T.C.
KONYA TEKNİK ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

GEZGİN SATICI PROBLEMİNİN ÇÖZÜMÜ İÇİN GUGUK KUŞU ARAMA ALGORİTMA TABANLI YENİ BİR HİBRİT METASEZGİSEL YÖNTEM

Mustafa Furkan BERKAYA

YÜKSEK LİSANS TEZİ

Endüstri Mühendisliği Anabilim Dalı

Temmuz-2021
KONYA
Her Hakkı Saklıdır

TEZ KABUL VE ONAYI

Mustafa Furkan BERKAYA tarafından hazırlanan "GEZGİN SATICI PROBLEMİNİN ÇÖZÜMÜ İÇİN GUGUK KUŞU ARAMA ALGORİTMA TABANLI YENİ BİR HİBRİT METASEZGİSEL YÖNTEM" adlı tez çalışması 01/07/2021 tarihinde aşağıdaki jüri tarafından oy birliği ile Konya Teknik Üniversitesi Lisansüstü Eğitim Enstitüsü Endüstri Mühendisliği Anabilim Dalı'nda YÜKSEK LİSANS TEZİ olarak kabul edilmiştir.

Jüri Üyeleri	**İmza**
Başkan Prof. Dr. Orhan ENGİN	……..
Danışman Doç. Dr. Ahmet SARUCAN	……..
Üye Dr. Öğr. Üyesi Kadir BÜYÜKÖZKAN	……..

Yukarıdaki sonucu onaylarım.

Prof. Dr. Saadettin Erhan KESEN
Enstitü 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.

Mustafa Furkan BERKAYA

01.07.2021

ÖZET

YÜKSEK LİSANS TEZİ

GEZGİN SATICI PROBLEMİNİN ÇÖZÜMÜ İÇİN GUGUK KUŞU ARAMA ALGORİTMA TABANLI YENİ BİR HİBRİT METASEZGİSEL YÖNTEM

Mustafa Furkan BERKAYA

Konya Teknik Üniversitesi
Lisansüstü Eğitim Enstitüsü
Endüstri Mühendisliği Anabilim Dalı

Danışman: Doç. Dr. Ahmet SARUCAN

2021, 51 Sayfa

Jüri
Doç. Dr. Ahmet SARUCAN
Prof. Dr. Orhan ENGİN
Dr. Öğr. Üyesi Kadir BÜYÜKÖZKAN

Bu tez çalışmasında, Gezgin Satıcı Problemi (GSP) çözümü için hibrit bir metasezgisel yöntem önerilmiştir. GSP, n tane şehrin bulunduğu ve bu şehirlerin birbirleri arasındaki uzaklıkların belli olduğu, her bir şehrin bir sefer ziyaret edildiği bir problemdir. Problemde en kısa turun bulunarak başlangıç şehrine dönülmesi hedeflenir. Şehir sayısının artması ile problemin optimum çözümü kabul edilebilir sürelerde bulunamamaktadır. Metasezgisel yöntemler, GSP'de kabul edilebilir iyi sonuçlar elde edilebilmesine olanak sağlamaktadır. Önerilen yöntem, 3-Opt yöntemi ve umut vadeden sonuçlar vererek çeşitli çalışmalarla başarısını kanıtlamış Guguk Kuşu Arama algoritmasının birleştirilmesi ile geliştirilmiş hibrit bir yöntemdir. Önerilen yöntemin performansı, GSP için sıkça kullanılan TSPLIB kütüphanesindeki 41 farklı test problemi ile ölçülmüş ve sonuçları literatürdeki son yıllarda geliştirilmiş yöntemler ile kıyaslanmıştır. Önerilen yöntem, 150 şehirden daha küçük boyutlu problemlerin tümünde optimum sonucu bulmuştur. Buna ek olarak, problem boyutu arttıkça diğer yöntemlere kıyasla, elde edilen sonuçların optimuma daha fazla yaklaştığı görülmüştür. Literatürde yer alan çalışmalar ile aynı problem kullanılarak karşılaştırıldığında benzer veya daha üstün sonuçlar elde edilmiştir.

Anahtar Kelimeler: Gezgin satıcı problemi, Guguk kuşu arama, Kesikli guguk kuşu arama Metasezgisel optimizasyon, NP-Zor problem.

ABSTRACT

MS THESIS

A NEW HYBRID METAHEURISTIC METHOD BASED ON CUCKOO SEARCH ALGORITHM FOR SOLVING THE TRAVELING SALESMAN PROBLEM

Mustafa Furkan BERKAYA

Konya Technical University
Institute of Graduate Studies
Department of Industrial Engineering

Advisor: Assoc. Prof. Dr. Ahmet SARUCAN

2021, 51 Pages

Jury
Assoc. Prof. Dr. Ahmet SARUCAN
Prof. Dr. Orhan ENGİN
Asst. Prof. Dr. Kadir BÜYÜKÖZKAN

In this thesis, a hybrid metaheuristic method is proposed to solve the Traveling Salesman Problem (TSP). TSP is a problem in which there are n cities and the distances between these cities are known, and each city is visited once. In the problem, it is aimed to find the shortest tour by returning to the starting city. With the increase in the number of cities, the optimum solution to the problem cannot be found in an acceptable time. Metaheuristic methods allow obtaining acceptable good results in TSP. The proposed method is a hybrid method developed by combining the 3-Opt method and the Cuckoo Search algorithm, which has proven its success in various studies with promising results. The performance of the proposed method was measured with 41 different test problems in the TSPLIB library, which is frequently used for TSP, and the results were compared with the methods developed in the literature in recent years. The proposed method found the optimum result for all problems with a size smaller than 150 cities. In addition, as the problem size increases, it has been observed that the obtained results get closer to the optimum compared to other methods. When compared with the studies in the literature using the same problem, similar or superior results are obtained.

Keywords: Cuckoo search, Discrete cuckoo search, Metaheuristic optimization, NP-Hard problem, Traveling salesman problem.

ÖNSÖZ

Tez çalışmam süresince danışmanlığımı yürüten, desteğini ve rehberliğini esirgemeyen değerli hocam Doç. Dr. Ahmet SARUCAN'a teşekkür ederim.

Çalışmalarım süresince sağlamış oldukları destek ve hoşgörü için değerli mesai arkadaşlarıma teşekkür ederim.

Sevgisi, cesaretlendirmesi, karşılaştığım zorluklarda gösterdiği fedakârlık ve maddi manevi bütün destekleri için biricik eşim Selcan KAPLAN BERKAYA'ya, umudum, canım oğlum Alperen Yasin BERKAYA'ya, her zaman yanımda olan beni yetiştirip bu günlere ulaşmamı sağlayan, üzerimde büyük hakları olan aileme teşekkür eder, minnettarlığımı sunarım.

Mustafa Furkan BERKAYA
KONYA-2021

İÇİNDEKİLER

ÇİZELGELER LİSTESİ

ŞEKİLLER LİSTESİ

SİMGELER VE KISALTMALAR

Simgeler

d_{ij}	: i ve j şehirleri arasındaki uzaklık
D	: Şehirler arası uzaklık matrisi
n	: Şehir sayısı
$\pi(i)$	: i. adımdaki ziyaret edilen şehir
π	: 1'den n'ye kadar şehirlerin tamamını içeren bir tur
p_a	: Yumurtanın ev sahibi kuş tarafından keşfedilme olasılığı oranı
p_c	: Akıllı guguk kuşlarının oranı
α	: Levy uçuşu adım büyüklüğü
F_i	: Uygunluk değeri
T	: Mevcut tur
$f(T)$	: Mevcut turun uzunluğu
T'	: Yeni bulunan tur
$f(T')$	: Yeni turun uzunluğu
z	: Değişim
z_{min}	: Mevcut aramadaki en iyi değişim

Kısaltmalar

YAK	: Yapay Arı Kolonisi
KKO	: Karınca Kolonisi Optimizayonu
KKS	: Karınca Kolonisi Sistemi
KS	: Karınca Sistemi
KD	: Kara Delik
GKA	: Guguk Kuşu Arama
KGKA	: Kesikli Guguk Kuşu Arama
GA	: Genetik Algoritma
GKO	: Gri Kurt Optimizasyonu
GSP	: Gezgin Satıcı Problemi
İKGKA	: İyileştirilmiş Kesikli Guguk Kuşu Arama
MMKS	: Maks-Min Karınca Sistemi
MGKO	: Modifiye Gri Kurt Optimizasyonu
PKKO	: Paralel İşbirlikçi Karınca Kolonisi Optimizayonu
PSO	: Parçacık Sürü Optimizasyonu
KGKAA	: Kuantum esinli Guguk Kuşu Arama Algoritması
TBHHGA	: Tavlama Benzetimi ile Hibrit Hücresel Genetik Algoritma
TA	: Tabu Arama
BOA	: Balina Optimizasyonu Algoritması
DKKBOA	: Değişken Komşuluk ile Kesikli Balina Optimizasyonu Algoritması

1. GİRİŞ

Optimizasyon, çeşitli yazılımların kullanım kolaylığının artması, yüksek hızlı ve paralel işlemcilerin geliştirilmesi ile birlikte bilgisayar teknolojisindeki hızlı ilerleme sayesinde son yıllarda daha da aktif bir araştırma alanı haline gelmiştir.

Optimizasyon, genel anlamıyla kısıtlı kaynak kullanılarak maksimum faydayı elde etmek olarak tanımlanabilir. Karar vermeyi içeren herhangi bir problemin merkezinde optimizasyon yer almaktadır. Karar verme süreci, çeşitli alternatifler arasında seçim yapmayı gerektirir. Bu seçim ile ulaşılmak istenen, en iyi kararı vermektir. Alternatiflerin en iyisinin ölçülmesi için bir amaç fonksiyonu tanımlanır. Optimizasyon yöntemlerinde, verilen amaç fonksiyonu doğrultusunda en iyi alternatif seçilir. Bu yöntemler ile karar verme süreçleri hızlanır ve karar kalitesi artırılır. Bu sayede gerçek hayatta karşılaşılan problemler etkin ve doğru bir şekilde gerçek zamanlı olarak çözülmüş olur (Chong ve Zak, 2004).

Gezgin Satıcı Problemi (GSP) yaygın olarak karşılaşılan ve üzerinde çok çalışılan optimizasyon problemlerinden biridir. Bu problem, farklı problemlerin GSP olarak modellenebilmesi sayesinde birçok alanda kullanılabilmektedir. Günümüzde dünyayı etkisi altına alan COVID-19 pandemisi nedeniyle birçok sektörde değişiklik yaşanmış ve yaşanmaya da devam etmektedir. Başta üretim ve dağıtım süreçlerinin işleyişi olmak üzere, lojistik süreçler önemini oldukça artırmıştır. GSP'nin bu alana modellenmesi ile de problemin etkili bir şekilde ve gerçek zamanlı olarak çözülmesi yeni geliştirilecek olan yöntemler sayesinde sağlanabilir.

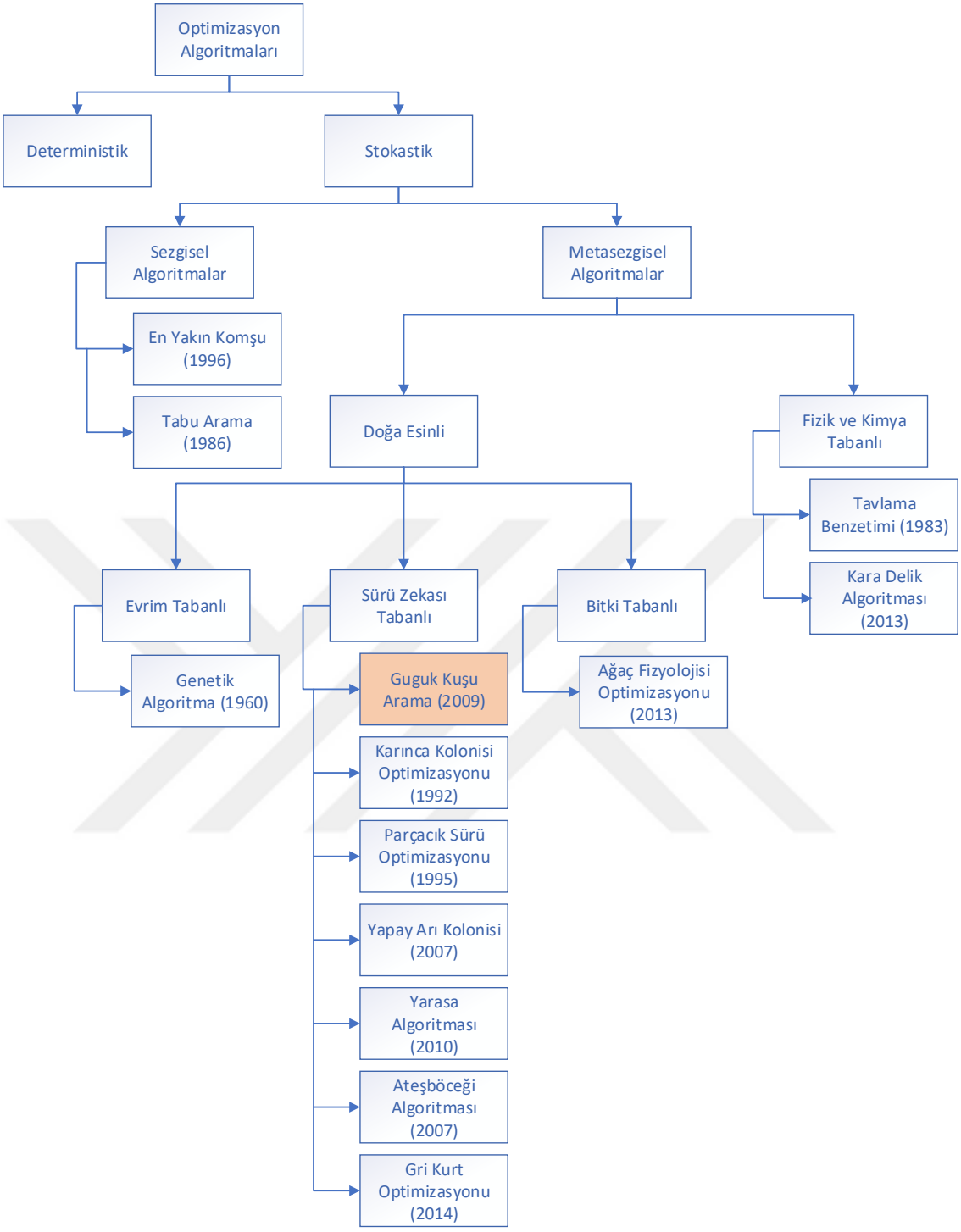
Bu problemi çözmek amacıyla birçok araştırmacı yeni ve farklı algoritmalar uygulamaktadır. GSP'nin tanımı, anlaşılması ve formüle edilmesi basit olmasına rağmen, özellikle değişken sayısının artması ile birlikte kabul edilebilir süreler içerisinde optimum çözümü elde etmek veya bu çözüme yaklaşmak zorlaşmaktadır. Problemin çözümünde geleneksel yöntemlerin kullanılması, artan değişken sayısıyla birlikte yetersiz kalmaktadır.

Özellikle karmaşık optimizasyon problemlerini çözmek için kullanılan sezgisel yöntemler, geleneksel yöntemler çok yavaş olduğunda bir problemi daha hızlı çözmek veya geleneksel yöntemler herhangi bir kesin çözüm bulamadığında yaklaşık bir çözüm bulmak için tasarlanmış yöntemlerdir. Genellikle doğada meydana gelen olaylardan veya biyolojik sistemlerden esinlenen problem çözmeye yönelik yaklaşımlardır. Metasezgisel algoritmalar, arama uzayını daha etkili bir şekilde keşfetmek için arama

stratejileri kullanır ve genellikle arama uzayının bazı umut vadeden bölgelerine odaklanır. Bu yöntemler bir dizi başlangıç çözümü veya bir başlangıç popülasyonu ile başlar ve daha sonra, ilgilenilen problemin optimal çözümüne ulaşmak veya yakınlaşmak için bir dizi çözümü adım adım inceler. Metasezgisel yöntemler ile GSP'nin çözümünde kabul edilebilir iyi sonuçlar elde edilebilmektedir. Birçok yönüyle probleme adapte edilerek sonuçların daha iyiye gelmesini sağlayan metasezgisel yöntemler arasında Guguk Kuşu Arama (GKA), Karınca Kolonisi Optimizasyonu (KKO) ve geliştirilmiş türevleri, Tabu Arama (TA), Yapay Arı Kolonisi (YAK), Parçacık Sürü Optimizasyonu (PSO), Genetik Algoritma (GA), Gri Kurt Optimizasyonu (GKO), Kara Delik (KD) algoritması gibi algoritmalar yer almaktadır. Bu yöntemler, özet olarak Şekil 1.1'de verilmiştir.

Bu tez çalışmasında, GSP'nin çözümü için, diğer başka uygulamalar ve veri setlerinde yüksek performans sağlayan sezgisel yöntemlerden biri olan GKA algoritmasına dayalı yeni bir hibrit yöntem önerilmiştir. Önerilen yöntemin performansını ölçmek amacıyla, bu problem için literatürde sıkça kullanılan TSPLIB (Reinelt, 1991) kütüphanesindeki sık kullanılan problemler ele alınarak, literatürdeki diğer metasezgisel yöntemlerle karşılaştırılmıştır. Özellikle değişken sayısının artmasıyla birlikte diğer karşılaştırılan yöntemlerde bulunan optimum çözüm veya bu çözüme yaklaşan çözüm sayısı azalırken, önerilen yöntem sayesinde ele alınan problemlerde daha iyi performans elde edilmiştir.

Bu tez çalışmasının planı şu şekildedir: Bölüm 2, GSP'yi ve uygulama alanlarını tanıtmaktadır. Bölüm 3, son yıllarda literatürde bu alanda yapılan çalışmalar ve yaygın olarak kullanılan yöntemler hakkında bilgi vermektedir. Bölüm 4'te, kullanılan veri seti detaylı bir şekilde açıklanmakta ve önerilen yöntem tanıtılmaktadır. Yapılan kapsamlı deneysel çalışmalar ile elde edilen sonuçlar ve bu sonuçların literatürdeki son yıllarda geliştirilen yöntemler ile karşılaştırılması Bölüm 5'te verilmiştir. Son olarak, sonuçlar ve çalışma ile ilgili genel değerlendirmeler Bölüm 6'da yer almaktadır.



Şekil 1.1. Optimizasyon algoritmaları (Halim ve Ismail, 2019)

2. GEZGİN SATICI PROBLEMİ

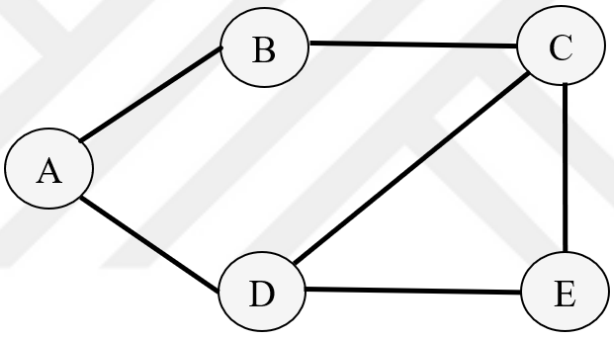
2.1. Çizge Teorisi

Çizge, düğümler ve bu düğümleri birbiri ile birleştiren bağlantılardan oluşan bir yapıdır. Bu yapı geometrik veya konumsal bir bilgi vermez. Bir G çizgesi, (V, E) ikilisinden oluşur. Burada V düğümler kümesi, E ise bağlantılar kümesi olarak tanımlanmaktadır (West, 2001). Her bağlantı, V düğümler kümesinde yer alan v_i ve v_j gibi bir çift düğümden oluşur ve (v_i, v_j) şeklinde gösterilir. Şekil 2.1'de örnek bir çizge verilmiştir. Bu G çizgesine ait düğümler ve bağlantılar kümesi aşağıdaki gibidir:

G = (V, E)

V = {A, B, C, D, E}

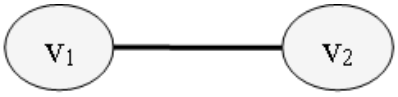
E = {(A, B), (A, D), (B, C), (C, D), (C, E), (D, E)}



Şekil 2.1. Örnek bir çizge

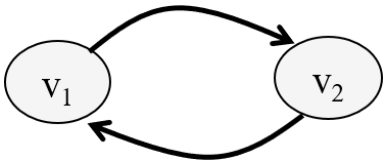
2.1.1. Yönlü ve yönsüz çizgeler

Bir çizgede düğümler arasındaki bağlantılar yön bilgisine sahip değilse, yönsüz çizge olarak tanımlanır. Buradaki yön bilgisi bağlantının başlangıç ve bitiş düğümlerini ifade eder. Çizge yapısında tüm bağlantılar aynı türden olmalıdır. Şekil 2.2'de yönsüz çizgeye bir örnek verilmiştir. Şekil 2.2'de görüldüğü gibi, iki düğümden oluşan bu yönsüz çizge için v_1 ve v_2 düğümleri arasında tek bir bağlantı vardır ve bu bağlantı başlangıç ve bitiş düğüm sırasına bakılmaksızın birbirine eşittir (v_1, v_2) = (v_2, v_1).



Şekil 2.2. Örnek bir yönsüz çizge

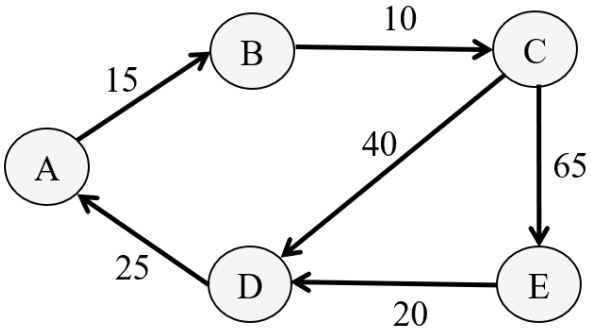
Eğer bir çizgede düğümler arasındaki bağlantılar yön bilgisine sahipse, yönlü çizge olarak adlandırılır. Şekil 2.3'te yönlü çizgeye bir örnek verilmiştir. Şekil 2.3'te görüldüğü gibi, yönlü çizge için v_1'den v_2'ye olan bağlantı v_2'den v_1'e olan bağlantıya eşit değildir $(v_1, v_2) \neq (v_2, v_1)$.



Şekil 2.3. Örnek bir yönlü çizge

2.1.2. Ağırlıklı çizgeler

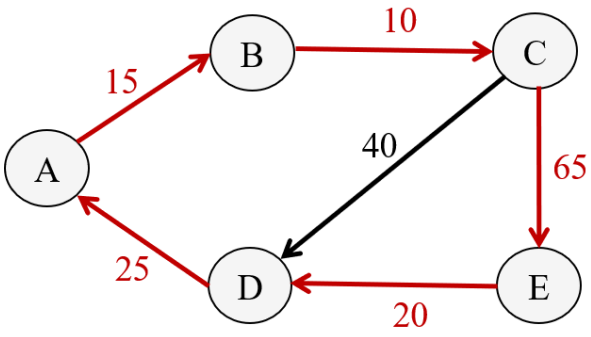
Eğer bir çizgede düğümler arasındaki bağlantılar ağırlık bilgisine sahipse, ağırlıklı çizge olarak adlandırılır. Ağırlıklı çizge olabilmesi için aradaki bağlantıların değerleri farklı olmalıdır. Burada ağırlık bilgisinin anlamı uygulama alanına göre değişiklik gösterir. Örneğin ağırlık, bir uygulamada şehirler arası uzaklığa eşit olabileceği gibi, başka bir uygulamada bant genişliğine karşılık gelebilir. Şekil 2.4'te ağırlıklı çizgeye bir örnek verilmiştir.



Şekil 2.4. Örnek bir ağırlıklı çizge

2.1.3. Hamilton turu

19. yüzyılda yaşamış bir matematikçi olan William Hamilton, her bir düğümün tam bir tur oluncaya kadar sadece bir sefer kullanılmasıyla tamamlanan, başlangıç düğümüne ulaşılmasıyla sona eren çizgeyi Hamilton turu olarak tanımlamıştır (West, 2001). Şekil 2.5'te, Şekil 2.4'te verilen çizgeye ait örnek bir Hamilton turu verilmiştir. GSP, Hamilton turunun kullanıldığı en yaygın problemlerden biridir.



Şekil 2.5. Örnek Hamilton turu

2.2. Problem Tanımı

GSP, *n* tane şehrin bulunduğu ve bu şehirlerin birbirleri arasındaki uzaklıkların belli olduğu, her bir şehrin bir sefer gezgin satıcı tarafından ziyaret edildiği bir problemdir (Mahi ve ark., 2015). Problemde en kısa rotanın bulunarak başlangıç şehrine dönülmesi hedeflenir. Gezgin satıcı, ilk şehir seçimini *n* tane farklı şehir arasından yapabilir. İkinci şehre geldiğinde, *n*-1 tane farklı şehir arasından seçim yapabilecektir. Bu şekilde sonuncu şehre kadar devam ettiğinde toplamda *n*! tane tur arasından seçim yapabilir. Örnek olarak, 50 şehir içeren bir problemde olası tur sayısı $3{,}04\times10^{64}$ olacaktır.

Karmaşıklık, şehir sayısıyla birlikte üssel olarak arttığı için problem NP-Zor sınıfında yer almaktadır. Genel olarak NP-Zor sınıfında yer alan problemler, kabul edilebilir bir zaman ölçeğinde bilinen herhangi bir algoritma tarafından verimli bir şekilde çözülemez (Gutin ve Punnen, 2006). Bu sınıfta yer alan birçok problemin tanımlanması ve anlaşılması basit olmasına rağmen, optimum çözümün bulunması alternatif çözümleri oluşturan kombinasyonların sayısı ilgilenilen problemin boyutu ile katlanarak arttığı ve her olası kombinasyonu aramanın uzun zaman alması nedeniyle hesaplama açısından gerçekçi değildir.

GSP ve varyasyonları, gezgin satıcının rota planlama probleminin yanı sıra matematik, bilgisayar bilimi, genetik, mühendislik ve elektronik dahil olmak üzere çeşitli alanlarda uygulanabilir. GSP çözümü birçok farklı ve önemli uygulama alanında kullanılmaktadır (Gutin ve Punnen, 2006). Bu alanlardan bazıları aşağıdaki şekilde sıralanabilir:

- Donanım (devre) tasarımı,
- Telekomünikasyon,

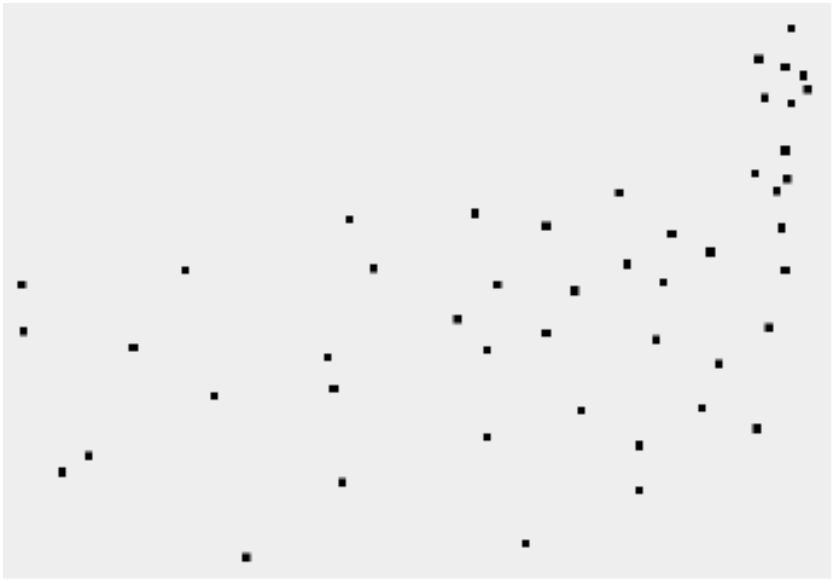
- Bilgisayar ağları,
- Üretim sistemlerinde robotların kontrolü,
- X-ışını kristalografisi,
- Çizelgeleme,
- Araç rotalama problemleri

GSP, matematiksel olarak n tane şehir ve her iki şehir arasındaki uzaklığı gösteren $D = (d_{ij})_{n\times n}$ uzaklık matrisi ile tanımlanmaktadır. Uzaklık değeri d_{ij} Denklem 2.1 ile hesaplanır. Problemde amaç, daha önce de bahsedildiği gibi en kısa turun bulunmasıdır. Eğer $\pi(i)$ i. adımdaki ziyaret edilen şehir olarak ele alınırsa, 1'den n'ye kadar şehirlerin döngüsel permütasyonları $\pi=(\pi(1), \pi(2),\ldots, \pi(n))$ bir tur olarak ifade edilebilir. Bir permütasyonun (turun) uzunluğu Denklem 2.2 ile tanımlanmaktadır.

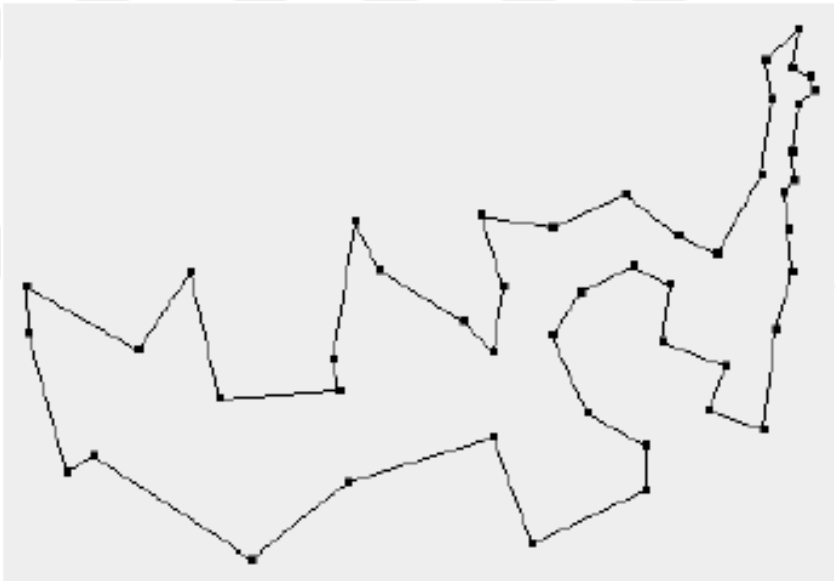
$$d_{ij} = \sqrt{(x_i - x_j)^2+(y_i - y_j)^2} \tag{2.1}$$

$$f(\pi) = \sum_{i=1}^{n-1} d_{\pi(i)\pi(i*1)} + d_{\pi(n)\pi(1)} \tag{2.2}$$

Eğer $1\leq i$ ve $j\leq n$ için $d_{ij}=d_{ji}$ ise, bu durumda problem, simetrik GSP olarak adlandırılmaktadır. GSP örneği Şekil 2.1'de verilmiştir. Her bir GSP çözümü bir Hamilton turuna karşılık gelir (Şekil 2.2) ve GSP için optimum çözüm en kısa Hamilton turu olur (Gutin ve Punnen, 2006).



Şekil 2.6. GSP örneği (Shahab, 2019)



Şekil 2.7. GSP çözüm örneği (Shahab, 2019)

3. KAYNAK ARAŞTIRMASI

GSP ve varyasyonları için etkin bir algoritma henüz bulunamamıştır. Bu problemlerde optimum olmasa bile optimuma yakın iyi çözümleri hızlı bir şekilde bulma ihtiyacı, metasezgisel gibi çeşitli algoritmaların geliştirilmesine yol açmıştır. Bununla birlikte metasezgisel algoritmalar, çok çeşitli optimizasyon problemlerinin çözümünde potansiyellerini ve etkinliklerini göstermiştir ve geleneksel algoritmalara göre avantajlıdırlar (Quaarab ve ark., 2014).

GSP için kullanılan çözüm yöntemleri ikiye ayrılmaktadır. Bunlardan ilki dal-sınır ve dal-kesme algoritmalarını içeren deterministik algoritmalar, diğeri ise sezgisel ve metasezgisel algoritmaları içeren stokastik algoritmalardır. Bu sezgisel algoritmalar; en yakın komşuluk yöntemi, açgözlü algoritma gibi tur oluşturan sezgiselleri, KKO, YAK, PSO, k-Opt, TA, GA gibi yöntemleri içeren tur geliştiren sezgiselleri ve birden fazla yöntemin güçlü yönlerinin birleşiminden faydalanılarak geliştirilen hibrit yöntemleri içermektedir.

Literatürde bu alanda yapılan çalışmalara ait detaylı bilgiler Çizelge 3.1'de verilmiştir. Çizelgede verilen veri setlerindeki optimum sonuç bulunan problemler koyu renk ile işaretlenmiştir.

Çizelge 3.1. Literatür taraması

Kaynak	Yöntem	Veri Seti	Açıklama
Karaboğa ve Görkemli (2011)	Kombinatoryal YAK	kroB150, kroA200	Önerdikleri yöntemde geliştirilmiş bir mutasyon operatör, işçi ve gözcü arıların komşuluk seçme mekanizmasına adapte edilmiştir.
Al-Obaidi (2013)	Dağınık arama, GKA-Dağınık arama	fri26, dantzig42, att48, eil51, eil101, kroa100, kroB100, kroC100, kroD100, kroE100, kroB200, lin105, lin318, pr76, pr124, pr299, pr439, pr1002, nrw1379, berlin52, bier127, a280	Standart dağınık arama ve GKA ile geliştirilmiş dağınık arama kıyaslaması sonucunda, hibrit yöntem %23,2 oranında daha iyi sonuç vermiştir.
Peker ve ark. (2013)	KK Sistemi (KKS)	**berlin52, eil51,** eil76**,** pr76**,** kroA100, kroC100**,** eil101, lin105**,** pr1002, d1291	Bu çalışmada, GSP için KKS kullanılarak bir çözüm önerilmiş ve Taguchi yönteminden parametre optimizasyonu alınmıştır. Parametrelerin sistem üzerindeki etki değerlerini belirlemek için varyans analizi yapılmıştır.

Çizelge 3.1. Literatür taraması (devam)

Kaynak	Yöntem	Veri Seti	Açıklama
Blazinskas ve ark. (2013)	Modifiye yerel arama	bier127, ch130, d657, dsj1000a, eil101, fl417, gil262, gr120, gr137, gr22, gr431, lin318, pcb442, pr107, pr264, pr439, rat783, rd100, rd400, si535, **gr96, kroA100, kroB100, kroC100, kroD100, kroE100, lin105, pr124, rat99, tsp225**	Değişiklikler esas olarak genişletilmiş komşu yapılarının kullanılmasından kaynaklanmaktadır. Yinelenen yerel arama yönteminin performansı da incelenmiştir.
Quaarab ve ark. (2014)	Kesikli GKA (KGKA), İyileştirilmiş KGKA (İKGKA)	kroA200, lin318**, eil51, berlin52, st70, pr76, eil76, kroA100, kroB100, eil101, bier127, ch130, ch150, kroa150**	Bu çalışmada, standart guguk kuşu araması algoritması yeniden yapılandırılmış ve daha akıllı olan yeni bir guguklu kategorisi sunarak Levy uçuşları aracılığıyla genişletilmiştir. İKGKA simetrik GSP çözümü için kullanılmıştır. Performansı literatürde yer alan 2 farklı yöntem ile karşılaştırılmıştır. Karşılaştırmanın sonuçları, İKGKA'nın GSP'yi çözmek için diğer iki yöntemden daha iyi performans sunduğunu göstermiştir.
Saenphon ve ark. (2014)	KKO ile hızlı karşıt gradyan araması	att48, berlin52, st70, eil76, pr76, kroa100, rd100, eil101, kroa200, **eil51**	Bu çalışmada önerilen yöntem TA, GA, PSO, KKO, PSO–KKO, GA–PSO–KKO ile karşılaştırılmıştır. Sonuçlar, algoritmanın daha az iterasyonda her durumda daha kısa mesafeler elde ettiğini göstermiştir.
Karagül (2014)	GKA	Sırbistan'ın Niş şehrindeki 20 bölge	Sırbistan'ın Niş şehrindeki 20 bölgeye konumlandırılmış plastik atık konteynerlerinde yer alan atıkların kamyonlarla toplanmasına ilişkin problem ele alınmıştır.
Escario ve ark. (2015)	Genişletilmiş KK	ry48p, eil51, st70, ftv70, pr76, rat99, kroA100, eil101, gr137	Genişletilmiş KK yönteminde devriye gezenler ve toplayıcılar olmak üzere iki tür karınca vardır. Bunların kendi aralarında görev bölüşümü yapmak ve arama işlemi sırasında her bir tür karıncanın sayısını kontrol etmek için bir düzenleme politikası yürütmek gibi iki önemli özelliği vardır. Ek olarak, klasik KKO tarafından kullanılan yapı grafiğini kullanmaz. Bunun yerine, arama bir durum uzayı keşif yaklaşımı kullanılarak gerçekleştirilir. Yöntem test edilen neredeyse her GSP örneğinde KKS ve Maks – Min Karınca Sisteminden (MMKS) daha iyi performans göstermektedir.

Çizelge 3.1. Literatür taraması (devam)

Kaynak	Yöntem	Veri Seti	Açıklama
Ansari ve Katiyar (2015)	GA, KKO, KKO-GA, KKO-GKA	6 özgün test problemi	KKO, GA ve GKA algoritması ile birleştirilerek, performansı artırmak ve daha iyi sonuçlar elde etmek için hibrit algoritmalar geliştirilmiştir. Bu çalışmada kullanılan algoritmalar karşılaştırıldığında, KKO-GA'nın GA ve KKO-GKA algoritmasına göre daha iyi sonuç verdiği sonucuna varılmıştır. Sezgisel bilgi kullanan yöntemlerin hızlı oran ve yakınsama açısından iyi sonuçlar verdiği sonucuna varılmıştır.
Laha (2015)	Kuantum esinli GKA Algoritması (KGKAA)	berlin52, st70, eil76, rat99, rd100, kroA100, kroB100, kroC100, kroD100, kroE100, eil101, lin105, pr107, bier127, ch130, pr144, ch150, kroA150, kroB150, pr152, u159, rat195, d198, **eil51, pr76, pr124**	Bu çalışmada, kuantum hesaplama kavramı ve GKA metasezgisel arama algoritması birleştirilerek yeni bir hibrit algoritma önerilmiştir. Bu yaklaşım, diğer önemli evrime dayalı algoritmalardan daha iyi veya bunlara yaklaşık çözümler ürettiğini göstermiştir.
Mahi ve ark. (2015)	PSO–KKO–3Opt	rat99, st70, kroA200, ch150, **eil51, berlin52, eil76, kroA100, lin105, eil101**	GSP çözümü için PSO, KKO ve 3-Opt algoritmasını içeren yeni bir hibrit yöntem önerilmiştir. Bu yöntemde, PSO, KKO'nun performansını etkileyen alfa ve beta parametrelerini belirlemek için kullanılır ve 3-Opt, KKO'da bulunan yerel çözümden kurtulmak için kullanılır. Önerilen bu yöntemin performansı ortalama değer, ortalama rota uzunluğu, standart sapma ve yüzde bağıl hata değerleri dikkate alınarak ölçülmüştür. Yazarlar çalışmalarında, farklı sayıdaki karıncaların KKO'daki etkilerini de analiz etmiştir. Deney sonuçlarına göre önerilen yöntemin performansı, daha az sayıda karıncaya bağlı olarak daha iyi hale gelmektedir. Bu çalışmada elde edilen sonuçlar, önerilen yöntemin performansının, karşılaştırılan diğer yöntemlere kıyasla daha iyi veya benzer olduğunu göstermektedir.
Munlin ve Anantathanavit (2015)	Böl-Yönet stratejisi	**burma14, bayg29, berlin52**	K-ortalamalar algoritması, şehirleri kümelemek ve daha sonra PSO ile sırası verilen bir alt şehir dizisini çözmek için kullanılır. PSO, yerel optimuma takılmaması için genetik algoritma operatörleri, yani mutasyon dahil edilerek değiştirilmiştir.

Çizelge 3.1. Literatür taraması (devam)

Kaynak	Yöntem	Veri Seti	Açıklama
Quaarab ve ark. (2015)	Rasgele anahtarlı GKA	pr136, ch130, rat195, **eil51, berlin52, st70, pr76, eil76, kroA100, eil101, bier127, pr144 , rd100, pr124**	Önerilen bu yaklaşımda, GSP'yi çözmek için rasgele anahtarlı GKA algoritması geliştirmek için GKA ile birlikte rastgele anahtar kodlama şeması kullanılmıştır. GKA'nın iyi bir arama mekanizması sağlamasına olanak sağlayan sürekli ve birleşik arama alanı arasında geçiş yapmak için rasgele anahtarlar kullanılmıştır. Performansı GA, KKO, PSO vb. yöntemlerle karşılaştırılmıştır. Önerilen yöntemin iyileştirmeye açık birçok yönü vurgulanmıştır.
Odili ve Kahar (2016)	Afrika Bufalo Optimizasyonu	st70, pr76, kroA100, eil101, ch150, tsp225. att48, pr152, rd400, pr1002, d1291, fn14461, **berlin52, eil76, gil262, brd14051**	Yöntem; PSO, KKO, YAK, MMKS ve birçok farklı hibrit algoritma ile karşılaştırılmıştır. Önerilen yöntem, diğer 11 yöntemden daha hızlı ve optimum veya optimuma yakın çözümler elde etme kapasitesi yüksektir.
Ismkhan (2017)	KKO için etkili sezgiseller	fl1577, fl3795, fnl4461, rl5915, pla7397, rl11849, usa13509, brd14051, b15112, b18512, **eil51, eil76, kroA100, lin105, d198, kroA200, a280, lin318, pcb442, att532, rat783, pr1002**	Çalışma, büyük ölçekli problemlerin üstesinden gelmek için KKO için etkili sezgisel yöntemleri sunmaktadır. KKO'nun mevcut varyasyonlarının bazı dezavantajları vardır. Bunlar: (1) Feromonu ana bellekte tutmanın karmaşıklığı yüksektir ve bu sorunu çözmek için mevcut en ünlü çözüm, birçok durumda çok esnek olmayan aday feromon değerlerini kullanmaktır. (2) KKO'nun mevcut varyasyonları, temel algoritmalarında yerel arama algoritmalarını kullanır, ancak daha verimli çalışmak için feromon bilgisini kullanamazlar. (3) Bir sonraki hamleyi seçmenin zaman karmaşıklığı yüksektir. Önerilen yöntem 20,000'e kadar olan büyük örneklerde kullanılabilmektedir. Diğer KKO'lar, şehir sayısı 1,500'den büyük olduğunda nadiren uygulanabilmektedir. KKO ile kıyaslandığında önerilen yöntemin daha iyi performans elde ettiği görülmüştür.
Tian ve ark. (2017)	Kesik genelleştirilmiş beta dağılımı	ulysses22, berlin52, pr76,rat99, **kroA100, pr299, lin318, rd400, d493, rat575**	GSP çözümü için optimum tur uzunluklarının olasılık dağılımı için yinelemeli kesik genelleştirilmiş beta dağılımı kullanılmıştır. Sonuçta, *K* değeri arttıkça gerçek optimuma yaklaşılabildiği bulunmuştur.

Çizelge 3.1. Literatür taraması (devam)

Kaynak	Yöntem	Veri Seti	Açıklama
Kefi ve ark. (2017)	KS, Elitist K, En iyi-En kötü KS, MMKS, KKS	ch150, kroA200, **eil51, berlin52, eil76, st70**	Beş farklı iyi bilinen KKO algoritması kullanılmıştır. Ek olarak, çalışmada, sabit parametre ayarı ile yerel arama olmadan KKO ile karşılaştırıldığında en iyi çözümü elde etmek için yerel arama yaklaşımı ile 2-Opt ve 3-Opt algoritmaları ile birleştirilmiş KKO yaklaşımı açıklamaktadır.
Gulcu ve ark. (2018)	Paralel İşbirlikçi Karınca Kolonisi Optimizayonu (PKKO)-3Opt	rat99, eil76, st70, kroa100, lin105, kroA200, ch150, eil101, **eil51, berlin52**	Çalışmada, KKO algoritması GSP'nin çözümlerini üretir ve 3-Opt algoritması bu çözümleri geliştirir. Her koloni, algoritmayı bağımsız olarak çalıştırır ve kendi en iyi çözümünü periyodik olarak birbirleriyle paylaşır. Böylece, algoritma nispeten iyi sonuçlar elde edebilir. Önerilen yöntemin, dağıtılmış bir sistem üzerinde paralel çalışma kabiliyeti vardır.
Hatamlou (2018)	KD	ulysses22, bays29, bayg29, att48, eil51, berlin52, st70, eil76, gr96, eil101	Kara delik olgusundan esinlenen popülasyon tabanlı bir metasezgisel algoritma önerilmiştir. Deneysel sonuçlar, KD algoritmasının GA, KKO ve PSO algoritmalarına kıyasla yüksek kaliteli çözümleri daha hızlı bulabildiğini göstermektedir.
Sopto ve ark. (2018)	Modifiye GKO (MGKO)	gr21, ulysses22, gr24, fri26, bays29, hk48, eil51, berlin52, st70, eil76, gr96, kroA100, **burma14, ulysses16, gr17**	Önerilen yöntem, GA, KKO ve PSO ile karşılaştırıldığında çoğu durumda daha iyi bir performans göstermiştir.
Faris ve ark. (2018)	GKO	-	GKO farklı versiyonlarıyla, öznitelik seçimi, yapay sinir ağlarının eğitimi, destek vektör makinesinin optimize edilmesi ve kümeleme uygulamaları gibi makine öğrenmesi uygulamaları, kontrolörlerin tasarımı, güç dağıtımı, robotik ve yol planlama, çizelgeleme ve benzeri mühendislik uygulamaları, kablosuz sensor ağları, çevre modelleme uygulamaları, görüntü işleme, medikal ve biyoinformatik uygulamalar gibi birçok alanda kullanılmaktadır.

Çizelge 3.1. Literatür taraması (devam)

Kaynak	Yöntem	Veri Seti	Açıklama
Ekmekçi (2019)	En iyi çözümleri hatırlayan KKO	**eil51, kroA100**	Önerilen yöntem, kombinatoryal problemler için düğümler arasındaki feromon haritasını görebilme avantajını sağlar.
Halim ve İsmail (2019)	En yakın komşu, Tavlama benzetimi, TA, GA, KKO ve Ağaç fizyolojisi optimizasyonu	eil51, berlin52, st70, ril76, pr76, rat99, kroa100, eil101, ch130, ch150, rat195, d198, a280, rd400, pcb442	GSP için 6 farklı sezgisel algoritma karşılaştırılmıştır. TA diğer algoritmalara kıyasla 6 kat, KKO diğer algoritmalara kıyasla 3 kat daha uzun işlem süresi gerektirir. En yakın komşu en düşük hesaplama süresine sahiptir. İstatistiksel karşılaştırmada optimum rotaya en yakın noktaya ulaşan algoritmalar TA, ağaç fizyolojisi optimizasyonu ve GA şeklindedir. Bununla birlikte, TA'nın doğruluğu daha büyük problem boyutları ($n \geq 400$) için azalır. GA'daki mutasyon, olası çözümü daha iyi sonuca götüren önemli bir faktördür. Tavlama benzetimi diğer algoritmalar arasında en yavaş yakınsama dinamiğine sahiptir. Ağaç fizyolojisi optimizasyonu daha küçük boyutlarda daha iyi sonuçlara yaklaşmaktadır.
Sahana (2019)	İçiçe Hibrit KKS	berlin 52, eil51, eil76, eil101, kroA100, kroB100, kroC100, kroD100, kroE100, bier127	Önerilen yöntem, GA ve sezgisel tarama ile KKS'yi birleştirmiştir. Az sayıda şehir için, kesin algoritmalardan daha iyi performans gösterir, aynı zamanda küçük ve çok sayıda şehir için sezgisel ve yaklaşık algoritmalardan daha iyi performans göstermektedir.
Thirugnanasam bandam (2019)	KS	att48, bays20, berlin52, eil51, eil76, pr76, st70	Parametrelerin sonuçlara etkisi incelenmiştir. Karınca sayısı, feromon buharlaşma miktarı, alfa ve beta parametreleri ile farklı setler oluşturup sonuçlar gözlemlenmiştir.
Yu (2019)	KKO-PSO	Çin'in Henan Eyaletindeki 17 şehir	Bu çalışmada, KKO'nun global araması PSO algoritması ile geliştirilmiş ve karınca kolonisi yerel aramasına belirli bir ağırlık verilerek Geliştirilmiş KKO elde edilmiştir. Lojistik dağıtım sürecine örnek olarak Çin'in Henan Eyaletindeki 17 şehir alınmıştır. Deneysel sonuçlar, Geliştirilmiş KKO'nun etkili ve uygulanabilir olduğunu göstermektedir.

Çizelge 3.1. Literatür taraması (devam)

Kaynak	Yöntem	Veri Seti	Açıklama
Khan ve Maiti (2019)	YAK-8 güncelleme kuralı ve k-Opt	eli51, kroA200, ftv56**,gr17,bays29 swiss42, berlin52, st70, eli76, rat99, kroA100, eli101, lin105, pr124, pr152br17, ftv33, ry48)**	YAK ile 2-Opt ve 3-Opt yöntemleri ayrı ayrı birleştirilmiştir. YAK-3Opt yönteminin performansının daha iyi olduğu görülmüştür. Deneyler için simetrik ve asimetrik GSP'ler kullanılmıştır.
Shahab (2019)	Sezgisel algoritma	eil76, rat99, kroB100, eil101, bier127, ch130, pr136, gr137, ch150, kroA150,kroB150, si175, brg180, rat195, kroA200 ,kroB200 ,gr202, ts225 ,tsp225, gr229, gil262, pr299, lin318, rd400, fl417, gr431 , pr439 , pcb442, d493, att532, ali535 , si535, pa561 , u574, rat575 , p654 , d657, gr666 , u724, rat783 , dsj1000, pr1002 ,si1032 , u1060, **burma14 ,ulysses16, gr17 , gr21 , ulysses22 , gr24 , fri26 , bayg29 ,bays29 , dantzig42, swiss42, att48 , gr48,hk48, eil51, berlin52, brazil58, st70, , pr76, gr96, , kroA100, , kroC100 , kroD100, kroE100, rd100 ,, lin105 , pr107, gr120, pr124, pr144, 2 pr152, u159 , , d198 , pr226, pr264, a280**	Önerilen sezgisel algoritmanın sonuçları K değerine bağlıdır. Çalışmada sadece varsayım kullanılarak K ve I değerleri belirlenmiştir. Önerilen yöntem toplamda 80 TSPLIB test problem üzerinde denenmiş, 36 problemde başarılı sonuçlar elde edilmiştir.
Dokeroglu ve ark. (2019)	YAK, Bakteriyel Toplama, Yarasa Algoritması, Biyocoğrafya tabanlı optimizasyon, GKA, Ateşböceği Algoritması, Yerçekimsel Arama Algoritması, GKO, Krill Sürüsü, Sosyal Örümcek Optimizasyonu, Balina Optimizasyonu Algoritması (BOA)	-	Yöntemleri kıyaslarken parametre yoğunluğu, keşif ve sömütü durumu, hibrit mevcudiyeti, yerel aramasının olup olmadığı dikkate alınmıştır. Sonraki çalışmalar için bu algoritmalar arası kıyaslama yapılması ihtiyacını karşılamışlardır.

Çizelge 3.1. Literatür taraması (devam)

Kaynak	Yöntem	Veri Seti	Açıklama
Almufti ve ark. (2019)	Fil sürüsü optimizasyonu, U-dönen KKO	ch150**, eil51, berlin52, st70, eil76, pr76, kroA100, kroB100, kroC100, kroD100, eil101, lin105, pr107, pr124, pr136, kroA150, kroB150**	Simetrik GSP için iki farklı yöntem ele alınmıştır. Her iki algoritma da aynı deneysel koşullarda aynı problemi çözmek için çalıştırıldığında, 17 problemden 9 problemde Fil sürüsü optimizasyonunun, U-Dönen KKO yönteminden daha iyi çözüme sahip olduğu, U-Dönen KKO'nun 2 problemde daha iyi çözüme sahip olduğu ve kalan 6 problemde aynı çözümü verdikleri sonucuna ulaşılmıştır. Fil sürüsü optimizasyonu, 15 problemde optimal çözüme ulaşırken, U-Dönen KKO 7 problemde optimal çözüme ulaşmıştır.
Gao (2020)	Yeni KKO	ch150, kroA200, a280, lin318, rd400, rat575, **att48, eil51, berlin52, st70, pr76, eil76, kroA100, eil101, lin105, pr124, bier127, ch130, pr136, tsp225**	Çalışmada karıncaların arama alanını genişleterek olası çözümleri çeşitlendiren yeni bir KKO önerilmiştir. Problemin karmaşıklığı arttıkça, yeni KKO algoritmasının hesaplama verimliliği standart KKO'ya göre artmıştır.
Lin ve ark. (2020)	Açgözlü-Levy KKO	ch150, a280, kroA200, kroB200, gr202, pr226, ts225, tsp225, gr229, pr299, gil262, lin318) berlin52, eil51, eil76, eil101, korA100, lin105, rat99, st70	Karmaşık kombinatoryal optimizasyon problemlerini çözmek için epsilon açgözlü ve Levy uçuşundan yararlanarak, bu iki yaklaşımı birleştiren bir açgözlü-Levy KKO önerilmiştir. Önerilen yöntem, MMKS ile karşılaştırıldığında daha iyi bir performans göstermiştir.
Deng ve ark. (2021)	Tavlama Benzetimi ile Hibrit Hücresel GA (TBHHGA)	att48, berlin52, bier127, ch130, eil51, eil76, kroA100, kroA200, oliver30, pr107, pr136, pr152, **chn31**	TBHHGA, hücresel otomataya dayalı geliştirilmiş bir GA'dır. TBHHGA'da seçim işlemi hücrenin durumuna göre yapılır. Tavlama benzetimi ile birleştirilerek, yakınsama hızını artırmak için elitist bir strateji sunulmuştur. Deneysel sonuçlar, çoğu durumda, kural 2 kullanılarak TBHHGA tarafından elde edilen sonuçların, kural 1 ve kural 3'ü kullanmanın sonuçlarından daha iyi ve daha kararlı olduğunu göstermektedir. Aynı zamanda, TBHHGA; GA, tavlama benzetimi ve Hücresel GA ile karşılaştırılmıştır. Karşılaştırma sonuçları, önerilen algoritma ile elde edilen mesafenin, teorik optimum değere daha yakınsadığını ve diğer üç algoritmaya kıyasla ortalama %7 oranında kısalttığını göstermektedir.

Çizelge 3.1. Literatür taraması (devam)

Kaynak	Yöntem	Veri Seti	Açıklama
Zhang ve ark. (2021)	Değişken Komşuluk ile Kesikli BOA (DKKBOA)	eil51, st70 ,eil76, pr76, kroa100, pr107 ,ch150, d198, tsp225, fl417, **oliver30 ,berlin52**	Balina optimizasyon algoritması, sürekli optimizasyon problemlerinin çözümünde iyi optimizasyon sonuçları elde eden yeni bir tür sürü zekâsı biyonik optimizasyon algoritmasıdır. Önerilen yöntem 12 TSPLIB problemi ile GKO, yarasa algoritması, güve alevi optimizasyonu, PSO, Kesikli BOA yöntemleri ile karşılaştırılmıştır.

Çizelge 3.1'de verilen literatürdeki çalışmalara ait bilgiler incelendiğinde, GKA algoritması 7 farklı çalışmada kullanılmıştır. Bu çalışmaların bazılarında GKA algoritması farklı metasezgiseller ile birleştirilerek hibrit yöntemler elde edilmiştir. Bu çalışmada ise, daha önce GKA algoritması ile birlikte kullanılmamış 3-Opt algoritması kullanılmıştır.

4. MATERYAL VE YÖNTEM

Çalışmada veri seti olarak, literatürde sıkça kullanılan farklı koordinatlar ve farklı şehir sayılarını içeren TSPLIB (Reinelt, 1991) kütüphanesindeki 41 örnek problem kullanılmıştır. Bu 41 test problemi, yeni hibrit yöntem ile literatür taramasında karşılaşılan, iyi sonuçlar veren diğer yöntemlerle kıyaslanabilmesi ve performansının test edilebilmesine olanak sağlaması göz önüne alınarak seçilmiştir. İKGKA algoritması üzerinde 3-Opt yöntemi eklenerek iyileştirme yapılan yeni bir yöntem önerilmiştir. Önerilen yöntem, IntelliJ IDEA Community Edition 2019.3.3 platformunda, Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz işlemci ve 16 GB RAM'e sahip bir bilgisayar ile değerlendirilmiştir. Kullanılan veri setindeki problemler ve önerilen yöntem hakkındaki detaylı bilgiler ilerleyen alt bölümlerde verilmiştir.

4.1. Kullanılan Veri Seti

TSPLIB (Reinelt, 1991), çeşitli kaynaklardan ve çeşitli türlerden GSP ve ilgili problemleri için örnek problemlerden oluşan bir kütüphanedir. Kullanılan veri seti literatürde en sık kullanılan ve diğer yöntemlerle rahatlıkla kıyaslama yapılmasına imkan sağlayan bir veri setidir. GSP konusunda yapılan araştırmalarda, bu veri setinin sık kullanılıyor olmasının nedenleri arasında tüm araştırmacıların kullanımına açık oluşu, örnek problem sayısının fazlalığı, bu problemlerin içerdiği şehir sayılarının değişkenliği ve içerdiği şehir yerleşimlerinin birbirinden farklı oluşu, büyük, orta veya küçük ölçekte testleri yapabilmeye imkan sağlayacak kadar çeşitliliğe sahip olması yer almaktadır. Çalışmada, bu kütüphane altındaki simetrik GSP için verilen örnek problemler kullanılmıştır. Çalışmada kullanılan 41 probleme ait veriler Çizelge 4.1'de verilmiştir. Problem adlandırmasında kullanılan sayılar aynı zamanda ilgili probleme ait şehir sayısını göstermektedir. Bu problemlerde her bir düğüm çifti için bir dizi *n* düğüm ve aralarındaki uzaklık verildiğinde, her düğümü tam olarak bir kez ziyaret eden minimum toplam uzunlukta bir tur bulunmalıdır. Bu problemlerde *i* düğümünden *j* düğümüne olan uzaklık, *j* düğümünden *i* düğümüne olan uzaklık ile aynı olması sebebiyle simetrik GSP olarak adlandırılmıştır.

Çizelge 4.1. Kullanılan problemlerin içeriği

Problem	Şehir Sayısı	Optimum Sonuç	Problem	Şehir Sayısı	Optimum Sonuç
eil51	51	426	pr152	152	73,682
berlin52	52	7,542	rat195	195	2,323
st70	70	675	d198	198	15,780
pr76	76	108,159	kroA200	200	29,368
eil76	76	538	kroB200	200	29,437
kroA100	100	21,282	ts225	225	126,643
kroB100	100	22,141	tsp225	225	3,916
kroC100	100	20,749	pr226	226	80,369
kroD100	100	21,294	gil262	262	2,378
kroE100	100	22,068	pr264	264	49,135
eil101	101	629	a280	280	2,579
lin105	105	14,379	pr299	299	48,191
pr107	107	44,303	lin318	318	42,029
pr124	124	59,030	rd400	400	15,281
bier127	127	118,282	fl417	417	11,861
ch130	130	6,110	pr439	439	107,217
pr136	136	96,772	rat575	575	6,773
pr144	144	58,537	rat783	783	8,806
ch150	150	6,528	pr1002	1002	259,045
kroA150	150	26,524	nrw1379	1379	56,638
kroB150	150	26,130			

4.2. Önerilen Yöntem

Önerilen yöntemde, İKGKA algoritması yeniden yapılandırılarak tur geliştirici sezgiseller arasında yer alan 3-Opt yöntemi, standart algoritmada kullanılan çift köprü yöntemi yerine eklenmiştir. Algoritma hakkında ayrıntılı bilgiler ilerleyen alt bölümlerde açıklanmaktadır.

4.2.1. Guguk kuşu arama algoritması

Optimizasyon problemlerinin çözümü için Yang ve Deb (2009) tarafından geliştirilen GKA algoritması, Quaarab ve ark. (2014) tarafından GSP probleminin çözümüne uyarlanmıştır. 2009'da geliştirilmesiyle birlikte, planlama, yönlendirme, rastgele değişkenleri seçme, sinir ağlarında izleme ağırlıkları, üretim, durum izleme, sağlık izleme vb. çok sayıda uygulamada kullanılmıştır. Çeşitli çalışmalarda bu algoritmaya dayanan yeni metasezgisel yöntemler önerilmiştir (Al-Abaji, 2020). Bu algoritma, bazı guguk kuşu türlerinin zorunlu kuluçka paraziti davranışı ile bazı kuşların ve meyve sineklerinin Levy uçuş davranışının birlikte kullanılmasına dayanmaktadır (Joshi ve ark., 2017, Quaarab ve ark., 2014).

Guguk kuşları, güzel seslerinin yanında agresif üreme stratejileri olan kuşlardır. Bazı guguk kuşu türleri, kendi yumurtalarının yaşama olasılığını artırmak için yuva sahibi kuşun yumurtalarını atarlar ve kendi yumurtalarını bırakırlar. Yuvayı seçerken

renk, boyut ve desen açısından kendi yumurtasına benzer ölçütte yumurtaların olduğu yuvaları tercih eder. Bu, yumurtalarının fark edilme olasılığını azaltır. Oldukça fazla sayıda tür, yumurtalarını diğer ev sahibi kuşların yuvalarına bırakarak zorunlu kuluçka parazitizmine girer. Ev sahibi kuş, yumurtaların kendilerine ait olmadığını keşfederse ya bu yabancı yumurtaları atar ya da yuvasını terk eder ve başka bir yerde yeni bir yuva kurar (Yang ve Deb, 2009).

Ayrıca bazı türlerin yumurtlama zamanlaması da şaşırtıcıdır. Genel olarak, guguk kuşu yumurtaları, dişi guguk kuşu yumurtlamadan önce ısı seviyelerini bir süre yüksek tuttuğu için ev sahibi yumurtalarından biraz daha erken yumurtadan çıkar. Guguk kuşu yavrusu yumurtadan çıktıktan sonra yapacağı ilk içgüdüsel eylem, ev sahibi kuşun yumurtalarını körü körüne yuvadan dışarı atmaktır, bu da guguk kuşu yavrusunun ev sahibi kuş tarafından sağlanan yiyecek payını arttırır. Aynı zamanda guguk kuşu yavruları üvey annenin getirdiği yiyeceğin hepsini yeme içgüdüsüne sahiptir. Bu yüzden diğer yavrular yumurtadan çıkabilseler de ya açlıktan ya da guguk kuşu yavrusunun saldırılarından dolayı hayatlarını kaybederler.

Fransız matematikçi Paul Levy tarafından Levy uçuşu olarak adlandırılan uçuşlar, bir güç yasası dağılımına uyan adım uzunlukları ile karakterize edilen bir rastgele yürüyüş modelini temsil eder. Yapılan çalışmalar, birçok hayvan ve böceğin uçuş davranışının Levy uçuşlarının tipik özelliklerini sergilediğini göstermiştir. Avcıların av arama davranışları üzerine yapılan araştırmalar da Levy uçuşlarının tipik özelliklerini sergilediklerini göstermektedir. Buradan yola çıkılarak bu davranış optimizasyon problemlerine uygulanmış ve umut vadeden sonuçlar vermiştir.

4.2.2. İyileştirilmiş kesikli guguk kuşu arama algoritması

Yang ve Deb (2009) tarafından geliştirilen standart GKA algoritmasında bir guguk kuşu Levy uçuşu aracılığıyla yeni bir yuva arar. Bu algoritmada ideal olarak üç temel kural kullanılmaktadır:

1) Her guguk kuşu her seferinde bir yumurta bırakır ve yumurta bırakacağı yuvayı rastgele seçer.
2) En yüksek yumurta kalitesine sahip olan en iyi yuvalar gelecek nesillere aktarılacaktır.
3) Mevcut ev sahibi yuva sayısı sabittir ve guguk kuşunun yumurtladığı yumurta, ev sahibi kuş tarafından $p_a \in [0, 1]$ olasılıkla keşfedilebilir. Bu durumda ev sahibi kuş ya yumurtayı atabilir ya da yuvayı terk ederek

tamamen yeni bir yuva yapabilir. Basitleştirmek amacıyla, bu son varsayımda, n yuvanın p_a oranı kadarı yeni yuvalarla (yeni rasgele çözümlerle) değiştirilir.

Burada yuvadaki her bir yumurta bir çözümü temsil eder. Bir guguk kuşu yumurtası yeni bir çözümü temsil eder. Amaç, yeni ve potansiyel olarak daha iyi çözümleri (guguk kuşları) yuvalardaki yeterince iyi olmayan çözümlerle değiştirmektir. Bu üç kurala dayanan GKA'nın temel adımları Algoritma 4.1'de verilmiştir.

Algoritma 4.1. GKA sözde kodu

Amaç fonksiyonu $f(x)$,$x=(x_1, \ldots, x_d)^T$

n adet başlangıç popülasyonu üret $x_i(i=1, \ldots, n)^T$

while (t<*Maksimum iterasyon*) veya (*durdurma kriteri*) **do**

Levy uçuşu ile rassal bir guguk kuşu al

F_i uygunluk değerini hesapla

n yuva içerisinde rassal olarak (j olsun) bir yuva seç

if ($F_i > F_j$) **then**

j'yi yeni çözüm ile değiştir

end if

En kötü yuvalardan p_a oranı kadarını terk et ve yenilerini inşa et

En iyi çözümleri tut (kaliteli çözüme sahip yuvalar)

Çözümleri sırala ve o anki en iyi çözümü bul

end while

Algoritma tamamlandı, sonuçları göster

Guguk kuşu i için Levy uçuşu ile yeni çözümler üretilirken (x^{t+1}) Denklem 4.1 kullanılır. Burada α (α>0) adım büyüklüğü olarak adlandırılmaktadır ve çoğu durumda α=1 kullanılır. Adım büyüklüğü Levy dağılımını takip eder ve Denklem 4.2 ile hesaplanır. Burada s Levy dağılımından elde edilen adım büyüklüğüdür.

$$x_i^{(t+1)} = x_i^t + \alpha \oplus \text{Levy}\ (s, \lambda) \tag{4.1}$$

$$\text{Levy}\ (s, \lambda) \sim s^{-\lambda},\ (1 < \lambda \leq 3) \tag{4.2}$$

İKGKA algoritmasında guguk kuşlarının çok daha iyi çözümler bulması için biraz zekaya sahip olabileceği varsayılır. Guguk kuşları ev sahibi olma olasılığı yüksek olan yuvalar üzerinde gözlem yapabilmektedirler. Yumurtaların terk edilmesini önlemek için kuluçka sırasında konak yuvasını değiştirme yeteneğine sahip yeni bir akıllı guguk kuşu kategorisi oluşturmak için bu davranıştan esinlenilmiştir. Bu akıllı kategorisindeki guguk kuşları, yuvanın en iyi seçim olup olmadığına karar vermek için ev sahibi yuvanın gözlemlenmesi gibi kuluçka öncesi ve sonrası mekanizmaları kullanır. Bu durumda akıllı guguk kuşlarının p_c oranında eklenmesi ile Algoritma 4.1, üç tip guguk kuşu ile Algoritma 4.2'deki gibi yeniden yapılandırılır:

1) Bir guguk kuşu, popülasyondan rasgele seçilen bir çözümden daha iyi yeni çözümleri içerebilecek alanlar (en iyi konumdan) arar.
2) p_a oranı kadar guguk kuşu en iyi çözümden uzak yeni çözümler arar.
3) p_c oranı kadar guguk kuşu mevcut konumdaki çözümleri iyileştirmek için çözümler arar. Yerel optimumda takılmaksızın her bölgedeki en iyi çözümü bulmak için Levy uçuşları aracılığıyla bir bölgeden diğerine hareket ederler.

Algoritma 4.2. İKGKA sözde kodu

Amaç fonksiyonu $f(x)$,$x=(x_1, \ldots, x_d)^T$
n adet başlangıç popülasyonu üret $x_i(i=1, \ldots, n)^T$
while ($t<$*Maksimum iterasyon*) veya (*durdurma kriteri*) **do**
    Aramaya p_c oranı kadar akıllı guguk kuşları ile başla
    Levy uçuşu ile rassal bir guguk kuşu al
    F_i uygunluk değerini hesapla
    n yuva içerisinde rassal olarak (j olsun) bir yuva seç
    if ($F_i > F_j$**) then**
        j'yi yeni çözüm ile değiştir
    end if
    En kötü yuvalardan p_a oranı kadarını terk et ve yenilerini inşa et
    En iyi çözümleri tut (kaliteli çözüme sahip yuvalar)
    Çözümleri sırala ve o anki en iyi çözümü bul
end while
Amaç fonksiyonu
    Levy uçuşu ile rassal bir guguk kuşu al

4.2.3. Gezgin satıcı probleminin guguk kuşu arama algoritması ile çözümü

GKA algoritması GSP için uyarlanırken GKA'da kullanılan yumurta, yuva, amaç fonksiyonu, arama alanı ve Levy uçuşları terimlerinin karşılıklarının belirlenmesi ve ilişkilendirmelerin yapılması gerekir.

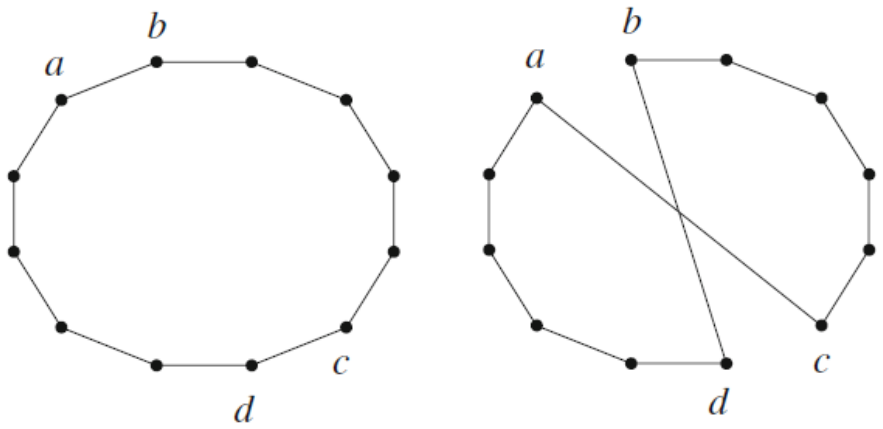
- **Yumurta**: Bir yumurta bir Hamilton turuna karşılık gelir. Bir guguk kuşunun bir yuvaya tek bir yumurta bıraktığı varsayılır.
- **Yuva**: Bir yuva kendi Hamilton turuna sahip olan bir popülasyona karşılık gelir ve popülasyon sayısı, yuva sayısına eşittir.
- **Amaç fonksiyonu**: Arama uzayındaki her çözüm, sayısal bir değerle ilişkilendirilir. Yani bir çözümün kalitesi amaç fonksiyonunun değeri ile orantılıdır. GSP için çözümün kalitesi, Hamilton turunun uzunluğu ile ilgilidir. En iyi çözüm, en kısa Hamilton döngüsüne sahip olandır.
- **Arama alanı**: GKA'da iki boyutlu durumda potansiyel yuvaların konumudur. Bir yuvanın konumunu değiştirmek için yalnızca koordinatlarının gerçek değerlerinin değiştirilmesi gerekir. GSP çözüm uzayında şehirlerin koordinatları, ziyaret edilen şehirlerin sabit koordinatlarıdır; ancak şehirler arasındaki ziyaret sırası değişebilir.
- **Levy uçuşu**: Bir çözüm etrafında yoğun bir araştırma ve ardından uzun vadede ara sıra büyük adımlar atma özelliğine sahiptir. Arama kalitesini iyileştirmek için, standart GKA'da belirtildiği gibi adım büyüklüğü Levy uçuşları tarafından oluşturulan değerler ile ilişkilidir.

İKGKA'da arama alanında ziyaret edilen şehirlerin sırasını değiştirerek mevcut bir çözümden yeni başka bir çözüm üretmede 2-Opt ve çift köprü yöntemleri kullanılmaktadır. 2-Opt yöntemi küçük adımlarla küçük değişiklikler, çift köprü yöntemi büyük adımlarla daha büyük değişiklikler yapabilmek için kullanılır. Önerilen yöntemde 3-Opt yöntemi, standart algoritmada kullanılan çift köprü yöntemi yerine eklenmiştir.

4.2.4. 2-Opt

Tur geliştirici sezgisel olan 2-Opt Croes (1958) tarafından geliştirilmiştir ve hibrit metasezgisel yöntemler oluşturulurken yaygın olarak kullanılmaktadır. Başlangıç çözüm olarak, herhangi bir uygun çözüm alınabilir veya basit sezgisellerle başlangıç çözümü oluşturulabilir. Öncelikle muhtemel çözüm üzerinden iki nokta çifti seçilir (a-b,

c-d). Sonraki aşamada turu bozmayacak şekilde iki nokta yer değiştirir (a-c, b-d). Özetle, GSP için Hamilton turu üstünde rasgele 2 kenar kaldırılır ve değiştirilerek yeniden birleştirilir. Noktaların değişimi Şekil 4.1'de gösterilmiştir.



Şekil 4.1. 2-Opt (Quaarab ve ark., 2014)

Yeni oluşan çözüm bir öncekine göre iyileşme sağlamışsa kabul edilir. Aksi halde bir önceki çözümle devam edilir. Bu şekilde yeni rota üzerinde sadece iki noktanın yeri değişir ve rotadaki diğer noktalar sabit kalır. İkili değişimle artık daha iyi sonuç bulunamayana kadar bu şekilde devam edilir ve tüm kombinasyonlar denenir. 2-Opt algoritması ile optimuma çok yakın bir değer, çok hızlı bir şekilde bulunabilir.

Algoritma 4.3'te, T mevcut tur, $f(T)$ mevcut turun uzunluğu, T' yeni bulunan tur, $f(T')$ yeni turun uzunluğu, z değişim ve z_{min} mevcut aramadaki en iyi değişim olacak şekilde algoritmanın sözde kodu verilmiştir.

Algoritma 4.3. 2-Opt sözde kodu

Adım 1: Başlangıç çözümünü ve tur uzunluğunu belirle: T, f (T)

Adım 2: i - j değerlerini sıfırla, $T' = T$, $z_{min} = 0$, z = 0

Adım 3: i - j ikilisini belirle ($i \neq j$ ve $i < j$)

Adım 4: Tur içerisinde i-j aralığını ters çevir (j-i olarak yeniden konumlandır):T'

Adım 5: Turun yeni uzunluğunu belirle: $f(T')$

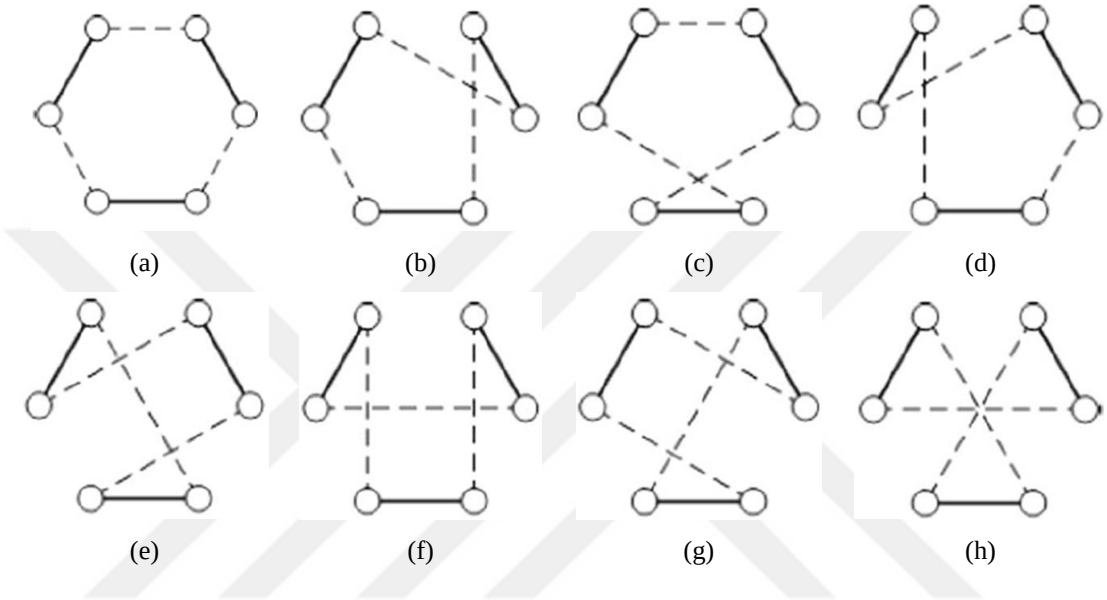
Adım 6: $f(T') < f(T)$ ise $z = f(T') - f(T)$,

Adım 7: $z < z_{min}$ ise $z_{min} = z$, $T'' = T'$, denenmemiş i-j varsa Adım 3'e git

Adım 8: $T = T''$; $z_{min} < 0$ ise Adım 2'ye git, yoksa sonlandır.

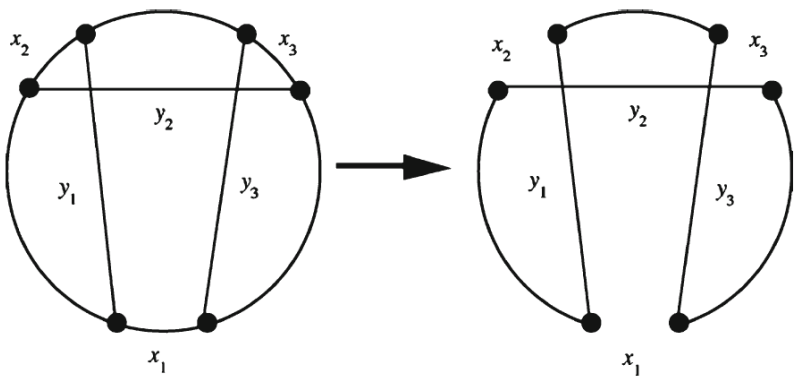
4.2.5. 3-Opt

Bu yöntemde, 2-Opt algoritmasından farklı olarak 2 kenar yerine 3 kenar seçilerek yerleri değiştirilir. Seçilen 3 kenar için farklı birleştirme kombinasyonları kullanılabilmektedir. Şekil 4.2'de 3-Opt bağlantı kombinasyonları verilmiştir. Bu şekilde 3-Opt uygulanmadan önceki tur Şekil 4.2a olarak ele alındığında, yedi farklı kombinasyonla yapılabilecek 3-Opt turları gösterilmiştir.



Şekil 4.2. 3-Opt bağlantı kombinasyonları (Gülcü ve ark., 2018)

Önerilen yöntemde yapılan birleştirme Şekil 4.3'te gösterilmiştir. Değişim sonucunda oluşan turun uzunluğu, değişim öncesi uzunluk ile kıyaslanarak iyileşme olması durumunda yeni tur eskisi ile değiştirilir. Yeni oluşan tur iyileşme sağlamıyorsa, yani eski turdan daha uzunsa, eski tur kullanılmaya devam edilir. Algoritmanın sözde kodu Algoritma 4.4'te verilmiştir.



Şekil 4.3. 3-Opt (Helsgaun, 2009)

Algoritma 4.4. 3-Opt sözde kodu

Adım 1: Başlangıç çözümünü ve tur uzunluğunu belirle: *T*, *f* (T)

Adım 2: *i* - *j* - *k* değerlerini sıfırla, $T' = T$, $z_{min} = 0$, z = 0

Adım 3: *i, j, k* değerlerini belirle ($i \neq j \neq k$ ve $i < j < k$)

Adım 4: 0 - *i* aralığına, önce (*j*+1) - *k* aralığını, sonra (*i*+1) - *j* aralığını ters çevirerek, en son (*j*+1) – 0 aralığını değiştirmeden ekle :*T'*

Adım 5: Turun yeni uzunluğunu belirle: *f (T')*

Adım 6: $f(T') < f(T)$ ise $z = f(T') - f(T)$,

Adım 7: $z < z_{min}$ ise $z_{min} = z$, $T'' = T'$, denenmemiş *i-j* varsa Adım 3'e git

Adım 8: $T = T''$; $z_{min} < 0$ ise Adım 2'ye git, yoksa sonlandır.

5. ARAŞTIRMA SONUÇLARI VE TARTIŞMA

Önerilen yöntem, esinlenilen İKGKA yöntemi ve literatürde son yıllarda geliştirilen yöntemlerle karşılaştırılmıştır. Önerilen yöntemde parametreler İKGKA yönteminde kullanıldığı şekliyle sabitlenip, p_a:0.2 p_c:0.6, m:3, n:20 olarak ayarlanmıştır. İterasyon sayıları sırasıyla 500 ve 1,000 alınıp 30'ar defa çalıştırılarak test edilmiştir. Bulunan en iyi değerler koyu renk ile işaretlemiştir. Yöntemlerin ayrı ayrı karşılaştırılması ilerleyen alt bölümlerde sunulmuştur.

5.1. Önerilen Yöntemin İKGKA ile Karşılaştırılması

Önerilen yöntem, adil bir karşılaştırma olması açısından İKGKA yönteminde izlenen metodoloji kullanılarak Çizelge 5.1 ve 5.2'de gösterildiği gibi en iyi, en kötü, ortalama sonuçlar ve süre (saniye) açısından İKGKA yöntemi ile karşılaştırılmıştır. Çizelge 5.1, önerilen yöntemin 1,000 iterasyon çalıştırıldığında pr136, kroA200, gil262, pr299, rd400, fl417, pr439, rat575, rat783, pr1002 ve nrw1379 problemlerinde İKGKA yöntemine göre optimumu bulma veya daha fazla yaklaşma açısından daha iyi sonuçlara ulaşabildiğini açıkça göstermektedir. rat195 için %0.09, tsp225 için %0.26 ve lin318 için %0.08 farkla İKGKA yöntemi daha iyi sonuçlara ulaşabilmiştir. Her iki yöntem de 200 şehirden daha küçük boyutlu problemlerin çoğunda optimum sonucu bulmuştur. Ancak, problem boyutu büyüdükçe önerilen yöntemin başarısı daha da fazla ortaya çıkmaktadır. Önerilen yöntemin optimum sonucu bulamadığı 3 problem için ise en fazla %0.26 sapma oranıyla iyi sonuçtan uzak kaldığı görülmektedir. İKGKA yönteminin optimum sonucu bulamadığı 11 problemde, İKGKA yönteme kıyasla %1'in altında sapma oranı ile 1,379 şehir içeren nrw1379 probleminde ise %1.08 sapma oranı ile daha iyi sonuç bularak önerilen yöntemin optimuma daha fazla yaklaştığı görülmektedir. Çizelge 5.2'de önerilen yöntem 500 iterasyon çalıştırıldığında elde edilen sonuçlar verilmiştir. Her iki çizelgeye ayrıntılı bir şekilde bakıldığında, aynı parametre seti ile 1,000 iterasyon çalıştırıldığında optimumu bulma veya optimuma yaklaşma açısından 500 iterasyon çalıştırılmasına göre daha iyi performans gösterdiği görülmektedir. İterasyon sayısını 1,000'in üzerine çıkarıldığında işlem süresi artmasına rağmen, elde edilen sonuçlarda önemli bir iyileşme sağlanamamıştır.

Önerilen yöntem ve İKGKA yöntemi aynı parametreler ile 500 iterasyon çalıştırılarak işlem süresi açısından karşılaştırıldığında, önerilen yöntemin 39 veri

setinde daha kısa işlem süresine sahip olduğu, sadece berlin52 veri setinde 0.33 saniye ve pr144 veri setinde 0.01 saniye daha uzun sürdüğü görülmektedir.

İterasyon sayısı ikiye katlandığında, önerilen yöntemin 500 iterasyon çalıştırılmasına göre, bulunan çözümler açısından daha iyi performans gösterdiği ve işlem süresinde de çok fazla artış olmadığı gözlemlenmiştir. Önerilen yöntem aynı parametreler ile 1,000 iterasyon çalıştırılarak işlem süresi açısından İKGKA yöntemi ile karşılaştırıldığında, veri setlerinin çoğunda işlem süresi açısından yine daha iyi performans gösterdiği görülmektedir. Bu kapsamda, önerilen yöntemin 37 veri setinde daha kısa işlem süresine sahip olduğu, sadece berlin52 veri setinde 0.76 saniye, kroA100 veri setinde 0.24 saniye, pr124 veri setinde 1.63 saniye ve pr144 veri setinde 3.59 saniye daha uzun sürdüğü görülmektedir.

Çizelge 5.1. Önerilen yöntemin (1,000 iterasyon) **İKGKA** ile karşılaştırılması

Veri seti	Optimum sonuç	Önerilen Yöntem				İKGKA (500 İterasyon)			
		En iyi	En kötü	Ortalama	Süre	En iyi	En kötü	Ortalama	Süre
eil51	426	**426**	428	426.77	0.81	**426**	426	426.00	1.16
berlin52	7,542	**7,542**	7,775	7,549.77	0.85	**7,542**	7,542	7,542.00	0.09
st70	675	**675**	684	677.43	1.43	**675**	675	675.00	1.56
pr76	108,159	**108,159**	109,085	108,357.7	1.72	**108,159**	108,159	108,159.00	4.73
eil76	538	**538**	546	539.23	1.67	**538**	539	538.03	6.54
kroA100	21,282	**21,282**	21,369	21,288.63	2.94	**21,282**	21,282	21,282.00	2.70
kroB100	22,141	**22,141**	22,295	22,168.60	3.13	**22,141**	22,157	22,141.53	8.74
kroC100	20,749	**20,749**	20,880	20,763.10	3.15	**20,749**	20,749	20,749.00	3.36
kroD100	21,294	**21,294**	21,731	21,380.57	3.03	**21,294**	21,389	21,304.33	8.35
kroE100	22,068	**22,068**	22,271	22,141.80	3.16	**22,068**	22,121	2,281.26	14.18
eil101	629	**629**	643	633.03	2.92	**629**	633	630.43	18.74
lin105	14,379	**14,379**	14,504	14,389.07	3.41	**14,379**	14,379	14,379.00	5.01
pr107	44,303	**44,303**	44,566	44,420.70	3.48	**44,303**	44,358	44,307.06	12.89
pr124	59,030	**59,030**	59,838	59,079.93	4.99	**59,030**	59,030	59,030.00	3.36
bier127	118,282	**118,282**	120,413	118,982.77	5.51	**118,282**	118,730	118,359.63	25.50
ch130	6,110	**6,110**	6,188	6,147.30	4.91	**6,110**	6,174	6,135.96	23.12
pr136	96,772	**96,772**	99,051	97,595.13	6.07	96,790	97,318	97,009.26	35.82
pr144	58,537	**58,537**	58,537	58,537.00	6.55	**58,537**	58,537	58,537.00	2.96
ch150	6,528	**6,528**	6,624	6,556.67	6.64	**6,528**	6,611	6,549.90	27.74
kroA150	26,524	**26,524**	27,083	26,642.90	6.74	**26,524**	26,767	26,569.26	31.23
kroB150	26,130	**26,130**	26,387	26,235.47	7.11	**26,130**	26,229	26,159.30	33.01
pr152	73,682	**73,682**	73,880	73,740.53	7.50	**73,682**	73,682	73,682.00	14.86
rat195	2,323	2,326	2,372	2,345.30	15.45	**2,324**	2,357	2,341.86	57.25
d198	15,780	**15,781**	15,859	15,811.50	19.41	**15,781**	15,852	15,807.66	59.95
kroA200	29,368	**29,368**	29,949	29,522.67	13.16	29,382	29,886	29,446.66	62.08
kroB200	29,437	**29,445**	30,044	29,663.27	12.66	29,448	29,819	29,542.49	64.06
ts225	126,643	**126,643**	127,161	126,755.67	17.10	**126,643**	126,810	126,659.23	47.51
tsp225	3,916	3,926	4,021	3,974.40	14.96	**3,916**	3,997	3,958.76	76.16
pr226	80,369	**80,369**	80,908	80,458.77	15.75	**80,369**	80,620	80,386.66	50.00
gil262	2,378	**2,379**	2,415	2,397.17	21.63	2,382	2,418	2,394.50	102.39
pr264	49,135	**49,135**	49,893	49,271.30	21.46	**49,135**	49,692	49,257.50	82.93
a280	2,579	**2,579**	2,647	2,597.97	24.12	**2,579**	2,623	2,592.33	115.57
pr299	48,191	**48,221**	48,807	48,481.00	33.59	48,327	48,753	48,470.53	138.20
lin318	42,029	42,158	42,896	42,469.63	33.33	**42,125**	42,890	42,434.73	156.17
rd400	15,281	**15,363**	15,587	15,481.13	53.94	15,447	15,704	15,533.73	264.94
fl417	11,861	**11,870**	11,989	11,905.03	58.10	11,873	11,975	11,910.53	274.59
pr439	107,217	**107,348**	110,448	108,188.40	74.16	107,447	109,013	107,960.50	308.75
rat575	6,773	**6,861**	6,979	6,928.90	130.50	6,896	7,039	6,956.73	506.67
rat783	8,806	**8,967**	9,118	9,035.43	278.09	9,043	9,171	9,109.26	968.66
pr1002	259,045	**264,171**	268,672	266,440.60	588.59	266,508	271,660	268,630.03	1,662.61
nrw1379	56,638	**58,322**	58,956	58,678.90	1,055.35	58,951	59,837	59,349.53	3,160.47

Çizelge 5.2. Önerilen yöntemin (500 iterasyon) İKGKA ile karşılaştırılması

Veri seti	Optimum	Önerilen Yöntem				İKGKA (500 İterasyon)			
	sonuç	En iyi	En kötü	Ortalama	Süre	En iyi	En kötü	Ortalama	Süre
eil51	426	**426**	432	427.13	0.41	**426**	426	426.00	1.16
berlin52	7,542	**7,542**	7,775	7,557.53	0.42	**7,542**	7,542	7,542.00	0.09
st70	675	**675**	682	677.53	0.71	**675**	675	675.00	1.56
pr76	108,159	**108,159**	109,118	108,407.83	0.87	**108,159**	108,159	108,159.00	4.73
eil76	538	**538**	548	539.37	0.87	**538**	539	538.03	6.54
kroA100	21,282	**21,282**	21,373	21,296.77	1.48	**21,282**	21,282	21,282.00	2.70
kroB100	22,141	**22,141**	22,410	22,193.50	1.48	**22,141**	22,157	22,141.53	8.74
kroC100	20,749	**20,749**	21,010	20,797.87	1.45	**20,749**	20,749	20,749.00	3.36
kroD100	21,294	**21,294**	21,679	21,371.57	1.47	**21,294**	21,389	21,304.33	8.35
kroE100	22,068	**22,068**	22,361	22,138.77	1.47	**22,068**	22,121	2,281.26	14.18
eil101	629	**629**	641	634.30	1.41	**629**	633	630.43	18.74
lin105	14,379	**14,379**	14,556	14,399.77	1.59	**14,379**	14,379	14,379.00	5.01
pr107	44,303	**44,303**	44,621	44,396.20	1.75	**44,303**	44,358	44,307.06	12.89
pr124	59,030	**59,030**	59,838	59,100.70	2.38	**59,030**	59,030	59,030.00	3.36
bier127	118,282	**118,282**	120,523	118,967.13	2.56	**118,282**	118,730	118,359.63	25.50
ch130	6,110	**6,110**	6,195	6,149.17	2.47	**6,110**	6,174	6,135.96	23.12
pr136	96,772	**96,781**	98,878	97,683.53	2.91	96,790	97,318	97,009.26	35.82
pr144	58,537	**58,537**	58,537	58,537.00	2.97	**58,537**	58,537	58,537.00	2.96
ch150	6,528	**6,528**	6,631	6,552.60	3.24	**6,528**	6,611	6,549.90	27.74
kroA150	26,524	**26,524**	26,879	26,629.30	3.39	**26,524**	26,767	26,569.26	31.23
kroB150	26,130	**26,130**	26,475	26,209.07	3.38	**26,130**	26,229	26,159.30	33.01
pr152	73,682	**73,682**	73,818	73,772.67	3.53	**73,682**	73,682	73,682.00	14.86
rat195	2,323	2,330	2,390	2,349.90	5.16	**2,324**	2,357	2,341.86	57.25
d198	15,780	**15,781**	15,897	15,824.13	5.72	**15,781**	15,852	15,807.66	59.95
kroA200	29,368	**29,368**	29,713	29,505.77	6.05	29,382	29,886	29,446.66	62.08
kroB200	29,437	**29,446**	30,014	29,628.63	6.79	29,448	29,819	29,542.49	64.06
ts225	126,643	**126,643**	127,345	126,844.80	8.02	**126,643**	126,810	126,659.23	47.51
tsp225	3,916	3,923	4,042	3,984.50	7.07	**3,916**	3,997	3,958.76	76.16
pr226	80,369	**80,369**	80,912	80,495.93	7.86	**80,369**	80,620	80,386.66	50.00
gil262	2,378	2,384	2,423	2,401.90	9.74	**2,382**	2,418	2,394.50	102.39
pr264	49,135	**49,135**	50,140	49,380.50	10.53	**49,135**	49,692	49,257.50	82.93
a280	2,579	**2,579**	2,623	2,596.03	11.47	**2,579**	2,623	2,592.33	115.57
pr299	48,191	**48,197**	49,270	48,546.07	16.40	48,327	48,753	48,470.53	138.20
lin318	42,029	42,234	42,873	42,563.83	19.54	**42,125**	42,890	42,434.73	156.17
rd400	15,281	**15,433**	15,715	15,560.43	29.00	15,447	15,704	15,533.73	264.94
fl417	11,861	11,883	12,022	11,921.07	28.60	**11,873**	11,975	11,910.53	274.59
pr439	107,217	**107,365**	110,663	108,537.17	38.88	107,447	109,013	107,960.50	308.75
rat575	6,773	**6,873**	7,030	6,955.63	57.26	6,896	7,039	6,956.73	506.67
rat783	8,806	**9,001**	9,155	9,084.60	136.65	9,043	9,171	9,109.26	968.66
pr1002	259,045	**265,467**	270,638	268,248.23	290.50	266,508	271,660	268,630.03	1,662.61
nrw1379	56,638	**58,531**	59,645	59,220.30	506.74	58,951	59,837	59,349.53	3,160.47

5.2. Önerilen Yöntemin Literatürdeki Yöntemlerle Karşılaştırılması

Literatürdeki diğer çalışmalarda geliştirilen yöntemlerle karşılaştırma yapılırken, literatürdeki yöntemlerle elde edilen en iyi sonuç değerleri çizelgelere dahil edilmiştir. Bu sonuçlar önerilen yöntemin 1,000 iterasyon çalıştırılması ile elde edilen sonuçlarla karşılaştırılmıştır. Çizelgelerde verilen veri setleri önerilen yöntem ve literatürden alınan karşılaştırılacak yöntemin ortak olarak kullandığı problemlerdir. Karşılaştırma yapılırken gösterge olarak literatürdeki diğer çalışmalarda performans göstergesi olarak kullanılan ve her çalışma için ulaşılabilen en iyi sonuç ve sapma oranı değerleri kullanılmıştır. Sapma oranı hesabı Denklem 5.1 ile yapılmaktadır.

$$\text{Sapma Oranı} = \frac{\text{Yöntemin En İyi Sonucu} - \text{Optimum Sonuç}}{\text{Optimum Sonuç}} \times 100 \qquad (5.1)$$

Çizelge 5.3, önerilen yöntemin Zhang ve ark. (2021) tarafından geliştirilen DKKBOA yöntemi ile 11 farklı test problemi üzerinde çalıştırılması ile elde edilen karşılaştırma sonuçlarını vermektedir. DKKBOA yöntemi sadece berlin52 probleminde optimum sonuca ulaşabilirken, önerilen yöntemin berlin52 test problemi dahil 8 farklı test probleminde optimumu bulduğu, kalan 3 farklı test probleminde de optimuma daha fazla yaklaştığı görülmüştür.

Çizelge 5.3. Önerilen yöntemin performansının DKKBOA (Zhang ve ark., 2021) ile karşılaştırılması

Sıra	Veri seti	Optimum Sonuç	Önerilen Yöntem			DKKBOA		
			En İyi Sonuç	Sapma Oranı (%)	Süre	En İyi Sonuç	Sapma Oranı (%)	Süre
1	eil51	426	**426**	0.00	0.81	429	0.70	10.47
2	berlin52	7,542	**7,542**	0.00	0.85	**7,542**	0.00	10.79
3	st70	675	**675**	0.00	1.43	676	0.15	17.69
4	pr76	108,159	**108,159**	0.00	1.72	108,353	0.18	20.52
5	eil76	538	**538**	0.00	1.67	554	2.97	20.05
6	kroA100	21,282	**21,282**	0.00	2.94	21,721	2.06	35.24
7	pr107	44,303	**44,303**	0.00	3.48	45,030	1.64	38.09
8	ch150	6,528	**6,528**	0.00	6.64	6,863	5.13	77.37
9	d198	15,780	**15,781**	0.01	19.41	16,313	3.38	145.24
10	tsp225	3,916	**3,926**	0.26	14.96	4,136	5.62	195.54
11	fl417	11,861	**11,870**	0.08	58.10	12,462	5.07	2,485.61

Çizelge 5.4. Önerilen yöntemin performansının TBHHGA (Deng ve ark., 2021) ile karşılaştırılması

Sıra	Veri seti	Optimum Sonuç	Önerilen Yöntem			TBHHGA		
			En İyi Sonuç	Sapma Oranı (%)	Süre	En İyi Sonuç	Sapma Oranı (%)	Süre
1	eil51	426	**426**	0.00	0.81	428.87	0.67	-
2	berlin52	7,542	**7,542**	0.00	0.85	7,544.37	0.03	-
3	eil76	538	**538**	0.00	1.67	547.17	1.70	-
4	kroA100	21,282	**21,282**	0.00	2.94	21,285.44	0.02	-
5	pr107	44,303	**44,303**	0.00	3.48	**44,301.68**	0.00	-
6	bier127	118,282	**118,282**	0.00	5.51	118,293.52	0.01	-
7	ch130	6,110	**6,110**	0.00	4.91	6,183.03	1.20	-
8	pr136	96,772	**96,772**	0.00	6.07	96,795.40	0.02	-
9	pr152	73,682	**73,682**	0.00	6.55	73,683.64	0.00	-
10	kroA200	29,368	**29,368**	0.00	13.16	29,533.06	0.56	-

Çizelge 5.4, önerilen yöntemin Deng ve ark. (2021) tarafından geliştirilen TBHHGA yöntemi ile 10 farklı test problemi üzerinde çalıştırılması ile elde edilen karşılaştırma sonuçlarını vermektedir. TBHHGA yöntemi sadece pr107 probleminde optimum sonuca ulaşabilirken önerilen yöntem verilen 10 test probleminde de optimum sonuca ulaşmıştır.

Çizelge 5.5. Önerilen yöntemin performansının Shahab (2019) tarafından önerilen yöntem ile karşılaştırılması

Sıra	Veri seti	Optimum Sonuç	Önerilen Yöntem			Shahab (2019)		
			En İyi Sonuç	Sapma Oranı (%)	Süre	En İyi Sonuç	Sapma Oranı (%)	Süre
1	eil51	426	**426**	0.00	0.81	**426**	0.00	0.02
2	berlin52	7,542	**7,542**	0.00	0.85	**7,542**	0.00	0.02
3	st70	675	**675**	0.00	1.43	**675**	0.00	0.04
4	pr76	108,159	**108,159**	0.00	1.72	**108,159**	0.00	0.05
5	eil76	538	**538**	0.00	1.67	542	0.74	0.05
6	kroA100	21,282	**21,282**	0.00	2.94	**21,282**	0.00	0.10
7	kroB100	22,141	**22,141**	0.00	3.13	22,199	0.26	0.10
8	kroC100	20,749	**20,749**	0.00	3.15	**20,749**	0.00	0.11
9	kroD100	21,294	**21,294**	0.00	3.03	**21,294**	0.00	0.11
10	kroE100	22,068	**22,068**	0.00	3.16	22,069	0.00	0.11
11	eil101	629	**629**	0.00	2.92	631	0.32	0.11
12	lin105	14,379	**14,379**	0.00	3.41	**14,379**	0.00	0.12
13	pr107	44,303	**44,303**	0.00	3.48	**44,303**	0.00	0.12
14	pr124	59,030	**59,030**	0.00	4.99	**59,030**	0.00	0.18
15	bier127	118,282	**118,282**	0.00	5.51	118,682	0.34	0.19
16	ch130	6,110	**6,110**	0.00	4.91	6,148	0.62	0.20
17	pr136	96,772	**96,772**	0.00	6.07	96,874	0.11	0.22
18	pr144	58,537	**58,537**	0.00	6.55	**58,537**	0.00	0.27
19	ch150	6,528	**6,528**	0.00	6.64	6,553	0.38	0.29
20	kroA150	26,524	**26,524**	0.00	6.74	26,584	0.23	0.30
21	kroB150	26,130	**26,130**	0.00	7.11	26,132	0.01	0.30
22	pr152	73,682	**73,682**	0.00	7.50	**73,682**	0.00	0.30
23	rat195	2,323	**2,326**	0.13	15.45	2,337	0.60	0.53
24	d198	15,780	15,781	0.01	19.41	**15,780**	0.00	0.58
25	kroA200	29,368	**29,368**	0.00	13.16	29,429	0.21	0.59
26	kroB200	29,437	**29,445**	0.03	12.66	**29,445**	0.03	0.60
27	ts225	126,643	**126,643**	0.00	17.10	127,737	0.86	0.80
28	tsp225	3,916	**3,926**	0.26	14.96	3,974	1.48	0.80
29	pr226	80,369	**80,369**	0.00	15.75	**80,369**	0.00	0.80
30	gil262	2,378	**2,379**	0.04	21.63	2,396	0.76	1.23
31	pr264	49,135	**49,135**	0.00	21.46	**49,135**	0.00	1.22
32	a280	2,579	**2,579**	0.00	24.12	**2,579**	0.00	1.35
33	pr299	48,191	**48,221**	0.06	33.59	48,266	0.16	1.69
34	lin318	42,029	**42,158**	0.31	33.33	42,488	1.09	1.89
35	rd400	15,281	**15,363**	0.54	53.94	15,468	1.22	4.03
36	fl417	11,861	11,870	0.08	58.10	**11,862**	0.01	4.39
37	pr439	107,217	**107,364**	0.14	74.16	108,380	1.08	4.91
38	rat575	6,773	**6,882**	1.61	130.50	6,909	2.01	9.79
39	rat783	8,806	**8,967**	1.83	278.09	8,972	1.89	22.88
40	pr1002	259,045	264,171	1.98	588.59	**263,217**	1.61	47.50

Çizelge 5.5, önerilen yöntemin Shahab (2019) tarafından önerilen yöntem ile 40 farklı test problemi üzerinde çalıştırılması ile elde edilen karşılaştırma sonuçlarını vermektedir. Karşılaştırılan yöntem önerilen yönteme göre sadece d198, fl147 ve pr1002 problemlerinde daha iyi sonuçlar vermiştir. Bu üç test problemi için önerilen yöntemin sapma oranları sırasıyla %0.01, %0.08 ve %1.98 olarak hesaplanmıştır.

Çizelge 5.6-8, önerilen yöntemin sırasıyla Thirugnanasambandam (2019) tarafından geliştirilen KS yöntemi, Sopto ve ark. (2018) tarafından geliştirilen MGKO

yöntemi ve Hatamlou (2018) tarafından geliştirilen KD yöntemi ile 5 farklı test problemi üzerinde çalıştırılması ile elde edilen karşılaştırma sonuçlarını vermektedir. Karşılaştırılan diğer tüm yöntemler, verilen test problemleri için optimum sonuca ulaşamazken, önerilen yöntem verilen tüm test problemlerinde optimum sonuca ulaşmıştır. Hatamlou (2018) tarafından geliştirilen yöntemde elde edilen en iyi sonuçlar çizelgeye eklenirken işlem süresi olarak çalışmalarında verilen yaklaşık değerler verilmiştir.

Çizelge 5.9, önerilen yöntemin Gulcu ve ark. (2018) tarafından geliştirilen PKKO-3OPT yöntemi ile 14 farklı test problemi üzerinde çalıştırılması ile elde edilen karşılaştırma sonuçlarını vermektedir. PKKO-3OPT yöntemi eil51, berlin52, eil76, kroA100, eil101 ve lin105 test problemlerinde optimum sonuca ulaşabilirken, önerilen yöntemin bu test problemleri de dahil toplam 9 farklı test probleminde optimumu bulduğu, kalan 5 farklı test probleminde de optimuma daha fazla yaklaştığı görülmüştür.

Çizelge 5.6. Önerilen yöntemin performansının KS (Thirugnanasambandam, 2019) ile karşılaştırılması

Sıra	Veri seti	Optimum Sonuç	Önerilen Yöntem			KS		
			En İyi Sonuç	Sapma Oranı (%)	Süre	En İyi Sonuç	Sapma Oranı (%)	Süre
1	eil51	426	**426**	0.00	0.81	456.91	7.26	71.26
2	berlin52	7,542	**7,542**	0.00	0.85	7,681.45	1.85	72.96
3	st70	675	**675**	0.00	1.43	721.24	6.85	104.99
4	pr76	108,159	**108,159**	0.00	1.72	118,693.70	9.74	116.91
5	eil76	538	**538**	0.00	1.67	563.18	4.68	117.84

Çizelge 5.2. Önerilen yöntemin performansının MGKO (Sopto ve ark., 2018) ile karşılaştırılması

Sıra	Veri seti	Optimum Sonuç	Önerilen Yöntem			MGKO		
			En İyi Sonuç	Sapma Oranı (%)	Süre	En İyi Sonuç	Sapma Oranı (%)	Süre
1	eil51	426	**426**	0.00	0.81	436.29	2.42	-
2	berlin52	7,542	**7,542**	0.00	0.85	8,289.12	9.91	-
3	st70	675	**675**	0.00	1.43	800.14	18.54	-
4	eil76	538	**538**	0.00	1.67	629.24	16.96	-
5	kroA100	21,282	**21,282**	0.00	2.94	28,340.42	33.17	-

Çizelge 5.3. Önerilen yöntemin performansının KD (Hatamlou, 2018) ile karşılaştırılması

Sıra	Veri seti	Optimum Sonuç	Önerilen Yöntem			KD		
			En İyi Sonuç	Sapma Oranı (%)	Süre	En İyi Sonuç	Sapma Oranı (%)	Süre
1	eil51	426	**426**	0.00	0.81	438.95	3.04	44.39
2	berlin52	7,542	**7,542**	0.00	0.85	7,804.66	3.48	43.40
3	st70	675	**675**	0.00	1.43	721.57	6.90	45.33
4	eil76	538	**538**	0.00	1.67	579.60	7.73	46.54
5	eil101	629	**629**	0.00	2.92	717.95	14.14	45.83

Çizelge 5.4. Önerilen yöntemin performansının PKKO-3OPT (Gulcu ve ark., 2018) ile karşılaştırılması

Sıra	Veri seti	Optimum Sonuç	Önerilen Yöntem			PKKO-3OPT		
			En İyi Sonuç	Sapma Oranı (%)	Süre	En İyi Sonuç	Sapma Oranı (%)	Süre
1	eil51	426	**426**	0.00	0.81	**426**	0.00	2.39
2	berlin52	7,542	**7,542**	0.00	0.85	**7,542**	0.00	2.10
3	st70	675	**675**	0.00	1.43	676	0.15	6.97
4	eil76	538	**538**	0.00	1.67	**538**	0.00	8.18
5	kroA100	21,282	**21,282**	0.00	2.94	**21,282**	0.00	21.10
6	eil101	629	**629**	0.00	2.92	**629**	0.00	20.79
7	lin105	14,379	**14,379**	0.00	3.41	**14,379**	0.00	14.57
8	ch150	6,528	**6,528**	0.00	6.64	6,570	0.64	79.35
9	kroA200	29,368	**29,368**	0.00	13.16	29,533	0.56	213.12
10	rd400	15,281	**15,363**	0.54	53.94	15,578	1.94	1,496.00
11	fl417	11,861	**11,870**	0.08	58.10	11,972	0.94	2,046.00
12	pr439	107,217	**107,364**	0.14	74.16	108,482	1.18	2,132.00
13	rat575	6,773	**6,882**	1.61	130.50	7,003	3.40	5,004.00
14	rat783	8,806	**8,967**	1.83	278.09	9,111	3.46	14,277.00

Çizelge 5.10. Önerilen yöntemin performansının PSO–KKO–3OPT (Mahi ve ark., 2015) ile karşılaştırılması

Sıra	Veri seti	Optimum Sonuç	Önerilen Yöntem			PSO–KKO–3OPT		
			En İyi Sonuç	Sapma Oranı (%)	Süre	En İyi Sonuç	Sapma Oranı (%)	Süre
1	eil51	426	**426**	0.00	0.81	**426**	0.00	140.50
2	berlin52	7,542	**7,542**	0.00	0.85	**7,542**	0.00	170.46
3	st70	675	**675**	0.00	1.43	676	0.15	256.89
4	eil76	538	**538**	0.00	1.67	**538**	0.00	220.68
5	kroA100	21,282	**21,282**	0.00	2.94	21,301	0.09	301.32
6	eil101	629	**629**	0.00	2.92	631	0.32	302.15
7	lin105	14,379	**14,379**	0.00	3.41	**14,379**	0.00	294.35
8	ch150	6,528	**6,528**	0.00	6.64	6,538	0.15	286.90
9	kroA200	29,368	**29,368**	0.00	13.16	29,468	0.34	303.23

Çizelge 5.10, önerilen yöntemin Mahi ve ark. (2015) tarafından geliştirilen PSO–KKO–3OPT yöntemi ile 9 farklı test problemi üzerinde çalıştırılması ile elde edilen karşılaştırma sonuçlarını vermektedir. PSO–KKO–3OPT yöntemi eil51, berlin52, eil76 ve lin105 test problemlerinde optimum sonuca ulaşabilirken, önerilen yöntemin verilen tüm test problemlerinde optimumu bulduğu görülmüştür.

Çizelge 5.11, önerilen yöntemin Laha (2015) tarafından geliştirilen KGKAA yöntemi ile 23 farklı test problemi üzerinde çalıştırılması ile elde edilen karşılaştırma sonuçlarını vermektedir. KGKAA yöntemi sadece eil51, pr76 ve pr130 problemlerinde optimum sonuca ulaşabilirken önerilen yöntemin verilen 21 test probleminde optimum sonuca ulaştığı, kalan 2 test probleminde optimuma daha fazla yaklaştığı görülmüştür.

Çizelge 5.5. Önerilen yöntemin performansının KGKAA (Laha, 2015) ile karşılaştırılması

Sıra	Veri seti	Optimum Sonuç	Önerilen Yöntem			KGKAA		
			En İyi Sonuç	Sapma Oranı (%)	Süre	En İyi Sonuç	Sapma Oranı (%)	Süre
1	eil51	426	**426**	0.00	0.81	**426**	0.00	4.96
2	berlin52	7,542	**7,542**	0.00	0.85	7,544	0.03	5.01
3	st70	675	**675**	0.00	1.43	677	0.30	7.95
4	pr76	108,159	**108,159**	0.00	1.72	**108,159**	0.00	8.89
5	eil76	538	**538**	0.00	1.67	544	1.12	8.92
6	kroA100	21,282	**21,282**	0.00	2.94	21,285	0.01	12.73
7	kroB100	22,141	**22,141**	0.00	3.13	22,210	0.31	12.76
8	kroC100	20,749	**20,749**	0.00	3.15	20,820	0.34	12.70
9	kroD100	21,294	**21,294**	0.00	3.03	21,526	1.09	12.70
10	kroE100	22,068	**22,068**	0.00	3.16	22,146	0.35	12.67
11	eil101	629	**629**	0.00	2.92	646	2.70	12.96
12	lin105	14,379	**14,379**	0.00	3.41	14,382	0.02	13.68
13	pr107	44,303	**44,303**	0.00	3.48	44,480	0.40	14.37
14	pr124	59,030	**59,030**	0.00	4.99	**59,030**	0.00	0.00
15	bier127	118,282	**118,282**	0.00	5.51	118,882	0.51	11.56
16	ch130	6,110	**6,110**	0.00	4.91	6,159	0.80	12.59
17	pr144	58,537	**58,537**	0.00	6.55	58,615	0.13	14.85
18	ch150	6,528	**6,528**	0.00	6.64	6,624	1.47	16.07
19	kroA150	26,524	**26,524**	0.00	6.74	27,318	2.99	15.45
20	kroB150	26,130	**26,130**	0.00	7.11	26,709	2.22	15.40
21	pr152	73,682	**73,682**	0.00	7.50	74,472	1.07	15.98
22	rat195	2,323	**2,326**	0.13	15.45	2,365	1.81	24.04
23	d198	15,780	**15,781**	0.01	19.41	16,042	1.66	24.60

Çizelge 5.3-11 işlem süresi açısından incelendiğinde, Shahab (2019) tarafından geliştirilen yöntem haricinde diğer tüm yöntemlerde verilen tüm veri setleri için önerilen yöntemin daha kısa işlem süresine sahip olduğu açıkça görülmektedir. Shahab (2019) çalışmasında ise karşılaştırılan tüm veri setleri için daha kısa işlem süresi verilmiştir.

6. SONUÇLAR VE ÖNERİLER

Optimizasyon problemleri arasında yaygın olarak karşılaşılan ve üzerinde çok çalışılan problemlerin en önemlilerinden birisi GSP'dir. Bu problem, farklı problemlere uyarlanabilmesi sayesinde birçok alanda kullanılabilmektedir. GSP, *n* tane şehrin bulunduğu ve bu şehirlerin birbirleri arasındaki uzaklıkların belli olduğu, her bir şehrin bir sefer ziyaret edildiği bir problemdir. Problemde en kısa rotanın bulunarak başlangıç şehrine dönülmesi hedeflenir. Şehir sayısının artması ile problemin optimum çözümü kabul edilebilir sürelerde bulunamamaktadır.

Çoğunlukla doğada meydana gelen olaylardan esinlenen metasezgisel algoritmalar, GSP'nin çözümünde kabul edilebilir iyi sonuçlar elde edilebilmesine olanak sağlamaktadır. Birçok yönüyle probleme adapte edilerek sonuçların daha iyiye gelmesini sağlayan metasezgisel yöntemler arasında GKA, KKO, TA, YAK, PSO, GA, GKO gibi algoritmalar yer almaktadır. GSP'nin etkili bir şekilde ve gerçek zamanlı olarak çözülmesi bunlar gibi yeni geliştirilecek olan yeni veya hibrit yöntemler sayesinde sağlanabilir.

Bu tez çalışmasında İKGKA algoritmasına dayanan, 3-Opt yöntemi ile geliştirilmiş yeni bir hibrit yöntem önerilmiştir. Önerilen yöntemin performansı, GSP için sıkça kullanılan TSPLIB kütüphanesindeki 41 test problemi ile ölçülmüş ve sonuçları literatürdeki son yıllarda geliştirilen yöntemler ile kıyaslanmıştır. Test edilen 41 problemin 27 tanesinde optimum sonuç bulunmuş, 5 tanesinde %0.10'dan daha küçük sapma oranı, 5 tanesinde %1.00'dan daha küçük sapma oranı ve geri kalan 4 tanesinde de %3.00'dan daha küçük sapma oranı ile optimuma yaklaşan iyi sonuçlar elde edilmiştir. Önerilen yöntem 150 şehirden daha küçük boyutlu problemlerin tümünde optimum sonucu bulmuştur. Bunun yanında, problem boyutu büyüdükçe diğer yöntemlere göre optimuma daha fazla yaklaştığı görülmektedir. Literatürde yer alan çalışmalar ile aynı problemler kullanılarak karşılaştırıldığında hem doğruluk hem de işlem süresi açısından benzer veya daha üstün sonuçlar elde edilmiştir.

Önerilen yöntemde kullanılan çeşitli parametreler üzerinde parametre optimizasyonu yapılarak daha iyi sonuçlar elde edilebilir. Aynı zamanda farklı metasezgisel yöntemler ile birleştirilerek farklı hibrit yöntemler de geliştirilebilir.

KAYNAKLAR

Al-Abaji, M. A., 2020, A literature review of cuckoo search algorithm, *Journal of Education and Practice*, 11 (8), 1-8.

Almufti, S., Marqas, R. and Asaad, R., 2019, Comparative study between elephant herding optimization (EHO) and U-turning ant colony optimization (U-TACO) in solving symmetric traveling salesman problem (STSP), *Journal of Advanced Computer Science & Technology*, 8 (2), 32.

Al-Obaidi, A. T. S., 2013, Improved scatter search using cuckoo search, *International Journal of Advanced Research in Artificial Intelligence*, 2 (2), 61-67.

Ansari, A. Q. and Katiyar, S., 2015, December, Comparison and analysis of solving travelling salesman problem using GA, ACO and hybrid of ACO with GA and CS, *2015 IEEE Workshop on Computational Intelligence: Theories, Applications And Future Directions (WCI)*, IEEE, 1-5.

Blazinskas, A., Lenkevičius, A. and Misevičius, A., 2013, Modified local search heuristics for the symmetric traveling salesman problem, *Information Technology and Control*, 42 (3), 217-230.

Chong, E. K. and Zak, S. H., 2004, An introduction to optimization, *John Wiley & Sons*.

Croes, G. A., 1958, A method for solving traveling-salesman problems, *Operations Research*, 6(6), 791-812.

Deng, Y., Xiong, J. and Wang, Q., 2021 A hybrid cellular genetic algorithm for the traveling salesman problem, *Mathematical Problems in Engineering*, 2021.

Dokeroglu, T., Sevinc, E., Kucukyilmaz, T. and Cosar, A., 2019, A survey on new generation metaheuristic algorithms, *Computers & Industrial Engineering*, 137, 106040.

Ekmekci, D., 2019, An Ant Colony Optimization Memorizing Better Solutions (ACO-MBS) for Traveling Salesman Problem. *2019 3rd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, IEEE, 1-5.

Escario, J. B., Jimenez, J. F. and Giron-Sierra, J. M., 2015, Ant colony extended: experiments on the travelling salesman problem, *Expert Systems with Applications*, 42 (1), 390-410.

Faris, H., Aljarah, I., Al-Betar, M. A. and Mirjalili, S., 2018, Grey wolf optimizer: a review of recent variants and applications, *Neural Computing and Applications*, 30 (2), 413-435.

Gao, W., 2020, New ant colony optimization algorithm for the traveling salesman problem, *International Journal of Computational Intelligence Systems*, 13 (1), 44-55.

Gutin, G. ve Punnen, A. P., 2006, The traveling salesman problem and its variations, *Springer Science & Business Media*, 12.

Gulcu, S., Mahi, M., Baykan, O. K. and 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.

Halim, A. H. and Ismail, I., 2019, Combinatorial optimization: comparison of heuristic algorithms in travelling salesman problem, *Archives of Computational Methods in Engineering, 26* (2), 367-380.

Hatamlou, A., 2018, Solving travelling salesman problem using black hole algorithm, *Soft Computing*, 22 (24), 8167-8175.

Helsgaun, K., 2009, General k-Opt submoves for the Lin–Kernighan TSP heuristic, *Mathematical Programming Computation*, 1(2), 119-163.

Ismkhan, H., 2017, Effective heuristics for ant colony optimization to handle large-scale problems, *Swarm and Evolutionary Computation*, 32, 140-149.

Joshi, A. S., Kulkarni, O., Kakandikar, G. M. and Nandedkar, V. M., 2017, Cuckoo search optimization-a review, *Materials Today: Proceedings*, 4(8), 7262-7269.

Karaboga, D. and Gorkemli, B., 2011, A combinatorial artificial bee colony algorithm for traveling salesman problem, *2011 IEEE International Symposium on Innovations in Intelligent Systems and Applications, INISTA-2011*, İstanbul, 50-53.

Karagül, K., 2014, Cuckoo search algorithm: A plastic waste collection example, *15th International Symposium on Econometrics, Operations Research and Statistics*, 1, Suleyman Demirel University, Isparta, TURKEY.

Kefi, S., Rokbani, N. and Alimi, A. M. 2016, Solving the traveling salesman problem using ant colony metaheuristic, a review, *International Conference on Hybrid İntelligent Systems,* Springer, Cham, 421-430.

Khan, I. and Maiti, M. K., 2019, A swap sequence based artificial bee colony algorithm for traveling salesman problem, *Swarm and Evolutionary Computation*, 44, 428-438.

Laha, S., 2015, A quantum-inspired cuckoo search algorithm for the travelling salesman problem. *2015 International Conference on Computing, Communication and Security (ICCCS),* IEEE. 1-6.

Liu, Y., Cao, B. and Li, H., 2020, Improving ant colony optimization algorithm with epsilon greedy and Levy flight, *Complex & Intelligent Systems*, 1-12.

Mahi, M., Baykan, Ö. K., and Kodaz, H., 2015, A new hybrid method based on particle swarm optimization, ant colony optimization and 3-Opt algorithms for traveling salesman problem, *Applied Soft Computing*, 30, 484-490.

Munlin, M. A. ve Anantathanavit, M., 2015, Hybrid K-means and Particle Swarm Optimization for symmetric Traveling Salesman Problem. *2015 IEEE 10th Conference on Industrial Electronics and Applications (ICIEA)*, 671-676.

Odili, J. B. and Mohmad Kahar, M. N., 2016, Solving the traveling salesman's problem using the african buffalo optimization, *Computational Intelligence And Neuroscience*, 2016.

Ouaarab, A., Ahiod, B., and Yang, X. S., 2014, Discrete cuckoo search algorithm for the travelling salesman problem, *Neural Computing and Applications*, 24 (7-8), 1659-1669.

Ouaarab, A., Ahiod, B., & Yang, X. S., 2014, Improved and discrete cuckoo search for solving the travelling salesman problem, *Cuckoo Search And Firefly Algorithm*, Springer, Cham, 63-84.

Peker, M., ŞEN, B. and Kumru, P. Y., 2013, An efficient solving of the traveling salesman problem: The ant colony system having parameters optimized by the Taguchi method, *Turkish Journal of Electrical Engineering & Computer Sciences*, 21 (Sup. 1), 2015-2036.

Reinelt, G., 1991, TSPLIB—a traveling salesman problem library, *ORSA Journal on Computing*, 3 (4), 376-384.

Saenphon, T., Phimoltares, S., and Lursinsap, C., 2014, Combining new fast opposite gradient search with ant colony optimization for solving travelling salesman problem, *Engineering Applications of Artificial Intelligence*, 35, 324-334.

Sahana, S. K., 2019, Hybrid optimizer for the travelling salesman problem, *Evolutionary Intelligence*, 12 (2), 179-188.

Shahab, M. L., 2019, New heuristic algorithm for traveling salesman problem, *Journal of Physics: Conference Series*, IOP Publishing, 1218 (1), 012038.

Sopto, D. S., Ayon, S. I., Akhand, M. A. H. and Siddique, N., 2018, Modified Grey Wolf Optimization to Solve Traveling Salesman Problem, *2018 International Conference on Innovation in Engineering and Technology (ICIET)*, IEEE, 1-4.

Thirugnanasambandam, K., 2019, Experimental analysis of ant system on travelling salesman problem dataset TSPLIB, *EAI Endorsed Transactions on Pervasive Health and Technology*, 5 (19).

Tian, W., Huang, C. and Wang, X., 2017, A near optimal approach for symmetric traveling salesman problem in euclidean space, *6th International Conference on Operations Research and Enterprise Systems*, 281-287.

West, D. B., 2001, Introduction to graph theory (Vol. 2), *Upper Saddle River: Prentice hall*.

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

Yu, M., 2019, A solution of TSP based on the ant colony algorithm improved by particle swarm optimization, *Discrete and Continuous Dynamical Systems-S*, 12 (4-5), 979.

Zhang, J., Hong, L. and Liu, Q, 2021, An ımproved whale optimization algorithm for the traveling salesman problem, *Symmetry*, 13 (1), 48.