**T.C.**
**SAKARYA ÜNİVERSİTESİ**
**FEN BİLİMLERİ ENSTİTÜSÜ**

# BULANIK C VE K-ORTALAMALARLA BAŞLANGIÇ ÇÖZÜMÜ OLUŞTURULMUŞ TAVLAMA BENZETİMİ HİBRİT ALGORİTMASI İLE KAPASİTE KISITLI ARAÇ ROTALAMA PROBLEMİNİN ÇÖZÜMÜ

**YÜKSEK LİSANS TEZİ**

**Ahmet Fatih EKER**

**Enstitü Anabilim Dalı** : **ENDÜSTRİ MÜHENDİSLİĞİ**

**Tez Danışmanı** : **Prof. Dr. İbrahim ÇİL**

**Eylül 2020**

# BEYAN

Tez içindeki tüm verilerin akademik kurallar çerçevesinde tarafımdan elde edildiğini, görsel ve yazılı tüm bilgi ve sonuçların akademik ve etik kurallara uygun şekilde sunulduğunu, kullanılan verilerde herhangi bir tahrifat yapılmadığını, başkalarının eserlerinden yararlanılması durumunda bilimsel normlara uygun olarak atıfta bulunulduğunu, tezde yer alan verilerin bu üniversite veya başka bir üniversitede herhangi bir tez çalışmasında kullanılmadığını beyan ederim.

Ahmet Fatih EKER

06.08.2020

# TEŞEKKÜR

Lisans ve Yüksek Lisans eğitimim boyunca değerli bilgi ve deneyimlerinden yararlandığım, her konuda bilgi ve desteğini almaktan çekinmediğim, araştırmanın planlanmasından yazılmasına kadar tüm aşamalarında yardımlarını esirgemeyen, teşvik eden, aynı titizlikte beni yönlendiren değerli danışman hocam Prof. Dr. İbrahim ÇİL'e teşekkürlerimi sunarım.

Ayrıca çeviride bana yardımcı olan İngilizce öğretmenim Sedat KALKAN'a ve çalışmanın son kontrollerini yapan kardeşim Recep EKER'e teşekkür ederim.

# İÇİNDEKİLER

# SİMGELER VE KISALTMALAR LİSTESİ

| | |
|---|---|
| $\|S\|$ | : S Alt Kümesindeki Elemanların Sayısı |
| AARP | : Açık Araç Rotalama Problemi |
| ABD | : Amerika Birleşik Devletleri |
| AP | : Atama Problemi |
| ARP | : Araç Rotalama Problemi |
| ARUAP | : Açgözlü Randomize Uyarlanabilir Arama Prosedürü |
| BCO | : Bulanık C-ortalamalar |
| $c_{ij}$ | : i ve j Arasındaki Maliyet |
| $C_j$ | : j Kümesinin Merkezi |
| ÇGSP | : Çoklu Gezgin Satıcı Problemi |
| $d_{ij}$ | : i ve j Arasındaki Mesafe |
| E | : Enerji Miktarı |
| $e_i$ | : En Erken Kabul Zamanı |
| EKYA | : En Kısa Yayılan Ağaç |
| GA | : Genetik Algoritma |
| GSP | : Gezgin Satıcı Problemi |
| GTARP | : Geri Taşımalı Araç Rotalama Problemi |
| GTZPARP | : Geri Taşımalı ve Zaman Pencereli Araç Rotalama Problemi |
| HK | : Hiyerarşik Kümeleme |
| HKDYAK | : Hiyerarşileri Kullanarak Dengeli Yinelemeli Azaltma ve Kümeleme |
| KARP | : Kapasiteli Araç Rotalama Problemi |
| $^{k}B$ | : Fiziksel Boltzmann Sabiti |
| KKO | : Karınca Kolonisi Optimizasyonu |
| KKDH | : Kendi Kendini Düzenleyen Harita |
| KRP | : Konum Rotalama Problemi |
| $l_i$ | : En Geç Kabul Zamanı |

MEB : Medoidlerin Etrafında Bölümleme

MKKARP : Mesafe Kısıtlı Kapasiteli Araç Rotalama Problemi

$q^0$ : Başlangıç Sıcaklığı

$q_i$ : i Müşterisinin Talebi

$q^{max}$ : Maksimum Yineleme Sayısı

SKARP : Simetrik Kapasiteli Araç Rotalama Problemi

STKARP : Stokastik Talepli Kapasiteli Araç Rotalama Problemi

T : Mevcut Mutlak Sıcaklık

TA : Tabu Araması

TB : Tavlama Benzetimi

TDARP : Topla-Dağıt Araç Rotalama Problemi

TDZPARP : Topla-Dağıt ve Zaman Pencereli Araç Rotalama Problemi

$t^{max}$ : Maksimum Yineleme Sayısı

$u_{ij}$ : xi Nesnesinin j Kümesine Aitlik Derecesi

$x^0$ : Başlangıç Çözümü

$x^t$ : Mevcut Çözüm

ZPARP : Zaman Pencereli Araç Rotalama Problemi

$\delta$ : Soğutma Parametresi

$\eta_{ij}$ : Yolun Önsel Olarak İstenirliği

$\tau_{ij}$ : i'den j'ye Hareketin İz Seviyesi

# ŞEKİLLER LİSTESİ

# TABLOLAR LİSTESİ

# ÖZET

Anahtar kelimeler: Araç rotalama problemi, Tavlama benzetimi, Bulanık c-ortalamalar, Optimizasyon

Bu çalışmada, popüler bir problem olan Araç Rotalama Problemi (ARP) üzerinde çalışılmıştır. Bu problemde müşteriler veya şehirler ziyaret edilmeli ve ürünler haritadaki bir noktadan başlayarak her müşteriye veya şehire taşınmalıdır. Amaç, taşıma sorununu çözerek ürünleri teslim edebilmektir. Bu problem az sayıda şehir veya müşteri ile çözülmesi kolay gibi görünse de öyle değildir. Çünkü çok fazla kısıtı sağlamak zorundadır. Dolayısıyla mevcut hesaplama gücü ile bu sorun çözülemez. Müşteri sayısı arttıkça yapılacak hesaplamalar da katlanarak artmaktadır, çünkü her müşteri için tüm kısıtlar sağlanmalı ve nispeten iyi bir çözüme kısa sürede ulaşılmalıdır.

Bu çalışmadaki problemi çözmek için meta-sezgisel bir yöntem olan Tavlama Benzetimi (TB) kullanılmıştır. Genel olarak TB algoritması, metallerin tavlama işlemini taklit eden değişken sıcaklık parametresine göre tekrarlayan bir işlemdir. Bizim çalışmamız için bu yöntemin en büyük sorunu, algoritmayı başlatmak için kullanılan başlangıç çözümünü rassal olarak oluşturmasıdır. Bu sebepten dolayı optimum çözüme ulaşmak için kullanılan arama uzayı büyük olduğundan, çözüm süresi (veya iterasyon sayısı) artacaktır.

Daha iyi bir başlangıç çözümüyle optimum çözüme ulaşmak daha kısa zaman alacaktır. Ulaşmak istediğimiz optimum çözüm minimum mesafe olduğundan, başlangıç çözümünü iyileştirmek için K-ortalamalar (KO) ve Bulanık c-ortalamalar (BCO) kullanılarak rotalar kümelenmiştir. Bulanık mantık gereği, her verinin 0-1 arasında birden fazla kümeye dahil olabilmesi durumu, algoritmanın her çözümünde başlangıç çözümünü değiştireceğinden dolayı optimum çözüme yaklaşma durumu olacaktır.

Aynı veriler ve aynı parametreler kullanılarak rassal başlangıç çözümü kullanan TB ve BCO ile başlangıç çözümü iyileştirilmiş TB ile problem çözülmüştür. BCO başlangıç arama uzayını %57 oranında azaltmıştır. Dolayısıyla BCO aynı çözüm süresinde ve aynı iretasyon sayısında optimum çözüme daha yakın sonuçlar vermiştir. Çözüm sonuçları karşılaştırılmıştır.

## SOLUTION OF CAPACITATED VEHICLE ROUTING PROBLEM WITH SIMULATED ANNEALING HYBRID ALGORITHM WITH INITIAL SOLUTION CREATED WITH FUZZY C AND K-MEANS ALGORITHM

# SUMMARY


Keywords: Vehicle routing problem, Simulated annealing, Fuzzy c-means, Optimization

In this study, a popular problem, Vehicle Routing Problem (VRP), has been studied. In this problem, customers or cities should be visited and transported to the customer or city, starting from a point in the products. The aim is to deliver products by solving the transportation problem. While this problem seems easy to solve with a small number of cities or customers, it is not. Because it has to provide too many constraints. Therefore, this problem cannot be solved with the available computing power. As the number of customers increases, the calculations to be made increase exponentially, because all constraints must be met for each customer and a relatively good solution must be reached in a short time.

Simulation Annealing (SA), a meta-heuristic method, was used to solve the problem in this study. In general, the SA algorithm is a repetitive process based on the variable temperature parameter that mimics the annealing process of metals. The biggest problem with this method for our study is that it randomly generates the initial solution used to start the algorithm. For this reason, because the search space used to reach the optimum solution is large, the solution time (or number of iterations) will increase.

With a better initial solution, it will take less time to reach the optimum solution. Since the optimum solution we want to reach is the minimum distance, the routes have been clustered using K-means and Fuzzy c-mean to improve the initial solution. Due to fuzzy logic, each data can be included in more than one cluster between 0-1, since it will change the initial solution in every solution of the algorithm, there will be a case of approaching the optimum solution.

Using the same data and the same parameters, the problem was solved with SA using a random starting solution and with an initial solution optimized SA with FCM. FCM reduced the initial search space by 57%. Therefore, FCM gave results closer to the optimum solution in the same solution time and the same number of rashes. Solution results are compared.

# BÖLÜM 1. GİRİŞ

Endüstride lojistik, firmalar tarafından dikkate alınması gereken en önemli unsurlardan biridir. Birçok firma, müşteri ihtiyaçlarını en iyi şekilde karşılayabilmek için ürün ve hizmetlerinin tasarımına ve üretimine odaklansa da bu ürünler müşteriye ulaşmazsa, firma başarısız olur ve ayakta kalamaz. Bir firma, lojistiği etkin ve verimli bir şekilde yönetebiliyorsa firma maliyetlerini, enerjiyi ve zamanı, kârını etkileyecek şekilde optimize edebilir. Başka bir deyişle, firmalar bir dağıtım probleminde, ürünlerin nasıl dağıtılacağı ve lojistiğin nasıl düzgün bir şekilde organize edileceği ile ilgilenmelidirler, böylece yüksek bir kâr elde edilebilir.

Tedarik zinciri yönetiminin önemli bir parçası, tedarik zinciri içinde ürünlerin taşınması için lojistik operasyonların koordinasyonudur. Bir firmanın tedarik zincirindeki müşterilere hizmet vermek için teslimat rotalarını tasarlama görevi, literatürde Araç Rotalama Problemi (ARP) olarak bilinir. Uygulamada, müşteri talepleri müşteri lokasyonlarına varmadan önce kesin olarak bilinmeyebilir. Bu gibi durumlarda, araç kapasitesi gerçekte karşılaşılan talebi karşılayamayabilir. Bu nedenle müşterilere hizmet vermeye devam etmeden önce kapasiteyi yenilemek için merkezî bir depoya geri dönüş gerektirebilir. Bu potansiyel rota hatalarını etkili bir şekilde yöneten rotalar tasarlamak önemli bir husus ve zor bir iştir.

Son birkaç on yılda araştırmacılar, araç rotası ve kapasitesi stratejilerini etkili bir şekilde belirlemek için yöneylem araştırması, matematiksel programlama ve diğer optimizasyon tekniklerini kullanmaya odaklandılar. Şu anda ulaşım ve dağıtım yönetimi modellemesinin çekirdeği hâline gelen ARP, klasik bir optimizasyon sorununu çözmeye çalışır. Amaç; zaman kısıtlamaları, araç sayısı veya rota uzunluğu kısıtlarıyla maliyeti en aza indirmektir. Gezgin Satıcı Problemi'nin (GSP) daha genel bir sürümüdür.

Gerçek hayatta talepler, araç kapasiteleri, teslim tarihleri, araçların ortalama hızları, tüketicilerin farklı yerlerde bulunması vb. gibi dağıtım sürecinde yerine getirilmesi gereken birçok engel vardır. Bu sorunlarla başa çıkmak için, firmaların tüm engelleri ve kısıtlamaları karşılayabilmeleri için araç rotalarını optimize etmeleri gerekmektedir. Bu araç rotası optimizasyonu, ARP olarak bilinir. ARP, iyi bilinen bir kombinatoryal optimizasyon problemidir ve ilk olarak Dantzig ve Ramser tarafından tanıtılmıştır [1]. Acil durum hazırlığı ve afet yardımı, katı atık ve süt endüstrisi, sokak temizliği, engelli insanların taşınması, okul otobüsü rotalama vb. gibi birçok uygulama alanına sahiptir. ARP; malları, tüketicilere en uygun rota ile teslim etmeyi ve depoya girip çıkmak için kullanılan araç sayısını en aza indirmeyi amaçlamaktadır [2].

1964'te Clarke ve Wright [3] ARP'yi genişleterek Kapasiteli Araç Rotalama Problemi (KARP)'ni elde ettiler ve bu problem için ilk buluşsal yöntemi önerdiler. KARP'de bir depo, bir dizi müşteri (n), bir dizi kapasiteli (k) araç (m) ve her müşterinin bir talebi ($d_i$) vardır. KARP'deki görev, tüm müşterilere tam olarak bir kez hizmet verecek ve hiçbir aracın toplam kapasitesini (k) aşan bir müşteri grubunu ziyaret etmeyecek şekilde araç rotalarını oluşturmaktır [4]. Araştırma, araçların müşteriye ulaşana kadar taleplerin bilinmediği Stokastik Talepli Kapasiteli Araç Rotalama Problemi (STKARP) üzerine odaklanmıştır.

ARP'yi çözmek için kesin yöntemler, hesaplama açısından yoğundur ve müşteri sayısı büyük olduğunda makul hesaplama zamanında en uygun rotayı bulması garanti edilmez. Bu nedenle, ARP'leri çözmek için yaygın olarak kullanılan teknikler, sezgisel veya meta-sezgisel uygulamalara dayanan algoritmik yöntemlerin kullanımına odaklanır. Sezgisel yöntemler, nispeten sınırlı ancak iyi tasarlanmış bir arama alanı araştırması yapan ve genellikle nispeten kısa bir hesaplama süresi içinde kaliteli çözümler üreten optimizasyon teknikleridir. ARP'ye uygulanan sezgisel prosedürlerin örnekleri arasında Parçacık Sürü Optimizasyonu [5], Yapay Arı Kolonisi Optimizasyonu [6], Karınca Kolonisi Optimizasyonu [7], Kısıtlı Programlama Algoritmaları [8] ve Genetik Algoritmalar [9] vardır.

Bu çalışma, tüm sistem kısıtlarını göz önünde bulundurarak müşteri taleplerini karşılamak için araç filosu için en uygun rotayı bulmayı amaçlamaktadır. Kısıtlar, araç kapasitesi ve müşteri talebidir.

Araç filosu bir depodan başlar ve araç kapasitesi kısıtlarını ihlal etmeden yolundaki maksimum teslimatları karşılamaya çalışır ve ardından depoya geri döner. Tüm müşteriler sadece bir kez ziyaret edilir ve bu araçlar teslimattan sonra tekrar depoya geri döner. Bu nedenle problem, tamsayılı programlama ile çözülebilen Çoklu Gezgin Satıcı Problemi (ÇGSP)'dir.

Bölüm 2, benzer konularda literatür taramasından oluşmaktadır. Bölüm 3'te, problemin çözümünde kullanılan yöntemlere değinilmiş ve bu yöntemler hakkında genel bilgi verilmiştir. Bölüm 4'te, örnek bir problem oluşturulmuş ve sonuçları analiz edilmiştir. Bölüm 5'te, araştırma bulguları ve sonuçlar tartışılmıştır.

# BÖLÜM 2. KAYNAK ARAŞTIRMASI

1959'daki ilk çalışmadan bu yana birçok yayın yapılmış ve ARP'nin kapsamı genişletilmiştir. Son on yılda, büyük problemleri çözmek için teknik çözüm konusunda önemli ilerlemeler kaydedilmiştir. İlgi gören bir diğer konu, teknolojik yeniliklerin ARP'ye dâhil edilmesidir. Bunlar arasında küresel konumlandırma sistemleri, radyo frekansı tanımlama ve yüksek kapasiteli bilgisayarların bilgi işlemesi alanları bulunmaktadır [10].

Dantzig ve Ramer (1959) tarafından formüle edilen ARP, gezgin satıcı probleminin bir uzantısıdır. ARP, bir depoda bulunan ve bir dizi müşteriye hizmet vermek için rotalanması gereken bir dizi araç olduğunu varsayar. ARP'nin amacı, belirli bir dizi kısıtlama altında araç maliyetini ve toplam rota maliyetini en aza indirmektir [11]. ARP'nin ortaya çıkışından bu yana birçok varyasyonu gerçek dünyadaki durumlara uygulanmıştır ve literatürde kapsamlı araştırmalar mevcuttur. Bu çalışmalar için Min ve ark. [12] aşağıdaki tabloyu oluşturmuştur.

Tablo 2.1. ARP'nin sınıflandırılması.

| | Sınıflandırma standardı | Problem türleri | |
|---|---|---|---|
| 1 | Malların yönü | Tek yönlü | Çift yönlü |
| 2 | Talep türü | Deterministik | Dinamik |
| 3 | Depo sayısı | Tekli | Çoklu |
| 4 | Araç sayısı ve Tipi | Tekli | Çoklu |
| 5 | Araç kapasitesi | Belirli | Belirsiz |
| 6 | Depo kapasitesi | Belirli | Belirsiz |
| 7 | Kademe sayısı | Tekli | Çoklu |
| 8 | Zaman pencereleri | Sert ve Yumuşak | Yok |
| 9 | Amaç fonksiyonu | Tekli | Çoklu |

Balinski ve Quandt, toplam maliyeti, faaliyetler açısından tanımlamaktadırlar. Faaliyetlerdeki siparişlerin ağırlığını kullanarak toplam maliyeti hesaplamışlardır [13]. Ek kısıtlar ile birlikte problem en kolay kapasite kısıtlamalarından daha karmaşık kısıtlama kombinasyonlarına kadar çeşitli yönlerde genişletilebilir. Ghiani, Laporte, Musmanno, yaygın operasyonel kısıtların; araç sayısı, talep kapasitesi kısıtlaması, rota süresi, zaman aralıkları, müşteriye özel araç kısıtlaması ve müşteri önceliği ilişkileri olduğunu belirtmektedirler [14].

Klasik ARP modellerini çözmek için çeşitli kesin ve sezgisel algoritmalar geliştirilmiştir. Kesin algoritmaları Doğrudan Ağaç Arama Yöntemleri, Dinamik Programlama ve Tamsayılı Doğrusal Programlama olarak üç ana kategoride sınıflandırabiliriz. Alt Sınır Ataması ve Dal Sınır Algoritması, ARP ve ÇGSP arasındaki ilişkiyi kullanır. Diğer bir problem ise her müşteriyi sadece bir kez ziyaret ederken depoda başlayan ve biten bir dizi minimum maliyet rotasının oluşturulmasıdır [15]. Simetrik ARP için sabit sayıda ÇGSP'nin k-derece merkez ağacının gevşemesine dayanan başka bir çözüm algoritması geliştirilmiştir. 10-25 arasında değişen müşteri sayısı başarılı bir şekilde çözülebilmiştir [16]. Eilon, Watson-Gandy ve Christofides, ARP'yi sabit sayıda araçla çözmek için dinamik bir programlama modeli önermektedir. Optimum değeri hesaplamak için minimum maliyet yineleme işlevini bulmuşlardır [17]. Christofides, daha sonraki bir çalışmada 50 müşteri için çözümü iyileştirilmiş bir sonuç elde etmiştir [18]. Balinski ve Quandt, ARP'ler için bir Bölümleyici Kümeleme ve Sütun Oluşturma Algoritması geliştirmişlerdir. Bu algoritmanın zorluğu hesaplama yollarının maliyetli olması ve çok sayıda ikili değişkene sahip olma riskidir. Uygulanabilir çözümler çok az olduğu için sorun sıkı bir şekilde kısıtlanmış problemlerde işe yaramaktadır [13].

ARP için sezgisel algoritmalar, GSP için geliştirilen algoritmalardan türetilmiştir [19]. Clarke ve Wright'ın algoritması, ARP'leri sınırsız sayıda araçla çözer [20]. Algoritma, müşterileri maksimum tasarrufu sağlamak için birleştirir ve iyileştirme mümkün olmayana kadar devam eder. Probleme, araç sabit maliyetleri ve filo boyutları eklemek gibi yöntemin farklı varyasyonları önerilmiştir [21, 22, 23]. Wren ve Holliday, bir veya birkaç depoya sahip ARP'ler için bir tarama algoritması ve

kutupsal koordinatlarla temsil edilen Öklid düzleminde bulunan köşeler önermektedir. Algoritma, en küçük açıya sahip bir müşteriyi seçer ve mevcut kapasitesi varsa müşteriyi bir araca yerleştirir. Oluşturulan her rota için, rotalarda kat edilen toplam mesafeyi en aza indirmek için bir GSP çözülür [24]. Gendreau, Hertz ve Laporte, bir dizi çözüm ve iyileştirme adımı oluşturmak için bir Tabu Arama yöntemi geliştirdiler [25]. Algoritma tarafından oluşturulan rotalar uygulanabilir değilse, fizibilite sapmasını göstermek için amaç fonksiyonunda bir ceza eklenmektedir.

Zaman Pencereli Araç Rotalama Problemi (ZPARP)'nde, her müşteri, en erken ($e_i$) ve en geç ($l_i$) kabul zamanlarını temsil ettiği bir seyahat maliyeti, seyahat süresi, talep ve zaman penceresi ile ilişkilendirilir. Zaman pencereleri, sorunun niteliğine bağlı olarak sert veya yumuşak kısıtlamalar olarak kabul edilebilir. ZPARP için kesin yöntemleri, Lagrange gevşemeye dayalı yöntemler, sütun oluşturma ve dinamik programlama olarak üç kategoride sınıflandırabiliriz [26].

Okul otobüsü rotalama ve çizelgeleme problemi ayrıca zaman pencerelerini de dikkate alır. Swersey ve Ballard, sorunu rotalama ve zamanlama olarak incelemişlerdir [27]. Rotaların önceden oluşturulduğunu varsayarlar, bu nedenle sadece zamanlama problemi çözülür. Her rotaya tek bir otobüs atanır, öğrencileri alır ve belirli bir zaman aralığında okula varır. Problem, her otobüsün yalnızca bir okula tahsis edildiğini varsayar. Amaç kullanılan okul otobüsü sayısını en aza indirmektir. Swersey ve Ballard problemi çözerek iki tamsayı programlama modelini formüle ettiler. Formülasyonları ABD'deki bir bölgenin gerçek verilerine uyguladılar. Yöntem tatmin edici sonuçlar verdi ve ihtiyaç duyulan otobüs sayısını %25 oranında azaltmıştır.

Sezgisel algoritmalar ARP'yi çözmek için yaygın olarak kullanılmaktadır. Baker ve Schaffer, Clarke ve Wright'ın ARP sezgisel tasarruf yönteminin bir uzantısı olan Sezgisel Rota Oluşturma yöntemini tasarlamışlardır [28]. Tasarrufları, mesafe ve zamanın bir kombinasyonu olarak tanımlamışlardır. Solomon, benzer bir buluşsal yöntem tanımlamıştır ancak zaman; tasarruf işlevine dâhil edilmemiştir [29].

Landeghem, müşteriler arasındaki zamanı ölçmek için bir sezgisel tasarruf yöntemi geliştirmiştir [30]. Bu buluşsal yöntemler zaman karmaşıklığına sahiptir. r-Opt Sezgiseli, r müşterilerini diğer r müşterileri ile değiştirir. 2-Opt veya 3-Opt, zaman aralığı ihlali riskine rağmen sorunu çözebilmektedir. Potvin ve Rousseau, bu tür problemleri çözmek için 2-Opt ve r-Opt algoritmalarını geliştirmiştir [31]. Meta-sezgisel yöntemler de ARP çözümünde kullanılır. Bu yöntemler arasında Tavlama Benzetimi [32], Tabu Araması [33] ve Genetik Algoritma [34] yer alır.

Son zamanlarda, bazı daha karmaşık ARP uzantıları tüm dünyadaki akademisyenlerin büyük ilgisini çekmektedir. Bu problemlerin bir örneği de [35] ve diğer birçok makalede ele alınan Açık ARP (AARP)'dir. Belirsiz ARP, birçok akademisyenin ilgisini çeken bir başka araştırma alanıdır. Liu ve Lai [36] ve Erbao ve Mingyong [37] tarafından hazırlanan çalışmalar, belirsiz talepleri dikkate alan yayınlara örnektir.

Müşterileri ARP çözmede gruplamak literatürde yeni değildir, Klose [38] müşterileri, açık tesislere atanan rotalara karşılık gelen kümeler halinde gruplamıştır. Bu kümeleme yaklaşımı araç rotalama problemlerini sezgisel olarak çözmek için başarıyla kullanmıştır. Yoshiike ve Takefuji [39], birinci aşamanın, müşterileri gruplandırmak için maksimum nöron modelini kullanarak birleştirildiği ve ikinci aşamanın, elastik ağ modelini kullanarak her müşteri grubu için bir GSP çözme ile çalışan iki aşamalı bir algoritma önermişlerdir. Zarandi ve ark. [40], sert c-ortalamalar, bulanık c-ortalamalar ve olasılıksal c-ortalamaları, rastgele ve sezgisel olarak oluşturulmuş çözümlere karşı test etmişlerdir. Konum Rotalama Problemi (KRP)'nin ARP'nin bir uzantısı olduğunu bilerek, Barreto ve ark. [41], müşterileri ARP ve türevlerinde gruplara ayıran başka bir çalışma yapmışlardır. Çeşitli hiyerarşik ve hiyerarşik olmayan kümeleme tekniklerini sıralı bir sezgisel algoritmaya entegre etmişlerdir.

# BÖLÜM 3. MATERYAL VE YÖNTEM

## 3.1. Araç Rotalama Problemine Genel Bakış

### 3.1.1. Gezgin satıcı problemi

ARP ve GSP, 50 yıldan fazla bir süredir tüm araştırmacılar için ilgi uyandırdığından rotalama yönetimlerinin temelini oluşturmaktadır [42]. Bu ikisi, günümüzde kullanımda olan kesin ve sezgisel algoritmaların en önemli varyantlarının büyümesine katkıda bulunmuştur.

ARP'yi çözmek için en ünlü algoritmalardan biri GSP'dir. İlk formülasyon, optimizasyon probleminde en çok çalışılan 1930 yılında yapıldı. Burada temel amaç, tüm düğümleri tam olarak bir kez ziyaret eden ve başlangıç noktasına dönüşten sonra en kısa turu bulmaktır. ARP'nin diğer varyantlarını çözmek için kullanılan çok sayıda algoritma vardır [43].

Mesafelere ve bunların her düğümü nasıl karşıladığına bağlı olarak, sorun simetrik veya asimetrik olabilir. Spesifik olarak simetrik; düğüm i'den düğüm j'ye veya düğüm j'den düğüm i'ye olan mesafenin sağlandığı zamandır. Tersi olduğu durumda sorun asimetriktir.

Araştırmacılar tarafından bilinen en basit ve muhtemelen en ünlü rotalama problemleri, belirli bir V = {1, 2, ... , N} kümesindeki N noktalarının her birini A rotası boyunca yalnızca bir kez ziyaret ederek minimum toplam uzunlukta bir rota arayan GSP'dir. A rotasındaki (i, j) tüm nokta kombinasyonları arasındaki mesafe (i, j) $\in$ V ve i $\neq$ j olarak bilinir. Aşagıdaki gösterim Rardin [44] tarafından tanıtılmıştır. Karar değişkeni $x_{ij}$ aşağıdaki şekilde tanımlanır:

$$x_{ij} = \begin{cases} 1, & bir\ satıcı\ i\ noktasından\ j\ noktasına\ giderse \\ 0, & diğer\ durumlarda \end{cases}$$

Amaç Fonksiyonu;

$$\min z = \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{3.1}$$

Kısıtlar;

$$\sum_{i=1}^{N} x_{ij} = 1 \qquad \forall_j \in \{2, \dots, N\} \tag{3.2}$$

$$\sum_{j=1}^{N} x_{ij} = 1 \qquad \forall_i \in \{2, \dots, N\} \tag{3.3}$$

$$\sum_{i\in S}\sum_{j\in S} x_{ij} \leq |S| - 1 \qquad \forall S \subset V \tag{3.4}$$

$$x_{ij} \in \{0, 1\} \qquad \forall_{i,j} \in \{2, \dots, N\} \tag{3.5}$$

Problemin amacı (3.1)'de kat edilen toplam mesafeyi en aza indirir. Her bir düğüm (3.2) ve (3.3)'e göre, aynı zamanda derece kısıtlamaları olarak da anılır ve sadece bir kez ziyaret edilmesini sağlar. Alt turlar (3.4)'ün tanımlanması ile elenmiş olur. |S|, S alt kümesindeki elemanların sayısını gösterir.

İhtiyaç duyulan temel araç, GSP'de optimal çözümlere yakın çözümlere ulaşabilmek için yerel aramalar yapmaktır. Ayrıca, Lin [45], Renaud ve ark. [46] GSP'yi iyileştirici çözümler sunmuşlardır.

GSP'yi farklı algoritma ile çözen araştırmacıların örnekleri:

- Genetik algoritma

- Tavlama benzetimi

- Tabu araması

- Tabu araması ile hibrit algoritma

Önemli bir örnek de GSP'nin veri sayısının fazla olduğu problemlerde çözüm sağlamak amacıyla kullanılmış olmasıdır. Örneğin, 7397 düğüm sayısına sahip devre kartlarının üretimi, ABD'de 13509 düğüm sayısından fazla büyükşehir problemi, İsveç'te 24978 düğümlü şehir problemi gibi birçok büyük problem GSP algoritması kullanılarak çözülmüştür. Tüm bu problemler yüksek iyimser sonuçlar vermiştir.

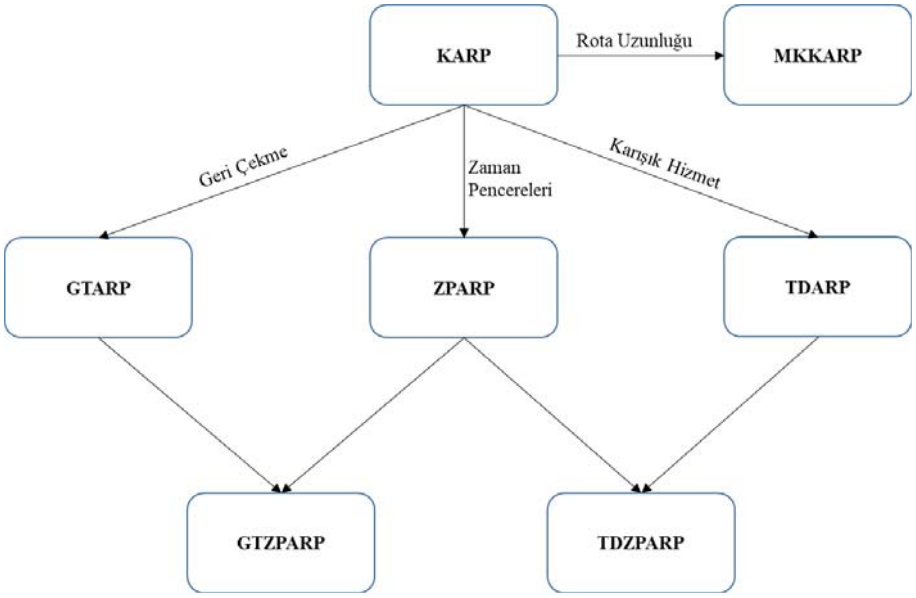
### 3.1.2. Araç rotalama problemi

ARP, dağıtım yönetiminin kalbini ifade eder. İlk olarak 1959'da Dantzig ve Ramsey tarafından duyurulmuştur. O günlerdeki asıl problem, şirketlerin ulaşmak zorunda oldukları durumların, literatürden çıkmış gerçek sorunlardan daha karmaşık olması ve zamanla değişmeseydi. Bu sorunların çözülebilmesi için, karmaşık ve pratik araç rotalama problemleri çözecek entegre bir modelleme ve optimizasyon tekniği sunmaktı. ARP, hizmetlerin ve malların müşteriye sunumundaki temel sorundur. Uygulamada ARP, taşınan malların türüne, hizmet kalitesi seviyesine ve müşterilerin ve araçların özelliklerine göre değişir. Bu nedenle ana kısıtlardan bazıları, farklı depo lokasyonlarına sahip araçlar, aynı araç tipleri ile çatışan müşteriler, müşterilere belirli zaman aralığında hizmet verebilmek, farklı günlerde teslimat süreleri ve karışık araç rotalarıdır. Çok sayıda kısıt olmasına rağmen, ARP'de minimum maliyetle bu rotayı elde etmeye odaklanılmıştır.

Algoritmalar ve yeni yazılım türleri hakkında yapılan araştırmalar, esnek optimizasyonları sistemlerin çeşitli pratik bağlamlara uyum sağlamak için değişmesine yardımcı olan sınırlı sayıda prototip problemine odaklanır [1]. ARP, fiziksel dağıtım ve lojistik kısmında temel bir özelliğe sahiptir ve bir grup müşteriye hizmet verilen deponun başlangıç ve bitiş noktası olan daha az maliyetli teslimat rotasını bulmak için hedef olarak belirlenir. Daha sonra, bir grup araç tarafından

müşterilere veya tedarikçilere hizmet etmek için kullanılan optimal bir rota setinin belirlenmesini gerektiren verilerin zor bir kombinasyonu oluşturulur [47]. Bu çözümün zor kısmı, her müşteriye hizmet verebilmek için sadece bir aracın kullanılması gerektiğidir.

ARP'nin temel uzantıları aşağıdaki şekilde gösterilmektedir:




Şekil 3.1. ARP'nin temel sınıflandırılması ve ara bağlantıları [11].

Şekilden de görebileceğimiz gibi, temel kategori KARP'dir. Çünkü diğerleri bunun bir parçasıdır. Özellikle KARP, rotalama problemleri için en yaygın ve en basit yöntem olan Gezgin Satıcı Probleminin bir parçasıdır.

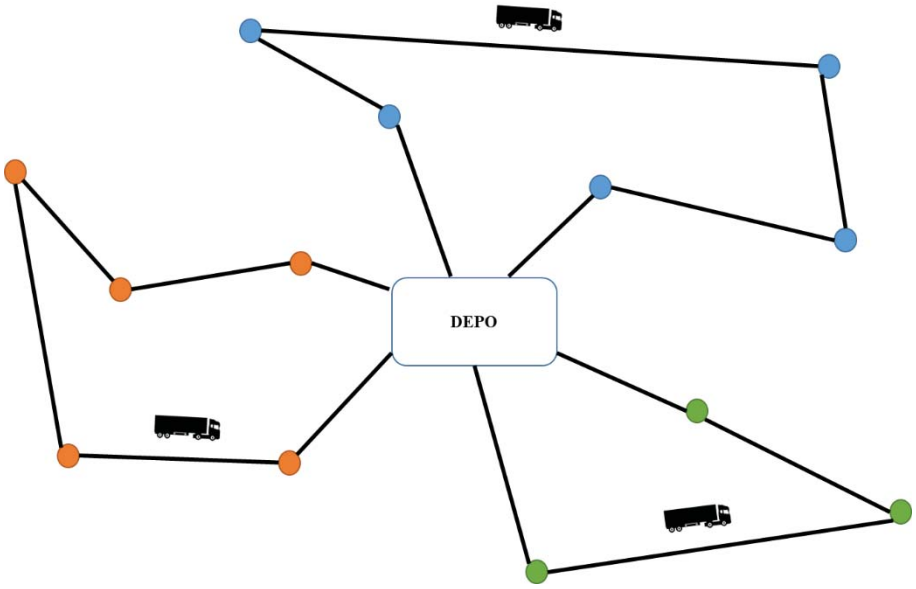
İşletme ve ekonomide imalat, konum, çizelgeleme ve rotalama problemlerini içeren karar problemleri optimizasyon problemleri olarak ifade edilebilir. Spesifik olarak, bu gibi problemler faydalı bir açıklama ile çözülemeyecek kadar zordur ve sezgisel yöntemler tercih edilen yöntemler hâline gelmiştir [48]. Müşteri sayısının gerçek hayattaki verilerle bağlantılı olduğu ve temsilî bir çözüm sağlamadığı durumlar vardır. Bu durumda, yaklaşım algoritmaları veya sezgisel yöntemler uygulanabilir [49].

Farklı rotalar için uygulanabilir bir çözüm oluşturmak için bir seçenek de NP-hard kombinatoryal problemler sınıfına ait olan ARP kullanmaktır. Bir ARP problemini çözmenin temel bileşenleri, minimum araç sayısını, minimum seyahat süresini veya seyahat edilen rotaların minimum maliyetini bulmaktır. Gerçek yaşam durumunda önemli unsurlardan bazıları, aracın kapasitesi veya her müşteriye hizmet verilmesi gereken zaman aralığı, her müşteri arasındaki ve depolar arasındaki mesafelerdir [50]. Nakliye maliyetlerini en aza indirmeye çalışmanın önemli olduğunu belirtmek de önemlidir. NP-hard olan ARP, yalnızca küçük örnekler için bir çözüm sunmaktadır.

Araç rotalama problemine çözüm bulmak için en uygun çözümü bulmaya yardımcı olacak verilere ihtiyacımız vardır. Bunun için adımlar; veri oluşturmak, bir model oluşturmak ve oluşturulan bu modelin sonuçlarını yorumlayabilmektir. Birinci adımda yaratıcılık ve ayrıntılara odaklanmak gerekir. Diğer iki adım tamamen teknik bilgi gerektirir.

ARP'de firmaların sahip olduğu müşteri sayısı, müşteri lokasyonu, talep dağılımı ve her aracın belli bir kapasite ile tamamladığı ortalama rota sayısı gibi ARP'yi tanımlayan kısıtlar vardır. Bu kısıtların tanımlanması ile otomatik olarak farklı örnek türlerine ulaşabiliriz [51].

Standart bir ARP, aynı zamanda kapasiteli araç rotalama problemi (KARP) olarak da adlandırılan yükleme kapasitesi sınırına sahip bir araç rotalama problemidir. Her araç maksimum bir kapasiteye sahiptir. Aşağıdaki Şekil 3.2.'de gösterildiği gibi en temel araç yönlendirme problemidir. Orta depo; yerel depo olabilir ve her aracın gerçek hayattaki uygulamalarda olduğu gibi kapasitesi vardır.




Şekil 3.2. Temel bir ARP örneği.

Diğer araç rotalama problemleri KARP'ye dayanmaktadır. Araç akış formülasyonları, bir rotanın optimal çözüme dâhil edilip edilmediğini belirtmek için tamsayı değişkenleri kullanan formülasyonların daha yaygın olanıdır. Aşağıda, araç rotalama problemi için standart bir matematiksel model verilmiştir. Parametreler, araç sayısı (M), müşteri sayısı (N), i. ve j. nokta arasındaki mesafe ($d_{ij}$), i. Müşterinin talep miktarı ($q_i$), araç kapasitesi (C), ($y_i$, alt turları engellemek için kullanılan rastgele değişkendir). Karar değişkeni $x_{ijk}$ aşağıdaki şekilde tanımlanır:

$$x_{ijk} = \begin{cases} 1, & k\ nolu\ araç\ i\ noktasından\ j\ noktasına\ giderse \\ 0, & diğer\ durumlarda \end{cases}$$

Amaç Fonksiyonu:

$$\min z = \sum_{i=0}^{N} \sum_{j=0}^{N} \sum_{k=1}^{M} d_{ij} x_{ijk} \qquad i \neq j \tag{3.6}$$

Kısıtlar:

$$\sum_{k=1}^{M}\sum_{j=1}^{N} x_{0jk} = M \tag{3.7}$$

$$\sum_{k=1}^{M}\sum_{j=0}^{N} x_{ijk} = 1 \qquad i \neq j \qquad i = 1, \dots, N \tag{3.8}$$

$$\sum_{k=1}^{M}\sum_{i=0}^{N} x_{ijk} = 1 \qquad i \neq j \qquad i = 1, \dots, N \tag{3.9}$$

$$\sum_{i=1}^{N} x_{i0k} \leq 1 \qquad k = 1, \dots, M \tag{3.10}$$

$$\sum_{i=1}^{N} q_i \sum_{j=0}^{N} x_{ijk} \leq C \qquad i \neq j \qquad k = 1, \dots, M \tag{3.11}$$

$$y_i - y_j + N\sum_{k=1}^{M} x_{ijk} \leq N - 1 \qquad i \neq j \qquad i, j = 1, \dots, M \tag{3.12}$$

Amaç fonksiyonu (3.6) toplam mesafenin minimize edilmesini ifade etmektedir. (3.7) nolu kısıt, başlangıç noktasından harekete geçecek araç sayısının M olduğunu, (3.8) ve (3.9) kısıtlar bir müşterinin sadece bir araç tarafından ziyaret edilmesini ve müşteriye gelen ve müşteriden çıkan rotadan yalnızca bir tanesinin kullanılması gerektiğini ifade etmektedir. (3.10) nolu kısıt bir aracın yalnızca bir rotalamada kullanılacağını ifade eder. (3.11) nolu kısıt ise araç kapasitesinin (C) aşılamayacağını ifade eder. (3.12)'de verilen kısıt ile alt turların oluşması engellenir.

#### 3.1.2.1. Kapasiteli araç rotalama problemi

Klasik KARP, bilinen talebe sahip müşterilerin sabit yükleme kapasiteli ve/veya seyahat süresi (mesafe) kısıtlaması olan bir V (>1) araç filosu ile tek bir depodan

tedarik edildiği zorlu bir kombinasyonel optimizasyon problemidir. Kapasiteli araç rotalama problemi aşağıdaki şartları karşılamak zorundadır:

- Her rota, depoda başlar ve biter.
- Her müşteri bir kez bir araç tarafından ziyaret edilir.
- Her güzergâhtaki müşterilerin toplam talepleri araç kapasitesini aşmaz.
- Her rotanın toplam süresi (mesafe), her bir aracın izin verilen seyahat süresini (mesafesini) aşmaz.
- Toplam rotalama maliyeti (zaman veya mesafe) en aza indirilir.

#### 3.1.2.2. Araç rotalama probleminde sezgisel yöntemler

ARP için birkaç sezgisel arama ailesi önerilmiştir. Bunlar genel olarak iki ana sınıfa ayrılabilir: Çoğunlukla 1960 ile 1990 arasında geliştirilen klasik buluşsal yöntemler ve Laporte ve ark. tarafından tartışılan ve son on yılda büyüyen meta-sezgisel yöntemler [52]. Çözümü baştan inşa eden buluşsal yöntemler, rotaları sırayla veya paralel olarak inşa ederken, modeli iyileştirici buluşsal yöntemler, nesnel değeri artırmak için mevcut çözüme farklı bildirimler uygulamaya çalışır. Bu yöntemler, sınırlı bir arama alanını keşfeder ve kısa hesaplama süreleri içinde nispeten iyi kalitede çözümler üretir.

Meta-sezgisel yöntemler, belirlenen arama alanı, en iyi alanın derin bir araştırmasını gerçekleştirmeye odaklanan ve kısmi hareketlerle yerel optimumda sıkışıp kalmaktan kaçınmak için denetleyici kontrol stratejileri tarafından yönlendirildiği klasik sezgisel iyileştirmeler olarak görülebilir. Bu yöntemlerle üretilen çözümlerin kalitesi genellikle klasik sezgisel yöntemler kullanılarak elde edilenden daha iyidir, ancak artan hesaplama süresi bu yöntemlerin seçilmesini zorlaştırabilir.

Klasik sezgisel yöntemler arasında ünlü Clarke ve Wright (1964) tasarruf algoritmaları, Gillett ve Miller (1974) sürükleme algoritmaları, Bramel ve Simchi-Levi (1995) iki fazlı sezgisel yöntemler ve Fisher ve Jayakumar (1981) küme-birinci, rota-ikinci algoritmaları bulunmaktadır. ARP için klasik ve modern sezgisel yöntemler için geniş bir anket yapılmıştır [52]. Son yıllarda, Karınca Kolonisi algoritmalarını kullanarak araç rotalama problemlerini çözmek için girişimlerde bulunulmuştur. Bullnheimer ve ark. [53], kapasite kullanımı ve tasarruflu sezgisel bir hibrit karınca koloni sistemi öne sürmüşlerdir.

### 3.1.2.3. Dal sınır yöntemi

Dal sınır yöntemi, KARP ve ana varyantlarını çözmek için son yıllarda kullanılmıştır. Laporte ve Nobert [54], kesin yöntemlere adanmış kapsamlı anketlerinde, dal sınır algoritmalarının eksiksiz ve ayrıntılı bir analizini verdiler. Simetrik Kapasiteye Sahip ARP ile Asimetrik Kapasiteye Sahip ARP arasındaki açık ayrım gerekli olmadığında, sadece KARP kullanılır. KARP, iyi bilinen bir GSP uzantısıdır ve minimum maliyetle bir dizi noktayı tam olarak bir kez ziyaret eder. Bu nedenle, KARP için pek çok kesin yaklaşım, GSP'nin kesin çözümü için yapılan kapsamlı ve başarılı çalışmalardan miras alınmıştır. 1980'lerin sonlarına kadar, KARP için en etkili kesin yaklaşımlar, esas olarak Atama Problemi (AP) ve En Kısa Yayılan Ağaç (EKYA) gibi temel kombinatoryal gevşemeleri kullanan dal sınır algoritmalarıydı.

Held ve Karp [55] tarafından GSP için önerilen 1-tree Relaxation yönteminin genişletilmesiyle Simetrik Kapasiteli Araç Yönlendirme Problemi (SKARP) için genişleyen ağaçlara dayalı birkaç gevşeme önerilmiştir. Küçük boyutlu örnekleri çözebildiği kanıtlanan bu tür gevşemeye dayanan en eski dal sınır algoritması Christofides ve Mingozzi [56] tarafından önerilmiştir

### 3.1.2.4. Geri taşımalı ve zaman pencereli araç rotalama problemi

Gelinas ve ark. [57], sütun oluşturmaya dayalı dallanma ve bağlı yaklaşımlar için yeni bir dallanma stratejisi önermiştir. Bu algoritma, 100 müşteriye kadar farklı test problemlerine en uygun çözümleri bulmuştur. Kontoravdis ve Bard [58], Zaman Pencereli Araç Rotalama Problemi (ZPARP) için Açgözlü Randomize Uyarlanabilir Arama Prosedürü (ARUAP) duyurmuşlardır. Bu algoritma aynı zamanda müşteri önceliği olmadan ZPARP'yi de çözebilmektedir. Yazarlar, kümeler hâlinde dağıtılan müşterilerle ilgili sorunlar için deneysel sonuçlar bildirmişlerdir. Cheung ve Chang [59], ZPARP'yi çözmek için Etiket Eşleştirme Algorithması'nı geliştirmişlerdir. Sezgisel yaklaşımları, farklı kapasitelerde araçlar ve erken gelen araçlar için cezalar uygulanması gibi karmaşık gerçek dünya kısıtlamalarının eklenmesiyle başa çıkabilmektedir.

Geri Taşımalı ve Zaman Pencereli Araç Rotalama Problemi (GTZPARP) şu şekilde ifade edilebilir: Belirli talep, belirli zaman aralıkları ve belirli hizmet gereksinimleri (teslim alma ve/veya teslim etme) olan bir müşteri grubu homojen bir şekilde olmalıdır. Belirli bir zaman aralığı olan merkezî bir depodan başlayıp biten sabit kapasiteli araç filosu olmalı ve her hizmet gereksinimi için (teslim alma veya teslim etme), her müşteri bir araca tam olarak bir kez atanmalıdır. Müşteriler iki gruba ayrılır: Taleplerinin yerine getirilmesi gereken müşterileri ve taleplerinin karşılanması gereken müşterileri (bir müşteri hem teslim alma hem de teslim etmeyi gerektiriyorsa, iki ayrı müşteri olarak modellenir). GTZPARP'nin iki amacı vardır: Rota sayısını en aza indirmek ve tüm rotaların toplam mesafesini en aza indirmek. GTZPARP, iki durumda sınıflandırılabilir: Müşteri önceliğine sahip GTZPARP'de, tüm müşterilerine, her rotadaki herhangi bir müşterisinden önce hizmet verilmelidir. Müşteri önceliği olmayan GTZPARP'de, müşteriler belirli bir rotaya serpiştirilebilir.

Literatürde toplama ve dağıtma hizmetiyle ilgili çeşitli rotalama problemleri incelenmiştir. Savelsbergh ve Sol [60], Genel Topla ve Dağıt Problemi adını verdikleri kategoride genel bir rotalama problemini tanımlamaktadır. Böylece

toplama ve dağıtma problemlerinin çoğu bu genel problemin özel durumları olarak ifade edilebilir.

Literatürde tartışılan klasik bir problem olan Dial-a-Ride problemi, müşterileri önceden belirlenmiş konumlardan almak ve belirli bir depodaki araçları kullanarak bilinen teslimat yerlerine taşınması durumudur. Her bir teslim alma konumu, teslim alma faaliyetine öncelik verilen konum çiftleri oluşturan bir teslimat konumu ile ilişkilendirilir. Problem; araç kapasitesi, müşteriler için maksimum seyahat süreleri ve zaman aralıklarında kısıtlamalara sahip olmasıdır. Aşağıdaki hedeflerin hiyerarşik bir şekilde en aza indirilmesi amaçlanmaktadır:

- Araç sayısı

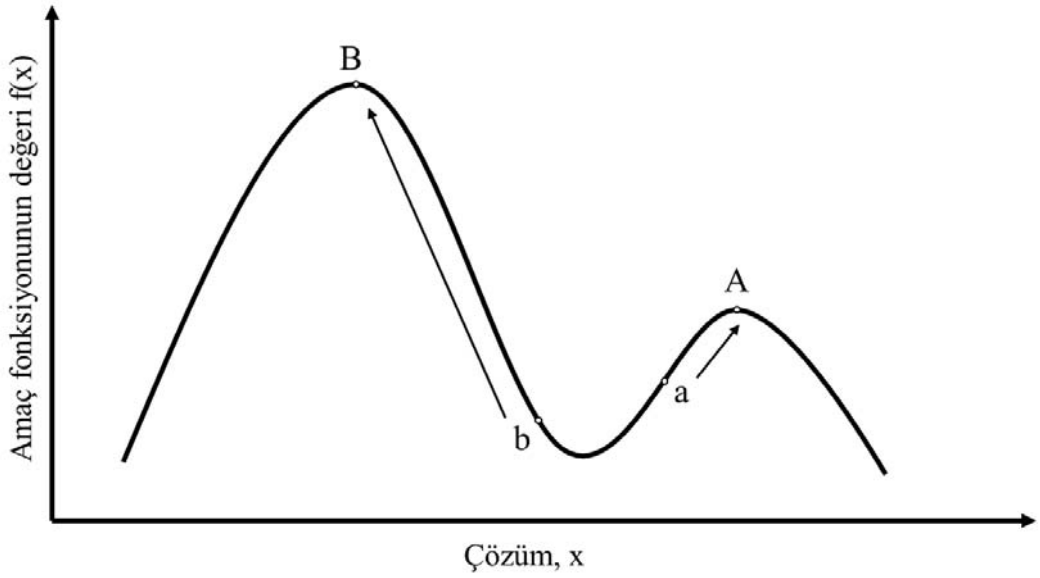
- Kat edilen toplam mesafe

- Etkili toplama ve dağıtma süreleri ile müşteriler tarafından istenen süreler arasındaki fark

Tam dinamik programlama algoritmaları, Psaraftis [61, 62] tarafından Dial-a-Ride Probleminin statik ve dinamik versiyonlarının yanı sıra zaman pencereli problemin bir varyantını çözmek için geliştirilmiştir. Desrosiers ve ark. [63], zaman pencereleri varyantı için daha verimli bir algoritma önerdi ve Psaraftis [64], Sexton ve Bodin [65] ve Sexton ve Choi [66] bu problem için sezgisel yöntemler önerdiler. Bu kategorideki diğer bir rotalama sorunu, teslimatlara öncelik verildiği ve tüm teslimatların herhangi bir teslim alma işleminden etkilenmeden önce yapılması gereken ana taşıyıcılarla ilgili problemdir. Tek bir araç olması durumunda, ana taşıyıcılı GSP tanımlanır. Gendreau ve ark. [67], bu problem için birkaç iki aşamalı buluşsal yöntem geliştirmiştir.

### 3.2. Meta-sezgisel Yöntemler

İyileştirilmiş sezgisel arama, yakın komşulukta hem uygulanabilir hem de iyileştirici bir çözüm bulunmayana kadar uygulanır. Yerleşik çözüm daha sonra yerel bir optimum olarak adlandırılır. Sezgisel yöntemlerin avantajı, optimallik garanti edilemese bile iyi uygulanabilir çözümlerin bulunabilmesidir. Dezavantajı, çözümlerin gerçekte optimal olana ne kadar yaklaştığına dair belirsizliğin var olmasıdır. İlk çözüm, bulunan yerel optimumun optimalliğini olumsuz etkileyebileceğinden, sezgisel yöntemlerin dezavantajı burada yatmaktadır.




Şekil 3.3. Global optimum ve yerel optimum noktalar.

Şekil 3.3.'teki basitleştirilmiş resme bakıldığında eğer sezgisel yöntem A'daki bir çözümle başlarsa, yalnızca A'da yerel optimuma ulaşana kadar gelişebileceğine dikkat edilmeli. Aynı sezgisel, B'deki bir çözümle başlarsa yerel değere ulaşabilir. B'de optimum olan aynı zamanda küresel optimum da olur.

Meta-sezgiler, mevcut çözüme kıyasla daha düşük objektif fonksiyon değerlerine sahip çözümlerle sonuçlanan komşuluk geçici olarak, hareketlere izin vererek küresel optimal çözümleri bulmak için algoritmaları yönlendirmek için akıllı karar verme tekniklerini kullanan ana stratejilerdir. Şekil 3.3.'te, B'ye doğru gelişmeden önce

A’dan başlayıp B’ye doğru kötüleşen bir çözümle kötüleşen hareket gösterilmektedir. Geçici olarak bozulma hareketlerini kabul etmekle ilgili bir sorun ortaya çıkmaktadır. A ve B’yi komşu çözümler olarak düşünün. Böylelikle her çözüme diğerinden tek bir tedirginlik ile ulaşılabilir. A’dan B’ye bir hareket, geçici olarak kötüleşen bir hareket olarak kabul edilebilir. Bununla birlikte, B’den A’ya bir hareket, nesnel işlevi geliştirdiği için her zaman kabul edilecektir. Bu, tek bir çözüm etrafında belirsiz döngüye yol açabilir. Özel adlandırma kuralları, analojileri ve ayrıntılı rutinleri değişiklik gösterse de tüm metasezgiseller benzer bir süreci takip eder.

- Başlatma, bir başlangıç çözümü bulma sürecidir. Genetik Algoritma gibi bazı meta-sezgiseller, uygulanabilir olması gerekmeyen rastgele oluşturulmuş ilk çözümlerle iyi performans gösterirken, Tabu Araması ilk çözümün kalitesine oldukça duyarlıdır.

- Çeşitlendirme, altta yatan sezgisel yöntemlerin çeşitlendirilmiş bir komşuluk aramasını ve böylece yerel optimum içinde sıkışıp kalmamasını sağlayan mekanizmadır.

- Yoğunlaştırma, sezgisel yöntemlerin tek bir çözüme doğru yakınlaştırmaya başlamasını sağlayan bir mekanizma. İyi bir komşuluklar belirlenir ve çözüm alanının bu alanları daha detaylı araştırılır.

Çeşitlendirme ve Yoğunlaştırma sonsuza kadar tekrar edebilir ve bu nedenle meta-sezgiyi sonlandırmak için bir durdurma kriteri gerektirir [68]. Meta-sezgisel çalışma ne kadar uzun sürerse küresel optimuma yakınsama olasılığı o kadar yüksek olur.

Temel bir meta-sezgisel, her biri tek bir yerel optimum sağlayan birden çok başlangıç çözümü ile gelişen arama sezgisel yöntemini çalıştırmayı gerektirecektir. Bu yerel optimumların en iyisi, ^x ile gösterilen yerleşik çözümdür. Böyle bir süreç, ilk çözümlerin seçimine büyük ölçüde bağlı olacaktır ve ilk çözümlerin dikkatli bir şekilde seçilip üretilmemesi durumunda daha düşük yerel optimallik sağlayabilir.

### 3.2.1 Tabu arama

TA, Glover [69] tarafından tanıtılan, iyileştirici bir çözüm arayışında komşu çözüm alanını (komşuluk) araştıran ve iyileştirme hamleleri yapıldığında yerleşik çözümü güncelleyen bellek tabanlı bir yerel arama meta-bilgisidir. Bozulan hamlelere izin verilir ve meta-sezgisel, yakın zamanda ziyaret edilen çözümlere yönelik hamleleri tabu olarak ilan ederek döngüye alır. TA'nin kapsamlı ve yeni bir incelemesi, yazarların ARP'nin zaman penceresi varyantlarına odaklandıkları Braysy ve Gendreau [70]'da bulunabilir.

Bir başlangıç çözümü $x^0$ ve bir durdurma kriteri gereklidir. Bu durumda durdurma kriteri önceden ayarlanmış maksimum yineleme sayısı $t^{max}$ olarak belirlenir. Algoritma, yineleme sayımını sıfıra ayarlayarak, başlangıç çözümünü yerleşik çözüm olarak ayarlayarak ve tabu listesini temizleyerek başlatılır. Amaç fonksiyon değeri c(x), x çözümünün bir fonksiyonu olarak ifade edilir. Mevcut çözüm $x^t$ etrafındaki komşulugu temsil eden uygun bir hareket seti $M^{xt}$ üretilir. Komşuluk, herhangi bir permütasyon yoluyla kurulur. Tabu olmayan bir hareket $\Delta x \in M$, önceden ayarlanmış yineleme sınırı içinde mevcut çözüm $x^t$'nin uygulanabilir bir komşusuna yol açmazsa veya bazı aspirasyon kriterleri karşılanırsa, meta-sezgisel sona erer ve görevdeki çözüm ^x yaklaşık optimaldir. Değilse yeni komşu geçerli çözüm olur ve tabu listesine eklenir. Mevcut çözüm, daha üstün bir hedef fonksiyon değerine sahip olan $c(x^{t+1})$ görevlinin yerini alır. Yineleme sayısı artırılır, yeni mevcut çözüm için bir hareket kümesi oluşturulur ve işlem tekrarlanır.

### 3.2.2. Tavlama benzetimi

Yerel bir arama yönteminin aksine, TB rastgele bir arama yöntemidir [71]. Optimizasyonda benzetilmiş tavlama kavramını anlamak için ilk olarak Kirkpatrick ve ark. tarafından sunulan fiziksel tavlama sisteminin benzetmesine bakmak gerekir [72]. Bir katının, örneğin çeliğin temel durumu, atomlarının veya parçacıklarının minimum enerji konfigürasyonunda düzenlendiği durumdur (katının en kararlı hâli).

Bir metalin temel durumu, fiziksel tavlama işlemi ile elde edilebilir. Metal, katıdan sıvıya dönüşümünü sağlamak için önce yüksek bir sıcaklığa ısıtılır. Bu sıcaklığa metalin erime noktası denir. Sıvı halde metal kararsızdır, parçacıklar herhangi bir set konfigürasyonunda düzenlenmediklerinden yüksek enerji sergileyerek serbestçe hareket ederler. Daha sonra parçacıkların minimum enerji düzenlemesine yavaş yavaş yerleşmesine izin vermek için sıcaklık dikkatlice düşürülür ve temel durum elde edilir.

Benzer şekilde TB, katının temel durumuna karşılık gelen bir optimizasyon probleminin amaç fonksiyonunun minimum değerini elde etmeyi amaçlamaktadır [73]. Katı maddenin herhangi bir başka hâli, optimizasyon problemi için uygun bir çözüme karşılık gelir ve katının bir hâlinin enerjisi, bir çözümün amaç fonksiyon değerine eşdeğerdir. Fiziksel sistemin sıcaklığına benzer bir kontrol parametresi q, amaç fonksiyon değerini bozan hareketlerin kabulünü düzenleyerek TB algoritmasının kademeli olarak global optimuma yakınsamasını kontrol etmek için kullanılır. Katı atomlarının küçük yer değiştirmesine benzer şekilde, t, $x^t$ aşamasındaki mevcut çözelti, optimum çözüme yaklaşırken $\Delta x$ küçük karışıklıklara uğrar.

Şekil 3.4.'te sağlanan genel TB meta-sezgisi, bir x0 başlangıç çözümü ve bir durdurma kriteri gerektirir. Alfa ve ark. [74], iyi çözümler bulmak için gereken hesaplama süresinin başlangıç çözümlerinin kalitesine duyarlı olduğunu belirtmektedir. TA algoritmasına gelince, bir yineleme limiti sayımı tmax kullanılır. Algoritma ayrıca, qmax ile gösterilen yeterli sayıda yinelemeden sonra sistemin sıcaklığını düşüren bir başlangıç sıcaklığı q0 ve bir soğutma parametresi δ gerektirir.

TB algoritmasının başlatılması, hem yineleme sayımının hem de sıcaklık kontrol sayısının sıfıra ayarlanmasını, sıcaklığın başlangıç sıcaklığına ayarlanmasını ve başlangıç çözümünün yerleşik x^ olarak atanmasını gerektirir. Komşuluk, herhangi bir permütasyon yoluyla kurulur.

**Girdi**: İlk uygulanabilir çözüm $x^0$; Yineleme sınırı $t^{max}$
**Girdi**: İlk sıcaklık $q^0 > 0$; Sıcaklık sınırı $q^{max}$; Soğutma faktörü $0 < \delta \leq 1$

$t \leftarrow 0$
$q^{sayma} \leftarrow 0$
$q \leftarrow q^0$
$x^{\wedge} \leftarrow x^t$
Uygulanabilir bir hareket seti oluştur $M^{xt}$
**while** $\Delta x \in M^{xt}$ **and** $t < t^{max}$ **do**
    $x' \leftarrow x^t + \Delta x$
    **if** $q^{sayma} = q^{max}$ **then**
        $q \leftarrow \delta q$
        $q^{sayma} \leftarrow 0$
    **else**
        $q^{sayma} \leftarrow q^{sayma} + 1$
    **endif**
    **if either** $c(x') < c(x^{\wedge})$ **or** Olasılık $\left(e^{\frac{c(x^{\wedge})-c(x')}{q}}\right)$ **then**
        $x^{t+1} \leftarrow x'$
        **if** $c(x^{t+1}) < c(x^{\wedge})$ **then**
            $x^{\wedge} \leftarrow x^{t+1}$
        **endif**
        **else**
            $x^{t+1} \leftarrow x^t$
        **endif**
    $t \leftarrow t + 1$
    Uygulanabilir bir hareket seti oluştur $M^{xt}$
**endw**

Şekil 3.4. Tavlama benzetimi algoritması.

Yineleme sayım sınırı $t^{max}$'a ulaşılırsa veya mevcut çözüm $x^t$ için komşu hareket kümesi $M^{xt}$'de $\Delta x$ daha fazla uygulanabilir hareket yoksa algoritma sona erer. Aksi takdirde hareket kabul için test edilir. Hareket, amaç fonksiyon değerini iyileştiriyorsa 1 olasılıkla kabul edilir. Hareket kötüleşiyorsa yine de olasılıkla kabul edilecektir.

$$p = e^{\left(\frac{c(\wedge x)-c(x\prime)}{q}\right)} \tag{3.13}$$

Bir katının fiziksel tavlaması ile TB arasındaki analojiye dönersek, TB algoritması için kabul kriteri Metropolis kriterinden çıkarılır. Metropolis ve ark. tarafından sunulan Metropolis algoritması [75] bir katının fiziksel tavlanmasını simüle etmek için basit bir algoritmadır. $E_i$ enerjili katının mevcut hâli i verildiğinde, $E_j$ enerjisine sahip müteakip bir j durumunun, katının atomlarının küçük bir yer değiştirmesi yoluyla üretildiğini belirtir. Ortaya çıkan enerji farkı, $E_j$ - $E_i$, sıfırdan küçük veya sıfıra eşitse, j yeni mevcut durum olarak kabul edilir. Bununla birlikte, enerji farkının sıfırdan büyük olması gerekiyorsa, j durumu yalnızca olasılıkla kabul edilecektir.

$$p = e^{\left(\frac{E_i - E_j}{kB^T}\right)} \tag{3.14}$$

T, katının mevcut mutlak sıcaklığıdır ve $^kB$, fiziksel Boltzmann sabiti olarak bilinir. Kirkpatrick ve ark. [72], sıcaklığın yalnızca bir kontrol parametresi olduğu için Boltzmanns sabitinin ihmal edilebileceğini belirtirler. Kontrol parametresi, algoritmanın ilk aşamalarında neredeyse tüm bozulan hareketlere izin verecek şekilde formüle edilmiştir. Kontrol parametresi kademeli olarak azaldığından, kötüleşen hareketleri kabul etme olasılığı da azalır ve algoritma global optimuma yakınsar.

Robuste ve ark. [76] TB uygulamasında, bir kişinin çözümün kalitesi açısından büyük problemler için algoritmadan daha iyi performans gösterebileceğini belirtmektedir. TB'nın gelişimi, Van Breedam'ın [77] TB'nın varyantlarını gözden geçirmesi ve karşılaştırmasıyla devam etmiştir. Tan ve ark. [73] TB meta-sezgisinin bir dizi avantajı olduğunu ifade eder:

- Keyfî sistemler ve maliyet fonksiyonları ile ilgilenir.

- İstatistiksel olarak en uygun çözümü garanti eder (yeterli işlem süresi sağlandığında).

- Karmaşık problemler için bile nispeten kolay kodlama.

- Genellikle makul işlem süresi içinde iyi bir çözüm sunar.

Van Breedam [78] tarafından TA ve TB arasındaki çözüm kalitesindeki farkın değerlendirmesinde hiçbir zaman %4'ü aşmadığını belirtmiştir. Üç meta-sezgiselin karşılaştırmalı analizinde, Tan ve ark. [73], TB'nın hesaplama çabası ve çözüm kalitesi arasında iyi bir uzlaşma olduğu sonucuna varmıştır.

### 3.2.3. Genetik algoritma

GA'lar 1975 yılında John Holland tarafından geliştirilmiş ve yayınlanmıştır. GA'lar, biyolojik evrimi taklit ederek rastgele arama yöntemlerini akıllıca kullanarak küresel optimal çözümleri arayan algoritmalardır [44]. Genetik evrim ve optimizasyon arasındaki ilişkiler şunlardır:

- Popülasyonlar, her biri uygun bir çözümü temsil eden gruplar tarafından temsil edilir.

- Bir popülasyonda, ebeveynler doğal seleksiyona göre çiftleşir. Bu, rastgele seçilen uygulanabilir ana çözümlere benzer.

- Yavrular, seçilen ebeveynlerin çiftleşmesi ile üretilir ve yeni oluşturulan çözümleri temsil eder.

- Doğada, kromozomlar yeni kromozom dizileri oluşturmak için değiştirildiğinden, yavrular her ebeveynin bazı özelliklerini sergiler. Algoritma, ana çözümlerin parçaları üzerinde takas gibi tedirginlikler kullanarak iki yeni yavru çözüm oluşturarak benzetmeden yararlanır. GA'larda tedirginlikler genellikle geçitler olarak adlandırılır.

- En uygun olanın hayatta kalması, bir çözümün uygunluğu, onun nesnel işlev değeriyle ilişkili olabileceği için dâhil edilir. En uygun çözümler, sonraki

nesilde en uygun çözümün hayatta kalmasını sağlamak için tipik olarak yeniden üretilecektir.

- Çeşitlilik için mutasyon, meta-sezgisel kromozomların rastgele modifikasyonu, yani muhtemel çözümler ile temsil edilir.

Goldberg [79], arama stratejilerindeki GA uygulamalarını ve optimizasyonu incelemiştir. Genel GA meta-sezgisi, nesil 0'ı oluşturmak için gereken p benzersiz uygulanabilir ilk çözümü temsil eder. Filipec ve ark. [80] GA'larını çeşitli popülasyon büyüklükleriyle test ettiler ve çok küçük bir popülasyonun çeşitlendirme tehlikeye girdiği için algoritmayı erken sonlandırabileceği sonucuna varırken, çok büyük popülasyonlar daha fazla nesile ihtiyaç duyulduğu için (artan hesaplama çabası) yakınsama oranını yavaşlatarak kaliteli çözümler elde edilir. Başlangıç çözümleri rastgele veya sezgisel rota oluşturma yöntemi kullanılarak oluşturulur [81]. Skrlec ve ark. [82] ve Tan ve ark. [73], yakınsama oranını iyileştirmek için yalnızca sezgisel yöntemlerin kullanılmasını önermektedir.

Algoritma, yalnızca yeterli sayıda nesil mevcut olduğunda sona erer. T neslinin en iyi çözümleri tam olarak nesil t+1'e klonlandığı için en uygun olanın hayatta kalması sağlanır. Bir dizi yeni göçmen çözümü $p_i$, üretilir ve t+1 nesline dâhil edilir. T+1 oluşumunun dengesi, t neslinden rastgele seçilen $p_c/2$ çözümleri üzerinde çeşitli geçiş permütasyonları gerçekleştirilerek oluşturulur. GA'larla kısıtlı ARP'leri çözmek için literatürde iki farklı yaklaşım bulunmuştur:

Birinci küme, ikinci rota (Cluster first, route second): Thangiah ve ark. [83], ZPARP'yi çözmek için kullanılan bir GA programı olan GIDEON'u geliştirdi. O zamanlar, Solomon [84] tarafından sunulan 56 kıyaslama probleminden 41'i için en iyi bilinen çözümleri ürettiği için ZPARP için mevcut en iyi algoritmaydı.

Birinci rota, ikinci küme (Route first, cluster second): Bir kromozomdaki ayrı yolları belirtmek için fazladan bölümleme karakterlerinin kromozoma eklenmesi gerekir. Bu ekstra karakterler GA'yı işe yaramaz hale getirebilir. GA'lar, ARP varyasyonlarını

çözmek için iki aşama kullanır: Her kromozom, tüm müşteriler arasında belirli bir yolu temsil eder. İlk yönlendirme aşamasında GA, tüm müşteriler için bir GSP çözerek uzun kromozom dizisini iyileştirir. İkinci kümeleme aşaması, uzun rotanın dışındaki her araç için bir rota oluşturur. Bu, müşterileri bir araca yalnızca zaman aralıkları ihlal edilmediği takdirde, araç dolana kadar ekleyen başka bir algoritma tarafından yapılır. Tek yol kromozomundaki aşağıdaki müşteri daha sonra bir sonraki araca atanır.

Tan ve ark. [73], ARP varyantları için üç popüler meta-sezgisel, TA, TB ve GA'yı karşılaştıran ilk kişi olmuştur. GA'ların ZPARP'yi çözmede başarılı olduğu sonucuna vardılar, ancak tüm rotalama problemlerini çözecek kadar tek bir genel meta-sezgisel yöntem olmadığı sonucuna varmışlardır.

### 3.2.4. Karınca kolonisi optimizasyonu

KKO algoritmaları, kombinatoryal optimizasyon problemlerine çözüm bulmak için yinelemeli, olasılıksal meta-sezgisel olarak sınıflandırılır. KKO, Dorigo ve Stützle [85] tarafından önerilen ve tüm karınca algoritmalarını içeren genel bir terimdir. Karınca algoritması, birkaç nesil yapay karıncanın iyi çözümler aradığı evrimsel bir yaklaşımdır. Bir neslin her karınca, çeşitli kararlardan geçerek adım adım bir çözüm geliştirir. İyi çözümler bulan karıncalar, yolun kenarlarına feromon yerleştirerek karar alanında yollarını işaretlerler. Gelecek neslin karıncaları feromona çekilir ve iyi çözümlerin yakınında çözüm alanını arama olasılıkları daha yüksektir [86].

Karınca algoritmaları, yuvalarından besin kaynaklarına en kısa yolu bulmaya çalışan gerçek karıncaların kullandığı yiyecek arama mekanizmasından ilham alır. Toplayıcı bir karınca, izine bir miktar feromon dağıtarak yolunu işaretler, böylece diğer yiyecek arayan karıncaları da aynı yolu izlemeye teşvik eder, ancak zorlamaz [87]. Feromon, aynı türdeki diğer organizmalarda reaksiyonu tetiklemek için bir organizma tarafından salgılanan herhangi bir endojen kimyasal maddenin jenerik adıdır. İletişim yoluyla karıncaların davranışında bir değişikliğe neden olmak için ortamı değiştirme ilkesi, damgalama olarak bilinir. Stigmergy'nin etkisi, yiyecek

arama davranışı ve yapay karınca meta-sezgiseli için temel sağlar. Dorigo ve ark. [87], karınca kolonilerinin sosyal yapısının yuva ile besin kaynakları arasındaki en kısa yolları belirleyebileceğini öne süren deneyleri yapmışlardır. Bununla birlikte resmî bir kanıt yoktur.

KKO algoritmasının merkezinde, her yinelemede her bir karıncanın daha eksiksiz bir kısmi çözüme karşılık gelen bir i durumundan başka bir j'ye hareket ettiği (bir adım gerçekleştirdiği) bir döngü vardır.

Maniezzo ve ark. [88], her başlangıç-hedef çifti (i, j) için çekicilik $\eta_{ij}$ olarak bilinen yolun önsel olarak istenirliğini gerekir. Çekicilik genellikle sezgisel bilgi olarak da adlandırılır [89]. i'den j'ye hareketin iz seviyesi $\tau_{ij}$ gereklidir ve bu belirli hareketi yapmanın geçmişte ne kadar faydalı olduğunu gösterir. İz seviyesi, bu nedenle, hareketin istenebilirliğinin bir posteriori (deney sonucu elde edilen) göstergesini temsil eder. Daha önce tartışılan meta-sezgisellikte olduğu gibi, bir iterasyon sınırı $t^{max}$ KKO'yu sonlandırır.

Her karınca k için önceden ayarlanmış parametrelerle ağırlıklandırılan hem çekicilik hem de iz seviyesi kullanılarak aşamalı olarak bir çözüm oluşturulur. Her karıncanın tabu hareketleri hafızası buna göre güncellenerek, yalnızca uygun çözümlerin yaratıldığından emin olunur. Tüm karıncalar çözümlerini bulduktan sonra, feromon iz matrisi, kaç karıncanın (çözüm) belirli kenarlardan (i, j) geçtiğini belirleyerek güncellenir. Yineleme sayısı artırılır ve işlem, maksimum yineleme sayısına ulaşılana kadar yinelenir.

Bullnheimer ve ark. [53], karşılaştırmalı değerlendirme problemleri için bulunan en iyi çözümleri geliştiremedi. ARP varyantlarına yaklaşımın yerleşik ve iyi araştırılmış meta-sezgisellere kıyasla olgunlaşmamış olduğu göz önüne alındığında, KKO'nun rekabet gücü takdire değerdir.

### 3.3. Kümeleme Algoritmaları

Günümüzde, biriken büyük miktardaki yüksek boyutlu veriyi anlamak önemlidir. Küme analizi, bu verileri analiz etmek için anahtar bir teknoloji hâline geldi. Son birkaç on yılda birçok kümeleme algoritması geliştirilmiştir. Jain ve ark. [90] ve Everitt ve ark. [91], mevcut bireysel kümeleme algoritmalarını kabaca iki kategoriye ayırdı: hiyerarşik kümeleme ve bölümlü kümeleme.

Kümeleme algoritmalarının diğer bir yaygın taksonomisi, Berkhin [92], Xu ve Wunsch [93] tarafından tanıtıldı. Bu sınıflandırma, kümeleme algoritmalarını beş kategoriye ayırır. İlk iki kategori hâlâ hiyerarşik kümeleme ve bölümlü kümelemedir. Diğer üçü yoğunluk tabanlı kümeleme, ızgara tabanlı kümeleme ve model tabanlı kümelemedir.

Yoğunluğa dayalı kümeleme, bir veri uzayındaki nesnelerin bağlantısına ve yoğunluğuna dayalı olarak nesneleri kümelemektir [92, 93]. Bu tür kümeleme yöntemleri, bir bölgedeki nesnelerin bir küme hâlinde oluşturulup oluşturulamayacağını belirlemek için bir yoğunluk eşiği kullanır. Bir dizi nesnenin yoğunluğu eşiği aşarsa nesneler bir kümeye atanacaktır; aksi takdirde, aynı kümeye atanmazlar. Yoğunluğa dayalı kümeleme, keyfî şekilli kümeleri algılama ve yüksek boyutlu verilerle başa çıkma becerisine sahiptir [92, 93]. Ek olarak, yoğunluk temelli kümeleme, verilerdeki gürültü ve aykırı değerlerle başa çıkmak için yeterince sağlamdır.

Izgara tabanlı kümeleme süreci, çok seviyeli bir taneciklik yapısına dayanmaktadır [92, 93]. Izgara tabanlı kümeleme, veri alanını eşit şekilde birkaç ızgaraya böler. Kümeleme, her ızgarada gerçekleştirilir. Zaman karmaşıklığı yalnızca ızgaraların sayısıyla ilgili olduğundan, ızgara tabanlı kümeleme, büyük veri kümeleriyle uğraşmak için iyi bir hıza sahiptir [92, 93]. STING (istatistiksel bir bilgi ızgarası yaklaşımı) [94] ve WaveCluster [95] gibi bazı tipik ızgara tabanlı kümeleme yöntemleri vardır.

Model tabanlı kümeleme yöntemleri, verilen verilerin kümelerinin bir dizi olasılık dağılımıyla belirlenebileceği varsayımına dayanarak kümeleme gerçekleştirir [92, 93]. Model tabanlı kümeleme, her bir küme için bir modeli (bir yoğunluk dağılımı işlevi gibi) varsayar ve daha sonra bu modelleri karşılık gelen kümeler için uygun nesneleri bulmak için kullanır [96, 97]. İstatistiksel model tabanlı kümeleme ve sinir ağı modeline dayalı kümeleme, en yaygın araştırma konularıdır. Model tabanlı kümeleme, keyfî biçimli kümeleri tespit etme ve kümelenmenin daha iyi kararlılıklarını algılama yeteneğine sahiptir [93], ancak büyük veri kümeleriyle başa çıkamaz ve ilklendirmeye duyarlıdır [92, 93]. Tipik model tabanlı kümeleme yöntemleri, Kendi Kendini Düzenleyen Harita (KKDH) [96], Autoclass [98] ve Cobweb'dir [97].

Son birkaç yılda kümeleme yöntemlerinin yeni bir taksonomisi ortaya çıktı. Kümeleme sonuçlarının esnekliğine göre, bu sınıflandırma, bireysel kümeleme yöntemlerini iki türe sınıflandırır: sert kümeleme yöntemleri ve bulanık kümeleme yöntemleri [92, 93]. Sert kümeleme, her nesnenin kalıcı olarak bir ve yalnızca bir kümeye atandığı anlamına gelir. K-ortalamalar, Hiyerarşik Kümeleme (HK), Hiyerarşileri Kullanarak Dengeli Yinelemeli Azaltma ve Kümeleme (HKDYAK), Kendi Kendini Düzenleyen Harita (KKDH) ve Medoidlerin Etrafında Bölümleme (MEB) gibi kümeleme algoritmalarının tümü, sert kümelemeye aittir. Bunun tersine, bulanık kümeleme, her nesneyi bir üyelik işlevi ile birden fazla kümeye atar [93]. Burada üyelikler olasılıklar biçiminde sunulur. Tipik bir bulanık kümeleme yöntemi, Bulanık C-Ortalamalar (BCO) algoritmasıdır [99, 100].

### 3.3.1. K-ortalamalar kümelemesi

K-ortalamalar Kümeleme algoritması [101], iyi bilinen bir kümeleme algoritmasıdır. K-ortalamalı kümeleme algoritmasının temel adımları şu şekilde özetlenmiştir [90, 102]:

- K kümelerinin merkezleri olarak rastgele k nesneler seçin.

- Geri kalan nesneleri en yakın küme merkezlerine atayın.

- K kümelerinin her biri için küme merkezini (kümedeki tüm nesnelerin ortalaması olarak tanımlanan) yeniden hesaplayın.

- Nesneleri yeni küme merkezlerine göre k kümelere yeniden tahsis edin.

- Her kümede daha fazla değişiklik olmayana veya sonlandırıcı koşullara ulaşılana kadar 3 ve 4. adımları tekrarlayın.

K-ortalamalı kümeleme algoritmasının temel avantajları aşağıdaki gibidir [90, 92, 93, 102]:

- Büyük veri kümelerini analiz etmek için iyidir.

- İyi (düşük) bir zaman karmaşıklığına sahiptir.

- Uygulaması kolaydır.

K-ortalamalı kümeleme algoritmasının temel sınırlamaları ve sorunları aşağıdaki gibidir [90, 92, 93, 102]:

- Nesneler arasındaki mesafeleri tanımlamak için farklı mesafe ölçütleri kullanmak, k-ortalamalarının sonuçlarını saptırabilir.

- K-ortalamalarının her çalışması farklı bir kümeleme sonucu oluşturabilir.

- Küme boyutlarının önemli ölçüde değiştiği veriler için iyi ölçeklenmez.

- İlklendirmeye duyarlıdır. Bu nedenle, uygun olmayan bir başlatma, k-ortalamalarının sonuçlarını saptırabilir.

- Verilerdeki gürültüye ve aykırı değerlere duyarlıdır.

- Dışbükey olmayan şekilli kümelerle baş edemez.

- Giriş verilerinin sırasına duyarlıdır.

- Kategorik verilerle baş edemez.

### 3.3.2. Bulanık c-ortalamalar kümelemesi

BCO algoritması, en yaygın olarak kullanılan bulanık kümeleme algoritmasıdır [90, 100, 102]. İlk geliştirme Dunn [99] tarafından yapılmış ve Bezdek [100] tarafından ikinci kez geliştirilmiştir. Sert kümeleme ile bulanık kümeleme arasındaki en önemli fark, bulanık kümelemenin her nesneyi kalıcı olarak bir kümeye atamamasıdır. Bulanık kümeleme, nesnelerin kümelere nasıl ait olduğunu ifade etmek için bir "derece" kavramı kullanır. Her nesne için, derecelerin toplamı (bulanık katsayılar olarak da adlandırılır) 1'dir [99, 100].

BCO'nin amaç işlevi Üyelik Fonsiyonu olarak adlandırılmıştır [99, 100]. BCO'nin "iyi" bir kümeleme çözümünün hedef işlevi en aza indirmesi beklenir. Üyelik Fonksiyonu (3.15), (3.16) ve (3.17) denklemleri olarak tanımlanır [99, 100]:

$$F_m = \sum_{i=1}^{n}\sum_{j=1}^{k} u_{ij}^{m}\left\|x_i - C_j\right\|^2 \qquad (1 \le m < \infty) \tag{3.15}$$

$$u_{ij} = \frac{1}{\sum_{q=1}^{k}\left(\frac{\left\|x_i - C_j\right\|}{\left\|x_i - C_q\right\|}\right)^{\frac{2}{m-1}}} \tag{3.16}$$

$$Cj = \frac{\sum_{i=1}^{n} u_{ij}^{m} x_i}{\sum_{i=1}^{n} u_{ij}^{m}} \tag{3.17}$$

Burada n, belirli bir veri kümesindeki nesnelerin sayısıdır. k, küme sayısıdır, m, 1'den büyük veya 1'e eşit herhangi bir gerçek sayı olabilir. $u_{ij}$, j kümesine ait olan $x_i$

nesnesinin derecesini gösteren bulanık katsayıdır. $C_j$, j kümesinin merkezidir. $||x_i-C_j||$, $x_i$ nesnesi ile $C_j$ küme merkezi arasındaki mesafeyi belirtir.

BCO'nin temel adımları şu şekilde özetlenmiştir [99, 100]:

- Küme sayısının tahminî sayısına göre her nesneye bulanık katsayılar atayın.
- Bulanık katsayılara göre küme merkezlerini hesaplayın.
- Yeni küme merkezlerine dayalı olarak bulanık katsayıları yeniden hesaplayın.
- Bulanık katsayıların varyansını değerlendirin (önceki bulanık katsayılardan biriyle karşılaştırarak).
- Varyansı önceden tanımlanmış bir duyarlılık eşiğiyle karşılaştırın.
- Bulanık katsayıların varyansı duyarlılık eşiğinden az olana kadar 2, 3, 4 ve 5. adımları tekrarlayın.

BCO'nin ana avantajları aşağıdaki gibidir [90, 102]:

- BCO, belirsiz verilerle başa çıkmak için sağlam ve esnektir.

BCO'nin temel sınırlamaları ve sorunları aşağıdaki gibidir [90, 102]:

- BCO, kenar ve kabuk gibi şekillerle verileri analiz edemez.
- Farklı mesafe metrikleri kullanmak, BCO sonuçlarını saptırabilir.
- Verilerdeki gürültü ve aykırı değerlerle başa çıkamaz.
- BCO, küme sayısı hakkında önceden bilgi gerektirir.

- BCO'nin sonuçları yerel minimumlar içinde sıkışmış olabilir.

### 3.4. Önerilen Çözüm Yöntemi

Bu çalışmada, Tavlama Benzetimi yönteminin başlangıç çözümü K-ortalamalar ve Bulanık c-ortalamalar ile iyileştirilmiştir. Veriler aracılığı ile örtüşen kümeler bulmakta kullanılan bu metot, esnek bir yapıya sahip olması ve diğer kümeleme algoritmalarına göre daha iyi sınıflandırma yeteneğine sahip olduğundan tercih edilmiştir. Bu metodun çözülmesinde MATLAB R2017A Programının Fuzzy seti kullanılmıştır.

BCO yönteminin uygulanması için gerekli olan şehirlerin enlem ve boylam koordinatları Google Haritalar üzerinden belirlenerek Ek:1'deki tabloda listelenmiştir. Ayrıca tabloda yer alan diğer veriler:

- Sıra: Şehirlerin indis numaralarıdır. 0 depoyu ve diğer sayılar şehirleri temsil eder.

- Kod: Şehirlerin plaka kodları.

- Şehir Adı: Şehirlerin isimleri.

- Enlem: BCO algoritmasının y değeri.

- Boylam: BCO algoritmasının x değeri.

- Talep: Müşterilerin talepleri.

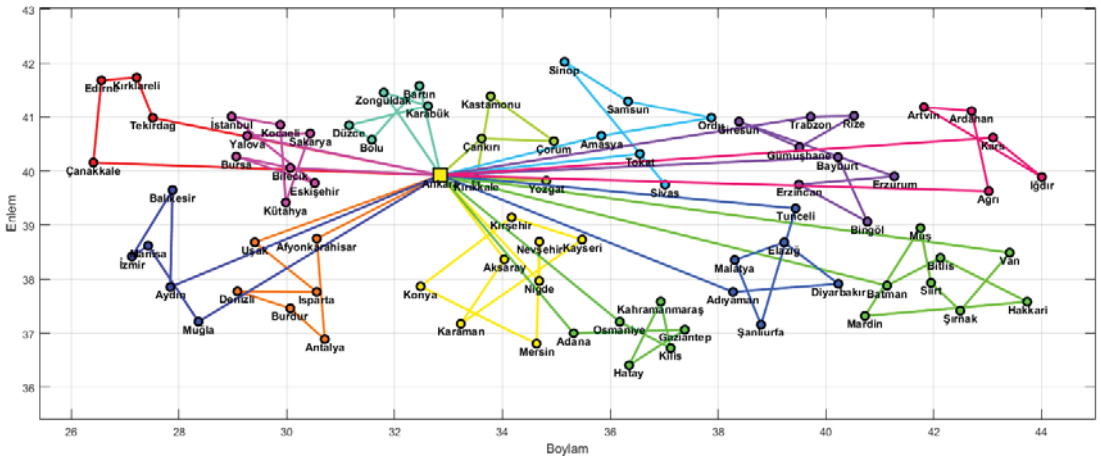
TB algoritmasının çalışabilmesi ve başlangıç çözümünü optimum çözüme ulaştırılabilmesi için gereken program, modüller hâlinde el ile yazılmıştır.

Model Oluşturma: Bu bölüme ait kodlar Ek:2'de gösterilmiştir. Buradaki amaç Ek:1'de belirtilen tablodaki tüm verileri okuyarak programın anlayabileceği şekilde "model" adı verilen bir veri seti oluşturmaktır. Bu bölümde ayrıca tüm şehirler arası uzaklık matrisi ve tüm şehirlerin depo ile olan uzaklıkları kilometre cinsinden hesaplanmıştır. "model" veri setine kaydedilen veriler şu şekildedir: nokta (şehir) sayısı, küme (rota) sayısı, müşterilerin talepleri, araç kapasiteleri, maksimum x değeri, minimum x değeri, maksimum y değeri, minimum y değeri, tüm noktaların x değerleri, tüm noktaların y değerleri, deponun x değeri, deponun y değeri, uzaklık matrisi ve tüm şehirlerin depoya olan uzaklıkları.

Model Kaydetme: Bu bölüme ait kodlar Ek:3'te gösterilmiştir. Model Oluşturma modülü tek başına çalışamaz. Model Kaydetme modülüne bağlıdır ve bu modülle beraber çalışır. İşlem yapılacak veri setinin kolay seçilebilmesi için dosya ismine nokta sayısı ve küme sayısı bilgileri girilerek programın okuyabileceği şekilde (.m uzantısında) veri setini kaydeder.

Veri Seti Seçme: Bu bölüme ait kodlar Ek:4'te gösterilmiştir. İşleme alınacak veri setini seçmek için bir seçim penceresi oluşturur. İstenen veri seti seçilir ve içindeki veriler programa aktarılır.

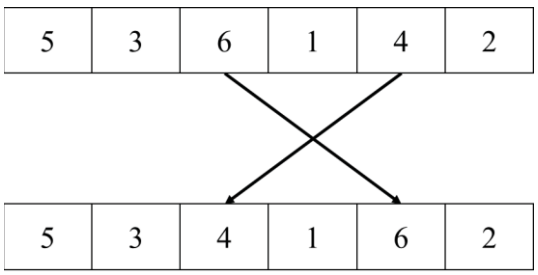
Başlangıç Çözümü Oluşturma: Bu bölüme ait kodlar Ek:6'da gösterilmiştir. Burada, başlangıç çözümünün oluşturulabilmesi için veri setinin içinden BCO algoritmasının kullanabileceği x ve y verilere çekilir. Araçların kapasite kısıtları dikkate alınmaksızın toplam mesafeyi minimum yapacak şekilde belirlenen küme sayısına göre kümeleme yapılır. Her bir kümeye ait olan noktalar rota oluşturacak şekilde renklendirilir ve noktalar birbirine bağlanarak rota oluşturulur. Oluşturulan her rota tek tek farklı renklerde birleştirilerek ve şehir isimleri de eklenerek başlangıç çözümü ekrana çizdirilir. Oluşturulan toplam rota bir dizi oluşturacak şekilde bir değişkene atanır.




Şekil 3.5. BCO ile örnek bir başlangıç çözümü.

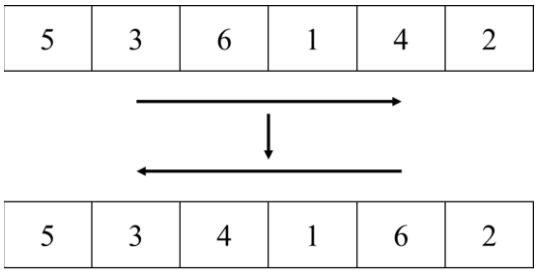
Komşu Oluşturma: Bu bölüme ait kodlar Ek:9'da gösterilmiştir. Her iterasyonda toplam rotayı oluşturan dizide rassal olarak seçilen 3 farklı komşuluk oluşturma yöntemi bulunmaktadır.

Takas, iki rastgele indis numarası seçilir ve bu indislerin elemanları birbirleriyle değiştirilir. Komşu çözümlerin üretilmesinin en kolay ve en yaygın kullanılan yoludur. Anlamaya yardımcı olmak için Şekil 3.6.'da Takas yöntemini tanıtıyoruz. Örneğin, üçüncü ve beşinci eleman seçilirse, konumları birbirleriyle değiştirilecektir.



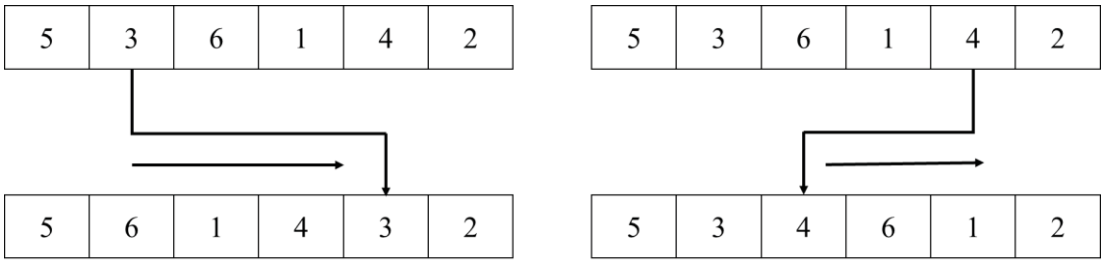

Şekil 3.6. Takas yöntemi.

Ters çevirme, iki rastgele indis numarası seçilir ve rastgele seçilen iki indis arasındaki yön tersine çevrilir. Anlamaya yardımcı olmak için Şekil 3.7.'de Ters çevirme yöntemini tanıtıyoruz. Örneğin, ikinci ve beşinci eleman seçilirse, konumları arasındaki yön tersine çevrilecektir.




Şekil 3.7. Ters çevirme yöntemi.

Araya sokma, iki rastgele indis numarası seçilir ve seçilen indislerin elemanlarının değerine dikkat ederek, rastgele seçilen iki öge arasındaki yön tersine çevrilir. Sayı 1, sayı 2'den küçük ise sayı 1'i, sayı 1'den sayı 2'ye kadar olan sıralamanın sonuna taşı (sayı 2'den hemen sonra). Diğer durumda, sayı 1'i, sayı 1'den sayı 2'ye kadar olan sıralamanın başına taşı (sayı 2'den hemen sonra). Anlamaya yardımcı olmak için Şekil 3.8.'de Araya sokma yöntemini tanıtıyoruz. Örneğin, ikinci ve beşinci eleman seçilirse konumları arasındaki yön aşağıdaki gibi olacaktır:

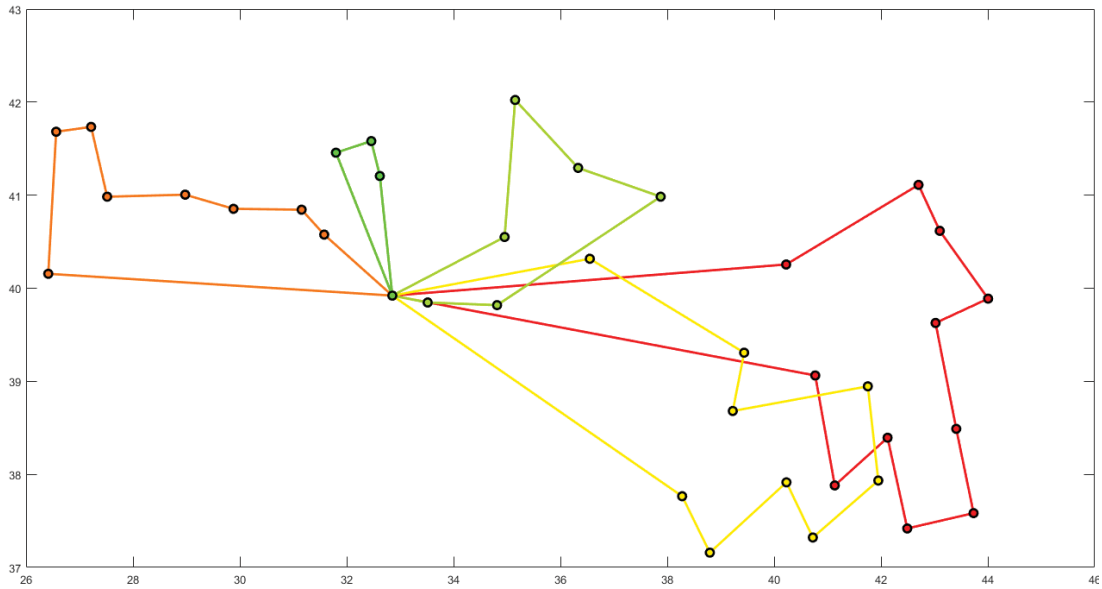



Şekil 3.8. Araya sokma yöntemi.

Sayısal Çözüm: Bu bölüme ait kodlar Ek:7'de gösterilmiştir. Sayısal hesaplamaların tamamı bu bölümde gerçekleşir. Yeni komşuluklar oluşturulduktan sonra yeni oluşan rotanın sayısal hesaplamaları burada yapılır ve kaydedilir. Kaydedilen veriler şu şekildedir: Yeni oluşan rotalar, bu rotaların mesafeleri, maksimum rota mesafesi, minimum rota mesafesi, toplam mesafe, kullanılan kapasite, fazla kapasite, ortalama fazla kapasite, toplam fazla kapasite ve çözümün uygulanabilir olup olmadığıdır. Ayrıca amaç fonksiyonu kullandığı verileri buradan alır ve amaç fonksiyonunun değeri hesaplanır.

Görsel Çözüm: Bu bölüme ait kodlar Ek:10'da gösterilmiştir. Burada amaç, yeni oluşan rotayı sayısal çözüm modülünde hesaplandıktan sonra burada minimum-

maksimum x ve minimum-maksimum y değerleri sınırlarında çizdirerek ekrana yansıtmaktır. Her bir kümenin rotası farklı renkte olacak şekilde çizilir ve noktalar birbirine bağlanarak toplam rota oluşturulur.

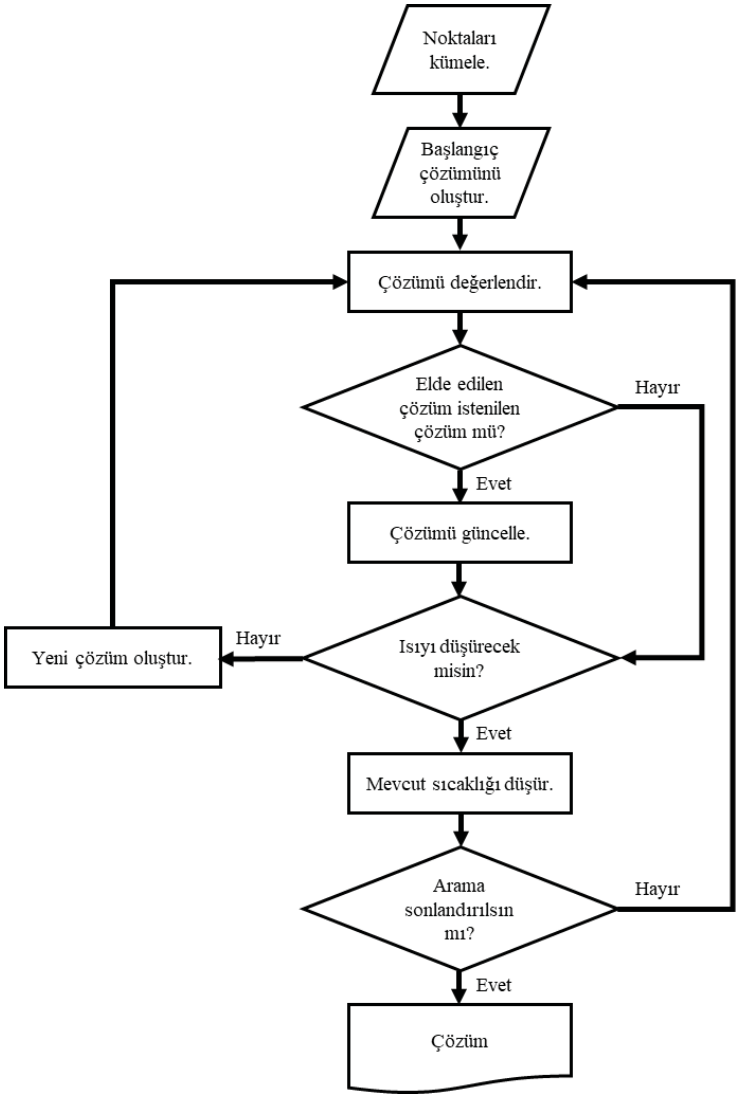



Şekil 3.9. Örnek bir görsel çözüm.

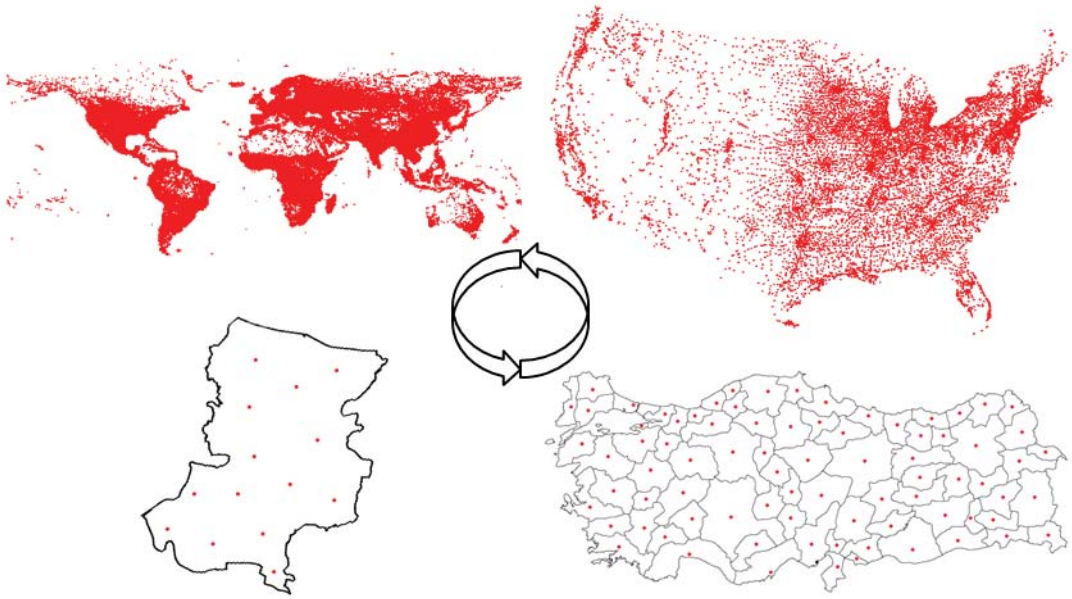
Tavlama Benzetimi: Bu bölüme ait kodlar Ek:11'de gösterilmiştir. Model Oluşturma ve Kaydetme modülleri hariç diğer modüller bu modüle bağlıdır. Burada kullanılan parametreler: üst iterasyon sayısı, alt iterasyon sayısı, ilk sıcaklık değeri, sıcaklık düşürme oranıdır. Belirtilen iterasyon sayısına gelene kadar her bir iterasyonda sıcaklık değeri düşürülür ve bağlı tüm modüller çalıştırılır. Ayrıca çalışma esnasında komut ekranında bize çeşitli bilgiler verilir. Yeni oluşturulan bir rotanın amaç fonksiyonunun değeri bir önceki çözümün amaç fonksiyonunun değerinden iyi ise yeni rota koşulsuz olarak kabul edilir. Eğer tam tersi bir durum varsa Boltzman Dağılımı'na göre bir değer hesaplanır ve bu değer 0-1 arasında oluşturulan rastgele bir sayıdan büyük veya eşit ise amaç fonksiyonunun değeri daha kötü olsa bile çözüm kabul edilir. Bu sayede algoritmanın arama uzayında çıkmaza girmesi engellenmiş olur.

Çalıştır: Bu bölüme ait kodlar Ek:12'de gösterilmiştir. Tavlama Benzetimi modülünü çalıştırır. tic-tok komutu kullanılarak algoritma başlangıcında zaman tutulmaya

başlanır ve son iterasyon da tamamlandıktan sonra süre durdurulur. Tüm algoritmanın çalışma zamanı tespit edilir ve bir değişkene atanır.




Şekil 3.10. Önerilen modelin akış şeması.



Şekil 3.11. Önerilen modelin ölçeklenebilirliği.

Önerilen model basit ve küçük problemlerden kapsamlı ve zor problemlere kadar çok çeşitli zorluk seviyelerinde kullanılabilir. Önerilen model ile beraber problem çözüldüğünde sonuçlar ümit vericidir. Bu model ile birlikte daha kısa sürelerde optimum çözüme daha yakın sonuçlar elde edilmiştir. Bu durum, daha karmaşık problemlerde daha iyi sonuçlar vereceğinin kanıtıdır. Bu da önerilen modelin ölçeklenebilirliğinin yüksek olduğunu gösterir.

# BÖLÜM 4. ARAŞTIRMA BULGULARI

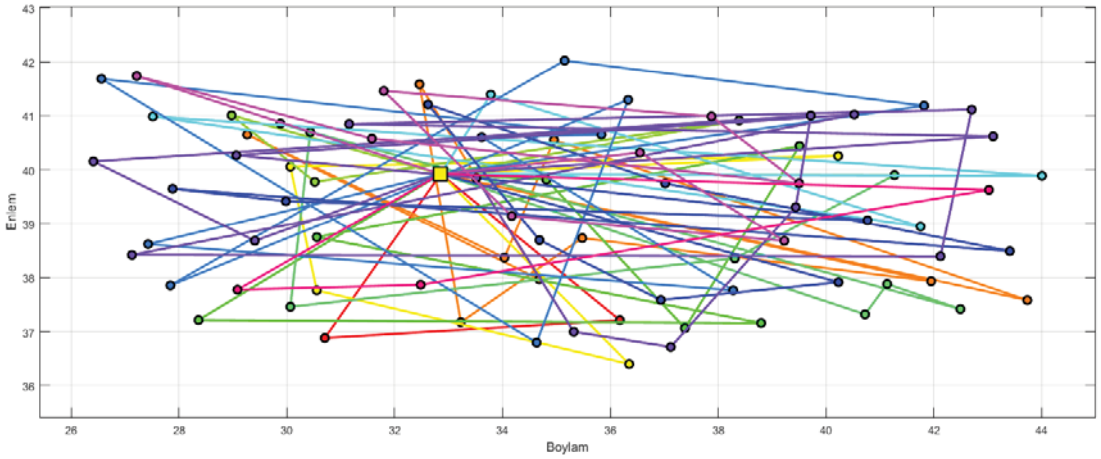
Aşağıdaki tablolarda rassal başlangıç çözümü, k-ortalamalar ile kümelenmiş başlangıç çözümü ve bulanık c-ortalamalar ile kümelenmiş başlangıç çözümü için sırasıyla 12, 13 ve 14 kümeli olacak şekilde problem çözülmüştür. Her yöntem ve küme sayısı için 15 adet deney yapılmıştır. Deney sonuçları aşağıdaki tablolarda gösterilmiştir.

Tablo 4.1. 12 kümeli rassal başlangıçlı çözüm sonuçları.

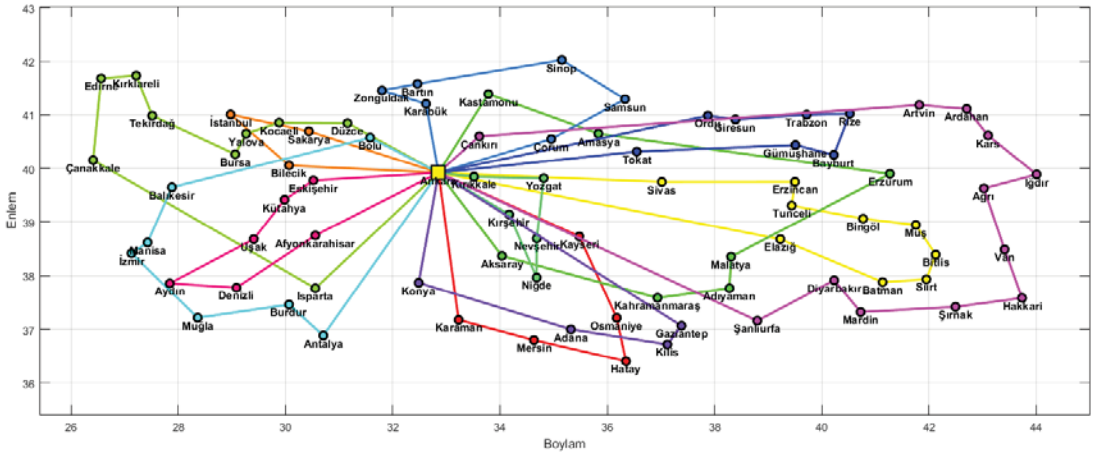
| Deney No | Başlangıç Çözümü | En İyi Çözüm | Küme Sayısı | Minimum Mesafe | Maksimum Mesafe | Toplam Mesafe | Toplam Fazla Kapasite | Uygulanabilir Çözüm | Çözüm Süresi |
|---|---|---|---|---|---|---|---|---|---|
| **1** | **52998,19** | **16172,86** | **12** | **654,7745** | **2466,0136** | **16172,86** | **0** | **1** | **33,734** |
| 2 | 47459,99 | 16751,71 | 12 | 917,3224 | 2396,0454 | 16585,85 | 1 | 0 | 33,635 |
| 3 | 46745,85 | 17546,34 | 12 | 964,4904 | 2378,118 | 17546,34 | 0 | 1 | 33,576 |
| 4 | 53429,29 | 16476 | 12 | 802,9382 | 2372,0571 | 16312,87 | 1 | 0 | 33,647 |
| 5 | 51885,52 | 16197,37 | 12 | 660,3134 | 2412,1949 | 16037 | 1 | 0 | 33,457 |
| 6 | 51682,93 | 17300,69 | 12 | 743,5212 | 2312,4084 | 17129,4 | 1 | 0 | 33,652 |
| 7 | 48234,08 | 16516,12 | 12 | 690,152 | 2450,572 | 16516,12 | 0 | 1 | 33,385 |
| 8 | 48215,6 | 16791,78 | 12 | 864,4684 | 2469,1795 | 16625,52 | 1 | 0 | 33,422 |
| 9 | 52232,35 | 16860,09 | 12 | 768,0105 | 2160,8319 | 16693,15 | 1 | 0 | 33,559 |
| 10 | 48666,38 | 16323,06 | 12 | 688,5869 | 2324,833 | 16323,06 | 0 | 1 | 33,48 |
| 11 | 48382,72 | 17274,75 | 12 | 696,2411 | 2465,6799 | 17274,75 | 0 | 1 | 33,513 |
| 12 | 45842,16 | 16953,36 | 12 | 684,9929 | 2273,6208 | 16785,5 | 1 | 0 | 33,722 |
| 13 | 50908,31 | 16488,77 | 12 | 579,4687 | 2197,1699 | 16165,46 | 2 | 0 | 33,836 |
| 14 | 51821,18 | 17682,1 | 12 | 985,4014 | 2275,3059 | 17682,1 | 0 | 1 | 33,592 |
| 15 | 50834,62 | 17036,83 | 12 | 726,2422 | 2141,6565 | 16868,15 | 1 | 0 | 33,597 |
| Ort. | 49955,94 | 16824,79 | 12 | 761,7949 | 2339,7125 | 16714,54 | 0,66667 | 0,4 | 33,587 |

Tüm araçların kapasitesinin eşit olduğu için toplam müşteri talebini bir aracın toplam kapasitesine böldüğümüzde çıkan sonuç 12'dir. Bu problemin çözülebilmesi için minimum 12 rotaya ihtiyaç vardır. Problem, 12 kümeli rassal başlangıç ile çözüldüğünde en iyi sonucu 1 numaralı deney vermiştir. Ortalama başlangıç çözümü 49955,94 km, ortalama en iyi çözüm 16824,79 km, ortalama minimum mesafe

761,79 km, ortalama maksimum mesafe 2339,71 km, ortalama toplam mesafe 16714,15 km, ortalama çözüm süresi 33,59 saniye olarak bulunmuştur. 15 çözümden 6 tanesi uygulanabilir çözümdür. Şekil 4.1. başlangıç çözümünü; Şekil 4.2. nihai çözümü göstermektedir.




Şekil 4.1. 12 kümeli rassal başlangıçlı en iyi çözümün başlangıç çözümü.

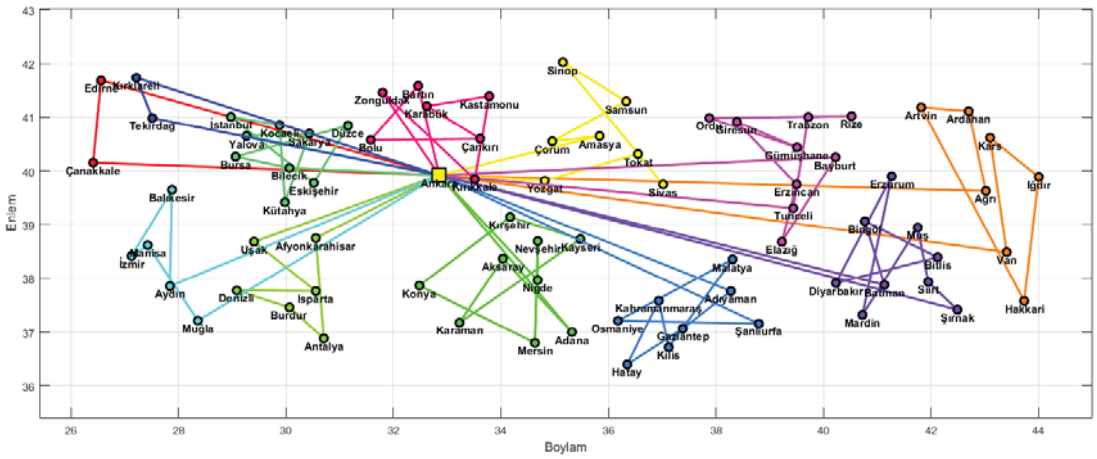



Şekil 4.2. 12 kümeli rassal başlangıçlı en iyi çözümün nihai çözümü.

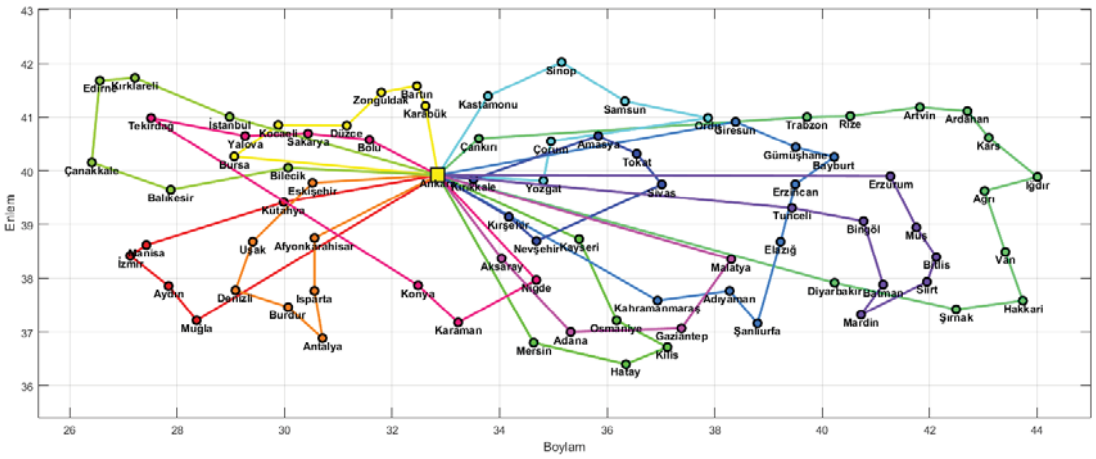
Tablo 4.2. 12 kümeli k-ortalamalar ile kümelenmiş başlangıçlı çözüm sonuçları.

| Deney No | Başlangıç Çözümü | En İyi Çözüm | Küme Sayısı | Minimum Mesafe | Maksimum Mesafe | Toplam Mesafe | Toplam Fazla Kapasite | Uygulanabilir Çözüm | Çözüm Süresi |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 21633,17 | 16772,07 | 12 | 740,9403 | 2375,2377 | 16606,01 | 1 | 0 | 34,442 |
| 2 | 20219,73 | 15860,55 | 12 | 589,6186 | 2418,5602 | 15703,52 | 1 | 0 | 34,183 |
| **3** | **22205,92** | **16336,63** | **12** | **858,6412** | **2360,802** | **16336,63** | **0** | **1** | **34,093** |
| 4 | 21434,53 | 15977,35 | 12 | 677,5239 | 2126,6589 | 15819,15 | 1 | 0 | 33,99 |
| 5 | 21456,43 | 16454,51 | 12 | 877,0236 | 2149,1044 | 16291,6 | 1 | 0 | 33,924 |
| 6 | 23330,91 | 16428,91 | 12 | 698,7528 | 2508,551 | 16266,24 | 1 | 0 | 34,111 |
| 7 | 21337,88 | 16591,98 | 12 | 783,6667 | 2648,542 | 15801,89 | 5 | 0 | 33,967 |
| 8 | 22306,11 | 15871,81 | 12 | 660,3134 | 2193,6065 | 15714,67 | 1 | 0 | 33,958 |
| 9 | 22296,78 | 16449,37 | 12 | 702,6832 | 2333,1864 | 16286,5 | 1 | 0 | 33,922 |
| 10 | 20953,83 | 15521,79 | 12 | 579,4687 | 2286,5694 | 15368,11 | 1 | 0 | 33,953 |
| 11 | 22854,22 | 16393,55 | 12 | 794,9418 | 2392,6719 | 16231,24 | 1 | 0 | 33,916 |
| 12 | 20851,98 | 16018,61 | 12 | 661,7122 | 2087,1751 | 15704,52 | 2 | 0 | 34,421 |
| 13 | 25038,33 | 17569,47 | 12 | 835,6869 | 2347,0975 | 17395,52 | 1 | 0 | 33,976 |
| 14 | 20478,62 | 15916,72 | 12 | 856,2059 | 2374,7066 | 15759,13 | 1 | 0 | 34,166 |
| 15 | 20726,11 | 16097,31 | 12 | 858,5333 | 2145,1019 | 15937,93 | 1 | 0 | 33,901 |
| Ort. | 21808,3 | 16284,04 | 12 | 745,0475 | 2316,5048 | 16081,51 | 1,26667 | 0,06667 | 34,062 |

Problem, 12 kümeli k-ortalamalar ile kümelenmiş başlangıç ile çözüldüğünde en iyi sonucu 3 numaralı deney vermiştir. Ortalama başlangıç çözümü 21808,3 km, ortalama en iyi çözüm 16284,04 km, ortalama minimum mesafe 745,05 km, ortalama maksimum mesafe 2316,5 km, ortalama toplam mesafe 16081,51 km, ortalama çözüm süresi 34,06 saniye olarak bulunmuştur. 15 çözümden 1 tanesi uygulanabilir çözümdür. Şekil 4.3. başlangıç çözümünü; Şekil 4.4. nihai çözümü göstermektedir.




Şekil 4.3. 12 kümeli k-ortalamalar ile kümelenmiş başlangıçlı en iyi çözümün başlangıç çözümü.



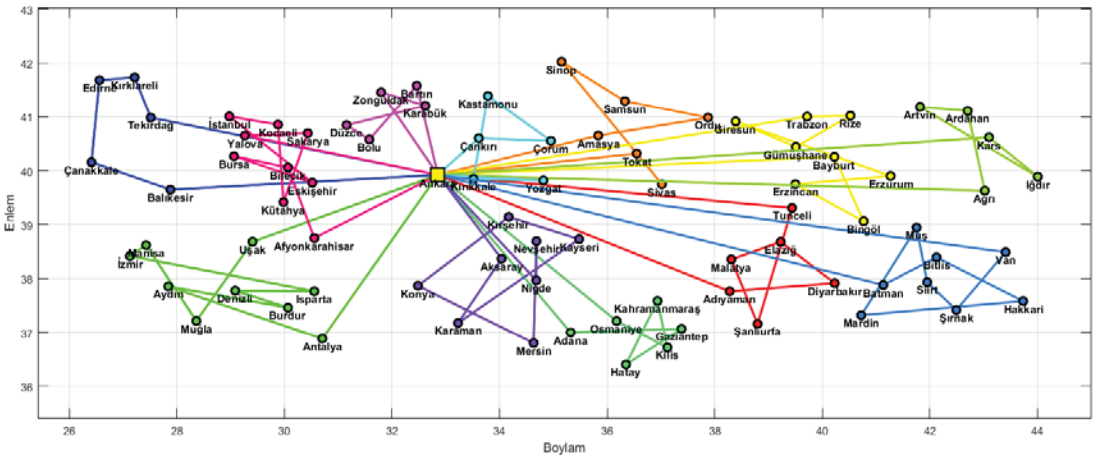

Şekil 4.4. 12 kümeli k-ortalamalar ile kümelenmiş başlangıçlı en iyi çözümün nihai çözümü.

Tablo 4.3. 12 kümeli bulanık c-ortalamalar ile kümelenmiş başlangıçlı çözüm sonuçları.

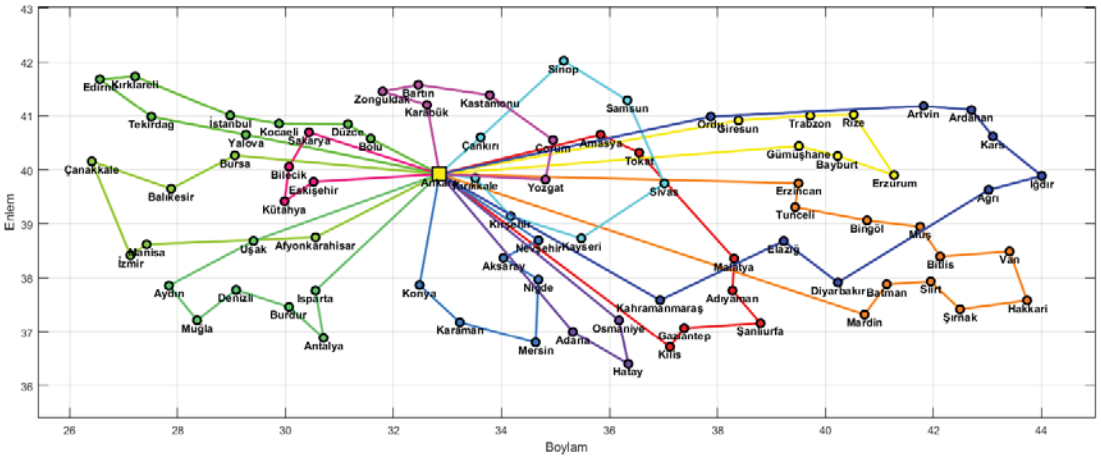
| Deney No | Başlangıç Çözümü | En İyi Çözüm | Küme Sayısı | Minimum Mesafe | Maksimum Mesafe | Toplam Mesafe | Toplam Fazla Kapasite | Uygulanabilir Çözüm | Çözüm Süresi |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 20856,12 | 16118,12 | 12 | 693,6245 | 2183,4083 | 16118,12 | 0 | 1 | 33,976 |
| 2 | 20274,13 | 16845,24 | 12 | 660,3134 | 2406,8446 | 16845,24 | 0 | 1 | 34,218 |
| 3 | 21717,91 | 16312,69 | 12 | 716,5878 | 2293,8122 | 16151,18 | 1 | 0 | 33,569 |
| 4 | 20856,12 | 15633,44 | 12 | 698,7528 | 2405,2975 | 15178,1 | 3 | 0 | 33,76 |
| 5 | 20856,12 | 16207,63 | 12 | 777,7182 | 2226,8662 | 16047,15 | 1 | 0 | 33,438 |
| 6 | 21717,91 | 16009,52 | 12 | 684,0689 | 2119,1895 | 16009,52 | 0 | 1 | 33,811 |
| 7 | 20274,13 | 15845,13 | 12 | 679,5598 | 2265,8588 | 15845,13 | 0 | 1 | 33,421 |
| 8 | 20856,12 | 16118,91 | 12 | 695,5593 | 2465,6468 | 15959,31 | 1 | 0 | 33,664 |
| 9 | 20856,12 | 16201,83 | 12 | 660,3134 | 2191,0233 | 16041,42 | 1 | 0 | 33,393 |
| 10 | 20856,12 | 15858,93 | 12 | 698,7528 | 2347,6152 | 15858,93 | 0 | 1 | 33,642 |
| **11** | **20321,44** | **15523,38** | **12** | **632,1836** | **2214,8394** | **15523,38** | **0** | **1** | **33,725** |
| 12 | 21260,06 | 16606,6 | 12 | 856,3641 | 2187,1448 | 16606,6 | 0 | 1 | 33,553 |
| 13 | 20856,12 | 16325,43 | 12 | 759,9745 | 2199,9643 | 16163,8 | 1 | 0 | 33,505 |
| 14 | 20856,12 | 16792,72 | 12 | 999,9048 | 2211,1039 | 16792,72 | 0 | 1 | 33,622 |
| 15 | 20856,12 | 16171,05 | 12 | 619,1711 | 2382,0533 | 15853,97 | 2 | 0 | 33,44 |
| Ort. | 20884,71 | 16171,38 | 12 | 722,1899 | 2273,3779 | 16066,31 | 0,66667 | 0,53333 | 33,649 |

Problem 12 kümeli bulanık c-ortalamalar ile kümelenmiş başlangıç ile çözüldüğünde en iyi sonucu 11 numaralı deney vermiştir. Ortalama başlangıç çözümü 20884,71 km, ortalama en iyi çözüm 16171,38 km, ortalama minimum mesafe 722,19 km,

ortalama maksimum mesafe 2273,38 km, ortalama toplam mesafe 16066,31 km, ortalama çözüm süresi 33,65 saniye olarak bulunmuştur. 15 çözümden 8 tanesi uygulanabilir çözümdür. Şekil 4.5. başlangıç çözümünü; Şekil 4.6. nihai çözümü göstermektedir.




Şekil 4.5. 12 kümeli bulanık c-ortalamalar ile kümelenmiş başlangıçlı en iyi çözümün başlangıç çözümü.

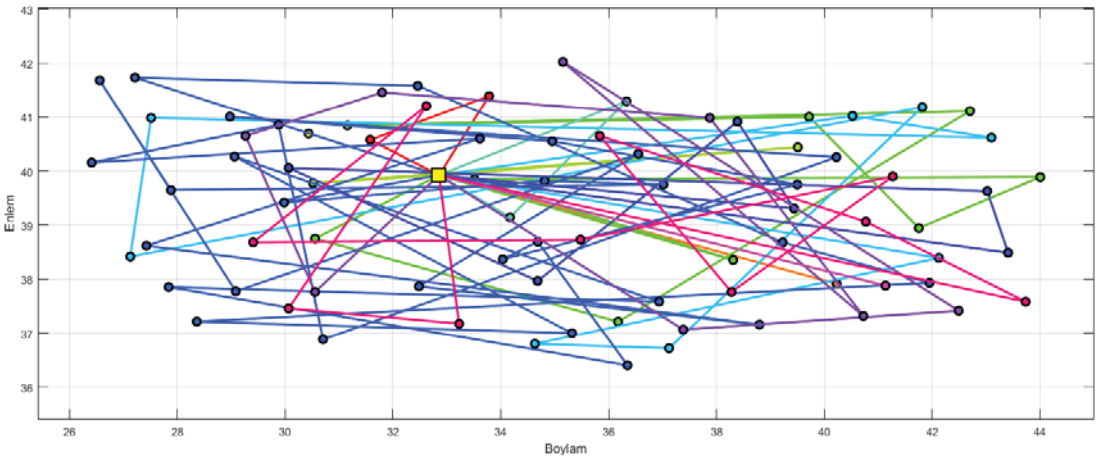



Şekil 4.6. 12 kümeli bulanık c-ortalamalar ile kümelenmiş başlangıçlı en iyi çözümün nihai çözümü.

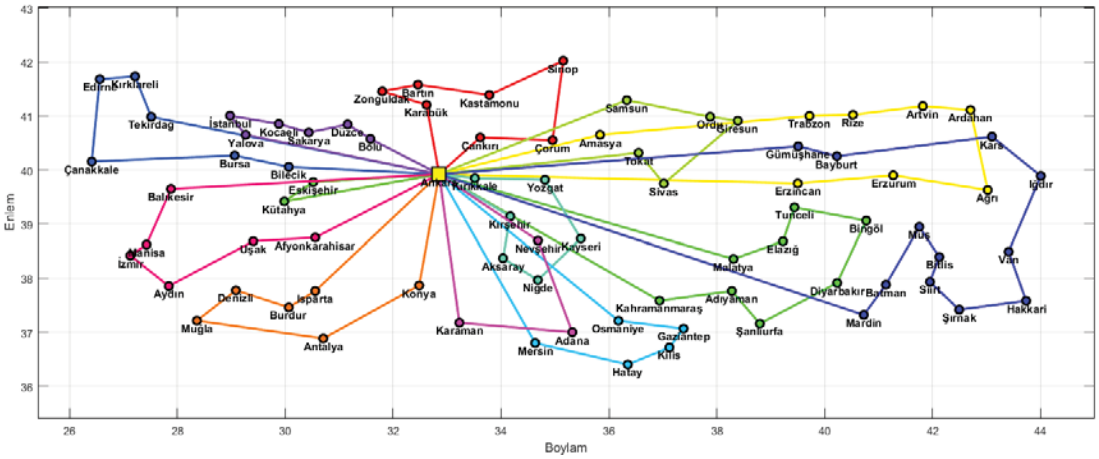
Tablo 4.4. 13 kümeli rassal başlangıçlı çözüm sonuçları.

| Deney No | Başlangıç Çözümü | En İyi Çözüm | Küme Sayısı | Minimum Mesafe | Maksimum Mesafe | Toplam Mesafe | Toplam Fazla Kapasite | Uygulanabilir Çözüm | Çözüm Süresi |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 49317,17 | 15833,97 | 13 | 428,7202 | 2378,7921 | 15833,97 | 0 | 1 | 34,906 |
| 2 | 52090,7 | 15899,82 | 13 | 417,556 | 2400,3257 | 15742,39 | 1 | 0 | 34,45 |
| 3 | 50413,92 | 15840,94 | 13 | 417,556 | 2308,6088 | 15840,94 | 0 | 1 | 34,723 |
| 4 | 50575,31 | 16019,47 | 13 | 474,9832 | 2149,9403 | 16019,47 | 0 | 1 | 34,669 |
| 5 | 48613,26 | 15892,07 | 13 | 505,64 | 2423,1411 | 15892,07 | 0 | 1 | 34,771 |
| 6 | 49264,25 | 15965,33 | 13 | 428,7202 | 2265,4637 | 15965,33 | 0 | 1 | 34,73 |
| 7 | 51860,19 | 15947,15 | 13 | 487,4924 | 2363,618 | 15789,26 | 1 | 0 | 34,47 |
| 8 | 47109,15 | 15834,81 | 13 | 539,7087 | 2356,8804 | 15834,81 | 0 | 1 | 34,539 |
| 9 | 53526,07 | 15732,05 | 13 | 450,1598 | 2369,7732 | 15732,05 | 0 | 1 | 34,725 |
| **10** | **48217,26** | **15800,67** | **13** | **512,8669** | **2511,1849** | **15800,67** | **0** | **1** | **34,425** |
| 11 | 51913,68 | 16125,1 | 13 | 399,7 | 2532,5885 | 16125,1 | 0 | 1 | 34,751 |
| 12 | 46184,96 | 16370,81 | 13 | 474,9832 | 2267,8834 | 16370,81 | 0 | 1 | 34,464 |
| 13 | 50333 | 15520,45 | 13 | 525,8825 | 2166,2095 | 15366,78 | 1 | 0 | 34,871 |
| 14 | 49490,56 | 16280,35 | 13 | 487,4924 | 2383,9985 | 15961,13 | 2 | 0 | 34,754 |
| 15 | 50086,68 | 16200,58 | 13 | 704,4885 | 2163,9992 | 16200,58 | 0 | 1 | 34,464 |
| Ort. | 49933,08 | 15950,91 | 13 | 483,73 | 2336,1605 | 15898,36 | 0,33333 | 0,73333 | 34,647 |

Problem 13 kümeli rassal başlangıç ile çözüldüğünde en iyi sonucu 10 numaralı deney vermiştir. Ortalama başlangıç çözümü 49933,08 km, ortalama en iyi çözüm 15950,91 km, ortalama minimum mesafe 483,73 km, ortalama maksimum mesafe 2336,16 km, ortalama toplam mesafe 15898,36 km, ortalama çözüm süresi 34,65 saniye olarak bulunmuştur. 15 çözümden 11 tanesi uygulanabilir çözümdür. Şekil 4.7. başlangıç çözümünü; Şekil 4.8. nihai çözümü göstermektedir.




Şekil 4.7. 13 kümeli rassal başlangıçlı en iyi çözümün başlangıç çözümü.



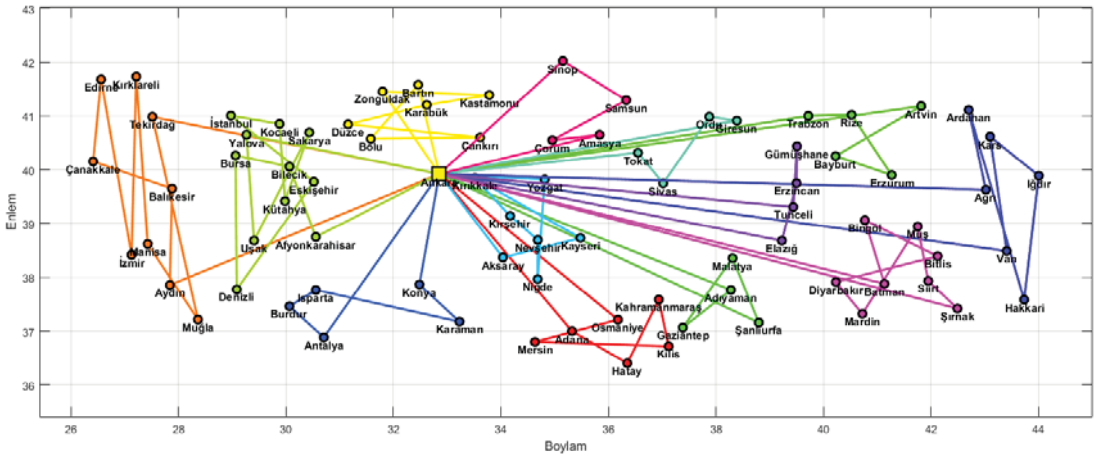

Şekil 4.8. 13 kümeli rassal başlangıçlı en iyi çözümün nihai çözümü.

Tablo 4.5. 13 kümeli k-ortalamalar ile kümelenmiş başlangıçlı çözüm sonuçları.

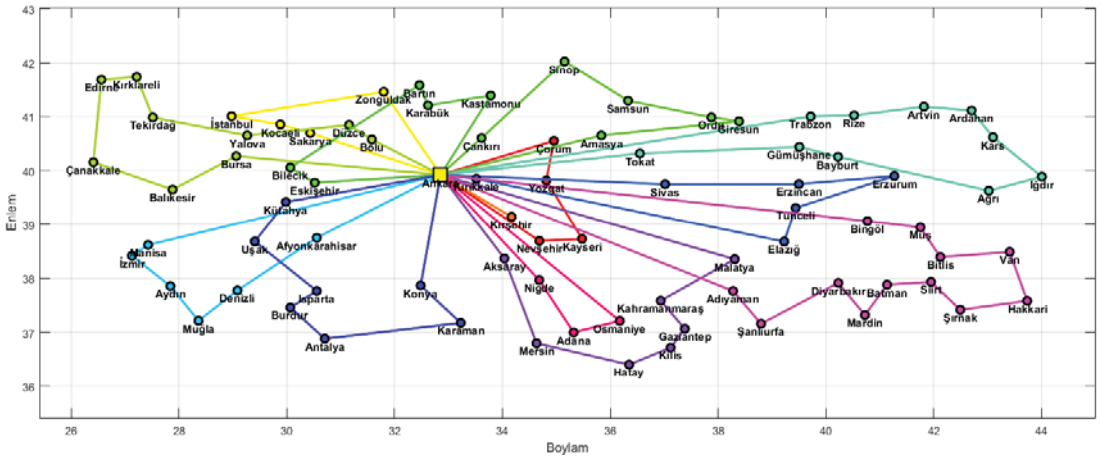
| Deney No | Başlangıç Çözümü | En İyi Çözüm | Küme Sayısı | Minimum Mesafe | Maksimum Mesafe | Toplam Mesafe | Toplam Fazla Kapasite | Uygulanabilir Çözüm | Çözüm Süresi |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 21088,79 | 15961,03 | 13 | 474,9832 | 2366,8461 | 15961,03 | 0 | 1 | 35,139 |
| 2 | 21065,04 | 15382,15 | 13 | 575,407 | 2153,8805 | 15229,85 | 1 | 0 | 35,376 |
| 3 | 21643,29 | 16233,29 | 13 | 417,556 | 2281,5682 | 16233,29 | 0 | 1 | 34,985 |
| 4 | 21962,92 | 15921,41 | 13 | 502,3368 | 2364,7775 | 15921,41 | 0 | 1 | 34,927 |
| 5 | 22844,8 | 16107,72 | 13 | 417,556 | 2241,3602 | 16107,72 | 0 | 1 | 35,038 |
| 6 | 21304,26 | 15602,79 | 13 | 539,7087 | 2184,9188 | 15448,31 | 1 | 0 | 35,273 |
| 7 | 19881,9 | 15945,61 | 13 | 417,556 | 2403,0887 | 15945,61 | 0 | 1 | 34,982 |
| 8 | 19980,16 | 15663,58 | 13 | 417,556 | 2609,7146 | 15663,58 | 0 | 1 | 35,049 |
| 9 | 20314,81 | 15866,33 | 13 | 651,0507 | 2292,0271 | 15866,33 | 0 | 1 | 35,314 |
| 10 | 22262,19 | 16066,28 | 13 | 399,7 | 2165,2007 | 16066,28 | 0 | 1 | 34,907 |
| 11 | 20730,12 | 15725,15 | 13 | 437,4377 | 2230,7999 | 15725,15 | 0 | 1 | 35,258 |
| 12 | 23448,68 | 16171,27 | 13 | 295,7122 | 2454,5769 | 16171,27 | 0 | 1 | 35,058 |
| 13 | 22912,49 | 16026,98 | 13 | 450,1598 | 2359,7754 | 16026,98 | 0 | 1 | 35,07 |
| 14 | 23445,87 | 15762,67 | 13 | 539,7087 | 2223,9769 | 15762,67 | 0 | 1 | 34,995 |
| **15** | **22921,48** | **15461** | **13** | **284,548** | **2226,8662** | **15461** | **0** | **1** | **35,085** |
| Ort. | 21720,45 | 15859,82 | 13 | 454,7318 | 2303,9585 | 15839,37 | 0,13333 | 0,86667 | 35,097 |

Problem 13 kümeli k-ortalamalar ile kümelenmiş başlangıç ile çözüldüğünde en iyi sonucu 15 numaralı deney vermiştir. Ortalama başlangıç çözümü 21720,45 km, ortalama en iyi çözüm 15859,82 km, ortalama minimum mesafe 454,73 km, ortalama

maksimum mesafe 2303,96 km, ortalama toplam mesafe 15839,37 km, ortalama çözüm süresi 35,1 saniye olarak bulunmuştur. 15 çözümden 13 tanesi uygulanabilir çözümdür. Şekil 4.9. başlangıç çözümünü; Şekil 4.10. nihai çözümü göstermektedir.




Şekil 4.9. 13 kümeli k-ortalamalar ile kümelenmiş başlangıçlı en iyi çözümün başlangıç çözümü.

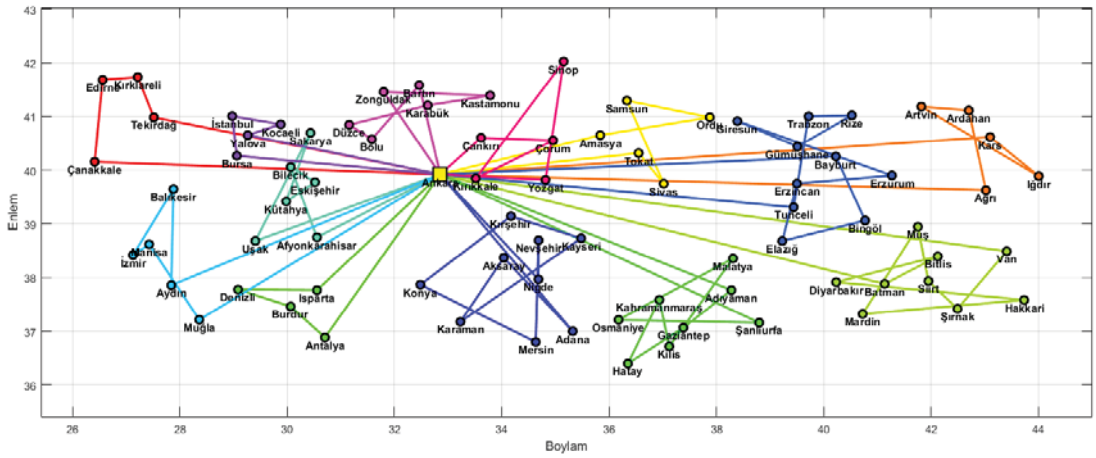



Şekil 4.10. 13 kümeli k-ortalamalar ile kümelenmiş başlangıçlı en iyi çözümün nihai çözümü.

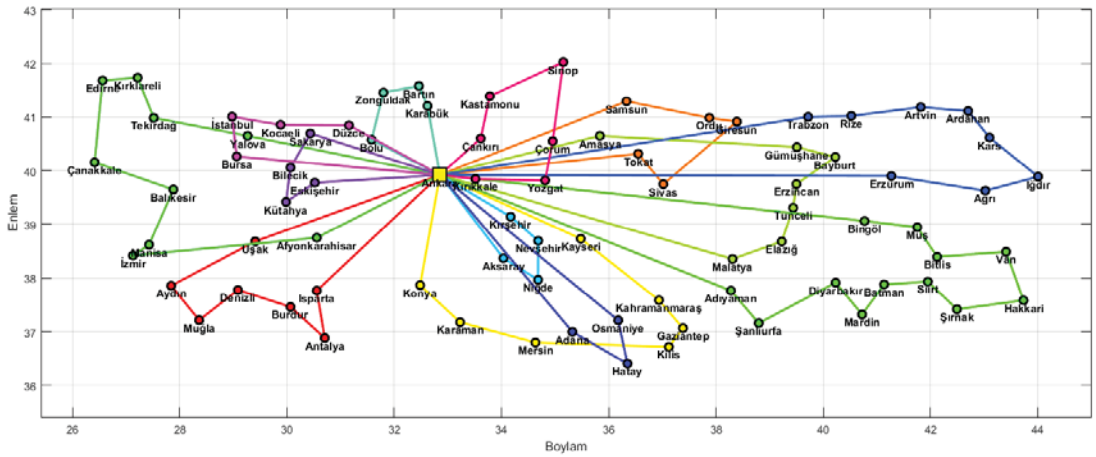
Tablo 4.6. 13 kümeli bulanık c-ortalamalar ile kümelenmiş başlangıçlı çözüm sonuçları.

| Deney No | Başlangıç Çözümü | En İyi Çözüm | Küme Sayısı | Minimum Mesafe | Maksimum Mesafe | Toplam Mesafe | Toplam Fazla Kapasite | Uygulanabilir Çözüm | Çözüm Süresi |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 20806,95 | 15832,4 | 13 | 437,4377 | 2346,2753 | 15832,4 | 0 | 1 | 34,469 |
| 2 | 21476,36 | 15589,93 | 13 | 441,9619 | 2364,5582 | 15589,93 | 0 | 1 | 34,585 |
| 3 | 20806,95 | 15441,87 | 13 | 417,556 | 2371,5039 | 15441,87 | 0 | 1 | 34,59 |
| 4 | 21255,83 | 15721,4 | 13 | 417,556 | 2475,6356 | 15721,4 | 0 | 1 | 34,561 |
| **5** | **21351,41** | **15008,19** | **13** | **474,9832** | **2226,8662** | **15008,19** | **0** | **1** | **34,905** |
| 6 | 20806,95 | 15491,48 | 13 | 417,556 | 2226,8662 | 15491,48 | 0 | 1 | 34,636 |
| 7 | 21130,08 | 15819,58 | 13 | 437,4377 | 2170,9131 | 15819,58 | 0 | 1 | 34,503 |
| 8 | 20806,95 | 15563,24 | 13 | 428,7202 | 2168,3869 | 15563,24 | 0 | 1 | 34,436 |
| 9 | 22092,89 | 15794,72 | 13 | 541,9302 | 2222,4666 | 15794,72 | 0 | 1 | 34,895 |
| 10 | 21118,83 | 15484,17 | 13 | 417,556 | 2150,4097 | 15484,17 | 0 | 1 | 34,557 |
| 11 | 21572,44 | 15605,05 | 13 | 555,2454 | 2367,1425 | 15605,05 | 0 | 1 | 34,493 |
| 12 | 21536,05 | 15440,78 | 13 | 568,371 | 2339,3172 | 15287,9 | 1 | 0 | 34,508 |
| 13 | 21340,09 | 15658,85 | 13 | 417,556 | 2347,6152 | 15658,85 | 0 | 1 | 34,372 |
| 14 | 20806,95 | 15495,43 | 13 | 399,7 | 2157,0576 | 15495,43 | 0 | 1 | 34,386 |
| 15 | 21150,86 | 15301,07 | 13 | 417,556 | 2197,445 | 15301,07 | 0 | 1 | 34,844 |
| Ort. | 21203,97 | 15549,88 | 13 | 452,7416 | 2275,4973 | 15539,69 | 0,06667 | 0,93333 | 34,583 |

Problem 13 kümeli bulanık c-ortalamalar ile kümelenmiş başlangıç ile çözüldüğünde en iyi sonucu 5 numaralı deney vermiştir. Ortalama başlangıç çözümü 21203,97 km, ortalama en iyi çözüm 15549,88 km, ortalama minimum mesafe 452,74 km, ortalama maksimum mesafe 2275,5 km, ortalama toplam mesafe 15539,69 km, ortalama çözüm süresi 34,58 saniye olarak bulunmuştur. 15 çözümden 14 tanesi uygulanabilir çözümdür. Şekil 4.11. başlangıç çözümünü; Şekil 4.12. nihai çözümü göstermektedir.




Şekil 4.11. 13 kümeli bulanık c-ortalamalar ile kümelenmiş başlangıçlı en iyi çözümün başlangıç çözümü.



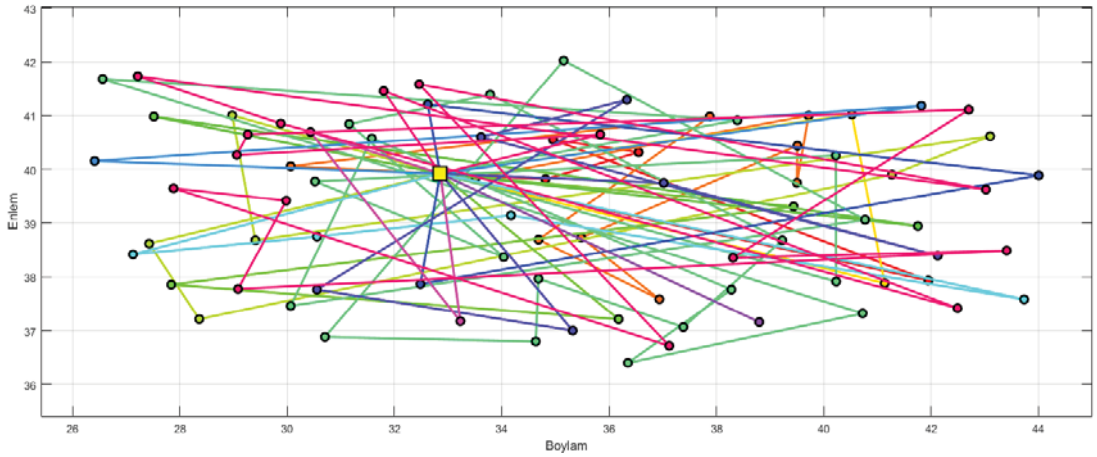

Şekil 4.12. 13 kümeli bulanık c-ortalamalar ile kümelenmiş başlangıçlı en iyi çözümün nihai çözümü.

Tablo 4.7. 14 kümeli rassal başlangıçlı çözüm sonuçları.

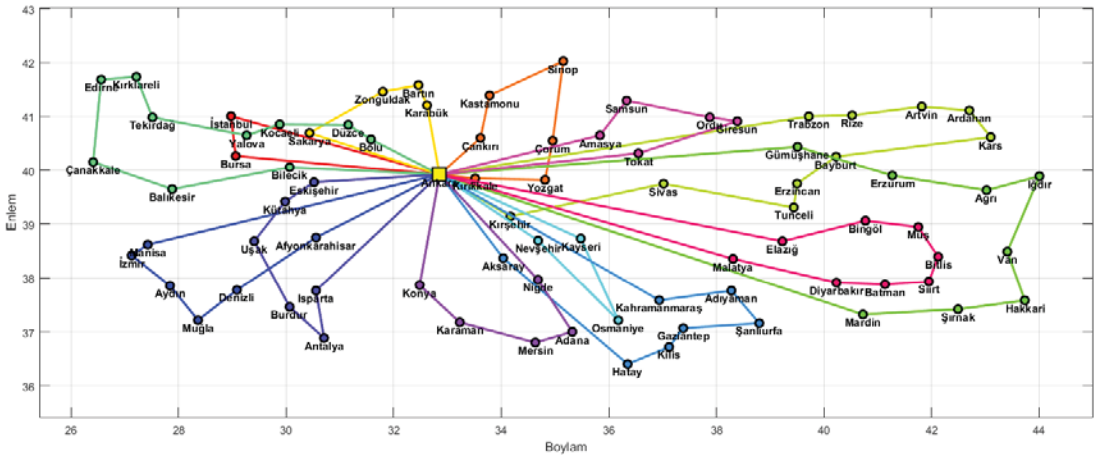
| Deney No | Başlangıç Çözümü | En İyi Çözüm | Küme Sayısı | Minimum Mesafe | Maksimum Mesafe | Toplam Mesafe | Toplam Fazla Kapasite | Uygulanabilir Çözüm | Çözüm Süresi |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 52008,33 | 16731,49 | 13 | 609,5057 | 2466,4281 | 16731,49 | 0 | 1 | 36,18 |
| 2 | 47148,91 | 15984,36 | 13 | 437,4377 | 2240,8385 | 15984,36 | 0 | 1 | 34,225 |
| 3 | 50346,08 | 15883,95 | 13 | 619,9646 | 2255,6597 | 15883,95 | 0 | 1 | 34,035 |
| **4** | **50941,97** | **15731,82** | **13** | **609,5057** | **2249,0385** | **15731,82** | **0** | **1** | **33,995** |
| 5 | 44933,86 | 16167,57 | 13 | 335,718 | 2206,8023 | 16167,57 | 0 | 1 | 33,815 |
| 6 | 48562,83 | 15940,38 | 13 | 403,5373 | 2152,8486 | 15782,55 | 1 | 0 | 33,791 |
| 7 | 50279,74 | 16368,3 | 13 | 487,4924 | 2257,3432 | 16368,3 | 0 | 1 | 34,574 |
| 8 | 53506,97 | 16064,95 | 13 | 512,8669 | 2430,3724 | 16064,95 | 0 | 1 | 34,043 |
| 9 | 47993,34 | 16009,88 | 13 | 539,7087 | 2354,5874 | 16009,88 | 0 | 1 | 33,702 |
| 10 | 49408,72 | 15849,3 | 13 | 441,9619 | 2338,5959 | 15849,3 | 0 | 1 | 33,878 |
| 11 | 49290,78 | 16055,1 | 13 | 474,9832 | 2370,7283 | 15896,14 | 1 | 0 | 34,109 |
| 12 | 46872,04 | 16278,27 | 13 | 539,7087 | 2383,8769 | 16278,27 | 0 | 1 | 33,955 |
| 13 | 45774,85 | 15996,04 | 13 | 437,4377 | 2359,6672 | 15996,04 | 0 | 1 | 33,788 |
| 14 | 52372,4 | 15968,74 | 13 | 524,4559 | 2121,9523 | 15968,74 | 0 | 1 | 34,033 |
| 15 | 52414,15 | 16342,58 | 13 | 568,371 | 2117,6578 | 16342,58 | 0 | 1 | 34,039 |
| Ort. | 49457 | 16091,51 | 13 | 502,8437 | 2287,0931 | 16070,39 | 0,13333 | 0,86667 | 34,144 |

Problem 14 kümeli rassal başlangıç ile çözüldüğünde en iyi sonucu 4 numaralı deney vermiştir. Ortalama başlangıç çözümü 49457 km, ortalama en iyi çözüm 16091,51 km, ortalama minimum mesafe 502,84 km, ortalama maksimum mesafe 2287,09 km,

ortalama toplam mesafe 16070,39 km, ortalama çözüm süresi 34,14 saniye olarak bulunmuştur. 15 çözümden 13 tanesi uygulanabilir çözümdür. Şekil 4.13. başlangıç çözümünü; Şekil 4.14. nihai çözümü göstermektedir.




Şekil 4.13. 14 kümeli rassal başlangıçlı en iyi çözümün başlangıç çözümü.

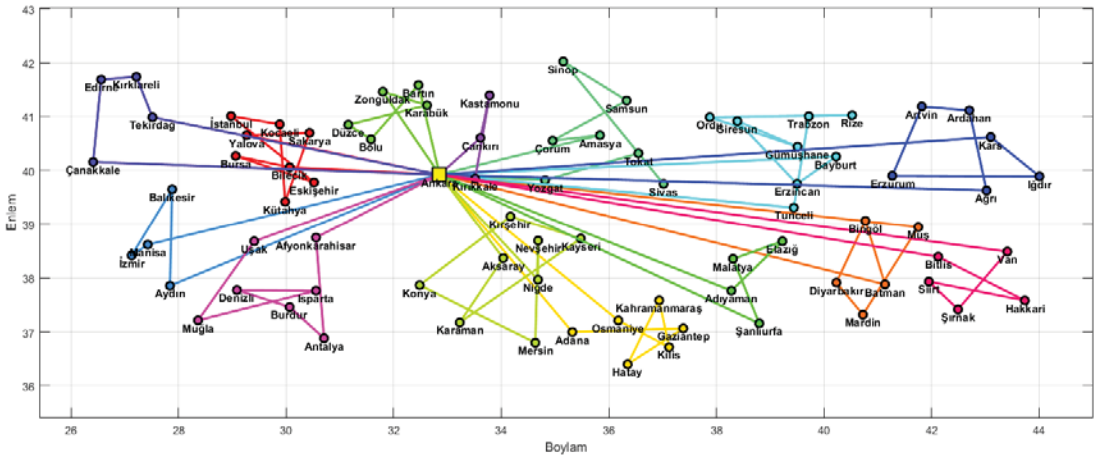



Şekil 4.14. 14 kümeli rassal başlangıçlı en iyi çözümün nihai çözümü.

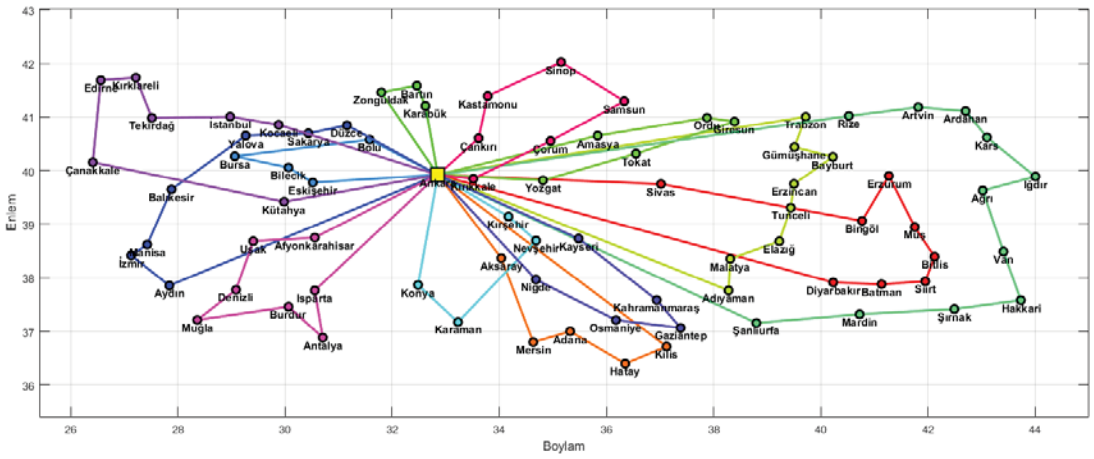
Tablo 4.8. 14 kümeli k-ortalamalar ile kümelenmiş başlangıçlı çözüm sonuçları.

| Deney No | Başlangıç Çözümü | En İyi Çözüm | Küme Sayısı | Minimum Mesafe | Maksimum Mesafe | Toplam Mesafe | Toplam Fazla Kapasite | Uygulanabilir Çözüm | Çözüm Süresi |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 22217,79 | 15817,38 | 13 | 437,4377 | 2294,7109 | 15817,38 | 0 | 1 | 35,715 |
| 2 | 22047,22 | 15530,86 | 13 | 437,4377 | 2302,2477 | 15530,86 | 0 | 1 | 35,495 |
| 3 | 22528,74 | 15504,85 | 13 | 566,824 | 2378,6627 | 15504,85 | 0 | 1 | 35,765 |
| 4 | 21910,16 | 16129,25 | 13 | 437,4377 | 2380,5542 | 16129,25 | 0 | 1 | 35,653 |
| 5 | 22582,23 | 15932,78 | 13 | 417,556 | 2308,4881 | 15932,78 | 0 | 1 | 35,786 |
| 6 | 21587,39 | 15993,71 | 13 | 544,6952 | 2332,1429 | 15993,71 | 0 | 1 | 35,507 |
| 7 | 23344,58 | 15832,66 | 13 | 428,7202 | 2161,2942 | 15832,66 | 0 | 1 | 35,556 |
| 8 | 20845,77 | 15783,25 | 13 | 437,4377 | 2228,2082 | 15783,25 | 0 | 1 | 35,517 |
| 9 | 23228,78 | 15836,08 | 13 | 417,556 | 2069,6788 | 15836,08 | 0 | 1 | 35,512 |
| 10 | 23260 | 15655,21 | 13 | 417,556 | 2388,2071 | 15655,21 | 0 | 1 | 35,404 |
| 11 | 22103,8 | 16291,94 | 13 | 417,556 | 2186,4128 | 16291,94 | 0 | 1 | 35,55 |
| 12 | 21797,01 | 15955,1 | 13 | 393,8493 | 2572,4541 | 15955,1 | 0 | 1 | 35,457 |
| **13** | **21775,36** | **15456,97** | **13** | **437,4377** | **2384,9126** | **15456,97** | **0** | **1** | **35,398** |
| 14 | 21610,4 | 15598,05 | 13 | 539,7087 | 2331,7571 | 15598,05 | 0 | 1 | 35,564 |
| 15 | 25758,08 | 15639,73 | 13 | 502,64 | 2171,0697 | 15484,89 | 1 | 0 | 35,433 |
| Ort. | 22439,82 | 15797,19 | 13 | 455,59 | 2299,3867 | 15786,87 | 0,06667 | 0,93333 | 35,554 |

Problem 14 kümeli k-ortalamalar ile kümelenmiş başlangıç ile çözüldüğünde en iyi sonucu 13 numaralı deney vermiştir. Ortalama başlangıç çözümü 22439,82 km, ortalama en iyi çözüm 15797,19 km, ortalama minimum mesafe 455,59 km, ortalama maksimum mesafe 2299,39 km, ortalama toplam mesafe 15786,87 km, ortalama çözüm süresi 35,55 saniye olarak bulunmuştur. 15 çözümden 14 tanesi uygulanabilir çözümdür. Şekil 4.15. başlangıç çözümünü; Şekil 4.16. nihai çözümü göstermektedir.




Şekil 4.15. 14 kümeli k-ortalamalar ile kümelenmiş başlangıçlı en iyi çözümün başlangıç çözümü.



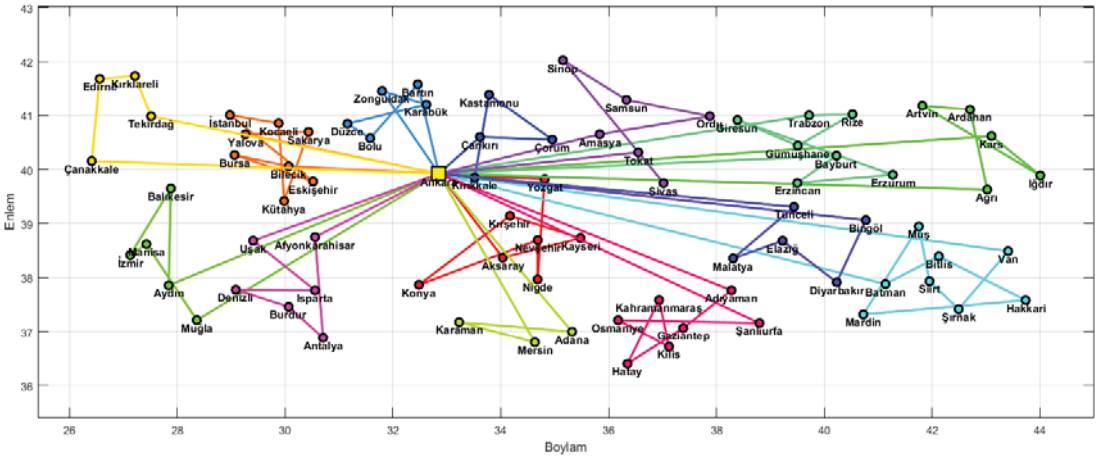

Şekil 4.16. 14 kümeli k-ortalamalar ile kümelenmiş başlangıçlı en iyi çözümün nihai çözümü.

Tablo 4.9. 14 kümeli bulanık c-ortalamalar ile kümelenmiş başlangıçlı çözüm sonuçları.

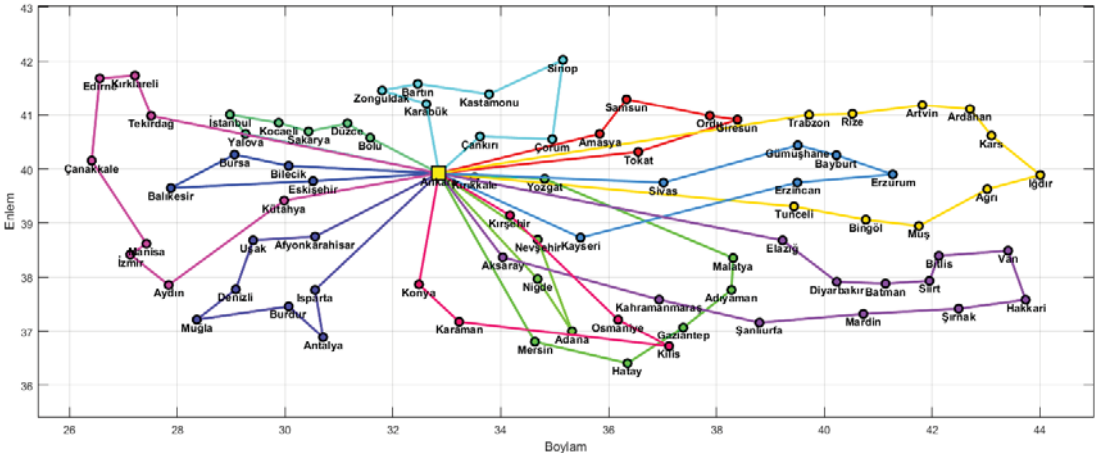
| Deney No | Başlangıç Çözümü | En İyi Çözüm | Küme Sayısı | Minimum Mesafe | Maksimum Mesafe | Toplam Mesafe | Toplam Fazla Kapasite | Uygulanabilir Çözüm | Çözüm Süresi |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 22186,53 | 15979,19 | 13 | 539,7087 | 2260,3659 | 15979,19 | 0 | 1 | 35,581 |
| **2** | **21447,97** | **15470,79** | **13** | **114,0158** | **2188,5518** | **15470,79** | **0** | **1** | **34,872** |
| 3 | 21170,9 | 15597,02 | 13 | 417,556 | 2265,1814 | 15597,02 | 0 | 1 | 35,012 |
| 4 | 21470,31 | 15836,12 | 13 | 437,4377 | 2172,7982 | 15836,12 | 0 | 1 | 35,16 |
| 5 | 21632,33 | 15849,41 | 13 | 660,4332 | 2166,6857 | 15849,41 | 0 | 1 | 34,525 |
| 6 | 21257,43 | 15480,09 | 13 | 474,9832 | 2307,7338 | 15480,09 | 0 | 1 | 34,603 |
| 7 | 22551,88 | 15627,18 | 13 | 437,4377 | 2233,6559 | 15627,18 | 0 | 1 | 35,093 |
| 8 | 21635,58 | 15722,26 | 13 | 428,7202 | 2423,9696 | 15722,26 | 0 | 1 | 35,236 |
| 9 | 22055,7 | 16044,53 | 13 | 437,4377 | 2282,5015 | 16044,53 | 0 | 1 | 34,68 |
| 10 | 22021,56 | 16196,93 | 13 | 441,9619 | 2353,1074 | 16196,93 | 0 | 1 | 34,508 |
| 11 | 22725,12 | 15902,6 | 13 | 558,7668 | 2160,8319 | 15902,6 | 0 | 1 | 34,546 |
| 12 | 21775,46 | 15751,93 | 13 | 417,556 | 2392,6633 | 15751,93 | 0 | 1 | 35,068 |
| 13 | 21257,43 | 15739,58 | 13 | 417,556 | 2197,1216 | 15739,58 | 0 | 1 | 34,816 |
| 14 | 22186,53 | 15463,61 | 12 | 632,1836 | 2069,6788 | 15310,51 | 1 | 0 | 34,754 |
| 15 | 22601,84 | 15554,53 | 13 | 417,556 | 2146,771 | 15554,53 | 0 | 1 | 34,895 |
| Ort. | 21865,1 | 15747,72 | 12,93 | 455,554 | 2241,4412 | 15737,51 | 0,06667 | 0,93333 | 34,89 |

Problem 14 kümeli bulanık c-ortalamalar ile kümelenmiş başlangıç ile çözüldüğünde en iyi sonucu 2 numaralı deney vermiştir. Ortalama başlangıç çözümü 21865,1 km, ortalama en iyi çözüm 15747,72 km, ortalama minimum mesafe 455,55 km, ortalama

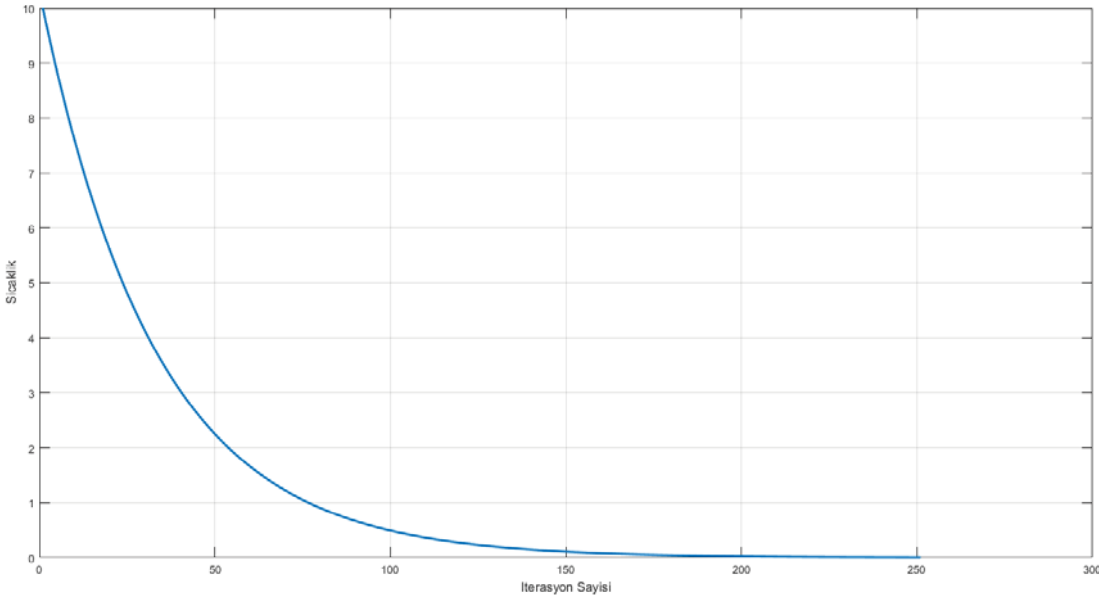
maksimum mesafe 2241,44 km, ortalama toplam mesafe 15737,51 km, ortalama çözüm süresi 34,89 saniye olarak bulunmuştur. 15 çözümden 14 tanesi uygulanabilir çözümdür. Şekil 4.17. başlangıç çözümünü; Şekil 4.18. nihai çözümü göstermektedir.



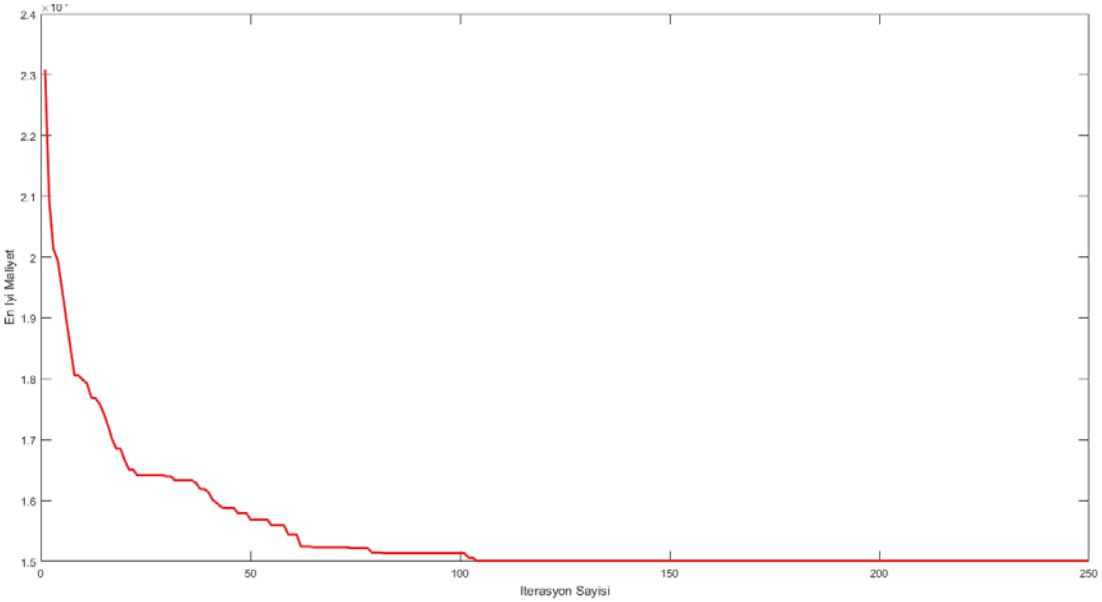

Şekil 4.17. 14 kümeli bulanık c-ortalamalar ile kümelenmiş başlangıçlı en iyi çözümün başlangıç çözümü.




Şekil 4.18. 14 kümeli bulanık c-ortalamalar ile kümelenmiş başlangıçlı en iyi çözümün nihai çözümü.




Şekil 4.19. Soğuma grafiği.




Şekil 4.20. En iyi çözümün amaç fonksiyonu değişimi.

Eşlenik t testinin uygulanması: Başlangıç çözümünün belirlenmesinde üç farklı yöntem kullanılmıştır. K-ortalamalar ile kümeleme ve bulanık c-ortalamalar ile kümeleme yöntemlerinin rassal başlangıç çözümüne karşı iyi sonuçlar verir vermediği test edilmiştir. Bu test için eşlenik t testi kullanılmıştır. Bu test başlangıç çözümü ve toplam mesafe değerleri için uygulanmıştır. Test sonuçları sırasıyla Tablo

4.10.'da ve Tablo 4.11.'de gösterilmiştir. En iyi sonuçlar 13 kümeli modelde elde edildiğinden test için de 13 kümeli model kullanılmıştır. Sonuçlar aşağıdaki tabloda verilmiştir.

Tablo 4.10. Başlangıç çözümü için test sonuçları.

| | Rassal ve K-ortalamalar | | Rassal ve Bulanık c-ortalamalar | | K-ortalamalar ve Bulanık c-ortalamalar | |
|---|---|---|---|---|---|---|
| Ortalama | 49933,07 | 21720,45 | 49933,07 | 21203,97 | 21720,45 | 21203,97 |
| Varyans | 3839999,49 | 1486920,02 | 3839999,49 | 139787,04 | 1486920,02 | 139787,04 |
| Gözlem | 15 | 15 | 15 | 15 | 15 | 15 |
| Pearson Korelasyonu | -0,4562 | | 0,4652 | | -0,0627 | |
| df | 14 | | 14 | | 14 | |
| t Stat | 39,8795 | | 61,2678 | | 1,5415 | |
| P(T<=t) tek-uçlu | 0,0000 | | 0,0000 | | 0,0727 | |
| t Kritik tek-uçlu | 1,7613 | | 1,7613 | | 1,7613 | |
| **P(T<=t) iki-uçlu** | **0,0000** | | **0,0000** | | **0,1455** | |
| t Kritik iki-uçlu | 2,1448 | | 2,1448 | | 2,1448 | |

Başlangıç çözümü için test sonuçlarına bakıldığında rassal başlangıç çözümü ve k-ortalamalar başlangıç çözümü için $P < 0,05$ olduğundan değerler arasında fark yoktur hipotezi reddedilir. Yani değerler arasında fark vardır. K-ortalamalar başlangıç çözümü rassal başlangıç çözümüne göre daha iyi sonuçlar vermiştir. Rassal başlangıç çözümü ve bulanık c-ortalamalar başlangıç çözümü için $P < 0,05$ olduğundan değerler arasında fark yoktur hipotezi reddedilir. Yani değerler arasında fark vardır. Bulanık c-ortalamalar başlangıç çözümü rassal başlangıç çözümüne göre daha iyi sonuçlar vermiştir. K-ortalamalar başlangıç çözümü ve bulanık c-ortalamalar başlangıç çözümü için $P > 0,05$ olduğundan değerler arasında fark yoktur hipotezi kabul edilir. Yani değerler arasında fark yoktur.

Tablo 4.11. Toplam mesafe için test sonuçları.

| | Rassal ve K-ortalamalar | | Rassal ve Bulanık c-ortalamalar | | K-ortalamalar ve Bulanık c-ortalamalar | |
|---|---|---|---|---|---|---|
| Ortalama | 15898,35 | 15839,36 | 15898,35 | 15539,68 | 15839,36 | 15539,68 |
| Varyans | 54041,80 | 83928,95 | 54041,80 | 50242,40 | 83928,95 | 50242,40 |
| Gözlem | 15 | 15 | 15 | 15 | 15 | 15 |
| Pearson Korelasyonu | -0,0725 | | -0,4254 | | -0,1073 | |
| df | 14 | | 14 | | 14 | |
| t Stat | 0,5944 | | 3,6034 | | 3,0159 | |
| P(T<=t) tek-uçlu | 0,2809 | | 0,0014 | | 0,0046 | |
| t Kritik tek-uçlu | 1,7613 | | 1,7613 | | 1,7613 | |
| **P(T<=t) iki-uçlu** | **0,5617** | | **0,0029** | | **0,0093** | |
| t Kritik iki-uçlu | 2,1448 | | 2,1448 | | 2,1448 | |

Toplam mesafe (nihai çözüm) için test sonuçlarına bakıldığında rassal başlangıç çözümü ve k-ortalamalar başlangıç çözümü için P > 0,05 olduğundan değerler arasında fark yoktur hipotezi kabul edilir. Yani değerler arasında fark yoktur. K-ortalamalar başlangıç çözümü rassal başlangıç çözümüne göre daha iyi sonuçlar vermemiştir. Rassal başlangıç çözümü ve bulanık c-ortalamalar başlangıç çözümü için P < 0,05 olduğundan değerler arasında fark yoktur hipotezi reddedilir. Yani değerler arasında fark vardır. Bulanık c-ortalamalar başlangıç çözümü rassal başlangıç çözümüne göre daha iyi sonuçlar vermiştir. K-ortalamalar başlangıç çözümü ve bulanık c-ortalamalar başlangıç çözümü için P < 0,05 olduğundan değerler arasında fark yoktur hipotezi reddedilir. Yani değerler arasında fark vardır. Bulanık c-ortalamalar başlangıç çözümü, k-ortalamalar başlangıç çözümüne göre daha iyi sonuçlar vermiştir.

# BÖLÜM 5. TARTIŞMA VE SONUÇ

Problemi çözmek için kullanılan Tavlama Benzetimi algoritmasında üst iterasyon sayısı 250 ve alt iterasyon sayısı 500 olmak üzere, arama uzayında toplamda 125.000 adet rota oluşturacak kombinasyon test edilmiştir. İlk sıcaklık değeri 10, sıcaklık düşürme oranı 0,97 olarak belirlenmiştir. Problemde, 26 birim kapasiteli özdeş araçlar kullanılmıştır. Problemdeki müşterilerin toplam talebi 311'dir. Problemi minimum araç sayısı ile çözmek istersek, 12 (311/26≈12) araca ihtiyaç vardır. Farklı araç sayıları için sonuçları gözlemleyebilmek için 12, 13 ve 14 araçlı olacak şekilde modeller oluşturulmuştur. 3 farklı araç sayısı için rassal başlangıç çözümü, k-ortalamalar ile kümelenmiş başlangıç çözümü ve bulanık c-ortalamalar ile kümelenmiş başlangıç çözümü olmak üzere toplamda 6 adet model oluşturulmuştur. Her bir modelde 15 adet olmak üzere toplamda 90 deney yapılmıştır. K-ortalamalar ile kümelenmiş modellerin rassal başlangıç çözümlü modellere göre daha iyi sonuçlar verdiği gözlenmemiştir. Aşağıda sadece uygulanabilir çözümlerin istatistikleri verilmiştir.

12 araç için rassal çözüm oluşturulduğunda ortalama arama uzayı 50834,62 kilometre, ortalama en iyi çözüm diğeri 16919,2 kilometre, tüm çözümlerde oluşan rota(küme) sayısı 12, minimum rotaların ortalaması 779,94 kilometre, maksimum rotaların ortalaması 2393,42 kilometre, ortalama çözüm süresi 33,55 saniye olarak bulunmuştur. 15 deneyden 6 tanesi uygulanabilir çözümdür. Uygulanabilir çözüm bulunma ihtimali %40'tır. 12 araç için bulanık c-ortalamalar ile kümelenmiş çözüm oluşturulduğunda ortalama arama uzayı 20802 kilometre, ortalama en iyi çözüm değeri 16199,96 kilometre, tüm çözümlerde oluşan rota (küme) sayısı 12, minimum rotaların ortalaması 730,81 kilometre, maksimum rotaların ortalaması 2242 kilometre, ortalama çözüm süresi 33,75 saniye olarak bulunmuştur. 15 deneyden 8 tanesi uygulanabilir çözümdür. Uygulanabilir çözüm bulunma ihtimali %53'tür.

13 araç için rassal çözüm oluşturulduğunda ortalama arama uzayı 49565,61 kilometre, ortalama en iyi çözüm değeri 15965,0 8 kilometre, tüm çözümlerde oluşan rota (küme) sayısı 13, minimum rotaların ortalaması 485,23 kilometre, maksimum rotaların ortalaması 2338,93 kilometre, ortalama çözüm süresi 34,65 saniye olarak bulunmuştur. 15 deneyden 11 tanesi uygulanabilir çözümdür. Uygulanabilir çözüm bulunma ihtimali %73'tür. 13 araç için bulanık c-ortalamalar ile kümelenmiş çözüm oluşturulduğunda ortalama arama uzayı 21180,25 kilometre, ortalama en iyi çözüm değeri 15557,67 kilometre, tüm çözümlerde oluşan rota (küme) sayısı 13, minimum rotaların ortalaması 444,48 kilometre, maksimum rotaların ortalaması 2270,94 kilometre, ortalama çözüm suresi 34,59 saniye olarak bulunmuştur. 15 deneyden 14 tanesi uygulanabilir çözümdür. Uygulanabilir çözüm bulunma ihtimali %93'tür.

14 araç için rassal çözüm oluşturulduğunda ortalama arama uzayı 49538,56 kilometre, ortalama en iyi çözüm değeri 16105,94 kilometre, tüm çözümlerde oluşan rota (küme) sayısı 13, minimum rotaların ortalaması 512,63 kilometre, maksimum rotaların ortalaması 2290,99 kilometre, ortalama çözüm süresi 34,17 saniye olarak bulunmuştur. 15 deneyden 13 tanesi uygulanabilir çözümdür. Uygulanabilir çözüm bulunma ihtimali %87'dir. 14 araç için bulanık c-ortalamalar ile kümelenmiş çözüm oluşturulduğunda ortalama arama uzayı 21842,15 kilometre, ortalama en iyi çözüm değeri 15768,01 kilometre, oluşan rota (küme) sayısı bir deney için 12, on dört deney için 13, minimum rotaların ortalaması 442,94 kilometre, maksimum rotaların ortalaması 2253,71 kilometre, ortalama çözüm süresi 34,9 saniye olarak bulunmuştur. 15 deneyden 14 tanesi uygulanabilir çözümdür. Uygulanabilir çözüm bulunma ihtimali %93'tür.

Tüm modeller için ortalamalara bakıldığında en iyi sonucu 13 araç için bulanık c-ortalamalar ile kümelenmiş çözümlü model vermiştir. Tüm çözümlere genel olarak bakıldığında yine aynı model içerisindeki 5 numaralı deney en iyi sonucu vermiştir. Bu deneyin sonuçları: Arama uzayı 21351,41 kilometre, en iyi çözüm 15.008,19 kilometre, rota (küme) sayısı 13, minimum rota 447,98 kilometre, maksimum rota 2226,87 kilometre, çözüm süresi 34,9 saniyedir. En iyi çözümün 12 araç için uygulanan modeller olması beklenmiştir. Ama bu modellerde müşterilerin toplam

taleplerini karşılayabilmek için toplam araç kapasitelerinin %99,68 oranında kullanılacağından uygulanabilir bir çözüm bulamadan algoritma çıkmaza girer. 14 araç için uygulanan modeller de amaç fonksiyonu optimum çözüme ulaşmaya çalıştığında algoritmanın en rahat çalışacağı araç sayısına düşmüştür. Karar vericiler 12 araç için fazla mesafe katetmeyi göze alırlarsa bu model kullanılabilir. Bir aracın, rotasını tamamlaması için gereken maliyet, fazladan katedilen mesafeden büyük ise 12 araçlı model; diğer durumlarda 13 araçlı model kullanılabilir.

Bu model ile her başlangıçta başlangıç çözümü farklı olduğundan ve noktalar, mesafe olarak birbirine en yakın olacak şekilde kümelendiğinden ve arama alanında ortalama olarak %57 oranında azaltma yapıldığından dolayı çıkan sonuçlar ümit vericidir. Hesaplama sonuçları, kümeleme tabanlı bir başlangıç çözümü kullanmanın, literatürdeki önceki algoritmalara kıyasla daha iyi çözümlere ulaşmada etkili olduğunu göstermektedir. Kümeleme algoritması parametrelerini ayarlama ve çözümün kalitesine nasıl katkıda bulunduklarını analiz etme gibi bazı muhtemel ileri araştırma alanları da vardır. TB ile Genetik Algoritma veya Tabu Arama gibi diğer iyi bilinen sezgisel yöntemler ve meta-sezgisellerin karşılaştırılması da mümkündür.

## KAYNAKLAR


[1] Dantzig G.B., Ramser J.G., The truck dispatching problem, Management Science, 6 (1), 80-91, 1959.

[2] Jaegere N.D., Defraeye M., Nieuwenhuyse I.V., The Vehicle Routing Problem: State of the Art Classification and Review, FEB Research Report KBI1415, Leuven, Belçika, 2014.

[3] Borcinova Z., Two models of the capacitated vehicle routing problem, Croatian Operational Research Review, 8 (2), 463-469, 2017.

[4] Ropke S., Heuristic and exact algorithms for vehicle routing problems, DTU library, Kopenhag, Danimarka, 2006.

[5] Lei, K., Zhu, X., Hou, J., and Huang, W. (2014). Decision of multimodal transportation scheme based on swarm intelligence. Mathematical Problems in Engineering, 2014, 1–10.

[6] Szeto, W.Y., Wu, Y., and Ho, S.C. (2011). An artificial bee colony algorithm for the capacitated vehicle routing problem. European Journal of Operational Research, 215(1), 126–135.

[7] Yu, S.P. and Li, Y.P. (2012). An improved ant colony opti-mization for vrp with time windows. Applied Mechanics and Materials, 263-266, 1609.

[8] Rego, C. (2006). Operations Research/Computer Science In-terfaces Series: Metaheuristic Optimization via Memory and Evolution : Tabu Search and Scatter Search. Springer US.

[9] Gracia, C., Diezma-Iglesias, B., and Barreiro, P. (2013). A hybrid genetic algorithm for route optimization in the bale collecting problem. Spanish Journal of Agricultural Research, 11(3), 603–614.

[10] Ghannadpour S.F., Noori S., T.-Moghaddam R., Ghoseiri K., A multi-objective dynamic vehicle routing problem with fuzzy time windows: Model, solution and application, Applied Soft Computing, 14, 504-527, 2014.

[11] Toth P., Vigo D., Models, Relaxations And Exact Approaches For The Capacitated Arc Routing Problem, Discrete Applied Mathematics, 123 (1-3), 487-512, 2002.

[12] Min H., Jayaraman V., Srivastava R., Combined location-routing prob- lems: A synthesis and future research directions, European Journal of Operational Research, 108 (1), 1-15, 1998.

[13] Balinski M.L., Quandt R.E., On an Integer Program for a Delivery Problem, Operations Research, 12 (2), 300-304, 1694.

[14] Ghiani G., Laporte G., Musmanno R., Introduction to Logistics Systems Management, John Wiley & Sons, Chichester, Birleşik Krallık, 2013.

[15] Lenstra J.K., Kan A.H.G.R., Some Simple Applications of the Travelling Salesman Problem, Operational Research Quarterly (1970-1977), 26 (4), 717, 1975.

[16] Christofides N., Mingozzi A., Toth P., Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations, Mathematical Programming, 20 (1), 255-282, 1981.

[17] Burrows C., Eilon S., Watson-Gandy C., Christofides N., Distribution management: Mathematical Modelling and Practical Analysis, Applied Statistics, 21, 337, 1972.

[18] Christofides N., Vehicle Routing. In Lawler E.L., Lenstra J.K., Rinnooy Kan A.H.G., Shmoys, D.B., editors, The Traveling Salesman Problem. A Guided Tour of Combinatorial Optimization, chapter Vehicle Routing, Wiley, Chichester, Birleşik Krallık, 431-448, 1985.

[19] Laporte G., The vehicle routing problem: An overview of exact and approximate algorithms, European Journal of Operational Research, 59 (3), 345-358, 1992.

[20] Clarke G., Wright J.W., Scheduling of Vehicles from a Central Depot to a Number of Delivery Points, Operations Research, 12 (4), 568-581, 1964.

[21] Gaskell T.J., Bases for Vehicle Fleet Scheduling, Operational Research Quarterly, 18 (3), 281-295, 1967.

[22] Yellow P.C., A Computational Modification to the Savings Method of Vehicle Scheduling, Operational Research Quarterly (1970-1977), 21 (2), 281, 1970.

[23] Paessens H., The savings algorithm for the vehicle routing problem, European Journal of Operational Research, 34 (3), 336-344, 1988.

[24] Wren A., Holliday A., Computer Scheduling of Vehicles from One or More Depots to a Number of Delivery Points, Journal of the Operational Research Society, 23 (3), 333-344, 1972.

[25] Gendreau M., Hertz A., Laporte G., A Tabu Search Heuristic for the Vehicle Routing Problem, Management Science, 40 (10), 1276-1290, 1994.

[26] El-Sherbeny N.A., Vehicle routing with time windows: An overview of exact, heuristic and metaheuristic methods, Journal of King Saud University - Science, 22 (3), 123-131, 2010.

[27] Swersey A.J., Ballard W, Scheduling School Buses, Management Science, 30 (7), 844-853, 1984.

[28] Baker E.K., Schaffer J.R., Solution Improvement Heuristics for the Vehicle Routing and Scheduling Problem with Time Window Constraints, American Journal of Mathematical and Management Sciences, 6 (3-4), 261-300, 1986.

[29] Solomon M.M., Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints, Operations Research, 35 (2), 166-324, 1987.

[30] Landeghem H., A bi-criteria heuristic for the vehicle routing problem with time windows, European Journal of Operational Research, 36 (2), 217-226, 1988.

[31] Potvin J.Y., Rousseau J.M., An exchange heuristic for routing problems with time windows, Journal of the Operational Research Society, 46 (12), 1433-1466, 1995.

[32] Chiang, W. C. and Russell, R. A. (1996). Simulated annealing metaheuristics for the vehicle routing problem with time windows. Annals of Operations Research.

[33] Taillard, E., Badeau, P., Gendreau, M., Guertin, F., and Potvin, J.-Y. (1997). A Tabu Search Heuristic for the Vehicle Routing Problem with Soft Time Windows. Transportation Science, 31(2):170–186.

[34] Thangiah, S. R. (1995). Vehicle Routing with Time Windows using Genetic Algorithms. In Chambers, L., editor, Application Handbook of Genetic Algorithms: New Frontiers, volume 2, pages 253–277. CRC Press.

[35] Li F., Golden B., Wasil E., The open vehicle routing problem: Algorithms, large-scale test problems, and computational results, Computers & Operations Research, 34 (10), 2918-2930, 2007.

[36] Liu C., Lai M.Y., The vehicle routing problem with uncertain demand at nodes, Transportation Research Part E Logistics and Transportation Review, 45 (4), 517-524, 2009.

[37] Erbao C., Mingyong L., A hybrid differential evolution algorithm to vehicle routing problem with fuzzy demands, Journal of Computational and Applied Mathematics, 231 (1), 302-310, 2009.

[38] Klose, A., 1995. Using clustering methods in problems of combined location and routing. Operations Research Proceedings. Springer, pp. 411–416.

[39] Yoshiike N., Takefuji Y., Solving vehicle routing problems by maximum neuron model, Advanced Engineering Informatics, 16 (2), 99-105, 2002.

[40] Zarandi, M. H. F., Davari, S., & Hemmati, A. 2010. Fuzzy Clustering for Initialization of Simulated Annealing Algorithm to Solve a Capacitated Vehicle Routing Problem. In Proceedings of the 2010 International Conference on Industrial Engineering and Operations Management Dhaka, Bangladesh.

[41] Barreto S., Ferreira C., Paixao J., Santos B.S., Using clustering analysis in a capacitated location-routing problem, European Journal of Operational Research, 179 (3), 968-977, 2007.

[42] Laporte, G., 2012. The Traveling Salesman Problem, the Vehicle Routing Problem, and their Impact on Combinatorial Optimization.. In: Canada: HEC Montréal,.

[43] Han, L., December 2016. Metaheuristic Algorthm for the vehicle routing problem with time window and skill set constraints., Halifax, Nova Scotia: Dalhousie University.

[44] Rardin, R. (1998). Optimization in Operations Research. Prentice Hall, Upper Saddle River, New Jersey.

[45] Lin, S., 1965. Computer solutions of the travelling salesman problem. Bell System Technical Journal 44, pp. 2245-2269.

[46] Renaud, J. B. F. F. &. L. G., 1996. A fast composite heuristic for the symmetric traveling salesman problem. INFORMS Journal on Computing,, pp. 134-143.

[47] Karel Jerabek., P. M. K. V., 2016. Application of Clark and Wright´s Savings Algorithm, s.l.: s.n.

[48] El-Sherbeny, N. A., 2009. Vehicle routing with time windows: An overview. Journal of King Saud University, 1 July.

[49] Tuyttens, D. T. J. E.-S. N., 2004. A particular multiobjective vehicle routing problem solved by simulated annealing.. Germany, Lecture Notes in Economics and Mathematical Systems, vol. 535. Springer-Vertag.

[50] Gold, T. C. a. H., 2008. Vehicle Routing Problem, Vienna, Austria: In-Teh.

[51] Uchoa, E. A. P. D. A. P. A. A. P. M. A. V. T. A. S. A., 2017. New Benchmark Instances for the Capacitated Vehicle Routing Problem. European Journal of Operational Research,, pp. 257(3), 845-858.

[52] Laporte, G., Gendreau, M., Potvin, J-Y., Semet, F. (2000). Classical and modern heuristics for the vehicle routing problem. International Transactions in Operational Research 7, 285-300.

[53] Bullnheimer, B., Hartl, R. F., and Strauss, C. (1999). An improved ant system algorithm for the vehicle routing problem. Annals of Operations Research, 89:319–328.

[54] Laporte, G. and Nobert, Y. (1987). Exact algorithms for the vehicle routing problem. In Martello, S., Laporte, G., Minoux, M., and Ribeiro, C., editors, Sureys in Combinatorial Optimization, volume 31 of Annals of Discrete Mathematics, chapter 5, pages 147–184. Elsevier Science (North-Holland), Amsterdam.

[55] Held, M. and R.M.Karp. (1971), „The traveling salesman problem and minimum spanning trees: Part II“, Mathematical Programming 1, 6-25.

[56] Christofides, N., A.Mingozzi (1989), „Vehicle routing: Practical and algorithmic aspects“, In Logistics: Where ends have to Meet, Oxford, UK, Pergamon Press, 30-48.

[57] Gelinas, S., Desrochers, M., Desrosiers, J. and Solomon, M. (1995), „A new branching strategy for time constrained routing problems with application to backhauling“, Annals of Operations Research 61, 91–109.

[58] Kontoravdis, G. and Bard, J., (1995) „A GRASP for the vehicle routing problem with time windows“, ORSA Journal on Computing 7(1), 10–23.

[59] Cheung, R.K. and Hang, D.D. (2003), „Multi-attribute label matching algorithms for vehicle routing problems with time windows and backhauls“, IIE Transactions 35(3), pp. 191–205.

[60] Savelsbergh, M.W.P. and Sol, M. (1995), „The general pickup and delivery problem“, Transp. Sci. 29, 17-29.

[61] Psaraftis, H. (1980), „A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem“, Transp. Sci., 130-154.

[62] Psaraftis, H (1983), „An exact algorithm for the single vehicle many-to-many dial-a-ride problem with time windows“. Transp. Sci. 17, 351-360.

[63] Desrosiers, J., Dumas, Y. and Soumis, F. (1986), „A dynamic programming solution of a large scale single vehicle dial a ride problem with time windows“. Am. J. Math. Manage. Sci. 6, 301-326.

[64] Psaraftis, H. (1983a), „K-interchange procedures for local search in a precedence-constrained routing problem“, Eur. J. Oper. Res. 13, 391-402.

[65] Sexton, T. and Bodin, L. (1985), „Optimizing single vehicle many-to-many operations with desired delivery times: II Routing“, Transp. Sci. 19, 378-435.

[66] Sexton, T. and Choi, Y. (1986), „Pick-up and delivery of partial loads with time windows“, Am. J. Math. Manage. Sci. 6, 369-398.

[67] Gendreau, M., Hertz, A. and Laporte, G. (1996), „The traveling salesman problem with backhauls“, Comput. Oper. Res. 23, 501-508.

[68] Van Breedam, A. (2001). Comparing descent heuristics and metaheuristics for the vehicle routing problem. Computers & Operations Research, 28(4):289–315.

[69] Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. Computers & Operations Research, 13(5):533–549.

[70] Braysy, O. and Gendreau, M. (2001). Tabu search heuristics for the vehicle routing problem with time windows. Report stf42 a01022, SINTEF Applied Mathematics, Research Council of Norway.

[71] Brucker, P. (2004). Scheduling algorithms. Springer-Verlag, Berlin, Germany, 4th edition.

[72] Kirkpatrick S., Gelatt C., Vecchi M., Optimization by simulated annealing, Science, 220 (4598), 671-680, 1983.

[73] Tan K.C., Lee L.H., Zhu Q.L., Ou K., Heuristic methods for vehicle routing problem with time windows, Artificial Intelligence in Engineering, 15 (3), 281-295, 2001.

[74] Alfa A.S., Heragu S.S., Chen M., A 3-opt ased simulated annealing algorithm for vehicle routing problems, Computers & Industrial Engineering, 21 (1-4), 635-639, 1991.

[75] Aarts E., Korst J., Simulated Annealing and Boltzmann Machines. Wiley-Interscience Series in Discrete Mathematics, John Wiley & Sons, Chichester, Birleşik Krallık, 1989.

[76] Robust´e F., Daganzo C.F., Souleyrette R.R.I., Implementing vehicle routing models, Transportation Research Part B, 24 (4), 263-286, 1990.

[77] Van Breedam A., Improvement heuristics for the vehicle routing problem based on simulated annealing, European Journal of Operational Research, 86 (5), 480-490, 1995.

[78] Van Breedam A., Comparing descent heuristics and metaheuristics for the vehicle routing problem, Computers & Operations Research, 28 (4), 289-315, 2001.

[79] Goldberg, D. E. (1989). Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading, Massachusetts.

[80] Filipec, M., Skrlec, D., and Krajcar, S. (1998). An efficient implementation of genetic algorithms for constrained vehicle routing problem. In Intelligent Systems for Humans in a Cyberworld, volume 3 of IEEE International Conference on Systems, Man, and Cybernetics, pages 2231–2236, Florida. Systems, Man, and Cybernetics Society of the Institute of Electrical and Electronic Engineers, Inc, IEEE.

[81] Vas, P. (1999). Artificial-intelligence-based electrical machines and drives: application of fuzzy, neural, fuzzy-neural, and genetic-algorithm-based techniques. Oxford University Press, Oxford.

[82] Skrlec, D., Filipec, M., and Krajcar, S. (1997). A heuristic modification of genetic algorithm used for solving the single depot capacitated vehicle routing problem. In Intelligent Information Systems ‘97, pages 184–188. IEEE.

[83] Thangiah, S. R., Nygard, K. E., and Juell, P. L. (1991). GIDEON: A genetic algorithm system for vehicle routing with time windows. In Seventh IEEE Conference on Artificial Intelligence for Applications, volume 1, pages 322–328. IEEE.

[84] Solomon, M. (1987). Algorithms for the vehicle routing and scheduling problems with time windows. Operations Research, 35(2):254–265.

[85] Dorigo, M. and Stutzle, T. (2002). Ant Colony Optimization. MIT Press, Cambridge.

[86] Middendorf, M., Reischle, F., and Schmeck, H. (2002). Multi colony ant algorithms. Journal of Heuristics, 8(3):305–320.

[87] Dorigo, M., Di Caro, G., and Gambardelle, L. M. (1999). Ant algorithms for discrete optimization. Artificial Life, 5(2):137–172.

[88] Maniezzo, V., Gambardella, L. M., and de Luigi, F. (2004). Ant colony optimization. In Onwubolu, G. C. and Babu, B. V., editors, New Optimization Techniques in Engineering, chapter 5, pages 101–117. Springer-Verlag, Berlin.

[89] Meuleau, N. and Dorigo, M. (2002). Ant colony optimization and stochastic gradient decent. Artificial Life, 8(2):103–121.

[90] Jain A.K., Murty M.N., Flynn P.J., Data clustering: a review, ACM Computing Surveys, 31 (3), 264-323, 1999.

[91] Everitt, B., Landau, S. and Leese, M., “Cluster Analysis,” London: Arnold, 2001.

[92] Berkhin, P., “Survey of clustering data mining techniques,” Technical Report, Accrue Software, 2002.

[93] Xu, R., and Wunsch, D., “Survey of clustering algorithms,” II, IEEE Transactions on Neural Networks, vol. 16, no. 3, pp. 645- 678, 2005.

[94] Wang, W., Yang, J. and Muntz, R., “STING: A Statistical Information Grid Approach to Spatial Data Mining,” In: Proc. of the 23rd Very Large Databases Conf, 1997.

[95] Sheikholeslami, G., Chatterjee, S. and Zhang, A., “WaveCluster: A Multi-Resolution Clustering Approach for Very Large Spatial Databases,” Proceedings of the 24rd International Conference on Very Large Data Bases, pp.428-439, 1998.

[96] Kohonen, T., “Automatic formation of topological maps of patterns in a self-organizing system,” Proceedings of 2SCIA, pp. 214-220, 1981a.

[97] Fisher, D.H., “Knowledge Acquisition via Incremental Conceptual Clustering,” Machine Learning, vol. 2, pp. 139-172, 1987.

[98] Cheeseman, P. and Stutz, J., “Bayesian Classification (AutoClass): Theory and Results,” Advances in Knowledge Discovery and Data Mining, Cambridge, 1996.

[99] Dunn J.C., A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters, Journal of Cybernetics, 3 (3), 32-57, 1973.

[100] Bezdek J.C., Pattern Recognition with Fuzzy Objective Function Algorithms, Plenum Press, New York, ABD, 1981.

[101] Mcqueen, J., “Some methods for classification and analysis of multivariate observations,” In Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, pp. 281–297, 1967.

[102] Pandey G., Kumar V., Steinbach M., Computational Approaches for Protein Function Prediction, Supported by the National Science Foundation under Grant Nos. IIS-0308264 and ITR-0325949, 2007.

# EKLER

## EK 1: Uygulamada Kullanılan Veriler

Tablo 7.1. Uygulamada kullanılan veriler.

| Sıra | Kod | İl Adı | Enlem | Boylam | Talep |
|---|---|---|---|---|---|
| 0 | 6 | Ankara | 39,920770 | 32,854110 | 0 |
| 1 | 1 | Adana | 37,000000 | 35,321333 | 12 |
| 2 | 2 | Adıyaman | 37,764751 | 38,278561 | 2 |
| 3 | 3 | Afyonkarahisar | 38,750714 | 30,556692 | 2 |
| 4 | 4 | Ağrı | 39,626922 | 43,021596 | 4 |
| 5 | 68 | Aksaray | 38,368690 | 34,036980 | 2 |
| 6 | 5 | Amasya | 40,649910 | 35,835320 | 2 |
| 7 | 7 | Antalya | 36,884140 | 30,705630 | 2 |
| 8 | 75 | Ardahan | 41,110481 | 42,702171 | 2 |
| 9 | 8 | Artvin | 41,182770 | 41,818292 | 2 |
| 10 | 9 | Aydın | 37,856041 | 27,841631 | 2 |
| 11 | 10 | Balıkesir | 39,648369 | 27,882610 | 2 |
| 12 | 74 | Bartın | 41,581051 | 32,460979 | 6 |
| 13 | 72 | Batman | 37,881168 | 41,135090 | 1 |
| 14 | 69 | Bayburt | 40,255169 | 40,224880 | 2 |
| 15 | 11 | Bilecik | 40,056656 | 30,066524 | 7 |
| 16 | 12 | Bingöl | 39,062635 | 40,769610 | 3 |
| 17 | 13 | Bitlis | 38,393799 | 42,123180 | 2 |
| 18 | 14 | Bolu | 40,575977 | 31,578809 | 2 |
| 19 | 15 | Burdur | 37,461267 | 30,066524 | 2 |
| 20 | 16 | Bursa | 40,266864 | 29,063448 | 10 |
| 21 | 17 | Çanakkale | 40,155312 | 26,414160 | 1 |
| 22 | 18 | Çankırı | 40,601343 | 33,613421 | 1 |
| 23 | 19 | Çorum | 40,550556 | 34,955556 | 5 |
| 24 | 20 | Denizli | 37,776520 | 29,086390 | 4 |
| 25 | 21 | Diyarbakır | 37,914410 | 40,230629 | 2 |
| 26 | 81 | Düzce | 40,843849 | 31,156540 | 2 |
| 27 | 22 | Edirne | 41,681808 | 26,562269 | 2 |
| 28 | 23 | Elazığ | 38,680969 | 39,226398 | 5 |
| 29 | 24 | Erzincan | 39,750000 | 39,500000 | 3 |
| 30 | 25 | Erzurum | 39,900000 | 41,270000 | 9 |

Tablo 7.1. (Devamı).

| Sıra | Kod | İl Adı | Enlem | Boylam | Talep |
|---|---|---|---|---|---|
| 31 | 26 | Eskişehir | 39,776667 | 30,520556 | 7 |
| 32 | 27 | Gaziantep | 37,066220 | 37,383320 | 5 |
| 33 | 28 | Giresun | 40,912811 | 38,389530 | 9 |
| 34 | 29 | Gümüşhane | 40,438588 | 39,508556 | 2 |
| 35 | 30 | Hakkâri | 37,583333 | 43,733333 | 2 |
| 36 | 31 | Hatay | 36,401849 | 36,349810 | 5 |
| 37 | 76 | Iğdır | 39,887984 | 44,004836 | 3 |
| 38 | 32 | Isparta | 37,764771 | 30,556561 | 3 |
| 39 | 34 | İstanbul | 41,005270 | 28,976960 | 12 |
| 40 | 35 | İzmir | 38,418850 | 27,128720 | 3 |
| 41 | 46 | Kahramanmaraş | 37,585831 | 36,937149 | 1 |
| 42 | 78 | Karabük | 41,206100 | 32,620350 | 3 |
| 43 | 70 | Karaman | 37,175930 | 33,228748 | 4 |
| 44 | 36 | Kars | 40,616667 | 43,100000 | 2 |
| 45 | 37 | Kastamonu | 41,388710 | 33,782730 | 3 |
| 46 | 38 | Kayseri | 38,731220 | 35,478729 | 6 |
| 47 | 71 | Kırıkkale | 39,846821 | 33,515251 | 3 |
| 48 | 39 | Kırklareli | 41,733333 | 27,216667 | 2 |
| 49 | 40 | Kırşehir | 39,142490 | 34,170910 | 5 |
| 50 | 79 | Kilis | 36,718399 | 37,121220 | 4 |
| 51 | 41 | Kocaeli | 40,853270 | 29,881520 | 1 |
| 52 | 42 | Konya | 37,866667 | 32,483333 | 4 |
| 53 | 43 | Kütahya | 39,416667 | 29,983333 | 5 |
| 54 | 44 | Malatya | 38,355190 | 38,309460 | 7 |
| 55 | 45 | Manisa | 38,619099 | 27,428921 | 8 |
| 56 | 47 | Mardin | 37,321163 | 40,724477 | 2 |
| 57 | 33 | Mersin | 36,800000 | 34,633333 | 2 |
| 58 | 48 | Muğla | 37,215278 | 28,363611 | 7 |
| 59 | 49 | Muş | 38,946189 | 41,753893 | 3 |
| 60 | 50 | Nevşehir | 38,693940 | 34,685651 | 9 |
| 61 | 51 | Niğde | 37,966667 | 34,683333 | 4 |
| 62 | 52 | Ordu | 40,983879 | 37,876411 | 5 |
| 63 | 80 | Osmaniye | 37,213026 | 36,176261 | 9 |
| 64 | 53 | Rize | 41,020050 | 40,523449 | 2 |
| 65 | 54 | Sakarya | 40,693997 | 30,435763 | 7 |
| 66 | 55 | Samsun | 41,292782 | 36,331280 | 5 |
| 67 | 56 | Siirt | 37,933333 | 41,950000 | 3 |
| 68 | 57 | Sinop | 42,023140 | 35,153069 | 3 |
| 69 | 58 | Sivas | 39,747662 | 37,017879 | 3 |
| 70 | 63 | Şanlıurfa | 37,159149 | 38,796909 | 2 |
| 71 | 73 | Şırnak | 37,418748 | 42,491834 | 1 |
| 72 | 59 | Tekirdağ | 40,983333 | 27,516667 | 3 |

Tablo 7.1. (Devamı).

| Sıra | Kod | İl Adı | Enlem | Boylam | Talep |
|---|---|---|---|---|---|
| 73 | 60 | Tokat | 40,316667 | 36,550000 | 4 |
| 74 | 61 | Trabzon | 41,001450 | 39,717800 | 2 |
| 75 | 62 | Tunceli | 39,307355 | 39,438778 | 3 |
| 76 | 64 | Uşak | 38,682301 | 29,408190 | 6 |
| 77 | 65 | Van | 38,489140 | 43,408890 | 3 |
| 78 | 77 | Yalova | 40,650000 | 29,266667 | 2 |
| 79 | 66 | Yozgat | 39,818081 | 34,814690 | 5 |
| 80 | 67 | Zonguldak | 41,456409 | 31,798731 | 4 |

## EK 2: Model_Olustur.m

```
function model=Model_Olustur(nokta_sayisi,kume_sayisi)
[sayi,metin,ham_veri]=xlsread('veri_seti.xlsx');
x=(sayi(2:end,5)).';
x0=sayi(1,5);
x_en_az=min(x)-1;
x_en_cok=max(x)+1;
y=(sayi(2:end,4)).';
y0=sayi(1,4);
y_en_az=min(y)-1;
y_en_cok=max(y)+1;
sehir_isim=(metin(2:end,3)).';
talep=(sayi(2:end,6)).';
arac_kapasitesi=(ones(kume_sayisi,1)*26).';
for uz1=1:length(x)
for uz2=1:length(y)
uzaklik_matrisi(uz1,uz2)=deg2km(distance([y(uz1) x(uz1)],[y(uz2)
x(uz2)]));
end
merkezden_uzaklik(uz1)=deg2km(distance([y0 x0],[y(uz1) x(uz1)]));
end
model.nokta_sayisi=nokta_sayisi;
model.kume_sayisi=kume_sayisi;
model.x=x;
model.x0=x0;
model.x_en_az=x_en_az;
model.x_en_cok=x_en_cok;
model.y=y;
model.y0=y0;
model.y_en_az=y_en_az;
model.y_en_cok=y_en_cok;
model.sehir_isim=sehir_isim;
model.talep=talep;
model.arac_kapasitesi=arac_kapasitesi;
model.uzaklik_matrisi=uzaklik_matrisi;
model.merkezden_uzaklik=merkezden_uzaklik;
end
```

**EK 3: Model_Olustur_Ve_Kaydet.m**

```
function Model_Olustur_Ve_Kaydet()
nokta_sayisi=80;
kume_sayisi=13;
model=Model_Olustur(nokta_sayisi,kume_sayisi);
model_ismi=['veri_seti(' num2str(model.nokta_sayisi) 'X'
num2str(model.kume_sayisi) ')'];
save(model_ismi,'model');
end
```

**EK 4: Veri_Seti_Sec.m**

```
function model=Veri_Seti_Sec()
dosya_filtresi={'*.mat','mat Dosyalari (*.mat)'
'*.*','Tum Dosyalar (*.*)'};
[dosya_ismi,dosya_yolu]=uigetfile(dosya_filtresi,'Veri Seti
Sec');
if dosya_ismi==0
model=[];
return;
end
tam_dosya_ismi=[dosya_yolu dosya_ismi];
veri_seti=load(tam_dosya_ismi);
model=veri_seti.model;
end
```

**EK 5: Baslangic_Cozumu_Olustur.m (Rassal Başlangıç)**

```
function baslangic_rotasi=Baslangic_Cozumu_Olustur(model)
nokta_sayisi=model.nokta_sayisi;
kume_sayisi=model.kume_sayisi;
baslangic_rotasi=randperm(nokta_sayisi+kume_sayisi-1);
en_iyi_cozum.Cozum=Sayisal_Cozum(baslangic_rotasi,model);
figure(1);
Gorsel_Cozum(en_iyi_cozum.Cozum,model);
end
```

**EK 6: Baslangic_Cozumu_Olustur.m (BCO Başlangıç)**

```
function baslangic_rotasi=Baslangic_Cozumu_Olustur(model)
nokta_sayisi=model.nokta_sayisi;
kume_sayisi=model.kume_sayisi;
x=model.x;
x0=model.x0;
x_en_az=model.x_en_az;
x_en_cok=model.x_en_cok;
y=model.y;
y0=model.y0;
y_en_az=model.y_en_az;
y_en_cok=model.y_en_cok;
sehir_isim=model.sehir_isim;
bco_veri=[x.' y.'];
[merkezler,U_matris]=fcm(bco_veri,kume_sayisi);
U_en_cok=max(U_matris);
for k=1:kume_sayisi
bas_kume{k}=find(U_matris(k,:)==U_en_cok);
end
renkler=hsv(kume_sayisi);
figure(1);
for k=1:kume_sayisi
X=[x0 x(bas_kume{k}) x0];
Y=[y0 y(bas_kume{k}) y0];
renk=renkler(k,:);
plot(X,Y,'- o',...
'Color',renk,...
'MarkerFaceColor',renk,...
'MarkerEdgeColor','black',...
'LineWidth',2,...
'MarkerSize',7);
hold on;
end
plot(x0,y0,'ks',...
'LineWidth',2,...
'MarkerSize',15,...
'MarkerEdgeColor','black',...
'MarkerFaceColor','yellow');
text([x0 x],[y0
y],sehir_isim,'FontWeight','bold','horizontalAlignment','center',
'verticalAlignment','top');
hold off;
xlabel('Boylam');
ylabel('Enlem');
grid on;
axis equal;
xlim([x_en_az x_en_cok]);
ylim([y_en_az y_en_cok]);
baslangic_rotasi=[bas_kume{1}];
for k=2:kume_sayisi
baslangic_rotasi=[baslangic_rotasi nokta_sayisi+k-1 bas_kume{k}];
    end
end
```

**EK 7: Sayisal_Cozum.m**

```
function cozum=Sayisal_Cozum(baslangic_rotasi,model)
nokta_sayisi=model.nokta_sayisi;
kume_sayisi=model.kume_sayisi;
talep=model.talep;
arac_kapasitesi=model.arac_kapasitesi;
uzaklik_matrisi=model.uzaklik_matrisi;
merkezden_uzaklik=model.merkezden_uzaklik;
pozisyon_silme=find(baslangic_rotasi>nokta_sayisi);
baslangic=[0 pozisyon_silme]+1;
bitis=[pozisyon_silme nokta_sayisi+kume_sayisi]-1;
rota_kumasi=cell(kume_sayisi,1);
rota_mesafesi=zeros(1,kume_sayisi);
kullanilan_kapasite=zeros(1,kume_sayisi);
for j=1:kume_sayisi
rota_kumasi{j}=baslangic_rotasi(baslangic(j):bitis(j));
if ~isempty(rota_kumasi{j})
rota_mesafesi(j)=merkezden_uzaklik(rota_kumasi{j}(1));
for k=1:numel(rota_kumasi{j})-1

rota_mesafesi(j)=rota_mesafesi(j)+uzaklik_matrisi(rota_kumasi{j}(
k),rota_kumasi{j}(k+1));
end

rota_mesafesi(j)=rota_mesafesi(j)+merkezden_uzaklik(rota_kumasi{j
}(end));
kullanilan_kapasite(j)=sum(talep(rota_kumasi{j}));
end
end
fazla_kapasite=max(kullanilan_kapasite./arac_kapasitesi-1,0);
toplam_fazla_kapasite=sum(max(kullanilan_kapasite-
arac_kapasitesi,0));
ortalama_fazla_kapasite=mean(fazla_kapasite);
cozum.rota_kumasi=rota_kumasi;
cozum.rota_mesafesi=rota_mesafesi;
cozum.en_az_rota_mesafesi=min(rota_mesafesi);
cozum.en_cok_rota_mesafesi=max(rota_mesafesi);
cozum.toplam_mesafe=sum(rota_mesafesi);
cozum.kullanilan_kapasite=kullanilan_kapasite;
cozum.fazla_kapasite=fazla_kapasite;
cozum.ortalama_fazla_kapasite=ortalama_fazla_kapasite;
cozum.toplam_fazla_kapasite=toplam_fazla_kapasite;
cozum.uygulanabilir_mi=(ortalama_fazla_kapasite==0);
end
```

**EK 8: Amac_Fonksiyonu.m**

```
function [z,cozum]=Amac_Fonksiyonu(baslangic_rotasi,model)
cozum=Sayisal_Cozum(baslangic_rotasi,model);
z1=cozum.toplam_mesafe;
z2=(z1*0.01)*cozum.toplam_fazla_kapasite;
z3=cozum.en_cok_rota_mesafesi;
z4=(cozum.en_cok_rota_mesafesi-cozum.en_az_rota_mesafesi);
z=z1+z2;
end
```

**EK 9: Komsu_Olustur.m**

```
function yeni_rota=Komsu_Olustur(baslangic_rotasi)
komsuluk_durumu=randi([1 3]);
switch komsuluk_durumu
case 1
yeni_rota=karsilikli_degisme(baslangic_rotasi);
case 2
yeni_rota=tersine_cevirme(baslangic_rotasi);
case 3
yeni_rota=ekleme_yapma(baslangic_rotasi);
end
end

function yeni_rota=karsilikli_degisme(baslangic_rotasi)
eleman_sayisi=numel(baslangic_rotasi);
sayi=randsample(eleman_sayisi,2);
sayi1=sayi(1);
sayi2=sayi(2);
yeni_rota=baslangic_rotasi;
yeni_rota([sayi1 sayi2])=baslangic_rotasi([sayi2 sayi1]);
end

function yeni_rota=tersine_cevirme(baslangic_rotasi)
eleman_sayisi=numel(baslangic_rotasi);
sayi=randsample(eleman_sayisi,2);
sayi1=min(sayi(1),sayi(2));
sayi2=max(sayi(1),sayi(2));
yeni_rota=baslangic_rotasi;
yeni_rota(sayi1:sayi2)=baslangic_rotasi(sayi2:-1:sayi1);
end

function yeni_rota=ekleme_yapma(baslangic_rotasi)
eleman_sayisi=numel(baslangic_rotasi);
sayi=randsample(eleman_sayisi,2);
sayi1=sayi(1);
sayi2=sayi(2);
if sayi1<sayi2
yeni_rota=[baslangic_rotasi(1:sayi1-1)
baslangic_rotasi(sayi1+1:sayi2) baslangic_rotasi(sayi1)
baslangic_rotasi(sayi2+1:end)];
else
yeni_rota=[baslangic_rotasi(1:sayi2) baslangic_rotasi(sayi1)
baslangic_rotasi(sayi2+1:sayi1-1) baslangic_rotasi(sayi1+1:end)];
end
end
```

**EK 10: Gorsel_Cozum.m**

```
function Gorsel_Cozum(cozum,model)
kume_sayisi=model.kume_sayisi;
x=model.x;
x0=model.x0;
x_en_az=model.x_en_az;
x_en_cok=model.x_en_cok;
y=model.y;
y0=model.y0;
y_en_az=model.y_en_az;
y_en_cok=model.y_en_cok;
sehir_isim=model.sehir_isim;
rota_kumasi=cozum.rota_kumasi;
renkler=hsv(kume_sayisi);
for j=1:kume_sayisi
if isempty(rota_kumasi{j})
continue;
end
X=[x0 x(rota_kumasi{j}) x0];
Y=[y0 y(rota_kumasi{j}) y0];
renk=renkler(j,:);
plot(X,Y,'- o',...
'Color',renk,...
'MarkerFaceColor',renk,...
'MarkerEdgeColor','black',...
'LineWidth',2,...
'MarkerSize',7);
hold on;
end
plot(x0,y0,'ks',...
'LineWidth',2,...
'MarkerSize',15,...
'MarkerEdgeColor','black',...
'MarkerFaceColor','yellow');
hold off;
xlabel('Boylam');
ylabel('Enlem');
grid on;
axis equal;
xlim([x_en_az x_en_cok]);
ylim([y_en_az y_en_cok]);
end
```

**EK 11: Tavlama_Benzetimi.m**

```
close all;
clear;
clc;
model=Veri_Seti_Sec();
tic;
maliyet_fonksiyonu=@(baslangic_rotasi)
Amac_Fonksiyonu(baslangic_rotasi,model);
iterasyon_sayisi_1=250;
iterasyon_sayisi_2=500;
ilk_sicaklik=10;
sicaklik_sogurma_orani=0.97;
noktalar.Pozisyon=Baslangic_Cozumu_Olustur(model);
[noktalar.Maliyet,noktalar.Cozum]=maliyet_fonksiyonu(noktalar.Poz
isyon);
en_iyi_cozum=noktalar;
disp(['Iterasyon 0 -- En Iyi Maliyet = '
num2str(en_iyi_cozum.Cozum.toplam_mesafe)]);
en_iyi_maliyet=zeros(iterasyon_sayisi_1,1);
sicaklik=ilk_sicaklik;
for iterasyon1=1:iterasyon_sayisi_1
for iterasyon2=1:iterasyon_sayisi_2
noktalar_yeni.Pozisyon=Komsu_Olustur(noktalar.Pozisyon);

[noktalar_yeni.Maliyet,noktalar_yeni.Cozum]=maliyet_fonksiyonu(no
ktalar_yeni.Pozisyon);
if noktalar_yeni.Maliyet<=noktalar.Maliyet
noktalar=noktalar_yeni;
else
delta(iterasyon1*iterasyon2)=noktalar_yeni.Maliyet-
noktalar.Maliyet;
p(iterasyon1*iterasyon2)=exp(-
delta(iterasyon1*iterasyon2)/sicaklik(iterasyon1));
if rand<=p(end)
noktalar=noktalar_yeni;
end
end
if noktalar.Maliyet<=en_iyi_cozum.Maliyet
en_iyi_cozum=noktalar;
end
end
en_iyi_maliyet(iterasyon1)=en_iyi_cozum.Maliyet;
if en_iyi_cozum.Cozum.uygulanabilir_mi
isaret=' *';
else
isaret='';
end
disp(['Iterasyon ' num2str(iterasyon1) ' -- En Iyi Maliyet = '
num2str(en_iyi_maliyet(iterasyon1)) isaret]);
sicaklik(iterasyon1+1)=sicaklik_sogurma_orani*sicaklik(iterasyon1
);
figure(2);
Gorsel_Cozum(en_iyi_cozum.Cozum,model);
```

```
pause(0);
end
text([model.x0 model.x],[model.y0
model.y],model.sehir_isim,'FontWeight','bold','horizontalAlignmen
t','center','verticalAlignment','top');
figure(3);
plot(en_iyi_maliyet,'r','LineWidth',2);
xlabel('Iterasyon Sayisi');
ylabel('En Iyi Maliyet');
figure(4);
plot(delta,'LineWidth',2);
xlabel('Iterasyon Sayisi');
ylabel('Delta');
figure(5);
plot(p,'LineWidth',2);
xlabel('Iterasyon Sayisi');
ylabel('p');
figure(6);
plot(sicaklik,'LineWidth',2);
xlabel('Iterasyon Sayisi');
ylabel('Sicaklik');
grid on;
```

**EK 12: Calistir.m**

```
Tavlama_Benzetimi;
sure=toc;
disp(['Islem Suresi = ' num2str(sure) ' saniye']);
```

# ÖZGEÇMİŞ

Ahmet Fatih Eker, 14 Temmuz 1994'te Konya'da doğdu. İlk, orta ve lise eğitimini Bursa'da tamamladı. 2012 yılında Ali Osman Sönmez Anadolu Teknik Lisesinin Tekstil Teknolojisi Bölümünden mezun oldu. 2012 yılında başladığı Sakarya Üniversitesi Endüstri Mühendisliği Bölümünden 2016 yılında mezun oldu. 2017 yılında Sakarya Üniversitesi Endüstri Mühendisliği Bölümünde yüksek lisans eğitimine başladı. Hâlâ Sakarya Üniversitesi Endüstri Mühendisliği Bölümünde öğrenci olarak öğrenim hayatına devam etmektedir.