T.C.
İSTANBUL AYDIN ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

SEZGİSEL ALGORİTMA KULLANILARAK EN İYİ YOL ROTALANMASI VE BİR UYGULAMA

YÜKSEK LİSANS TEZİ

Mehmet ŞİRİN
(Y1413.010027)

Bilgisayar Mühendisliği Anabilim Dalı
Bilgisayar Mühendisliği Programı

Tez Danışmanı: Prof. Dr. Ali GÜNEŞ

MART, 2018

T.C.
İSTANBUL AYDIN ÜNİVERSİTESİ
FEN BİLİMLER ENSTİTÜSÜ MÜDÜRLÜĞÜ

Yüksek Lisans Tez Onay Belgesi

Enstitümüz Bilgisayar Mühendisliği Ana Bilim Dalı Bilgisayar Mühendisliği Tezli Yüksek Lisans Programı **Y1413.010027** numaralı öğrencisi **MEHMET ŞİRİN**'nin **"SEZGİSEL ALGORİTMA KULLANILARAK EN İYİ YOL ROTALAMASI VE BİR UGULAMA"** adlı tez çalışması Enstitümüz Yönetim Kurulunun 28/02/2018 tarih ve 2018/04 sayılı kararıyla oluşturulan jüri tarafından oybirliği ile Tezli Yüksek Lisans tezi olarak kabul edilmiştir.

Öğretim Üyesi Adı Soyadı	**İmzası**
Tez Savunma Tarihi : 19/03/2018	
1)Tez Danışmanı: PROF. DR. ALİ GÜNEŞ	.....
2) Jüri Üyesi : PROF. DR. ZAFER ASLAN	.....
3) Jüri Üyesi : DOÇ. DR. TUĞBA ALTINTAŞ	.....

Not: Öğrencinin Tez savunmasında **Başarılı** olması halinde bu form **imzalanacaktır**. Aksi halde geçersizdir.

YEMİN METNİ

Yüksek Lisans tezi olarak sunduğum "SEZGİSEL ALGORİTMA KULLANILARAK EN İYİ YOLDAN ROTALANMASI VE BİR UYGULAMA" adlı çalışmanın, tezin proje safhasından sonuçlanmasına kadarki bütün süreçlerde bilimsel ahlak ve geleneklere aykırı düşecek bir yardıma başvurulmaksızın yazıldığını ve yararlandığım eserlerin Bibliyografya'da gösterilenlerden oluştuğunu, bunlara atıf yapılarak yararlanılmış olduğunu belirtir ve onurumla beyan ederim. (19.03.2018)

Mehmet ŞİRİN

Aileme,

ÖNSÖZ

Akademik çalışmalarımda önemli bir yeri olan tez çalışmamın bütün aşamalarında desteğini esirgemeyen, engin bilgi birikimini ve tecrübesini benimle paylaşarak farklı bir bakış açısı kazanmama vesile olan, danışman hocam değerli bilim insanı Prof. Dr. Ali GÜNEŞ'e teşekkürü borç bilirim. Tez hazırlama sürecinde her türlü desteklerini aldığım İstanbul Halk Ekmek A.Ş. İnsan Kaynakları Müdürü Mustafa HONİ ve İstanbul Halk Ekmek A.Ş İnsan Kaynakları Şefi Fatih AKTÜRK'e hassaten teşekkür ederim.

Nefes alıp verdiğim her anımda olduğu gibi bu tez sürecinde de anlayışlarını ve desteklerini evlatlarından esirgemeyen annem ve babama şükranlarımla…

Mart, 2018 **Mehmet ŞİRİN**

İÇİNDEKİLER

Sayfa

KISALTMALAR

CPU : Merkezi İşlem Birim
ÇPP : Çinli Postacı Problemi
DGSP : Dinamik Gezgin Satıcı Problemi
DKA : Değişken Komşuluk Araması
EKS : Elitist Karınca Sistemi
ETDARP : Eş Zamanlı Topla Dağıt Araç Rotalama Problemi
EYK : En Yakın Komşu
GA **:** Genetik Algoritmalar
GPS : Küresel Konumlandırma Sistemi
GSP : Gezgin Satıcı Problemi
KKA : Karınca Kolonisi Algoritması
KKE : Karınca Kolonisi Eniyilemesi
KKO : Karınca Kolonisi Optimizasyonu
KKS : Karınca Koloni Sistemi
KSA : Karınca Sistemi Algoritması
Np-Zor **:** Çokgensel Olmayan Zor Problemler
PSO : Parçacık Sürü Optimizasyonu
RHO : Feromon Sıvısı Buharlaşma Oranı
TGP **:** Temel Güzergâh Problemleri
TSP : Gezgin Satıcı Problemi (Travelling Salesman Problem)
TSPLIB : Gezgin Satıcı Problem Örnekleri Kütüphanesi

SİMGELER

A	: Feromon katsayısı
B	: Görünürlük katsayısı
P	: Buharlaşma katsayısı
m	: Karınca sayısı
n	: Büfe sayısı
pijk	: k karıncasının i noktasından j noktasına geçme ihtimali
T	: Döngü sayacı
t0	: Başlangıç feromon madde miktarı
τij (t)	: t döngüsünde (i,j) güzergâhı üzerinde bulunan feromon sıvısı miktarı
Δτijk(t)	: Döngü sonunda k karıncasının (i,j) arasına bırakılacak iz miktarı
Q	: İz miktarının belirlenmesinde kullanılan sabit
Lk	: k karıncası tarafından oluşturulan çözümün tur uzunluğu
Nt	: Tur sayısı (iterasyon)
Nmin	**:** En kısa tura kaçıncı turda ulaşıldığı
Ψ0	: Başlangıç çözümü

ÇİZELGE LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

SEZGİSEL ALGORİTMA KULLANARAK EN İYİ ROTALAMA OLUŞTURMA VE BİR UYGULAMA

ÖZET

Bu çalışmada çözülmesi zor problemlerden biri olan Gezgin Satıcı Problemi ele alınmıştır. GSP sezgisel yöntemlere yol göstermesi amacıyla en iyi sonuçların elde edilebilmesi için çözümüne yönelik karınca kolonisi algoritması kullanılmıştır. Amaç GSP üzerinde karınca kolonisi algoritması kullanılarak İstanbul Halk Ekmek şirketinin ekmek dağıtım araçlarının rotalama probleminin çözülmesidir. Karıncaların aralarındaki iletişimi sağlamaları için salgıladıkları feromon adlı sıvı haberleşmeleri açısından en temel madde olarak bilinmektedir. Bu algoritmada bulundukları alanda en kısa yolu bulma mantığıyla çalışan yapay karıncalardan faydalanılmıştır. Programımızdaki amaç, dağıtım araçlarının büfelere en kısa yoldan rota hesaplaması yapılmasıdır. Programda oluşturulan rota Google maps ile harita üzerinden görüntülenmektedir. Yapılan uygulamanın etkinliğini ve performansını ölçmek için aynı rotalar Excel- Solver (Evolutionary) çözülüp karşılaştırılmıştır.

Anahtar Kelimeler: *Karınca Kolonisi Algoritması, Gezgin Satıcı Problemi, Web Tabanlı Uygulama, Rotalama*

CREATING THE BEST ROUTING USING HEURISTIC ALGORITHM,

AND AN APPLICATION

ABSTRACT

In this study, Travelling Salesman Problem (TSP), an NP-hard problem, is addressed. In order to get the best results with a view to directing TSP heuristics, the ant colony algorithm was used for solution purposes. The purpose was to solve the problem of setting a course for the bread distribution trucks of Istanbul Halk Ekmek (Public Bread) Company using the ant colony algorithm on TSP. A liquid called Pheromone, which ants release in order to establish communication among them, is known as the most fundamental matter to provide this communication. In this research, artificial ants, which function with the logic of finding the shortest path in the area where they are located, were utilized. The purpose of our programme is to determine the shortest route for the arrival of the distribution trucks to the kiosks where bread is sold to the public. The route developed by the programme is displayed over Google maps. In order to measure the efficiency and performance of the implemented application, the same roots Excel-Solver (Evolutionary) were solved and compared.

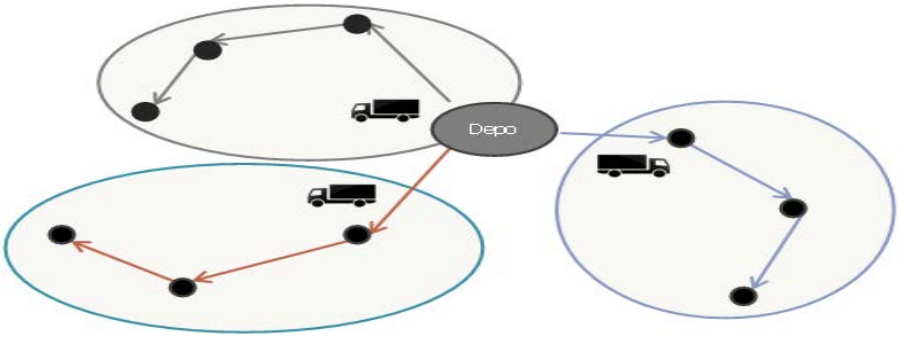
Keywords: *Ant Colony Algorithm, Travelling Salesman Problem, Web-Based Application, Routing*

1. GİRİŞ

Şirketlerin artan rekabet ortamında bilginin, hammaddenin, mal ve hizmetlerin dağıtım sorunları giderek değer kazanmaktadır. İşletmelerin dağıtım faaliyetlerindeki rekabet, rotalama şekillerinin farklılaşmasına neden olmuştur. İşletmelerin dağıtım ağı süreçlerini iyileştirmek ve en verimli hale getirmek amacıyla özellikle taşıma ve dağıtım sorunlarına çeşitli çözümlerin geliştirilmesi gerekmektedir. Bu kararlardan birisi de fabrika depolarından müşterilere gerçekleştirilecek olan rotalama kararlarının alınmasıdır. İşletmelerin yönetmek zorunda kaldığı bu problemler literatürde yeni problemlerin çıkmasına ve bu problemlerin çözümleri için yeni tekniklerin geliştirilmesine neden olmuştur. Gezgin Satıcı Problemi son yılarda araştırmacıların en çok üzerinde çalıştığı optimizasyon problemlerinin başında gelmektedir. Gezgin Satıcı Problemi (GSP) bir noktadan başlayıp, listedeki tüm noktalara sadece bir kez ziyaret edip, tekrar başladığı noktaya varan yolu veya turu bulmayı amaçlayan bir optimizasyon problemidir. GSP, çözüm zorluğu bakımından NP-Zor problem sınıfında yer almaktadır [1].

Literatürde, araç rotalama problemi GSP daha genişletilmiş hali olarak ifade edilmektedir. Bunun nedeni ise birden çok araç kullanılması ve birden çok kısıtlar eklenmiş olmasıdır.

İlk kez 1959 yılında Dantzig ve Ramser Araç Rotalama Problemi üzerinde araştırma yapmışlardır. Yapmış oldukları bu çalışmada, benzin istasyonlarındaki araçların en kısa yoldan ve en az maliyetle araç rotalamalarının minimize edilmesi hedeflenmiştir [2]. Şekil 1.1 ‘de GSP problemine ait gösterim verilmektedir.



Şekil 1.1: Gezgin Satıcı Problem Gösterimi

2. LİTERATÜR ÖZETİ

Temel güzergâh problemleri gezgin satıcı problemlerinin ortaya çıkması ve çeşitli kısıtlar eklenerek çeşitlenmesinden meydana gelmiştir [3]. Bu alanda araştırmalar 1950 yılında yapılmaya başlamıştır. GSP, rotalama ve karınca kolonisi algoritması konularında yapılan araştırmalar aşağıda incelenmiştir.

Şükran (2017), çalışmasında Stokastik Araç Rotalama Problemini çözümü üzerinde durmuştur ve çözüm yöntemi olarak Genetik Algoritma kullanmıştır. Yapılan çalışma sonucunda çözüm uzayı çok büyük olan araç rotalama da problemin belirli bir matematiksel modelle ifade edilemediği, geleneksel eniyileme yöntemlerinden istenen sonucun alınmadığı alanlarda araştırmada kullanılan yöntem daha etkili ve kullanışlıdır [4].

Rabia (2017), eş zamanlı topla dağıt araç rotalama problemi üzerinde durulmuştur. Karşılaşılan bu problemin çözümü için karışık tamsayılı matematiksel model kullanılmıştır ve sezgisel bir algoritma geliştirilmiştir. Araştırma sonucunda kullanılan algoritmanın etkili sonuç verdiği görülmüştür [5].

Can (2017), yapılan araştırmada Eş Zamanlı Topla Dağıt Araç Rotalama Problemi'nin (ETDARP) üzerinde durulmuştur. Problemin çözmek için Karınca Koloni Sistemi (KKS) ile Değişken Komşuluk Araması'na (DKA) dayanan melez bir meta sezgisel algoritma geliştirilmiştir. Geliştirilen yaklaşımın hem çözüm kalitesi hem de CPU süresinde güçlü ve etkili olduğu sayısal sonuçlar ile kanıtlanmıştır [6].

Gözde (2017), araştırmasında Seçici Kümelendirilmiş Gezgin Satıcı Problemi ve Seçici Genelleştirilmiş Gezgin Satıcı Problemini incelemiştir. Bu iki problemin çözümü için iki tane ayrıt tabanlı matematiksel model önermiştir. Çalışmasının sonunda önerilen modellerin performansları analiz edilmiş ve sonuç olarak her iki problem için de ayrıt tabanlı matematiksel modelin verimli olduğu görülmüştür [7].

Saniye (2017), araç rotalama problemi araştırılmış ve meta sezgisel yöntem olan genetik algoritma yardımı ile çözüm aranmıştır. Geliştirilen algoritmanın etkinliğini ve performansını ölçmek için aynı rotalar Excel- Solver (Evolutionary) Genetik

Algoritmalar Programı ve En Yakın Komşu Algoritması ile de optimize edilmektedir. Sonuç olarak geliştirilen algoritma sonuçları diğer yöntemlere göre daha başarılıdır [8].

Çakır (2016), Sakarya ilinde faaliyet göstermekte olan, bir tıbbi atık toplama ve sterilizasyon tesisinin verileri kullanılarak, tıbbi atıkların en doğru ve maliyeti etkin bir biçimde taşınması konusunu ele almıştır. Tersine lojistiğin kapsamında, tersine lojistik faaliyetlerinden biri olan atık yönetimi incelemiş ve tıbbi atıkların toplanması konusunu ele almıştır. Bu çalışma kapsamında yapılan uygulamanın amacı, tıbbi atıkların toplanmasında kullanılan araçların kat edecekleri mesafeyi en aza indirerek, en uygun rotayı belirlemektir. Uygulamada kat edilen mesafenin en aza indirilmesi, tıbbi atıkların toplanmasındaki taşımacılık maliyetlerinin azalması amaçlanmıştır [9].

Gizem (2015), Çalışmanın temel amacı NP-zor optimizasyon problemlerini çözmede Grafik İşlemci Birimi kullanımının avantajlarını göstermektir. Bu nedenle, gezgin satıcı problemini çözmek üzere 2-opt ve 3-opt yöntemleri paralel olarak uygulanmıştır. Bu araştırmanın amacı Grafik İşlemci Biriminin etkin kullanımıyla ilgili faydalı bilgiler vermek ve optimizasyon alanındaki araştırmacıların bu konuya aşina olmalarını sağlamaktır [10].

Ahmet (2015), çalışmasında Simetrik Gezgin Satıcı Problemine üzerinde durulmuş ve çözüm için 3 farklı algoritma (Yakın Çift, Solucan, Örümcek Ağı) geliştirilmiştir. Çalışma süreleri ve çözümün değerleri karşılaştırılmıştır. Sonuç olarak Örümcek Ağı algoritması bu yarışın galibi olmuştur [11].

Alpaslan (2015), Araç Rotalama Problemleri İçin Matematiksel Modeller ve Çözüm Yöntemleri adlı çalışmasında, araç rotalama problemlerini incelemiş ve bu problemlere yönelik yeni karma tam sayılı tek amaçlı ve çok amaçlı modeller geliştirilmiş, literatürde daha önceden ele alınmayan, kullanılan araç türü en küçüklenmesi amaçlanmıştır. Algoritma, literatürdeki test problemleri üzerinde denenmiş ve elde edilen hesaplamalı sonuçlar kıyaslamalı bir şekilde sunulmuştur [12].

Atasagun (2015), araştırmasında eş zamanlı topla-dağıt araç rotalama problemi ile zaman bağımlı araç rotalama problemlerini bir arada çözebilmek için zamana bağımlı eş zamanlı topla-dağıt araç rotalama problemini birleştirmiştir. Problemin çözümü için bir matematiksel model geliştirilerek literatürdeki bulunan test

problemleri ile deneysel çalışmalar yapmış ve yorumlamıştır [13].

Eldem, H. (2014), Karınca kolonisi optimizasyonu yönteminin ürettiği sonuçların iyileştirilmesi için, Parçacık sürü optimizasyonu yönteminin birlikte çalışabileceğini ele alınmıştır. Literatürde optimizasyon problemlerinin çözümünde sıkça kullanılan test fonksiyonlarından Gezgin Satıcı Probleminin çözümünde, önerilen yöntem kullanılmıştır. Bu şekilde sıralı şekilde optimum başlangıç yolları sonuçlarını veren KKO algoritmasının yaptığı çıkarımların, PSO tarafından optimize edilmesi sağlanmıştır. Bu çalışmada, KKO ve PSO algoritmalarının temel halleri işlenmiş ve ürettiği sonuçlar değerlendirildiğinde, tavsiye edilen sıralı yaklaşımın, temel KKO ve temel PSO algoritmalarından elde edilen sonuçlar ile karşılaştırıldığında daha iyi sonuçlar elde ettiği görülmüştür [14].

Hasan ve diğerleri (2014), Bu çalışmada, rota planlama problemlerinden olan gezgin satıcı probleminin (GSP) çözümünü gerçekleştirmek için yapay zekâ tekniklerinden olan karınca kolonisi ve genetik algoritmaların performansları karşılaştırılmıştır. Araştırma sonucunda karınca kolonisi algoritmasının hem rota mesafesi hem de başarım süresi yönünden genetik algoritmalara göre daha üstün olduğu gözlemlenmiştir [15].

Ekmekçi ve Kosif (2012), araç rotalama yöntemlerinden biri olan tasarruf algoritmasını kullanmıştır. Ülkemizde bulunan bir lojistik firmasının müşterilerinin ihtiyaçlarına yönelik, mevcut durumun optimize edilmesi ve lojistik maliyetlerinin en aza indirilmesini sağlayan bir çalışma yapmıştır [16].

Gökalp (2012), Birbirinden bağımsız yapay karınca kolonileri tarafından üretilen feromon tabloları, birer kromozom olarak düşünülmektedir. Bunlara çaprazlama işlemleri uygulandıktan sonra, bir sonraki soydaki karınca kolonilerine feromon tablosu olarak yansıtılmaktadır. Çaprazlamanın uygulanmasındaki temel düşünce, farklı koloniler tarafından feromon izleri şeklinde depolanmış olan yerel bilginin paylaşılması, güçlü yönlerinin birleştirilmesi ve yerel en iyiye takılmadan, evrensel en iyi çözüme ulaşılma olasılığının artırılmasıdır. Yeniden yapılandırılan bu yöntemler Gezgin Satıcı Problemi ile TSPLIB' de var olan bir takım karşılaştırma problemleri ile Maksimum Minimum KSA üzerinde denenmiş ve alınan sonuçlar yansıtılmıştır. Bu çalışmada, çaprazlama mekanizmalarının, KKE algoritmalarının performansını iyileştirdiğini göstermiştir [17].

Suvaydan (2011), mobil robotların yol planlama problemlerinin çözümü hakkında literatür taraması yapıp, yol planlaması hakkında bilgi birikimi sağlamak ve sezgisel bir teknik olan Karınca Kolonisi Algoritması yardımıyla bir noktadan hedeflenen bir noktaya engellere çarpmaksızın optimum kriterlere uygun yol planlaması yapılmıştır. Buna paralel olarak Karınca Kolonisi Algoritmasının lokal ve global feromon güncellemesine göre sonuçlar elde edilmeye çalışılmış bir simülasyon programı elde edilmiştir. Bu simülasyon programı sayesinde gereken değerlerler girilip değerlendirmeler yapılarak en sağlıklı sonucun bulunması hedeflenmiştir [18].

Çalışkan (2011), yapmış olduğu çalışmasında karınca kolonisi optimizasyonu ile araç rotalama probleminin maliyetlerinin kümeleme tekniği ile iyileştirilmesini hedeflemiştir. Araç rotalama problemine karınca kolonisi optimizasyonu uygulanarak taleplerin toplam gerçekleşme süresi, toplam mesafe ve kullanılan toplam araç sayısının değişimleri izlenmiştir [19].

Seçkin (2010), yapmış olduğu tez çalışmasında genetik algoritmanın çalışma mantığına üzerinde durulmuş ve geliştirmiş olduğu genetik algoritmayı kullanarak gezgin satıcı problemini çözümünü hedeflemiştir. Genetik olduğu algoritmaların avantajı ve dezavantajına değinmiştir. Araştırmanın sonucunda genetik algoritmaların az zamanda en iyiye yakın sonuç verdiği görülmektedir [20].

Keskintürk (2009), Araç Rotalama Problemlerinin Global Karınca Koloni Optimizasyonu ile Çözümü adlı tezinde özetle araç rotalama problemlerinin çözümüne yönelik global KKO önerilmiştir. Farklı tipteki araç rotalama problemlerine uygulanan yöntem literatürden alınan problemler üzerinde test edilmiş ve mevcut karınca koloni algoritmalarıyla karşılaştırılmıştır. Ayrıca bir şirkete ait dağıtım filosun yöntemle rotalanıp çözüm mevcut durumla karşılaştırılmış ve sonuçlar yorumlanmıştır [21].

Hasan Timur (2007), araştırmasında karınca kolonisi algoritmasının çalışma şeklinin ve prensiplerinin gösterilmesidir. Farklı Karınca kolonisi algoritmalarında incelendiği çalışmada örnek bir GSP problemine yer verilmiş ve sonuçlar diğer yöntemlerin sonuçları ile karşılaştırılmıştır [22].

3. TEMEL GÜZARGAH PROBLEMLERİ

Temel Güzergâh Problemleri (TGP), tedarik ve planlama işlemlerin en son ve en önemli parçası olan ürünlerin müşterilere dağıtımı problemi olarak karşımıza çıkmaktadır. Temel Güzergâh Problemleri arasından en sık araştırma konusu olarak Gezgin Satıcı Problemi tercih edilmektedir. Diğer Temel Güzergâh Problemleri ise, Gezgin Satıcı Probleminin türevleri olarak görülmektedir [23].

3.1 Gezgin Satıcı Problemi (GSP)

Gezgin Satıcı Problemi bir noktadan başlayıp, listedeki tüm noktalara sadece bir kez ziyaret edip, tekrar başladığı noktaya varan yolu veya turu bulmayı amaçlayan bir optimizasyon problemidir. Gezgin satıcı problemi başlangıç ve bitiş noktaları farklı olan problemlerin geliştirilmesi ile ortaya çıkmıştır. Aracın başlangıç noktası olan depoya dönmesi, turun gerçekleşmesi için gereklidir. GSP başladığı noktaya geri dönmesi problemin çözümünü zorlaştırmaktadır. GSP çözmekteki asıl amaç tüm müşterileri noktalarını en kısa zamanda en az yol giderek turun nasıl gidileceğinin bulunmasıdır. Atık toplama, okul taşıt güzergâhlarının belirlenmesi gibi problemleri örnek olarak verebiliriz. Bu problemlere genel ifade ile “Gezgin Satıcı Problemi" denmektedir [24].

3.2 Dinamik Gezgin Satıcı Problemi

Dinamik Gezgin Satıcı Problemi (DGSP) ise, kenarların ağırlık değerlerinin, düğümlerin sayı ve yerlerinin zamanla değişebildiği bir çizgedeki en kısa Hamilton turunu bulmayı amaçlayan bir problemdir [25]. Dinamik gezgin satıcı probleminde GSP’den farklı olarak düğüm ya da bir başka deyişle şehir sayısı zamanla değişebilir yani zamanla bazı düğümler kaybolabilir veya bazı yeni düğümler ortaya çıkabilir. Benzer biçimde düğümlerin pozisyonları ya da iki düğüm arasındaki maliyet değeri de değişime uğrayabilir. İki düğüm arasındaki maliyeti simgeleyen bu değer problemin türüne göre uzaklık, zaman ya da para

gibi bir değer olabilir.

Dinamik gezgin satıcı problemi temel özellikler açısından gezgin satıcı problemine çok benzer olmasına rağmen, kendine özgü bazı karakteristik özelliklere de sahiptir [25]. Bunları şu şekilde sıralayabiliriz:

- **Süreklilik (Continuity):** Problem zamanla kısmen ya da nicel olarak değişikliğe uğrayabilir.
- **Sağlamlık (Robustness):** Dinamik GSP, bir düğümün silinmesi ya da eklenmesi gibi beklenmeyen durumlara hızlı bir biçimde cevap verebilmeyi gerektirir.
- **Etkinlik (Efficiency):** Dinamik GSP, en iyi turun makul bir sürede bulunmasını gerektirir.

3.3 Çinli Postacı Problemi (ÇPP)

Çinli Postacı Problemi tüm gidilecek noktalara en az bir defa uğramak şartı ile en az maliyetle problemin çözülmesidir. ÇPP, 1962 yılında Çinli matematikçi Mei-Ko Kwan ilk kez araştırmıştır. Bu problem, postacının mektuplarını en kısa yoldan şehirdeki tüm sokaklara uğrayarak mektupları dağıtması gerektiğinden ortaya çıkmıştır. Mektupları dağıtan postacı başladığı noktaya posta haneye geri dönmek zorundadır [26]. ÇPP, gezgin satıcı problemine benzerlikleri olmasına rağmen farklılıkları vardır. GSP, noktaların birleşmesi problemi olup tüm noktalara yalnızca bir defa uğrayarak en kısa turun (Hamilton turun) bulunmasıdır. ÇPP ve GSP karşılaştırıldığında ise ÇPP'nde düğümler yerine bu düğümleri birbirine bağlayan ayrıtlardan en az bir kez geçilmesi şartıdır [27]. ÇPP eğer tam bir turu (Euler) tamamlayamıyorsa ayrıtlardan birden fazla geçilebilmektedir.

3.4 Çoklu Gezgin Satıcı Problemi

Gezgin satıcı problemlerinin birden çok araç eklenerek en kısa yolun bulunması durumudur. Bir satıcıda bulunan N sayıda araç aynı başlangıç noktasından çıkarak tekrar başlangıç noktasına geri dönmesidir. Her araç en az bir noktaya gitmesi zorunludur. Birden fazla düğüm noktasına uğramasında bir kısıtlama yoktur.

3.5 Tek Depo, Çok Araç ve Çok Duraklı Dağıtım Problemi

En yaygın karşılaşılan satıcı problemidir. Karşılaşılan bu problemin çözüm yöntemi, tek depodan çıkan araçların tüm noktalara en kısa rotayı bularak tekrar aynı başlangıç noktasına dönmelerine sağlayacak rotanın bulunmasıdır. Bu problemde tüm noktadaki taleplerin deterministik olarak bulunur. Bu konu ile ilgili Golden tarafından araştırmalar gerçekleştirilmiştir [28].

3.6 Çok Depo, Araç ve Çok Duraklı Dağıtım Problemi

Bir satıcıda bulanan tüm araçların, birden fazla depodan yola çıkmasıdır. Bu problemde her aracın aynı depodan çıkıp aynı depoya dönmesi gerekmektedir. Gillett ve Johnson, atama-süpürme (assignment-sweep) yöntemi ile bu problemi iki adımda çözmüşlerdir. İlk olarak dağıtım noktaları depolara atanmaktadır. İkinci olarak da depo ve ona atanmış dağıtım noktaları için çözüm uygulanmaktadır. Her iki aşama birbirinden ayrı olarak ele alınmaktadır.

3.7 Tek Depo, Çok Araç ve Tahmini Talepli Dağıtım Problemi

Bu problemin TGP farkı dağıtım noktalarındaki talebin miktarının kesin olarak bilinmemesidir. Talebin miktarının bilinmemesinden dolayı olasılık dağılımından faydalanılmaktadır. Karşılaşılan bu problemin çözümü için sezgisel çözüm yöntemi uygulanmaktadır.

3.8 Kapasite Tahditli Dağıtım Problemi

Kapasite Tahditli Dağıtım probleminde tüm noktaları depo olarak kullanılmaktadır. Her bir noktaya tek bir aracın uğraması ile oluşturulacak araç rotalama modelidir. Araçların kapasiteleri sınırlı olduğundan rota oluşturulurken bu sınırlar aşılmamalıdır. Amaç araçların ve rotanın toplam maliyetini minimize etmektir.

3.9 Maliyet Tahditli TGP

Kapasite tahditli dağıtım problemden farkı araç kapasite tahditinin kullanmak yerine araç maliyet tahditinin kullanılmasıdır.

3.10 Maliyet ve Kapasite Tahditli Dağıtım Problemi

Kapasite tahditli ve Maliyet Tahditli problemlerin çözümü için kullanılan yöntemler bu problem içinde kullanılabilmektedir. Fakat her iki kısıt da birden uygulanarak model oluşturulmaktadır.

3.11 Zaman Tahditli Dağıtım Problemi

Araç rotalama problemleri arasında çözümü en zor olan problemdir. Karşılaşılan bu problemde araçların her bir noktaya belli bir zaman aralığında gitmesi ve buna göre rotalama oluşturulması gerekmektedir.

4. GSP UYGULAMA ALANLARI

GSP probleminin en yaygın kullanıldığı alanlardan bazıları şunlardır,

- ✓ Çeşitli araçların rotalanması (Servis, devriye araçlarının vs.),
- ✓ Satıcıların ürünlerini bir veya daha fazla depodan satış noktalarına dağıtımı,
- ✓ Uçak güzergâhlarının planlanması,
- ✓ E ticaret sitelerinde yapılan satışların teslimat problemi,
- ✓ Ürün dağıtım problemleri,
- ✓ Depo malzemeleri toplama problemleri,
- ✓ Tüm toplama ve dağıtma problemleri

5. GSP ÇÖZÜM YÖNTEMLERİ

Gezgin Satıcı Problemlerinin çözümü için bulunan birden fazla yöntem bulunmaktadır. Bunlar kesin, klasik sezgiseller ve meta sezgiseller olarak gruplanmaktadır. Bulunan kesin çözüm yöntemi en iyi sonucu göstermektedir fakat süre bakımında uzun sürebilmektedir. Diğer çözüm yöntemleri ise en iyiye yakın çözümleri kısa sürede bize sunmaktadır.

5.1 Kesin Çözüm Yöntemleri

Kesin çözüm yöntemi en iyi sonucu göstermektedir. Karşılaşılan en büyük problem ise problemin boyutuna bağlı olarak çözüm süresi üstel olarak artmaktadır. Boyutu büyük ve karmaşık problemlerin dışında kalan problemler için uygundur. Çözüm yöntemi kolay ve ispatlanabilir biçimdedir. Bütün araç rotalama problemlerine uygun ve kesin çözüm bulan yöntem bulunmamaktadır [29] [30].

5.1.1 Kesme Düzlem Algoritması

Kesme düzlem algoritmaları ayarlanmış olan en uygun çözüm parametrelerinin bazı parçaları yok sayılarak, tam sayılı en iyi sonuca ulaşmak istenir [31]. Diğer bir ifade ile kısıtlar kullanılarak tam sayı sonucunun sağlanmasıdır. Kısıtlar eklendikten sonra bulunan yeni çözüm tam sayı ise en iyi çözüm bulunmuştur. Bulunan sonuç kesirli ise tam sayılı sonuç bulanana kadar çözme işlemine devam edilir [32].

5.1.2 Dal ve Sınır Algoritması

1983 yılında Balas ve Toth tarafından dal ve sınır algoritması geliştirildi. Alt tur kısıtları yok sayılarak atama problemi haline getirilmiştir. Satır ve sütün ekleme yöntemiyle rotaları bulmayı hedefler. Çözüme uygun olmayan rotalara atama yapılmasını engellemek için maliyet matrisinde ceza katsayısı uygulanır ve tüm dallar için uygun çözümler bulunana kadar işlem tekrar edilir.

5.1.3 Dal ve Kesme Algoritması

Dal ve kesme düzlem yöntemlerinin birleşiminden meydana gelmiştir [33]. Problemin çözümü için fonksiyon ve kısıtlar kullanılarak çözüm modeli oluşturulur. Modelin çözümünde oluşan alt turlar sıfıra eşitlenir ve problem dallara ayrılır. Oluşan alt dallara kısıtlayıcısı eklenerek optimum sonuç bulunur [34].

5.1.4 Dinamik Programlama

Dinamik programlama problemi birbirinden bağımsız alt problemleri ayırmaktadır. Bu oluşan alt problemlerin çözümleri kullanılarak uygun araç rotaları bulunur [35].

5.2 Klasik Sezgisel Çözüm Yöntemleri

GSP için en uygun çözümü üretebilecek olası rotalama çözüm yöntemlerinden birisidir. Tüm olasılıklar denenebilirse, en iyi çözüme ulaşılmaktadır. Fakat büyük boyutlu problemlerin çözümünde tüm kombinasyonlarının denenmesi ve en iyi çözümün bulunması zaman açısından mümkün değildir. En iyi çözüme ulaşmak için çok fazla zaman gerekmektedir. Bundan dolayı en iyi sonuca yakın ve aynı zamanda hızlı çözüme ihtiyaç duyulması sezgisel yöntemleri geliştirmiştir [36] [37].

5.2.1 Tur Kurucu Sezgiseller

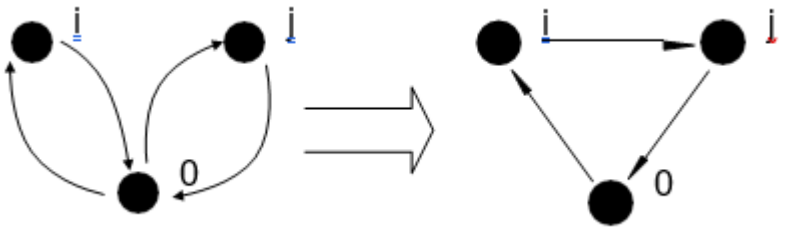
1959 – 1990 yıllarında geliştirilmiştir. Tur kurucu sezgiseller, gerçekleşmeyecek atamalar ile çözüme başlar, her seferinde iki düğüm arasına bir kenar ekleyerek en iyi çözümü hedefler. Kenar eklenirken araç kapasite uyulup uyulmadığı denetlenir.

5.2.2 Dantzig ve Ramser'in Yöntemi

Araç Rotalama Problemi ilk olarak Dantzig ve Ramser 1959 yılında araştırmıştır. Bu çalışmada, benzin istasyonlarındaki müşterilerin belirttiği kısıtlara göre araçların en kısa yoldan ve en az maliyetle rota hesaplama yapılması için araç rotalamalarının minimize edilmesi hedeflenmiştir [38]. Kullanılan algoritmaya göre, herhangi iki noktanın toplam talebi araç kapasitesinin yarısından fazla ise bu iki noktanın birbirine bağlanmaması gerektiğini ifade ederler [2].

5.2.3 Clarke ve Wright Tasarruf Algoritması

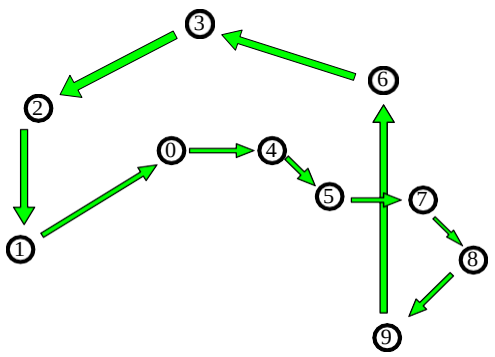
1964 yılında Clarke ve Wright tarafından Dantzig ve Ramser'in çalışmasından yararlanılarak geliştirilmiştir. Tur kurucu sezgisel algoritmalar arasında en çok kullanılanıdır. Herhangi iki nokta birbirine bağlanmasa da, depodan çıkan araç, noktaların her birine gidip dönecektir. Noktalar birbirine bağlandıktan sonara araç sırayla birinci ve ikinci noktaya gidecek ve depoya geri dönecektir. Böylelikle iki noktanın bağlanması ve bağlanmaması arasındaki farka göre tasarruf oluşacaktır. Tasarrufun oluşması Şekil 2.1 'de gösterilmiştir.



Şekil 2.1: Tasarrufun Oluşturulması

5.2.4 En Yakın Komşu (EYK)

1966 yılında Bellmore ve Nemhauser tarafından EYK geliştirilmiştir. Gezgin Satıcı Probleminin çözümü için geliştirilen basit bir algoritmadır. İlk olarak başlangıç noktasından en yakın noktaya gidilir. Bundan sonraki adımlarda ise bir sonraki en yakın noktaya gidilerek en son başlangıç noktasına geri dönülür. En yakın komşu algoritmasının bir örneği Şekil 3.1 'te gösterilmiştir.



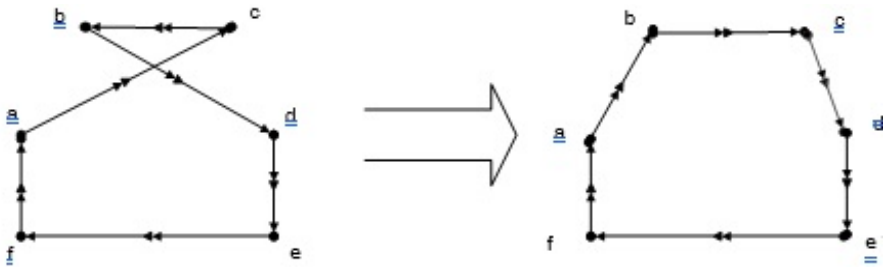
Şekil 3.1: En yakın komşu algoritması

5.2.5 Tur Geliştirici Sezgiseller

Tur geliştirici sezgiseller ilk oluşturulan tur çözümü başlangıç çözümü olarak alınır ve çözüm geliştirilir. Her tekrarda rota değiştirilir ve değişimin daha iyi bir çözüm olup olmadığı kontrol edilir [39] [40].

5.2.6 2-Opt Algoritması

1958 yılında ilk olarak Croes tarafından geliştirilmiştir. Başlangıç çözümü olarak rastgele çözüm alınacağı gibi uygun görülen çözümde seçilebilir. Algoritma ile ilk önce muhtemel çözüm üzerinden iki nokta çifti belirlenir (a-c, b-d). Daha sonra turu bozmayacak şekilde iki nokta yer değiştirir (a-b, c-d). Noktaların değişimi Şekil 4.1 ’de gösterilmiştir.

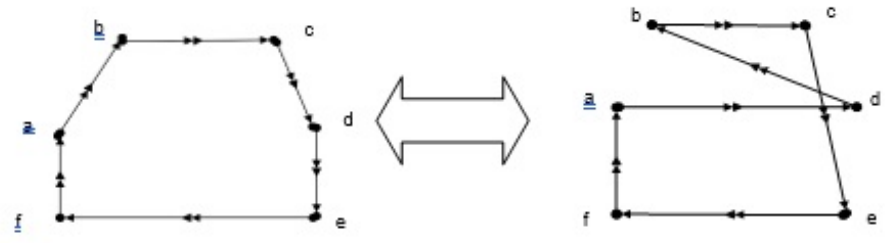


Şekil 4.1: 2-opt örneği

Yeni bulunan çözüm bir öncekine göre daha iyi ise kabul edilir. Aksi halde bir önceki çözümle devam edilir. Bu yöntem ile sadece iki noktanın yeri değişir ve rotadaki diğer noktalar sabit kalır. İkili değişimle daha iyi sonuç bulunamayana kadar devam edilir ve tüm ihtimaller denenir.

5.2.7 3-Opt Algoritması

3-opt algoritması da 2-opt algoritmasına benzer fakat 2 nokta yerine 3 nokta seçilerek yerleri değiştirilir. Bu değişim Şekil 4.1 ve Şekil 5.1 ’de gösterildiği gibi iki farklı şekilde gerçekleşebilir [40].



Şekil 5.1: 3-opt gösterimi

5.2.8 K-Opt Algoritması

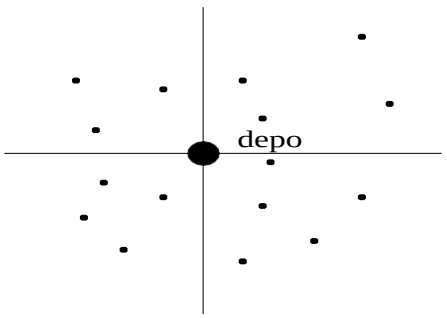
K-Opt Algoritması 4 ya da daha fazla noktanın seçilmesidir. Fakat k-opt algoritmasının çözüm için çok fazla zaman harcanmaktadır. Bulunan çözüm 2-opt ve 3-opt'a göre çok az bir iyileşme göstermektedir [40].

5.2.9 İki Aşamalı Metotlar

İki aşamalı metotlarda birinci aşamasında, noktalar araçlara kapasiteyi aşmayacak şekilde paylaştırılır. İkinci aşamada ise her bir araç için rota oluşturulur [39].

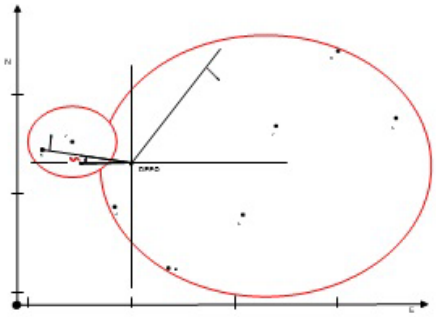
5.2.10 Süpürme (Sweep)

1971 yılında Gillett ve Miller tarafından geliştirilmiştir ve iki aşamalı bir algoritmadır. İlk aşamada, depo merkez seçilerek tüm noktalar koordinat düzleminde Şekil 6.1 'de gösterildiği gibi çizilir. İkinci aşamada ise her i noktası için merkez depo ve X(doğu-batı) ekseni ile olan θi açı değeri hesaplanır.



Şekil 6.1: Depo ve noktalar kutupsal koordinat düzleminde gösterimi

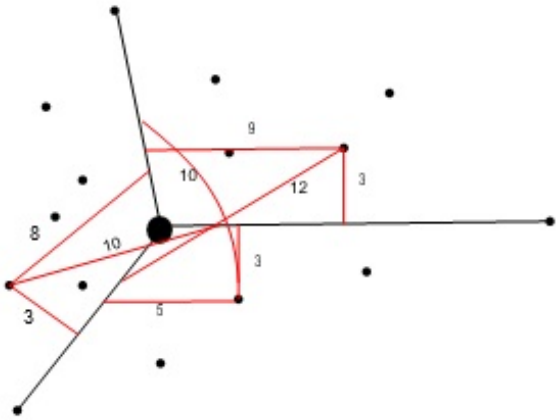
θi değeri en küçük olan nokta bulunur ve kapasitesine ulaşana kadar taranarak noktalar bir küme içerisine alınır. Limite ulaşınca bir sonraki nokta eklenmez ve yeni ikinci nokta kümesi oluşturulur. Bu şekilde tüm rotalar Şekil 7.1 'deki gibi oluşturulur. İkinci aşamada oluşan kümeler gezgin satıcı problemine göre çözülür. İlk aşamadaki tarama saat yönünde veya saat yönünün tersinde yapılabilir [41].



Şekil 7.1: Noktaların gruplanması

5.2.11 Fisher ve Jaikumar Algoritması

1981 yılında Fisher ve Jaikumar tarafında iki aşamalı metodu olarak geliştirilmiştir. İlk aşamada müşteriler arası mesafeler hesaplanır. Her araç için müşteri sayısı belirlenir. Birbirlerine olan mesafeleri en uzak olan müşteriler seçilir. Daha sonra ise noktalara oluşturulan rotalara ve depo arasındaki uzaklıkları hesaplanır. Kapasite limiti dikkate alınarak en yakın noktalar rotalar Şekil 8.1 'deki örnekte görüldüğü gibi atanır. İkinci aşamada gruplar GSP çözülür [29].



Şekil 8.1: Koordinat düzleminde deponun ve noktaların gösterimi

5.2.12 Christofides, Mingozzi ve Toth

1979 yılında Christofides, Mingozzi ve Toth tarafından iki aşamalı yöntem olarak geliştirilmiştir. Bu yöntem, zaman, mesafe veya kapasite kısıtlı araç rotalama problemlerinin çözümünün bulunması için geliştirilmiştir. Kullanıcı tarafından girilen $\lambda \geq 1$ ve $\mu \geq 1$ parametrelerine göre sonuç üretir. İki çözümden en iyisi alınır. Bu işlem λ ve μ parametrelerinin farklı değerleri için tekrar yapılır. Birinci aşamada seriler, ikinci aşamada ise paralel rotalar bulunur [42] [43].

5.3 Meta Sezgisel Çözüm Yöntemleri

Meta sezgisel çözüm yöntemleri son yıllardaki araştırmalar ile büyük gelişim göstermişlerdir. Temel anlamda çalışma mantığında %100 kesinlik verme garantisi olmayan bu algoritmalarda her zaman aynı performansta çalışmaz veya her zaman bir sonuç vermeyi ya da sonuçlarında kesinlik sağlamayı garanti etmez fakat problemleri optimize etmek için genel anlamda kullanışlıdırlar. Büyük çaplı ve karmaşık problemlerin kesin çözüm yöntemi kullanılarak çözümü çok uzun sürmektedir. Meta sezgisel çözüm yöntemleri ile bu problemlerin çözümü daha kısa sürmektedir. Bundan dolayı büyük ve karmaşık problemlerin çözümü için en kullanışlı yoldur. En iyi çözüme yakın sonucu kısa sürede verebildikleri için de yaygın olarak kullanılmaktadırlar [44] [36] [45]. Sezgisel algoritmalarda kullanılan yöntemlerde olması gereken bazı özellikler vardır. En uygun veya en uygun çözüme yakın çözümler sağlayabilmeli, kısa sürede sonuç verebilmeli, anlaması kolay olmalı, mevcut probleme benzer problemlere uyarlanabilir olmalıdır. Sezgisel algoritmalar her zaman kesin sonuç vermeseler bile çözüme ulaşma süresi makuldür. Ulaşılan sonucun doğruluğunun ispatlanabilir oluşu önemsenmemektedir.

5.3.1 Tabu Arama

1989 Glover ve Laguna tarafından geliştirilmiştir. En iyi sonuca çok yakın çözümü kısa sürede bulabilmesi sebebi ile tabu arama algoritması en çok kullanılan yöntemlerdendir [44] [40].

Yasaklı liste tabu aramanın en önemli özelliğidir. Bu liste sürekli olarak güncellenir ve algoritmanın tekrar etmemesini sağlar. Ancak bulanan çözüm o ana kadar bulanan en iyi çözümden daha iyiyse, yasaklı listede de olsa çözüm kabul edilir ve çözüme devam edilir. Genelde çözümler önce kötüleşir, sonra iyileşmeye başlar.

Bu algoritmada çözüme giden adımların son adımı olan kısımda çözümün dairesel hareketler yapmasını tekrar etmesini engellemesi veya cezalandırmasıdır. Bu şekilde engellenen tekrarlamalar sayesinde yeni çözüm yollarına teşvik eder ve sonrasındaki yolların araştırılabilir olmasını sağlamak amacıyla kısmi araştırmaya olanak sağlamaktadır.

Böylece her bir tekrarda aday seçenekler değerlendirilir, mevcut problemden bir sonraki probleme doğru adım adım ilerleme sağlanır. Engellemeler veya cezalandırmalar yapılırken bazı adımların, hareketlerin tabu olarak nitelendirilmesi, seçimlerin tekrar etmesini engellemeye yönelik atılmış adımlar olacaktır. Bazı durumlarda bu engellemelere rağmen yeni adımlarda seçilen elemanlar tabu olarak nitelendirilen, tekrarlanmaması gereken adımlardan seçilebilmektedir. Tabu arama algoritmasının kullanıldığı en temel iki problem, komşuluk arama ve yakın zamanda olması muhtemel kısa süreli hafızadır. [46].

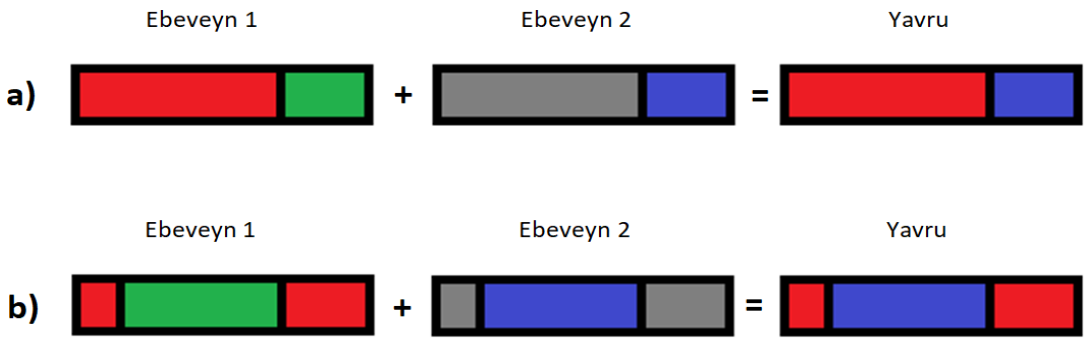
5.3.2 Genetik Algoritma

Genetik algoritma problemlerinin çözümünde doğal oluşan süreçlerin bilgisayar yardımı ile taklit edilmesidir. 1975 yılında ilk kez Holland tarafından problem çözme paradigması önerilmiştir.

İlk olarak diğer sezgisel yöntemlerden elde edilen çözüm ile başlanır ve buna başlangıç popülâsyonu denir. Çözümün her bir parametresi bir gen olarak ifade edilir, çözümün tamamı ise kromozom olarak kabul edilir. İlk kromozomlar oluşturulduktan sonra, mevcut iki kromozomun çaprazlanmasıyla yöntemi veya kromozomun değiştirilmesi (mutasyon) ile yeni kromozomlar oluşturulur. Kromozomların değiştirilmesi ile genlerin değerlerinde ufak değişiklikler yapılır. Böylelikle lokal minimumlardan azalması amaçlanır. Tüm kromozomlar uygunluk fonksiyonundan geçerek fonksiyonuna en uzak kromozomlar bulunur ve eklenir. En iyi kromozom yani en uygun çözüme ulaşılır.

GA' da optimize edilecek olan sorunlara ait çözüm yollarının her biri ayrı kromozom biçiminde temsil edilir. Kromozomlar sorun yapısına bağlı olacak biçimde nitelendirilir. Kromozomların hangi şekilde nitelendirileceğinin belirlenmesinden sonra, sorunun şekline göre en uygun çaprazlama şekli belirlenir. Çaprazlama işi ise seçimi yapılan iki ayrı ana kromozomun birbirleriyle gen alışverişinde bulunması

sonucu yeni bir kromozomun meydana getirilmesi olarak tanımlanabilir. Şekil 9.1 ' deki görüldüğü gibi Genetik algoritmada tek noktalı ve çift noktalı örnekleri verilmiştir.



Şekil 9.1: Genetik algoritma çaprazlama çeşitleri a) Tek noktalı b) Çift Noktalı

Genetik çeşitliliği artırmak amacıyla çaprazlama işleminden elde edilen yeni yavru bireylere mutasyon işlemi uygulanmaktadır. Mutasyon işlemi, mevcut kromozomun gelişigüzel seçimi yapılan genlerinin yapısının değiştirilmesi ya da gezgin satıcı probleminde olduğu gibi bütünleştirmeci sorunlarda, genlerin kendi aralarında yer değiştirmesi biçiminde uygulanabilmektedir [47].

5.3.3 Benzetilmiş Tavlama

1983 yılında Kirkpatrick ve diğerleri tarafından geliştirilmiştir. Yöntemin amacı, pozitif yöndeki ilerlemenin yerine negatifin seçilmesi ihtimalinin tekrarlanarak seçilme ihtimalinin azalmasıdır. Böylelikle ilk başlarda pozitif bölgelerde sıçramalar olurken en iyi sonuca yakın çözüme ulaşılırken olasılık değeri sıfıra yaklaşır ve çözüm bölgesi daralır [44].

Adını demirin yüksek derecede ısıtılması veya demirin tavlanmasından alan bu algoritmanın amacı, genel anlamda genel optimizasyon elde etmektir. Doğada katı halde bulunan bir maddenin, önce ısıtılarak sıvı hale gelmesi ve ardından belli bir seviyede soğutularak tekrar eski haline dönmesiyle meydana gelen maddenin yeni yapısı bir sistemdeki parçacık olarak yansıtılırsa, bu tavlama işleminden benzetilmiş tavlama yöntemi elde edilmiş olur [46].

Isıtmadan sonra yapılan soğutma yöntemiyle yeni komşuların bulunması, algoritmadaki önemli adımlardan biridir. Doğada bulunan katı bir maddeyi ısıttığımızda meydana gelen kinetik enerji artıyor, gaz bir maddeyi soğuttuğumuzda ise potansiyel enerji azalarak minimum noktaya geldiğinde kristalleşme oluyor. Benzetilmiş tavlamada bu yapıya benzer olarak kendi ürettiğimiz bir fonksiyonu, yapıyı minimize etmeye çalışıyoruz [46].

Bu algoritma genellikle, seyahat problemleri, yol bulma problemleri, görüntü işleme ve elektronik devre tasarımları gibi birçok problemin çözümünde kullanılmaktadır.

5.3.4 Karınca Kolonisi

1991 yılında Dorigo ve diğerleri tarafından gezgin satıcı probleminin çözümü için geliştirilmiştir. Doğadaki karıncaların hareketlerinden faydalanılarak matematiksel modele dönüştürmüştür. Karınca kolonisi 6 bölümde detaylı olarak anlatılacağından bu kısımda detaya girilmemiştir.

5.3.5 Yapay Arı Kolonisi

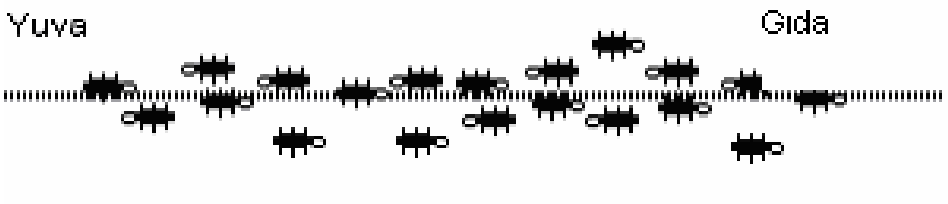
2005 yılında Karaboğa tarafından doğada bulunan arıların yiyecek arama şekillerinden faydalanılarak geliştirilmiş bir algoritmadır. Algoritma görevli ve görevsiz işçi arılar, yiyecekler ile negatif ve pozitif geri bildirimlerinden meydana gelmektedir. Görevli işçi arılar, belirli kaynaklardan getirdikleri besinlerin yer ve kalite bilgilerini diğer arılara bildirirler. Görevsiz işçi arılar, kâşif ve gözcü olmak üzere iki ayrılır ve yeni besin kaynaklarını ararlar. Kovandaki tüm arıların %5- 10'u kâşif arılardan oluşur [44].

5.3.6 Parçacık Sürü Optimizasyonu

1995 yılında Kennedy ve Eberhart tarafından kuş sürülerinin davranışlarından yararlanılarak geliştirilmiştir. Kuşlar besin ararken etkileşim içerisinde olurlar ve en yakın kuşun peşinden giderler. Bu etkileşimler muhtemel çözüm parçacıkları olarak belirlenir ve o anki en iyi parçacığı izleyerek çözüm bulmaya çalışırlar. Algoritmada kullanılan parametrenin az olması nedeniyle uygulanması kolaydır [48].

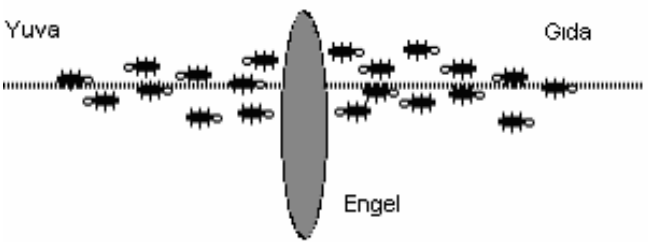
6. KARINCA KOLONİ OPTİMİZASYONU

Karınca kolonisi algoritması 1996 yılında Dorigo ve arkadaşları tarafında kuadratik arama ve gezgin satıcı probleminin çözümü için geliştirilmiştir. Doğada karıncalar besin ihtiyaçlarını sağlamak üzere çevreyi tarar ve bu taramada görme duyularını kullanmadan gerçekleştirirler. Çevredeki değişimlere çabuk alışma özellikleri bulunmaktadır. Besin aramak için yuvalarından çıkarlar ve rastsal olarak bir yol takip ederler. Şekil 10.1 'deki karıncaların yiyecek bulma amacıyla çıktıkları yoldaki hareketlerini inceleyelim.



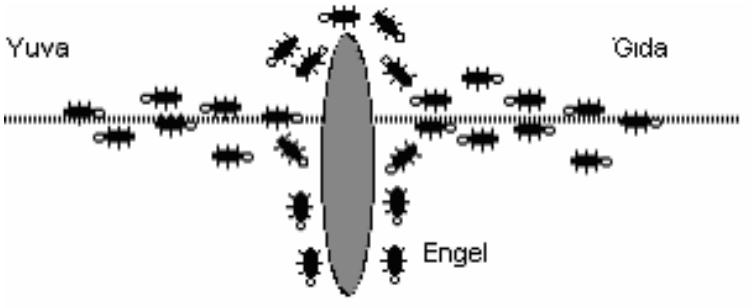
Şekil 10.1: Yuvadan çıkıp besine doğru aldıkları yol

Şekil 10.1 'de karıncalar yuvalarından çıkıp herhangi bir engelle karşılaşmadan ileride duran besine ulaşıyor ve yine bu yoldan geri dönüyorlar. Şekil 11.1 'deki görüldüğü gibi önlerine bir engel konulduğunda önlerindeki yolu daha zor bir hale getirelim.



Şekil 11.1: Önlerine engel konulan karıncalar

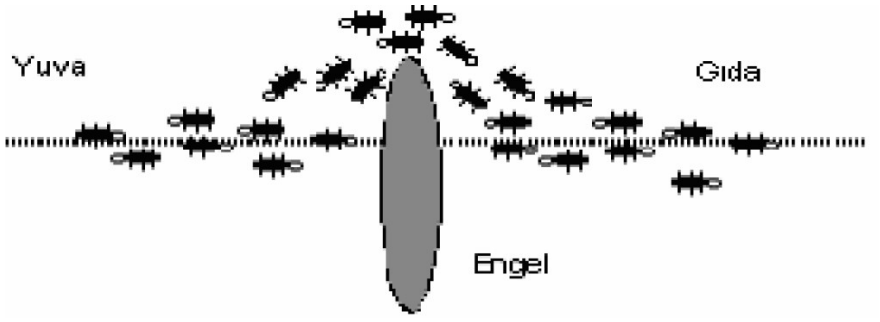
Engelle karşılaşan karıncalar önceden tek bir yoldan besine gidebiliyorken engelle karşılaştıkları için artık iki ayrı yolla karşı karşıya kalmışlardır. Bu iki farklı yol için seçilim ihtimalleri eşittir. Bu seçilim karıncalar tarafından rassal olarak yapılmaktadır.



Şekil 12.1: Engelle karşılaşan karıncaların bir sonraki yol seçim durumları

Şekil 12.1 'de görüldüğü gibi neredeyse eşit olarak iki yoldan giden karıncalar, salgıladıkları feromon sıvısı sayesinde hangi yolun kısa hangi yolun zor olduğu kanısına varır ve bir sonraki besine gitme yollarını bu sıvının yaydığı kokunun yoğunluğuna bağlı olarak seçmektedirler.

Şekil 13.1 ' de görüldüğü gibi bir süre sonra feromon sıvısının yoğun olduğu yoldaki karıncalarda artma meydana gelecek ve bu doğrultuda feromon sıvısının az olduğu yoldaki karınca sayısında zamanla önce azalma sonra tamamen bitme durumu yaşanacaktır.



Şekil 13.1: Feromon sıvısına bağlı olarak seçilen yolun son durumu

Yürüdükleri yolda feromon adı verilen bir maddeyi salgılarlar. Feromon kokulu bir sıvıdır. Belli bir miktar koku yayar ve bir süre sonra bu madde buharlaşır. Feromon,

kendilerinden sonra gelen karıncalar için yol gösterici olur. Karıncaların önlerine bir engel konulduğunda, feromonu takip edemedikleri için rastsal olarak bir yol seçerler. Kısa yoldaki birim zamandaki geçiş diğer yola göre daha fazla olacağından, bu yoldaki feromon kokusu daha yoğundur. Besine ulaştıktan sonra en çok kullanılan yoldan yani feromon salgısının en fazla olduğu yoldan yuvalarına geri dönerler. Karınca kolonisi sistemindeki bu karıncalar normal karıncalardan farklıdır. Farklı yönleri bir hafızalarının olması ve kısmen görme yeteneklerin olmasıdır [49].

6.1 Karınca Kolonisi Eniyilemesi

KKO, doğal karınca topluluklarının doğadaki yiyecek arama davranışlarından esinlenerek optimizasyon sorunlarına çözüm bulmak amacıyla üretilmiştir [50]. Önceki senelerde KKO ile birçok farklı alanda yapılan çalışmaların bir kısmı Çizelge 1.1 ’de gösterilmiştir.

Çizelge 1.1: KKO uygulaması alanında yapılan çalışmalar

Çizelgeleme Problemi	Kılıç ve Kahraman	MMAS for FPFSP	2007
	Kılıç ve Kahraman	MMAS	2006
	Kılıç ve Kahraman	Fuzzy ACO	2006
	Blum	Beam-ACO	2005
	Ying ve Liao	ACS	2004
	Shyu ve ark.	ACO	2004
	Kılıç ve Kalyan	AS-Schedule	2003
	T'kindt ve ark.	ACO	2002
	Gravel ve ark.	ACO	2002
	McMullen	ACO	2001
	Stützle	AS-FSP	1998
	Colorni ve ark.	AS-JSP	1994
Kümeleme	Yang ve Kamel	Multi-ant colonies	2006
	Kuo ve ark.	Ant k_means	2005
	Shelokar ve ark.	ACO	2004
Sınıflandırma	Shelokar ve ark.	ACO classifier system	2004
Kısıt Tatmini	Solnon	Ant-P-solver	2000
Ardışık Sıralama	Gambardella ve Dorigo	HAS-SOP	2000
Şebeke Rotalama	Di Caro ve Dorigo	AntNet	1998
	Schoonderwoerd ve ark.	ABC	1996
Ağ Boyama	Costa Hertz	ANTCOL	1997
Çizelgeleme Problemi	Kılıç ve Kahraman	MMAS for FPFSP	2007
	Kılıç ve Kahraman	MMAS	2006
	Kılıç ve Kahraman	Fuzzy ACO	2006
	Blum	Beam-ACO	2005
	Ying ve Liao	ACS	2004
	Shyu ve ark.	ACO	2004
	Kılıç ve Kalyan	AS-Schedule	2003
	T'kindt ve ark.	ACO	2002
	Gravel ve ark.	ACO	2002
	McMullen	ACO	2001
	Stützle	AS-FSP	1998
	Colorni ve ark.	AS-JSP	1994
Kümeleme	Yang ve Kamel	Multi-ant colonies	2006
	Kuo ve ark.	Ant k_means	2005
	Shelokar ve ark.	ACO	2004
Sınıflandırma	Shelokar ve ark.	ACO classifier system	2004
Kısıt Tatmini	Solnon	Ant-P-solver	2000
Ardışık Sıralama	Gambardella ve Dorigo	HAS-SOP	2000
Şebeke Rotalama	Di Caro ve Dorigo	AntNet	1998
	Schoonderwoerd ve ark.	ABC	1996
Ağ Boyama	Costa Hertz	ANTCOL	1997

6.2 Parametrelerin Belirlenmesi

Karınca kolonisi optimizasyonun ilk adımı parametrelerin belirlenmesi aşamasıdır. Çıktıların doğruluğu, mevcut probleme uygun değerler verilmesiyle doğrudan ilişkilidir. Karınca kolonisi optimizasyonunda kullanılan parametreler aşağıdaki gibidir:

α (alpha): Mevcut problem için kullanılacak feromon izi miktarı ile alakalıdır. Feromon izi seviyesinin ne kadar kullanılacağıyla ilgilidir.

β(beta): Mevcut problem için sezgiselliğin ne derece kullanılacağıyla alakalıdır.

ρ (rho): Mevcut problem için feromonun buharlaşma oranıyla alakalıdır.

m: Karınca Kolonisi optimizasyonunda, kullanılan algoritma için ihtiyaç duyulan karınca sayısı ile alakalıdır.

Sonlandırma koşulu (İterasyon Sayısı): Mevcut problem için uygulanacak

iterasyon sayısı ile alakalıdır.

6.3 İlk kullanılacak Feromon Miktarı

Problemin çözümüne başlarken, algoritmada ilk olarak kullanılacak feromon miktarı sıfıra yakın bir değer olarak seçilir. Karıncalar belli bir süre sonra hareket edeceklerinden, feromon miktarı, kullanılan yollara bağlı olarak artış gösterecektir.

6.4 Çözümlerin Oluşturulması

Kullanılan karıncalar, başladıkları noktadan diğer noktalara doğru hareket ederler. Başlangıç noktasından sonra sezgisel olarak seçenekler arasından birini tercih ederler. Zamanla çözüme ulaşma çabasında olan karıncalar, feromon izlerini takip ederler ve sezgilerinden yararlanırlar. En çok seçilen yoldaki feromon miktarı az seçilen yola göre daha fazla olacağından, karıncalar feromon miktarının fazla olduğu yoldan ilerleyeceklerdir.

6.5 Yerel Arama

Çözüm yolları oluşturulduktan sonra, mevcut çözümleri daha da iyileştirmek amacıyla kullanılır. Belli bir alanda yoğunlaşan karıncalar, yaptıkları işi tekrar ederek

çözümlerin en iyisini belirlerler. Yerel arama her problemde kullanılmaz. Uygulanmak istenen probleme uygun olduğu durumlarda kullanılır ve isteğe bağlıdır [51].

6.6 Feromon İzi Güncelleme

Karıncalar yürürken arkalarından yola salgıladıkları feromonun takibi sayesinde en iyi çözüme ulaşmayı hedeflemektedirler. Feromon izi güncellemesi iki aşamada yapılmaktadır. Birinci aşamada feromon sıvısı belirlenen bir oranda buharlaşmaktadır. Buna bağlı olarak buharlaşma feromon miktarını azaltmaktadır. İkinci aşamada ise çok seçilen yollara ait çözümlerdeki feromon miktarı arttırılarak başarılı sonuçlara ait feromon miktarlarının daha da arttırılarak seçilme oranının arttırılması hedeflenmektedir.

6.7 Karınca Kolonisi Optimizasyonu Algoritmaları

Literatürde bugüne kadar birçok karınca kolonisi optimizasyonu algoritması kullanılmıştır. İlk olarak kullanılanı Dorigo tarafından 1996 'da Karınca Sistemi adıyla, Gezgin Satıcı Problemi için kullanılmıştır [52].

Sonrasında bu algoritmanın, Elitist Karınca Sistemi, Karınca Kolonisi Sistemi, Sıra Tabanlı Karınca Sistemi, En iyi En kötü Karınca Sistemi, Max-Min Karınca Sistemi adlarındaki algoritmaları türetilmiştir. Bu tez çalışmasında Karınca Kolonisi Optimizasyonu için GSP örneği kullanılmıştır.

6.8 Karınca Sistemi

Karınca Kolonisi Optimizasyonu algoritmalarından olan Karınca Sistemi literatürde ilk olma özelliği taşımaktadır. Problemin çözümü aşamalarına geçildiğinde m sayıdaki karıncanın oluşturduğu çözüm yollarına göre feromon miktarları güncellenmektedir.

(1) $$\tau_{ij} \leftarrow (1-\rho) . \tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}{}^{k}$$

Karınca Sisteminde feromon miktarı formül 1'e göre güncellenmektedir. Formüldeki Tij i ve j noktaları arasında bulunan yoldaki feromon miktarını göstermektedir. Formüldeki m kullanılan karınca sayısını, ρ feromon miktarının buharlaşma katsayısını göstermektedir. ΔTijk ise k. Karıncanın i ve j noktaları arasındaki yola bırakılan feromon sıvısı seviyesini göstermektedir. Feromon miktarının buharlaşma kat sayısı 1'e yaklaştıkça daha önceden salgılanan feromon izi tamamen yok olmaktadır. Durum tam tersi olan 0'a yaklaştığında ise feromon izi korunmaktadır. Feromon izlerinin 1 değerine yaklaşması durumunda karıncaların önceki denemelerinden sağlanan bilgiler silinecek ve istenilen sonucun doğruluğu negatif olarak etkilenecektir. Feromon izlerinin 0 değerine yaklaşması durumunda ise karıncaların önceki denemelerinden sağlanan bilgiler tekrarlanacak ve karıncalar sürekli aynı yolu tercih edeceklerinden, yeni yolların denenmesi zorlaşacaktır.

(2) $$\Delta\tau_{ij}{}^{k} = \begin{cases} \frac{Q}{L_k}, & eğer\ k.karınca\ (i,j) kenarını\ turunda\ kullandıysa \\ 0, & kullanmadıysa \end{cases}$$

ΔTijk ise formül 2' ye göre hesaplanmaktadır. Bu formülde Lk kaçıncı karıncanın çözümde oluşturduğu tur mesafesini ifade etmektedir. Formüldeki bir diğer Q parametresi ise formülde sabittir. Kaçıncı karıncanın kullandığı yola bırakılan feromon seviyesi yapılan yol mesafesi ile ters orantılı olarak değişmektedir.

Karıncaların bulundukları noktadan bir diğer noktaya giderken yapacakları seçimi rassal olarak yaparlar. Feromon miktarının güncellenmesinden önce karıncalar gidecekleri yolu seçer ve bulundukları noktadan hangi farklı noktaya gideceklerini belirlerler.

(3) $$p_{ij}^{k} = \begin{cases} \frac{\tau_{ij}^{\alpha}\cdot\eta_{ij}^{\beta}}{\sum_{l\,\epsilon\,\Omega}\tau_{il}^{\alpha}\cdot\eta_{il}^{\beta}}, & eğer\ j\ \in\ \Omega\ ise \\ 0, & değilse \end{cases}$$

Kaçıncı karıncanın i noktasından çıkıp j noktasını seçme olasılığı formül 3'e göre hesaplanmaktadır. Formüldeki pikj i noktasından sonra seçilen noktanın j olma ihtimalini belirtmektedir. Ω ise i noktasından çıktıktan sonra seçilebilecek en yakın

listesini ifade etmektedir. Formüldeki Tij i ve j noktaları arasında bulunan yoldaki feromon miktarını göstermektedir. η i j ise i noktası ve j noktaları arasındaki sezgisel seçimi göstermektedir. Ω listesi, daha önceden uğranılmamış komşu noktalara ait listeyi saklamaktadır. Bu listeyi oluşturmadaki amaç, Gezgin Satıcı Problemi'nde uğranılacak noktalara gidiş ve dönüşte yalnızca bir defa uğranılacak olması kuralı gereği yapılmaktadır. α 'nın sıfıra yaklaşması durumunda, bir sonraki seçilecek komşu noktanın hangisi olacağı seçimi yapılırken önceki denemelerden elde edilen sonuçların bir önemi kalmayacaktır. Bu durumda komşu seçimleri sezgisel olarak yapılacaktır. Seçimin yalnızca sezgisel olarak yapılması durumunda açgözlü algoritmadaki seçim şekline benzeyecektir. β'nın sıfıra yaklaşması durumunda ise α 'da ki durumun tersi bir işleyiş gerçekleşecek ve karıncalar daha önceden yapılan denemelerden faydalanarak yol seçimlerini yapacaklardır. Çözümün kalitesi bu durumunda negatif olarak etkilenecektir.

(4) $$\eta_{ij} = \frac{1}{d_{ij}}$$

η ise formül 4' e göre hesaplanmaktadır. Formüldeki dij i noktasından çıkıp j noktasına giderken ki kat edilen yolun mesafesini ifade etmektedir. Yakın mesafede bulunan komşu noktalar, uzak mesafede bulunan noktalara göre tercih edilebilmesi daha yüksek seviyede olmaktadır. Şekil 14.1' de karınca sistemi algoritmasının 5 adımda işleyişi gösterilmiştir.

1. Adım: Gerekli parametreler için başlangıç değerleri belirlenir.

2. Adım: Bütün karıncaları şehirlere rastgele olarak yerleştirilir.

3. Adım: Karıncaları bulundukları şehirden olasılık formülüne göre başka bir şehre hareket etmesi için seç.

4. Adım: Her karınca turunu tamamladıktan sonra yolların uzunluğu hesaplanır ve en iyi değere göre feromen güncellemesi yapılır.

5. Adım: İterasyon sayısına veya başka bir kriteri sağlayıncaya kadar tabu listelerini boşalt ve 2. Adıma geri dön

Şekil 14.1: Karınca sistemi algoritmasının adımları

6.9 Elitist Karınca Sistemi

Bu algoritmadaki elitist terimi, genetik algoritmalarda var olan elitist stratejilerinden gelmektedir. EKS, Karınca sistemi algoritmasına nazaran, daha önceki denemelerden elde edilen sonuçların en iyi olan yola, feromon güncelleme aşamasında bir miktar daha feromon sıvısı eklenmesi mantığıyla çalışır. Eklenen bu feromon sıvısının miktarı e. Q/L* formülüyle hesaplanarak eklenir. Formülde bulunan e simgesi elitist karınca sayısını göstermektedir.

Formüldeki L* parametresi ise o zamana kadar denenmiş en iyi yolun mesafesini ifade etmektedir. Formüldeki bir diğer Q parametresi ise formülde sabittir.

Mevcut problemle ilgili yapılan çalışmalarda, elitist karınca sayısının miktarı büyük bir önem arz etmektedir. Elitist karınca sayısının olması gerekenden az olduğu durumlarda eğer karınca sayısı arttırılırsa, yerel en iyi bölgeler çok daha çabuk tespit edilir. Elitist karınca sayısının olması gerekenden daha fazla olduğu durumda ise yerel en iyi noktalarda elitist karıncaların daha fazla yoğunlaşması sebebiyle bu algoritmanın çözüm üretme kalitesinin negatif etkilendiği belirtilmektedir [52].

6.10 Karınca Kolonisi Sistemi

Karınca kolonisi sistemi algoritması, karınca sistemi algoritmasından bazı konularda farklılık göstermektedir. Feromon güncellemesi aşamasında görülen bu farklılaşmalardan birincisi, yerel feromon güncellemesi yapılırken, gidilecek yol tamamlanmadan önce i noktasından j noktasına gidilirken bu güzergâhta ki feromon

miktarını formül 5 ’e göre güncelliyor olmasıdır.

(5) $$\tau_{ij} = (1 - \varphi).\tau_{ij} + \varphi . \tau_0$$

Formülde kullanılan φ simgesi, feromon sıvısının azaltılma katsayısının kaç olacağını ifade etmektedir. Formüldeki Tij i ve j noktaları arasında bulunan yoldaki feromon miktarını göstermektedir. T0, problemin çözümüne başlarken ki ilk feromon miktarını göstermektedir. Formüldeki φ’e verilen değer 1 olduğu durumda, feromon seviyesi başlangıç değerine dönecektir. İlk değerine dönmesinin bir sonucu olarak feromon sıvısı değerinin 0 ’a inmemesi sağlanacaktır. Bu algoritmada kullanılan karıncaların kullandıkları güzergâhlar da mevcut feromon sıvısını azaltma istekleri, aslında gidilebilecek yol alternatiflerini arttırmaya yöneliktir. Karıncaların bu davranışları mevcut çözüm denemesinde daha önceden farklı karıncaların kullandığı güzergâhları daha az tercih etme eğilimleriyle açıklanmaktadır.

Karınca kolonisi sisteminin Karınca sisteminden diğer bir farkı, feromon güncellemesini yapılan denemeden sonra gerçekleştiriyor olmasıdır. Karınca sistemi algoritmasında feromon güncellemesi yapılırken, sistemdeki bütün karıncalarla güncelleme işlemi yapılmaktadır. Karınca kolonisi sisteminde ise farklı bir durum söz konusudur. Problemin çözümünde kullanılan karıncaların tümü değil, yalnızca en kısa mesafe kat eden karıncalar güncelleme işlemi yapmaktadır.

(6) $$\tau_{ij} \leftarrow \begin{cases} (1 - \rho).\tau_{ij} + \rho . \Delta\tau_{ij}, & eğer\ (i,j) eniyi\ tura\ aitse \\ \tau_{ij}, & değilse \end{cases}$$

Karınca kolonisi sistemi algoritmasındaki feromon sıvısı güncellemesi formül 6 ’ya göre yapılmaktadır. Formülde bulunan ΔTij = 1 / Len iyi olarak hesap edilmektedir. Len iyi, problemin çözümünde her bir denemedeki veya problemin tümündeki denemelerde alınan en iyi mesafe olarak kullanılmaktadır.

Karınca kolonisi sisteminin Karınca sisteminden diğer bir farkına, çözümlere gidildiği aşamada karşılaşırız. Aynı karınca sisteminde olduğu gibi i noktasından j noktasına uğrama ihtimali aynı formülle hesaplanır. Bu şekilde hesaplama

yapılmadan önce “pseudo random proportional” kuralı devreye girer. Uygulanacak kurala göre [0,1] aralığı için rastsal bir q sayısı seçilir. Seçilen bu sayı önceden seçilmiş q0 = [0,1] değerinden küçük veya eşit ise (q ≤ q0) ihtimal dâhilinde bir tercih yapılmaz. Bu durumda mevcut komşulardan en yüksek Tij. ηβij değerine sahip olan seçilir. Şayet q > q0 ise, Karınca Sistemi’nde kullanıldığı gibi bu algoritma içinde aynı olasılık sal nokta belirleme formülü devreye girer.

6.11 Sıra Tabanlı Karınca Sistemi

Bu algoritmada, Elitist karınca sisteminde kullanılan yöntemin dışında, çözüm için yapılan denemeler sonrasında, m sayıda karıncaya ait güzergâhlar, en iyiden en kötüye doğru listelenir. Yapılan bu sıralamanın ardından bütün karıncaların feromon sıvıları güncellenmez, yalnızca w sayıda karınca güncelleme işlemine tabi tutulur. Yapılan bu güncellemeyle bütün karıncalar için aynı oranda güncellenme yapılmaz. Formül 7, 8, 9 ve 10’da ki kurala göre güncelleme işlemleri yapılmaktadır [55].

(7) $$\tau_{ij} = \rho \,.\, \tau_{ij} + \Delta\tau_{ij} + \Delta\tau^{*}_{ij}$$

(8) $$\Delta\tau_{ij} = \sum_{\eta=1}^{\sigma-1} \Delta\tau^{\mu}_{ij}$$

(9) $$\Delta\tau^{\mu}_{ij} = \begin{cases} (\sigma - \mu)\frac{Q}{L_{\mu}}, & eğer\ \mu.eniyi\ karınca\ (i,j) kenarından\ geçtiyse \\ 0, & geçmediyse \end{cases}$$

(10) $$\Delta\tau^{*}_{ij} = \begin{cases} \sigma\ \frac{Q}{L^{*}}, & eğer\ (i,j) kenarı\ eniyi\ turun\ bir\ parçası\ ise \\ 0, & değilse \end{cases}$$

Formüldeki μ, sırlama indeksini ifade eder. ΔTμij, bu çözüme ait sıralamadaki μ. karıncanın i ve j noktaları arasına salgıladıkları feromon seviyesini belirtir. Lμ ise bu çözüme ait sıralamadaki μ. karıncanın gittiği yolun mesafesini göstermektedir. ΔT*ij, bu çözüme ait sıralamadaki elitist karıncaların i ve j noktaları arasına salgıladıkları

feromon seviyesini belirtir. Formüldeki σ ise problemde kullanılan elitist karınca sayısını göstermektedir. Son olarak L* ise elde edilen en kısa yol mesafesini ifade etmektedir.

Elitist karınca sistemindeki feromon güncelleme şeklini kullanıp ayrıca bu duruma ek olarak en kısa yoldan en uzun yola kadar sıraya dizilen w sayıda karıncanın belirli miktarlarda yaptıkları yollara feromon güncellemesi yoluyla feromon sıvısı eklemeleri sağlanmıştır. Bu algoritmada elitist karınca sistemine göre iyileştirme yapılmış, en kısa yollardan daha fazla istifade edilirken, gidilebilecek alternatif güzergâhları ortaya çıkarma işlemi daha etkin bir şekilde çalışmaktadır.

6.12 En İyi En Kötü Karınca Sistemi

Bu algoritmada karıncaların i noktasından j noktasına giderken yaptıkları seçimi Karınca sistemindeki yöntemin aynısını uygulayarak (olasılıksal seçim yöntemi) yaparlar. En iyi – en kötü karınca sisteminde feromon güncelleme işlemleri ise 3 farklı yöntemle yapılmaktadır [53].

Bu yöntemlerden birincisi, mevcut problem için yapılan çözüm iterasyonları sonucu ortaya çıkan sonuçlarda, en iyi ve en kötü çözümler birbirinden ayrılır. Ayrılan bu çözümlerde iyi olan turlar için feromon sıvısı seviyesi arttırılır. Kötü olan turlar için ise feromon sıvısı seviyesi düşürülür. Bir denemede bulunan bir yolun hem en iyi hem en kötü tur içinde yer alması mümkündür. Böyle bir durumda mevcut güzergâh en iyi çözümler arasında gösterilerek feromon sıvısı seviyesi arttırılır. Bu güzergâh için daha sonra feromon sıvısı azaltma işlemi yapılmamaktadır.

Çözüm arama aşamasında durağanlık durumuna rastlandığında arama süreci tekrarlanır. Bu işlem yapılırken feromon seviyeleri hepsi için ilk durumdaki haline döndürülür. Çözümde durağanlık, en iyi yol için hesaplanan feromon seviyesinin normalden fazla artması ve kalan yolların feromon sıvısı seviyesinin sıfıra doğru yönlenmesi durumunda tespit edilir ve arama süreci tekrarlanır.

Bir diğer yöntem ise aşağıda bulunan formül 11 ve 12 ’de verildiği gibi, başlangıçta az daha sonraki aşamalarda ise artarak çok miktarda mutasyon işlemi uygulanmaktadır.

(11)
$$\tau'_{rs} = \begin{cases} \tau_{rs+mut(it,\tau_{threshold})}, & eğer\ a = 0 \\ \tau_{rs-mut(it,\tau_{threshold})}, & eğer\ a = 1 \end{cases}$$

(12) $$\tau_{threshold} = \frac{\sum_{(R,S)\in S_{en\ iyi}} \tau_{rs}}{|S_{en\ iyi}|}$$

Formülde verilen a = {0,1},bu aralıktaki rastgele verilmiş bir sayıyı ifade etmektedir. Formüldeki it, anlık deneme numarasını Tthreshould ise en iyi yol için ortalama feromon sıvısı seviyesini ifade etmektedir. Mut(.) parametresi formül 13'teki şekilde belirtilmiştir.

(13) $$mut(it, \tau_{threshold}) = \frac{it - it_r}{Nit - it_r} . \sigma . \tau_{threshold}$$

Formülde verilen Nit, algoritmanın çözümü için yapılan deneme miktarını göstermektedir. $İt_r$ ise baştan başlatılan aramanın en sondaki deneme sayısını belirtmektedir. Gerçekleşen mutasyon miktarını σ parametresi belirtir. Bir örnekle açıklamak gerekirse, σ=4 eşitliğine göre, en son yeniden başlatma işleminden sonra en az %25'i denenen iterasyonlar için mutasyon seviyesi Tthreshold seviyesine eşit olacaktır.

Çözüm için algoritmanın yeniden başlatıldığı zamanlarda dikkat edilmesi gereken durum, mutasyon seviyesinin sıfıra sabitlenerek başlangıç şekline geri dönmektedir. Diğer algoritmalardan farklı olarak en iyi yollarda bulunan feromon sıvısı seviyesi daha çok arttırılır. Önceki bilinenlerden faydalanılarak mutasyonda kullanılarak yeni yolların keşfi amaçlanmıştır. Bu şekilde dengeli bir şekilde algoritmanın sonuç üretmesi beklenir.

6.13 Max Min Karınca Sistemi

Bu sistemde, feromon sıvısı güncelleme işlemi Karınca kolonisi sisteminde kullanılan feromon sıvısı güncelleme işlemiyle, yalnızca en iyi yol için feromon güncellemesi yapılması açısından benzerlik göstermektedir. Algoritma çalışmaya başladıktan sonra ilk zamanlarda sadece o anda yapılan denemeye ait en iyi yolun

feromon sıvısı güncellemesi yapılırken, sonraki zamanlarda ise bütün denemelerdeki en iyi yola ait feromon sıvısı güncellemesi daha sık bir şekilde yapılır. Bahsi geçen en iyi yol, hem o anki deneme için yapılan yol olurken hem de bütün denemelere ait yapılan yollarda dâhil edilebilir. Aslında bu algoritma iki yaklaşımı da melez olarak kullanabilmektedir.

Bu algoritmada da yol üzerindeki feromon sıvısı miktarının bir sınırı vardır. Bu sınır Tmin alt sınır ve Tmax üst sınır olarak tanımlanmıştır. Yani bir yol üzerinde bulunan feromon sıvısı miktarı belirlenen alt-üst sınırı arasında bir yerde olmak zorundadır. Bu miktar çözüme ulaşılmaya çalışırken yapılan denemelerin en genel haline göre belirlenmektedir. Algoritmaya başlarken erken durağanlık yaşanmasının önüne geçebilmek, yeni yolların keşfini mümkün kılabilmek için, feromon sıvısı miktarı Tmax olarak belirlenir [54].

Max Min karınca sisteminde feromon sıvısının yola bıraktığı izi için düzeltme işlemi yapılır. Bu düzeltme işlemi için iki farklı formül kullanılmaktadır.

(14) $$\tau_{ij}^{*}(t) = \tau_{ij}(t) + \delta(\tau_{max}(t) - \tau_{ij}(t))$$

(15) $$0 < \delta < 1$$

Formül 14' te bulunan Tij*, yapılan feromon sıvısı düzeltme işleminden sonra i ve j noktaları arasında bulunan yoldaki feromon sıvısı miktarını göstermektedir. Tij(t), ise yapılan feromon sıvısı düzeltme işleminden önce i ve j noktaları arasında bulunan yoldaki feromon sıvısı miktarını göstermektedir. δ = 1 eşitliği olduğu durumda yol üzerinde bulunan bütün feromon sıvısı miktarı Tmax' a eşitlenir. Bu durumda feromon sıvısı güncellemesi ilklenmektedir. δ = 0 eşitliği olduğu durumda ise feromon izi düzeltme işlemi çalışmamaktadır. Feromon sıvısı izi düzenlemenin asıl görevi, bu algoritmanın durağanlık anına girmesini engellemek ve yeni noktaların keşfine katkı sunmaktır.

6.14 Dezavantajları

Sezgisel bir algoritma olması sebebiyle kesin bir sonuç vermemesi, en iyiye yakın sonucu bulmak için her bir problem için denenen iterasyon sayısı arttıkça cevap verme süresinin uzaması en önemli dezavantajları olarak sayılabilir.

6.15 Avantajları

Belli başlangıç noktasından başlayıp tüm noktalara sadece bir defa uğraması ve tekrar başlangıç noktasına dönmesi gerekmesidir. Ayrıca tüm bu noktaları dolaşırken en kısa yoldan gidip dönmesidir. Rota hesaplanırken çıkan sonucun kesin sonuç vermek zorunda olmaması ve bundan dolayı hızlı çalışmasıdır.

7. UYGULAMA

Uygulamamız Visual Studio 2017 yazılım geliştirme aracı üzerinde. NET Platformu ile birlikte C# dili kullanılarak web form uygulaması yazılmıştır. Web uygulaması olarak geliştirilmesinin en büyük sebebi kurulum gerekmemesi ve çok sayıda kullanıcının rahatlıkla erişe bilmesidir. Uygulama HTML ve JavaScript kodlarının kullanılarak Google Api servislerinden yararlanılarak geliştirilmiştir. Ayrıca rota çizme işlemini sınıf kullanarak gerçekleştirmektedir.

Ayrıca KKA' nın uygulanabilmesi için bazı parametrelere ihtiyaç duyulur. Yapılan testler sonucunda sabit parametrelere en uygun değerler atanmıştır. Bu değerler problemimizde bulunan noktalar için en uygun sonucu verecek şekilde test edilerek belirlenmiştir. Bu parametreler aşağıda verilmiştir:

Karinca_listesi: Algoritmamızda kullanılan her karınca bu listede tutulmaktadır. Her karıncanın lokasyonu ve kaç tur yapacağı bu kısımda tutulur.

Mekân_listesi: Algoritmamızda kullanılacak her bir büfe bu listeye eklenmesi gerekmektedir.

en_iyi_tur_listesi: Bu listede algoritmamızda kullanılan karıncaların yaptıkları yollar için sağlanan en kısa mesafe için elde edilen verilerin tutulduğu listedir. Karıncaların gittikleri yollar için en iyi turlar bu listede tutulur.

en_iyi_tur_uzunluk: Bu parametrenin ilk değeri -1' dir. Eğer bir tur, daha önce hesaplanan turlardan daha kısa ise bu parametre yeni değerini alır.

Mesafeler: Büfe sayısı kadar değer alır. Her bir büfe diğer noktalarla arasında bulunan mesafelerin ölçülüp atandığı değerlerdir.

Feromonlar: Seçilen büfe sayısı kadar değer alır. Başlangıçta her feromon, 1/büfe sayısı cinsinden değer alır. Algoritmanın gidişatına bağlı olarak bu değerler güncelleştirilebilir.

Mekân_sayisi: Algoritmanın uygulanacağı, seçilen toplam büfe sayısını belirtir.

Karinca_sayisi: Yapılan testler sonucunda toplam 16 adet büfe için 90 adet karıncanın yeterli olduğu görülmüştür.

Feromon_konsantresi: Feromon sıvısının yoğunluk miktarını belirtir. Feromon sıvısı güncellemesi yapılırken bu feromon yoğunluğu dikkate alınır. Uygulamamız için kullanılan feromon sıvısı yoğunluk miktarı 0,7 olarak belirlenmiştir.

Alpha: Turlar yapılırken feromon sıvısı bilgisinden yararlanılır. Karıncaların salgıladıkları bu sıvıya ait bilgiyi etkiler. Bu parametrenin 0 olduğu durumda feromon sıvısına ait bilgi kullanılmayacaktır. Kullanılmamasının sebebi, 0 olduğu durumda karıncaların tümünün aynı geçiş güzergâhını kullanacağından algoritmada durağanlığa sebep olacaktır.

Beta: Turlar yapılırken sezgisellikten de faydalanılır. Noktasal arası uzaklığın tersi olan sezgisel durum bu parametre tarafından kullanılır. Bu parametrenin 0 olması durumunda sezgisel bilgi karar alması göz ardı edilerek algoritma feromon iz bilgisine göre çalışmaktadır.

Rho: Feromon sıvısı bozunma oranını ifade eder. Belli bir süre sonra feromon sıvısı buharlaşacağı için feromon sıvısı güncellemesi bozunma oranı dikkate alınarak yapılır. Algoritmamızdaki değeri 0,7 olarak belirlenmiştir.

İterasyonlar: Algoritmamızda belirtilen iterasyon sayısı kadar fonksiyonlar tekrar edilir. Bu sayede algoritma öğrenmesi sağlanır. İterasyon sayısı ne kadar çok artarsa problemimiz için yapılan sıralama o kadar kaliteli sonuç verecektir. İterasyon sayısının çok yüksek seçilmesi, sonuca ulaşmamızı süre olarak daha geç gerçekleştirecektir.

Rastgele_sayi: Algoritma başlatıldığında, karıncalar mekânlara rastgele olarak dağıtılır. Bu rastgele dağıtım için seçilecek sayının belirlendiği fonksiyonu sağlar.

Uygulamamız üç kısımdan oluşmaktadır.

1) Uygulamanın İncelenmesi
2) Web form ara yüz çalışması
3) Google Api kullanılarak geliştirilen web form arka plan kod kısmı

7.1 Uygulamanın İncelenmesi

Örnek bir GSP problemi için İstanbul Halk Ekmek A.Ş ekmek dağıtım araçların rotalanması ele alınmıştır. Halk Ekmek Şirketinin sahip olduğu 3 fabrikaya göre büfeler gruplandırılmıştır ve fabrika bazlı hatlar oluşturulmuştur. Cebeci fabrikasına ait Cebeci ilçesi için araçların rotası incelenmiştir. Rotada bulunan büfelerin birbirine ve fabrikaya olan uzaklıkları belirlenmiştir. Fabrikada bulunan tüm araçlar 246 kasa ekmek kapasitelidir. Problem Karınca Koloni algoritmaları ile çözülmüştür. Bu maksatla oluşturulan Cebeci bölgesine ait 16 büfenin mesafeleri aşağıdaki Çizelge 2.1 'de gösterilmiştir.

Çizelge 2.1: Cebeci İlçesi Fabrika ve büfelerin mesafe matrisi

	Büfe 1	Büfe 2	Büfe 3	Büfe 4	Büfe 5	Büfe 6	Büfe 7	Büfe 8	Büfe 9	Büfe 10	Büfe 11	Büfe 12	Büfe 13	Büfe 14	Büfe 15	Büfe 16	FABRİKA
Büfe 1	0	450	437	814	432	902	829	1000	1000	2000	2000	3000	2000	2000	2000	3000	3000
Büfe 2	450	0	363	742	602	451	692	1000	1000	2000	2000	2000	2000	2000	2000	2000	4000
Büfe 3	437	363	0	378	931	1000	433	792	856	1000	1000	2000	2000	2000	2000	2000	3000
Büfe 4	814	742	378	0	712	1000	752	546	805	1000	1000	2000	2000	2000	2000	3000	2000
Büfe 5	432	602	931	712	0	712	1000	2000	2000	2000	2000	3000	3000	3000	2000	3000	4000
Büfe 6	902	451	1000	1000	712	0	643	1000	2000	2000	3000	3000	2000	2000	2000	2000	4000
Büfe 7	829	692	433	752	1000	643	0	542	528	1000	1000	2000	2000	2000	2000	3000	2000
Büfe 8	1000	1000	792	546	2000	1000	542	0	462	463	479	960	1000	2000	2000	2000	2000
Büfe 9	1000	1000	856	805	2000	2000	528	462	0	586	593	841	817	1000	1000	2000	2000
Büfe 10	2000	2000	1000	1000	2000	2000	1000	463	586	0	355	858	1000	1000	2000	2000	2000
Büfe 11	2000	2000	1000	1000	2000	3000	1000	479	593	355	0	481	667	1000	2000	1000	2000
Büfe 12	3000	2000	2000	2000	3000	3000	2000	960	841	858	481	0	406	881	2000	1000	1000
Büfe 13	2000	2000	2000	2000	3000	2000	2000	1000	817	1000	667	406	0	566	1000	3000	2000
Büfe 14	2000	2000	2000	2000	3000	2000	2000	2000	1000	1000	1000	881	566	0	763	709	2000
Büfe 15	2000	2000	2000	2000	2000	2000	2000	2000	1000	2000	2000	2000	1000	763	0	725	3000
Büfe 16	3000	2000	2000	3000	3000	2000	3000	2000	2000	2000	1000	1000	3000	709	725	0	2000
FABRİKA	3000	4000	3000	2000	4000	4000	2000	2000	2000	2000	2000	1000	2000	2000	3000	2000	0

GSP Karınca Kolonisi Algoritması ile Çözüm

Karınca kolonisi optimizasyonunda kullanılan parametreler aşağıdaki gibidir:

α (alpha): Mevcut problem için kullanılacak feromon izi miktarı ile alakalıdır. Feromon izi seviyesinin ne kadar kullanılacağıyla ilgilidir.

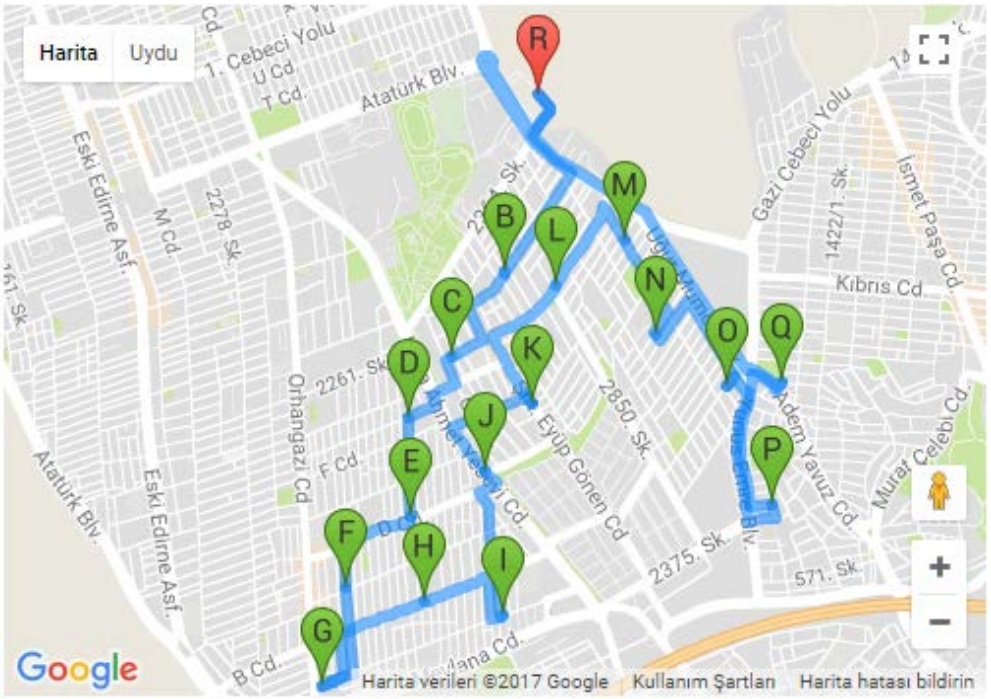
B (beta): Mevcut problem için sezgiselliğin ne derece kullanılacağıyla alakalıdır.

ρ (rho): Mevcut problem için feromonun buharlaşma oranıyla alakalıdır.

m : Karınca Kolonisi optimizasyonunda, kullanılan algoritma için ihtiyaç duyulan karınca sayısı ile alakalıdır.

Sonlandırma koşulu (İterasyon Sayısı): Mevcut problem için uygulanacak iterasyon sayısı ile alakalıdır.

Uygulamamızda bulunan 16 adet büfe için en uygun parametre değerleri alpha 1, beta 1, Rho 0,7, iterasyon sayısı 250 ve karınca sayısı 90 olarak belirlenmiştir. Uygulama Cebeci ilçesi için çalıştırıldığında oluşan en kısa yol Şekil 15.1 ’ de görünmektedir.



Şekil 15.1: Cebeci ilçesi en iyi yol gösterimi.

Uygulama en iyi rotayı oluştururken izlediği yolu adım adım inceleyelim. Uygulama fabrikadan başlayarak 16 büfeye uğrayarak tekrar fabrikaya gelinen yolun rotalanması Şekil 15.1 ‘ de gösterilmiştir. Yapılan uygulamada çizilen rotalamanın detaylı mesafe bilgisi Çizelge 3.1‘ de görülmektedir.

Çizelge 3.1: Gidilen yol mesafeleri

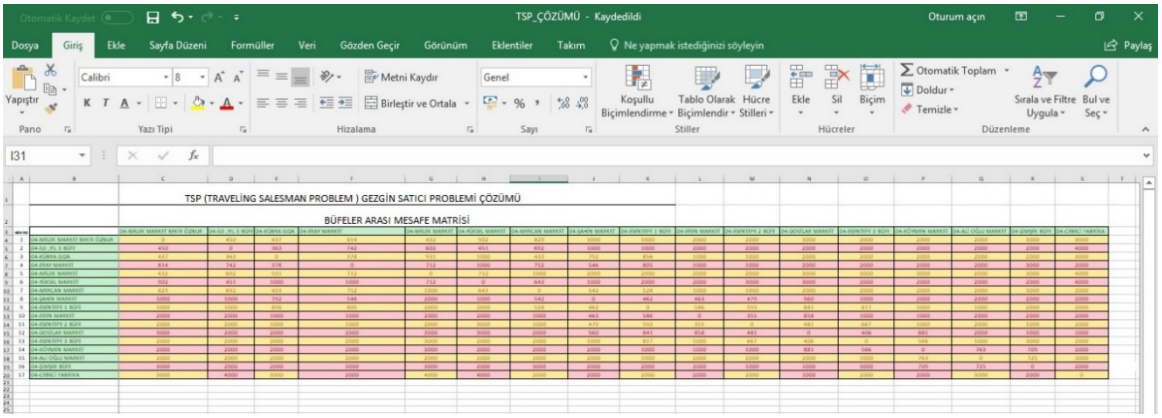
Gidilen Yer	Mesafe (metre)
FABRİKA	2000
Büfe 12	1000
Büfe 13	406
Büfe 14	566
Büfe 16	709
Büfe 15	725
Büfe 9	1000
Büfe 7	528
Büfe 6	643
Büfe 2	451
Büfe 5	602
Büfe 1	432
Büfe 3	437
Büfe 4	378
Büfe 8	546
Büfe 10	463
Büfe 11	355
Toplam Mesafe	11241

İstanbul Halk Ekmek fabrikası Cebeci ilçesi de bulunan 16 adet büfe için uygulanan KKA ile en iyi rota 11241 metre olarak bulunmuştur.

7.2 GSP Excel- Solver (Evolutionary) ile Çözümü

Uygulamada kullanılan büfeler ve mesafelerin aynısını Excel-Solver kullanılarak çözümü yapılmıştır. Bunun için öncelikle büfeler arası mesafe matrisi oluşturuldu. Çizelge 4.1 ‘de görülmektedir. Daha sonra mesafe, sıra no ve büfe adı bulanan tablo oluşturulmuştur Çizelge 5.1 ‘de görülmektedir. Çizelge 5.1 ‘de mesafeyi bulmak için indis formülü kullanılmıştır. Örnek olarak “=İNDİS(C4:S20;E47;E31)” kodu kullanılmıştır. Büfe adını Çizelge 5.1 ’de bulmak için ise ara komutu kullanılmıştır. Örnek olarak “=ARA(E31;A4:A20; B4:B20)” kodu kullanılmıştır. Mesafelerin toplamı sarı olarak işaretlenmiştir.

Çizelge 4.1: Excel büfe mesafe matrisi

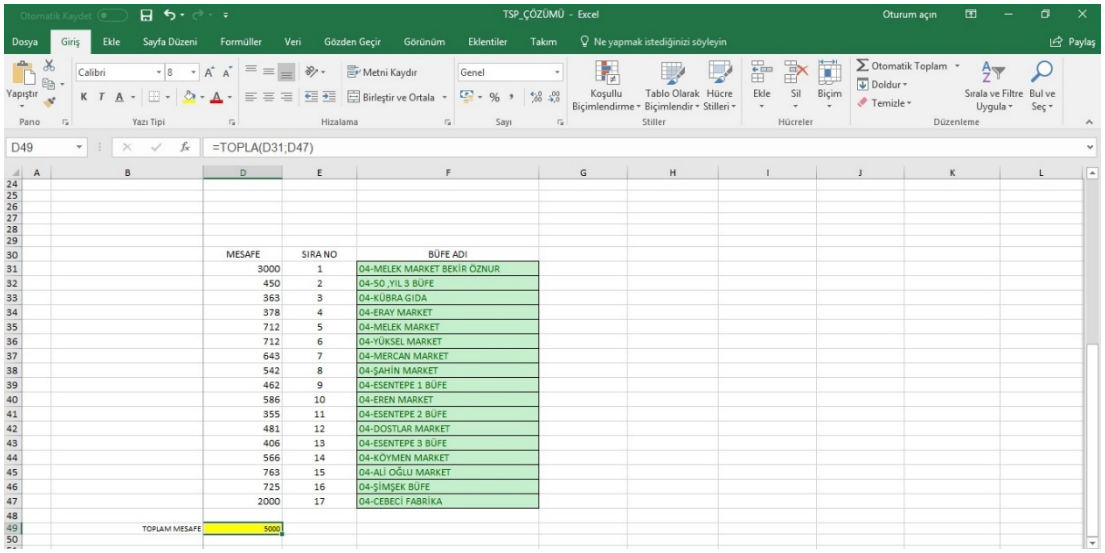


TSP (TRAVELİNG SALESMAN PROBLEM) GEZGİN SATICI PROBLEMİ ÇÖZÜMÜ

BÜFELER ARASI MESAFE MATRİSİ

Son adım olarak ise Excel-Solver kullanarak GSP probleminin çözülmesidir. Bunun için bazı parametreler eklenmiştir. Şekil 16.1 ’de görüldüğü üzere hedef ayarla yerine toplam sonucu bulduğumuz Excel hücresini yazıyor. Hedef olarak en küçük yazanı seçiyoruz. Değişken hücreleri değiştirerek yazan parametre ise Çizelge 4.1 ’de yaptığımız sıra no listesinin tamamını seçiyoruz. Excel Solver bu sıra numaraları arasında rastgele seçim yaparak en kısa yolu bulmaktadır.

Çizelge 5.1: Excel büfe mesafe hesaplama

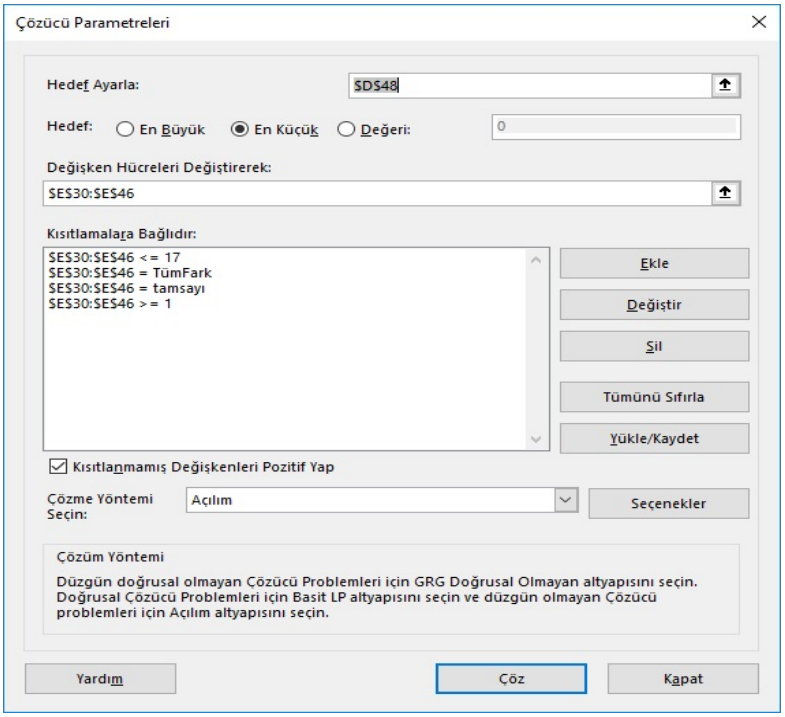


=TOPLA(D31:D47)

MESAFE	SIRA NO	BÜFE ADI
3000	1	04-MELEK MARKET BEKİR ÖZNUR
450	2	04-50_YIL 3 BÜFE
363	3	04-KÜBRA GIDA
378	4	04-ERAY MARKET
712	5	04-MELEK MARKET
712	6	04-YÜKSEL MARKET
643	7	04-MERCAN MARKET
542	8	04-ŞAHİN MARKET
462	9	04-ESENTEPE 1 BÜFE
586	10	04-EREN MARKET
355	11	04-ESENTEPE 2 BÜFE
481	12	04-DOSTLAR MARKET
406	13	04-ESENTEPE 3 BÜFE
566	14	04-KÖYMEN MARKET
763	15	04-ALİ OĞLU MARKET
725	16	04-ŞİMŞEK BÜFE
2000	17	04-CEBECİ FABRİKA
TOPLAM MESAFE	5000	

Bazı kısıtlamalar ekledik bunlar seçili sıra numaraların <=17, Tüm Fark, tamsayı ve >=1'den büyük olması kısıtlarıdır. Son olarak kullana bileceğimiz 3 farklı metot bulunmaktadır. Biz problemimize en uygun olan açılım metodunu kullandık. Sonuç olarak bulduğumuz değer en kısa rota 14829 metredir.

GSP probleminin çözümü için yapılan uygulama ve Excel Solver çözümünün kıyaslanmasının sonucu olarak uygulamanın bulduğu rota daha kısadır.



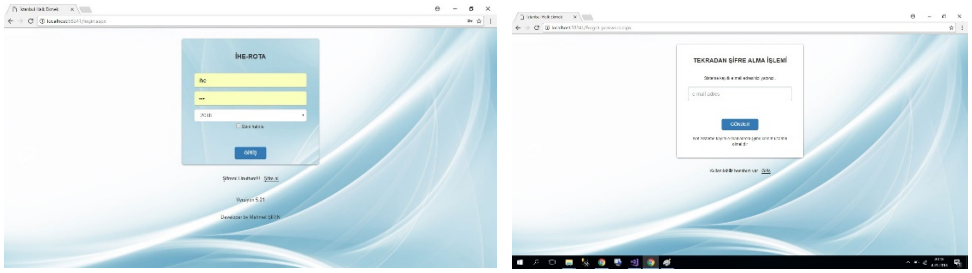
Şekil 16.1: Excel Solver parametreler

7.3 Uygulama Ara yüz Çalışmaları

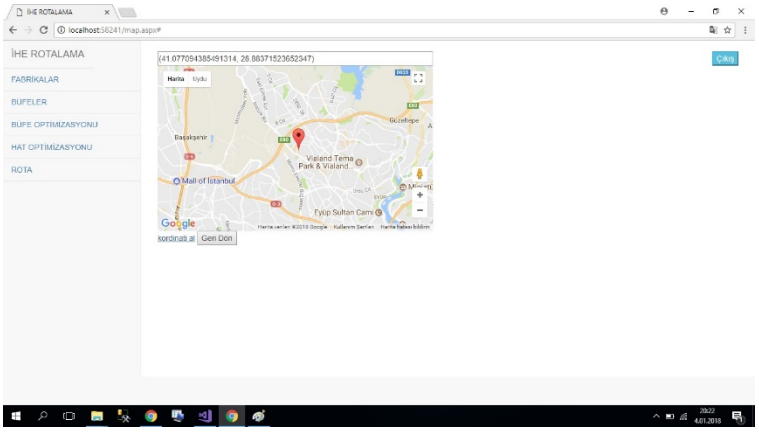
Yapılan uygulama 5 ana kısımdan oluşmaktadır. Bunlar şunlardır;

1. Fabrika ekleme, silme ve güncelleme sayfası
2. Büfe ekleme, silme ve güncelleme sayfası
3. Büfe optimizasyon sayfası
4. Hat optimizasyon sayfası
5. Rota çizme sayalsı

Ayrıca uygulamada kullanıcı girişi, şifre sıfırlama sayfası ve haritadan kolay koordinat bulunması için harita sayfası bulunmaktadır. Şekil 17.1 ve Şekil 18.1 ’de görülmektedir.



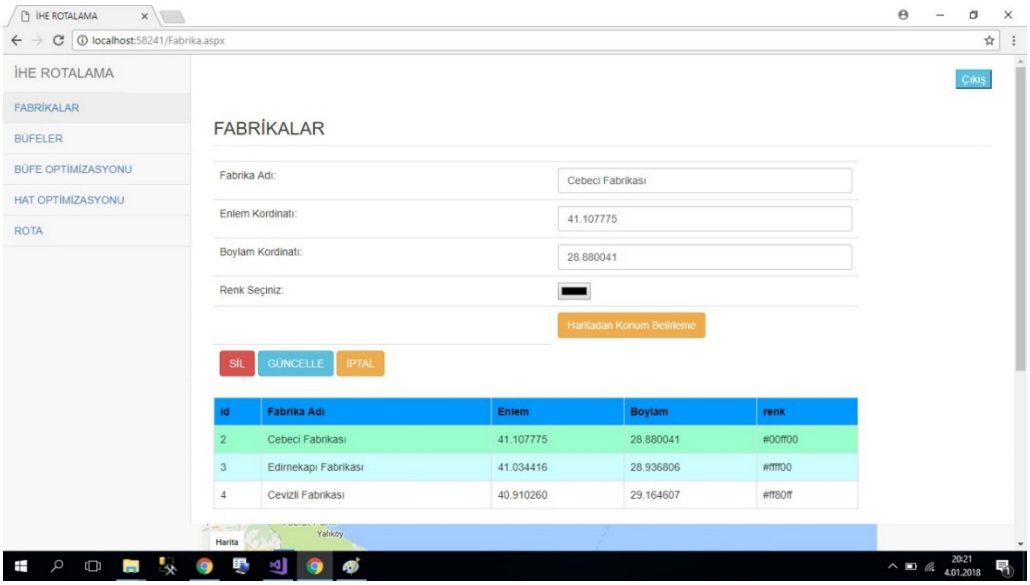
Şekil 17.1: Giriş ve şifre hatırlatma sayfası



Şekil 18.1: Haritadan Koordinat bulma

7.3.1 Fabrika Sayfası

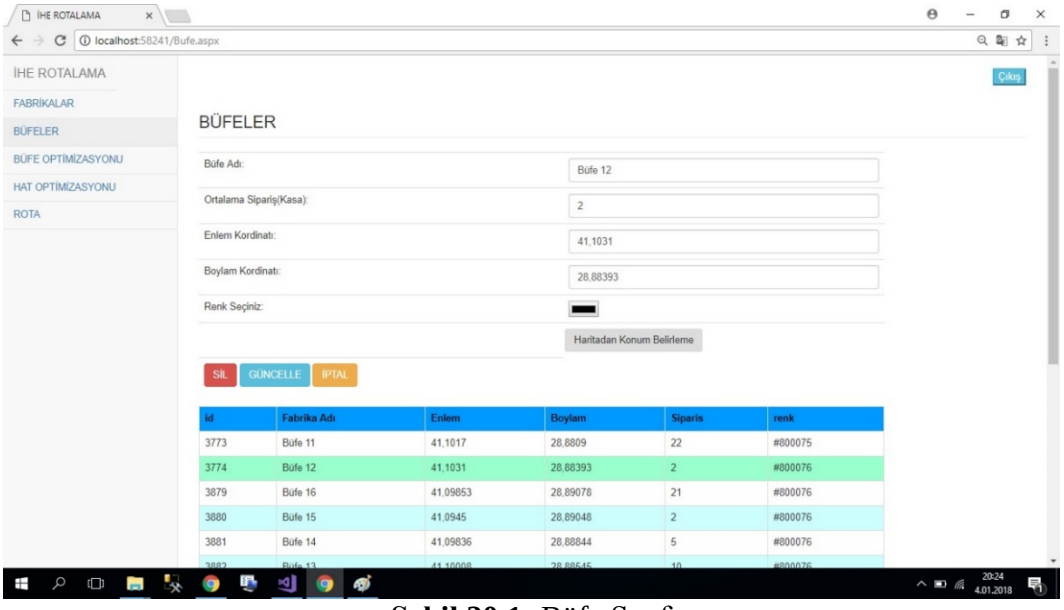
Bu sayfa uygulamada kullanılmak üzere fabrika ekleme işlemini yapıyoruz. Eklenecek fabrikanın adı, koordinatları sayısı ve haritada görünecek marker (ikon) rengini seçiyoruz. Şekil 19.1 ’de görüldüğü gibi.



Şekil 19.1: Fabrika ekleme, güncelleme ve silme işleminin yapıldığı sayfa

7.3.2 Büfe Sayfası

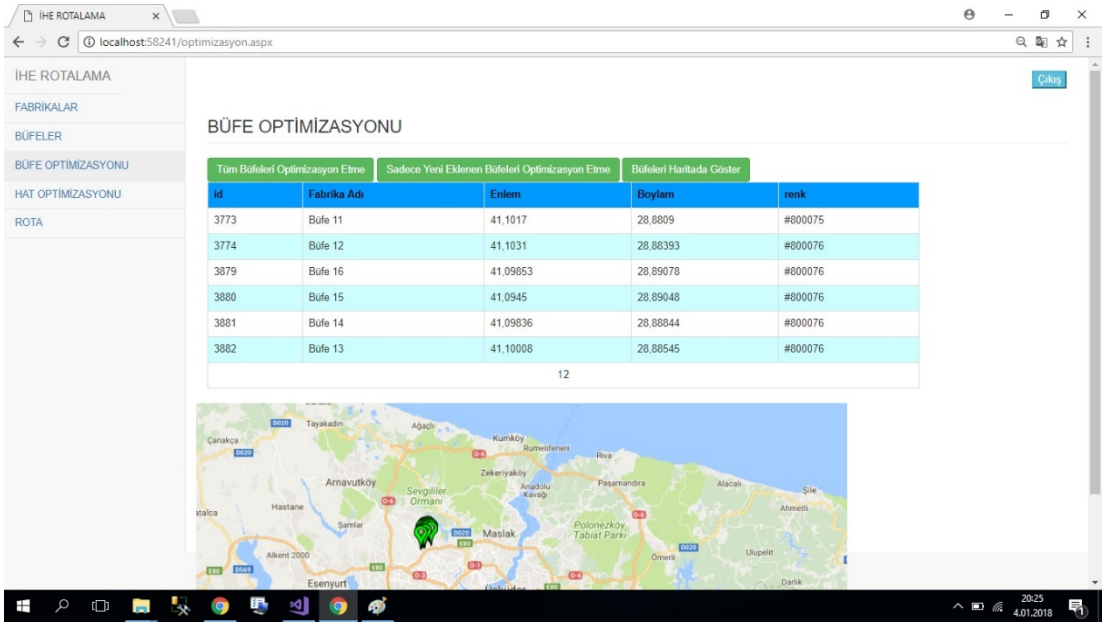
Bu sayfa uygulamada kullanılmak üzere fabrika ekleme işlemini yapıyoruz. Eklenecek fabrikanın adı, ortalama sipariş, koordinatları sayısı ve haritada görünecek marker (ikon) rengini seçiyoruz. Şekil 20.1 ’de görüldüğü gibi.



Şekil 20.1: Büfe Sayfası

7.3.3 Büfe Optimizasyon Sayfası

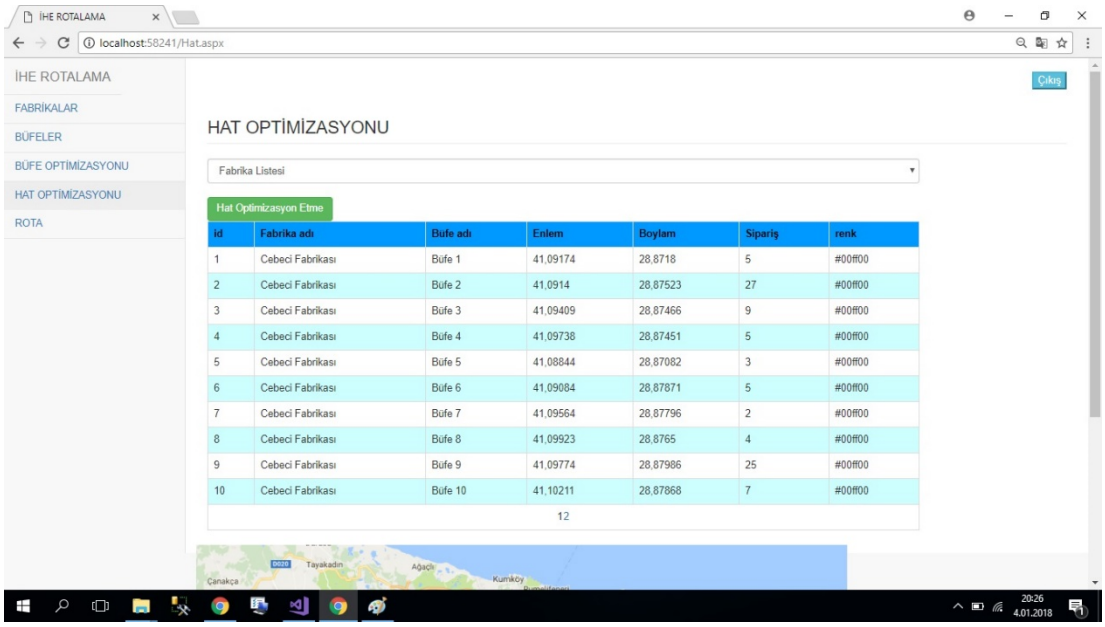
Uygulamanın bu sayfasında eklenen tüm büfeler kendilerine en yakın firmaya dâhil edilmektedir. Böylelikle fabrikaların araç sayılırını belirleyeceğiz. Ayrıca bu sayfada tüm büfeleri ekledikten sonra yeni büfeler eklenirse sadece o büfelerin optimize edilmesi ve zamanda tasarruf edilmesi için imkân sunulmuştur. Bu sayfada optimize işlemi bittikten sonra büfelerin hangi fabrikaya dâhil olduğunu haritada Şekil 21.1' de görebilmekteyiz.



Şekil 21.1: Büfe Optimizasyon Sayfası

7.3.4 Hat Optimizasyon Sayfası

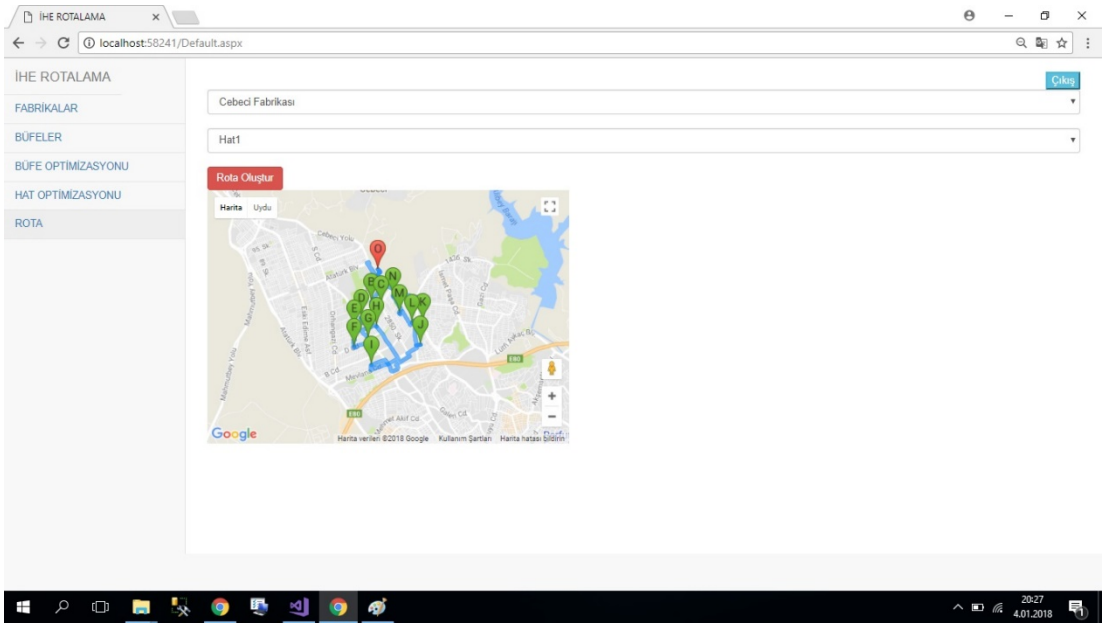
Uygulamanın bu sayfasında hat optimizasyonu yapılamaktadır. İlk önce dropdown liste bulunan fabrika listesinde fabrika seçiyoruz. Daha sonra optimizasyon işlemini başlatıyoruz. Uygulama büfelerin sipariş sayılarını ve fabrikaya yakınlıklarını baz alarak hatlara bölme işlemini gerçekleşmektedir. Hatlar oluşturulduktan sonra bu veriler kullanılarak rota çizme işlemi gerçekleştirilmiştir. Ekran görüntüsü Şekil 22.1 ' de görülmektedir.



Şekil 22.1: Hat Optimizasyon Sayfası

7.3.5 Rota Çizme Sayfası

Uygulamanın bu sayfasında seçilen hat'ın rotası çizilmektedir. İlk önce dropdown listen fabrika seçilmelidir daha sonra hat seçilmelidir. Böylelikle rota çizme işlemi gerçekleşmiştir. Rota çizimi Şekil 23.1 'de görülmektedir.



Şekil 23.1: Rota Çizme Sayfası

7.4 Uygulama Kod Kısmı

Bu kısımda tüm uygulama kodları anlatılmayacak sadece kullanılan özel class'lar gösterilecektir. Uygulamada kullanılan 3 farlı class bulunmaktadır. Bunlar;

1. Ant class
2. Googlemapsmodel class
3. Algoritma class

Ant class: Bu class' da karıncaların o anki konumları daha sonra alacakları konumlar ve toplam mesafe gibi bilgiler bulunmaktadır.

```
 public Ant(int startPos, int num_cities)
{
   currentLocation = startPos;
   nextLocation = 0;
   tour_number = 0;
   haveBeenList = new List<int>();
   tourList = new List<int>();
   distanceTraveled = 0;
   for (int k = 0; k < num_cities; k++)
   {
      haveBeenList.Add(0);
      tourList.Add(0);
   }
}

public void update_total_distance(double distance)
{
   distanceTraveled += distance;
}


public void set_current_location(int location)
{
   currentLocation = location;
}

public void set_next_location(int next)
{
   nextLocation = next;
}

public int get_current_location()
{
   return currentLocation;
}
public double getDistance()
```

```
    {
        return distanceTraveled;
    }

    public void resetDistance()
    {
        distanceTraveled = 0;
    }
```

Googlemapsmodel class: Bu class' da şehirler yani uygulamamızda büfe noktaların koordinatları, büfenin adı gibi parametereleri tutmaktayız. Ayrıca bu parametreleri googlemap model oluşturarak Google map'de kullanmaktayız.

```
public abstract class BaseCity : ICity
{

    public double X { get; set; }

    public double Y { get; set; }

    public string CityName { get; set; }
}

public class GoogleMapModel : BaseCity
{
    public string title
    {
        get { return CityName; }
        set { CityName = value; }
    }

    public double lat
    {
        get { return X; }
        set { X = value; }
    }
    public double lng
    {
        get { return Y; }
        set { Y = value; }
    }
```

Algoritma class: Bu class'da parametre değerlerin atanması, hesaplamaların yapıldığı, feromon update işlemi, mesafe hesaplamaları ve en iyi turun bulunması gibi çoğu işlem bu kısımda yapılmaktadır.

```
        public int num_cities { get { return city_list.Count; } }
    public int num_ants = 90;
    public int pherom_const = 100;
    public double ALPHA = 1.0;
    public double BETA = 1.0;
    public double RHO = .7;
```

```
public int iterations = 250;
public int click_counter = 0;

public Algoritma(List<GoogleMapModel> city_list)
{
   ant_list = new List<Ant>();
   best_tour_list = new List<int>();
   best_tour_length = -1;
   this.city_list = city_list;
   distances = new double[num_cities, num_cities];
   pheromones = new double[num_cities, num_cities];
   initAnts();
   initPherom();


   for (int i = 0; i < num_cities; i++)
      for (int k = 0; k < num_cities; k++)
      {
         double x = Math.Pow((double)city_list[i].X -
            (double)city_list[k].X, 2.0);
         double y = Math.Pow((double)city_list[i].Y -
            (double)city_list[k].Y, 2.0);
         distances[i, k] = Math.Sqrt(x + y);
      }
}



private void initPherom()
{
   for (int from = 0; from < num_cities; from++)
   {
      for (int to = 0; to < num_cities; to++)
      {
         pheromones[from, to] = 1.0 / (double)num_cities;
         pheromones[to, from] = 1.0 / (double)num_cities;
      }
   }
}



private void initAnts()
{
   int rand_city = 0;
```

```
    ant_list.Clear();
    for (int i = 0; i < num_ants; i++)
    {
        rand_city = rand_gen.Next(0, num_cities);
        ant_list.Add(new Ant(rand_city, num_cities));
        ant_list[i].tourList[0] = ant_list[i].get_current_location();
        ant_list[i].haveBeenList[ant_list[i].get_current_location()] = 1;
        ant_list[i].tour_number = 1;
    }
}


private void goToNextCity(Ant current_ant)
{
    double sum_prob = 0;
    double move_prob = 0;
    int current_city = current_ant.get_current_location();
    for (int i = 0; i < num_cities; i++)
    {
        if (current_ant.haveBeenList[i] == 0)
        {
            sum_prob += Math.Pow(pheromones[current_city, i], ALPHA) *
                Math.Pow(1.0 / distances[current_city, i], BETA);
        }
    }

    int destination_city = 0;
    double rand_move = 0;
    int count = 0;

    while (count < 400)
    {
        if (current_ant.haveBeenList[destination_city] == 0)
        {
            move_prob = (Math.Pow(pheromones[current_city, destination_city], ALPHA) *
                Math.Pow(1.0 / distances[current_city, destination_city], BETA)) / sum_prob;
            rand_move = rand_gen.NextDouble();
```

```
            if (rand_move < move_prob) break;
        }
        destination_city++;
        if (destination_city >= num_cities) destination_city = 0;
        count++;
    }
     current_ant.set_next_location(destination_city);
    current_ant.haveBeenList[destination_city] = 1;
    current_ant.tourList[current_ant.tour_number] = destination_city;
    current_ant.tour_number++;
     current_ant.update_total_distance(distances[current_ant.get_current_location(),
destination_city]);

     if (current_ant.tour_number == num_cities)
    {
        current_ant.update_total_distance(
            distances[current_ant.tourList[num_cities - 1], current_ant.tourList[0]]);
    }

    current_ant.set_current_location(destination_city);
}



private bool ants_stop()
{
    int moved = 0;
    for (int i = 0; i < num_ants; i++)
    {
        if (ant_list[i].tour_number < num_cities)
        {
            goToNextCity(ant_list[i]);
            moved++;
        }
    }
    if (moved == 0)
    {
        return true;
```

```
    }
    else return false;
}

public void evaporatePheromones()
{
     for (int i = 0; i < num_cities; i++)
        for (int k = 0; k < num_cities; k++)
        {
            pheromones[i, k] *= (1.0 - RHO);
             if (pheromones[i, k] < 1.0 / (double)num_cities)
            {
                pheromones[i, k] = 1.0 / (double)num_cities;
            }
        }
}



private void updatePheromones()
{
    for (int i = 0; i < num_ants; i++)
    {
        for (int k = 0; k < num_cities; k++)
        {
            int from = ant_list[i].tourList[k];
             int to = ant_list[i].tourList[((k + 1) % num_cities)];
            pheromones[from, to] += (double)pherom_const / ant_list[i].getDistance();
            pheromones[to, from] = pheromones[from, to];
        }
    }
}

private void best_tour()
{
    double best_local_tour = ant_list[0].getDistance();
    int save_index = 0;
```

```
    for (int i = 1; i < ant_list.Count; i++)
    {
        if (ant_list[i].getDistance() < best_local_tour)
        {
            best_local_tour = ant_list[i].getDistance();
            save_index = i;
        }
    }
     if (best_local_tour < best_tour_length || best_tour_length == -1)
    {
        best_tour_list = ant_list[save_index].tourList;
        best_tour_length = best_local_tour;
     }
}
 public int fact(int n)
{
    if (n == 0) return 1;
    else return n * fact(n - 1);
}

public void Hesapla()
{
            acoWatch.Start();

    for (int k = 0; k < iterations; k++)
    {
        for (int i = 0; i < num_cities; i++)
            if (ants_stop())
            {
                best_tour();
            }
    }
    acoWatch.Stop();
    var totalTimeACO = acoWatch.Elapsed.TotalSeconds.ToString();
    acoWatch.Reset();
}
```

8. DENESEL ÇALIŞMALAR

Bu bölümde, uygulamamızda kullanılan KKA için kullanılan parametre değerleri test edilerek en uygun sonuca ulaşılmaya çalışılmıştır. Uygulamamızda yer alan 16 adet büfe için en iyi şekilde sıralanabilmesi için, alpha, beta, rho, iterasyon sayısı ve karınca sayısı ayrı ayrı testlere tabi tutulmuştur. Uygulamamızın, belirlenen konumlardaki büfeler için en kısa mesafeyi verebilmesi amacıyla belirtilen parametrelerin test kombinasyonları yapılmıştır. Başarı ölçütü olarak belirtilen parametreler ışığında, en kısa mesafeyi veren değerler olarak belirlenmiştir.

8.1 Alpha Parametresi Değişiminin Etkileri

Bu parametre, feromon sıvısı ağırlığına etki etmektedir. Karıncaların feromon sıvısı yardımı ile bıraktıkları izin artmasına veya azalmasına doğrudan etki etmektedir. Alpha parametresi için 0.25, 0.50, 0.75, 1, 1.25, 1.50, 1.75 ve 2 değerleri verilmiş ve bu değerlere göre ortaya çıkan sonuçlar Çizelge 6.1 ' de gösterilmiştir. Alpha parametresine farklı değerler verilirken diğer parametreler sabit tutulmuştur.

Çizelge 6.1: Alpha parametresi değişimi ile elde edilen mesafe sonuçları

ALPHA	BETA	RHO	İTERASYON S.	BÜFE S.	KARINCA S.	MESAFE
0,25	1	0,5	100	16	100	2259
0,50	1	0,5	100	16	100	2092
0,75	1	0,5	100	16	100	1779
1	1	0,5	100	16	100	1694
1,25	1	0,5	100	16	100	1711
1,50	1	0,5	100	16	100	1726
1,75	1	0,5	100	16	100	1730
2	1	0,5	100	16	100	1809

8.2 Beta Parametresi Değişiminin Etkileri

Beta parametresi, noktalar arası uzaklığa bağlı bir değişkendir. Beta değerinin artması sezgisel seçim ihtimalini arttırmaktadır. Bir sonraki komşunun seçilmesi beta parametresinin miktarına bağlıdır. Beta parametresi için 1, 1,5, 2, 2,5, 3, 3,5, 4 ve 5 değerleri verilmiş ve bu değerlere göre ortaya çıkan sonuçlar Çizelge 7.1 'de gösterilmiştir. Beta parametresine farklı değerler verilirken diğer parametreler sabit tutulmuştur.

Çizelge 7.1: Beta parametresi değişimi ile elde edilen mesafe sonuçları

ALPHA	BETA	RHO	İTERASYON S.	BÜFE S.	KARINCA S.	MESAFE
1	1	0,5	100	16	100	1698
1	1,5	0,5	100	16	100	1705
1	2	0,5	100	16	100	1708
1	2,5	0,5	100	16	100	1710
1	3	0,5	100	16	100	1718
1	3,5	0,5	100	16	100	1723
1	4	0,5	100	16	100	1729
1	5	0,5	100	16	100	1739

8.3 RHO parametresi değişiminin etkileri

Rho parametresi feromon sıvısı bozunma oranını ifade eder. Belli bir süre sonra feromon sıvısı buharlaşacağı için feromon sıvısı güncellemesi bozunma oranı dikkate alınarak yapılır. Rho parametresi için 0,2, 0,3, 0,4, 0,5, 0,6, 0,7, 0,8 ve 0,9 değerleri verilmiş ve bu değerlere göre ortaya çıkan sonuçlar Çizelge 8.1 'de gösterilmiştir. Rho parametresine farklı değerler verilirken diğer parametreler sabit tutulmuştur.

Çizelge 8.1: RHO parametresi değişimi ile elde edilen mesafe sonuçları

ALPHA	BETA	RHO	İTERASYON S.	BÜFE S.	KARINCA S.	MESAFE
1	1	0,2	100	16	100	1752

1	1	0,3	100	16	100	1745
1	1	0,4	100	16	100	1744
1	1	0,5	100	16	100	1734
1	1	0,6	100	16	100	1728
1	1	0,7	100	16	100	1709
1	1	0,8	100	16	100	1754
1	1	0,9	100	16	100	1762

8.3 İterasyon Parametresi Değişiminin Etkileri

Bu parametre, fonksiyonlarımızın kaç kere tekrar edeceğini belirtir. Bu sayede algoritma öğrenmesi sağlanır. İterasyon sayısı ne kadar çok artarsa problemimiz için yapılan sıralama o kadar kaliteli sonuç verecektir fakat cevap verme süresi uzayacağı için bu parametre değeri 250 olarak belirlenmiştir. İterasyon parametresi için 50, 100, 150, 200, 250, 300, 350 ve 400 değerleri verilmiş ve bu değerlere göre ortaya çıkan sonuçlar Çizelge 9.1 ’de gösterilmiştir. İterasyon parametresine farklı değerler verilirken diğer parametreler sabit tutulmuştur.

Çizelge 9.1: İterasyon parametresi değişimi ile elde edilen mesafe sonuçları

ALPHA	BETA	RHO	İTERASYON S.	BÜFE S.	KARINCA S.	MESAFE
1	1	0,7	50	16	100	1756
1	1	0,7	100	16	100	1744
1	1	0,7	150	16	100	1724
1	1	0,7	200	16	100	1702
1	1	0,7	250	16	100	1718
1	1	0,7	300	16	100	1696
1	1	0,7	350	16	100	1698
1	1	0,7	400	16	100	1697

Çizelge 9.1 ’ de görüldüğü gibi, en iyi sonuçlar İterasyon parametresi sürekli artarken gözlemlenmiş ve uygulamamızın cevap verme süresi göz önünde bulundurularak 250 olarak belirlenmiştir.

8.4 Karınca Sayısı Parametresi Değişiminin Etkileri

Bu parametre, belirlenene noktalar için kullanılan karınca miktarını belirtir. Karıncaların miktarı, salgıladıkları feromon sıvısının meydana getirdiği takip etme içgüdüsü sebebi ile çok önemlidir. Karınca sayısının gereğinden az olması, takip edilme oranını azaltacağı gibi, gereğinden çok olması, belli bölgelerde yoğunluk yaratacağından kalitesiz sonuç vermeye yol açacaktır. Karınca sayısı parametresi için 50, 70, 90, 110, 130, 150, 170 ve 190 değerleri verilmiş ve bu değerlere göre ortaya çıkan sonuçlar Çizelge 10.1 ’de gösterilmiştir. Karınca sayısı parametresine farklı değerler verilirken diğer parametreler sabit tutulmuştur. Çizelge 10.1 ’de görüldüğü gibi, en iyi sonuç Karınca sayısı parametresi 90 iken sağlanmıştır.

Çizelge 10.1: Karınca sayısı parametresi değişimi ile elde edilen mesafe sonuçları

ALPHA	BETA	RHO	İTERASYON S.	BÜFE S.	KARINCA S.	MESAFE
1	1	0,7	250	16	50	1756
1	1	0,7	250	16	70	1768
1	1	0,7	250	16	90	1698
1	1	0,7	250	16	110	1702
1	1	0,7	250	16	130	1727
1	1	0,7	250	16	150	1729
1	1	0,7	250	16	170	1732
1	1	0,7	250	16	190	1734

Uygulamamızda bulunan 16 adet büfe için en uygun parametreler belirlenmiş olup, değerleri Alpha 1, Beta 1, Rho 0,7, İterasyon Sayısı 250 ve Karınca Sayısı 90 olarak belirlenmiştir. Bu sonuçlarla büfe koordinatları üzerinden yapılan sıralamanın kalitesi artmıştır.

9. SONUÇ VE ÖNERİLER

Bu çalışmada, GSP Problemi tanıtılmış ve çözüm yöntemleri açıklanmıştır. Araçların büfelere en kısa güzergâhtan rotalanması üzerine, web ortamında çalışacak bir uygulama yazılmış ve bu uygulama için KKA kullanılmıştır.

Bu çalışma sonucunda şirket araçlarının zamandan tasarruf etmesini sağlayan bir uygulama geliştirilmiştir. Pilot ilçe olarak seçilen Cebeci ilçesine ait 16 adet büfe noktası için KKA kullanılarak hem zamandan tasarruf edilmiş hem de maliyetin düşürülmesi için en kısa yoldan büfelere dağıtımın yapılması sağlanmıştır.

Geliştirilen uygulamanın web tabanlı olması hem kullanışlı hem de tüm cihazları desteklemesi uygulanılan şirket açısından çok faydalı oldu. Böylelikle uygulama hem telefonlardan hem de tabletlerden kullanılması sağlandı.

GSP probleminin çözümü için yapılan uygulama ve Excel Solver çözümünün kıyaslanmasının sonucunda uygulamanın bulduğu rota daha kısa olduğu görülmüştür.

Çalışmada kullanılan 16 büfe ve fabrika sayısı daha da arttırılabilir, sabit noktalar için uyarlanabilir bir yapıdadır. Yaptığım bu çalışma ile GSP çözüm tekniklerinin tanıtılması, bu konuda araştırma yapan akademisyenlere ve yöneticilere genel bir fikir vermek amaçlanmıştır.

KAYNAKLAR

[1] **G. Laporte And I. H. Osman, Potvin, J., Y.,** (1996) "Genetic Algorithms For The Travelling Salesman Problem", Forthcoming İn Annals Of Operations Research On "Metaheuristics İn Combinatorial Optimization", Eds. 63: 339-370

[2] **Dantzig, G.B., Ramser J.B.,** (1959) The Truck Dispatching Problem. Management Sci. 6, 80-91

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

[4] **Şükran Şeker** (2017) "Araç Rotalama Problemleri Ve Zaman Pencereli Stokastik Araç Rotalama Problemine Genetik Algoritma Yaklaşımı" Fbe Endüstri Mühendisliği Anabilim Dalında Endüstri Mühendisliği Yıldız Teknik Üniversitesi

[5] **Rabia Gökçen Büyükyılmaz** (2017) "Eş Zamanlı Topla Dağıt Araç Rotalama Problemi İçin Yeni Bir Çözüm Önerisi" Sakarya Üniversitesi Fen Bilimleri Enstitüsü

[6] **Can Kaya** (2017) "Eş Zamanlı Topla Dağıt Araç Rotalama Problemi İçin Karınca Koloni Sistemi İle Güçlendirilmiş Değişken Komşuluk Arama Algoritması "Pamukkale Üniversitesi Fen Bilimleri Enstitüsü Endüstri Mühendisliği Anabilim Dalı

[7] **Gözde Gürkan Altunsoy** (2017) "Genelleştirilmiş Seçici Gezgin Satıcı Problemleri İçin Yeni Matematiksel Modeller" Başkent Üniversitesi Fen Bilimleri Enstitüsü

[8] **Saniye Çeyrekoğlu** (2017) "Araç Rotalama Problemine Genetik Algoritma Yaklaşımı Ve Örnek Bir Uygulama" İstanbul Üniversitesi Fen Bilimleri Enstitüsü

[9] **Çakır E. T.** (2016) "Bir Tersine Lojistik Faaliyeti Olarak Tıbbi Atıkların Toplanmasında Araç Rotalama Uygulaması" Sakarya Üniversitesi, Sosyal Bilimler Enstitüsü İşletme Anabilim Dalı Üretim Yönetimi Ve Pazarlama Bilim Dalı

[10] **Gizem Ermiş** (2015) Acceleratıng Local Search Algorıthms For Travellıng Salesman Problem Usıng Gpu Effectıvely Ubmitted To The Graduate School Of Engineering And Natural Sciences İn Partial Fulfillment Of The Requirements For The Degree Of Master Of Science Sabancı University July

[11] **Ahmet Sedat** Kaya (2015) "Travelıng Salesman Problem: Herustıcs And Empırıcal Evaluatıon" In Partıal Fulfıllment Of The Requırements For The Degree Of Master Of Scıence In The Department Of Computer Engıneerıng

[12] **Melis Alpaslan** (2015) "Araç Rotalama Problemleri İçin Matematiksel Modeller Ve

Çözüm Yöntemleri" Anadolu Üniversitesi Fen Bilimleri Enstitüsü Endüstri Mühendisliği Anabilim Dalı

[13] Atasagun, Gözde Can (2017) "Zaman Bağımlı Eş Zamanlı Topla Dağıt Araç Rotalama Problemi" Selçuk Üniversitesi Fen Bilimleri Enstitüsü

[14] Eldem, H. (2014). "Karınca Kolonisi Optimizasyonu (Aco) Ve Parçacık Sürü Optimizasyonu (Pso) Algoritmaları Temelli Bir Hiyerarşik Yaklaşım Geliştirilmesi" Selçuk Üniversitesi, Yayımlanmış Yüksek Lisans Tezi, Konya.

[15] Hasan Dikmen, Hüseyin Dikmen, Ahmet Elbir, Ziya Ekşi, Fatih Çelik (2014) "Gezgin Satıcı Probleminin Karınca Kolonisi Ve Genetik" 18(1), 8-13, 2014

[16] Burak Kosif, İsmail Ekmekçi (2012) Araç Rotalama Sistemleri Ve Tasarruf Algoritması Uygulaması Cilt 11, Sayı 21, Sayfalar 41 – 51

[17] Osman Gökalp (2012) "Karınca Kolonisi Eniyilemesi Algoritmaları İçin Çaprazlama Yöntemleri Geliştirilmesi" Ege Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı

[18] Suvaydan F. (2011)" Mobil Robotlar İçin Yol Planlama Problemi Ve Karınca Kolonisi Algoritması İle Yol Planlama Problemlerinin Optimal Çözümü" Düzce Üniversitesi Fen Bilimleri Enstitüsü Elektrik Eğitimi Anabilim Dalı

[19] Çalışkan Özyer, Kâmil Tansel (2011) "Karınca Kolonisi Optimizasyonu İle Araç Rotalama Probleminin Maliyetlerinin Kümeleme Tekniği İle İyileştirilmesi" Tobb Ekonomi Ve Teknoloji Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı

[20] Serçin Özkan (2010) "Satıcı Probleminin Çözümüne Yönelik Algoritmik Yaklaşımlar" Yüksek Lisans Tezi Endüstri Mühendisliği Gazi Üniversitesi Fen Bilimleri Enstitüsü

[21] Keskintürk T. (2009)" Araç Rotalama Problemlerinin Global Karınca Koloni Optimizasyonu İle Çözümü" İstanbul Üniversitesi Sosyal Bilimler Enstitüsü İşletme Anabilim Dalı Sayısal Yöntemler Bilim Dalı

[22] Hasan Söyler, Timur Keskintürk (2007)" Karınca Kolonisi Algoritması İle Gezen Satıcı Probleminin Çözümü" 8. Türkiye Ekonometri Ve İstatistik Kongresi24-25 Mayıs – İnönü Üniversitesi Malatya

[23] Bellman, R. (1958), "On A Routing Problem", İn Quarterly Of Applied Mathematics, 16(1): 87-90.

[24] Cruyssen, V. D. P., Rijkaert. M., (1978), "Heuristic For The Asymetric Travelling Salesmanproblem", The Journal Of The Operational Research Society, 29(7):697–701.

[25] Liu, Z. Ve Kang, L., 2004, A Hybrid Algorithm Of N-Opt And Ga To Solve Dynamic Tsp, Proceedings Of The Grid And Cooperative Computing.

[26] Ahuja, R. K., Magnanti, T. L., Orlin, J. B. (1993), "Network Flows: Theory, Algorithmsand Applications", Prentice Hall: New Jersey, 37:211-276.

[27] Eiselt, H. A., Gendreau, M., Laporte, G. (1995), "Arc Routing Problems, Part 1: Thechinese Postman Problem", Operations Research, 43(2): 231–242.

[28] Golden, B., Bodin L., Doyle T., Stewart W, (1980), "Approximate Traveling Salesman Algorithms", Operations Research, June 1, 694-711.

[29] Ropke, S., (2005). Heuristic And Exact Algorithms For Vehicle Routing Problems.

[30] Kumar, S.N., (2012). Panneerselvam, R., A Survey On The Vehicle Routing Problem And Its Variants. Intelligent Information Management, 4, 66-74.

[31] Pan, L., (2015). Cutting Plane Method. The Chinese University Of Hong Kong, Operations Research And Logistics Jan. 20.

[32] Toth, P., Vigo, D., (2002). The Vehicle Routing Problem.
Windows And Transshipment. Inform, Vol. 44, No. 217-227, Issn:0315-5986

[33] Araque, J.R., Kudva, G., Morin, T.L., Pekny, J.F., (1994). A Branch-And-Cut Algorithm For Vehicle Routing Problems. Annals Of Operations Research 50, 37-59.

[34] Başkaya Z., Avcı Öztürk, B., (2005). Tamsayılı Programlamada Dal Kesme Yöntemi Ve Bir Ekmek Fabrikasında Oluşturulan Araç Rotalama Problemine Uygulanması. Uludağ Üniversitesi İktisadi Ve İdari Bilimler Fakültesi Dergisi Cilt Xxıv, Sayı 1, S. 101-114.

[35] Chauhan, C., Gupta, R., Pathak, K., (2012). Survey Of Methods Of Solving Tsp Along With İts Implementation Using Dynamic Programming Approach. International Journal Of Computer Applications (0975–8887) Volume 52– No.4.

[36] Cordeau, J.F., Gendreau, M., Laporte, G., Potvin, J.Y., Semet, F., (2002). A Guide To Vehicle Routing Heuristics. The Journal Of The Operational Research Society, Vol. 53, No. 5 Pp. 512-522.

[37] Bozyer, Z., Alkan, A., Fığlalı, A., (2014). Kapasite Kısıtlı Araç Rotalama Probleminin Çözümü İçin Önce Grupla Sonra Rotala Merkezli Sezgisel Algoritma Önerisi. Bilişim Teknolojileri Dergisi, Cilt: 7, Sayı: 2

[38] Atmaca, E., (2012). Bir Kargo Şirketinde Araç Rotalama Problemi Ve Uygulaması. Türk Bilim Araştırma Vakfı Dergisi, Cilt:5, Sayı:2, Sayfa: 12-27

[39] Eryavuz, M., Gencer, C., (2001). Araç Rotalama Problemlerine Ait Bir Uygulama.Süleyman Demirel Üniversitesi İktisadi Ve İdari Bilimler Fakültesi C.6, S.1, S.139-155.

[40] Nilsson, C., (2003). Heuristics For The Traveling Salesman Problem. Linköping University.

[41] Nurcahyo, G.W., Alias, R.A., Shamsuddın, S.M., Sap., M.N.M., (2002), Sweep Algorithm İn Vehicle Routing Problem For Public Transport. Jurnal Antarabangsa (Teknologi Maklumat) 2(2002): 51-64.

[42] Düzakın, E., Demircioğlu, M., (2009). Araç Rotalama Problemleri Ve Çözüm

Yöntemleri. Çukurova Üniversitesi İibf Dergisi Cilt:13. Sayı:1, Ss.68-87.

[43] Laporte, G., (1992). The Vehicle Routing Problem: An Overview Of Exact And Approximate Algorithms. European Journal Of Operational Research 59, 345-358.

[44] Şahin, Y., Eroğlu, A., (2014). Kapasite Kısıtlı Araç Rotalama Problemi İçin Metasezgisel Yöntemler: Bilimsel Yazın Taraması.Süleyman Demirel Üniversitesi İktisadi Ve İdari Bilimler Fakültesi Dergisi C.19, S.4, S.337-355.

[45] Genreau, M., Potvin, J.Y., Braysy, O., Hasle, G., Lokketangen, A., (2007). Metaheuristics For The Vehicle Routing Problem And İts Extensions: A Categorized Bibliography. Cırrelt-2007-27.

[46] Cura, T., 2008, Modern Sezgisel Teknikler Ve Uygulamaları 1.Baskı, Papatya Yayıncılık, İstanbul, Isbn: 978-975-6797-79-2

[47] Obitko, M., "Crossover And Mutation", Http://Www.Obitko.Com/Tutorials/Genetic-Algorithms/Crossover-Mutation.Php (Erişim Tarihi: 01 Ocak 2018).

[48] Çevik, K.K., Koçer, H.E., (2013). Parçacık Sürü Optimizasyonu İle Yapay Sinir Ağları Eğitimine Dayalı Bir Esnek Hesaplama Uygulaması. Süleyman Demirel Üniversitesi Fen Bilimleri Enstitüsü Dergisi, 17 (2), 39-45.

[49] Cordon, O., Herrera, F., & Stutzle, T. A Review On The Ant Colony Optimization Metaheuristic: Basis, Models And New Trends. Mathware And Soft Computing, 9(2–3), :141–175, 2002

[50] Dorigo, M., Birattari, M. And Stützle, T., (2006), Ant Colony Optimization: Artificial Ants As A Computational İntelligence Technique, Ieee Computational Intelligence Magazine, 1(4):28-39.

[51] Aydın, D., 2011, Sürü Zekâsı Yaklaşımlarının Renkli Görüntü Kesimlemeye Uyarlanması Ve Tanıma Sistemleri Üzerinde Gerçekleştirimi, Doktora Tezi, Ege Üniversitesi Fen Bilimleri Enstitüsü, 113s.

[52] Dorigo M., Maniezzo V. And Colorni A., 1996, Ant System: Optimization By A Colony Of Cooperating Agents, Systems, Man, And Cybernetics, Part B: Cybernetics, Ieee Transactions On, 26(1):29-40.

[53] Cordon, O., Viana, I.F. And Moreno, L., (2000), A New Aco Model İntegrating Evolutionary Computation Concepts: The Best-Worst Ant System, Proceedings Of Ants2000, 22-29.

[54] Stützle, T. And Hoos, H.H., 2000, Max–Mın Ant System, Future Generation Computer Systems, 16:889-914.

[55] Bullnheimer, B., Hartl, R.F. And Strauss, C., 1997, A New Ranked-Based Version Of The Ant System: A Computational Study, Central European Journal For Operational Research And Economics, 7:25-38.

EKLER

Ek A: Web Tabanlı Araç Rotalama Uygulamasının Kodlanmış Kısmı Cd'de teslim edilecektir

ÖZGEÇMİŞ

Mehmet ŞİRİN, 20.03.1988 tarihinde Bitlis'te doğdu. 2001'de Adanan Menderes İlköğretim Okulu'ndan mezun oldu. 2006 yılında İstanbul Dr. Sadik Ahmet Lisesi'nden mezun oldu. 2011 yılında Anadolu Eskişehir Üniversitesi İşletme Fakültesi'nden mezun oldu. 2011 yılında Bitlis Eren Üniversitesi Bilgisayar Programcılığı'ndan dikey geçiş ile İstanbul Aydın Üniversitesi Bilgisayar Mühendisliğine geçiş yaptı. 2014 yılında İstanbul Aydın Üniversitesi Bilgisayar Mühendisliği Fakültesi'nden mezun oldu. 2014 yılında İstanbul Aydın Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği'nde Yüksek Lisans eğitimine başladı. İyi derecede İngilizce dilini bilmektedir. İlgi alanlarından bazıları, Yapay Zekâ, Sürü Zekâsı, Bulanık Mantık, Android Programlamadır.