**ZEHRA YILDIZ AÇIKGÜL**

**191180002**

**BM469**

# FİNAL PROJESİ RAPORU

# GENETİK ALGORİTMALAR ve PROGRAMLAMA

# İÇİNDEKİLER

**Sayfa**

# ŞEKİLLERİN LİSTESİ

**Şekil** **Sayfa**

# 1. ÖZET

Bu makale, en uzun turun uzunluğunu en aza indirmek amacıyla Çoklu Gezgin Satıcı Problemini (mTSP) çözmek için hibrit bir genetik algoritma önermektedir. Genetik algoritma, her bireyin temsili olarak bir TSP dizisi kullanır ve bireyi değerlendirmek ve verilen şehir dizisi için en uygun mTSP çözümünü bulmak için dinamik bir programlama algoritması kullanılır. Yeni bir çapraz geçiş operatörü, iki ebeveynden gelen benzer turları birleştirmek için tasarlanmıştır ve nüfusa büyük çeşitlilik sunar. Oluşturulan yavruların bir kısmı için, kesişimsiz bir çözüm elde etmek amacıyla turlar arasındaki kesişimleri tespit edip kaldırıyoruz. Bu özellikle min-maks mTSP için kullanışlıdır. Oluşturulan yavrular aynı zamanda kendi kendini uyarlayan rastgele yerel arama ve kapsamlı bir mahalle aramasıyla da geliştirilir. Algoritmamız, literatürde bulunan birden fazla kıyaslama seti ile test edildiğinde, benzer kesme süresi eşikleriyle ortalama olarak mevcut tüm algoritmalardan daha iyi performans göstermektedir. Ayrıca dört kıyaslama setinde 89 bulut sunucusundan 21'i için en iyi bilinen çözümleri geliştiriyoruz.

# 2. GİRİŞ

Gezgin Satıcı Problemi (TSP), yaygın olarak tanınan bir kombinatoryal optimizasyon problemidir. Bu problem, belirli bir şehir kümesini ve bunların mesafelerini kullanarak, her şehri tam olarak bir kez ziyaret eden minimum maliyetli Hamilton döngüsünü belirler. Taşımacılık, lojistik ve üretim alanlarında TSP'nin çok sayıda uygulaması bulunmaktadır.

Bu makale Çoklu Gezgin Satıcı Problemini (mTSP) incelemektedir. TSP'nin bir çeşidi olarak mTSP, turlarını aynı depoda başlatan ve bitiren birden fazla satıcıyı içerir. Sorunu temsil etmek için aşağıdaki grafik gösterimi kullanılmıştır. Ziyaret edilecek şehirleri temsil eden n köşeli tam yönsüz bir grafik olan G = (V, E)'yi düşünüyoruz. M'yi görev için uygun olan satıcı sayısı ve d'yi de her satıcının turunu başlattığı ve bitirdiği depo tepe noktası olarak kabul edelim. i köşesi ile j köşesi arasındaki mesafeyi cij ile gösteririz.

mTSP'nin amacı, her satıcı için bir tane olmak üzere, her şehrin satıcılardan biri tarafından tam olarak bir kez ziyaret edilmesini sağlayacak şekilde bir dizi m turu bulmaktır. mTSP'yi çözmek için iki ortak amaç fonksiyonu vardır. İlk amaç, minimum toplam mTSP olarak bilinen bir problem olan, tüm satıcıların kat ettiği toplam mesafeyi en aza indirmeyi içerir (Lin ve Kernighan, 1973). Alternatif amaç, en uzun turla satış görevlisinin kat ettiği toplam mesafeyi en aza indirmektir ve buna genellikle minimum-maksimum amaç fonksiyonu denir (França ve diğerleri, 1995). Makalemizde minimum maksimum mTSP problemini çözmeye odaklanıyoruz. .

mTSP çözümü, depo tepe noktası d'de başlayan ve biten, her satıcı için bir tane olmak üzere m ayrı TSP turu seti kullanılarak temsil edilebilir. Başka bir deyişle, V köşeleri kümesini m adet ayrık alt kümeye V1, V2, ..., Vm olarak bölmemiz gerekir; burada her bir alt küme, tek bir satıcı tarafından ziyaret edilen şehirleri temsil eder. Her satıcının turu, karşılık gelen şehir alt kümesinde Hamilton döngüsünü oluşturan bir dizi köşe noktası olarak temsil edilebilir. O halde genel çözüm, satıcıların şehirleri ziyaret etme sırasına göre m Hamilton çevriminin bir birleşimi olarak temsil edilebilir.

mTSP'lerin TSP'lerle benzer uygulamalara sahip olduğunu ve birden fazla aracının aynı anda bir dizi konumu kapsaması gereken senaryolarda tanıtıldıklarını unutmayın. mTSP, örneğin lojistik ve taşımacılıkta birden fazla araç veya sürücü için teslimat rotalarını optimize etmek amacıyla kullanılabilir. mTSP, iletişim kuleleri ve veri merkezleri gibi tesislerin yerleşimini optimize etmek için ağ tasarımında kullanılabilir. Üretimde mTSP, bir ürünü üretmek için birlikte çalışması gereken birden fazla makinenin veya robotun rotasını optimize etmek için kullanılabilir.

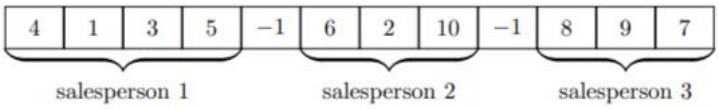
mTSP, NP-zor problem sınıfına aittir; bu, büyük veri kümeleriyle uğraşırken kesin çözümünün hesaplama açısından pratik olmadığı anlamına gelir. Bu nedenle makul bir süre içerisinde optimuma yakın çözümler elde etmek için sezgisel algoritmaların kullanılması gerekmektedir. Popüler yaklaşımlardan biri, doğal seçilim ve genetik prensiplerinden ilham alan stokastik optimizasyon algoritmaları olan genetik algoritmaları (GA) kullanmaktır.

Bu yazıda mTSP'yi çözmek için hibrit bir genetik algoritma öneriyoruz. Algoritmamız, genetik algoritmadaki her bireyin bir TSP çözümünü temsil ettiği böl ve yönet yaklaşımını kullanır. Belirli bir TSP dizisi için en uygun mTSP çözümünü belirlemek amacıyla dinamik programlamaya dayalı bir algoritma kullanırız. Nüfusun çeşitliliğini arttırmak için yeni bir geçiş fonksiyonu geliştirdik. Bu işlev, iki ebeveynden gelen benzer turlar arasında genetik bilgi alışverişini teşvik eder, böylece bireyler arasında daha fazla çeşitlilik ortaya çıkar.

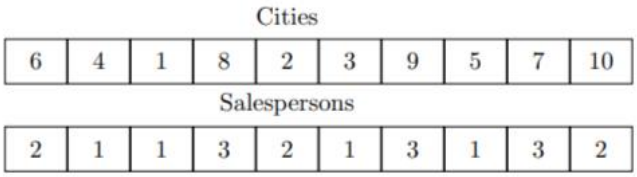
Çözümleri geliştirmek için yerel arama yöntemlerinin bir kombinasyonunu kullanıyoruz. Bu yöntemler, her bir çözümün mahallesini keşfederek turları daha da optimize etmeyi amaçlamaktadır. Ek olarak, turlar arasındaki kesişmeleri tespit etmek ve kaldırmak için özel olarak tasarlanmış bir fonksiyon sunuyoruz. Bu işlev, oluşturulan çözümlerin kesişme olmamasını sağlar; bu da özellikle mTSP'de yüksek kaliteli çözümlere ulaşmak için faydalıdır. Ancak mTSP'nin optimal çözümünde kesişimler bulunabileceğinden popülasyondaki her bireye uygulanmaz.

Makalemiz dört önemli katkı sağlamaktadır. İlk olarak mTSP ile mücadele etmek için dinamik programlama ile genetik algoritmaları birleştiren bir yöntem kullanıyoruz. Bu yaklaşım, her iki tekniğin de güçlü yanlarından yararlanarak, bu sorun bağlamında

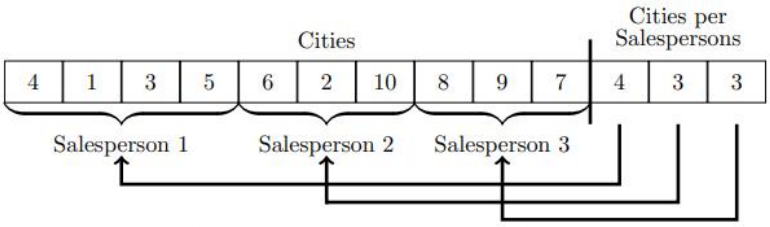
çözümleri iyileştirmeyi amaçlamaktadır. İkinci olarak, iki avantaj sunan yeni bir çaprazlama fonksiyonu sunuyoruz: hesaplama açısından ucuz ve popülasyonları çeşitlendirmede oldukça etkili. Bu, daha geniş bir çözüm alanının keşfedilmesine katkıda bulunur ve algoritmanın performansını artırır. Üçüncü olarak, turlar arasındaki kesişmelerin kaldırılmasının min-maks mTSP'de daha iyi çözümlere yol açabileceğini gösteriyoruz. Son olarak, kıyaslama örnekleri üzerinde kapsamlı deneyler yaparak algoritmamızın etkinliğini doğruluyoruz. Sonuçlarımız yalnızca mevcut algoritmaların sonuçlarını aşmakla kalmıyor, aynı zamanda birden fazla örnek için belirli bir zaman diliminde bulunan en iyi çözümlerde dikkate değer gelişmeler olduğunu da gösteriyor. Bu katkılar toplu olarak mTSP'yi çözmek için önerdiğimiz yaklaşımımızın değerini ve etkinliğini göstermektedir.



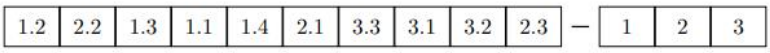

(a) Representation used in Tang et al. (2000). The tours of different salesmen are separated by −1.



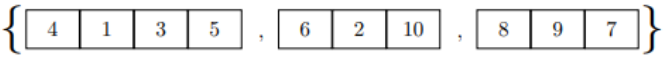

(b) Two chromosome representation. The first chromosome is the sequence of cities, the second is the index of salesmen visiting them.



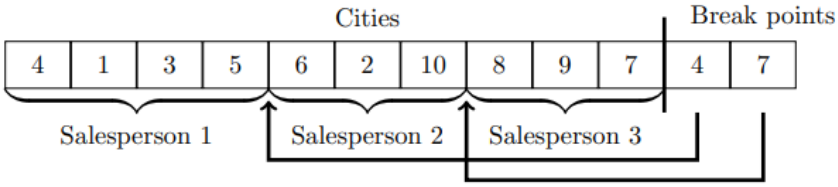

(c) Two-part chromosome representation.



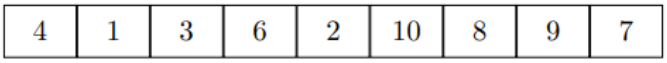

(d) GGA chromosome encoding. The value of 3.2 in the gene position nine means that salesperson three visits city nine, and it is the second city on the route. The second part comprises the group section of the chromosome




(e) Many-chromosome representation.




(f) Two-part chromosome representation with breakpoints.




(g) Chromosome representation used in this study, which is a sequence of all cities.

Şekil 1: Literatürde kullanılan farklı kromozom kodlamaları

Çoklu seyahat eden satıcı problemi (mTSP), literatürde önemli bir ilgi toplamış ve bu durum kapsamlı araştırma çabalarına yol açmıştır. Önerilen çeşitli yaklaşımlar arasında Genetik Algoritma (GA), mTSP'yi çözmek için en popüler ve başarılı yöntemlerden biri olarak ortaya çıkmıştır. Bu bölümde, mTSP ile mücadelede GA'nın kullanımına odaklanan mevcut literatürü gözden geçireceğiz ve sorunun çözümünde GA'nın verimliliğini ve etkinliğini vurgulayacağız.

Daha sonra tartışmamızı mTSP'nin çözümünde kullanılan diğer buluşsal yöntemleri araştırmaya kaydırıyoruz. Son olarak mTSP'nin çeşitlerini inceleyeceğiz ve bu çalışmalara adanmış ilgili literatürü tartışacağız. Dikkat çekici bir şekilde, bu çalışmaların çoğu, belirli problem türlerini ele almak için birincil araç olarak evrimsel

algoritmaları da kullanıyor. Bu üç kategorideki literatürün kapsamlı bir incelemesini sunarak, mTSP'yi çevreleyen araştırma ortamına ve onun çeşitli çözüm yöntemlerine ilişkin bütünsel bir anlayış sunmayı amaçlıyoruz.

Literatürde mTSP problemlerinin çözümünde genetik algoritmalar baskın yaklaşım olarak ortaya çıkmıştır. Tang ve diğerleri (2000) mTSP'yi sürekli çizelgeleme sisteminin bir parçası olarak ele alır ve her satış elemanı için turları ayırmak üzere sınırlayıcılarla birlikte düğüm dizisini içeren bir kromozom kodlama şeması kullanır (bkz. Şekil 1a). Park (2001), iki kromozomlu kodlama kullanarak Zaman Pencereli Araç Rotalama Problemini (VRPTW) çözmek için bir GA geliştirir. Birinci kromozom şehirleri, ikincisi ise o şehirleri ziyaret eden satıcıları temsil etmektedir (Şekil 1b).

Çalışma kısmen eşlenmiş çaprazlamayı (PMX) içeriyor ve dört farklı mutasyon fonksiyonunu araştırıyor. Carter ve Ragsdale (2006), iki parçalı kromozom kodlaması kullanarak mTSP'yi çözmek için bir genetik algoritma önermektedir. İlk bölüm şehirlerin sırasını temsil eder, ikinci bölüm ise her bir satış elemanının ziyaret ettiği şehir sayısını gösterir (bkz. Şekil 1c). Ayrıca Chen ve Chen (2011), yukarıda bahsedilen iki parçalı kromozom kodlamasına dayalı olarak farklı geçiş ve mutasyon kombinasyonlarını araştırmak için bir deney tasarımı yürütmektedir. Bu yaklaşım, çeşitli genetik operatörlerin mTSP'yi çözmedeki etkinliğinin değerlendirilmesine olanak tanır.

Son yıllarda, yeni geçişler veya yeni kromozom kodlamalarının tanıtılması yoluyla mTSP'yi çözmek için genetik algoritmaları geliştirmeyi amaçlayan birçok çalışma vardır. Özellikle, Yuan ve diğerleri (2013), özellikle iki parçalı kromozom temsili için uyarlanmış, TCX adında yeni bir çaprazlama önermektedir. TCX, kromozomun ilk kısmındaki çaprazlama işlemi sırasında her satıcıyı ayrı ayrı ele alarak çalışır ve sonuç olarak algoritmanın etkinliğini artırır. Kromozom kodlamaya alternatif bir yaklaşım, mTSP çözümlerini kodlamak için Gruplandırma Genetik Algoritmasını (GGA) kromozom temsilini tanıtan Brown ve diğerleri (2007) tarafından araştırılmıştır (bkz. Şekil 1d). Bu kodlamada her şehir, kendisine atanan satıcı ve satıcının turu içerisindeki siparişi ile ilgili bilgileri taşır. Singh ve Baghel (2009), kromozomun her bir parçasının belirli bir satıcı için bir turu temsil ettiği, çok kromozomlu bir temsili kullanarak bir

gruplandırma genetik algoritması geliştirdi (bkz. Şekil 1e). Ayrıca yeni bir çaprazlama stratejisi de tasarlıyorlar: her yinelemede ebeveyn rastgele seçiliyor ve seçilen ebeveynden gelen en kısa tur yavruya ekleniyor. Bu tura ait olan şehirler diğer ebeveynden çıkarılır. Bu işlem m−1 kez tekrarlanır ve ardından açgözlü veya rastgele bir yaklaşımla geri kalan şehirler eklenir. Bu çalışma kendi crossover gelişimimiz için ilk ilham kaynağı oldu. Ayrıca, Wang ve diğerleri (2017), minimum-maks mTSP'yi ele almak için Sıralı Değişken Mahalle İnişi (MASVND) kullanan çok kromozomlu bir temsile sahip bir Memetik Algoritma geliştirmiştir. Bu yenilikçi yaklaşımları birleştirerek, bu çalışmalar mTSP'nin çözümüne yönelik genetik algoritmaların devam eden ilerlemesine ve iyileştirilmesine katkıda bulunmuştur.

Bu çalışma, mTSP'yi çözmek için genetik algoritmayı, dinamik programlamayı ve yerel aramaları birleştiren hibrit bir yaklaşım önermektedir. Bu yaklaşımı kullanmanın motivasyonu, Drone'larla Gezgin Satıcı Sorununu (TSPD) çözmek için böl ve yönet yaklaşımının kullanıldığı önceki çalışmamızdan (Mahmoudinazlou ve Kwon, 2023) kaynaklanmaktadır. Bu çalışmada kamyon ve drone dizilerini içeren kromozom temsillerine sahip bir GA kullanılmıştır. Katıl adı verilen dinamik programlama tabanlı bir algoritma kullanılarak, drone'nun fırlatılması ve inişi için en uygun konum belirlenir. Bu önceki çalışmanın üzerine inşa edilen mevcut çalışma, TSP dizilerinden oluşan basit bir kromozom temsiline sahip bir GA kullanarak benzer bir yaklaşımı benimsiyor (bkz. Şekil 1g). mTSP çözümünü elde etmek amacıyla, verilen TSP dizisi için en uygun sınırlayıcıları belirlemek amacıyla Split adı verilen dinamik bir programlama algoritması kullanılır. Bu yaklaşım, karar vermenin bir kısmını GA'dan Split algoritmasına aktarır, böylece karar verme etkinliğini arttırır ve GA'nın karmaşıklığını azaltır. Bu basitleştirme GA'nın yakınsamaya daha hızlı ulaşmasını sağlar. Ayrıca bu yaklaşım iki farklı bölgede eş zamanlı arama yapılmasına olanak sağlar. GA, TSP mahallesini keşfetmeye odaklanırken, yerel arama fonksiyonları mTSP bölgesi içindeki mTSP çözümünü iyileştirmeyi amaçlamaktadır. GA, dinamik programlama ve yerel aramaları birleştiren bu hibrit yaklaşım, mevcut yöntemlere kıyasla gelişmiş yakınsama hızı, dinamik programlama yoluyla optimum karar verme ve genel çözüm kalitesini artırmak için birden fazla alanda eşzamanlı aramalar dahil olmak üzere çeşitli avantajlar sunar.

mTSP'yi çözmek için çeşitli alternatif buluşsal yöntemler önerilmiştir. Bu yöntemler sorunun üstesinden gelmek için farklı bakış açıları ve stratejiler sunar. Junjie ve Dingwei (2006), minimum toplam mTSP'yi çözmek için bir Karınca Kolonisi Optimizasyonu (ACO) algoritması uyguluyor. Benzer şekilde, Liu ve diğerleri (2009) mTSP'yi birden fazla hedefle ele almak için bir Karınca Kolonisi sistemi geliştirmiştir. Venkatesh ve Singh (2015) iki meta-sezgisel algoritma tanıtmaktadır: biri Yapay Arı Kolonisi (ABC) algoritmasını temel alır, diğeri ise İstilacı Ot Optimizasyonu (IWO) algoritmasını temel alır. Bu algoritmalar, şehirleri en uzun turdan diğer turlara açgözlü bir şekilde yeniden tahsis etmek için 2 seçenekli bir algoritma ve buluşsal yöntem içeren bir yerel arama prosedürüyle birleştirilir. Soylu (2015), mTSP için bir Genel Değişken Mahalle Araması (GVNS) önermektedir. Çalışma, arama alanını etkili bir şekilde keşfetmek için tek noktalı hareket, iki noktalı hareket, Or-opt2 hareketi, Or-opt3 hareketi, üç noktalı hareket ve 2-opt hareket dahil olmak üzere altı farklı yerel arama fonksiyonundan yararlanıyor. He ve Hao (2022), mTSP'yi çözmek için Mahalle Azaltma ile Hibrit Arama (HSNR) algoritmasını tanıttı.

HSNR algoritması, yakınsamayı sağlamak için turlar arası ve tur içi aramaları içeren iki adımlı yinelemeli bir süreci izler. Turlar arası adımda, turlar arasındaki arama uzayını keşfetmek için Tabu Aramayı temel alan iki komşuluk arama fonksiyonu, bir ekleme operatörü ve bir çapraz değişim kullanılır. Tur içi optimizasyon, bireysel turların iyileştirilmesine odaklanan TSP buluşsal EAX kullanılarak gerçekleştirilir. Zheng ve diğerleri (2022), birinci aşama olarak başlatmayı ve ikinci aşama olarak iyileştirmeyi içeren yinelenen iki aşamalı bir sezgisel algoritma (ITSHA) önermektedir. İlk aşamada, yazarlar ya rastgele açgözlü bir süreç kullanarak ya da C-ortalamalı kümelemeyi kullanarak başlangıç çözümleri yaratırlar. İyileştirme aşaması, 2 seçenekli bir hareket, bir ekleme hareketi ve bir takas hareketi içeren değişken bir komşuluk aramasını içerir.

Bildiğimiz kadarıyla Zheng ve arkadaşlarının (2022) mTSP sonuçları, bu makalenin yazıldığı sırada yaygın olarak kullanılan kıyaslama setleri arasında en iyisidir. Ancak He ve Hao'nun (2023) Memetik Algoritmasının (MA) daha uzun kesme süresiyle kendi algoritmalarından daha iyi performans gösterdiğini belirtmekte fayda var.

Bununla birlikte, yaygın olarak kullanılan kesme zamanını kullanan MA'ları için deneysel sonuçlar mevcut değildir. mTSP'yi çözmek için takviyeli öğrenmeyi kullanan bir dizi değerli çalışma yapılmıştır. Bahsetmek istediğimiz bu makalelerden bazıları aşağıdadır. Hu ve diğerleri (2020), mTSP'yi çözmek için eğitim aracı olarak GNN'leri kullanan çok etmenli takviyeli öğrenme algoritması önermektedir. Min-maks mTSP'yi çözmek için Park ve diğerleri (2023) tarafından çok etmenli takviyeli öğrenme algoritması ScheduleNet önerilmiştir. Kim ve diğerleri (2023), mTSP'nin çözümü için çapraz değişim mahallesini kullanacak bir aracıyı eğitmek amacıyla GNN'leri kullanır. Algoritmaları Neuro Cross Exchange (NCE) algoritması olarak biliniyor ve bazı kıyaslama setlerindeki sonuçları umut verici görünüyor. Bu nedenle yöntemimizi değerlendirmek için temel algoritmalardan biri olarak NCE'yi seçtik.

mTSP ailesi, dikkati hak eden bir dizi değişken içerir. Alves ve Lopes (2015) hem kat edilen toplam mesafeyi hem de en uzun turu en aza indirmeye çalışan iki amaçlı bir mTSP önermektedir. Sunulan iki yaklaşım vardır; biri çok amaçlı GA'dan oluşur, diğeri ise iki amacı birleştiren düzenli GA'dan oluşur. Bu çalışmada iki kromozom kodlaması kullanılmaktadır (Şekil 1b). Bulgularının bir sonucu olarak, çok amaçlı GA, bu iki amaçlı mTSP için daha iyi performans göstermektedir. Li ve diğerleri (2013) tarafından tanıtılan mTSP'nin bir çeşidinde, belirli şehirler belirli bir satış elemanıyla sınırlandırılmıştır. Bu soruna mTSP* adı verilir. Sorun ayrıca iki kromozom kodlamalı GA kullanılarak da çözüldü. Benzer bir problem Liu ve diğerleri (2021) tarafından Ziyaret Kısıtlı Çoklu Seyahat Eden Satıcı Problemi (VCMTSP) adı altında incelenmiştir. Bu soruna göre her şehir yalnızca önceden belirlenmiş belirli satıcıların ziyaret etmesiyle sınırlandırılmıştır. Bu çalışmada çok kromozomlu kodlamaya (Şekil 1e) ve beş yerel arama fonksiyonuna sahip bir evrimsel algoritma önerilmektedir.

Ayrıca birden fazla deponun bulunduğu bir mTSP çeşidi de vardır. Bu sorun çok depolu mTSP veya M-mTSP olarak bilinir. En yeni M-mTSP çalışmalarından bazılarının bir incelemesi sunulmaktadır. Lu ve Yue (2019) tarafından M-mTSP'nin çözümü için yeni bir karınca kolonisi optimizasyon algoritması önerilmiştir. Jiang ve diğerleri (2020), M-mTSP'yi çözmek için hibrit bir Partheno Genetik Algoritması (PGA) ve bir karınca kolonisi optimizasyon algoritması önermektedir. Çözüm, iki parçalı bir kromozom gösterimi kullanılarak kodlanır; ikinci parça, kesme noktalarını

temsil eder (Şekil 1f). Wang ve diğerleri (2020), üreme mekanizmasını istilacı ot algoritmasına dahil ederek PGA'yı iyileştirmektedir.

| Algorithm | Problem | Objective | Approach |
|---|---|---|---|
| Tang et al. (2000) | mTSP | min-sum | GA |
| Park (2001) | VRP | min-sum | GA |
| Carter and Ragsdale (2006) | mTSP | both | GA |
| Junjie and Dingwei (2006) | mTSP | min-sum | AC |
| Brown et al. (2007) | mTSP | both | GA |
| Singh and Baghel (2009) | mTSP | both | GGA |
| Liu et al. (2009) | mTSP | both | AC |
| Chen and Chen (2011) | mTSP | min-sum | GA |
| Yuan et al. (2013) | mTSP | both | GA |
| Li et al. (2013) | mTSP* | min-max | GA |
| Venkatesh and Singh (2015) | mTSP | both | BC/IWA |
| Soylu (2015) | mTSP | both | GVNS |
| Alves and Lopes (2015) | mTSP | bi-objective | GA |
| Wang et al. (2017) | mTSP | min-max | MA |
| Lu and Yue (2019) | mTSP | min-max | AC |
| Jiang et al. (2020) | M-mTSP | min-sum | AC/PGA |
| Wang et al. (2020) | M-mTSP | min-sum | PGA/IWA |
| Karabulut et al. (2021) | M-mTSP | both | ES |
| Liu et al. (2021) | VCMTSP | min-sum | GA |
| He and Hao (2022) | mTSP | both | TS/EAX |
| Zheng et al. (2022) | mTSP | both | VNS |
| He and Hao (2023) | M-mTSP | both | MA |
| This study | mTSP | min-max | HGA/DP |

Şekil 2: Literatür Taraması

## 3. YÖNTEM

Bir sonraki adım, dinamik programlama tabanlı Split algoritmamızın ayrıntılarını ve bireyi nasıl değerlendirdiğini tartışmaktır. Algoritmada kullanılan çeşitli yerel arama fonksiyonlarını keşfetmeye devam ediyoruz. Son olarak, turlar arasındaki kesişmelerin kaldırılması uygulamasını inceliyoruz ve bunun min-maks mTSP'ye faydalarını vurguluyoruz.

Bu çalışmada Vidal ve diğerleri (2012)'nin GA yapısının basitleştirilmiş versiyonu kullanılmıştır. GA'mızda herhangi bir uygun olmayan bölge bulunmamasına rağmen popülasyon kontrolü, birey sayısının $\mu$ ile $\mu + \lambda$ arasında değiştiği Vidal ve ark.(2012)'ya benzer. Algoritma 1'de prosedüre genel bir bakış sunulmaktadır. Durdurma koşulu sağlanana kadar aşağıdaki adımlar tekrarlanır (satır 2-15). Algoritma, iyileştirme yapılmayan yinelemelerin sayısı Itni'ye ulaştığında veya

algoritmanın toplam çalışma süresi önceden belirlenen kesme süresine (satır 2) ulaştığında sona erer.

İlk olarak popülasyon, Bölüm 3.1'de açıklanan uygunluk ve çeşitlilik faktörüne (satır 3) göre sıralanır. Turnuva Seçimi yöntemini kullanarak popülasyondan ebeveyn olarak iki birey seçilir (satır 4). Her ebeveyni seçmek için turnuva bireyleri rastgele seçilir ve içlerinden en iyileri seçilir. Popülasyondan iki ebeveynin seçilmesi üzerine, çocuğun kromozomunu oluşturmak için onlara Benzer Tur Çaprazlaması (STX) gerçekleştirilir (satır 5).

Üretilen kromozom, uygun bir mTSP çözümüne dönüştürülmesi gereken tüm şehirlerin bir permütasyonudur. Verilen diziyi en uygun şekilde m turlara bölmek için kromozom üzerinde Bölme algoritması kullanılır (satır 6).

**Algorithm 1** Hybrid Genetic Algorithm

1: $\Omega =$ `initial_population()` ▷ Section 3.2
2: **while** Stopping condition is not met **do**
3: `sort`$(\Omega)$ ▷ Based on fitness and diversification factor
4: Select $\omega_1$ and $\omega_2$ from $\Omega$
5: $c \leftarrow$ `crossover`$(\omega_1, \omega_2)$ ▷ Section 3.3
6: $\omega \leftarrow$ SPLIT$(c)$ ▷ Algorithms 2 and 3
7: `educate`$(\omega)$ ▷ Section 3.4
8: $\Omega_F \leftarrow \Omega_F \cup \{\omega\}$
9: **if** `size`$(\Omega) = \mu + \lambda$ **then**
10: `select_survivors`$(\Omega)$
11: **end if**
12: **if** `best`$(\Omega)$ not improved for $It_{DIV}$ iterations **then**
13: `diversify`$(\Omega)$
14: **end if**
15: **end while**
16: Return `best`$(\Omega)$

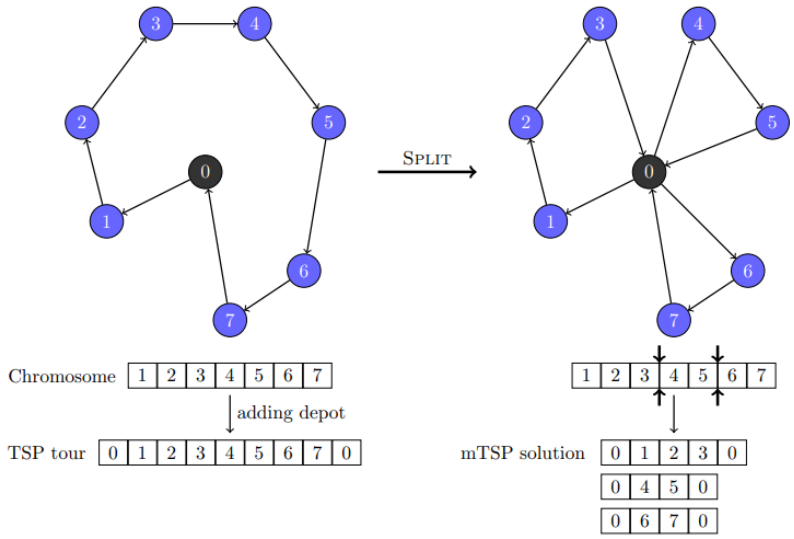
Şekil 3: Algoritma 1

Daha sonra çocuk, popülasyona eklenmeden önce yerel arama fonksiyonları ve iyileştirmelerin bir kombinasyonundan geçer (satır 7). Popülasyon büyüklüğü μ + λ'ya ulaşırsa, bireyler uygunluklarına göre sıralanacak (minimum-maksimum hedef), ilk μ bireyleri hayatta kalacak, geri kalanlar ise atılacaktır (satır 9-11). Algoritmanın yerel bir optimuma takılıp kalmasını önlemek için, Itdiv nesillerinden sonra herhangi bir iyileştirme yapılmadığında, en iyi sayıda birey tutularak ve başlangıç popülasyonunun

nesline benzer yeni kromozomlar eklenerek popülasyon çeşitlendirilmeye çalışılır. popülasyon boyutunu tekrar μ'ye yükseltin (satır 12-14).

Daha önce de belirtildiği gibi GA'mız, kromozom temsili olarak bir TSP dizisini (depo hariç) kullanır. Bu yazıda, bir TSP turunu girdi olarak alan, sınırlayıcıları tanımlayan ve ardından mTSP bağlamında verilen dizi için en uygun çözümü hesaplayan dinamik bir programlama algoritması öneriyoruz. Şekil 2'de gösterildiği gibi, her bir kromozom temsili, başlangıca ve sona depo eklendikten sonra bir TSP turudur. Şehirlerin sırasını değiştirmeden, Bölme algoritması başlangıç dizisini m farklı diziye böler ve bunların her biri depo eklendikten sonra bir tur haline gelir.

İlk olarak Prins (2004) tarafından toplam seyahat mesafesini en aza indirmek amacıyla VRP için tanıtılan Split algoritması, çalışmamızda min-maks mTSP için uyarlandı ve değiştirildi. Anlamayı kolaylaştırmak için,




Şekil 4: Bölünmüş algoritmanın çizimi

Şimdi algoritmada kullanılan gösterimi sağlıyoruz:

- $n$: the number of customer nodes
- $S = \{S_1, S_2, ..., S_n\}$: the given TSP sequence
- $m$: the number of salesmen
- $t_{S_i,S_j}$: travel time from node $S_i$ to node $S_j$
- $k$: position of the current node, $k \in \{1, 2, ..., n\}$
- $r$: the number of completed routes
- $R_k$: set of all possible numbers of completed routes at position $k$, where

$$R_k = \begin{cases} \{0\} & \text{if } k = 0, \\ \{1, ..., \min(k, m-1)\} & \text{if } 1 \leq k < n, \\ \{1, ..., m\} & \text{if } k = n \end{cases}$$

- $V_r^k$: the makespan (min-max objective) of the path from the beginning depot to the node in position $k$, while completing $r$ routes

- $P_r^k$: the predecessor of $k$, while $r$ routes has been completed

**Algorithm 2** SPLIT for *min-max* mTSP

$V_0^0 \leftarrow 0$
$V_r^k \leftarrow +\infty \quad \forall k = 1$ to $n$ , $\forall r$ in $R_k$
**for** $k = 1$ to $n$ **do**
  **for** $r \in R_{k-1}$ **do**
    **if** $V_r^{k-1} < \infty$ **then**
      $T \leftarrow 0; j \leftarrow k$
      **while** $j \leq n$ **do**
        **if** k=j **then**
          $T \leftarrow t_{0,S_j} + t_{S_j,0}$ ▷ 0 represents depot.
        **else**
          $T \leftarrow T - t_{S_{j-1},0} + t_{S_{j-1},S_j} + t_{S_j,0}$
        **end if**
        **if** $r + 1 \in R_j$ **then**
          **if** $\max(V_r^{k-1}, T) < V_{r+1}^j$ **then**
            $V_{r+1}^j \leftarrow \max(V_r^{k-1}, T)$
            $P_{r+1}^j \leftarrow k - 1$
          **end if**
        **end if**
        $j \leftarrow j + 1$
      **end while**
    **end if**
  **end for**
**end for**

Şekil 5: Algoritma 2

Algoritma 2, Split algoritmasının ayrıntılarını sunar. Amacımız, bir dizi r turu oluşturarak, verilen TSP dizisinde depodan k'inci düğüme kadar olan en uzun turu en aza indirmektir.

**Algorithm 3** Extracting the mTSP solution

```
for r = 1 to m do
    tour(r) ← ∅
end for
r ← m
j ← n
while r > 1 do
    i ← P_r^j
    for k = i + 1 : j do
        tour(r) ← tour(r) ∪ S_k
    end for
    r ← r − 1
    j ← i
end while
```

Şekil 6: Algoritma 3

Bölünmüş algoritma, bir mTSP çözümü oluşturmak için bir TSP turunda kullanılabilir. Farklı TSP turlarının farklı mTSP çözümleriyle ve dolayısıyla daha çeşitli bir popülasyonla sonuçlanması bekleniyor.

Bu çalışmada dört TSP algoritması kullanılmaktadır: Concorde'da uygulanan tam bir algoritma (Applegate ve diğerleri, 2002) ve en yakın ekleme, en uzak ekleme ve en ucuz eklemeden oluşan üç basit buluşsal algoritma. Başlangıç popülasyonunun geri kalanını oluşturmak için, mevcut TSP turlarından birini rastgele seçiyoruz, değiştiriyoruz ve ardından bireyi elde etmek için Bölme algoritmasını uyguluyoruz.

mTSP çözümünü iki ebeveynden kopyalamak için yeni bir çaprazlama fonksiyonu geliştiriyoruz. Önerilen çaprazlamamız şu şekilde çalışır: İlk ebeveynden rastgele bir tur seçilir ve ardından ikinci ebeveynden kendisiyle maksimum sayıda ortak şehre sahip bir tur seçilir.

Daha sonra seçilen turlar arasında basit bir iki noktalı geçiş uygulanır ve ortaya çıkan tur alt öğeye eklenir. Başka bir deyişle, her iki turun menzilindeki iki konum rastgele seçiliyor ve ortaya çıkan tur, turda arada kalan şehirleri ve daha az sayıda şehri ve diğer turdan seçilen konumların dışındaki şehirleri içeriyor. Bu işlem çocuk m tur yapana kadar tekrarlanmalıdır. Süreç sonunda bazı şehirler birden fazla karşımıza çıkabiliyor, bazıları ise çözümden yoksun olabiliyor. Tekrarlanan şehirler çocuğun turundan çıkarılacaktır. Son adımda açgözlü bir yaklaşımla geri kalan tüm şehirler

çocuğun turuna atanacak. Her şehir için en uzun tur dışında olası tüm eklemeler incelenmeli ve şehir en az artış sağlayacak konuma yerleştirilmelidir. Bu geçiş, şehirlerdeki en fazla benzerliği paylaşan turları birleştiriyor; dolayısıyla Benzer Tur Geçişi (STX) adı seçilmiştir.

Genetik algoritmamız hibrit olarak adlandırılıyor çünkü her yavru kendi kalitesini artırmak ve genel olarak popülasyonun gen havuzunun kalitesini artırmak için tasarlanmış işlemlere tabi tutuluyor.

Yavruların gelişiminde birbirini takip eden üç katman yer alır. İlk aşamada çözümün geometrisine ve farklı turlar arasındaki kesişimlerin kaldırılmasına dikkat edilir. İkinci katman, en uzun turu en aza indiren amaç fonksiyonuna bakılmaksızın farklı turlarda iyileştirmeler yapmaya çalışır. Son olarak üçüncü katman, en uzun turun uzunluğunun azaltılmasına odaklanıyor.

Bu çalışma sırasında, min-maks mTSP için optimal veya optimale yakın çözüm turlarının ayrı olma eğiliminde olduğunu ve özellikle depo merkezde olduğunda en az kesişmeye sahip olduğunu keşfettik.

Bu nedenle yerel aramalara başlamadan önce Premove olasılığı ile oluşturulan yavrular üzerinde kesişimleri tespit eden ve kesişen turların alt dizilerini değiştirerek bunları ortadan kaldıran bir fonksiyon çalıştırılır.

Şimdi algoritmamızın yerel aramalar gerçekleştirmek için nasıl çalıştığını tartışıyoruz. mTSP'de (ve VRP) en sık kullanılan mahalleler turlar arası mahalleler ve tur içi mahalleler olmak üzere iki kategoriye ayrılabilir. Turlar arası mahallelerde turlar arasında değişiklik yapılarak iyileştirmeler yapılırken, tur içi mahallelerde ise turlar içerisinde iyileştirmeler yapılmaktadır.

# 4. SONUÇLAR ve TARTIŞMA

HGA'mızın etkinliğini değerlendirmek için dört örnek örnek seti kullanılır. Algoritma Julia programlama dili kullanılarak 16 GB RAM'e sahip bir Mac bilgisayarda uygulanmıştır ve bir Apple M1 işlemci. HGA'mızda kullanılan parametreler $\mu = 10$, $\lambda = 20$, kturnuva = 2, Itdiv = 1000, nbest = 0,2μ, nelite = 0,2npopülasyon, Premove = 0,1, nimprv = 100, n1 local = 100, n2 local = 1000 , nclose = 0,1n.

HGA'mızı değerlendirmek ve onu literatürdeki mevcut en iyi yöntemlerle karşılaştırmak için aşağıdaki kıyaslama örnekleri kullanılır:

• SetI: Bu setin örneklerindeki şehir sayısı 50, 100 ve 200'dür. Her boyut için üç farklı sayıda satıcı vardır, bu da toplam dokuz ayar anlamına gelir. Şehirler rastgele oluşturulmuş düğümlerdir.

• SetII: Necula ve diğerleri (2015) tarafından tanımlandığı gibi bu sette toplam 16 örnek yer almaktadır.

TSPLIB'den m = 2, 3, 5 ve 7 sayıda satıcıyla çözülen eil51, berlin52, eil76 ve rat99 olmak üzere dört TSP örneği vardır.

• SetIII: Carter ve Ragsdale (2006) tarafından tanımlanan, 51, 100 ve 150 şehirli üç Öklidyen iki boyutlu simetrik TSP örneğini içeren, geniş çapta incelenen örnekler kümesi. Her örneği farklı sayıda satıcı için çözmek (m = 3, 5, 10, 20 ve 30'dan itibaren) bize toplamda 12 örnek verir.

• SetIV : Wang ve diğerleri (2017) tarafından tanımlanan ve TSPLIB'den ch150, kroA200, lin318, att532, rat783 ve cb1173 olmak üzere altı TSP örneğinden oluşan 24 örnekten oluşan bir set. Bu sette m=3, 5, 10 ve 20 adet satıcı kullanılmaktadır.

HGA'yı MA ile karşılaştırmak için (He ve Hao, 2023), Set III, Set IV ve rand100 adı verilen ek bir örnek üzerinde deneyler yapıyoruz. Bu deneyler için kullanılan kesme süresi, MA tarafından kullanıldığı şekliyle (n/100) × 4 dakikadır. Bu bölümdeki temel

algoritmalar HSNR (He ve Hao, 2022), ITSHA (Zheng ve diğerleri, 2022) ve MA'dır (He ve Hao, 2023). ITSHA için rapor edilen sonuçlar He ve Hao (2023) tarafından sağlanmaktadır ve biz onların rapor edilen sonuçlarına güvenerek bunları olduğu gibi sunuyoruz. Sonuç olarak, aynı analizi HGA için de sözde Öklid mesafelerini kullanarak yapıyoruz.

| | | | HSNR | | ITSHA | | MA | | HGA | | Gap over MA | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Instance | $m$ | BKS | best | avg | best | avg | best | avg | best | avg | best | avg |
| rand-100 | 3 | 3031.95 | **3031.95** | 3032.67 | **3031.95** | 3033.65 | **3031.95** | 3031.95 | **3031.95** | 3031.95 | 0.00 | 0.00 |
| | 5 | 2409.63 | 2411.68 | 2415.00 | 2411.68 | 2414.65 | **2409.63** | 2409.65 | **2409.63** | 2409.63 | 0.00 | 0.00 |
| | 10 | 2299.16 | **2299.16** | 2299.16 | **2299.16** | 2299.16 | **2299.16** | 2299.16 | **2299.16** | 2299.16 | 0.00 | 0.00 |
| | 20 | 2299.16 | **2299.16** | 2299.16 | **2299.16** | 2299.16 | **2299.16** | 2299.16 | **2299.16** | 2299.16 | 0.00 | 0.00 |
| ch150 | 3 | 2401.63 | 2407.34 | 2435.49 | 2407.34 | 2416.87 | **2401.63** | 2401.86 | **2401.63** | 2401.92 | 0.00 | 0.00 |
| | 5 | 1741.13 | 1741.71 | 1743.48 | **1741.13** | 1752.06 | **1741.13** | 1741.13 | **1741.13** | 1741.13 | 0.00 | 0.00 |
| | 10 | 1554.64 | **1554.64** | 1554.64 | **1554.64** | 1554.64 | **1554.64** | 1554.76 | **1554.64** | 1554.64 | 0.00 | -0.01 |
| | 20 | 1554.64 | **1554.64** | 1554.64 | **1554.64** | 1554.64 | **1554.64** | 1554.64 | **1554.64** | 1554.64 | 0.00 | 0.00 |
| kroA200 | 3 | 10691.00 | 10748.10 | 10987.69 | 10700.57 | 10819.85 | **10691.00** | 10691.41 | **10691.03** | 10700.06 | 0.00 | 0.08 |
| | 5 | 7412.12 | 7418.87 | 7494.44 | 7449.22 | 7513.67 | **7412.12** | 7414.21 | 7418.87 | 7420.23 | 0.09 | 0.08 |
| | 10 | 6223.22 | **6223.22** | 6223.22 | **6223.22** | 6223.22 | **6223.22** | 6249.10 | **6223.22** | 6223.22 | 0.00 | -0.41 |
| | 20 | 6223.22 | **6223.22** | 6223.22 | **6223.22** | 6223.22 | **6223.22** | 6223.22 | **6223.22** | 6223.22 | 0.00 | 0.00 |
| lin318 | 3 | 15663.50 | 15902.50 | 16207.05 | 15930.04 | 16088.56 | **15663.50** | 15699.92 | **15663.54** | 15714.31 | 0.00 | 0.09 |
| | 5 | 11276.80 | 11295.20 | 11596.35 | 11430.65 | 11601.67 | **11276.80** | 11291.59 | 11288.52 | 11297.35 | 0.10 | 0.05 |
| | 10 | 9731.17 | **9731.17** | 9731.17 | **9731.17** | 9731.17 | **9731.17** | 9731.17 | **9731.17** | 9731.17 | 0.00 | 0.00 |
| | 20 | 9731.17 | **9731.17** | 9731.17 | **9731.17** | 9731.17 | **9731.17** | 9731.17 | **9731.17** | 9731.17 | 0.00 | 0.00 |
| att532 | 3 | 9966.00 | 10231.00 | 10565.30 | 10158.00 | 10344.50 | **9966.00** | 10064.00 | 9973.00 | 10038.05 | 0.07 | -0.26 |
| | 5 | 6986.00 | 7067.00 | 7334.00 | 7067.00 | 7156.80 | 6986.00 | 7070.95 | **6950.00** | 7003.80 | -0.52 | -0.95 |
| | 10 | 5709.00 | **5709.00** | 5738.90 | 5731.00 | 5787.50 | 5770.00 | 5796.75 | **5698.00** | 5716.75 | -1.25 | -1.38 |
| | 20 | 5580.00 | **5580.00** | 5580.00 | 5583.00 | 5601.75 | **5580.00** | 5589.35 | **5580.00** | 5580.00 | 0.00 | -0.17 |
| rat783 | 3 | 3052.41 | 3187.90 | 3237.29 | 3131.99 | 3180.79 | **3052.41** | 3083.52 | 3086.70 | 3106.39 | 1.12 | 0.74 |
| | 5 | 1961.12 | 2006.46 | 2044.32 | 2018.44 | 2058.65 | 1961.12 | 1989.68 | **1959.16** | 1981.51 | -0.10 | -0.41 |
| | 10 | 1313.01 | 1334.76 | 1345.88 | 1357.65 | 1381.69 | 1313.01 | 1325.54 | **1304.70** | 1321.33 | -0.63 | -0.32 |
| | 20 | 1231.69 | **1231.69** | 1231.69 | **1231.69** | 1231.84 | **1231.69** | 1235.37 | **1231.69** | 1231.69 | 0.00 | -0.30 |
| pcb1173 | 3 | 19569.50 | 20813.80 | 21144.92 | 20288.75 | 21144.92 | **19569.50** | 19858.77 | 19724.24 | 20016.94 | 0.79 | 0.80 |
| | 5 | 12406.60 | 13032.30 | 13216.99 | 12816.55 | 13216.99 | **12406.60** | 12636.49 | 12517.65 | 12655.64 | 0.90 | 0.15 |
| | 10 | 7623.59 | 7758.26 | 7897.20 | 7801.18 | 7910.09 | 7623.59 | 7745.00 | **7512.43** | 7617.21 | -1.46 | -1.65 |
| | 20 | 6528.86 | **6528.86** | 6528.86 | **6528.86** | 6534.75 | **6528.86** | 6548.87 | **6528.86** | 6528.86 | 0.00 | -0.31 |
| Average | | 6922.17 | 7042.20 | 7139.50 | 7016.30 | 7115.04 | 6924.71 | 6967.85 | 6928.72 | 6962.13 | -0.03 | -0.15 |

Şekil 7: (n/100) × 4 dakika kesme süresine sahip HGA sonuçları

En iyi sütunlardaki kalın değerler, hesaplama süresine bakılmaksızın literatürde en iyi bilinen çözümler olan BKS'ye eşit veya ondan daha iyi olan değerlerdir. Gölgeli değerler HGA tarafından elde edilen yeni en iyi çözümlerdir.

Bu makale, minimum-maksimum amaç fonksiyonuyla çoklu seyahat eden satıcı problemini (mTSP) çözmek için hibrit bir genetik algoritma (HGA) sundu. HGA, çözümleri temsil etmek için TSP dizilerini kullandı ve her dizi için en uygun mTSP çözümünü belirlemek amacıyla Split adı verilen dinamik programlama tabanlı bir algoritma kullandı. Yavrular için turlar oluşturmak amacıyla, ana çözümlerden benzer turları birleştiren STX adı verilen yeni bir çaprazlama algoritması tasarlandı.

HGA, yerel arama olarak farklı mahalleleri kullanan çözümleri daha da geliştirdi. Ayrıca makale, farklı turlar arasındaki kesişmeleri tespit etmek ve kaldırmak için yeni bir yaklaşım ortaya koydu ve bunun minimum-maksimum hedeflere sahip mTSP'ler için oldukça etkili olduğu kanıtlandı. HGA, toplam 89 örnekten oluşan dört farklı veri kümesinde değerlendirildi. 89 problemin 76'sında HGA'nın en iyi çözümü, mevcut en iyi bilinen çözümlerle ya eşleşti ya da onları aştı. Ayrıca HGA, 21 örnekte en iyi bilinen çözümleri geliştirdi. Ayrıca HGA, temel algoritmayla karşılaştırıldığında 89 problemin 78'inde eşit veya üstün ortalama performans gösterdi.

Genel olarak makale, mTSP'yi minimum-maksimum bir hedefle çözmek için TSP dizilerini, dinamik programlamayı, yeni çaprazlamayı, yerel aramayı ve kesişim tespit tekniklerini kullanan hibrit bir genetik algoritma sundu. Deneysel sonuçlar, önerilen yaklaşımın yüksek kaliteli çözümlere ulaşma ve hem en iyi hem de ortalama performans açısından temel algoritmalardan daha iyi performans gösterme konusundaki etkinliğini gösterdi.

Mevcut bulgulara ek olarak, minimum-maksimum çoklu seyahat eden satıcı problemi (mTSP) alanında gelecekteki araştırmalar için umut verici birkaç yol bulunmaktadır. Potansiyel yönlerden biri, önerilen hibrit genetik algoritmanın (HGA) çok depolu mTSP'yi yönetecek şekilde genişletilmesidir. Bu, birden fazla depoyu barındırmak için Bölme algoritmasında ve çaprazlama işlevlerinde değişiklikler yapılmasını gerektirecektir. Ayrıca, gelişen drone teknolojisi alanı göz önüne alındığında, drone'ların mTSP'ye ajan olarak entegrasyonunu keşfetme fırsatı da var. Bu, limited flight time gibi zorlukların ele alınmasını içerecektir.

# KAYNAKLAR


1. Alves, R. M., C. R. Lopes. 2015. Using genetic algorithms to minimize the distance and balance the routes for the multiple traveling salesman problem. 2015 IEEE Congress on Evolutionary Computation (CEC). IEEE, 3171–3178.

2. Applegate, D., W. Cook, S. Dash, A. Rohe. 2002. Solution of a min-max vehicle routing problem. INFORMS Journal on Computing 14(2) 132–143.

3. Brown, E. C., C. T. Ragsdale, A. E. Carter. 2007. A grouping genetic algorithm for the multiple traveling salesperson problem. International Journal of Information Technology & Decision Making 6(02) 333–347.

4. Carter, A. E., C. T. Ragsdale. 2006. A new approach to solving the multiple traveling salesperson problem using genetic algorithms. European Journal of Operational Research 175(1) 246–257.

5. Chen, S.-H., M.-C. Chen. 2011. Operators of the two-part encoding genetic algorithm in solving the multiple traveling salesmen problem. 2011 International Conference on Technologies and Applications of Artificial Intelligence. IEEE, 331–336.

6. França, P. M., M. Gendreau, G. Laporte, F. M. M¨uller. 1995. The m-traveling salesman problem with min-max objective. Transportation Science 29(3) 267–275.

7. He, P., J.-K. Hao. 2022. Hybrid search with neighborhood reduction for the multiple traveling salesman problem. Computers & Operations Research 142 105726.

8. He, P., J.-K. Hao. 2023. Memetic search for the min-max multiple traveling salesman problem with single and multiple depots. European Journal of Operational Research 307(3) 1055–1070.

9. Helsgaun, K. 2017. An extension of the lin-kernighan-helsgaun tsp solver for constrained traveling salesman and vehicle routing problems.

10. Roskilde: Roskilde University 12. Hu, Y., Y. Yao, W. S. Lee. 2020. A reinforcement learning approach for optimizing multiple traveling salesman problems over graphs. Knowledge-Based Systems 204 106244. Jiang, C., Z.

11. Wan, Z. Peng. 2020. A new efficient hybrid algorithm for large scale multiple traveling salesman problems. Expert Systems with Applications 139 112867.

12. Junjie, P., W. Dingwei. 2006. An ant colony optimization algorithm for multiple travelling salesman problem. First International Conference on Innovative Computing, Information and ControlVolume I (ICICIC'06), vol. 1. IEEE, 210–213.

13. Karabulut, K., H. Oztop, L. Kandiller, M. F. Tasgetiren. 2021. Modeling and optimization of ¨ multiple traveling salesmen problems: An evolution strategy approach. Computers & Operations Research 129 105192.

14. Kim, M., J. Park, J. Park. 2023. Learning to CROSS exchange to solve min-max vehicle routing problems. The Eleventh International Conference on Learning Representations. URL https: //openreview.net/forum?id=ZcnzsHC10Y.

15. Larranaga, P., C. M. H. Kuijpers, R. H. Murga, I. Inza, S. Dizdarevic. 1999. Genetic algorithms for the travelling salesman problem: A review of representations and operators. Artificial Intelligence Review 13 129–170.

16. Li, J., Q. Sun, M. Zhou, X. Dai. 2013. A new multiple traveling salesman problem and its genetic algorithm-based solution. 2013 IEEE International Conference on Systems, Man, and Cybernetics . IEEE, 627–632.

17. Lin, S., B. W. Kernighan. 1973. An effective heuristic algorithm for the traveling-salesman problem. Operations Research 21(2) 498–516.

18. Liu, W., S. Li, F. Zhao, A. Zheng. 2009. An ant colony optimization algorithm for the multiple traveling salesmen problem. 2009 4th IEEE Conference on Industrial Electronics and Applications. IEEE, 1533–1537. Liu, X.-X., D.

19. Liu, Q. Yang, X.-F. Liu, W.-J. Yu, J. Zhang. 2021. Comparative analysis of five local search operators on visiting constrained multiple traveling salesmen problem. 2021 IEEE Symposium Series on Computational Intelligence (SSCI). IEEE, 01–08.

20. Lu, L.-C., T.-W. Yue. 2019. Mission-oriented ant-team aco for min–max mtsp. Applied Soft Computing 76 436–444. Mahmoudinazlou, S., C. Kwon. 2023. A hybrid genetic algorithm with type-aware chromosomes for traveling salesman problems with drone. arXiv preprint arXiv:2303.00614 .

21. Necula, R., M. Breaban, M. Raschip. 2015. Tackling the bi-criteria facet of multiple traveling salesman problem with ant colony systems. 2015 IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI). IEEE, 873–880.

22. Park, J., C. Kwon, J. Park. 2023. Learn to solve the min-max multiple traveling salesmen problem with reinforcement learning. Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems. AAMAS '23, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 878–886.

23. Park, Y.-B. 2001. A hybrid genetic algorithm for the vehicle scheduling problem with due times and time deadlines. International Journal of Production Economics 73(2) 175–188.

24. Perron, L., V. Furnon. 2023. OR-Tools v9.6. URL https://developers.google.com/optimization/.

25. Prins, C. 2004. A simple and effective evolutionary algorithm for the vehicle routing problem. Computers & Operations Research 31(12) 1985–2002.

26. Singh, A., A. S. Baghel. 2009. A new grouping genetic algorithm approach to the multiple traveling salesperson problem. Soft Computing 13 95–101.

27. Soylu, B. 2015. A general variable neighborhood search heuristic for multiple traveling salesmen problem. Computers & Industrial Engineering 90 390–401.

28. Tang, L., J. Liu, A. Rong, Z. Yang. 2000. A multiple traveling salesman problem model for hot rolling scheduling in shanghai baoshan iron & steel complex. European Journal of Operational Research 124(2) 267–282.

29. Venkatesh, P., A. Singh. 2015. Two metaheuristic approaches for the multiple traveling salesperson problem. Applied Soft Computing 26 74–89.

30. Vidal, T., T. G. Crainic, M. Gendreau, N. Lahrichi, W. Rei. 2012. A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. Operations Research 60(3) 611–624.

31. Wang, Y., Y. Chen, Y. Lin. 2017. Memetic algorithm based on sequential variable neighborhood descent for the min-max multiple traveling salesman problem. Computers & Industrial Engineering 106 105–122.

32. Wang, Z., X. Fang, H. Li, H. Jin. 2020. An improved partheno-genetic algorithm with reproduction mechanism for the multiple traveling salesperson problem. IEEE Access 8 102607–102615.

33. Yuan, S., B. Skinner, S. Huang, D. Liu. 2013. A new crossover approach for solving the multiple travelling salesmen problem using genetic algorithms. European Journal of Operational Research 228(1) 72–82.

34. Zheng, J., Y. Hong, W. Xu, W. Li, Y. Chen. 2022. An effective iterated two-stage heuristic algorithm for the multiple traveling salesmen problem. Computers & Operations Research 143 105772.