T.C.
İSTANBUL AYDIN ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

SEZGİSEL ALGORİTMALAR KULLANILARAK TARİHSEL MEKANLARIN EN KISA YOLDAN ROTALANMASI ÜZERİNE BİR MOBİL UYGULAMA

YÜKSEK LİSANS TEZİ

Mehmet Rıdvan ALTUNKUM

(Y1513.010016)

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

Tez Danışmanı: Prof. Dr. Ali GÜNEŞ

Ağustos, 2017

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 **Y1513.010016** numaralı öğrencisi **Mehmet Rıdvan ALTINKUM** 'un **"SEZGİSEL ALGORİTMALAR KULLANILARAK TARİHSEL MEKÂNLARIN EN KISA YOLDAN ROTALANMASI ÜZERİNE BİR MOBİL UYGULAMA"** adlı tez çalışması Enstitümüz Yönetim Kurulunun 01.08.2017 tarih ve 2017/17 sayılı kararıyla oluşturulan jüri tarafından Oy birliği ile Tezli Yüksek Lisans tezi olarak .kabul....edilmiştir.

Öğretim Üyesi Adı Soyadı	İmzası
Tez Savunma Tarihi : 28/08/2017	
1)Tez Danışmanı: Prof. Dr. Ali GÜNEŞ	.....
2) Jüri Üyesi : Doç. Dr. Metin ZONTUL	.....
3) Jüri Üyesi : Yrd. Doç. Dr. Ferdi SÖNMEZ	.....

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

YEMİN METNİ

Yüksek Lisans tezi olarak sunduğum “Sezgisel Algoritmalar Kullanılarak Tarihsel Mekanların En Kısa Yoldan Rotalanması Üzerine Bir Mobil Uygulama” adlı çalışmanın tezin projesi safhasından sonuçlanmasına kadarki bütün süreçlerde bilimsel ahlak ve geleneklere aykırı düşecek bir yardıma başvurulmaksızın yazıldığını ve yararlandığım eserlerin Bibliyografya ‘da gösterilenlerde oluştuğunu, bunlara atıf yapılarak yararlanılmış olduğunu belirtir ve onurumla beyan ederim. (28/08/2017)

Mehmet Rıdvan ALTUNKUM

Aileme,

ÖNSÖZ

İlk olarak tez çalışmamın hazırlanmasında her türlü yardımı esirgemeyen ayrıca değerli görüş ve yorumları, rehberliği ve desteği için değerli danışman hocam sayın Prof. Dr Ali GÜNEŞ'e Teşekkür ederim. Bu tez çalışma süresi boyunca sabrı, anlayışı ve desteği ile bana yardımcı olan annem ve babama sonsuz Teşekkür ederim.

Ağustos,2017 **Mehmet Rıdvan ALTUNKUM**

İÇİNDEKILER

Sayfa

KISALTMALAR

KKA **:** Karınca kolonisi algoritması
KSA **:** Karınca sistemi algoritması
KKO **:** Karınca kolonisi optimizasyonu
GSP **:** Gezgin satıcı problemi
EKS : Elitist Karınca Sistemi
GA **:** Genetik Algoritmalar
GPS **:** Küresel Konumlandırma Sistemi
RHO **:** Feromon sıvısı buharlaşma oranı

SİMGELER

α : Feromon katsayısı
β : Görünürlük katsayısı
ρ : Buharlaşma katsayısı
m : Karınca sayısı
n : Tarihsel mekan sayısı
p_{ij}^{k} : k karıncasının i noktasından j noktasına geçme ihtimali
t : Döngü sayacı
t_0 : Başlangıç feromon madde miktarı
$\tau ij\ (t)$: t döngüsünde (i,j) güzergahı üzerinde bulunan feromon sıvısı miktarı
$\Delta\tau_{ij}^{k}(t)$: Döngü sonunda k karıncasının (i,j) arasına bırakılacak iz miktarı
Q : İz miktarının belirlenmesinde kullanılan sabit
L_k : k karıncası tarafından oluşturulan çözümün tur uzunluğu
N_t : Tur sayısı (iterasyon)
N_{min} : En kısa tura kaçıncı turda ulaşıldığı
Ψ^0 : Başlangıç çözümü

ÇİZELGE LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

SEZGİSEL ALGORİTMALAR KULLANILARAK TARİHSEL MEKANLARIN EN KISA YOLDAN ROTALANMASI ÜZERİNE BİR MOBİL UYGULAMA

ÖZET

Günümüzde hızla gelişen teknoloji sayesinde kullanıcılar artık her zaman teknolojiye hızlıca erişebilme gücüne ihtiyaç duymaktadırlar. Her yaştan ve her guruptan insanın teknolojiyle ilişkisi günden güne artmaktadır. Hızla ilerleyen teknoloji, kullanıcılarının ihtiyaçlarına cevap verebilmek için sürekli gelişmekte ve bu durum kullanıcıları farklı ihtiyaçlara yönlendirmektedir. Son yıllarda sayıları hızla artan mobil cihazlar geçmişe göre teknolojik cihazların günlük hayatta çok daha fazla kullanılmasına sebep olmaktadır. Önceleri belli bir yerden sabit olarak erişebildiğimiz cihazların bir kısmını artık cebimizde taşıyabiliyoruz. İnternet teknolojisinin de artık kablolu dışında kablosuza geçmesiyle, akıllı telefonlar, tabletler, şarj cihazları ve daha bir çok cihazı taşınılabilir ve mekan sınırlaması olmadan kullanılabilmektedir. Yeni nesil akıllı telefonlarda günlük hayatta kullandığımız birçok uygulama bulunmaktadır. Bilmediğimiz bir adrese navigasyonla gidebilir, en yakın otobüs durağını bulabilir, taksi çağırabilir, yazışmalar yapabilir, mail gönderebilir, alışveriş yapabilir, sosyal platformlarda paylaşımlar yapabiliriz. Bir çok alanda yapılan mobil uygulamalar bulunmasına rağmen, İstanbul ilinde bulunan tarihsel mekanlar hakkında hem bilgi veren hem de bu tarihsel mekanları birden fazla yolla gezdiren, rotalayan bir uygulama bulunmamaktadır. İstanbul'a gelen yerli ve yabancı turistlerin zaman sınırlarının olması ve gezebilecekleri tarihsel mekanlar hakkında bilgi sahibi olmama ihtimalleri, bu mekanları gezerken hangi sıraya göre gezebileceklerini her zaman kestirememeleri ihtimali üzerine Gezgin Satıcı Problemi üzerinde karınca kolonisi algoritması kullanılarak bu program geliştirilmiştir. Karınca kolonisi algoritması, doğadaki karıncaların izlenmesi ve bu izlenimlerin yorumlanması sonucu gerçek karıncaların davranışlarından esinlenerek geliştirilmiş bir algoritmadır. Karıncaların aralarındaki iletişimi sağlamaları için salgıladıkları feromon adlı sıvı haberleşmeleri açısından en temel madde olarak görülmektedir. Bulundukları alanda en kısa yolu bulmak mantığıyla çalışan yapay karıncalardan faydalanılmıştır. Programımızdaki amaç, bu doğrultuda tarihsel mekanlar arasından yapılan seçimler için en kısa yoldan birbirlerine ve aynı zamanda kullanıcının bulunduğu noktaya GPS yardımı ile bağlantı yaparak rotalamaktır. Microsoft Visual Studio yazılım geliştirme aracı ile Xamarin Cross Platform teknolojisi kullanılarak arayüz geliştirmesi yapılmıştır. Harita üzerinden rotalanmış mekanlar hazırlamış olduğumuz web sitesi ile Google maps üzerinden görüntülenmesi sağlanmıştır.

Anahtar Kelimeler: *Karınca Kolonisi Algoritması, Rotalama, Tarihi Mekanlar, Mobil Uygulama.*

AN MOBILE APPLICATION ON SHORTEST ROUTING OF HISTORICAL PLACES BY USING HEURISTIC ALGORITHMS

ABSTRACT

Today, thanks to rapidly evolving technology users; no longer require the power; to quickly access each time technology of all ages and in a relationship with every group of people is increasing day by day.The technology rapidly advancing. It's to be able to respond to the needs of it's users are constantly evolving and that directs users to the different needs in recent years, the rapidly increasing numbers of mobile devices based on past technological devices to be used much more in everyday life. At first, internet technology has also now apart from the passing of wired to wireless smart phones, tablets, chargers, and more device are lightweight and can be used without limitation venue, next-generation smart phones. There are many applications that we use in daily life. We can go with navigations to an address, you can find the nearest bus stop, you can call a taxi, you can send the mail, do shopping, correspondence, we can do a lot of social platforms, shares in mobile applications. Despite the presence of mobile applications in many areas ,there is no application that gives information about the historical places in Istanbul and also circulates these historical places in more than one way.Time limitations of domestic and foreign tourists coming to Istanbul and they may not have the knowledge of the historical places to visit. According to what order they could visit while visiting these places there is a possibility of their failure to predict. For this reason, the traveling salesman problem using ant colony algorithm on this program has been developed. Ant Colony algorithm is monitoring of ants in nature and the result of the interpretation of the behavior of real ants inspired by this impression improved. To provide communication between the ants secrete pheromones in terms of communications at liquid is regarded as the most basic item. Artificial ants has benefited.They have the logic of finding the shortest path. The purpose of our program, For the choices made in historical sites, the route is routed from the shortest route to each other and to the same point with the help of GPS. Interface development using Microsoft Visual Studio software development tool and Xamarin Cross Platform technology has been done. We have provided you with a web site to view Routed locations via map via Google Maps.

Keywords: *Ant Colony Algorithm, Routing, Historic Sites, Mobile Application*

1 GİRİŞ

Mobil uygulamalar, günümüzde birçok kişinin kullandığı, mobil akıllı telefon veya tabletlere özgü dizayn edilmiş ve kodlanmış programlara mobil uygulama denmektedir. Genel olarak sıklıkla kullanılan IOS ve Android işletim sistemine uygun olarak yazılırlar. Kaba masaüstü cihazlar yerine artık akıllı telefonlar ve tabletlerle dilediğimiz bütün masaüstü yazılımları mobil cihazlara uyumlu olarak yazmak ve dolayısıyla kullanmak mümkün hale gelmiştir.

Akıllı telefonlarla artık çok fazla vakit geçirilmekte, birçok alanda mobil uygulamalardan yararlanılmaktadır. Eskiden lüks olan internet ve akıllı telefonlar neredeyse bir ihtiyaç haline gelmiştir. Çok fazla kullanıcısı olan mobil cihazların aynı doğrultuda çok fazla mobil uygulama çeşitliliğine ihtiyacı vardır. Değişik alanlarda yüzbinlerce sayıda yazılan mobil yazılımlar yine de yeterli gelmemekte daha başka yazılımlara ihtiyaç duyulmaktadır.

İstanbul'a gelen yerli ve yabancı turistlerin tarihsel mekanları gezerken yaşadıkları zorluklar düşünüldüğünde, tarihsel mekan hakkında yeterli bilgi birikimine sahip olunamayışları, gelişen ve değişen çevre şartlarına uyumlu olmayan rehber materyaller, turist rehberi olmadan kendi başına gezebilme isteği, tarihsel mekana ulaşımdaki güçlükler gibi bir çok problem yaşamaktadırlar. Bahsedilen problemler göz önüne alındığında bu alanda bir mobil uygulamaya ihtiyaç duyulduğu anlaşılmaktadır.

Yazılacak mobil uygulamayla İstanbul'a gelen yerli ve yabancı turistler, tarihsel mekanlar hakkında bilgi sahibi olabilecek, pilot olarak belirlenen 34 tarihsel mekan arasından istediklerini işaretleyip, hangi sırayla gezebileceklerini görebilecek veya bulundukları noktadan gezmek istedikleri tarihsel mekan arasında kalan tarihsel mekanları da listelerine ekleyebilecek ve en son başladıkları noktaya geri dönebileceklerdir. Uygulamanın Türkçe dışında İngilizce dil desteği de bulunmaktadır. Seçilen tarihsel mekanlar harita üzerinde ve rotalanmış bir şekilde görüntülenebilecektir.

Bu çalışmada, doğadaki doğal karıncaların hareketlerinin incelenmesi sonucu, yuvalarından çıkıp yiyeceğe giden yollar arasındaki en kısa yolu bulup, bunu koloni şeklinde devam ettiren karıncalardan ilham alınarak oluşturulmuş, yapay karıncaların kullanıldığı, KKA kullanılarak mevcut problem üzerinde benzetim yapılarak uygulanmıştır. Kullanılan bu sisteme Karınca Sistemi denilmektedir. Microsoft Visual Studio 2015 programı üzerinde bulunan Xamarin Cross Platform teknolojisi kullanılarak yazılan bu uygulama android işletim sistemini desteklemektedir.

1.1 Neden bir mobil uygulamaya ihtiyaç duyuldu?

Mobil uygulama, gelişen teknoloji ve ihtiyaçlar için günümüzde hemen hemen herkesin yanında bulundurduğu, akıllı telefon ve tabletler için kodlanmış, tasarlanmış yazılımlardır. Masaüstü cihazlardan internete girişler yerini yavaş yavaş mobil cihazlara bırakmaktadır. Günlük yaşantımızın bir parçası haline gelen mobil cihazlarla artık daha fazla zaman geçirmekteyiz. Mobil uygulamalar en yakın eczaneyi bulmak, sinema filmi izlemek için en uygun salonu seçmek, hava yolları seyahatlerinde elektronik bileti cihaza okutmak, bilmediğimiz bir yerde seyahat ederken istediğimiz noktaya yönlendirilmek, anlık anılar paylaşmak, elektronik postalarımızı kontrol etmek, bankacılık işlemlerimizi yapmak ve daha birçok özellik için hayatımızda önemli bir yer tutmaktadırlar. Bu denli önemli etkileri olan mobil uygulamalar, taşınılabilir cihazlar sayesinde her anımızda yanımızda bulunabilirler. Gelişen teknoloji ile birlikte akıllı telefonlar ve tabletler kolay taşınma ve kullanım imkanı sağlamaktadır. Yaptığım çalışmada kullanıcıların birden fazla noktaya seyahat edecekleri, seçtikleri noktaları değiştirebilecekleri, en son nokta olarak başladıkları yere dönecekleri göz önüne alındığında, bir mobil uygulamanın her an ulaşılabilir olması, kullanıcılar için büyük kolaylıklar sağlayacaktır. Kullanıcıların gidecekleri noktaların listesini sırası ile vermesinin dışında, bir noktadan başka bir noktaya hangi yollardan ve nasıl gidebileceklerinin bilgisini sunmaktadır.

1.2 Uygulamanın Özgünlüğü:

Quick İstanbul adlı uygulama için yapılan literatür çalışmalarında, bu güne kadar turistler için tarihsel mekanların gezilmesi, gezilecek tarihsel mekanların en kısa mesafeyle sıralanması, gezinti sonrası başlanılan yere geri dönülmesi, gezilmek istenen tarihsel mekanlar için resim ve bilgi verilmesi çalışmalarına bir bütün olarak rastlanılmamıştır. Bu uygulamada kullanıcıların tarihsel mekanlara rotalanması için iki ayrı seçenek bulunmaktadır. Birinci seçenekte uygulamada var olan, kullanıcının seçmiş olduğu tarihsel mekanları kendi aralarında, hem harita üzerinde hem liste şeklinde kullanıcıya sunulmaktadır. Kullanıcı istediği herhangi birinden başlayacak şekilde tasarlanmıştır. Başlangıç noktası seçme zorunluluğu yoktur. İkinci seçenekte ise, kullanıcının başlangıç noktası GPS tarafından tespit edilmektedir. Tespit edilen nokta başlangıç noktası olarak kabul edilir. Uygulamada var olan tarihsel mekanlardan kullanıcının gezmek istediği bir nokta seçilir. Kullanıcının bulunduğu başlangıç noktasından, gezmek istediği nokta arasında kalan, belli bir çaptaki alan içerisinde bulunan farklı tarihsel mekanlar kullanıcıya liste şeklinde sunulur ve kullanıcı isterse bu noktalar arasından bir kaçını veya hepsini gezilecek yerler arasına ekler. Seçilen veya seçilmeyen bu nokta veya noktalardan sonra, rotalama yapılır.

1.3 Neden karınca kolonisi algoritması

Bir noktadan başlayıp gidilecek noktalara sadece bir kez uğraması ve tekrar aynı noktaya dönme ihtiyacının olması, bu noktalara uğrarken en kısa yoldan gidip dönmesi, hızlı yanıt vermesi, rota oluşturulurken çıkan sonucun kesin sonuç vermek zorunda olmaması, problemimize olan uygunluğu bu algoritmanın esas seçilme sebeplerinden bazılarıdır. KKA' nın tarihsel mekanlar ile olan ilişkisi, gezilecek her mekana sadece bir defa uğranılacak olması, gezinti yapacak turistin yine gezintiye başladığı yere dönecek olması, kısıtlı zamana sahip olan turistlerin zamanını en verimli bir şekilde kullanması amacıyla gezilmek istenen mekanlar arasında mantıklı bir sıralama yapabiliyor olması kullanılan bu algoritmanın çalışma mantığıyla örtüştüğü görülmektedir.

1.4 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.

2 LİTERATÜR ÖZETİ

Araç rotalama problemleri alanındaki çalışmalar 1950' li yıllarda ortaya çıkmıştır. Özellikle 2003 yılından beri büyük bir aşama kaydettiği göze çarpmaktadır. Aşağıda, genel olarak rotalama ve karınca kolonisi algoritması konusunda gerçekleştirilen çalışma örnekleri yer almaktadır.

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.

Demircioğlu (2009), zaman pencereli araç rotalama problemine sahip bir dağıtım firmasında uygulama yapılarak en uygun dağıtım rotası bulunmaya çalışılmış, zaman pencereli araç rotalama probleminde sezgisel çözüm teknikleri kullanarak, en uygun dağıtım yolları belirlenmiştir. Belirlenen bu dağıtım yollarının dağıtımda geçen süreler ve kullanılan maliyetleri üzerindeki etkileri belirlenmiştir.

Alpaslan (2015), Araç Rotalama Problemleri İçin Matematiksel Modeller Ve Çözüm Yöntemleri adlı çalışmasında, klasik araç rotalama problemi, açık uçlu araç rotalama problemi ve bölünmüş talepli 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.

Akay (2016), gerçek hayatta karşılaşılan, büyük boyutlu personel ve malzeme nakil görevlerinin rota tespiti yapılmıştır. Rota tespit edilirken, uçuş süresini minimize etmek ve böylece maliyetleri düşürmek amaçlanmıştır. Bu amaçla literatürde bulunan; Eş Zamanlı, Topla-Dağıt Araç Rotalama Probleminde kullanılan matematiksel model incelenerek 13 karakollu problemlere kadar optimal sonuçlar bulunmuş, iki aşamalı (Önce rotala, sonra kümele) sezgisel bir

yöntem C++ dilinde kodlanmış ve elde edilen sonuçlar, hem optimal sonuçlarla hem de manuel yapılan hesaplamalarla karşılaştırılmıştır. Edilen sonuçlar, hem optimal sonuçlarla hem de manuel yapılan hesaplamalarla karşılaştırılmıştır.

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 küçükleyecek, en uygun rotayı belirlemektir. Uygulamada Kat edilen mesafenin en küçüklenmesi, tıbbi atıkların toplanmasındaki taşımacılık maliyetlerinin de en küçüklenmesini sağlamak amaçlanmıştır.

Kuram (2016), müşterilerine istekleri doğrultusunda hizmet vermek bunun yanında toplam araç taşıma maliyetlerini en aza indirmek amacıyla araç rotalama probleminin özel bir durumu olan zaman pencereli araç rotalama problemi çalışılmıştır. Bu rotalama sürecinde kesin çözüm yöntemleri hızlı cevap verememesinden dolayı en iyi çözümü garanti etmeyen fakat en iyi çözüme kısa sürede ve yakın çözümler veren popülasyon tabanlı sezgisel yöntemlerin kullanıldığı bir çalışma yapılmıştır.

Atasagun (2015), literatürde henüz çalışılmamış olan ve topla-dağıt araç rotalama probleminin çeşitlerinin en genel hali olarak kabul edilen eş zamanlı topla-dağıt araç rotalama problemi ile zaman bağımlı araç rotalama problemlerini birlikte çözebilmek için zaman bağımlı eş zamanlı topla-dağıt araç rotalama problemi için bir matematiksel model geliştirilerek literatürdeki mevcut test problemleri ile deneysel çalışmalar yapmış ve yorumlamıştır.

Gangal (2015), kablosuz Algılayıcı Ağlarda, çok duraklı rotalama yöntemlerinden Karınca Koloni Algoritmalı Rotalama ile Enerji Etkin Çok Duraklı Rotalama yöntemleri incelenmiştir. Yapılan benzetimlerde, bu iki yöntemin enerji tüketimleri, ağ ömürleri ve baz istasyonuna ulaştırabildikleri toplam paket sayıları karşılaştırılmıştır. Her iki yöntemde de Heinzelman'ın Mikrosensör Ağlar için önerdiği enerji modeli kullanılmıştır. Karınca Koloni

Algoritmalı Rotalama yönteminde, baz istasyonu konumunun, buharlaşma katsayısının, düğüm menzillerinin ve rota güncelleme periyodunun ağ ömrüne etkileri araştırılmıştır. Ayrıca yeni bir karınca paket yapısı önerilmiştir. Karınca hafızasının uzunluğunun (paket boyu), baz istasyonuna ulaştırılan paket sayısını önemli derecede etkilemediği görülmüştür.

Eldem (2014), Karınca kolonisi optimizasyonu yönteminin ürettiği sonuçların iyileştirilmesi için, bulduğu çözümleri hiyerarşik bir yapıda, ayrıklaştırılmış Parçacık sürü optimizasyonu yönteminin iyileştirmesi üzerine bu iki yöntemin birlikte çalışabileceği ele alınmıştır. Literatürde optimizasyon problemlerinin çözümünde sıkça kullanılan test fonksiyonlarından Gezgin Satıcı Problemi (Traveling Salesman Problem - GSP) nin çözümünde, önerilen yöntem kullanılmıştır. Bu şekilde sıralı şekilde optimum başlangıç yolları sonuçlarını veren KKO algoritmasının yaptığı çıkarımların, PSO tarafından optimize edilmesi sağlanmıştır. Ayrıca bu iki yöntemin hiyerarşik bir yaklaşımla komşuluk operatörleri yardımıyla birlikte çalışmaları test edilerek performans sonuçları verilmiştir. 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.

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 eniyiye takılmadan, evrensel eniyi çözüme ulaşılma olasılığının artırılmasıdır. Yeniden yapılandırılan bu yöntemler Gezgin Satıcı Problemi ile TSPLIB de var olan bir takım karşılaştırma problemleri ile Maximum Minimum KSA üzerinde denenmiş ve alınan sonuçlar yansıtılmıştır. Bu çalışmada, çaprazlama mekanizmalarının, KKE algoritmalarının performansını iyileştirdiğini göstermiştir.

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.

Çalışkan (2011), karınca kolonisi optimizasyonu ile araç rotalama probleminin maliyetlerinin kümeleme tekniği ile iyileştirilmesi adlı tezinde özetle, araç rotalama problemine karınca kolonisi optimizasyonu k ortalama kümeleme tekniği ile birlikte uygulanarak taleplerin toplam gerçekleşme süresi, toplam mesafe ve kullanılan toplam araç sayısının değişimleri izlenmiştir.

Ekizler (2011), çalışmasında, araç rotalama problemlerinden, Kapasite Kısıtlı Araç Rotalama Problemleri incelenmiştir. Metasezgisel yöntemlerden biri olan Karınca Kolonisi Sistemi kullanılarak çözüm önerisi sunulmuştur. Kullanılan bu yöntem literatürde var olan test problemlerine uygulanmış ve elde edilen çözümler problemlerin bilinen en iyi çözümleri ile karşılaştırılmıştır.

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.

3 TARİH NEDİR

Tarih, türlü insan topluluklarının daha evvelki yaşadıklarını sebep sonuç ilişkisi içerisinde mekânını ve zamanını doğru olarak belirterek ve bunları belgelere isnat ederek tarafsız bir şekilde tetkik eden bir bilim dalıdır. Tarih geçmişteki yaşanmışlıklara ait bilgilerin keşfedilmesi, düzenlenmesi, bir araya getirilmesi ve aktarılması bilimidir. Tarihi bilgi, geçmişteki yaşanmışlıklarla ilgili bütün bilgilerin, durumların yaşanmış olduğu zamanın şartları göz önüne alınarak, mümkün olan en objektif bir şekilde aktarılması ile meydana gelir. Tarih, meydana gelen olayların bir daha yaşanabilmesi gibi bir ihtimal olmadığı için diğer bilim dalları gibi deney ve gözleme dayanamaz. Bu bilimde her şey bir defa meydana gelir ve sonlanır [1].

Tarih, insanoğluna bir aitlik hissiyatı vererek, kişilerin kendilerinin, mensubu oldukları toplumlukların ve insanoğlunun, kültürel yapılarını ve geçmişten gelen mirası anlamlandırmasına yardımcı olur. İnsanoğlunun mensupları olarak nereden gelip nereye gittiklerini ve mevcut zamanda bulundukları yerdeki yaşadıkları dünyanın, bir ferdi olarak içinde yaşadıkları toplumun, milletin ve insanların geleceğinin hazırlanabilmesi için hangi zorlu ve sıkıntılı durumlardan geçerek geldiklerini, neyin ve kimlerin nasıl bir bedel ödediklerini kolayca anlamalarını sağlar (Turan ve Ulusoy, 2013: 141).

3.1 Bir yere tarihsel mekan denilebilmesi için neler gereklidir

Bir yerin tarihsel mekan olarak kabul edilebilmesi için En geniş ifadesiyle "etrafımızı kuşatan geçmişe ait bütün unsurlardır" (Anderson ve Moore'dan akt, Safran ve Ata, 2006: 53). Tarihsel çevreyi oluşturan etmenler, müzelerde sergilenen eserler ve tarihsel yapılardır. "İnsanlığın, geçmiştin günümüze değişen ihtiyaçlara cevap vermek için geliştirdiği araç ve gereçler, binalar yollar da tarihsel çevrenin unsurlarıdır." (Safran ve Ata, 2006: 53).

Tarih için bir malzeme gurubundan söz etmek gerekirse, heykeller, zafer taçları, abideler, lahitler ve mezar taşlarını örnek olarak gösterebiliriz (Özçelik, 2011:

86). Tarihi çevre, tarih öğretiminde kullanılabilecek önemli unsurlardan birsidir. "Geçmişten gelen ve bizi saran her şey olarak tanımlanan tarihi çevre; binalar, açık alanda bulunan tarihi mekânlar ve taşınabilir tarihi materyallerden oluşmaktadır." (Copeland, 1991: Akt. Demircioğlu, 2010: 131).

Tarihi mekân dendiğinde ilk olarak akla gelen, geçmişte yaşamış insan topluluklarının bırakmış oldukları yapılar veya yaşadıkları çevre akla gelmektedir. "Bu tür mekânlar kale, mezar, medrese, türbe, ev, cami ve çeşme gibi insanların geçmişte etkileşimde bulunduğu yapılar ya da yerler olarak tanımlanabilir." (Ilgaz ve Örten, 2015: 287).

"Tarihi çevre; 2863 sayılı Kültür ve Tabiat Varlıklarını Koruma Kanunu dikkate alınarak şu şekilde tanımlanabilir: Tarih öncesinden günümüze kadar gelen çeşitli medeniyetlerin ürünü olup, yaşadıkları devirlerin sosyal, ekonomik, mimari ve benzeri özelliklerini yansıtan kent ve kent kalıntılarının; önemli tarihi olayların yaşandığı yerlerin ve yer üstünde, yer altında veya su altında bulunan korunması gerekli taşınır ve taşınmaz kültür ve tabiat varlıklarının oluşturduğu çevredir." (Alkış ve Oğuzoğlu, 2005: 348).

Bu çalışmada İstanbul ilindeki 34 farklı tarihi mekan kullanıldı. Seçilen tarihi mekanlar, Mihrimah Sultan Camii, Hırka-i Şerif Camii, İstiklal Caddesi, Sent Antuan Kilisesi, Galata Kulesi, Kıztaşı, Bozdoğan Su Kemeri, İtfaiye Müzesi, Vefa Bozacısı, Laleli Sebili, Süleymaniye Camii, Yeni Camii, Mısır Çarşısı, Beyazıt Kulesi, Mirahur Hasan Ağa Çeşmesi, Çinili Han, Büyük Postane, Gotlar Sütunu, Gülhane Parkı, Topkapı Sarayı, Cağaloğlu Hamamı, Aya İrini, Ayasofya Müzesi, Yerebatan Sarnıcı, Çemberlitaş Sütunu, Ayasofya Hürrem Sultan Hamamı, Alman Çeşmesi, Dikilitaş, Sultan Ahmet Camii, Tarihi Şifa Hamamı, Küçük Ayasofya, İstanbul Arkeoloji Müzeleri, Fatih Sultan Mehmet Camii ve Büyük Çarşı'dır [5].

3.2 Bir tarihi mekanı gezmenin zorlukları

Rotalama problemi: Turistlerin bulundukları yerden gitmek istedikleri tarihi mekana nasıl gidebilecekleri, gitmek istedikleri yerler birden fazla ise bu yerler arasında nasıl bir sıralama yapacakları ve bu yaptıkları seçimin en kısa rota olup olmadığının her zaman tahmin edilemeyiş olması, gitmek istedikleri tarihi

mekanların sayısı arttıkça rota seçiminin zorlaşması, rotalama açısından ciddi problemler teşkil etmektedir.

Gezilecek tarihi mekanlar hakkında bilgi sahibi olunamaması: Gezilecek tarihi mekanın görsel olarak ve tarihi geçmişi, yapısı açısından bilgi sahibi olabilmek, gezilecek tarihi mekanların daha kolay bulunabilmesi ve gezilebilmesine olanak sağlar. Turistler gezilecek tarihi mekan hakkında ne kadar çok bilgi sahibi olurlarsa o kadar işleri kolaylaşacak ve daha bilinçli gezme imkanı bulacaklardır.

Güncel olmayan gezi rehberleri: Turistlerin ellerindeki mevcut haritalarıyla bir tarihi mekana gitmeye çalışmaları ve bu tarihi mekan hakkında bilgi sahibi olmaya çalışmaları, güncel olmayan haritalarda ve güncel olmayan yazılı kitaplarda turistlerin işini zorlaştıracak, yanlış yönlendirecektir.

Turist rehberiyle gezmek istemeyen turistler: Gezi turlarıyla gezmeyen, tarihi mekanları kendi başına gezmek isteyen turistlerin, anlık olarak konumunu bulması, zamansal açıdan kısıtlarının olması, kısa zamanda gezebilecekleri tarihi mekanların sayısını kısıtlamaktadır. Bu kısıtlı zamanda yapılacak yanlış rota seçimleri gezilebilecek tarihi mekan sayısını en aza indirecektir.

3.2.1 İstanbul'da tarih

İstanbul'un, bir yerleşim yeri olarak kullanıldığının tarihi 300 bin, kentsel olarak kullanıldığı tarihi yaklaşık olarak 3 bin, başkentlik olarak kullanıldığı tarihi 1600 yıla kadar uzanır. Avrupa ile Asya kıtalarının kesişim noktasında bulunan bir dünya şehridir. İstanbul, yüzyıllar boyunca birçok uygarlık ve kültürlere ev sahipliği yapmıştır ve aynı zamanda asırlar boyu türlü din, dil ve ırktan insan topluluklarının birlikte yaşadığı komplike şeklini muhafaza etmiş ve bu tarihi dönemlerde benzersiz bir şekil almıştır [2].

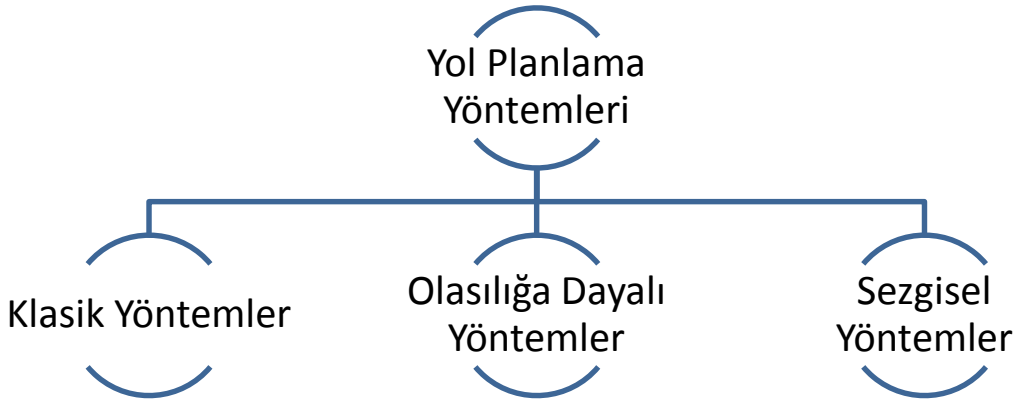
Eski İstanbul olarak da adlandırılan tarihi yarımada ise İstanbul'un ticaret ve yönetim merkezi konumundadır. Yarımada üzerinde tarih boyunca oluşmuş fiziksel, kültürel ve toplumsal değişimler, İstanbul'un günümüzdeki kimliğinin oluşmasını sağlamıştır. Bölge İstanbul'un en zengin kültürel, sosyal ve mimari mirasa sahip bölgelerinden biridir. Aynı zamanda dünya kültür mirasının da önemli bir parçasıdır.

3.3 Tarihsel Mekanların Seçilmesi

Uygulamamızda kullanmak üzere belirlediğimiz tarihsel mekanların seçimi yapılırken, özellikle Fatih Tarihi Yarımadası üzerinde bulunan tarihsel mekanlar seçilmiştir. Bu bölgenin seçilmesindeki amaç İstanbul' a gelen yerli ve yabancı turistlerin İstanbul' da en çok gezdikleri bölge olması sebebiyledir. Bu bölgede bulunan tarihsel mekanlar turistlerinde rahatça gezebilmesi açısından çok kapsamlı bir ulaşım ağına sahiptir. Tarihsel mekanlar belirlendikten sonra uygulamamızda kullanılacak Karınca Kolonisi Algoritması için gerekli olan koordinatlar Google Map haritası üzerinden tespit edilmiştir. Her tarihsel mekanın enlem, boylam ve fiziksel adres bilgileri tespit edilerek kullanılmıştır. İlgili ekranlarda rotalama yapılırken, harita üzerinde sadece bu mekanların işaretleri gösterilmiş ve kullanıcıları için kolaylık sağlanmıştır.

4 ROTALAMA NEDİR

Bir noktadan başka bir noktaya ulaşması veya ulaştırılması gereken kişileri, hizmeti, ürünleri gidilecek noktalar arasındaki mesafeyi en aza indirip yönlendirme işlemidir. Bu yönlendirme sürecinde bazı sorunlarla karşı karşıya kalınmaktadır. Bunlar, servis süresi kısıtları, araç kapasite kısıtları, araç deposu yakıt kısıtları, mevcut ürün kullanımlarının yeteri kadar planlı bir şekilde paylaştırılamaması karşılaşılan bazı problemler olarak sıralanabilir. Rotalama problemlerinin çözümü için bir çok farklı yöntem kullanılmaktadır. Bu yöntemler en genel anlamda Şekil 4.1'de görüldüğü gibi 3 başlık altında toplanmıştır.

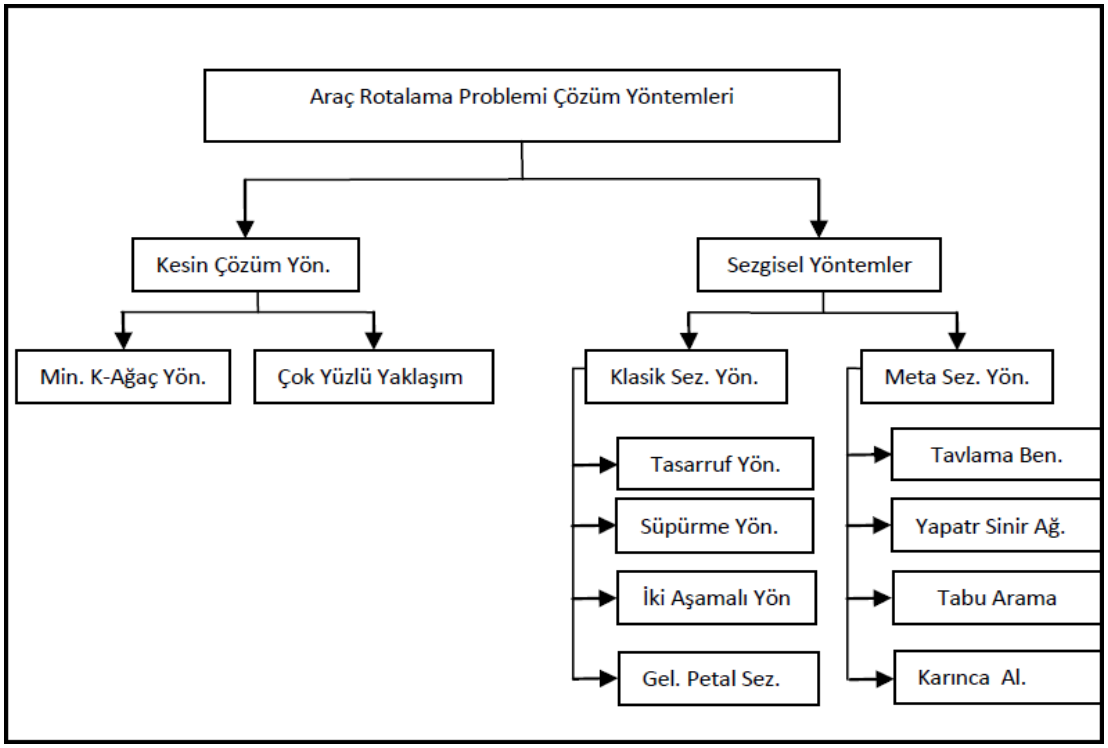


Şekil 4.1: Yol Planlama Yöntemleri

Günümüzde rotalama birçok alanda uygulanmaktadır. Bunlar nakit para veya ziynet eşya dağıtımı, çöp toplama, petrol ürünleri dağıtımı, postalama hizmetleri, havayolu üzerinden kargo ve yolcu taşınması, atık yağ toplama, servis hizmeti veren işyerlerindeki yolcuların taşınması gibi gerçek hayatta birçok örneği bulunmaktadır. Günümüzde, bahsettiğim bu uygulama alanlarındaki rotalama sorunları bazen herhangi bir yardım almadan veya

planlanmadan yapılabildiği gibi, teknolojiden faydalanılarak ta yapılmaktadır. Teknolojiyi kullananlar birçok alanda fayda sağlamış ve yüksek oranda tasarruf sağlayarak hem daha çevreci hem daha ekonomik hem de zamandan maksimum kazanç elde etmişlerdir.

Bu alanda en mantıklı rotalamalar üzerine yapılan çalışmalarda bazı sezgisel algoritmalar kullanılmış ve mevcut sorunlar eniyilenerek bazı sonuçlar elde edilmiştir.



Şekil 4.2: Arp Çözüm Yöntemleri [6]

Araç rotalama problemleriyle ilgili kullanılan algoritmalar ve eniyileme örnekleri Şekil 4.2' de gösterilmektedir. Bu yöntemlerin bazıları sırasıyla açıklanacaktır.

4.1 Algoritma çeşitleri

Belli bir problemin çözülmesi için izlenen sıralı yollar bütününe algoritma denir.

Akış diyagramı ise çözülmesi istenen problemin, sorunun çözümü için izlenecek sıralı yolları, işlemleri bir şema ile gösteren grafiksel halidir. Bir yazılım yapılırken, yapılması gereken her işlem adım adım yapılır. Ortada bir problem

vardır ve bu problemi çözmek için çeşitli yollar, basamaklar vardır. Çözümün probleme uygulanabilmesi için bir plan dahilinde adım adım sorunlar işlenerek bir sonuca varılır.

Algoritmalar en geniş anlamda iki başlıkta sıralanır. Birincisi kesin sonuç veren algoritmalarken ikincisi kesin sonuç vermeyen algoritmalardır.

Kesin sonuç veren algoritmalara örnek olarak, doğrusal programlama, dinamik programlama, dal-sınır algoritmalarını(branch-and-bound algorithm) gösterebiliriz.

4.1.1 Çözüm kurucu algoritmalar

Probleme iyi bir çözümü iteratif olarak kurar. Bu algoritmalar, başlangıçta bir çözüm parçasını seçer ve probleme tam bir çözüm elde edinceye kadar adım adım çözüm parçalarını kısmi çözüme ekler. Herhangi bir adımda hangi çözüm parçasının kısmi çözüme Ekleneceğine rassal ya da sezgisel bir kural ile karar verir. Genellikle, her adımda bir sezgisel fonksiyon ile tahmin edilen ve en büyük faydanın söz konusu olduğu çözüm parçası kısmi çözüme eklenir. Bu tür algoritmalar (sezgisel kural kullanan), Açgözlü çözüm kurucu sezgisel (greedy constructive heuristic), Açgözlü sezgisel (greedy heuristic) Bu yöntemlerin en büyük dezavantajı farklı problemler için farklı çözüm kurucu algoritmalara ihtiyaç duyulmasıdır.

4.1.2 Yerel Arama Algoritmaları

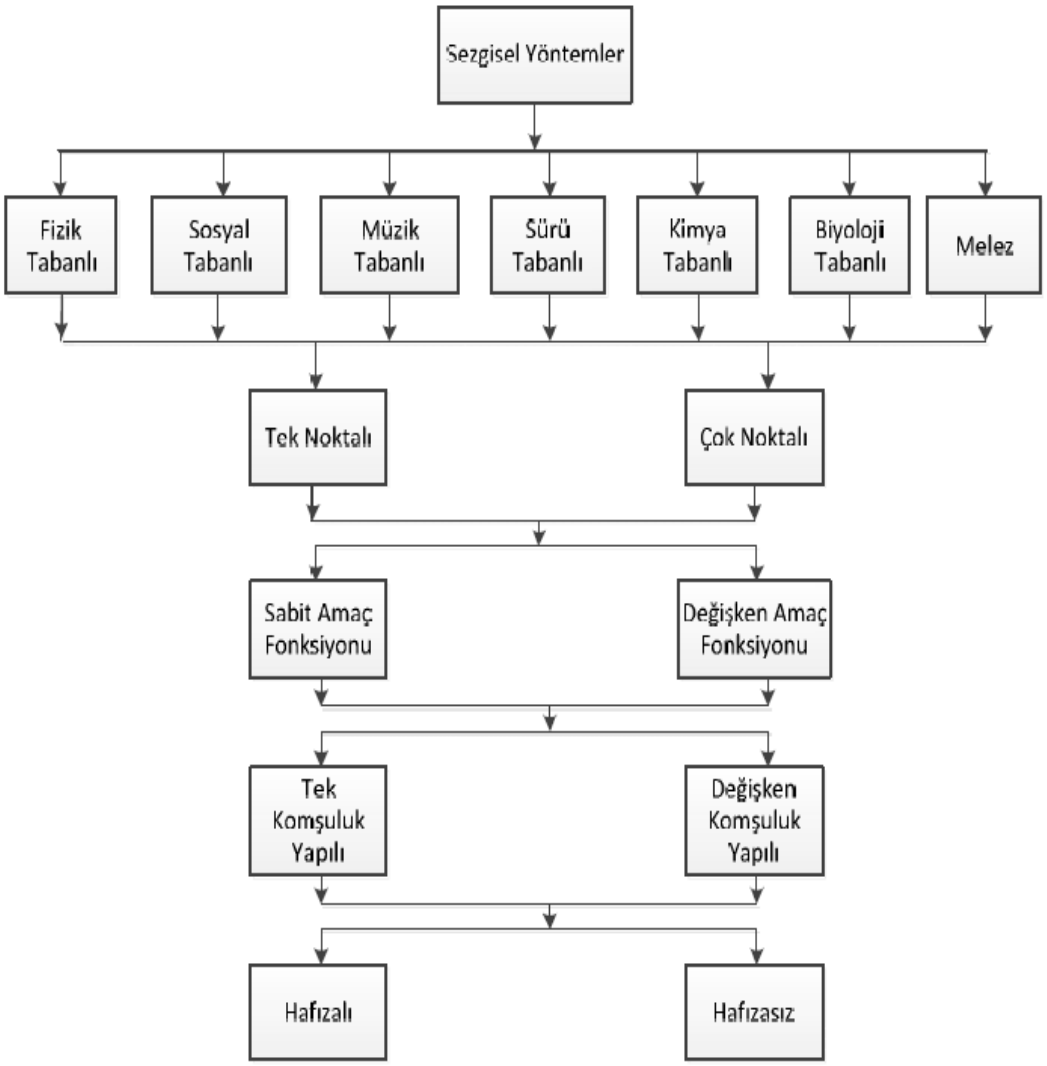
Yerel arama algoritmaları (YAA), en eski ve en kolay eniyileme yöntemlerinden birisidir. YAA, NP-zor sınıfında yer alan problemlere makul zamanda kaliteli çözümü bulmak için kullanılan genel bir yaklaşımdır. Bir başlangıç çözümü ile baslar ve yerel değişmeler ile bu çözümü iyileştirmeye çalışır. Bu algoritmaları uygulayabilmek için aday çözümlerin kümesi S üzerinde "komşuluk yapısının" tanımlanması gerekir. "Komşuluk yapısı", algoritmanın mevcut bir çözümden hareket edebileceği mümkün çözümlerin kümesini tanımlar [4].

Kesin sonuç vermeyen, yaklaşık sonuç veren(sezgisel) algoritmalara örnek olarak, çözüm kurucu algoritmaları, yerel arama algoritmaları(local search), genel amaçlı algoritmaları örnek olarak gösterebiliriz.

4.1.3 Sezgisel algoritmalar

Temel anlamda çalışma mantığında %100 kesinlik verme garantisi olmayan bu algoritmalarda her zaman aynı performansta çalışmaz veya her zaman bir sonuç vermeyi yada sonuçlarında kesinlik sağlamayı garanti etmez fakat problemleri optimize etmek için genel anlamda kullanışlıdırlar. 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.

Genel amaçlı sezgisel optimizasyon algoritmaları, doğadan ilham alarak geliştirilmiştir. Fizik, biyoloji, kimya, müzik, sosyal ve sürü tabanlı olmak üzere altı guruptan oluşurlar. Sürü tabanlı algoritmalarda bazı hayvan çeşitlerinin hareketlerinden ilham alınarak bu algoritmalar türetilmiştir. Ayrıca bunların birleşimi olan melez yöntemler de vardır. Bahsedilen bu yöntemler Şekil 4.3’ te sunulmaktadır.



Şekil 4.3: Sezgisel Yöntemler

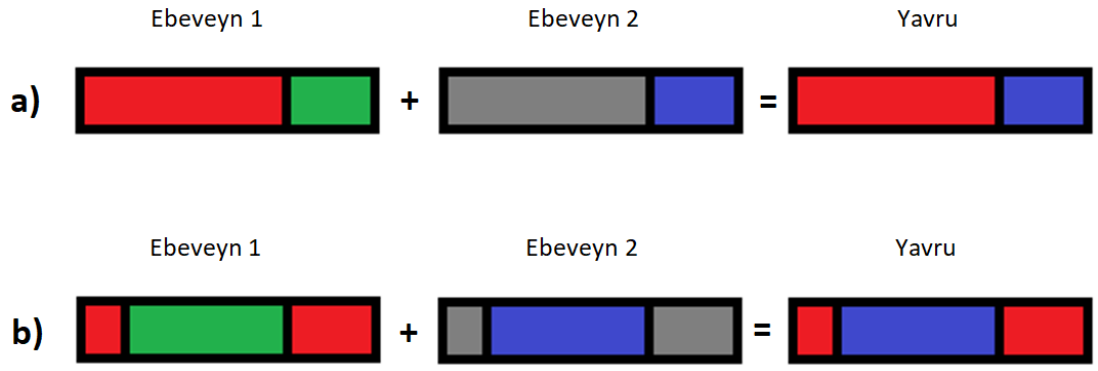
Genetik algoritmalar, Tabu Arama, Tavlama Benzetimi, Değişken Komşu Arama, Karınca Kolonisi Optimizasyonu, Kuş Sürüsü Optimizasyonu, Tırmanış Araması, Açgözlü en iyi öncelikli arama, Yapay Balık Sürüsü Algoritması**,** Yapay Arı Kolonisi, ateş böceği algoritması, Kurt Kolonisi Algoritması**,** Bakteriyel Besin Arama Optimizasyon Algoritması, ateş böceği sürü optimizasyonu, Parçacık Sürü Optimizasyonu sezgisel algoritmalardan bazılarıdır [3].

4.1.3.1 Genetik algoritma

Genetik Algoritma tabii yollarla gelişen yollarla oluşan seçimlerden ilham alınarak Holland (1975) tarafından tavsiye edilmiş bir optimizasyon algoritmasıdır. Genetik biliminde mevcut olan kromozom, çaprazlama ve

mutasyon kavramlarını kullanır. Var olan bu terimleri kullanarak optimizasyon sorunlarına entegre ederek bu problemler için çözümler yaratmaya çalışır.

GA' da optimize edilecek olan sorunlara ait çözüm yollarının her biri ayrı 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 4.4' te görüldüğü gibi Genetik algoritmada tek noktalı ve çift noktalı örnekleri verilmiştir.



Şekil 4.4: 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 (Obitko, 2012).

4.1.3.2 Tabu Arama

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. (Cura, 2008).

4.1.3.3 Benzetilmiş Tavlama

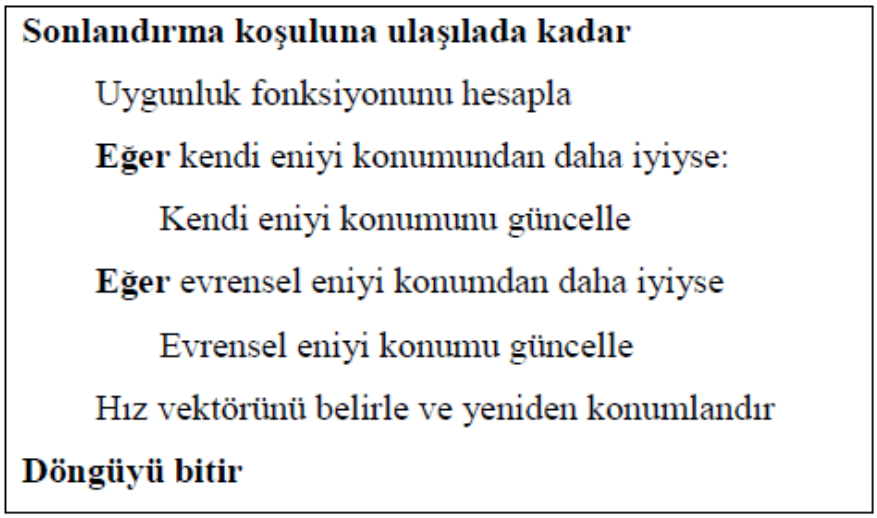
Adını demirin yüksek derecede ısıtılması veya demirin tavlanmasından alan bu algoritmanın amacı, genel anlamda genel optimizasyon elde etmektir. Doğada katı halde bulunan bir maddenin, önce ısıtılarak sıvı hale gelmesi ve ardından belli bir seviyede soğutularak tekrar eski haline dönmesiyle meydana gelen maddenin yeni yapısı bir sistemdeki parçacık olarak yansıtılırsa, bu tavlama işleminden benzetilmiş tavlama yöntemi elde edilmiş olur (Cura, 2008).

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 (Cura, 2008).

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

4.1.3.4 Parçacık Sürü Optimizasyonu

Parçacık sürü optimizasyonu, 1995 yılında, Eberhart ve Kennedy tarafından geliştirilmiş, popülasyon temelli bir sezgisel optimizasyon tekniğidir. Çevrelerindeki insanlarla sosyal anlamda ilişki içerisinde olan kişilerin davranışları incelenerek geliştirilmiştir. Parçacık sürü optimizasyonu, parçacık zekası olarak da isimlendirilir. Çözüm uzayında seçilmiş bir fonksiyonun, genel optimum noktasını bulmak üzere çalışmaktadır. Şekil 4.5' te Parçacık sürü optimizasyonu çalışma adımları gösterilmektedir.

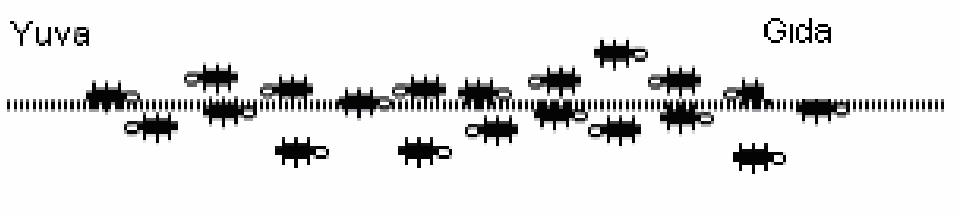


Şekil 4.5: Parçaçık sürü optimizasyonu çalışma adımları

Parçalardan oluşan çözüm elemanları tek başlarına bir şey ifade etmezler. Parçaların birbirleriyle olan etkileşimleriyle ilerleme sağlanır. Parçalar kendi bölgelerinde en iyi noktayı bulmaya çalışır, sonra birbirleriyle iletişim kurup genel bir en iyi çözümde birleşirler. Bu işlem en iyi sonucu bulana kadar devam eder. Çevre bilgisi, parçaların geçmişlerine yönelik bilgi, komşuların deneyimlerine dair bilgi sosyal ağın şekillenmesi açısından büyük önem taşımaktadır (Cura, 2008).

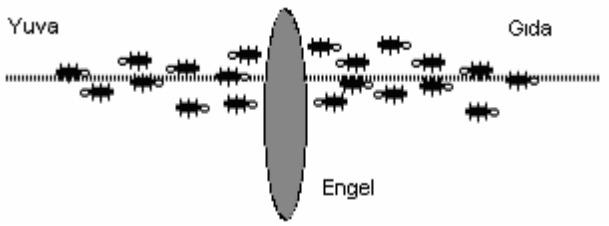
5 KARINCA KOLONİSİ OPTİMİZASYONU

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 5.1' de karıncaların yiyecek bulma amacıyla çıktıkları yoldaki hareketlerini inceleyelim.



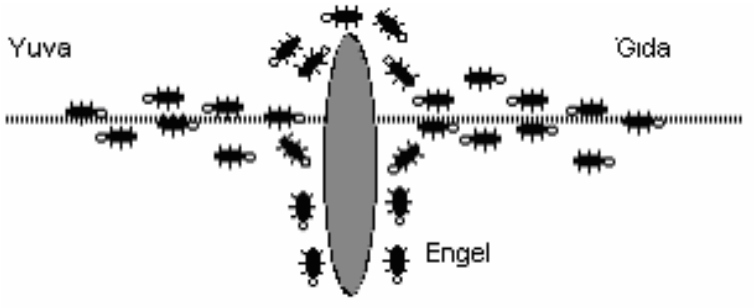
Şekil 5.1: Yuvadan çıkıp besine doğru aldıkları yol

Şekil 5.1' de karıncalar yuvalarından çıkıp herhangi bir engelle karşılaşmadan ileride duran besine ulaşıyor ve yine bu yoldan geri dönüyorlar. Şekil 5.2' de görüldüğü gibi önlerine bir engel konulduğunda önlerindeki yolu daha zor bir hale getirelim.



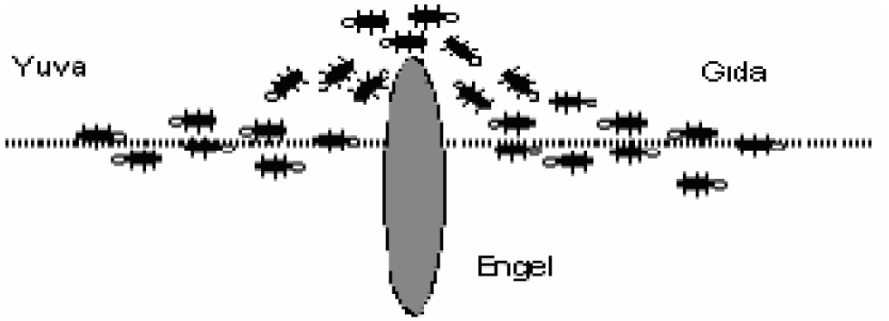
Şekil 5.2: Önlerine engel konulan karıncalar

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 5.3: Engelle karşılaşan karıncaların bir sonraki yol seçim durumları

Şekil 5.3' te 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 5.4' te 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 5.4: 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 (Cordon ve Herrera, 2012: 141-175).

5.1 Karınca Kolonisi Eniyilemesi

KKO, doğal karınca topluluklarının doğadaki yiyecek arama davranışlarından esinlenerek optimizasyon sorunlarına çözüm bulmak amacıyla üretilmiştir (Dorigo, Birattari ve Stützle, 2006: 28-39).

Önceki senelerde KKO ile bir çok farklı alanda yapılan çalışmaların bir kısmı Çizelge 5.1'de gösterilmiştir.

Çizelge 5.1: KKO uygulama alanları ve önceki senelerde 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

5.1.1 Parametrelerin belirlenmesi

Karınca kolonisi optimizasyonunda ilk adım 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.

5.1.2 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.

5.1.3 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.

5.1.4 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 (Aydın, 2011: 113).

5.1.5 Feromon izi 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.

5.2 Karınca Kolonisi Optimizasyonu Algoritmaları

Literatürde bu güne kadar birçok karınca kolonisi optimizasyonu algoritması kullanılmıştır. İlk olarak kullanılanı Dorigo tarafından 1996 'da Karınca Sistemi adıyla, Gezgin Satıcı Problemi için kullanılmıştır (Dorigo et al., 1996).

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.

5.2.1 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.

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

Karınca Sisteminde feromon miktarı formül 1'e göre güncellenmektedir. Formüldeki T_{ij} i ve j noktaları arasında bulunan yoldaki feromon miktarını göstermektedir. Formüldeki m kullanılan karınca sayısını, ρ feromon miktarının buharlaşma katsayısını göstermektedir. $\Delta T_{ij}{}^{k}$ 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 yakşalması durumunda ise karıncaların önceki denemelerinden sağlanan bilgiler tekrarlanacak ve karıncalar sürekli aynı yolu tercih edeceklerinden, yeni yolların denenmesi zorlaşacaktır.

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

$\Delta T_{ij}{}^{k}$ ise formül 2 ye göre hesaplanmaktadır. Bu formülde L_k 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 olasılıksal olarak yaparlar. Feromon miktarının güncellenmesinden önce karıncalar gidecekleri yolu seçer ve bulundukları noktadan hangi farklı noktaya gideceklerini belirlerler.

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

Kaçıncı karıncanın i noktasından çıkıp j noktasını seçme olasılığı formül 3 e göre hesaplanmaktadır. Formüldeki $p_i{}^{k}{}_j$ i noktasından sonra seçilen noktanın j olma ihtimalini belirtmektedir. Ω ise i noktasından çıktıktan sonra seçilebilecek en yakın listesini ifade etmektedir. Formüldeki T_{ij} i ve j noktaları arasında bulunan yoldaki feromon miktarını göstermektedir. η_{ij} 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.

$$\eta_{ij} = \frac{1}{d_{ij}} \tag{4}$$

η ise formül 2 ye göre hesaplanmaktadır. Formüldeki d_{ij} 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 5.5'te 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 5.5: Karınca sistemi algoritmasının adımları

5.2.2 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 (Dorigo et al., 1996).

5.2.3 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 güzergahta ki feromon miktarını formül 5 ’e göre güncelliyor olmasıdır.

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

Formülde kullanılan φ simgesi, feromon sıvısının azaltılma katsayısının kaç olacağını ifade etmektedir. . Formüldeki T_{ij} i ve j noktaları arasında bulunan yoldaki feromon miktarını göstermektedir. T_0, 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 güzergahlar da mevcut feromon sıvısını azaltma istekleri, aslında gidilebilecek yol alternatiflerini arttırmaya yöneliktir. Karıncaların bu davranışları mevcut çözüm denemesinde daha önceden farklı karıncaların kullandığı güzergahları 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.

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

Karınca kolonisi sistemi algoritmasındaki feromon sıvısı güncellemesi formül 6 'ya göre yapılmaktadır. Formülde bulunan $\Delta T_{ij} = 1 / L_{en\ iyi}$ olarak hesap edilmektedir. $L_{en\ iyi}$, 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 $q_0 \in [0,1]$ değerinden küçük veya eşit ise ($q \leq q_0$) ihtimal dahilinde bir tercih yapılmaz. Bu durumda mevcut komşulardan en yüksek $T_{ij} . \eta^{\beta}_{ij}$ değerine sahip olan seçilir. Şayet $q > q_0$ ise, Karınca Sistemi'nde kullanıldığı gibi bu algoritma içinde aynı olasılık sal nokta belirleme formülü devreye girer.

5.2.4 Sıra- tabanlı karınca sistemi

Bu algoritmada, Elitist karınca sisteminde kullanılan yöntemin dışında, çözüm için yapılan denemeler sonrasında, m sayıda karıncaya ait güzergahlar, 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 (Bullnheimer et al., 1999).

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

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

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

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

Formüldeki μ, sırlama indeksini ifade eder. ΔT^{μ}_{ij}, bu çözüme ait sıralamadaki μ. karıncanın i ve j noktaları arasına salgıladıkları feromon seviyesini belirtir. L_{μ} 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 güzergahları ortaya çıkarma işlemi daha etkin bir şekilde çalışmaktadır.

5.2.5 En iyi- 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 (Cordon et al., 2000).

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 güzergah en iyi çözümler arasında gösterilerek feromon sıvısı

seviyesi arttırılır. Bu güzergah 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.

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

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

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

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

Formülde verilen *Nit*, algoritmanın çözümü için yapılan deneme miktarını göstermektedir. it_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 $T_{threshold}$ 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.

5.2.6 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 dahil edilebilir. Aslında bu algoritma iki yaklaşımı da melez olarak kullanabilmektedir.

Bu algoritmada da yol üzerindeki feromon sıvısı miktarının bir sınırı vardır. Bu sınır T_{min} alt sınır ve T_{max} 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 T_{max} olarak belirlenir (Stützle and Hoos, 2000).

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

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

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

Formül 14' te bulunan T_{ij}^{*}, yapılan feromon sıvısı düzeltme işleminden sonra i ve j noktaları arasında bulunan yoldaki feromon sıvısı miktarını göstermektedir. $T_{ij}(t)$, ise yapılan feromon sıvısı düzeltme işleminden önce i ve j noktaları arasında bulunan yoldaki feromon sıvısı miktarını göstermektedir. $\delta = 1$ eşitliği olduğu durumda yol üzerinde bulunan bütün feromon sıvısı miktarı T_{max}' a eşitlenir. Bu durumda feromon sıvısı güncellemesi ilklenmektedir. $\delta = 0$ 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 UYGULAMA

Uygulamamız Visual Studio 2015 yazılım geliştirme aracı üzerinde Xamarin Cross Platform teknolojisi ile birlikte C# dili kullanılarak yazılmıştır. Xamarin Crosss Platform kullanılmasının sebeplerinden biri, IOS ve Windows mobil uygulamalarına destek vermesi ve bu uygulamanın ileriki zamanlarda bu platformlarda da uygulanabilir olmasıdır. Uygulama android işletim sistemi üzerinde çalışacak şekilde hazırlanmıştır. Uygulamada bulunan harita ekranı IIS üzerinde publish edilmiş web sitesinden yararlanılarak gösterimi sağlanmıştır. HTML ve JavaScript kodlarının kullanılarak Google Api servislerinden yararlanıldığı bir web sitesi tercih edilmiştir.

Xamarin mobil uygulama yazılımı açısından piyasada yeni bir teknoloji olduğu için Google Api'nin sunmuş olduğu servisleri kullanımı için esnek bir yapıya sahip değildir. Esnek olmadığından dolayı harita ekranını kullanırken bir web sitesi üzerinden gösterim tercih edilmiştir. Web sitesi MVC (Model View Controller) teknolojisinden yararlanılarak C# dilinde yazılmıştır.

QuickİstanbuI uygulamasında 2 sınıf oluşturulmuştur. Birincisi karınca sınıfı ikincisi ise tarihsel mekan sınıfıdır. Bu sınıflar Karınca Kolonisi Algoritması kullanılması esnasında bize kolaylık sağlamaktadır. Bu sınıflar ve sınıflara ait özellikler aşağıdaki gibidir:

```
public class Karinca
    {
        private int simdikiLokasyon;
        private int siradakiLokasyon;
        public int tur_numarasi;
        public List<int> karincaNeredeListesi;
        public List<int> turListesi;
        public double gidilenMesafe;
```

```
        public Karinca()
        {
            simdikiLokasyon = 0;
            siradakiLokasyon = 0;
            tur_numarasi = 0;
            karincaNeredeListesi = new List<int>();
            turListesi = new List<int>();
            gidilenMesafe = 0;
        }
        public Karinca(int baslangicYeri, int sehirSayisi)
        {
            simdikiLokasyon = baslangicYeri;
            siradakiLokasyon = 0;
            tur_numarasi = 0;
            karincaNeredeListesi = new List<int>();
            turListesi = new List<int>();
            gidilenMesafe = 0;
            for (int k = 0; k < sehirSayisi; k++)
            {
                karincaNeredeListesi.Add(0);
                turListesi.Add(0);
            }
        }
        public void toplam_mesafeyi_guncelle(double mesafe)
        {
            gidilenMesafe += mesafe;
        }
        public float UzaklikOlcumu(PointF a, PointF b)
        {
            return (float)Math.Pow((Math.Pow(a.X - b.X, 2F) + Math.Pow(a.Y
- b.Y, 2F)), .5F);
```

```
    }
    public void simdiki_lokasyonu_setle(int lokasyon)
    {
        simdikiLokasyon = lokasyon;
    }
    public void siradaki_lokasyonu_setle(int siradaki)
    {
        siradakiLokasyon = siradaki;
    }
    public int simdiki_lokasyonu_getir()
    {
        return simdikiLokasyon;
    }
    public double uzakligi_getir()
    {
        return gidilenMesafe;
    }
    public void uzakligi_sifirla()
    {
        gidilenMesafe = 0;
    }

}
public class Mekan
{
    PointF lokasyon;
    public Mekan()
    {
        lokasyon = new PointF(0, 0);
    }
    public Mekan(PointF p)
```

```
        {
            lokasyon = p;
        }
        public void lokasyonu_setle(int x, int y)
        {
            lokasyon.X = x;
            lokasyon.Y = y;
        }
        public PointF lokasyonu_getir()
        {
            return lokasyon;
        }
        public string Ad { get; set; }
        public int Id { get; set; }
        public bool Checked { get; set; }
    }
```

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:

1) karinca_listesi: Algoritmamızda kullanılan her karınca bu listede tutulmaktadır. Her karıncanın lokasyonu ve kaç tur yapacağı bu kısımda tutulur.
2) mekan_listesi: Algoritmamızda kullanılacak her tarihsel mekanın bu listeye eklenmesi gerekmektedir. Tarihsel mekan listesini ve bu noktalara ait lokasyonları barındırır. Lokasyonların isimlerini verilen spesifik Id' lerini ayrıca tarihsel mekanın seçilip seçilmediğinin kontrol edildiği boolean değerini barındırır.
3) 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.

4) 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.

5) mesafeler: Seçilen tarihsel mekan sayısı kadar değer alır. Her tarihsel mekanın diğer noktalarla arasında bulunan mesafelerin ölçülüp atandığı değerlerdir.

6) feromonlar: Seçilen tarihsel mekan sayısı kadar değer alır. Başlangıçta her feromon, 1/tarihsel mekan sayısı cinsinden değer alır. Algoritmanın gidişatına bağlı olarak bu değerler güncelleştirilebilir.

7) mekan_sayisi: Algoritmanın uygulanacağı, seçilen toplam tarihsel mekan sayısını belirtir.

8) Karinca_sayisi: Yapılan testler sonucunda toplam 34 adet tarihsel mekan için 90 adet karıncanın yeterli olduğu görülmüştür.

9) 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.

10) 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 güzergahını kullanacağından algoritmada durağanlığa sebep olacaktır.

11) 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.

12) 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.

13) iterasyonlar: 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.

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

Uygulamamız iki kısımdan oluşmaktadır.

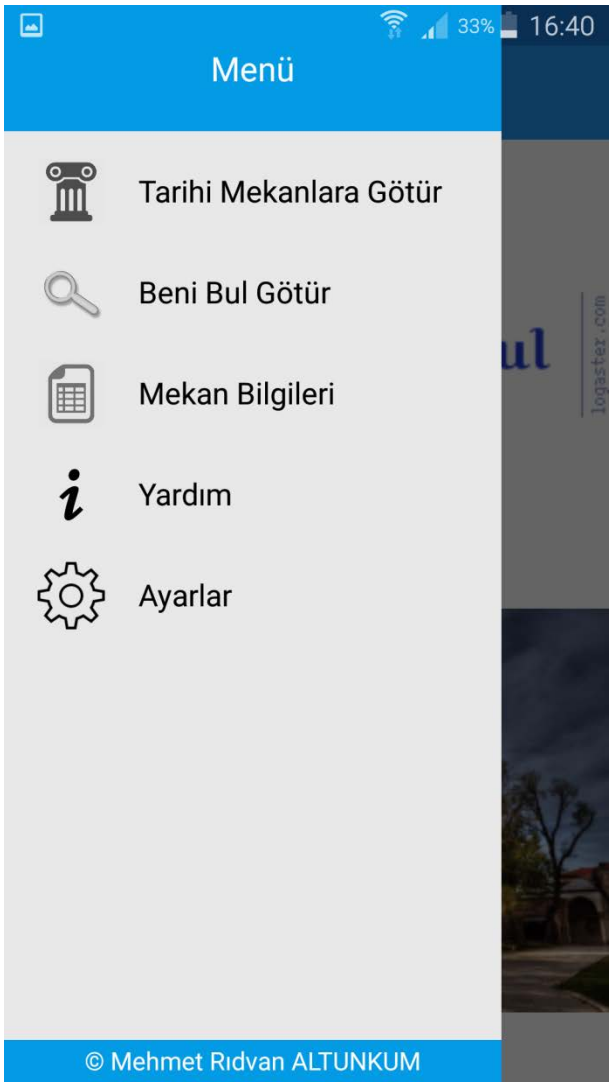
1) Xamarin platformu arayüz çalışması
2) Google Api kullanılarak web sitesinin kodlanması

6.1 Xamarin platformu arayüz çalışması

Quick İstanbul adlı uygulamamız 5 menüden oluşmaktadır. Bu menüler ve menülere ait görseller bu kısımda gösterilmektedir.

Şekil 6.1: Uygulama giriş ekranı

Şekil 6.1' de gösterildiği gibi, uygulamanın giriş ekranı kısmında uygulamanın ismi Quick İstanbul, logosu ve tarihsel mekanlara ait görseller bulunmaktadır.



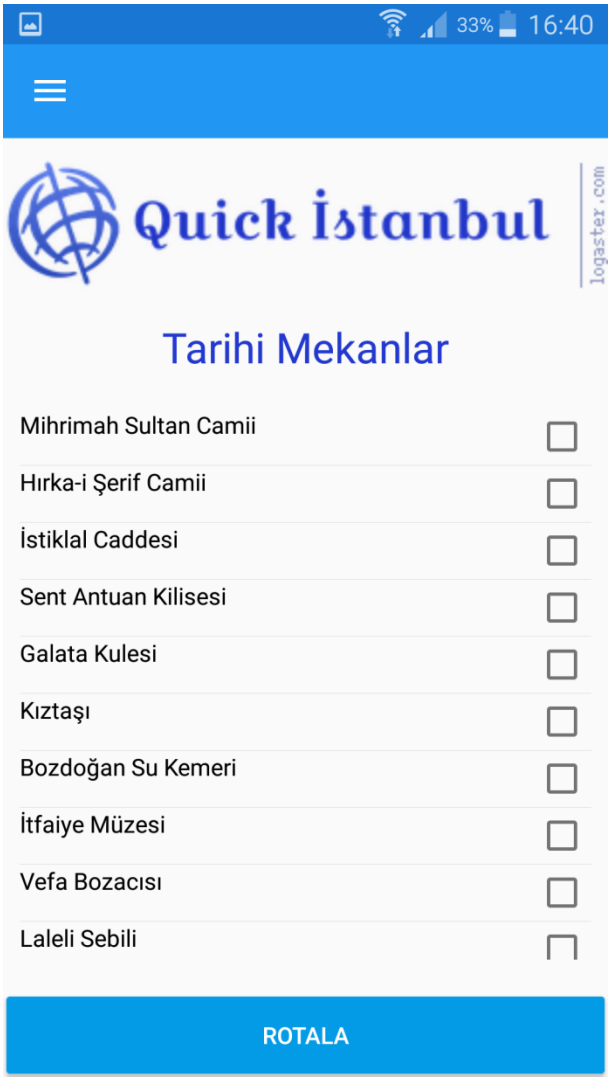
Şekil 6.2: Açılır Kapanır Menü

Şekil 6.2' de gösterildiği gibi, giriş ekranında sol üst köşede bulunan buton ile açılır kapanır menüye erişim sağlanmaktadır. Bu butona basıldığında, Tarihi Mekanlara Götür, Beni Bul Götür, Mekan Bilgileri, Yardım ve Ayarlar menüleri görünmektedir.

6.1.1 Tarihi Mekanlara Götür

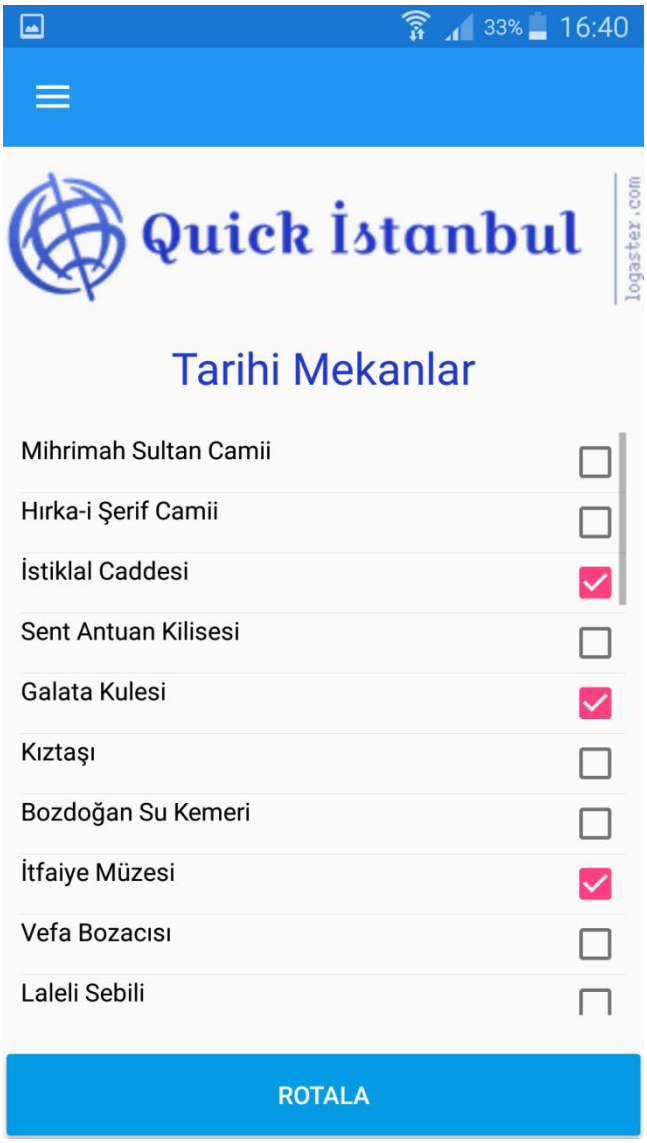
Bu kısımda, İstanbul ilinde bulunan ve pilot olarak seçilen 34 adet tarihsel mekanın listesi görüntülenmektedir. Her bir mekanın yanında checkbox'lar bulunmaktadır. Tarihsel mekanları gezmek isteyen kullanıcı, ilgili tarihsel mekanlara ait kutucukları işaretlemesi gerekmektedir. Gezilmek istenen tarihi mekanlar seçildikten sonra aynı ekranın alt kısmında bulunan Rotala butonuna basmalıdır. Rotala butonuna basıldığında, uygulamamızda rotalama amacıyla kullandığımız Karınca Kolonisi Algoritması' nın kullandığı fonksiyonu

çağırarak seçilen tarihsel mekanlar arasında bir sıralama yapar ve bu şekilde turistler için sıralı bir rota oluşturur.



Şekil 6.3: Tarihi Mekanlara Götür menüsü

Şekil 6.3' te gösterildiği gibi ekranda tarihsel mekanlar listelenir. Kullanıcı bu ekrandan gezmek istediği tarihsel mekanları işaretlemelidir.



Şekil 6.4: Tarihi Mekanlara Götür menüsünde seçili lokasyonlar

Şekil 6.4' te gösterildiği gibi, gezilmek istenilen tarihsel mekanlar işaretlenir ve ardından rotala butonuna basılır. Rotala butonu KKA' nın kullanıldığı fonksiyonu çağırır. Bu fonksiyonun adımları şu şekildedir:

Kullanıcı tarafından seçilen tarihsel mekanların listesi alınır.

```
public void baslangic()
{
    mekan_listesi = new List<Mekan>();
    karinca_listesi = new List<Karinca>();
    en_iyi_tur_listesi = new List<int>();
}
1 reference
private void Temizle()
{
    en_iyi_tur_uzunluk = -1;
    feromonlar = new double[mekan_sayisi, mekan_sayisi];
    mesafeler = new double[mekan_sayisi, mekan_sayisi];
}
```

Şekil 6.5: Başlangıç parametre değerleri atanması

Şekil 6.5' te görüldüğü üzere önceden belirlediğimiz parametrelerimizden mekan_listesi, karinca_listesi ve en_iyi_tur_listesi için yeni ve boş listeler tanımlanır. En_iyi_tur_uzunluk parametresi -1 default değerine atanır. Feromonlar parametresi seçilirken tarihsel mekan sayısı kadar değer alabileceği belirtilir. Mesafeler parametreside seçilen tarihsel mekan sayısı kadar değer alabileceğini belirtir. Mekan_listesi parametresine seçilen tarihsel mekan bilgileri eklenir.

```
private void MekanlariBaslat()
{

    for (int i = 0; i < mekan_sayisi; i++)
        for (int k = 0; k < mekan_sayisi; k++)
        {
            double x = Math.Pow(mekan_listesi[i].lokasyonu_getir().X -
                (double)mekan_listesi[k].lokasyonu_getir().X, 2.0);
            double y = Math.Pow((double)mekan_listesi[i].lokasyonu_getir().Y -
                (double)mekan_listesi[k].lokasyonu_getir().Y, 2.0);
            mesafeler[i, k] = Math.Sqrt(x + y);
        }

}
```

Şekil 6.6: Mekanlar arası mesafe hesaplama

Şekil 6.6' da gösterildiği gibi parametre değerlerimizi belirledikten sonra her bir noktanın diğer noktalarla arasındaki mesafe tespit edilerek mesafeler parametresine set edilir. Bunun için noktaların mekan sınıfında bulunan lokasyonların X ve Y değerleri alınarak aradaki uzaklık hesaplanır.

```
private void KarincalariBaslat()
{
    int rand_mekan = 0;
    karinca_listesi.Clear();
    for (int i = 0; i < karinca_sayisi; i++)
    {
        rand_mekan = rastgele_sayi.Next(0, mekan_sayisi);
        karinca_listesi.Add(new Karinca(rand_mekan, mekan_sayisi));
        karinca_listesi[i].turListesi[0] = karinca_listesi[i].simdiki_lokasyonu_getir();
        karinca_listesi[i].karincaNeredeListesi[karinca_listesi[i].simdiki_lokasyonu_getir()] = 1;
        karinca_listesi[i].tur_numarasi = 1;
    }
}
```

Şekil 6.7: Karıncaların mekanlara rastgele dağıtılması

Şekil 6.7' de gösterildiği gibi karıncalar rastgele bir şekilde noktalara dağıtılır. Öncelikle rastgele bir tarihsel mekan belirlenir. Karınca listesine yeni karıncanın başlangıç yeri ve toplam nokta sayısı belirtilerek eklenir. Karınca listesinin ilk tur listesine karıncanın anlık bulunduğu lokasyon set edilir. Karınca_nerde_listesine karıncanın uğrayacağı diğer noktadan önce hangi noktaya uğradığı bu listeye set edilir. Bu karıncanın bu noktaya uğradığı anlamındadır. Karınca listesindeki tur numarası ilk uğradığı mekan olduğu için 1. Tur olarak atanır. Bu işlemler karınca sayısı kadar tekrar edilir.

```
private void FeromonlariBaslat()
{
    for (int i = 0; i < mekan_sayisi; i++)
    {
        for (int j = 0; j < mekan_sayisi; j++)
        {
            feromonlar[i, j] = 1.0 / (double)mekan_sayisi;
            feromonlar[j, i] = 1.0 / (double)mekan_sayisi;
        }
    }
}
```

Şekil 6.8: Feromon sıvısının dağıtılması

Şekil 6.8' de belirtildiği gibi mekanların ve karıncaların dağılımlarının ardından feromon sıvısı dağıtımı yapılır.

```
private void Hesapla()
{
    for (int k = 0; k < iterasyonlar; k++)
    {
        for (int i = 0; i < mekan_sayisi; i++)
            if (karincalari_durdur())
            {
                feromonlariBuharlastir();
                feromonlariGuncelle();
                iyi_tur();
                KarincalariBaslat();
            }
    }
}
```

Şekil 6.9: Algoritmanın uygulanması ve rotanın hesaplanması

Şekil 6.9' da gösterildiği üzere algoritmanın uygulanması ve rota hesaplanması için iterasyon sayısı kadar fonksiyonlar tekrarlanır. Yapılan testler sonucunda 34 tarihsel mekan için iterasyon sayısı 250 olarak belirlenmiştir. Buna bağlı olarak her iterasyon için her tarihsel mekan güzergahı için güncelleme gerçekleşecektir.

```
private bool karincalari_durdur()
{
    int hareket = 0;
    for (int i = 0; i < karinca_sayisi; i++)
    {
        if (karinca_listesi[i].tur_numarasi < mekan_sayisi)
        {
            siradakiMekanaGit(karinca_listesi[i]);
            hareket++;
        }
    }
    if (hareket == 0)
    {
        return true;
    }
    else return false;
}
```

Şekil 6.10: Karıncaların her mekana uğramasının sağlanması

Şekil 6.10' da gösterildiği gibi eğer her karınca bütün turlarını tamamladıysa işlemler false değerini alıp sonlandırılacaktır. Eğer karınca her tarihsel mekana uğramadıysa seçili karınca bir sonraki noktaya yönlendirilecektir. Bir sonraki noktaya gidilebilmesi için, seçili karıncaların karınca nerede listesindeki mekanlardan, uğramamış olduğu mekanlar dikkate alınır.

```
private void siradakiMekanaGit(Karinca secili_karinca)
{
    double toplam_prob = 0;
    double hareket_prob = 0;
    int secili_mekan = secili_karinca.simdiki_lokasyonu_getir();
    for (int i = 0; i < mekan_sayisi; i++)
    {
        if (secili_karinca.karincaNeredeListesi[i] == 0)
        {
            toplam_prob += Math.Pow(feromonlar[secili_mekan, i], ALPHA) *
                Math.Pow(1.0 / mesafeler[secili_mekan, i], BETA);
        }
    }
    int varis_sehri = 0;
    double rastgele_hareket = 0;
    int sayi = 0;
    while (sayi < iterasyonlar)
    {
        if (secili_karinca.karincaNeredeListesi[varis_sehri] == 0)
        {
            hareket_prob = (Math.Pow(feromonlar[secili_mekan, varis_sehri], ALPHA) *
                Math.Pow(1.0 / mesafeler[secili_mekan, varis_sehri], BETA)) / toplam_prob;
            rastgele_hareket = rastgele_sayi.NextDouble();
            if (rastgele_hareket < hareket_prob) break;
        }
        varis_sehri++;
        if (varis_sehri >= mekan_sayisi) varis_sehri = 0;
        sayi++;
    }
    secili_karinca.siradaki_lokasyonu_setle(varis_sehri);
    secili_karinca.karincaNeredeListesi[varis_sehri] = 1;
    secili_karinca.turListesi[secili_karinca.tur_numarasi] = varis_sehri;
    secili_karinca.tur_numarasi++;
    secili_karinca.toplam_mesafeyi_guncelle(mesafeler[secili_karinca.simdiki_lokasyonu_getir(), varis_sehri]);
    if (secili_karinca.tur_numarasi == mekan_sayisi)
    {
        secili_karinca.toplam_mesafeyi_guncelle(
            mesafeler[secili_karinca.turListesi[mekan_sayisi - 1], secili_karinca.turListesi[0]]);
    }
    secili_karinca.simdiki_lokasyonu_setle(varis_sehri);
}
```

Şekil 6.11: Karıncanın bir sonraki mekana gitmesi için gerekli fonksiyon

Şekil 6.11' de bir karıncanın bir sonraki mekana giderken kullanacağı fonksiyon gösterilmiştir. Karıncanın bulunduğu noktadan bir sonraki noktaya geçerken uğradığı mekandaki feromon izi miktarının azalması gerekmektedir. Bu sebepten dolayı sonraki noktalara giderken feromon iz miktarı hesaplanır. Bu hesaplama işlemi için Alpha parametresi dikkate alınır. Ayrıca bulunduğu noktadan diğer noktalara gitmek için aradaki mesafe hesaplanırken Beta parametresi kullanılır. Elde edilen feromon izi miktarı ile mesafe değeri çarpılarak toplam iş gücü hesaplanır.

Seçilen karıncanın uğramadığı her nokta için rastgele olarak hareket ettirilmesi sağlanır. İlk hareketinden sonra seçtiği nokta ile bu noktadan sonra seçimini yapacağı nokta arasındaki mesafe karşılaştırılır. Buna göre kısa olan nokta tercih edilir. Toplam iterasyon sayısına ulaşılıncaya kadar bu işlem her nokta için tekrar ettirilir.

En kısa mesafe ve en uygun feromon sıvısı miktarı ile karıncanın sıradaki lokasyonu set edilir. Karınca nerede listesinden yeni tarihsel mekan seçili hale getirilir. Seçili karıncanın tur listesinde tur numarasına bağlı olarak varış noktası belirtilerek tur numarası bir arttırılır. Seçili karıncanın toplamda ilerlemiş olduğu mesafe güncellenir. Eğer karınca her mekana uğradıysa toplam mesafe son nokta ile ilk nokta arasındaki uzaklık hesaplanarak güncellenir. Seçili karıncanın güncel lokasyonu belirlenen varış noktası olarak set edilir. Eğer karıncanın gideceği bir nokta kalmadıysa hesaplama işlemi tamamlanmış demektir.

Karıncalar bir mekandan diğer bir mekana giderken karıncaların arkalarında bıraktığı feromon izleri doğal koşullar gereği buharlaşır. Bundan dolayı ilerlerken bu feromon izlerinin de hesaplanması gerekmektedir.

```
public void feromonlariBuharlastir()
{

    for (int i = 0; i < mekan_sayisi; i++)
        for (int k = 0; k < mekan_sayisi; k++)
        {
            feromonlar[i, k] *= (1.0 - RHO);

            if (feromonlar[i, k] < 1.0 / (double)mekan_sayisi)
            {
                feromonlar[i, k] = 1.0 / (double)mekan_sayisi;
            }
        }
}
```

Şekil 6.12 : Feromonların buharlaştırılması

Şekil 6.12' de feromon buharlaştırma fonksiyonu gösterilmiştir Feromon sıvısı miktarı, bozunma oranına göre buharlaşır. Her tarihsel mekan için tarihsel mekan sayısı kadar feromon sıvısı değerleri bozunma oranı kadar azaltılır.

```
private void feromonlariGuncelle()
{
    for (int i = 0; i < karinca_sayisi; i++)
    {
        for (int k = 0; k < mekan_sayisi; k++)
        {
            int from = karinca_listesi[i].turListesi[k];
            int to = karinca_listesi[i].turListesi[((k + 1) % mekan_sayisi)];
            feromonlar[from, to] += (double)feromon_konsantresi / karinca_listesi[i].uzakligi_getir();
            feromonlar[to, from] = feromonlar[from, to];

        }
    }
}
1 reference
```

Şekil 6.13 : Feromon güncellemesi

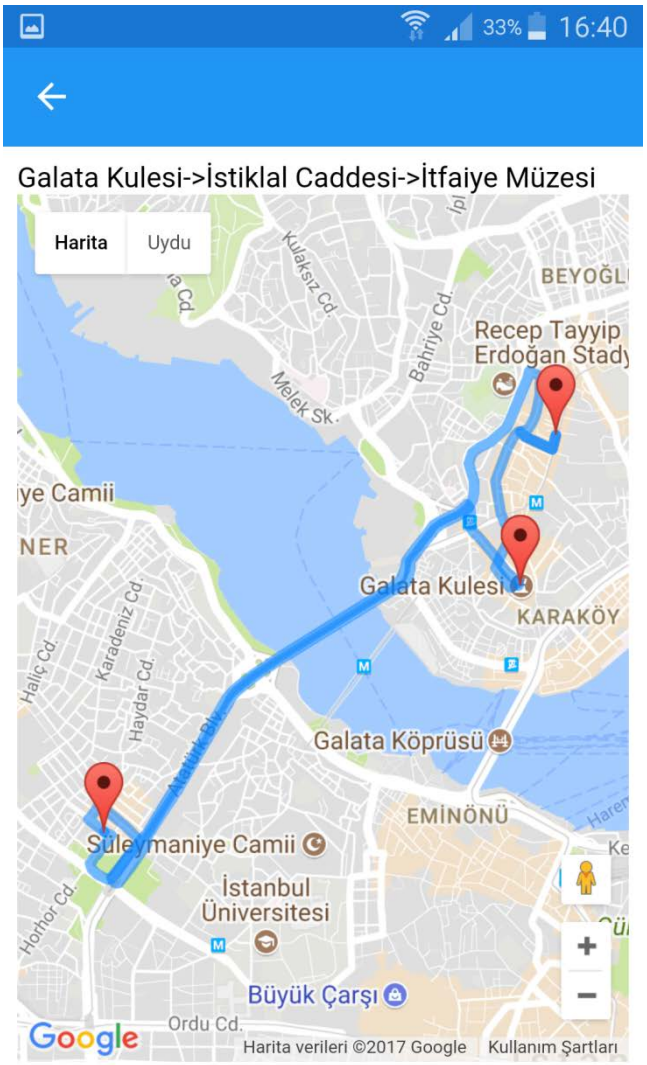
Şekil 6.13’ te görüldüğü üzere bozunma oranı dikkate alınarak azaltılan feromonların güncellemesi yapılırken her karıncanın, her noktada bıraktığı iz miktarı, feromon yoğunluğunun iki nokra arasındaki uzaklığa bölünmesi ile elde edilir.

```
private void iyi_tur()
{
    double en_iyi_tur = karinca_listesi[0].uzakligi_getir();
    int index_kaydet = 0;
    for (int i = 1; i < karinca_listesi.Count; i++)
    {
        if (karinca_listesi[i].uzakligi_getir() < en_iyi_tur)
        {
            en_iyi_tur = karinca_listesi[i].uzakligi_getir();
            index_kaydet = i;
        }
    }

    if (en_iyi_tur < en_iyi_tur_uzunluk || en_iyi_tur_uzunluk == -1)
    {
        en_iyi_tur_listesi = karinca_listesi[index_kaydet].turListesi;
        en_iyi_tur_uzunluk = en_iyi_tur;
    }
}
```

Şekil 6.14 : En iyi turun belirlenmesi

Şekil 6.14’ te gösterdiği gibi feromon güncelleme işlemlerinin ardından, en iyi tur listesine en kısa yoldan en uzun yola doğru güncelleme işlemi yapılır. Bu sayede, seçilen noktalardan birbirine en yakın olanların en kısa mesafeli olandan en uzun mesafeli olana doğru listelenip sıralaması yapılır. Her karıncanın her noktaya uğramasının sağlanması için, karıncalar yeniden noktalara dağıtılarak bu işlemler tekrar edilir.



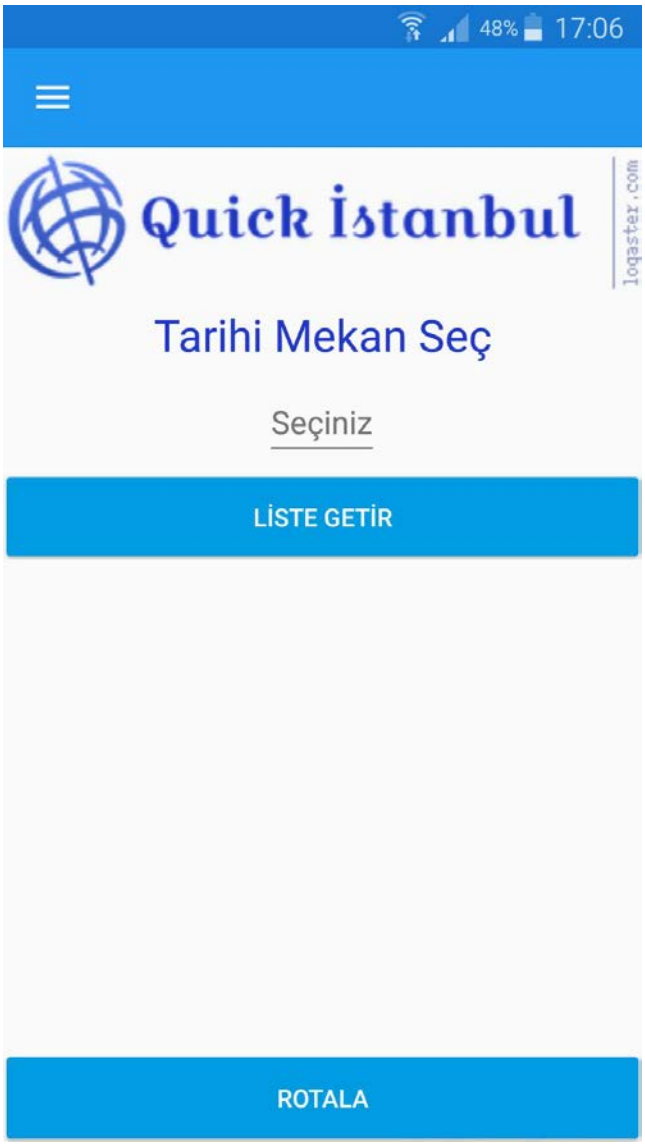
Şekil 6.15: Google api kodları ile seçilen noktaların harita üzerinde gösterimi

Şekil 6.15' te gösterildiği gibi, seçilen tarihsel mekanların rotala butonuna basıldıktan sonra Google maps üzerinde rotalanması görüntülenmektedir. Ekranın üst kısmında noktalar arasında sıralaması yapılan mekanların listesi görüntülenmektedir. Harita üzerinde yapılan rotalama, üst kısımda görüntülenen sıralamaya göre yapılmaktadır.

6.1.2 Beni Bul Götür

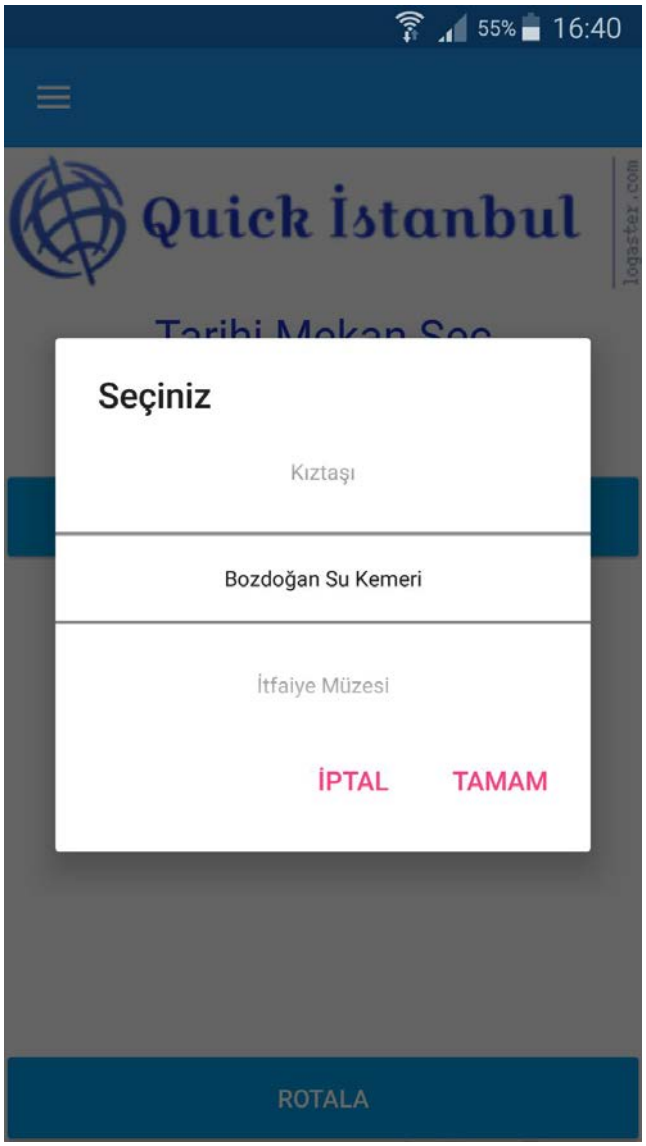
Bu menümüzde öncelikle gezip görülmek istenilen asıl tarihsel mekan belirlenir. Listeden gidilmek istenilen tarihsel mekan seçimi yapılır. Bu menünün asıl işlevi, bulunduğumuz noktadan gezilmek istenilen tarihsel mekan arasında kalan, gezilebilecek, yol üstünde ve belli bir çap genişliği içerisinde olan( ) diğer tarihsel mekanları öneri olarak sunmasıdır. Bu özelliğin aktif olarak

kullanılabilmesi için android cihazımızın konum bulma özelliğinin bulunması ve aktif durumda olması gerekmektedir.



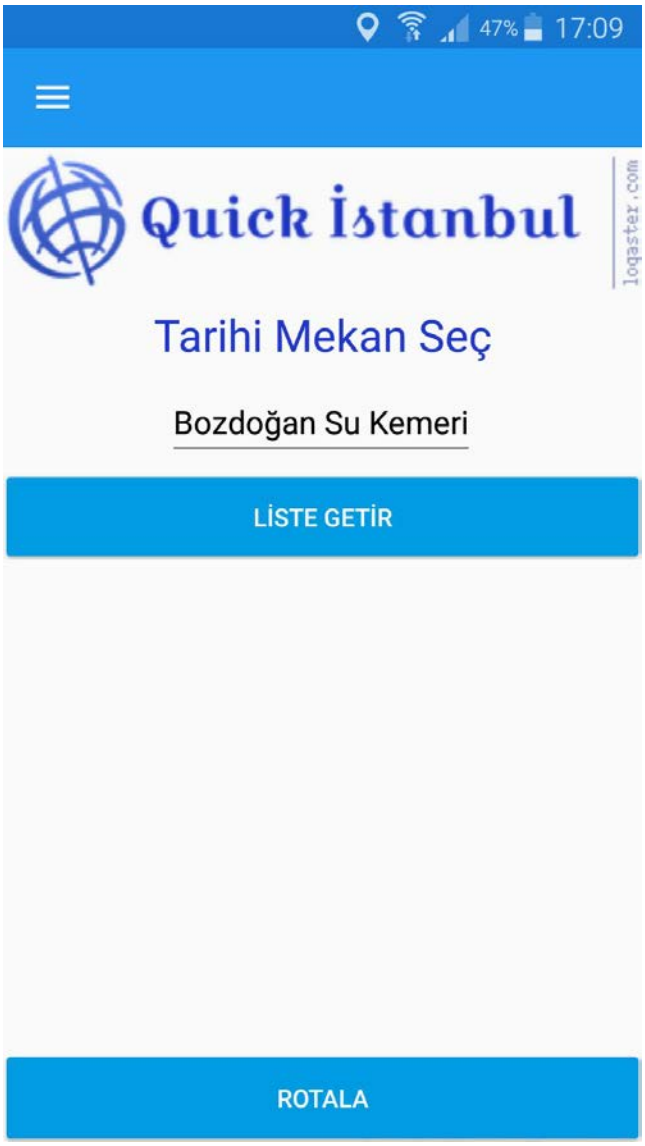
Şekil 6.16: Beni bul götür menüsü giriş ekranı

Şekil 6.16' da görüldüğü üzere beni bul götür menüsünün giriş ekranı gösterilmektedir. Bu ekran üzerinden sadece 1 mekan seçimi yapılabilir. Gezilmek istenen mekan listesi bu ekrandan, seçiniz butonuna basarak görüntülenir.



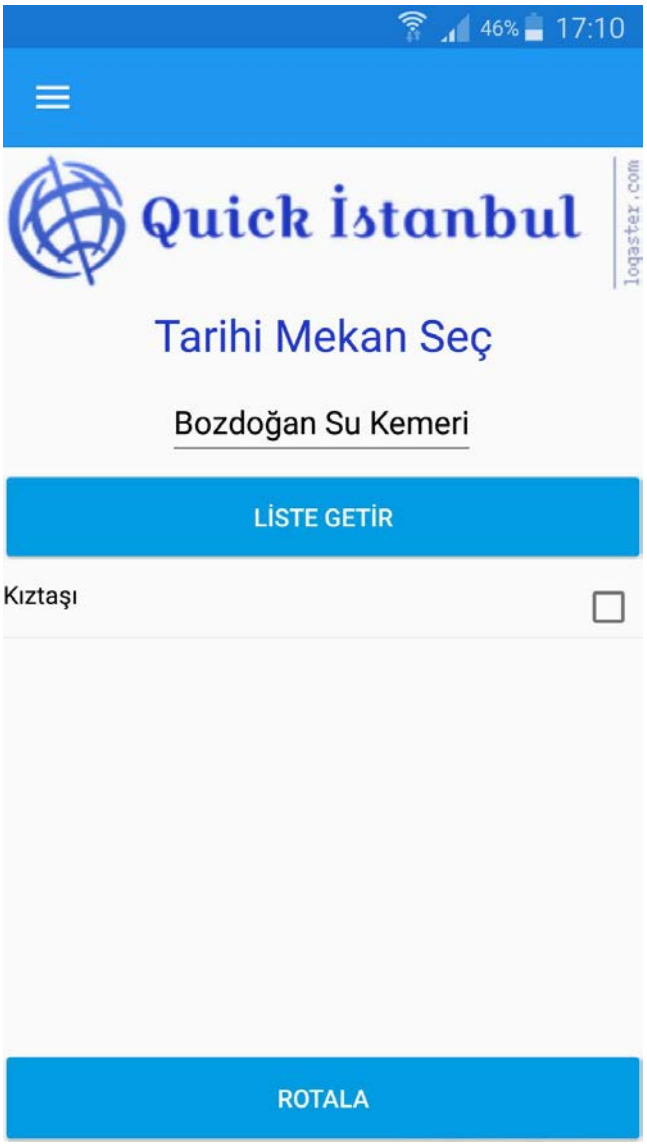
Şekil 6.17: Beni bul götür menüsü seçiniz ekranı

Şekil 6.17' de gösterildiği gibi seçiniz butonuna basıldıktan sonra karşımıza çıkan tarihsel mekan listesi ile karşılaşılmaktadır. Kullanıcı bu ekrandan istediği noktayı seçip ardından tamam butonuna basmalıdır.



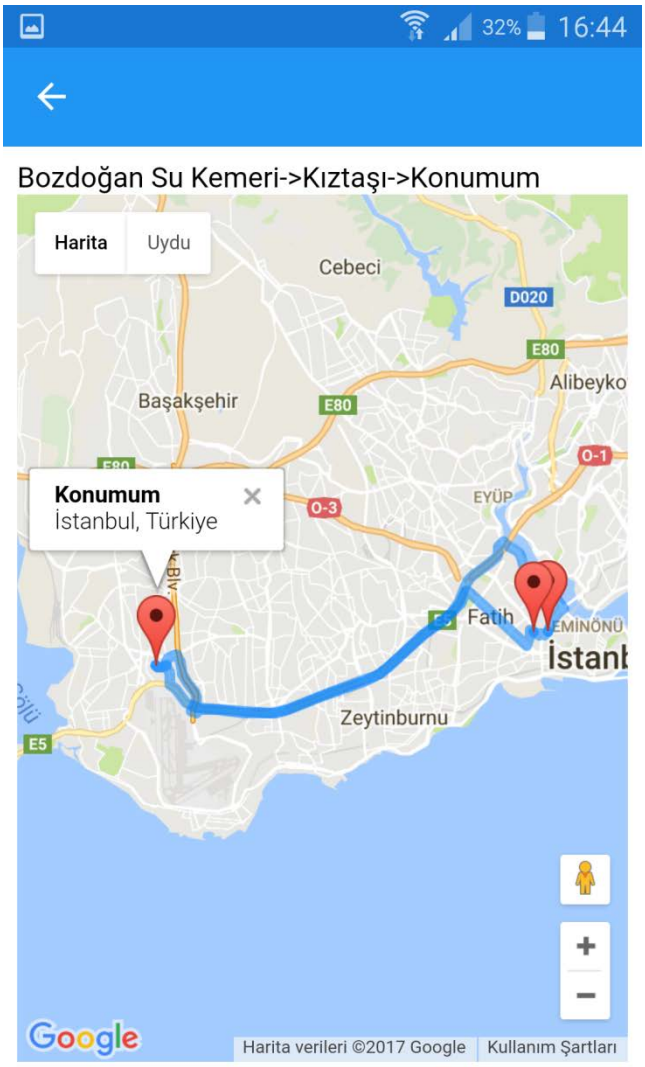
Şekil 6.18: Beni bul götür menüsü liste getir ekranı

Şekil 6.18' de gösterildiği gibi, seçilen tarihi mekan bu ekranda görüntülenir. Gezilmek istenilen tarihsel mekan seçildikten sonra, liste getir butonuna basılır ve GPS yardımı ile kullanıcının konumu belirlenir. Yol üstünde bulunan diğer tarihsel mekanlar için mekan öneri listesi açılmaktadır Bu listede üzerinden isteğe bağlı olarak diğer tarihsel mekan veya mekanlar seçilerek eklenebilir. Seçilen yerler ve kullanıcının konumu arasında bir sıralama yaparak, kullanıcıya sıralı bir liste oluşturulur. Aynı zamanda bu listedeki anlık konumumuz ve seçtiğimiz tarihsel mekanlar, Karınca Kolonisi Algoritması yardımı ile rotalama yapıldıktan sonra Google maps üzerinde de yol tarifi verecek şekilde gösterilir.



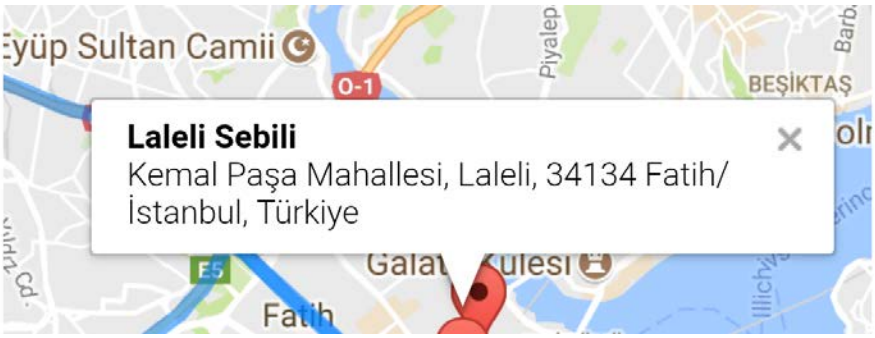
Şekil 6.19: Beni bul götür menüsü önerilen tarihsel mekan listesi

Şekil 6.19' da gösterildiği gibi, bu ekranda kullanıcının seçmiş olduğu tarihsel mekan ile anlık konumu arasında, belli bir çap içerisinde kalan(%+0.15 - %-0.15) tarihsel mekan veya mekanların listesi görüntülenmektedir. Kullanıcı isterse yol üstünde bulunan bu noktaya da uğrayabilecektir.



Şekil 6.20: Beni bul götür menüsü rotalanması ve listelenmesi

Şekil 6.20' de gösterildiği gibi, kullanıcıya sunulan öneri listesinden seçmiş olduğu mekan ile birlikte asıl seçmiş olduğu mekan ve GPS yardımı ile bulunan konumu bir arada rotalanmıştır. Yapılan rotalama liste olarak ekranın üst kısmında gösterildiği gibi aynı zamanda harita üzerinden de gösterilmiştir. Tarihsel Mekanlar menüsünde uygulanan Karınca Kolonisi Algoritması ve Java script teknolojisi burada da kullanılmaktadır. Tarihsel Mekanlar menüsünden farklı olarak GPS ile konumumuzun bulunması, tarihsel mekanlar listesine bu noktanın da eklenmesi ve konumumuz ile gidilmek istenilen nokta arasında bulunan tarihsel mekan arasındaki yerlerin kısıtlanması en önemli farklarıdır.

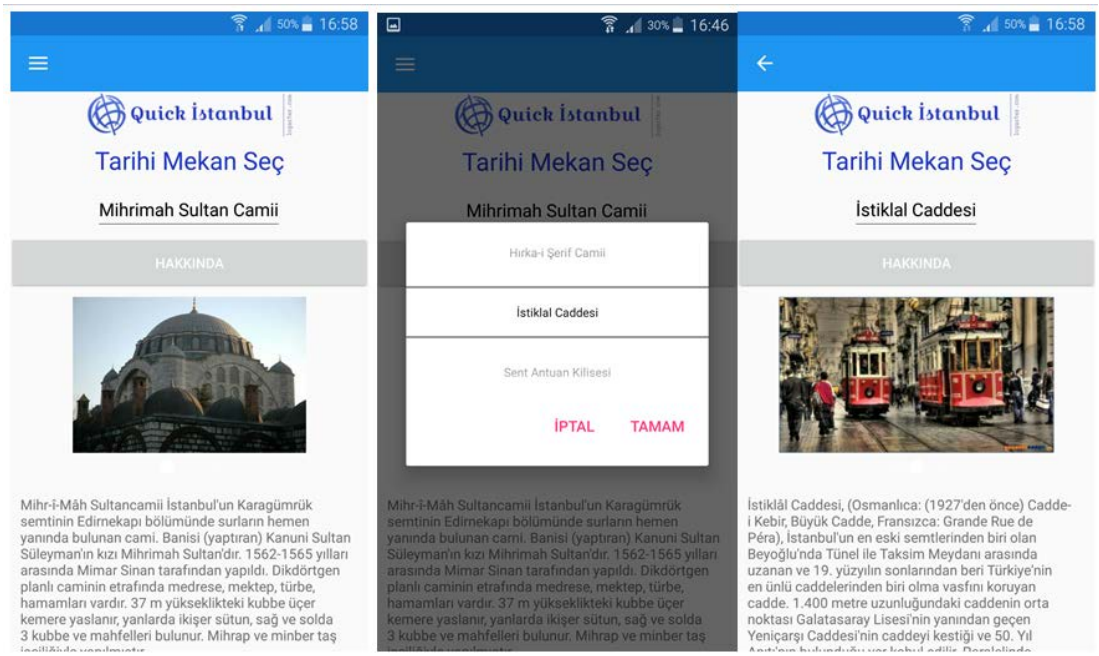


Şekil 6.21: Beni bul götür menüsü maker üzerinde adres gösterimi

Şekil 6.21' de gösterildiği gibi, gidilmek istenilen tarihsel mekan ekranda rotalanırken aynı zamanda maker üzerine tıklayan kullanıcıya o mekana ait adres bilgisi de vermektedir. Bu şekilde gezilmek istenilen nokta hakkında daha detaylı bilgi sahibi olunabilmektedir.

6.1.3 Mekan Bilgileri

Bu menümüzde uygulamamızda kullandığımız, İstanbul ilinde bulunan ve pilot olarak seçilen 34 farklı tarihsel mekana ait görseller ve tarihsel bilgiler yer almaktadır. Bu menünün konulmasındaki amaç herhangi bir arama motoru kullanmaya gerek kalmadan, bu tarihsel mekanlar hakkında bilgi sahibi olmaya imkan sağlamasıdır.

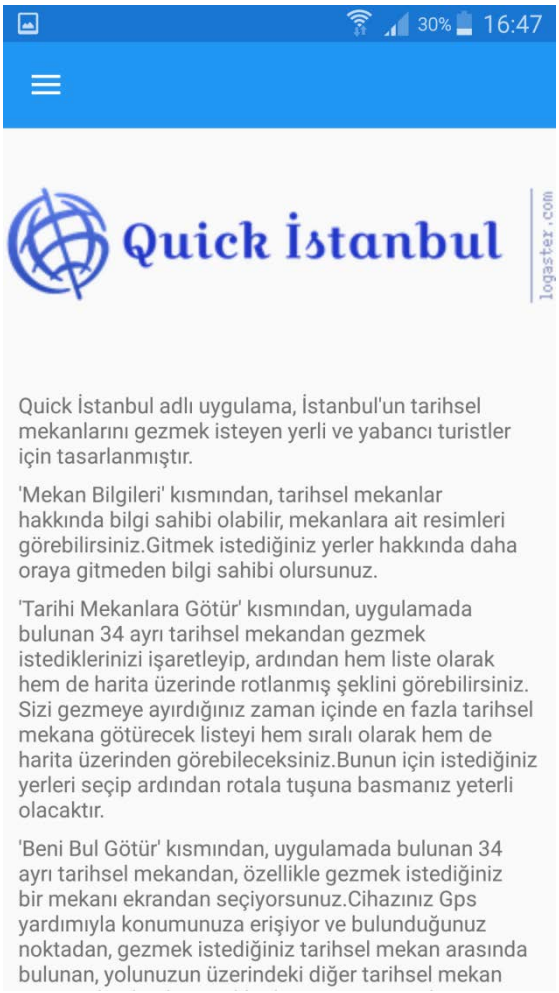


Şekil 6.22: Mekan Bilgileri menüsü ekranları

Şekil 6.22’ de gösterildiği gibi, uygulamamızdaki her tarihsel mekan için üç adet görsel ve kısa bilgiler bulunmaktadır. Bu menüdeki listeden, istenilen tarihsel mekan seçilip ardından hakkında butonuna basılması ile ilgili yer hakkında görsellere ve bilgilere ulaşmış olunur.

6.1.4 Yardım

Bu menümüzde Quick İstanbul adlı uygulamamız için rehber niteliğinde bir bölüm bulunmaktadır.

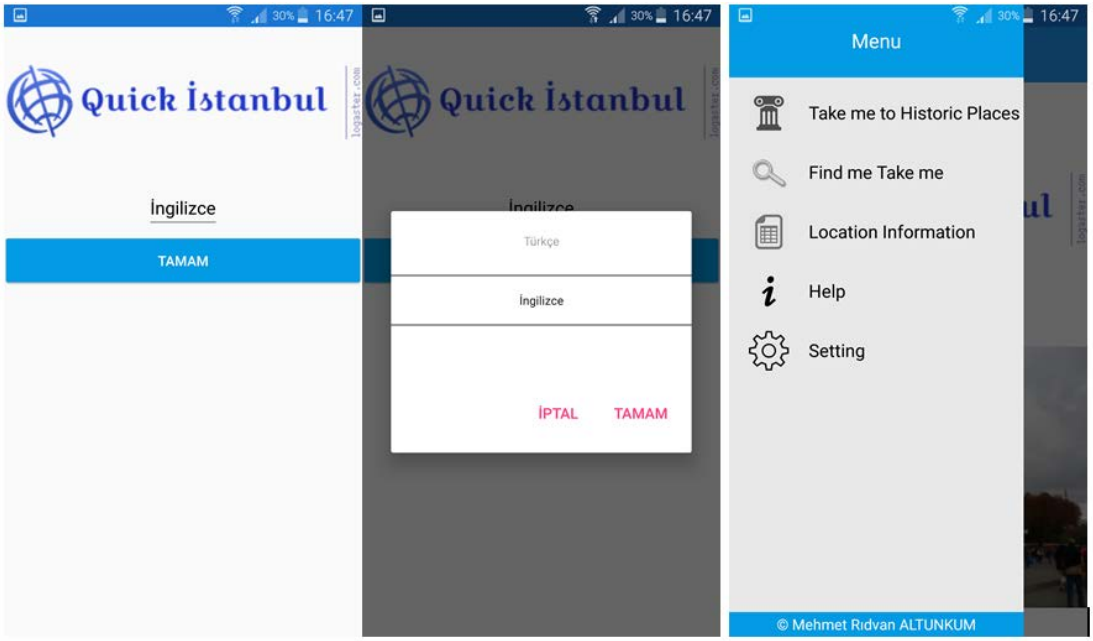


Şekil 6.23: Yardım menüsü ekranı

Şekil 6.23’ te gösterildiği gibi, yazılan uygulama hakkında hangi menünün, hangi kategorinin ne amaçla ve nasıl kullanılabileceği üzerine bilgi vermektedir. Bu sayede uygulama hakkında bilgi sahibi olmayan kullanıcıların, uygulamayı kolayca kavraması amaçlanmıştır.

6.1.5 Ayarlar

Quick İstanbul adlı uygulamamızın yerli ve yabancı turistlere hitap edebilmesi amacı ile birden fazla dil seçeneği eklenmiştir.



Şekil 6.24: Ayarlar menüsü dil seçeneği ekranı

Şekil 6.24' te gösterildiği gibi, bu menümüzde başlangıçta Türkçe olarak çalışan uygulamamız, bu kısımda eklenen dil seçeneği sayesinde istenildiği zaman tamamen İngilizce olarak çalışabilmektedir. Bu sayede yabancı turistler için de faydalı ve yol gösterici olacaktır.

6.2 Google Api kullanılarak web sitesinin kodlanması

Elimizde artık en iyi tur listesi olduğuna göre bu noktalar web sitesine parametre olarak gönderilir. Gelen parametreler ile tanımlı mekanlar arasındaki rotalama işlemi Google api kodları yardımı ile gösterimi sağlanır. Google maps üzerinden rotalama yapılırken javascript teknolojisinden yararlanılır.

7 DENEYSEL ÇALIŞMALAR

Bu bölümde, uygulamamızda kullanılan KKA için kullanılan parametre değerleri test edilerek en uygun sonuca ulaşılmaya çalışılmıştır. Uygulamamızda yer alan 34 adet tarihsel mekanın en iyi şekilde sıralanabilmesi için, alpha, beta, rho, iterasyon sayısı ve karınca sayısı ayrı ayrı testlere tabi tutulmuştur. Uygulamamızın, belirlenen konumlardaki tarihsel mekanlar için en kısa mesafeyi verebilmesi amacıyla belirtilen parametrelerin test kombinasyonları yapılmıştır. Başarı ölçütü olarak belirtilen parametreler ışığında, en kısa mesafeyi veren değerler olarak belirlenmiştir.

7.1 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 7.1' de gösterilmiştir. Alpha parametresine farklı değerler verilirken diğer parametreler sabit tutulmuştur.

Çizelge 7.1: Alpha parametresi değişimi ile elde edilen mesafe sonuçları

ALPHA	BETA	RHO	İTERASYON S.	MEKAN S.	KARINCA S.	MESAFE
0,25	1	0,5	100	34	100	2259
0,50	1	0,5	100	34	100	2092
0,75	1	0,5	100	34	100	1779
1	1	0,5	100	34	100	1694
1,25	1	0,5	100	34	100	1711
1,50	1	0,5	100	34	100	1726
1,75	1	0,5	100	34	100	1730
2	1	0,5	100	34	100	1809

Çizelge 7.1' de görüldüğü gibi, en iyi sonuç alpha parametresi 1 iken sağlanmıştır.

7.2 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 7.2' de gösterilmiştir. Beta parametresine farklı değerler verilirken diğer parametreler sabit tutulmuştur.

Çizelge 7.2: Beta parametresi değişimi ile elde edilen mesafe sonuçları

ALPHA	BETA	RHO	İTERASYON S.	MEKAN S.	KARINCA S.	MESAFE
1	1	0,5	100	34	100	1698
1	1,5	0,5	100	34	100	1705
1	2	0,5	100	34	100	1708
1	2,5	0,5	100	34	100	1710
1	3	0,5	100	34	100	1718
1	3,5	0,5	100	34	100	1723
1	4	0,5	100	34	100	1729
1	5	0,5	100	34	100	1739

Çizelge 7.2' de görüldüğü gibi, en iyi sonuç beta parametresi 1 iken sağlanmıştır.

7.3 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 7.3' te gösterilmiştir. Rho parametresine farklı değerler verilirken diğer parametreler sabit tutulmuştur.

Çizelge 7.3: RHO parametresi değişimi ile elde edilen mesafe sonuçları

ALPHA	BETA	RHO	İTERASYON S.	MEKAN S.	KARINCA S.	MESAFE
1	1	0,2	100	34	100	1752
1	1	0,3	100	34	100	1745
1	1	0,4	100	34	100	1744
1	1	0,5	100	34	100	1734
1	1	0,6	100	34	100	1728
1	1	0,7	100	34	100	1709
1	1	0,8	100	34	100	1754
1	1	0,9	100	34	100	1762

Çizelge 7.3' te görüldüğü gibi, en iyi sonuç Rho parametresi 0,7 iken sağlanmıştır.

7.4 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 7.4' te gösterilmiştir. İterasyon parametresine farklı değerler verilirken diğer parametreler sabit tutulmuştur.

Çizelge 7.4: İterasyon parametresi değişimi ile elde edilen mesafe sonuçları

ALPHA	BETA	RHO	İTERASYON S.	MEKAN S.	KARINCA S.	MESAFE
1	1	0,7	50	34	100	1756
1	1	0,7	100	34	100	1744
1	1	0,7	150	34	100	1724
1	1	0,7	200	34	100	1702
1	1	0,7	250	34	100	1718
1	1	0,7	300	34	100	1696
1	1	0,7	350	34	100	1698
1	1	0,7	400	34	100	1697

Çizelge 7.4' te görüldüğü gibi, en iyi sonuçlar İterasyon parametresi sürekli artarken gözlemlenmiş ve uygulamamızın cevap verme süresi göz önünde bulundurularak 250 olarak belirlenmiştir.

7.5 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 7.5' te gösterilmiştir. Karınca sayısı parametresine farklı değerler verilirken diğer parametreler sabit tutulmuştur.

Çizelge 7.5: Karınca Sayısı parametresi değişimi ile elde edilen mesafe sonuçları

ALPHA	BETA	RHO	İTERASYON S.	MEKAN S.	KARINCA S.	MESAFE
1	1	0,7	250	34	50	1756
1	1	0,7	250	34	70	1768
1	1	0,7	250	34	90	1698
1	1	0,7	250	34	110	1702
1	1	0,7	250	34	130	1727
1	1	0,7	250	34	150	1729
1	1	0,7	250	34	170	1732
1	1	0,7	250	34	190	1734

Çizelge 7.5' te görüldüğü gibi, en iyi sonuç Karınca sayısı parametresi 90 iken sağlanmıştır.

Uygulamamızda bulunan 34 adet tarihsel mekan 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 tarihsel mekanların koordinatları üzerinden yapılan sıralamanın kalitesi artmıştır.

8 SONUÇ VE ÖNERİLER

Teknolojinin gelişimi ve hızla yaygınlaşması ile mobil cihazların ve mobil uygulamaların kullanımı her geçen gün artmaktadır. Mobil cihaz ve uygulama kullanım oranlarının artması ile rotalama ve bilgi edinme üzerine birçok çalışma yapılmaktadır.

Bu tez çalışmasında Tarihsel mekanların en kısa yoldan rotalanması üzerine, android işletim sisteminde çalışacak bir mobil uygulama yazılmış ve bu uygulama için KKA kullanılmıştır.

Bu çalışmada gerçekleştirilen testler sonucunda, yerli ve yabancı turistler için zamandan tasarruf etmeyi sağlayan bir uygulama geliştirilmiştir. İstanbul ilinde bulunan ve pilot olarak seçilen 34 adet tarihsel mekan için KKA kullanılarak hem bilgi sahibi olmayı hem de zamandan tasarruf ederek en kısa yoldan bu mekanları gezmeyi sağlamaktadır.

Uygulama yardımı ile listelenen tarihsel mekanlar hakkında görsellere ve bilgilere ulaşılabilir, belli sayıdaki tarihsel mekanları kendi aralarında yakından uzağa doğru sıralayabilir, bulunduğunuz yerdeki konumunuzdan gezmek istediğiniz tarihi mekana rotalama yaparken arada kalan mekanlara da uğrayabilirsiniz. Birden fazla dil seçeneği sayesinde farklı dillerde, bahsedilen olanaklardan yararlanılabilmektedir.

Çalışmada kullanılan 34 tarihsel mekan sayısı daha da arttırılabilir, başka illerdeki tarihsel mekanlara uyarlanabilir veya farklı amaçlar doğrultusunda, sabit noktalar için uyarlanabilir bir yapıdadır. Çalışmanın, bu konuda çalışma yapmak isteyen araştırmacılara yardımcı olması beklenmektedir.

KAYNAKLAR

Akay E T.C. Kara Harp Okulu Savunma Bilimleri Enstitüsü Harekât Araştırması Ana Bilim Dalı Helikopter Rotalama Problemi: Bir Eş-Zamanlı Topla-Dağıt Araç Rotalama Problemi Modeli Ve Sezgisel Bir Çözüm Yaklaşımı Ankara-2016 Yüksek Lisans Tezi

Alkış, S., Oğuzoğlu, Y. (2005). Ülkemiz Koşullarında Tarihi Çevre Eğitiminin Önemini Ve Gerekliliğini Arttıran Etkenler, Uludağ Üniversitesi, Eğitim Fakültesi Dergisi, Xvııı (2).

Alpaslan M Yüksek Lisans Tezi Endüstri Mühendisliği Anabilim Dalı Mayıs-2015 Araç Rotalama Problemleri İçin Matematiksel Modeller Ve Çözüm Yöntemleri Anadolu Üniversitesi Fen Bilimleri Enstitüsü Endüstri Mühendisliği Anabilim Dalı

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

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.

Burak K, İsmail E 09.07.2012 Araç Rotalama Sistemleri Ve Tasarruf Algoritması Uygulaması (Araştırma Makalesi) İstanbul Ticaret Üniversitesi..Fen Bilimleri Dergisi Yıl: 11 Sayı: 21 Bahar 2012 S.41-51

Can A G Selçuk Üniversitesi Fen Bilimleri Enstitüsü Endüstri Mühendisliği Anabilim Dalı Zaman Bağımlı Eş Zamanlı Topla Dağıt Araç Rotalama Problemi 2015

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

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.

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

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

Çağrı K Zaman Pencereli Araç Rotalama Problemlerinin Popülasyon Tabanlı Sezgisel Yöntemler İle Optimize Edilmesi İstanbul Üniversitesi Fen Bilimleri Enstitüsü Endüstri Mühendisliği Ana Bilim Dalı 2016

Çakır E T İşletme Anabilim Dalı Üretim Yönetimi Ve Pazarlama Bilim Dalı Bir Tersine Lojistik Faaliyeti Olarak Tıbbi Atıkların Toplanmasında Araç Rotalama Uygulaması Sakarya Üniversitesi, Sosyal Bilimler Enstitüsü 2016

Çalışkan K Karınca Kolonisi Optimizasyonu İle Araç Rotalama Probleminin Maliyetlerinin Kümeleme Tekniği İle İyileştirilmesi Tobb Ekonomi Ve

Teknoloji Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı 2011

Demircioğlu, İ. H. (2010). Tarih Öğretiminde Öğrenci Merkezli Yaklaşımlar. Ankara: Anı Yayıncılık.

Demircioğlu, M Araç Rotalama Probleminin Sezgisel Bir Yaklaşım İle Çözümlenmesi Üzerine Bir Uygulama Doktora Tezi Adana /2009 T.C. Çukurova Üniversitesi Sosyal Bilimler Enstitüsü İşletme Anabilim Dalı

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.

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.

Ekizler H Araç Rotalama Probleminin Çözümünde Karınca Kolonisi Optimizasyonu Algoritmasının Kullanılması İstanbul Üniversitesi Fen Bilimleri Enstitüsü Endüstri Mühendisliği Anabilim Dalı 2011

Eldem H Selçuk Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı Karınca Koloni Optimizasyonu (Kko) Ve Parçacık Sürü Optimizasyonu (Pso) Algoritmaları Temelli Bir Hiyerarşik Yaklaşım Geliştirilmesi 2014

Gangal V Kablosuz Algılayıcı Ağlarda Karınca Koloni Algoritmalı Rotalama İle Enerji Etkin Rotalamanın İncelenmesi Karadeniz Teknik Üniversitesi Fen Bilimleri Enstitüsü Elektrik-Elektronik Mühendisliği Anabilim Dalı 2015

Gökalp O Karınca Kolonisi Eniyilemesi Algoritmaları İçin Çaprazlama Yöntemleri Geliştirilmesi Ege Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı 2012

Ilgaz, S., Örten, H. (2015). *Sosyal Bilgiler Eğitiminde Tarihi Ve Sanatsal Mekânlara Dayalı Öğrenme Ortamları*, Sosyal Bilgiler Eğitiminde Mekânsal Öğrenme Ortamları, (Ed. Ramazan Sever Ve Erol Koçoğlu). Ankara: PegemAkademi.

Keskintürk T 2009 Araç Rotalama Problemlerinin Global Karınca Koloni Optimizasyonu İle Çözümü T. C. İstanbul Üniversitesi Sosyal Bilimler Enstitüsü İşletme Anabilim Dalı Sayısal Yöntemler Bilim Dalı

Knox J. (1994), "Tabu Search Performance On The Symmetric Traveling Salesman Problem", Computers And Operations Research, C. 21, Sf. 867-876

Obitko, M., "Crossover And Mutation", Http://Www.Obitko.Com/Tutorials/Genetic-Algorithms/Crossover-Mutation.Php (Erişim Tarihi: 22 Şubat 2017).

Özçelik, İ. (2011). Tarih Araştırmalarında Yöntem Ve Teknikler. Ankara: Nobel Yayınları.

Safran, M., Ata, B. (2006). "Okul Dışı Tarih Öğretimi", Tarih Eğitimi Makale Ve Bildiriler, Ankara: Gazi Kitabevi.

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

Suvaydan F Düzce Üniversitesi Fen Bilimleri Enstitüsü Elektrik Eğitimi Anabilim Dalı Mobil Robotlar İçin Yol Planlama Problemi Ve Karınca Kolonisi Algoritması İle Yol Planlama Problemlerinin Optimal Çözümü 2011

Turan, R., Ulusoy, K. (2013). Sosyal Bilgilerde Tarihin Yeri Önemi, Sosyal Bilgilerin Temelleri. Ankara: Pegem Akademi.

İNTERNET KAYNAKLARI

[1] http://www.yazilibilgi.com/2015/02/tarih-nedir-tarih-hakknda-ksa-bilgi.html

[2] https://tr.wikipedia.org/wiki/%C4%B0stanbul_tarihi

[3] http://bilgisayarkavramlari.sadievrenseker.com/2008/12/22/sezgi-ustu-algoritmalar-ustsezgisel-algoritmalar-meta-heuristic-algorithms/

[4] http://w3.gazi.edu.tr/~ikaraoglan/1203720.html

[5] http://www.istanbul.net.tr/istanbul-rehberi/tarihi-eserler/6/3

[6] ARP ÇÖZÜM YÖNTEMLERİ (Çukurova Üniversitesi İİBF Dergisi) Cilt:13. Sayı:1.Haziran 2009 ss.73 Araç Rotalama Problemleri ve Çözüm Yöntemleri Erkut DÜZAKIN veMert DEMİRCİOĞLU

EKLER

Ek A: Quick İstanbul Adlı Uygulamanın Android İşletim Sistemi İle Kodlanmış Kısmı Cd'de teslim edilecektir

Ek B: Quick İstanbul Adlı Uygulamanın Haritada Gösterim Bölümü İçin Kodlanan Web Sitesi Kısmı Cd'de teslim edilecektir

ÖZGEÇMİŞ

Mehmet Rıdvan ALTUNKUM, 01.08.1990 tarihinde Siirt'te doğdu. 2001'de Atatürk İlkokulu'ndan mezun oldu. 2004 yılında Mehmetcik Ortaokulu'ndan, 2008 yılında Siirt Atatürk Anadolu Lisesi'nden mezun oldu. 2011 yılında İstanbul Aydın Üniversitesi, Eğitim Fakültesi, Bilgisayar ve Öğretim Teknolojileri Öğretmenliği Bölümü'nü kazandı. 2012 yılı güz döneminde Siirt Üniversitesi, Eğitim Fakültesi Bilgisayar ve Öğretim Teknolojileri Öğretmenliği Bölümü'ne yatay geçiş yaptı. 2015 yılında bu bölümden mezun oldu. 2015 yılında İstanbul Aydın Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği'nde yüksek lisans eğitimine başladı. İyi derecede İngilizce ve Arapça dili bilmektedir. İlgi alanlarından bazıları, Yapay Zeka, Sürü Zekası, Bulanık Mantık, Android Programlama'dır.