ŞEHİR İÇİ ULAŞIMDA
ZAMAN BAĞIMLI ROTA BELİRLEME

Fatih ŞAHİN

YÜKSEK LİSANS TEZİ
ENDÜSTRİ MÜHENDİSLİĞİ

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

HAZİRAN 2009
ANKARA

Fatih ŞAHİN tarafından hazırlanan ŞEHİR İÇİ ULAŞIMDA ZAMAN BAĞIMLI ROTA BELİRLEME adlı bu tezin Yüksek Lisans tezi olarak uygun olduğunu onaylarım.

Yrd. Doç. Dr. Ediz ATMACA ……..
Tez Danışmanı, Endüstri Mühendisliği Ana Bilim Dalı

Bu çalışma, jürimiz tarafından oy birliği / oy çokluğu ile Endüstri Mühendisliği Anabilim Dalında Yüksek Lisans olarak kabul edilmiştir.

Prof. Dr. Hadi GÖKÇEN ……..
Endüstri Mühendisliği, Gazi Üniversitesi

Yrd. Doç. Dr. Ediz ATMACA ……..
Endüstri Mühendisliği, Gazi Üniversitesi

Yrd.Doç.Dr.Hasan Şakir BİLGE ……..
Bilgisayar Mühendisliği, Gazi Üniversitesi

Tarih: 26.06.2009

Bu tez, Gazi Üniversitesi Fen Bilimleri Enstitüsü tez yazım kurallarına uygundur.

Prof.Dr.Nail ÜNSAL ……..
Fen Bilimleri Enstitü Müdürü

TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, ayrıca tez yazım kurallarına uygun olarak hazırlanan bu çalışmada bana ait olmayan her türlü ifade ve bilginin kaynağına eksiksiz atıf yapıldığını bildiririm.

Fatih ŞAHİN

ŞEHİR İÇİ ULAŞIMDA
ZAMAN BAĞIMLI ROTA BELİRLEME
(Yüksek Lisans Tezi)

Fatih ŞAHİN

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
Haziran 2009

ÖZET

En kısa yol problemleri, yöneylem araştırması, yapay zekâ ve bilgisayar bilimlerinde yaygın olarak kullanılmaktadır. Çalışmada, literatürde yer alan en kısa yol problemleri incelenmiş ve zaman bağımlı en kısa yol problemi üzerinde durulmuştur. Zaman bağımlı en kısa yolu belirlemedeki amaç, kullanıcıyı en kısa zamanda istenilen noktaya ulaştırmada alternatifler sunmaktır. Çalışmanın uygulama kısmında, Java yazılım dili ile bir web uygulaması geliştirilmiştir ve Ankara ili için uygulanmıştır. Uygulama, şehir içi taşımacılıkta kullanılan kaynaklar arasında aktarmalar yaparak zaman bağımlı en kısa yolu belirlemeye çalışmaktadır. Zaman bağımlı rota belirleme uygulaması, benzer uygulamalar ile karşılaştırılmış ve uygulamanın avantajları üzerinde durulmuştur. Uygulamada bulunan çözümler, Google Maps API yardımıyla görselleştirilmiştir

Bilim Kodu : 906.1.148
Anahtar Kelimeler : Zaman bağımlı en kısa yol problemi, graf, dijkstra
Sayfa Adedi : 83
Tez Yöneticisi : Yrd. Doç. Dr. Ediz ATMACA

DEFINING TIME DEPENDENT ROUTE
IN URBAN TRANSPORTATION
(M.Sc. Thesis)

Fatih ŞAHİN

GAZI UNIVERSITY
INSTITUTE OF SCIENCE AND TECHNOLOGY
June 2009

ABSTRACT

The shortest path problems are used widely such as operational research, artificial intelligence and computer science. In this study, the shortest path problem in literature were examined and time-dependent shortest path problem has been emphasized. Purpose of defining time-dependent shortest path is providing alternatives to reach user target point. In the application part of the study, a web application was developed with Java programming language and was implemented for Ankara. The application is trying to determine the time dependence of the shortest path that used in urban transportation resources by making the transfer. Defining time-dependent shortest path application is compared with similar applications and focused on its advantages. The solutions of the application are visualised by the help of Google Maps API.

Science Code : **906.1.148**
Key Words : **Time dependent shortest path problem, graph, dijkstra**
Page Number : **83**
Adviser : **Assist. Prof. Dr. Ediz ATMACA**

TEŞEKKÜR

Çalışmalarım boyunca tecrübelerinden faydalandığım, yardım ve katkılarıyla beni yönlendiren danışman hocam Yrd. Doç. Dr. Ediz ATMACA' ya, tez kapsamındaki uygulama için verilerini kullanmama izin veren, değerli vakitlerini ve ilgilerini esirgemeyen EGO çalışanlarına teşekkürü bir borç bilirim. Ayrıca çalışma esnasındaki desteklerinden dolayı bilgisayar mühendisi Sema ÇEREKÇİ' ye, öğrenim hayatım boyunca bana en iyi imkânları sağlamak için hiçbir fedakârlıktan kaçınmamış olan aileme çok teşekkür ederim.

İÇİNDEKİLER

Sayfa

Sayfa

Sayfa

ÇİZELGELERİN LİSTESİ

Çizelge **Sayfa**

ŞEKİLLERİN LİSTESİ

Şekil **Sayfa**

Şekil **Sayfa**

RESİMLERİN LİSTESİ

Resim **Sayfa**

Resim **Sayfa**

HARİTALARIN LİSTESİ

Harita **Sayfa**

SİMGELER VE KISALTMALAR

Bu çalışmada kullanılmış bazı simgeler ve kısaltmalar, açıklamaları ile birlikte aşağıda sunulmuştur.

Simgeler	**Açıklama**
A	Sonlu m ayrıt seti
d_i	i düğümü
D	Düğümler kümesi
G	Graf kümesi
k	Komşuluk değeri
K	Ayrıtlar kümesi
l_{ij}	Uzunluk veya ağırlık değerleri
M	Bitişiklik matrisi
n	Düğüm sayısı
N	Sonlu n düğüm seti

1. GİRİŞ

Şehir içi ulaşımda zaman bağımlı en kısa yolu belirlemedeki amaç, kullanıcıya en kısa zamanda istenilen noktaya ulaşımda alternatifler sunmaktır. Sunulan alternatifler ile ulaşım süreleri dolayısıyla ulaşım maliyetleri düşmekte ve şehir içi taşımacılıkta kullanılan kaynakların verimliliği artmaktadır. Aynı zamanda ulaşım kaynakları etkin kullanılarak atıl kaynaklar da kullanılabilir hale gelmektedir.

Şehir içi ulaşımda zaman bağımlı en kısa yol uygulamasında ulaşım kaynaklarından kullanıcıya kadar uzanan bir bilgi akışına ihtiyaç duyulmaktadır. Bunun gerçekleştirilebilmesi için, düğüm noktaları (durak, metro istasyonu vb.), düğüm noktaları arasında hareket eden araçlar (otobüs, metro, minibüs, tren vb.) , araçların rotaları, araçların düğüm noktalarına daha önceki geliş süreleri ve araçların tarife bilgilerinin çok iyi şekilde belirlenmesi gerekmektedir. Bütün olasılıkları göz önünde bulundurarak, hızlı sonuç üreten bir yöntem olması nedeniyle uygulamadan pek çok alanda yararlanılabilmektedir.

Çalışmanın birinci bölümü giriştir. İkinci bölümde, graf teorisi ve graf algoritmaları hakkında bilgi verilmiştir. Graf teorisinin tarihsel gelişimi ve grafın kullanım alanları anlatılmaktadır. Graflar ile ilgili çözüm yaklaşımları bu bölümde verilmiş ve en kısa yol algoritmaları incelenmiştir.

Üçüncü bölümde uygulamaya konu olan seyahat şebekelerinin modelleme teknikleri üzerinde durulmaktadır. Seyahat şebekeleri modellenirken en kısa yol algoritmalarından faydalanılmıştır ve modelin zaman bağımlı bir model olduğu vurgulanmıştır. Modele ayrıca yürüyüş yolları da dahil edilerek model daha detaylı hale getirilmiştir.

Çalışmanın dördüncü bölümünde çalışma ile ilgili zaman bağımlı en kısa yol problemleri için literatür araştırması, tarihsel gelişimi içerisinde verilmiştir. Literatür bölümünde, herbir çalışmanın zaman bağımlı en kısa yol problemlerine olan katkıları detaylı olarak incelenmektedir.

Beşinci bölümde, zaman bağımlı en kısa yol problemi için yapılan uygulama hakkında bilgi verilmektedir. Bu bölümde, uygulamada kullanılan algoritma ve algoritmaya yapılan ek kısımlar anlatılmaktadır. Zaman bağımlı en kısa yol problemi öncelikle örnek bir graf üzerinde gösterilerek çözülmüştür. En kısa yol problemi daha sonra, geliştirilen web uygulaması ile çözülerek sonuçlar incelenmiştir. Kullanılan kaynaklar, uygulama sonuçları, diğer uygulamalar ile zaman bağımlı en kısa yol uygulamasının benzer ve farklı özellikler uygulama bölümünde verilmektedir. Uygulama bölümünde ayrıca, hat, güzergah, tarife bilgisi ve aktarmalı ulaşımın boyutları gibi çeşitli bütünleşme unsurları dünya ve ülkemiz kentlerindeki, uygulamalarla değerlendirilmektedir.

Çalışmanın sonuç bölümünde ise uygulamanın sonuçları genel olarak değerlendirilmekte ve çalışmanın en kısa yol problemi uygulamalarına yaptığı katkılar vurgulanmaktadır. Toplu taşıma sistemlerinde aktarmayı ortaya çıkaran ihtiyaçları, aktarmaların ortaya çıkması ile birlikte aktarma yapılan türler incelenmektedir. Aktarmalı yolculuklar, hem işleticiler hem de yolcular açısından daha etkin ve kolay hale getirmek için yapılan düzenlemeler ve teknolojik gelişmelerin bütünleşmeye getirdiği kolaylıklar dünya kentlerinden örnekleriyle incelenmekte, Ankara ili için yapılması gerekenler özetlenmektedir. Çalışmanın devamında uygulama için yapılabilecek faaliyetler ve bu faliyetlerden bazılarının zorlukları üzerinde durulmaktadır. Uygulamanın daha iyi sonuçlar verebilmesi için ileriki çalışmalara önerilerde bulunulmaktadır.

2. GRAF TEORİSİ VE GRAF ALGORİTMALARI

Graf Teorisi, bir olay ve ifadenin, düğüm ve çizgiler kullanılarak gösterilmesi biçimidir. Graf teorisinden, problemleri tanımlama ve yapısal olarak ilişkileri belirlemekte de faydalanılmaktadır. İlk olarak 1736 yılında Matematikçi Leonhard Euler tarafından önerilmiş ve yine aynı bilim adamı tarafından ünlü Könisberg köprüsü probleminin çözümünde uygulanmıştır. Graf teorisinin uygulamaları, modern hayatın karmaşık ve geniş kapsamlı birçok probleminin çözümü için kullanılmaktadır. Bu uygulamalar; ekonomi, yönetim bilimi, satış pazarlama, bilgi iletimi ve taşıma planlaması gibi alanları kapsamaktadır.

Graf, düğüm noktaları ve bu noktaları birbirine bağlayan ayrıtlardan oluşan bir ağ yapısıdır. Bir grafı tanımlamak için öncelikle düğümler ve ayrıtlar kümesinin tanımlanması gerekmektedir. Daha sonra hangi ayrıtların hangi düğümleri bağladığı belirtilmelidir. Bir ayrıt her iki ucunda da bir düğüm olacak şekilde tanımlandığından graftaki tüm ayrıtların uç noktalarını bir düğüm ile ilişkilendirmek gerekmektedir.

Graflar mühendislik, sosyal bilimler ve temel bilimlerde, problemlere algoritmik bir çözüm sunmaktadırlar. Sistemin davranışının veya problemin tanımının düğümler ve düğümler arasındaki ilişkilerin ayrıtlarla yapılabildiği bu durumda graflarla ilgili önceden yapılan tanımlar, teoremler ve algoritmalar bu tür problemlerin çözümüne yardımcı birer araç olurlar. Graf Teorisi günümüzde bilgisayar uygulamalarında ve modellemelerde çokca kullanılmaktadır. Şehir içi taşımacılıktaki duraklar, durakların birbirlerine olan yol bağlantıları ve bu yolların özellikleri graflar ile tanımlanabilmektedir. Yerleşim birimlerinin birbirlerine olan yol bağlantıları ve bu yolların özellikleri graflar ile gösterilerek ifade edilebilir; bunun dışında, internet üzerindeki bilgisayar ağlarının birbirlerine olan bağlantıları da bir grafla gösterilebilmektedir [Jonathan G, 1999].

Graflarla ifade edilebilecek problem ve olayları incelemek ve bunları temel alarak çözümler üretmek takip edilen en ideal yoldur. Bu gibi problemlerin çözümü, yapıları itibariyle graf veri modeline yakındır. Eğer bir problemin çözümü graf veri

modeli yapısına benzetilebiliyorsa, bu problem için algoritmik bir durum elde edilmiş olur ve problemin çözümünde tanımlanmış tüm graf aksiyom ve graf teorileri kullanılabilir.

Graf Teorisinin bir konusu olan en kısa yol problemi, belirlenen iki düğüm arasında, düğümlerin oluşturduğu ayrıtların ağırlıkları toplamını minimize etmeyi konu alan bir problemdir. Bir noktadan başka bir noktaya giderken en kısa yolun tercih edilmesini ve böylece maliyetin azaltılmasını amaçlayan bir yaklaşımdır. Bir yol haritası üzerinde, bir yerden başka bir yere giderken en kısa yolu bulma en kısa yol problemine örnek olarak verilebilir. Bu durumda, hedef düğüm noktası, gidilecek olan yeri, ayrıtlar yolların bölümlerini, ağırlıklar ise ayrıtların uzunluklarını temsil etmektedir. En kısa yol problemleri, düğüm ve ayrıtlardan oluşan graflar yardımıyla modellenebilir. Şehirler arasındaki karayolu ulaşım sistemi ve uçuşların gösterildiği havayolu sistemi buna örnek olarak gösterilebilir. Şehir içi taşımacılıktaki duraklar, durakların birbirlerine olan yol bağlantıları ve bu yolların özellikleri graflar ile gösterilerek ifade edilebilir. Eğer problem uzaklıklarla ilgiliyse yollar kilometre birimiyle, kar-zarar gibi değerleri içeriyorsa yollar para birimi ile, zaman ile ilgili ise saat birimiyle isimlendirilir. Bunlara ağırlıklı graf denir. Bir çok problem ağırlıklı graflarla temsil edilebilir [Akman, 2002].

2.1. Graf Kavramları ve Tanımlar

Graflar, ayrık matematiksel yapıların önemli konularından biridir. Graflar konusunda matematiksel anlamda geliştirilen çok sayıda teorem, aksiyom ve algoritmalar yazılım dünyasında yoğun olarak kullanılmaktadır. Aralarında yol anlamında bağlantı olan her çeşit uygulama graf veri modeline uyarlanabilmektedir.

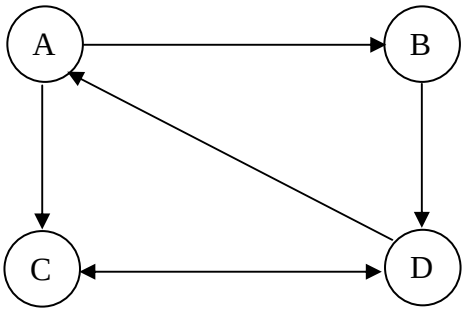
Graf, matematiksel tanım olarak, düğümler ve bu düğümler arasındaki ilişkiyi gösteren ayrıtlardan meydana gelen bir kümedir. Buradaki mantıksal ilişki düğüm ile düğüm veya düğüm ile ayrıt arasında kurulmaktadır. Grafın, grafik gösterilimi Eş. 2.1'de gösterildiği gibi düğümler ve bunları birbirine bağlayan ayrıtlarla yapılmaktadır. Bir graf üzerinde n tane düğüm ve m tane ayrıt varsa, matematiksel

ifadesi, düğümler ve ayrıtlar kümesinden elemanların ilişkilendirilmesiyle sağlanmaktadır.

$$\begin{aligned} D &= \{d_0, d_1, d_2 \ldots\ldots d_{n-1}, d_n\} \rightarrow \text{ Düğümler Kümesi} \\ K &= \{k_0, k_1, k_2 \ldots\ldots k_{m-1}, k_m\} \rightarrow \text{ Ayrıtlar Kümesi} \\ G &= \{D, K\} \Rightarrow Graf \end{aligned} \qquad (2.1)$$

Bir G grafı üzerindeki d_i ve d_j adlı iki düğüm, ayrıtlar kümesinde bulunan bir ayrıtla ilişkilendiriliyorsa bu iki düğüm birbirine komşu düğümlerdir; k ={d_i, d_j} biçiminde gösterilmektedir ve k ayrıtı hem d_i hem de d_j düğümleriyle bitişik olarak belirtilmektedir. Başka bir ifadeyle k ayrıtı d_i ve d_j düğümlerini birbirine bağlar veya d_i ve d_j düğümleri k ayrıtının uç noktalarıdır denilmektedir.

Bir G grafı üzerindeki ayrıtlar bağlantılarının nereden başlayıp nerede sonlandığını belirten yön bilgilerini içeriyorsa yönlü graf veya yönlendirilmiş graf (directed graf) olarak adlandırılmaktadır. Yönlü graf üzerindeki bir ayrıt, iki düğüm arasında bağlantı olduğunu, fakat ilişkinin oka bağlı olarak tek yönlü olduğunu göstermektedir. İki düğüm arasında karşılıklı bir ilişki varsa zıt yönde iki ayrıt kullanılmaktadır. Yönlü graflar, matematiksel olarak gösterilirken < > karakter çiftiyle gösterilmektedir. İki düğüm arasında karşılıklı bağlantı bulunuyorsa farklı yönde iki ayrıt kullanılmaktadır. <C, D> ve <D, C> olduğu gibi. Yönlü grafların komşuluk matrisleri, her bir bağlantı için birebir karşı bağlantısı yoksa simetrik olmamaktadırlar. Şehir içi ulaşım ağları yönlü graflara bir örnektir. Şekil 2.1'de yönlü graf örneği verilmektedir.



Şekil 2.1. Yönlü graf örneği

Şekil 2.1 ‘deki yönlü grafın matematiksel gösterimi verilmektedir.

$$G_{dd} = \{< A,B >, < A,C >, < B,D >, < C,D >, < D,C >\}$$

Bir G grafı üzerindeki ayrıtların ağırlıkları veya sayısal değerleri varsa ve bu değerler eşit değilse, ayrıca her biri farklı bir değer alabiliyorsa, bu graf maliyetli graf (weighted graph) olarak adlandırılmakta ve maliyet bilgisi ile birlikte gösterilmektedir. Eğer tüm ayrıtların maliyeti bir birim veya birbirine eşitse maliyetli graf olarak adlandırılmamaktadır. Eğer yön bilgisi de bulunmuyorsa bu graflara basit graf denilmektedir. Basit graf yönsüz ve maliyetsiz olan graftır. Maliyetli graflarda bazen ayrıtların simetrili gösterimine ihtiyaç duyulabilmektedir. Böyle bir durumda, graf üzerindeki ayrıt iki kez gösterilmiş olabilmektedir.

Düğümlerden düğümlere olan bağlantıyı göstermek için kullanılan kare matrise komşuluk matrisi (Adjacency Matrice) denilmektedir. Komşuluk matrisinin elemanları k_i değerlerinden oluşmaktadır. Komşuluk matrisi G_{dd}’nin matris biçiminde gösterilmesinden meydana gelmektedir. Eğer komşuluk matrisi $G_{dd} = [a_{ij}]$ ise, yönlü-maliyetsiz graflar için komşuluk matris şartı Eş. 2.2’deki gibi olmaktadır.

$$a_{ij} = \begin{cases} 1; (d_i , d_j) \in K \text{ ise} \\ 0; \text{Diğer durumlarda} \end{cases} \qquad (2.2)$$

Basit graflar için ise, Eş. 2.3’deki gibidir

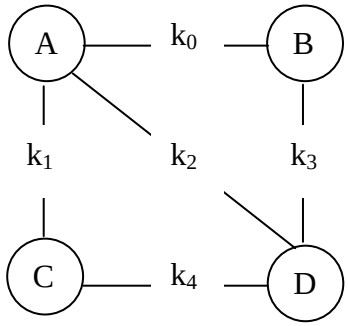
$$a_{ij} = \begin{cases} 1; (d_i , d_j) \in K \text{ veya } (d_j , d_i) \in K \text{ ise} \\ 0; \text{Diğer durumlarda} \end{cases} \qquad (2.3)$$

Düğümlerle ayrıtlar arasındaki bağlantı ilişkisini göstermek için kullanılan matrise bitişiklik matrisi (Incedence Matrice) denilmektedir. Matrisin satır sayısı düğüm, sütün sayısı da ayrıt sayısı kadar olmaktadır. Bitişiklik matrisi de G_{dk}’nın matris

şeklinde gösterilmesinden meydana gelmektedir. Eğer bitişiklik matrisi $G_{dk} = [m_{ij}]$ ise maliyetsiz graflar için bitişiklik şartı Eş. 2.4'deki gibidir.

$$m_{ij} = \begin{cases} 1; (d_i , d_j) \text{ Bağlantısı varsa} \\ 0; \text{Diğer durumlarda} \end{cases} \quad (2.4)$$

Bir grafın komşuluk ve bitişiklik matrisi Şekil 2.2'de gösterilmiştir.



A	A	B	C	D
A	∞	4	3	5
B	4	∞	∞	8
C	3	∞	∞	7
D	5	8	7	∞

M	k_0	k_1	k_2	k_3	k_4
A	4	3	5	∞	∞
B	4	∞	∞	8	∞
C	∞	3	∞	∞	7
D	∞	∞	5	8	7

a b c

Şekil 2.2 . a) Graf b) Komşuluk matrisi c) Bitişiklik matrisi

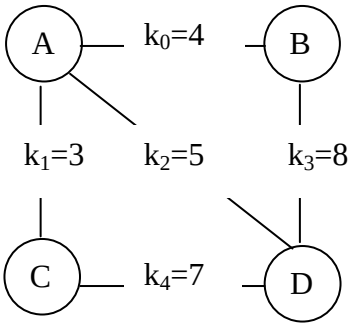
Şekil 2.2 a'da verilen grafın 4 adet düğümü ve 5 adet ayrıtı vardır. Düğümler D={A,B,C,D}, ayrıtlar da K = {k_0, k_1, k_2, k_3, k_4 } kümesinden oluşmaktadır. A olarak adlandırılan 4 x 4' lük komşuluk matrisinin hem satır hem de sütunları A, B, C, D olarak adlandırılmıştır. İki düğümün arasında doğrudan bir ayrıt var ise o iki satır ve sütunun kesiştiği yere bağlantı vardır anlamında 1 yazılmaktadır. Bağlantı yoksa 0 yazılır. Bir satırdaki sıfırdan farklı değerler o satıra ilişkin düğümlerin hangi düğümlere bağlantısı olduğunu göstermektedir.

Şekil 2.2 c'de verilen 4 x 6' lık M matrisinde satırlardaki sıfırdan farklı değerler o satırla ilgili düğüme bağlı olan ayrıtları göstermektedir. Bitişiklik matrisinde satırlardaki sıfır değerleri ilgili düğümün sütunda ifade edilen ayrıta bitişik olmadığını belirtmektedir.

Yönsüz graflarda komşuluk matrisi simetrik özellik taşımaktadır. Çünkü graf yönlü olmadığından A düğümünden B düğümüne olan bağlantı, aynı zamanda B

düğümünden A düğümüne olan bağlantıdır. Matrisin simetrik olma özelliği köşegen (diyagonal) çizgisine göredir.

Maliyetli graflarda, verilen graf maliyetlidir; ayrıtların pozitif, negatif ya da sıfır olabilen maliyetleri söz konusudur. Böyle bir durumda basit grafta olduğu gibi bağlantı yok anlamında sıfır sayısı kullanılmamaktadır. Çünkü sıfır maliyetli ayrıt olabilmektedir. Bunun yerine sonsuz (∞) yazılmalıdır. Şekil 2.3 'de gösterilen 4 düğümlü maliyetli grafın A komşuluk matrisinde, basit graftaki gibi 0 veya 1 değerleri değil de maliyet değerleri ve ∞ sayısı bulunmaktadır. Şekil 2.3'de graf, komşuluk matrisi ve bitişiklik matrisi verilmiştir.



a

A	A	B	C	D
A	∞	4	3	5
B	4	∞	∞	8
C	3	∞	∞	7
D	5	8	7	∞

b

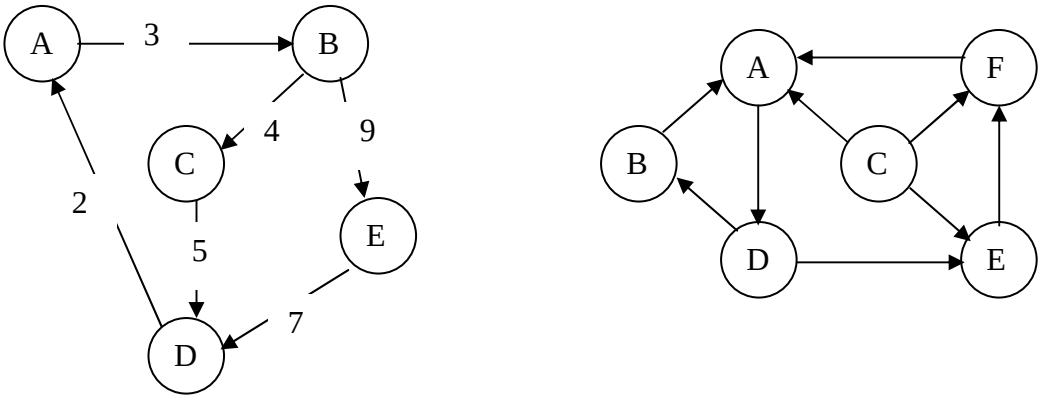
M	k_0	k_1	k_2	k_3	k_4
A	4	3	5	∞	∞
B	4	∞	∞	8	∞
C	∞	3	∞	∞	7
D	∞	∞	5	8	7

c

Şekil 2.3. a) Graf
b) Komşuluk matrisi
c) Bitişiklik matrisi

Bitişiklik matrisi M' de ise, her satırdaki sayısal değerler, o satıra bitişik olan ayrıtların varlığını ve maliyetini göstermektedir. Burada dikkat edilmesi gereken, her sütunda iki sayısal değerin varlığı ve bunların eşit olmasıdır. Çünkü bir ayrıtın iki ucu bulunmaktadır. Maliyetli graflarda komşuluk matrisi köşegen çizgisine göre simetrik özellik taşımaktadır. Çünkü bir düğümden diğer düğüme olan yön bilgisi olmadığından iki yönlüdür. Dolayısıyla A-B arasında bir yol varsa, B-A arasında da vardır. Matrisin simetrikliği çizilen diyagonal çizgisine göredir. Yönlü ve yönlü-maliyetli grafların komşuluk matrisleri simetrik özellik taşımamaktadır. Her bir ayrıt sadece bir düğümden diğerine olan bağlantıyı göstermektedir, ters yönde bir bağlantı olamayacağından simetrik olmaz. Bu tür graflara ait komşuluk matrislerinin simetrik

olabilmesi için her bir ayrıtın ters yönde eşleniğinin olması şarttır. Şekil 2.4'de yönlü maliyetli ve yönlü graf örnekleri gösterilmiştir. Bu grafların komşuluk matrisleri de Şekil 2.5'de gösterilmiştir.



a b

Şekil 2.4. a) Yönlü maliyetli graf b) Yönlü graf

A	A	B	C	D	E
A	∞	3	∞	∞	∞
B	∞	∞	4	∞	9
C	∞	∞	∞	5	∞
D	2	∞	∞	∞	∞
E	∞	∞	∞	7	∞

a

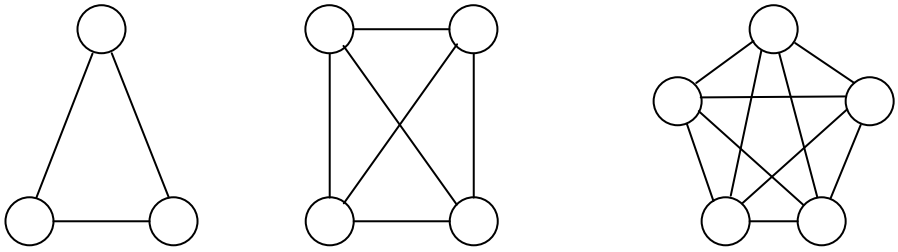
A	A	B	C	D	E	F
A	∞	∞	∞	1	∞	∞
B	1	∞	∞	∞	∞	∞
C	1	∞	∞	∞	1	1
D	∞	1	∞	∞	1	∞
E	∞	∞	∞	∞	∞	1
F	1	∞	∞	∞	∞	∞

b

Şekil 2.5. a) Yönlü-maliyetli graf b) Yönlü graf için komşuluk matrisi

Düğüme bağlı toplam uç sayısına o düğümün düğüm derecesi (node degree) denlmektedir. Çevrimli ayrıtlar, aynı düğüme hem çıkış hem de giriş yapabilmektedirler. Bu durum düğümün derecesini iki artırmaktadır. Yönlü graflarda, düğüm derecesi giriş derecesi ve çıkış derecesi olmak üzere ayrı olarak belirtilmektedir. Yönlü graflarda düğümün derecesi, düğümün giriş ve çıkış derecelerinin toplamına eşittir.

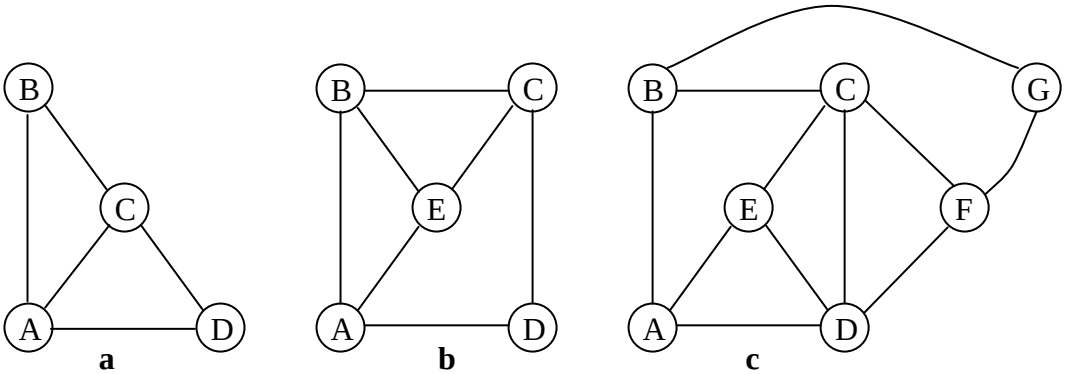
Her bir düğümü arasında doğrudan bağlantı olan graflara tamamlanmış graf (Complete Graph) adı verilmektedir. Tamamlanmış bir graf üzerindeki tüm düğümlerin dereceleri eşittir ve düğüm sayısından 1 eksiktir. Tamamlanmış bir grafın düğüm sayısı n ise bu graftaki her bir düğümün derecesi n–1 olur. Ayrıtların sayısı da n(n-1)/2'dir. Tamamlanmış graf yönlü olursa bu tanımlar geçerli değildir. Yönlü tamamlanmış graflarda her düğümden her düğüme, düğümün kendisi de dâhil, bir yol vardır. Şekil 2.6'da tamamlanmış graf ve dereceleri görülmektedir.



Şekil 2.6. Tamamlanmış graf örnekleri

Tüm düğümlerin derecelerinin eşit olduğu grafa Düzenli Graf (Regular Graph) denilmektedir. Tamamlanmış graf aynı zamanda düzenli bir graftır. Fakat bunun tersi her zaman geçerli değildir. Yani, düzenli graf her zaman tamamlanmış graf değildir.

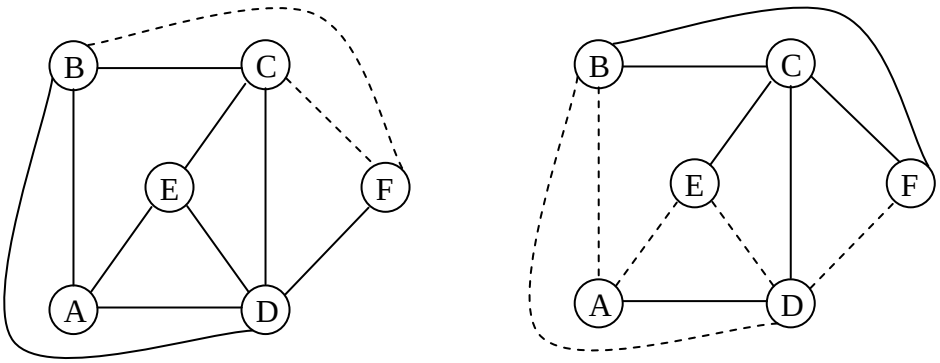
Hiçbir ayrıtı birbirini kesmeyen grafa Düzlemsel Graf (Planar Graph) adı verilmektedir. Düzlemsel graflar, ayrıtları kesişmeden herhangi bir düzleme rahatlıkla çizilebilirler. Bu yüzden harita graf ya da kabaca harita olarak da adlandırılırlar. Şekil 2.7'de düzlemsel özellik taşıyan graf örnekleri verilmiştir.



Şekil 2.7. Düzlemsel graf örnekleri

İki düğüm arasında doğrudan veya dolaylı olarak bir bağlantının ya da erişimin olmasına yol (path) denilmektedir. İki farklı düğüm arasındaki yol üzerinde aynı düğüme iki defa uğranılmamalıdır. Herhangi bir düğümden, gidilecek olan diğer bir düğüme doğrudan ulaşılabileceği gibi, başka düğümlerden geçilerek de ulaşılabilmektedir. Böyle bir durumda yol, geçilecek olan düğümlerin (d_0, d_2, d_5, d_1) şeklinde listelenmesiyle gösterilebileceği gibi, takip edilen ayrıtların (d_0, d_2), (d_2, d_5), (d_5, d_1) şeklinde yazılmasıyla da ifade edilebilmektedir. Birbirine komşu iki düğüm arasındaki yolların uzunluğu 1'dir, n uzunluğundaki yol, iki düğüm arasında n tane ayrıt geçilerek bağlantı yapılması anlamına gelmektedir.

Bir graf üzerinde, herhangi bir düğümden, diğer tüm düğümlere herhangi bir çevrim oluşturmadan gidilen yola, yol ağacı (spanning tree) adı verilmektedir. Grafta birden fazla yol ağacı olabilmektedir. Bir yol ağacı üzerinde ayrıtların sayısı, eğer grafta ayrık bir düğüm yoksa n-1'dir. Şekil 2.8'de A-B arasında iki ayrıtlı bir yol ve yol ağacı verilmektedir.



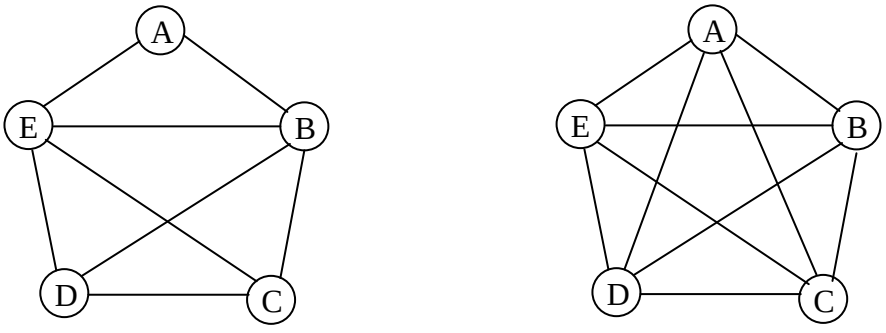
Şekil 2.8. Bir graf üzerinde yol ve yol ağacı
a) İki uzunluktaki bir yol (Kesikli çizgiler)
b) Tipik bir yol ağacı (Kesikli çizgiler)

Başlangıç ve bitiş düğümü aynı olan kapalı yol çevrim (cycle) olarak adlandırılmaktadır. Çevrim uzunluğu, kapalı yol üzerindeki ayrıtların sayısı kadardır. Kendi üzerinde, tek bir düğümü içine alan yolun uzunluğu 1 olur. Bir graf üzerinde geçilen ayrıtların tekrarlanmadığı bir dolaşıma gezi (trail) adı verilmektedir.

Birbirine bitişik olan düğüm-ayrıt-düğüm sıralanmasına dolaşım (walk) denilmektedir. Diğer bir ifadeyle düğümler arasında kendilerine bağlı bulunan ayrıtlar üzerinden gidilebilmektir. Eğer başlangıç düğümü ile bitiş düğümü aynı ise kapalı, değilse açık dolaşım yapılmıştır. Dolaşımın uzunluğunu geçilen ayrıt sayısı oluşturmaktadır.

Maliyetli bir grafta tüm düğümleri içine alan ve en az maliyetle yapılan kapalı bir dolaşımın bulunması problemine gezgin satıcı (traveling salesman) problemi adı verilmektedir. Bu şekilde, tüm düğümlere uğranılmaktadır ve dolaşım bitip başlangıç noktasına geri dönüldüğünde diğer alternatiflere göre en az maliyetli yol gidilmektedir. Bu tür problemler, dağıtım şirketleri vs. gibi uygulama alanlarında kullanılabilmektedir.

Her düğümden yalnızca bir kez geçilmesi esasına dayanan ve kapalı bir dolaşım gerçekleştirilebilen graf Hamilton Grafı (Hamilton's Graph) olarak adlandırılmaktadır. Graftaki her düğümü içine alan kapalı bir yolun olmasına Hamilton Çevrim denilmektedir. Gezgin satıcı problemi en az maliyetli Hamilton çevrim bulunmasına dayanan bir problem türüdür. Şekil 2.9 a'da Hamilton ve Euler grafına örnek verilmiştir.



a b

Şekil 2.9. a) Hamilton Grafı
b) Euler Grafı

Graf üzerinde ayrıtların tekrarlanmadığı kapalı bir dolaşımın yapılabileceği grafa Euler Grafı (Euler's Graph) adı verilmektedir. Bu tür graflar üzerinde bütün

düğümlerin dereceleri çift olmaktadır. Graf üzerinde her ayrıtı kapsayan, kapalı bir gezi yapılabilecek bir yolun bulunmasına Euler çevrim denilmektedir. Şekil 2.9 b'de bu grafa bir örnek verilmiştir.

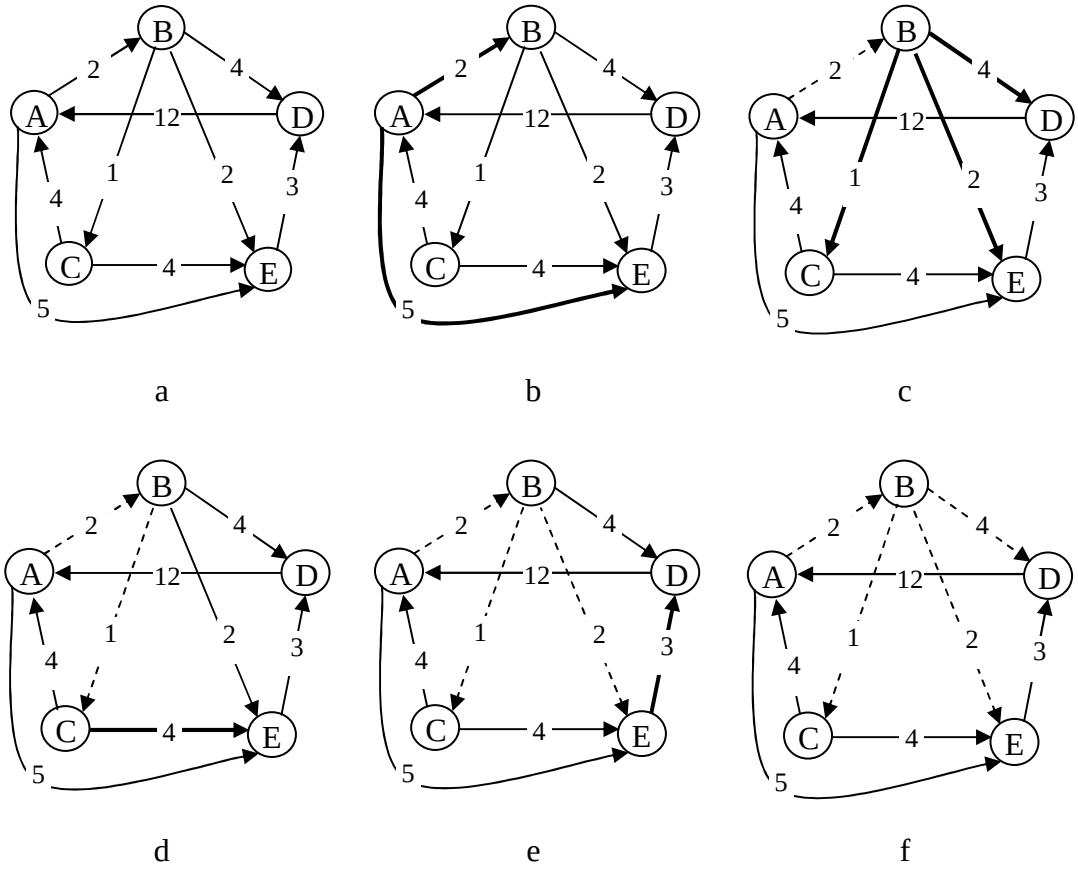
2.2. Greedy Yaklaşımı / Yöntemi

Greedy yaklaşımı, herhangi bir graf üzerinde belirli bir konuda en uygun sonucu veya en iyi sonucu bulmak amacıyla dolaşılırken bir sonraki düğümü belirlemek amacıyla kullanılan karar verme yöntemidir. Greedy yaklaşımında amaç, o andaki seçenekler arasından en iyi olanı seçmektir, bu seçimin ölçütleri yerel değerlendirmelere göre yapılmakta ve seçilenin, genel olarak tüm sistem için en iyi seçim olduğu varsayılmaktadır.

Greedy yaklaşımı her zaman en iyi sonuca ulaştırmayabilir. Ama çoğunluk olarak uygun bir sonuç elde edilmektedir. Greedy yaklaşımı özellikle bilgisayar bilimlerinde graflara uyarlanmış birçok problemin çözümünde, gelinen düğümden sonraki düğümün belirlenmesinde seçme elemanı olarak kullanılmaktadır. En kısa yol ağacı ve en kısa yol bulan algoritmalar buna örnektir. Greedy yaklaşımının tam tersi dinamik programlamayla yapılan değerlendirmedir. Dinamik programlamada en iyi olan yerel değil, genel bir değerlendirme ile belirlenmektedir. Yani, dinamik programlamada, bir sonraki karar elemanı, programın sonuna kadar gidilip görülmesi ve geriye dönüldükten sonra karar verilmesi esasına dayanmaktadır. Bunun için, dinamik programlamaya dayalı bir çözümün olabilecek uygun çözüm olduğu söylenilebilmektedir. Greedy yaklaşımında, yönlü-maliyetli graf üzerinde en kısa yol maliyetinin belirlenmesi problemi ele alınmaktadır. Problemde, belirli bir başlangıç noktasına göre diğer düğümlere erişmek için gereken en az maliyetli yolu bulmaya çalışılmaktadır. Greedy yaklaşımında en önemli adım, o anda bulunan düğümden bir sonraki düğüme geçişte yerel bir karar vererek gidilecek en iyi düğümü seçmektir. Greedy yaklaşımı birçok graf algoritmasında ilerleme seçim unsuru olarak kullanılmaktadır. Eksi maliyeti olmayan maliyetli graflar üzerinde, bir düğümden diğerine en kısa yol maliyetlerini ve en kısa rotayı belirlemede kullanılan Dijkstra algoritması, Greedy yaklaşımına dayanmaktadır.

2.3. Graf Algoritmaları

Graflar mühendislik, sosyal bilimler ve temel bilimlerde problemlere algoritmik bir çözüm sunmaktadır. Sistemin davranışının veya problemin tanımının düğümler ve düğümler arasındaki ilişkilerin ayrıtlarla yapılabildiği bu durumda graflarla ilgili önceden yapılan tanımlar, teoremler ve algoritmalar bu tür problemlerin çözümüne yardımcı birer araç olurlar. Örneğin iki şehir arasındaki olası en kısa yolun bulunmasında graf algoritmaları kullanılarak algoritmik çözümler elde edilebilmektedir. Şekil 2.10'da yönlü maliyetli graflara örnek verilmiştir [Çölkesen, 2002].

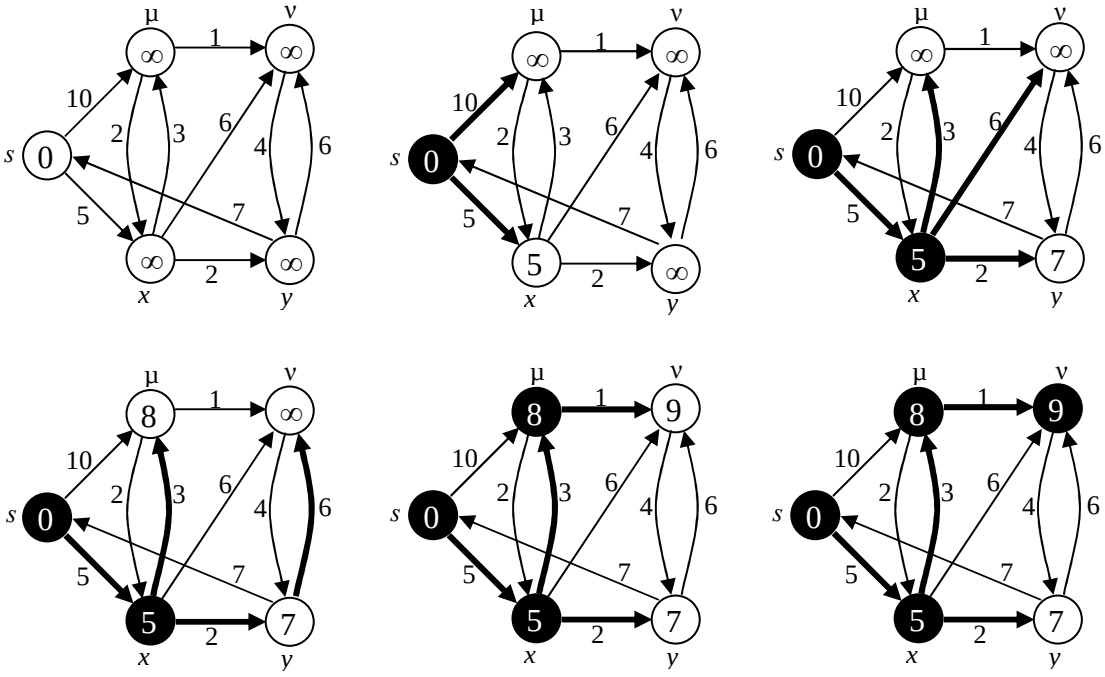


Şekil 2.10. a) Örnek yönlü maliyetli graf
b) A başlangıç düğümü
c) B düğümüne en kısa yol 2
d) C düğümüne en kısa yol 3
e) E düğümüne en kısa yol 4
f) D düğümüne en kısa yol 6

2.3.1. Dijkstra algoritması

En kısa yol probleminin çözümü için kullanılan ilk algoritma olan Dijkstra algoritması, Alman bilim adamı Dijkstra tarafından 1959 yılında bulunmuştur. Algoritma, baslangıç düğümü ile birinci düğümün arasındaki en kısa yolun bulunması, baslangıç ile ikinci düğüm arasındaki en kısa yolun bulunması böylece tekrarlı olarak baslangıç ile diğer tüm düğümler arasındaki en kısa yolların bulunmasına kadar devam etmektedir [Dijkstra, 1959].

Dijkstra algoritması, belirli bir başlangıç noktasına göre en kısa yolu bulan bir algoritmadır. Bir başlangıç düğümünden, diğer tüm düğümlere olan en kısa yolu belirler. Ağırlıklı ve yönlü graflar için geliştirilmiştir. Graf üzerindeki her bir ayrıtın ağırlığı sıfır veya sıfırdan büyük olmalıdır. Ancak burada da eksi maliyetli çevrim olmaması gerekmektedir. Dijkstra algoritması en kısa yolu belirlerken Greedy yaklaşımını kullanmaktadır. Her defasında, bir sonraki düğüme ilerleme Greedy yaklaşımına göre yapılmaktadır. Şekil 2.11’de 6-düğümlü örnek maliyetli graf üzerinde Dijkstra algoritmasının davranışı gösterilmiştir.



Şekil 2.11. Djkstra algoritması davranışı

Şekil 2.11'de maliyetli bir grafın başlangıç durumu ve birinci, ikinci, üçüncü, dördüncü, beşinci ve son adımdaki davranışı görülmektedir. Başlangıç düğümü A olarak kabul edilen maliyetli grafta ilk işlem olarak, A düğümünden doğrudan gidilebilecek düğümlere ait maliyet ve rota bilgilerinin güncellenmektedir. Bir sonraki düğüme gitme işlemi Greedy yaklaşımına göre gerçekleştirilmektedir.

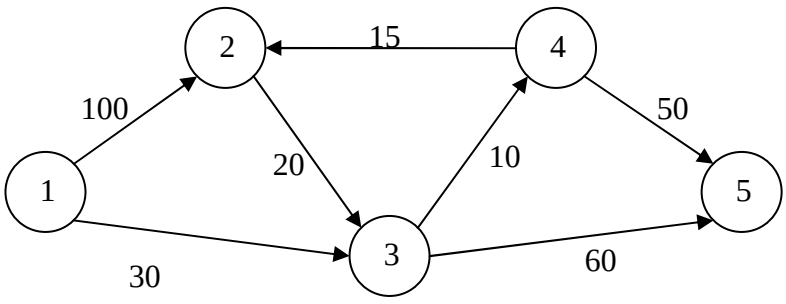
Dijkstra algoritmasında her düğüm için etiket verilmektedir. Bu etiketler geçici ve kalıcı olarak değer almaktadırlar. Daha iyi bir yol bulunana kadar etiket değeri geçicidir. Eğer en iyi yol bulunmuşsa etiket kalıcıya dönüştürülmektedir. Bu algoritma sadece, seçilen iki düğüm noktası arasındaki en kısa yolu vermektedir. Algoritmanın adımları şunlardır:

0. Adım : Kaynak düğümünü kalıcı olarak etiketle. Adım sayısı i' yi 1 yap.

i. Adım : j'nin "kalıcı" etiketlenmemiş olması şartıyla, i düğümünden ulaşılabilen her j düğümü için geçici mesafeleri hesapla. Eğer j düğümü başka bir k düğümü ile zaten etiketlenmişse ve daha iyi bir i, j bağlantısı bulunuyorsa j, k bağlantısını i, j bağlantısı ile değiştir. Tüm düğümlerde "kalıcı" etiketi varsa dur. Aksi halde tüm "geçici" etiketleri arasından en kısa mesafeli bağlantıyı seç ve i. adımı tekrarla.

Dijkstra algoritması örneği:

Dijkstra algoritmasının adımları ve çözümü Şekil 2.12'deki şebeke kullanılarak gösterilmektedir.



Şekil 2.12. Dijkstra algoritması örneği

0. adım:

1.düğüme [0 , -] kalıcı etiketi atanır.

1. adım:

2. ve 3. düğümlere 1.düğümden (en son kalıcı etiketlenen) ulaşılır ve düğümler Çizelge 2.1'deki gibi etiketlenir.

Çizelge 2.1. Dijkstra algoritması örneği birinci adım

Düğüm	Etiket	Statü
1	[0, -]	kalıcı
2	[0+100,1]=[100,1]	geçici
3	[0+30,1]=[30,1]	geçici

3. düğüm en kısa yolu verdiği için bir sonraki çizelgede statüsü kalıcı olarak işaretlenmektedir.

2. adım:

4. ve 5. düğümlere 3.düğümden ulaşılır. Yeni etiketleme Çizelge 2.2'deki gibidir.

Çizelge 2.2. Dijkstra algoritması örneği ikinci adım

Düğüm	Etiket	Statü
1	[0, -]	Kalıcı
2	[100,1]	geçici
3	[30,1]	Kalıcı
4	[30+10,3]=[40,3]	geçici
5	[30+60,3]=[90,3]	geçici

3. adım:

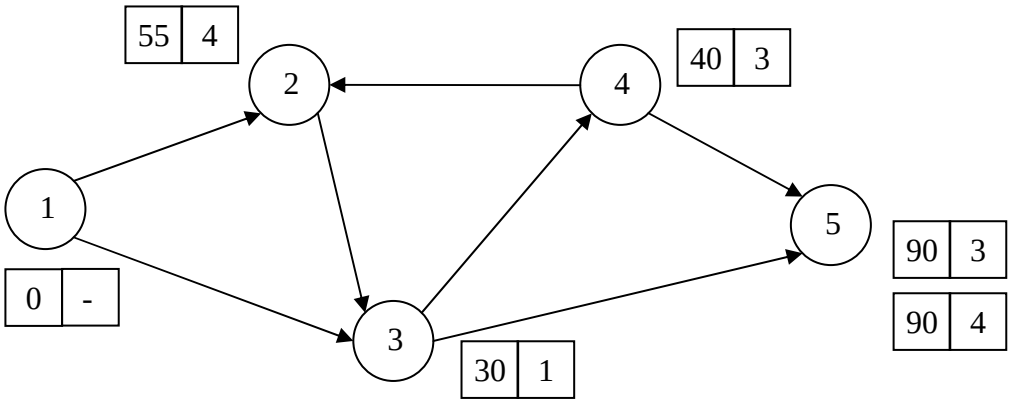
2. ve 5. düğümlere 4.düğümden ulaşılabilir. 4. düğümün statüsü değiştirilir. 2. düğümün etiketi daha kısa yol bulunduğundan değiştirilmektedir. 5. düğümün 2 seçeneği bulunmaktadır. Değişiklikler Çizelge 2.3'te gösterilmektedir.

Çizelge 2.3. Dijkstra algoritması örneği üçüncü adım

Düğüm	Etiket	Statü
1	[0, -]	kalıcı
2	[40+15,4]=[55,4]	geçici
3	[30,1]	kalıcı
4	[40,3]	kalıcı
5	[90,3]veya [40+50,4]=[90,4]	geçici

4.adım:

2. düğümden sadece 3. düğüme gidilebilmektedir. 3. düğümün etiketi kalıcı olduğu için yeniden etiketlenememektedir. 2. düğümdeki etiket de kalıcı olarak işaretlenmiştir. 5. düğümden diğer düğümlere gidiş olmadığı için işlem tamamlanmıştır.



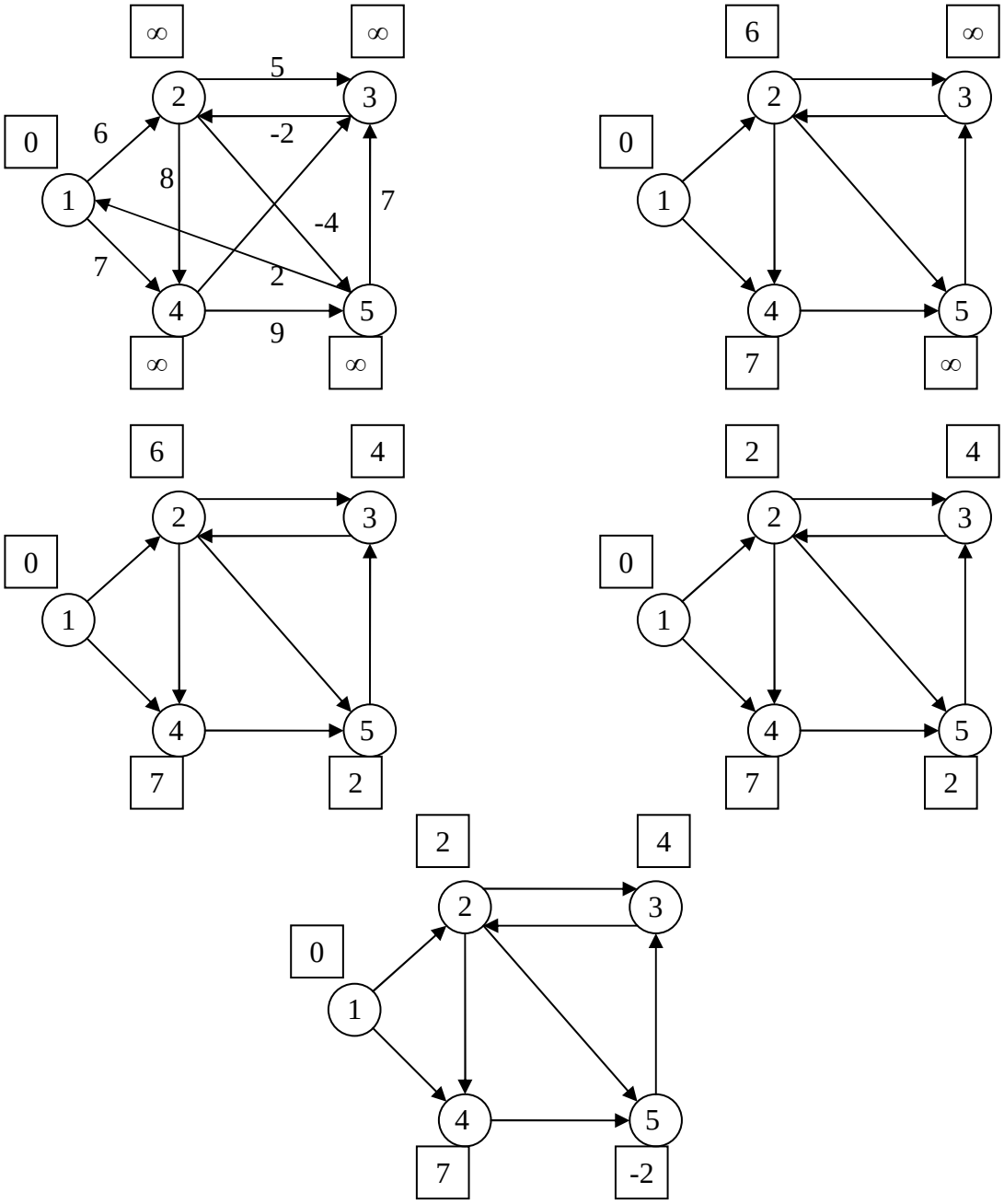
Şekil 2.13. Dijkstra algoritması örneği çözüm grafı

2.3.2. Bellman - Ford algoritması

Bellman-Ford algoritması, tek kaynaklı en kısa yol problemleri için etkili bir yöntemdir. Bellman-Ford algoritması, Dijkstra algoritmasında olduğu gibi bir başlangıç düğümünden diğer tüm düğümlere olan en kısa yolu bulmaktadır. Bazı durumlarda grafın eksi maliyetli değerler alması söz konusu olabilmektedir. Dijkstra algoritması ayrıt maliyetleri sıfır veya artı olan graflar için doğru sonuç verirken

Dijkstra'dan farklı olarak Bellman-Ford algoritmasının ayrıt maliyetleri eksi olan graflar için de uygulanabilmektedir.

Bellman-Ford Algoritması Örneği :



Şekil 2.14. Bellman-Ford algoritması çözüm adımları

Bellman-Ford algoritmasının adımları şunlardır:

1. Adım: Her düğümün diğer düğümlere olan uzaklıkları hesaplanarak bir tabloda tutulmaktadır.

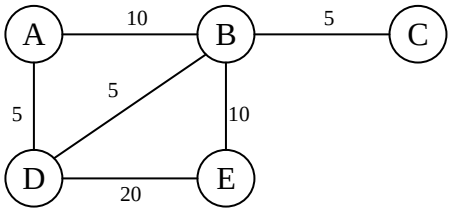
2. Adım: Her düğümün tablosu, komşu düğümlere gönderilmektir

3. Adım: Her düğüm, komşusundan uzaklıklar tablosunu aldığında, diğer düğümlere olan en kısa yollar hesaplanır ve tablo değerleri değişiklikler ile güncellenmektir.

2.3.3. Floyd algoritması

Floyd algoritması Dijkstra algoritmasının daha genel halidir. Çünkü Floyd algoritması, şebekedeki herhangi iki düğüm arasındaki en kısa yolu belirlemektedir. Floyd algoritması graf üzerinde her bir düğüm için diğer düğümlere olan en kısa yolları bulan en genel algoritmadır. Floyd algoritmasının verdiği sonuç, Dijkstra algoritmasının graftaki düğüm sayısı kadar çalıştırılmasıyla da elde edilebilmektedir. Floyd algoritmasının yoğun graflarda daha iyi sonuçlar vermektedir. Bunun sebebi, çok seyrek graflarda Dijkstra algoritmasının düğüm sayısının katı kadar çalıştırılmasıyla iyi sonuçlar alınabilmektedir. Floyd algoritması, grafın komşuluk matrisi, sonucun tutulacağı uzaklık matrisi ve en kısa yol bilgilerinin tutulacağı rota matrisi bilgilerini kullanmaktadır. Başlangıç düğümünün komşuluk matrisindeki maliyeti, başlangıç maliyeti olarak kabul edilmektedir. Floyd algoritması, n düğümlü şebekeyi n satırlı ve n sütunlu kare matris olarak göstermektedir. Matrisin (i , j) elemanı, i. düğümden j. düğüme olan uzaklığı vermektedir. Doğrudan i, j' ye bağlıysa düğümler bağlantı değeri almaktadır. Eğer düğümler arasında doğrudan bağlantı yok ise bu düğümler arasındaki maliyetler ∞ yapılmaktadır. Uzaklık matrisinde ise, başlangıçta hiçbir bilgi olmadığını göstermek amacıyla tüm bölmeler -1 ile doldurulmaktadır. Şekil 2.15' de Floyd algoritması ile çözüm adımları verilmiştir [Floyd ve ark., 1983].

Floyd algoritması örneği :



Şekil 2.15. Floyd algoritması graf örneği

Şekil 2.15'de verilen graf için A düğümünden E düğümüne olan yol bilgileri Çizelge 2.4'de verilmiştir.

Çizelge 2.4. Floyd algoritması yol bilgileri

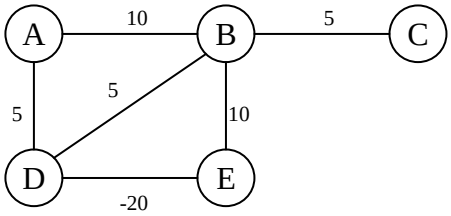
Yol 1	:	A	->	B	->	E		20
Yol 2	:	A	->	D	->	E		25
Yol 3	:	A	->	B	->	D	-> E	35
Yol 4	:	A	->	D	->	B	-> E	20

Burada dikkat edilecek noktalar şunlardır:

1. İki düğüm arasında birden fazla düğüm bulunabilmektedir.
2. Rotada bulunan düğüm sayısı önemli değildir (yol 4, 4 düğüm içerirken, yol 2, 3 düğüm içermektedir).
3. En kısa uzunluğu veren birden fazla yol olabilmektedir.

Graftan diğer bilgiler de gözlemlenebilmektedir. Her yol göz önünde bulundurularak, her düğümden en az bir kere geçilmelidir.

Negatif ağırlıklara izin verildiğinde problem farklı olarak değerlendirilmektedir. Negatif ağrılıklar döngüde gözüktüğü zaman en kısa yol bulunamamaktadır. Şekil 2.16'da negatif ağırlıklı Floyd algoritması grafı görülmektedir.



Şekil 2.16. Negatif ağırlıklı Floyd algoritması graf örneği

Şekil 2.16'daki grafda B,E,D,B düğümleri arasında bir döngü bulunmaktadır. B Başlangıç düğümü aynı zamanda son düğümdür. Bu döngünün maliyeti 10–20+5=–5 değerini almaktadır. Eğer bu döngü herhangi bir yola eklenirse, yolun değerini 5 azaltacaktır. İki defa eklenirse, yolun değerini 2 * 5 = 10 azaltacaktır.

Floyd algoritması yolu bulmaz sadece uzunlukları bulmaktadır. Negatif değerlerin döngüye girmediği varsayılmaktadır. En kısa uzaklıkları bulmak için izlenecek adımlar listelenmektedir.

Çizelge 2.5. Floyd algoritması başlangıç matrisi

	A	B	C	D	E
A	0	10	0	5	0
B	10	0	5	5	10
C	0	5	0	0	0
D	5	5	0	0	20
E	0	10	0	20	0

A düğümü ile herhangi iki düğüm arasındaki boşluklar en kısa yolu bulmak için işleme alınmaktadır.

Çizelge 2.6. Floyd algoritması 1. matrisi

	A	B	C	D	E
A	0	10	0	5	0
B	10	0	5	5	10
C	0	5	0	0	0
D	5	5	0	0	20
E	0	10	0	20	0

B düğümü ile herhangi iki düğüm arasındaki boşluklar en kısa yolu bulmak için işleme alınmaktadır.

Çizelge 2.7. Floyd algoritması 2. matrisi

	A	B	C	D	E
A	0	10	15	5	20
B	10	0	5	5	10
C	15	5	0	10	15
D	5	5	10	0	15
E	20	10	15	15	0

Aynı işlemler C,D ve E için de yapılarak en kısa yolu bulmak amaçlanmaktadır.

Çizelge 2.8. Floyd algoritması 3. , 4. ve 5. matrisi

	A	B	C	D	E
A	0	10	15	5	20
B	10	0	5	5	10
C	15	5	0	10	15
D	5	5	10	0	15
E	20	10	15	15	0

	A	B	C	D	E
A	0	10	15	5	20
B	10	0	5	5	10
C	15	5	0	10	15
D	5	5	10	0	15
E	20	10	15	15	0

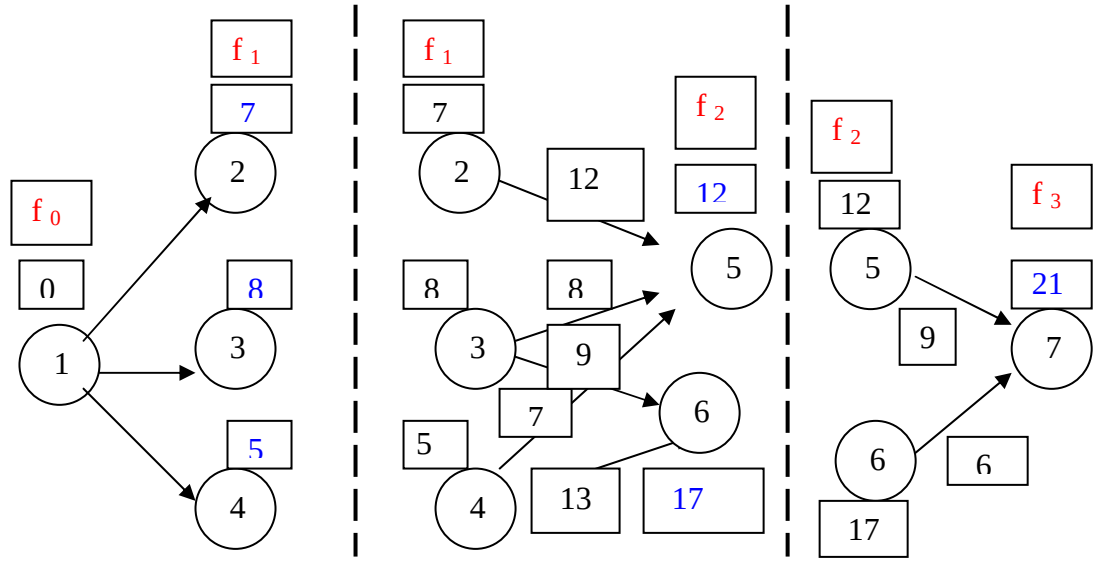
	A	B	C	D	E
A	0	10	15	5	20
B	10	0	5	5	10
C	15	5	0	10	15
D	5	5	10	0	15
E	20	10	15	15	0

2.3.4. Deterministik dinamik programlama

Dinamik programlama, n değişkenli bir problemin optimum çözümünü, problemi n aşamaya ayırarak ve her aşamada tek değişkenli bir alt problemi çözerek belirlemektedir. Hesaplamalar yinelenerek yapıldığı için bir alt problemin optimum çözümü bir sonraki problemin girdisini oluşturmaktadır. Son alt problem çözüldüğünde optimum çözüme ulaşılmaktadır. Algoritmada önemli olan konu problemin nasıl parçalara ayrıştırılacağıdır.

<u>Deterministik dinamik programlama örneği :</u>

Problemi dinamik programlama ile çözebilmek için önce parçalara ayırmak gerekmektedir. Şekil 2.17’de deterministik şebeke örneği verilmiştir.



Şekil 2.17. Deterministik şebeke modeli

1. adım:

2. düğüme en kısa uzaklık = 7 (1.düğümden)

3. düğüme en kısa uzaklık = 8 (1.düğümden)

4. düğüme en kısa uzaklık = 5 (1.düğümden)

2. adım:

5. düğüme en kısa uzaklık = 12 (4.düğümden)

6. düğüme en kısa uzaklık = 17 (3.düğümden)

3. adım:

7. düğüme en kısa uzaklık = 21 (5.düğümden)

7.düğüme en kısa uzaklık 21'dir. 3. aşamanın sonucundan hareket edilerek 7.düğüm, 5.düğüme bağlanmıştır. Daha sonra 2.aşamanın sonucuna bakılarak 5.düğüm, 4.düğüme bağlanmıştır. Son olarak 1.aşamanın sonucuna bakılarak 4.düğüm, 1. düğüme bağlanmıştır. Optimum yol 1, 4, 5 ve 7.düğümlerden geçmektedir.

3. SEYAHAT ŞEBEKESİNİN MODELLENMESİ

En kısa yol algoritmalarında, farklı şebekeler, bir veya daha fazla en düşük maliyetli yolu bulabilmek için farklı algoritmalara ihtiyaç duymaktadırlar. En basit olanları düğüm noktalarının ağırlıkları statik ve deterministik olanlarıdır. Eğer düğümlerin ağırlıkları statik değilse ve farklı zamanlarda değişkenlik gösteriyorsa, şebeke zaman bağımlıdır. Örnek olarak, toplu taşıma araçları olan otobüs ve trenlerin oluşturduğu şebekenin düğüm noktaları, tek bir deterministik ağırlığa sahip değildir. Bunun yerine herbir düğüm olasılıklı rastgele bir ağırlığa sahiptir. Bu tarz şebekeler stokastik şebeke olarak kabul edilmektedir ve genellikle trafik hesaba katıldığında, seyahat sürelerindeki belirsizlikten dolayı taşımacılıkta kullanılmaktadır.

Standart en kısa yol algoritmaları, şebekedeki yolların maliyetleri deterministik ve toplanabilir olduğu zaman en kısa yolu bulmada kullanılabilmektedir. Fakat, seyahat süreleri, gerçek taşımacılık şebekelerinde, özellikle trafikteki yoğunluk dikkate alındığında tahmin edilememektedir. Bu yüzden gerçek dünya taşımacılığındaki şebekeler stokastik ve zaman bağımlı olarak kabul edilmektedir.

En kısa sürede istenilen noktaya ulaşma isteği kullanıcıyı, ulaşım kaynaklarını etkin ve verimli kullanmaya, gerektiğinde kaynaklar arası aktarmalar yaparak daha hızlı ulaşım sağlamaya yönlendirmektedir. Günümüzün en önemli kaynaklarından biri olan zamanın verimli bir şekilde kullanılabilmesi için ulaşımda kullanılan kaynakların yazılım desteği ile kayıt altına alınıp kontrol edilme gerekliliği ortaya çıkmaktadır. Bilişim teknolojilerinin gelişmesiyle bir noktadan başka bir noktaya en az maliyetle gidilmesi problemi, maliyetin azaltılması yaklaşımının etkinliğini artırmış, bu nedenle de pek çok uygulama geliştirilmiştir.

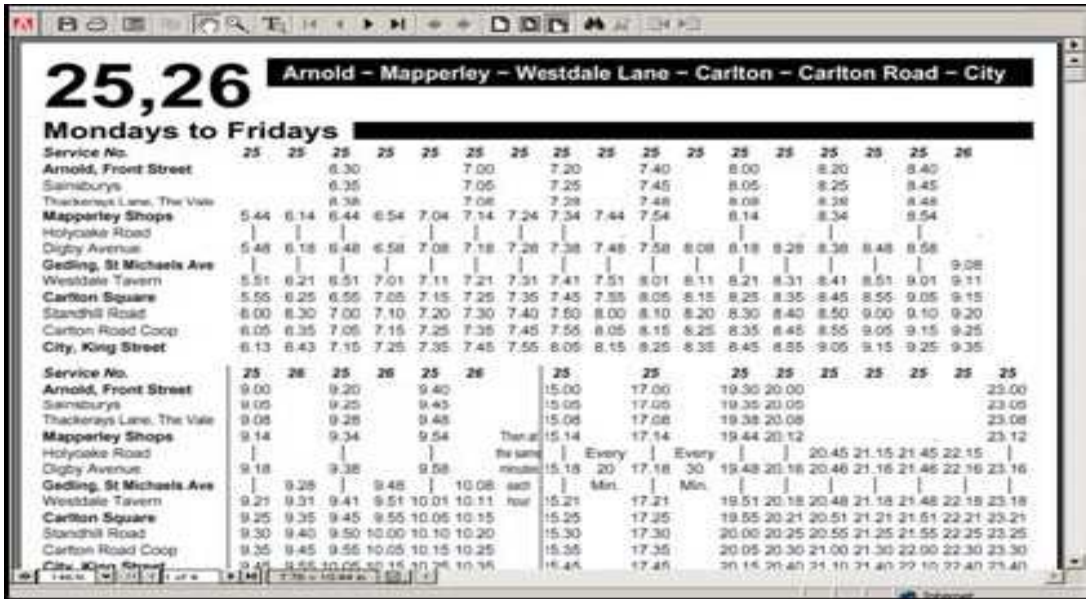
Otobüsle seyahatte rota bulmak için, otobüsün tarife bilgilerine ihtiyaç duyulmaktadır. Herhangi iki nokta arasındaki uzunluk ve süre bilgileri direk verilmediğinden bilgi otobüs tarife bilgilerinden alınmaktadır. Çizelge 3.1’ de otobüs bölge bilgilerinin hangi formatta olacağı gösterilmektedir.

Çizelge 3.1. Otobüs bölge bilgileri formatı

Durak No	Bölge	Adres	X Koordinatı	Y Koordinatı
Aa	Şehir	Angelrd	457050.58	339922.27
Ab	Şehir	Angelrd	457036.02	339928.82
Ac	Şehir	Angelrd	457005.98	339943.61
Ad	Şehir	Angelrd	456986.82	339951.85

3.1. Otobüs Tarife Programının Düzenlenmesi

Otobüs seyahat şebekesinin sunulabilmesi için otobüs tarife programının modellenmesi gerekmektedir. Şekil 3.1'de örnek bir otobüs tarife programı verilmektedir.



Şekil 3.1. 25,26 Numaralı otobüslerin tarife programı

İki otobüs durağı arasındaki beklenen seyahat süresini ölçmek için, iki durak arasına çizgi çekilerek iki durak arası mesafe ölçülmektedir. Aynı zamanda, otobüsün hareket saati ve bir sonraki durağa varış süresi gerekmektedir. Tarife programındaki her bir giriş bağlantılar ile ilgili bilgidir.

3.2. Yürüyüş Yollarının Modellenmesi

Seyahat şebekesine yürüyüş bağlantılarının eklenmesiyle yeni bir problem ortaya çıkmaktadır. Yürüyüş yollarının da dahil edildiği bir şebekede rotanın bulunabilmesi için iki nokta arasındaki yürüyüş yollarının uzunluğuna ihtiyaç duyulmaktadır. Model belirlenirken iki tip ayrıt bilgisinin tutarlı olması gerekmektedir. Farklı insanlar farklı hızlarda yürüyebilmektedir. Farklı yürüme hızları seyahat sürelerinde farklılık oluşturabilmektedir. Modele, seyahat süresi farklılıklarından dolayı tutarsızlık karışmaktadır.

Yürüyüş yolları arasındaki bağlantılar ile otobüs durakları arasındaki bağlantılar tutarlı bir şekilde modellenmelidir. Bunun sebebi, otobüs veya yürüme bağlantılarının aynı zamanda modelde kullanılabilmeleridir. Bu işlemin yapılabilmesi için, otobüs bölge bilgileri, kullanılabilir zaman bilgilerine dönüştürülmelidir. Seyahat şebekesi oluşturulurken, iki otobüs durağı arasındaki fiziksel uzaklık, beklenen yürüyüş süre bilgilerine çevrilebilmektedir. Örnek olarak, i ve j noktaları arasındaki fiziksel uzaklık Eş.3.1 ile hesaplanabilmektedir.

- distance(i,j)=sqrt((x(i)-x(j))**2+(y(i)-y(j))**2) (3.1)

Ortalama yürüme süresinin 5 km/s olması varsayımı altında i ve j noktaları arasında beklenen yürüme süreleri hesaplanabilmektedir.

- walktime(i,j)=distance(i,j)/(5000/60) (3.2)

Uzaklıkların yürüyüş sürelerine dönüştürülmesi sayesinde, otobüs ile seyahat veya yürüyerek seyahat tutarlı hale gelmektedir.

3.3. İki Modelli Şebekenin Sunumu

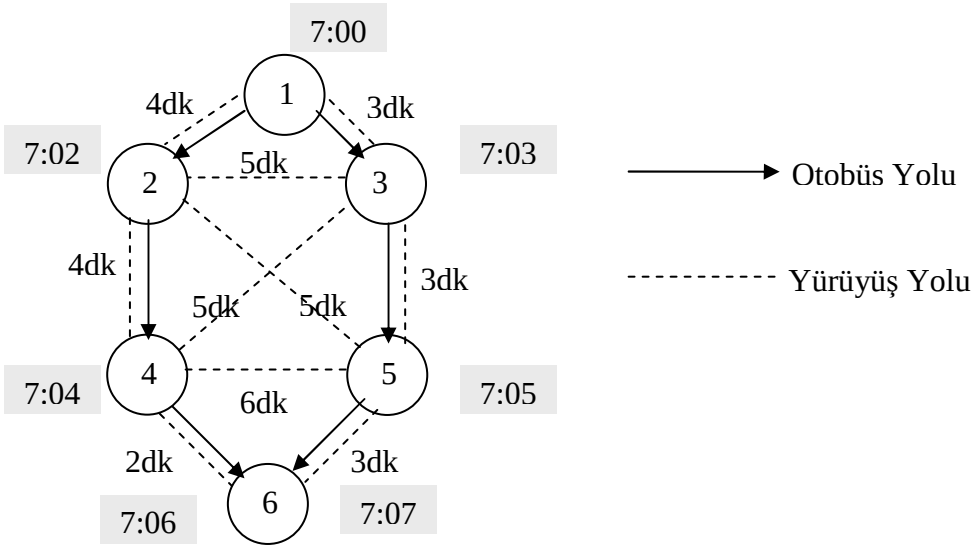
Tanımlama ve modellere bağlı olarak toplu taşıma şebekeleri tanımlanabilmektedir. Otobüs sistemindeki rotayı bulmak için G = (N, A) grafiği kullanılmaktadır.

N : sonlu n düğüm seti,

A : sonlu m ayrıt seti, arc(ij) Є A

l_{ij} : Uzunluk veya ağırlık değerleri, $l_{ij} \geq 0$

Modelin çözümü ile bulunan yol, taşımacılıkta kullanılacak olan şebekeyi ifade etmektedir. Düğümler otobüs duraklarını, arklar ise iki düğüm arasında ki yürüyüş ya da otobüsle seyahatinin alacağı süreyi göstermektedir. Modelde amaç, taşımacılık şebekesindeki iki nokta arasında, en kısa yol serisinden bir tanesini bulmaktır. Şekil 3.2'de şehir içi taşımacılık şebekesi verilmektedir.



Şekil 3.2. Şehir içi taşımacılık şebekesi

Şekil 3.2' de verilen örnek bir grafikte, numaralı alanlar düğüm noktalarını, çizgiler ve oklar da ayrıtları ifade etmektedir.

Şebekede gösterildiği gibi, bir sonraki rotayı bulma problemi, bağlantıların stokastik özelliği dikkate alındığında daha karmaşık bir hale gelmektedir. Zaman bağımlı stokastik en kısa yol algoritması için kullanılan çözüm tekniği, potansiyel çözüm olarak kabul edilmekte ve tarifeli taşımacılık şebekesinde uygulanmaktadır.

4. LİTERATÜR ARAŞTIRMASI

Bu bölümde, ayrıt ağırlıkları statik olmayan en kısa yol problemlerinin literatür incelemesi yapılmaktadır. Literatürde, bir klasik olarak kabul edilen rotalama problemleri için en kısa yol algoritmaları geliştirilmiş ve bu algoritmalar ile en kısa yol problemler çözülmüştür. Rotalama problemleri ve bu problemlerin türevleri çok sayıda araştırmacı tarafından incelenmiştir. Gruplanabilen problemler, ayrıt özellikleri (seyahat zamanı, maliyet) dikkate alınarak deterministik veya stokastik olarak ele alınmıştır. Bu alandaki birçok problem, bu iki grubun kombinasyonları şeklinde artmaktadır. Araştırmacıların çoğu, ayrıtların değerleri kesin olan deterministik şebekeler üzerinde çalışmışlardır.

4.1. Zaman Bağımlı En Kısa Yol Algoritmaları

Düğüm ağırlıkları statik olmayan ve zaman bağımlı olarak değişen şebekeler için bir takım algoritmalar geliştirilmiştir. Zaman bağımlı en kısa yol algoritmaları genellikle, otoyolun yoğunluğuna göre seyahat sürelerinin farklı olduğu şebekelerde kullanılmaktadır.

Zaman bağımlı şebekede, herhangi iki nokta arasındaki en kısa yolu bulabilmek için “Tek kaynaklı, negatif ağırlık ihtimalli” algoritması geliştirilmiştir [Bellman, 1958]. Bellman’ın dinamik programlama yaklaşımı için optimumluk prensibi geliştirilmiştir [Cooke ve ark. , 1966].

Standart Dijkstra algoritması, zaman bağımlı en kısa yol algoritması ile uğraşabilmek için değiştirilmiştir. Dijkstra’nın algoritmasını kullanarak zaman bağımlı en kısa yolu belirleme önerilmiştir [Drefyus, 1969].

İlk olarak Dreyfus’un çalışmasını sınırlandırılmıştır ve i noktasından ayrılış gecikmeli olabilir ya da döngü optimal yolu içermektedir [Halpern, 1977].

Olasılıklı durumlar için daha genel olarak, zaman bağımlı en kısa yol problemi üzerinde ilk olarak çalışılmıştır. Hat gecikmeleri için olasılık dağılımının, problemi koşullu olasılık teorisini kullanarak çözmek için dinamik programlama algoritması tayin edeceğini ve alt-optimal sonuçları vereceğini gösterilmiştir [Hall ve ark., 1986].

Zaman bağımlı en kısa yol problemini analiz edilmiş ve Halpern'in sonuçları, farklı durumlardaki bağlantı gecikme fonksiyonlarının özel durumları için geliştirilmiştir. Ayrıca Malandraki, tutarlılık varsayımına bağlantı fonksiyonunun birinci dereceden türevinin negatif değerlere geçmediği doğrulanması ile zaman bağımlı en kısa yol algoritmasına cevap vermiştir [Malandraki, 1989].

Zaman bağımlı şebekeler için, sürekli zamanlı bir şebekenin nasıl maliyetlere dönüştürülerek nasıl daha basit bir en kısa yol algoritmasına olarak kullanılabileceği üzerinde çalışılmıştır. Birçok tip düğüm noktalarında bekleme senaryolarını çalışılmış ve bu farklı durumlar için farklı algoritmalar önerilmiştir. Düğümlerde beklemelere izin veriliyorsa, tutarlılık varsayımının gerekmediğini gösterilmiştir. Şebekede başka bir yerde beklemeye izin verilmiyorsa, kaynak noktada optimal beklemeyi tanımlayabilmek için bir algoritma belirlenmiştir. Ayrıca, beklemelere izin verilmediği durumlarda, tutarlılık varsayımı olmaksızın elde edilen yolların basit olmayacağını ve problemin sürekli zaman versiyonunun NP-Hard olduğu gösterilmiştir [Orda ve ark., 1990].

Trafik denge problemi için, kaynak bekleme durumlarını analiz edilmiştir. Rota seçiminin hareket saatinden bağımsız olduğu varsayımı üzerinde çalışılmıştır ve optimal kaynak bekleme zamanlarının, kullanıcı için toplam maliyeti minimize etmeye bağlı olduğunu hesaplanmıştır [Friesz, 1993].

Zaman aralığı şebeke formülasyonu ve eğer problem tutarlılık varsayımını yerine getiriyorsa beklenen hat gecikmelerinin problem çözümünde kullanılabileceğini gösterilmiştir. Miller ve ark., hat gecikmelerinin olasılık dağılımı vasıtasıyla Pareto-optimal yolu elde etmek için etkin bir etiket doğrulama algoritması geliştirmişlerdir [Kaufman ve ark., 1993].

Çalışmada, zaman bağımlı en kısa yol problemine etkili bir çözüm yaklaşımı sağlanmıştır. Etkin çözümü sağlayabilmek için problem t zaman periyoduna bölünmüş ve bütün kaynaklar için pseudo-polinom zaman algoritması geliştirilmiştir. Kaynaklar, m kaynak noktasına sahip şebeke için O(m3t2) karmaşıklığına sahiptir. Bu algoritma, şebekede, en düşük maliyet ve en kısa yolu tanımlayabilmiştir. Şebeke, başlama zaman setini vermesi, ilk giren ilk çıkar kuralına cevap vermesi gerekli olmayan genel bağlantı seyahat maliyetlerine sahiptir [Ziliaskopoulos ve ark., 1993].

Zaman bağımlı en kısa yol algoritmaları için gerçekçi trafik hat bekleme fonksiyonlarını belirlemişlerdir. Gerçek zamanlı ve geçmiş gecikme bileşenlerinin ağırlıklarını veren, hat bekleme fonksiyonlarının etkili olması ve tayin edilebilmesini sağlamışlardır [Koutsopoulos ve ark., 1994].

Taşıma şebekesindeki bağlantılar ile zaman bağımlı doğal riskleri birleştirilmiştir. Miller aynı zamanda, tehlikeli maddelerin nakliyeleri için dinamik, minimum risk rotaları için zaman bağımlı en kısa yol uygulamasını önermiştir [Miller, 1994].

Çalışmada, araçların, kalabalık şehir içi şebekelerdeki dönüşlerinin seyahat zamanları için önemini belirtilmiştir. Ayrıca kavşak hareketleri ve hareket yasaklarını içeren bir şebekeyi tanımlamak, ileri-yıldız yapısının gelişmiş hali olan, etkin bir etiket-düzeltme prosedürü tanımlanmıştır [Mahmassani ve ark., 1996].

Yapılan çalışmada, akıllı taşıma sistemi uygulaması için etkili bir prosedür geliştirilmiştir [Ziliaskopoulos ve ark., 1997].

Bir şebekede zaman bağımlı gecikmeler ve maliyetleri bulmak için zaman bağımlı en kısa yol problemi analiz edilmiştir. Toplam gecikmeler önceden belirlenmiş değerlere küçük veya eşittir. Orda ve ark.'nın yaptığı çalışmadaki konulara benzer olarak birçok beklemeli model üzerinde çalışmışlardır [Cai ve ark., 1997].

Çalışmada, benzer bir problem analiz edilmiştir. Problemin bu çalışmadan farkı zaman değişkenini ve sırt çantası sabitlerini dikkate almasıdır [Haquari ve ark., 1997].

Zaman farklı ve düğüm ağırlıkları zaman bağımlı olan bir algoritma sunulmuştur [Chabini, 1998].

Yapılan çalışmada, maliyetin farklı versiyonlarını kullanılarak benzer sonuçlar verilmiş ve algoritmalarının etkinliğini kanıtlanmıştır [Sung ve ark., 2000].

Miller ve ark. çalışmasına benzer olarak Fu, ayrıt sürelerinin rastgele değişkenler ile modellenebildiği şebekeler için optimal rotalama algoritması geliştirmiştir. Optimal çözüm, gerçek zamanlı ayrıt süreleri ile tahmin edilerek gerçekleştirilmiştir [Fu, 2001].

Walter ve ark., problemi, sınırlı uzay ve ayrıt sürelerinin maliyetine bağlı olan statik bir şebekede ele almışlardır. Bu limitli bağımlılık, ayrıt maliyetlerinin dağıtılması ile modellenebilmiştir. Gerçek zamanlı en kısa yol problemlerini anlayabilmek için muhtemel gerçek zamanlı çözümlerin ilişkileri üzerinde çalışmışlardır. Problemin özel durumları için etkin bir algoritma geliştirilmiştir [Walter ve ark., 2002]
Diğer algoritmalar sürekli zaman modellerinde minimum maliyetli yol üzerinde yoğunlaşmışlardır.

4.2. Stokastik Network Algoritmaları

Diğer bir algoritma sınıfı ise düğüm noktalarına olasılık fonksiyonu ile rastgele değerler atanabilmesi ve seyahat maliyetlerini olasılıklı olarak gösteren çalışmalardır. Bu tür algoritmalar aynı zamanda, seyahat sürelerinin, kesinlik olarak bilinemeyen, trafikte geçen sürelerin sadece tahmin edilebildiği taşımacılık şebekeleri içinde kullanılabilir. Stokastik şebekelerde seçilen yol üzerinde, beklenen en az seyehat süresini bulma çalışmaları devam etmektedir.

Köşe ağırlıklarını, beklenen değerleri ile değiştirerek problemi standart en kısa yol problemi ile çözmeye çalışmıştır [Frank, 1969].

Çalışmada, bu stratejinin herzaman kullanılamıyacağını ve alt-optimal yol oluşturacağına dikkat çekilmektedir. Kesin sürelerde varışın, fayda fonksiyonunu kullanarak nasıl avantajlı olabileceği gösterilmiştir [Loui, 1983].

Verilen fayda fonksiyonun doğrusal veya üstel olduğu zaman, Dijkstra benzer bir algoritmanın kullanılabileceğini gösterilmiştir [Eiger ve ark., 1985].

Bu problemin çok karmaşık bir versiyonu üzerinde çalışılmış ve stokastik şebeke üzerindeki belirsizliği azaltan sezgisel yöntem sunulmuştur [Bard ve ark., 1991].

Çalışma, doğrusal ve ikinci dereceden fayda fonksiyonlarını kapsamaktadır. Bu fonksiyonlar ilk olarak Mirchandini ve ark. tarafından sunulmuştur [Mirchandini ve ark., 1985].

Murthy ve ark., bu alanda sürekli olarak çalışmışlar ve zaman kısıt fonksiyonu ile birlikte stokastik en kısa yol algoritmasını bulmuşlardır [Murthy ve ark., 1996].

Chang ve ark., ayrıt zaman değerlerinin değişken olduğu stokastik network üzerinde çalışmıştır. Stokastik zaman değişkenli ve seyahat süresi normal dağılımlı şebekede, muhtemel rotayı seçen çok kriterli bir algoritma geliştirilmiştir. Şebekeyi kontrol etmeyen yolları bulmak için bir algoritma önermişlerdir. Algoritmalarını küçük ve büyük şebekelerde deneyerek parametrik değişimlerin etkilerini incelemişlerdir [Chang ve ark., 2005]

Nielsen ve ark., stokastik değişkenli şebekelerde, iki kriterli muhtemel rota seçim problemi üzerinde çalışmışlardır. Beklenen süre ve maliyetleri minimize eden etkin yol setini bulmak için iki fazlı yaklaşım içeren bir algoritma önermişlerdir. Algoritmalarını beklenen, minimum ve maksimum değerler ile test etmişlerdir [Nielsen ve ark., 2006]

4.3. Stokastik Zaman Bağımlı Network Algoritmaları

Sonuç olarak, köşe ağırlıkları stokastik ve aynı zamanda zaman bağımlı olan algoritmalar vardır. Bu algoritmalar en iyi toplu taşımacılık şebekelerinde gösterilebilmektedir.

Ağırlık değişimlerinden dolayı, stokastik zaman bağımlı şebekeler Dijkstra benzeri algoritmalara indirgenemez. Algoritmalar optimallik prensibini izlemek durumundadır.

İlk olarak, stokastik zaman bağımlı şebekede, optimal yolu bulan bir çözüm önerilmiştir. Verilen şebeke, beklenen en erken varış süreleri için yüksek seviyeli tanımlama içermektedir [Hall, 1986].

Optimallik prensibine alternatif olarak stokastik tutarlılığı önerilmiştir. Stokastik tutarlılık, verilen zamanda varış ihtimali anlamına gelmektedir. Wellman, Hall'ın algoritmasını takip etmiş ve stokastik tutarlılık prensibine uyan şebekedeki yol sayısını azaltan bir optimizasyon üzerinde çalışmıştır. Uygulama, özellikle Hall' ın algoritmasının çalışma süresini azaltmaktadır [Wellman ve ark.,1990].

Hall'ın algoritmasına, son yolun seyahat sürelerinin üst ve alt sınırları bulmak için sezgisel önerilmiştir. Bu yüzden birçok yol dikkate alınmaktadır. Zaman bağımlı stokastik grafik için, en kısa yol algoritmalarının hepsi varış süresini minimize etmeye çalışmamaktadır [Kaufman ve ark., 1993].

Muhtemel en az zaman yolu bulmak için bir algoritma geliştirilmiştir. Bu algoritma muhtemel en az zaman yolunu ve seyahat süresinin gerçekleşme ihtimalini vermiştir [Miller ve ark., 1994].

Sürekli en kısa yola stokastik tutarlılık ve stokastik üstünlük, sürekli fonksiyonu alt ve üst sınırlar ile sıkıştırarak ulaşmaya çalışmıştır [Wellman ve ark., 1995].

Gao ve ark., stokastik zaman bağımlı şebekelerde, optimal rotalama problemleri için teorik bir yapı kurmuşlardır. Optimal rotalama problemlerini türleri, stokastik bağımlılık ve gerçek zamanlı verinin var olmasına göre sınıflandırmışlardır. Problemin bazı değişkenleri üzerinde çalışmışlardır [Gao ve ark., 2006]

5. ZAMAN BAĞIMLI EN KISA YOL PROBLEMİ UYGULAMASI

Bu bölümde, zaman bağımlı en kısa yol problemi için yapılan uygulama hakkında bilgi verilmektedir. Uygulamada kullanılan algoritma ve algoritmaya yapılan ek kısımlar anlatılmaktadır.

5.1. Uygulama Adımları

Ankara ili şehir içi ulaşımda zaman bağımlı en kısa yol probleminin çözümü için aşağıdaki adımlar takip edilmiştir.

- En kısa yol problemi için verilerin toplanması
- Veri yapılarının tasarımı
- Uygulamanın kodlanması için yazılım dili ve diğer kaynakların belirlenmesi
- En kısa yol için kullanılacak algoritmanın seçilmesi ve uygulanması
- Belirlenen rotanın alternatifleri ile harita üzerinde gösterilmesi
- Uygulamanın diğer uygulamalar ile karşılaştırılması

5.2. Verilerin Toplanması

Problemin çözümü için gerekli olan veriler, EGO Genel Müdürlüğünde bulunan istatistik biriminden alınmıştır. EGO Genel Müdürlüğü'nden temin edilen veriler ve verilerin uygulamada kullanılma biçimi bu bölümde anlatılmaktadır.

Bölge

Ankara şehir içi toplu taşımacılığında kullanılan alan, beş bölgeye ayrılmıştır. Bu bölgeler şunlardır:

1. Bölge Şube Müdürlüğü (Çankaya)
2. Bölge Şube Müdürlüğü (Yenimahalle)

3. Bölge Şube Müdürlüğü (Mamak)
4. Bölge Şube Müdürlüğü (Altındağ)
5. Bölge Şube Müdürlüğü (Sincan)

Hat

Her bölge otobüs hatlarından oluşmaktadır. Otobüs hatları sistemde tanımlanırken kullanılan bilgiler dört kısımdan oluşmaktadır. Bu kısımlar şunlardır:

1. Hat Numarası
2. Hat Adı
3. Uzunluk
4. Süre

Durak

Otobüs hatları duraklardan oluşmaktadır. Durakları oluşturan bilgiler iki kısımdan oluşmaktadır. Bu kısımlar şunlardır:

1. Durak Numarası
2. Durak Adı

Hareket

Otobüslerde günlük olarak tutulan ve servis başlama ve bitiş sürelerini içeren hareket bilgileri altı kısımdan oluşmaktadır. Bu kısımlar şunlardır:

1. Hat Numarası
2. Hareket Tarihi
3. Servis Başlama Saati
4. Servis Bitiş Saati
5. Süre
6. Yolcu Sayısı

EGO Genel Müdürlüğünden alınan bölge, hat, hareket bilgileri ile uygulamada diğer kullanılan alanlar veri yapılarının ve veri tablolarının tasarlanması bölümünde anlatılmaktadır.

5.3. Veri Yapılarının ve Veri Tablolarının Tasarlanması

EGO Genel Müdürlüğünden alınan bilgilere ek olarak, sistemde kullanılacak diğer gerekli bilgiler ise şunlardır:

1. Ayrıt
2. Güzergah

Ayrıt

Ayrıt tablosu iki durak, aradaki mesafe ve süreleri içeren bir veribanı tablosudur. Ayrıt tablosunu oluşturan alanlar şunlardır:

1. Ayrıt Numarası
2. Başlangıç Durak Numarası
3. Bitiş Durak Numarası
4. Uzunluk
5. Süre

Ayrıtlar yönsüz olarak düşünülmüştür. Ayrıtların uzunlukları hesaplanırken Google MAPS API' den faydalanılmıştır. Uzaklıklar hesaplanırken ayrıca enlem ve boylam bilgleri belirli iki nokta arası mesafeleri hesaplamaya yardımcı olan matematiksel fonksiyondan yararlanılmıştır. Ayrıtların süreleri hesaplanırken üç tane işlem adımı takip edilmektedir.

Bütün hatlar için ortalama servis süresi hesaplanır. Herbir hat için hareket tablosuna bakılarak son bir aylık hareket bilgileri alınır. Hareket bilgileri arasında yer alan servis süresi alanının aritmetik ortalaması alınarak herbir hat için ortalama servis

süresi hesaplanmış olunur. Hatların uzunluk ve süre bilgisi alınır. Hat tablosundan, toplam hat uzunluğu ve toplam hat ve süre bilgisi alınır.

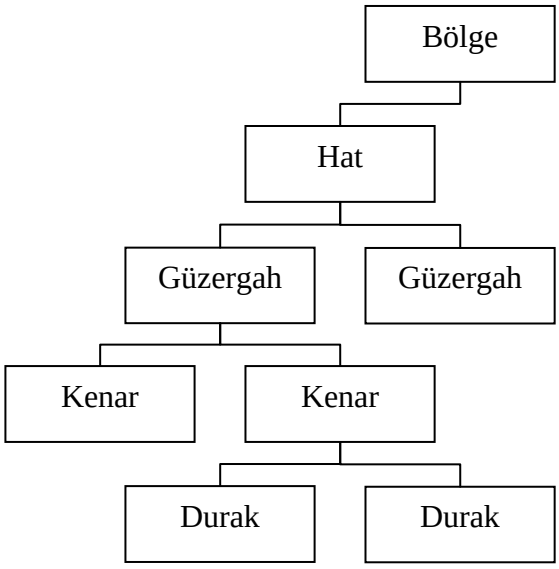
Herbir ayrıt için süre bilgisi hesaplanır. Hatın toplam servis süresi, toplam uzunluğa oranlanarak bir katsayı hesaplanır. Hesaplanan katsayı herbir ayrıtın uzunluğu ile çarpılarak her bir ayrıt için süre bilgisi hesaplanmıştır.

Güzergâh

Güzergâh, bir hattın hangi duraklardan geçtiğini gösteren bir tablodur. Ayrıtlar, güzergâh tablosunda belirli bir sıraya uygun olarak bulunmatadır. Güzergâh tablosunda yer alan bilgiler şunlarıdır:

1. Hat Numarası
2. Ayrıt Numarası
3. Sıra Numarası

Duraklar ayrıtları, ayrıtlar güzergâhları, güzergâhlar hatları, hatların ise bölgeleri oluşturduğu görülmektedir. Şebeke elemanları arasındaki hiyerarşi Şekil 5.1'de gösterilmektedir.



Şekil 5.1. Şehir içi ulaşım şebeke yapılanması

5.4. Uygulamada Kullanılan Kaynakların Belirlenmesi

Uygulama için bir takım kaynaklara ihtiyaç duyulmaktadır. Uygulamanın gerçekleştirilmesinde kullanılan kaynaklar şunlardır:

1. Yazılım Dili
2. Veri Tabanı
3. Diğer Kaynaklar

5.4.1. Yazılım dili

Uygulama yazılım dili olarak Java yazılım dili tercih edilmiştir. Uygulamada Java yazılım diline bağlı olarak Struts - EJB kullanılmıştır. Uygulamanın yazılım dilinde Java'nın tercih edilmesinin nedenleri şunlardır:

1. Platform bağımsız olması
2. Nesneye dayalı bir yazılım dili olması
3. Büyük uygulamalarda başarılı olması
4. Geniş bir kullanım alanına sahip olması

Java teknolojisi kullanılarak aynı uygulama farklı platformlarda kullanılabilmektedir. Uygulma, kod kısmında herhangi bir değişiklik yapılmadan farklı işletim sistemleri ve donanımlarda (PC, Macintosh, Cep Telefonlarında) kolaylıkla çalıştırılabilmektedir. Platform bağımsız olma özelliği ile diğer birçok yazılım dilinden daha avantajlı hale gelmektedir.

Java nesneye dayalı bir yazılım dilidir. Gerçek dünya nesnelerden oluşmaktadır. Çözülmek istenen problemi oluşturan nesneler, gerçek dünyadaki yapılarına benzer şekilde bilgisayar ortamında modellenebilmektedir. Birbirinden türetilen nesneler birbirlerinin özelliğini göstermektedirler. Sınıf ve nesne yapılarıyla daha kolay ve defalarca kullanılabilen modüller yaratılabilmektedir. Bu özellikler kullanım kolaylığını artırarak kodlama karmaşasını gidermektedir. Nesneye dayalı bir yazılım

dili olmasından dolayı diğer birçok fonksiyonel yazılım dillerinden daha avantajlı hale gelmektedir.

Modüler büyük çaplı ticari uygulamalar Java dili ile geliştirildiğinde başarılı sonuçlar ortaya koymaktadır. Java yazılım dili, uygulamanın büyüklüğüne karşılık verebilecek bir alt yapıya sahiptir. Büyük uygulamalar geliştirilirken, uygulama birden fazla programcı tarafından geliştirilmektedir. Java, modüler yapısı sayesinde çoklu kod geliştirilmesine de olanak sağlamaktadır.

Java yazılım dili, avantajlı özellikleri bulunmasından dolayı dünyada yaygın olarak kullanılmaktadır. Çok fazla geliştiricisi olduğundan ortaya çıkabilecek problemler rahatlıkla çözülebilmektedir. Birçok alanda kullanıldığından benzer alanlar içinde benzer uygulamalar geliştirilebilmektedir. Yaygın kullanım alanı olmasından dolayı diğer birçok yazılım dilinden daha avantajlı hale gelmektedir.

5.4.2. Veri tabanı

Gerekli olan verilerin tutulması için, seçilen yazılım dili ile uyumu, kullanım kolaylığı ve performansı dikkate alınarak MS SQL Server veri tabanı kullanılmıştır. MS SQL Server, ilişkisel bir veritabanıdır. Ayrıca çevrimiçi analitik işleme (OLAP) sunucusu, yerleşik replikasyon ve Veri Dönüştürme Servisleri'nde (DTS) veri ayıklama, dönüştürme ve yükleme (ETL) araçlarını içermektedir. Avantajlı özelliklerinden dolayı SQL Server kullanılmıştır.

5.4.3. Diğer kaynaklar

Uygulamada yazılım dili ve veritabanından farklı olarak Google MAP API kaynak olarak kullanılmıştır. Google Maps, Google tarafından hizmete sokulmuş ve ücretsiz online harita servisidir. Google Maps ayrıca diğer sitelerinde hizmetten yararlanması için API desteği de sunmaktadır. Google Maps API seçilmesinin nedenleri şunlardır:

1. Performans
2. Kullanılabilirlik
3. Yaygın kullanım alanı

Geliştirmeleri sürekli devam eden bir araç olmasından ve büyük bir yazılım şirketi olan Google'ın desteğinden dolayı performans yönünden başarılıdır.

Javascript altyapısını kullandığı için kullanımı kolay bir yapıdadır. Bütün web uygulamalarında rahatlıkla kullanılabilmektedir.

Google MAPS API' nin çok fazla sayıda kullanıcısı ve geliştiricisi bulunmaktadır. Bu nedenle problemlere hızlı ve doğru çözümler üretilebilmektedir. Yaygın kullanım alanı farklı uygulama örneklerini de ortaya çıkarmaktadır.

Avantajlı özelliklerinden dolayı uygulama geliştirme esnasında Google Maps API kaynak olarak kullanılmıştır.

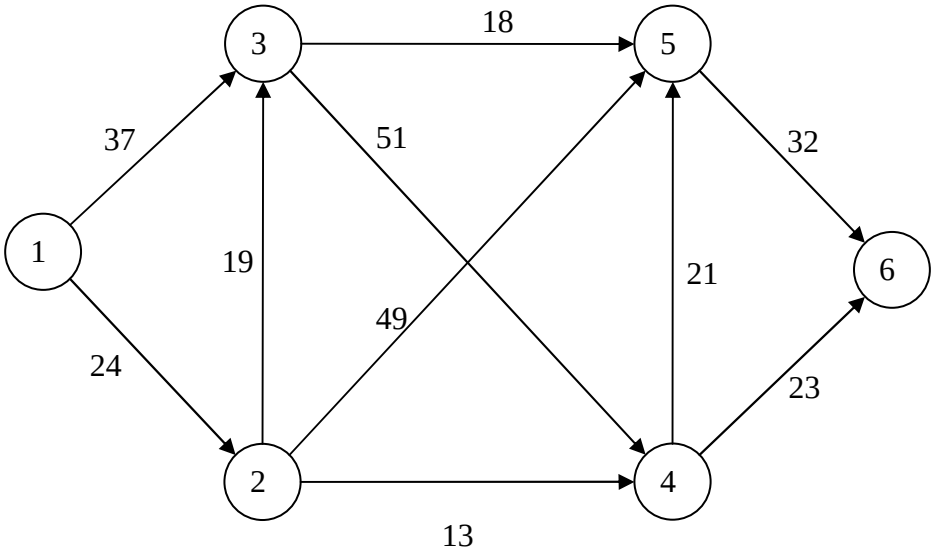
5.5. En Kısa Yol Problemi İçin Kullanılacak Algoritmanın Seçilmesi

Problemin çözümü için graf teorisi ve graf algoritmaları bölümünde belirtildiği üzere birçok yol mevcuttur. Algoritma seçilirken kolay kodlanabilir ve etkin olan bir algoritma seçilmesine dikkat edilmiştir. Seçilen algoritmaya ek olarak, en kısa yol bulunduktan sonra, alternatif en kısa yollar ve bunların zaman bağımlılıklarını dikkate alan kodlamalar yapılmıştır. Algoritma seçildikten sonra toplanan veriler ile uygulama gerçekleştirilmiştir.

Uygulamada, kolay ve etkin kodlama, nesneye dayalı programlama metodolojisine uyum avantajlarından dolayı Dijkstra Algoritması kullanılmıştır. Algoritmanın, uygulamada kullanımı için Dijkstra Sınıf'ı oluşturulmuştur. Dijkstra Sınıf'ı EK-1'de belirtilmiştir.

5.5.1. Seçilen algoritmanın örnek üzerinde gösterimi

Bu bölümde en kısa yol probleminin çözümü için seçilen Dijkstra algoritması ve geliştirilen ek kod işlemleri örnek şebeke üzerinde gösterilmiştir. Şekil 5.2'de verilen graf üzerinde 1 numaralı düğümü başlangıç düğümünü 6 numaralı düğüm ise bitiş düğümü olarak seçilmiştir.



Şekil 5.2. Örnek graf

Şekil 5.2'de verilen ağırlıklı grafın ağırlık değerleri sürelerden oluşmaktadır. Çizelge 5.1'de graf üzerinde bulunan düğüm noktalarının birbirlerine olan süre değerleri gösterilmektedir.

Sınıf içinde yer alan algoritmanın çalışması için gerekli adımlar şunlardır:

1. Süreler matrisinin oluşturulması
2. Algoritmanın çalıştırılarak en kısa süreli rotanın belirlenmesi
3. Grafın oluşturulması

Çizelge 5.1. Örnek süreler matrisi

	D1	**D2**	**D3**	**D4**	**D5**	**D6**
D1	0	24	37	0	0	0
D2	24	0	19	13	49	0
D3	37	19	0	51	18	0
D4	0	13	51	0	21	23
D5	0	49	18	21	0	32
D6	0	0	0	23	32	0

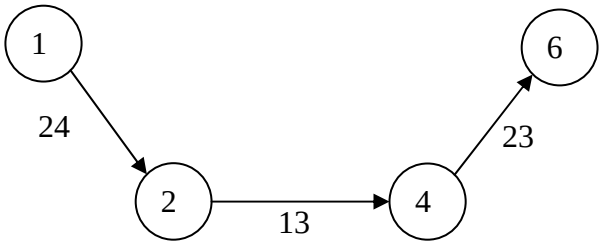
Çizelge 5.1’de verilmiş olan duraklar arası süre bilgileri veritabanından alınmaktadır. Duraklar arası süreler matrisi, kullanıcı tarafından seçilen başlangıç düğümünden başlamakta ve bitiş durağına kadar devam etmektedir.

Dijkstra algoritması, örnek süreler matrisini kullanılarak, başlangıç ve bitiş düğümü arasındaki en kısa yolu bulmaktadır. Dijkstra algoritması en kısa yolu bulurken, bütün düğümleri dikkate almaktadır. Uygulamada kullanılmak üzere oluşturulmuş olan Dijkstra Sınıf’ının oluşturduğu sonuçlar şunlardır:

1. Başlangıç ve bitiş düğümü arasındaki en kısa süre bilgisi
2. Başlangıç ve bitiş düğümleri arasıda takip edilen düğümlerin listesi

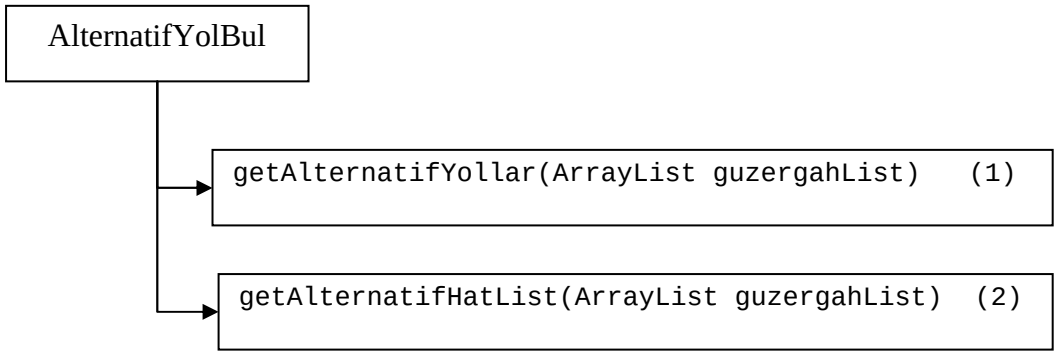
Dijkstra algoritmasının kodlamalarında genel olarak en kısa yolun süre bilgisi ile ilgilenilmektedir. Dijkstra algoritması kodlanırken, en kısa süre bilgisine ek olarak takip edilen düğümlerin listesini elde eden bölümler eklenmiştir.

Dijkstra Sınıf’ı en kısa yolu bulduktan sonra, bulunan en kısa yoldan geçen düğüm ve ayrıtları içeren bir graf oluşturmaktadır. Şekil 5.3’ de bulunan en kısa yol ve en kısa yol üzerinde bulunan düğüm ve ayrıtlardan oluşan graf gösterilmektedir.



Şekil 5.3. En kısa yol

En kısa yol grafının oluşturulmasından sonraki adım ise bulunan en kısa yol üzerinden geçen hatların bulunarak alternatif yollar listesinin elde edilmesidir. Alternatif yolların bulunması için AlternatifYolBul Sınıf'ı kullanılmaktadır. AlternatifYolBul Sınıf' ının metod ve bu metodların görevleri Şekil 5.4'de gösterilmektedir.

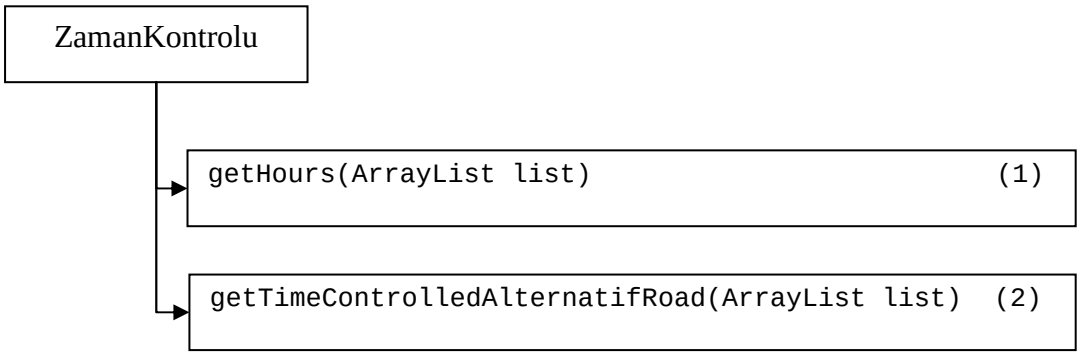


Şekil 5.4. AlternatifYolBul Sınıf'ı

- (1) Dijkstra Sınıf'ının bulduğu en kısa yol üzerinden geçen hat bilgilerinin listelenmesi
- (2) Google Maps a kaynak teşkil edecek şekilde alternatif yolların geçtiği düğümleri içeren listenin oluşturulması

Düğüm noktalarından geçen hatlar belirlendikten sonra, hatların kartezyen çarpımı ile bulunan en kısa yol bilgisi alternatif hat bilgilerine dönüştürülmektedir. Alternatif hat listesi, kullanıcıya, aynı en kısa yol üzerinden aktarmalı olarak tercih edebileceği hat bilgilerini içermektedir. Alternatif hat listesinin bulunmasından sonraki adım ise

bulunan alternatif yolların zaman kontrolünün yapılmasıdır. Zaman kontrolü yapılarak uygulama zaman bağımlı hale getirilmektedir. ZamanKontrolu Sınıf'ının metodları ve bu metodların görevleri Şekil 5.5'da gösterilmektedir.



Şekil 5.5. ZamanKontrolu Sınıf'ı

ZamanKontrolu Sınıf'ı, alternatif hat listesi içersindeki her bir hattın, en kısa yol üzerinde hareket ederken istenilen zamanda aktarma noktalarında olup olmayacağını kontrol etmektedir. Şekil 5.5'de gösterilen ZamanKontrolu Sınıfının görevleri şunlardır:

- (1) Sistem tarihi ve saati dikkate alınarak hangi saatlerde aktarma noktalarında olunması gerektiğinin belirlenmesi
- (2) Alternatiflerin, zaman bilgileri kontrol edilerek uygun alternatif olup olmadıklarının belirlenmesi

Alternatif hat listesi, alternatiflerin zaman bilgileri kontrol edildikten sonra, uygun alternatif hat listesi ve uygun olmayan alternatif hat listesi olarak iki bölüme ayrılmaktadır. Uygun alternatif hat listesi içerisinden belirlenen herhangi bir hat listesi, başlangıç ve bitiş düğümü arasında zaman bağımlı en kısa yol belirlemektedir.

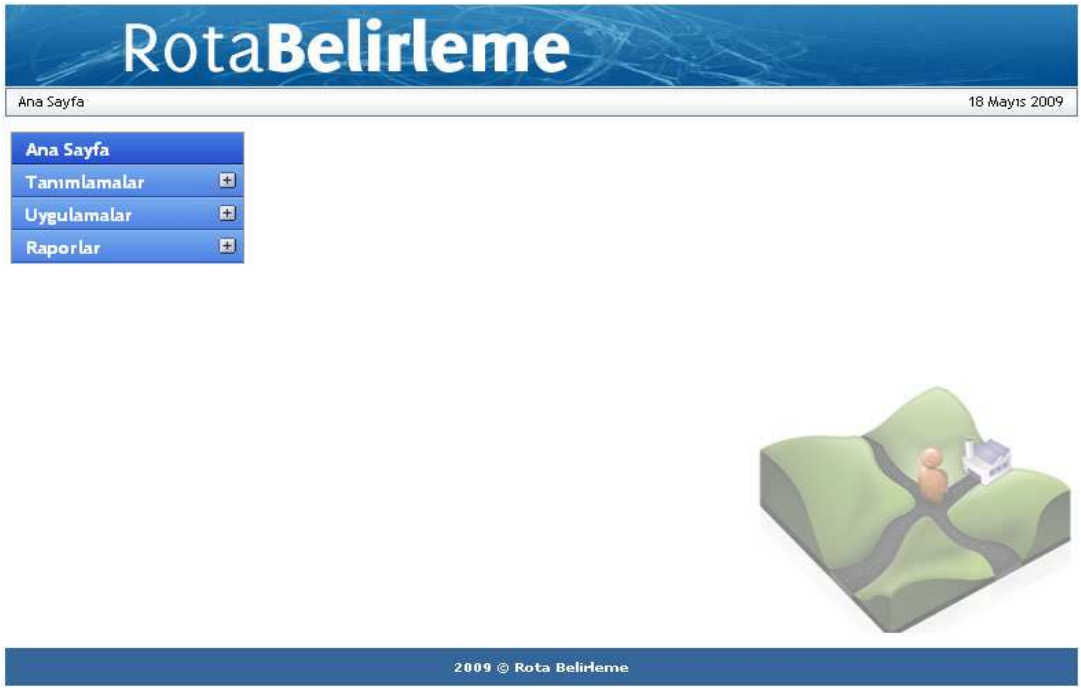
5.6. En Kısa Yol Problemi İçin Kullanılacak Algoritmanın Uygulaması

Bu bölümde en kısa yol probleminin çözümü için seçilen Dijkstra algoritması ve geliştirilen ek kod işlemleri uygulama üzerinden gösterilmektedir.

Uygulama üzerinden zaman bağımlı en kısa yol belirlenirken kullanılan modüller şunlardır:

1. Tanımlama
2. Uygulama
3. Raporlama

Resim 5.1.'de uygulamanın ana ekranı gösterilmektedir.



Resim 5.1. Zaman bağımlı en kısa yol uygulaması ana ekranı

5.6.1 Tanımlama

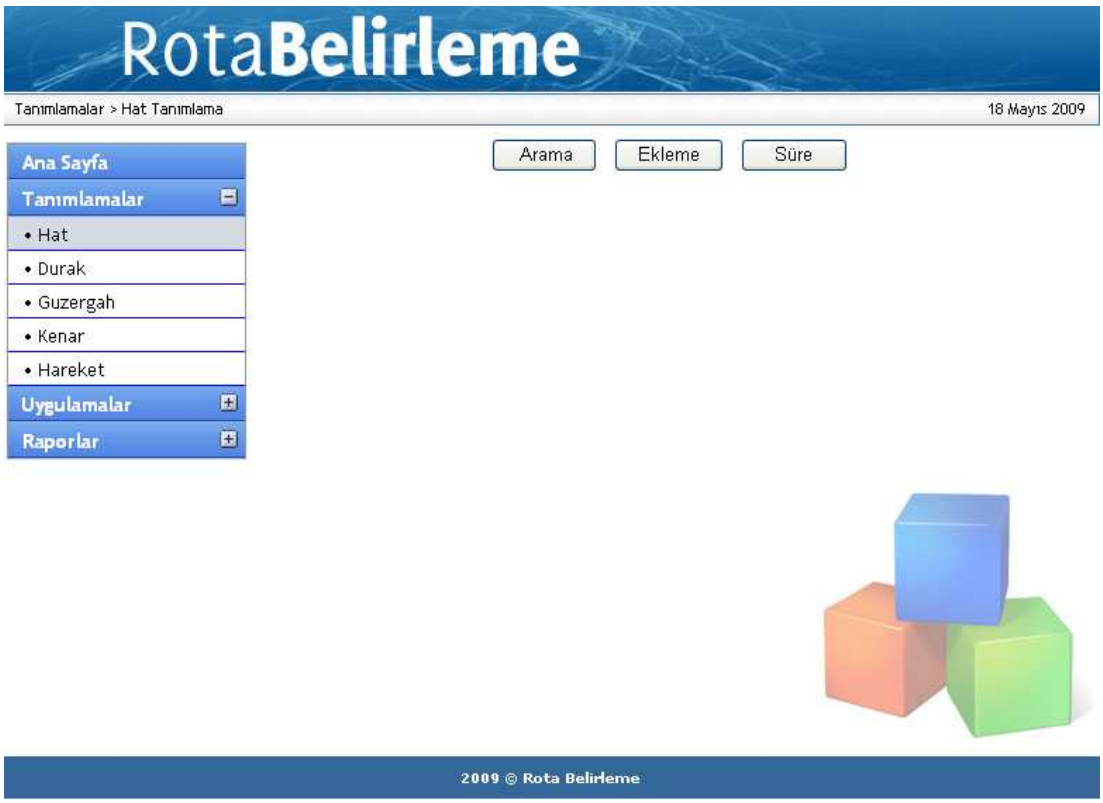
Tanımlama modülü, uygulama için gerekli tanımlamaların yapıldığı bölümdür. Tanımlama işleminin yapıldığı alanlar şunlardır:

1. Hat Tanımlama
2. Durak Tanımlama
3. Güzergâh Tanımlama

4. Ayrıt Tanımlama
5. Hareket Tanımlama

Hat

Tanımlama modülü içerisinde, hat tanımlama işlemlerinin yapıldığı kısmın ekran görüntüsü Resim 5.2'de verilmektedir.



Resim 5.2. Hat tanımlama

Hat tanımlama kısmında veritabanında kayıtlı olan hatlar, hat adı veya hat numarası alanları ile sorgulanabilmektedir. Hatlar için ayrıca yeni hat ekleme, hat silme ve hat güncelleme işlemleri de bu kısımdan yapılmaktadır.
Tanımlamalardan, hat aramanın yapıldığı kısım Resim 5.3'de gösterilmektedir.

Resim 5.3. Hat arama

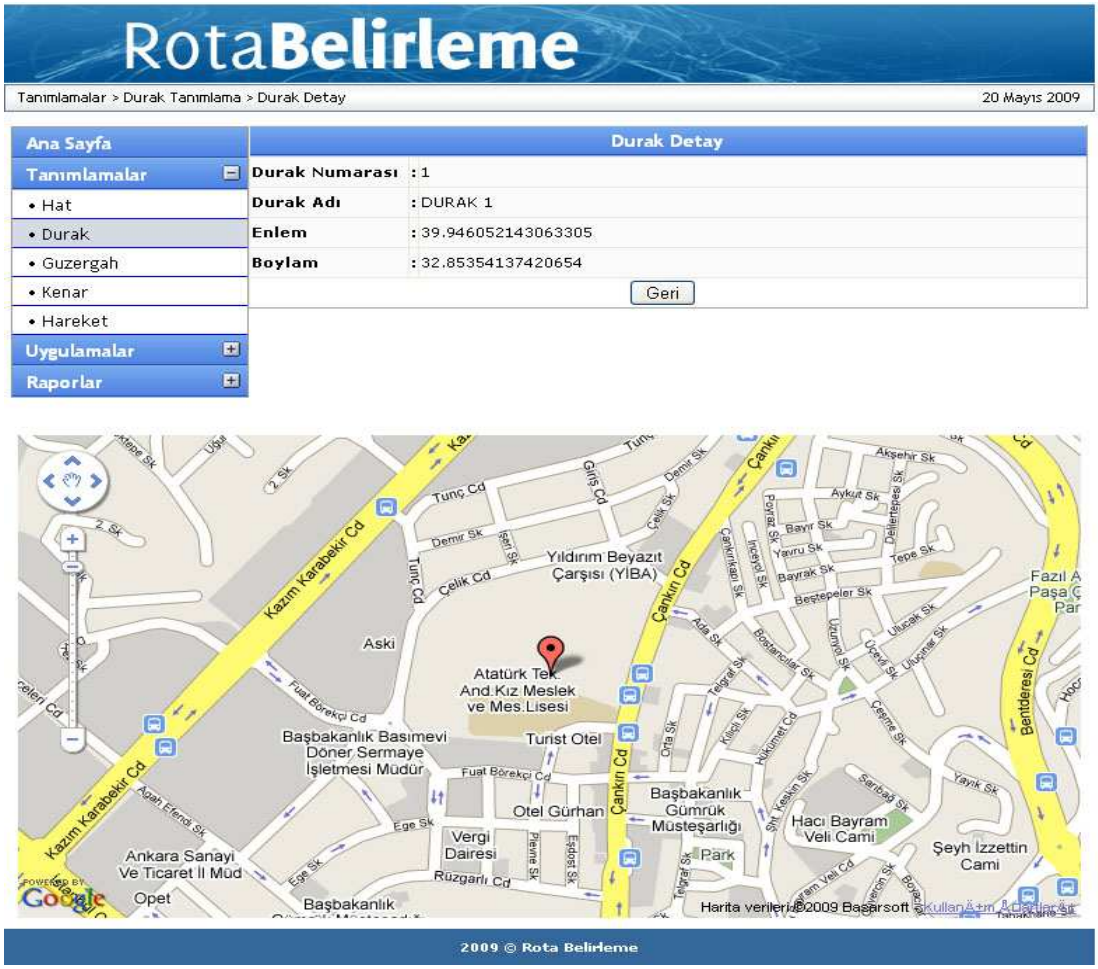
Durak

Tanımlama modülü içerisinde yeralan durak tanımlama kısmında, veritabanında kayıtlı olan duraklar, durak numarası, durak adı, enlem ve boylam arama kriterleri ile aranabilmektedir.

Resim 5.4. Durak arama

Duraklar için ayrıca yeni durak ekleme, durak silme ve durak güncelleme işlemleri de bu kısımdan yapılmaktadır. Tanımlama modülü içerisinde, durak arama

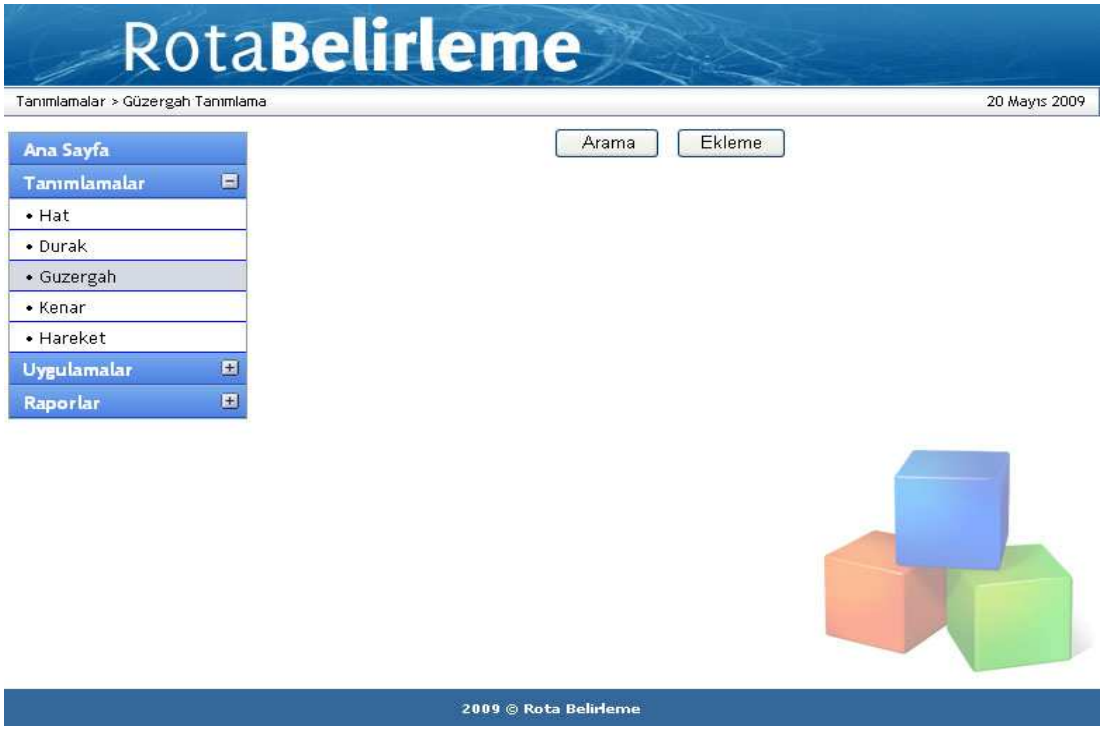
işlemlerinin yapıldığı kısım Resim 5.4'de verilmektedir. Durak arama işlemi yapıldıktan sonra detayı görüntülemek istenen durak satırına tıklanmaktadır. Satıra tıklanıldığında, durak ile ilgili detaylı bilgilere ulaşılabilmektedir. Resim 5.5'de durak detay kısmı görüntülenmektedir.



Resim 5.5. Durak detayı

Güzergâh

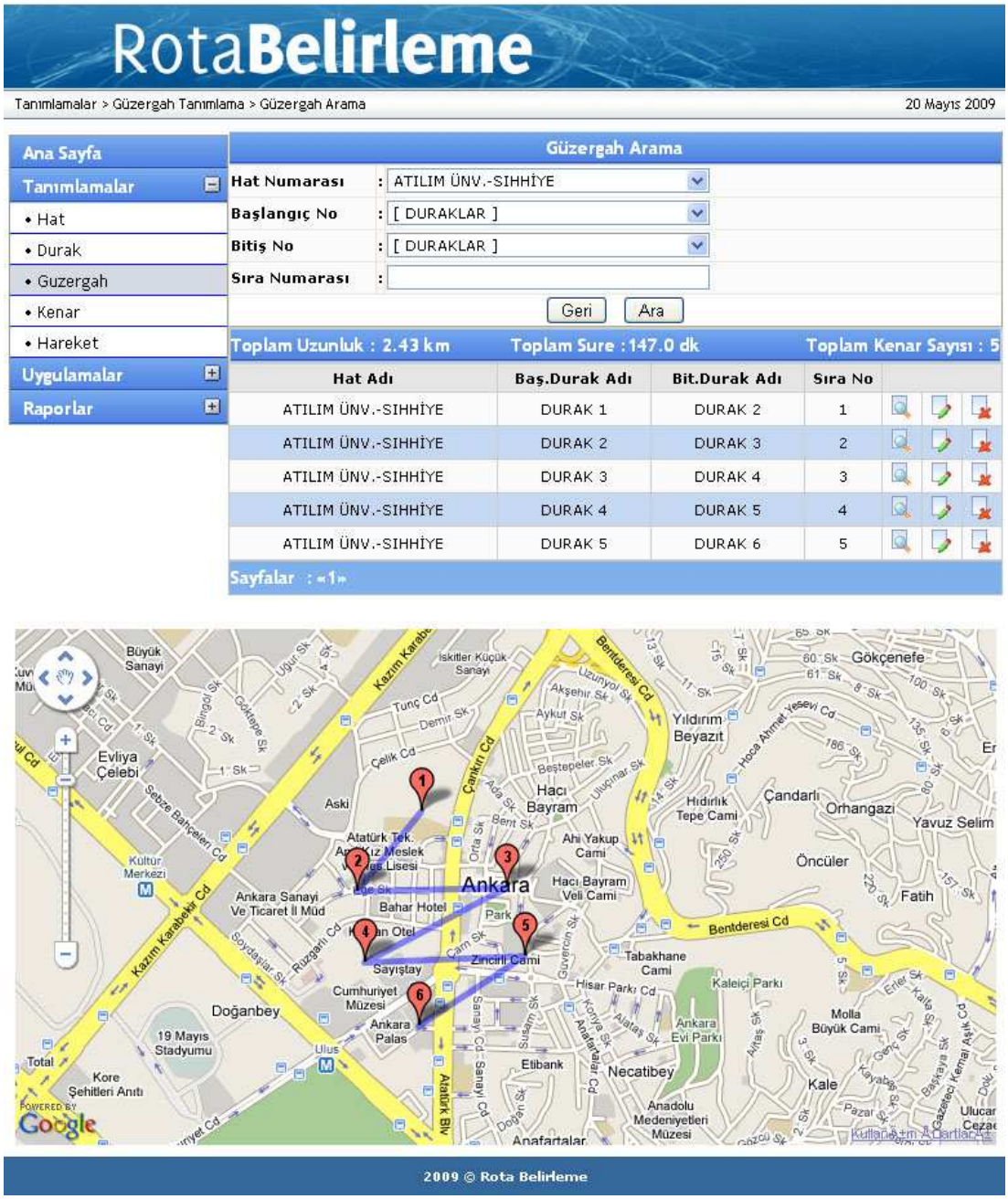
Tanımlama modülü içerisinde yeralan güzergâh tanımlama kısmı, güzergâh ekleme, silme ve güncelleme işlemlerinin yapıldığı kısımdır. Güzergâh tanımlama kısmında veritabanında kayıtlı güzergâhlar, hat numarası başlangıç durak numarası bitiş durak numarası ve sıra numarası arama kriterleri ile sorgulanabilmektedir. Güzergah tanımlama işlemlerinin yapıldığı kısım Resim 5.6'de gösterilmektedir.



Resim 5.6. Güzergah tanımlama

Güzergâh ekleme işleminde güzergâhı oluşturan hat, başlangıç durak numarası, bitiş durak numarası ve sıra bilgileri girilmektedir. Bir hattın güzergâh bilgisi oluşturulurken, ayrıt bilgisi, girilen başlangıç ve bitiş durak numarlarına göre otomatik olarak oluşturulmaktadır.

Otomatik olarak oluşturulan ayrıt, süre ve uzunluk bilgilerini içermektedir. Uzunluk bilgisi başlangıç durağının enlem ve boylam bilgisi ile bitiş durağının enlem ve boylam bilgileri dikkate alınarak matematiksel fonksiyon ile hesaplanmaktadır. Süre bilgisi ise ayrıt uzunluğunun hattın toplam uzunluğuna oranlanması ile oluşan katsayının, hattın ortalama servis süresi ile çarpılması sonucu hesaplanmaktadır. Güzergah arama işlemlerinin yapıldığı kısmın ekran görüntüsü Resim 5.7.'de gösterilmektedir.



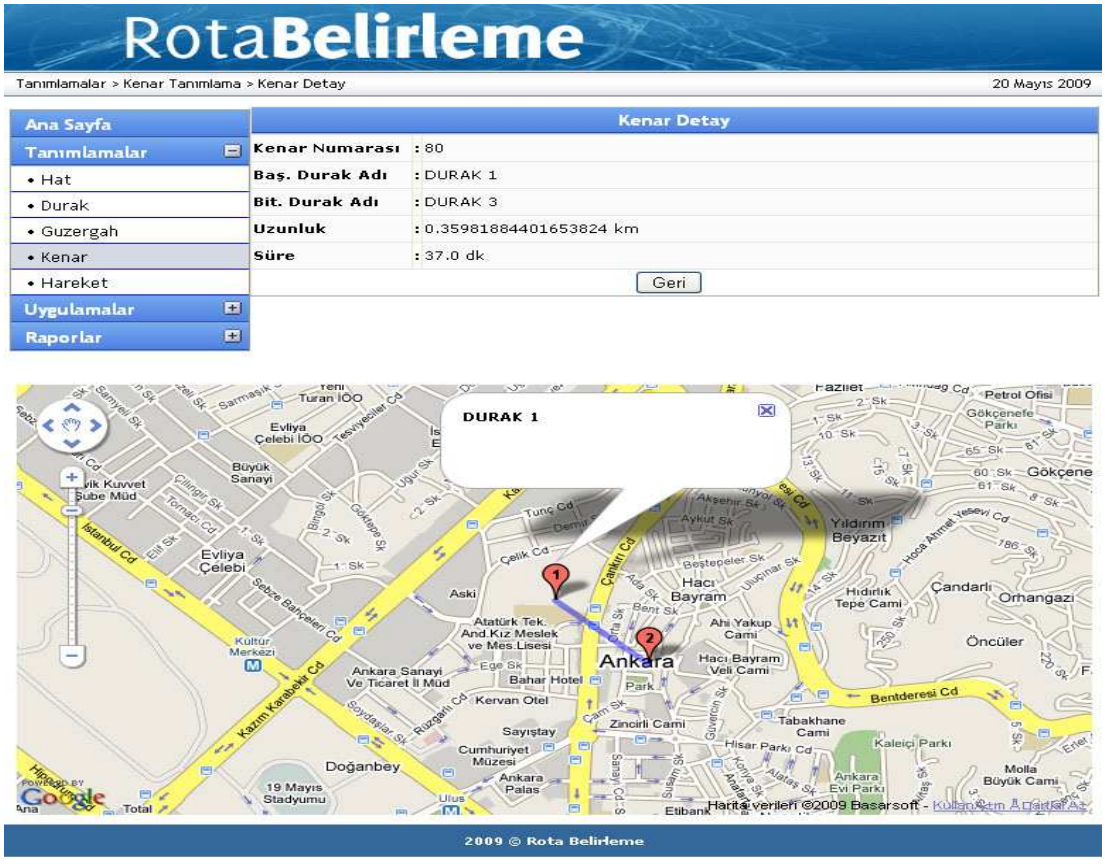
Resim 5.7. Güzergah arama

Ayrıt

Güzergah tanımlama sırasında oluşabilen ayrıtlar için ayrıca ayrıt tanımlama kısmı da bulunmaktadır. Tanımlama modülü içerisinde yeralan ayrıt tanımlama kısmı ayrıt ekleme, silme ve güncelleme işlemlerinin yapıldığı kısımdır. Ayrıt tanımlama kısmında veritabanında kayıtlı ayrıtlar, ayrıt numarası, başlangıç durak numarası,

bitiş durak numarası arama kriterleri ile sorgulanabilmektedir. Ayrıt arama işleminin yapıldığı kısım Resim 5.8'de ve ayrıt detayı Resim 5.9'da görüntülenmektedir.

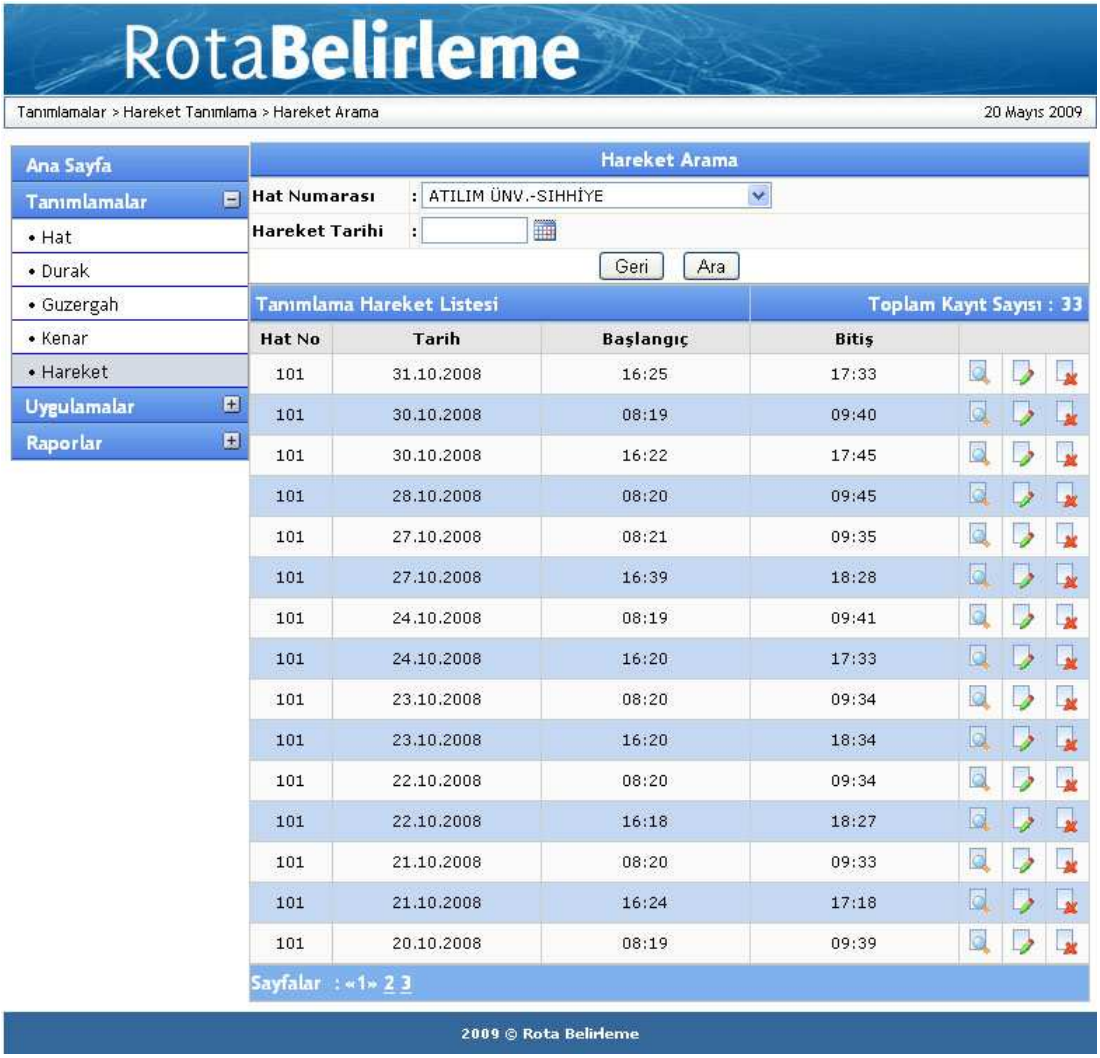
Resim 5.8. Ayrıt arama



Resim 5.9. Ayrıt detay

Hareket

Tanımlama modülü içerisinde yeralan hareket tanımlama kısmı hareket ekleme, silme ve güncelleme işlemlerinin yapıldığı kısımdır. Hareket tanımlama kısmında veritabanında kayıtlı hareketler, hat numarası ve hareket tarihi arama kriterleri ile sorgulanabilmektedir. Hareket arama işlemlerinin yapıldığı kısım Resim 5.10'da gösterilmektedir.



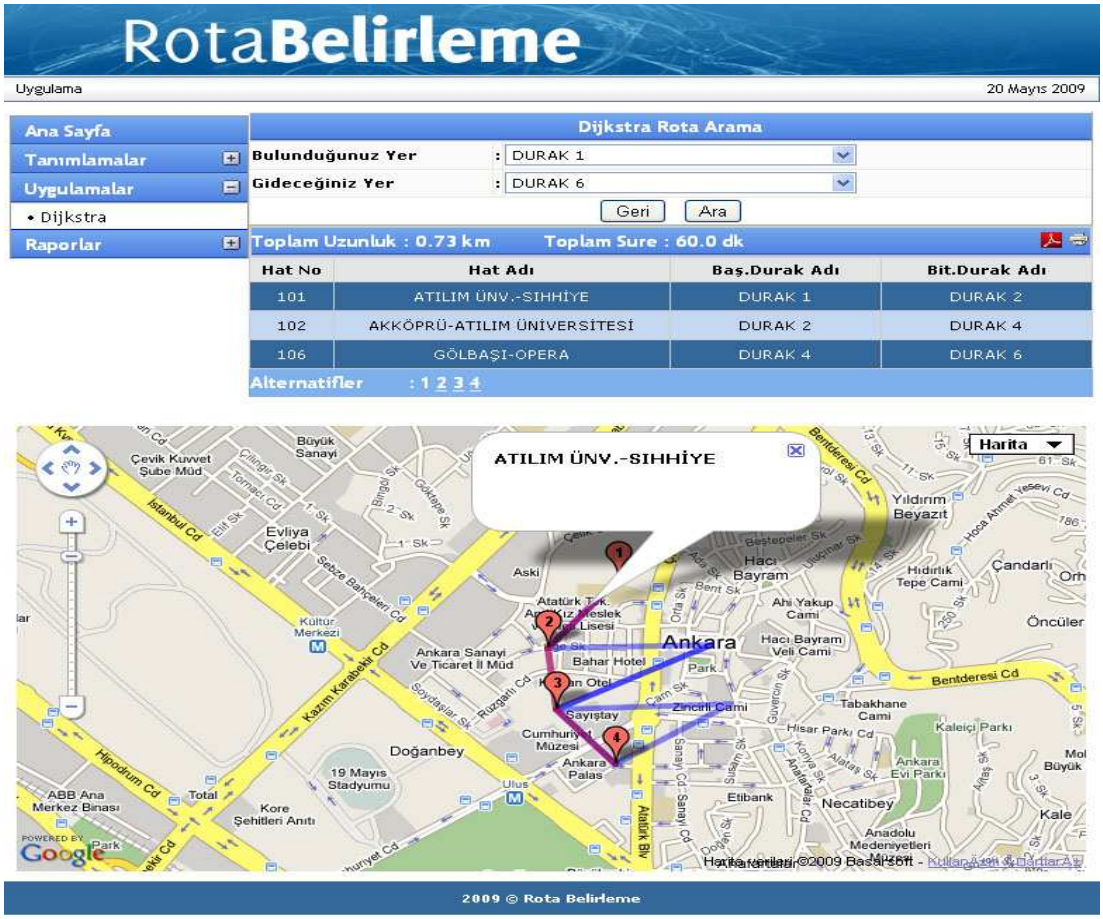
Resim 5.10. Hareket arama

Ortalama servis sürelerinin hesaplanmasında, hareket bilgilerinden yararlanılmaktadır.

5.6.2 Uygulama

Uygulama kısmında, kullanıcıdan, Resim 5.11'de belirtildiği gibi bulunduğu yer ve gideceği yer olmak üzere iki durak ismi seçmesi beklenmektedir. Sorgulama işlemi tamamlandıktan sonra kullanıcıya verilen bilgiler şunlardır:

1. Başlangıç ve bitiş durağı arasındaki en kısa yolun süre bilgisi
2. Başlangıç ve bitiş durağı arasındaki en kısa yolun uzunluk bilgisi
3. Başlangıç ve bitiş durağı arasıda takip edilen durakların listesi
4. Alternatif en kısa yol bilgileri
5. Alternatif en kısa yol haritaları



Resim 5.11. Uygulama

Ekranda bulunan alternatiflerin numaralarına tıklanılarak alternatif yol ve harita bilgileri arasında geçişler yapılabilmektedir. Haritalar arasında geçişler yapılırken alternatifte yer alan hatların hangi duraklardan geçtiği bilgisine de ulaşılabilmektedir. Alternatife yer alan hatların geçtiği yollar mavi renkte seçilen en kısa yolun geçtiği yollar ise kırmızı renkte gösterilmektedir. Ayrıca seçilen en kısa yol üzerindeki duraklar takip ediliş sırasına göre numaralandırılmıştır. Bu kısımdan seçilen yolun hangi durakları takip ettiği bilgisi de görülmektedir. Duraklar arası yolların üzerine tıklandığında, görüntülenen alternatif içerisinde hangi hat ile birlikte geçildiği bilgisine de ulaşılabilmektedir.

Bilgiler, listenin üzerinde bulunan bilgiler buton yardımıyla pdf tipinde dökümü alınarak saklanabilmektedir. Döküm kısmında, uygulamanın çalıştırılması sonucu elde edilen sonuçlar yeralmaktadır. Döküm kısmında yeralan bilgiler Resim 5.12'de gösterilmektedir.

DİJKSTRA
GÜZERGAH LİSTESİ

GÜZERGAH BİLGİLERİ

Buluduğunuz Yer	: DURAK 1	Toplam Uzunluk	: 0.73 km
Gideceğiniz Yer	: DURAK 6	Toplam Süre	: 60.0 dk
		Tarih / Saat	: 18.06.2009 / 10:02

ALTERNATİF GÜZERGAHLAR

Alternatif 1 :

Sıra	Hat No	Hat Adı	Başlangıç Durak Adı	Bitiş Durak Adı	Saat
1	101	ATILIM ÜNV.-SIHHİYE	DURAK 1	DURAK 2	10:02
2	102	AKKÖPRÜ-ATILIM ÜNİVERSİTESİ	DURAK 2	DURAK 4	10:26
3	106	GÖLBAŞI-OPERA	DURAK 4	DURAK 6	10:39

Alternatif 2 :

1	102	AKKÖPRÜ-ATILIM ÜNİVERSİTESİ	DURAK 1	DURAK 2	10:02
2	102	AKKÖPRÜ-ATILIM ÜNİVERSİTESİ	DURAK 2	DURAK 4	10:26
3	106	GÖLBAŞI-OPERA	DURAK 4	DURAK 6	10:39

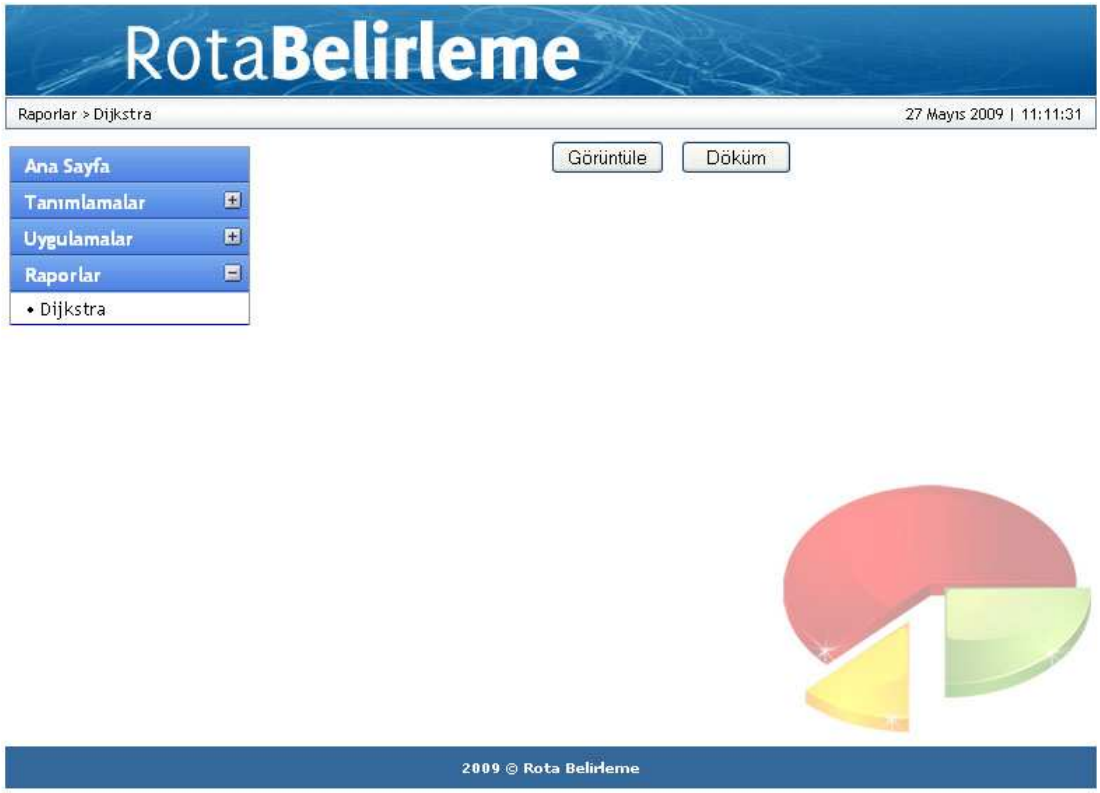
Resim 5.12 Uygulama sonuç dökümü

Ayrıca oluşturulan bilgiler yazıcıya gönderilerek çıktı alınabilmektedir.

Uygulamada elde edilen sonuçlar raporlama bölümünde raporlanarak uygulamanın etkin sonuç verip vermediği kontrol edilmektedir.

5.6.3 Raporlama

Uygulama gerçekleştirildikten sonra sonuçların inceleneceği bölüm raporlama bölümüdür. Oluşturulan rapor ekranda görüntülenebileceği gibi pdf dökümü de alınabilmektedir. Resim 5.13'de raporlama işlemlerinin yapıldığı kısım görüntülenmektedir.



Resim 5.13. Raporlama

Raporlama bölümünde, belirlenen alternatif yollardan geçen hatların:

1. Toplam uzunluk ve toplam süre bilgileri
2. Başlangıç durak ve bitiş durak bilgisi yer almaktadır.

Seçilen en kısa yol ile toplam uzunluk ve toplam süre bilgileri karşılaştırılmaktadır

Bu karşılaştırma sayesinde uygulamanın etkin ve verimli bir sonuç verdiği görülmektedir. Resim 5.14.'de alternatifleri liste halinde gösteren raporun ekran görüntüsü ve Resim 5.15'de raporun dökümü gösterilmektedir.

Resim 5.14. Alternatiflerin raporlanması

DİJKSTRA
RAPOR LİSTESİ

RAPOR BİLGİLERİ

Buluduğunuz Yer	: DURAK 1	Toplam Uzunluk	: 0.73 km
Gideceğiniz Yer	: DURAK 6	Toplam Süre	: 60.0 dk
		Tarih / Saat	: 18.06.2009 / 10:48

ALTERNATİF SÜRELER

Alternatif 1 :

Hat Adı	Başlangıç Durak Adı	Bitiş Durak Adı	Uzunluk	Süre
ATILIM ÜNV.-SIHHİYE	DURAK 1	DURAK 6	2.43	147.0
AKKÖPRÜ-ATILIM ÜNİVERSİTESİ	DURAK 1	DURAK 6	0.73	60.0
GÖLBAŞI-OPERA	DURAK 3	DURAK 6	0.81	74.0

Alternatif 2 :

AKKÖPRÜ-ATILIM ÜNİVERSİTESİ	DURAK 1	DURAK 6	0.73	60.0
GÖLBAŞI-OPERA	DURAK 3	DURAK 6	0.81	74.0

Resim 5.15. Alternatiflerin rapor dökümü

5.7. Diğer Uygulamalar ile karşılaştırılması

Zaman bağımlı en kısa yol problemi uygulamasına benzeyen çok sayıda yerli ve yabancı uygulama bulunmaktadır. Bu uygulamalardan öne çıkan ikisi yerli ikisi de yabancı olmak üzere toplam dört uygulama detaylı olarak incelenmektedir. İncelenecek olan uygulamalar şunlardır:

1. Google directions
2. Transport for London
3. Iett Yolculuk planlama sistemi
4. Ego güzergâh haritası

5.7.1. Google directions

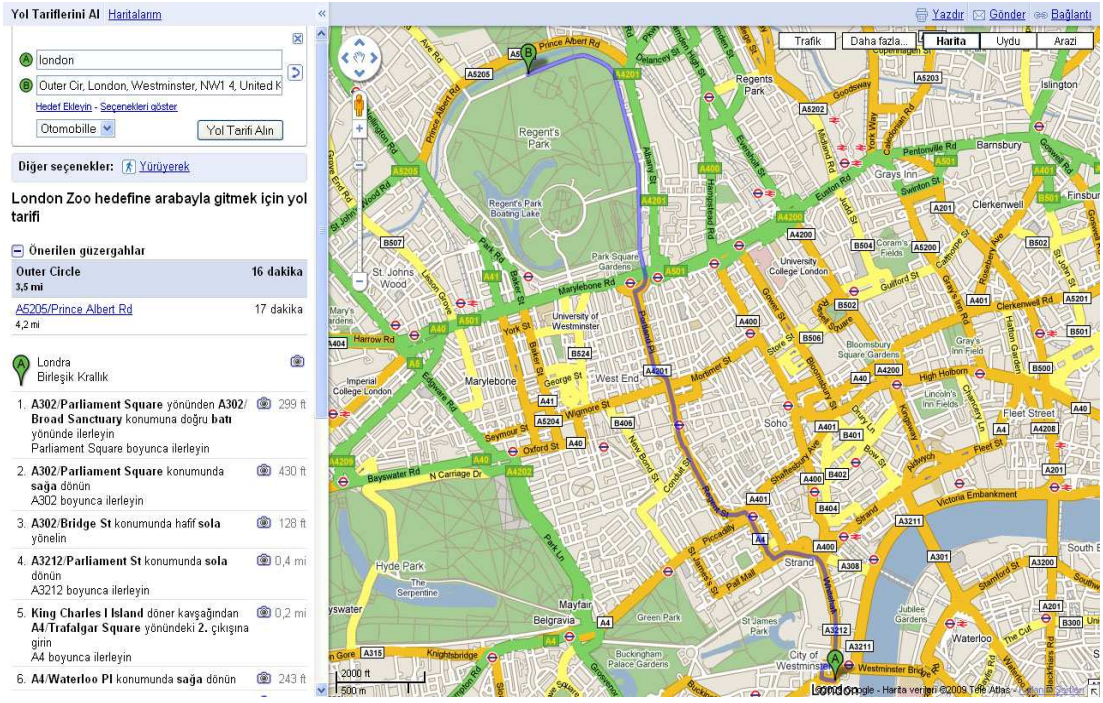
Google Directions, Google tarafından üretilmiş ve geliştirmeleri devam eden bir uygulamadır. Google MAPS API'yi kullanan uygulama henüz tüm dünya için geçerli bir uygulama değildir. Amerika, Kanada, İngiltere ve bazı Avrupa ülkeleri için sonuç veren uygulama, kullanım alanını her geçen gün daha da artırmaktadır. Uygulama başlangıç ve bitiş noktası arasındaki en kısa yolu bulmaya çalışmaktadır. Başlangıç ve bitiş noktaları yazılarak belirlenebildiği gibi navigasyon menüsünde bulunan ilgili kısım belirlenen iki nokta arasına bırakılarak da belirlenebilmektedir. Uygulamanın çalışabilmesi için istediği gerekli olan bilgiler şunlardır:

1. Başlangıç noktası
2. Bitiş noktası
3. Seyahat şekli (otomobille, toplu taşıma ile, yürüyerek)

Bilgiler girildikten sonra yol tarifi al butonuna tıklanarak başlangıç ve bitiş noktası arasında yol tarifi alınmaktadır. Uygulama, seyahat şeklini de dikkate alarak belirlenen iki nokta arasındaki en kısa yolu bulup, takip edilmesi gereken yolları yönleri ile beraber liste halinde sunmaktadır. Liste halinde sunulan bilgiler, harita üzerinden işaretlenerek, görsel hale getirilmektedir. Harita 5.1.'de Google Direction

uygulaması, Resim 5.16'de alınan yol tarifinin görsel hali verilmektedir.

Harita 5.1. Google Direction

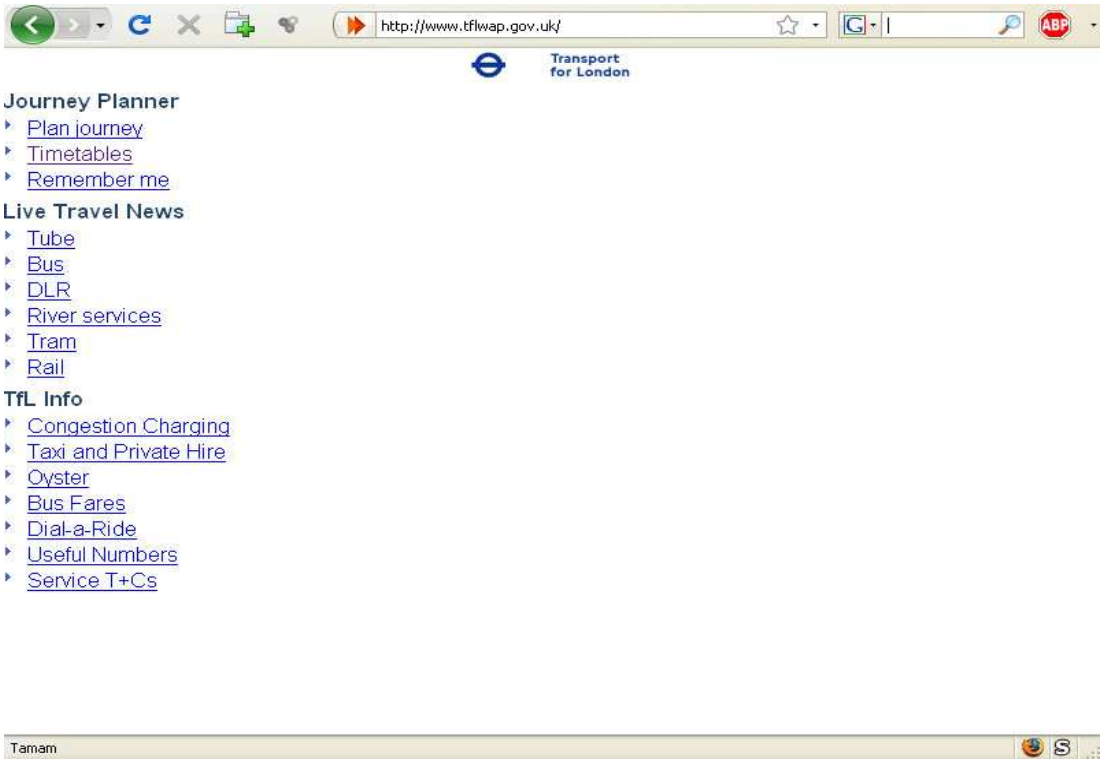


Resim 5.16. Google direction yol tarifi

Google Direction uygulaması, hız, kapsadığı alan, yazılı ve görsel öğeler açısından başarılı sonuçlar vermektedir. Alınan yol tarifinde önemli noktalar kamera görüntüleri ile desteklenmektedir. Google Direction, en kısa yolu bulurken düğümler arası uzunlukları hesaba katmaktadır. Ancak, zaman bağımlı en kısa yol uygulaması, Google Direction'dan farklı olarak zaman bilgisini dikkate almaktadır. Zaman bağımlı en kısa yol uygulaması çözümü alternatifler ile desteklemektedir. Google Direction, belirtilen özellikler ile zaman bağımlı en kısa yol uygulamasından farklılıklar göstermektedir.

5.7.2. Transport for London

İngiltere'de uygulanmakta olan Transport for London uygulaması, toplu taşımacılıkda kullanılan, internet ve wap tabanlı bir uygulamadır. Transport for London uygulaması, birçok alternatif içerisinden en uygun yolu bulmaya yardımcı olmaktadır. Resim 5.17' da Transport for London uygulaması gösterilmektedir.



Resim 5.17. Transport for London

Transport for London, metro, otobüs ve banliyö hatlarını kapsamaktadır. Uygulama Londra şehir içi taşımacılığını tahmin edilebilir kılmaktadır. Seyahatler daha kolay ve konforlu ve daha az stersli hale gelmiştir.

Uygulama, araçların bir sonraki istasyona ne zaman ulaşılacağı bilgisini vermemektedir. Sadece metro, otobüs ve banliyö hatları internet ve wap üzerinden takip edilebilmekte ve araçların tam olarak nerde oldukları bilgisine ulaşılabilmektedir.

Wap servisinde, metronun hat hat hareket süreleri bulunmaktadır. Buna benzer olarak banliyöde istasyonlardan hareket sürelerine ulaşılabilmektedir. Otobüsler için ise aktarma ve değişim zamanları gibi daha fazla bilgi yer almaktadır.

Uygulama bir takım bilgileri sunarak, daha akılcı kararlar verilmesini amaçlamaktadır. En kısa yol kullanıcı tarafından seçilecektir. Bu yönüyle uygulama zaman bağımlı en kısa yol uygulaması ile farklılık göstermektedir. Zaman bağımlı en kısa yol uygulaması kullanıcılara bir rota önerisi sunmaktadır.

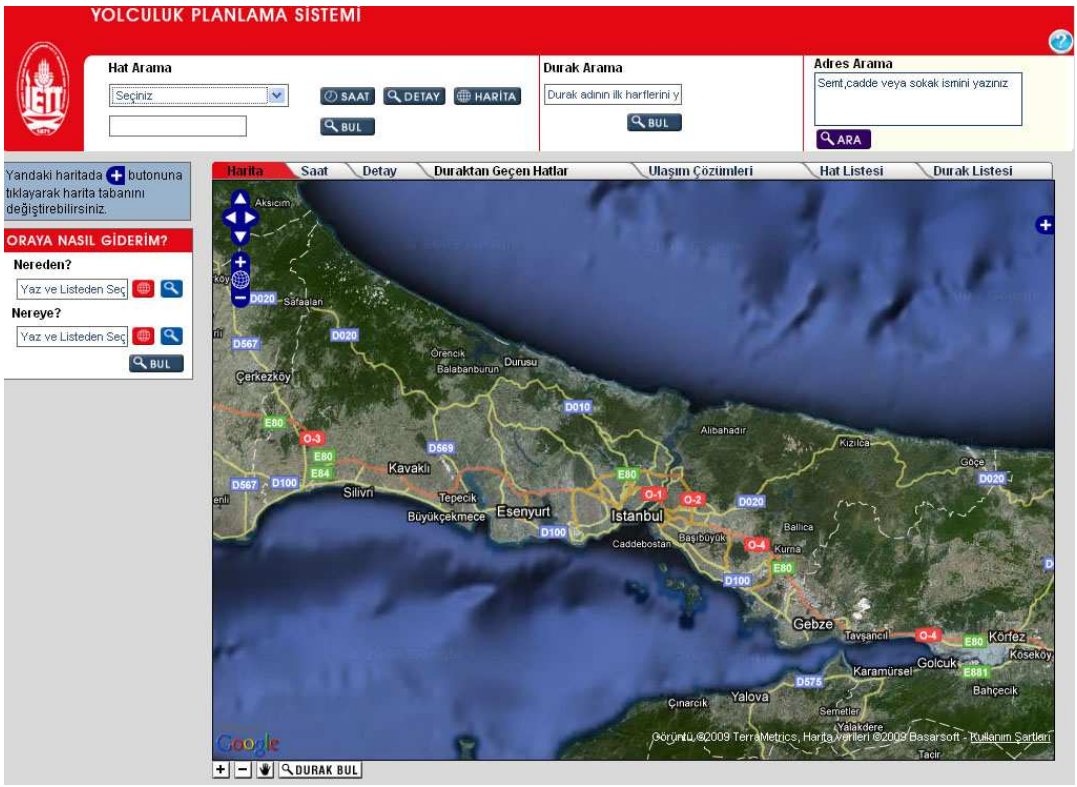
5.7.3. Yolculuk planlama sistemi

Yolculuk planlama sistemi, İstanbul Elektrik Tramvay ve Tünel İşletmeleri tarafından geliştirilmiş ve İstanbul için geçerli olan bir uygulamadır. Rota belirlenirken; en kısa yol algoritması ya da en hızlı yol algoritmaları kullanılmaktadır. Sistem, kullanıcının bineceği ve ineceği hat duraklarının alınmasıyla başlayarak veritabanında eşleştirme yapılma suretiyle çalışmaktadır. Kullanıcıya sunulan çözümler arasında sıralama yapılırken en az durak mesafesi, en kısa süre, en az maliyet sırasına göre sunulmaktadır. En kısa yol algoritmaları harita verileri üzerinde çalışmaktadır. Uygulamadan elde edilebilecek bilgiler şunlardır:

1. Hatların hareket saatleri, geçtiği duraklar ve harita bilgisi
2. Durakların detaylı bilgileri ve hangi hatların bu duraktan geçtiği bilgisi
3. Başlangıç ve bitiş durakları arası ulaşım bilgisi

4. Adres arama işlemi

Yolculuk Planlama Sistemi uygulaması, harita sistemi olarak Google Maps API'yi kullanmaktadır. Uygulamada harita normal, hibrit, fiziksel ve uydu olarak görüntülenebilmektedir. Uygulamanın modülleri ve harita görüntüleri Resim 5.18'te gösterilmektedir.



Resim 5.18. Yolculuk planlama sistemi

<u>Hat</u>

Hatların hareket saatleri, geçtiği duraklar ve harita bilgilerinin öğrenildiği kısımdır. Bilgi alınmak istenilen hat açılır kutudan seçilmektedir. Hat seçildikten sonra, seçilen hatta ait kalkış saatlerini öğrenmek için saat butonu, hangi duraklardan geçtiğini öğrenmek için detay butonu, hattın haritasını ve duraklarını harita üzerinde görmek için harita butonu kullanılmaktadır. Resim 5.19'de bir hattın saat bilgisi, Resim 5.20'de geçtiği durak listesi ve Resim 5.21'de hattın haritası gösterilmektedir.

Resim 5.19. Yolculuk planlama sistemi hat saat bilgileri

Resim 5.20. Yolculuk planlama sistemi hatta ait durak bilgileri

Resim 5.21. Yolculuk planlama sistemi hatta ait harita gösterimi

Durak

Duraklar, durak arama kısmında bulunan metin kutusuna durağın ismi yazılarak sorgulanabilmektedir. Arama işlemi sonunda gelen ekranda, durağın yanındaki butonları kullanılarak o duraktan geçen tüm hatlar listelenebilmekte ve durağın harita üstündeki yeri görüntülenebilmektedir. Resim 5.22, Resim 5.23, Resim 5.24’ de bu işlem örneklenmiştir.

Resim 5.22. Yolculuk planlama sistemi durak bilgisi

Resim 5.23. Yolculuk planlama sistemi duraktan geçen hatların listesi

Resim 5.24. Yolculuk planlama sistemi durak harita bilgisi

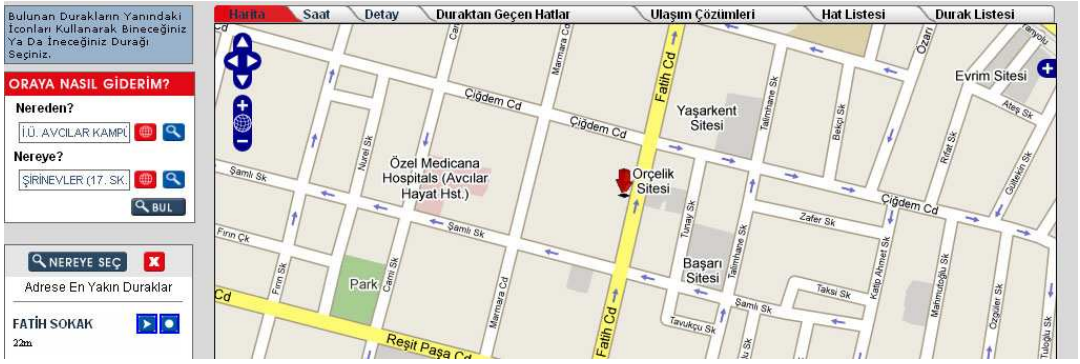
Ulaşım Çözümleri

Ulaşım çözümleri bölümünde, başlangıç ve bitiş durakları arasındaki yollar ulaşım çözümü olarak verilmektedir. Nereden metin kutusuna başlangıç noktası, nereye metin kutusuna ise bitiş noktası girilerek ulaşım çözümleri elde edilmektedir. Ulaşım çözümleri Resim 5.25'de gösterilmektedir.

Resim 5.25. Yolculuk planlama sistemi ulaşım çözümleri

Adres arama

Adres arama işlemi, bulunmak istenilen adresin adres metin kutusuna yazılması ile gerçekleşmektedir. Aranan adres harita üzerinde gösterilmekte ve adrese en yakın duraklar listelenmektedir. Aranılacak olan adres sokak, cadde, mahalle, semt, ilçe şeklinde yazılarak daha doğru sonuçlara ulaşılabilmektedir. Aranılan adres bulunduktan sonra harita o noktaya odaklanır ve 500 m çap içerisinde bulunan tüm duraklar formun alt kısmında görülmektedir. Bulunan durakların yanındaki butonlar kullanılarak nereden ya da nereye kutucuklarına atama yapılarak ulaşım çözümleri bulunabilmektedir. Resim 5.26'de adres arama işlemi gösterilmektedir.



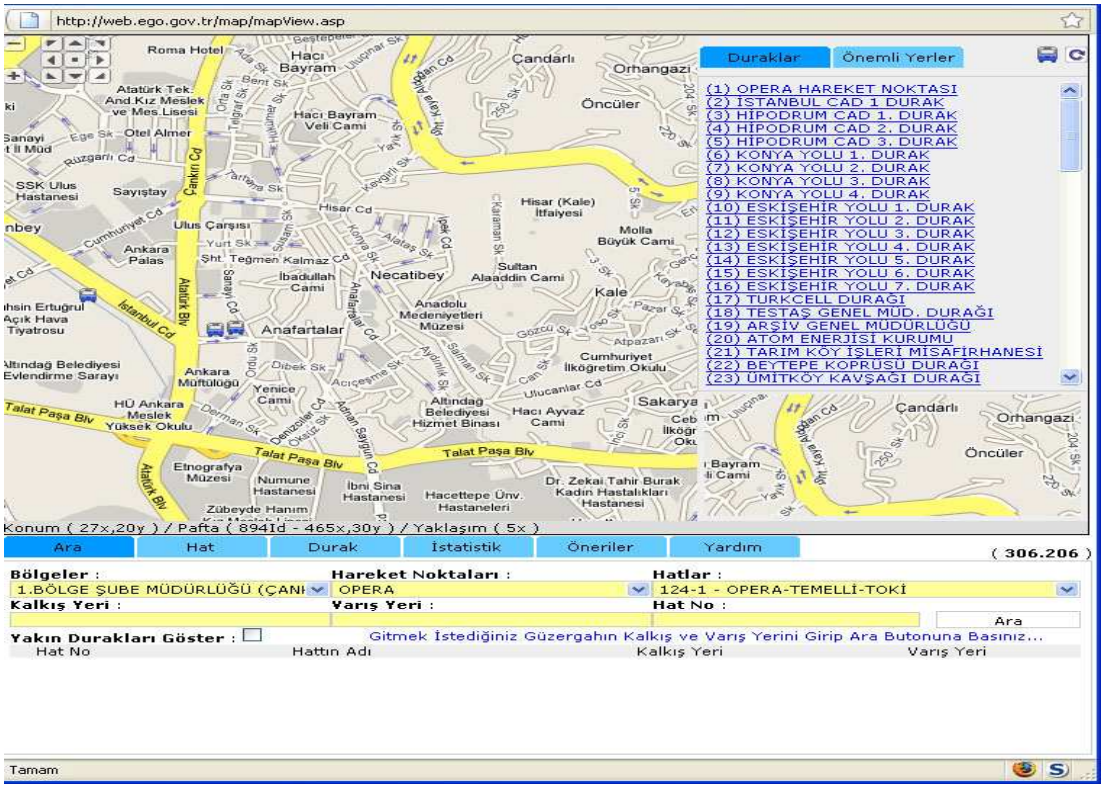
Resim 5.26. Yolculuk planlama sistemi adres arama

Yolculuk planlama sistemi, zaman bağımlı en kısa yol uygulaması ile benzerlikler içermektedir. Hat, durak ve ulaşım çözümleri zaman bağımlı en kısa yol

uygulamasına benzeyen kısımlardır. Yolculuk planlama sisteminde farklı olarak, adres bulma ve bulunan adres etrafındaki durakları listeleme işlemi bulunmaktadır. Yolculuk planlama sistemi diğer birçok uygulama gibi uzunlukları dikkate almaktadır. Zaman bağımlı en kısa yol uygulaması gerçek zamanlı süre bilgileri ile çalışmakta ve yolculuk süresini en aza indirmeyi amaçladığından dolayı yolculuk planlama sistemi ile farklılık göstermektedir.

5.7.3. EGO güzergâh haritası

EGO Güzergâh Haritası, Ankara ili şehir içi taşımacılıkta kullanılan otobüslerin hat bilgileri, hareket saat bilgileri, kalkış ve varış yeri girildiğinde belirtilen iki nokta arasındaki önemli yerleri gösteren bir uygulamadır. Resim 5.27'de EGO güzergâh haritası gösterilmektedir.



Resim 5.27. EGO güzergâh haritası

Uygulamada, girilen kalkış ve varış noktalarını içeren hat bilgileri listelenmektedir. Listelenen hatlardan incelenmek istenen hat seçildiğinde uygulamadan alınabilen bilgiler şunlardır:

1. Hat bilgisi
2. Durak bilgisi ve önemli yerler bilgisi
3. Harita bilgisi

Resim 5.28. EGO güzergâh haritası durak, önemli yer bilgileri

Hat

Uygulamada, hat ile ilgili bölge, hareket noktası, kalkış yeri, varış yeri, mesafe, süre bilgileri yeralmaktadır. Ayrıca hattın hareket saatleri ve geçtiği güzergâhlar bilgisine de bu bölümden ulaşılabilmektedir. Resim 5.29'da EGO güzergâh haritası hareket saati ve güzergâh bilgileri gösterilmektedir.

Resim 5.29. EGO güzergâh haritası hareket saati ve güzergâh bilgileri

Durak ve önemli yerler

Hatlar duraklardan oluşmaktadır. Uygulamada tanımlı olan bütün duraklar mavi otobüs resmi ile gösterilmektedir. Duraklar, durak ve önemli yerler alanından seçildiğinde seçilen durak, harita üzerinde kırmızı otobüs resmi ile belirtilmektedir. Duraklardan farklı olarak bir hat seçildiğinde o hattın geçtiği önemli yerler (meydan, aktarma noktası vb.) bilgi olarak verilmektedir. Önemli yer bilgisi, güzergâh bulmada kullanılan ek bilgilerdendir. EGO güzergâh haritası durak bilgileri Resim 5.30'da gösterilmektedir.

Resim 5.30. EGO güzergâh haritası durak bilgileri

Harita

Uygulamanın görsel aracı olarak, kendi içerisinde yeralan statik bir harita bulunmaktadır. Uygulama, Google Maps Api gibi herhangi bir dış araç kullanmamaktadır. Herhangi bir hat seçildiğinde, seçilen hat bütün durakları ile harita üzerinde görüntülenmektedir.

EGO güzergâh haritası uygulaması hareket saati, güzergâh ve durak bilgilerini içermektedir. En kısa uzunluk ve en kısa zamanlı yolu tercih etmekte herhangi bir öneride bulunmamaktadır. Bu yönüyle Zaman Bağımlı En Kısa Yol uygulamasından farklılık göstermektedir. EGO güzergâh haritası uygulaması kendi içerisinde yer alan statik haritaları kullanmaktadır. Bundan dolayı haritalar güncel değildir. Google Maps Api'nin haritaları sürekli güncellenmektedir.

6. SONUÇ

Kentlerin yayılarak büyümesiyle yolculuk mesafeleri artmış ulaşım sistemi araçlı yolculuklara bağımlı hale gelmiş ve artan yolculuk taleplerini karşılamaya yönelik özel ve kamu kesimi tarafından işletilen ulaşım türleri bugünkü ulaşım karmaşasını doğurmuştur. Farklı ulaşım türleri ile çeşitlenen şehir içi ulaşım sisteminde bir yolculuk sırasında kullanılacak ulaşım türlerinin olumlu özelliklerinin birleştirilmesi, olumsuz özelliklerinin azaltılması ve bu ulaşım türlerinin bir bütünün parçaları olarak çalışabilmeleri için bütünleştirilmeleri (entegrasyonu) gerekmektedir.

Toplu ulaşım hatları, servisleri ve türleri arasında aktarma gereksinmesi yolcu, işletici, ya da her ikisi açısından bir ihtiyaç veya zorunluluk olarak ortaya çıkabilmektedir. Başlangıç ve bitiş noktaları arasında doğrudan yolculuk yapmaya olanak veren bir şebeke bulunmasına karşılık, aktarma yapıldığında yolcu bazı avantajlar sağlıyorsa, bu kez yolcuya sunulan aktarmalı yolculuk daha çekici olmakta, yolcu mecbur olmamasına rağmen aktarmalı yolculuğu yapmaktadır. Sonuçta yolcuların zaman kazancı ve sistemin verimliliği artacak, daha çekici ve etkin bir toplu ulaşım sistemi oluşacaktır.

Şehir içi ulaşımda zaman bağımlı en kısa yol uygulamasında, aktarmalı ve etkin bir ulaşımın gerçekleşebilmesi için ulaşım kaynaklarından kullanıcıya kadar uzanan bir bilgi akışı oluşturulmaya çalışılmıştır. Zaman bağımlı en kısa yol uygulamasının, benzer uygulamalar ile farklı yönleri bulunmaktadır. Google direction, yolculuk planlama sistemi ve güzergâh haritası uygulamalarında en kısa yol önerisi verilirken uzaklıklar dikkate alınmaktadır. Zaman bağımlı en kısa yol uygulaması, toplu taşıma araçlarını kullananlar için daha önemli olan süreyi en aza indirmeyi amaçlamaktadır. Transport for London ise araçların gerçek zamanlı olarak nerede oldukları bilgisini vermektedir. Zaman bağımlı en kısa yol uygulaması, transport for London gibi gerçek zamanlı veriler ile çalışmakta, ek olarak da en kısa zamanlı rota önerisinde bulunmaktadır.

Çalışmada, graf teorisinin temel kavramları ele alınmış, problemin graf teorisiyle modellenmesi ve algoritmik çözümü ile ilgili çalışmalar incelenmiştir. Literatürde yer alan, zaman bağımlı en kısa yol algoritmaları üzerinde yapılan çalışmalar ele alınmıştır, çalışmaların birbirleri ile farklı ve benzer yönleri karşılaştırılmıştır. İncelenen algoritmalar içerisinden, kolay kodlanabilir ve etkin sonuç verme özellikleri dikkate alınarak Dijkstra algoritması seçilmiştir. Dijkstra algoritmasının yanında alternatif yolların bulunması ve zaman kontrolünün yapılması amacıyla ek kodlamalar yapılmış ve uygulama olarak Ankara ili şehir içi taşımacığında zaman bağımlı en kısa yol problemi için java teknolojisi ile bir web uygulaması geliştirilmiştir. Uygulamada alternatif en kısa yolların da bulunması sağlanmış ve zaman bilgileri kontrol edilerek uygulama zaman bağımlı hale getirilmiştir.

Zaman bağımlı rota belirleme uygulaması, toplu taşımadan sorumlu kamu ve özel kurumlar ile kullanıcılar açısından bir takım faydalar sağlamaktadır. Kamu ve özel kurumlar için sağladığı faydalar şunlardır; toplu taşıma daha etkin kullanılmakta ve bu sayede kullanıcılar toplu taşımaya teşvik edilmektedir. Atıl kaynaklar kullanılarak kaynaklar verimli hale gelmektedir. Kaynakların yazılım desteği ile entegrasyonu sağlanmakta ve bu sayede kaynaklar kolay kontrol edilebilir hale gelmektedir. Kullanıcılar açısından sağladığı faydalar şunlardır; kullanıcılar daha akılcı kararlar verilebilmekte ve seyahat esnasında oluşabilecek zaman kayıpları ve maliyetler en aza inmektedir.

Uygulamanın gerçek bilgilere daha yakın sonuçlar verebilmesi için duraklar arası sürelerin daha güncel ve doğru bir biçimde ölçülmesi gerekmektedir. Duraklar arası sürelerin daha iyi hesaplanabilmesi için durak ve otobüsler arasında sinyalizasyon çalışması yapılması gerekmektedir. Uygulama, sinyalizasyondan gelen duraklar arası süre bilgileri web servisler aracılığıyla kullanılarak anlık en kısa süreli rota bilgisini verebilecek şekilde geliştirilebilir. Ancak sinyalizasyon çalışması maliyetler ve sistemin kontrolü açısından zor bir çalışmadır. Ayrıca uygulamanın çok daha karmaşık graf modellerinde daha etkili çalışabilmesi için “graf parçalama yöntemi” kullanılabilir. Böylece gerekli olan iş yükü işlemciler arası bölünerek çalışma zamanından tasarruf edilebilir.

KAYNAKLAR

Baker B.M., Carreto A.C., "Visual Interactive Approach To Vehicle Routing",***Computers & Operations Research***, 30 (3) : 321–337 (2003).

Bellman, R. E. 1958, "On a Routing Problem", ***Quarterly of Applied Mathematics***, 16 : 87–90 (1958).

Blasum U., Hochstättler W. "Application of the branch-and-cut method to the vehicle routing problem", ***Technical report, Universitat zu Koln***, 184 (2002).

Bodin L., Golden B., Assad A., Ball M., "Routing and Scheduling of Vehicles and Crews", ***The State of the Art, Computers and Operations Research***, 10 : 63–211 (1983).

Braysy O., Gendreau M., "Tabu Search Heuristics for the Vehicle Routing Problem with Time Windows SINTEF Applied Mathematics", ***Research Council of Norway*** 255-260 (2001).

Cai, X., Kloks T., Wong C.K. "Time-Varying Shortest Path Problems with Constraints", ***Networks***, 29 (3) : 141–149 (1997).

Chang T., Nozick N.K., Turnquist M.A.. "Multiobjective path finding in stochastic dynamic networks, with application to routing hazardous materials shipments", ***Transportation Science***, 39(3) : 383–399 (2005).

Chen, Y. L. Tang Kwei., "Minimum Time Paths in a Network with Mixed Constraints". ***Computers and Operations Research***, 25 (10) : 793–805 (1998).

Clarke G., Wright J.W., "Scheduling Of Vehicles From A Central Depot To A Number Of Delivery Points", ***Oper. Research,*** 12 : 568–581 (1964).

Cook, K. L., Halsey E., "The Shortest Route Through a Network With Time-Dependent Internodal Transit Times", ***Journal of Mathematical Analysis and Application.*** 493–498 (1966).

Çölkesen R., "Veri Yapıları ve Algoritmalar", ***Papatya yayıncılık***, İstanbul, 424 (2002).

Dantzig G.B., Ramser J.H., "The truck dispatching problem", ***Management Science***,6 : 80–91 (1959).

Dijkstra, E.W., "A Note on Two Problems in Connexion with Graphs", ***Journal of Numerical Mathematics***, 1 : 269–271 (1959).

Dreyfus, S.E., "An Appraisal of Some Shortest Path Algorithms", ***Operations Research***, 17 : 395–412 (1969).

Eiselt H.A., “Arc routing problems: part I: The chinese postman problem”, ***Operations Research***, 43 : 231–242 (1995).

Friesz, T.L., Bernstein D, Smith T.E., Tobin R.L., Wie B.W. “Variational Inequality Formulation of the Dynamic Network User Equilibrium Problem”, ***Operations Research***, 41(1) : 179–191 (1993).

Fu L., “An adaptive routing algorithm for in–vehicle route guidance systems with real–time information”. ***Transportation Research Part B***, 35 : 749–765 (2001).

Gao S., Chabini I., “Optimal routing policy problems in stochastic time-dependent networks”. ***Transportation Research Part B***, 40 : 93–122 (2006).

Gendreau M., Hertz A., Laporte G., “New insertion and postoptimization procedures for the travelling salesman problem”, ***Operations Research***, 40 : 1086–1094 (1992).

Grant F.H., Solberg J.J., “Variance reduction techniques in stochastic shortest route analysis: application procedures and results”, ***Mathematics and Computers in Simulation***, 25 (4) : 366–375 (1983)

Hall, R. W. , “The Fastest Path Through a Network With Random Time–Dependent Travel Times”, ***Transportation Scienc***, 20 : 182–188 (1986).

Halpern, J. L., “Shortest Route With Time-Dependent Length of Edges and Limited Delay Possibilities in Nodes”, ***Zeitschrift fur Operations Research***. 21 : 117–124 (1977).

Taha H. A., “Operations Research”, Çeviri Editörü/Editörleri, ***Ş.Alp Baray***, ***Şakir Esnaf***, İstanbul, 485 (2000)

Haquari M., Dejax P., “Time Dependent Shortest Path Problem: Solution and Application to Routing Problems”, ***Recherche Operationnelle***, 31(2) : 117–31 (1997).

Ichoua S., Gendreau M., Jean-Yves P., “Vehicle Dispatching With Time Dependent Travel Times”, ***European Journal of Operations Research***, 144 : 379–396 (2003).

Kaufman D.E., Smith R.L., “Fastest Paths in Time–Dependent Networks for IVHS Application”, ***IVHS Journal***, 1: 1–11 (1993).

Koutsopoulos, H.N., Xu H., “An information discounting routing strategy for ATIS”, ***Transportation Research Part C (Emerging Technologies)***, 1C(3) : 249–264 (1994).

Lau H.C., “Vehicle Routing Problem With Time Windows and A Limited Number Of Vehicles”, ***European Journal of Operational Research***, 148 (3) : 559–569 (2003).

Malandraki C., “The Time–Dependent Vehicle Routing Problem”, Doctoral Dissertation, ***Northwestern University***, 175 (1989).

Miller, E.D., “Optimal Routing in Time–Varying Stochastic Networks: Algorithms and Implementation”, Doctoral Dissertation. ***University of Texas, Austin***, 49 (1994).

Miller, E.D., Mahmassani, H.S., and Ziliaskopoulous, A., “A. Path search techniques for Transportation Networks with Time–Dependent, Stochastic Arc Costs”, ***IEEE International Conference on Systems, Man, and Cybernetics. Humans, Information and Technology***. 2:1716–1721 (1994).

Minieka E., “Optimization Algorithms for Networks and Graphs”, ***New York: Marcel Dekker Inc.***, 356 (1978).

Nielsen L.R., Andersen K.A., Pretolani D., “Bicriterion a priori route choice in stochastic time–dependent Networks”, ***Working Paper*** , 1– 23 (2006).

Orda, A., Rom R., “Routing with Packet Duplication and Elimination in Computer Networks”, ***IEEE Transactions on Communications***, *36* : 860–866 (1988).

Orda, A., Rom R. “Shortest–Path and Minimum–Delay Algorithms in Networks With Time–Dependent Edge–Lengths”, ***Journal of the Association for Computing Machinery. IVHS Journal***, 37 : 603–625 (1990).

Rochat Y., Taillard E.D., “Probabilistic diversification and intensification in local search vehicle routing”, ***Journal of Heuristics***, 1 : 147–167 (1995).

Slavenko C., “Application of The Chinese postman Problem Model To The Toronto Transportation Network Within GIS–Based Software”, Thesis of Master, ***University of Toronto***, Toronto, 103 (2001).

Sinha S.M., “Dynamic Programming”, Mathematical Programming Theory and Methods Pages, ***Elsevier India P Ltd*** , Delhi, 502–533 (2007).

Taillard E.D., “Parallel Iterative Search Methods for Vehicle Routing Problems”, ***Networks***, 23 : 661–673 (1993).

Waller S.T., Ziliaskopoulos A., “On the online shortest path problem with limited arc cost dependencies”, ***Networks***, 40(4):216–227 (2002).

Wu, Q. , “Time–Dependent Stochastic Shortest Path Algorithms for a Scheduled Transportation Network”, ***IJSSST*** , 6 (7) : 53–60 (2005).

Ziliaskopoulos A. , “Time–Dependent Shortest Path Algorithm for Real– Time Intelligent Vehicle Highway System Applications”, ***Transportation Research Record***, 1408 : 94–100 (1993).

Ziliaskopoulos, A. , “Note on least time path computation considering delays and prohibitions for intersection movements”, ***Transportation Research, Part B (Methodological).***30B (5) : 359–367 (1996).

Ziliaskopoulos, A., “Design and Implementation of Paralel Time–Dependent Least Time Path Algorithms for Intelligent Transportation Systems Applications”, ***Transportation Research Part C***, 5(2) : 95–107 (1997).

İnternet: Wikipedia “Bellman–Ford algoritmaları”
http://en.wikipedia.org/wiki/Bellman-Ford_algorithm#Algorithm (2001)

İnternet: U.S. National Institute of Standards and Technology “Bellman–Ford algoritmaları”
http://www.itl.nist.gov/div897/sqg/dads/HTML/bellmanford.html (2005)

İnternet: Rensselaer Polytechnic Institute “Bellman–Ford Screen”
http://www.cs.rpi.edu/~musser/gp/algorithm-concepts/bellman-ford-screen.pdf (2003)

İnternet: Computer Programming “One Source Shortest Path: The Bellman-Ford Algorithm”
http://compprog.wordpress.com/2007/11/29/one-source-shortest-path-the-bellman-ford-algorithm/ (2007)

İnternet: Computer Programming “One Source Shortest Path: The Bellman-Ford Algorithm”
http://compprog.wordpress.com/2007/11/15/all-sources-shortest-path-the-floyd-warshall-algorithm/ (2007)

İnternet: Tübitak Marmara Araştırma Merkezi Enerji Sistemleri ve Çevre Araştırma Enstitüsü “Java Programlama Dili”
http://www.javadili.com/forum/f25/java-programlama-dili-tcoban-t199.html (2004)

İnternet: Akademik Bilişim Konferansı “İnternete dayalı uzaktan eğitim”
http://ab.org.tr/ab02/tammetin/47.doc (2002)

EKLER

EK–1 Dijkstra algoritmasının kodlanması

```
 public class dijkstra {

    // minpath find the lengths of the minimal
    // paths from the first node to every other node

    public static int[] minpath(int [] [] C){
      int [] D;          // the minimal path lengths
      boolean [] S;      // the set of completed vertices
      int i;

      D = new int[(C[0]).length];
      S = new boolean[(C[0]).length];

      // initialize the set of completed vertices
      S[0] = true;
      for (i = 1; i < (C[0]).length; i++) { S[i] = false; }

      // initialize the path lengths
      D[0] = 0;
      for (i = 1; i < (C[0]).length; i++) { D[i] = C[0][i]; }
//      int dmin = 0;
      // take one vertex at a time and settle its path length
      for (i = 1; i < (C[0]).length; i++) {
        int j, k, dmin;

        dmin = 0;
//      System.out.println("dmin is : " + dmin);
        k = -1;

        // choose a vertex k in -S such that D[k] in minimal
        for (j = 1; j < (C[0]).length; j++) {
//              k =j;
          if (!S[j]) {
            if (!(D[j] == 0)) {
              if (dmin == 0 || D[j] < dmin) {
                dmin = D[j];
                k = j;
                System.out.println("The path to: " + j+ " ");
                System.out.println("dmin is: " + dmin + " C[" + k +
"][" +j + "] is: " + C[k][j]);
                System.out.println("D[" + k + "] is: " + D[j]);
                System.out.println(" ");
              }
            }
          }
        }
//      k =j;
        if (k != -1) { S[k] = true; }
         if ( k != -1){
                //System.out.println("THis is k: " + k);
        for (j = 1; j < (C[0]).length; j++) {
                System.out.println("This is C["+k+"]["+j+"]: " +
C[k][j]);
          if (!S[j] && C[k][j] != 0) {
```

EK–1 (Devam) Dijkstra algoritması

```
System.out.println("This is D[" + j + "]: " + D[j] + " D[" + k + "]:
" + D[k] + "C[" + k + "][" + j + "]: " + C[k][j]);
        if ( D[j] != 0)
        {
            D[j] = Math.min(D[j],D[k]+C[k][j]);
        }
        else { D[j] =D[k]+C[k][j];}
           }
          System.out.println("D[" + j + "]final is: " + D[j]);
        }
         }

      }
      return D;
    }
```

EK–2 Hat bilgileri

HAT NO	HAT ADI
301	Örnek-Ulus-Balgat
302	Örnek-Y.Doğan-Kzl-Öveçler
303	Beştepe-Kızılay-Örnek
304	Örnek-Kızılay-A.Ayrancı
305	Örnek-Emek-Bahçeli
306	Örnek-Çalışkanlar-Kızılay
307	Örnek-Hıdırlık-Y.Doğan-Kızılay
308	Örnek-Kızılay-Beştepe-G.O.E
309	Örnek-Babür-Kzl- Bakanlık
310	Örnek-Y.Doğan-Kzl-Bakanlık
312	Karapürçek-Beşikkaya-Ulus
313	Kavaklı-Aydıncık-Peçenek-Tatlar
314	Karapürçek-Ekin Mah.-Cebeci
315	Karapürçek-Beşikkaya-Cebeci
316	Karapürçek-Eskiyol-Cebeci
317	Bağcılar-İç Aydınlık
318	Bağcılar-Ulus
320	Karapürçek-Ekin Mah.-Ulus
322	Karapürçek-Eskiyol-Ulus
323	Odtü
324	Çatak-Çigiltepe-Kızılay
325	Y.Beyazıt-Ulus
326	Toki-Dikimevi
327	Bostancık-Çigiltepe-Kızılay
328	60 Evler-Çigiltepe-Ulus
329	60 Evler-Kızılay
330	Ege Mah.-Lojman-Kızılay
331	Zerdalitepe-Ddy
332	Kayaş-E.Yol-Kızılay
333	İmrahor-Ulus-D.Evi-Toprak
334	Ege Mah.-Dikimevi
336	Yakupabdalköyü-Ulus
337	Metehan-Boğaziçi-Siteler
338	Ege Mah.-6.-Cad.-Ulus
339	Ege Mah.-Sıhhiye-Ulus
340	Ege Mah.-Kızılay-B.Evler-DDY

EK–2 (Devam) Hat Bilgileri

HAT NO	HAT ADI
341	Ege Mah.-Kızılay
343	Altıağaç-Ulus
344	Altıağaç-Kızılay
345	Karapürçek-Beşevler-T. Okullar
346	Tepecik-Ç.Çeşmesi-Kızılay
347	Tepecik-.Ç.Çeşmesi-Ulus
348	Kayaş-E.Yol-Kzl-T.Doğan-B.Evler
349	Derbent-Araplar-Ulus
350	Kızılcaköy-Ulus
351	Akşemsettin-B.İçi-Sudeposu
352	Kutludüğün - Ulus
353	Boğaziçi-4. Cadde-Ulus
354	Elmadağ
355	Zerdalitepe-Ulus
356	Kayaş-Ulus-Yeniyol
357	Kıbrısköyü-Ulus
358	B.Deresi-S.Yolu-Site-Kayaş
359	Gökçeyurt-Ulus
360	Ortaköy-Ulus
361	Yeşilbayır-Ulus
362	Tuzluçayır-Kızılay
363	Kusunlar-Ulus
364	Şahapgürler-Ulus
365	Misket-Ulus-Sıhhiye
366	Araplar-Ddy-Kızılay
367	Ulus-Şirintepe
368	Gökçeyurt-Siteler
370	Tekmezar-Kızılay
371	Kıbrısköyü-Ddy
372	Ulus-Akdere-İmrahor
373	Mutlu-Z.Blokları-Ulus
374	Zerdalitepe-Dikimevi
375	Akdere-Karşıyaka-Ulus
376	Yeşilbayır-Kızılay
377	Şahapgürler-Kızılay-T.Doğan
378	Üreğil-Kızılay
380	Peyamisefa-Kızılay

EK–3 Durak Bilgileri

NO	DURAK ADI	ENLEM	BOYLAM
1	Ege Hareket Noktası	39,89212258	32,9336214
2	Natoyolu Caddesi 1. Durak	39,89003169	32,9283428
3	Natoyolu Caddesi 2. Durak	39,89108538	32,9273987
4	Natoyolu Caddesi 3. Durak	39,89720961	32,9213047
5	Natoyolu Caddesi 4. Durak	39,89982706	32,9199529
6	Natoyolu Caddesi 5. Durak	39,9031193	32,9180646
7	Natoyolu Caddesi 6. Durak	39,90603281	32,9173994
8	Natoyolu Caddesi 7. Durak	39,90962102	32,9183221
9	Natoyolu Caddesi 8. Durak	39,91225445	32,9152322
10	Natoyolu Caddesi 9. Durak	39,91548027	32,912786
11	Natoyolu Caddesi 10. Durak	39,91687917	32,911799
12	Natoyolu Caddesi 11. Durak	39,91989083	32,9075074
13	Natoyolu Caddesi 12. Durak	39,9224745	32,9048038
14	Tıp Fakültesi Caddesi 1. Durak	39,92532136	32,9019284
15	Tıp Fakültesi Caddesi 2. Durak	39,92581502	32,8972721
16	Tıp Fakültesi Caddesi 3. Durak	39,92665424	32,8956413
17	Tıp Fakültesi Caddesi 4. Durak	39,92805292	32,8920472
18	Tıp Fakültesi Caddesi 5. Durak	39,92800355	32,8921866
19	Tıp Fakültesi Caddesi 6. Durak	39,93039769	32,8868544
20	Tıp Fakültesi Caddesi 7. Durak	39,93267657	32,8828955
21	Cemal Gürsel Cad. 1. Durak	39,93295628	32,8784108
22	Cemal Gürsel Cad. 2. Durak	39,93145076	32,8758359
23	Cemal Gürsel Cad. 3. Durak	39,92959966	32,8723383
24	Ziya Gökalp Caddesi 1. Durak	39,92724663	32,8663301
25	Ziya Gökalp Caddesi 2. Durak	39,92467959	32,8628111
26	Ziya Gökalp Caddesi 3. Durak	39,92125673	32,8566098
27	G.M.K Bulvarı 1.Durak	39,9241201	32,8487778
28	G.M.K Bulvarı 2.Durak	39,92945156	32,8460312
29	G.M.K Bulvarı 3.Durak	39,93304678	32,8416538
30	G.M.K Bulvarı 4.Durak	39,92468782	32,8480697
31	G.M.K Bulvarı 5.Durak	39,92193144	32,8509021
32	1.Cadde 1.Durak	39,91462445	32,9826736

EK–4 Hareket Bilgileri

HAT NO	TARIH	BAS	BIT	SURE	YOLCU
301	31.10.2008	05:10	05:51	41	51
301	31.10.2008	06:30	07:57	87	51
301	31.10.2008	06:35	07:16	41	69
301	31.10.2008	06:59	08:54	115	177
301	31.10.2008	07:15	09:22	127	237
301	31.10.2008	07:30	09:36	126	210
301	31.10.2008	07:45	09:51	126	169
301	31.10.2008	08:00	10:00	120	51
301	31.10.2008	09:00	10:53	113	51
301	31.10.2008	09:23	11:18	115	96
301	31.10.2008	11:00	11:20	20	9
301	31.10.2008	11:20	12:54	94	46
301	31.10.2008	11:59	12:22	23	51
301	31.10.2008	12:22	13:55	93	51
301	31.10.2008	12:30	14:33	123	110
301	31.10.2008	14:01	16:20	139	182
301	31.10.2008	14:32	16:28	116	51
301	31.10.2008	15:00	17:09	129	188
301	31.10.2008	15:30	17:50	140	254
301	31.10.2008	15:57	18:11	134	212
301	31.10.2008	16:44	19:08	144	226
301	31.10.2008	17:00	19:26	146	51
301	31.10.2008	17:40	20:00	140	166
301	31.10.2008	18:07	20:30	143	165
301	31.10.2008	18:43	20:45	122	51
301	31.10.2008	19:29	21:01	92	119
301	31.10.2008	21:31	22:51	80	111
301	31.10.2008	22:00	23:35	95	57
301	31.10.2008	22:55	23:36	41	7
301	30.10.2008	06:30	08:08	98	178
301	30.10.2008	06:35	07:13	38	36
301	30.10.2008	06:44	08:41	117	51
301	30.10.2008	06:59	09:12	133	252

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, adı : ŞAHİN, Fatih

Uyruğu : T.C.

Doğum tarihi ve yeri : 08.08.1982 Ankara

Medeni hali : Bekar

Telefon : 0 (312) 595 87 70

Faks : 0 (312) 369 33 09

e-mail : fatihsahin06@yahoo.com.

Eğitim

Derece	Eğitim Birimi	Mezuniyet tarihi
Lisans	İstanbul Üniversitesi Endüstri Müh Bölümü	2003
Lise	Çankırı Süleyman Demirel Fen Lisesi	1999

İş Deneyimi

Yıl	Yer	Görev
2005-2006	Nurus A.Ş.	ERP Uygulama Danışmanı
2006-2007	PCK Medikal	ERP Sistem Yöneticisi
2007-	SGK	Çözümleyici

Yabancı Dil

İngilizce

Hobiler

Bilgisayar teknolojileri, Basketbol, Müzik, Sinema