

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

ARAÇ ROTALAMA İÇİN KÜMELEME YAKLAŞIMI

YÜKSEK LİSANS TEZİ

Çağla Ebru BİLİR

Endüstri Mühendisliği Anabilim Dalı

Endüstri Mühendisliği Bilim Dalı

ARALIK 2024

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

ARAÇ ROTALAMA İÇİN KÜMELEME YAKLAŞIMI

YÜKSEK LİSANS TEZİ

Çağla Ebru BİLİR

Endüstri Mühendisliği Anabilim Dalı

Endüstri Mühendisliği Bilim Dalı

Tez Danışmanı: Prof. Dr. Özer UYGUN

ARALIK 2024

Çağla Ebru BİLİR tarafından hazırlanan “Araç Rotalama için Kümeleme Yaklaşımı” adlı tez çalışması 25.12.2024 tarihinde aşağıdaki jüri tarafından oy birliği/oy çokluğu ile Sakarya Üniversitesi Fen Bilimleri Enstitüsü Endüstri Mühendisliği Anabilim Dalı Endüstri Mühendisliği Bilim Dalı’nda Yüksek Lisans tezi olarak kabul edilmiştir.

Tez Jürisi

Jüri Başkanı:	Prof. Dr. Özer UYGUN Sakarya Üniversitesi	
Jüri Üyesi:	Doç. Dr. Seher ARSLANKAYA Sakarya Üniversitesi	
Jüri Üyesi:	Dr. Öğr. Üyesi Yasin KIRELLİ Kütahya Dumlupınar Üniversitesi	



ETİK İLKE VE KURALLARA UYGUNLUK BEYANNAMESİ

Sakarya Üniversitesi Fen Bilimleri Enstitüsü Lisansüstü Eğitim-Öğretim Yönetmeliğine ve Yükseköğretim Kurumları Bilimsel Araştırma ve Yayın Etiği Yönergesine uygun olarak hazırlamış olduğum “Araç Rotalama için Kümeleme Yaklaşımı” başlıklı tezin bana ait, özgün bir çalışma olduğunu; çalışmamın tüm aşamalarında yukarıda belirtilen yönetmelik ve yönergeye uygun davrandığımı, tezin içerdiği yenilik ve sonuçları başka bir yerden almadığımı, tezde kullandığım eserleri usulüne göre kaynak olarak gösterdiğimi, bu tezi başka bir bilim kuruluna akademik amaç ve unvan almak amacıyla vermediğimi ve 20.04.2016 tarihli Resmi Gazete’de yayımlanan Lisansüstü Eğitim ve Öğretim Yönetmeliğinin 9/2 ve 22/2 maddeleri gereğince Sakarya Üniversitesi’nin abonesi olduğu intihal yazılım programı kullanılarak Enstitü tarafından belirlenmiş ölçütlere uygun rapor alındığını, çalışmamla ilgili yaptığım bu beyana aykırı bir durumun ortaya çıkması halinde doğabilecek her türlü hukuki sorumluluğu kabul ettiğimi beyan ederim.

(25/12/2024).

(imza)

Çağla Ebru BİLİR





Eşime ve aileme...



TEŞEKKÜR

Tez çalışmamda beni destekleyen ve yönlendiren değerli danışman hocam Prof. Dr. Özer UYGUN'a teşekkür ederim. Hayatım boyunca hep yanımda olan en büyük motivasyon kaynağım aileme sonsuz sevgi ve teşekkürlerimi sunarım.

Çağla Ebru BİLİR



İÇİNDEKİLER

Sayfa

ETİK İLKE VE KURALLARA UYGUNLUK BEYANNAMESİ	v
TEŞEKKÜR	ix
İÇİNDEKİLER	xi
KISALTMALAR	xiii
SİMGELER	xv
TABLO LİSTESİ	xvii
ŞEKİL LİSTESİ	xix
ÖZET	xxi
SUMMARY	xxiii
1. GİRİŞ	1
1.1. Tezin Kapsamı	1
1.2. Tezin Amacı	1
1.3. Literatür Taraması	2
1.3.1. Kümeleme	2
1.3.1.1. K-means kümeleme algoritması	3
1.3.1.2. Bulanık c-means kümeleme algoritması	5
1.3.2. Araç rotalama problemi	6
1.3.2.1. Araç rotalama probleminin matematiksel modellenmesi	7
1.3.2.2. Yapay arı kolonisi algoritması	8
1.3.2.3. Ateş böceği algoritması	10
2. UYGULAMA	11
2.1. Giriş	11
2.2. K-means ile Kümeleme ve Yapay Arı Kolonisi ile Rotalama	13
2.2.1. K-means kümeleme sonuçları	14
2.2.2. Yapay arı kolonisi algoritması ile araçların rotasyonu	15
2.3. K-means ile Kümeleme ve Ateş Böceği Algoritması ile Rotalama	15
2.3.1. K-means ile kümeleme sonuçları	16
2.3.2. Ateş böceği algoritması ile araçların rotasyonu	16
2.4. Bulanık c-means ile Kümeleme ve Yapay Arı Kolonisi ile Rotalama	17
2.4.1. Bulanık c-means algoritması ile kümeleme	17
2.4.2. Yapay arı kolonisi algoritması ile araçların rotasyonu	18
2.5. Bulanık c-means ile Kümeleme ve Ateş Böceği Algoritması ile Rotalama ..	19
2.5.1. Bulanık c-means ile kümeleme sonuçları	20
2.5.2. Ateş böceği algoritması ile araçların rotasyonu	20
3. SONUÇ VE DEĞERLENDİRME	23
KAYNAKLAR	27
ÖZGEÇMİŞ	79



KISALTMALAR

ABC	: Yapay Arı Kolonisi (Artificial Bee Colony)
AGV	: Automated Guided Vehicle
ARP	: Araç Rotalama Problemi
DEA	: Data Envelopment Analysis
FA	: Ateş Böceği Algoritması (Firefly Algorithm)
GA	: Genetik Algoritmalar
GEMS	: Grounding-Enterprise-Market-Surrounding



SİMGELER

A	: Araç kümesi
B	: Bayi kümesi
D	: Alt tur engelleme karar değişkeni
K	: Kapasite
M_{ij}	: i.bayi ile j.bayi arasındaki mesafe
N	: Düğüm kümesi
S_{iA}	: A aracına atanan i.bayinin tüm siparişleri
Y	: Ürün taban yüzölçümü



TABLO LİSTESİ

	<u>Sayfa</u>
Tablo 1.1. Kümeleme çalışmalarında literatürde kullanılan metotlar.	2
Tablo 1.2. Araç Rotalama Problemleri örnekleri.	6
Tablo 1.3. Yapay Arı Kolonisi	9
Tablo 1.4. Ateş Böceği Algoritması	10
Tablo 3.1. Çözümlerin karşılaştırılması	24



ŞEKİL LİSTESİ

	<u>Sayfa</u>
Şekil 2.1. Uygulama aşamaları.	12
Şekil 2.2. Araç taban ölçüleri.....	13
Şekil 2.3. Elbow Yöntemi kullanılarak optimal küme sayısı	14





ARAÇ ROTALAMA İÇİN KÜMELEME YAKLAŞIMI

ÖZET

Günümüzde işletmelerin en önemli amaçlarından biri ürünlerin müşterilere zamanında teslim edilebilmesidir. Ürünlerin müşteriye teslim sürecinde dağıtım yapacak araç sayısı, müşteriler ve dağıtım merkezleri arası mesafeler, teslimat zamanlaması, doğru ürünün doğru miktarda teslimi, ürünlerin araç içi yerleştirilmeleri, vb. çok fazla kriter dikkate alınmaktadır. Araç Rotalama Problemleri tüm bu kriterleri dikkate alarak hem müşteri memnuniyetini sağlanmasında hem de sürecin maliyetlerinin minimize edilmesinde kullanılmaktadır. Bu nedenle araç rotalama problemleri hem literatürde hem de uygulamada oldukça sıklıkla çalışılmaktadır.

Rotalama yapılırken kapasite, mesafe, depo sayısı vb kısıtlara bağlı olarak kapasite kısıtlı, mesafe kısıtlı, çok depolu önce dağıt sonra topla, eşzamanlı ve karışık topla-dağıt, zaman pencereli vb farklı teknikler kullanılmaktadır. Bu tekniklerin ortak amaçları: sevkiyatta maliyetlerin azaltılması, en az sefer sayısı ile en fazla teslimatı yapacak rotanın belirlenmesi ve bunları sağlarken teslimat ve hizmet kalitesinin artırılmasıdır. Rotalamada faydalanan yaklaşımlardan biri dağıtım yapılacak merkezleri gruplandırmak için kullanılan kümeleme yaklaşımıdır. K-means Kümeleme Algoritması, Hiyerarşik Kümeleme Algoritması, Model Tabanlı Kümeleme metotları, Grid Temelli metotlar, Yoğunluk Temelli metotlar vb. kümeleme algoritmaları kullanılabilir.

Bu çalışmada beyaz eşya sektöründe faaliyet gösteren bir imalat işletmesinin nihai ürünlerinin bayilere doğru ve verimli şekilde sevkiyatı için araç rotalama uygulaması anlatılmaktadır. Problemin çözümünde teslimatların yapılacağı bayilerin rotalanmasında “önce kümele sonra rotala” yaklaşımı kullanılmıştır.

Öncelikle bayi konumları, sipariş hacimleri ve bilgileri gibi veriler elde edilmiştir. Bayilerin ana depoya olan uzaklıkları dikkate alınarak Google Maps uygulaması aracılığı ile mesafe matrisi oluşturulmuştur. Önce kümele sonra rotala yaklaşımına dayanarak bayilerin kümelemesinde K-means ve Bulanık c-means olmak üzere iki farklı kümeleme algoritması kullanılmıştır. Her iki algoritma ile belirlenen kümeler üzerinden Yapay Arı Kolonisi Algoritması ve Ateş Böceği Algoritması olmak üzere iki farklı sezgisel algoritma ile rotalama yapılmıştır. Sonuç olarak K-means ve Yapay Arı Kolonisi, K-means ve Ateş Böceği, Bulanık c-means ve Yapay Arı Kolonisi, Bulanık c-means ve Ateş Böceği eşleştirmeleri ile dört farklı rota elde edilmiştir. Hem kümeleme hem de rotalama için kullanılan tüm algoritmalara ait kodlar Phyton programında çalıştırılmıştır. Bu rotalar: araç sayısı, kat edilen toplam mesafe, araç kapasitelerinin dolulukları açısından kıyaslanmıştır. Çözümler arasından en uygun yöntem belirlenmiştir.



CLUSTERING APPROACH FOR VEHICLE ROUTING

SUMMARY

One of the primary objectives of businesses today is to ensure the timely delivery of products to customers. In the product delivery process, numerous criteria must be considered, such as the number of vehicles available for distribution, distances between customers and distribution centers, delivery timing, accurate delivery of the correct quantity of products, and the internal arrangement of products within vehicles. Vehicle Routing Problems (VRPs) are utilized to address these criteria, aiming to enhance customer satisfaction while minimizing operational costs. Consequently, VRPs are widely studied both in the academic literature and in practice.

Vehicle Routing Problems are the problems of determining the optimum number of distribution/collection routes for a given number of vehicles to distribute/collect products from at least one main warehouse (or more than one warehouse) to a set of customers or dealers who are scattered in different physical locations and have known demand. In solving vehicle routing problems, the following objectives are taken into consideration: minimizing total transportation costs by taking into account the total travel time of the vehicles and their fixed costs, minimizing the number of vehicles to be used by ensuring that no customer/dealer is left without receiving products/services, balancing the determined routes in terms of vehicle load and distance traveled, and preventing partial distribution of orders to customers/dealers. Depending on the nature of the problem, it may sometimes be necessary to create a model to achieve more than one and/or conflicting objectives. Vehicle Routing Problems, for which many different exact methods and many general heuristic methods can be used, are integrated optimization problems that have remained popular for many years. Many different types of Vehicle Routing Problems, which have been studied frequently both in literature and in practical applications since the 1950s, have been introduced by adding new constraints, and many mixed models and algorithms have been developed for these problem types.

The main objective in Vehicle Routing Problems is to provide all the desired constraints depending on the structure of the problem, keep the number of vehicles to a minimum, keep the total travel time and/or distance to a minimum and at the same time minimize the cost function. While all these basic objectives are provided in Vehicle Routing Problems, the secondary objective that needs to be provided is to keep customer satisfaction at the maximum level. In order to achieve all these goals, businesses produce solutions based on their past experiences without using mathematical models, which results in high distribution costs (such as labor, vehicles, time), and ineffective/inefficient distribution routes can create extra costs for businesses. In addition, inefficient vehicle routes can affect delivery time and delivery accuracy, thus reducing customer satisfaction.

During the routing process, various techniques are employed depending on constraints such as vehicle capacity, distance, and the number of depots. These techniques include capacitated, distance-constrained, multi-depot, delivery-then-pickup, simultaneous

pickup and delivery, and time-windowed routing methods. The common objectives of these techniques are to reduce distribution costs, determine routes that maximize deliveries with the fewest trips, and enhance delivery and service quality.

One of the approaches employed in routing is clustering, used to group delivery points effectively. Clustering algorithms such as K-means Clustering Algorithm, Hierarchical Clustering Algorithm, Model-Based Clustering methods, Grid-Based methods, and Density-Based methods are among the techniques that can be utilized.

In this study, a vehicle routing application is presented for a manufacturing company operating in the white goods sector, aiming to ensure the accurate and efficient shipment of final products to its dealers. The “cluster first, route second” approach was adopted for solving the routing of dealers. Initially, data such as dealer locations, order volumes, and order details were collected. A distance matrix was then created using Google Maps, taking into account the distances between the dealers and the main depot. Based on the “cluster first, route second” approach, two different clustering algorithms — K-means and Fuzzy C-means — were applied to cluster the dealers.

Subsequently, two different metaheuristic algorithms — Artificial Bee Colony (ABC) Algorithm and Firefly Algorithm — were employed to perform routing based on the clusters formed by each clustering method. As a result, four different routing solutions were obtained by pairing the clustering and routing algorithms: (1) K-means & ABC, (2) K-means & Firefly, (3) Fuzzy C-means & ABC, and (4) Fuzzy C-means & Firefly. All algorithms were coded and executed using Python.

The resulting routes were compared in terms of the number of vehicles used, total distance traveled, and vehicle load capacities.

When clustering was performed using the K-means algorithm, the optimum number of clusters was determined as four based on the Elbow Method. Since Cluster 2 contained a volume of products exceeding the capacity of a single vehicle, five vehicles were assigned to Cluster 2. Consequently, a total of eight vehicles were used in each routing solution based on K-means clustering. Therefore, when comparing the two routing solutions derived from K-means, the primary comparison criterion is the total distance traveled.

When clustering was performed using the Fuzzy C-means algorithm, the optimum number of clusters was found to be ten. In the routing solution using the Firefly Algorithm, Clusters 2 and 3 exceeded the capacity of a single vehicle, thus requiring two vehicles each. Therefore, a total of twelve vehicles were used in the routing solution combining Fuzzy C-means clustering and the Firefly Algorithm. Thus, when comparing the two routing solutions derived from Fuzzy C-means, both the number of vehicles and the total distance traveled must be considered.

Upon examining the results of all solutions, it was observed that the combination of K-means clustering and Firefly routing provided the optimum solution, achieving both the minimum number of vehicles (eight) and the shortest total distance traveled (1886.98 km).

In the current distribution practice of the white goods company under study, product deliveries to dealers are planned based on the sales personnel’s knowledge and experience. Since clustering is not applied during distribution planning, orders from geographically distant dealers may be loaded onto the same vehicles. Furthermore, due to the absence of a systematic routing method, the number of vehicles used and the total distances traveled are substantially higher.

The proposed solution, utilizing K-means clustering and Firefly routing, achieved significant improvements in both the number of vehicles required and the total distance traveled.





1. GİRİŞ

Yeni dünya düzeninde yoğun rekabet içerisindeki şirketler, müşterilerin artan beklentilerini karşılayabilmek adına hızlı ve yenilikçi çözümler üretmek ve bu çözümleri sürdürülebilir kılmak durumundadırlar. Rekabetin yoğun olmasından dolayı müşteri sadakatini ve bağlılığını sağlamak her geçen gün daha da zorlaşmaktadır. Teknoloji ve taşıma sistemlerindeki son gelişmelerin bir sonucu olarak üreticiler, müşterilerinin taleplerine hızlı bir şekilde cevap verebilmelidir. Bunun içinde müşteri talebinin doğru şekilde belirlenip üretimin istenilen kalitede ve düşük maliyetle yapılmasının yanı sıra aynı zamanda da dağıtım planının da doğru şekilde yapılması gerekmektedir. Doğru dağıtım planlamasının, dağıtım maliyetlerini düşüren ve dağıtım araçlarının aldıkları yolu ve dağıtım süresini azaltan bir plan olması önemlidir. Bu konuları kapsayan problemlerin çözümünde Araç Rotalama teknikleri kullanılmaktadır.

1.1. Tezin Kapsamı

Bu çalışmada beyaz eşya üretim işletmesinin bayilerine yaptığı ürün dağıtımının optimizasyonu için kümeleme ve araç rotalama teknikleri kullanılacaktır. 59 bayinin bulunduğu bölgede bir tane ana depo bulunmaktadır. Dağıtım ana depodan yapılacaktır. Öncelikle bayiler arasındaki mesafelerin hesaplanabilmesi mesafe matrisi hesaplanmıştır. Belirlenen mesafeleri dikkate alarak k-means, bulanık c-means yöntemleri ile kümeleme yapılmış ve rotalama öncesi kümeler belirlenmiştir. Kümeleme işleminden sonra araç rotalama için sezgisel yöntemler kullanılmıştır. Yapay Arı Kolonisi algoritması ve Ateş Böceği algoritması yöntemleri ile iki farklı rotalama yapılmıştır. Ve sonuç bölümünde kullanılan yöntemlerin karşılaştırılması yapılmıştır.

1.2. Tezin Amacı

Çalışma beyaz eşya üretilen bir işletmenin dağıtım problemini çözmek amacı ile yapılmıştır. İşletmenin Ankara ilinde bulunan ana deposundan, civar bölgelerde bulunan teslimat noktalarına siparişlerin dağıtımı yapılırken mevcut durumda birtakım

problemler yaşanmaktadır. Problemler, mevcut dağıtım planının lojistik personelinin deneyimine dayanılarak yapılmasından da kaynaklanmaktadır. Bu nedenle bu çalışmada dağıtım planında yaşanan sıkıntıların azaltılması için sezgisel yöntemleri de kapsayan bilimsel yöntemlerle dağıtım modeli kurulması amaçlanmaktadır

1.3. Literatür Taraması

1.3.1. Kümeleme

Kümeleme analizi; tanımlanmamış verilerin benzer özelliklere veya niteliklere sahip olanları aynı kümeye ait olacak şekilde anlamlı halde gruplandırılmasına olanak sağlamaktadır. Geleneksel kümeleme yöntemleri genel olarak beş kategoriye ayrılabilir: hiyerarşik, bölümsel, yoğunluk tabanlı, ızgara tabanlı ve model tabanlı kümeleme yöntemleri (Chen ve ark, 2024).

Veri kümeleme, küme içindeki benzerliği en üst düzeye çıkarmak ve kümeler arası benzerliği ise en aza indirmek için örnekleri yoğun gruplara ayırmayı amaçlamaktadır (Oskouei ve ark, 2024).

Kümelemenin temel varsayımlarından birisi: verilerin farklı özelliklerin bir koleksiyonu olarak kabul edilmesidir. Çoğu kümeleme yöntemi, farklı veriler arasındaki bireysel özelliklerin benzerlik derecesini hesaplayarak kümeleme işlemini gerçekleştirir (Chen ve ark, 2024).

Veri kümeleme verilerin dağıtımına dair içgörü sağlar ve ayrıca öneri sistemi ve ayrıca öneri sistemleri, Pazar bölümlenme, sosyal ağ analizi, görüntü işleme, eğitim, yer bilimleri (coğrafya, jeoloji, vb), psikoloji ve biyolojik veri analizi gibi çok çeşitli bilimsel uygulamalarda kullanılmaktadır (Xu ve Wunsch, 2005). Kümeleme çalışmaları ile ilgili literatür örnekleri Tablo 1.1’de verilmektedir.

Tablo 1.1. Kümeleme çalışmalarında literatürde kullanılan metotlar.

Referans	Çalışma alanı	Sene	Metot
Hosseinzadeh ve ark. (2024)	İnsansız hava aracı	2024	Balina Optimizasyon Algoritması
Oskouei ve ark. (2024)	---	2024	Bulanık c-means
Wang ve ark. (2024)	Görüntü işleme	2024	Bulanık c-means
Heidari ve ark. (2024)	Görüntü işleme	2024	K-means ve K-medoid
Yu ve ark. (2024)	Enerji	2024	DEA ve GEMS
Chen ve ark. (2024)	Deneyisel veri	2024	K-means ve yapay sinek kuşu algoritması
Meng ve ark. (2024)	Eğitim	2024	K-means
Wu ve ark. (2024)	Deneyisel veri	2024	K-means

Tablo 1.1.(Devamı) Kümeleme çalışmalarında literatürde kullanılan metotlar.

Paul ve ark. (2024)	Görüntü sıkılaştırma	2024	K-means
Xiang (2024)	Sağlık	2024	K-means
Raj ve Palaniappan (2024)	Biyomedikal	2024	K-means
Dugan ve ark. (2024)	Sağlık	2024	K-means
Zhang ve ark. (2024)	Görüntü işleme	2024	K-means
Lee (2024)	Uzay çalışmaları	2024	K-means
Burada ve ark. (2024)	Sağlık-deri kanseri erken teşhis	2024	Bulanık c-means

Standart kümeleme süreci aşağıdaki adımlardan oluşmaktadır (Xu ve Tian, 2015):

- Özellik çıkarma ve seçme: orijinal veri setinden temsili özellikleri çıkarın ve seçin,
- Kümeleme algoritması tasarımı: kümeleme algoritmasını problemin özelliklerine göre tasarlayın,
- Sonuç değerlendirmesi: kümeleme sonucunu değerlendirin ve algoritmanın geçerliliğini sorgulayın,
- Sonuç açıklaması: kümeleme sonucunu pratik bir şekilde açıklayın.

Genellikle farklı başlangıç noktaları ve kriterler farklı sınıflandırmalara yol açabilir. Yaygın olarak kabul edilen bir çerçeve, oluşturulan kümelerin özelliklerine dayalı olarak klasik kümeleme tekniklerinin hiyerarşik kümeleme, bölümlü kümeleme, bulanık teoriye dayalı kümeleme, dağıtım dayalı kümeleme, yoğunluğa dayalı kümeleme, grafik teorisine dayalı kümeleme, ızgaraya dayalı kümeleme, fraktal teoriye dayalı kümeleme, modele dayalı kümeleme olarak ayrıldığıdır (Xu ve Tian, 2015).

Hiyerarşik kümeleme, veri nesnelerini tekil kümelerden tüm bireyleri içeren bir kümeye veya tam tersi şekilde bir dizi bölüme gruplandırırken, bölümsel kümeleme, veri nesnelerini hiyerarşik yapı olmadan doğrudan önceden belirlenmiş sayıda kümeye bölmektedir (Xu ve Wunsch, 2005).

1.3.1.1. K-means kümeleme algoritması

K-means tipi kümeleme yöntemleri, etiketlenmemiş verilerden türetilen benzer girdi çiftlerine göre veri dağılımını öğrenmeyi amaçlamaktadır (Yang ve ark, 2024). K-means kümeleme algoritması, süreçte oluşturulması gereken önceden tanımlanmış kümelerin sayısını ve gruplar arasındaki varyasyonu en üst düzeye çıkarmak ve gruplar

içindeki varyasyonu en aza indirmek için, kümenin merkez konumunu tanımlayan optimum K değerini bulmaya çalışmaktadır (Sadeghi ve ark, 2024). K-ortalama kümeleme sürecini uygulamadan önce k'yi önceden ayarlamak çok önemlidir (Zhou ve Sun, 2023). K değerini bulmada Elbow (Dirsek) yöntemi kullanılabilir.

N tane etiketlenmemiş veriyi temsil eden D kümesi şu şekilde gösterilir: $D = [X_1, X_2, X_3, \dots, X_N]$. Küme içine atama işleminde her veri için Öklid mesafesi belirlenir ve en yakın küme mesafesine atanır. Küme merkezleri daha sonra yeni atamalara göre güncellenir. Küme üyeliği sabitlenene veya maksimum yinleme sayısına ulaşana kadar atama ve güncelleme işlemleri tekrar eder (Zhang ve ark, 2024).

K -means algoritmasının ana fikri, aynı kümedeki veriler arasındaki mesafeyi mümkün olduğu kadar küçük, farklı kümelerdeki veriler arasındaki mesafeyi ise mümkün olduğu kadar büyük olacak şekilde verileri k kümeye bölmektir. K- means'in adımları aşağıdaki gibidir (Li ve ark, 2024):

Adım 1: Veri ön işleme. Veri setini normalleştirin. D ile $[-1,1]^d$ ve n veri noktası elde edin.

$$p_1, p_2, \dots, p_n .$$

Adım 2: Rastgele başlatma. Başlangıç merkezleri olarak n veri noktasından rastgele veri noktaları seçin.

$$c_1, c_2, \dots, c_n .$$

Adım 3: Veri noktalarını atayın. Ve her veri noktası ve merkez noktası arasındaki Öklid mesafesini hesaplayın, veri noktalarını en yakın kümelere bölün.

Küme şu şekilde hesaplanır:

$$d_{ist}(c, p) = \sqrt{\sum_{j=1}^d (c_{i,j} - p_j)^2} \quad (1.1)$$

Adım 4: Kümeleme merkezlerini güncelleyin. Her kümedeki veri noktalarının ortalama değerini hesaplayın

$$c'_i = \frac{s_i}{|c_i|} = \frac{\sum_{p \in c_i} p}{|c_i|} \quad (1.2)$$

Merkez noktasını güncelleyin ve k-means algoritmasının yeni kümeleme merkezini belirleyin. Her küme için Karesel Hataların Toplamını (Sum of Squarred Errors-SSE) hesaplayın.

$$SSE = \sum_{i=1}^k \sum_{p \in c_i} (d_{ist}(c'_i, p))^2 \quad (1.3)$$

Adım 5: Hedef işlev SSE yakınsayana veya önceden belirlenen maksimum yineleme sayısına ulaşana kadar Adım 3-Adım 4'ü tekrarlayın.

1.3.1.2. Bulanık c-means kümeleme algoritması

Geleneksel K-ortalımalı kümeleme yaklaşımında kümeler, her kümedeki nesnelerin ortalama değerlerine göre oluşturulur. Bu işlem, orijinal veriler için, 0 s ve 1 s ile temsil edilen ikili değerleri içeren bir üyelik matrisiyle sonuçlanır. Bu nedenle, geleneksel K-ortalımlar algoritmasına sıklıkla sert kümeleme yöntemi denir. Bununla birlikte, bu özellik, gerçek dünya senaryolarında pratik uygulanabilirliğini önemli ölçüde kısıtlamaktadır (Yang ve ark, 2024). Bu kısıtlamayı gidermek için Bulanık c-means algoritması geliştirilmiştir. Bulanık c-means kümeleme algoritması Bezdek ve ark. (1984) tarafından geliştirilmiştir.

Bulanık c-means modeli aşağıdaki gibi gösterilmektedir (Wang ve ark, 2024)

$$\sum_{i=1}^n \sum_{k=1}^c p_{ki}^\alpha \|X_i - v_k\|_2^2, \quad (1.4)$$

$$s. t. P \geq 0, (p_i)^T 1_c \quad (1.5)$$

Bulanık c-means kümeleme algoritması hızlı yakınsama hızına sahip klasik bir bölümlerme algoritmasıdır. Ancak bulanık c-means kümeleme algoritmasının güçlü gürültü ve aykırı değerlere karşı dayanıklılığı zayıftır. Algoritmadaki bu kusurun üstesinden gelmek için, Krishnapuram ve Keller (1993) tarafından olası c-means kümeleme algoritması önerilmiştir. Bununla birlikte olası c-means kümeleme algoritması hala ciddi bir çakışan kümelenme fenomeni ve başlatma duyarlılığı konusunda sıkıntılıdır (Benjamin ve Yang, 2022, Wu ve Guo, 2022). Olası c-means kümeleme algoritmasının eksikliklerini gidermek için Pal ve ark. (2005), bulanık c-means ve olası c-means algoritmalarını birleştirerek “olası bulanık c-ortalımlar” kümelemesini (PFCM) sunmuştur. Geliştirilen algoritma kısmen bulanık c-means

algoritmasının kararlılığını ve olası c-means algoritmasının sağlamlığını göstermektedir ve birçok pratik alanda geniş çapta araştırılmış ve uygulanmıştır (Yu ve ark, 2024).

1.3.2. Araç rotalama problemi

Temel Araç Rotalama Problemi (ARP), coğrafi olarak merkezi depoya dağılmış n müşteriye, başlangıçta merkezi bir depoda konuşlandırılmış, aynı kapasiteye sahip m araçla belirli miktarda malın teslim edilmesi olarak tanımlanabilir (Nielsen ve ark, 2024).

Araç rotalama problemleri yaygın olarak karşılaşılan problemlerdir. Literatürdeki örnekleri Tablo 1.2’de görülmektedir.

Tablo 1.2. Araç Rotalama Problemleri örnekleri.

Referans	Çalışma alanı
Xiao ve ark. (2024)	Düşük karbonlu araç rotalama- Büyük komşuluk arama algoritması
Cavaliere ve ark. (2024)	Kapasiteli araç rotalama problemi- heuristik
Alvarez ve ark. (2024)	Stokastik süreçler ve talepler içeren araç rotalama problemi
Kou ve ark. (2024)	Bölünmüş teslimat aracı rotalama problemi
Parada ve ark. (2024)	Monoton başvuruya sahip Stokastik araç rotalama problemi
Yu ve ark. (2024)	Çok depolu iki kademeli araç rotalama problemi – simüle edilmiş tavlama buluşsal yöntemi
Rodriguez-Esparza ve ark. (2024)	Kapasiteli elektrikli araç rotalama problemi – tavlama ve güçlendirme öğrenmesine dayalı hiper-sezgisel yöntem
Şimşir ve Ekmekçi (2019)	Kapasiteli araç rotalama – yapay arı kolonisi algoritması
Garside ve ark. (2024)	Yeşil araç rotalama problemi
Wang ve ark. (2024)	İki kademeli, çok ürünlü, çok modlu araç rotalama problemi
Fragkogios ve ark. (2024)	Zaman pencere, çok yolculuklu araç rotalama problemi
Su ve ark. (2024)	Zaman pencere, çok depolu araç rotalama problemi
Lehmann ve Winkenbach (2024)	Karışık alım ve teslimat talebi ve zaman pencereleri olan iki kademeli çoklu yolculuk araç rotalama problemi
Polimeni ve Vitetta (2024)	Karayolu taşımacılığı
Ferreira ve ark. (2024)	Yeşil araç rotalama problemi

1.3.2.1. Araç rotalama probleminin matematiksel modellenmesi

Araç Rotalama Problemlerinin (ARP) matematiksel modellenmesi yapılırken, ilgili problemin kapsamındaki kısıtlara göre farklı matematiksel ifadelerle zenginleştirilebilirler.

ARP için kullanılan matematiksel denklemler aşağıda sıralanmıştır (Boz ve Aras, 2024):

Kümeler:

$B\{1,2,3,\dots,n\}$	Bayi Kümesi
$N\{0,1,2,3,\dots,n\}$	Düğüm Kümesi (Depo 0. Düğümdür)
$A\{1,2,3,\dots,n\}$	Araç Kümesi

Parametreler:

A	araç sayısı
K	kapasite
Y	ürün taban yüzölçümü
D	alt tur engelleme karar değişkeni
S_{iA}	A aracına atanan i.bayinin tüm siparişleri
M_{ij}	i.bayi ile j.bayi arasındaki mesafe

$$\sum_j \sum X_{ijA} = 1 \quad \forall_i \quad (2.1)$$

$$\sum_i X_{ijA} - \sum_j X_{jiA} = 0 \quad \forall_{i,j,A} \quad (2.2)$$

$$\sum_i X_{oiA} = 1 \quad \forall_A \quad (2.3)$$

$$\sum_i X_{iA} = 1 \quad \forall_A \quad (2.4)$$

$$\sum_j \sum_i Y_{N+1} X_{ijA} \leq K \quad \forall_A \quad (2.5)$$

$$D_j \geq D_{i+1} - N(-\sum_A X_{ijA}) \quad \forall_{i,j} \quad (2.6)$$

$$D_j \geq 0 \quad \forall_j \quad (2.7)$$

$$\sum_A S_{iA} = 1 \quad \forall_i \quad (2.8)$$

$$X_{ijA}, S_{iA} \in \{0,1\} \quad \forall_{ij} \quad (2.9)$$

$$\sum_i \sum_j \sum_A M_{ij} X_{ijA} \quad \forall_{ijA} \quad (2.10)$$

Denklem (2.1), her bayiye sadece bir araç ile sipariş teslimi olacağını göstermektedir. Denklem (2.2), rotanın devam edebilmesi için her aracın bir önceki bayiden (veya depodan) çıkıp sonraki bayiye (veya depoya) gitmesinin gerekliliği göstermektedir. Denklem (2.3), araçların rotalarına depodan başlamasını sağlamaktadır. Denklem (2.4), araçların rotalarını depoda bitirmesini sağlamaktadır. Denklem (2.5), araçların kapasite sınırı olduğunu göstermektedir. Kapasite dolduğunda yeni araca geçilecektir. Denklem (2.6) ve (2.7), uygulamada alt-turların oluşmasını engellemeyi sağlamaktadır. Denklem (2.8), bayi siparişlerinin yalnızca bir araca yüklenmesini sağlamaktadır. Denklem (2.9), işaret kısıtlarını göstermektedir. Denklem (2.10), amaç fonksiyonunu göstermektedir. Rotalar oluşturulurken bayiler arasında kat edilen mesafenin minimizasyonu amaçlanmaktadır.

1.3.2.2. Yapay arı kolonisi algoritması

Yapay Arı Kolonisi algoritması (ABC), 2005 yılında Karaboğa (2005) tarafından önerilmiştir. Başlangıçta sürekli sayısal optimizasyon problemlerini çözmek için kullanılmasına rağmen, ABC yalnızca uygulanması kolay bir algoritma değil aynı zamanda da mükemmel bir algoritma verimliliğine de sahiptir. ABC algoritması, ikinci dereceden atama problemi, minimum yayılan ağaç problemi, sırt çantası problemi, atölye planlama problemi, araç rotalama problemi ve gezgin satıcı problemi gibi ayrık kombinatoriyal optimizasyon problemlerini çözmek için değiştirilmiştir (li ve ark, 2024).

ABC algoritması arıların yiyecek arama sürecini simüle ederek geliştirilen akıllı bir optimizasyon algoritmasıdır (Ma ve ark, 2024). Temel mekanizması dört bölümden oluşur: besin kaynağı, görevli arılar, takipçi arılar ve keşif arıları (Jin ve ark, 2024; Anuradha ve ark, 2024):

1. Yiyecek kaynağı: Yiyecek kaynağı, belirli bir gösterim şemasıyla verilen, çözülmesi gereken optimizasyon probleminin uygulanabilir çözümüne karşılık gelir. Yiyecek kaynağının kalitesi, yani çözümün uygulanabilirliği genellikle uygunluk endeksi kullanılarak değerlendirilir.
2. Görevli arı aşaması: Görevli arı sayısı, yiyecek kaynaklarının sayısına karşılık gelir. Yiyecek kaynağını bulduktan sonra, görevli arılar yiyecek kaynaklarını bir seçim stratejisi aracılığıyla takip eden arılarla paylaşırlar.

3. Arı aşamasını takip edin: Görevli arıların yönlendirmesine göre arıların mevcut besin kaynaklarının yakınında yeni yiyecek kaynakları aramasını takip edin ve daha iyi yiyecek kaynaklarını seçin.
4. Keşif arı aşaması: Keşif arılar yuvanın yakınında yeni yiyecek kaynakları ararlar. Bir yiyecek kaynağı birçok keşiften sonra yenilenmezse, görevli arı bir keşif arısına dönüşür ve yeni ve değerli bir yiyecek kaynağı arar.

Son yıllarda ABC, iyi bilinen bir optimizasyon tekniği haline gelmiştir. Literatürdeki örnekleri Tablo 1.3'te gösterilmektedir.

Tablo 1.3. Yapay Arı Kolonisi

Referans	Çalışma alanı
Wang ve ark. (2024)	Yanardağ basınç kaynağı parametresinin ters çevrilmesi
Zhang ve ark. (2024)	Izgara tabanlı yapay arı kolonisi algoritması ile atölye planlaması
Jiang ve ark. (2023)	Yapay arı kolonisi algoritması ile gemilerin çok amaçlı optimizasyonu
Huang ve ark. (2024)	Çok amaçlı ayrık yapay arı kolonisi algoritması ile montaj hattı dengelemesi
Wang ve ark. (2024)	Hibrit akış atölyeleri için ayrık bir yapay arı kolonisi algoritması uygulaması
Ren ve ark. (2024)	Yapay arı kolonisi algoritması ile iki amaçlı söküm hattının planlanması
Mumtaz ve ark. (2024)	Genetik-yapay arı kolonisi algoritması ile hat dengeleme ve AGV planlama
Li ve ark. (2023)	Bilgi tabanlı yapay arı kolonisi algoritması ile permütasyon akış atölyesi planlaması
Wang ve ark. (2023)	Öz uyarlamalı çok stratejili bir yapay arı kolonisi algoritması ile kablosuz sensör ağlarında kapsama optimizasyonu
Guo ve ark. (2024)	Gelişmiş hibrit yapay arı kolonisi algoritması ile söküm hattı dengelemesi
Pan ve ark. (2024)	Geliştirilmiş yapay arı kolonisi algoritması ile karmaşık optimizasyon problemlerinin çözümü
Han ve ark. (2023)	Yapay arı kolonisi algoritması ile rüzgar türbini alt montajlarının güvenilirlik analizi
Yang ve ark. (2023)	Gevşek hızlandırma tabanlı ayrık yapay arı kolonisi algoritması ile kaynak kısıtlı ameliyathane planlaması
Cui ve ark. (2024)	Öğrenme tabanlı yapay arı kolonisini kullanarak çok robotlu yol planlama

1.3.2.3. Ateş böceği algoritması

Ateş Böceği (FA) Algoritması metasezgisel bir algoritmadır. Metasezgisel algoritmalar, problemlere makul bir sürede optimum bir çözüm bulmak için çözüm uzayını arayan yerel optimizasyon çıkış mekanizmalarına sahip rastgele arama algoritmalarıdır. En yaygın kullanılan ve etkili algoritmalar arasında genetik algoritma (GA) ve ateş böceği algoritması yer almaktadır (Sahani ve ark, 2024). GA, başlangıçta doğrusal, dışbükey ve benzeri problemleri çözmek amacıyla klasik optimizasyon yöntemleri için geliştirilmiştir, ancak daha sonraları ayrık, sürekli ve doğrusal olmayan problemlerin çözümünde de kullanılmışlardır. FA algoritması ise doğadan ve rastgelelikten esinlenen, optimizasyon, mühendislik problemlerinin hemen hemen tüm alanlarında kullanılan grup zeka yöntemlerinden ve metasezgisel algoritmalarından biridir. En düşük seviyesinde, FA algoritması ateş böceklerinin hareketlerini ve göz kırpmada desenlerini taklit ederek bir arama alanı içinde çözümler üretmeye ve en iyi hayatta kalma çözümünü seçmeye odaklanır. Rastgele arama, optimum yerel tuzağa yakalanmaktan kaçınır (Anshory ve ark, 2024).

FA algoritması ile yapılan literatür çalışmalarından örnekler Tablo 1.4'te gösterilmektedir.

Tablo 1.4. Ateş Böceği Algoritması

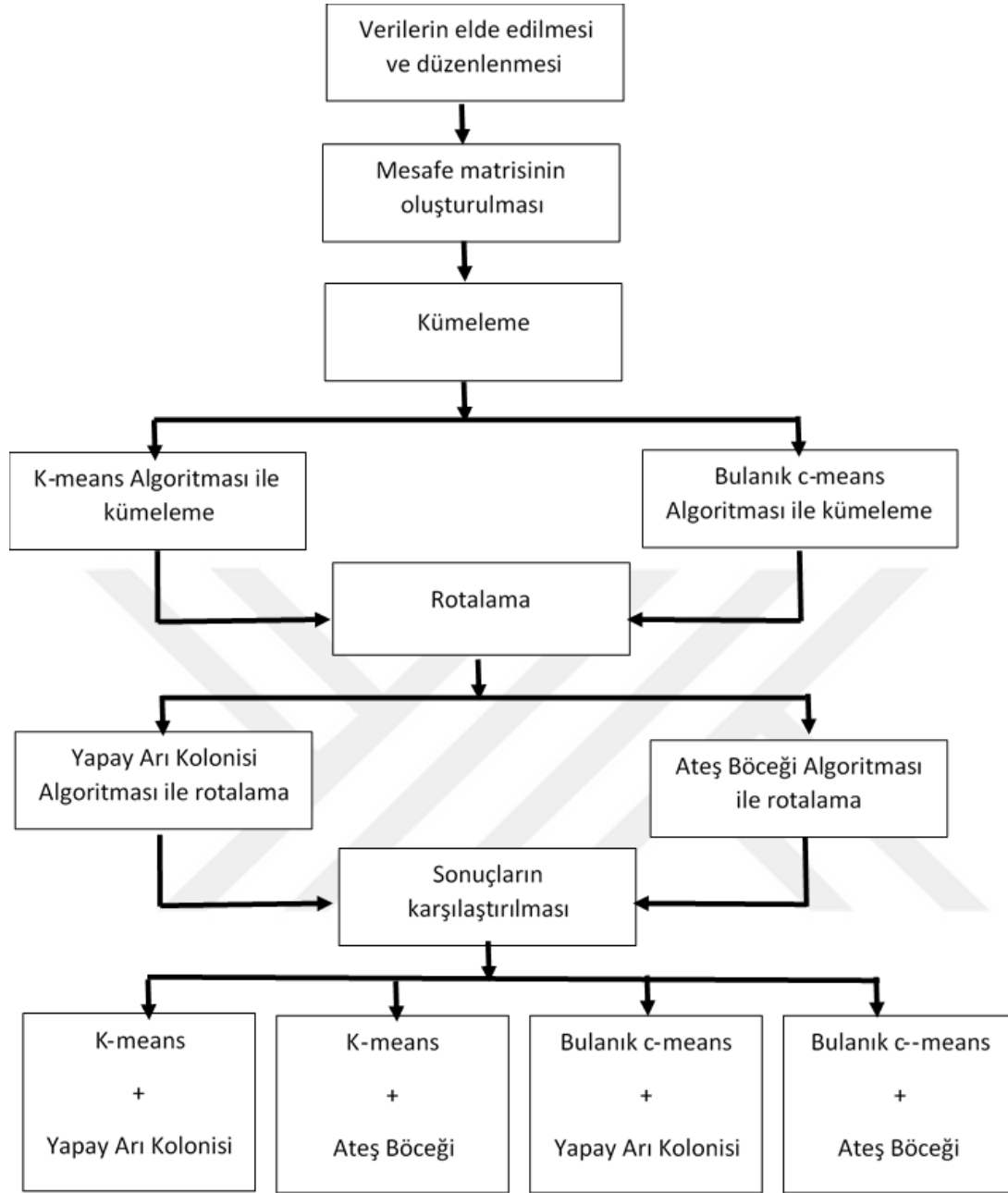
Referans	Çalışma alanı
Zhao ve ark. (2024)	Yüksek boyutlu özellik seçimi için çok amaçlı ateş böceği algoritması
Yuan ve ark. (2024)	Hibrit yenilenebilir enerji sistemlerinin en iyi şekilde boyutlandırılmasında ateş böceği algoritması
Kumar ve ark. (2024)	Binalara entegre fotovoltaik sistemlerde enerji tahmini için ateş böceği algoritması
Ahsan ve ark. (2024)	Hibrit enerji depolama sistemlerinin optimum boyutlandırılması ve maliyet analizinde ateş böceği algoritması

2. UYGULAMA

2.1. Giriş

Çalışmada beyaz eşya üretim işletmesine ait Ankara’da bulunan ana depodan Ankara ve yakın bölgelere yapılacak teslimatlar için dağıtım planlaması yapılması amaçlanmaktadır. Hali hazırda teslimat ana depodaki lojistik personelinin deneyim ve inisiyatifine dayalı olarak yapılmaktadır. Bu da müşteri isteklerinin zamanında karşılanmasında gecikmelere sebep olmaktadır. Bu nedenle mesafeleri ve talepleri göz önünde bulundurarak kümeleme ve rotalama uygulaması yapılacaktır.

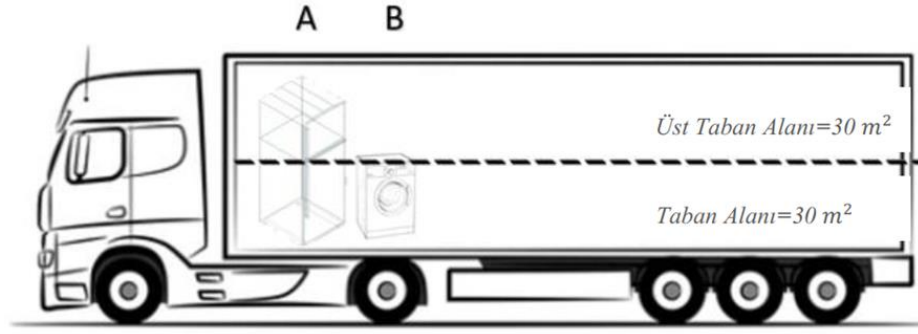
Uygulamanın aşamaları Şekil 2.1’de gösterilmektedir.



Şekil 2.1. Uygulama aşamaları.

Problemde 59 adet bayi seçilerek veri seti oluşturulmuştur. Veri seti olarak bayilerin adresleri ve sipariş miktarları alınmıştır. Bayilerin bulundukları noktaların birbirlerine ve ana depoya olan uzaklıkları Google haritalar üzerinden belirlenerek uzaklık matrisi oluşturulmuştur. Uzaklık matrisi Ek-1’de verilmektedir.

Araç için alt ve üst taban ölçüleri Şekil 2.2’de yer almaktadır.



Şekil 2.2. Araç taban ölçüleri.

Aracın taşıma kapasitesi ile ilgili ölçüleri $3 \times 5 \text{ m}^2$ 'dir. Problem boyunca aracın ölçüleri aynı kabul edilecektir. Ürün yerleştirmesi yapılırken ürünlerin en ve boylarına göre yerleştirme yapılacaktır. Problemden iki grup ürün bulunmaktadır. Buzdolabı, derin dondurucu vb. ürünler A grubunu oluştururken; çamaşır makinesi, bulaşık makinesi, fırın vb. ürünler B grubunu oluşturacaktır. A grubu ürünler üst üste konulamayan ürünlerdir, B grubu ürünler ise iki tane ürün üst üste konulacak şekilde yerleştirilebilir.

Teslimat noktalarının kümelenmesinde Python programlama dili kullanılmıştır. Python'da ilgili kütüphaneler kullanılarak k-means ve Bulanık c-means olmak üzere iki farklı algoritma ile uygulama yapılmıştır. Her iki algoritma ile elde edilen kümeleme çözümleri için Yapay Arı Kolonisi ve Ateş Böceği Algoritması olmak üzere iki farklı rotalama uygulaması yapılmıştır.

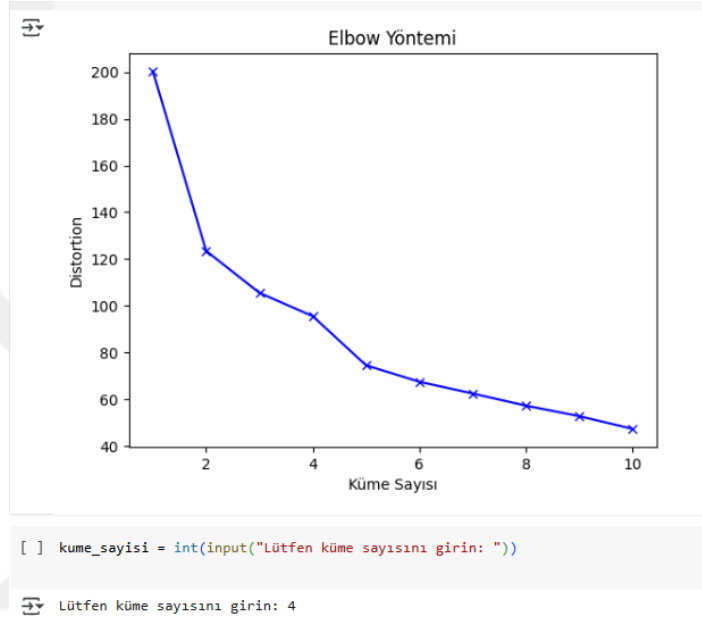
2.2. K-means ile Kümeleme ve Yapay Arı Kolonisi ile Rotalama

Python programlama dilinde K-means ile teslimat bayileri kümelenmiş ve sonra Yapay Arı Kolonisi ile teslimat yapacak araçların güzergahları rotalanmıştır. K-means ile kümeleme yapılırken, kümelerin sayısını belirlemek için Elbow yöntemi kullanılmıştır. Depodan ilk bayiye olan mesafeyi ve bayiler arası mesafeleri optimize etmek için Yapay Arı Kolonisi algoritması kullanılmıştır. Yapay Arı Kolonisi algoritması, arıların davranışını simüle ederek en kısa mesafeyi bulmaya çalışır. Rastgele başlatılan çözümler arasında en iyi rotayı bulmak için iteratif olarak çalışır. Arılar farklı rotalar dener ve en kısa mesafeyi sağlayan çözümü bulmaya çalışırlar. Python programlama dilinde yazılan kodlar Ek-1'de verilmektedir.

- **K-means Elbow Yöntemi:** Bayileri en uygun kümelere ayırır.

- **Yapay Arı Kolonisi Algoritması:** Her kümedeki bayiler arasındaki en uygun rotayı optimize eder.
- **Depo (1. Bayi):** Başlangıç noktası olarak kabul edilir ve ilk bayiye olan mesafe hesaplamalara dahil edilir.
- **Tır Yükleme:** Siparişlerin tırlara en uygun şekilde yerleştirilmesi sağlanır ve her tırın rotası optimize edilir.

Grafik Analizi



Şekil 2.3. Elbow Yöntemi kullanılarak optimal küme sayısı .

Bu grafik, Elbow Yöntemi kullanılarak optimal küme sayısını belirlemek için çizilmiştir. Grafik üzerinde "dirsek" (elbow) olarak adlandırılan nokta, eğrinin belirgin bir şekilde azaldığı ve sonra daha yavaş bir azalma gösterdiği yerdir. Bu noktada eğrinin eğimi önemli ölçüde azalır.

Bu grafiğe baktığımızda, "dirsek" noktası 3. ve 4. kümeler arasında yer alıyor gibi görünüyor. Bu nedenle, küme sayısını 3 veya 4 olarak seçmek mantıklı olacaktır. Ancak genellikle bu yöntemde, dirseğin en belirgin olduğu nokta tercih edilir, bu durumda küme sayısı olarak 4 seçilebilir.

2.2.1. K-means kümeleme sonuçları

K-means algoritması ile elde edilen kümeler aşağıda verilmektedir. Ana depo 1 ile, bayiler 2'den başlayarak 60'a kadar numaralandırılmıştır. Örneğin küme 1'de 60., 34. ve 40. bayiler bulunmaktadır.

Küme 1: [60, 34, 40]

Küme 2: [23, 50, 30, 2, 21, 46, 16, 26, 45, 31, 39, 58, 27, 14, 51, 13, 20, 59, 38, 5, 7, 11, 57, 25, 24, 15, 52, 36, 54, 6, 35, 32, 42, 28, 29, 22, 41, 48, 47, 55, 43, 33, 37, 8, 49, 10]

Küme 3: [56, 9, 19]

Küme 4: [53, 17, 18, 4, 3, 12, 44]

2.2.2. Yapay arı kolonisi algoritması ile araçların rotasyonu

Yapay Arı Kolonisi Algoritması ile elde edilen araç rotaları ve kat edilen mesafeler aşağıda gösterilmektedir.

Küme 1:

Araç 1-Rota: [1, 60, 40, 34, 1]-Mesafe: 358,3 km

Küme 2:

Araç 1-Rota: [1, 39, 48, 43, 52, 13, 55, 54, 16, 29, 20, 1]-Mesafe: 180,8 km

Araç 2-Rota: [1, 26, 14, 51, 27, 31, 36, 57, 42, 1]-Mesafe: 97,7 km

Araç 3-Rota: [1, 2, 24, 25, 11, 49, 35, 5, 22, 41, 47, 37, 1]-Mesafe: 227,89 km

Araç 4-Rota: [1, 45, 23, 33, 10, 28, 15, 50, 46, 8, 6, 38, 32, 58, 30, 1]-Mesafe: 293,2 km

Araç 5-Rota: [1, 7, 59, 21, 1]-Mesafe: 64,3 km

Küme 3:

Araç 1-Rota: [1, 9, 19, 56, 1]-Mesafe: 349,1 km

Küme 4:

Araç 1-Rota: [1, 44, 3, 18, 12, 17, 53, 4, 1]-Mesafe: 372 km

2.3. K-means ile Kümeleme ve Ateş Böceği Algoritması ile Rotalama

Python programlama dilinde K-means ile teslimat bayileri kümelenmiş ve sonra Ateş Böceği Algoritması ile teslimat yapacak araçların güzergahları rotalanmıştır. K-means ile kümeleme yapılırken, kümelerin sayısını belirlemek için Elbow yöntemi kullanılmıştır. Depodan ilk bayiye olan mesafeyi ve bayiler arası mesafeleri optimize etmek için Ateş Böceği Algoritması kullanılmıştır. Bu algoritma, ateşböceklerinin

birbirlerine olan çekim kuvvetine dayalı olarak çalışır. Mesafe ile ters orantılı olarak, ateşböcekleri birbirlerine çekilir ve daha iyi çözümler (daha kısa mesafeler) bulmaya çalışır. Beta: Ateşböceklerinin birbirlerine olan çekim kuvvetini hesaplamak için kullanılır. Alpha: Rastgele bir perturbasyon eklemek için kullanılır, bu da aramayı çeşitlendirir. Gamma: Çekim kuvvetinin uzaklıkla ne kadar hızlı azaldığını kontrol eder. Best_distance: Bulunan en iyi çözümün mesafesini depolar.

Pyhton programlama dilinde yazılan kodlar Ek-2’de verilmektedir.

- **K-means Elbow Yöntemi:** Bayileri en uygun kümelere ayırır.
- **Ateşböceği Algoritması:** Her kümedeki bayiler arasındaki en uygun rotayı optimize eder.
- **Depo (1. Bayi):** Başlangıç noktası olarak kabul edilir ve ilk bayiye olan mesafe hesaplamalara dahil edilir.
- **Tır Yükleme:** Siparişlerin tırlara en uygun şekilde yerleştirilmesi sağlanır ve her tırın rotası optimize edilir.

2.3.1. K-means ile kümeleme sonuçları

Bu çözüm grubunda (k-means ve ateş böceği algoritması) da kümeleme işlemi K-means algoritması ile yapılmıştır. Sonuçlar bölüm 3.2.1 de verilmiştir.

2.3.2. Ateş böceği algoritması ile araçların rotasyonu

Ateş Böceği Algoritması ile elde edilen araç rotaları ve kat edilen mesafeler aşağıda gösterilmektedir.

Küme 1:

Araç 1-Rota: [1, 34, 40, 60, 1]-Mesafe: 358,3 km

Küme 2:

Araç 1-Rota: [1, 8, 23, 27, 58, 47, 39, 14, 13, 26, 31, 1]-Mesafe: 135,9 km

Araç 2-Rota: [1, 11, 54, 2, 45, 7, 16, 59, 55, 5, 35, 1]-Mesafe: 203,29 km

Araç 3-Rota: [1, 30, 21, 24, 49, 37, 46, 52, 10, 38, 22, 28, 51, 1]-Mesafe: 205,2 km

Araç 4-Rota: [1, 6, 36, 15, 20, 48, 25, 32, 50, 29, 41, 57, 43, 33, 1]-Mesafe: 244,89 km

Araç 5-Rota: [1, 42, 1]-Mesafe: 6,6 km

Küme 3:

Araç 1-Rota: [1, 9, 19, 56, 1]-Mesafe: 349,1 km

Küme 4:

Araç 1-Rota: [1, 3, 18, 17, 12, 44, 53, 4, 1]-Mesafe: 383,7 km

2.4. Bulanık c-means ile Kümeleme ve Yapay Arı Kolonisi ile Rotalama

Python programlama dilinde Bulanık c-means ile teslimat bayileri kümelendi ve sonra Yapay Arı Kolonisi ile teslimat yapacak araçların güzergahları rotalandı. Bulanık c-means bayileri belirli bir küme sayısına göre kümeler ayırır. Bu algoritma, verilerin birden fazla kümeye ait olmasına izin verir ve her bir veri noktası, her bir küme merkezine olan mesafeye dayalı olarak bir üyelik derecesine sahip olur. Depodan ilk bayiye olan mesafeyi ve bayiler arası mesafeleri optimize etmek için Yapay Arı Kolonisi algoritması kullanılmıştır. Yapay Arı Kolonisi algoritması, arıların yiyecek arama davranışını taklit ederek en kısa mesafeyi bulmaya çalışır. Rastgele başlatılan çözümler arasında en iyi rotayı bulmak için iteratif olarak çalışır. Arılar farklı rotalar dener ve en kısa mesafeyi sağlayan çözümü bulmaya çalışırlar.

- **Bulanık C-Means:** Bayileri, belirli bir küme sayısına göre kümeler ayırır. Bayilerin her biri, farklı kümeler belirli bir üyelik derecesine sahip olabilir.
- **Yapay Arı Kolonisi Algoritması:** Her kümedeki bayiler arasındaki en uygun rotayı optimize eder. Arılar, yiyecek arama davranışını taklit ederek, her iterasyonda daha iyi çözümler bulmaya çalışırlar.
- **Depo (1. Bayi):** Başlangıç noktası olarak kabul edilir ve ilk bayiye olan mesafe hesaplamalara dahil edilir.
- **Tır Yükleme:** Siparişlerin tırlara en uygun şekilde yerleştirilmesi sağlanır ve her tırın rotası optimize edilir.

Python programlama dilinde yazılan kodlar Ek-3'te verilmektedir.

2.4.1. Bulanık c-means algoritması ile kümeleme

Bulanık c-means algoritması ile elde edilen kümeler aşağıda verilmektedir. Örneğin küme 4'te 4. ve 34. bayiler bulunmaktadır.

Küme 1: [9]

Küme 2: [54, 45, 37, 35, 22, 2, 29, 41, 38, 12]

Küme 3: [21, 6, 36, 8, 52, 39, 23, 42, 50, 58]

Küme 4: [4, 34]

Küme 5: [13, 59, 47, 48, 20, 30, 57, 55, 15, 14, 27, 28, 10, 43, 5, 51, 26]

Küme 6: [18, 3, 17, 44]

Küme 7: [7, 33, 53, 31]

Küme 8: [25, 46, 32, 24, 49, 16, 11]

Küme 9: [19, 56]

Küme 10: [40, 60]

2.4.2. Yapay arı kolonisi algoritması ile araçların rotasyonu

Yapay Arı Kolonisi Algoritması ile elde edilen araç rotaları ve kat edilen mesafeler aşağıda gösterilmektedir.

Küme 1:

Araç 1-Rota: [1, 9, 1]-Mesafe: 262,0 km

Küme 2:

Araç 1-Rota: [1, 37, 29, 2, 38, 41, 45, 22, 54, 35, 12, 1]-Mesafe: 159,8 km

Küme 3:

Araç 1-Rota: [1, 42, 58, 52, 36, 8, 6, 39, 50, 21, 23, 1]-Mesafe: 46,89 km

Küme 4:

Araç 1-Rota: [1, 34, 4, 1]-Mesafe: 214,79 km

Küme 5:

Araç 1-Rota: [1, 57, 43, 28, 30, 51, 47, 27, 10, 5, 55, 1]-Mesafe: 93,5 km

Küme 5:

Araç 1-Rota: [1, 26, 20, 15, 48, 14, 59, 13, 1]-Mesafe: 32,7 km

Küme 6:

Araç 1-Rota: [1, 44, 3, 18, 17, 1]-Mesafe: 211,7 km

Küme 7:

Araç 1-Rota: [1, 53, 31, 7, 33, 1]-Mesafe: 106,19 km

Küme 8:

Araç 1-Rota: [1, 24, 46, 49, 11, 16, 32, 25, 1]-Mesafe: 87,79 km

Küme 9:

Araç 1-Rota: [1, 19, 56, 1]-Mesafe: 263 km

Küme 10:

Araç 1-Rota: [1, 40, 60, 1]-Mesafe: 351,3 km

2.5. Bulanık c-means ile Kümeleme ve Ateş Böceği Algoritması ile Rotalama

Python programlama dilinde Bulanık c-means ile teslimat bayileri kümelendi ve sonra Ateş Böceği Algoritması ile teslimat yapacak araçların güzergahları rotalandı. Bulanık c-means Bayileri belirli bir küme sayısına göre kümeler ayırır. Bu algoritma, verilerin birden fazla kümeye ait olmasına izin verir ve her bir veri noktası, her bir küme merkezine olan mesafeye dayalı olarak bir üyelik derecesine sahip olur. Depodan ilk bayiye olan mesafeyi ve bayiler arası mesafeleri optimize etmek için Ateş Böceği Algoritması kullanılmıştır. Bu algoritma, ateşböceklerinin birbirlerine olan çekim kuvvetine dayalı olarak çalışır. Mesafe ile ters orantılı olarak, ateşböcekleri birbirlerine çekilir ve daha iyi çözümler (daha kısa mesafeler) bulmaya çalışır. Beta: Ateşböceklerinin birbirlerine olan çekim kuvvetini hesaplamak için kullanılır. Alpha: Rastgele bir perturbasyon eklemek için kullanılır, bu da aramayı çeşitlendirir. Gamma: Çekim kuvvetinin uzaklıkla ne kadar hızlı azaldığını kontrol eder. Best_distance: Bulunan en iyi çözümün mesafesini depolar.

- **Bulanık C-Means:** Bayileri, belirli bir küme sayısına göre kümeler ayırır. Bayilerin her biri, farklı kümeler belirli bir üyelik derecesine sahip olabilir.
- **Ateşböceği Algoritması:** Her kümedeki bayiler arasındaki en uygun rotayı optimize eder. Ateşböceklerinin birbirlerine olan çekim kuvvetine dayalı olarak en kısa mesafe bulunur.
- **Depo (1. Bayi):** Başlangıç noktası olarak kabul edilir ve ilk bayiye olan mesafe hesaplamalara dahil edilir.
- **Tır Yükleme:** Siparişlerin tırlara en uygun şekilde yerleştirilmesi sağlanır ve her tırın rotası optimize edilir.
- **Sonuçlar:** Tüm araçların yaptığı toplam mesafeler tablo olarak gösterilir.

Python programlama dilinde yazılan kodlar Ek-4'te verilmektedir.

2.5.1. Bulanık c-means ile kümeleme sonuçları

Bu çözüm grubunda (bulanık c-means ve ateş böceği algoritması) da kümeleme işlemi Bulanık c-means algoritması ile yapılmıştır. Sonuçlar bölüm 3.4.1’de verilmiştir.

2.5.2. Ateş böceği algoritması ile araçların rotasyonu

Ateş Böceği Algoritması ile elde edilen araç rotaları ve kat edilen mesafeler aşağıda gösterilmektedir.

Küme 1:

Araç 1-Rota: [1, 4, 34, 53, 44, 3, 1]-Mesafe: 284,1 km

Küme 2:

Araç 1-Rota: [1, 20, 48, 14, 59, 5, 13, 10, 30, 57, 43, 1]-Mesafe: 55,0 km

Araç 2-Rota: [1, 51, 26, 55, 1]-Mesafe: 24,2 km

Küme 3:

Araç 1-Rota: [1, 37, 29, 2, 41, 22, 54, 16, 45, 32, 46, 1]-Mesafe: 133,3 km

Araç 2-Rota: [1, 11, 35, 38, 1]-Mesafe: 62,09 km

Küme 4:

Araç 1-Rota: [1, 60, 40, 1]-Mesafe: 351,3 km

Küme 5:

Araç 1-Rota: [1, 56, 19, 1]-Mesafe: 263 km

Küme 6:

Araç 1-Rota: [1, 9, 1]-Mesafe: 262 km

Küme 7:

Araç 1-Rota: [1, 42, 8, 31, 7, 52, 58, 6, 21, 39, 50, 36, 23, 1]-Mesafe: 77,5 km

Küme 8:

Araç 1-Rota: [1, 27, 47, 25, 28, 24, 49, 15, 33, 1]-Mesafe: 120,69 km

Küme 9:

Araç 1-Rota: [1, 12, 18, 1]-Mesafe: 134,2 km

Küme 10:

Araç 1-Rota: [1, 17, 1]-Mesafe: 180,6 km





3. SONUÇ VE DEĞERLENDİRME

Bu tez çalışmasında beyaz eşya üretimi yapan bir işletmenin, Ankara’da yer alan ana dağıtım deposundan Ankara ve civarındaki 59 adet bayiye yapacağı ürün teslimatı ile ilgili dağıtım problemi ele alınmaktadır.

Google Maps uygulaması ile bayiler ve ana depo arasındaki mesafeler dikkate alınarak mesafe matrisi oluşturulmuştur. Bayiler ağırlıklı olarak Ankara civarında yer almaktadır. Bayiler K-means algoritmasında Elbow yöntemi kullanılarak ve Bulanık c-means algoritması ile olmak üzere iki farklı şekilde kümelenmiştir. Kümelerde ihtiyaç durumuna göre bir veya daha fazla araç kullanılmıştır. Araç içi yüklemede, ürünler üst üste yüklenebilir ve yüklenemez olarak iki sınıfa ayrılmıştır. Ve yükleme esnasında bu sınıflandırma dikkate alınmıştır.

Çalışmada rotalama için Yapay Arı Kolonisi Algoritması ve Ateş Böceği Algoritması kullanılmasının sebebi, bu algoritmaların tek bir çözümde sıkışıp kalmamayı ve yeni çözümler keşfederek daha iyi çözümler bulmayı amaçlayan sezgisel yöntemler olmasıdır.

Çalışmada ürünleri depodan bayilere ulaştırmak için dört farklı uygulama yapılmıştır. Uygulamalarda kat edilen mesafe ve araç kapasitesi kısıt olarak dikkate alınmıştır. Birinci uygulamada k-means algoritması ile bayiler kümelenirken yapay arı kolonisi algoritması ile araç rotalama yapılmıştır. İkinci uygulamada k-means algoritması ile bayiler kümelenirken ateş böceği algoritması ile araç rotalama yapılmıştır. Üçüncü uygulamada bulanık c-means algoritması ile bayiler kümelenirken yapay arı kolonisi algoritması ile araç rotalama yapılmıştır. Dördüncü uygulamada ise bulanık c-means algoritması ile bayiler kümelenirken ateş böceği algoritması ile araç rotalama yapılmıştır.

Sonuçlar Tablo 3.1’de gösterilmektedir.

Tablo 3.1. Çözümlerin karşılaştırılması

	Araç sayısı	Toplam mesafe
K-means algoritması + Yapay Arı Kolonisi algoritması	8	1943,29 km
K-means algoritması + Ateş Böceği algoritması	8	1886,98 km
Bulanık c-means algoritması + Yapay Arı Kolonisi algoritması	10	1839,66 km
Bulanık c-means algoritması + Ateş Böceği algoritması	12	1947,98 km

Tablo 3.1.'de görüldüğü üzere; Problem, Phyton programı üzerinde K-means algoritması ile kümeleme ve Yapay Arı Kolonisi algoritması ile rotalama metodlarına göre çalıştırıldığında çıkan sonuca göre ürün teslimatı yapmak için 8 araç kullanılacaktır. Toplam kat edilen mesafe 1943,29 km'dir. Diğer çözümler de benzer şekilde yorumlanabilir.

K-means algoritması ile kümeleme çözümü yaptırıldığında Elbow metoduna göre optimum küme sayısı 4 olarak bulunmuştur. Küme 2, tek bir araç kapasitesinden fazla ürün içerdiği için küme 2'ye 5 araç atanmıştır. Bu nedenle K-means algoritması ile ulaşılan çözümlerde 8'er araç kullanılmıştır. Dolayısı ile K-means algoritması ile yapılan iki farklı rotalama çözümünü karşılaştırırken dikkat edilmesi gereken kriter mesafedir.

Bulanık c-means algoritmasına göre çalıştırıldığında ise küme sayısı 10 olarak bulunmuştur. Ateş Böceği algoritmasına göre yapılan rotalamada küme 2 ve küme 3, tek bir araç kapasitesinin üzerinde ürün bulundurdıkları için bu kümelere 2'şer araç atanmıştır. Bu nedenle Bulanık c-means Algoritması ile kümeleme ve Ateş Böceği Algoritması ile rotalama çözümünde toplam araç sayısı 12'dir. Dolayısı ile Bulanık c-means algoritması ile yapılan iki farklı rotalama çözümünü karşılaştırırken dikkat edilmesi gereken kriterler hem araç sayısı hem de mesafedir.

Tüm çözümlerin sonuçlarına bakıldığında gerek araç sayısının en düşük olması gerekse de mesafenin kısalığı dolayısı ile K-means algoritması ile kümeleme ve Ateş Böceği Algoritması ile rotalama şeklinde yapılan çözümün optimum çözüm olduğu görülmektedir (8 araç ve 1886,98 km).

Uygulamaya konu olan beyaz eşya firması mevcut durumunda bayilere ürün dağıtımını satış personelinin bilgi ve deneyimine dayalı olarak planlamaktadır. Dağıtım planı yapılırken kümeleme yapılmadığı için birbirinden uzakta olan bayilerin

siparişleri aynı araçlara yüklenebilmektedir. Ve rotalamada belli bir yöntem kullanılmadığı için de araç sayısı fazla çıkmakta ve araçların kat ettiği mesafeler çok daha fazla olmaktadır. Önerilen K-means algoritması ile kümeleme ve Ateş Böceği Algoritması ile rotalama çözümü ile hem araç sayısında hem de kat edilen mesafede iyileştirme sağlanmıştır.





KAYNAKLAR

- [1] Chen, L., Fan, M., & Wang, J. (2024). A controlled data envelopment analysis clustering approach based on individual perspective. *Information Sciences*, 677, 120932, <https://doi.org/10.1016/j.ins.2024.120932>
- [2] Oskouei, A.G., Samadi, N., & Tanha, J. (2024). Feature-weight and cluster-weight learning in fuzzy c-means method for semi-supervised clustering. *Applied Soft Computing*, 161, 111712, <https://doi.org/10.1016/j.asoc.2024.111712>
- [3] Chen, X., Liu, C., Lin, B., Lai, J., & Miao, D. (2024). AHA-3WKM: The optimization of K-means with three-way clustering and artificial hummingbird algorithm. *Information Sciences*, 672, 120661, <https://doi.org/10.1016/j.ins.2024.120661>
- [4] Xu, R., & Wunsch, D. (2005). Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, vol. 16, no. 3, pp. 645-678, doi: 10.1109/TNN.2005.845141
- [5] Hosseinzadeh, M., Tanveer, J., Alanazi, F., Aurangzeb, K., Yousefpoor, M.S., Yousefpoor, E., Darwesh, A., Lee, S.W., & Rahmani, A.M. (2024). An intelligent clustering scheme based on whale optimization algorithm in flying ad hoc networks. *Vehicular Communications*, 49, 100805, <https://doi.org/10.1016/j.vehcom.2024.100805>
- [6] Wang, J., Wu, Y., Huang, X., Zhang, C., & Nie, F. (2024). Projected fuzzy c-means clustering algorithm with instance penalty. *Expert Systems with Applications*, In Press, 124563, <https://doi.org/10.1016/j.eswa.2024.124563>
- [7] Heidari, J., Daneshpour, N., & Zangeneh, A. (2024). A novel K-means and K-medoids algorithms for clustering non-spherical-shape clusters non-sensitive to outliers. *Pattern Recognition*, 155, 110639, <https://doi.org/10.1016/j.patcog.2024.110639>
- [8] Yu, H., J, L., Fan, J., Xie, S., Lan, R. (2024). A feature-weighted suppressed possibilistic fuzzy c-means clustering algorithm and its application on color image segmentation. *Expert Systems with Applications*, 241, 122270, <https://doi.org/10.1016/j.eswa.2023.122270>
- [9] Meng, J., Abed, A.M., Elsehrawy, M.G., Agha, A.A., Abdullah, N., Elattar, S., Abbas, M., Garalleh, H.A., & Assilzadeh, H. (2024). Nano-integrating green and low-carbon concepts into ideological and political education in higher education institutions through K-means clustering. *Heliyon*, vol 10, issue 10, e31244, <https://doi.org/10.1016/j.heliyon.2024.e31244>
- [10] Wu, W., Wang, W., Jia, X., & Feng, X. (2024). Transformer autoencoder for k-means efficient clustering. *Engineering Applications of Artificial Intelligence*, col 133, part F, 108612, <https://doi.org/10.1016/j.engappai.2024.108612>

- [11] Paul, R.K., Jena, S., Chandran, S., Bandyopadhyay, A., & Swain, S. (2024). Image compression scheme based on optimized K-means clustering and higher-level decomposed DWT. *Procedia Computer Science*, vol 235, page 642-655, <https://doi.org/10.1016/j.procs.2024.04.063>
- [12] Xiang R.F. (2024). Use of n-grams and k-means clustering to classify data from free text bone marrow reports. *Journal of Pathology Informatics*, 15, 100358, <https://doi.org/10.1016/j.jpi.2023.100358>
- [13] Raj, E.J.S.J., & Palaniappan, R. (2024). A novel method for movement quality analysis of lower limb joints using surface electromyography signals and k-means clustering technique. *Biomedical Signal Processing and Control*, vol 95, part A, 106455, <https://doi.org/10.1016/j.bspc.2024.106455>
- [14] Dugan E., Barbuto, A.E., Masterson, C.M., & Shilt, J. (2024). Multivariate functional principal component analysis and k-means clustering to identify kinematic foot types during gait in children with cerebral palsy. *Gait & Posture*, vol 113, page 40-45, <https://doi.org/10.1016/j.gaitpost.2024.05.032>
- [15] Zhang, X., Li, F., Shi, Z., & Yang, M. (2024). Multi-view reduced dimensionality k-means clustering with σ -norm and Schatten p -norm. *Pattern Recognition*, 155, 110675, <https://doi.org/10.1016/j.patcog.2024.110675>
- [16] Lee, S.S. (2024). Satellite constellation method for ground targeting optimized with k-means clustering and genetic algorithm. *Engineering Applications of Artificial Intelligence*, 129, 107509, <https://doi.org/10.1016/j.engappai.2023.107509>
- [17] Burada, S., Manjunathswamy, B.E., & Kumar, M.S. (2024). Early detection of melanoma skin cancer: A hybrid approach using fuzzy c-means clustering and differential evolution-based convolutional neural network. *Measurement: Sensors*, 33, 101168, <https://doi.org/10.1016/j.measen.2024.101168>
- [18] Xu, D., & Tian, Y. A. (2015). A comprehensive survey of clustering algorithms. *Annals of Data Science*, vol 2, page 165-193, <https://doi.org/10.1007/s40745-015-0040-1>
- [19] Yang, X., Zhao, W., Xu, Y., Wang, C.D., Li, B., & Nie, F. (2024). Sparse K-means clustering algorithm with anchor graph regularization. *Information Sciences*, 667, 120504, <https://doi.org/10.1016/j.ins.2024.120504>
- [20] Sadeghi, M., Casey, P., Carranza, E.J.M., & Lynch, E.P. (2024). Principal components analysis and K-means clustering of till geochemical data: Mapping and targeting of prospective areas for lithium exploration in Västernorrland Region, Sweden. *Ore Geology Reviews*, 167, 106002, <https://doi.org/10.1016/j.oregeorev.2024.106002>
- [21] Zhou, Q., & Sun, B. (2023). Adaptive k-means clustering based under-sampling methods to solve the class imbalance problem. *Data and Information Management*, 100064, in press, <https://doi.org/10.1016/j.dim.2023.100064>
- [22] Zhang, H., Li, J., Zhang, J., & Dong, Y. (2024). Speeding up k-means clustering in high dimensions by pruning unnecessary distance computations. *Knowledge-Based Systems*, 284, 111262, <https://doi.org/10.1016/j.knosys.2023.111262>

- [23] Li, Y., Song, X., Tu, Y., & Liu, M. (2024). GAPBAS: Genetic algorithm-based privacy budget allocation strategy in differential privacy k-means clustering algorithm. *Computers & Security*, 139, 103697, <https://doi.org/10.1016/j.cose.2023.103697>
- [24] Bezdek, J.C., Ehrlich, R., & Full, W. (1984). FCM: The fuzzy c-means clustering algorithm. *Computer & Geosciences*, 10 (2–3), page 191-203, <https://www.sciencedirect.com/science/article/abs/pii/0098300484900207>
- [25] Krishnapuram, R. & Keller, J.M. (1993). A possibilistic approach to clustering. *IEEE Transactions on Fuzzy Systems*, 1 (2), page 98-110 <https://www.scopus.com/record/display.uri?eid=2-s2.0-0027595430&origin=inward&txGid=389a25eefb02448e6cb4b17d0e4ab0e4>
- [26] Benjamin, J.B.M., & Yang, M.S. (2022). Weighted Multiview possibilistic c-means clustering with L2 regularization. *IEEE Transactions on Fuzzy Systems*, 30 (5), page 1357-1370, 10.1109/TFUZZ.2021.3058572
- [27] Wu, C.M., & Guo, X.K. (2022). A novel interval-valued data driven type-2 possibilistic local information c-means clustering for land cover classification. *International Journal of Approximate Reasoning*, 148, page 80-116, 10.1016/j.ijar.2022.05.007
- [28] Pal, N. R., Pal, K., Keller, J. M., & Bezdek, J. C. (2005). A possibilistic fuzzy c-means clustering algorithm. *IEEE Transactions on Fuzzy Systems*, 13(4), page 517-530. <https://doi.org/10.119/TFUZZ.2004.840099>.
- [29] Yu, R., Wang, J., Cheng, T.C.E., & Yu, P. (2024). Assessment of new energy industrial clusters: An MCDM approach using DEA and GEMS. *Expert Systems with Applications*, 252, 124231, <https://doi.org/10.1016/j.eswa.2024.124231>
- [30] Nielsen, P., Dahanayaka, M., Perera, H.N., Thibbotuwawa, A., & Kilic, D.K. (2024). A systematic review of vehicle routing problems and models in multi-echelon distribution networks. *Supply Chain Analytics*, 7, 100072, <https://doi.org/10.1016/j.sca.2024.100072>
- [31] Xiao, J., Liu, X., Zhang, H., Cao, Z., & Kang, L. (2024). The low-carbon vehicle routing problem with dynamic speed on steep roads. *Computers & Operations Research*, 169, 106736, <https://doi.org/10.1016/j.cor.2024.106736>
- [32] Cavaliere, F., Accorsi, L., Lagana, D., Musmanno, R., & Vigo, D. (2024). An efficient heuristic for very large-scale vehicle routing problems with simultaneous pickup and delivery. *Transportation Research Part E: Logistics and Transportation Review*, 186, 103550, <https://doi.org/10.1016/j.tre.2024.103550>
- [33] Alvarez, A., Cordeau, J.F., & Jans, R. (2024). The consistent vehicle routing problem with stochastic customers and demands. *Transportation Research Part B: Methodological*, 186, 102968, <https://doi.org/10.1016/j.trb.2024.102968>
- [34] Kou, S., Golden, B., & Bertazzi, L. (2024). Estimating optimal split delivery vehicle routing problem solution values. *Computers & Operations Research*, 169, 106714, <https://doi.org/10.1016/j.cor.2024.106714>

- [35] Parada, L., Legault, R., Cote, J.F., & Gendreau, M. (2024). A disaggregated integer L-shaped method for stochastic vehicle routing problems with monotonic recourse. *European Journal of Operational Research*, vol 318, issue 2, page 520-533, <https://doi.org/10.1016/j.ejor.2024.05.012>
- [36] Yu, V.F., Lin, S.W., Zhou, L., & Baldacci, R. (2024). A fast simulated annealing heuristic for the multi-depot two-echelon vehicle routing problem with delivery options. *Transportation Letter*, in press, <https://doi.org/10.1080/19427867.2023.2257923>
- [37] Rodriquez-Esparza, E., Masegosa, A.D., Oliva, D., & Onieva, E. (2024). A new hyper-heuristic based on adaptive simulated annealing and reinforcement learning for the capacitated electric vehicle routing problem. *Expert Systems with Applications*, vol 252, part B, 124197, <https://doi.org/10.1016/j.eswa.2024.124197>
- [38] Şimşir, F., & Ekmekçi, D. (2019). A metaheuristic solution approach to capacited vehicle routing and network optimization. *Engineering Science and Technology, an International Journal*, vol 22, issue 3, page 272-735, <https://doi.org/10.1016/j.jestch.2019.01.002>
- [39] Garside, A.K., Ahmad, R., & Muhtazaruddin, M.N.B. (2024). A recent review of solution approaches for green vehicle routing problem and its variants. *Operations Research Perspectives*, 12, 100303, <https://doi.org/10.1016/j.orp.2024.100303>
- [40] Wang, S., Zhu, X., Shang, P., Lin, X., Yang, L., & Tavasszy, L. (2024). Two-echelon multi-commodity multimodel vehicle routing problem considering user heterogeneity in city logistics. *Expert Systems with Applications*, vol 252, Part A, 124141, <https://doi.org/10.1016/j.eswa.2024.124141>
- [41] Fragkogios, A., Qui, Y., Saharidis, G.K.D., & Pardalos, P.M. (2024). An accelerated benders decomposition algorithm for the solution of the multi-trip time-dependent vehicle routing problem with time windows. *European Journal of Operational Research*, vol 317, issue 2, page 500-514, <https://doi.org/10.1016/j.ejor.2024.04.013>
- [42] Su, Y., Zhang, S., & Zhang, C. (2024). A lightweight genetic algorithm with variable neighborhood search for multi-depot vehicle routing problem with time windows. *Applied Soft Computing*, 161, 111789, <https://doi.org/10.1016/j.asoc.2024.111789>
- [43] Lehman, J., & Winkenbach, M. (2024). A matheuristic for the two-echelon multi-trip vehicle routing problem with mixed pickup and delivery demand and time windows. *Transportation Research Part C: Emerging Technologies*, 160, 104522, <https://doi.org/10.1016/j.trc.2024.104522>
- [44] Polimeni, A., & Vitetta, A. (2024) Network design and vehicle routing problems in road transport systems: Integrating models and algorithms. *Transportation Engineering*, 16, 100247, <https://doi.org/10.1016/j.treng.2024.100247>
- [45] Ferreira, K.M., Queiroz, T.A., & Munari, P. (2024). A variable neighborhood search for the green vehicle routing problem with two dimensional loading constraints and split delivery. *European Journal of Operational Research*, vol 316, issue 2, page 597-616, <https://doi.org/10.1016/j.ejor.2024.01.049>

- [46] Boz, E., & Aras, N. (2024). Bütünleşik sipariş gruplama ve zaman pencereli araç rotalama problem. *Gazi Üniversitesi Mühendislik ve Mimarlık Fakültesi Dergisi*, 39:4, 2223-2237
- [47] Karaboga, D. (2005). An idea based on Honey bee swarm for numerical optimization. (Vol. 200, pp. 1-10). Technical report-tr06, Erciyes University, Engineering Faculty, Computer Engineering Department.
- [48] Li, X., Zhang, S., & Shao, P. (2024). Discrete artificial bee colony algorithm with fixed neighborhood search for traveling salesman problem. *Engineering Applications of Artificial Intelligence*, 131, 107816, <https://doi.org/10.1016/j.engappai.2023.107816>
- [49] Ma, Y., Chen, J., Lv, W., Qui, X., Zhang, Y., & Liu, W. (2024). An improved artificial bee colony algorithm to minimum propagation latency and balanced load for controller placement in software defined network. *Computer Networks*, 250, 110600, <https://doi.org/10.1016/j.comnet.2024.110600>
- [50] Jin, Q., Zhang, S., Xu, J., Jiang, K., Cheng, Y., & Zhang, W. (2024). Remanufacturing system scheduling of batch products with the consideration of dynamic changes in machine efficiency using an improved artificial bee colony algorithm. *Computers & Industrial Engineering*, 187, 109817, <https://doi.org/10.1016/j.cie.2023.109817>
- [51] Anuradha, K., Krishna, M.V., & Mallik, B. (2024). Bio inspired Boolean artificial bee colony based feature selection algorithm for sentiment classification. *Measurement: Sensors*, 32, 101034, <https://doi.org/10.1016/j.measen.2024.101034>
- [52] Wang, L., Xie, L., & Xi, C. (2024). Improved artificial bee colony algorithm for pressure source parameter inversion of Sakurajima volcano from InSAR data. *Geodesy and Geodynamics*, In press, <https://doi.org/10.1016/j.geog.2024.05.004>
- [53] Zhang, B., Che, A., Wang, Y. (2024). Grid-based artificial bee colony algorithm for multi-objective job scheduling with manual loading and unloading tasks. *Expert Systems with Applications*, 245, 123011, <https://doi.org/10.1016/j.eswa.2023.123011>
- [54] Jiang, C., Yang, S., Nie, P., & Xiang, X. (2023). Multi-objective structural profile optimization of ships based on improved artificial bee colony algorithm and structural component library. *Ocean Engineering*, 283, 115124, <https://doi.org/10.1016/j.oceaneng.2023.115124>
- [55] Huang, Y., Sheng, B., Luo, R., Lu, Y., Fu, G., & Yin, X. (2024). Solving human-robot collaborative mixed-model two-sided assembly line balancing using multi-objective discrete artificial bee colony algorithm. *Computers & Industrial Engineering*, 187, 109776, <https://doi.org/10.1016/j.cie.2023.109776>
- [56] Wang, Z.C., Pan, Q.K., Gao, L., Miao, Z.H., & Sang, H.Y. (2024). A variable-representation discrete artificial bee colony algorithm for a constrained hybrid flow shop. *Expert Systems with Applications*, 254, 124349, <https://doi.org/10.1016/j.eswa.2024.124349>

- [57] Ren, Y., Gao, K., Fu, Y., Li, D., Suganthan, P.N. (2024). Ensemble artificial bee colony algorithm with Q-learning for scheduling Bi-objective disassembly line. *Applied Soft Computing*, 155, 111415, <https://doi.org/10.1016/j.asoc.2024.111415>
- [58] Mumtaz, J., Minhas, K.A., Rauf, M., Yue, L., Chen, Y. (2024). Solving line balancing and AGV scheduling problems for intelligent decisions using a genetic-artificial bee colony algorithm. *Computers & Industrial Engineering*, 189, 109976, <https://doi.org/10.1016/j.cie.2024.109976>
- [59] Li, Y.Z., Gao, K., Meng, L.L., & Suganthan, P.N. (2023). A problem-specific knowledge based artificial bee colony algorithm for scheduling distributed permutation flowshop problems with peak power consumption. *Engineering Applications of Artificial Intelligence*, vol 126, Part C, 107011, <https://doi.org/10.1016/j.engappai.2023.107011>
- [60] Wang, J., Liu, Y., Rao, S., Zhou, X., Hu, J. (2023). A novel self-adaptive multi-strategy artificial bee colony algorithm for coverage optimization in wireless sensor networks. *Ad Hoc Networks*, 150, 103284, <https://doi.org/10.1016/j.adhoc.2023.103284>
- [61] Guo, J., Li, Y., Du, B., & Wang, K. (2024). Multi-objective fuzzy partial disassembly line balancing considering preventive maintenance scenarios using enhanced hybrid artificial bee colony algorithm. *Expert Systems with Applications*, vol 27, Part C, 121681, <https://doi.org/10.1016/j.eswa.2023.121681>
- [62] Pan, X., Wang, Y., Lu, Y., & Sun, N. (2024). Improved artificial bee colony algorithm based on two-dimensional queue structure for complex optimization problems. *Alexandria Engineering Journal*, vol 86, page 669-679, <https://doi.org/10.1016/j.aej.2023.12.011>
- [63] Han, F., Li, X., Qi, S., Wang, W., & Shi, W. (2023). Reliability analysis of wing turbine subassemblies based on the 3-P weibull model via an ergodic artificial bee colony algorithm. *Probabilistic Engineering Mechanics*, 73, 103476, <https://doi.org/10.1016/j.probengmech.2023.103476>
- [64] Yang, M.C., Pan, Q.K., Sang, H.Y., Li, W.M., & Wang, Y.L. (2023). A slack speed-up based discrete artificial bee colony algorithm for resource-constrained operating room scheduling problem. *Computers & Industrial Engineering*, 186, 109760, <https://doi.org/10.1016/j.cie.2023.109760>
- [65] Cui, Y., Hu, W., & Rahmani, A. (2024). Multi-robot path planning using learning-based artificial bee colony algorithm. *Engineering Applications of Artificial Intelligence*, 129, 107579, <https://doi.org/10.1016/j.engappai.2023.107579>
- [66] Sahani, S., Barman, T., Sarkar, B., Gunasekaran, A., & Pareek, S. (2024). Retail management policy through firefly algorithm under uncertainty using Dempster-Shafer theory for production firm. *Journal of Retailing and Consumer Services*, vol 79, 103760, <https://doi.org/10.1016/j.jretconser.2024.103760>
- [67] Anshory, I., Jamaaludding, J., Wisaksono, A., Sulistiyowati, I., Hindarto, Rintyarna, B.S., Fudholi, A., Rahman, Y.A., & Sopian, K. (2024). Optimization DC-DC boost converter of BLDC motor drive by solar panel

- using PID and firefly algorithm. *Results in Engineering*, vol 21, 101727, <https://doi.org/10.1016/j.rineng.2023.101727>
- [68] Zhao, J., Lv, S., Xiao, R., Ma, H., & Pan, J.S. (2024). Hierarchical learning multi-objective firefly algorithm for high-dimensional feature selection. *Applied Soft Computing*, vol 165, 112042, <https://doi.org/10.1016/j.asoc.2024.112042>
- [69] Yuan, T., Mu, Y., Wang, T., Liu, Z., & Pirouzi, A. (2024). Using firefly algorithm to optimally size a hybrid renewable energy system constrained by battery degradation and considering uncertainties of power sources and loads. *Heliyon*, vol 10, issue 7, e26961, <https://doi.org/10.1016/j.heliyon.2024.e26961>
- [70] Kumar, S.S., Karthick, A., & Shankar, R. (2024). Energy forecasting of the building integrated photovoltaic system based on deep learning Dragonfly-firefly algorithm. *Energy*, 132926, <https://doi.org/10.1016/j.energy.2024.132926>
- [71] Ahsan, M.B.F., Mekhilef, S., Soon, T.K., Usama, M., Mubin, M.B., Seyedmahmoudian, M., Stojcevski, A., Mokhlis, H., Shrivastava, P., & Alshammari, O. (2024). Optimal sizing and cost analysis of hybrid energy storage system for EVs using metaheuristic PSO and firefly algorithms. *Results in Engineering*, vol 23, 102358, <https://doi.org/10.1016/j.rineng.2024.102358>



EKLER

EK-1

```
{
  "nbformat": 4,
  "nbformat_minor": 0,
  "metadata": {
    "colab": {
      "provenance": []
    },
    "kernelspec": {
      "name": "python3",
      "display_name": "Python 3"
    },
    "language_info": {
      "name": "python"
    }
  },
  "cells": [
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "ZVRo4AZGX0xZ",
        "outputId": "96907bcb-58b9-4bb0-95f4-b913a9f32243"
      },
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "Mounted at /content/drive\n"
          ]
        }
      ],
      "source": [
        "\n",
        "# Google Drive Bağlantısı\n",
        "from google.colab import drive\n",
        "drive.mount('/content/drive')\n"
      ]
    },
    {
      "cell_type": "code",
      "source": [
        "import pandas as pd\n",
        "import numpy as np\n",
        "import matplotlib.pyplot as plt\n",

```

```

"from sklearn.cluster import KMeans\n",
"from scipy.spatial.distance import cdist\n",
"import random\n",
"\n",
"# Google Drive'dan dosyaları yükleme\n",
"mesafe_path =
\"/content/drive/MyDrive/end_/Mesafe.xlsx\"\n",
"siparis_path =
\"/content/drive/MyDrive/end_/siparis.xlsx\"\n",
"depo_path =
\"/content/drive/MyDrive/end_/Depo.xlsx\"\n",
"\n",
"mesafe_df = pd.read_excel(mesafe_path)\n",
"siparis_df = pd.read_excel(siparis_path)\n",
"depo_df = pd.read_excel(depo_path)\n",
"\n",
"# Bayi ID'lerini al\n",
"bayi_ids = mesafe_df.iloc[:, 0].values # 2'den 60'a
kadar bayi ID'leri\n",
"mesafe_mat = mesafe_df.iloc[:, 1:].values # 59x59
mesafe matrisi\n",
"\n",
"# Depo mesafesini doğru indeksleme ile alalım\n",
"depo_mesafeleri = depo_df.iloc[0, 2:].values # Depodan
2'den 60'a kadar bayilere mesafeler\n",
"\n",
"# Kümeleri belirlemek için Elbow yöntemi\n",
"def elbow_method(data):\n",
"    distortions = []\n",
"    K = range(1, 11)\n",
"    for k in K:\n",
"        kmeans = KMeans(n_clusters=k, random_state=0,
n_init=10)\n",
"        kmeans.fit(data)\n",
"        distortions.append(sum(np.min(cdist(data,
kmeans.cluster_centers_, 'euclidean'), axis=1)) /
data.shape[0])\n",
"    plt.plot(K, distortions, 'bx-')\n",
"    plt.xlabel('Küme Sayısı')\n",
"    plt.ylabel('Distortion')\n",
"    plt.title('Elbow Yöntemi')\n",
"    plt.show()\n",
"\n",
"# Kullanıcıdan küme sayısını al\n",
"elbow_method(mesafe_mat)
],
"metadata": {
"colab": {
"base_uri": "https://localhost:8080/",
"height": 472
},
"id": "KFqRo53H0bjX",
"outputId": "ffd37ee2-ea83-406d-9221-8c7ae511446f"
},
"execution_count": null,
"outputs": [
{
"output_type": "display_data",

```

```

"data": {
  "text/plain": [
    "<Figure size 640x480 with 1 Axes>"
  ],
  "image/png":
    "iVBORw0KGgoAAAANSUhEUgAAAJsAAAHHCAYAAABZbpmkAAAAOnRFWHRTb2Z0d2Fy
    ZQBNYXRwbG90bGliIHZlcnNpb24zLjEwLjAsIGh0dHBzOi8vbWF0cGxvdGxpYi5vc
    mcvlHJYcGAAAAAwSFlzAAAPYQAAD2EBQD+naQAAVIRJREFUeJzt3XlcVXX+x/HXZR
    VBQEHBFJXctzTTUVwyjdwl0zLLjPamXHOZtN+IW41LZWm5jNVozbbOWO5lqLjhnkt
    lZrmRBo4pIJaIch5/nOHqDSxA4FzufT8fj/PgnoVzPzdmuu/Od7MZhmEgIiIi4qI8
    rC5AREREpCQp7IiIiIhLU9gRERERl6awIyIiIi5NYUdERERcmsKOiIiIuDSFHRERE
    XFpCjsiIiLi0hR2RERExKUp7IjIH7LZbEycONG+P3HiRGw2G2fOnLGuqOsUEBDAQw
    89BEDPnj2pWbOmpfVcr0WLFmGz2Th27JjVpYg4JYUdETeU++V4rW3btm1Wl1hoiYm
    JeHh4MG7cuHzPT58+HZvNxsqVK3nrrbd48sknARg9ejSvvpqkd9369atTJw4kdTU
    1CLfQ0RKlpfVBYiIdSZPnkxUVFSe47Vr17agmusTHR3Nk08+ycsvv8wDDzxAo0aN7
    OeOHZ/O5MmTueeee+jRo4fD7912223X9b5bt25l0qRJPPTQQwQHB1/XvYpq0KBBDB
    gwAF9fX0veX8TZKeyIuLFu3brRokULq8sOnTOMTeOzzz7jySefZNOMtdhsNgCGDh2
    Kt7c3s2bNsrjCkuHp6Ymnp6fVZYg4LTvjIUiRnDlzhv79+xMYGEhoaCjDhw/n4sWL
    DtdcvnyZKVOMUKtWLXx9falZsybPPfccmZmZ9mtGjhxJaGgohmHYjw0d0hSbzcbs2
    bPtx1JSURDZbMybN++aNUFBTFrliY2bNnCM2++CcDSpUtZvvnw506ZNo0qVKly4cI
    FRo0YRGRmJr68v9erV46WXXnJ4fzD7Kg0ZMoRPP/2Uxo0b4+vrS6NGjVizZo39mok
    TJzJmzBgAoqKi7M2AV/edWbx4Mbfcctg+fn6EhIQwYMAAkPKSHN7rtttuo3Hjxuzf
    v58OHTpQvnx5ateuzSeffaJAQkICrVqlws/Pj3r16vHl1186/L767Ij8MYUdETeWl
    pbGmTnNHLZffvmlQL/bv39/L168yNSpU+nevTuzZ8/miSeecLjmscceIy4ujubNm/
    PKK6/QoUMHpK6dyoABA+zXtG/fnrNnz/LNN9/Yj23atAkPDw82bdrkcAzglltv/cO
    6cpuqnn32WY4cOcLw4cNp06YNTz75JIZh0Lt3b1555RW6du3KzJkzqVevHmPGjGHk
    yJF57rV582aefvppBgwYwIwZM7h48SL9+vWz/zPq27cv9913HwCvvPIK7777Lu+++
    y6VK1UC4IUXXuDBBx+kTp06zJw5kxEjRhAfH8+tt96ap4/PuXPn6NmzJ61atWLGjB
    n4+voyYMAAPvzwQwYMGED37t2ZNM0aFy5c40677+b8+fn/9icSkVyGiLidhQsXGkC
    +m6+vr80lgDFhwgT7/oQJEwzA6N27t8N1Tz/9tAEY+/btMwzDMPbu3WsAxmOPPeZw
    3ejRow3AWLdunWEYhnH69GkDMObOnWsYhmGkpqYaHh4exj333GOEHYXZf2/YsGFGS
    EiIkZOT86ef79ixY4a/v78REhJieHt7GwcOHDAMwzA+/fRTAzCef/55h+vvvvwtu2
    azGT/88IPD5/bx8XE4tm/fPgMwXnvtNfuxF1980QCMo0eP5qnB09PTEOGFFxyOHzh
    wwPDy8nI43qFDBWmWlixZYj/23XffGYDh4eFhbNu2zX78888/NwBj4cKF9mO5f8/f
    1yAiJj3ZEXFjc+bMYe3atQ7b6tWrC/S7gwcPdtgfOnQoAKtWrXL4+fsnJqNGjQJg5
    cqVAFSsqVIn69euzceNGALZs2YKnpYdjxowhJSWFw4cPA+aTnXbt2tn74fyRGjVqMG
    HCBM6ePcvIkSNp3LixvSZPT0+GDRuWpybDMPJ89piYGGvQmXfv+mmmwgMDOTIkSN
    /WsN//vMfcNjY6N+/v80Ts/DwcOrUqcP69esdrG8ICHB44lWvXj2Cg4Np0KABrVq1
    sh/PfV2QGkTEPa7KIm7sL3/5S5E7KNepU8dhv1atWnh4eNj7jRw/fhwPD488I7vCw
    8MJDg7m+PHj9mPt27e3h6NNmzbRokULWrRoQUhICJs2bSIsLIx9+/Zx//33F7i+li
    1bAjh8vuPHjxmREUGFChUcrm3QoIH9/NWqV6+e574VK1bk3Llzf/r+hw8fxjCMPP+
    ccnl7ezvsV6tWLU+QCwoKIjIyMs8xoEA1iHJYUdEisWlnrgU5ElMu3bteOONNzhy
    5AibNm2iffv22Gw22rVrx6ZNM4iIiCanJ4f27dsXd9l/6FoJnIzfdWbOT05ODjabj
    dWrV+d7n4CAgAK91/XUICImhR0RKZLDhw87zNHzzw8/kJOTY5+NueaNGuTk5HD48G
    H7kxMwR1WlpqZSo0YN+7HcELN27Vp27tzJ2LFjAbMz8rx584iIIMdf359bbrnlumq
    uUaMGX375JefPn3d4uvPdd9/ZzxftwCJcrVq1MAyDqKgo6tatW7SCRaRYqM+OiBTJ
    nDlZHPZfe+01wJy7B6B79+4AeWYnnjlzJoDD5H5RUVFUrVqVV155haysLNq2bQuYI
    ejHH3/kk08+oXXrlnh5Xd9/n3Xv3p3s7Gxeff/11h+OvvPIKNpvNXnth+Pv7A+QZXd
    W3b188PT2ZNGlSnqcwhmEUeNSbiFw/PdkRcWOrV6+2P9W4Wps2bbjxxhv/8HePHj1
    K79696dq1K4mJiSxevJj777+fpk2bAtC0aVNiY2NZsGABqampdOjQgR07dvD222/T
    p08fOnbs6HC/9u3b88EH9CkSRMqVqVqIQQPmzfH39+f7778vH+da+nwYqxcd03bk/
    /7v/zH27BhNmzbliY++4LPPPmPEiBEONZELKvdp0//93/8xYMAAvL296dWrf7Vq1e
    L5559n3LhxHdt2jd59+lChQgWOHj3K0qVLEKJJxg9evR1fyYR+XMKOYJuLC4uLt/
    jCxcu/Now8+GHHxIXF8fYsWPx8vJiyJAhpVjiiw7XvPnm9x4440sWrSIPuXEH4e
    zrhx45gwYUKe++WGnXbt2tmPeX15ER0dzZdfflks/XU8PDxYtmwZcXfXfPjhhycu
    JCaNWvy4osv2keJFVbLli2ZMmUK8+fPZ82aNeTk5HD06FH8/f0ZO3YsdevW5ZVXXm
    HSpEkAREZG0rlzZ3r37n3dn0dECsZmqJebiIiIuDD12RERERGXprAjIiIiLklhR0R
    ERfYawo6IiIi4NIUdERERcWkKOyIiIuLSNM805ho2p06dokKFCgVax0dERESsZxgG

```

58+fJyIiAg+Paz+/UdgBTp06lWdlYRERESkbpKSqFat2jXPK+yAfUHApKQkAgMDL
a5GRERECiI9PZ3IyEiHhX3zo7DDLWLAwMDFXZERETKmD/rgqIOyiIiIuLSFHZERE
TEpSnsiIiIiEtT2BERERGXprAjIiIiLklhR0RERFyawo6IiIi4NIUdERERcWkKOyI
iIuLSFHZERETEpVkadqZOnUrLli2pUKEClStXpk+fPhw6dMjhmosXLzJ48GBCQ0MJ
CAigX79+pKSkOfxz4sQJevToQfny5alCuTJjxozh8uXLpflR7CZOhClT8j83ZYp5X
kREREqPpWEnISGBWYMHs23bNtauXUtWVhadO3fmwoUL9mueeeYZli9fzscff0xCQg
KnTp2ib9++9vPZ2dn06NGDS5cusXXrVt5++20WLVpEXFycFR8JT0+Ii8sbeKZMMY9
7elpSloiIiPsynMjp06cNwEhISDAMwzBSU1MNB29v4+OPP7Zfc/DgQQMwEhMTDcMw
jFWrVhkeHh5GcnKy/Zp58+YZgYGBRmZmZoHeNy0tzQCMtLS0YvkckycbBhjGxImGk
ZhoGHFx5v7kycVyexERETEK/v3tVH120tLSAAgJCQFg9+7dZGVlERMTY7+mfV36VK
9encTERAASEXnp0qQJYWFh9mu6dOlCeno633zzTS1Wf8X48TB5stlkFR1tvp482Tw
uIiIipctpwk50Tg4jRoygbdu2NG7cGIDk5GR8fHwIDg52uDYsLIzk5GT7NVcHndzz
uefyk5mZSXp6usNW3MaPB4///dP18FDQERERsYrThJ3BgwFz9ddf88EHH5T4e02dO
pWgoCD7FhkZWezvMWUK5OSYr3NyzCc7IiIiUvqcIuwMGTKEFStWsh79eqpVq2Y/Hh
4ezqVLl0hNTXW4PiUlhfDwcPslvx+dlbufe83vjRs3jrSONPuWlJRujJ/mSmfk8eO
hXDnz2IQJ1x6lJSIiIiXH0rBjGAZDhgXh6dKlrfu3jqioKIfzt9xyC97e3sTHx9uP
HTp0iBMnThAdHQ1AdHQ0Bw4c4PTp0/Zrlq5dS2BgIA0bNsz3fX19fQkMDHTYiktu0
Mntp3P77ebxmJj8R2mJiIhIyfyKy8s0HDx7MkiVL+Oyzz6hQoYK9j01QUBB+fn4EBQ
Xx6KOPMnLkSEJCQggMDGT00KFER0fTunVrAdp37kzDhg0ZNGgQM2bMIDk5mb///e8
MHjwYX1/fUv9M2dmOnZF79oSVK+HXX83j2dmlXpKIiIhbsxmGYVj25jZbvscXLLzI
Qw89BJiTC04aNYr333+fzMxMunTpwtY5cx2aqI4fP85TTz3Fhg0b8Pf3JzY2lmnTp
uHlVbAsl56eTlBQEGlpacX6lAcgKQmqVzc7KaekwA03FOvtRURE3FZBv78tDTvOoi
TDDkCzZrBvH7z7LjzWQLHfXkRExC0V9PvbKTooU7qePc2fK1ZYW4eIiIg7UtgpBbl
hZ80ayMqythYRERF307BTCLq2hEqVIC0Ntmyxuh0RERH3orBTCjw90xt387WaskRE
REqXwk4p6dHD/KmwIyIiUroUdkpJ587g5QWHDsHhw1ZXIyIi4j4UdkpJUBDceqv5e
uVKA2sRERFfxJwo7pUHD0EVEREQfwk4pyg07CQmQnm5tLSIiIu5CYacUlakDdevC5c
vwXrdWVyiIuIeFHZKmqYRERESpFctinLDTurVmKfDBERkdKgsFpk2rWDwED4739
h506rqxEREXF9CjulzNsbunY1X6spS0REpOQp7FhA/XZERERKj8KOBbplA5sN9u2D
pCSrqxEREXFtCjsWuOEGiI42X2s2ZRErkZKlsGMRNWWJiIiUD0Udi+SGnfh4+PVXA
2sRERFfxZQo7FmncGKpXh4sXYd06q6sRERFxxQo7FrHz1JQlIiJSGhR2LHR12DEMa2
sRERFfxVQo7FurYECqXh5MnzWHOIiIiUvUdixUrhzExJiv1ZQlIiJSMhR2LKZ+OyI
iIiVLYcdi3bubP3fsgJQUa2sRERFfxRQo7FqtaFzo3NzSor15tdTUiIiKuR2HHCeQ2
ZWnpCBERkeKnsOMEcsPO55/DpUvWliIiIuJqFHacwC23QFgYnD8PmzZZXY2IiIhrU
dhxAh4e0KOH+VqjskRERIqXwo6TyG3KWR5csymLiIgUJ4UdJxETAz4+8OOP8P33Vl
cjIiLiOiWNOxs3bqRXr15ERERgs9n49NNPHc5nZGQwZMgQqlWrhp+fHw0bNmT+/Pk
O1ly8eJHBgwcTGhpKQEAA/frlI6UMTlhToQLcdpv5WklZiIiixcfSsHPhwgWaNm3K
nDlz8j0/cuRIlqxZw+Lfizl48CAjRoxgyJAHLfu2zH7NM888w/Lly/n4449JSEjg1
KlT903bt7Q+QrHSbMoiIiLFz2YYztFDxGazsXTpUvr06WM/1rhxY+69917Gjx9vP3
bLLbfQrVs3nn/+edLS0qhUqRJLlizh7rvvBuC7776jQYMGJCYm0rp16wK9d3p6OkF
BQaSlpREYGFisn6swjhyBwRXA0xPONIHgYmtKERERcXoF/f526j47bdq0YdmyZZw8
eRLDMFi/fj3ff/89nTt3BmD37t1kZWURk7uaJlC/fn2qV69OYmKiVWUX2Y03QoMGk
JltzrkjIiIi18+pw85rr71Gw4YNqVatGj4+PnTt2pU5c+Zw6623ApCcnIyPjw/Bv3
sEEhYWRnJy8jXvm5mZSXp6usPmLNSUJSIiUrycPuxs27aNZcuWsXv3bl5++WUGDx7
Ml19+eV33nTp1KkFBQfYtMjKymCq+frlhZ9Uq8wmPiIiIXB+nDTu//fYbzz33HDNn
zqRXr17cdNNNDBkyhHvvvZeXXnoJgPDwcC5dukRqaqrD76akpBAeHn7Ne48bN460t
DT7lpSUVJIfpVdatDH76pw9C9u2WV2NiIhI2ee0YScrK4usrCw8PBxL9PT0JCCnBz
A7K3t7exMfH28/f+jQIU6cOEF0dPQ17+3r60tgYKDD5iy8vKBbN/OlmrJERESun5e
Vb56RkcEPP/xg3z969Ch79+4lJCSE6tWr06FDB8aMGYOfnx81atQgISGBd955h5kz
ZvIQFBTEo48+ysiRIwkJCSEwMJChQ4cSHR1d4JFYzqhnt3j/fTPStJlqdtUiIiJlm
6VDzZds2EDHjh3zHI+NjWXRokUkJyczbw4vvjiC86ePUuNGjV44okneOaZZ7DZbI
A5qeCoUaN4//33ycZMpEuXLSydO/cPm7F+z1mGnuc6exYqVYKcHDh6FGRWtLoiERE
R51PQ72+nmWfHSS4WdgBuvdVcAf312HwYKURERERcT4uMc+OO9MQdBERkeKhsOOk
csPOunWQkWftLSIiImWZwo6TatAAoqLg0iW4arCZiIiIFJLCjpOy2dsUJSiIUhWud
pxYbthZudIcmSUiIiKfP7DjxDp0AH9/+Pln+Oorq6sREREpmxR2nJivL/xvgXclZY
mIiBSRwo6TU78dERGR66Ow4+S6dZd/7tplNmeJiIhI4SjsOLnwcGjZ0ny9apWltYi
IiJRFCjtlgJqyREREik5hpwzIDTtr10JmPrWliIiIldUKO2XAZtdDRARcuAAJCvZX
IyIiUrYo7QBNhv06GG+VlOWiIhI4SjslBFX99sxDGtrERERKUsUdsqI2283Jxk8e
hQOHrS6GhERkbJDYaeM8PeHTp3Ml2rKEHERKtiFnTJEQ9BFREQKT2GnDMntpLxlC5

w9a20tIiIiZYXCThlSowY0aQI5ObBmjdxViIiI1A0KO2WmmrJEREQKR2GnjMkNO6t
Xw+XLltYiIiJSFijslDGtWkFoKKSmtatVlcjIiLi/BR2yhhPT+je3XytpiwREZE/
p7BTBqnfjoiISMEp7JRBnTubT3gOHOqff7S6GhEREeemsFMGBQdD+/bm65UrLSlFR
ETE6Sns1FFqyhiRESkYhZ0yKjfsbNgA589bWoqIiIhTU9gpo+rWhdq1ISsL1q61uh
oRERHnpbBTRtlasasoSEREpCIWdMiw37Kxcaa6XJSIiInkp7JRh7dtDhQpw+jTs2mV
1NSIiIs7J0rCzceNGevXqRUREBDabju8//TPNQcPHqR3794EBQXh7+9Py5YtOXHi
hP38xYsXGTx4MKGhoQQEBCvXz9SULJK8VNYx8cHunQxX6spS0REJH+Whp0LFy7Qt
GlT5syZk+/5H3/8kXbt2lG/fn02bnJa/v37GT9+POXKlbnf88wzz7B8+XI+/vhjEh
ISOHXqFH379i2tj2A59dsRERH5YzbDMAyriwCw2WwsXbqUPn362I8NGDAAb29v3n3
33Xx/Jy0tjUqVKrFkyRLuvvtuAL777jsaNGhAYmIirVu3LtB7p6enExQURFpaGoGB
gdf9WUrT6dMQHg6GAT/9BFWrWl2RiIhI6Sjo97ft9tnJyclh5cqV1K1bly5dulC5c
mVatWrl0NSle/dusrKyImJsR+rX78+latXJzEx0YKqS1/lyuZK6KDZlEVERPLjtG
Hn9OnTZGRkMG3aNLp27coXX3zBXXfdRd++fU1ISAAgOTkZHx8fgoODHX43LCyM5OT
ka947MzOT9PR0h60sU1OWiIjItTlt2Mn531jqO++8k2eeeYZmzZoxduxYevbsyFz5
86/r3lOnTiUoKMi+RUZGFkfJlskNO19+Cb/9Zm0tIiIizsZpw84NN9yAl5cXDRs2d
DjeoEED+2is8PBwLl26RGpqqsm1KSphIeHX/Pe48aNIy0tzb4lJSUve/2l6aaboF
olM+isX291NSIiIs7Fac00j48PLVu25NChQw7Hv//+e2rUqAHALbfcgre3N/Hx8fb
zhw4d4sSJE0RHR1/z3r6+vgQGBjpsZdnVsymr346IiIgjLyvfPCMjgx9++MG+f/To
Ufbu3UtISAjVqldnzJgx3Hvvvdx666107NiRNWvWshZ5cjZs2ABAUFAQjz76KCNHj
iQkJITAwECGDh1KdHR0gUdiuYqePWH+fLPfzuuvmwFIRERELB56vmHDBjp27JjneG
xsLIswLQLgX//6F1OnTuWnn36iXr16TJo0iTvvnN+7cWLFxklahTvv/8+mZmZdOn
Shblz5/5hM9bvleWh57l++w1CQ82f+/dDkyZWVYQiIlKyCvr97TTz7FjJfCIOQK9e
5pOdf/wDxo2zuhoREZGSVebn2ZHC0xB0ERGRvBR2XEiPHubPxEQ4c8baWkRERJyFw
o4LqVYNmjUzl45YvdrqakRERJyDwo6LUVOWiIiI4UdF5Mbdtagawsa2sRERFxBg
o7LqZlS6hUCdLTYfNm6sRERGXnsKOi/HwuNJRWU1ZiIiIcjsuSf12RERERlDYcUF
33AHe3vD99+YmIiLizhr2XFBgIHToYL7WwqAiIuLuFHZclJqyRERETAo7Lio37Gzc
CGlpltyIiIjIjYUdF1WrFtSvD5cvwxdfWF2NiIIdRR2XJiGoIuIiIcjsuLTcpqxVq
yA729paRERERKKw48LatoWgIHMF9B07rK5GRETEGgo7LszbG7p2NV+rKUteRNyWwo
6L0xB0ERFxdwo7Lq5rV309rP374cQJq6sREREPfQo7Lu6GGyA62nyt2ZRFRMQdKey
4ATVliYiIOlPYcQO5YSc+Hi5csLYWERGR0qaw4wYaNYIaNSAZe9ats7oaERGR0qWw
4wZsnJvliYiI+1LYcRO5YwflSjAMa2sREREPtQo7buK226B8eTh5Evbts7oaERGR0
qOw4ybKlYM77jBfqylLRETcicKOG1G/HRERcUcK026ke3fz544dkJJibS0iIiKlRW
HHjUREwC23mB2UV6+2uhoREZHSobDjZtSUJSIi7kZhx83khp3PP4dL16ytRUREpDQ
o7LiZ5s0hPBwyMmDjRqurERERKXmWhp2NGzfsq1cvIiIisNlsfPrpp9e89q9//Ss2
m41XX33V4fjZs2cZOHAggYGBBACH8+ijj5KRkVGyhZdhHh7Qo4f5Wk1ZiIiLiDiwNO
xcuXKBp06bMmTPnD69bunQp27ZtIyIiIs+5gQMH8s0337B27VpWrFjBxo0beekJJ0
qqZJeQ25S1fLlmUxYREdfnZeWbd+vWjW7duv3hNSdPnmTo0KF8/vnn9Mh9JPE/Bw8
eZM2aNezcuZMWLVoa8Npr9G9e3deeu1fMORQEwM+PjAKSNw6BDUr291RSIiIiXH
qfvs50TkMGjQIMaMGUojRo3ynE9MTCQ4ONgedABiYmLw8PBg+/bt17xvZmYm6enpD
ps7CQiajh3N12rKEhERV+fUYWf69014eXkxbNiwm8nJydTuXJlH2NeXl6EhISQnJ
x8zftOnTqVoKAg+xYZGVmsdZcFGoIuIiLuwmnDzu7du5klaxaLfi3CZrMV673HjRt
HWlqafUtKSirW+5cFuS2CmzfDuXPWliIiI1KSnDbsbNq0idOnT109enW8vLzw8vLi
+PHjjBolipolawIQHh706dOnHX7v8uXLnD17lvDw8Gve29fxl8DAQIfn3URFQANGk
JltzrkjIiLiQpw27AwaNIj9+/ezd+9e+xyREcGYMWP4/H/fztHR0aSmprJ7927776
1bt46cnBxatWplVellhpqyRETEHVg6Gisji4MffvjBvn/06FH27t1LSEGIlatXJzQ
010F6b29vwsPDqVevHgANGjSga9euPP7448yfp5+srCyGDBnCGAEDNBKRAhr2hOnT
zXWyLl8GL0v/lyAiI1IyLH2ys2vXLm6++WZuvvlMAEaOHMnNN99MXFxcge/x3nvvU
b9+fw6//Xa6d+9Ou3btWLBgQUmV7FJat4aQEDh7FrZts7oaERGRkmEzjMJPk5eSks
Lo0aOJj4/n9OnT/P4W2dnZxVZgaUHPtycoKIi0tDS367/zwAPw3nvw7LMwbZrV1Yi
IiBRcQb+/i9Rw8dBDD3HixAnGjx9PlSpVin20lJSenj3NsLNihcKOiIi4piKfnc2b
N7Np0yaaNWTWzOViaevSBTW94Ztv4OhRc5SWiIiIKylSn53IyMg8TVdSNLWSCG3bm
q9XrrS2FhERKZJQpLDz6quvMnbsWI4d01bM5YgVNARdRERcWZE6KFesWJFff/2Vy5
cvU758eb9vR30nz17ttgKLA3u3EEZ4OBBANjQXBz011/MtbNEREScXYL2UH711Ve
LWpc4ofr14cYbzVXQv/wS+vSxuiIREZHiU6SwExsbW9xliIVsNrMpa/ZssyLLYUdE
RFxJkefMzc705tNPP+XgwYMANGrUiN69e+Pp6VlsxUnpyQ07K1dCTg54001CiIiI
oVTPd47P/zwa927d+fkyZP2pRsOHTPeZGQkK1eupFatWsVeaEly9z47AJmZcMMNkJ
EBO3dCixZWVYQiIvLHCvr9XaT/fh82bBi1atUiKSmJPXv2sGfPhk6cOEfUVBTdhg0
rctFiHV9f6NzZfK1RWSIi4kqKFHYSEhKYMWMGISEh9mOhoaFMmzaNhISEYitOSlfu

EHTNTyMiIq6kSGHH19eX8+fP5zmekZGBj4/PdRcl1uje3fy5axf8/LO1tYiIiBSXI
oWdnj178sQTT7B9+3YMw8AwDLZt28Zf//pXevfuXdw1SikJC40//MV8vWqVtbWIIi
gUlyKFndmzZ1OrVi2io6MpV64c5cqVo23bttSuXZtZs2YVd41SijSbsoiIuJoijcb
KdfjwYb777jsAGjRoQO3atYutsNKK0VhXfPUVNG80/v5w5gyUK2d1RSiIiVkr0RmU
c9WpU4c6depczy3EyTRrBhERcOoUJCSYq6KLiIiUZQUOoYNHjmTKlCn4+/szcuTIP
7x25syZ112YWCN3NuUFC8ymLIUdEREp6wocdr766iuySrLsr8V1XR12Zs82A5CiE
hZdV19dlyF+uw4+vVXCA2Fixfh66+hUSOrKxIREcmrRGdQfuSRR/KdZ+fChQs88sg
jRbm1OJHy5aFTJ/O1RmWJiEhZV6Sw8/bbb/Pbb7/1Of7bb7/xzjvvXHdRYj0NQRcR
EVdRqNFY6enp9kkEz58/T7mrxiVnZ2ezatUqKleuXOxFSunr0cP8uXUr/PKL2awlI
iJSFhUq7AQHB2Oz2bDZbNStWzfPeZvNxrJk4qtOLF09epw002wzfz+sWQMDB1pdkY
iISNEUKuysX78ewzDo1KkT//73vx0WAvXx8aFGjRpEREQUe5FijZ49zbCzYoXCjoi
l1F2FCjsdOnTg8uXLxMbG0qJFCyIjI0uqLnECPXvCP/5hPtnJygJvb6srEhERKbxC
d1D28vLik08+ITs7uyTqESfyl7/ADTDaaqrZd0dERKQsKtJorE6dOpGQkFDctYgTm
TjRfKrTvbU5f/WorClTzPMiIiJlQZHWxurWrRtjx47lwIED3HLLLfj7+zuc7927d7
EUJ9bx9IS400jf39xfsQJefNEMOnFxmHmytfWJiIgUVJFmUPbwuPYDIzVNVuaauDS
Dcv5yg42HB+TkWPDhMGUWGXTGj7e60hERcXcluup5Tk5OkQuTsiM30MTFmT9nzYJx
4xR0RESkbClSnx1xH+PHO47C+uAD2LvXsnJEREQKrchhJyEhgV69elG7dm1q165N7
9692bRpU3HWJk5gyhTHYedHj0J0NGhVEBERKSuKFHYWL15MTEwM5cuXZ9iwYQwbNg
w/Pz9uv/12lixZUtwlikWu7ox86ZLZhAXmauixsTBkiHlcRETEqRlFUL9+fWpMzJl
5jr/88stG/frlC3yfhIQEo2fPnkaVKlUMWFi6dKn93KVLl4y//elvRuPGjY3y5csb
VapUMQYNGmScPHnS4R6//PKLcf/99xsVKlQwgoKCjEceecQ4f/58oT5PWlqaARhpa
WmF+j1XNnmyYYD582qTJpnHc7c2bQzjd38SERGRUlhQ7+8iPdk5cuQIvXrlynO8d+
/eHD16tMD3uXdhAk2bNmXOnDl5zv3666/s2bOH8ePHs2fPhv7zn/9w6NChPMPaBw4
cyDffffMPatWtZsWIFGzdu5Iknnij8hxIH2dn5j7rKfdJz330QFGRONnjLLaAWTBER
cVpFSVKlatUy5s+fn+f4vHnzjNqlaxfllnme7ORnx44dBmAcP37cMAzD+Pbbbw3A2
Llzp/2alatXGzabLc8ToD+iJztFc/iwYTRubD7h8fIyJfMzDCMnx+qqRETEXRT0+7
tIQ89HjRrFsGHD2Lt3L23atAFgy5YtLFq0iFmzZhVXDssjLS0Nm81GcHAWAImJiQQ
HB90iRQv7NTEMXh4eLB9+3buuuuufO+TmZlJZmamfT89Pb3EanZltWvDtm3w+OPw
/vvmPDzbt8Mbb0D58lZXJyIiYipS2HnqqacIDw/n5Zdf5qOPPgKgQYMGfPjhh9x55
53FWmCuixcv8uyzz3LffffzJw5Ktk6mcuXKdtd5eXkREhJCcnLyNe81depUJk2aVC
Jluht/f3jvPXMdrdGjYckS+Ppr+M9/ofYtq6sTERG5jqHnd911F5s3b+aXX37hl19
+YfPmzSUWdLKysujfvz+GYTBv3rZrvt+4ceNIS0uzb0lJScVQpfuy2WDECFi3DsLC
YP9+aNECVq2yujIREZEihp0bb7yRX375Jc/x1NRUbrzxXusu6mq5Qef48eOsXbvWY
Tro8PBwTp8+7XD95cuXOXv2LOHh4de8p6+vL4GBgQ6bXL9bb4Xdu6F1a30l9J49Yd
Ikc6kJERERqxQp7Bw7dizf9a8yMzM5efLkdReVKzfoHD58mC+//JLQ0FCH89HR0aS
mprJ79277sXXrlpGtK0OrVq2KrQ4puKpVISEBnn7aHJw+cSL07g3nz1ldmYiIuKtC
9dlZtmyZ/fXnn39OUFCqfT87O5v4+HhqlqxZ4Pt1ZGTwww8/2PePHj3K3r17CQkJo
UqVKtx9993s2bOHFStWkJ2dbe+HEXISgo+PDw0aNBBr1648/vjjzJ8/n6ysLIYMGc
KAAQOIiIgozEeTYutJA3PmmP14/vpXWLkSWrY0+/HcdJPVlYmIiLsplKrnuaud22w
2fv9r3t7e1KxZk5dffpmePXsW6H4bNmygY8eOeY7HxsYyceJEOqKi8v299evXc9tt
twFw9uxZhgWzWvLly/Hw8KBfv37Mnj2bgICagn4srXpegr76Cvr2hWPHwM8P3nwT7
r/f6qpERMQVFPT7u1BhJ1dUVBQ7d+7khhtuuK4inYXCTsn65Rcz4Hzxhbk/YgTMmO
G4wKiIiEhhfFT7u0h9do4ePZon6KSmpHblVuIGQkPNkVnPPWfuv/oqxMTAH8wOICI
iUmyKFHamT5/Ohx9+aN+/5557CAkJOWrVquzbt6/YihPX4ekJL7wAS5dChQqwcaO5
zERiotWViYiIqytS2Jk/fz6RkZEARf27li+//JI1a9bQrVs3xowZU6wFimvp0wd27
oSGDeHUKejQAebNM0duiYi1lIQihZ3k5GR72FmxYgX9+/enc+fO/O1vf2Pnzp3FWq
C4nnrlzGUm7r4bsrLMYeoPPwy//WZ1ZSIi4oqKFHYqVqxon3V4zZolxMTEAGAYRr7
z74j8XoUK8NFH8OKL4OEBb78NbdvC0aNVVYyIiIq6mSGGnb9++3H///dxxxx388ssv
dOvWDYcVvvqK2rVrF2uB4rpsNnM9rbVr4YYbzGHqLVpcGbUlIiJSHIoUdl1555RWGD
BlCw4YNWbt2rXlOm59//pmnn366WASU19epE+zZY048ePYsdO1qdmBWMhMi1lIcij
TPjqvRPDvO4eJFGDYm3njD3L/zTrN566qJukVEROWK+vld4OUililbRrdu3fD29nZ
YNiI/vXv3Lnl1Iv9TrhwsWGAuMzF4MHZ2mf166VJz9JaIiEhRFPjJjoeHB8nJyVSu
XNm+bES+N7TzylwnZT3ZcT47d0K/fpCUBP7+sHAh3HOP1VWJiIgzKfYZLHnycqhc
bL99bW2shZ0xDMlbAm7d5v9eS5cgp79YcwYuHzZ6spERKSsKXQH5ZychP71r3/Rs2
dPGjduTJMnTbjzzjt555138iwOKnI9KlWCzz+Hv/3N3H/pJejcGU6ftrYuEREpWwo
VdgzDoHfv3jz22GOCpHmSJK2a0KhRI44dO8ZDDz3EXXfdVVJlipvy8oLp0+HjjyEg
ANavN5eZ2LHD6spERKSsKHAHZYBFixaxceNG4uPj6dixo8O5devW0adPH9555x0ef
PDBYi1S5O67oVEjuOsuOHQI2reH1l+Hxx+3ujIREXF2hXqy8/777/Pcc8/lCToAnt

p1YuzYsbz33nvFVpzI1Ro0MJ/o3HUXXLoETzwBjz1mDlkXERG51kKFnf3799O1a9d
rnu/WrZtWPZcSFRgI//43TJ1qLjPx11vmU54TJ6yuTEREnFWHws7Zs2cJCwu75vmw
sDDOnTt33UWJ/BGbDcaOhTVrIDQUdu0y+/HEx1tdmYiIOKNChZ3s7Gy8vK7dzcfT0
5PLGhsspeSOO8zh6c2bw5kz5kitGTNAGwJFRORqheqgbBgGDz30EL6+vvmesz8mLJ
aiRAqqRg3YvBmefhoWLYJnnzX79SxcaK6sLiIiUqiWExsb+6fXaCSWlDY/P/jXv6B
VK3NtrX//G779Fv7zH6hf3+rqRETEaloIFC0X4Uq2bTOHqZ88aT7Zefttc/SWiIi4
nmJfLkKkLGjd2uzH06EDnD8PfftCu3YwaVL+10+ZAhMnlmqJiIJSyHR2xOWEhcHat
TBypLm/ZYsZaMaNc7xuyhSIiwNPz1IvUURESPhCjrgkb294+WX44AMoX948Nm0aPP
WU+To36EyeDOPHWleniIiUPPXZQX12XN3XX5v9dn74wdz39ITsbAUdEZGyTn12RP6
ncWPYURN69TL3s7PNn998Y3ZoFHER16awI24hOBhatDBf22zmzw8/hOhos1PzBx9A
VpZl5YmISAlS2BG3MGUKTJhgN13l5JiTEILZpLV909x3H0RFmWtu/fKLtbWKiEjxU
tgRl5dfZ+Q5c8z97Gzo2NEcwXXyJDz3HERGwpNPMhMTiohI2aewIy7vWp2Rx483j9
96Kxw/bk5AePPN8NtvsGABNGoEXbrA6tXm0yARESmBNBoLjcaSKwWDNm2CWbPg00+
vhJx69WD4cHjwQfD3t7REERH5H43GEikCm8180vPvf5tD1UeOhMBAOHTI7OdTrZq5
2OiJE1ZXKiIiBaWwI3INUVHmxIQ//QSzZ00tWpCaCjNmWi03Qv/+sHWr+TRIREScl
6VhZ+PGjftQ1YuIiAhsNhuffvqpW3nDMIiLi6NKlSr4+fKREXPD4cOHHA45e/YsAw
cOJDawKODgYB599FEyMjJK8VOIq6tQAYYONZ/uLFsGnTqZ/YA+/hjatjVXWl+yBC5
dsrpSERHJj6Vh58KFCzRt2pQ5c+bke37GjBnMnj2b+fPns337dvz9/enSpQsXL160
XzNw4EC++eYblq5dy4oVK9i4cSNPPFEaX0EcSOenubEhPHxsG8fPPoo+PqaEXYOH
Gg+CfrHP+DMGasrFRGRqz1NB2WbzcBSpUvp06cPYD7ViYiIYNSOUYwePRqAtLQ0ws
LCWLRoEQMGDODgYM0bNiQnTt30uJ/M8atWbOG7t2789NPPxEREVGg91YHZSmq//4
X/vlPcyh7crJ5rFw5eOABs0Nz48bw1lici4srKfAflo0ePkpypcTExmJp1YUFAQrVq1
IjExEYDExESCg4PtQQcgJiYGDw8Ptm/ffs17Z2Zmkp6e7rCJFEWlSvD3v5tD1999F
5o3h4sX4c03oUkTuOMOWLlSQ9dFRKzktGEn+X//mRwWFuZwPCwszH4u0TmZypUrO5
z38vIiJCTEFkl+pk6dSlBQkH2LjIws5urF3fj4mE9zdu0yh6736wceHvD1l9CzJ9S
vbz79UXcyEZHS57RhpysNGzeOtLQ0+5aUlGR1SeIibDZolw4++QR+/BFGj4agIDh8
GIYMMYeuJxljPgkSEZHS4bRhJzw8HICUlBSH4ykpKfZz4eHhnd592uH85cuXOXv2r
P2a/Pj6+hIYGOiwiRS3mjXhxRfNoeuvvw516kBaGrz0kjl0/Z57YPNmDV0XESlpTh
t2oqKiCA8PJz4+3n4sPT2d7du3Ex0dDUB0dDSpqans3r3bfs26devIycmhVatWpV6
zSH4CamDwYPjuOlixAmJizD48n3wc7dtDy5aweLGGrouIlBRLw05GRgZ79+5l7969
gNkpee/evZw4cQKbzcaIESN4/vnnWbZsGQcOHODBBx8kIiLCpmKrQYMGdO3alccff
5wdO3awZcsWhgwZwoABAwo8EkuktHh4QI8esHYtHDgAjz1mjtZavRsGDYIaNeD558
0RXiIiUnwsHXq+YcMGOnbsmOd4bGwsixYtwjAMJkyYwIIFC0hNTaVdu3bMnTuXunX
r2q89e/YsQ4YMYfny5Xh4eNCvXz9mz55NQEBAGevQ0HOxypkz5qKjr78OP/9sHvP1
vTJ0vUkTa+stEXFmBf3+dpp5dqyksCNWu3TJbNZ65RVzRFeu2283Q0+PHuYK7Z6ee
VdvB5gyxZzVeeLEUitZRMRYZX6eHRF34uMD998PO3bAlilM52UPD3O25t69zVXXd+
6EuDgz2FxtYhTzuKenNbWLiDg7hr0RJ2KzQZs28NFHCOSIOUw9ONhcgX3VKrOJKy7
OXI0drgSdyZPzf+IjIiJqXgLUjCXO7cIFeOcdPVV+P77K8c9Pc2mKwUdEXFXasYS
cRH+/vDUU3DwoPl0p3Nn83h2ttnUNW6ctfWJiDg7hr2RMsLDA7p1M2dozpWTAzfdZ
K7HJSIi+VPYESlDru6j85//mElZBw9Cw4ag9WxFRPKnsCNSRvy+M/Jdd5kTFPr4wN
Gj0KCBJiQUEcmPwo5IGZFfz+SOHWHrVihfHk6dMpu4TpywrkYREWek0VhoNJaUfYc
OmR2XT5yAqlXhiy/Mpi0REVem0VgibqRePXMMywgYN4ORJc4HR7dutrKpExDko7Ii4
iGrVYNMmaNUKzp4115pYu9bqqkRErKewI+JCQkPhyy/hjjvMyQh79ICPP7a6KHERa
ynsiLiYgABYvhz694esLLj3Xpg/3+qqRESso7Aj4oJ8fWHJENjySTAMcwbmF14wX4
uIuBuFHREX5ekJ8+bB3/9u7v/97+YCojk51tYl1lLaFHZEXJjNZk5G+Mor5v6rr0J
srNm8JSLiLhR2RNzAiBHmyumenrB4MfTtC7/9ZnVViiKlQ2FHxE0MGgSffgrlysGK
Fdc1C6SmWl2ViEjJU9gRcSM9e5qzKwcFmXPY3HYbJCdbXZWISMLs2BFxM+3bQ0Ich
IXBvn3melPhjldhlyYhVHYEXFDTZuay0tERcGPP0LbtrB/v9VViYiUDIUDetdVq5
YZeJo0MZuyOnQw90VEXI3CjogbqlLFbNjQ29bsrHzHHbBqldVViYgUL4UdETdXsaL
Zabl7d3M4+p13wnvvWV2ViEjxUdGRECqXN4elP/AAXL5s/pw92+qqRESKh8KoIADg
7Q1vvw3Dhpn7w4dDXJzW0xKRsk9hR0TsPDzMSUMTzb3p0yBwYmH09vSskRErovCj
og4sNlg/HiY09d8PW8eDBwily5ZXZMSISNEo7IhIvp56Ct5/32ze+vBD6NULLLywui
oRkcJT2BGRa7r3Xli+3OzA/MUXEBMDZ89aXZWISOEo7IjIH+rSBeLjzSHq27bBrbf
CyZNWVYUiUnAKOyLyp1q3NhcOjYiAb74xJyE8fnjqkREckZhR0QKpFEjczmJOnXg
+HEz8Hz1ldVViYj8OacO09nZ2Ywfp56oqCj8/PyoVasWU6ZMwbhq4g/DmiLi6NKl
Sr4+fKREXPDYf0np0iJqFkTNm+Gm2+G//7XXE9rwwarqxIR+WN0HXamT5/OvHnzeP
311zl48CDTP09nxowZvPbaa/ZrZsyYwezZs5k/fz7bt2/H39+fLl26cPhIRQsrF3F
dlSubAadDBzh/Hrp2hc8+s7oqEZFrsgmG886P2rNnT8LCwnjrbrfsx/r164efnx+L

Fy/GMAwiIiIYNWoUo0ePBiAtLY2wsDAWLVRgEAEDCvQ+6enpBAUFkZaWRmBgYI18F
hFXc/Ei3HefucyEhwe8+SY8/LDVVYmIOyno97dTP9lp06YN8fHxfP/99wDs27ePzZ
s3061bNwCOHj1KcnIyMTE9t8JCgqiVatWJCYmXvO+mZmZpKenO2wiUjjlySHHH5s
BJycHHnkEXnrJ6qpERPLysrqAPzJ27FjS09OpX78+np6eZGdn88ILLzBw4EAAkpOT
AQgLC3P4vbCwMPu5/EydOpVJkyaVXOEibsLLC956C264AV58EcaMgTNnYOpUc/Z1E
RFn4NRPdj766CPee+89lixZwp49e3j77bd56aWXePvtt6/rvuPGjSMtLc2+JSU1FV
PFIu7HZoMZM2D6dHN/+nR4/HFz9XQREWfg1E92xowZw9ixY+19b5o0acLx48eZOnU
qsbGxhIeHA5CSkkKVK1Xsv5eSkkKzZs2ueV9fX198fX1LtHYRd/O3v0FICDz5pPm0
5+xZWLLb04SEbGSUz/Z+fXXX/HwcCzR090TnJwcAKKioggPDyc+Pt5+Pj09ne3bt
xMdHV2qtYoIPPaY2Y/HxweWLoUePcwRWYiIvNlQsNOrVy9eeOEFVq5cybFjxli6dC
kzZ87krrvuAsBmszFixAief/551ilbxoEDB3jwwQeJiIigT58+lhYv4qb69oXVqyE
gANatg06dzDl5REss4tRDz8+fP8/48eNZunQpp0+fJiIigvuu4+4uDh8fHwAc1LB
CRMmsGDBAlJTU2nXrh1z586lbt26BX4fDT0XKX67dkG3bmaH5XrlzIVEqle3uioRc
SUF/f526rBTWhR2RERGoUNwxx2QlATVqpmBp0EDq6sSEVfhEvPsiEjZVq+euZ5Wgw
bw00/Qvj3s2GF1VSLibhR2RKRErUaaK6b/5S/wyy/Qrh089FD+106ZAhMnlmZ1IuI
OFHZEPMsfhkJ8vNmklZUFb78Nv1/NZcoUiIsDT09rahQR16WwIyKlIiAAlI+He+4x
9z/8Ehr3Nl/nBp3Jk2H8eOtqFBHX5NSTCoqIa/H1hffffNycf/Oc/zfDj6Wmurfx3v
yvoiEjJ0JMdESlVnp4wb54ZbsAMomCurdWzJ7zxBvzB0nYiIoWmJzsiUupsNnOWZT
DDT3Y2ZGbcYpXmZrNBq1Zw553mVr++FhYVkaLTkx0RKXVX99G5fBkmTTKP3347tGw
JhgHbtsG4cdCwoTmEfcwY2LzZDEYiIoWhsCMipSq/zsi5+/Hx0KuXOSfPvHnQtav5
BOjwYXjpJXOenipV4JFH4LPP4NdfRf0sIlI2aAZlNI0ySGmaONFsusqVM/KUKeaTm
6vn2klPh88/N8PNypWQmnrlnJ+fOZz9zjvNkFSpUgkXLYJOrctFFILCjkjZkZJv1Tl
D42Wfmdvz4lXM2G7Rpc6WfTyGWxxORMkphpxAUDkTKHsOA/fuvBJ89exzP169/Jfi
0agUearQXcTkKO4WgsCNS9iUlwbJlZvBZv97s+JwrLMxs5rrzTrMTtJ+fdXWKSFPFR
2CkEhR0R15KWBqtXm8Fn1Sqz30+u8uWhSxcz+PTsaS5lISJlk8JOISjsiLiuS5cgI
eFKc9dPP1055+FhLkya29xVq5ZldYpI4SnsFILCjoh7MAZ46qsrwWffPsfzjRpdCT
4tWqifj4izU9gpBIUdEfd07NiVfj4JCY4TFkZEXOnn06mTua6XiDgXhZ1CUNgRkXP
nzP49n31m9vfJyLhyLiDAnODwzjuhRw+oWLHw8wWJSPEr6Pe3HtKKiGAGmIED4aOP
4MwZM/D89a/me56MDPjkExg0yJy4sFMn2LHDnPl5yhTH++TOEO3pac3nEJG89GQHP
dkRkVwLyYHdu6/08/n667zXdOgAzz8PX3xhhp2r18IQkZKjZqxGUNgRkYL68ccr/X
w2bTLD0NUiIqBvX2jdGqKjISpKK7aLlBSFnUJQ2BGRovjlF3O9rocfzht6clWqdCX
4tG5truoeEFC6dYq4qoJ+f3uVYk0iIi4lNNRcnysnxlyd/diluOceqFoVEhPNJSz+
+19YvtzcwBzO3rjx1fDTurW5jpeGuYuUHP3fS0SkiHI7IO+eDJmZ5s+PP4aQENi2z
Zy50TERXnkF+veH6tXNYLR/P/zzn+YToQYN4IYboFs3mDTJXOH93DmrP5mIalEzFm
rGEpHCuzroXN0Z+VrHc506Bdu3myFo2zbYtQt++y3vdQ0aXHnyEx0NDRtqhJfI76n
PTiEo7IhIYRXXPdtZWXDgwJXws20b/PBD3usCAuAvf7kSflq1MvsDibgzH21CUNgR
EWfy3/+aT39yw8/27Y6THOaqVcux8/NNN4G3d+nXK2IVhZ1CUNgREWeWnQ3ffnsl/
CQmwsGDea/z8zPX9Mpt/mrd2hwKL+KqFHYKQWFHRMqalFRzFufc8LNTm3ns96pXdw
w/zZs7rvOlZS+kLNPQcxERFxyCDJ07mxuYo7wOH3YMPwcOwIkT5vbRR+Z1Pj5w881
Xws/58zBzpnnuWh2tRco6Pd1BT3ZExDVLZJijvXLDT2Ki2R/o9wICzGvvuAP+/nf4
8ksteyFlg5qxCKfhr0TcgWHAsWOOI7+++gouX857bdWq5txAbdpA27ZQpUqplyvyp
xR2CkFhR0TclW+/mTM9b9sGf/vbtZe9qFnTDD1t2phbkyya90esV9Dvb6efQfnkyZ
M88MADhIaG4ufnR5MmTdila5f9vGEYxMXFUaVKFfz8/IiJieHw4cMWviwiUnb4+Zk
h5tdfryx7AXD33fD009CsmbmUxbFj8N57MHiw2ecnONhs9po40VztPT3dus8g8mec
OuycO3eOtm3b4u3tzerVq/n22295+eWXqVixov2aGTNmMHv2bObPn8/27dvx9/enS
5cuXLx40cLKRUTKjvyWvfjkEwgPN5u5zp2DtWvNYNO5M1SoYPbx+fJLc4mLL13M8N
O0KTz1FCxeDEeOmM1Is7AqZuxxo4dy5YtW9i0aVO+5w3DICIiglGjRjF69GgA0tL
SCAsLY9GiRQwYMKBA76NmLBfXV0VZ9iI7G775BrZuhS1bzJ9HjuS9d3j41T4/bdqY
T4SuHvYucr1cos9Ow4YN6dKlCz/99BMJCQlUrVqVp59+mscfxyAI0eOUktWLB766
iuaNWtm/70OHTTrQrFkzZs2ale99MzMzyczMtO+np6cTGRmpsCMibqe45tn5+Wez43
Nu+Nm921wK42q+vtCy5ZUAFB2tJS/k+rHE2ClXrhwaIOeO5J577mHnzp0MHZ6c+fP
nExsby9atW2nbtI2nTp2iylVDBfr374/NZuPDDz/M974TJ05k0qRJeY4r7IiIFI/f
fjMDZ9VPf86cyXtd3bqOT3/q1zf7CIkUheEHR8fH1q0aMHWrvvtx4YNG8bOnTtJT
EwsctjRkx0RkdJlGOakh1eHn2+/zXtdxYrmE5/cANSyJfj71369Uja4xAzKVapUoW
HDhg7HGjRowL//W8AawsPDAUhJSXEIOykpKQ7NWr/n6+uLrxqORURKjclmPsWpWxc
eesg8dvasOeQ9NwBt3252h161ytzAbGJrluzKk5+2baFatbz317IX8kec+mFh27Zt
OXTokMox77//nholagAQFRVFeHg48fHx9vPp6els376d6OjoUq1VREQKJyQEuneH5
5+H9eshLQ127oRZs8wJDatWNUPK7t0wezYMGACRkeZ6X/fdB6+9Zs4RdPmyGXTi4s

xgc7XcjtaaE8jNGU5sx44dhpeXl/HCCy8Yhw8fNt577z2jfPnyxuLfi+3XTJs2zQg
ODjY+++wzY//+/cadd95pREVFGB/99luB3yctLc0AjLS0tJL4GCIiUkThJxvG++8b
xpAhhtG8uWF4ehqG2Sh2ZStf3jA6djSMDh3M/eeeM3938mRzf/JkSz+ClKCCfn87d
Z8dgBUrVjBu3DgOHZ5MVfQUI0eOtI/GAnP4+YQJELiWYAGpQam0a9eOuXpNurdu3Q
K/h4aei4iUDRkZ5mrvuU1fiYnmE6Hfs9nMKNS1K4wdaw5717/eXY9LdFAuLQo7IiJ
1U0602dH56o7PP/yQ/7V16sAtt0Dz5ubPm28200RL2aWwUwgKOYIiriG3j46Xl9mX
p359uHABkpLyvz4qyjEANW8ON9xQujVL0bnEaCwREZGC+v2sz1fv//Wv5tIXu3ebn
Zp374ajR69sn3xy5T6RkXkD0P8G/0oZpbAjIiJlXn7LW+T+jIu7st+585XfOXcubw
A6fNh8CpSUBJ9+euXaKlXyBqCqVc2+QeL81IyFmrFERMq64ppnJz3dDEB79lwJQN9
9l/+ippUrm6Hn6gBUo4YCUGlSn51CUNgREZFryciAffscA9C335oB6vdCQvIGoFq1
FIBKisJOISjsiIhIYfz2Gxw44NgE9vXXeRc/BQgKMkd+XR2A6tbNfw0wzQRdOOqgL
CIiUkL8/OAvfzG3XJmZzuC5+gnQ/v3mPEAbNphbroAAcxmM3PDTvLk5cix3JmhwDD
xX90mSwtOTHfRkR0RESkZWltnkdXUA2rvXfDL0e35+0LSpOXfQjh3w1FPm0hnTpuX
tfC0mNWMVgsKOiIiUluxsOHTIsQnsq6/MvkHX0qED/N//mSvCBwSUXq30TmGnEBR2
RETESjk55rD3q58ArV+f9zpPT7P/T7t20L69+bNy5dKv11ko7BSCwo6IiDiT3D463
t5mU9hNN5l9f44fz3ttvXpm8MndatZ0n9FfBf3+zcqvuiIiIiFjl6s7Ily6ZP/fvh0
cfhRMn4L33zBmhGzc2rz90CN58E2Jj4cYbzRmg77sP5s4lR4z15Fj7eZyBRmOJiIg
4iYLOBH3//ebsr2fNBVA3bTK3Xbvg5En44ANzAwgOhrZtrzz5adECfHxK9WNZTs1Y
qBlLREScw/XOs/Prr7B9+5Xwk5hoLoR6tXLloFWrK+EnOhoqVCjOT1F61GenEBR2R
ETEFV2+bi702rz5SgA6c8bxGg8Pc86fq/v9lJVOzwo7haCwIyIi7sAwzD4+ucFn0y
Y4dizvdXXrOoafqCjn7PSssFMICjsiIuKufvrJMfx8/XXeayIirgxlB98emjTJf7m
L0qawUwgKOYIiIqZz5/J2ev79ml9BQXk7Pfv6XjlfWmt8aW0sERERKbSKFaFnT3MD
s9Pzjh2OnZ7T0mDVKnMDs9PzX/5y5enP5cswaZJ5zhnW+NKTHfRkR0REpKAuXzbX9
7q60/N//+t4jYcHhIXBzz/Dvfeaa3wtWFD8a3ypGasQFHZERESKxjDg++8d+/0cPZ
r/tcW9mKnCTiEo7IiIiBSfkyevBJ9588xA50ltzghdnLRchIiIiFiialUYMADCw82
g4+NjdnKeMsWaehR2REREpNhd3Rk5M9P8GRdnTeDRaCwREREpVgVd46u0KOYIiIhI
scrOzr8zcu5+dnbplqM0yqiDsoiISfMkDsoiIiIiKOYIiIiIiIiPYEREREZemsCMiI
iIuTWfHREXJrCjoiIiLi0MhV2pk2bhs1mY8SIEfZjFy9eZPDgwYSghhIQEEC/fv
1ISUmxrkgRERFxmUm7OzcuZN//vOf3HTTTQ7Hn3nmGZYvX87HH39MQkICp06dom/
fvhZVKSiIIs6mTISdjIwMBg4cyBtvvEHFihXtx9PS0njrrbeYOXMmnTp14pZbbmHh
woVs3bqVbdu2WVixiIiIOIsyEXYGDx5Mjx49iImJcTi+e/dusrKyHI7Xrl+f6tWrk
5iYeM37ZWZmkp6e7rCJiIiIa3L6tbE++OAD9uzZw86dO/Ocs05OxsfHh+DgYIfjYW
FhJCcnX/OeU6dOZdKkSXmOK/SIiIiUHbnf23+28pVTh52kpCSGDx/O2rVrKVeuxLH
dd9y4cYwcOdK+f/LkSRo2bEhkZGSxvYeIiIiUjvPnzxMUFHTN804ddnbv3s3p06dp
3ry5/Vh2djYbN27k9ddf5/PPP+fSpUukpqY6PN1JSUkhPDz8mvf19fXF19fXvh8QE
EBSUhIVKlTAZrOVyGcpy9LT04mMjCQpKUKLPtoJ/U2ci/4ezkV/D+dSkN8PwzA4f/
48ERERf3idu4ed22+/nQMHDjgce/jhh6lfvz7PPvsskZGREht7Ex8fT79+/QA4dOg
QJ06cIDo6usDv4+HhQbVq1Yq1dlcUGBIOF3E4Gf1NnIv+Hs5Ffw/nULJ/jz96opPL
qcNohQoVaNy4scMxf39/QkND7ccfffrRRo4cSUhICIGBgQwdOpTo6Ghat25tRckiI
iLiZJw67BTEK6+8goeHB/369SMzM5MuXbowd+5cq8sSERERJlHmws6GDRsc9suVK8
ecOXOYM2eONQW5AV9fXyZMmODQz0mspb+Jc9Hfw7no7+FcnOHvYTP+bLyWiIiISBl
WJiYVFBERESkqhR0RERFxaQo7IiIi4tIUdkRERMSlKezINU2dOpWWLVtSoUIFKleu
TJ8+fTh06JDVZcn/TJs2DZvNxogRI6wuxW2dPHmSBx54gNDQUPz8/GjSpAm7du2yu
iy3lZ2dzfjx44mKisLPz49atWoxZcQUP103SYrHxo0b6dWrfXEREDhsNj799FOH84
ZhEBcXR5UqVfDz8yMmJobDhw+XSm0KO3JNCQkJDB48mG3btrF27VqysrLo3LkzFy5
csLo0t7dz507++c9/ctNNN1ldits6d+4cbdu2xdvbm9WrV/Ptt9/y8ssvU7FiRatL
clvTp09n3rx5vP766xw8eJDP06czY8YMXnvtNatLcwsXLlygadOm15wKZsaMGcyeP
Zv58+ezfft2/P396dKlCxcvXizx2jT0XArsv//9L5UrVyYhIYFbb73V6nLcVkJZGBS
2bN2fu3Lk8//zzNGvWjFdfdfXqstz02LFj2bJlC5s2bbK6FPmfnj17EhYwxltvvWU
/1q9fP/z8/Fi8eLGF1bkfM83G0qVL6dOnD2A+1YmIiGDUqFGMHj0agLS0NMLCwli0
aBEDBgwo0Xr0ZEcKLC0tDYCQkBCLK3FvggcPpkePHsTExFhdiltbtmWZLVq04J577
qFy5crcfPPNvPHGG1aX5dbatGLDfHw833//PQD79u1j8+bNdOvWzeLkX5j27YqQnJz
v8eysoKIhWrVqRmJhY4u9f5mZQFmV5k0QwYsQI2rZtm2e9Mik9H3ZwAKAv20GHnzp1
Wl+L2kjhw5wrx58xg5ciTPPfcc03fuZNIwYfj4+BAbG2tleW5p7NixpKenU79+fTw9
PcnOzuaFF15g4MCBVpfm9pKTkwEICwtzOB4WFmY/V5IudqRABg8ezNdf83mzZutL
sVtJSU1MXz4cNauXUu5cuWsLsft5eTk0KJFC/7xj38AcPPNN/P1118zf/58hR2LFP
TRR7z33nssWbKERO0asXfvXkaMGFERIT+Jm5OzVjyp4YMGcKKFStYv3491apVs7o
ct7V7925Onz5N8+bN8fLywsvLi4SEBGbPno2XlxfZ2dlWl+hWqlSpQsOGDR2ONWjQ

```

gBMnTlhUkYwZM4axY8cyYMAAmjRpwqBBg3jmmWeYOnWqlaW5vfDwcABSULIcjqekp
NjPlSSFHbkmwzAYMmQIS5cuZd26dURFRVldklu7/fbbOXDgAHv37rVvLVq0YODAge
zduxdpT0+rS3Qrbdu2zTMVw/fff0+NGjUsqkh+/fVXPDwcv9Y8PT3JycmxqCLJFRU
VRXh4OPHx8fZj6enpbN++nejo6BJ/fzVjyTUNHjyYJUuW8NlnnlGhQgV7u2pQUBB+
fn4WV+d+KlSokKe/1L+/P6GhoepHZYFnnmGNm3a8I9//IP+/fuzY8cOFixYwIIFC
6wuzW316tWLF154gerVq9OoUSO++uorZs6cySOPPGJlaW4hIyODH374wb5/90hR9u
7dS0hICNWrV2fEiBE8//zz1KlTh6ioKMaPH09ERIR9xFajMkSuAch3W7hwodWlyf9
06NDBGD58uNVluK3ly5cbjRs3Nnx9fY369esbCxYssLokt5aenm4MHz7cqF69ulGu
XDnjxhtvNP7v//7PyMzMtLo0t7B+/fp8vzNiY2MNwzCMnJwcY/z48UZYZWJjh6+tr3
H777cahQ4dKpTbNsyMiIiIuTX12REREXKUp7IiIiIhLU9gRERERl6awIyIiIi5NYU
dERERcmsKOiIiIuDSFHREREXFpCjsiUuImTpxIs2bNAGjWrBkTJ060tB4RcS8KOyJ
yXR566KE8071/8sknlCtXjpdfhfmA0aNH29fEiY+PZ/To0aVdpt0bb7xB06ZNCQgI
IDg4mJtvvlkLRYq4OK2NJSLF6s0332Tw4MHMnz+fHx9+GICAgAACAgIACA0Ntay2f
/3rX4wYMYLZs2fToUMHMjMz2b9/P19//bVlNYlIydOTHREpNjNmzGDo0KF88MEH9q
CT350fESNGcNttt9n3b7vtNoYOHcQIESOoWLEiYWFhvPHGGly4cIGHH36YChUqULT
2bVavXulwn6+//ppu3boREBBAWFgYgWYN4syZM9esb9myZfTv359HH32U2rVr06hR
I+677z5eeOEF+zU7d+7kjjvu4IYbbiAoKIgOHTqwZ88e+/lHHnmEnj170tw3KyuLy
pUr89Zbb9k/z4gRI+zn586dS506dShXrhhYWHcffffdDp/96mtFpPgp7IhIsXj22W
eZMmUKKlas4K677ir077/99tvccMMN7Nixg6FDh/LUU09xzz330KZNG/bs2UPnzp0
ZNGgQv/76KwCpqa106tSjM2++mV27drFmzRpSULlo37//Nd8jPDycbdu2cfz48Wte
c/78eWJjY9m8eTPbmt2jTp06dO/enfPnzPw2GOPSWbNGn7++Wf776xYsYJff/2Ve
++9N8/9du3axbBhw5g8eTKHDh1izZo13HrrrYX+5yMil6FULhsVEZcVGxtr+Pj4GI
ARHx+f7/k777zT4djw4cONDh062Pc7dOhgtGvXzr5/+fJlw9/f3xg0aJD92M8//2w
ARmJiomEYhjFlyhSjc+fODvdNSkoygGuupHzq1CmjdevWBmDURvVXiI2NNT788EMj
Ozv7mp8vOzvbqFChgrF8+XL7sYYNGxrTp0+37/fqlct46KGHHD5P7mr0//73v43Aw
EAjPT093/tr5XqRkqcnOyJy3W666SZq1qzJhAkTyMjIKPI9cnl6ehIaGkqTJk3sx8
LCwgA4ffo0APv27WP9+vX2/kABAQHUr18fgB9//DHf96hSpQqJiYkcoHCA4cOHc/n
yZWJjY+natSs5OTkApKSk8Pjj1OnTh2CgoIIDAwkIyODEyd020/z2GOPSXDhQvv1
q1ev5pFHHsn3Pe+44w5q1KjBjTfeyKBBg3jvvffsT6dEpHqo7IjIdatatSobNmzg5
MmTd03ald7kA+Dh4YFhGA7XZ2Vl5bmHt7e3w77NZnM4ZrPZA0yhJCMjg169erF371
6H7fDhw3/aTNS4cWOefvppFi9ezNqla1m7di0JCQkAxMbGsnfvXmbNmsXWrVvZu3c
voaGhXLP0yf77Dz74IEeOHCEXmZHFixcTFRVF+/bt832vChUqsGfPhT5//32qVKlC
XFwcTZs2JTU19Q9rFJHio7AjIsWiRo0aJCQkkJyc7BB4KlWq5NC/BWDv3r3X/X7Nm
zfnm2++oWbNmtSuXdth8/f3L/B9GjZsCMCFcxcA2LJlC8OGDaN79+40atQIX1/fPJ
2eQ0ND6dOnDwsXLmTRokX2ztjX4uXlRUxMDDNmzGD//v0c03aMdevWFFITi0hRKey
ISLGJjIxkw4YNnD59mi5dupCenk6nTp3YtWsX77zzDocPH2bChAnFMtR78ODBnD17
lvvu4+d03fy448/8vnnn/Pww+TnZ2d7+889dRTTJkyhS1btnD8+HG2bdvGgw8+S
KVKlYiOjgagTp06vPvuuxw8eJdt27czcOBA/Pz88tzrscce4+233+bgwYPExsZes8
4VKlYwe/Zs9u7dy/Hjx3nnnXfIycmhXr161/3PQEQKRMFHRIPvtWrV2LBhA2fOnKF
Lly5ER0czfvx4/va3v9GyZUvOnz/Pgw8+eN3vExERwZYtW8jOzqZz5840adKEESNG
EBwcjIdH/v9qi4mJYdu2bdxzzz3UrVuXfv36Ua5cOeLj4+3z/7z111uc03e05s2bM
2jQIIYNG0blypXzvVeVKlXo0qULERERl6wzODiY//znP3Tq1IkGDRowf/583n//fR
olanTd/wxEpGBsxu8b00VE5E9lZGRQtWpVfi5cSN++fa0uR0T+gGZQFhEphJychM6
cOcPLL79MchAwvXv3trokefkTCjsiIoVw4sQJoqKiqFatGosWLCllS/8aFXF2asYS
ERERl6YOyiIiIuLSFHZERETEpSnsiIiIiEtT2BERERGxprAjIiIiLk1hR0RERFyaw
o6IiIi4NIUDERERCWkKOyIiIuLS/h8OFa/cf+1cZAAAAABJRu5ErkJggg==\n"

```

```

    },
    "metadata": {}
  }
},
{
  "cell_type": "code",
  "source": [
    "kume_sayisi = int(input(\"Lütfen küme sayısını girin:
\")(\"\"))\n"
  ],
  "metadata": {}
}

```

```

"colab": {
  "base_uri": "https://localhost:8080/"
},
"id": "0-KY-9X_OeD-",
"outputId": "89862ad8-1244-4235-9215-458353d947e2"
},
"execution_count": null,
"outputs": [
  {
    "name": "stdout",
    "output_type": "stream",
    "text": [
      "Lütfen küme sayısını girin: 4\n"
    ]
  }
]
},
{
  "cell_type": "code",
  "source": [
    "\n",
    "# K-Means kümeleme\n",
    "def cluster_bayiler(data, n_clusters):\n",
    "    kmeans = KMeans(n_clusters=n_clusters,\n",
    random_state=0, n_init=10)\n",
    "    labels = kmeans.fit_predict(data)\n",
    "    return labels\n",
    "\n",
    "labels = cluster_bayiler(mesafe_mat, kume_sayisi)\n",
    "kume_dict = {i: [] for i in range(kume_sayisi)}\n",
    "for i, label in enumerate(labels):\n",
    "    kume_dict[label].append(bayi_ids[i])\n",
    "\n",
    "# Yapay Arı Kolonisi Algoritması (ABC) ile En Kısa
Rota\n",
    "\n",
    "def calculate_route_distance(route, distance_matrix,\n",
    depo_mesafeleri, bayi_ids):\n",
    "    total_distance = 0\n",
    "    for i in range(len(route) - 1):\n",
    "        if route[i] == 1: # Depo çıkışı\n",
    "            idx2 = np.where(bayi_ids ==\n",
    route[i+1])[0][0]\n",
    "            distance = depo_mesafeleri[idx2]\n",
    "        elif route[i+1] == 1: # Depoya dönüş\n",
    "            idx1 = np.where(bayi_ids ==\n",
    route[i])[0][0]\n",
    "            distance = depo_mesafeleri[idx1]\n",
    "        else:\n",
    "            idx1 = np.where(bayi_ids ==\n",
    route[i])[0][0]\n",
    "            idx2 = np.where(bayi_ids ==\n",
    route[i+1])[0][0]\n",
    "            distance = distance_matrix[idx1, idx2]\n",
    "            total_distance += distance\n",
    "    return total_distance\n",
    "\n"
  ]
}

```

```

def artificial_bee_colony(kume, mesafe_matrix,
depo_mesafeleri, bayi_ids, max_iter=100):\n",
    "    best_route = None\n",
    "    best_distance = float('inf')\n",
    "    num_bayiler = len(kume)\n",
    "\n",
    "    for _ in range(max_iter):\n",
    "        random.shuffle(kume)\n",
    "        route = [1] + kume + [1] # Depodan çıkıp tekrar
depoya döner\n",
    "        distance = calculate_route_distance(route,
mesafe_matrix, depo_mesafeleri, bayi_ids)\n",
    "        if distance < best_distance:\n",
    "            best_distance = distance\n",
    "            best_route = route\n",
    "    return best_route, best_distance\n",
    "\n",
    "rotalar = {}\n",
    "mesafeler = {}\n",
    "for kume_id, bayiler in kume_dict.items():\n",
    "    rota, mesafe = artificial_bee_colony(bayiler,
mesafe_mat, depo_mesafeleri, bayi_ids)\n",
    "    rotalar[kume_id] = rota\n",
    "    mesafeler[kume_id] = mesafe\n",
    "\n",
    "# Kapasite aşımı kontrolü ve ek araç yönetimi\n",
    "MAX_KAPASITE = 5 * 3 # Araç kapasitesi\n",
    "araclar = {}\n",
    "\n",
    "def arac_yonet(rota, siparis_df, kume_id):\n",
    "    arac_sayaci = 1\n",
    "    kapasite = 0\n",
    "    araclar[kume_id] = {}\n",
    "    araclar[kume_id][arac_sayaci] = []\n",
    "\n",
    "    for bayi in rota:\n",
    "        if bayi == 1:\n",
    "            continue\n",
    "        siparis = siparis_df[siparis_df['Lokasyonlar']
== bayi][['A', 'B']].values.flatten()\n",
    "        siparis_miktari = siparis[0] + siparis[1]\n",
    "\n",
    "        if kapasite + siparis_miktari >
MAX_KAPASITE:\n",
    "            arac_sayaci += 1\n",
    "            kapasite = 0\n",
    "            araclar[kume_id][arac_sayaci] = []\n",
    "\n",
    "            araclar[kume_id][arac_sayaci].append((bayi,
siparis[0], siparis[1]))\n",
    "            kapasite += siparis_miktari\n",
    "\n",
    "for kume_id, rota in rotalar.items():\n",
    "    arac_yonet(rota, siparis_df, kume_id)\n",
    "\n",
    "# Çıktılar\n",
    "print("\n\n📌 1. KÜMELEME SONUÇLARI")\n",
    "for kume_id, bayiler in kume_dict.items():\n",

```

```

        print(f"Küme {kume_id+1}: {bayiler}\n")\n",
        "\n",
        "print(\n\n🚗 2. ROTASYON SONUÇLARI\n)\n",
        "for kume_id, rota in rotalar.items():\n",
        "    print(f"Küme {kume_id+1} - Rota: {rota}\n")\n",
        "\n",
        "print(\n\n🚚 3. HER ARAÇ İÇİN ROTASYON VE
MESAFELELER\n)\n",
        "for kume_id, arac_list in aracalar.items():\n",
        "    for arac_id, rota in arac_list.items():\n",
        "        arac_mesafe = calculate_route_distance([1] +
[x[0] for x in rota] + [1], mesafe_mat, depo_mesafeleri,
bayi_ids)\n",
        "        print(f"Küme {kume_id+1} - Araç {arac_id} -
Rota: {[1] + [x[0] for x in rota] + [1]} - Mesafe: {arac_mesafe}
km\n")\n",
        "        for bayi, a_miktar, b_miktar in rota:\n",
        "            print(f"        Araç {arac_id} - Bayi {bayi}:
{a_miktar} A, {b_miktar} B\n")\n"
    ],
    "metadata": {
        "colab": {
            "base_uri": "https://localhost:8080/"
        },
        "id": "tOB1sxKbOgfw",
        "outputId": "0b6367d7-7897-43c9-8d70-10ac7ea56e25"
    },
    "execution_count": null,
    "outputs": [
        {
            "output_type": "stream",
            "name": "stdout",
            "text": [
                "\n",
                "📌 1. KÜMELEME SONUÇLARI\n",
                "Küme 1: [60, 34, 40]\n",
                "Küme 2: [23, 50, 30, 2, 21, 46, 16, 26, 45, 31, 39,
58, 27, 14, 51, 13, 20, 59, 38, 5, 7, 11, 57, 25, 24, 15, 52, 36,
54, 6, 35, 32, 42, 28, 29, 22, 41, 48, 47, 55, 43, 33, 37, 8, 49,
10]\n",
                "Küme 3: [56, 9, 19]\n",
                "Küme 4: [53, 17, 18, 4, 3, 12, 44]\n",
                "\n",
                "🚗 2. ROTASYON SONUÇLARI\n",
                "Küme 1 - Rota: [1, 60, 40, 34, 1]\n",
                "Küme 2 - Rota: [1, 39, 48, 43, 52, 13, 55, 54, 16,
29, 20, 26, 14, 51, 27, 31, 36, 57, 42, 2, 24, 25, 11, 49, 35, 5,
22, 41, 47, 37, 45, 23, 33, 10, 28, 15, 50, 46, 8, 6, 38, 32, 58,
30, 7, 59, 21, 1]\n",
                "Küme 3 - Rota: [1, 9, 19, 56, 1]\n",
                "Küme 4 - Rota: [1, 44, 3, 18, 12, 17, 53, 4, 1]\n",
                "\n",
                "🚚 3. HER ARAÇ İÇİN ROTASYON VE MESAFELELER\n",
                "Küme 1 - Araç 1 - Rota: [1, 60, 40, 34, 1] - Mesafe:
358.3 km\n",
                "        Araç 1 - Bayi 60: 1 A, 0 B\n",
                "        Araç 1 - Bayi 40: 1 A, 0 B\n"
            ]
        }
    ]
}

```

" Araç 1 - Bayi 34: 1 A, 1 B\n",
 "Küme 2 - Araç 1 - Rota: [1, 39, 48, 43, 52, 13, 55,
 54, 16, 29, 20, 1] - Mesafe: 180.80000000000004 km\n",
 " Araç 1 - Bayi 39: 0 A, 2 B\n",
 " Araç 1 - Bayi 48: 1 A, 1 B\n",
 " Araç 1 - Bayi 43: 0 A, 2 B\n",
 " Araç 1 - Bayi 52: 0 A, 1 B\n",
 " Araç 1 - Bayi 13: 0 A, 1 B\n",
 " Araç 1 - Bayi 55: 0 A, 1 B\n",
 " Araç 1 - Bayi 54: 0 A, 2 B\n",
 " Araç 1 - Bayi 16: 1 A, 1 B\n",
 " Araç 1 - Bayi 29: 1 A, 0 B\n",
 " Araç 1 - Bayi 20: 0 A, 1 B\n",
 "Küme 2 - Araç 2 - Rota: [1, 26, 14, 51, 27, 31, 36,
 57, 42, 1] - Mesafe: 97.7 km\n",
 " Araç 2 - Bayi 26: 1 A, 0 B\n",
 " Araç 2 - Bayi 14: 0 A, 3 B\n",
 " Araç 2 - Bayi 51: 0 A, 2 B\n",
 " Araç 2 - Bayi 27: 0 A, 3 B\n",
 " Araç 2 - Bayi 31: 0 A, 1 B\n",
 " Araç 2 - Bayi 36: 0 A, 1 B\n",
 " Araç 2 - Bayi 57: 0 A, 1 B\n",
 " Araç 2 - Bayi 42: 1 A, 1 B\n",
 "Küme 2 - Araç 3 - Rota: [1, 2, 24, 25, 11, 49, 35,
 5, 22, 41, 47, 37, 1] - Mesafe: 227.89999999999995 km\n",
 " Araç 3 - Bayi 2: 1 A, 1 B\n",
 " Araç 3 - Bayi 24: 0 A, 1 B\n",
 " Araç 3 - Bayi 25: 1 A, 0 B\n",
 " Araç 3 - Bayi 11: 0 A, 2 B\n",
 " Araç 3 - Bayi 49: 1 A, 0 B\n",
 " Araç 3 - Bayi 35: 0 A, 2 B\n",
 " Araç 3 - Bayi 5: 1 A, 0 B\n",
 " Araç 3 - Bayi 22: 0 A, 2 B\n",
 " Araç 3 - Bayi 41: 0 A, 1 B\n",
 " Araç 3 - Bayi 47: 0 A, 1 B\n",
 " Araç 3 - Bayi 37: 0 A, 1 B\n",
 "Küme 2 - Araç 4 - Rota: [1, 45, 23, 33, 10, 28, 15,
 50, 46, 8, 6, 38, 32, 58, 30, 1] - Mesafe: 293.2 km\n",
 " Araç 4 - Bayi 45: 0 A, 1 B\n",
 " Araç 4 - Bayi 23: 1 A, 0 B\n",
 " Araç 4 - Bayi 33: 1 A, 0 B\n",
 " Araç 4 - Bayi 10: 0 A, 1 B\n",
 " Araç 4 - Bayi 28: 1 A, 1 B\n",
 " Araç 4 - Bayi 15: 1 A, 0 B\n",
 " Araç 4 - Bayi 50: 1 A, 0 B\n",
 " Araç 4 - Bayi 46: 0 A, 1 B\n",
 " Araç 4 - Bayi 8: 1 A, 0 B\n",
 " Araç 4 - Bayi 6: 1 A, 0 B\n",
 " Araç 4 - Bayi 38: 0 A, 1 B\n",
 " Araç 4 - Bayi 32: 1 A, 0 B\n",
 " Araç 4 - Bayi 58: 0 A, 1 B\n",
 " Araç 4 - Bayi 30: 0 A, 1 B\n",
 "Küme 2 - Araç 5 - Rota: [1, 7, 59, 21, 1] - Mesafe:
 64.3 km\n",
 " Araç 5 - Bayi 7: 0 A, 1 B\n",
 " Araç 5 - Bayi 59: 0 A, 1 B\n",
 " Araç 5 - Bayi 21: 1 A, 0 B\n",

```

        "Küme 3 - Araç 1 - Rota: [1, 9, 19, 56, 1] - Mesafe:
349.1 km\n",
        "    Araç 1 - Bayi 9: 1 A, 0 B\n",
        "    Araç 1 - Bayi 19: 1 A, 0 B\n",
        "    Araç 1 - Bayi 56: 1 A, 0 B\n",
        "Küme 4 - Araç 1 - Rota: [1, 44, 3, 18, 12, 17, 53,
4, 1] - Mesafe: 372.00000000000006 km\n",
        "    Araç 1 - Bayi 44: 1 A, 0 B\n",
        "    Araç 1 - Bayi 3: 1 A, 0 B\n",
        "    Araç 1 - Bayi 18: 0 A, 1 B\n",
        "    Araç 1 - Bayi 12: 0 A, 1 B\n",
        "    Araç 1 - Bayi 17: 0 A, 1 B\n",
        "    Araç 1 - Bayi 53: 1 A, 0 B\n",
        "    Araç 1 - Bayi 4: 0 A, 2 B\n"
    ]
}
]
}
]
}

```

```

{
  "nbformat": 4,
  "nbformat_minor": 0,
  "metadata": {
    "colab": {
      "provenance": []
    },
    "kernelspec": {
      "name": "python3",
      "display_name": "Python 3"
    },
    "language_info": {
      "name": "python"
    }
  },
  "cells": [
    {
      "cell_type": "code",
      "execution_count": 1,
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "ntqI_53RbzFX",
        "outputId": "ddfbf134-38c9-4b41-8ada-0f416d9ebfbe"
      },
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "Mounted at /content/drive\n"
          ]
        }
      ],
      "source": [
        "# Google Drive Bağlantısı\n",
        "from google.colab import drive\n",
        "drive.mount('/content/drive')\n"
      ]
    },
    {
      "cell_type": "code",
      "source": [
        "import pandas as pd\n",
        "import numpy as np\n",
        "import matplotlib.pyplot as plt\n",
        "from sklearn.cluster import KMeans\n",
        "from scipy.spatial.distance import cdist\n",
        "import random\n",
        "\n",
        "# Google Drive'dan dosyaları yükleme\n",
        "mesafe_path =\n"/content/drive/MyDrive/end_/Mesafe.xlsx"\n",
        "siparis_path =\n"/content/drive/MyDrive/end_/siparis.xlsx"\n",

```

```

"depo_path =
\"/content/drive/MyDrive/end_/Depo.xlsx\"\\n",
"\\n",
"mesafe_df = pd.read_excel(mesafe_path)\\n",
"siparis_df = pd.read_excel(siparis_path)\\n",
"depo_df = pd.read_excel(depo_path)\\n",
"\\n",
"# Bayi ID'lerini al\\n",
"bayi_ids = mesafe_df.iloc[:, 0].values # 2'den 60'a
kadar bayi ID'leri\\n",
"mesafe_mat = mesafe_df.iloc[:, 1:].values # 59x59
mesafe matrisi\\n",
"\\n",
"depo_mesafeleri = depo_df.iloc[0, 2:].values # Depodan
bayilere mesafeler\\n",
"\\n",
"# Kümeleme için Elbow yöntemi\\n",
"def elbow_method(data):\\n",
"    distortions = []\\n",
"    K = range(1, 11)\\n",
"    for k in K:\\n",
"        kmeans = KMeans(n_clusters=k, random_state=0,
n_init=10)\\n",
"        kmeans.fit(data)\\n",
"        distortions.append(sum(np.min(cdist(data,
kmeans.cluster_centers_, 'euclidean'), axis=1)) /
data.shape[0])\\n",
"        plt.plot(K, distortions, 'bx-')\\n",
"        plt.xlabel('Küme Sayısı')\\n",
"        plt.ylabel('Distortion')\\n",
"        plt.title('Elbow Yöntemi')\\n",
"        plt.show()\\n",
"\\n",
"# Kullanıcıdan küme sayısını al\\n",
"elbow_method(mesafe_mat)"
],
"metadata": {
"colab": {
"base_uri": "https://localhost:8080/",
"height": 472
},
"id": "l46U3yBACXeP",
"outputId": "c70cea7f-cbed-4809-fe11-919ebb621ea9"
},
"execution_count": 3,
"outputs": [
{
"output_type": "display_data",
"data": {
"text/plain": [
"<Figure size 640x480 with 1 Axes>"
],
"image/png":
"iVBORw0KGgoAAAANSUhEUgAAAJsAAAHHCAYAAABZbpmkAAAAOnRFWHRTb2Z0d2Fy
ZQBNYXRwbG90bGliIHZlcnNpb24zLjEwLjAsIGh0dHBzOi8vbWF0cGxvdGxpYi5vc
mcvLHJYcGAAAAAwSFlzAAAPYQAAD2EBQD+naQAQAAVIRJREFUeJzt3XlVXXUy
VBQEHBFJXctzTTUVwyjdwl0zLLjPamXHOZtN+IW41LZWm5jNVozbbOWO5lqLjhkt
lZrmRBo4pIJaiCh5/nOHqDSxA4FzufT8fj/PgnoVzPzdmu/Od7MZhmEgIiIi4qI8

```

rC5AREREpCQp7IiIiIhLU9gRERERl6awIyIiIi5NYUdERERcmsKOiIiIuDSFHRERE
XFpCjsiIiLi0hR2RERExKup7IjIH7LZbEycONG+P3HiRGw2G2fOnLGuqOsUEBDAQw
89BEDPnj2pWbOmpfVcr0WLFmGz2Th27JjVpYg4JYUdETeU++V4rW3btm1Wl1hoiYm
JeHh4MG7cuHzPT58+HZvNxsqVK3nrrbd48sknARg9ejSvvvpqkd9369atTJw4kdTU
1CLfQ0RKlpfVBYiIdSZPnkxUVFSe47Vr17agmusTHR3Nk08+ycsvv8wDDzxAo0aN7
OeOHZ/O5MmTueeee+jRo4fD7912223X9b5bt2510qRJPPTQQwQHB1/XvYpq0KBBDB
gwAF9fX0veX8TZKeyIuLFu3brRokULq8soNtOmTeOzzz7jySefZNOMtdhsNgCGDh2
Kt7c3s2bNsrjCkuHp6Ymnp6fVZYg4LTVjiUiRnDlzhv79+xMYGEhoaCjDhw/n4sWL
DtdecvnyZKVOmUKtWLXx9falZsybPPfccmZmZ9mtGjhxJaGgohmHYjw0dOhSbzcbs2
bPtx1JSUrDZbMybN++aNQUFBTFr1iy2bNnCM2++CcDSpUtZvvnw506ZNo0qVKly4cI
FRo0YRGRmJr68v9erV46WXXnJ4fzD7Kg0ZMoRPP/2Uxo0b4+vrS6NGjVizZo39mok
TJzJmzBgAoqKi7M2AV/edWbx4Mbfcggt+fn6EhIQwYMAAkpKSHN7rtttuo3Hjxuzf
v58OHTpQvnx5ateuzSeffAJAQkICrVqlws/Pj3r16vHl1186/L767Ij8MYUdETeWl
pbGmTNNhLZffvmlQL/bv39/Ll68yNSpU+nevTuzZ8/miSeecLjmscceIy4ujubNm/
PKK6/QoUMHpk6dyoABA+zXtG/fnrNnz/LNN9/Yj23atAkPDw82bdrkcAzg1ltv/co
6cpuqnn32WY4cOcLw4cNp06YNTz75JIZh0Lt3b1555RW6du3KzJkzqVevHmPGjGHk
yJF57rV582aefvppBgwYwIwZM7h48SL9+vWz/zPq27cv9913HwCvvPIK7777Lu+++
y6VKlUC4IUXXuDBBx+kTp06zJw5kxEjRhAfH8+tt96ap4/PuXPn6NmzJ61atWLGjB
n4+voyYMAAPvzwQwYMGED37t2ZNm0aFy5c40677+b8+fN/9icSkVyGiLidhQsXGkC
+m6+vr80lgDFhwgT7/oQJEwzA6N27t8N1Tz/9tAEY+/btMwzDMPbu3WsAxmOPPeZw
3ejRow3AWLdunWEYhnh69GkDMObOnWsYhmGkpqYaHh4exj333GOEHYXZf2/YsGFGS
EiIkZOT86ef79ixY4a/v78REhJieHt7GwcOHDAMwzA+/fRTAzCef/55h+vvvvvtuw2
azGT/88IPD5/bx8XE4tm/fPgMwXnvtNfuxF1980QCMo0eP5qnB09PTEOGFFxyOHzh
wwPDy8nI43qFDBwMwlixZYj/23XffGYDh4eFhbNu2zX78888/NwBj4cKF9m05f8/f
lyAiJj3ZEXFjc+bMYe3atQ7b6tWrC/S7gwcPdtgfOnQoAKtWrXL4+fsnJqNGjQJg5
cqVAFSqvIn69euzceNGALZs2YKnpdyjxowhJSWFw4cPA+aTnXbt2tn74fyRGjVqMG
HCBM6ePcvIkSNp3LixvSZPT0+GDRuWpybDMPJ89piYGGrVqmXfv+mmmwgMDOTIkSN
/WsN//vMfcNjy6N+/v8OTs/DwcOrUqcP69esdrg8ICHB44lWvXj2Cg4Np0KABrVq1
sh/Pfv2QGkTEpA7KiM7sL3/5S5E7KNepU8dhvlatWnh4eNj7jRw/fhwPD488I7vCw
8MJDg7m+PHj9mPt27e3h6NNmzbRokULWrRoQUhICJs2bSIsLix9+/Zx//33F7i+li
1bAjh8vuPHjxMREUGFChUcrn3QoIH9/NWqV6+e574VK1bk3Llzf/r+hW8fxjCMPP+
ccnl7ezvsV6tWLU+QCwoKIjIyMs8xoEAliIhJYUdEisWlnrgU5ElMu3bteOONNzhy
5AibNm2iffv22Gw22rVrx6ZNm4iIiCAnJ4f27dsXd9l/6FoJnIzfdWbOT05ODjabj
dWrV+d7n4CagAK91/XUICImhR0RKZLDhw87zNHzwW8/kJOTY5+NueAnGuTk5HD48G
H7kxMwR1WlpqZSo0YN+7HcELN27Vp27tzJ2LFjAbMz8rx584iIiMDf359bbrnlumq
uUaMGX375JefPn3d4uvPdd9/ZzxftwcJcrVq1MAyDqKgo6tatW7SCRaRYqM+OiBTJ
nDlZHPZfe+0lwJy7B6B79+4AeWYnnjlzJoDD5H5RUVFUrVqVV155haysLNq2bQuYI
ejHH3/kk08+oXXr1nh5Xd9/n3Xv3p3s7Gxef/11h+OvvPIKNpvNXnth+Pv7A+QZXd
W3b188PT2ZNGlSnqcwhmEUeNSbiFw/PdkRcWOrV6+2P9W4Wps2bbjxxhv/8HePHj1
K79696dq1K4mJiSxevJj777+fpk2bAtC0aVNiY2NZsGABqampdOjQgR07dvD222/T
p08fOnbs6HC/9u3b88EHH9CkSRMqVqWIQPPmzfH39+f7778vVH+da+nVqxcd03bk/
/7v/zh27BhNmzbliyy++4LPPmPEiBEOnZELKvdp0//93/8xYMAAvL296dWrF7Vqle
L5559n3LhxHdt2jD59+1ChQgWOHj3K0qVLeeKJJxg9evR1fyYR+XMKOyJuLC4uLt/
jCxcu/NOw8+GHHxIXF8fYsWPx8vJiyJAhvPjiiw7XvPnmm9x4440sWrSIpUuXEh4e
zrhx45gwYUke++WGnXbt2tmPeXl5ER0dzZdfllks/XU8PDxYtmwZcXfxfPjhhyxcu
JCaNwvy4osv2keJFVbLli2ZMmUK8+fpZ82aNeTk5HD06FH8/f0ZO3YsdevW5ZVXXm
HSpEkAREZG0rlz3r37n3dn0dECsZmqJebiIiIuDD12RERERGXprAjIiIiLk1hR0R
ERFyawo6IiIi4NIUdERERcWkKOyIiIuLSNM805ho2p06dokKFCgVax0dERESsZxgG
58+fJyIiAg+Paz+/UdgBTp06lWdlYRERESkbpKsqFat2jXPK+yAfUHApKQkAgMDL
a5GRERECiI9PZ3IyEiHhX3zo7DDlVWLAWMDFXZERETKmD/rgqIOyiIiIuLSFHZERE
TEpSnsiIiIiEt2BERERGXprAjIiIiLk1hR0RERFyawo6IiIi4NIUdERERcWkKOyI
iIuLSFHZERETEpVkadqZOnUrLli2pUKEClStXpk+fPhw6dMjhmosXLZ48GBQOMJ
cAigX79+pKSkOfxZ4sQJevToQfny5alcuTJjxozh8uXLpflR7CZOhClT8j83Zp5X
kREREqPpWENISGBwYMHs23bNtauXUtWVhadO3fmwoUL9mueeeYZli9fzscff0xCQg
KnTp2ib9+++9vPZ2dn06NGDS5cusXXrVt5+++20WLVpEXFycFR8JT0+Ii8sbeKZMMY9
7elpSloiIiPsynMjp06cNwEhISDAMwzBSU1MNB29v4+OPP7Zfc/DgQQMwEhMTDcMw
jFWrVhkeHh5GcnKy/Zp58+YZgYGBRmZmZoHeNy0tzQCMtLS0YvkckycbBhjGxImGk
ZhoGHFx5v7kycVyexERETEK/v3tVH120tLSAAgJCQFg9+7dZGVlERMTY7+mfV36VK

9encTERAASExNp0qQJYWFh9mu6dOlCeno633zzTS1Wf8X48TB5stlkFR1tvp482Tw
uIiIipctpwk5OTg4jRoygbdu2NG7cGIDk5GR8fHwIDg52uDYSLiZk5GT7NVChndzz
uefyk5mZSXp6usNW3MaPB4///dP18FDQERERsYrThJ3BgwFz9dddf88EH5T4e02dO
pWgoCD7FhkZWezvMWUK5OSYr3NyzCc7IiIiUvqcIuwMGTKEFStWsH79eqpVq2Y/Hh
4eqzVLl0hNTXW4PiUlhfDwcPslvx+dlbufe83vjRs3jrSONPuWlJRujJ/mSmfk8eO
hXDnz2IQJlX6lJSIiIiXH0rBjGAZDhgXh6dKlRfu3jqioKIftz9xyC97e3sTHx9uP
HTp0iBMnThAdHQ1AdHQ0Bw4c4PTp0/Zr1q5dS2BgIA0bNsz3fX19fQkMDHTYiktu0
Mntp3P77ebxmJj8R2mJiIhIyfyKy8s0HDx7MkiVL+Oyzz6hQoYK9j01QUBB+fn4EBQ
Xx6KOPMnLkSEJCQggMDGT00KFER0fTunVrADp37kzDhg0ZNGgQM2bMIDk5mb///e8
MHjwYX1/fUv9M2dmOnZF79oSVK+HXX83j2dmlXpKIiIhbsxmGYVj25jZbvscXLlZi
Qw89BJiTC0a4aNyR333+fzMXMunTpwtY5cx2aqI4fP85TTz3Fhg0b8Pf3JzY2lmnTp
uHlVbAsl56eTlBQEGlpacX6lAcgKQmqVzc7KaekwA03FOvtrURE3FZBv78tDTvOoi
TDDKcZzRbVh7z7LjzwQLHfXkRExC0V9PvbKTooU7qePc2fKlZYW4eIiIg7UtgpBbl
hZ80ayMqythYRERF3o7BTClq2hEqVIC0NtmYxuhORERH3orBTCjw9oXt387WaskRE
REqXwk4p6dHD/KmwIyIiUroUdkpJ587g5QWHDsHhw1ZXIyIi4j4UdkpJUBDceqv5e
uVKA2sRERFxFjwo7pUhd0EVEREQfWk4pyg07CQmQnm5tLSIiIu5CYacU1akDdevC5c
vwXrdWVyMiIuIeFHZKmZqyRERESpfCTinLDTurVmkFdBERkdKgsFPK2rWDwED4739
h506rqxEREXF9CjulzNsbunYlX6spS0RepOQp7FhA/XZERERKj8KOBbp1A5sN9u2D
pCSrqxEREXFtCjsWuOEGiI42X2s2ZRERkZKlsGMRNWWJiIiUDoUdi+SGnfh4+PVXa
2sRERFxFzQo7FmncGKpXh4sXYd06q6sRERFXXQo7FrHZ1JQlIiJSGhR2LHR12DEMa2
sRERFxFVQo7FurYEcqXh5MnzWHOiIiIuVwUdixUrhZExJiv1ZQlIiJSMhR2LKZ+OyI
iIiVLyCdi3bubP3fsgJQUa2sRERFxFRQo7FqtaFZo3NzSor15tdTUIiIKuR2HHCeQ2
ZWnpCBERkeKnsOMEcsPO55/DpUvWlIiIiUJqFHacwC23QFgYnD8PmzZZXY2IiIhrU
dhxAh4e0KOH+VqjSkRERIqXwo6TyG3KwR5csymLiIgUJ4UdJxETAz4+8OOP8P33Vl
cjIiLiOiWNOxs3bqRXr15ERERGs9n49NNPHc5nZGQwZMgQqlWrhp+fHw0bNmT+/Pk
O1ly8eJHBgwcTGhpKQEA/fRlI6UMTlhtToQLcdpv5WklZiIiixcfSsHPhwgWaNm3K
nDlZ8j0/cuRI1qxZw+LFizl48CAjRoxgyJAHLFu2zH7NM88w/Lly/n4449JSEjg1
KlT903bt7Q+QrHSbMoiIiLFz2YYztFDxGazsXTpUvr06WM/1rhxY+69917Gjx9vP3
bLLbfQrVs3nn/+edLS0qhUqRJLlizh7rvvBuC7776jQYMGJCYm0rpl6wK9d3p6OkF
BQaSlpREYGFisn6swjhyBWRXA0xPONIHgYMTKERERCXoF/f526j47bdq0YdmyZZw8
eRLDMFi/fj3ff/89nTt3BmD37tlkZWURk7uaJlC/fn2qV69OYmKiVWUX2Y03QoMGk
JltzrkjIiIi18+pw85rr71Gw4YNqVatGj4+PnTt2pU5c+Zw6623ApCcnIyPjw/Bv3
sEEhYWRnJy8jXvm5mZSXp6usPmLNSUJSIiUrycPuxs27aNZcuWsXv3bl5++WUGDx7
Ml19+eV33nTp1KkFBQfYtMjKymCq+fRlhZ9Uq8wmPiIiIXB+nDTu//fYbzz33HDNn
zqRXr17cdNNNDBkyhHvvvZeXXnoJgPDwcC5dukRqaqrD76akpBAeHn7Ne48bN460t
DT7lpSUVJIfpVDatDH76pw9C9u2WV2NiIhI2ee0YScrK4usrCw8PBxL9PT0JCcnBz
A7K3t7exMfH28/f+jQIU6cOEF0dPQ17+3r60tgYKDD5iy8vKBbN/OlmrJERESun5e
Vb56RkcEPP/xg3z969Ch79+4lJCSE6tWr06FDB8aMGYOfnx81atQgISGBd955h5kz
ZwIQFBTEo48+ysiRIwkJCSEwMJChQ4cSHR1d4JFYzqhnt3j/fTPsTJlqdTUIiIjlm
6VDzds2EDHjh3zHI+NjWXRoKUKJyczbT4vVjiC86ePUuNGjv44okneOaZZ7DZbI
A5qeCoUa4//33yczMpEuXLSydO/cPm7F+zlmGnuc6exYqVYKcHDh6FGrWtLoiERE
R51PQ72+nmWfHSS4WdgBuVdVcAf3112HwYKURERERCT4uMc+009MQdBERkeKhsOOk
csPOunWQkWfTlSiImWZwo6TataAoqLg0iW4arCziIiIFJLCjpOy2dSUJSIiUhwUd
pxYbthZudIcmsUiIiKfP7DjxDp0AH9/+Pln+Oorq6sREREpMxR2nJivL/xvgXclZY
mIiBSRwo6TU78dERGR66Ow4+S6dzd/7tPlNmeJiIhI4SjsOLnwcGjZ0ny9apWltYi
IiJRFCjtlgJqyREREik5hpwzIDTtr10JmPrWlIiIiLDUKO2XAZtDDRARcuAAJCVZX
IyIiUrYo7JQBNhv06GG+VlOWiIhI4SjslBFX99sxDGtrERERKUsUdsqI2283Jxk8e
hQOhrS6GHERkbJDYaeM8PeHTp3Ml2rKEHERKtiFnTJEQ9BFREQKT2GnDMntpLxlC5
w9a20tIiIiZYXCThlSowY0aQI5ObBmjdxViIiIla0KO2WMMrJEREQKR2GnjMkNo6t
Xw+XLltYiIiJSFijslDGtWkFoKKSmtatVlcjIiLi/BR2yhhPT+je3XytpiwREZE/
p7BTBqnffjoiISMEp7JRBnTubT3gOHOqff7S6GHEREEemsFMGBqdD+/bm65UrLSlFR
ETE6Sns1FFqyhIRESkYhZ0yKjfsbNgA589bWoqIiIhTU9gpo+rWhdq1ISsLlq61uh
oRERHnpbBTRtIsasoSEREpCIWdMiW37KxCAA6XJSIiInkp7JRh7dtDhQpw+jTs2mV
lNSIiIs7J0rCzcNGevXqRUREBDabjU8//TTPNQcPHqR3794EBQXh7+9Py5YtOXHi
hP38xYsXGTx4MKGhoQqEBNCvXz9SULJK8VNYx8cHunQxX6spS0REJH+Whp0LFy7Qt
GlT5syZk+/5H3/8kXbt2lG/fn02bNjA/v37GT9+POXKlbnf88wzz7B8+XI+/vhjEh
ISOHXqFH379i2tj2A59dsRERH5YzbDMAyriwCw2WwsXbqUPn362I8NGDAAb29v3n3
33Xx/Jy0tjUqVKrFkyRLuvvtuAL777jsaNGhAYmIirVu3LtB7p6enExQURFpaGoGB

gdf9WUrT6dMQHg6GAT/9BFWrWl2RiIhI6Sjo97ft9tnJyclh5cqVlK1bly5dulC5c
mVatWrl0NSle/dusrKyIImJsR+rX78+latXJzEx0YKqS1/lyuZK6KDZlEVERPLjtG
Hn9OntZGRkMG3aNLp27coXX3zBXXfdRd++fUlISAAG0TkZHx8fgoODHX43LCyM5OT
ka947MzOT9PR0h60sU1OWiIjItTlt2Mn531jqO++8k2eeeYzmZoxduxYevbsyFz5
86/r3lOntiUoKMi+RUZGfKfJlSkNO19+Cb/9Zm0tIiIizsZpw84NN9yAl5cXDRs2d
DjeoEED+2is8PBwLl26RGpqqSMlKSkphIeHX/Pe48aNIy0tzb4lJSUve/2l6aaboF
o1M+isX291NSIiIs7FacOOj48PLVu25NChQw7Hv//+e2rUqAHALbfcgre3N/Hx8fb
zhw4d4sSJE0RHR1/z3r6+vgQGBjpsZdnVsymr346IiIgJLyvfPCMjgx9++MG+f/To
Ufbu3UtISAjVqldnzJgx3HvvvdX666107NiRNWvWshZ5cjZs2ABAUFQJz76KCNHj
iQkJITAwECGDh1KdHR0gUdiuYqePWH+fLPfzuuvmwFIRERELB56vmHDBjp27JjneG
xsLiSWLQLgX//6FlOntuWnn36iXr16TJo0iTvvnN+7cWLFxklahTvv/8+mZmZdOn
Shblz5/5hM9bvleWh57l++w1CQ82f+/dDkyZWVYQiIlKyCvr97Ttz7FjJFcIOQK9e
5pOdf/wDxo2zuhoREZGSVeBn2ZHC0xB0ERGRvBR2XEiPHubPxEQ4c8baWkRERJyFw
o4LqVYNmjUzl45YvdrqakRERJyDwo6LUVOWiIiII4UdF5MbdTasgawsa2sRERFXBg
o7LqZlS6hUCdLTYfNmQ6sRERGXnsKOi/HwuNJRWU1ZiIiIcjsuSf12RERERlDYcUF
33AHe3vD99+YmIiLizhR2XFBgIHToYL7WwqAiIuLuFHZclJqyRERETAo7Lio37Gzc
CGLp1tYiIiJiJYUdF1WrFtSvD5cvwxdfWF2NiIiIdRR2XJiGoIuIiIcjsuLTcpqxVq
yA729paRERERKKw48LatoWgIHMF9B07rK5GRETEGgo7LszbG7p2NV+rKUteRNyVwo
6L0xB0ERFXdwo7Lq5rV3O9rP374cQJq6sREREPfQo7Lu6GGyA62nyt2ZFRMQdKey
4ATVliYiIO1PYcQO5YSc+Hi5csLYWERGR0qaw4wYaNYIaNSAzE9ats7oaERGR0qWw
4wZsNjVliYiI+1LYcRO5YwflSjAMa2sREREPtQo7buK226B8eTh5EVBts7oaERGR0
qOw4ybKlYM77jBfqylLRETcicKOG1G/HRERcUcKO26ke3fz544dkJJibS0iIiKlRW
HHjUREwC23mB2UV6+2uhoREZHSobDjZtSUJSiI7kZhX83khp3PP4dLl6ytrUREpDQ
o7LiZ5s0hPBwyMmDjRqurERERKXmWhp2NGzfSq1cvIiIisNlsfPrpp9e89q9//Ss2
m41XX33V4fjZs2cZOHAggYGBBACH8+ijj5KRkVGyhzdhHh7Qo4f5Wk1ZiIiLiDiwNO
xcuXKBp06bMmTPnD69bunQp27ZtIyIiIs+5gQMH8s0337B27VpWrFjBxo0beeKJJ0
qqZJeq25S1fLlMuxYREdfnzEwbd+vWjW7duv3hNSdPnmTo0KF8/vnn9Mh9JPE/Bw8
eZM2aNezcuZMWLVoa8Npr9G9e3deeuMfMORQEwM+PjAKSNw6BDUr291RSiIiIXH
qfvs50TkMGjQIMAGUOjRo3ynE9MTCQ4ONGedABiYmLw8PBg+/bt17xvZmYm6enpD
ps7CQiAjh3N12rKEhERV+fUYWf69014eXkxbNiWfM8nJydTuXJlh2NeXl6EhISQnJ
x8zftOntTqVoKAg+xyZGVmsdZcFGoIuIiLuwmnDzu7du5klaxaLFi3CZrMV673HjRt
HWLqafUtKSirW+5cFuS2CmzfDuXPWliIiIiKSnDbsbNq0idOnt109enW8vLzw8vLi
+PHjjBolipolawIQHh706dOnHX7v8uXLNd17lvDw8Gve29fXl8DAQIfn3URFQaNgk
JltzrkjIiLiQpw27AwaNIj9+/ezd+9e+xyREcGYMWP4/H/fztHR0aSmprJ7927776
1bt46cnBxatWplVellhpqyRETEHVg6GisjI4MffvjBvn/06FH27t1LSEgIlatXJzQ
01OF6b29vwsPDqVevHgANGjSga9euPP7448yfp5+srCyGDBnCGAEDNBKRAhr2hOnt
zXWyLl8GL0v/lyAiIiIyLH2ys2vXlM6++WZuvvlmAEaOHMnNN99MXFxcge/x3nvvU
b9+fw6//Xa6d+9Ou3btWLBgQUmV7FJat4aQEDh7FrZts7oaERGRkmEzjMJPk5eSks
Lo0aOJj4/n9Ont/P4W2dnZxVZgaUhPTYcoKiI0tDS367/zwAPw3nvw7LMwbZrVlYi
IiBRcQb+/i9Rw8dBDD3HixAnGjx9PlSpVin20lJSenj3NsLNihcKOiIi4piKFnc2b
N7Np0yaaNWTwZOVIaevSBTW94Ztv4OhRc5SWiIiIKylSn53IyMg8TVdSNlWsCG3bm
q9XrrS2FhERkZJQpLDz6quvMnbsWI4d0lbM5YgVNARDRERcWZE6KFesWJFff/2Vy5
cvU758eby9vR3Onz17ttgKLA3u3EEZ4OBBanJQXBz011/MtbNEREScXYl2UH711Ve
LWpc4ofrl4cYbzVXQv/wS+vSxuiIREZHiU6SwExsbW9xliIVsNrMpa/ZssylLYUde
RFxJkefMzc7O5tNPP+XgWYMANGrUiN69e+Pp6VlSxUnpyQ07KldCTg54001CiIiI
oVTPd47P/zWA927d+fkyZP2pRsOHTPEZGQkKleupFatWsVeaEly9z47AJmZcMMNkJ
EBO3dCixZWVYQiIvLHCvr9XaT/fh82bBilatUiKSmJPXv2sGfPHk6cOEFUVBTdhg0
rctFiHV9f6NzZfKlRWSiI4kqKFHYSEhKYMWMGISEh9mOhoaFMmzaNhISEYitOSlfu
EHTNtyMiIq6kSGHH19eX8+fP5zmekZGBj4/PdRcl1uje3fy5axf8/LO1tYiIiBSXI
oWdnj178sQTT7B9+3YmW8AwDLZt28Zf//pXevfuXdw1SikJC40//MV8vWqVtbWiiI
gUlyKFndmzZlOrVi2io6MpV64c5cqVo23bttSuXZtZs2YVd41SijSbsoiIuJoiJcb
KdfjwYb777jsAGjRoQO3atYutsNKK0VhXbfPUVNG80/v5w5gyUK2d1RSiIiIvkr0RmU
c9WpU4c6depczy3EyTRrBHERcOoUJCSYq6KLiiUzQUOoYNHjMTklCn4+/szcuTIP
7x25syZ112YWCN3NuUFC8ymLIUdEREP6wocdr766iuySrLsr8V1XR12Zs82A5CiE
hZdV19dlyF+uw4+vVXCA2Fixfh66+hUSOrKxIREcmrRGdQfuSRR/KdZ+fChQs88sg
jRbmLOJHy5aFTJ/OlRmWJiEhZV6Sw8/bbb/Pbb7/lof7bb7/xzjvvXHdRyJ0NQRcR
EVdRqNFY6enp9kkEz58/T7mrxiVnZezatUqKleuXOXFSunr0cP8uXUr/PKL2awlI
iJSFhUq7AQHB2Oz2bDZbNStWzfPeZvNxqRJK4qtOLF09epw002wfz+sWQMDB1pdkY

iISNEUKuysX78ewzDo1KkT//73vx0WAvXx8aFGjRpEREQUe5FijZ49zbCzYoXCjoi
IlF2FCjsdOnTg8uXLxMbG0qJFCyIjI0uqLnECPXvCP/5hPtnJygJvb6srEhERKbxC
d1D28vLik08+ITS7uyTqESfyl7/ADTDaaqrZd0dERKQsKtJorE6dOpGQkFDctYgTm
TjRfKrTvb5f/WorClTzPMiIiJlQZHWxurWrRtjx47lwIED3HLLLfj7+zuc7927d7
EUJ9bx9IS400jf39xfsQJefNEMOnFxmHmytFWJiIgUVJFmUPbwuPYDIZvNVuaauDS
Dcv5yg42HB+TkWPDhMGUWGXTGj7e6OhERCXcluup5Tk5OkQuTsiM30MTFmT9nzYJx
4xR0RESkbClSnx1xH+PHO47C+uAD2LvXsnJEREQKrchhJyEhgV69elG7dm1ql65N7
9692bRpU3HWJk5gyhTHYedHj0JONGhVEBERKSuKFHYWL15MTEwM5cuXZ9iwYQwbNg
w/Pz9uv/12lixZUtwlikWu7ox86ZLZhAXmauixsTBkiHlcRETEqRlFUL9+fWPmzJl
5jr/88stG/frlC3yfhIQEo2fPnkaVKlUMWfi6dKn93KVLl4y//e1vRuPgjY3y5csb
VapUMQYNGmScPHnS4R6//PKLcf/99xsVKlQwgoKCjEceecQ4f/58oT5PWLqaARhpa
WmF+j1XNnmyYYD582qTJpnHc7c2bQzjd38SERGRU1HQ7+8iPdk5cuQIvXr1ynO8d+
/eHD16tMD3uXDhAk2bNmXONd15zv3666/s2bOH8ePHs2fPHv7zn/9w6NChPMPaBw4
cyDffMPatWtZsWIFGzdu5Iknnij8hxIH2dn5j7rKfdJz330QFGRONnjLLaAWTBER
cVpFSVKlatUy5s+fn+f4vHnzjNqlaxfllnme7ORnx44dBmAcP37cMAZd+Pbbbw3A2
Llzp/2a1atXGzabLc8ToD+iJztFc/iwYTRubD7h8fIyJFmzDCMnx+qqRETEXRT0+7
tIQ89HjRrFsGHD2Lt3L23atAFgy5YtLFq0iFmzZhVXDssjLS0Nm81GcHAwAImJiQQ
HB90iRQv7NTEMXh4eLB9+3buuuuufO+TmZlJZmamfT89Pb3EanZltWvDtm3w+OPw
/vvmPDzbt8Mbb0D58lZXJyIiYipS2HnqqacIDw/n5Zdf5qOPPqKgQYMGfPjhh9x55
53FWmCuixcv8uyzz3LffffZJw5KTK6mcuXKDtd5eXkREhJCcnLYNe81depUJk2aVC
Jluht/f3jvPXMdrdGjYckS+Ppr+M9/oFYtq6sTERG5jqHnd911F5s3b+aXX37h119
+YfPmzSUWdLKysujfvz+GYTBv3rzzvt+4ceNIS0uzb0lJScVQpfuy2WDECFi3DsLC
YP9+aNECVq2yujIREZEihp0bb7yRX375Jc/x1NRUbrzxxusu6mq5Qef48eOsXbvWY
Tro8PBwTp8+7XD95cuXOXv2LOHh4de8p6+vL4GBgQ6bXL9bb4Xdu6F1a30l9J49Yd
Ikc6kJERERqxQp7Bw7dizf9a8yMzM5efLkdReVKzfoHD58mC+/JLQ0FCH89HR0aS
mprJ79277sXXrlpGtK0OrVq2KrQ4puKpVISEBnn7aHJw+cSL07g3nzl1ldmYiIuKtC
9dlZtmyZ/fXnn39OUFCQfT8705v4+HhqlqxZ4Pt1ZGTwww8/2PePhj3K3r17CQkJo
UqVKtx9993s2bOHFStWkJ2dbe+HEXISgo+PDw0aNKBr1648/vjjzJ8/n6ysLIYMGc
KAAQOIiIgozEeTYuTjA3PmmP14/vpXWLkSWrY0+/HcdJPVlYmIiLsp1Krnuaud22w
2fv9r3t7elKxZk5dfpmePXsW6H4bNmygY8eOeY7HxsYyceJEoqKi8v299evXc9tt
twFw9uxZhgwZwvLly/Hw8KBfv37Mnj2bgICAg4srXpegr76Cvr2hWPHwM8P3nwT7
r/f6qpERMQVFPT7u1BhJldUVBQ7d+7khhtuuK4inYXCTsn65Rcz4Hzxhbk/YgTMmO
G4wKiIiEhhFft7u0h9do4ePZon6KSmpbhlVuIGQkPNkVnPPWfuv/oqxMTAH8wOICI
iUmyKFHamT5/Ohx9+a+/5557CAkJOWrVquzbt6/YihPX4ekJL7wAS5dChQqWcaO5
zERiotWViYiIqytS2Jk/fz6RkZEArF27li+/Jl1a9bQrVs3xowZU6wFimvp0wd27
oSGDeHUKejQAebNM0duiYiIliQihZ3k5GR72FmxYgX9+/enc+fO/0lvf2Pnzp3FWq
C4nnrlzGUmr4bsrLMYeoPPwy//WZlZSIi4oqKFHYqVqxon3V4zZolxMTEAGAYRr7
z74j8XoUK8NFH8OKL4OEBb78Nbdc0aNNVYyIiIq6mSGGnb9++3H//dxxxx388ssv
dOvWDYCVvvqK2rVrF2uB4rpsNnM9rbVr4YYbzGHqLVpcGbUlIiJSHIoUdl555RWGD
BlCw4YNWbt2rXlOm59//pmnn366WASU19epE+zZY048ePYsd0lqdmBWMhMiIlicij
TPjqvRPDvO4eJFGDYm3njD3L/zTrN566qJukVEROWK+vld4OUililbRrdu3fd29nZ
YNiI/vXv3LnliIv9TrhwsWGAuMzF4MHZ2mfl66VJz9JaIiEhRFPjJjoeHB8nJyVSu
XNm+bES+N7TZylwnZT3ZcT47d0K/fpCUBP7+sHAh3HOP1VWJiIgZKfYzLHNycqhc
bL99bW2shZ0xDmlbAm7d5v9eS5cgP79YcwYhZz6spERKSsKXQH5ZychP71r3/Rs2
dPGjduTJMmTbjzzjt555138iwOKni9KlWCzz+Hv/3N3H/pJejcGU6ftrYuEREpWwo
VdgzDoHfv3jz22GocPHmSjk2a0KhRI44d08ZDDz3EXXfdVvJlipvy8oLp0+HjjyEg
ANavN5eZ2LHD6spERKSsKHAHZYBfixaxceNG4uPj6dix0805devW0adPH9555x0ef
PDBYi1S5O67oVEjuOsuOHQI2reh11+Hxx+3ujIREXF2hXqy8/777/Pcc8/lCToAnt
p1YuzYsbz33nvFVpzI1Ro0MJ/o3HUXXLoETzwBjz1mDlkXERG5lkKFnf379901a9d
rnu/WrZtWPZcSFRgi//43TJlqLjPx1lvuU54TJ6yuTEREnFWHws7Zs2cJCwu75vmw
sDDONt33UWJ/BGbDcaOhTVrIDQUduOy+/Hex1tdmYiIOKNChZ3s7Gy8vK7dzcft0
5PLGhsspeS008zh6c2bw5kz5kitGTNAgyJFRORqheqgbBgGDz30EL6+vvmesz8ZMLJ
aiRAqQrg3YvBmefhoWLYJnnzX79SxagK6sLiLiUqiWExsb+6fXaCSWlDY/P/jXv6B
VK3NtrX//G779Fv7zh6hf3+rqRETEaloIFC0X4Uq2bTOHqZ88aT7Zefttc/SWiIi4
nmJfLkKkLGjd2uzH06EDnD8PfftCu3YwaVL+10+ZAhMnlmqJiIJSyhr2xOWehcHat
TBypLm/ZYsZaMaNc7xuyhSiwNPz1IvUURESPhCjrgkb294+WX44AMOX948Nm0aPP
WU+To36EyeDOPHWleniIiUPPXZQX12XN3XX5v9dn74wdz39ITsBAUdeZGyTn12RP6
ncWPYuRN69TL3s7PNn998Y3ZoFHER16awI24hOBhatDBf22zmzw8/hOhos1PzBx9A

VpZl5YmISAlS2BG3MGUKTJhgNl3l5JiTEILZpLV9O9x3H0RFmWtu/fKLtbWKiEjxU
tgRl5dfZ+Q5c8z97Gzo2NEcwXXyJDz3HERGwpNPmhMTiohI2aewIy7vWp2Rx483j9
96Kxw/bk5AePPN8NtvsGABNGoEXbrA6tXm0yARESmBNBoLjcaSKwwDNm2CWbPg00+
vhJx69WD4cHjwQfD3t7REERH5H43GEikCm8l80vPvf5tDlUeOhMBAOHTI7OdTrZq5
2OiJE1ZXKiIiBaWwI3INUVHmxIQ//QSzZ0OtWpCaCjNmWI03Qv/+sHWr+TRIREScl
6VhZ+PGjftQlYuIiAhsNhuffvqpw3nDMIiLi6NKlSr4+fkRExPD4cOHHa45e/YsAw
cOJDAwkODgYB599FEyMjJK8VOIq6tQAYYONZ/uLFsGnTqZ/YA+/hjatjVXWl+yBC5
dsrpSERHJj6Vh58KFCzRt2pQ5c+bke37GjBnMnj2b+fPns337dvz9/enSpQsXLl60
XzNw4EC++eYblq5dy4oVK9i4cSNPPPFExX0EcSOenubEhPHxsG8fPPoo+PqaExYOH
Gg+CfrHP+DMGasrFRGRqzlnB2WbzcBspUvp06cPYD7ViYiIYNSoUYwePRqAtLQ0ws
LCWLROEQMGDODgWYM0bNiQnTt30uJ/M8atWbOG7t2789NPPxEREVGg9lYHZSmq//4
X/vlPcyh7crJ5rFw5eOABs0Nz48bWlici4srKfAflo0ePkpypcTExmJp1YUFAQRvQl
IjExEYDExESCg4PtQQcgJiYGDw8Ptm/ffs17Z2Zmkp6e7rCJFEWlSvD3v5tDl999F
5o3h4sX4c03oUkTuOMOWLlSQ9dFRKzktGEN+X//mRwWFuZwPCwszH4u0TmZypUrO5
z38vIiJCTEfkl+pk6dSlBQkH2LjIws5urF3fj4mE9zdu0yh6736wceHvDl19CzJ9S
vbz79UXcyEZHS57RhySNGzeOtLQ0+5aUlGRlSeIibDZolw4++QR+/BFGj4agIDh8
GIYMMYeuJx1jPgkSEZHS4bRhJzw8HICULBSH4ykpKfZz4eHhnd592uH85cuXOXv2r
P2a/Pj6+hIYGoiwiRS3mjXhxRfNoeuvvw5l6kBaGrz0kjl0/Z57YPNmDV0XESlpTh
t2oqKiCA8PJz4+3n4sPT2d7du3Ex0dDUB0dDSpqans3r3bfs26devIycmhVatWpV6
zSH4CAmDwYPjuOlixAmJizD48n3wC7dtDy5aweLGGrouIlBRLw05GRgZ79+5l7969
gNkpee/evZw4cQKbzcaIESN4/vnnWbZsGQC0HODDBx8kIiLCPmKrQYMGd03alccff
5wd03awZcsWhgwZwoABAwo8EkuktHh4QI8esHYtHDGajzlmjtzavRsGDYIaNeD558
0RXiIiUnwsHXq+YcMGOnbsmOd4bGwsixYtwjAMJkyYwIIFC0hNTaVdu3bMnTuXunX
r2q89e/YsQ4YMYfny5Xh4eNCvXz9mz55NQEBAgevQ0HOxypkz5qKjr78OP/9sHvP1
vTJ0vUkTa+stEXFmBf3+dpp5dqyksCNWu3TJbNZ65RVzRFeu2283Q0+PHuYK7Z6ee
VdvB5gyxZzVeeLEUitZRMryZX6eHRF34uMD998PO3bAlilm52UPD3O25t69zVXXd+
6EuDgz2FxtyhTzuKenNbWLiDg7hR0RJ2KzQZs28NFHcOSIOUw9ONhcgX3VKrOJKy7
OXI0drgSdyZPzf+IjIiJqXgLUjCXO7cIFeOcdPVV+P77K8c9Pc2mKwUdEXFXasYS
cRH+/vDUU3DwoPl0p3Nn83h2ttnUNW6ctfWJiDg7hR2RMsLDA7p1M2dozpWTAzfdZ
K7HJSIi+VPYESlDru6j85//mElZBw9Cw4ag9WxFRPKnsCNSRvy+M/Jdd5kTfPr4wN
Gj0KCBjiQUEcmPwo5IGZFfz+SOHWHrVihfHk6dMpu4TpywrkYREWek0VhoNJaUfYc
OmR2XT5yAqlXhiy/Mpi0REVem0VgibqRePXMWgYN4ORJc4HR7dutrKpExDko7Ii4
iGrVYNMmaNUKzp41l5pYu9bqqkRErKewI+JCQkPhyy/hjjvMyQh79ICPP7a6KHERa
ynsiLiYgABYvhz694esLLj3Xpg/3+qqRESso7Aj4oJ8fWHJEnjySTAMcwbmF14wX4
uIuBuFHREX5ekJ8+bB3/9u7v/97+YCojk51tYl1lLaFHZEXJjNZk5G+Mor5v6rr0J
srNm8JSLiLhR2RNzAiBHmyumenrB4MfttC7/9ZnVVIiKlQ2FHxE0MGgSffgrlysGK
FdClC6SmWl2ViEjJU9gRcSM9e5qzKwcFmXPY3HYbJCdbXZWISMlS2BFxm+3bQ0Ich
IXBvn3melpHjldlYhIyVHYEXFDTZuay0tERcGPP0LbtrB/v9VViYiUDIudETdVq5
YZeJo0MZuyOnQw90VEXI3CjogbqlLFbNjQ29bsrHzHHbBqlDVViYgUL4UdETdXsaL
Zabl7d3M4+p13wnvvWV2ViEjxUdgREcqXN4elP/AAXL5s/pw92+qqRESKh8KOiADg
7Q1vvw3Dhpn7w4dDXJzW0xKRsk9hR0TsPDzMJSUMtzb3p0yBwYmH09vSskRErovCj
og4sNlg/HiYO9d8PW8eDBwIly5ZXZmISNeo7IhIvp56Ct5/32ze+vBD6NULllywui
oRkcJT2BGRa7r3Xli+3Oza/MUXEBMDZ89aXZWISOEo7IjIH+rSBeLjzSHq27bBrbf
CyZNVVYUiUnAKOyLyp1q3NhcoJYiAb74xJyE8fNjqqkRECKzhR0QKpFEjczmJOnXg
+HEz8Hz1ldVViYj8Oac009nZ2Ywfp56oqCj8/PyoVasWU6ZMwbhq4g/DMIiLi6NKl
Sr4+fkRExPDYf0np0iJqFkTNm+Gm2+G//7XXE9rwwarqxIR+WNOHXamT5/OvHnzeP
311zl48CDTP09nxowZvPbaa/ZrZsyYwezZs5k/fz7bt2/H39+fLl26cPHiRQsrF3F
dlSubAadDBzh/Hrp2hc8+s7oqEZFrsmG886P2rNnT8LCwnjrrbfsx/r164efnx+L
Fy/GMAwiIiIYNWoUo0ePBiAtLY2wsDAWLVRgEAEDCvQ+6enpBAUFkZaWRmBgYl18F
hFXc/Ei3HefucyEhwe8+SY8/LDVVYmIOyno97dTP9lp06YN8fHxfP/99wDs27ePzZ
s3061bnWCOHj1KcnIyMTEx9t8JCgqiVatWJCymXvO+mZmZpKenO2wiUjjlysHHH5s
BJycHHnkEXnrJ6qpERPLysrqAPzJ27FjS09OpX78+np6eZGdn88ILLzBw4EAAkpOT
AQgLC3P4vbCwMPu5/EydOpVJkyaVXOEibSLLC956C264AV58EcaMgTnNYOpUC/ZlE
RFn4NRpdj766CPee+89lixZwp49e3j77bd56aWXePvtt6/rvuPGjSmtLc2+JSU1FV
PFiU7HZoMZM2D6dHN/+nR4/HFz9XQREWfglE92xowZw9ixY+19b5o0acLx48eZOnU
qsbGxhIeHA5CSkkKVklXsv5eSkKzZs2ueV9fX198fX1LtHYRd/O3v0FICDz5pPm0
5+xZWLLbE04SEbGSUz/Z+fXXX/HwcZr090TnJwcAKKioggPDyc+Pt5+Pj09ne3bt
xMdHV2qtYoIPPaY2Y/HxweWLoUePcwRWYiIvNLqsNOrVy9eeOEFVq5cybFjxli6dC

kzZ87krrvuAsBmszFixAief/55lilbxoEDB3jwwQeJiIigT58+1hYv4qb69oXVqyE
gANatg06dzDl5RESS4tRDz8+fP8/48eNZunQpp0+fJiIigvvuu4+4uDh8fHwAc1LB
CRMmsGDBAlJTU2nXrh1z586lbt26BX4fDT0KXK67dkG3bmaH5Xr1zIVEqle3uioRc
SUF/f526rBTWhR2RErGoUNwxx2QlATVqpmBp0EDq6sSEVfhEvPsiEjZVq+euZ5Wgw
bw00/Qvj3s2GF1VSLibhR2RKRErUaaK6b/5S/wyy/Qrh089FD+106ZAhMnlmZ1IuI
OFHZEPMsFhkJ8vNmklZUFb78Nvl/NZcoUiIsDT09rahQR16WwIyKlIiAAlI+He+4x
9z/8Ehr3Nl/nBp3Jk2H8eOtqFBHX5NSTCoqIa/H1hfffNycf/Oc/zfDj6WmurfX3v
yvoiEjJ0JMdESlVnp4wb54ZbsAMOmCurdWzJ7zxBvzB0nYiIoWmJzsiUupsNnOWZT
DDT3Y2ZGbCypXmZrNBq1Zw553mVr++FhYVkaLTkx0RKXVX99G5fBkmTTKP3347tGw
JhgHbtsG4cdCwoTmEfcwY2LzZDEYiIoWhsCMipSq/zsi5+/Hx0KuXOSfPvHnQtav5
BOjwYXjpJXOenipV4JFH4LPP4NdfRf0sIlI2aAZlNI0ySGmaONFsusqvM/KUKeaTm
6vn2klPh88/N8PNypWQmnrlnJ+fOZz9zjvNkFSpUgkXLYJORctFFILCjkjZkJVlTl
D42Wfmdvz4lXM2G7Rpc6WfTyGWxxORMkphpxAUdkTKHsOA/fuvBJ89exzP169/Jfi
0agUearQXcTkK04WgsCNS9iUlwbJlZvBZv97s+JwrLMxs5rrzTrMTtJ+fdXWKSFPFR
2CkEhR0R15KWBqtXm8Fn1Sqz30+u8uWhSxcz+PTsaS5lISJlk8JOISjsiLiuS5cgI
eFKc9dPP1055+FhLkya29xVq5Z1dYpI4SnsFILCjoh7MAz46qsrwWffPsfzjRpdCT
4tWqifj4izU9gpBIUdEfd07NiVfj4JCY4TFkZEXOnn06mTua6XiDgXhZ1CUNgRkXP
nzP49n3l9vfJyLhyLiDAnODwzjuhRw+oWLHw8wWJSPEr6Pe3HtKKiGAGmIED4aOP
4MwZM/D89a/me56MDPjkExg0yJy4sFMn2LHDnPl5yhTH++TOEO3pac3nEJG89GQHP
dkRkWvLyYHdu6/08/n667zXdOgAzz8PX3xhhp2rl8IQkZKjZqxCUNgRkYL68ccr/X
w2bTLD0NUiIqBvX2jdGqKjISpKK7aLlBSFnUJQ2BGRovjlF309rocfzht6clWqdCX
4tG5truoeEFC6dYq4qoJ+f3uVYk0iIi4lNNRcnysnx1yd/diluOceqFoVEhPNJSz+
+19YvtzCwBzO3rjx1fDTurW5jpeGuYuUHP3fS0SkiHI7IO+eDJmZ5s+PP4aQENi2z
Zy50TERXnkF+veH6tXNYLR/P/zzn+YtoQYN4IYboFs3mDTJXOH93DmrP5mIa1EzFm
rGEpHCuzroXN0Z+VrHc506Bdu3myFo2zbYtQt++y3vdQ0aXHnyEx0NDRtqhJfI76n
PTiEo7IhIYRXPDtZWXdGwJXws20b/PBD3usCAuAvf7kSflqlMvsDibgzhZ1CUNgR
EWfy3/+aT39yw8/27Y6THOaqVcux8/NNN4G3d+nXK2IVhZ1CUNgREWEWnQ3ffnsl/
CQmwsGDea/z8zPX9Mpt/mrd2hwKL+KqFHYKQWFHRMqa1FRzFufc8LNTm3ns96pXdw
w/zZs7rvOlZS+kLNPQcxERFxYcDJ07mxuYo7wOH3YMPwcOwIkT5vbRR+Z1Pj5w881
Xws/58zBzpnnuWh2tRco6PdlBT3ZEXDVLZJijvXLDt2Ki2R/o9wICzGvvuAP+/nf4
8ksteyFlg5qxCKfHr0TcgWHASWOOI7+++gouX857bdWq5txAbdpA27ZQpUqplyvyp
xR2CkFhR0Tc1W+/mTM9b9sGf/vbtZe9qFnTDD1t2phbkya90esV9Dvb6efQfnkyZ
M88MADhIaG4ufnR5MmTdila5f9vGEYxMXFUaVKFfz8/IiJieHw4cMWViwiUnb4+Zk
h5tdfryx7AXD33fD009CsmbmUxbFj8N57Mhiw2ecnONhs9po40VztPT3dus8g8mec
OuyC03eOtm3b4u3tzerVq/n22295+eWxqVixov2aGTNmMHv2bObPn8/27dvx9/enS
5cuXLx40cLKRUTKjvyWvfjkEwgPN5u5zp2DtWvNYNO5M1SoYPbx+fJLc4mLL13M8N
O0KTz1FCxeDeeOmMlms7AqZuxxo4dy5YtW9i0aVO+5w3DICIiglGjRjF69GgA0tL
SCAsLY9GiRQwYMKBA76NmLBfxV0VZ9iI7G775BrZuhS1bzJ9HjuS9d3j4lT4/bdqY
T4SuHvYucr1cos9Ow4YN6dKlCz/99BMJCQlUrVqVp59+mscfxfyAI0eOUkTWLb766
iuaNWtm/70OHTqrFkzZs2ale99MzMzyCzMtO+np6cTGRmpsCMibqe45tn5+Wez43
Nu+Nm921wK42q+vtCy5ZUAFB2tJS/k+rhe2ClXrhWAI0eO5J577mHnzp0MHZ6c+fP
nExsby9atW2nbt12nTp2iy1VDBfr374/NZuPDDz/M974TJ05k0qRJeY4r7IiIFI/f
fjMDz9VPf86cyXtd3bqOT3/q1zf7CikUheUEHR8fHlq0AMHwVvtx4YNG8bOnTtJT
EwsctjRkx0RkdJlGOakh1eHn2+/zXtdxYrme5/cANSyJfj71369Uja4xAzKVapUoW
HDhg7HGjRowL//W8AwsPDAUhJSXEIOykpKQ7NWr/n6+uLrxqORURKjclmPsWpWxc
eesg8dvasOeQ9NwBt3252hl61ytzAbGJrluzKk5+2baFatbz317IX8kec+mFh27Zt
OXTokMOx77//nholagAQFRVFeHg48fHx9vPp6els376d6OjoUq1VREQKJyQEuneH5
5+H9eshLQ127oRZs8wJDatWNUPK7t0wezYMGACRkeZ6X/fdB6+9Zs4RdPmyGXTi4s
xgc7Xcjtaae8jNGU5sx44dhpeXl/HCCy8Yhw8fnt577z2jfPnyxuLfi+3XTJs2zQg
ODjY++++wzY//+/cadd95pREVFGb/99luB3yctLc0AjlS0tJL4GCIiUkTHjxvG++8b
xpAhhtG8uWF4ehqG2Sh2ZStf3jA6djSMDh3M/eeeM3938mRzf/JkS+ClKCCfn87d
Z8dgBURvJBU3DgOHZ5MVfQUI0eOtI/GANP4+YQJElIwYAGpqam0a9eOuXpNurdu3Q
K/h4aei4iUDRKZ5mrVU1fiYnmE6Hfs9nMKNS1K4wdaw5717/eXY9LdFAuLQo7IiJ
1U0602dH56o7PP/yQ/7V16sAtt0Dz5ubPm28200RL2aWwUwgKOYIiriG3j46X19mX
p359uHABkpLyvz4qyjEANW8ON9xQujVL0bnEaCwREZGC+v2sz1fv//Wv5tIXu3ebn
Zp374ajR69sn3xy5T6RkXkD0P8G/0oZpbAjIiJlXn7LW+T+jIu7st+585XfOXcubw
A6fNh8CpSUBJ9+euXaKlXyBqCqVc2+QeL81IyFmrFERMq64ppnJz3dDEB79lwJQN9
9l/+ippUrm6Hn6gBUo4YCUGlSn51CUNgREZFryciAffscA9C335oB6vdCQvIGoFq1

FIBKisJOISjsiIhIYfz2Gxw44NgE9vXXeRc/BQgKMkd+XR2A6tbNfw0wzQRdOOqgL
CIiUkL8/OAvfzG3XJmZzuC5+gnQ/v3mPEAbNphbroAAcxmM3PDTvLk5cix3JmhwDD
xX90mSwtOTHfRkR0RESkZWltnkdXUA2rvXfDL0e35+0LSpOXfQjh3w1FPM0hnTpuX
tfC0mNWMVgsKOiIiUluxeOHTIsQnsq6/MvkHX0qED/N//mSvCBwSUXq30TmGnEBR2
RETESjk55rD3q58ArV+f9zpPT7P/T7t20L69+bNy5dKv11ko7BSCwo6IiDiT3D463
t5mU9hNN5l9f44fz3ttvXpm8MndatZ0n9FfBf3+zqcvuIiIiFjl6s7Ily6ZP/fvh0
cfhRMn4L33zBmhGzc2rz90CN58E2Jj4cYbzRmg77sP5s41R4z15Fj7eZyBRmOJiIg
4iYLOBH3//ebrs2fNBVA3bTK3Xbvg5En44ANzAwgOhrZtrzz5adECfHxK9WNZTs1Y
qBlLREScw/XOs/Prr7B9+5Xwk5hoLoR6tXLloFWrK+EnOhoqVCjOT1F61GenEBR2R
ETEFV2+bI702rz5SgA6c8bxGg8Pc86fq/v9lJVOzwo7haCwIyIi7sAwzD4+ucFn0y
Y4dizvdXXrOoafqCjn7PSssFMICjsiIuKufvrJMfx8/XXeayIirgx1b98emjTJf7m
L0qawUwgKOyIiIqZz5/J2ev79m19BQXk7Pfv6Xj1fWmt8aW0sERERKbSKFaFnT3MD
s9Pzjh2OnZ7T0mDVKNMDs9PzX/5y5enP5cswaZJ5zhnW+NKTHfRkR0REpKAuXzbX9
7q60/N//+t4jYcHhIXBzz/Dvfeaa3wtWFD8a3ypGasQFHZERESKxjDg++8d+/0cPZ
r/tcW9mKnCTiEo7IiIiBSfkyevBJ9588xA501tZghdnLRchIiIiFiialUYMADCw82
g4+NjdnKeMsWaehR2REREPNh3Rk5M9P8GRdnTeDRaCwREREPVgVd46u0KOyIiIhI
scrOzr8zcu5+dnbplqM0yqiDsoiISfMkDsoiIiIiKOyIiIiI1PYEREREZemsCMiI
iIuTWFHREREXJrCjoiIiLi0MhV2pk2bhs1mY8SIEfZjFy9eZPDgwYSghhIQEEC/fv
1ISUmrxkgRERFXmUm7OzcuZN//vOf3HTTTQ7Hn3nmGZYvX87HH39MQkICp06dom/
fvhZVKSIIiIs6mTISdjIwMBg4cyBtvvEHFihXtx9PS0njrrbeYOXMmnTp14pZbbmHh
woVs3bqVbdu2WVixiIiIOIsyEXYGDx5Mjx49iImJcTi+e/dusrKyHI7Xr1+f6tWrk
5iYeM37ZWZmkp6e7rCJiIiIa3L6tbE++OAD9uzZw86dO/OcS05OxsFhH+DgYIfjYW
FhJCcnX/OeU6dOZdKkSXmOK/SIiIiUHbnf23+28pVTh52kpCSGDx/O2rVrKVeuXLH
dd9y4cYwcOdK+f/LkSRo2bEhkZGSxvYeIiIiUjvPnzxMUFHTN804ddnbv3s3p06dp
3ry5/Vh2djYbN27k9ddf5/PPP+fSpUukpqY6PN1JSUkhPDz8mvf19fXF19fXvh8QE
EBSUhIVK1TAZrOVyGcPy9LT04mMjCQpKUKLPtoJ/U2ci/4ezkV/D+dSkn8PwzA4f/
48ERERf3idU4ed22+/nQMHDjgce/jhh6lfvz7PPvsskZGReht7Ex8fT79+/QA4dOg
QJ06cIDo6usDv4+HhQbVq1Yq1dlcUGBIOF3E4Gf1NnIv+Hs5Ffw/nUlJ/jz96opPL
qcNoHqoVaNy4scMxf39/QkND7ccffRRRo4cSUhICIGBgQwdOpTo6Ghat25tRckiI
iLiZJw67BTEK6+8goeHB/369SMzM5MuXbowd+5cq8sSERERJ1Hmws6GDRsc9suVK8
ecOXOYM2eONQW5AV9fXyZMmODQz0mspb+Jc9Hfw7no7+FcnOHvYTP+bLyWiIiISBl
WJiYVFBERESkqhR0RERFXaQo7IiIi4tIUdkRERMSlKezINU2dOpWWLVtSoUIFKleu
TJ8+fTh06JDVZcn/TJs2DZvNxogRI6wuxW2dPHmSBx54gNDQUPz8/GjSpAm7du2yu
iy3lZ2dzfjx44mKisLPz49atWoxZcqUP103SYrHxo0b6dWrfXEREdhsNj799FOH84
ZhEBcXR5UqVfDz8yMmJobDhw+XSm0KO3JNCQkJDB48mG3btrF27VqysrLo3LkzFy5
csLo0t7dz507++c9/ctNNN1ldits6d+4cbdu2xdvbm9WrV/Ptt9/y8ssvU7FiRatL
clvTp09n3rx5vP766xw8eJDp06czY8YMXnvtNatLcwsXLlygadOm15wKZsaMGcyeP
Zv58+ezfft2/P396dKlCxcvXizx2jT0XArsv//9L5UrVyYhIYFbb73V6nLcVkJZGBs
2bN2fu3Lk8//zzNGvWjFdfdxqstzO2LFj2bJlC5s2bbK6FPmfnj17EhYwXltvWU
/lq9fP/z8/Fi8eLGLf1bkfm83G0qVL6dOnD2A+1YmIiGDUqFGMHj0agLS0NMLCwli0
aBEDBgwo0Xr0ZEckKLC0tDYCQkBCLK3FvgwcPpkePHsTExFhdiltbtmwZLVq04J577
qFy5crcfPPNvPHGG1aX5dbatGldfHw833//PQD79ulj8+bNdOvWzeLK5OjRoyQnJz
v8eysoKIhWrVqRmJhY4u9f5mZQFmvk5OQwYsQI2rZtm2e9Mik9H3zwAXv27GHnzp1
Wl+L2jhw5wrx58xg5ciTPPfccO3fuZNIwYfj4+BAbG2tleW5p7NixpKenU79+fTw9
PcnOzuaFF15g4MCBVpfm9pKTKwEICwtzOB4WFmY/V5IUdqRABg8ezNdf83mzZutL
sVtJSULMXz4cNauXUu5cuWsLsft5eTk0KJFC/7xj38AcPPNN/P1118zf/58hR2Lfp
TRR7z33nssWbKERO0asXfvXkaMGEFERIT+Jm5OzVjyp4YMGcKKFStYv3491apVs7o
ct7V7925Onz5N8+bN8fLywsvLi4SEBGBpno2XlxfZ2dlWl+hWqlSpQsOGDR2ONWjQ
gBMnTlhUkYwZM4axY8cyYMAAmjRpwqBBg3jmmWeYOnWqlaW5vfdwcABSULicjgekp
NjPlSSFHbkmwzAYMmQIS5cuZd26dURFRVldklu7/fbbOXDgAHv37rVvLVq0YODAge
zduxdpT0+rS3Qrdbu2zTMVw/fff0+NGjYUsqkh+/fVXPDwcv9Y8PT3JycmxqCLJFRU
VRXh4OPHx8fzj6enpbN++nejo6BJ/fzVjYtUNHjyYJUuW8Nlnn1GhQgV7v2pQUBB+
fn4WV+d+KlSokKe/1L+/P6GhoephZYfnnnmGNm3a8I9//IP+/fuzY8cOFixYwIIFC
6wuzW316tWLF154gerVq9OoUSO++uorZs6cySOPPGJ1aW4hIyODH374wb5/9OhR9u
7ds0hICNWrV2fEiBE8//z1KlTh6ioKMaPH09ERIR9xFaJmKSuaAch3W7hwodWlyf9
06NDBGD58uNVluK3ly5cbjRs3Nnx9fY369esbCxYssLokt5aenm4MHZ7cqF69ulGu
XDnjxhtvNP7v//7PyMzMtLo0t7B+/fp8vzNiY2MNwzCMnJwcY/z48UZyWJjh6+tr3
H777cahQ4dKpTbNsyMiIiIuTX12REREXKUP7IiIiIhLU9gRERERl6awIyIiIi5NYU

```

dERERcmsKOiIiIuDSFHREREXFPcjsiUuImTpxIs2bNAGjWrBkTJ060tB4RcS8KOyJ
yXR566KE8071/8sknlCtXjpdfhMA0aNH29fEiY+PZ/To0aVdpt0bb7xB06ZNCQgI
IDg4mJtvvlkLRYq4OK2NJSLF6s0332Tw4MHMnz+fHx9+GICAgAACAgIACA0Ntay2f
/3rX4wMYLZs2fToUMHMjMz2b9/P19//bVlNYlIydOTHREpNjNmzGDo0KF88MEH9q
CT35OfESNGcNttt9n3b7vtNoYOHcqIESOoWLEiYWFhvPHGG1y4cIGHH36YChUqULT
2bVavXu1wn6+//ppu3boREBBAWFgYgWYN4syZM9esb9myZfTv359HH32U2rVr06hR
I+677z5eeOEF+zU7d+7kjjvu4IYbbiAoKIgOHTqwZ88e+/lHHnmEnj170tw3KyuLy
pUr89Zbb9k/z4gRI+zn586dS506dShXrhxhYWHcfffDp/96mtFpPgp7IhIsXj22W
eZMmUKK1as4K677ir077/99tvccMMN7Nixg6FDh/LUU09xzz330KZNG/bs2UPnzp0
ZNGGQv/76KwCpqa106tSJm2++mV27drFmzRpSUlLo37//Nd8jPDycbdu2cfz48Wte
c/78eWJjY9m8eTPbtm2jTp06dO/enfPnzWPw2GOPsWbNGn7++Wf776xYsYJff/2Ve
++9N8/9du3axbBhw5g8eTKHDh1izZo13HrrrYX+5yMil6FULhsVEZcVGxtr+Pj4GI
ARHx+f7/k777zT4djw4cONDh062Pc7dOhgtGvXzr5/+fJlw9/f3xg0aJD92M8//2w
ARmJiomEYhjFlyhSjc+fODvdNSkoygGuupHzq1CmjdevWBmDUrVvXiI2NNT788EMj
Ozv7mp8vOzvbqFChgrF8+XL7sYYNGxrTp0+37/fq1ct46KGHHD5P7mr0//73v43Aw
EAjPT093/tr5XqRkqcnOyJy3W666SZqlqzJhAkTyMjIKPI9cnl6ehIaGkqTJk3sx8
LCwG4A4ffo0APv27WP9+vX2/kABAQHUr18fgB9//DHf96hSpQqJiYkcOHCA4cOHc/n
yZWJjY+natSs50TkApKSk8Pjjj1OnTh2CgoIIDAwkIyODEyd020/z2GOPsXDhQvv1
ql5v5pFHHsn3Pe+44w5q1KjBjTfeyKBBg3jvffsT6dEpHQo7IjI datatSobNmzg5
MmTd03a1d7kA+Dh4YFhGA7XZ2Vl5bmHt7e3w77NZnM4ZrPZA0yHJCMjg169erF371
6H7fDhw3/atNS4cWOefvppFi9ezNq1alm7di0JCQkAxMbGsnfvXmbNmsXWrvVvZu3c
voaGhXLp0yf77Dz74IEeOHCEXMHFfixcTFRVF+/bt832vChUqsGfPHt5//32qVKlC
XFwcTZs2JTU19Q9rFJHio7AjIsWiRo0aJCQkkJyc7BB4KlWq5NC/BWDv3r3X/X7Nm
zfnm2++oWbNmtSuXdth8/f3L/B9GjZsCMCFcXcA2LJlC8OGDaN79+40atQIX1/fPJ
2eQ0ND6dOnDwsXLmTRokX2ztjX4uXlRUxMDDNmzGD//v0c03aMdevWFFiTi0hRKey
ISLGJjIxkw4YNnD59mi5dupCenk6nTp3YtWsX77zzDocPH2bChAnFMTr78ODBnD17
lvvuu4+d03fy448/8vnnn/Pwww+TnZ2d7+889dRTTJkyhS1btnD8+HG2bdvGgw8+S
KVKlYiOjgagTp06vPvuuxw8eJDt27czcOBA/Pz88tzrscce4+233+bgwYPExsZes8
4VKlYwe/Zs9u7dy/Hjx3nnnXfIycmhXr161/3PQEQRmFHRiPvTWrV2LBhA2fOnKF
Lly5ER0czfvx4/va3v9GyZUvOnz/Pgw8+eN3vExERwZytW8jOzqZz5840adKEESNG
EBwcjIdH/v9qi4mJYdu2bdxzzz3UrVuXfv36Ua5cOeLj4+3z/7z11luc03e05s2bM
2jQIIYNG0blypXzvVeVKlXo0qULERER16wzODiY//znP3Tq1IkGDRowf/583n//fR
olanTd/wxEpGBsxu8b00VE5E9lZGRQtWpVFf5cSN++fa0uR0T+gGZQFhEphJycHM6
cOcPLL79MchAwvXv3trokEfkTCjsiIoVw4sQJoqKiqFatGosWLCllS/8aFXF2asYS
ERERl6YOyiIiIuLSFHZERETEpSnsiIiIiEtT2BERERGXprAjIiIiLklhR0RERFyaw
o6IiIi4NIUDERERcWkKOyIiIuLS/h0Fa/cf+1cZAAAAABJRU5ErkJggg==\n"

```

```

    },
    "metadata": {}
  }
],
},
{
  "cell_type": "code",
  "source": [
    "kume_sayisi = int(input(\"Lütfen küme sayısını girin:
\")(\"))"
  ],
  "metadata": {
    "colab": {
      "base_uri": "https://localhost:8080/"
    },
    "id": "_shZbZz6WsLF",
    "outputId": "0bb8f320-869f-4183-9635-a69dec3fbc"
  },
  "execution_count": 4,
  "outputs": [
    {
      "name": "stdout",

```

```

        "output_type": "stream",
        "text": [
            "Lütfen küme sayısını girin: 4\n"
        ]
    }
]
},
{
    "cell_type": "code",
    "source": [
        "# K-Means kümeleme\n",
        "def cluster_bayiler(data, n_clusters):\n",
        "    kmeans = KMeans(n_clusters=n_clusters,\n",
        "random_state=0, n_init=10)\n",
        "    labels = kmeans.fit_predict(data)\n",
        "    return labels\n",
        "\n",
        "labels = cluster_bayiler(mesafe_mat, kume_sayisi)\n",
        "kume_dict = {i: [] for i in range(kume_sayisi)}\n",
        "for i, label in enumerate(labels):\n",
        "    kume_dict[label].append(bayi_ids[i])\n",
        "\n",
        "# Mesafe hesaplama fonksiyonu\n",
        "def calculate_route_distance(route, distance_matrix,\n",
        "depo_mesafeleri, bayi_ids):\n",
        "    total_distance = 0\n",
        "    for i in range(len(route) - 1):\n",
        "        if route[i] == 1:\n",
        "            idx2 = np.where(bayi_ids ==\n",
        "route[i+1])[0][0]\n",
        "            distance = depo_mesafeleri[idx2]\n",
        "            elif route[i+1] == 1:\n",
        "                idx1 = np.where(bayi_ids ==\n",
        "route[i])[0][0]\n",
        "                distance = depo_mesafeleri[idx1]\n",
        "            else:\n",
        "                idx1 = np.where(bayi_ids ==\n",
        "route[i])[0][0]\n",
        "                idx2 = np.where(bayi_ids ==\n",
        "route[i+1])[0][0]\n",
        "                distance = distance_matrix[idx1, idx2]\n",
        "            total_distance += distance\n",
        "    return total_distance\n",
        "\n",
        "# Ateşböceği Algoritması ile En Kısa Rota\n",
        "class FireflyAlgorithm:\n",
        "    def __init__(self, kume, distance_matrix,\n",
        "depo_mesafeleri, bayi_ids, max_iter=100):\n",
        "        self.kume = kume\n",
        "        self.distance_matrix = distance_matrix\n",
        "        self.depo_mesafeleri = depo_mesafeleri\n",
        "        self.bayi_ids = bayi_ids\n",
        "        self.max_iter = max_iter\n",
        "\n",
        "    def optimize(self):\n",
        "        best_route = None\n",
        "        best_distance = float('inf')\n",
        "        num_bayiler = len(self.kume)

```

```

"\n",
"        for _ in range(self.max_iter):\n",
"            random.shuffle(self.kume)\n",
"            route = [1] + self.kume + [1]\n",
"            distance = calculate_route_distance(route,
self.distance_matrix, self.depo_mesafeleri, self.bayi_ids)\n",
"            if distance < best_distance:\n",
"                best_distance = distance\n",
"                best_route = route\n",
"        return best_route, best_distance\n",
"\n",
"rotalar = {}\n",
"mesafeler = {}\n",
"for kume_id, bayiler in kume_dict.items():\n",
"    fa = FireflyAlgorithm(bayiler, mesafe_mat,
depo_mesafeleri, bayi_ids)\n",
"    rota, mesafe = fa.optimize()\n",
"    rotalar[kume_id] = rota\n",
"    mesafeler[kume_id] = mesafe\n",
"\n",
"# Kapasite aşımı kontrolü ve ek araç yönetimi\n",
"MAX_KAPASITE = 5 * 3 # Araç kapasitesi\n",
"araclar = {}\n",
"\n",
"def arac_yonet(rota, siparis_df, kume_id):\n",
"    arac_sayaci = 1\n",
"    kapasite = 0\n",
"    araclar[kume_id] = {}\n",
"    araclar[kume_id][arac_sayaci] = []\n",
"\n",
"    for bayi in rota:\n",
"        if bayi == 1:\n",
"            continue\n",
"        siparis = siparis_df[siparis_df['Lokasyonlar']
== bayi][['A', 'B']].values.flatten()\n",
"        siparis_miktari = siparis[0] + siparis[1]\n",
"\n",
"        if kapasite + siparis_miktari >
MAX_KAPASITE:\n",
"            arac_sayaci += 1\n",
"            kapasite = 0\n",
"            araclar[kume_id][arac_sayaci] = []\n",
"\n",
"            araclar[kume_id][arac_sayaci].append((bayi,
siparis[0], siparis[1]))\n",
"            kapasite += siparis_miktari\n",
"\n",
"for kume_id, rota in rotalar.items():\n",
"    arac_yonet(rota, siparis_df, kume_id)\n",
"\n",
"# Çıktılar\n",
"print("\n🚚 1. KÜMELEME SONUÇLARI")\n",
"for kume_id, bayiler in kume_dict.items():\n",
"    print(f"Küme {kume_id+1}: {bayiler}")\n",
"\n",
"print("\n🚚 2. ROTASYON SONUÇLARI")\n",
"for kume_id, rota in rotalar.items():\n",
"    print(f"Küme {kume_id+1} - Rota: {rota}")\n",

```

```

"\n",
"print(\"\\n🚗 3. HER ARAÇ İÇİN ROTASYON VE
MESAFELELER\\n\")\n",
"for kume_id, arac_list in aracler.items():\n",
"    for arac_id, rota in arac_list.items():\n",
"        arac_mesafe = calculate_route_distance([1] +
[x[0] for x in rota] + [1], mesafe_mat, depo_mesafeleri,
bayi_ids)\n",
"        print(f\"Küme {kume_id+1} - Araç {arac_id} -
Rota: {[1] + [x[0] for x in rota] + [1]} - Mesafe: {arac_mesafe}
km\\n\")\n",
"        for bayi, a_miktar, b_miktar in rota:\n",
"            print(f\"        Araç {arac_id} - Bayi {bayi}:
{a_miktar} A, {b_miktar} B\\n\")\n"
],
"metadata": {
"colab": {
"base_uri": "https://localhost:8080/"
},
"id": "Wedzrs6cWpPI",
"outputId": "8b7c0c94-f60e-4e85-e3ae-8244792bd2ff"
},
"execution_count": 6,
"outputs": [
{
"output_type": "stream",
"name": "stdout",
"text": [
"\n",
"📌 1. KÜMELEME SONUÇLARI\n",
"Küme 1: [34, 40, 60]\n",
"Küme 2: [27, 47, 16, 58, 31, 24, 10, 57, 43, 38, 51,
8, 37, 52, 55, 48, 7, 2, 23, 26, 54, 15, 46, 33, 30, 22, 49, 36,
14, 11, 29, 13, 35, 39, 5, 32, 42, 41, 50, 21, 20, 45, 28, 25, 6,
59]\n",
"Küme 3: [56, 19, 9]\n",
"Küme 4: [18, 3, 4, 44, 53, 12, 17]\n",
"\n",
"🚗 2. ROTASYON SONUÇLARI\n",
"Küme 1 - Rota: [1, 34, 40, 60, 1]\n",
"Küme 2 - Rota: [1, 8, 23, 27, 58, 47, 39, 14, 13,
26, 31, 11, 54, 2, 45, 7, 16, 59, 55, 5, 35, 30, 21, 24, 49, 37,
46, 52, 10, 38, 22, 28, 51, 6, 36, 15, 20, 48, 25, 32, 50, 29,
41, 57, 43, 33, 42, 1]\n",
"Küme 3 - Rota: [1, 9, 19, 56, 1]\n",
"Küme 4 - Rota: [1, 3, 18, 17, 12, 44, 53, 4, 1]\n",
"\n",
"🚗 3. HER ARAÇ İÇİN ROTASYON VE MESAFELELER\n",
"Küme 1 - Araç 1 - Rota: [1, 34, 40, 60, 1] - Mesafe:
358.3 km\n",
"    Araç 1 - Bayi 34: 1 A, 1 B\n",
"    Araç 1 - Bayi 40: 1 A, 0 B\n",
"    Araç 1 - Bayi 60: 1 A, 0 B\n",
"Küme 2 - Araç 1 - Rota: [1, 8, 23, 27, 58, 47, 39,
14, 13, 26, 31, 1] - Mesafe: 135.9 km\n",
"    Araç 1 - Bayi 8: 1 A, 0 B\n",
"    Araç 1 - Bayi 23: 1 A, 0 B\n",

```

" Araç 1 - Bayi 27: 0 A, 3 B\n",
 " Araç 1 - Bayi 58: 0 A, 1 B\n",
 " Araç 1 - Bayi 47: 0 A, 1 B\n",
 " Araç 1 - Bayi 39: 0 A, 2 B\n",
 " Araç 1 - Bayi 14: 0 A, 3 B\n",
 " Araç 1 - Bayi 13: 0 A, 1 B\n",
 " Araç 1 - Bayi 26: 1 A, 0 B\n",
 " Araç 1 - Bayi 31: 0 A, 1 B\n",
 "Küme 2 - Araç 2 - Rota: [1, 11, 54, 2, 45, 7, 16, 59, 55, 5, 35, 1] - Mesafe: 203.29999999999998 km\n",
 " Araç 2 - Bayi 11: 0 A, 2 B\n",
 " Araç 2 - Bayi 54: 0 A, 2 B\n",
 " Araç 2 - Bayi 2: 1 A, 1 B\n",
 " Araç 2 - Bayi 45: 0 A, 1 B\n",
 " Araç 2 - Bayi 7: 0 A, 1 B\n",
 " Araç 2 - Bayi 16: 1 A, 1 B\n",
 " Araç 2 - Bayi 59: 0 A, 1 B\n",
 " Araç 2 - Bayi 55: 0 A, 1 B\n",
 " Araç 2 - Bayi 5: 1 A, 0 B\n",
 " Araç 2 - Bayi 35: 0 A, 2 B\n",
 "Küme 2 - Araç 3 - Rota: [1, 30, 21, 24, 49, 37, 46, 52, 10, 38, 22, 28, 51, 1] - Mesafe: 205.2 km\n",
 " Araç 3 - Bayi 30: 0 A, 1 B\n",
 " Araç 3 - Bayi 21: 1 A, 0 B\n",
 " Araç 3 - Bayi 24: 0 A, 1 B\n",
 " Araç 3 - Bayi 49: 1 A, 0 B\n",
 " Araç 3 - Bayi 37: 0 A, 1 B\n",
 " Araç 3 - Bayi 46: 0 A, 1 B\n",
 " Araç 3 - Bayi 52: 0 A, 1 B\n",
 " Araç 3 - Bayi 10: 0 A, 1 B\n",
 " Araç 3 - Bayi 38: 0 A, 1 B\n",
 " Araç 3 - Bayi 22: 0 A, 2 B\n",
 " Araç 3 - Bayi 28: 1 A, 1 B\n",
 " Araç 3 - Bayi 51: 0 A, 2 B\n",
 "Küme 2 - Araç 4 - Rota: [1, 6, 36, 15, 20, 48, 25, 32, 50, 29, 41, 57, 43, 33, 1] - Mesafe: 244.89999999999998 km\n",
 " Araç 4 - Bayi 6: 1 A, 0 B\n",
 " Araç 4 - Bayi 36: 0 A, 1 B\n",
 " Araç 4 - Bayi 15: 1 A, 0 B\n",
 " Araç 4 - Bayi 20: 0 A, 1 B\n",
 " Araç 4 - Bayi 48: 1 A, 1 B\n",
 " Araç 4 - Bayi 25: 1 A, 0 B\n",
 " Araç 4 - Bayi 32: 1 A, 0 B\n",
 " Araç 4 - Bayi 50: 1 A, 0 B\n",
 " Araç 4 - Bayi 29: 1 A, 0 B\n",
 " Araç 4 - Bayi 41: 0 A, 1 B\n",
 " Araç 4 - Bayi 57: 0 A, 1 B\n",
 " Araç 4 - Bayi 43: 0 A, 2 B\n",
 " Araç 4 - Bayi 33: 1 A, 0 B\n",
 "Küme 2 - Araç 5 - Rota: [1, 42, 1] - Mesafe: 6.6 km\n",
 " Araç 5 - Bayi 42: 1 A, 1 B\n",
 "Küme 3 - Araç 1 - Rota: [1, 9, 19, 56, 1] - Mesafe: 349.1 km\n",
 " Araç 1 - Bayi 9: 1 A, 0 B\n",
 " Araç 1 - Bayi 19: 1 A, 0 B\n",
 " Araç 1 - Bayi 56: 1 A, 0 B\n",

```
"Küme 4 - Araç 1 - Rota: [1, 3, 18, 17, 12, 44, 53,
4, 1] - Mesafe: 383.70000000000005 km\n",
"    Araç 1 - Bayi 3: 1 A, 0 B\n",
"    Araç 1 - Bayi 18: 0 A, 1 B\n",
"    Araç 1 - Bayi 17: 0 A, 1 B\n",
"    Araç 1 - Bayi 12: 0 A, 1 B\n",
"    Araç 1 - Bayi 44: 1 A, 0 B\n",
"    Araç 1 - Bayi 53: 1 A, 0 B\n",
"    Araç 1 - Bayi 4: 0 A, 2 B\n"
]
}
]
}
]
}
```



EK-3

```
{
  "nbformat": 4,
  "nbformat_minor": 0,
  "metadata": {
    "colab": {
      "provenance": []
    },
    "kernelspec": {
      "name": "python3",
      "display_name": "Python 3"
    },
    "language_info": {
      "name": "python"
    }
  },
  "cells": [
    {
      "cell_type": "code",
      "execution_count": 1,
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "OD-vVYgfZ1tv",
        "outputId": "9f5052cf-e9c5-4efb-ef8c-5d2af96cfe1d"
      },
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "Mounted at /content/drive\n"
          ]
        }
      ],
      "source": [
        "# Google Drive Bağlantısı\n",
        "from google.colab import drive\n",
        "drive.mount('/content/drive')\n"
      ]
    },
    {
      "cell_type": "code",
      "source": [
        "import pandas as pd\n",
        "import numpy as np\n",
        "import matplotlib.pyplot as plt\n",
        "import subprocess\n",
        "import sys\n",
        "\n",
        "# skfuzzy modülü eksikse yükleme\n",
        "try:\n",
        "    import skfuzzy as fuzz\n",
        "except ImportError:\n",
        "    subprocess.check_call([sys.executable, "-m",\n",
        "\"pip\", \"install\", \"scikit-fuzzy\"])\n",

```

```

        "    import skfuzzy as fuzz\n",
        "\n",
        "from scipy.spatial.distance import cdist\n",
        "import random\n",
        "\n",
        "# Google Drive'dan dosyaları yükleme\n",
        "mesafe_path =
\"/content/drive/MyDrive/end_/Mesafe.xlsx\"\n",
        "siparis_path =
\"/content/drive/MyDrive/end_/siparis.xlsx\"\n",
        "depo_path =
\"/content/drive/MyDrive/end_/Depo.xlsx\"\n",
        "\n",
        "mesafe_df = pd.read_excel(mesafe_path)\n",
        "siparis_df = pd.read_excel(siparis_path)\n",
        "depo_df = pd.read_excel(depo_path)\n",
        "\n",
        "# Bayi ID'lerini al\n",
        "bayi_ids = mesafe_df.iloc[:, 0].values # 2'den 60'a
kadar bayi ID'leri\n",
        "mesafe_mat = mesafe_df.iloc[:, 1:].values # 59x59
mesafe matrisi\n",
        "\n",
        "depo_mesafeleri = depo_df.iloc[0, 2:].values # Depodan
bayilere mesafeler\n",
        "\n",
        "# Optimum küme sayısını belirlemek için Fuzzy C-Means
yöntemi\n",
        "def fuzzy_c_means_optimal_clusters(data):\n",
        "    distortions = []\n",
        "    K = range(2, 11)\n",
        "    for k in K:\n",
        "        cntr, u, _ = fuzz.cluster.cmeans(data.T, k, 2, error=0.005, maxiter=1000)\n",
        "        compactness = np.sum(np.min(cdist(data, cntr,
'euclidean'), axis=1)) / data.shape[0]\n",
        "        distortions.append(compactness)\n",
        "        optimal_k = K[np.argmin(distortions)]\n",
        "        return optimal_k\n",
        "\n",
        "# En uygun küme sayısını belirle\n",
        "kume_sayisi =
fuzzy_c_means_optimal_clusters(mesafe_mat)
    ],
    "metadata": {
        "id": "r0sGR0f8bXXe"
    },
    "execution_count": 12,
    "outputs": []
},
{
    "cell_type": "code",
    "source": [
        "# Fuzzy C-Means kümeleme\n",
        "def fuzzy_c_means_clustering(data, n_clusters):\n",
        "    cntr, u, _ = fuzz.cluster.cmeans(data.T,
n_clusters, 2, error=0.005, maxiter=1000)\n",
        "    labels = np.argmax(u, axis=0)\n",

```

```

        "    return labels\n",
        "\n",
        "labels = fuzzy_c_means_clustering(mesafe_mat,
kume_sayisi)\n",
        "kume_dict = {i: [] for i in range(kume_sayisi)}\n",
        "for i, label in enumerate(labels):\n",
        "    kume_dict[label].append(bayi_ids[i])\n",
        "\n",
        "# Mesafe hesaplama fonksiyonu\n",
        "def calculate_route_distance(route, distance_matrix,
depo_mesafeleri, bayi_ids):\n",
        "    total_distance = 0\n",
        "    for i in range(len(route) - 1):\n",
        "        if route[i] == 1:\n",
        "            idx2 = np.where(bayi_ids ==
route[i+1])[0][0]\n",
        "            distance = depo_mesafeleri[idx2]\n",
        "        elif route[i+1] == 1:\n",
        "            idx1 = np.where(bayi_ids ==
route[i])[0][0]\n",
        "            distance = depo_mesafeleri[idx1]\n",
        "        else:\n",
        "            idx1 = np.where(bayi_ids ==
route[i])[0][0]\n",
        "            idx2 = np.where(bayi_ids ==
route[i+1])[0][0]\n",
        "            distance = distance_matrix[idx1, idx2]\n",
        "        total_distance += distance\n",
        "    return total_distance\n",
        "\n",
        "# Yapay Arı Kolonisi Algoritması (ABC) ile En Kısa
Rota\n",
        "class ArtificialBeeColony:\n",
        "    def __init__(self, kume, distance_matrix,
depo_mesafeleri, bayi_ids, max_iter=100):\n",
        "        self.kume = kume\n",
        "        self.distance_matrix = distance_matrix\n",
        "        self.depo_mesafeleri = depo_mesafeleri\n",
        "        self.bayi_ids = bayi_ids\n",
        "        self.max_iter = max_iter\n",
        "\n",
        "    def optimize(self):\n",
        "        best_route = None\n",
        "        best_distance = float('inf')\n",
        "        num_bayiler = len(self.kume)\n",
        "\n",
        "        for _ in range(self.max_iter):\n",
        "            random.shuffle(self.kume)\n",
        "            route = [1] + self.kume + [1]\n",
        "            distance = calculate_route_distance(route,
self.distance_matrix, self.depo_mesafeleri, self.bayi_ids)\n",
        "            if distance < best_distance:\n",
        "                best_distance = distance\n",
        "                best_route = route\n",
        "        return best_route, best_distance\n",
        "\n",
        "rotalar = {}\n",
        "mesafeler = {}

```

```

        "for kume_id, bayiler in kume_dict.items():\n",
        "    abc = ArtificialBeeColony(bayiler, mesafe_mat,\n",
depo_mesafeleri, bayi_ids)\n",
        "    rota, mesafe = abc.optimize()\n",
        "    rotalar[kume_id] = rota\n",
        "    mesafeler[kume_id] = mesafe\n",
        "\n",
        "# Kapasite aşımı kontrolü ve ek araç yönetimi\n",
        "MAX_KAPASITE = 5 * 3 # Araç kapasitesi\n",
        "araclar = {}\n",
        "\n",
        "def arac_yonet(rota, siparis_df, kume_id):\n",
        "    arac_sayaci = 1\n",
        "    kapasite = 0\n",
        "    araclar[kume_id] = {}\n",
        "    araclar[kume_id][arac_sayaci] = []\n",
        "\n",
        "    for bayi in rota:\n",
        "        if bayi == 1:\n",
        "            continue\n",
        "        siparis = siparis_df[siparis_df['Lokasyonlar']\n",
== bayi][['A', 'B']].values.flatten()\n",
        "        siparis_miktari = siparis[0] + siparis[1]\n",
        "\n",
        "        if kapasite + siparis_miktari >\n",
MAX_KAPASITE:\n",
        "            arac_sayaci += 1\n",
        "            kapasite = 0\n",
        "            araclar[kume_id][arac_sayaci] = []\n",
        "\n",
        "            araclar[kume_id][arac_sayaci].append((bayi,\n",
siparis[0], siparis[1]))\n",
        "            kapasite += siparis_miktari\n",
        "\n",
        "for kume_id, rota in rotalar.items():\n",
        "    arac_yonet(rota, siparis_df, kume_id)\n",
        "\n",
        "# Çıktılar\n",
        "print("\n\n📌 1. KÜMELEME SONUÇLARI")\n",
        "for kume_id, bayiler in kume_dict.items():\n",
        "    print(f"Küme {kume_id+1}: {bayiler}")\n",
        "\n",
        "print("\n\n🚗 2. ROTASYON SONUÇLARI")\n",
        "for kume_id, rota in rotalar.items():\n",
        "    print(f"Küme {kume_id+1} - Rota: {rota}")\n",
        "\n",
        "print("\n\n🚚 3. HER ARAÇ İÇİN ROTASYON VE\n",
MESAFELELER)\n",
        "for kume_id, arac_list in araclar.items():\n",
        "    for arac_id, rota in arac_list.items():\n",
        "        arac_mesafe = calculate_route_distance([1] +\n",
[x[0] for x in rota] + [1], mesafe_mat, depo_mesafeleri,\n",
bayi_ids)\n",
        "        print(f"Küme {kume_id+1} - Araç {arac_id} -\n",
Rota: {[1] + [x[0] for x in rota] + [1]} - Mesafe: {arac_mesafe}\n",
km)\n",
        "        for bayi, a_miktar, b_miktar in rota:\n",

```

```

"          print(f"      Araç {arac_id} - Bayi {bayi}:
{a_miktar} A, {b_miktar} B")"
],
"metadata": {
  "colab": {
    "base_uri": "https://localhost:8080/"
  },
  "id": "ga82IdmLbw1B",
  "outputId": "61e09392-4ca9-48f3-86a9-5507603a9ec9"
},
"execution_count": 13,
"outputs": [
  {
    "output_type": "stream",
    "name": "stdout",
    "text": [
      "\n",
      "📌 1. KÜMELEME SONUÇLARI\n",
      "Küme 1: [9]\n",
      "Küme 2: [54, 45, 37, 35, 22, 2, 29, 41, 38, 12]\n",
      "Küme 3: [21, 6, 36, 8, 52, 39, 23, 42, 50, 58]\n",
      "Küme 4: [4, 34]\n",
      "Küme 5: [13, 59, 47, 48, 20, 30, 57, 55, 15, 14, 27,
28, 10, 43, 5, 51, 26]\n",
      "Küme 6: [18, 3, 17, 44]\n",
      "Küme 7: [7, 33, 53, 31]\n",
      "Küme 8: [25, 46, 32, 24, 49, 16, 11]\n",
      "Küme 9: [19, 56]\n",
      "Küme 10: [40, 60]\n",
      "\n",
      "🚗 2. ROTASYON SONUÇLARI\n",
      "Küme 1 - Rota: [1, 9, 1]\n",
      "Küme 2 - Rota: [1, 37, 29, 2, 38, 41, 45, 22, 54,
35, 12, 1]\n",
      "Küme 3 - Rota: [1, 42, 58, 52, 36, 8, 6, 39, 50, 21,
23, 1]\n",
      "Küme 4 - Rota: [1, 34, 4, 1]\n",
      "Küme 5 - Rota: [1, 57, 43, 28, 30, 51, 47, 27, 10,
5, 55, 26, 20, 15, 48, 14, 59, 13, 1]\n",
      "Küme 6 - Rota: [1, 44, 3, 18, 17, 1]\n",
      "Küme 7 - Rota: [1, 53, 31, 7, 33, 1]\n",
      "Küme 8 - Rota: [1, 24, 46, 49, 11, 16, 32, 25,
1]\n",
      "Küme 9 - Rota: [1, 19, 56, 1]\n",
      "Küme 10 - Rota: [1, 40, 60, 1]\n",
      "\n",
      "🚚 3. HER ARAÇ İÇİN ROTASYON VE MESAFELELER\n",
      "Küme 1 - Araç 1 - Rota: [1, 9, 1] - Mesafe: 262.0
km\n",
      "Araç 1 - Bayi 9: 1 A, 0 B\n",
      "Küme 2 - Araç 1 - Rota: [1, 37, 29, 2, 38, 41, 45,
22, 54, 35, 12, 1] - Mesafe: 159.8 km\n",
      "Araç 1 - Bayi 37: 0 A, 1 B\n",
      "Araç 1 - Bayi 29: 1 A, 0 B\n",
      "Araç 1 - Bayi 2: 1 A, 1 B\n",
      "Araç 1 - Bayi 38: 0 A, 1 B\n",
      "Araç 1 - Bayi 41: 0 A, 1 B\n",
      "Araç 1 - Bayi 45: 0 A, 1 B

```

" Araç 1 - Bayi 22: 0 A, 2 B\n",
 " Araç 1 - Bayi 54: 0 A, 2 B\n",
 " Araç 1 - Bayi 35: 0 A, 2 B\n",
 " Araç 1 - Bayi 12: 0 A, 1 B\n",
 "Küme 3 - Araç 1 - Rota: [1, 42, 58, 52, 36, 8, 6,
 39, 50, 21, 23, 1] - Mesafe: 46.89999999999999 km\n",
 " Araç 1 - Bayi 42: 1 A, 1 B\n",
 " Araç 1 - Bayi 58: 0 A, 1 B\n",
 " Araç 1 - Bayi 52: 0 A, 1 B\n",
 " Araç 1 - Bayi 36: 0 A, 1 B\n",
 " Araç 1 - Bayi 8: 1 A, 0 B\n",
 " Araç 1 - Bayi 6: 1 A, 0 B\n",
 " Araç 1 - Bayi 39: 0 A, 2 B\n",
 " Araç 1 - Bayi 50: 1 A, 0 B\n",
 " Araç 1 - Bayi 21: 1 A, 0 B\n",
 " Araç 1 - Bayi 23: 1 A, 0 B\n",
 "Küme 4 - Araç 1 - Rota: [1, 34, 4, 1] - Mesafe:
 214.79999999999998 km\n",
 " Araç 1 - Bayi 34: 1 A, 1 B\n",
 " Araç 1 - Bayi 4: 0 A, 2 B\n",
 "Küme 5 - Araç 1 - Rota: [1, 57, 43, 28, 30, 51, 47,
 27, 10, 5, 55, 1] - Mesafe: 93.5 km\n",
 " Araç 1 - Bayi 57: 0 A, 1 B\n",
 " Araç 1 - Bayi 43: 0 A, 2 B\n",
 " Araç 1 - Bayi 28: 1 A, 1 B\n",
 " Araç 1 - Bayi 30: 0 A, 1 B\n",
 " Araç 1 - Bayi 51: 0 A, 2 B\n",
 " Araç 1 - Bayi 47: 0 A, 1 B\n",
 " Araç 1 - Bayi 27: 0 A, 3 B\n",
 " Araç 1 - Bayi 10: 0 A, 1 B\n",
 " Araç 1 - Bayi 5: 1 A, 0 B\n",
 " Araç 1 - Bayi 55: 0 A, 1 B\n",
 "Küme 5 - Araç 2 - Rota: [1, 26, 20, 15, 48, 14, 59,
 13, 1] - Mesafe: 32.7 km\n",
 " Araç 2 - Bayi 26: 1 A, 0 B\n",
 " Araç 2 - Bayi 20: 0 A, 1 B\n",
 " Araç 2 - Bayi 15: 1 A, 0 B\n",
 " Araç 2 - Bayi 48: 1 A, 1 B\n",
 " Araç 2 - Bayi 14: 0 A, 3 B\n",
 " Araç 2 - Bayi 59: 0 A, 1 B\n",
 " Araç 2 - Bayi 13: 0 A, 1 B\n",
 "Küme 6 - Araç 1 - Rota: [1, 44, 3, 18, 17, 1] -
 Mesafe: 211.7 km\n",
 " Araç 1 - Bayi 44: 1 A, 0 B\n",
 " Araç 1 - Bayi 3: 1 A, 0 B\n",
 " Araç 1 - Bayi 18: 0 A, 1 B\n",
 " Araç 1 - Bayi 17: 0 A, 1 B\n",
 "Küme 7 - Araç 1 - Rota: [1, 53, 31, 7, 33, 1] -
 Mesafe: 106.19999999999999 km\n",
 " Araç 1 - Bayi 53: 1 A, 0 B\n",
 " Araç 1 - Bayi 31: 0 A, 1 B\n",
 " Araç 1 - Bayi 7: 0 A, 1 B\n",
 " Araç 1 - Bayi 33: 1 A, 0 B\n",
 "Küme 8 - Araç 1 - Rota: [1, 24, 46, 49, 11, 16, 32,
 25, 1] - Mesafe: 87.79999999999998 km\n",
 " Araç 1 - Bayi 24: 0 A, 1 B\n",
 " Araç 1 - Bayi 46: 0 A, 1 B\n",
 " Araç 1 - Bayi 49: 1 A, 0 B\n",

```

"    Araç 1 - Bayi 11: 0 A, 2 B\n",
"    Araç 1 - Bayi 16: 1 A, 1 B\n",
"    Araç 1 - Bayi 32: 1 A, 0 B\n",
"    Araç 1 - Bayi 25: 1 A, 0 B\n",
"Küme 9 - Araç 1 - Rota: [1, 19, 56, 1] - Mesafe:
263.0 km\n",
"    Araç 1 - Bayi 19: 1 A, 0 B\n",
"    Araç 1 - Bayi 56: 1 A, 0 B\n",
"Küme 10 - Araç 1 - Rota: [1, 40, 60, 1] - Mesafe:
351.3 km\n",
"    Araç 1 - Bayi 40: 1 A, 0 B\n",
"    Araç 1 - Bayi 60: 1 A, 0 B\n"
]
}
]
}
]
}

```



```

{
  "nbformat": 4,
  "nbformat_minor": 0,
  "metadata": {
    "colab": {
      "provenance": []
    },
    "kernelspec": {
      "name": "python3",
      "display_name": "Python 3"
    },
    "language_info": {
      "name": "python"
    }
  },
  "cells": [
    {
      "cell_type": "code",
      "execution_count": 1,
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "XNZdOyj1YJ7C",
        "outputId": "303e9948-c103-48fc-fb3a-31e789837fec"
      },
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "Mounted at /content/drive\n"
          ]
        }
      ],
      "source": [
        "# Google Drive Bağlantısı\n",
        "from google.colab import drive\n",
        "drive.mount('/content/drive')\n"
      ]
    },
    {
      "cell_type": "code",
      "source": [
        "import pandas as pd\n",
        "import numpy as np\n",
        "import matplotlib.pyplot as plt\n",
        "import subprocess\n",
        "import sys\n",
        "\n",
        "# skfuzzy modülü eksikse yükleme\n",
        "try:\n",
        "    import skfuzzy as fuzz\n",
        "except ImportError:\n",
        "    subprocess.check_call([sys.executable, "-m",\n",
        "\npip", "install", "scikit-fuzzy"])\n",

```

```

"    import skfuzzy as fuzz\n",
"\n",
"from scipy.spatial.distance import cdist\n",
"import random\n",
"\n",
"# Google Drive'dan dosyaları yükleme\n",
"mesafe_path =
\"/content/drive/MyDrive/end_/Mesafe.xlsx\"\n",
"siparis_path =
\"/content/drive/MyDrive/end_/siparis.xlsx\"\n",
"depo_path =
\"/content/drive/MyDrive/end_/Depo.xlsx\"\n",
"\n",
"mesafe_df = pd.read_excel(mesafe_path)\n",
"siparis_df = pd.read_excel(siparis_path)\n",
"depo_df = pd.read_excel(depo_path)\n",
"\n",
"# Bayi ID'lerini al\n",
"bayi_ids = mesafe_df.iloc[:, 0].values # 2'den 60'a
kadar bayi ID'leri\n",
"mesafe_mat = mesafe_df.iloc[:, 1:].values # 59x59
mesafe matrisi\n",
"\n",
"depo_mesafeleri = depo_df.iloc[0, 2:].values # Depodan
bayilere mesafeler\n",
"\n",
"# Optimum küme sayısını belirlemek için Fuzzy C-Means
yöntemi\n",
"def fuzzy_c_means_optimal_clusters(data):\n",
"    distortions = []\n",
"    K = range(2, 11)\n",
"    for k in K:\n",
"        cntr, u, _ = fuzz.cluster.cmeans(data.T, k, 2, error=0.005, maxiter=1000)\n",
"        compactness = np.sum(np.min(cdist(data, cntr,
'euclidean'), axis=1)) / data.shape[0]\n",
"        distortions.append(compactness)\n",
"    optimal_k = K[np.argmin(distortions)]\n",
"    return optimal_k\n",
"\n",
"# En uygun küme sayısını belirle\n",
"kume_sayisi =
fuzzy_c_means_optimal_clusters(mesafe_mat)
],
"metadata": {
"    id": "jehhij8JYz-N"
},
"execution_count": 3,
"outputs": []
},
{
"cell_type": "code",
"source": [
"# Fuzzy C-Means kümeleme\n",
"def fuzzy_c_means_clustering(data, n_clusters):\n",
"    cntr, u, _ = fuzz.cluster.cmeans(data.T,
n_clusters, 2, error=0.005, maxiter=1000)\n",
"    labels = np.argmax(u, axis=0)\n",

```

```

        return labels\n",
    "\n",
    "labels = fuzzy_c_means_clustering(mesafe_mat,
kume_sayisi)\n",
    "kume_dict = {i: [] for i in range(kume_sayisi)}\n",
    "for i, label in enumerate(labels):\n",
    "    kume_dict[label].append(bayi_ids[i])\n",
    "\n",
    "# Mesafe hesaplama fonksiyonu\n",
    "def calculate_route_distance(route, distance_matrix,
depo_mesafeleri, bayi_ids):\n",
    "    total_distance = 0\n",
    "    for i in range(len(route) - 1):\n",
    "        if route[i] == 1:\n",
    "            idx2 = np.where(bayi_ids ==
route[i+1])[0][0]\n",
    "            distance = depo_mesafeleri[idx2]\n",
    "            elif route[i+1] == 1:\n",
    "                idx1 = np.where(bayi_ids ==
route[i])[0][0]\n",
    "                distance = depo_mesafeleri[idx1]\n",
    "            else:\n",
    "                idx1 = np.where(bayi_ids ==
route[i])[0][0]\n",
    "                idx2 = np.where(bayi_ids ==
route[i+1])[0][0]\n",
    "                distance = distance_matrix[idx1, idx2]\n",
    "            total_distance += distance\n",
    "    return total_distance\n",
    "\n",
    "# Ateşböceği Algoritması ile En Kısa Rota\n",
    "class FireflyAlgorithm:\n",
    "    def __init__(self, kume, distance_matrix,
depo_mesafeleri, bayi_ids, max_iter=100):\n",
    "        self.kume = kume\n",
    "        self.distance_matrix = distance_matrix\n",
    "        self.depo_mesafeleri = depo_mesafeleri\n",
    "        self.bayi_ids = bayi_ids\n",
    "        self.max_iter = max_iter\n",
    "\n",
    "    def optimize(self):\n",
    "        best_route = None\n",
    "        best_distance = float('inf')\n",
    "        num_bayiler = len(self.kume)\n",
    "\n",
    "        for _ in range(self.max_iter):\n",
    "            random.shuffle(self.kume)\n",
    "            route = [1] + self.kume + [1]\n",
    "            distance = calculate_route_distance(route,
self.distance_matrix, self.depo_mesafeleri, self.bayi_ids)\n",
    "            if distance < best_distance:\n",
    "                best_distance = distance\n",
    "                best_route = route\n",
    "        return best_route, best_distance\n",
    "\n",
    "rotalar = {}\n",
    "mesafeler = {}\n",
    "for kume_id, bayiler in kume_dict.items():\n",

```

```

        "    fa = FireflyAlgorithm(bayiler, mesafe_mat,
depo_mesafeleri, bayi_ids)\n",
        "    rota, mesafe = fa.optimize()\n",
        "    rotalar[kume_id] = rota\n",
        "    mesafeler[kume_id] = mesafe\n",
        "\n",
        "# Kapasite aşımı kontrolü ve ek araç yönetimi\n",
        "MAX_KAPASITE = 5 * 3 # Araç kapasitesi\n",
        "araclar = {}\n",
        "\n",
        "def arac_yonet(rota, siparis_df, kume_id):\n",
        "    arac_sayaci = 1\n",
        "    kapasite = 0\n",
        "    araclar[kume_id] = {}\n",
        "    araclar[kume_id][arac_sayaci] = []\n",
        "\n",
        "    for bayi in rota:\n",
        "        if bayi == 1:\n",
        "            continue\n",
        "        siparis = siparis_df[siparis_df['Lokasyonlar']
== bayi][['A', 'B']].values.flatten()\n",
        "        siparis_miktari = siparis[0] + siparis[1]\n",
        "\n",
        "        if kapasite + siparis_miktari >
MAX_KAPASITE:\n",
        "            arac_sayaci += 1\n",
        "            kapasite = 0\n",
        "            araclar[kume_id][arac_sayaci] = []\n",
        "\n",
        "            araclar[kume_id][arac_sayaci].append((bayi,
siparis[0], siparis[1]))\n",
        "            kapasite += siparis_miktari\n",
        "\n",
        "for kume_id, rota in rotalar.items():\n",
        "    arac_yonet(rota, siparis_df, kume_id)\n",
        "\n",
        "# Çıktılar\n",
        "print(\"\\n📌 1. KÜMELEME SONUÇLARI\\n\")\n",
        "for kume_id, bayiler in kume_dict.items():\n",
        "    print(f\"Küme {kume_id+1}: {bayiler}\\n\")\n",
        "\n",
        "print(\"\\n🚗 2. ROTASYON SONUÇLARI\\n\")\n",
        "for kume_id, rota in rotalar.items():\n",
        "    print(f\"Küme {kume_id+1} - Rota: {rota}\\n\")\n",
        "\n",
        "print(\"\\n📊 3. HER ARAÇ İÇİN ROTASYON VE
MESAFELER\\n\")\n",
        "for kume_id, arac_list in araclar.items():\n",
        "    for arac_id, rota in arac_list.items():\n",
        "        arac_mesafe = calculate_route_distance([1] +
[x[0] for x in rota] + [1], mesafe_mat, depo_mesafeleri,
bayi_ids)\n",
        "        print(f\"Küme {kume_id+1} - Araç {arac_id} -
Rota: {[1] + [x[0] for x in rota] + [1]} - Mesafe: {arac_mesafe}
km\\n\")\n",
        "        for bayi, a_miktar, b_miktar in rota:\n",
        "            print(f\"    Araç {arac_id} - Bayi {bayi}:
{a_miktar} A, {b_miktar} B\\n\")

```

```

],
"metadata": {
  "colab": {
    "base_uri": "https://localhost:8080/"
  },
  "id": "_uafI1kzZWKq",
  "outputId": "137358fa-81e6-452f-c11d-d2a3b6211992"
},
"execution_count": 4,
"outputs": [
  {
    "output_type": "stream",
    "name": "stdout",
    "text": [
      "\n",
      "📌 1. KÜMELEME SONUÇLARI\n",
      "Küme 1: [44, 34, 53, 4, 3]\n",
      "Küme 2: [20, 14, 43, 55, 48, 57, 5, 26, 13, 10, 51,
59, 30]\n",
      "Küme 3: [38, 54, 46, 41, 2, 11, 35, 32, 37, 45, 16,
29, 22]\n",
      "Küme 4: [60, 40]\n",
      "Küme 5: [19, 56]\n",
      "Küme 6: [9]\n",
      "Küme 7: [39, 6, 52, 31, 21, 36, 58, 50, 8, 7, 23,
42]\n",
      "Küme 8: [25, 28, 15, 49, 47, 24, 33, 27]\n",
      "Küme 9: [12, 18]\n",
      "Küme 10: [17]\n",
      "\n",
      "🚗 2. ROTASYON SONUÇLARI\n",
      "Küme 1 - Rota: [1, 4, 34, 53, 44, 3, 1]\n",
      "Küme 2 - Rota: [1, 20, 48, 14, 59, 5, 13, 10, 30,
57, 43, 51, 26, 55, 1]\n",
      "Küme 3 - Rota: [1, 37, 29, 2, 41, 22, 54, 16, 45,
32, 46, 11, 35, 38, 1]\n",
      "Küme 4 - Rota: [1, 60, 40, 1]\n",
      "Küme 5 - Rota: [1, 56, 19, 1]\n",
      "Küme 6 - Rota: [1, 9, 1]\n",
      "Küme 7 - Rota: [1, 42, 8, 31, 7, 52, 58, 6, 21, 39,
50, 36, 23, 1]\n",
      "Küme 8 - Rota: [1, 27, 47, 25, 28, 24, 49, 15, 33,
1]\n",
      "Küme 9 - Rota: [1, 12, 18, 1]\n",
      "Küme 10 - Rota: [1, 17, 1]\n",
      "\n",
      "🚚 3. HER ARAÇ İÇİN ROTASYON VE MESAFELER\n",
      "Küme 1 - Araç 1 - Rota: [1, 4, 34, 53, 44, 3, 1] -
Mesafe: 284.1 km\n",
      "Araç 1 - Bayi 4: 0 A, 2 B\n",
      "Araç 1 - Bayi 34: 1 A, 1 B\n",
      "Araç 1 - Bayi 53: 1 A, 0 B\n",
      "Araç 1 - Bayi 44: 1 A, 0 B\n",
      "Araç 1 - Bayi 3: 1 A, 0 B\n",
      "Küme 2 - Araç 1 - Rota: [1, 20, 48, 14, 59, 5, 13,
10, 30, 57, 43, 1] - Mesafe: 55.0 km\n",
      "Araç 1 - Bayi 20: 0 A, 1 B\n",
      "Araç 1 - Bayi 48: 1 A, 1 B\n",

```

" Araç 1 - Bayi 14: 0 A, 3 B\n",
 " Araç 1 - Bayi 59: 0 A, 1 B\n",
 " Araç 1 - Bayi 5: 1 A, 0 B\n",
 " Araç 1 - Bayi 13: 0 A, 1 B\n",
 " Araç 1 - Bayi 10: 0 A, 1 B\n",
 " Araç 1 - Bayi 30: 0 A, 1 B\n",
 " Araç 1 - Bayi 57: 0 A, 1 B\n",
 " Araç 1 - Bayi 43: 0 A, 2 B\n",
 "Küme 2 - Araç 2 - Rota: [1, 51, 26, 55, 1] - Mesafe:
 24.200000000000003 km\n",
 " Araç 2 - Bayi 51: 0 A, 2 B\n",
 " Araç 2 - Bayi 26: 1 A, 0 B\n",
 " Araç 2 - Bayi 55: 0 A, 1 B\n",
 "Küme 3 - Araç 1 - Rota: [1, 37, 29, 2, 41, 22, 54,
 16, 45, 32, 46, 1] - Mesafe: 133.3 km\n",
 " Araç 1 - Bayi 37: 0 A, 1 B\n",
 " Araç 1 - Bayi 29: 1 A, 0 B\n",
 " Araç 1 - Bayi 2: 1 A, 1 B\n",
 " Araç 1 - Bayi 41: 0 A, 1 B\n",
 " Araç 1 - Bayi 22: 0 A, 2 B\n",
 " Araç 1 - Bayi 54: 0 A, 2 B\n",
 " Araç 1 - Bayi 16: 1 A, 1 B\n",
 " Araç 1 - Bayi 45: 0 A, 1 B\n",
 " Araç 1 - Bayi 32: 1 A, 0 B\n",
 " Araç 1 - Bayi 46: 0 A, 1 B\n",
 "Küme 3 - Araç 2 - Rota: [1, 11, 35, 38, 1] - Mesafe:
 62.099999999999994 km\n",
 " Araç 2 - Bayi 11: 0 A, 2 B\n",
 " Araç 2 - Bayi 35: 0 A, 2 B\n",
 " Araç 2 - Bayi 38: 0 A, 1 B\n",
 "Küme 4 - Araç 1 - Rota: [1, 60, 40, 1] - Mesafe:
 351.3 km\n",
 " Araç 1 - Bayi 60: 1 A, 0 B\n",
 " Araç 1 - Bayi 40: 1 A, 0 B\n",
 "Küme 5 - Araç 1 - Rota: [1, 56, 19, 1] - Mesafe:
 263.0 km\n",
 " Araç 1 - Bayi 56: 1 A, 0 B\n",
 " Araç 1 - Bayi 19: 1 A, 0 B\n",
 "Küme 6 - Araç 1 - Rota: [1, 9, 1] - Mesafe: 262.0
 km\n",
 " Araç 1 - Bayi 9: 1 A, 0 B\n",
 "Küme 7 - Araç 1 - Rota: [1, 42, 8, 31, 7, 52, 58, 6,
 21, 39, 50, 36, 23, 1] - Mesafe: 77.5 km\n",
 " Araç 1 - Bayi 42: 1 A, 1 B\n",
 " Araç 1 - Bayi 8: 1 A, 0 B\n",
 " Araç 1 - Bayi 31: 0 A, 1 B\n",
 " Araç 1 - Bayi 7: 0 A, 1 B\n",
 " Araç 1 - Bayi 52: 0 A, 1 B\n",
 " Araç 1 - Bayi 58: 0 A, 1 B\n",
 " Araç 1 - Bayi 6: 1 A, 0 B\n",
 " Araç 1 - Bayi 21: 1 A, 0 B\n",
 " Araç 1 - Bayi 39: 0 A, 2 B\n",
 " Araç 1 - Bayi 50: 1 A, 0 B\n",
 " Araç 1 - Bayi 36: 0 A, 1 B\n",
 " Araç 1 - Bayi 23: 1 A, 0 B\n",
 "Küme 8 - Araç 1 - Rota: [1, 27, 47, 25, 28, 24, 49,
 15, 33, 1] - Mesafe: 120.69999999999999 km\n",
 " Araç 1 - Bayi 27: 0 A, 3 B\n",

```

"    Araç 1 - Bayi 47: 0 A, 1 B\n",
"    Araç 1 - Bayi 25: 1 A, 0 B\n",
"    Araç 1 - Bayi 28: 1 A, 1 B\n",
"    Araç 1 - Bayi 24: 0 A, 1 B\n",
"    Araç 1 - Bayi 49: 1 A, 0 B\n",
"    Araç 1 - Bayi 15: 1 A, 0 B\n",
"    Araç 1 - Bayi 33: 1 A, 0 B\n",
"Küme 9 - Araç 1 - Rota: [1, 12, 18, 1] - Mesafe:
134.2 km\n",
"    Araç 1 - Bayi 12: 0 A, 1 B\n",
"    Araç 1 - Bayi 18: 0 A, 1 B\n",
"Küme 10 - Araç 1 - Rota: [1, 17, 1] - Mesafe: 180.6
km\n",
"    Araç 1 - Bayi 17: 0 A, 1 B\n"
]
}
]
}
]
}

```

ÖZGEÇMİŞ

Ad-Soyad : Çağla Ebru BİLİR

ÖĞRENİM DURUMU:

- **Lisans** : 2016, Sakarya Üniversitesi, Mühendislik Fakültesi, Endüstri Mühendisliği Bölümü
- **Yüksek Lisans** : 2024, Sakarya Üniversitesi, Fen Bilimleri Enstitüsü, Endüstri Mühendisliği Anabilim Dalı, Endüstri Mühendisliği Bilim Dalı

MESLEKİ DENEYİM:

- 2017 – 2019 Yılları arasında Kariyer.net - İnsan Kaynakları Çözüm Ortağı
- 2021 – 2022 Aydın Endüstri-Planet Entegre A.Ş. - Satın Alma ve Stok Sorumlusu

TEZDEN TÜRETİLEN ESERLER:

- Bilir, Ç.E. & Uygun, Ö (2023). Araç Rotalama Problemi İçin Kümeleme Yaklaşımı, Abant 2nd International Conference On Current Scientific Researches, December 28 - 30, 2023 - ABANT