırt çantası problemleri bu türden problemlerdendir.

Orijinal MBO algoritması ile ikili problemi çözebilmek için çözüm permütasyonları "0" veya "1" ile doldurulmalıdır. Bu durumda yeni çözüm üretimi ve mevcut çözüme ait komşu çözüm üretim aşamasında orijinal MBO algoritmasından farklı bir yöntem uygulanmalıdır. Başlangıç popülasyonu ve komşu çözüm üretim aşamaları aşağıda başlıklar halinde verilmiştir.

3.3.3.1. BMBO başlangıç popülasyonu

BMBO'da her birey bir çözümü temsil etmektedir. Başlangıç popülasyonu orijinal MBO algoritmasında olduğu gibi rastgele oluşturulur. Bireyler içerisinde bulunan tek boyutlu vektörün büyüklüğü problem boyutuna bağlı olarak dinamik bir şekilde oluşturulur. Geliştirilen BMBO algoritmasında bu vektör başlangıçta "0" değeri ile doldurulur. Ardından problemin kısıtları dikkate alınarak vektörün herhangi bir hücresi rastgele seçilir. Seçilen bu hücre içerisindeki "0" değeri "1" olarak güncellenir. Vektördeki bu güncelleme işlemine problem veya problemin kısıtlarına göre devam edilir. Oluşturulan bireyler orijinal MBO algoritmasında olduğu gibi "V" formasyonunda yerleştirilir. Popülasyonun büyüklüğü probleme göre farklılık gösterebilir. İdeal popülasyon büyüklüğü ele alınan probleme göre test edilerek belirlenir. Örnek bir çözüm permütasyonu Şekil 3.10'da verilmiştir.

0	0	1	1	0	1	1	0	1

Şekil 3.10. İkili MBO algoritmasının örnek bir çözüm permütasyonu

3.3.3.2. BMBO için komşu çözüm üretimi ve paylaşımı

BMBO'da komşu çözüm üretimi orijinal MBO algoritmasından farklı uygulanır. İkili kodlanmış bir çözüm permütasyonunda komşu çözüm üretme, genellikle rastgele seçilen bir hücrenin değerinin terslenmesiyle elde edilmektedir. Şekil 3.10'daki örnek permütasyon üzerinden rastgele bir hücre seçilir. Hücre içerisindeki değer eğer "1" ise "0", "0" ise "1" yapılarak yeni bir ikili permütasyon elde edilir. Bu değişim problemin kısıtlarını bozmayacak şekilde yapılır. Eğer hücre içerisindeki değer değişikliği kısıtları bozuyorsa bu durumda değeri "1" olan hücreler arasından rastgele bir hücre seçilir ve yeni değer "0" olarak güncellenir. Üretilecek komşu çözümlerin ve paylaştırılacak çözümlerin sayısı orijinal MBO algoritmasında olduğu gibi Denklem 3.1, 3.2 ve 3.3'e göre belirlenir.

Komşu çözüm paylaşımı orijinal MBO algoritmasındaki gibi gerçekleştirilir. Aynı şekilde BMBO'nun durdurma kriteri de orijinal MBO algoritması ile aynıdır. Ancak durdurma kriteri zaman sınırlaması veya önceden belirlenen bir iterasyon değeri ile de belirlenebilir.

3.3.4. Çok Sürülü İkili MBO algoritması (ÇS-BMBO)

Çok sürülü ikili MBO (ÇS-BMBO) algoritması geliştirilmiş ÇS-MBO algoritmasından esinlenilmiştir. İkili problemlerin çözümü için üretilen BMBO algoritmasının performansını artırmak amacıyla BMBO'nun geliştirilmiş versiyonudur. ÇS-MBO algoritmasında olduğu gibi, ÇS-BMBO algoritmasında tek bir sürü yerine birden çok sürü kullanılmıştır. Arama uzayının daha iyi taranabilmesi amacıyla geliştirilen ÇS-BMBO algoritması BMBO algoritmasından türetilmiştir.

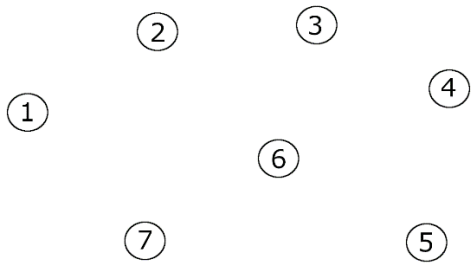
Başlangıç popülasyonun oluşumunda BMBO algoritmasında olduğu gibi bireyler ikili verileri tutacak şekilde bir çözüm permütasyonuna sahiptirler. Sürüyü oluşturan her birey V formasyonunda yerleştirilir. BMBO'dan farklı olarak bireylerin tamamı tek bir sürü içerisine değil birden çok sürü içerisine yerleştirilirler. Sürüler arasındaki bireyler komşu paylaşım mekanizması ile etkileşim halinde olurken, sürüler birbirinden bağımsız arama işlemi yaparlar. ÇS-BMBO algoritmasında başlangıç popülasyonunun oluşumu esnasında birden çok sürünün oluşturulması haricinde BMBO algoritmasındaki başlangıç popülasyonu, komşu üretimi ve paylaşımı ile lider değişimi adımları BMBO ile aynıdır.

4. ARAŞTIRMA SONUÇLARI VE TARTIŞMA

Bu tez çalışmasında, orijinal MBO algoritması ve onun geliştirilmiş versiyonu olan ÇS-MBO, PSO-ÇS-MBO algoritmaları ile yeni üretilmiş olan BMBO ve ÇS-BMBO algoritmaları; GSP, ÇBİYSBP, İSÇP, ÇBİSÇP, GBP ve TBKP problemleri üzerine uygulanmıştır. Tüm algoritmalar Qt Creator ortamında C++ programlama dili kullanılarak yazılmış ve testlerin tamamı Linux Ubuntu 14.04 işletim sistemi üzerinde gerçekleştirilmiştir. Bahsi geçen problemlerin algoritmalara nasıl uygulandığı ve elde edilen sonuçlar, bu bölümde sunulmuştur.

4.1. GSP'nin MBO, ÇS-MBO ve PSO-ÇS-MBO ile Çözümü

Bu bölümde daha önce tanımı verilen GSP'nin MBO, ÇS-MBO ve PSO-ÇS-MBO algoritmaları ile çözümünden bahsedilmiştir. Konunun daha iyi anlaşılabilmesi için somut bir örnek verilmiştir. Verilen örnek, yedi şehirden oluşmakta ve bu şehirlerin yerleşimi Şekil 4.1'de gösterilmiştir. Bu örnek, simetrik bir GSP olarak düşünülmüş ve şehirler arasındaki mesafeler Çizelge 4.1'de verilmiştir. Simetrik GSP'de iki şehir arasındaki gidiş ve geliş mesafesi aynıdır.



Şekil 4.1. Örnek GSP için yedi şehrin yerleşimi

Şekil 4.1'de görülen yedi şehrin tamamı arasında birbirlerine bağlı bir yolun olduğu kabul edilir. Buna göre şehirlerin arasındaki mesafeler Çizelge 4.1'deki gibidir.

Çizelge 4.1. Örnek GSP'ye ait şehirler arası mesafeler

Şehir No	Şehir No	Mesafe	Şehir No	Şehir No	Mesafe
1	2	10	3	4	7
1	3	15	3	5	12
1	4	25	3	6	4
1	5	27	3	7	20
1	6	18	4	5	9
1	7	11	4	6	13
2	3	8	4	7	16
2	4	12	5	6	6
2	5	16	5	7	14
2	6	4	6	7	8
2	7	9			

Verilen örnek problem algoritmalara uygulanırken çözümü temsil eden bireylerin sahip oldukları tek boyutlu çözüm vektörleri oluşturulur. Oluşturulan vektörlerin uzunluğu problemde ele alınan şehir sayısı kadardır. Bu çözüm vektörlerinin her hücresine rastgele bir şehir atanır. Bu atama esnasında aynı şehrin iki veya daha fazla atanmaması ve ayrıca bütün şehirlerin atama işlemine dahil olmasına dikkat edilir. Bütün şehirler sadece bir defa olmak üzere vektöre yerleştirilerek bir çözüm permütasyonu elde edilir. Örnek bir çözüm permütasyonu Şekil 4.2'de verilmiştir.

Şekil 4.2. Rastgele oluşturulmuş örnek bir çözüm permütasyonu

Şekil 4.2'de çözüm permütasyonuna rastgele yerleşen şehirler vektörün ilk elemanından başlanarak sırayla ziyaret edilir. Şekil 4.2'ye göre şehirlere ziyaret sırası 3-5-2-6-4-7-1-3 şeklinde olur. Burada dikkat edilecek olursa tüm şehirleri gezmesi gereken bir satıcı yukarıdaki sıralamaya göre başladığı yere tekrar dönmektedir. Bu sıralamaya göre Çizelge 4.1'deki mesafe bilgileri kullanılarak elde edilen toplam mesafe Denklem 3.4'e göre 12(3→5) + 16(5→2) + 4(2→6) + 13(6→4) + 16(4→7) + 11(7→1) + 15(1→3) = 87 olarak hesaplanır. Bulunan bu değer GSP için amaç fonksiyonunun değerini temsil eder. Popülasyondaki her birey için bu şekilde çözüm permütasyonları oluşturulur ve amaç fonksiyonunun değeri hesaplanır.

Tez çalışması kapsamında, algoritmaların GSP için komşu çözüm üretme yöntemi araya ekleme olarak seçilmiştir. Komşu çözüm, mevcut çözümü geliştirmek amacıyla çözümün permütasyonundan faydalanıp permütasyon sırasında küçük değişiklikler yaparak üretilir. Komşu çözüm, mevcut çözümün vektöründen rastgele iki indis seçilerek

başlanır. İlk seçilen indis içerisindeki şehir, permütasyon sırasından geçici olarak çıkartılacak şehirdir. Bu şehir ikinci seçilen indisin belirttiği şehrin yanına yerleştirilir ve diğer şehirler birer kaydırılır. Eğer ilk seçilen indis ikinci seçilenden küçükse ikinci indisin solunda kalan şehirler ilk indisin boşalan yerine doğru (sola) birer kaydırılır. Tersi durumda sağa doğru birer kayrılır. Böylece yeni bir çözüm permütasyonu elde edilir. Elde edilen bu yeni permütasyonun uygunluk değeri Denklem 3.4'e göre yeniden hesaplanır. Bu işlem algoritmaların yerel arama yapmasını sağlar ve popülasyondaki tüm bireyler için uygulanır. Her bireyden ne kadar komşu çözüm üretileceği Denklem 3.1, 3.2 ve 3.3'e göre belirlenir. Örnek bir komşu çözüm üretimi Şekil 4.3'te verilmiştir.

Şekil 4.3. Komşu çözüm üretme

Şekil 4.3'te mevcut çözüm permütasyonunun ikinci elemanı beşinci ve altıncı elemanların arasına yerleştirilerek yeni bir çözüm permütasyonu elde edilmektedir. Bu işlem sonucunda elde edilen yeni çözüm permütasyonu Şekil 4.4'te verilmiştir.

Şekil 4.4. Araya ekleme yöntemiyle elde edilen yeni çözüm permütasyonu

Şekil 4.4'te elde edilen yeni çözüm permütasyonu Denklem 3.4'e göre değerlendirilir. Buna göre Şekil 4.4'teki yeni çözümün toplam tur uzunluğu 74 olarak hesaplanır. Şekil 4.2'deki sıralamaya göre mevcut çözümün toplam mesafesi 87'dir. Problemdeki amaç en kısa yolu bulmak olduğundan komşu çözüm üretme yöntemi ile elde edilen Şekil 4.4'teki yeni çözüm permütasyonunun mevcut çözümden daha kaliteli bir sonuca sahip olduğu görülmektedir. Mevcut çözümden üretilen komşu çözümler çözüm kalitesine göre en iyiden kötüye doğru sıralanırlar. Komşu çözümler arasındaki en iyi çözüm mevcut çözüm ile karşılaştırılır. Eğer en iyi komşu çözüm mevcut çözümden daha iyi ise komşu çözüm mevcut çözümün yerini alır ve bir sonraki iterasyonda mevcut çözüm olarak işleme dahil edilir. Diğer komşu çözümler sırasıyla kendisinden sonra gelen çözümlerle paylaştırılır. Bu durum Şekil 3.2'de gösterilmiştir. Aynı "V" diziliminde komşu çözüm arayışı önceden belirlenen kanat çırpma parametresi (m) kadar devam

ettirilir. Ardından sürünün lider değişimi gerçekleştirilir. Lider değişiminde çözüm permütasyonlarında herhangi bir değişiklik yapılmaz sadece mevcut "V" dizilimindeki bireylerin yerleri değiştirilir. Lider değişimi ilk olarak sürünün sol tarafı üzerinde gerçekleştirilir. İkinci değişiklik sürünün sağ tarafına uygulanır. Her lider değişiminde *m* parametresi sıfırlanır. İterasyon tamamlanıncaya kadar lider değişimi sol ve sağ tarafa uygulanarak devam edilir. Bu durum Şekil 3.4'te gösterilmiştir.

4.1.1. Deneysel Sonuçlar

Orijinal MBO algoritması, ÇS-MBO algoritması ve PSO-ÇS-MBO algoritması TSPLIB kütüphanesinden (Reinelt, 1997) seçilmiş 22 simetrik GSP örneği üzerine uygulanmıştır. Seçilen GSP örneklerinin en küçüğü 51 en büyüğü 654 şehir içermektedir. Problem ismindeki sayı problemin kaç şehir içerdiğini göstermektedir. Seçilen problemlerden birisine ait detaylar aşağıda verilmiştir.

Dosya adı "berlin52.tsp" olan problem dosya ismindeki sayıdan da anlaşılacağı üzere 52 şehir içermektedir. Dosya uzantısındaki "tsp" ifadesi ise problemin simetrik bir GSP problemi olduğunu belirtmektedir. Dosya içeriğinin bir kısmı Şekil 4.5'te verilmiştir.

```
NAME: berlin52
TYPE: TSP
COMMENT: 52 locations in Berlin (Groetschel)
DIMENSION: 52
EDGE_WEIGHT_TYPE: EUC_2D
NODE_COORD_SECTION
1 565.0 575.0
2 25.0 185.0
3 345.0 750.0
4 945.0 685.0
5 845.0 655.0
6 880.0 660.0
7 25.0 230.0
```

Şekil 4.5. berlin52 dosyasının içeriği

"Name" ile ifade edilen, problemin ismini göstermektedir. "Type" problem tipini ifade ederken (TSP simetrik bir GSP olduğunu göstermektedir), "Comment" ilave yorumların (genellikle problem örneğinin veya geliştiricisinin adının verildiği yerdir) eklendiği bölümdür. "Dimension" problem boyutunu (problem içerisindeki şehir sayısı), "Edge weight type" mesafelerin (kenar ağırlıklarının) nasıl verildiğini belirtir. EUC_2D iki boyuttaki öklit uzaklığına göre işlem yapılacağını belirtir. "Node coord section" ise problem içerisindeki her bir şehrin koordinat verilerinin olduğu bölümü ifade etmektedir.

Şekil 4.5'e göre birinci şehrin iki boyutlu düzlemdeki (x,y) koordinat bilgileri (565,575) ve ikinci şehrin aynı düzlemdeki koordinat bilgileri (25,185) olduğu görülmektedir. Bu iki şehir arasındaki mesafe Denklem (4.1'e göre hesaplanır.

$$f(U) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{4.1}$$

Burada, *f(U),* koordinatları sırasıyla (x_1,y_1) ve (x_2,y_2) olan A ve B noktaları arasındaki uzaklığı ifade etmektedir. Buna göre Şekil 4.5'te verilen birinci şehir (565,575) ve ikinci şehir (25,185) arasındaki mesafe aşağıdaki gibi hesaplanır.

$$f(U) = \sqrt{(565 - 25)^2 + (575 - 185)^2} = 666{,}1081$$

Yukarıda hesaplanan mesafe en yakın tam sayıya yuvarlanır. Yani sonuç 666 olarak bulunur. Tüm şehirler arasında bir yolun var olduğu kabul edilir ve bu hesaplama işlemi problemdeki tüm şehirler arası mesafeyi bulmak için uygulanır. Simetrik GSP için iki şehir arasındaki gidiş ve dönüş mesafeleri aynıdır.

Algoritmaların başarısını kıyaslayabilmek için her algoritmada ortak olan parametreler aynı değerlerle çalıştırılmıştır. Ayrıca problemin çözümünde kullanılacak en iyi parametreleri tespit etmek için parametre testleri yapılmıştır. Testler sonucunda elde edilen uygun parametreler Çizelge 4.2'de verilmiştir.

Çizelge 4.2. GSP problemi için elde edilen en iyi parametreler

Parametreler	Değerleri
Kuş sayısı (*p*)	71
Komşu sayısı (*k*)	3
Kanat çırpma sayısı (*m*)	30
Komşu paylaşım sayısı (*x*)	1
Sürü sayısı (*z*)	15
c_1 sabiti	1
c_2 sabiti	1

Çizelge 4.2'de orijinal MBO algoritmasında kullanılan parametreler (*p, k, m* ve *x*) ÇS-MBO ve PSO-ÇS-MBO algoritmaları için de geçerlidir. ÇS-MBO algoritmasında bu parametrelere ilave olarak *z* parametresi de eklenmiştir. PSO-ÇS-MBO algoritmasında ise ÇS-MBO algoritmasına ek olarak c_1 ve c_2 parametreleri kullanılmıştır. Çizelge 4.2'de verilen parametreler kullanılarak her üç algoritma yirmi iki GSP örneği üzerinde bağımsız

olarak otuz defa çalıştırılmış ve elde edilen sonuçlar Çizelge 4.3'te verilmiştir. Çizelge 4.3'te her problem için problem ismi, şehir sayısı, bilinen en iyi sonuçlarla birlikte ilgili algoritmadan elde edilen en iyi, en kötü ve ortalama sonuçlar sütunlar halinde verilmiştir.

Çizelge 4.3. TSPLIB kütüphanesinden seçilen 22 GSP için orijinal MBO, ÇS-MBO ve PSO-ÇS-MBO algoritmalarından elde edilen sonuçlar

Problem	Şehir sayısı	Opt.	Orijinal MBO			ÇS-MBO			PSO-ÇS-MBO		
			En iyi	Ortalama	En kötü	En iyi	Ortalama	En kötü	En iyi	Ortalama	En kötü
Eil51	51	426	441	465,40	491	435	442,90	450	**426**	**427,66**	431
Berlin52	52	7542	8114	8523,16	9193	7711	8030,03	8265	**7542**	**7568,43**	7799
St70	70	675	743	823,40	922	704	746,30	771	**675**	**682,36**	695
Eil76	76	538	583	620,56	654	564	580,46	600	**538**	**546,20**	557
Pr76	76	108159	122862	130258,70	140751	114295	121320,60	128367	**108159**	**109496,46**	111392
KroA100	100	21282	24591	29716,13	33343	24438	26349,70	27642	**21282**	**21851,70**	22593
KroB100	100	22141	25703	30288,17	36143	24656	26637,13	27891	**22220**	**22792,80**	23331
Eil101	101	629	703	744,63	794	693	709,73	740	**634**	**647,30**	654
Lin105	105	14379	18798	21071,87	24809	16521	18297,50	19661	**14783**	**15127,33**	15465
Pr124	124	59030	81623	98647,30	114464	75721	87039,33	94101	**59519**	**60239,63**	61147
Bier127	127	118282	141340	152450,80	168914	136110	142295,36	148183	**119469**	**123273,56**	125519
Ch130	130	6110	7905	8522,30	9197	7240	7728,00	8019	**6161**	**6351,16**	6487
Pr136	136	96772	124389	142910,30	155380	119445	127120,33	133489	**100274**	**102923,93**	105353
Ch150	150	6528	8499	9460,60	10314	7922	8458,26	9075	**6630**	**6803,80**	6924
KroA150	150	26524	33522	40307,97	47236	31443	35827,33	37697	**27216**	**27855,40**	28476
KroB150	150	26130	34297	40706,63	46068	33261	35168,20	36478	**26764**	**27285,70**	27798
U159	159	42080	58022	68255,23	76714	56738	60836,03	64571	**42282**	**44085,00**	45903
KroA200	200	29368	42636	48423,13	54647	40956	44151,56	46996	**30809**	**31434,03**	31923
Tsp225	225	3919	5702	6157,56	6810	5406	5630,20	5993	**4082**	**4177,53**	4253
A280	280	2579	4197	4607,03	5243	3946	4231,20	4494	**2788**	**2874,20**	2963
Fl417	417	11861	47487	57292,10	75325	38835	48808,96	53203	**13308**	**13894,03**	14551
P654	654	34643	181616	225682,83	274856	184007	212538,06	225960	**42031**	**43593,66**	44487

Çizelge 4.3'te literatürde iyi bilinen yirmi iki simetrik GSP, orijinal MBO, ÇS-MBO ve PSO-ÇS-MBO algoritmaları ile çözülmüş ve algoritmalardan elde edilen sonuçların en iyi, ortalama ve en kötü sonuçları listelenmiştir. Çizelge 4.3'te problem sütunu problemin ismini, şehir sayısı ilgili problemdeki şehir sayısını (problem boyutunu), optimum sonuç ise yine ilgili problemin bilinen en iyi sonucunu göstermektedir. Aynı çizelgede kalan diğer sütunlar sırasıyla orijinal MBO, ÇS-MBO ve PSO-ÇS-MBO algoritmalarından elde edilen sonuçların en iyi, ortalama ve en kötü değeleri göstermektedir. Koyu renkte belirtilen değerler ilgili problemin, her üç algoritma ile çözümünden elde edilen en iyi değerini göstermektedir. Buna göre PSO-ÇS-MBO algoritmasından elde edilen sonuçların, ele alınan yirmi iki problemin tamamında orijinal MBO ve ÇS-MBO algoritmalarına göre daha iyi olduğu görülmektedir. Aynı çizelgeye göre, ÇS-MBO algoritması da orijinal MBO algoritmasına göre daha iyi sonuçlar üretmiştir. Her üç algoritmadan elde edilen ortalama değerler incelendiğinde PSO-ÇS-MBO algoritmasının yine tüm problemlerde daha iyi sonuçlar verdiği açıkça görülebilmektedir. Ayrıca PSO-ÇS-MBO algoritmasından elde edilen en kötü sonucun orijinal MBO algoritmasından elde edilen en iyi sonuçtan daha iyi olduğu da görülmektedir. Bu sonuçlara göre; geliştirilen ÇS-MBO ve PSO-ÇS-MBO algoritmaları GSP problemi için orijinal MBO algoritmasına göre daha başarılıdır. Ayrıca PSO-ÇS-MBO algoritmasının ÇS-MBO algoritmasına göre daha başarılı olduğu görülmektedir.

Çizelge 4.3'e göre iyileştirilen her iki yöntemin de orijinal MBO'ya göre daha iyi olduğu görülmesine karşın bu farkın anlamlı olup olmadığını anlamak için orijinal MBO, ÇS-MBO ve PSO-ÇS-MBO algoritmaları arasında istatistiksel F testi uygulanmıştır. Çizelge 4.4'te verilen sayısal sonuçlar F testindeki p değerini ifade etmektedir. yapılan testlerde önem seviyesi %95 alınmıştır. Bunun anlamı hesaplanan p değerleri %5'ten (0.05) büyükse H_0 hipotezi (sütunlardaki ilk yöntem) kabul edilir aksi halde diğer yöntem (sütunlardaki ikinci yöntem) kabul edilir. Yapılan istatistiksel test verileri 30 bağımsız çalıştırmadan elde edilen değerler ile hesaplanmıştır. Orijinal MBO ile ÇS-MBO arasındaki istatistiksel testte sadece Pr124 probleminde H_0 hipotezi kabul edilmiş (orijinal MBO ile ÇS-MBO algoritmaları arasında anlamlı bir fark yok) diğerlerinin tamamında reddedilmiştir. Bu sonuçlar, orijinal MBO ile ÇS-MBO arasında pr124 problemi haricinde diğer problemler için anlamlı bir fark olduğu anlamına gelmektedir. Benzer şekilde üçüncü sütunda orijinal MBO ile PSO-ÇS-MBO arasındaki istatistiksel analizde ele alınan tüm GSP problemleri için her iki algoritma arasında anlamlı bir farkın olduğu görülmektedir. Son sütunda iyileştirilmiş ÇS-MBO ile PSO-ÇS-MBO arasındaki

istatistiksel analize göre yine tüm problemler için anlamlı bir farkın olduğu ortaya çıkmıştır. GSP için yapılan testlerde PSO-ÇS-MBO algoritmasının ÇS-MBO ve orijinal MBO algoritmalarından daha başarılı sonuçlar verdiği açıkça görülmektedir.

Çizelge 4.4. Algoritmalara göre F istatistiki testinden elde edilen p değerleri

Problem	Orijinal MBO & ÇS-MBO (p)	Orijinal MBO & PSO-ÇS-MBO (p)	ÇS-MBO & PSO-ÇS-MBO (p)
Eil51	2,98E-06	3,16E-13	7,88E-04
Berlin52	1,06E-04	1,11E-12	7,49E-05
St70	1,23E-02	2,18E-12	3,23E-07
Eil76	6,94E-03	5,90E-17	6,51E-11
Pr76	5,35E-09	1,01E-16	5,28E-04
KroA100	1,19E-04	1,29E-12	7,50E-05
KroB100	1,30E-05	4,81E-14	5,85E-05
Eil101	1,05E-05	1,04E-16	6,25E-07
Lin105	7,13E-06	1,89E-15	9,25E-06
Pr124	**8,54E-02**	1,60E-18	1,30E-14
Bier127	1,27E-06	9,51E-21	2,43E-09
Ch130	1,01E-03	4,83E-16	8,66E-09
Pr136	3,71E-02	5,32E-20	3,27E-15
Ch150	1,11E-04	4,07E-19	3,14E-10
KroA150	2,86E-08	1,09E-19	9,06E-07
KroB150	6,94E-06	3,05E-31	3,88E-20
U159	8,52E-05	2,40E-16	1,16E-07
KroA200	8,94E-07	2,96E-18	4,51E-07
Tsp225	7,55E-06	1,46E-22	6,98E-12
A280	4,07E-08	1,25E-21	1,63E-08
Fl417	4,18E-04	4,11E-33	1,52E-24
P654	3,41E-04	2,20E-11	1,90E-08

GSP'nin çözümünde diğer yöntemlerden daha başarılı kabul edilen PSO-ÇS-MBO algoritmasının literatürdeki diğer algoritmalar ile kıyaslaması ise Çizelge 4.5 ve Çizelge 4.6'da verilmiştir. PSO-ÇS-MBO algoritması ayrık yabani ot optimizasyon algoritması (DIWO) (Zhou ve ark., 2015), geliştirilmiş karınca koloni optimizasyon algoritması (IACO) (Tuba ve Jovanovic, 2013), ayrık yapay arı koloni algoritması (DABC) (Kıran ve ark., 2013), parçacık sürü optimizasyon algoritması (PSO) (Shi ve ark., 2007), melez diferansiyel evrim algoritması (HDE) (Wang ve Xu, 2011) ve geliştirilmiş meyve sineği optimizasyon algoritması (Huang ve ark., 2017) ile kıyaslanmıştır. Çizelgelerdeki "NA" ifadesi ilgili problemin ilgili algoritma ile çözülmediği anlamına gelmektedir.

Çizelge 4.5. PSO-ÇS-MBO algoritmasının literatürden seçilen diğer algoritmalar ile karşılaştırılması

Problem	Opt.	DIWO		IACO		DABC		PSO-ÇS-MBO	
		En iyi	Hata(%)	En iyi	Hata(%)	En iyi	Hata(%)	En iyi	Hata(%)
Eil51	426	428	0.4694	427	0.2347	428	0.4694	426	**0**
Berlin52	7542	7544	0.0265	7542	**0**	7544	0.0265	7542	**0**
St70	675	677	0.2962	675	**0**	677	0.2962	675	**0**
Eil76	538	NA	NA	538	**0**	NA	NA	538	**0**
Pr76	108159	NA	NA	108358	0.1839	108159	**0**	108159	**0**
KroA100	21282	NA	NA	21282	**0**	21285	0.0140	21282	**0**
Eil101	629	NA	NA	NA	NA	653	3.8155	634	**0.7949**
Lin105	14379	NA	NA	14379	**0**	NA	NA	14783	2.8096
Pr124	59030	NA	NA	59030	**0**	NA	NA	59519	0.8283
Pr136	96772	NA	NA	96781	**0.0093**	NA	NA	100274	3.6188
A280	2579	NA	NA	NA	NA	2818	9.2671	2788	**8.1039**

Çizelge 4.6. PSO-ÇS-MBO algoritmasının literatürden seçilen diğer algoritmalar ile kıyaslaması

Problem	Opt.	PSO		HDE		IFOA		PSO-ÇS-MBO	
		En iyi	Hata(%)	En iyi	Hata(%)	En iyi	Hata(%)	En iyi	Hata(%)
Eil51	426	427	0,2347	439	3,0516	426	**0**	426	**0**
Berlin52	7542	7542	**0**	7542	**0**	7542	**0**	7542	**0**
St70	675	675	**0**	684	1,3333	675	**0**	675	**0**
Eil76	538	546	1,4869	558	3,7174	540	0,3717	538	**0**
Pr76	108159	108280	0,1118	109491	1,2315	NA	NA	108159	**0**
KroA100	21282	NA	NA	NA	NA	21282	**0**	21282	**0**
KroB100	22141	NA	NA	NA	NA	22219	**0,3522**	22220	0,3568
Eil101	629	NA	NA	NA	NA	653	0,9538	634	**0,7949**
Lin105	14379	NA	NA	NA	NA	14379	**0**	14783	2,8096
Ch150	6528	NA	NA	NA	NA	6558	**0,4595**	6630	1,5625

Kıyaslamalar için verilen çizelgelerde algoritmalardan elde edilen en iyi sonuçlar ile en iyi değere göre hesaplanan hata değerleri kıyaslanmıştır. Hata değeri Denklem (4.2'ye göre hesaplanmıştır.

$$\text{Hata} = \frac{bulunansonuç - optimum}{optimum} \times 100 \tag{4.2}$$

Çizelge 4.5'e göre PSO-ÇS-MBO algoritmasının IACO yöntemiyle yaklaşık sonuçlar verdiği buna karşın diğer yöntemlere göre daha başarılı olduğu görülmektedir. Hem PSO-ÇS-MBO hem de IACO yöntemleri on problemin altısında optimum sonucu bulmuşlardır. Benzer şekilde Çizelge 4.6'da ise PSO-ÇS-MBO algoritması IFOA algoritması ile yakın sonuçlar verdiği gözlenmiştir. PSO ve HDE algoritmaları sırasıyla iki ve bir problemde optimum sonuca ulaşabilirken IFOA algoritması beş problemde optimum değeri bulmuştur. PSO-ÇS-MBO algoritmasının on problemin altısında optimum değere ulaştığı görülmüştür. Bu sonuçlardan anlaşılabiliyor ki, iyileştirilen

PSO-ÇS-MBO algoritması, GSP'nin çözümünde literatürde mevcut bulunan algoritmalara güçlü bir alternatif yöntem olmayı kanıtlamıştır.

4.2. ÇBİYSBP'nin MBO, ÇS-MBO ve PSO-ÇS-MBO ile Çözümü

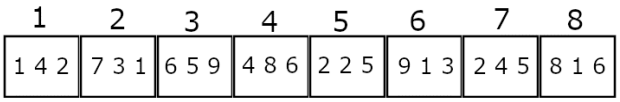
ÇBİYSBP'nin orijinal MBO algoritması ve türevleriyle çözümünde takip edilen adımlar bu bölümde detaylı bir şekilde açıklanmıştır. Başlık 3.2.2'de tanımı yapılan ve küçük bir örnek ile açıklanmaya çalışılan bu problemin algoritmalara nasıl uygulandığı yine aynı örnek ile desteklenmiştir.

Aşağıdaki gibi eleman sayısı sekiz ve problem boyutu üç olan bir veri kümesi verilmiş olsun.

SEK = {(1 4 2), (7 3 1), (6 5 9), (4 8 6), (2 2 5), (9 1 3), (2 4 5), (8 1 6)}

ÇBİYSBP'deki amaç *SEK* kümesini dengeli bir şekilde iki alt kümeye ayırmaktır. Böyle bir problemin çözülebilmesi için öncelikle *SEK* kümesindeki her bir eleman (üçlü sayı grubu) bireyin çözüm vektörünün bir hücresine yerleştirilir. Bu durumda her bireyin çözüm vektörünün uzunluğu *SEK* kümesinin eleman sayısı kadar belirlenir.

Başlangıç popülasyonu oluşturulurken *SEK* kümesi içerisindeki her eleman kodlanır. Bu kodlama sadece bir defa yapılır ve popülasyondaki tüm bireyler için rastgele çözümler üretilirken bu kodlamadan faydalanılır. Verilen küme içerisindeki üçlü gruplar bir vektörün hücrelerine sırasıyla yerleştirilir ve vektörün indis numarası üçlü sayı grubunun kodunu temsil eder. Bu durum Şekil 4.6'da gösterilmiştir.

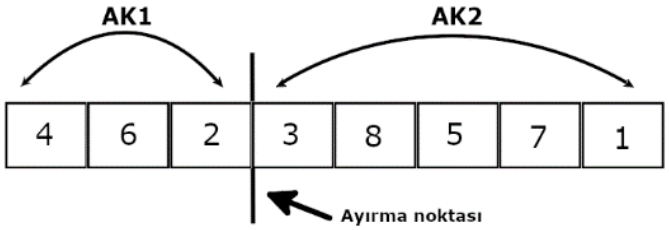


Şekil 4.6. Veri kümesinin kodlanması

Şekil 4.6'da sekiz elemanlı bir vektör içerisine sekiz elemanlı *SEK* kümesi içerisindeki üçlü sayı gruplarının sırayla yerleştirildiği görülmektedir. Üçlü sayı gruplarının üzerindeki ardışık sayılar vektörün indis numaralarını temsil etmektedir.

SEK kümesindeki kodlanmış her grup rastgele seçilir ve çözüm vektörünün hücrelerine sırayla yerleştirilir. Bu aşamada çözüm vektörü içerisine sayı grupları yerine onları temsil eden kodlar yerleştirilir. Son olarak vektörü ikiye ayıracak rastgele bir nokta seçilir ve *SEK* kümesinin iki alt kümesi elde edilmiş olur. Böylece her birey için çözüm

permütasyonu elde edilir. Bir bireye ait örnek bir çözüm permütasyonu Şekil 4.7'de verilmiştir.



Şekil 4.7. Rastgele oluşturulmuş bir çözüm permütasyonu

Şekil 4.7'de rastgele oluşturulmuş bir çözüm permütasyonu ($Ç_1$) görülmektedir. Şekle göre $Ç_1$ iki alt kümeye (AK_1 ve AK_2) ayrılmıştır. Çözüm permütasyonundaki ayırma noktası rastgele belirlenmiştir. Buna göre rastgele oluşturulmuş $Ç_1$ kümesi ve bu kümeye ait AK_1 ve AK_2 alt kümeleri aşağıdaki gibidir.

$Ç_1$ = {(4 8 6), (9 1 3), (7 3 1), (6 5 9), (8 1 6), (2 2 5), (2 4 5), (1 4 2)}

AK_1 ={(4 8 6), (9 1 3), (7 3 1)}

AK_2 ={(6 5 9), (8 1 6), (2 2 5), (2 4 5), (1 4 2)}

Çözüm permütasyonu ve alt kümeler belirlendikten sonra Denklem 3.5'e göre $Ç_1$'in uygunluk değeri hesaplanır.

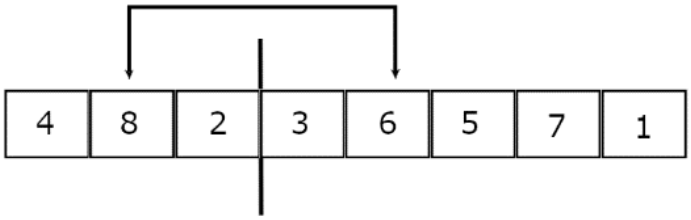
AK_1'in toplamı = {(20 12 10)}

AK_2'nin toplamı = {(19 16 27)}

AK_1 ve AK_2 kümeleri arasındaki fark = {(1 4 17)}

$f(Ç_1)$ = max{(1 4 17)} = 17

ÇBİYSBP'nin çözümünde komşuluk yöntemi olarak hem takas hem de araya ekleme yöntemleri birlikte kullanılmıştır. Orijinal MBO'nun yapısından dolayı popülasyondaki her birey için en az üç komşu çözüm üretilmiştir. Çalışmada üretilecek komşu çözümler sırasıyla birinci komşu için takas, ikinci komşu için ayırma noktasını kaydırma ve üçüncü komşu için yine takas olarak belirlenmiştir. Eğer üçten daha fazla komşu çözüm üretilecekse yine bu sıra dikkate kalınarak komşu çözüm üretmeye devam edilir. Bu örnekte üç komşu çözüm üretildiği düşünülmüştür. Şekil 4.7'deki çözüm permütasyonundan takas yöntemi ile elde edilecek yeni bir çözüm permütasyonu Şekil 4.8'deki gibidir.



Şekil 4.8. Takas yöntemi ile elde edilen birinci komşu çözüm

Takas yöntemi ile komşu çözüm üretirken çözüm vektörünün iki indisi rastgele belirlenir. Ancak seçilen iki indisin aynı alt kümeye ait olmaması kontrol edilir. Aynı alt kümeden seçilen iki indis içerisindeki kodlanmış elemanların yer değiştirmesi Denklem 3.5'e göre sonucu değiştirmeyecektir. Bundan dolayı seçilen indislerin Şekil 4.8'deki gibi iki farklı kümeden seçilmesi gerekmektedir. Seçilen iki indis içerisindeki kodlamalar karşılıklı yer değiştirilir ve böylece yeni bir çözüm permütasyonu elde edilir. Şekil 4.8'de ikinci ve beşinci indisler içerisindeki sekiz ve altı kodlu sayı grupları karşılıklı yer değiştirerek yeni bir dizilim elde edilmiştir. Bu yeni çözüm permütasyonuna göre çözümün uygunluk değeri Denklem 3.5'e göre yeniden hesaplanır. Bu hesaplamanın adımları aşağıda verilmiştir.

$KÇ_1$ = {(4 8 6),(8 1 6),(7 3 1),(6 5 9),(9 1 3),(2 2 5),(2 4 5),(1 4 2)}

Burada $KÇ_1$ üretilen birinci komşu çözümü belirtmektedir. $KÇ_1$'e ait alt kümler ($AK_{1,KÇ1}$ ve $AK_{2,KÇ1}$) çözümün devamında verilmiştir.

$AK_{1,KÇ1}$ = {(4 8 6),(8 1 6),(7 3 1)}

$AK_{2,KÇ1}$ = {(6 5 9),(9 1 3),(2 2 5),(2 4 5),(1 4 2)}

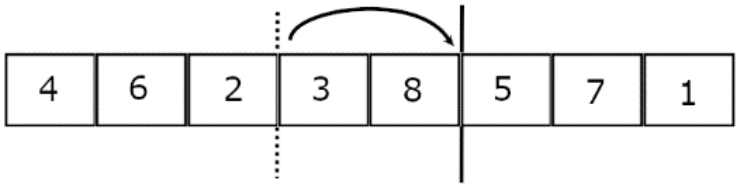
$AK_{1,KÇ1}$'in toplamı = {(19 12 13)}

$AK_{2,KÇ1}$'in toplamı = {(20 16 29)}

$AK_{1,KÇ1}$ ve $AK_{2,KÇ1}$ kümeleri arasındaki fark = {(1 4 16)}

$f(KÇ_1)$ = max{(1 4 16)} = 16

İkinci komşu çözüm üretme yöntemi mevcut çözümde ayırma noktasının sağa veya sola kaydırılması olarak belirlenmiştir. Mevcut çözüm üzerindeki ayırma noktasının konumu rastgele belirlenen bir başka konuma taşınarak yeni alt kümeler elde edilir. Çözüm permütasyonu değiştirmeden alt küme elemanlarının değişimiyle (ayırma noktasının değiştirilmesinden dolayı) elde edilen yeni çözüm permütasyonu Şekil 4.9'da verilmiştir.



Şekil 4.9. Ayırma noktasının konumunu değiştirerek elde edilen ikinci komşu çözüm

Şekil 4.9'da görüldüğü gibi mevcut çözümün permütasyonunda hiçbir değişiklik yapılmamıştır. Buna karşın mevcut çözüm üzerindeki ayırma noktası iki eleman sağa kaydırılmıştır. Böylelikle alt kümelerin elemanları arasında bir geçiş yapılmıştır. 3 ve 8 kodlu elemanlar daha önce ikinci alt kümede iken ayırma noktasının kaydırılması ile ikinci alt kümeden çıkartılıp birinci alt kümeye dahil edilmiştir. Bu işlem ile mevcut çözüm üzerinde rastgele belirlenen ayırma noktası komşu çözüm üretirken farklı konumlara yerleştirilerek daha dengeli iki alt küme bulunması amaçlanmıştır. Üretilen ikinci komşu çözüm ($KÇ_2$) ve ona ait alt kümeler ($AK_{1,KÇ2}$ ve $AK_{2,KÇ2}$) aşağıdaki gibidir.

$KÇ_2$ = {(4 8 6),(8 1 6),(7 3 1),(6 5 9),(9 1 3),(2 2 5),(2 4 5),(1 4 2)}

$AK_{1,KÇ2}$ = {(4 8 6),(9 1 3),(7 3 1),(6 5 9),(8 1 6)}

$AK_{2,KÇ2}$ = {(2 2 5),(2 4 5),(1 4 2)}

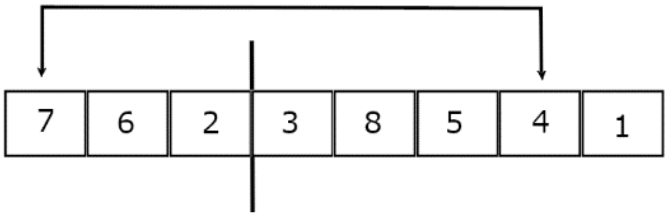
$AK_{1,KÇ2}$'in toplamı = {(34 18 25)}

$AK_{2,KÇ2}$'in toplamı = {(5 10 12)}

$AK_{1,KÇ2}$ ve $AK_{2,KÇ2}$ kümeleri arasındaki fark = {(29 8 13)}

$f(KÇ_2)$ = max{(29 8 13)} = 29

Üçüncü komşu çözüm tekrar takas yöntemiyle oluşturulur. Yine mevcut çözüm vektöründen rastgele iki indis seçilir (seçilen indisler ayırma noktasının farklı tarafında olmak şartı ile) ve bu indisler içerisindeki kodlanmış sayı grupları yer değiştirilir. Takas yöntemiyle üretilen üçüncü komşu çözüm Şekil 4.10'da verilmiştir.



Şekil 4.10. Takas yöntemiyle oluşturulan üçüncü komşu çözüm

Şekil 4.10'da üretilen üçüncü komşu çözüm ($KÇ_3$) mevcut çözümdeki ilk indis ile yedinci indis içerisindeki değerlerin yer değiştirmesiyle oluşturulmuştur. Buna göre $KÇ_3$'e ait çözüm kümesi, alt kümeleri ve çözümün uygunluk değeri aşağıdaki gibi hesaplanır.

$KÇ_3$ = {(2 4 5),(9 1 3),(7 3 1),(6 5 9),(8 1 6),(2 2 5),(4 8 6),(1 4 2)}

$AK_{1,KÇ3}$ = {(2 4 5),(9 1 3),(7 3 1)}

$AK_{2,KÇ3}$ = {(6 5 9),(8 1 6),(2 2 5),(4 8 6),(1 4 2)}

$AK_{1,KÇ3}$'ün toplamı = {(18 8 9)}

$AK_{2,KÇ3}$'ün toplamı = {(21 20 28)}

$AK_{1,KÇ3}$ ve $AK_{2,KÇ3}$ kümeleri arasındaki fark = {(3 12 19)}

$F(KÇ_3)$ = max{(3 12 19)} = 19

Tüm komşu çözümler üretildikten sonra bu çözümler uygunluk değerine göre en iyiden en kötüye doğru sıralanırlar. Bu problemdeki amaç en küçük farkı bulmak olduğuna göre komşu çözümler uygunluk değerine bakılarak küçükten büyüğe doğru sıralanırlar ($KÇ_1$ = 16, $KÇ_3$ = 19, $KÇ_2$ = 29). Bu çözümler arasındaki en iyi komşu çözüm ($KÇ_1$) mevcut çözümü ($Ç_1$) geliştirmek için kullanılır. $KÇ_1$'in uygunluk değeri $Ç_1$'in uygunluk değerinden daha iyi olduğu için $KÇ_1$, $Ç_1$'in yerini alır. Diğer komşu çözümler ise $Ç_1$'den sonra gelen çözümlerle paylaştırılır.

4.2.1. ÇBİYSBP için deneysel sonuçlar

Literatürde ÇBİYSBP ile ilgili mevcut çalışmalar olmasına rağmen henüz paylaşılmış bir test verisi bulunamamıştır. Bundan dolayı probleme ait veri setleri rastgele oluşturulmuştur. Farklı büyüklükte ve boyutta 126 adet veri kümesi oluşturulmuş ve küme elemanları gerçel sayılardan seçilmiştir. Örnek bir veri dosyasına ait detaylar aşağıda verilmiştir.

Problem ismi: 50_2_a

Problem ismindeki 50 sayısı kümenin eleman sayısını ifade etmektedir. Daha sonra gelen 2 değeri ise kümedeki her bir elemanın iki boyutlu olduğunu belirtmektedir. Problem isminin sonundaki "a" harfi ise aynı eleman sayısına ve boyuta sahip diğer problemleri birbirinden ayırmak için kullanılmıştır. Bu probleme ait veri kümesinin bir kısmı Şekil 4.11'de gösterilmiştir.

```
50
2
5066.196
1250.635
3387.766
7077.036
9509.299
2725.622
6912.584
1355.913
9453.943
6402.922
7926.986
4146.824
```

Şekil 4.11. ÇBİYSBP'ye ait 50 elemanlı ve 2 boyutlu bir problemin örnek veri seti

Şekil 4.11'de, veri setindeki ilk değer (50) problemdeki kümenin eleman sayısını, ikinci değer (2) ise her bir elemanın boyutunu ifade eder. Daha sonraki değerler ise küme içerisine yerleşen sayılardır. Üçüncü ve dördüncü satırda bulunan değerler kümenin ilk elemanını oluşturur. Kümenin diğer elemanları da aynı yolla elde edilir. Buna göre kümenin ilk iki elemanı aşağıdaki gibi gösterilebilir.

SEK = {(5066.196 1250.635), (3387.766 7077.036), …}

Oluşturulan bu küme Şekil 4.6'daki gibi kodlanıp her üç algoritma ile çözümü gerçekleştirilmiştir. ÇBİYSBP'nin çözümü için orijinal MBO, ÇS-MBO ve PSO-ÇS-MBO algoritmalarının parametreleri Çizelge 4.7'de verilmiştir.

Çizelge 4.7. ÇBİYSBP için elde edilen en iyi parametreler

Parametreler	Değerleri
Kuş sayısı (p)	25
Komşu sayısı (k)	11
Kanat çırpma sayısı (m)	20
Komşu paylaşım sayısı (x)	1
Sürü sayısı (z)	50
c_1 sabiti	1
c_2 sabiti	1

Çizelge 4.7'de verilen parametreler kullanılarak her üç algoritma 126 veri seti üzerinde bağımsız olarak otuz defa çalıştırılmış ve elde edilen sonuçlar Çizelge 4.8 ile 4.14 arasında verilmiştir. Algoritmaların durdurma kriteri 900 saniye olarak belirlenmiştir.

Çizelge 4.8. 50 boyutlu veri seti için sonuçlar

Problem	Orijinal MBO			ÇS-MBO			PSO-ÇS-MBO		
	En iyi	Ortalama	En kötü	En iyi	Ortalama	En kötü	En iyi	Ortalama	En kötü
50_2_a	50,51	99,77	174,43	0,77	3,10	5,29	**0,61**	**0,99**	1,47
50_2_b	52,82	72,39	92,35	1,18	3,49	6,30	**0,44**	**0,93**	1,33
50_2_c	3,67	32,97	77,25	1,53	2,71	4,90	**0,54**	**0,99**	1,72
50_3_a	296,59	370,16	582,12	40,95	62,92	84,26	**14,33**	**25,26**	36,14
50_3_b	156,86	277,21	445,79	**8,45**	36,11	56,87	15,04	**25,53**	35,47
50_3_c	56,10	373,95	711,75	37,70	48,53	83,52	**9,39**	**24,24**	34,41
50_5_a	947,36	1233,76	1640,61	**152,86**	310,91	418,61	235,19	**292,47**	390,81
50_5_b	1077,86	1275,65	1615,20	346,28	423,71	603,62	**40,56**	**240,25**	307,21
50_5_c	723,43	1368,64	1767,22	263,50	414,74	554,05	**94,57**	**257,28**	341,65
50_10_a	4576,17	4814,21	5137,60	1716,53	2349,67	2741,66	**1275,66**	**1837,35**	2593,70
50_10_b	3757,94	4183,74	4686,71	**1779,23**	2216,98	2446,73	2030,83	**2157,20**	2349,27
50_10_c	3687,78	4168,60	4872,37	**1897,14**	**2215,24**	2711,87	1918,53	2224,89	2385,17
50_15_a	5268,75	6207,74	6862,84	4070,78	**4375,41**	4706,48	**3838,31**	4462,25	4727,06
50_15_b	5727,86	6557,73	6891,89	**3067,27**	4280,04	4752,30	4089,65	**4253,54**	4444,78
50_15_c	5813,60	6539,22	7350,15	3866,28	4246,71	4498,49	**3600,91**	**4139,78**	4598,64
50_20_a	8528,63	9473,54	11283,14	**5646,70**	**6374,41**	7000,44	5865,81	6492,06	7102,60
50_20_b	8998,19	10233,55	12793,39	**5327,76**	**6111,68**	6689,89	5742,39	6284,28	6506,16
50_20_c	8611,27	9095,14	9802,66	**4412,49**	**6101,24**	6791,55	6109,65	6376,09	6666,68

Çizelge 4.9. 100 boyutlu veri seti için sonuçlar

Problem	Orijinal MBO			ÇS-MBO			PSO-ÇS-MBO		
	En iyi	Ortalama	En kötü	En iyi	Ortalama	En kötü	En iyi	Ortalama	En kötü
100_2_a	12,75	28,08	45,05	**0,33**	1,67	2,54	0,41	**0,96**	1,79
100_2_b	10,13	37,72	80,18	0,41	1,46	2,60	**0,27**	**0,70**	1,13
100_2_c	5,28	31,31	55,02	0,11	1,55	2,83	**0,05**	**0,56**	0,92
100_3_a	86,65	199,71	278,36	13,02	27,93	39,54	**9,67**	**16,46**	24,40
100_3_b	181,59	293,08	447,45	9,27	31,88	56,44	**7,27**	**20,26**	29,94
100_3_c	141,12	198,26	252,56	**8,64**	24,73	35,85	12,24	**17,05**	22,72
100_5_a	781,74	1119,35	1277,19	**205,42**	**288,61**	390,55	246,35	305,19	389,93
100_5_b	798,03	978,87	1052,89	**173,43**	**243,51**	334,52	174,96	260,57	363,10
100_5_c	730,11	1021,30	1163,66	**164,68**	284,29	370,86	210,51	**272,94**	328,62
100_10_a	2825,16	3730,45	4275,76	**1487,14**	**1847,31**	2321,47	1730,34	1979,38	2296,19
100_10_b	3531,74	3754,49	3967,67	**1203,16**	**1653,70**	1873,30	1626,05	2078,99	2429,13
100_10_c	3476,71	3788,65	4050,88	**1007,10**	**1767,38**	2013,10	1870,93	2035,91	2191,52
100_15_a	5299,44	5968,88	6481,80	**3489,45**	**3837,82**	4293,18	3982,08	4357,22	4894,36
100_15_b	5252,80	5961,08	6601,29	**3225,04**	**3693,84**	4116,07	3836,09	4168,08	4525,77
100_15_c	5372,25	5603,05	5802,67	**3226,18**	**3663,09**	4084,36	3643,52	4124,36	4504,04
100_20_a	7291,36	7838,15	8321,13	**5291,33**	**5551,41**	5958,57	5381,10	6135,36	6775,11
100_20_b	5852,76	7631,31	8443,24	**5130,22**	**5448,00**	5888,26	5025,18	5786,84	6319,97
100_20_c	6438,94	8146,66	8814,61	**4286,00**	**5307,75**	5675,00	5904,89	6185,19	6382,47

Çizelge 4.10. 200 elemanlı veri seti için sonuçlar

Problem	Orijinal MBO			ÇS-MBO			PSO-ÇS-MBO		
	En iyi	Ortalama	En kötü	En iyi	Ortalama	En kötü	En iyi	Ortalama	En kötü
200_2_a	4,17	14,82	25,14	**0,26**	**0,86**	1,38	0,65	1,02	1,61
200_2_b	4,31	12,43	29,52	**0,15**	**0,60**	0,97	0,38	1,13	1,76
200_2_c	7,50	9,71	13,44	**0,01**	**0,44**	1,19	0,44	0,98	1,46
200_3_a	79,27	128,85	169,26	6,17	**16,07**	24,90	**5,54**	19,28	29,11
200_3_b	79,89	141,41	176,53	**7,79**	**14,49**	24,27	11,16	22,33	32,04
200_3_c	28,80	71,71	104,67	**4,06**	**10,97**	15,92	15,57	20,28	24,72
200_5_a	363,49	607,17	878,23	**155,73**	**213,37**	270,86	184,49	241,03	305,73
200_5_b	512,19	791,96	1087,11	**104,96**	**219,33**	376,13	241,60	333,94	447,14
200_5_c	639,16	843,17	1089,31	**138,62**	**202,86**	284,87	191,70	245,00	305,95
200_10_a	2120,74	3101,37	3560,73	1396,10	**1707,08**	2033,30	**1363,16**	1723,97	1897,79
200_10_b	2298,47	2924,67	3664,71	**1278,10**	**1574,23**	1786,19	1419,35	1921,79	2231,25
200_10_c	1913,84	2653,89	3043,30	**1490,46**	**1691,58**	1991,17	1839,65	2119,50	2295,78
200_15_a	4559,75	5435,86	5859,01	**3370,60**	**3759,34**	4120,92	3642,77	4062,93	4306,69
200_15_b	4899,51	5480,55	5910,29	**3125,57**	**3552,11**	3935,05	3768,88	4029,54	4210,80
200_15_c	4448,86	5212,52	5926,46	**3100,85**	**3358,35**	3763,74	3329,22	3749,49	4073,58
200_20_a	6881,77	7265,05	7654,07	**4669,36**	**5177,91**	5431,39	5588,74	5772,82	6080,61
200_20_b	6784,02	7182,14	7418,98	**4424,27**	**4969,56**	5537,81	5328,66	5620,85	5883,27
200_20_c	6050,85	6492,54	7179,49	**4405**,41	**5201,63**	5821,92	5283,97	5665,27	5924,29

Çizelge 4.11. 300 boyutlu veri seti için sonuçlar

Problem	Orijinal MBO			ÇS-MBO			PSO-ÇS-MBO		
	En iyi	Ortalama	En kötü	En iyi	Ortalama	En kötü	En iyi	Ortalama	En kötü
300_2_a	7,54	10,52	14,10	**0,19**	**0,51**	1,07	0,26	0,72	1,17
300_2_b	5,70	14,99	25,20	**0,13**	**0,85**	1,58	0,67	1,21	1,67
300_2_c	5,27	11,28	16,97	**0,14**	**0,46**	0,79	0,71	1,19	1,91
300_3_a	70,78	88,52	112,72	**3,00**	**12,28**	21,46	11,70	16,57	23,33
300_3_b	30,15	80,87	141,53	**2,63**	**12,13**	19,16	16,91	21,48	27,74
300_3_c	45,41	104,65	152,02	**6,23**	**12,26**	21,84	9,02	19,27	38,68
300_5_a	466,83	589,43	707,56	**151,61**	**235,71**	290,34	213,43	271,91	349,45
300_5_b	469,96	713,10	868,23	**115,87**	**190,05**	263,54	238,86	284,51	314,75
300_5_c	330,93	527,66	673,94	**119,53**	**185,34**	252,95	186,94	225,49	260,95
300_10_a	2121,61	2758,45	3198,05	**1516,08**	**1734,59**	1953,29	1385,47	1957,85	2146,61
300_10_b	2830,30	3024,55	3113,54	**952,01**	**1531,98**	1918,92	1886,92	2014,12	2190,43
300_10_c	2659,68	2936,57	3330,66	**1099,55**	**1432,05**	1675,13	1331,73	1790,45	2084,35
300_15_a	4424,71	5044,22	5511,69	**2563,75**	**3453,72**	4016,09	3347,09	3843,68	4361,40
300_15_b	4203,03	4903,09	5333,90	**2854,33**	3444,81	3698,47	2989,69	**3428,92**	3619,91
300_15_c	4514,88	5053,10	5461,46	**2493,52**	**3237,84**	3768,12	3456,14	3717,37	3982,38
300_20_a	5852,63	6783,87	7558,84	**4266,82**	**5090,33**	5598,09	5229,10	5647,30	6008,35
300_20_b	6200,16	6648,64	7249,41	**4826,28**	**5291,54**	5704,92	5186,69	5352,58	5452,06
300_20_c	6379,59	6809,15	7225,76	**4349,06**	**5000,83**	5523,35	5159,57	5566,93	6009,62

Çizelge 4.12. 400 boyutlu veri seti için sonuçlar

Problem	Orijinal MBO			ÇS-MBO			PSO-ÇS-MBO		
	En iyi	Ortalama	En kötü	En iyi	Ortalama	En kötü	En iyi	Ortalama	En kötü
400_2_a	0,82	5,16	9,15	**0,05**	**0,35**	0,79	0,41	0,85	1,53
400_2_b	1,88	5,72	9,47	**0,10**	**0,48**	0,82	0,38	0,59	0,85
400_2_c	3,07	7,79	12,61	**0,27**	**0,54**	1,09	0,34	0,98	2,27
400_3_a	22,36	62,00	105,57	**4,01**	**10,60**	14,85	10,22	16,18	22,84
400_3_b	53,23	98,98	156,47	**5,91**	**10,89**	16,33	10,99	20,31	25,73
400_3_c	38,73	89,74	147,55	**4,76**	**12,45**	22,99	18,03	26,35	41,66
400_5_a	331,42	521,34	588,75	**111,21**	**204,16**	299,61	96,88	216,48	314,01
400_5_b	432,00	506,88	584,24	**149,55**	**202,12**	259,61	207,31	255,28	354,15
400_5_c	500,20	631,34	769,15	177,37	**220,21**	249,27	**77,28**	231,88	333,18
400_10_a	2369,64	2753,03	3046,81	**1270,18**	**1605,95**	1853,47	1556,74	1768,10	2024,52
400_10_b	2019,79	2413,60	2906,21	**1242,11**	**1632,29**	1883,26	1340,07	1591,78	1976,58
400_10_c	2202,05	2583,17	2855,13	**1163,41**	**1534,60**	1882,50	1548,32	1819,00	1974,05
400_15_a	4245,09	4625,92	5002,77	3202,18	**3438,41**	3676,91	**3198,96**	3387,83	3719,78
400_15_b	4072,67	4265,72	4471,41	3113,68	3474,72	3937,28	**3086,10**	**3364,21**	3702,62
400_15_c	4613,08	4800,28	4954,75	**3092,12**	3670,39	3992,84	3438,96	**3655,74**	4001,83
400_20_a	5997,45	6480,23	6780,93	**4985,47**	**5470,12**	5837,43	5421,22	5821,96	6197,42
400_20_b	5392,15	6018,64	6354,09	**4492,01**	**5167,28**	5667,01	4810,04	5379,26	5904,49
400_20_c	5903,94	6430,01	6917,70	**4559,85**	**5246,73**	5785,59	5015,64	5416,87	5709,30

Çizelge 4.13. 500 boyutlu veri seti için sonuçlar

Problem	Orijinal MBO			ÇS-MBO			PSO-ÇS-MBO		
	En iyi	Ortalama	En kötü	En iyi	Ortalama	En kötü	En iyi	Ortalama	En kötü
500_2_a	2,70	3,80	6,01	**0,26**	**0,64**	0,85	0,42	0,79	1,14
500_2_b	4,02	7,20	10,26	**0,11**	**0,53**	0,90	0,27	0,77	1,87
500_2_c	2,46	5,04	9,24	**0,05**	**0,52**	1,13	0,23	0,76	1,23
500_3_a	18,94	57,14	81,15	**5,03**	**11,75**	24,19	12,44	15,13	17,65
500_3_b	31,39	65,68	102,10	**4,36**	**14,31**	26,97	7,35	16,61	25,02
500_3_c	61,85	78,25	108,00	**5,01**	**13,55**	20,52	12,88	22,04	30,02
500_5_a	370,69	512,48	666,44	**88,43**	**205,17**	256,94	226,18	252,98	275,07
500_5_b	402,85	484,54	571,90	**105,05**	219,36	315,05	144,03	**200,70**	270,63
500_5_c	423,46	494,62	623,38	127,59	202,06	265,03	**102,83**	**198,29**	308,04
500_10_a	2232,44	2453,37	2780,36	**956,92**	**1663,46**	2110,03	1665,59	1884,99	2040,89
500_10_b	2617,45	2722,67	2837,41	**1468,97**	**1690,49**	1967,84	1698,76	1815,86	1980,16
500_10_c	1899,38	2518,68	2956,22	1579,45	1778,33	2120,70	**1280,95**	**1674,93**	1931,47
500_15_a	4268,07	4714,35	5241,42	**2657,45**	**3318,08**	3909,38	2844,44	3428,99	4150,83
500_15_b	3472,89	4358,65	5065,81	**2922,33**	**3466,90**	3980,65	3588,86	3828,25	3989,51
500_15_c	3788,58	4322,47	4670,28	**2782,07**	**3464,73**	3858,72	3021,30	3637,40	4004,90
500_20_a	6108,27	6555,99	6900,90	**4743,35**	**5476,98**	6017,23	5392,18	5704,82	5968,15
500_20_b	6255,10	6417,72	6618,89	**4750,79**	**5383,94**	5734,86	5124,83	5318,59	5571,27
500_20_c	5568,97	6150,69	6887,54	**4877,73**	**5236,79**	5614,04	5337,86	5596,50	5971,01

Çizelge 4.14. 1000 boyutlu veri seti için sonuçlar

Problem	Orijinal MBO			ÇS-MBO			PSO-ÇS-MBO		
	En iyi	Ortalama	En kötü	En iyi	Ortalama	En kötü	En iyi	Ortalama	En kötü
1000_2_a	2,32	3,40	4,51	**0,23**	0,86	1,50	0,31	**0,71**	1,15
1000_2_b	0,18	2,72	5,02	**0,08**	**0,82**	1,76	0,53	0,91	1,24
1000_2_c	1,28	2,93	4,15	0,63	1,14	2,17	**0,40**	**0,75**	1,02
1000_3_a	5,50	37,51	49,46	**3,44**	**17,59**	29,55	10,41	18,78	30,28
1000_3_b	30,49	49,55	64,49	**5,62**	18,38	30,75	7,61	**17,05**	34,25
1000_3_c	22,04	38,45	66,61	7,53	19,76	28,14	**7,33**	**17,58**	25,56
1000_5_a	238,70	389,63	524,49	**123,78**	263,43	371,27	143,17	**232,77**	303,62
1000_5_b	363,65	441,96	540,91	**133,67**	**258,45**	361,30	226,73	262,99	306,25
1000_5_c	271,74	335,87	447,16	**212,80**	266,15	326,25	227,75	**247,89**	283,39
1000_10_a	1850,05	2042,85	2215,27	**1505,39**	1959,97	2265,12	1806,43	**1932,22**	2155,24
1000_10_b	1943,95	2203,90	2396,50	**1308,36**	1738,72	2075,03	1437,01	**1747,32**	2334,15
1000_10_c	2108,94	2394,86	2564,83	**1401,82**	**1924,84**	2254,52	1786,57	1994,58	2151,13
1000_15_a	3905,06	4169,78	4421,72	3563,96	3994,00	4281,04	**2998,34**	**3809,39**	4281,13
1000_15_b	3669,35	3996,20	4560,80	**3463,57**	4140,49	4541,16	3765,32	**3987,77**	4428,02
1000_15_c	3702,45	3996,72	4368,53	**3305,72**	4067,10	4592,66	3404,13	**3710,73**	3978,02
1000_20_a	5753,75	5914,54	6260,72	**5114,24**	6233,98	7139,37	5631,88	**5872,12**	6221,56
1000_20_b	5171,12	5627,74	6019,55	6093,98	6567,71	6967,45	**4514,19**	**5355,76**	6046,20
1000_20_c	5782,54	5985,75	6055,46	**5136,73**	6549,72	7128,96	5856,08	**5957,44**	6067,46

Çizelge 4.8 ile 4.14 arasında sırasıyla 50, 100, 200, 300, 400, 500 ve 1000 boyutlu problemler, yine sırasıyla 2, 3, 5, 10, 15 ve 20 koordinata sahip ve her bir koordinat için üç farklı veri grubu olmak üzere toplamda 126 test problemi kullanılmıştır. Her bir problem orijinal MBO, ÇS-MBO ve PSO-ÇS-MBO algoritmaları ile çözülmüştür.

Çizelge 4.8'de her üç algoritmadan elde edilen en iyi sonuçlara göre, PSO-ÇS-MBO algoritması on sekiz problemin onunda en iyi değere sahiptir. Buna karşın yine elde edilen en iyi değerler karşılaştırıldığında on sekiz problemin sekizinde ÇS-MBO algoritması en iyi değere sahiptir. Ortalama değerler incelendiğinde on sekiz problemin on üçünde PSO-ÇS-MBO algoritması, kalan beş problemde ise ÇS-MBO algoritması en iyi değerlere sahiptir.

100 boyutlu veri setine sahip Çizelge 4.9'da, en iyi sütunlarındaki sonuçlara göre ÇS-MBO algoritması on dört problemde diğer algoritmalara göre daha iyi sonuçlar verirken PSO-ÇS-MBO algoritması kalan dört problemde en iyi sonucu vermiştir. Ortalama değerlere bakıldığında ÇS-MBO algoritması on bir problemde en başarılı sonuçlar üretirken kalan yedi problemde PSO-ÇS-MBO algoritması en iyi sonucu vermiştir. Ayrıca 100 boyutlu bu problemlerde ÇS-MBO ve PSO-ÇS-MBO algoritmalarından elde edilen sonuçların birbirine yakın olduğu da görülmüştür.

Çizelge 4.10'daki 200 boyutlu problemlerin çözümünde ÇS-MBO algoritması on sekiz problemin on altısında diğer algoritmalara göre daha başarılı sonuçlar verirken, PSO-ÇS-MBO algoritması sadece iki problemde başarılı sonuçlar üretmiştir. Ayrıca ortalama değerlere bakıldığında ÇS-MBO algoritmasının on sekiz problemin tamamında diğer algoritmalara göre daha başarılı olduğu açıkça görülmektedir.

300 boyutlu problemlerin bulunduğu Çizelge 4.11'de en iyi değerlerin bulunduğu sütunlardaki değerler incelendiğinde ÇS-MBO algoritmasının on sekiz problemin tamamında diğer algoritmalara göre daha başarılı olduğu görülmüştür. Ortalama değerlere göre PSO-ÇS-MBO algoritması sadece tek bir problemde en iyi ortalama değere sahipken, kalan on yedi problemin tamamında ÇS-MBO algoritması daha başarılı sonuçlar elde üretmiştir.

Çizelge 4.12'de her üç algoritmanın en iyi değerleri ile kıyaslama yapıldığında, ÇS-MBO algoritmasının on beş problemde diğer iki algoritmaya göre daha başarılı sonuçlar verdiği görülmüştür. Buna karşın üç problemde PSO-ÇS-MBO algoritması diğer iki algoritmaya göre daha başarılı olmuştur. Ortalama değerlere bakıldığında, ÇS-MBO algoritmasının on sekiz problemin on altısında diğer algoritmalara göre başarılı olduğu, PSO-ÇS-MBO algoritmasının ise sadece iki problemde öne çıktığı görülmüştür.

500 boyutlu problemlerin bulunduğu Çizelge 4.13'teki en iyi değerlere göre, ÇS-MBO algoritması on sekiz problemin on altısında diğer yöntemlere göre daha başarılı olduğu, iki problemde ise PSO-ÇS-MBO algoritmasının daha iyi sonuçlar elde ettiği görülmüştür. Ortalama değerlere göre, ÇS-MBO algoritması on beş problemde diğer yöntemlere göre daha başarılı iken, PSO-ÇS-MBO algoritması üç problemde diğer algoritmalara göre daha başarı sonuçlar vermiştir.

Çizelge 4.14'te bulunan 1000 boyutlu problemler için her üç algoritmadan elde edilen en iyi değerlere göre, ÇS-MBO algoritması on dört problemde daha iyi sonuçlar verirken, PSO-ÇS-MBO algoritması dört problemde diğer yöntemlere göre daha iyi sonuçlar vermiştir. Ortalamalara göre değerlendirildiğinde, PSO-ÇS-MBO algoritmasının öne çıktığı görülmüştür. On sekiz problemin on dördünde daha başarılı olan PSO-ÇS-MBO algoritmasına karşın, ÇS-MBO algoritması dört problemde daha iyi sonuçlar vermiştir.

Genel olarak tüm sonuçlar değerlendirildiğinde ÇS-MBO algoritmasının bu problem türünde daha başarılı olduğu gözlenmiştir. 50 boyutlu problemlerin çözümünde PSO-ÇS-MBO algoritmasının hem en iyi değerlere göre hem de ortalama değerlere göre orijinal MBO ve ÇS-MBO algoritmasından daha başarılı sonuçlar verdiği görülmüştür. Ayrıca 1000 boyutlu problemlerin çözümünden elde edilen ortalama değerlere göre PSO-ÇS-MBO algoritmasının diğer iki algoritmaya göre daha başarılı olduğu tespit edilmiştir. Buna karşın ÇS-MBO algoritmasının 100, 200, 300, 400 ve 500 boyutlu problemlerin çözümünden elde edilen en iyi ve ortalama değerlere göre orijinal MBO ve PSO-ÇS-MBO algoritmalarından çok daha iyi sonuçlar ürettiği açıkça görülmüştür.

Sonuç olarak, ÇBİYSBP'nin çözümünde, orijinal MBO algoritmasının iyileştirilmiş versiyonları olan ÇS-MBO ve PSO-ÇS-MBO algoritmalarının, orijinal MBO algoritmasına göre çok daha başarılı olduğu görülmüş ve yapılan iyileştirmelerin etkin ve verimli olduğu sonucu ortaya çıkmıştır.

Yapılan deneysel çalışmalardan elde edilen sonuçlar arasındaki farkın anlamlı olup olmadığı istatistiksel olarak da incelenmiştir. Bunun için F testi kullanılmış ve %95 önem seviyesine göre doğruluğu sınanmıştır.

Çizelge 4.15 ile 4.18 arasındaki çizelgelerdeki sayısal değerler, ilgili sütunda belirtilen iki algoritma için F testinden elde dilen p değerini göstermektedir. Koyu olarak gösterilen değerler p değerinin 0,05'ten küçük olduğu değerlerdir.

Çizelge 4.15. ÇBİYSBP için algoritmalara göre F istatistiki testinden elde edilen p değerleri

Problem	Orijinal MBO & ÇS-MBO (p)	Orijinal MBO & PSO-ÇS-MBO (p)	ÇS-MBO & PSO-ÇS-MBO (p)	Problem	Orijinal MBO & ÇS-MBO (p)	Orijinal MBO & PSO-ÇS-MBO (p)	ÇS-MBO & PSO-ÇS-MBO (p)
50_2_a	**1,15E-05**	**1,46E-08**	**6,92E-03**	100_2_a	**5,17E-05**	**2,04E-05**	6,63E-01
50_2_b	**4,75E-04**	**1,33E-06**	**1,43E-02**	100_2_b	**3,64E-07**	**8,91E-08**	5,12E-01
50_2_c	**5,08E-06**	**4,57E-07**	2,70E-01	100_2_c	**5,58E-06**	**2,58E-07**	1,65E-01
50_3_a	**9,13E-04**	**2,42E-04**	5,31E-01	100_3_a	**9,13E-04**	**2,37E-04**	5,25E-01
50_3_b	**1,13E-03**	**7,55E-05**	2,14E-01	100_3_b	**1,16E-03**	**3,20E-04**	5,44E-01
50_3_c	**2,45E-04**	**1,72E-05**	2,25E-01	100_3_c	**7,33E-03**	**1,46E-03**	4,40E-01
50_5_a	**3,82E-02**	**1,31E-02**	5,86E-01	100_5_a	**4,72E-02**	**3,75E-02**	9,03E-01
50_5_b	1,03E-01	2,79E-01	5,42E-01	100_5_b	1,26E-01	4,26E-01	4,29E-01
50_5_c	**1,61E-02**	**1,91E-02**	9,32E-01	100_5_c	6,73E-02	**1,99E-02**	5,25E-01
50_10_a	7,34E-01	3,50E-01	5,43E-01	100_10_a	8,73E-02	1,22E-01	8,48E-01
50_10_b	7,92E-01	1,35E-01	2,07E-01	100_10_b	4,34E-01	1,66E-01	5,18E-01
50_10_c	3,15E-01	8,35E-02	4,20E-01	100_10_c	**4,06E-02**	1,72E-01	4,21E-01
50_15_a	**3,56E-03**	2,32E-01	**4,14E-02**	100_15_a	1,06E-01	3,17E-01	4,98E-01
50_15_b	6,48E-01	**2,71E-02**	**1,10E-02**	100_15_b	3,46E-01	3,75E-01	9,53E-01
50_15_c	8,25E-02	5,90E-01	2,06E-01	100_15_c	4,17E-01	3,25E-01	8,57E-01
50_20_a	1,09E-01	9,96E-02	9,59E-01	100_20_a	2,18E-01	6,25E-01	9,61E-02
50_20_b	**1,38E-02**	**9,89E-03**	8,67E-01	100_20_b	**8,66E-03**	1,91E-01	1,14E-01
50_20_c	2,46E-01	1,53E-01	7,64E-01	100_20_c	1,98E-01	**1,23E-02**	1,47E-01

Çizelge 4.16. ÇBİYSBP için algoritmalara göre F istatistiki testinden elde edilen p değerleri

Problem	Orijinal MBO & ÇS-MBO (p)	Orijinal MBO & PSO-ÇS-MBO (p)	ÇS-MBO & PSO-ÇS-MBO (p)	Problem	Orijinal MBO & ÇS-MBO (p)	Orijinal MBO & PSO-ÇS-MBO (p)	ÇS-MBO & PSO-ÇS-MBO (p)
200_2_a	**4,17E-05**	**2,20E-05**	7,63E-01	300_2_a	**9,31E-04**	**1,47E-03**	8,27E-01
200_2_b	**1,16E-07**	**6,54E-05**	**9,47E-03**	300_2_b	**5,74E-05**	**6,82E-05**	9,35E-01
200_2_c	**1,16E-02**	**6,04E-03**	7,48E-01	300_2_c	**1,67E-05**	**7,76E-04**	8,77E-02
200_3_a	**1,40E-02**	**2,20E-02**	8,18E-01	300_3_a	7,36E-02	**2,95E-02**	6,26E-01
200_3_b	**4,85E-03**	**6,05E-03**	9,14E-01	300_3_b	**3,92E-04**	**7,77E-04**	7,45E-01
200_3_c	**3,77E-05**	**8,81E-04**	1,53E-01	300_3_c	**7,78E-04**	**4,25E-02**	6,56E-02
200_5_a	**3,11E-04**	**2,61E-02**	**4,68E-02**	300_5_a	**4,97E-02**	1,65E-01	5,00E-01
200_5_b	**2,06E-04**	**4,05E-02**	**2,08E-02**	300_5_b	**2,73E-02**	**1,24E-02**	6,89E-01
200_5_c	**3,73E-02**	**2,34E-02**	8,09E-01	300_5_c	1,18E-01	**1,81E-02**	3,24E-01
200_10_a	**2,86E-02**	8,62E-02	5,55E-01	300_10_a	1,05E-01	5,35E-01	2,88E-01
200_10_b	**7,85E-04**	3,21E-01	**6,40E-03**	300_10_b	1,20E-01	9,04E-01	1,47E-01
200_10_c	1,54E-01	1,01E-01	8,01E-01	300_10_c	7,48E-01	9,50E-01	7,02E-01
200_15_a	1,23E-01	1,98E-01	7,72E-01	300_15_a	5,85E-01	8,29E-01	4,49E-01
200_15_b	5,08E-01	1,34E-01	3,73E-01	300_15_b	**1,90E-02**	1,86E-01	2,24E-01
200_15_c	**1,91E-02**	2,38E-01	1,75E-01	300_15_c	5,46E-01	2,67E-01	5,97E-01
200_20_a	9,57E-01	4,32E-01	4,02E-01	300_20_a	1,80E-01	1,18E-01	7,98E-01
200_20_b	6,86E-01	7,23E-01	4,53E-01	300_20_b	4,17E-01	**8,22E-03**	**4,14E-02**
200_20_c	9,48E-01	4,42E-01	4,06E-01	300_20_c	7,66E-01	8,77E-01	6,52E-01

Çizelge 4.17. ÇBİYSBP için algoritmalara göre F istatistiki testinden elde edilen p değerleri

Problem	Orijinal MBO & ÇS-MBO (p)	Orijinal MBO & PSO-ÇS-MBO (p)	ÇS-MBO & PSO-ÇS-MBO (p)	Problem	Orijinal MBO & ÇS-MBO (p)	Orijinal MBO & PSO-ÇS-MBO (p)	ÇS-MBO & PSO-ÇS-MBO (p)
400_2_a	**2,04E-05**	**1,27E-03**	**6,84E-02**	500_2_a	**5,92E-03**	**1,21E-02**	7,24E-01
400_2_b	**3,81E-04**	**8,75E-05**	4,90E-01	500_2_b	**8,39E-06**	**1,29E-02**	**3,24E-03**
400_2_c	**2,37E-06**	**1,01E-02**	**1,21E-03**	500_2_c	**3,68E-03**	**4,65E-03**	9,09E-01
400_3_a	**1,97E-04**	**2,70E-03**	2,26E-01	500_3_a	**3,22E-03**	**3,69E-04**	3,10E-01
400_3_b	**2,56E-04**	**2,33E-03**	3,03E-01	500_3_b	**6,49E-03**	**3,17E-02**	4,30E-01
400_3_c	**2,02E-03**	**1,41E-02**	3,50E-01	500_3_c	**3,33E-02**	**4,77E-02**	8,50E-01
400_5_a	2,28E-01	6,51E-01	4,37E-01	500_5_a	**1,93E-03**	**8,75E-03**	4,67E-01
400_5_b	5,23E-01	9,54E-01	4,87E-01	500_5_b	5,67E-01	4,02E-01	7,84E-01
400_5_c	**1,09E-02**	6,55E-01	**2,64E-02**	500_5_c	2,73E-01	9,32E-01	2,40E-01
400_10_a	5,40E-01	3,46E-01	7,32E-01	500_10_a	4,61E-01	3,78E-01	1,19E-01
400_10_b	1,71E-01	5,07E-01	4,55E-01	500_10_b	1,43E-01	5,31E-01	3,75E-01
400_10_c	6,87E-01	4,19E-01	6,79E-01	500_10_c	6,54E-02	4,26E-01	2,54E-01
400_15_a	7,87E-02	4,40E-01	2,85E-01	500_15_a	9,49E-01	6,46E-01	6,02E-01
400_15_b	2,68E-01	3,22E-01	8,99E-01	500_15_b	7,67E-01	**3,11E-02**	5,43E-02
400_15_c	1,67E-01	5,47E-01	4,13E-01	500_15_c	6,04E-01	9,82E-01	6,19E-01
400_20_a	7,27E-01	8,73E-01	8,49E-01	500_20_a	4,50E-01	4,35E-01	1,38E-01
400_20_b	9,15E-01	9,58E-01	8,73E-01	500_20_b	2,86E-01	6,75E-01	5,05E-01
400_20_c	2,94E-01	5,08E-01	6,85E-01	500_20_c	1,15E-01	1,28E-01	9,50E-01

Çizelge 4.18. Algoritmalara göre F istatistiki testinden elde edilen p değerleri

Problem	Orijinal MBO & ÇS-MBO (p)	Orijinal MBO & PSO-ÇS-MBO (p)	ÇS-MBO & PSO-ÇS-MBO (p)
1000_2_a	1,47E-01	9,05E-02	7,78E-01
1000_2_b	7,00E-02	**3,17E-03**	1,34E-01
1000_2_c	1,70E-01	**9,41E-03**	1,38E-01
1000_3_a	3,00E-01	1,07E-01	5,25E-01
1000_3_b	1,26E-01	4,43E-01	4,13E-01
1000_3_c	1,66E-01	8,58E-02	7,00E-01
1000_5_a	3,08E-01	1,74E-01	7,11E-01
1000_5_b	5,34E-01	2,62E-01	6,01E-01
1000_5_c	**8,55E-03**	**8,24E-03**	9,86E-01
1000_10_a	9,01E-01	1,47E-01	1,81E-01
1000_10_b	1,63E-01	2,42E-01	**1,79E-02**
1000_10_c	1,89E-01	6,84E-01	9,49E-02
1000_15_a	6,79E-01	1,05E-01	2,10E-01
1000_15_b	7,64E-01	5,69E-01	3,90E-01
1000_15_c	4,82E-01	6,75E-01	2,70E-01
1000_20_a	8,92E-01	1,21E-01	9,53E-02
1000_20_b	1,04E-01	1,83E-01	**7,43E-03**
1000_20_c	4,52E-01	**3,16E-03**	**1,48E-02**

F testinden elde edilen p değerlerine göre, iyileştirilmiş ÇS-MBO ve PSO-ÇS-MBO algoritmalarından elde edilen sonuçların orijinal MBO algoritmasından elde edilen sonuçlara göre anlamlı olup olmadığı incelenmiştir. %95 güven seviyesinde yapılan testlerde iyileştirilen yöntemlerin orijinale göre 50, 100, 200 ve 300 boyutlu problem setleri için yaklaşık olarak yarısında anlamlı bir farkın olduğu tespit edilmiştir. Problem boyutunun 400 ve 500 olduğu problem setlerinde bu oranın kısmen düştüğü gözlenmiştir. 400 ve 500 boyutlu problem seti için on sekiz problemin yedisinde ÇS-MBO ile orijinal MBO algoritmaları arasında anlamlı farkların olduğu görülmüştür. Benzer şekilde 400 ve 500 boyutlu problem seti için sırasıyla on sekiz problemin altısında ve sekizinde PSO-ÇS-MBO ile orijinal MBO arasında anlamlı farklar olduğu tespit edilmiştir. Problem boyutunun 1000 olduğu veri setinde ise sadece bir problemde ÇS-MBO ile orijinal MBO arasında fark oluşurken, aynı problem setinde dört problem için PSO-ÇS-MBO ile orijinal MBO arasında farkın olduğu görülmüştür. Ayrıca her problem seti için ÇS-MBO ile PSO-ÇS-MBO algoritmaları arasındaki farklar da incelenmiştir. 50 ve 200 boyutlu problem setlerinde dört problem ve 400 ve 1000 boyutlu problem setlerinde ise üç problem için aralarında anlamlı bir fark gözlenmiştir. 500 ve 300 boyutlu problem setlerinde bir problemde anlamlı bir farkın olduğu görülürken 100 boyutlu problem setinin hiç birinde anlamlı bir fark görülmemiştir. Her üç algoritmadan elde edilen deneysel sonuçlar ile yapılan istatistiki testler gösteriyor ki, ÇS-MBO ve PSO-ÇS-MBO algoritmaları orijinal MBO algoritmasına göre ÇBİYSBP için daha başarılı sonuçlar üretmiştir.

ÇBİYSBP'nin çözümünde kullanılan problemler, CPLEX yöntemi ile de çözülmüştür. CPLEX yönteminden elde edilen sonuçlar, bu problemde daha başarılı olan ÇS-MBO algoritmasından elde edilen en iyi sonuçlar ile kıyaslanmıştır. Çizelge 4.19 ve 4.20'de verilen sonuçlar için ÇS-MBO algoritması 900 saniye çalıştırılırken CPLEX 1800 saniye çalıştırılmıştır.

Çizelge 4.19. CPLEX ve ÇS-MBO algoritmalarından elde edilen sonuçların karşılaştırılması

Problem	CPLEX	ÇS-MBO	Problem	CPLEX	ÇS-MBO	Problem	CPLEX	ÇS-MBO	Problem	CPLEX	ÇS-MBO
50_2_a	0,82	**0,77**	100_2_a	0,59	**0,33**	200_2_a	0,48	**0,26**	300_2_a	1,00	**0,19**
50_2_b	0,36	**1,18**	100_2_b	**0,31**	0,41	200_2_b	0,82	**0,15**	300_2_b	1,06	**0,13**
50_2_c	0,29	**1,53**	100_2_c	0,36	**0,11**	200_2_c	0,64	**0,01**	300_2_c	0,49	**0,14**
50_3_a	**15,92**	40,95	100_3_a	15,08	**13,02**	200_3_a	20,14	**6,17**	300_3_a	13,36	**3,00**
50_3_b	14,17	**8,45**	100_3_b	26,62	**9,27**	200_3_b	34,71	**7,79**	300_3_b	15,40	**2,63**
50_3_c	**16,86**	37,70	100_3_c	15,25	**8,64**	200_3_c	16,75	**4,06**	300_3_c	22,11	**6,23**
50_5_a	**126,77**	152,86	100_5_a	216,27	**205,42**	200_5_a	290,32	**155,73**	300_5_a	235,99	**151,61**
50_5_b	**202,92**	346,28	100_5_b	259,93	**173,43**	200_5_b	177,13	**104,96**	300_5_b	198,83	**115,87**
50_5_c	**162,64**	263,50	100_5_c	231,02	**164,68**	200_5_c	176,92	**138,62**	300_5_c	201,08	**119,53**
50_10_a	**1563,68**	1716,53	100_10_a	**1413,18**	1487,14	200_10_a	1562,50	**1396,10**	300_10_a	1600,50	**1516,08**
50_10_b	**1726,9**	1779,23	100_10_b	1443,46	**1203,16**	200_10_b	1405,14	**1278,10**	300_10_b	1211,64	**952,01**
50_10_c	**1335,88**	1897,14	100_10_c	1449,05	**1007,10**	200_10_c	1594,06	**1490,46**	300_10_c	1439,70	**1099,55**
50_15_a	**3018,78**	4070,78	100_15_a	**3304,12**	3489,45	200_15_a	**3223,49**	3370,60	300_15_a	2833,67	**2563,75**
50_15_b	**3067,27**	**3067,27**	100_15_b	**3056,41**	3225,04	200_15_b	3271,08	**3125,57**	300_15_b	3066,96	**2854,33**
50_15_c	**3302,69**	3866,28	100_15_c	**3181,99**	3226,18	200_15_c	**2933,27**	3100,85	300_15_c	3219,41	**2493,52**
50_20_a	**5646,7**	**5646,70**	100_20_a	**4943,96**	5291,33	200_20_a	**4528,35**	4669,36	300_20_a	4520,48	**4266,82**
50_20_b	5557,62	**5327,76**	100_20_b	**4613,20**	5130,22	200_20_b	4953,88	**4424,27**	300_20_b	5377,25	**4826,28**
50_20_c	5169,57	**4412,49**	100_20_c	4350,88	**4286,00**	200_20_c	5073,88	**4405**,41	300_20_c	4629,25	**4349,06**

Çizelge 4.20. CPLEX ve ÇS-MBO algoritmalarından elde edilen sonuçların karşılaştırılması

Problem	CPLEX	ÇS-MBO	Problem	CPLEX	ÇS-MBO	Problem	CPLEX	ÇS-MBO
400_2_a	0,76	**0,05**	500_2_a	0,25	**0,26**	1000_2_a	0,36	**0,23**
400_2_b	0,92	**0,10**	500_2_b	0,37	**0,11**	1000_2_b	0,89	**0,08**
400_2_c	0,78	**0,27**	500_2_c	0,51	**0,05**	1000_2_c	0,82	0,63
400_3_a	26,29	**4,01**	500_3_a	16,52	**5,03**	1000_3_a	9,06	**3,44**
400_3_b	14,38	**5,91**	500_3_b	34,47	**4,36**	1000_3_b	18,07	**5,62**
400_3_c	11,92	**4,76**	500_3_c	16,61	**5,01**	1000_3_c	17,18	7,53
400_5_a	200,65	**111,21**	500_5_a	211,98	**88,43**	1000_5_a	256,46	**123,78**
400_5_b	202,98	**149,55**	500_5_b	171,98	**105,05**	1000_5_b	237,90	**133,67**
400_5_c	**177,16**	177,37	500_5_c	370,07	**127,59**	1000_5_c	298,45	**212,80**
400_10_a	1525,54	**1270,18**	500_10_a	1465,15	**956,92**	1000_10_a	1596,56	**1505,39**
400_10_b	1544,82	**1242,11**	500_10_b	**1326,77**	1468,97	1000_10_b	1665,83	**1308,36**
400_10_c	1475,29	**1163,41**	500_10_c	**1200,92**	1579,45	1000_10_c	1618,68	**1401,82**
400_15_a	**3078,73**	3202,18	500_15_a	3358,48	**2657,45**	1000_15_a	3710,32	3563,96
400_15_b	**2976,72**	3113,68	500_15_b	3230,87	**2922,33**	1000_15_b	**2711,33**	3463,57
400_15_c	**3013,11**	3092,12	500_15_c	**2663,84**	2782,07	1000_15_c	3318,91	**3305,72**
400_20_a	5064,95	**4985,47**	500_20_a	5071,81	**4743,35**	1000_20_a	5832,68	**5114,24**
400_20_b	4982,15	**4492,01**	500_20_b	4969,66	**4750,79**	1000_20_b	**4601,66**	6093,98
400_20_c	4822,46	**4559,85**	500_20_c	4948,70	**4877,73**	1000_20_c	5538,17	**5136,73**

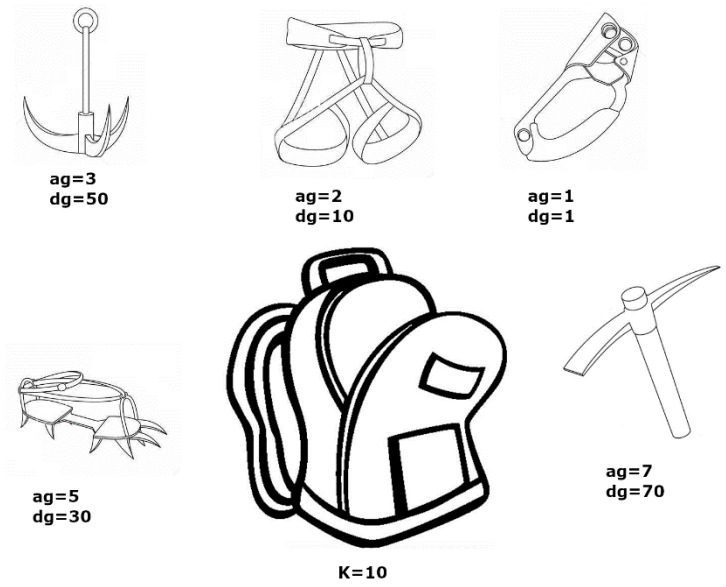
Çizelge 4.19'da 50, 100, 200 ve 300 boyutlu problemler için CPLEX ile ÇS-MBO algoritmaları kıyaslanmıştır. Elde edilen sonuçlara göre problemin karmaşıklığı arttıkça ÇS-MBO algoritmasının başarısı da artmaktadır. 50 boyutlu problemler için on problemde CPLEX, ÇS-MBO'ya göre daha iyi sonuçlar üretmiştir. Bu veri seti için iki problemde ise elde edilen sonuçlar aynıdır. 100 boyutlu problemlere geçildiğinde durumun tersine döndüğü görülmektedir. ÇS-MBO algoritması on sekiz problemin on birinde CPLEX'ten daha iyi sonuçlar üretmiştir. 200 boyutlu problemlerin kıyaslamasına bakıldığında CPLEX'in sadece üç problemde ÇS-MBO'ya göre daha iyi olduğu görülmektedir. 300 boyutlu problemlerde ise ÇS-MBO'nun tüm problemlerde CPLEX'i geride bıraktığı görülmüştür. Çizelge 4.20'de ise CPLEX, 400, 500 ve 1000 boyutlu problemlerde sırasıyla sadece dört, üç ve iki problemde ÇS-MBO'ya göre daha başarılı olmuştur. Geriye kalan tüm problemlerde ÇS-MBO'nun CPLEX'e göre daha iyi sonuçlar verdiği görülmüştür. İyileştirilen yöntemlerden ÇS-MBO algoritmasının özellikle büyük arama uzaylarında CPLEX yöntemine göre daha başarılı olduğu sonucuna varılmıştır.

4.3. İSÇP'nin BMBO ve ÇS-BMBO ile Çözümü

İkili sırt çantası problemi, bir dağcının tırmanışa geçmeden önce taşıyabileceği kadar gerekli malzemeyi bir çantanın içerisine yerleştirmesi olarak düşünülebilir. Dağcı için çok gerekli olan malzemelerin yanı sıra tırmanma esnasında malzemelerini yerleştirdiği çantayı da taşıyabilmesi önemlidir. Böyle bir durumda dağcı çantasını en verimli bir şekilde doldurabilmesi için hem çok gerekli hem de hafif olan malzemelerinden seçim yapmalıdır. Yerleştireceği malzemelerden sadece bir tane olduğu düşünülürse, çantaya yerleşen bir malzemeden ikinci bir tanesinin çantaya yerleşmesi mümkün olmayacaktır. Bu türden olan sırt çantası problemine ikili sırt çantası problemi de denmektedir.

Bu problemi çözerken bireylerin çözüm vektöründeki her bir hücre içerisine sadece "1" veya "0" değerlerinden bir tanesini yazılır. Vektörün boyu sırt çantasına yerleştirilmesi planlanan nesne sayısı kadar olmalıdır. Ayrıca her bir nesne bir ağırlığa ve fayda değerine sahiptir. Buna karşın nesnelerin yerleştirileceği çantanın da bir taşıma kapasitesi vardır.

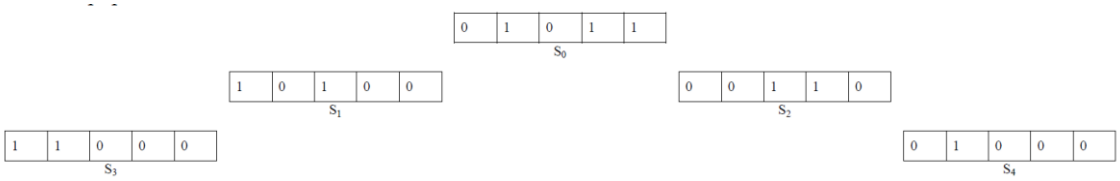
Şekil 4.12'de basit bir sırt çantası problemi örnek olarak verilmiştir.



Şekil 4.12. Sırt çantasının kapasitesi ve yerleştirilecek olan nesnelere ait ağırlıklar ve değerleri

Şekil 4.12'de taşıma kapasitesi 10 birim olan bir sırt çantası ile çantaya yerleştirilmesi planlanan beş nesnenin ağırlıkları (*ag*) ve değerleri (*dg*) görülmektedir. Buna göre ağırlıklar kümesi *ag = {2, 5, 7, 3, 1}* ve değerler kümesi *dg = {10, 30, 70, 50, 1}* olarak oluşturulabilir. Verilen örnekteki problemin boyutu 5, çantanın kapasitesi (*K*) 10 olarak tanımlıdır. Bu problemi çözmek için *ag* ve *dg* kümelerindeki eleman sayısı kadar bir çözüm vektörü oluşturulur. Oluşturulan vektörün indis numaraları *ag* veya *dg* kümesindeki elemanların sırası ile ilişkilendirilir. Yani ağırlığı 2 ve değeri 10 olan nesne vektörün ilk indisi ile temsil edilirken ağırlığı ve değeri 1 olan nesne vektörün son indisi olarak temsil edilir.

Başlangıç popülasyonunda çantaya yerleştirilecek nesne rastgele seçilir. Seçilen bu nesne çanta kapasitesini aşmıyorsa seçim işlemi onaylanır ve çözüm vektörünün ilgili indisi "1" değeri ile doldurulur. Ardından bir başka nesne daha seçilir. Seçilen bu nesnenin ağırlığı çanta içindeki daha önce seçilen diğer nesnelerin toplam ağırlığına dahil edilir. Ardından çanta kapasitesinin aşılma durumu kontrol edilir. Taşma gerçekleşmiyorsa seçim onaylanır. Nesnelerin seçim işlemi çantanın kapasitesi dolana kadar devam ettirilir. Seçilen nesneler çözüm vektöründe "1" değeri ile temsil edilirken seçilmemiş nesneler "0" değeri ile temsil edilir. Şekil 4.13'te beş bireyden oluşan bir popülasyonun çözüm vektörleri gösterilmektedir.



Şekil 4.13. Rastgele oluşturulmuş popülasyondaki çözüm vektörleri

Şekil 4.13'te "V" formasyonunda yerleştirilmiş popülasyon ve bireylerin çözüm vektörleri verilmiştir. Buna göre S_0 bireyi için sırt çantasına üç nesne yerleştirilmiştir. Bu nesnelerin ağırlıklarının toplamı (5 + 3 + 1) = 9 birim olarak hesaplanmıştır ki bu değer, kapasitesi 10 birim olan çantanın kapasitesinden daha azdır. Yine aynı birey için çantaya yerleştirilmiş olan nesnelerin değerlerinin toplamı, yani bireyin uygunluk değeri Denklem 3.6'ya göre (30 + 50 + 1) = 81 olarak hesaplanır. Bu şekilde tüm bireylerin uygunlukları hesaplanır. Ardından mevcut çözümlerin geliştirilmesi için komşu çözümler üretilir ve üretilen bu çözümlerden bazıları diğer çözümlerle paylaştırılır. Çözüm vektörü ikili kodlanmış algoritmada; komşu çözüm üretme işlemi, mevcut çözüm vektörü içerisindeki "0" değerli hücrelerin değerlerinin "1" yapılmasıyla elde edilir. Yani çantaya yeni bir nesne eklenir. Çantaya yeni eklenen nesnenin ağırlığı mevcut nesnelerin ağırlık toplamına eklenir. Bu durumda sırt çantasının kapasite aşımı gerçekleşmiş olabilir. Eğer böyle bir durum varsa son eklenen nesne haricinde rastgele belirlenen bir başka nesne sırt çantasından çıkartılır. Yani çıkartılacak olan nesnenin çözüm vektöründeki "1" olan değeri "0" ile değiştirilir. Eğer nesnelerin toplam ağırlıkları çanta kapasitesini hala aşıyorsa çantadan nesne çıkartma işlemine devam edilir. Elde edilen yeni bireyin uygunluğu yine Denklem 3.6'ya göre hesaplanır. Verilen örneğe ait başlangıç popülasyonu ve komşu paylaşımını içeren özet bir durum Çizelge 4.21'de verilmiştir.

Çizelge 4.21'de üretilen komşu çözümler (Km) uygunluk değerlerine göre en iyiden en kötüye doğru sıralanır. En iyi komşu çözüm mevcut çözümü geliştirmek için kullanılırken diğer komşu çözümler mevcut çözümü takip eden diğer çözümler ile paylaştırılır.

Çizelge 4.21. k = 3 ve x = 1 için her bireyin komşu çözüm üretimi ve paylaşımı

Birey	Permütasyon	Uygunluk	Komşu çözüm	Permütasyon	Uygunluk
S_0	01011	81	Km_{01}	00110	80
			Km_{02}	00101	71
			Km_{03}	11001	41
S_1	10100	80	Km_{11}	10101	81
			Km_{12}	00110	80
			Km_{13} (Paylaştırılmış=Km_{02})	00101	71
S_2	00110	120	Km_{21}	10010	60
			Km_{22}	00011	51
			Km_{23} (Paylaştırılmış=Km_{03})	11001	41
S_3	11000	40	Km_{31} (Paylaştırılmış=Km_{12})	00110	80
			Km_{32}	00010	50
			Km_{33}	11001	41
S_4	01000	30	Km_{41}	01010	80
			Km_{42} (Paylaştırılmış=Km_{22})	00011	51
			Km_{43}	01001	31

4.3.1. İSÇP için deneysel çalışmalar

İSÇP için literatürden seçilen otuz bir test problemi (Ortega) BMBO ve ÇS-BMBO algoritmaları ile çözülmüştür. PSO-ÇS-MBO algoritmasının yapısı ikili problemlere uygun olmadığı için bu tür problemlerin çözümünde kullanılmamıştır. Kullanılan otuz bir test problemi dört farklı gruba ayrılmıştır. Düşük boyutlu İSÇP problemlerinden on, geniş boyutlu İSÇP problemlerinden ise yirmi bir test verisi vardır. Problemlerin tamamı her algoritma için bağımsız olarak otuz kez çalıştırılmış ve elde edilen en iyi, ortalama ve en kötü sonuçlar kaydedilmiştir. Ele alınan test problemlerinin dosya formatları aşağıdaki gibidir.

Dosya ismi: kp_n_wmax

kp: problemin adı

n: Sırt çantasına yerleştirilecek nesne sayısı

wmax: sırt çantasının kapasitesi

Veri seti içeren dosya içeriğinin bir bölümü Şekil 4.14'te gösterilmiştir.

```
94 485
506 326
416 248
992 421
649 322
237 795
457 43
815 845
446 955
422 252
791 9
359 901
667 122
598 94
```

Şekil 4.14. İSÇP için örnek veri seti

Şekil 4.14'teki veri setine göre her satır iki sayıdan oluşmaktadır. Bu satırlardaki ilk sütun, nesnelerin değerini (*dg*) ikinci sütun ise aynı nesnelerin ağırlığını (*ag*) temsil etmektedir. Veri dosyası içerisinde sırt çantasına yerleştirilecek nesne sayısı kadar satır bulunmaktadır.

İSÇP'nin çözümü için kullanılan algoritmaların parametreleri Çizelge 4.22'de verilmiştir.

Çizelge 4.22. İSÇP'nin çözümünde kullanılan parametreler

Parametreler	Değerleri
Kuş sayısı (*p*)	71
Komşu sayısı (*k*)	5
Kanat çırpma sayısı (*m*)	30
Komşu paylaşım sayısı (*x*)	1
Sürü sayısı (*z*)	30

Her iki algoritma için İSÇP'nin çözümünden elde edilen sonuçlar Çizelge 4.23'te verilmiştir.

Çizelge 4.23. Literatürden seçilen 31 İSÇP için BMBO ve ÇS-BMBO algoritmalarından elde edilen sonuçlar

P.No	Problem	Opt.	BMBO			ÇS-BMBO		
			En iyi	Ortalama	En kötü	En iyi	Ortalama	En kötü
1	f1_l-d_kp_10_269	295	**295**	**295,00**	**295**	**295**	**295,00**	**295**
2	f2_l-d_kp_20_878	1024	**1024**	**1024,00**	**1024**	**1024**	**1024,00**	**1024**
3	f3_l-d_kp_4_20	35	**35**	**35,00**	**35**	**35**	**35,00**	**35**
4	f4_l-d_kp_4_11	23	**23**	**23,00**	**23**	**23**	**23,00**	**23**
5	f5_l-d_kp_15_375	481,06	**481,06**	**481,06**	**481,06**	**481,06**	**481,06**	**481,06**
6	f6_l-d_kp_10_60	52	**52**	**52,00**	**52**	**52**	**52,00**	**52**
7	f7_l-d_kp_7_50	107	**107**	**107,00**	**107**	**107**	**107,00**	**107**
8	f8_l-d_kp_23_10000	9767	**9767**	**9767,00**	**9767**	**9767**	**9767,00**	**9767**
9	f9_l-d_kp_5_80	130	**130**	**130,00**	**130**	**130**	**130,00**	**130**
10	f10_l-d_kp_20_879	1025	**1025**	**1025,00**	**1025**	**1025**	**1025,00**	**1025**
11	knapPI_1_100_1000_1	9147	**9147**	**9147,00**	**9147**	**9147**	**9147,00**	**9147**
12	knapPI_1_200_1000_1	11238	**11238**	**11238,00**	**11238**	**11238**	**11238,00**	**11238**
13	knapPI_1_500_1000_1	28857	**28857**	28818,06	28691	**28857**	**28857,00**	28857
14	knapPI_1_1000_1000_1	54503	54105	53856,13	53568	**54284**	**54126,53**	54032
15	knapPI_1_2000_1000_1	110625	108750	108260,60	108007	**108945**	**108809,46**	108634
16	knapPI_1_5000_1000_1	276457	261031	259686,93	257778	**263028**	**261979,66**	261118
17	knapPI_1_10000_1000_1	563647	502409	498882,60	494487	**519962**	**510033,33**	504663
18	knapPI_2_100_1000_1	1514	**1514**	1497,00	1442	**1514**	**1514,00**	**1514**
19	knapPI_2_200_1000_1	1634	1610	1560,66	1483	**1634**	**1622,20**	1610
20	knapPI_2_500_1000_1	4566	4352	4189,06	4038	**4448**	**4381,00**	4274
21	knapPI_2_1000_1000_1	9052	8298	8079,66	7644	**8576**	**8486,66**	8410
22	knapPI_2_2000_1000_1	18051	15754	15320,60	14985	**16284**	**15997,86**	15780
23	knapPI_2_5000_1000_1	44356	**37316**	35691,86	34678	36647	**36419,00**	36175
24	knapPI_2_10000_1000_1	90204	70231	69365,73	68552	**71451**	**70512,86**	70132
25	knapPI_3_100_1000_1	2397	**2397**	2336,86	2297	**2397**	**2397,00**	**2397**
26	knapPI_3_200_1000_1	2697	**2697**	2636,53	2597	**2697**	**2697,00**	**2697**
27	knapPI_3_500_1000_1	7117	6717	6683,06	6611	**6917**	**6850,00**	6817
28	knapPI_3_1000_1000_1	14390	13490	13283,26	13090	**13690**	**13496,40**	13390
29	knapPI_3_2000_1000_1	28919	26219	25681,26	25216	**26519**	**26324,33**	26117
30	knapPI_3_5000_1000_1	72505	61005	60282,73	59600	**62401**	**61437,06**	61004
31	knapPI_3_10000_1000_1	146919	115619	113075,40	111518	**116525**	**115296,60**	114314

Çizelge 4.23'te İSÇP'nin BMBO ve ÇS-BMBO algoritmaları ile çözümünden elde edilen sonuçlar listelenmiştir. İlk on problem düşük boyutlu problemlerden oluşurken on birinci problem ile on yedinci problemler arasındakiler büyük ölçekli ve ilişkili olmayan veri setlerini içermektedir. Yani *dg* ve *ag* [1,R] arasında rastgele seçilmiştir. On sekizinci problem ile yirmi dördüncü problemler arasında kalanlar yine büyük ölçekli zayıf ilişkili örnekleri içermektedir. Zayıf ilişki $ag \epsilon [1,R]$ ve $dg \epsilon [ag\text{-}10/R, ag+R/10]$ olarak tanımlanmıştır. Son olarak yirmi beşinci problem ile otuz birinci problemler arasında kalan örnekler büyük ölçekli ve güçlü ilişkili olan örneklerden oluşmaktadır. Güçlü ilişkide *ag* ve *dg* değerleri $ag \epsilon [1,R]$ ve $dg = ag+R/10$ olarak tanımlanmıştır. Buradaki R, [1000,10000] aralığında seçilmiş rastgele bir değerdir.

İlk grup problemde BMBO ve ÇS-BMBO algoritmalarının her ikisinden de çok başarılı sonuçlar elde edilmiştir. Öyle ki, bağımsız olarak gerçekleştirilen otuz testin tamamında algoritmalar bilinen en iyi değere (optimum) ulaşmışlardır. Geniş ölçekli problem örneklerinden 10, 11 ve 12. problemlerde de her iki algoritma yapılan tüm testlerde optimum değere ulaşabilmiştir. 13. problemde, BMBO yine optimum değere ulaşmıştır ancak tüm testlerde aynı başarıyı yakalayamamıştır. Buna karşın ÇS-BMBO algoritması bu problem için yine tüm testlerde optimum sonuca ulaşabilmiştir. 14 ve 17. problemlerde her iki algoritma da optimum değere hiçbir zaman ulaşamamışlardır. Ancak ÇS-BMBO algoritması BMBO algoritmasından daha iyi sonuçlar vermiştir. 18. problemde her iki algoritma da optimum değere ulaşabilmişlerdir. Buna ilaveten, ÇS-BMBO algoritmasının tüm testlerde optimum değere ulaştığı görülmüştür. 19. problemde sadece ÇS-BMBO algoritması optimum değere ulaşmıştır. 20, 21, 22. ve 24. problemlerde her iki algoritma da optimum değeri görememiştir. Buna karşılık, ÇS-BMBO algoritması hem en iyi değere göre hem de ortalama değerlere göre BMBO algoritmasından daha iyi sonuçlar üretmiştir. 23. problemde BMBO algoritması en iyi değere göre kıyaslandığında ÇS-BMBO algoritmasından daha iyi sonuç üretmiştir. Ancak bu başarısı ortalama değerlere yansımamıştır. 25 ve 26. problemlerde de her iki algoritma optimum değeri bulabilmişlerdir ve ÇS-BMBO algoritması tüm testlerde bu başarıyı yakalayabilmiştir. 27, 28, 29, 30 ve 31. problemlerde optimum değer her iki algoritma tarafından bulunamamıştır. Ancak ÇS-BMBO algoritmasının BMBO algoritmasına göre daha başarılı olduğu en iyi değer ve ortalama değerlere bakılarak anlaşılabilmektedir.

Çizelge 4.23'teki sonuçlara bakıldığında, düşük boyutlu problemlerde her iki algoritmanın da başarılı sonuçlar üretebildiği, problem boyutunun artmasıyla ÇS-BMBO algoritmasının BMBO algoritmasına göre daha çok öne çıktığı görülmektedir.

BMBO ve ÇS-BMBO algoritmalarından elde edilen sonuçlar arasında anlamlı bir farkın olup olmadığı F istatistiksel testi ile araştırılmıştır. Çizelge 4.24'te BMBO ile ÇS-BMBO algoritmaları için F testinden elde edilen p değerleri hesaplanmıştır. Ancak 1-12 nolu problemlerde her iki algoritma da tüm deneysel testlerde aynı sonucu buldukları için F testi bu problemlere uygulanmamıştır. Benzer şekilde 18, 25 ve 26 nolu problemlerde ÇS-BMBO algoritması tüm testlerde aynı sonucu verdiğinden bu problemler için de istatistiksel test uygulanmamıştır.

Çizelge 4.24. İSÇP için BMBO ve ÇS-MBO algoritmalarından elde edilen sonuçlara göre F istatistiki testinden elde edilen p değerleri

Problem	BMBO & ÇS-BMBO (p)
knapPI_1_500_1000_1	**3,92E-19**
knapPI_1_1000_1000_1	1,36E-01
knapPI_1_2000_1000_1	**4,72E-03**
knapPI_1_5000_1000_1	**1,96E-02**
knapPI_1_10000_1000_1	**3,99E-02**
knapPI_2_200_1000_1	**5,06E-08**
knapPI_2_500_1000_1	**1,53E-03**
knapPI_2_1000_1000_1	**2,33E-04**
knapPI_2_2000_1000_1	2,94E-01
knapPI_2_5000_1000_1	**2,62E-07**
knapPI_2_10000_1000_1	5,09E-01
knapPI_3_500_1000_1	9,34E-01
knapPI_3_1000_1000_1	7,05E-02
knapPI_3_2000_1000_1	**5,50E-03**
knapPI_3_5000_1000_1	1,12E-01
knapPI_3_10000_1000_1	**1,85E-02**

Çizelge 4.24'te p değeri hesaplanan on altı problemin on tanesinde ÇS-BMBO algoritmasının BMBO algoritmasına göre anlamlı derecede farkın olduğu görülmüştür. Bu sonuç, çok sürülü yapının ikili problemlerin çözümünde de daha başarılı olduğu göstermiştir.

ÇS-BMBO algoritmasından elde edilen sonuçlar literatürden seçilen diğer yöntemler ile de kıyaslanmıştır. Ancak incelenen birçok makalede standart sırt çantası problemi olarak Çizelge 4.23'teki ilk on problem kullanılmıştır. Diğer örneklerin ise rastgele oluşturulduğu görülmüştür. Bu sebeple ÇS-BMBO ile diğer yöntemlerin kıyaslaması sadece ilk on problem üzerinden yapılmıştır. Çizelge 4.25'te ikili guguk kuşu arama algoritması (BCS), harmony arama algoritması (NGHS) (Gherboudj ve ark., 2012)

ve ikili kelebek optimizasyon algoritması (BBO) (Feng ve ark., 2017), ÇS-BMBO algoritması ile karşılaştırılmıştır.

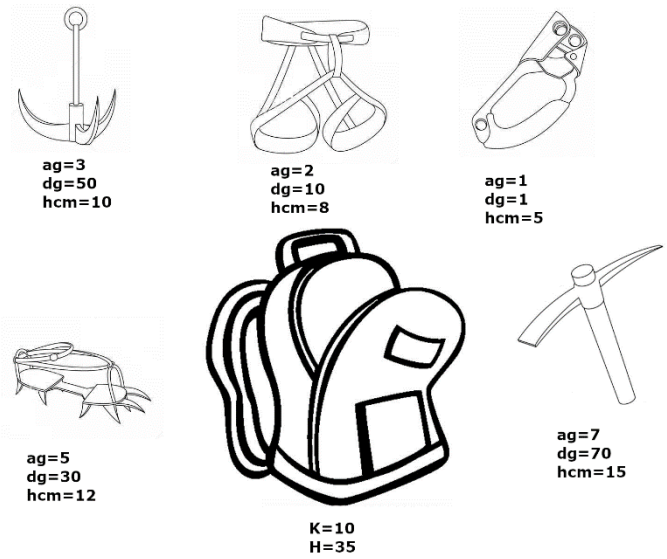
Çizelge 4.25. ÇS-BMBO algoritmasının literatürden seçilen diğer algoritmalar ile karşılaştırılması

Problem	BCS	NGHS	BBO	ÇS-BMBO
f1_l-d_kp_10_269	**295**	**295**	**295**	**295**
f2_l-d_kp_20_878	**1024**	**1024**	**1024**	**1024**
f3_l-d_kp_4_20	**35**	**35**	**35**	**35**
f4_l-d_kp_4_11	**23**	**23**	**23**	**23**
f5_l-d_kp_15_375	**481,06**	**481,06**	**481,06**	**481,06**
f6_l-d_kp_10_60	**52**	50	**52**	**52**
f7_l-d_kp_7_50	**107**	**107**	**107**	**107**
f8_l-d_kp_23_10000	**9767**	9761	**9767**	**9767**
f9_l-d_kp_5_80	**130**	**130**	**130**	**130**
f10_l-d_kp_20_879	**1025**	**1025**	**1025**	**1025**

Çizelge 4.25'te bulunan en iyi sonuçlar koyu olarak işaretlenmiştir. NGHS algoritması sadece iki problemde optimum değere ulaşamazken diğer yöntemlerin tamamı tüm problemlerde en iyi sonucu bulabilmişlerdir. Bu sonuçlar, BMBO ve ÇS-BMBO algoritmalarının İSÇP'nin çözümünde literatürdeki diğer yöntemlere alternatif bir algoritma olduğunu göstermektedir.

4.4. ÇBİSÇP'nin BMBO ve ÇS-BMBO ile Çözümü

Çok boyutlu ikili sırt çantası problemi çanta kısıtlarının birden fazla belirtilmesi olarak tarif edilebilir. Bir önceki başlıkta anlatılan İSÇP örneğinde bir dağcının tırmanışa geçmeden önce gerekli olan malzemelerini bir çanta içerisine yerleştirirken çantanın taşıyabileceği toplam ağırlık dikkate alınarak kısıt belirlenmişti. Eğer dağcı taşıyacağı eşyaları çanta içerisine yerleştirirken hem ağırlığı hem de eşyaların hacimleri dikkate alınacak olursa eşyaların yerleşiminde iki kısıt karşımıza çıkmaktadır. Yani çantanın taşıyabileceği maksimum ağırlık birinci kısıt, yine aynı çantanın alabileceği maksimum hacim ise ikinci kısıt olmaktadır. Bu türdeki sırt çantası problemleri çok boyutlu ikili sırt çantası problemi olarak isimlendirilir. Şekil 4.12'de verilen örnek, çok boyutlu sırt çantası problemi için Şekil 4.15'teki gibi gösterilebilir.



Şekil 4.15. Sırt çantasının kapasiteleri ve yerleştirilecek olan nesnelere ait ağırlıklar, hacimler ve değerleri

Önceki verilen örnekten farklı olarak her nesneye kapalı alanda kapladığı hacim değerleri verilmiştir. Bunun yanında nesnelerin yerleştirileceği çanta için ağırlık kısıtına ek olarak hacim kısıtı da getirilmiştir. Şekil 4.15'te "hcm" nesnenin hacmini gösterirken sırt çantasındaki "H" ise çantanın hacim kapasitesini göstermektedir.

ÇBİSÇP çözülürken tek boyutlu sırt çantası probleminin çözümünde kullanılan teknikler kullanılmıştır. Yani çözüm vektörleri içeresinde nesnelerin çantaya yerleştirildiği durumlar "1" yerleştirilmediği durumlar "0" olarak temsil edilmiştir. Ancak nesnenin yerleştirilme aşamasında birden çok olan kısıtlar tek tek kontrol edilmiştir. Başka bir ifadeyle, herhangi bir nesne çantaya yerleştiğinde bu nesnenin hem ağırlığı hem de hacmi çantanın kapasitesini işgal etmektedir. Yerleştirilen nesnenin hacmi küçük olup çantanın hacim kapasitesi sınırını aşmasa dahi nesnenin ağırlığı çok fazla olup çantanın ağırlık kapasite sınırını aşabilir. Böyle bir durumda bu nesne çanta içerisine yerleştirilemez. Kısacası nesnelerin çanta içerisine yerleştirilebilmesi için yerleştirilmek istenen nesnenin çantanın tüm kısıtları için sınırlar dışına taşmaması gerekir.

ÇBİSÇP'nin çözümünde bireylerin çözüm permütasyonlarının temsili Şekil 4.13'teki gibidir. Çözüm vektörünün büyüklüğü yine yerleştirilecek nesne sayısı kadardır. Çantaya yerleştirilen nesne, çözüm permütasyonunda "1" ile temsil edilirken çanta dışında kalan nesneler "0" olarak temsil edilir. Başlangıç popülasyonu yine İSÇP'de olduğu gibi rastgele oluşturulur. Rastgele oluşturulan bireylerin kapasite sınırlarını taşma durumları kontrol edilir.

Şekil 4.13'teki S_0 bireyinin çözüm permütasyonu dikkate alınırsa Denklem 3.8'e göre S_0 bireyinin uygunluk değeri (30 + 50 + 1) = 81 olarak hesaplanır. Bu bireyin toplam ağırlığı Denklem 3.9'a göre (5 + 3 + 1) = 9 ve toplam hacmi yine Denklem 3.9'a göre (12 + 10 + 5) = 27 olarak hesaplanır. Nesnelerin toplam ağırlıkları ve toplam hacimleri sırt çantasının ağırlık kapasitesi olan 10 ve hacim kapasitesi olan 35 sınır değerlerinden daha küçük olduğundan bu çözüm permütasyonu kabul edilebilir.

ÇBİSÇP'de komşu çözüm üretimi yine aynı şekilde İSÇP'de olduğu gibi mevcut bir bireyden üretilir. Mevcut çözüm permütasyonundan rastgele seçilen bir nesne (çözüm vektöründe "0" değer içeren hücre) çanta içerisine yerleştirilir. Rastgele seçilen bu nesne çanta içerisine yerleştirildikten sonra çanta içerisindeki tüm nesnelere ait toplam ağırlık ve hacim bilgileri güncellenir. Eğer çantanın kapasite sınırlarının herhangi birinde bir taşma olmuşsa, son eklenen nesne dışında herhangi bir nesne çantadan çıkartılır (çözüm permütasyonundaki "1" değeri "0" ile değiştirilir) ve toplam ağırlıklar ve hacim tekrar güncellenir. Eğer çantanın kapasite sınırları hala aşılıyorsa çantadan rastgele nesne çıkartma işlemine devam edilir. Bu şekilde yeni bir komşu çözüm elde edilmiş olur. Çözüm paylaşımı ve lider değişimi orijinal MBO algoritmasında olduğu gibidir.

4.4.1. ÇBİSÇP için deneysel çalışmalar

BMBO ve ÇS-BMBO algoritmalarının ÇBİSÇP üzerindeki performansı OR-Library kütüphanesinden seçilen otuz test problemi üzerinde incelenmiştir (Chu ve Beasley, 1998). "mknap1.txt" isimli dosya içerisinde birbirinden bağımsız yedi çok boyutlu sırt çantası problemi mevcuttur. Dosya içeriğine ait küçük bir örnek Şekil 4.16'da verilmiştir.

```
6 10 3800
100 600 1200 2400 500 2000
8 12 13 64 22 41
8 12 13 75 22 41
3 6 4 18 6 4
5 10 8 32 6 12
5 13 8 42 6 20
5 13 8 48 6 20
0 0 0 0 8 0
3 0 4 0 8 0
3 2 4 0 8 4
3 2 4 8 8 4
80 96 20 36 44 48 10 18 22 24
```

Şekil 4.16. ÇBİSÇP örnek veri seti

Şekil 4.16'daki dosya içeriğine göre ilk satırdaki ilk değer (6) altı adet nesnenin olduğunu ifade etmektedir. Hemen yanındaki değer (10) ise her bir nesnenin sahip olduğu kısıt sayısını ifade etmektedir. İlk satırdaki son değer (3800) bilinen en iyi sonucu ifade etmektedir. İkinci satırdaki sayılar sırasıyla nesnelerin değeler kümesini (dg) oluşturur. Üçüncü satırdan başlayıp on ikinci satıra kadar olan matris şeklindeki değerler topluluğu ise her bir nesnenin ilgili kısıtını göstermektedir. Satırlar her bir kısıtı ifade ederken sütunlar ise ilgili nesnenin o kısıt için değerini ifade etmektedir. Yani birinci kısıt için nesnelerin kısıtlar kümesi sırasıyla {8, 12, 13, 64, 22 ve 41} değerlerinden oluşur. En alt satır ise çantanın her bir kısıt için tanımlanmış kapasitelerini göstermektedir.

İlk üç problem "mknap1.txt" dosyasından seçilmiştir. Diğer problemler ise "mknapcb1", "mknapcb2", ...,"mknapcb9" dosyalarından seçilen problemlerdir. Seçilen bu otuz test problemi BMBO ve ÇS-BMBO algoritmaları ile bağımsız olarak otuz defa çalıştırılmıştır. Algoritmalardan elde edilen en iyi, ortalama ve en kötü sonuçlar Çizelge 4.27'de verilmiştir.

İSÇP'nin çözümü için kullanılan algoritmaların parametreleri Çizelge 4.26'da verilmiştir.

Çizelge 4.26. ÇBİSÇP'nin çözümünde kullanılan parametreler

Parametreler	Değerleri
Kuş sayısı (p)	51
Komşu sayısı (k)	3
Kanat çırpma sayısı (m)	10
Komşu paylaşım sayısı (x)	1
Sürü sayısı (z)	30

Çizelge 4.27. Literatürden seçilen 30 ÇBİSÇP için BMBO ve ÇS-BMBO algoritmalarından elde edilen sonuçlar

Problem No	Nesne Sayısı	Kısıt Sayısı	Bilinen en iyi sonuç	BMBO			ÇS-MBO		
				En iyi	Ortalama	En kötü	En iyi	Ortalama	En kötü
1	6	10	3800	**3800**	3800,00	**3800**	**3800**	**3800,00**	**3800**
2	39	5	10618	**10618**	10611,47	10604	**10618**	**10618,00**	**10618**
3	50	5	16537	**16537**	16534,93	16519	**16537**	**16537,00**	**16537**
4	100	5	24585	24168	23918,13	23761	**24281**	**24102,27**	24005
5	100	5	24538	23993	23871,60	23735	**24244**	**24070,20**	23937
6	100	5	23895	23494	23279,33	23093	**23520**	**23417,33**	23322
7	250	5	59442	58212	57864,53	57564	**58542**	**58253,07**	58028
8	250	5	61629	60123	59627,20	59339	**60310**	**60189,47**	60076
9	250	5	62259	60598	60220,67	59810	**60934**	**60749,27**	60618
10	500	5	12023	116440	116033,33	115558	**117172**	**116741,20**	116522
11	500	5	11795	114258	113943,00	113528	**114913**	**114670,27**	114407
12	500	5	12121	117468	117143,33	116749	**117847**	**117711,53**	117526
13	100	10	23480	**23055**	22726,93	22542	**23055**	**22927,80**	22710
14	100	10	23220	22478	22285,27	22089	**22570**	**22487,40**	22415
15	100	10	22493	22081	21777,40	21623	**22131**	**21974,33**	21848
16	250	10	59489	58002	57800,00	57647	**58358**	**58060,40**	57698
17	250	10	59024	57327	57098,67	56827	**57643**	**57509,73**	57333
18	250	10	58413	57267	56466,47	56133	**57032**	**56828,93**	56681
19	500	10	11801	114098	113704,20	113146	**114599**	**114400,73**	114172
20	500	10	11943	115992	115520,87	114876	**116260**	**115982,40**	115798
21	500	10	11940	115717	115297,47	114938	**116703**	**116022,80**	115408
22	100	30	22579	21660	21598,87	21547	**21946**	**21709,33**	21543
23	100	30	22367	21439	21324,47	21183	**21575**	**21478,87**	21368
24	100	30	21270	20391	20284,53	20173	**20548**	**20468,87**	20365
25	250	30	57430	55616	55341,00	55188	**55762**	**55591,47**	55240
26	250	30	59080	57134	56793,73	56646	**57336**	**57204,80**	56857
27	250	30	57176	55784	55393,60	55107	**55991**	**55796,27**	55611
28	500	30	11661	113366	112740,60	112364	**113730**	**113167,20**	112669
29	500	30	11537	112341	111642,60	111203	**112382**	**112194,53**	111732
30	500	30	11734	113590	113156,93	112784	**114182**	**113876,33**	113575

Çizelge 4.27'de, literatürden seçilmiş otuz çok boyutlu sırt çantası problemi BMBO ve ÇS-BMBO algoritmaları ile çözülmüş ve elde edilen en iyi, ortalama ve en kötü sonuçlar listelenmiştir. Koyu olarak belirtilen değerler, ilgili problemin çözümünden elde edilen en iyi değerleri ifade etmektedir. Her iki algoritmadan elde edilen sonuçlara bakıldığında her iki algoritmanın da birbirine yakın sonuçlar ürettiği görülmektedir. Her iki algoritma da otuz problemin sadece 1, 2 ve 3. problemlerinde optimum sonuca ulaşmıştır. 1. problemde her iki algoritma da yapılan tüm testlerde optimum değere ulaşırken 2. ve 3. problemlerde BMBO bazı testlerde optimum değeri bulabilmiştir. Buna karşın ÇS-BMBO algoritması 2. ve 3. problemler için de tüm testlerde optimum değere ulaşabilmiştir. Buna ilaveten 13. problemde her iki algoritma da optimum sonuca ulaşamamasına rağmen aynı en iyi değeri bulmuşlardır. Ancak ortalama değerlerine bakıldığında burada da ÇS-BMBO'nun daha önde olduğu görülmektedir. Diğer problemlerin tamamında her iki algoritma da optimum değeri bulamamış ancak yakın sonuçlar üretmişlerdir. Ayrıca yine diğer tüm problemlerde ÇS-BMBO algoritması BMBO algoritmasından hem en iyi hem de ortalama değerlere göre daha iyi sonuçlar vermiştir.

ÇBİSÇP'nin çözümünde kullanılan BMBO ve ÇS-BMBO algoritmaları arasındaki farkların anlamlılığı istatistiksel F testi ile kontrol edilmiştir. Çizelge 4.27'deki 1, 2 ve 3 numaralı problemler için ÇS-BMBO algoritması, yapılan tüm testlerde aynı optimum sonucu verdiğinden dolayı bu problemler istatistiksel teste dahil edilmemiştir. F testinde kullanılan önem seviyesi %95 olarak alınmıştır. F testinden elde edilen p değerleri Çizelge 4.28'de verilmiştir. Hesaplanan p değerlerine göre ÇS-BMBO algoritması dokuz problemde anlamlı sayılacak şekilde BMBO algoritmasından daha iyi sonuçlar verdiği görülmüştür.

Çizelge 4.28. ÇBİSÇP için BMBO ve ÇS-MBO algoritmalarından elde edilen sonuçlara göre F istatistiki testinden elde edilen p değerleri

Problem No	BMBO &ÇS-BMBO (p)
4	6,23E-02
5	8,55E-01
6	1,68E-01
7	2,16E-01
8	**2,77E-04**
9	**1,27E-03**
10	4,19E-01
11	1,13E-01
12	**8,04E-03**
13	1,87E-01
14	**1,44E-03**
15	4,27E-01
16	5,12E-02
17	2,24E-01
18	**1,26E-03**
19	**6,75E-03**
20	**8,85E-03**
21	1,63E-01
22	**1,66E-04**
23	6,88E-01
24	8,95E-01
25	3,91E-01
26	5,11E-01
27	**4,29E-02**
28	7,77E-01
29	1,71E-01
30	4,08E-01

ÇBİSÇP'nin çözümünde daha başarılı olduğu görülen ÇS-BMBO algoritmasından elde edilen sonuçlar literatürden seçilen iki karınca kolonisi optimizasyon algoritması (LMACO) (Leguizamon ve Michalewicz, 1999) ve (FACO) (Fidanova, 2002) ile akıllı su damlası algoritması (IWDA) (Shah-Hosseini, 2008) ve karınca algoritmaları (Ant) (Alaya ve ark., 2004) ile kıyaslanmıştır.

Çizelge 4.29. ÇS-BMBO algoritmasının literatürden seçilen diğer algoritmalar ile karşılaştırılması

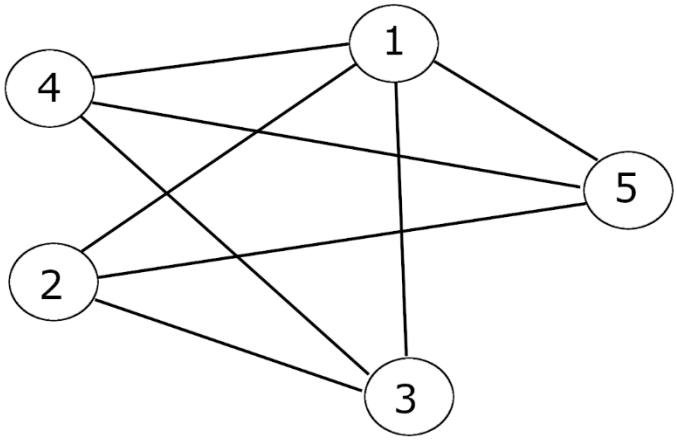
Problem No	LMACO	FACO	IWDA	Ant	ÇS-BMBO
4	**24381**	23984	24295	NA	24281
5	**24274**	24145	24158	NA	24244
6	**23551**	23523	23518	NA	23520
13	23057	NA	22936	**23064**	23055
14	**22801**	NA	22591	**22801**	22570
15	**22131**	NA	21969	**22131**	**22131**

Çizelge 4.29'daki sonuçlar, algoritmaların buldukları en iyi değerleri göstermektedir. ÇS-BMBO algoritması 15 numaralı problem için LMACO ve Ant

algoritmaları ile aynı en iyi sonucu vermiştir. 5 numaralı problemde ise FACO ve IWDA algoritmalarından daha iyi sonuç üretmiştir. 13 ve 6 nolu problemlerde IWDA algoritmasından, 4 numaralı problemde ise FACO algoritmasından daha iyi olduğu görülmektedir. Sadece 14 numaralı problemde diğer algoritmaların hepsinin gerisinde kalmıştır. Çizelge 4.29'daki sonuçlara bakıldığında tüm algoritmaların birbirine yakın sonuçlar ürettiği görülebilmektedir. Bu tablo, ÇS-BMBO algoritmasının ÇBİŞÇP'nin çözümünde alternatif bir yöntem olarak kullanılabileceğini göstermektedir.

4.5. GBP'nin MBO, ÇS-MBO ve PSO-ÇS-MBO ile Çözümü

Graf boyama problemi (GBP) ayrık bir kombinatoryal problem türüdür. GBP, düğümler ve bu düğümleri birbirine bağlayan kenarlardan oluşan bir grafta *(Grf)* birbirine kenarı olmayan düğümleri aynı renge boyayarak tüm grafı en az renk kullanarak boyanmasını amaçlayan bir problemdir. GBP'nin çözümü için öncelikle düğümler numaralandırılmıştır. Bu numaralar algoritmaların çözüm permütasyonuna yerleşecek değerleri oluşturacaktır. Düğümler arasında kenarların olup olmadığı oluşturulan komşu matris *Kmş(Grf)* kullanılarak kontrol edilmiştir. GBP'nin çözümü Şekil 4.17'de verilen örnek ile somut bir şekilde gösterilmeye çalışılmıştır.



Şekil 4.17. Örnek bir graf

Şekil 4.17'de beş düğüme sahip bir graf örneği verilmiştir. Bir numaralı düğüm dört kenara sahipken, kalan diğer düğümler üçer kenara sahiptir. Üç ve beşinci düğümler ile iki ve dördüncü düğümler arasında kenar bulunmamaktadır. Buna karşın birinci

düğümün diğer her düğümle bir kenarı vardır. Bu durumda komşu matris *Kmş(Grf)* Çizelge 4.30'daki gibi oluşur.

Çizelge 4.30. Verilen örnek grafa ait komşu matris *Kmş(Grf)*

Düğümler	**1**	**2**	**3**	**4**	**5**
1	0	1	1	1	1
2	1	0	1	0	1
3	1	1	0	1	0
4	1	0	1	0	1
5	1	1	0	1	0

Çizelge 4.30'da birinci düğümün diğer tüm düğümlerle kenarı olduğundan dolayı *Kmş(Grf)* matrisinin birinci satırı (kendisi hariç) "1" ile doldurulmuştur. Matris kare matris ve büyüklüğü düğüm sayısı kadardır. Ayrıca Şekil 4.17'de verilen graf yönsüz bir graf olduğundan dolayı oluşan *Kmş(Grf)* matrisi simetrik bir matristir. Birinci düğümden tüm düğümlere kenar olduğundan *Kmş(Grf)* matrisinin birinci satırı ve sütunu (kendisi hariç) "1" ile doldurulur. 2-4 düğümleri ile 3-5 düğümleri arasında bir kenar olmadığından dolayı *Kmş(Grf)[2,4]* ve *Kmş(Grf)[4,2]* hücreleri ile *Kmş(Grf)[3,5]* ve *Kmş(Grf)[5,3]* hücreleri "0" ile doldurulur. Problemin çözümünde hangi düğüm çiftleri arasında kenarın olduğu *Kmş(Grf)* matrisinden anlaşılabilmektedir. Bu aşamalardan sonra grafı boyamak için kullanılacak renkler belirlenir ve belirlenen renkler düğümlerde olduğu gibi numaralandırılır. Şekil 4.17'de verilen grafı boyamak için üç farklı renk (yeşil, mavi ve sarı) kullanılacağı kabul edilsin. Bu durumda renklerin numaralandırılması Şekil 4.18'deki gibi olacaktır.

Şekil 4.18. Kullanılan yeşil, mavi ve sarı renklerin numaralandırılması

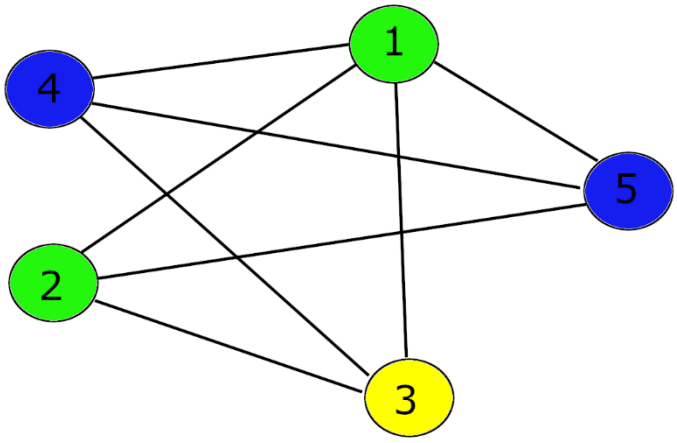
Düğümlerin numaralandırılması, *Kmş(Grf)* matrisinin oluşturulması ve renklerin kodlanmasından sonra problemin çözümündeki ön hazırlık aşaması tamamlanır. Sonraki adımda başlangıç popülasyonu oluşturulur. Popülasyondaki her bireyin sahip olduğu

çözüm vektörünün büyüklüğü, düğüm sayısı olarak belirlenir. Çözüm vektöründeki indis numaraları graf numarası ile eşleştirilirken vektör içerisine rastgele yerleşen değerler (numaralar) renk kodlarından oluşur ve böylece çözüm permütasyonu elde edilir. Elde edilen bu çözüm permütasyonunda aynı değerin (renk kodu) tekrarlanabilir olmasına dikkat edilir. Böylece birden çok düğüm aynı renge boyanabilir. Şekil 4.19'da örnek bir çözüm permütasyonu gösterilmiştir.

1	1	3	2	2

Şekil 4.19. GBP için bir bireye ait örnek bir çözüm permütasyonu

Şekil 4.19'da beş elemanlı bir çözüm vektörü içerisinde renk kodlarından oluşmuş bir çözüm permütasyonu görülmektedir. Bu dizilim ile birinci ve ikinci düğümler "1" numaralı (yeşil) renge, üçüncü düğüm "3" numaralı (sarı) renge boyanırken dört ve beşinci düğümler "2" numaralı (mavi) renge boyanmış sonucu çıkmaktadır. Buna göre Şekil 4.17'deki graf Şekil 4.20'deki gibi renklendirilir.



Şekil 4.20. Verilen örnek çözüm permütasyonuna göre renklendirilmiş graf

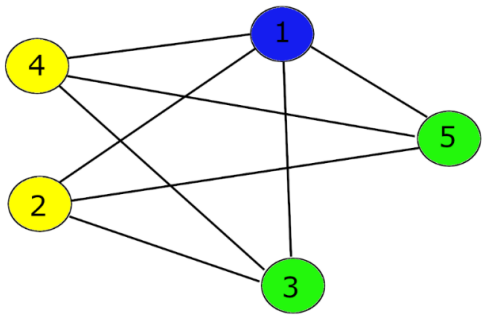
Şekil 4.19'da verilen çözüm permütasyonu Denklem 3.10'a göre değerlendirilir. Bu permütasyona göre bireyin kromatik entropi $\chi(Grf)$ (uygunluk değeri) "2" olarak hesaplanır. Bunun anlamı, iki çift düğüm farklı renklerde olması gerekirken aynı renge boyanmıştır. Şekil 4.20'ye bakıldığında 1-2 düğüm çifti ile 4-5 düğüm çiftleri arasında kenar bulunmasına rağmen aynı renge boyanmışlardır.

Bu problem için tezde kullanılan algoritmaların komşuluk yapısı değiştirilmiştir. Mevcut bir çözümden komşu çözüm üretirken bir bireyin mevcut çözüme ait çözüm vektörünün bir indisi rastgele seçilir. Seçilen bu indis içerisine kullanılan renk kodlarından bir tanesi yine rastgele seçilerek yerleştirilir. Böylelikle yeni bir çözüm permütasyonu elde edilmiş olur. Bu işlem Çizelge 4.31'de gösterilmiştir.

Çizelge 4.31. Komşu çözüm üretimi

Mevcut permütasyon	Mevcut uygunluk değeri	Rastgele seçilen pozisyon	Rastgele seçilen renk	Yeni permütasyon	Yeni uygunluk değeri
1-1-3-2-**2**	2	5	1	1-1-3-2-**1**	3

Çizelge 4.31'de elde edilen yeni permütasyona göre beşinci düğümün rengi maviden yeşile değiştirilmiştir. Bu değişim ile 4-5 düğüm çifti farklı renklendirilmiştir. Bu renklendirmeye göre $\chi(Grf) = 3$ olarak hesaplanmıştır. Yani 1-2, 1-5 ve 2-5 düğüm çiftleri, aralarında kenar olmasına rağmen aynı renge boyanmıştır. İterasyon süresince her bireyin çözümü geliştirmeye çalışılır. Şekil 4.17'de verilen örnek grafı en uygun bir şekilde boyayan dizilim 2-3-1-3-1 şeklinde bulunur. Bu dizilime göre örnek graf Şekil 4.21'deki gibi boyanır.



Şekil 4.21. En uygun şekilde renklendirilmiş graf

4.5.1. GBP için deneysel sonuçlar

Algoritmaların performanslarını kıyaslayabilmek için DIMACS kütüphanesinden seçilen test verileri kullanılmıştır (ThanhVu H. Nguye). Farklı boyutlarda yirmi beş test verisi incelenmiştir. Kütüphaneden seçilen bu veri dosyalarından bir tanesinin içeriği Şekil 4.22'de gösterilmiştir.

```
c FILE: david.col
c Translated from Stanford GraphBase File: david.gb
c Stanford GraphBase ID: book(?david?,87,0,1,64,0,0,0)
p edge 87 812
e 1 20
e 1 14
e 1 4
e 1 79
e 1 66
e 1 83
e 2 32
e 2 55
e 2 74
e 2 46
```

Şekil 4.22. GBP için kullanılan test veri dosyasının içeriğinin bir kısmı

Dosya içeriğinde "c" ile başlayan satırlar bilgi satırlarını ifade etmektedir. "p" ile başlayan satırdaki "87" değeri problemdeki düğüm sayılarını gösterirken, "812" değeri ise düğümler arasındaki kenar sayısını ifade etmektedir. "e" ile başlayan satırlar ise ilgili kenarın hangi iki düğüm arasında olduğunu ifade etmektedir. Başka bir deyişle "e 1 20" satırı birinci düğüm ile yirminci düğüm arasında bir kenarın olduğunu belirtmektedir.

İncelenen yirmi beş GBP, orijinal MBO, ÇS-MBO ve PSO-ÇS-MBO algoritmaları ile ayrı ayrı çözülmüştür. Kıyaslamanın doğruluğu açısından her algoritmada ortak olan parametreler aynı değerler ile bağımsız olarak otuz defa çalıştırılmıştır. GBP çözümünde kullanılan parametreler Çizelge 4.32'de verilmiştir.

Çizelge 4.32. GBP çözümünde kullanılan parametreler

Parametreler	Değerleri
Kuş sayısı (p)	51
Komşu sayısı (k)	3
Kanat çırpma sayısı (m)	30
Komşu paylaşım sayısı (x)	1
Sürü sayısı (z)	50
c_1 sabiti	1
c_2 sabiti	1

Her üç algoritma için GBP'nin çözümünden elde edilen sonuçlar Çizelge 4.33'te verilmiştir.

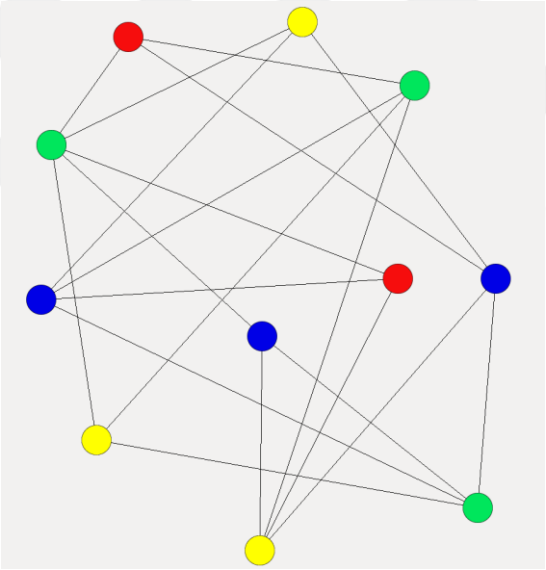
Çizelge 4.33. Literatürden seçilen 25 GBP için MBO, ÇS-MBO ve PSO-ÇS-MBO algoritmalarından elde edilen sonuçlar

P.No	Problem	Düğüm ve Kenar Sayıları	$\chi(Grf)$	Orijinal MBO			ÇS-MBO			PSO-ÇS-MBO		
				En iyi	Ortalama	En kötü	En iyi	Ortalama	En kötü	En iyi	Ortalama	En kötü
1	queen5_5.col	dgm=25;knr=160	5	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**
2	queen6_6.col	dgm=36;knr=290	7	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**
3	queen7_7.col	dgm=49;knr=476	7	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**
4	queen8_8.col	dgm=64;knr=728	9	**9**	**9**	**9**	**9**	**9**	**9**	**9**	**9**	**9**
5	queen8_12.col	dgm=96;knr=2736	12	**12**	**12**	**12**	**12**	**12**	**12**	**12**	**12**	**12**
6	queen9_9.col	dgm=81;knr=2112	10	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**
7	myciel3.col	dgm=11;knr=20	4	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**
8	myciel4.col	dgm=23;knr=71	5	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**
9	myciel5.col	dgm=47;knr=236	6	**6**	**6**	**6**	**6**	**6**	**6**	**6**	**6**	**6**
10	myciel6.col	dgm=95;knr=755	7	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**	**7**
11	huck.col	dgm=74;knr=301	11	**11**	**11**	**11**	**11**	**11**	**11**	**11**	**11**	**11**
12	jean.col	dgm=80;knr=254	10	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**
13	david.col	dgm=87;knr=406	11	**11**	**11**	**11**	**11**	**11**	**11**	**11**	**11**	**11**
14	myciel7.col	dgm=191;knr=2360	8	**8**	**8**	**8**	**8**	**8**	**8**	**8**	**8**	**8**
15	games120.col	dgm=120;knr=638	9	**9**	**9**	**9**	**9**	**9**	**9**	**9**	**9**	**9**
16	miles250.col	dgm=128;knr=387	8	**8**	**8**	**8**	**8**	**8**	**8**	**8**	**8**	**8**
17	anna.col	dgm=138;knr=493	11	**11**	**11**	**11**	**11**	**11**	**11**	**11**	**11**	**11**
18	miles500.col	dgm=128;knr=1170	20	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**	**20**
19	miles750.col	dgm=128;knr=2113	31	**31**	**31**	**31**	**31**	**31**	**31**	**31**	**31**	**31**
20	miles1000.col	dgm=128;knr=3216	42	**42**	**42**	**42**	**42**	**42**	**42**	**42**	**42**	**42**
21	mulsol.i.1.col	dgm=197;knr=3925	49	**49**	**49**	**49**	**49**	**49**	**49**	**49**	**49**	**49**
22	mulsol.i.4.col	dgm=185;knr=3946	31	**31**	**31**	**31**	**31**	**31**	**31**	**31**	**31**	**31**
23	fpsol2.i.1.col	dgm=496;knr=11654	65	**65**	**65**	**65**	**65**	**65**	**65**	**65**	**65**	**65**
24	inithx.i.1.col	dgm=864;knr=18707	54	**54**	**54**	**54**	**54**	**54**	**54**	**54**	**54**	**54**
25	inithx.i.3.col	dgm=621;knr=13969	31	**31**	**31**	**31**	**31**	**31**	**31**	**31**	**31**	**31**

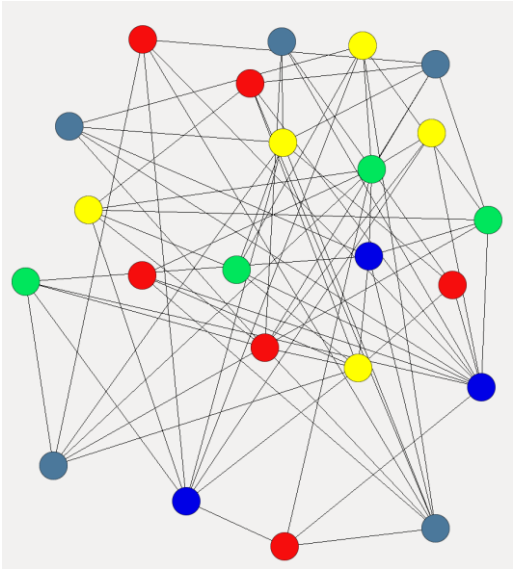
Çizelge 4.33'te literatürden seçilen yirmi beş test probleminin orijinal MBO, ÇS-MBO ve PSO-ÇS-MBO algoritmaları ile çözümünden elde edilen sonuçlar listelenmiştir. Problem sütunu örnek veri setini içeren problemin dosya ismini göstermektedir. dgm, graftaki düğüm sayısını gösterirken, knr graftaki toplam kenar sayısını temsil etmektedir. $\chi(Grf)$ ise kromatik entropi olarak isimlendirilen, problemin optimum sonucunu göstermektedir.

Her üç algoritma da farklı büyüklükteki tüm veri setleri üzerinde bağımsız olarak otuz defa çalıştırılmış ve yapılan testlerin tamamında optimum sonuca ulaşılmıştır. Buradan anlaşılabiliyor ki orijinal MBO ve geliştirilmiş versiyonları olan ÇS-MBO ve PSO-ÇS-MBO algoritmaları, GBP'nin çözümünde oldukça başarılı ve etkin bir algoritma olduklarını göstermişlerdir.

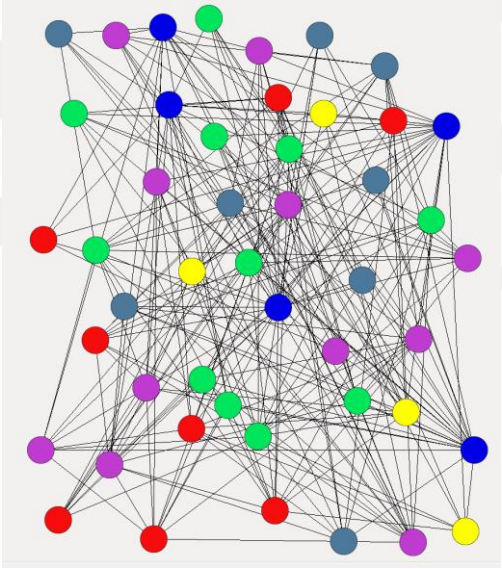
Kullanılan bazı veri setlerinden elde edilen sonuçlara göre renklendirilmiş graflar Şekil 4.23 ile 4.25 arasında verilmiştir.



Şekil 4.23. myciel3.col probleminin renklendirilmiş graf çıktısı



Şekil 4.24. myciel4.col probleminin renklendirilmiş graf çıktısı



Şekil 4.25. myciel5.col probleminin renklendirilmiş graf çıktısı

Her üç algoritma da tüm testlerde optimum sonucu verdiklerinden dolayı istatistiksel olarak incelenme gereği duyulmamıştır. Ancak her üç algoritmadan elde edilen bu sonuçlar, literatürden seçilen diğer yöntemlerle kıyaslanmıştır. Seçilen bu yöntemler, değiştirilmiş guguk kuşu optimizasyon algoritması (MCO) (Mahmoudi ve Lotfi, 2015), karınca algoritması (ABAC) (Bui ve ark., 2008) ve arı davranışlarına dayanan bir algoritmadır (BEECOL) (Faraji ve Javadi, 2011).

Tez çalışmasında GBP için test edilen üç algoritma (orijinal MBO, ÇS-MBO ve PSO-ÇS-MBO) da aynı sonuçları verdiklerinden dolayı Çizelge 4.34'te tez çalışmasından elde edilen sonuçlar kısaca $MBO_{tüm}$ olarak isimlendirilmiştir.

Çizelge 4.34. $MBO_{tüm}$ ile literatürden seçilen diğer algoritmaların karşılaştırılması

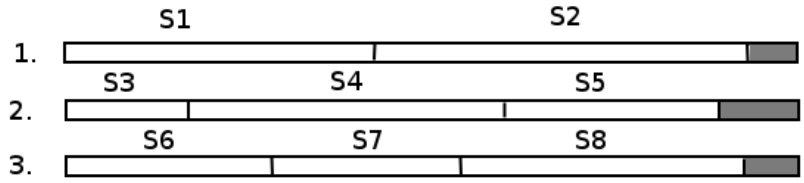
Problem No	MCO	ABAC	BEECOL	$MBO_{tüm}$
1	**5**	**5**	**5**	**5**
2	8	**7**	8	**7**
3	**7**	**7**	8	**7**
4	10	**9**	10	**9**
5	13	**12**	**12**	**12**
6	11	**10**	11	**10**
7	**4**	**4**	**4**	**4**
8	**5**	**5**	**5**	**5**
9	**6**	**6**	**6**	**6**
10	**7**	**7**	**7**	**7**
11	**11**	**11**	**11**	**11**
12	**10**	**10**	**10**	**10**
13	**11**	**11**	**11**	**11**
14	**8**	**8**	**8**	**8**
15	**9**	**9**	**9**	**9**
16	**8**	**8**	**8**	**8**
17	**11**	**11**	**11**	**11**
18	**20**	**20**	**20**	**20**
19	**31**	**31**	**31**	**31**
20	**42**	**42**	**42**	**42**
21	**49**	**49**	**49**	**49**
22	**31**	**31**	**31**	**31**
23	**65**	**65**	**65**	**65**
24	**54**	**54**	**54**	**54**
25	**31**	**31**	**31**	**31**

Çizelge 4.34'teki değerlere göre ABAC ve $MBO_{tüm}$ yöntemlerinin tüm problemlerde optimum sonucu buldukları görülmektedir. Buna karşın BEECOL ve MCO algoritmaları dört problemde optimum sonuca yakın sonuçlar üretmişlerdir. Bu sonuçlar, GBP'nin çözümünde orijinal MBO, ÇS-MBO ve PSO-ÇS-MBO algoritmalarının da güçlü bir alternatif yöntem olduğunu göstermiştir.

4.6. TBKP'nin MBO, ÇS-MBO ve PSO-ÇS-MBO ile Çözümü

Tek boyutlu kesim problemi (TBKP), ayrık kombinatoryal optimizasyon problemlerindendir. TBKP'nin algoritmalara uygulanmasında stoktaki hammadde büyüklüklerinin eşit ve hammadde sayısının sonsuz olduğu varsayılmıştır. Kesilecek parçaların farklı uzunluklara ve aynı boyuttaki parçalardan birden çok olabileceği varsayılmıştır. Bunun yanında kesim işlemi esnasında kesici bıçağın kalınlığı ve artan firelerin bir sonraki siparişte kullanılması da ihmal edilmiştir.

Şekil 4.26'da stokta bulunan eşit büyüklüğe sahip 3 hammadde ve bu hammaddelerden kesilecek olan 8 parça (S1, …, S8) gösterilmiştir. Buna ilaveten boyalı alanlar her hammadde üzerinde oluşan fireleri göstermektedir.



Şekil 4.26. Kesilecek 8 parçanın 3 hammadde üzerindeki kesim planı

Şekil 4.26'ya göre 8 parçanın tamamı birbirinden farklı uzunluklara sahiptir. Denklem 3.12'deki amaç fonksiyonu Şekil 4.26'daki boyalı olan fire toplamlarını en aza indirgemektir. Şekil 4.26'da kesilecek parçalar (siparişler) kodlanır. Kodlanmış her bir parça rastgele çözüm vektörüne sırayla atanır. Atama işleminde her bir parça yerleştirilmeden önce hammaddenin yerleştirilmeyen alanına uygun olup olmadığı kontrol edilir. Eğer uygun değilse (sipariş edilen parça hammaddenin kalan kısmından daha uzun ise) yeni bir hammadde üzerine yerleştirilir. Bu işleme tüm siparişler yerleştirilene kadar devam edilir. Böylece her bir bireyin çözüm permütasyonu elde edilir. Bu türden bir problemin çözümünde kesilecek olan parçaların hammadde üzerindeki yerleşim sırası direkt olarak fire miktarını etkileyecektir. Çözüm permütasyonunun uygunluğu Denklem 3.12'ye göre belirlenir.

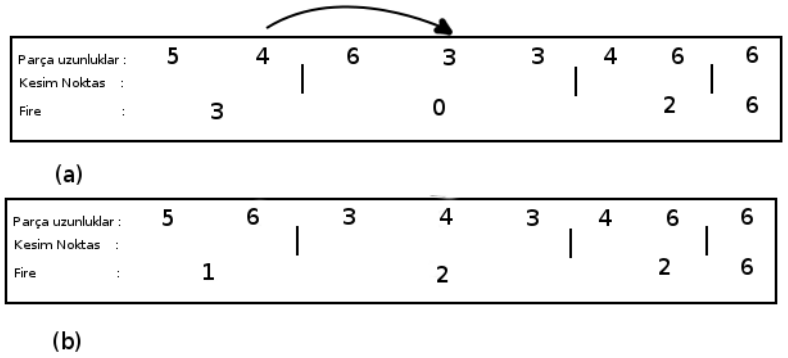
Şekil 4.27'de bir bireyinin örnek permütasyon dizilimi verilmiştir. Bu örnekte stoktaki her hammadde eşit uzunluğa sahip ve her birinin boyu 12 birimdir. Kesilecek olan parçaların adetleri ve uzunlukları ise şöyledir; 3 birim uzunluğunda 2 adet, 4 birim uzunluğunda 2 adet, 5 birim uzunluğunda 1 adet ve 6 birim uzunluğunda 3 adet. Şekil 4.27'de kesilecek olan tüm parçalar rastgele seçilip hammaddeler üzerine yerleştirilerek bir çözüm permütasyonu elde edilmiştir.

Şekil 4.27. Kesilecek 8 parçanın çözüm permütasyonundaki yerleşimi

Şekil 4.27'deki yerleşim incelendiğinde, 5 birim uzunluğa sahip parça ile 4 birim uzunluğa sahip parçalardan bir tanesi yerleştirildiğinde bu yerleşen iki parçanın toplam uzunluğu 9 birim etmektedir. Bu iki parçanın yanına rastgele seçilen siparişlerden 6 birimlik parçalardan bir tanesinin daha eklenmesi durumunda, 12 birim uzunluğa sahip

hammaddenin uzunluğu aşılmaktadır. Bu kontrol yapılarak eşit uzunluğa sahip birinci hammadde üzerine 5 ve 4 birimlik iki sipariş yerleştirilmiştir. Bu durumda ilk hammaddeden oluşacak fire miktarı 3 (12 – (5 + 4)) birim olarak belirlenir. Seçilen 6 birimlik üçüncü parça ikinci hammadde üzerine yerleştirilerek yerleştirme işlemine devam edilir. Kalan tüm siparişler de aynı kontrol aşamasından geçerek hammaddeler üzerine yerleştirildiğinde Şekil 4.27'ye göre toplamda 4 hammadde kullanıldığı ve bu hammaddelerde oluşan fire miktarlarının sırasıyla 3, 0, 2 ve 6 birim olduğu görülmektedir. Bireye ait bu permütasyon dizilimine göre her hammaddeden oluşan firelerin toplamı (3 + 0 + 2 + 6 = 11) o bireyin uygunluk değerini göstermektedir. Buna göre en az toplam fire miktarına sahip birey en iyi kesim planına sahip bireydir.

Her bireyin uygunluk değerleri hesaplandıktan sonra başlangıç çözümleri geliştirilmeye çalışılır. Bu işlem MBO algoritmasındaki komşuluk yapısı kullanılarak gerçekleştirilir. Komşuluk metodu olarak araya ekleme metodu tercih edilmiştir. Bu metoda göre, çözüm permütasyonundaki rastgele bir parça alınır ve yine rastgele belirlenen bir pozisyona yerleştirilir. Bu durum Şekil 4.28'de gösterilmiştir.



Şekil 4.28. Araya ekleme yöntemi kullanılarak örnek bir komşu çözüm üretme

Şekil 4.28 (a)'da çözüm permütasyonundaki rastgele seçilen ikinci sıradaki kesilecek parça aynı permütasyonda yine rastgele seçilen dördüncü pozisyona yerleştirilmiştir. Elde edilen yeni çözüm permütasyonu Şekil 4.28 (b)'de gösterilmiştir. Yeni dizilime göre birinci hammaddedeki fire 3 birimden 1 birime düşerken ikinci hammaddedeki fire 0 birimden 2 birime çıkmıştır. Örnekte verilen işlem üçüncü ve dördüncü hammaddelerin sırasını değiştirmediği için fire miktarları aynı kalmıştır.

4.6.1. TBKP için deneysel sonuçlar

Algoritmaların performansını kıyaslayabilmek için literatürden (Liang ve ark., 2002) seçilmiş beş problem kullanılmıştır. Literatürden seçilen örneklere ait veri setinden bir örnek Şekil 4.29'da verilmiştir. Verilen örneğe göre ilk değer (14) sabit olan stokların uzunluğunu göstermektedir. Daha sonraki her satırda bulunan değer çiftlerinden ilki (5) talepte bulunulan parça adedini ve aynı satırdaki ikinci değer (3) talep dilen parçaların uzunluğunu ifade etmektedir. İlk iki satıra göre değerler şu şekilde yorumlanabilir: her birisi 14 birim olan sonsuz sayıdaki stoktan 3 birim uzunluğunda 5 adet parça kesilmesi istenmektedir.

```
14
5    3
2    4
1    5
2    6
4    7
2    8
1    9
3    10
```

Şekil 4.29. TBKP için kullanılan örnek veri setleri

Her problem orijinal MBO, ÇS-MBO ve PSO-ÇS-MBO algoritmaları ile bağımsız olarak 30 kez çalıştırılmış, sonuçların en iyi, ortalama ve en kötü değerleri kaydedilmiştir. Problemin çözümü için algoritmaların aldığı parametreler Çizelge 4.35'de verilmiştir.

Çizelge 4.35. TBKP çözümünde kullanılan parametreler

Parametreler	Değerleri
Kuş sayısı (p)	71
Komşu sayısı (k)	5
Kanat çırpma sayısı (m)	30
Komşu paylaşım sayısı (x)	1
Sürü sayısı (z)	30
c_1 sabiti	1
c_2 sabiti	1

Her üç algoritma için TBKP'nin çözümünden elde edilen sonuçlar Çizelge 4.36'da verilmiştir.

Çizelge 4.36. Literatürden seçilen 5 TBKP için MBO, ÇS-MBO ve PSO-ÇS-MBO algoritmalarından elde edilen sonuçlar

Problem No	Stok Uzunluğu	Toplam Parça Sayısı	MBO			ÇS-MBO			PSO-ÇS-MBO		
			En iyi	Ortalama	En kötü	En iyi	Ortalama	En kötü	En iyi	Ortalama	En kötü
1a	14	20	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**
2a	15	50	**13**	**13**	**13**	**13**	**13**	**13**	**13**	**13**	**13**
3a	25	60	**25**	**25**	**25**	**25**	**25**	**25**	**25**	**25**	**25**
4a	25	60	**11**	**11**	**11**	**11**	**11**	**11**	**11**	**11**	**11**
5a	4300	126	**14130**	**14130**	**14130**	**14130**	**14130**	**14130**	**14130**	**14130**	**14130**

Çizelge 4.36'da literatürden seçilen beş farklı TBKP, orijinal MBO, ÇS-MBO ve PSO-ÇS-MBO algoritmalarının performansını test etmek için kullanılmıştır. Her problemde stok uzunluğu sabit olup sonsuz sayıda olduğu varsayılmıştır. Farklı uzunluklarda ve sayılardan oluşan taleplerin toplam adedi ise toplam parça sayısı sütununda verilmiştir. Koyu renkle belirtilen sonuçlar, ilgili problemden elde edilen en iyi ve ortalama değerleri ifade etmektedir. Buna göre, her üç algoritma da TBKP için yapılan tüm testlerde aynı değeri üretmişlerdir. Bu problemde de her üç algoritmanın sonuçları aynı olduğundan istatistiksel bir test uygulanmamıştır.

Orijinal MBO, ÇS-MBO ve PSO-ÇS-MBO algoritmalarından elde edilen sonuçlar literatürden seçilen diğer yöntemlerle kıyaslanmıştır. Her üç algoritma da aynı sonuçları verdiğinden Çizelge 4.37'de bu üç algoritma kısaca $MBO_{tüm}$ olarak isimlendirilmiştir. Direkt bir kıyaslama yapabilmek için toplam fire miktarlarının oranları hesaplanmış ve bu oran üzerinden kıyaslamalar yapılmıştır. Bu oran Denklem (4.3'e göre hesaplanmıştır. Evrimsel programlama (EP) (Liang ve ark., 2002), basit tabu arama (bTA) ve geliştirilmiş tabu arama (gTA) (Yang ve ark., 2006) algoritmaları ile kıyaslanmıştır.

$$TFO = \frac{ToplamFire}{U_t} \times 100 \tag{4.3}$$

TFO toplam fire oranı, *ToplamFire* ilgili problem için bulunan en iyi fire miktarı, U_t ise her problem için kesilecek parçaların toplam uzunluklarını ifade etmektedir.

Çizelge 4.37. $MBO_{tüm}$ ile literatürden seçilen diğer algoritmaların karşılaştırılması

Problem No	EP(TFO)	bTA(TFO)	gTA(TFO)	$MBO_{tüm}$(TFO)
1a	**2,44**	**2,44**	**2,44**	**2,44**
2a	**3,92**	8,43	**3,92**	**3,92**
3a	**0**	6,67	**0**	6,67
4a	**2,37**	7,76	**2,37**	**2,37**
5a	5,52	11,25	**5,29**	6,52

Çizelge 4.37'de literatürden seçilen algoritmalar orijinal MBO, ÇS-MBO ve PSO-ÇS-MBO algoritmalarından elde edilen sonuçlarla kıyaslanmıştır. 1a problemi için tüm algoritmalar aynı sonucu verirken, 2a probleminde bTA algoritması hariç diğerleri yine aynı sonucu üretmişlerdir. 3a probleminde EP ve gTA algoritmalarının toplam fireleri sıfır olarak hesaplanmıştır. Bu problem için bTA ve $MBO_{tüm}$ aynı fire oranına sahiptir. 4a

probleminde yine bTA dışındaki tüm yöntemler aynı fire oranını bulmuşlardır. Sadece problem 5a için öne çıkan tek algoritma gTA'dır.

TBKP için tüm bu sonuçlara bakıldığında orijinal MBO, ÇS-MBO ve PSO-ÇS-MBO algoritmalarının TBKP için alternatif bir yöntem olarak kullanılabileceğini göstermiştir.

5. SONUÇLAR VE ÖNERİLER

5.1. Sonuçlar

Bu tez çalışmasında ayrık kombinatoryal optimizasyon problemlerinin çözümü için tasarlanan MBO algoritması iyileştirilmiş ve farklı türden problemleri çözebilecek kabiliyet kazandırılmıştır.

Orijinal MBO algoritması, göçmen kuşların göç esnasında daha uzun mesafelere daha az enerji harcayarak uçtukları "V" formasyonundaki dizilimden esinlenmiştir. Orijinal MBO, popülasyon tabanlı bir optimizasyon algoritması olup, problemlerin çözümünde tek bir sürü yapısı kullanmaktadır. Sürü bireyleri fayda mekanizması olarak isimlendirilen bir komşu çözüm paylaşım yöntemiyle birbirleri arasında haberleştirilmektedir. Bu sayede sürüdeki tüm bireyler yine sürü içerisindeki en iyi bireye doğru yakınsamaları sağlanmıştır. Ancak bu yaklaşım, bazı optimizasyon problemlerinin çözümünde algoritmanın yerel optimum çözümlere takılmasına neden olabilmektedir. Çözüm uzayının daha iyi ve detaylı taranabilmesi için tek sürü kullanan orijinal MBO algoritmasına birden çok sürü dahil edilmiştir. Çok sürülü MBO (ÇS-MBO) algoritmasında her sürü içerisindeki bireyler orijinal MBO algoritmasında olduğu gibi komşu paylaşım yöntemiyle haberleştirilmiştir. Ancak bu etkileşim sadece aynı sürü içerisindeki bireyler arasında yapılmıştır. Yapılan deneysel çalışmalarda, bu yeni yaklaşımın orijinal MBO algoritmasından daha iyi sonuçlar verdiği gözlenmiştir. Ancak arama uzayı çok sürü kullanılarak her ne kadar daha detaylı taranmış olsa da global optimum çözümden uzakta olan sürü ve onun içerisindeki bireyler global optimum yakınlarında arama işlemi yapamamışlardır. Bundan dolayı ÇS-MBO algoritmasında sürülerin de birbirleriyle etkileşimde olması amaçlanmıştır. Bu etkileşimi sağlamak için yine popülasyon tabanlı bir başka algoritmadan faydalanılmıştır. Parçacık sürü optimizasyon (PSO) algoritması da MBO algoritması gibi popülasyon tabanlı olup bireylerin konum ve hız bilgilerinden faydalanarak sürü içerisindeki diğer bireylerin haberleşmesini sağlayabilmektedir. PSO'nun bu haberleşme yönteminden faydalanılarak PSO tabanlı çok sürülü MBO (PSO-ÇS-MBO) algoritması geliştirilerek orijinal MBO algoritması farklı bir yöntemle iyileştirilmiştir. PSO-ÇS-MBO algoritmasında PSO sadece sürülerin haberleştirilmesinde kullanılmıştır. Böylece orijinal MBO'nun temel yapısı da korunmuştur. Orijinal MBO algoritmasında sürü liderinin kritik bir önemi vardır. Sürü lideri, ürettiği komşu çözümleri sürünün her iki tarafına da

paylaştırabilmektedir. Bu sayede liderin etkisi sürünün her iki kanadı üzerinde de hissedilmektedir. Bundan dolayı sürüler arası etkileşim sadece sürü liderleri arasında gerçekleştirilmiştir. Yani her sürünün lideri PSO'daki bir bireye benzetilmiş ve sürü liderlerinin konum ve hız bilgilerinden faydalanılarak sürünün konumu güncellenmiştir. PSO-ÇS-MBO'da orijinal MBO'dan faklı olarak her sürü içerisindeki tüm bireylere konum ve hız bilgileri eklenmiştir. Çünkü her birey potansiyel bir sürü lideri adayıdır. Sürülerin haberleştirilmesi lider değişimi esnasında gerçekleştirilmiştir. Sürünün lideri olan yeni bireyin konum bilgisi, diğer sürülerdeki yeni liderlerin konum ve hız bilgilerinden etkilenerek değiştirilmiştir. Bu güncel konuma göre liderin çözüm permütasyonu da güncellenmiştir. Lider değişimiyle gerçekleşen bu güncellemelerden sonra sürü içerisindeki bireyler orijinal MBO'da olduğu gibi sadece kendi aralarında haberleştirilmiştir. Böylece konumu değiştirilen sürü yeni konumunda yerel arama yapmaya devam etmiştir.

Orijinal MBO ve iyileştirilmiş ÇS-MBO ve PSO-ÇS-MBO algoritmalarının performansları literatürde iyi bilinen GSP, ÇBİYSBP, GBP ve TBKP üzerinde test edilmiştir. Elde edilen sonuçlar göstermiştir ki ÇS-MBO ve PSO-ÇS-MBO algoritmaları orijinal MBO algoritmasından daha iyi sonuçlar üretmişlerdir. Ayrıca GBP'de tüm algoritmalar optimum sonucu bulabilmişlerdir. GSP'de, ÇS-MBO algoritması orijinal MBO'dan daha iyi, PSO-ÇS-MBO algoritması da ÇS-MBO'dan daha iyi sonuçlar üretmiştir. Ayrıca ÇBİYSBP için iyileştirilmiş algoritmalar en iyi ve ortalama sonuçlara göre orijinal MBO'dan daha iyi sonuçlar elde etmişlerdir. ÇS-MBO ve PSO-ÇS-MBO algoritmaları, problem boyutu ve koordinat sayısına bağlı olarak birbirlerine yakın sonuçlar üretmişlerdir. Ortalama değerlere göre ÇS-MBO ve PSO-ÇS-MBO algoritmalarının aralarında çok büyük farkların olmadığı da tespit edilmiştir.

Geliştirilmiş ikili MBO (BMBO) algoritması ile BMBO'nun iyileştirilmiş versiyonu olan ÇS-BMBO algoritmasının performansları yine literatürde iyi bilinen İSÇP ve ÇBİSÇP üzerinde test edilmiştir. Her iki problemde de ÇS-BMBO algoritmasından elde edilen ortalama sonuçlar BMBO algoritmasından daha iyidir. Buna rağmen aralarındaki farkın çok büyük olmadığı gözlenmiştir. Özellikle düşük boyutlu problemlerde her iki yöntemin de optimum sonuca ulaştığı görülmüştür. Ancak problem boyutu büyüdükçe ÇS-BMBO'nun BMBO'ya göre daha başarılı sonuçların elde edildiği görülmüştür.

Sonuç olarak; bu tez kapsamında iyileştirilen ÇS-MBO ve PSO-ÇS-MBO algoritmaları ile geliştirilen BMBO ve ÇS-BMBO isminde iyileştirilmiş ve geliştirilmiş

yeni meta-sezgisel algoritmalar önerilmiştir. Problemlerin çözümünde kaliteli ve tutarlı sonuçlar üretebilen bu algoritmalar ile literatüre bu alanda bir katkı sağlanmıştır.

5.2. Öneriler

Sürekli optimizasyon problemlerinin çözümünde kullanılacak bir sürekli MBO algoritması geliştirilebilir. Ayrıca geliştirilen sürekli optimizasyon algoritması, bu çalışmada olduğu gibi çok sürülü ve etkileşimli çok sürülü şekilde tasarlanabilir.

İkili problemlerin çözümünde kullanmak için ÇS-BMBO algoritmasındaki sürüler farklı bir yöntemle haberleştirilebilirler.

KAYNAKLAR

Akcay, Y., Li, H. J. ve Xu, S. H., 2007, Greedy algorithm for the general multidimensional knapsack problem, *Annals of Operations Research*, 150 (1), 17-29.

Alaya, I., Solnon, C. ve GHéDIRA, K., 2004, Ant algorithm for the multi-dimensional knapsack problem, *International Conference on Bioinspired Optimization Methods and their Applications (BIOMA 2004*.

Alidaee, B., Glover, F., Kochenberger, G. A. ve Rego, C., 2005, A new modeling and solution approach for the number partitioning problem, *Advances in Decision Sciences*, 2005 (2), 113-121.

Alkaya, A. F., Algin, R., Sahin, Y., Agaoglu, M. ve Aksakalli, V., 2014, Performance of Migrating Birds Optimization Algorithm on Continuous Functions, *Advances in Swarm Intelligence, Icsi 2014, Pt Ii*, 8795, 452-459.

Alonso, C. L., Caro, F. ve Montana, J. L., 2006, A flipping local search genetic algorithm for the multidimensional 0-1 knapsack problem, *Current Topics in Artificial Intelligence*, 4177, 21-30.

Alves, M. J. ve Almeida, M., 2007, MOTGA: A multiobjective Tchebycheff based genetic algorithm for the multidimensional knapsack problem, *Computers & Operations Research*, 34 (11), 3458-3470.

Arguello, M. F., Feo, T. A. ve Goldschmidt, O., 1996, Randomized methods for the number partitioning problem, *Computers & Operations Research*, 23 (2), 103-111.

Avanthay, C., Hertz, A. ve Zufferey, N., 2003, A variable neighborhood search for graph coloring, *European Journal of Operational Research*, 151 (2), 379-388.

Azadeh, A., Farahani, M. H., Eivazy, H., Nazari-Shirkouhi, S. ve Asadipour, G., 2013, A hybrid meta-heuristic algorithm for optimization of crew scheduling, *Applied Soft Computing*, 13 (1), 158-164.

Balev, S., Yanev, N., Freville, A. ve Andonov, R., 2007, A dynamic programming based reduction procedure for the multidimensional 0-1 knapsack problem, *European Journal of Operational Research*, 186 (1), 63-76.

Balev, S., Yanev, N., Fréville, A. ve Andonov, R., 2008, A dynamic programming based reduction procedure for the multidimensional 0–1 knapsack problem, *European Journal of Operational Research*, 186 (1), 63-76.

Banos, R., Ortega, J., Gil, C., Marquez, A. L. ve de Toro, F., 2013, A hybrid meta-heuristic for multi-objective vehicle routing problems with time windows, *Computers & Industrial Engineering*, 65 (2), 286-296.

Bauke, H., Franz, S. ve Mertens, S., 2004, Number partitioning as a random energy model, *Journal of Statistical Mechanics-Theory and Experiment*.

Belov, G. ve Scheithauer, G., 2002, A cutting plane algorithm for the one-dimensional cutting stock problem with multiple stock lengths, *European Journal of Operational Research*, 141 (2), 274-294.

Benkalai, I., Rebaine, D., Gagne, C. ve Baptiste, P., 2017, Improving the migrating birds optimization metaheuristic for the permutation flow shop with sequence-dependent set-up times, *International Journal of Production Research*, 55 (20), 6145-6157.

Berberler, M. E., Nuriyev, U. ve Yıldırım, A., 2011, A software for the one-dimensional cutting stock problem, *Journal of King Saud University-Science*, 23 (1), 69-76.

Berretta, R., Cotta, C. ve Moscato, P., 2004, Enhancing the performance of memetic algorithms by using a matching-based recombination algorithm, *Metaheuristics: Computer Decision-Making*, 86, 65-90.

Boussier, S., Vasquez, M., Vimont, Y., Hanafi, S. ve Michelon, P., 2010, A multi-level search strategy for the 0-1 Multidimensional Knapsack Problem, *Discrete Applied Mathematics*, 158 (2), 97-109.

Boyer, V., Elkihel, M. ve El Baz, D., 2009, Heuristics for the 0-1 multidimensional knapsack problem, *European Journal of Operational Research*, 199 (3), 658-664.

Bui, T. N., Nguyen, T. H., Patel, C. M. ve Phan, K. A. T., 2008, An ant-based algorithm for coloring graphs, *Discrete Applied Mathematics*, 156 (2), 190-200.

Cherri, A. C., Arenales, M. N., Yanasse, H. H., Poldi, K. C. ve Vianna, A. C. G., 2014, The one-dimensional cutting stock problem with usable leftovers - A survey, *European Journal of Operational Research*, 236 (2), 395-402.

Chu, P. C. ve Beasley, J. E., 1998, A genetic algorithm for the multidimensional knapsack problem, *Journal of Heuristics*, 4 (1), 63-86.

Debels, D., De Reyck, B., Leus, R. ve Vanhoucke, M., 2006, A hybrid scatter search/electromagnetism meta-heuristic for project scheduling, *European Journal of Operational Research*, 169 (2), 638-653.

Dell'Amico, M., Iori, M., Martello, S. ve Monaci, M., 2008, Heuristic and exact algorithms for the identical parallel machine scheduling problem, *Informs Journal on Computing*, 20 (3), 333-344.

Dorigo, M. ve Di Caro, G., 1999, Ant colony optimization: a new meta-heuristic, *Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406)*, 1470-1477.

Duan, H. B., Yu, Y. X., Zhang, X. Y. ve Shao, S. A., 2010, Three-dimension path planning for UCAV using hybrid meta-heuristic ACO-DE algorithm, *Simulation Modelling Practice and Theory*, 18 (8), 1104-1115.

Duman, E., Uysal, M. ve Alkaya, A. F., 2012, Migrating Birds Optimization: A new metaheuristic approach and its performance on quadratic assignment problem, *Information Sciences*, 217, 65-77.

Duman, E. ve Elikucuk, I., 2013, Solving Credit Card Fraud Detection Problem by the New Metaheuristics Migrating Birds Optimization, *Advances in Computational Intelligence, Pt Ii*, 7903, 62-+.

Durrett, G., Medard, M. ve O'Reilly, U. M., 2010, A Genetic Algorithm to Minimize Chromatic Entropy, *Evolutionary Computation in Combinatorial Optimization, Proceedings*, 6022, 59-+.

Eberhart, R. ve Kennedy, J., 1995, A new optimizer using particle swarm theory, *Micro Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on*, 39-43.

Eiben, A. E., Van der Hauw, J. K. ve Van Hemert, J. I., 1998, Graph coloring with adaptive evolutionary algorithms, *Journal of Heuristics*, 4 (1), 25-46.

Faraji, M. ve Javadi, H. H. S., 2011, Proposing a new algorithm based on bees behavior for solving graph coloring, *International Journal Contemp. Math. Sciences*, 6 (1), 41-49.

Feng, Y., Wang, G. G., Deb, S., Lu, M. ve Zhao, X. J., 2017, Solving 0-1 knapsack problem by a novel binary monarch butterfly optimization, *Neural Computing & Applications*, 28 (7), 1619-1634.

Fidanova, S., 2002, Evolutionary algorithm for multiple knapsack problem, *IN PROCEEDINGS OF PPSN-VII, SEVENTH INTERNATIONAL CONFERENCE ON PARALLEL PROBLEM SOLVING FROM NATURE, LECTURE*.

Fleurent, C. ve Ferland, J. A., 1996, Genetic and hybrid algorithms for graph coloring, *Annals of Operations Research*, 63, 437-461.

Fuksz, L., Pop, P. ve Zelina, I., 2013, Heuristic algorithms for solving the bi-dimensional two-way number partitioning problem, *Stud Univ Babes-Bolyai, Ser Inform LVIII*, 3, 17-28.

Galinier, P. ve Hao, J. K., 1999, Hybrid evolutionary algorithms for graph coloring, *Journal of Combinatorial Optimization*, 3 (4), 379-397.

Gao, K. Z., Suganthan, P. N. ve Chua, T. J., 2013, An Enhanced Migrating Birds Optimization Algorithm for No-wait Flow Shop Scheduling Problem, *Proceedings of the 2013 Ieee Symposium on Computational Intelligence in Scheduling (Cisched)*, 9-13.

Gao, L. ve Pan, Q. K., 2016, A shuffled multi-swarm micro-migrating birds optimizer for a multi-resource-constrained flexible job shop scheduling problem, *Information Sciences*, 372, 655-676.

Garey, M. R. ve Johnson, D. S., 1979, Computers and intractability: a guide to NP-completeness, WH Freeman and Company, San Francisco.

Geng, X. T., Chen, Z. H., Yang, W., Shi, D. Q. ve Zhao, K., 2011, Solving the traveling salesman problem based on an adaptive simulated annealing algorithm with greedy search, *Applied Soft Computing*, 11 (4), 3680-3689.

Gherboudj, A., Layeb, A. ve Chikhi, S., 2012, Solving 0-1 knapsack problems by a discrete binary version of cuckoo search algorithm, *International Journal of Bio-Inspired Computation*, 4 (4), 229-236.

Glover, F., 1989, Tabu search—part I, *ORSA Journal on computing*, 1 (3), 190-206.

Gradisar, M. ve Trkman, P., 2005, A combined approach to the solution to the general one-dimensional cutting stock problem, *Computers & Operations Research*, 32 (7), 1793-1807.

Granmo, O. C., Oommen, B. J., Myrer, S. A. ve Olsen, M. G., 2007, Learning automata-based solutions to the nonlinear fractional knapsack problem with applications to optimal resource allocation, *Ieee Transactions on Systems Man and Cybernetics Part B-Cybernetics*, 37 (1), 166-175.

Gulcu, S., Mahi, M., Baykan, O. K. ve Kodaz, H., 2018, A parallel cooperative hybrid method based on ant colony optimization and 3-Opt algorithm for solving traveling salesman problem, *Soft Computing*, 22 (5), 1669-1685.

Gwee, B., Lim, M. ve Ho, J., 1993, Solving four-colouring map problems using genetic algorithm, *Proceedings of International Two-Stream Conference on Artificial Neural Networks and Expert Systems, ANNEZ'93*, 322-323.

Hacibeyoglu, M., Tongur, V. ve Alaykiran, K., 2014, Solving the bi-dimensional two-way number partitioning problem with heuristic algorithms, *Application of Information and Communication Technologies (AICT), 2014 IEEE 8th International Conference on*, 1-5.

Hanafi, S. ve Wilbaut, C., 2008, Scatter search for the 0–1 multidimensional knapsack problem, *Journal of Mathematical Modelling and Algorithms*, 7 (2), 143-159.

Hanafi, S. ve Wilbaut, C., 2011, Improved convergent heuristics for the 0-1 multidimensional knapsack problem, *Annals of Operations Research*, 183 (1), 125-142.

Hayes, B., 2002, Computing science: The easiest hard problem, *American Scientist*, 90 (2), 113-117.

Helsgaun, K., 2000, An effective implementation of the Lin-Kernighan traveling salesman heuristic, *European Journal of Operational Research*, 126 (1), 106-130.

Hill, R. R., Cho, Y. K. ve Moore, J. T., 2012, Problem reduction heuristic for the 0-1 multidimensional knapsack problem, *Computers & Operations Research*, 39 (1), 19-26.

Horowitz, E. ve Sahni, S., 1974, Computing partitions with applications to the knapsack problem, *Journal of the ACM (JACM)*, 21 (2), 277-292.

Huang, L., Wang, G. C., Bai, T. ve Wang, Z., 2017, An improved fruit fly optimization algorithm for solving traveling salesman problem, *Frontiers of Information Technology & Electronic Engineering*, 18 (10), 1525-1533.

Johnson, D. S., Aragon, C. R., Mcgeoch, L. A. ve Schevon, C., 1991, Optimization by Simulated Annealing - an Experimental Evaluation .2. Graph-Coloring and Number Partitioning, *Operations Research*, 39 (3), 378-406.

Jolai, F., Rabiee, M. ve Asefi, H., 2012, A novel hybrid meta-heuristic algorithm for a no-wait flexible flow shop scheduling problem with sequence dependent setup times, *International Journal of Production Research*, 50 (24), 7447-7466.

Karaboga, D. ve Basturk, B., 2007, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, *Journal of Global Optimization*, 39 (3), 459-471.

Karapetyan, D. ve Gutin, G., 2011, Lin-Kernighan heuristic adaptations for the generalized traveling salesman problem, *European Journal of Operational Research*, 208 (3), 221-232.

Karmarkar, N. ve Karp, R. M., 1982, The Difierencing Method of Set Partitioning, *Technical Report UCB/CSD 82/113, Computer Science Division, University of California, Berkeley*.

Ke, L. J., Feng, Z. R., Ren, Z. G. ve Wei, X. L., 2010, An ant colony optimization approach for the multidimensional knapsack problem, *Journal of Heuristics*, 16 (1), 65-83.

KESKİN, F. D., 2015, TEK BOYUTLU KESME PROBLEMİ: BİR İŞLETME UYGULAMASI, *Gazi Üniversitesi İktisadi ve İdari Bilimler Fakültesi Dergisi*, 17 (1), 180-196.

Kiran, M. S., 2015, TSA: Tree-seed algorithm for continuous optimization, *Expert Systems with Applications*, 42 (19), 6686-6698.

Kıran, M. S., İşcan, H. ve Gündüz, M., 2013, The analysis of discrete artificial bee colony algorithm with neighborhood operator on traveling salesman problem, *Neural computing and applications*, 23 (1), 9-21.

Kirkpatrick, S., Gelatt, C. D. ve Vecchi, M. P., 1983, Optimization by simulated annealing, *science*, 220 (4598), 671-680.

Kojic, J., 2010, Integer linear programming model for multidimensional two-way number partitioning problem, *Computers & Mathematics with Applications*, 60 (8), 2302-2308.

Kong, M., Tian, P. ve Kao, Y. C., 2008, A new ant colony optimization algorithm for the multidimensional Knapsack problem, *Computers & Operations Research*, 35 (8), 2672-2683.

Kong, Y., Wang, F., Lim, A. ve Guo, S. S., 2003, A new hybrid genetic algorithm for the robust graph coloring problem, *Ai 2003: Advances in Artificial Intelligence*, 2903, 125-136.

Korf, R. E., 1998, A complete anytime algorithm for number partitioning, *Artificial Intelligence*, 106 (2), 181-203.

Koyutürk, M., 2000, Hypergraph based declustering for multi-disk databases, *Bilkent University*.

Kratica, J., Kojić, J. ve Savić, A., 2014, Two metaheuristic approaches for solving multidimensional two-way number partitioning problem, *Computers & Operations Research*, 46, 59-68.

Laporte, G., 1992, The Traveling Salesman Problem - an Overview of Exact and Approximate Algorithms, *European Journal of Operational Research*, 59 (2), 231-247.

Leguizamon, G. ve Michalewicz, Z., 1999, A new version of ant system for subset problems, *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, 1459-1464.

Liang, K. H., Yao, X., Newton, C. ve Hoffman, D., 2002, A new evolutionary approach to cutting stock problems with and without contiguity, *Computers & Operations Research*, 29 (12), 1641-1659.

Liu, F. ve Zeng, G. Z., 2009, Study of genetic algorithm with reinforcement learning to solve the TSP, *Expert Systems with Applications*, 36 (3), 6995-7001.

Liu, Y. ve Liu, C., 2009, A Schema-Guiding Evolutionary Algorithm for 0-1 Knapsack Problem, *Iacsit-Sc 2009: International Association of Computer Science and Information Technology - Spring Conference*, 160-164.

Ma, J., Yang, T., Hou, Z. G., Tan, M. ve Liu, D. R., 2008, Neurodynamic programming: a case study of the traveling salesman problem, *Neural Computing & Applications*, 17 (4), 347-355.

Mahajan, A. ve Ali, M. S., 2008, Hybrid Evolutionary Algorithm for Graph Coloring Register Allocation, *2008 Ieee Congress on Evolutionary Computation, Vols 1-8*, 1162-+.

Mahmoudi, S. ve Lotfi, S., 2015, Modified cuckoo optimization algorithm (MCOA) to solve graph coloring problem, *Applied Soft Computing*, 33, 48-64.

Maitra, T., Pal, A. J., Bhattacharyya, D. ve Kim, T.-h., 2010, Noise reduction in VLSI circuits using modified GA based graph coloring, *International Journal of Control and Automation*, 3 (2), 37-44.

Makas, H. ve Yumusak, N., 2013, New Cooperative and Modified Variants of the Migrating Birds Optimization Algorithm, *2013 International Conference on Electronics, Computer and Computation (Icecco)*, 176-179.

Marappan, R., Sethumadhavan, G. ve Srihari, R., 2016, New approximation algorithms for solving graph coloring problem–An experimental approach, *Perspectives in Science*, 8, 384-387.

Marinakis, Y., Marinaki, M. ve Migdalas, A., 2016, A hybrid discrete artificial bee colony algorithm for the multicast routing problem, *European Conference on the Applications of Evolutionary Computation*, 203-218.

Marx, D., 2004, Graph colouring problems and their applications in scheduling, *Periodica Polytechnica Electrical Engineering*, 48 (1-2), 11-16.

Meer, K., 2007, Simulated annealing versus metropolis for a TSP instance, *Information Processing Letters*, 104 (6), 216-219.

Meng, T., Pan, Q. K., Li, J. Q. ve Sang, H. Y., 2018, An improved migrating birds optimization for an integrated lot-streaming flow shop scheduling problem, *Swarm and Evolutionary Computation*, 38, 64-78.

Merkle, R. ve Hellman, M., 1978, Hiding information and signatures in trapdoor knapsacks, *IEEE transactions on Information Theory*, 24 (5), 525-530.

Mertens, S., 1998, Phase transition in the number partitioning problem, *Physical Review Letters*, 81 (20), 4281-4284.

Mertens, S., 2006, The easiest hard problem: Number partitioning, *Computational Complexity and Statistical Physics*, 125 (2), 125-139.

Mirjalili, S. ve Lewis, A., 2016, The Whale Optimization Algorithm, *Advances in Engineering Software*, 95, 51-67.

Nawrocki, J. R., Complak, W., Blazewicz, J., Kopczynska, S. ve Mackowiak, M., 2009, The Knapsack-Lightening problem and its application to scheduling HRT tasks, *Bulletin of the Polish Academy of Sciences-Technical Sciences*, 57 (1), 71-77.

Niroomand, S., Hadi-Vencheh, A., Sahin, R. ve Vizvari, B., 2015, Modified migrating birds optimization algorithm for closed loop layout with exact distances in flexible manufacturing systems, *Expert Systems with Applications*, 42 (19), 6586-6597.

Ortega, J. A., Instances of 0/1 Knapsack Problem, http://artemisa.unicauca.edu.co/~johnyortega/instances_01_KP/:

Oz, D., 2017, An improvement on the Migrating Birds Optimization with a problem-specific neighboring function for the multi-objective task allocation problem, *Expert Systems with Applications*, 67, 304-311.

Pan, W. T., 2012, A new Fruit Fly Optimization Algorithm: Taking the financial distress model as an example, *Knowledge-Based Systems*, 26, 69-74.

Parra-Hernandez, R. ve Dimopoulos, N. J., 2005, A new heuristic for solving the multichoice multidimensional knapsack problem, *Ieee Transactions on Systems Man and Cybernetics Part a-Systems and Humans*, 35 (5), 708-717.

Pedroso, J. P. ve Kubo, M., 2010, Heuristics and exact methods for number partitioning, *European Journal of Operational Research*, 202 (1), 73-81.

Poorzahedy, H. ve Rouhani, O. M., 2007, Hybrid meta-heuristic algorithms for solving network design problem, *European Journal of Operational Research*, 182 (2), 578-596.

Pop, P. C. ve Matei, O., 2013, A memetic algorithm approach for solving the multidimensional multi-way number partitioning problem, *Applied Mathematical Modelling*, 37 (22), 9191-9202.

Porumbel, D. C., Hao, J. K. ve Kuntz, P., 2008, A study of evaluation functions for the graph K-coloring problem, *Artificial Evolution*, 4926, 124-+.

Pragya, Dutta, M. ve Pratyush, 2015, TSP Solution Using Dimensional Ant Colony Optimization, *2015 5th International Conference on Advanced Computing & Communication Technologies Acct 2015*, 506-512.

Rao, A. R. M. ve Shyju, P. P., 2008, Development of a hybrid meta-heuristic algorithm for combinatorial optimisation and its application for optimal design of laminated composite cylindrical skirt, *Computers & Structures*, 86 (7-8), 796-815.

Reinelt, G., 1997, TSPLIB, http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsplib.html: [17.01.2018].

Rodriguez, F. J., Glover, F., Garcia-Martinez, C., Marti, R. ve Lozano, M., 2017, GRASP with exterior path-relinking and restricted local search for the multidimensional two-way number partitioning problem, *Computers & Operations Research*, 78, 243-254.

Salcedo-Sanz, S., Xu, Y. ve Yao, X., 2006, Hybrid meta-heuristics algorithms for task assignment in heterogeneous computing systems, *Computers & Operations Research*, 33 (3), 820-835.

Sbihi, A., 2007, A best first search exact algorithm for the Multiple-choice Multidimensional Knapsack Problem, *Journal of Combinatorial Optimization*, 13 (4), 337-351.

Scheithauer, G. ve Terno, J., 1995, The Modified Integer Round-up Property of the One-Dimensional Cutting Stock Problem, *European Journal of Operational Research*, 84 (3), 562-571.

Shah-Hosseini, H., 2008, Intelligent water drops algorithm: A new optimization method for solving the multiple knapsack problem, *International Journal of Intelligent Computing and Cybernetics*, 1 (2), 193-212.

Shi, H. X., 2006, Solution to 0/1 knapsack problem based on improved ant colony algorithm, *2006 IEEE International Conference on Information Acquisition, Vols 1 and 2, Conference Proceedings*, 1062-1066.

Shi, X. H., Liang, Y. C., Lee, H. P., Lu, C. ve Wang, Q. X., 2007, Particle swarm optimization-based algorithms for TSP and generalized TSP, *Information Processing Letters*, 103 (5), 169-176.

Singha, S., Bhattacharya, T. ve Chaudhuri, S. R. B., 2008, An Approach for Reducing Crosstalk in Restricted Channel Routing using Graph Coloring Problem and Genetic Algorithm, *Iccee 2008: Proceedings of the 2008 International Conference on Computer and Electrical Engineering*, 807-+.

Sioud, A. ve Gagne, C., 2018, Enhanced migrating birds optimization algorithm for the permutation flow shop problem with sequence dependent setup times, *European Journal of Operational Research*, 264 (1), 66-73.

Sipahioğlu, A. ve Saraç, T., 2010, Çok amaçlı sırt çantası probleminin çözümüne yeni bir yaklaşım: Konik skalerleştirme, *Endüstri Mühendisliği Dergisi*, 21 (4), 2-12.

Soto, R., Crawford, B., Almonacid, B. ve Paredes, F., 2015, A Migrating Birds Optimization Algorithm for Machine-Part Cell Formation Problems, *Advances in Artificial Intelligence and Soft Computing, Micai 2015, Pt I*, 9413, 270-281.

Soto, R., Crawford, B., Almonacid, B. ve Paredes, F., 2016, Efficient Parallel Sorting for Migrating Birds Optimization When Solving Machine-Part Cell Formation Problems, *Scientific Programming*.

Tanir, D., Ugurlu, O., Guler, A. ve Nuriyev, U., 2016, One-dimensional Cutting Stock Problem with Divisible Items, *arXiv preprint arXiv:1606.01419*.

ThanhVu H. Nguye, T. B., Graph Coloring Benchmark Instances, https://turing.cs.hbg.psu.edu/txn131/graphcoloring.html:

Truong, T. K., Li, K. L. ve Xu, Y. M., 2013, Chemical reaction optimization with greedy strategy for the 0-1 knapsack problem, *Applied Soft Computing*, 13 (4), 1774-1780.

Tuba, M. ve Jovanovic, R., 2013, Improved ACO Algorithm with Pheromone Correction Strategy for the Traveling Salesman Problem, *International Journal of Computers Communications & Control*, 8 (3), 477-485.

Umetani, S., Yagiura, M. ve Ibaraki, T., 2003, One-dimensional cutting stock problem to minimize the number of different patterns, *European Journal of Operational Research*, 146 (2), 388-402.

Vahrenkamp, R., 1996, Random search in the one-dimensional cutting stock problem, *European Journal of Operational Research*, 95 (1), 191-200.

Vance, P. H., 1998, Branch-and-price algorithms for the one-dimensional cutting stock problem, *Computational Optimization and Applications*, 9 (3), 211-228.

Vanderster, D. C., Dimopoulos, N. J., Parra-Hernandez, R. ve Sobie, R. J., 2009, Resource allocation on computational grids using a utility model and the knapsack problem, *Future Generation Computer Systems-the International Journal of Escience*, 25 (1), 35-50.

Vasquez, M. ve Hao, J.-K., 2001, A hybrid approach for the 0-1 multidimensional knapsack problem, *IJCAI*, 328-333.

Vimont, Y., Boussier, S. ve Vasquez, M., 2008, Reduced costs propagation in an efficient implicit enumeration for the 01 multidimensional knapsack problem, *Journal of Combinatorial Optimization*, 15 (2), 165-178.

Wagner, B. J., 1999, A genetic algorithm solution for one-dimensional bundled stock cutting, *European Journal of Operational Research*, 117 (2), 368-381.

Wang, L., Zheng, X. L. ve Wang, S. Y., 2013, A novel binary fruit fly optimization algorithm for solving the multidimensional knapsack problem, *Knowledge-Based Systems*, 48, 17-23.

Wang, L., Zeng, Y. ve Chen, T., 2015, Back propagation neural network with adaptive differential evolution algorithm for time series forecasting, *Expert Systems with Applications*, 42 (2), 855-863.

Wang, X. ve Xu, G. Y., 2011, Hybrid Differential Evolution Algorithm for Traveling Salesman Problem, *Ceis 2011*, 15.

Wang, Z. C., Geng, X. T. ve Shao, Z. H., 2009, An Effective Simulated Annealing Algorithm for Solving the Traveling Salesman Problem, *Journal of Computational and Theoretical Nanoscience*, 6 (7), 1680-1686.

Wilbaut, C., Salhi, S. ve Hanafi, S., 2009, An iterative variable-based fixation heuristic for the 0-1 multidimensional knapsack problem, *European Journal of Operational Research*, 199 (2), 339-348.

Wolpert, D. H. ve Macready, W. G., 1995, No free lunch theorems for search, *Technical Report SFI-TR-95-02-010, Santa Fe Institute*.

Wu, S. G. ve Li, S. K., 2007, Extending traditional graph-coloring register allocation exploiting meta-heuristics for embedded systems, *Icnc 2007: Third International Conference on Natural Computation, Vol 4, Proceedings*, 324-+.

Yang, C. T., Sung, T. C. ve Weng, W. C., 2006, An improved tabu search approach with mixed objective function for one-dimensional cutting stock problems, *Advances in Engineering Software*, 37 (8), 502-513.

Yang, X.-S. ve Deb, S., 2009, Cuckoo search via Lévy flights, *Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*, 210-214.

Yang, X.-S., 2010a, Nature-inspired metaheuristic algorithms, Luniver press, p.

Yang, X. S., 2010b, A New Metaheuristic Bat-Inspired Algorithm, *Nicso 2010: Nature Inspired Cooperative Strategies for Optimization*, 284, 65-74.

Yoon, Y. ve Kim, Y. H., 2013, A Memetic Lagrangian Heuristic for the 0-1 Multidimensional Knapsack Problem, *Discrete Dynamics in Nature and Society*.

Zeng, Y.-R., Peng, L., Zhang, J. ve Wang, L., 2016, An Effective Hybrid Differential Evolution Algorithm Incorporating Simulated Annealing for Joint Replenishment and Delivery Problem with Trade Credit, *International Journal of Computational Intelligence Systems*, 9 (6), 1001-1015.

Zhang, B., Pan, Q. K., Gao, L., Zhang, X. L., Sang, H. Y. ve Li, J. Q., 2017, An effective modified migrating birds optimization for hybrid flowshop scheduling problem with lot streaming, *Applied Soft Computing*, 52, 14-27.

Zhang, X. G., Huang, S. Y., Hu, Y., Zhang, Y. J., Mahadevan, S. ve Deng, Y., 2013, Solving 0-1 knapsack problems based on amoeboid organism algorithm, *Applied Mathematics and Computation*, 219 (19), 9959-9970.

Zhong, W.-h., Zhang, J. ve Chen, W.-n., 2007, A novel discrete particle swarm optimization to solve traveling salesman problem, *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, 3283-3287.

Zhong, Y. W., Wu, C., Li, L. S. ve Ning, Z. Y., 2008, The Study of Neighborhood Structure of Tabu Search Algorithm for Traveling Salesman Problem, *Icnc*

2008: Fourth International Conference on Natural Computation, Vol 1, Proceedings, 491-495.

Zhou, Y. Q., Luo, Q. F., Chen, H., He, A. P. ve Wu, J. Z., 2015, A discrete invasive weed optimization algorithm for solving traveling salesman problem, *Neurocomputing*, 151, 1227-1236.

Zou, D. X., Gao, L. Q., Li, S. ve Wu, J. H., 2011, Solving 0-1 knapsack problem by a novel global harmony search algorithm, *Applied Soft Computing*, 11 (2), 1556-1564.

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı	**:**	Vahit TONGUR
Uyruğu	**:**	T.C.
Doğum Yeri ve Tarihi	**:**	Konya – 19.05.1976
Telefon	**:**	
Faks	**:**	
e-mail	**:**	vtongur@konya.edu.tr, vahittongur@gmail.com

EĞİTİM

Derece		**Adı, İlçe, İl**	**Bitirme Yılı**
Lise	:	Gazi Lisesi, Meram, Konya	1993
Üniversite	:	Selçuk Üniversitesi – Bilgisayar Mühendisliği, Selçuklu, Konya	2001
Yüksek Lisans	:	Selçuk Üniversitesi – Bilgisayar Mühendisliği ABD, Selçuklu, Konya	2008
Doktora	:	Selçuk Üniversitesi – Bilgisayar Mühendisliği ABD, Selçuklu, Konya	2018

İŞ DENEYİMLERİ

Yıl	**Kurum**	**Görevi**
2001	Selçuk Üniversitesi Bozkır MYO	Öğr. Gör.
2011	Necmettin Erbakan Üniversitesi Bilgisayar Mühendisliği Yazılım ABD	Öğr. Gör.

UZMANLIK ALANI

Optimizasyon, Doğa Esinli Algoritmalar, Veritabanı

YABANCI DİLLER

İngilizce

YAYINLAR

Hacibeyoglu, M., Alaykiran, K., Acilar, A. M., Tongur, V. ve Ulker, E., 2018, A Comparative Analysis of Metaheuristic Approaches for Multidimensional Two-Way Number Partitioning Problem, *Arabian Journal for Science and Engineering*, 1-22. (Doktora tezinden yapılmıştır)

Tongur, V. ve Ülker, E., 2014, Migrating birds optimization for flow shop sequencing problem, *Journal of Computer and Communications*, 2 (04), 142. (Doktora tezinden yapılmıştır)

Tongur, V. ve Ülker, E., 2016, The analysis of migrating birds optimization algorithm with neighborhood operator on traveling salesman problem, In: Intelligent and Evolutionary Systems, Eds: Springer, p. 227-237. (Doktora tezinden yapılmıştır)

Tongur, V. ve Ülker, E., 2017a, Migrating Birds Optimization (MBO) Algorithm to Solve Graph Coloring Problem, *International Journal of Engineering Science*, 14545. (Doktora tezinden yapılmıştır)

Tongur, V. ve Ülker, E., 2017b, Migrating Birds Optimization (MBO) algorithm to solve 0-1 multidimensional knapsack problem, *Computer Science and Engineering (UBMK), 2017 International Conference on*, 786-789. (Doktora tezinden yapılmıştır)

Tongur, V. ve Ülker, E., 2018, PSO-based improved multi-flocks migrating birds optimization (IMFMBO) algorithm for solution of discrete problems, *Soft Computing*, 1-16. (Doktora tezinden yapılmıştır)

Ulker, E. ve Tongur, V., 2017, Migrating birds optimization (MBO) algorithm to solve knapsack problem, Procedia computer science, 111, 71-76. (Doktora tezinden yapılmıştır)