T.C.
PAMUKKALE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ARAÇ ROTALARININ EN KISA YOL ALGORİTMALARI KULLANILARAK BELİRLENMESİ VE .NET ORTAMINDA SİMÜLASYONU

Şahin BAYZAN

Yüksek Lisans Tezi

DENİZLİ – 2005

ARAÇ ROTALARININ EN KISA YOL ALGORİTMALARI KULLANILARAK BELİRLENMESİ VE .NET ORTAMINDA SİMÜLASYONU

Pamukkale Üniversitesi
Fen Bilimleri Enstitüsü
Tarafından Kabul Edilen
Bilgisayar Mühendisliği Anabilim Dalı
Yüksek Lisans Tezi

Şahin BAYZAN

Tez Savunma Tarihi: 25 08 2005

DENİZLİ – 2005

TEZ SINAV SONUÇ FORMU

Bu tez tarafımızdan okunmuş, kapsamı ve niteliği açısından Yüksek Lisans Tezi olarak kabul edilmiştir.

Yrd. Doç. Dr. Sezai TOKAT

(Yönetici)

Yrd. Doç. Dr. A.Kadir YALDIR

(Jüri Üyesi)

Yrd. Doç. Dr. Şevket CİVELEK

(Jüri Üyesi)

Pamukkale Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 14.09.2005 tarih ve 20/09 sayılı kararıyla onaylanmıştır.

Prof. Dr. Mehmet Ali SARIGÖL

Müdür

Fen Bilimleri Enstitüsü

TEŞEKKÜR

Yüksek lisans tez çalışmalarım süresince değerli zamanını benden esirgemeyen, bilgi ve tecrübesi ile her konuda bana yön gösteren, kendisinden çok şey öğrendiğim değerli danışman hocam Yrd. Doç. Dr. Sezai TOKAT'a emeklerinden dolayı teşekkürü bir borç bilirim.

Tezimdeki simülasyonun C# programlama dilinde geliştirilmesi aşamasında, benim için zaman ayırıp yardımlarını esirgemeyen Öğr. Gör. Önder ÇİVRİL'e gönülden teşekkür ederim.

Çeşitli konularda kıymetli fikirleri ile desteklerini esirgemeyen Bölüm Başkanım Doç. Dr. Halil KUMSAR başta olmak üzere Yrd. Doç. Dr. A. Kadir YALDIR'a ve tüm diğer bölüm öğretim üyelerine teşekkürlerimi sunarım.

Tez çalışmamın her safhasında düşünceleri ile katkıda bulunan, yardımlarını ve bilhassa desteklerini esirgemeyen Alper UĞUR'a ve tüm diğer çalışma arkadaşlarıma şükranlarımı sunarım.

Bugünlere gelmemde en büyük pay sahibi olan anneme, babama; ayrıca bana her konuda destek olan eşim Saliha BAYZAN'a, varlıklarıyla bana büyük moral kaynağı olan çocuklarım Muhammed Sadık BAYZAN ve Ömer Faruk BAYZAN'a teşekkür etmeyi bir vazife olarak görüyorum.

Ayrıca özel yaşantımda bana manevi destek olan bütün dost ve yakınlarıma; tüm sıkıntılarımı benimle paylaşan, bana sürekli moral desteği sağlayan herkese teşekkürlerimi sunarım.

Şahin BAYZAN

ÖZET

Bu tezde, graf veri modeline uyarladığımız bir coğrafi alandaki Araç Rotalama Probleminin (ARP) C# ortamında simülasyonunu geliştirilmiştir. Simülasyon kullanılarak ARP probleminde, araçların talep noktalarında bekleme sürelerinin taşıma maliyetine etkisini irdelenmiştir.

Değişik noktalardan farklı zaman aralıklarında gelen rasgele 50 talebin karşılanmasında bekleme süresinin aracın aldığı toplam yola etkisini görmek için iki algoritma önerilmiştir. Bunlardan ilki, talebin geldiği anda kendisine en yakın aracın talep listesine eklenmesi, ikincisi talebin geldiği anda araçların gittiği noktalardan yakın olanının listesine eklenmesi şeklindedir. Bu iki yaklaşımdan elde edilen sonuçlar karşılaştırılarak her iki durum için bekleme sürelerinin alınan toplam yola etkisi incelenmiştir. Böylece, verilen bir senaryo için önerilen yaklaşımlardan hangisinin daha uygun olduğu belirlenmeye çalışılmıştır.

Anahtar kelimeler: Araç Rotalama, Gezgin Satıcı Problemi, Graf Teorisi

ABSTRACT

In this thesis, a simulation program for the vehicle routing problem is developed where a geographical domain that is adapted to a graph data model on C#.NET platform is investigated. With the help of the simulation, the effects of waiting time of the vehicles at the demand points are examined considering the transport costs.

Two different algorithms are suggested to show the effect of waiting time on the total distance. Different scenarios are created for randomly selected 50 demands requested at various time instants from different nodes. In the first algorithm, the demand is simply appended to the demand list of the vehicle nearest to the demand point. In the second one, on the other hand, the demand is appended to the demand list of the vehicle nearest to the demand point at the time after meeting the current demand.

Considering both algorithms, the effects of waiting time to the total path are analyzed by comparing the simulation results and the appropriate algorithm for a given scenario is determined.

Keywords: Vehicle Routing Problem, Traveling Salesman Problem, Graph Theory

İÇİNDEKİLER

Sayfa

Birinci Bölüm

GİRİŞ

İkinci Bölüm

GRAF KURAMI VE GRAF VERİ MODELİ

Üçüncü Bölüm

ARAÇ ROTALAMA PROBLEMİ

Dördüncü Bölüm

NESNE YÖNELİMLİ PROGRAMLAMA VE C# PROGRAMLAMA DİLİ

Beşinci Bölüm

ARAÇ ROTALAMA PROBLEMİ SİMÜLASYONU

Altıncı Bölüm

SONUÇLAR VE ÖNERİLER

ŞEKİLLER DİZİNİ

BİRİNCİ BÖLÜM
GİRİŞ

1. GİRİŞ

Taşımacılık, çok gelişmiş ulusların ekonomilerinin en önemli parçasını oluşturur. Günümüz dünyasında bireysel firmalar için, işletme maliyetlerinin en önemli kısmını nakliyat maliyetlerinin oluşturduğu görülmektedir. Bu ekonomik önem, özel şirketleri ve akademik araştırmacıları, nakliyattaki verimliliği geliştirmek ve ilerletmek için yöneylem araştırması ve yönetim biliminin kullanımını takip etme noktasında oldukça motive etmiştir. Taşımacılık, diğer bir ifade ile nakliyatta en önemli problemlerden birisi Araç Rotalama Problemidir (ARP). Bu problem fiziksel dağıtımda ve lojistikte merkezi bir rol oynamaktadır.

Günümüzün küresel rekabeti karşısında, verimli lojistik planlama gereksinimi, en çok, üreten ve ürettiğini pazarlayan üretim ve dağıtım firmaları için öncelikle üzerinde durulması gereken bir sorun olmuştur. Bir şirketin başarısı; ürettiği ürünün kalitesiyle, müşteri sayısının fazlalığıyla ve ürettiği ürünleri en kısa zamanda ve en az maliyetle müşterilerine ulaştırmasıyla yakından alakalıdır.

Bir şirket ürünlerini müşterilerine ya doğrudan ya da bayilikleri aracılığıyla ulaştırmaktadır. Bayiler, bulundukları yerleşim alanının coğrafi yapısına göre, yerleşim alanının değişik noktalarına açtıkları depolar aracılığıyla müşterilerine hızlı ve en az maliyetle ulaşmaya çalışmaktadırlar. Müşterilerin ürün talepleri, talebin geldiği noktaya göre bu depolardan en uygun olanı kullanılarak karşılanmaktadır. Talebin geldiği noktaya göre kullanılacak en uygun deponun belirlenmesi, bu depo ile talep noktası arasındaki en kısa mesafenin hesaplanması ve takip edilecek rotanın belirlenmesi değişik kısa yol algoritmaları kullanılarak hesaplanabilmektedir. Burada amaç, ürünü en az maliyetle, en kısa zamanda ve en uygun yolu kullanarak müşteriye ulaştırmaktır

Coğrafi alanın yol ağı çok karmaşık olabileceği gibi çok basit de olabilir. Bir depodan müşteriye gitmek veya müşteriden depoya dönmek için, bu ağ içerisinde kullanabileceğimiz birçok yol seçeneği olabilir. Bu yol seçenekleri içerisinde en uygun yol seçeneği, gideceğimiz noktaya bizi en kısa zamanda ve en az maliyetle ulaştıran yoldur.

1.1 Araç Rotalama Problemi

ARP, bir depodan coğrafi bir bölge içerisinde belirli noktalara dağılmış müşterilere en uygun teslimat ve toplama yapacak araçların rotalarının belirlenmesi problemi olarak tanımlanabilir. Başka bir ifadeyle, bir yol ağı içerisinde, herhangi bir depodan harekete başlayan bir veya birden fazla aracın, en az maliyetle istenilen talep noktasına gidip harekete başladığı noktaya dönebileceği en uygun rotanın bulunması problemidir.

ARP, Bir şebekedeki belli bir arz ve talebe sahip düğümlerin en az maliyetle ziyaret edilmesini sağlayacak turu bulmayı amaçlayan bir problemdir (Dantzig ve Ramser, 1959). Problemdeki araçlar, okul servis araçları, çöp arabaları, kargo dağıtım araçları, şirket ürünlerini müşterilere dağıtan araçlar v.b olabilir. Buradaki en uygunluk, araçların en kısa sürede, en az maliyetle, en uygun varyasyonun kullanılarak hedefe varıp, harekete başladıkları noktaya dönmesi olarak ifade edilebilir.

Her rota depoda başlar ve depoda biter. Her müşteriye sadece bir araç ile servis yapılır. Problem, müşteriye servis yapacak olan her araç için maliyeti en aza indirgeyecek yan kısıtlamalar içermektedir. Bu yan kısıtlamalar şunlardır:

1. Kapasite kısıtlaması: her bir aracın yükü kapasitesini aşmayacak şekilde olmalıdır. Bu kapasite ağırlık ve hacim cinsinden ifade edilebileceği gibi bazı durumlarda da kapasite kısıtlaması aracın fiziksel kapasitesi ile alakalı olmayabilir. Örneğin, bankalar için para toplayan ya da dağıtan aracın kapasitesi güvenli olarak taşıyabileceği maksimum para miktarıdır.

2. Toplam zaman ve mesafe kısıtlaması: Bir sürücünün belirli bir saat devamlı araç kullanmasına izin verilmiştir. Mesela kanunlar, bir sürücünün on saatten fazla devamlı bir şekilde araç kullanmasına izin vermez.

3. Zaman aralığı kısıtlaması: Bazı müşteriler belirli zamanlarda teslimat ve toplama yapılmasını isterler. Eğer araç bu zaman aralığından önce müşteriye ulaşmışsa beklemek zorundadır. Mesela araç müşteriye belirtilen saat aralığında değil de erken varmışsa müşteri yerinde olmayabilir.

4. Öncelik ilişkisi: Bazı müşterilere rotanın başında ya da sonunda teslimat yapılabilir. Eğer teslimat ve toplama karışıksa, teslimat işlemi daima toplamadan önce olmalıdır.

ARP, değişik uygulama alanları bulmuş bir problemdir. Örneğin, bazı müşterilerin merkezi bir noktadan kargo, yiyecek, akaryakıt, beyaz eşya gibi ürünlerinin teslimat işlemlerinde kullanılacak araçların rotalarının belirlenmesi ARP'nin alanına girmektedir. Bazı taşımacılık işleri, araçların rotalanmasını gerektirmektedir. Bunları şu şekilde sıralamak mümkündür.

1. Okul servis araç rotalama: Belirli bir kapasitesi olan okul servis araçlarının öğrencileri her sabah belirlenen noktalardan alıp, okul bitiminde yine aynı noktaya bırakması işini öğrenci seyahat süresini ve aracın kat ettiği mesafeyi en aza indirgeyecek şekilde yapmasıdır.

2. Paket teslimatı ve toplanması: Kargo şirketlerinin gelen kargoları dağıtmada ve gidecek kargoları müşteriden toplamada araçların takip edecekleri en uygun güzergâhların belirlenmesi gerekmektedir.

3. Meşrubat türü içecek teslimatı: Bu tür ürünler üreten firmaların ürünlerini bakkal ve marketlere dağıtmada kullandıkları araçların gidecekleri en uygun rotaların belirlenmesi de aynı şekilde araçların rotalanmasını gerektirmektedir.

4. Bankalardan para toplama: Gün sonunda banka şubelerinden toplanılan paraların merkez bankaya getirilmesinde kullanılan araçların güzergâhlarının belirlenmesi de araç rotalama işlemi gerektirmektedir.

5. Atık Toplama: Atık yönetim şirketleri, her bir atık toplayan araç için, yerleşim yerlerinden topladıkları atıkları, atıkların biriktirildiği yere getirecekleri en uygun rotların belirlenmesi için de araçların rotalanmasına ihtiyaç vardır (Yetkin, 2004).

6. Yemek Teslimatı: Farklı şirketlere yemek yapan lokantalar, yemekleri şirketlerin yemek saatlerinden kısa bir süre önce teslim etmeleri gerekmektedir. Bu durumda yemekleri teslimat işinde kullanılan araçların en uygun güzergâhı taşıyacak şekilde rotalanmasına ihtiyaç vardır.

7. Süt Endüstrisi: Süt üreten mandıralardan sütleri işlenecekleri merkeze götürmek için toplanması da sütleri toplayan araçların en uygun şekilde rotalanmasını gerektirir.

8. Servis Endüstrisi: Fabrikada çalışan işçilerin servis araçlarıyla fabrikaya taşınması için belirli noktalardan toplanması ve mesai bitiminde tekrar aynı noktaya bırakılması bu servis araçlarının en uygun şekilde rotalanmasını gerektirir.

Sadece hesaplama karmaşıklığı açısından değil, içerdiği giriş bilgisi noktasından bakıldığında ARP çözümlemesi zor bir problemdir. Çok iyi bilinmekte olan Gezgin Satışçı Probleminin (GSP) genelleştirilmiş halidir. Kesin algoritmalar sadece küçük hacimli örnekleri çözerler. Pratik örnekler için birçok araştırmacı bulgusal ve uygulamalı konular üzerine yoğunlaşmışlardır.

ARP'nin önde gelen şebekesi, bir yol ağı ya da euclidean bir yüzeyde noktalarca tanımlanmış tamamlanmış bir graf olabilir. Birçok araştırmacı, her bir müşterinin yerinin (x,y) koordinatlarıyla belirtildiği Euclidean ARP üzerinde odaklanmıştır.

Müşteriler arasındaki mesafeler Euclidean mesafesi olarak düz bir çizgili yol olarak yaklaşık hesaplanmıştır. Yol ağının bu şekilde kabul edilmesi durumunda, problem tamamlanmış bir graf üzerinde ARP' ye dönüştürülebilir. Bu durumda her bir müşteri arasındaki en kısa mesafeyi hesaplama söz konusu olur. Bu çevirme uygulaması daima bir yolun topolojik yapısına zarar verir ve bu da ARP algoritmasını daha az verimli kılar. Örneğin, bu yol ağını basit bir yolu olan bir ağ varsayarsak, yol boyunca dizilmiş müşteriler birim talepler sahip olur. En uygun çözüm basit olarak, aracın üzerindeki yük bitene kadar, en uzaktaki müşterilere bir aracın üzerindeki yükün bir kısmını daha uzaktakine de diğeri kısmını tahsis etmektir. Bununla birlikte tamamlanmış bir graf üzerinde, bir örneğin ARP'ye çevrilmesi durumunda çözüm belli değildir.

Bu tezde, kısa yol algoritmaları kullanılarak en kısa yolu bulan aynı zamanda da üzerinde depolar, müşteriler, depolarda belirli hız ve kapasiteye sahip araçların bulunduğu ve müşterilerden gelen taleplerin en uygun güzergahın belirlenerek karşılandığı yani araçların rotalandığı bir simülasyon geliştirilmiştir.

1.2 Çizge (Graf) Kuramı ve Bilgisayar Modelleme

Tarih boyunca taşımacılık, dünya coğrafyasında çok önemli bir yere sahip olmuştur. İnsanlar ticaret için kendi memleketlerinden farklı yerlere gidip gelmişler ve ürünlerini satma gayreti içinde olmuşlardır. Bulundukları zamanın şartlarına ve coğrafi koşullarına göre ellerindeki ürünlerini en kısa yoldan ve en kısa zamanda müşterilerine ulaştırmak için çalışmışlar, bunları yaparken kervanlar oluşturmuşlar, ürünlerini taşımak için at, deve, katır gibi taşıma araçları kullanmışlardır.

Taşıma ve ulaşım araçlarının zamanla gelişmesi, özellikle gemi gibi deniz ulaşım araçlarının gelişimi yeni kıtaların keşfinde çok önemli yere sahip olmuştur. Dünya nüfusunun artması beraberinde şehirleşmeyi getirmiş, yeni yerleşim yerlerinin ortaya çıkmasına sebep olmuştur.

Yerleşim yerleri arasındaki ulaşım ağının gelişmesi, bir yerleşim yerinden diğerine gidilebilecek yol alternatiflerini artırmıştır. Bir noktadan diğer bir noktaya giderken ya da değişik noktalar arasında taşımacılık gibi bir faaliyette bulunurken hedeflenen şey sadece varılacak noktaya herhangi bir şekilde ulaşmak olmaktan çıkmış ve en kısa yoldan, en kısa zamanda, en az maliyetle nasıl gidileceği olmuştur. Noktalar arasındaki mesafe, zaman ve maliyet önem kazanmıştır.

Yerleşim yerleri arasındaki ulaşım ağının genişliği, bu noktalar arasındaki ulaşımda yol güzergâhının tespiti diğer bir ifade ile takip edilecek rotanın belirlenmesi gibi birtakım problemlerin ortaya çıkmasına sebep olmuştur. Bu tür problemlerin çözümü için değişik yöntemler ve modellemeler yapılmıştır. İşte Çizge Kuramı (Graph Theory)'da bu tür problemlerin çözümünde kullanılan yöntemlerden biridir.

Çizge kuramı, matematiksel araştırmalarda, elektrik mühendisliğinde, bilgisayar mühendisliğinde ve bilgisayar ağlarında, iş idaresinde, ekonomide, pazarlamada ve haberleşmede önemli bir araç olarak kullanılmaktadır. Özellikle bazı problemler, graflara uyarlanarak modellenebilmektedir. Şehir içi ve şehirlerarası yol ağı, yerel ve geniş alan ağları, haberleşme ağları ve benzeri ağlar graf veri modeline uyarlanarak çözümlenebilmektedir.

Bir graf, köşeler (vertices) ve bu köşeleri çizgelerle (edges) birleştiren kenarlardan oluşur. Gerçek hayatta yerleşim yerleri, yerleşim yerlerindeki caddelerin kavşak noktaları birer köşe, yerleşim yerlerini ve kavşakları birbirine bağlayan yollarda çizge olarak kabul edilebilir. Graf kuramının ilk uygulandığı problemlerden birisi Könisberg köprüsü problemidir ve ilk kez 1736 yılında Leonhard Euler tarafından çözümlenmiştir.

1.3 Simülasyon

Burada ele alınan problem, bir coğrafi alan içerisinde bulunan bir veya daha çok deponun kullanılarak, bir veya daha fazla müşteriden gelen talepleri karşılamak için kullanılacak kapasiteleri kısıtlı araçların, en az maliyetle talebi karşıladığı turu bulmaktır.

Her bir talep için kullanılacak depo, araç sayısı ve araçların takip edecekleri güzergâh belirlenecektir. Her talepten sonra araçların konumları, üzerlerindeki yük miktarı ve gelen taleplerin karşılanma durumları güncellenecektir. Problemin çözümü için yapılan kısıtlamalar ve varsayımlar şöyledir.

- Araçlar eşdeğer hız ve kapasiteye sahiptir.
- Depo sayısı ve depodaki araç sayısı kullanıcı tarafından belirlenecektir.
- Depo dışındaki her düğüm bir müşteri olarak kabul edilecektir ve taleplerin bu noktalardan geldiği varsayılacaktır.
- Gelen talepler öncelik sırasına göre karşılanacaktır.
- Üzerindeki yükü biten araç kendisine en yakın depoya dönecektir.
- Bir araç herhangi bir talebi karşıladığında üzerinde yük varsa sıradaki talebi karşılamak için hareket eder. Sırada talep yok ise bekler. Bekleme süresi kullanıcı tarafından belirlenir. Bekleme süresi biten araç en yakın depoya döner.

1.4 Bölüm Özeti

Bu bölümde kısaca taşımacılığın günümüz dünyası için önemi irdelenmiştir. Ülkelerin ekonomileri açısından taşımacılığın ne derece önemli olduğu belirtilmiş ve taşımacılıkla birlikte ortaya çıkan ve en uygun yolun bulunmasını inceleyen ARP problemine değinilmiştir. Bu tür problemlerin uyarlanabileceği graf veri modeli hakkında bilgi verilmiş ve geliştirilen simülasyondaki bazı kısıtlamalar anlatılmıştır. Bir sonraki bölümde Graf Teorisi detaylı olarak incelenecektir.

İKİNCİ BÖLÜM
GRAF TEORİSİ VE GRAF VERİ MODELİ

2. GRAF TEORİSİ VE GRAF VERİ MODELİ

İlk olarak 1736 yılında Matematikçi Leonhard Euler tarafından önerilen ve yine aynı bilim adamı tarafından ünlü Könisberg köprüsü probleminin çözümünde kullanılan Graf Teorisi, bir olay ve ifadenin, düğüm ve çizgiler kullanılarak gösterilmesi biçimidir. Kökleri 18. yüzyıla dayanan Graf Teorisi günümüzde bilgisayar uygulamalarında ve modellemelerde çokça kullanılmaktadır. Fizik, kimya gibi temel bilimlerde, mühendislik uygulamalarında ve tıp biliminde karşılaşılan birçok problemin çözümü ve modellenmesi Graf teorisi temel alınarak yapılmaktadır. 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 gösterilebilir (Jonathan G, 1999).

Graflarla ifade edilebilecek problem ve olayları incelemek ve bunları temel alarak çözümler üretmek takip edilen en ideal yoldur. Bir kimya problemindeki zincir yapısı, havacılıkta ve denizcilikte rotaların belirlenmesi, bilgisayar ağlarındaki veri paketlerinin yönlendirilmesi, şehir içi ve şehirlerarası yol ağı ya da bir yerleşim yerinin içme suyu altyapısı gibi bilimsel ve teknik modellemelerde ve bunların bilgisayar ortamında simülasyonlarının gerçekleştirilmesinde çoğunlukla graflar ve bunlara ait teoremler, aksiyomlar çözüm olabilmektedir. 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 teorileri kullanılabilir (Akman, 2002).

Bu bölümde, graf teorisinin genel bir tanımı yapıldıktan sonra graf türlerine değinilecek, graf tabanlı problemler ele alınacak ve graf tabanlı problemlerin çözümünde kullanılan algoritmalar hakkında temel bilgiler verilecektir.

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 uyarlanabilir.

Graf, matematiksel tanım olarak, düğümler ve bu düğümler arasındaki ilişkiyi gösteren kenarlardan meydana gelen bir kümedir; buradaki mantıksal ilişki düğüm ile düğüm veya düğüm ile kenar arasında kurulur. Grafın grafik gösterilimi (1)'de gösterildiği gibi düğümler ve bunları birbirine bağlayan kenarlarla yapılır. Bir graf üzerinde *n* tane düğüm ve *m* tane kenar varsa, matematiksel ifadesi, düğümler ve kenarlar kümesinden elemanların ilişkilendirilmesiyle yapılır.

$$\begin{aligned} D &= \{d_0, d_1 d_2 \ldots . d_{n-1}, d_n\} \longrightarrow \textit{Düğümler kümesi} \\ K &= \{k_0, k_1, k_2 \ldots . k_{m-1}, k_m\} \longrightarrow \textit{Kenarlar kümesi} \\ G &= \{D, K\} \Rightarrow \textit{Graf} \end{aligned} \qquad (1)$$

Tanım 2.1.1: Basit bir *G = {D, K}* grafı, boş olmayan *D* düğümler kümesine ve bu düğümler kümesindeki elemanları sıralı olma özelliğine bakmaksızın bağlayan veya ilişkilendiren *K* kenarlar kümesine sahiptir.

Tanım 2.1.2: Bir G grafı üzerindeki d_i ve d_j adlı iki düğüm, kenarlar kümesinde bulunan bir kenarla ilişkilendiriliyorsa bu iki düğüm birbirine komşu düğümlerdir; $k = \{d_i, d_j\}$ biçiminde gösterilir ve *k* kenarı hem d_i hem de d_j düğümleriyle bitişiktir denilir. Başka bir ifadeyle *k* kenarı d_i ve d_j düğümlerini birbirine bağlar veya d_i ve d_j düğümleri *k* kenarının uç noktalarıdır denilir.

Tanım 2.1.3: Bir *G* grafı komşuluk ilişkisiyle gösteriliyorsa G_{dd} *={(*d_i*,* d_j*)..}*, bitişiklik ilişkisiyle gösteriliyorsa G_{dk}*={(*d_i*,* k_j*)…}* şeklinde ifade edilir. (G_{dd}: düğümdüğüm, G_{dk}: düğümkenar)

Tanım 2.1.4: Bir G grafı üzerindeki kenarlar bağlantılarının nereden başlayıp nerede sonlandığını belirten yön bilgilerini içeriyorsa **yönlü graf veya yönlendirilmiş graf** (directed graf) olarak adlandırılır. Yönlü graf üzerindeki bir kenar, iki düğüm arasında bağlantı olduğunu, fakat ilişkinin oka bağlı olarak tek yönlü olduğunu gösterir. İki düğüm arasında karşılıklı bir ilişki varsa zıt yönde iki kenar kullanılır. Yönlü graflar, matematiksel olarak gösterilirken < > karakter çiftiyle gösterilir. Yönlü grafların komşuluk matrisleri, her bir bağlantı için birebir karşı bağlantısı yoksa simetrik olmazlar. Bilgisayar ağları yönlü grafa güzel bir örnektir. Şekil 2.1'deki grafların birisi 4, diğeri 5 düğümlü yönlü graftır. Bu grafların komşuluk ilişkisi şu şekilde yazılır.

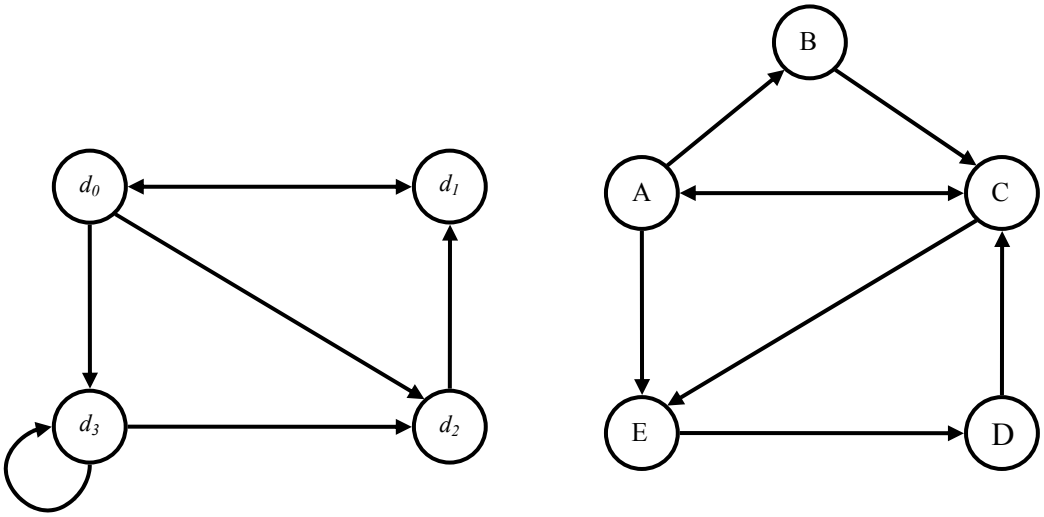
4-düğümlü için matematiksel gösterim

$$G_{dd} = \{< d_0, d_1 >, < d_0, d_2 >, < d_0, d_3 >, < d_1, d_0 >, < d_2, d_1 >, < d_3, d_2 >, < d_3, d_3 >\}$$

5-düğümlü için matematiksel gösterim

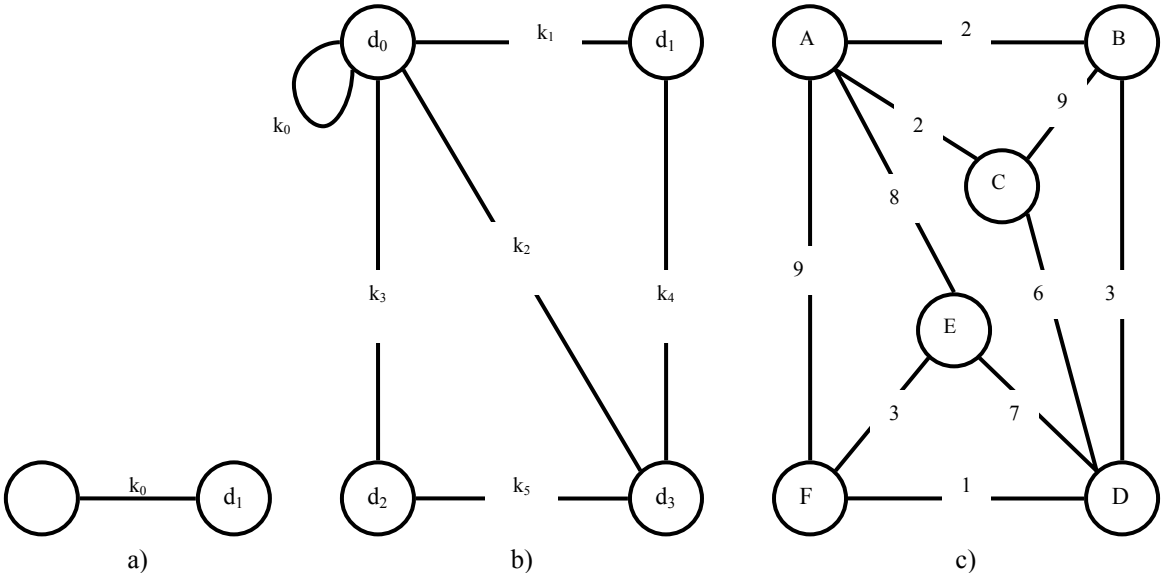
$$G_{dd} = \{< A, B >, < A, C >, < A, E >, < B, C >, < C, A >, < C, E >, < D, C >, < E, D >\}$$

İki düğüm arasında karşılıklı bağlantı bulunuyorsa farklı yönde iki kenar kullanılmaktadır. $<d_0,d_1>$ ve $<d_1,d_0>$, <A, C> ve <C, A> olduğu gibi.



Şekil 2.1: Yönlü graf örnekleri

Tanım 2.1.5: Bir G grafı üzerindeki kenarların ağırlıkları veya sayısal değerleri varsa ve bu değerler eşit değilse, ayrıca her biri farklı bir değer alabiliyorsa, bu graf **maliyetli graf** (weighted graph) olarak adlandırılır ve maliyet bilgisi ile birlikte gösterilir. Eğer tüm kenarların maliyeti 1 veya birbirine eşitse maliyetli graf olarak adlandırılmaz, eğer yön bilgisi de bulunmuyorsa bu graf **basit graf** olur. Basit graf yönsüz ve maliyetsiz olan graftır. Maliyetli graflarda bazen kenarların simetrili gösterimine ihtiyaç duyulabilir. Böyle bir durumda, graf üzerindeki kenar iki kez gösterilmiş olabilir. Örneğin Şekil 2.2c'de A'dan B'ye 2 maliyetli kenarın simetrisi de B'den A'ya da 2 maliyetli şeklinde ifade edilebilir. Bu durumda bir kenar iki kez gösterilmiş olur. Maliyetli ve yönlü graflar, ağ olarak da bilinirler ve bu şekilde adlandırılırlar.



Şekil 2.2: Çeşitli graf türleri

Tanım 2.1.6: Düğümlerden düğümlere olan bağlantıyı göstermek için kullanılan kare matrise **komşuluk matrisi (Adjacency Matrice)** denir. Komşuluk matrisinin elemanları k_i değerlerinden oluşur. Komşuluk matrisi G_{dd}'nin matris biçiminde gösterilmesinden meydana gelir. Eğer komşuluk matrisi $G_{dd} = [a_{ij}]$ ise, yönlü-maliyetsiz graflar için komşuluk matris şartı (2)'deki gibi olur.

$$a_{ij} = \begin{cases} 1;\ (d_i, d_j)\ \in K \text{ ise} \\ 0;\ \text{Diğer durumlarda} \end{cases} \qquad (2)$$

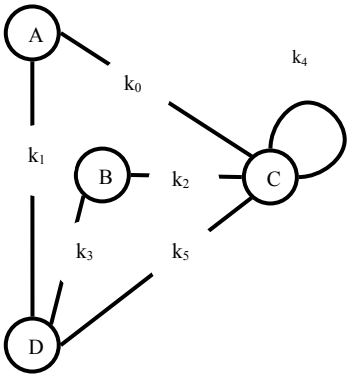
Basit graflar için ise, (3)'deki gibidir

$$a_{ij} = \begin{cases} 1;\ (d_i, d_j)\ \in K \text{ veya } (d_j, d_i)\ \in K \text{ ise} \\ 0;\ \text{Diğer durumlarda} \end{cases} \qquad (3)$$

Tanım 2.1.7: Düğümlerle kenarlar arasındaki bağlantı ilişkisini göstermek için kullanılan matrise **bitişiklik matrisi (Incedence Matrice)** denir. Matrisin satır sayısı düğüm, sütün sayısı kenar sayısı kadar olur. Bitişiklik matrisi de G_{dk}'nın matris şeklinde gösterilmesinden meydana gelir. Eğer bitişiklik matrisi $G_{dk} = [m_{ij}]$ ise maliyetsiz graflar için bitişiklik şartı (4)'deki gibidir.

$$m_{ij} = \begin{cases} 1;\ (d_i, k_j)\ \text{Bağlantısı varsa} \\ 0;\ \text{Diğer durumlarda} \end{cases} \qquad (4)$$

Basit bir grafın komşuluk ve bitişiklik matrisi, komşuluk ve bitişiklik matris tanımlarından yola çıkılarak Şekil 2.3'deki gibi elde edilir.



a) Graf

A	A	B	C	D
A	0	0	1	1
B	0	0	1	1
C	1	1	1	1
D	1	1	1	1

b) Komşuluk Matrisi

M	k_0	k_1	k_2	k_3	k_4	k_5
A	1	1	0	0	0	0
B	0	0	1	1	0	0
C	1	0	1	0	1	1
D	0	1	0	1	0	1

c) Bitişiklik Matrisi

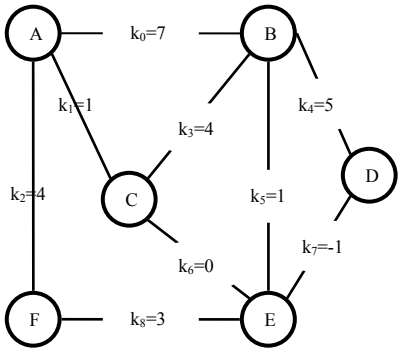
Şekil 2.3: a) Graf b) Komşuluk matrisi c) Bitişiklik matrisi

Şekilde 2.4a'da verilen grafın 4 adet düğümü ve 6 adet kenarı vardır. Düğümler D = {A, B, C, D}, kenarlarda K = {k_0, k_1, k_2, k_3, k_4, k_5} kümesinden oluşmaktadır. A olarak adlandırılan 4x4'lük komşuluk matrisinin hem satır hem de sütunları A, B, C, D olarak adlandırılmıştır. İki düğümün arasında doğrudan bir kenar var ise o iki satır ve sütunun kesiştiği yere bağlantı vardır anlamında 1 yazılır. Bağlantı yoksa 0 yazılır. Bir satırdaki sıfırdan farklı değerler o satıra ilişkin düğümlerin hangi düğümlere bağlantısı olduğunu gösterir.

Şekil 2.4c'de verilen 4x6'lık M matrisinde satırlardaki sıfırdan farklı değerler o satırla ilgili düğüme bağlı olan kenarları gösterir. Bitişiklik matrisinde satırlardaki sıfır değerleri; ilgili düğümün, sütunda ifade edilen kenara bitişik olmadığını gösterir.

Yönsüz graflarda komşuluk matrisi simetrik özellik taşır; çünkü graf yönlü olmadığından A düğümünden B'ye olan bağlantı, aynı zamanda B düğümünden A'ya olan bağlantıdır. Matrisin simetri olma özelliği köşegen (diyagonal) çizgisine göredir.

Maliyetli graflarda, verilen graf maliyetlidir; kenarların pozitif, negatif ya da sıfır olabilen maliyetleri söz konusudur. Böyle bir durumda basit grafta olduğu gibi bağlantı yok anlamında sıfır sayısı kullanılmaz. Çünkü sıfır maliyetli kenar olabilir; bunun yerine sonsuz (∞) yazılmalıdır. Şekil 2.4'de gösterilen 6 düğümlü maliyetli grafın A komşuluk matrisinde, basit graftaki gibi 0 veya 1 değerleri değil de maliyet değerleri ve ∞ sayısı bulunur.



a)

Komşuluk Matrisi

A	A	B	C	D	E	F
A	∞	7	1	∞	∞	4
B	7	∞	4	5	1	∞
C	1	4	∞	∞	0	∞
D	∞	5	∞	∞	-1	∞
E	∞	1	0	-1	∞	3
F	4	∞	∞	∞	3	∞

b)

Bitişiklik Matrisi

M	k_0	k_1	k_2	k_3	k_4	k_5	k_6	k_7	k_8
A	7	1	4	∞	∞	∞	∞	∞	∞
B	7	∞	∞	4	5	1	∞	∞	∞
C	∞	1	∞	4	∞	∞	0	∞	∞
D	∞	∞	∞	∞	5	∞	∞	-	∞
E	∞	∞	∞	∞	∞	1	0	-1	3
F	∞	∞	4	∞	∞	∞	∞	∞	3

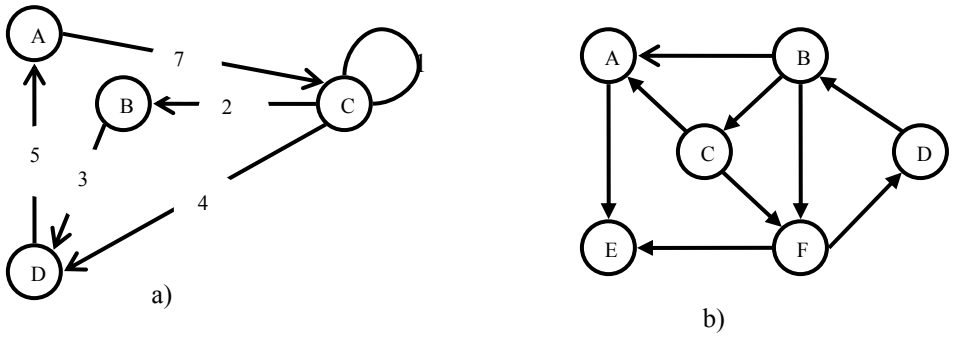
c)

Şekil 2.4: a) Maliyetli graf b) Komşuluk matrisi c) Bitişiklik matrisi

Bitişiklik matrisi M'de ise, her satırdaki sayısal değerler, o satıra bitişik olan kenarların varlığını ve maliyetini gösterir. Burada dikkat edilmesi gereken, her sütunda iki sayısal değerin varlığı ve bunların eşit olduğudur. Çünkü bir kenarın iki ucu vardır.

Maliyetli graflarda komşuluk matrisi köşegen çizgisine göre simetrik özellik taşır. Çünkü bir düğümden diğer düğüme olan yön bilgisi olmadığından iki yönlüdür. Dolayısıyla A-B arasında bir yol varsa, B-A arasında da vardır. Matrisin simetriliği çizilen diyagonal çizgisine göredir.

Yönlü ve yönlü-maliyetli grafların komşuluk matrisleri simetrik özellik taşımaz. Her bir kenar sadece bir düğümden diğerine olan bağlantıyı gösterir, ters yönde bir bağlantı olamayacağından simetrik olmaz. Bu tür graflara ait komşuluk matrislerinin simetrik olabilmesi için her bir kenarın ters yönde eşleniğinin olması şarttır. Şekil 2.5'da yönlü-maliyetli ve yönlü graf örnekleri gösterilmiştir. Bu grafların komşuluk matrisleri de Şekil 2.6'da gösterilmiştir.



Şekil 2.5: a)Yönlü maliyetli graf b) Yönlü graf

A	A	B	C	D
A	∞	∞	7	∞
B	∞	∞	∞	3
C	∞	2	1	4
D	5	∞	∞	∞

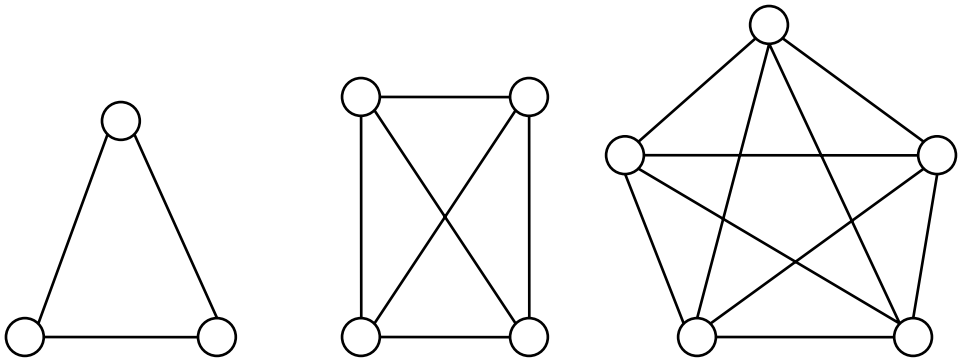
Yönlü grafların komşuluk Matrisleri, birebir bağlantı için birebir bağlantı yoksa simetrik olmaz

A	A	B	C	D	E	F
A	0	1	0	0	0	1
B	0	0	1	0	1	0
C	1	0	0	0	1	0
D	1	0	0	0	1	0
E	0	0	0	1	0	1
F	0	0	0	0	0	0

Şekil 2.6: Yönlü-maliyetli graf ve yönlü graf için komşuluk matrisi

Tanım 2.1.8: Düğüme bağlı toplam uç sayısına o düğümün **düğüm derecesi (node degree)** denir. Çevrimli kenarlar aynı düğüme hem çıkış hem de giriş yapabilir. Bu durum düğümün derecesini iki artırır. Yönlü graflarda, düğüm derecesi giriş derecesi ve çıkış derecesi olarak ayrı olarak belirtilir. Yönlü graflarda düğümün derecesi, düğümün giriş ve çıkış derecelerinin toplamına eşittir.

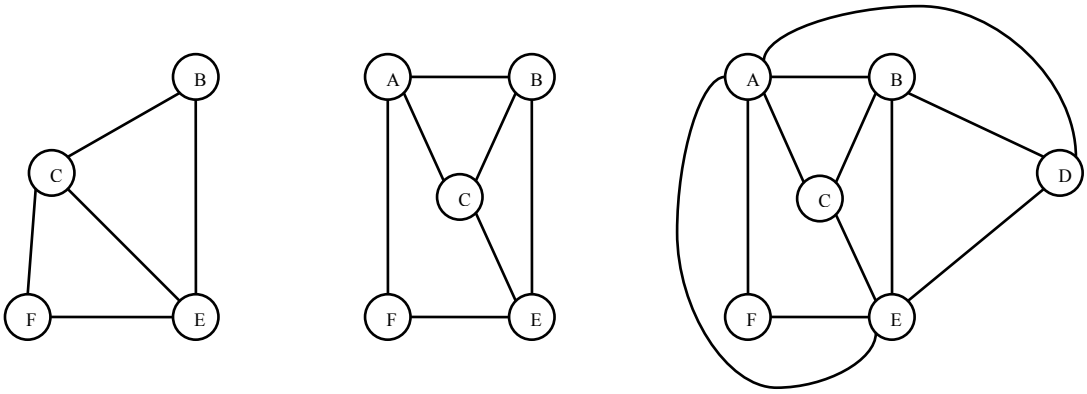
Tanım 2.1.9: Her bir düğümü arasında doğrudan bağlantı olan graflara **tamamlanmış graf (Complete Graph)** adı verilir. Tamamlanmış bir graf üzerindeki tüm düğümlerin dereceleri eşittir ve düğüm sayısından 1 eksiktir. Tamamlanmış bir grafın düğüm sayısı n ise bu graftaki her bir düğümün derecesi *n–1* olur. Kenarların 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 2.7'de tamamlanmış graf ve dereceleri görülmektedir.



Şekil 2.7: Tamamlanmış graf örnekleri

Tanım 2.1.10: Tüm düğümlerin derecelerinin eşit olduğu grafa **Düzenli Graf (Regular Graph)** denir; tamamlanmış graf aynı zamanda düzenli bir graftır. Fakat bunun tersi her zaman geçerli değildir. Yani, düzenli graf her zaman tamamlanmış graf değildir.

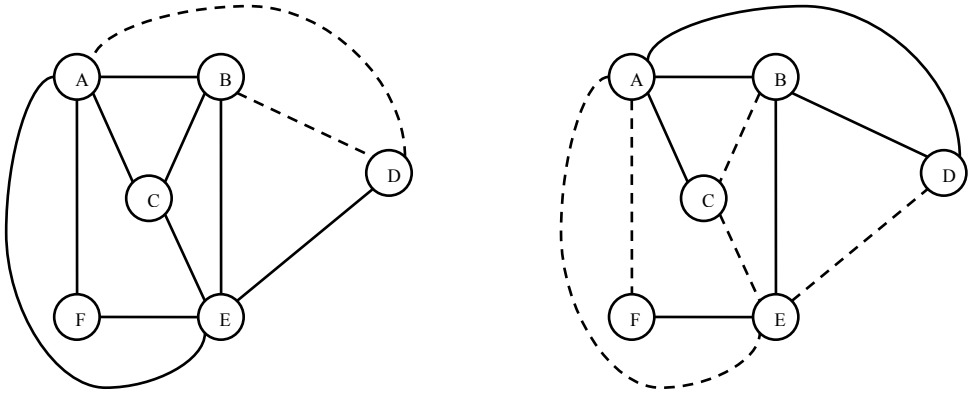
Tanım 2.1.11: Hiçbir kenarı birbirini kesmeyen grafa **Düzlemsel Graf (Planar Graph)** denir. Şekil 2.8'deki grafların tamamı düzlemsel özellik taşırlar. Düzlemsel graflar, kenarları (ayrıtları) kesişmeden herhangi bir düzleme rahatlıkla çizilebilirler. Bu yüzden harita graf ya da kabaca harita olarak da adlandırılırlar.



Şekil 2.8: Düzlemsel Graf örnekleri

Tanım 2.1.12: İki düğüm arasında doğrudan veya dolaylı olarak bir bağlantının ya da erişimin olmasına **yol (path)** denir. 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 ulaşılması mümkündür. Böyle bir durumda yol, geçilecek olan düğümlerin (d_0, d_2, d_5, $d_{1...}$) gibi listelenmesiyle gösterilebileceği gibi, takip edilen kenarların (d_0, d_2), (d_2, d_5), (d_5, d_1) yazılmasıyla da ifade edilebilir. Tüm birbirine komşu iki düğüm arasıdaki yolun uzunluğu 1'dir; n uzunluktaki yol, iki düğüm arasında n tane kenar geçilerek bağlantı yapılması anlamına gelir.

Tanım 2.1.13: Bir graf üzerinde, herhangi bir düğümden, diğer tüm düğümlere herhangi bir çevrim oluşturmadan gidilen yola **yol ağacı (spanning tree)** denir. Grafta birden fazla yol ağacı olabilir. Bir yol ağacı üzerinde kenarların sayısı, eğer grafta ayrık bir düğüm yoksa n-1'dir. Şekil 2.9'da A-B arasında iki kenarlı bir yol ve yol ağacı görülmektedir.



a) 2 Uzunlukta bir yol (Kesikli çizgiler) b) Tipik bir yol ağacı (Kesikli çizgiler)

Şekil 2.9: Bir graf üzerinde yol ve yol ağacı

Tanım 2.1.14: Başlangıç ve bitiş düğümü aynı olan kapalı yola **çevrim (cycle)** denir. Çevrim uzunluğu, kapalı yol üzerindeki kenarların sayısı kadardır. Kendi üzerinde, tek bir düğümü içine alan yolun uzunluğu 1 olur.

Tanım 2.1.15: Birbirine bitişik olan düğüm-kenar-düğüm sıralanmasına **dolaşı (walk)** denir. Diğer bir ifadeyle düğümler arasında kendilerine bağlı bulunan kenarlar üzerinden gidebilmektir. Eğer başlangıç düğümü ile bitiş düğümü aynı ise kapalı, değilse açık dolaşı yapılmış olunur. Dolaşının uzunluğu geçilen kenar sayısıdır.

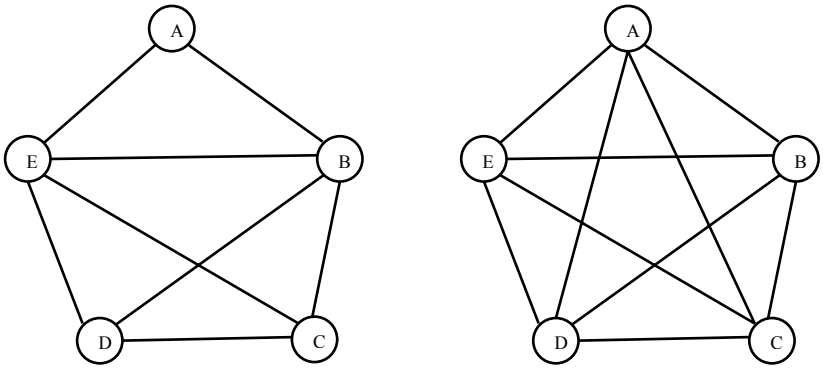
Tanım 2.1.16: Bir graf üzerinde geçilen kenarların tekrarlanmadığı bir dolaşıya **gezi (trail)** denir.

Tanım 2.1.17: Maliyeli bir grafta tüm düğümleri içine alan ve en az maliyetle yapılan kapalı bir dolaşının bulunması problemine **gezgin satıcı (traveling salesman) problemi** denir. Bu şekilde, tüm düğümlere uğranılır ve dolaşı bitip başlangıç noktasına geri dönüldüğünde diğer alternatiflere göre en az maliyetli yol gidilmiş olur. Bu tür problemler, dağıtım şirketleri vs. gibi uygulama alanlarında kullanılabilir.

Tanım 2.1.18: Her düğümden yalnızca bir kez geçilmesi esasına dayanan ve kapalı bir dolaşı gerçekleştirilebilen grafa **Hamilton Grafı (Hamilton's Graph)** denir. Hamilton Çevrim, graftaki her düğümü içine alan kapalı bir yolun olmasıdır. Gezgin satıcı

problemi en az maliyetli Hamilton çevrim bulunmasına dayanan bir problem türüdür. Şekil 2.10a'da örnek bir Hamilton graf gösterilmiştir.

Tanım 2.1.19: Graf üzerinde kenarların tekrarlanmadığı kapalı bir dolaşının yapılabileceği grafa **Euler Grafı (Euler's Graph)** denir. Bu tür graflar üzerinde bütün düğümlerin dereceleri çifttir. Euler çevrim, graf üzerinde her kenarı kapsayan, kapalı bir gezi yapılabilecek bir yolun bulunmasıdır. Şekil 2.10b'de bu grafa bir örnek verilmiştir.



a) Bir Hamilton grafı b) Bir Euler grafı

Şekil 2.10: Örnek Hamilton ve Euler grafları

2.2 Greedy Yaklaşımı / Yöntemi

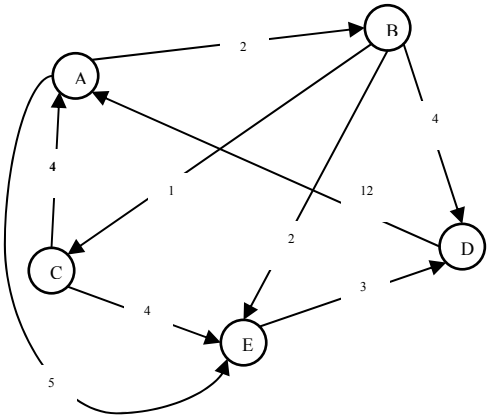
Greedy yaklaşımı, herhangi bir graf üzerinde belirli bir konuda en uygun sonucu veya en iyi sonucu bulmak amacıyla dolaşılırken bir sonraki düğümü belirlemek amacıyla kullanılan karar verme yöntemidir. Greedy yaklaşımında amaç, o andaki seçenekler arasından en iyi olanı seçmektir, bu seçimin ölçütleri 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

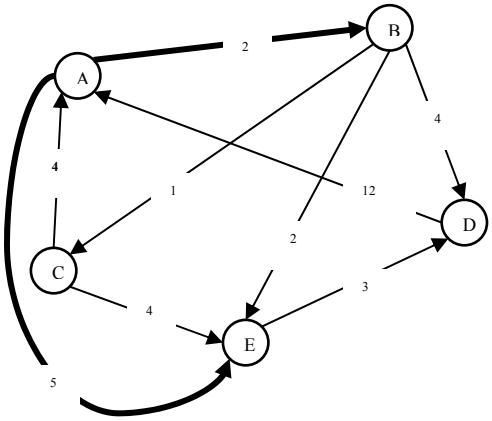
değil, genel bir değerlendirme ile belirlenir. Yani, dinamik programlamada, bir sonraki karar elemanı, programın sonuna kadar gidilip görülmesi ve geriye dönüldükten sonra karar verilmesi esasına dayanır. Bunun için, dinamik programlamaya dayalı bir çözümün olabilecek uygun çözüm olduğu söylenilebilir. Greedy yaklaşımını, yönlü-maliyetli graf üzerinde en kısa yol maliyetinin belirlenmesi problemi ele alınarak Şekil 2.11'de gösterilmiştir. Problem, belirli bir başlangıç noktasına göre diğer düğümlere erişmek için gereken en az maliyetli yolu bulan algoritmadır. Greedy yaklaşımında en önemli şey, o anda bulunulan düğümden bir sonraki düğüme geçişte yerel bir karar vererek gidilecek en iyi düğümü seçmektir. Greedy yaklaşımı birçok graf algoritmasında ilerleme seçim unsuru olarak kullanılır. Örneğin, eksi maliyeti olmayan maliyetli graflar üzerinde bir düğümden diğer en kısa yol maliyetlerini ve en kısa rotayı belirleyen Dijkstra algoritması Greedy yaklaşımına dayanmaktadır.

2.3 Graf Algoritmaları

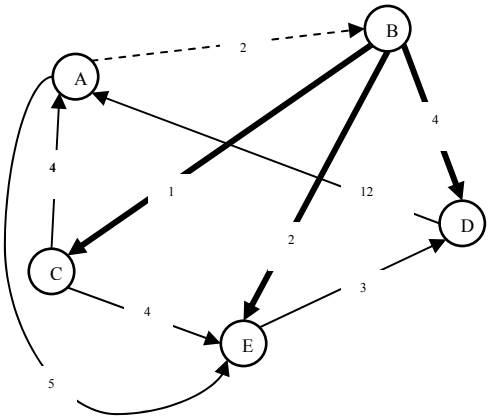
Graflar mühendislik, sosyal bilimler ve temel bilimlerde problemlere algoritmik bir çözüm sunar. Sistemin davranışının veya problemin tanımının düğümler ve düğümler arasındaki ilişkilerin kenarlarla yapılabildiği bu durumda graflarla ilgili önceden yapılan tanımlar, teoremler ve algoritmalar bu tür problemlerin çözümüne yardımcı birer araç olurlar (Çölkesen, 2002). Örneğin iki şehir arasındaki olası en kısa yolun bulunmasında graf algoritmaları kullanılarak algoritmik çözümler elde edilebilir.



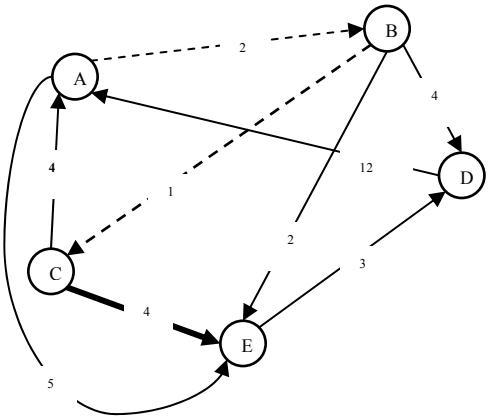
a) Örnek yönlü-maliyetli graf



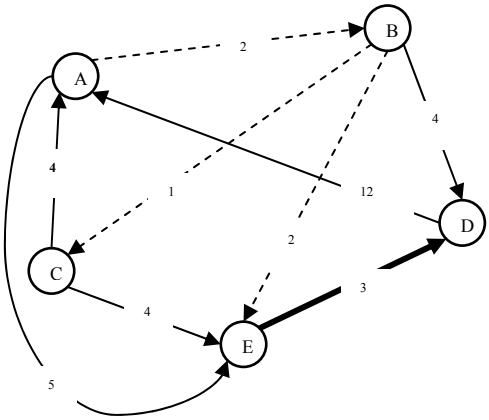
b) Başlangıç düğümü A



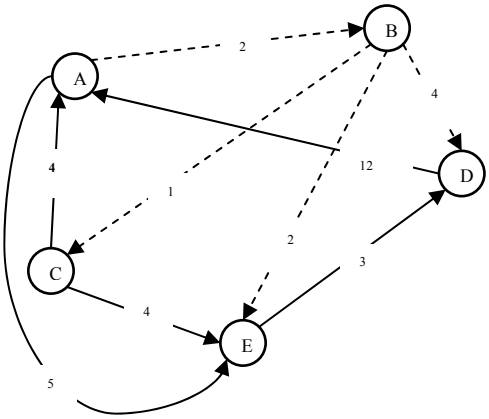
a) B düğümü için EKM=2



a) C düğümü için EKM=3



a) E düğümü için EKM=4

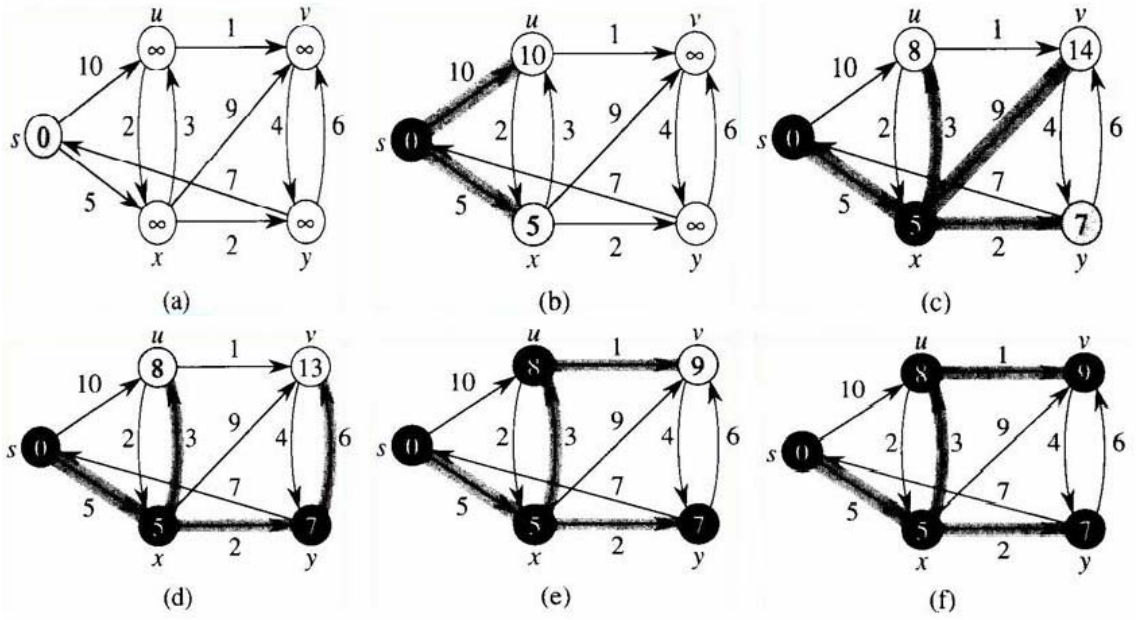


a) D düğümü için EKM=6

Şekil 2.11: Greedy yaklaşımına göre en kısa yol belirleme

2.3.1 Dijkstra Algoritması

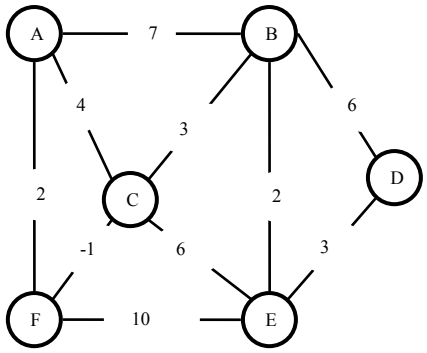
Dijkstra'nın algoritması belirli bir başlangıç noktasına göre en kısa yolu bulan bir algoritmadır; bir başlangıç düğümünden, diğer tüm düğümlere olan en kısa yolu belirler. Ağırlıklı ve yönlü graflar için geliştirilmiştir. Graf üzerindeki her bir kenarın ağırlığı sıfır veya sıfırdan büyük olmalıdır. Ancak burada da eksi maliyetli çevrim olmaması gerekir. Dijkstra'nın algoritması en kısa yolu belirlerken Greedy yaklaşımını kullanır. Her defasında, bir sonraki düğüme ilerleme Greedy yaklaşımına göre yapılır. Şekil 2.12'de 6-düğümlü örnek maliyetli graf üzerinde Dijkstra'nın algoritmasının davranış gösterilmiştir. Bu şekilde; örnek maliyetli bir grafın başlangıç durumu ve birinci, ikinci, üçüncü, dördüncü, beşinci ve son adımdaki davranışı görülmektedir. Başlangıç düğümü A olarak kabul edilen örnek maliyetli grafta ilk yapılan şey, A düğümünden doğrudan gidilebilecek düğümlere ait maliyet ve rota bilgilerinin güncellenmesi işidir. Her seferinde bir sonraki düğüme gitme işlemi Greedy yaklaşımına göre gerçekleştirilir.



Şekil 2.12: Dijkstra algoritmasının davranışı

2.3.2 Bellman ve Ford Algoritması

Bellman ve 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 kenar maliyetleri sıfır veya artı olan graflar için doğru sonuç verirken Dijkstra'dan farklı olarak Bellman ve Ford Algoritmasının kenar maliyetleri eksi olan graflar için de çalışıyor olmasıdır. Şekil 2.13'de Bellman ve Ford Algoritmasının çalışabileceği örnek bir graf gösterilmiştir.



Şekil 2.13: Bellman ve Ford Algoritmasının çalışabileceği bir graf

2.3.3 Floyd Algoritması

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 edilebilir. **Floyd Algoritmasının** yoğun graflarda kullanılması daha iyi bir seçimdir; çünkü çok seyrek graflarda Dijkstra algoritmasının düğüm sayısının katı kadar çalıştırılmasıyla daha iyi sonuçlar alınır. **Floyd Algoritmasına** göre grafın komşuluk matrisi, sonucun tutulacağı uzaklık matrisi ve en kısa yol bilgilerinin tutulacağı rota matrisi bulunmaktadır. Başlangıç düğümünün komşuluk matrislerindeki maliyetler başlangıç maliyetleri olarak kabul edilmiştir. Eğer düğümler arasında doğrudan bağlantı yok ise bu düğümler arasındaki maliyetler ∞ yapılır. Uzaklık matrisinde ise, başlangıçta hiçbir bilgi olmadığını göstermek amacıyla tüm bölmeler -1 ile doldurulur.

2.3.4 Kruskal Algoritması

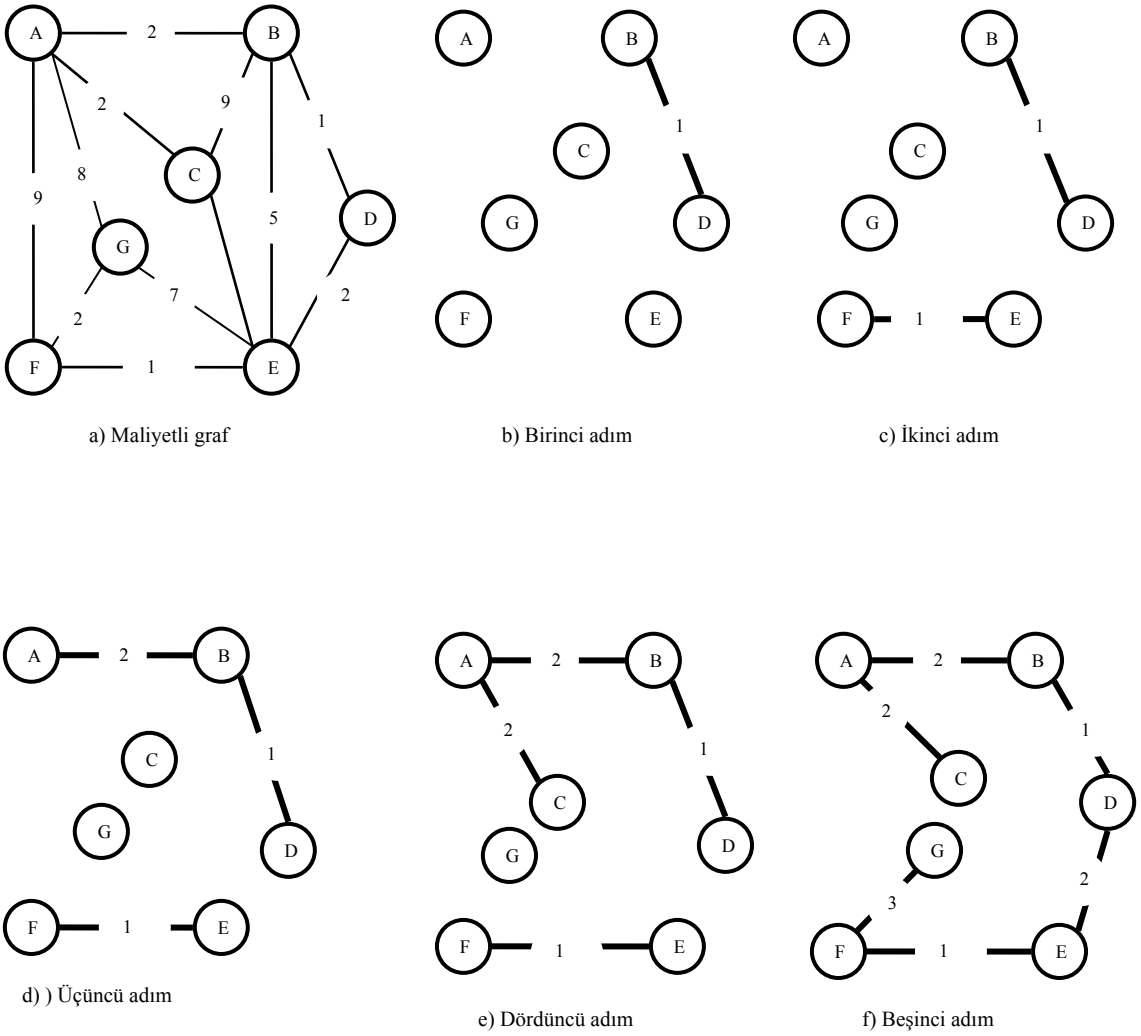
Kruskal Algoritması daha az maliyetli kenarları tek tek değerlendirerek yol ağacını bulmaya çalışan bir algoritmadır. En küçük yol ağacını belirlemek amacıyla kullanılır. Algoritmik ifadesi davranışsal olarak basit olmasına karşın, gerçeklenmesi için bazı yardımcı fonksiyonlara gereksinim duyar. Graf üzerindeki düğümler, aralarında bağlantı bulunmayan *N* tane bağımsız küme gibi düşünülür. Başka bir deyişle, *N* tane küme vardır ve her bir küme başlangıçta birer tane farklı düğüm içermektedir. Daha sonra bu kümeler maliyeti en az olan kenarlarla birleştirilerek düğümler arasında bağlantı olan tek bir küme oluşturulmaya çalışılır. Küme birleştirme işine en az maliyete sahip kenarlardan başlanır ve sonra kalan kenarlar arasından en az maliyetli olanlar seçilir.

Kruskal Algoritmasının davranışı Şekil 2.14'de 7 düğümlü bir graf üzerinde gösterilmiştir. Başlangıçta düğümler arasında herhangi bir bağlantı yok varsayılır ve sonra düğümler tek tek çevrim (cycle) oluşturmayacak şekilde birbirine bağlanır. Eğer bir düğümüm çevrim oluşturması söz konusu ise o düğüm atlanır. Kapsanmayan düğüm kalmayıncaya dek bu işlem yapılır. Yol ağacı bulunurken, yolun uzunluğu düğüm sayısından bir eksik (N–1) olur. Yani yol üzerindeki kenar sayısı düğüm sayısından her zaman 1 eksik olur.

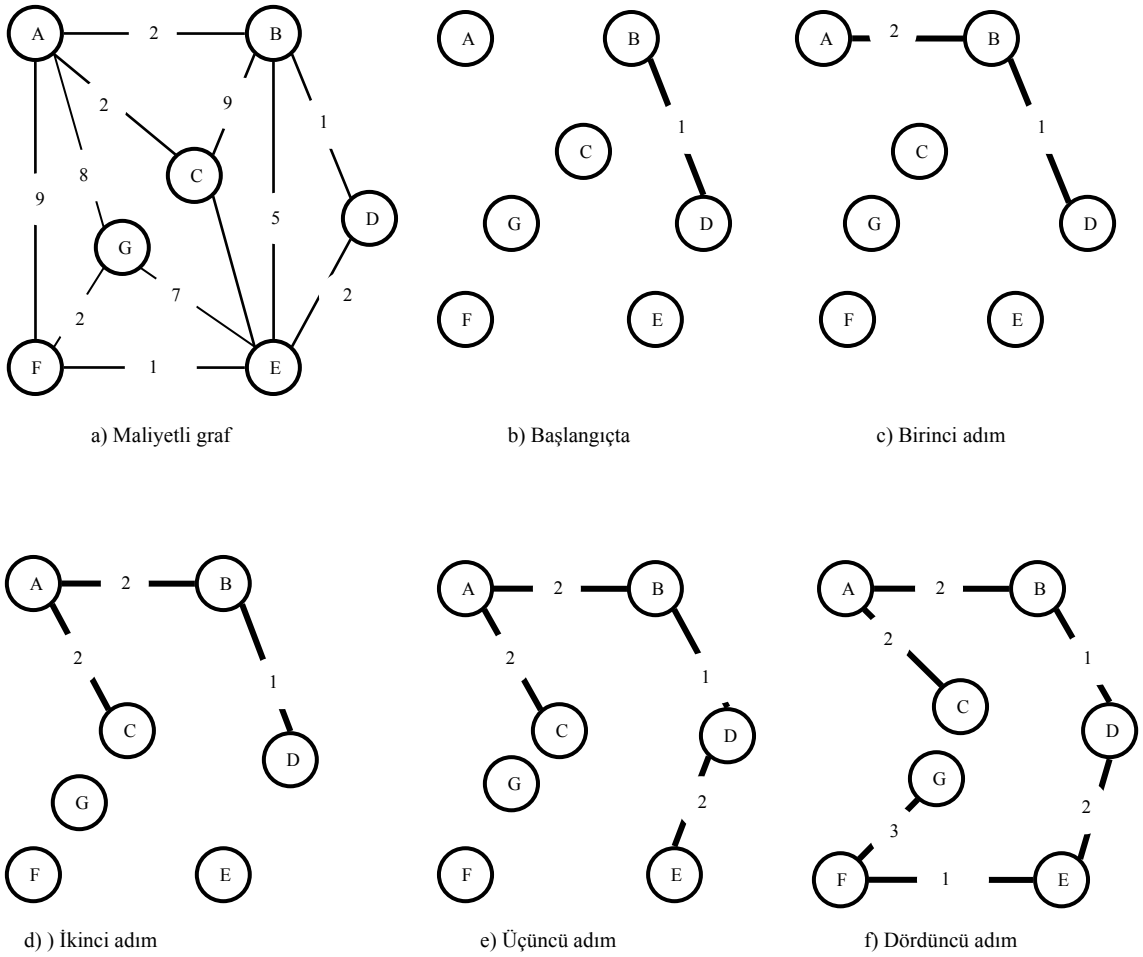
2.3.5 Prim Algoritması

Prim Algoritması en az maliyeli kenardan başlanarak, bu kenarın uçlarından en az maliyetle genişleyecek kenarın seçilmesi esasına dayanan bir algoritmadır. Tek kenarlık bir yol ağacından başlanarak adım adım diğer kenarların belirlenmesi esasına dayanır. Kruskal algoritmasındaki gibi, yol ağacının belirlenmesi aşamasında birden çok yol ağacı yoktur; sadece başlangıçta bir tane kenardan oluşan bir yol ağacı vardır ve sonraki adımlarda bu yol ağacına eklenmek için seçilen kenarın yol ağacı üzerinde bulunan bir düğüme komşu olması şarttır. Fakat seçilen kenarın her iki ucunun da yol ağacı üzerindeki birden fazla düğüme komşu olmaması gerekmektedir. Eğer iki uç birbirine komşu olursa çevrim oluşacağından ağaç tanımından çıkılır. Olması gereken yeni kenar

seçilirken Greedy yaklaşımına göre hareket edilmesidir. Şekil 2.15'de Prim algoritması yaklaşımı örnek bir graf üzerinde gösterilmiştir.



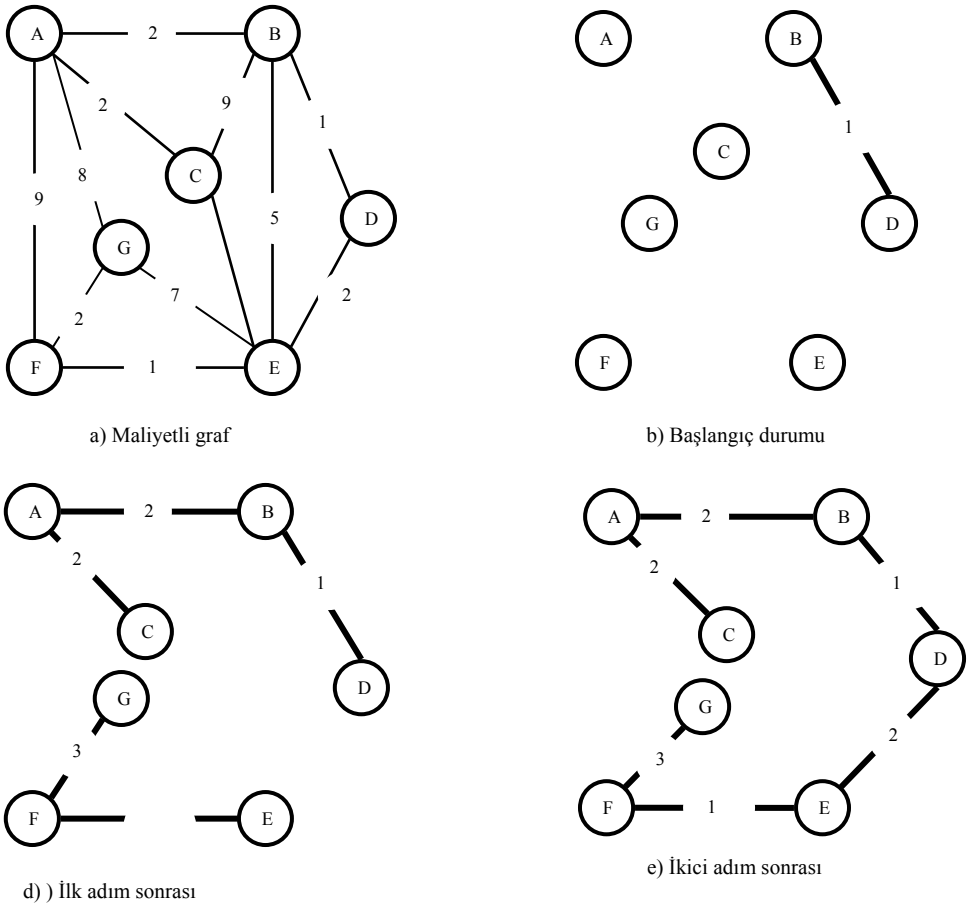
Şekil 2.14: Kruskal Algoritmasının davranışı



Şekil 2.15: Prim'in algoritmasının yaklaşımı

2.3.6 Sollin Algoritması

Sollin Algoritması aynı anda birden çok ağaçla başlayan ve sonraki adımlarda ağaçların birleşerek tek bir ağaca dönüştüğü bir algoritmadır. Bu algoritmada, bir adımda birden fazla kenar seçilir ve yol ağacı ara işlemlerde alt ağaç olarak bulunan ağaçlara eklenmek suretiyle bulunur. Yani yol ağacı belirlenene kadar ara işlemlerde birden fazla ağaç bulunur. Kruskal ve Prim'in algoritmalarına göre sonuca daha az adımda ulaşan Sollin algoritmasının ara işlemleri bu algoritmalara göre daha fazla olabilir. Şekil 2.16'da Sollin algoritmasının davranışı gösterilmiştir. Şekilde de görüleceği gibi örnek graf, Prim ve Kruskal algoritmalarında verilen örnek grafla aynıdır. Bu graf örnekleri dikkatle incelenirse Sollin algoritması ile sonuca daha az adımda ulaşıldığı açıkça görülecektir.



Şekil 2.16: Sollin algoritmasının davranışı

2.4 En Kısa Yol (Shortest Path) Problemi

İki düğüm arasında en az maliyetle gidilebilecek bir yolun belirlenmesi problemidir. Herhangi bir düğümden diğer tüm düğümlere ya da her bir düğümden diğer tüm düğümlere olan en kısa yolların belirlenmesi amacıyla birçok algoritma geliştirilmiştir. Bunların en çok kullanılanları,

- **Dijkstra** (Bir düğümden diğer tüm düğümlere olan en kısa yollar)
- **Bellman ve Ford** (eksi maliyetli graflarda bir düğümden diğerlerine en kısa yollar)
- **Floyd** (tüm düğümler için en kısa yollar)

Bu algoritmalardan bazıları bir düğümden diğerine olan en kısa yolu (single-sorce shortest path) bulurken, bazıları da her bir düğümden diğer tüm düğümlere olan en kısa yolları (all-pairs shortest path) bulur.

Bu algoritmaları şu şekilde sınıflandırabiliriz.

- Tek hedefe giden en kısa yollar (single-destination shortest path)
- Tek başlangıç düğümünden en kısa yollar (single-source shortest path)
- İki düğüm arasındaki en kısa yol (single-pair shortes path)
- Tüm düğümler arasındaki en kısa yollar (all-pairs shortest path)

Bir başlangıç noktasından tüm düğümlere olan en kısa yolları bulan algoritmalar, her defasında bir başlangıç noktası belirleyip N defa çalıştırılırsa genel algoritmalar gibi her bir düğümden diğer tüm düğümlere olan en kısa yolları bulmak amacıyla kullanılabilir. Fakat bu durum grafın yoğun olmasına göre toplam zaman maliyetinin artmasına sebep olabilir ve çoğu zaman da yürütülme süresi uzar.

2.5 En Küçük Yol Ağacı (Minimum Spanning Tree) Problemi

Yol ağacı, bir graf üzerinde tüm düğümleri içine alan ağaç şeklinde bir yoldur. Ağaç özelliğine sahip olduğundan çevrim içermez. Bir graf üzerinde birden çok yol ağacı olabilir. En az maliyetli olan yol ağacına en küçük yol ağacı (Minimum Spannin Tree) denir. Uygulamalar daha çok en küçük yol ağacının bulunması ile ilgilenir. Örnek bir graf için çeşitli yol ağaçları Şekil 2.6'daki gibidir. En küçük yol ağacının belirlenmesi amacıyla ilk akla gelen Kruskal, Prim ve Sollin algoritmaları geliştirilmiştir.

2.6 Graf Üzerinde Dolaşma Yöntemleri

Graf üzerinde dolaşma yöntemleri ve Greedy yaklaşımı, graf algoritmaları içinde önemli bir yer tutarlar; hem temel işlemleri tamamlarlar hem de diğer algoritmaların anlaşılması için teorik ve davranışsal model meydana getirirler. Bu sebepten dolayıdır ki bu iki yöntemin bilinmesi gerekmektedir. En kısa yol problemi düğümler arasındaki yollardan en kısa olanın bulunmasını gerçekleştirirken, yol ağacı algoritması tüm düğümleri içine alan yolun bulunmasını gerçekleştirmektedir. Gezgin satışçı problemi bir düğümden başlayarak diğer tüm düğümleri dolaştıktan sonra başlangıç noktasına gelme işinin en az maliyetle gerçekleşmesini amaçlamaktadır ve bunun üzerine yoğunlaşmıştır. Aynı şekilde şebeke problemlerinde de amaç kaynaklardan varış

noktasına en fazla akışın yapılmasıdır. Şebeke problemleri de bu tür sorunlar üzerine yoğunlaşmıştır.

Herhangi bir graf üzerinde dolaşma, grafın düğümleri ve kenarları üzerinde yapılması amaçlanan bir işi gerçekleştirecek ya da problemi çözecek şekilde hareket etmektir. Tüm düğümleri en az maliyetle dolaşma, iki düğüm arasındaki en kısa yolu bulma, tüm düğümler arasındaki en kısa yolları bulma, iki düğüm arasındaki en yüksek akış kapasitesini hesaplama, düğümler arasında *n* uzunlukta yol olup olmadığını belirleme, dolaşı yapma, gezi yapma ve bunlara benzer birçok işin yapılması graf üzerinde dolaşma konusuna girmektedir. Graf üzerinde çözülmesi gereken problemler şu başlıklar altında sıralanabilir.

- En az maliyetin bulunması
- Yol ağacının bulunması
- En fazla akışın bulunması
- Düğüm ya da bölge renklendirme
- Kümeleme yöntemiyle ayrıştırma
- *n* uzunlukta bağlantılılık belirlenmesi

Mühendislikte, sosyal bilimlerde ve temel bilimlerde çoğu problemler graf veri modeline yakınlaştırılarak çözülebilir. Bu çözümler genel olarak graflar üzerinde dolaşma algoritmaları kullanılarak gerçekleştirilir. Bu sebepledir ki graf üzerinde dolaşma algoritmaları bilgisayar biliminde önemli bir yere sahiptir.

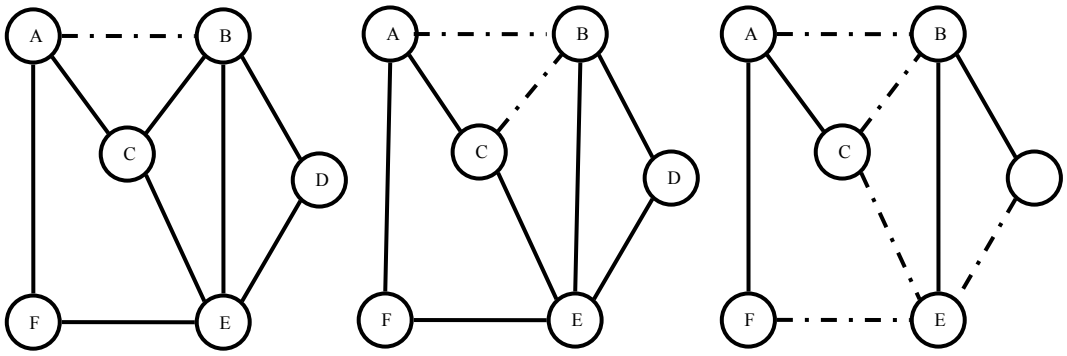
Graf üzerinde dolaşma yapan birçok yaklaşım yöntemi bulunmaktadır; bunlardan en çok kullanılan iki yöntem DFS (Depth First Search) ve BFS (Breadth First Search) olarak adlandırılmıştır ve graflar üzerine geliştirilen algoritmaların büyük bir kısmı bu yaklaşım yöntemine dayanmaktadır denilebilir.

2.6.1 Önce Derinlik Araması (Depth-First Search- DFS) Yöntemi

Önce derinlik araması olarak adlandırılan bu yöntem, graflar üzerinde dolaşma yöntemlerinden birisidir. Bu yöntem, graf üzerinde dolaşmaya başlangıç düğümünün bir

kenarından başlayarak o kenar üzerinde gidilebilecek en uzak düğüme kadar devam ettirir. Ara düğümlerde de benzer davranış söz konusudur, o düğümün bir kenarından gidilebilecek en uzak düğüme kadar gidilir; gidilemeyen düğüm varsa o düğümün başka bir kenarından denenir. Şekil 2.17'de DFS yönteminin çalışma şekli gösterilmiştir.

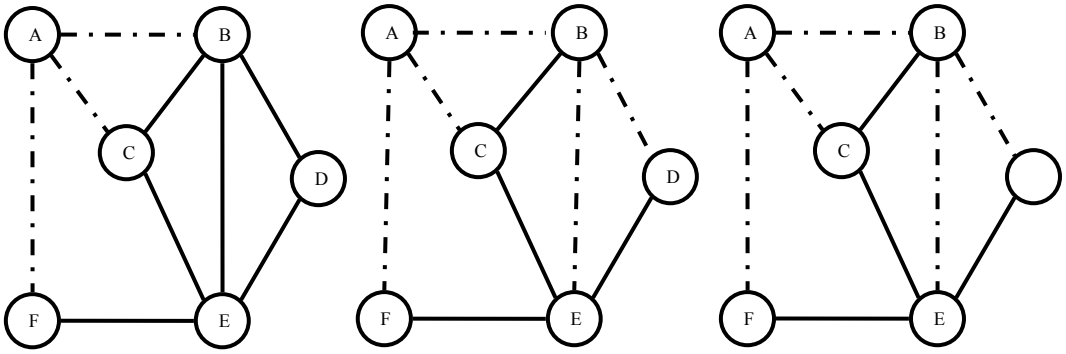
Graf üzerinde dolaşmaya A'dan başlandığı varsayılırsa A ile bağlantısı olan B'ye sonra B ile bağlantısı olan C'ye ve sonra da sıra ile E, D ve F'ye gidilir. Kullanılan algoritma DFS olduğundan, bir düğümden kendisine bağlı birçok düğümden herhangi birine gidilir, oradan da bağlı olan düğümlerden birine gidilir. Bu şekilde ilerlerken gidilmeyen düğümler olursa tekrarlanan program davranışında olduğu gibi geri dönülür ve o düğümden, sıradaki bir başka düğüme gidilir.



Şekil 2.17: DFS yönteminin çalışması

2.6.2 Önce Genişlik Araması (Breadth-First Search-BFS) Yöntemi

Önce genişlik araması olarak adlandırılan BFS, başlangıç düğümünden gidilebilecek tüm düğümlere gidilmesiyle yapılan bir dolaşı yöntemidir. DFS'den en önemli farkı da budur. DFS' de graf üzerinde dolaşmaya başlangıç düğümünün bir kenarından en uzağa gidilmesiyle başlanırken, BFS' de başlangıç düğümünden gidilebilecek tüm komşu düğümlere gidilmesiyle başlanır. Ara düğümlerde de başlangıç düğümü gibi hareket edilir. Bir ara düğüme varıldığında, o düğüme komşu ve daha önce gidilmemiş olan tüm düğümlere gidilir. Bu yöntem örnek bir graf üzerinde Şekil 2.18'de gösterilmiştir.



Şekil 2.18: BFS yönteminin çalışma şekli

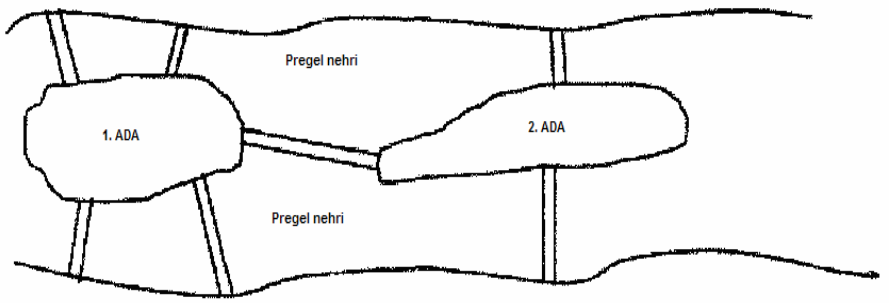
BFS yöntemine göre hareket ederek herhangi bir graf üzerinde dolaşma, graf üzerinde dolaşarak işlem yapan birçok algoritmanın ortaya çıkmasına sebep olmuştur demek yanlış olmaz. Örneğin, herhangi bir grafta eğer kenar maliyetleri yok ise veya eşitse, BFS yöntemi en kısa yol algoritması gibi bir şekil alır ve bu durumda bir düğümden her bir düğüme olan en kısa yolları bulur denilebilir.

2.7 Tur Belirleme Problemleri

Tur belirleme problemlerini çeşitli sınıflandırmalar altında incelemek mümkündür. Fakat genel olarak rota belirleme problemlerini iki başlık altında incelenmektedir.

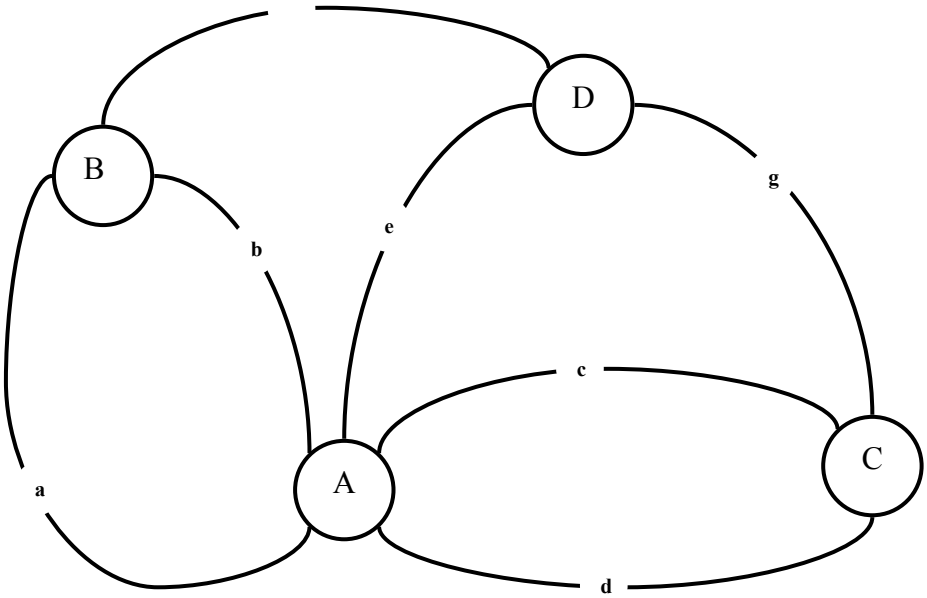
1. Hamilton Turlu Problemler (Düğümler için tur belirleme)
2. Euler Turlu Problemler (Kenarlar için tur belirleme)

Euler turu, şebeke kuramının kurucusu kabul edilen İsviçe'li matematikçi Leonhard Euler (1707–1783) tarafından Königsberg köprüleri için tasarlanan bir problemin çözümü için tasarlanmıştır (Minieka, 1978). Königsberg köprüleri problemi, eski adıyla Königsberg bugünkü adıyla Kaliningrad olan şehrin halkı tarafından, kentin içinden geçen Pregel nehri üzerinde bulunan 7 köprüden geçiş yapılması ile ilgili Pazar eğlencesi olarak tanımlanmış bir oyundur. Oyundaki temel soru, Her köprüden sadece bir kez geçmek şartıyla bir yerden başlayıp tekrar aynı yere dönülmesini sağlayacak bir yol var mıdır sorusudur. Şekil 2.19, bu probleme konu olan ve her iki adayı birbirine bağlayan nehirlerin üzerindeki yedi adet köprü görülmektedir.



Şekil 2.19: Konigsberg Köprülerinin şematik gösterilimi

Euler, bu problemi inceleyerek ilk defa bir problemin düğüm ve kenarlardan oluşan bir şebeke olduğu tanımlamasını yapmış her köprüyü birer kenar ve her düğümü de köprülerin bağladığı bölgeler olarak tanımlamıştır (Şekil 2.20). 1736 yılındaki bir çalışmasıyla da "Königsberg Köprüleri Problemi" ifadesini kullanarak, problemin tanımlanan şekliyle bir çözümünün olamayacağını ispatlamıştır. Bu ispatı yapabilmek için de bugün Euler turu olarak bilinen tur tanımını yapmıştır. Bu tanıma göre Euler turu, bir başlangıç noktasından yola çıkarak graftaki bütün kenarlardan yalnız bir kez geçip yine başlangıca dönen yoldur. Her grafta Euler tur olması şart değildir. Euler, bunun şartlarını da incelemiş ve eğer graftaki bütün düğümlerin dereceleri çift sayı ise Euler turunun olacağını, aksi halde olamayacağını göstermiştir.



Şekil 2.20: Königsberg Köprüleri Probleminin grafa uyarlanması

Hamilton turu ise, İrlandalı matematikçi William Rowan Hamilton (1805–1865) 1859 yılında tanımlanan bir matematiksel problem sonucu ortaya çıkmıştır (Gondran ve Minoux, 1984). Problem şöyle tanımlanmaktadır: Dünya yüzeyine dağılmış 20 tane şehir seçilerek bu şehirlere bir gezi düzenlenecektir. Amaç, her şehre yalnız bir kez uğramak ve geziye başlanan şehre geri dönmektir. Bu şartlar altında amaca uygun olarak hangi yollar izlenmelidir ve bu yollar nasıl belirlenmelidir?

Hamilton, problemi daha basit hale getirmek için dünyayı düzgün bir 12 yüzlü olarak kabul edip şehirlerin de bu şeklin köşelerinde olduğunun varsayılabileceğini söylemiştir. Böylece köşeleri birleştiren kenar çizgileri de şehirleri birleştiren yolları temsil edecektir. Düzgün 12 yüzlü, 12 tane eşit alanlı beşgenden oluşan küre benzeri hacimsel şekil olduğundan 20 tane köşe noktası ve 30 tane de beşgenlerin birleştiği kenar çizgisi vardır. Bu varsayımlar altında 12 yüzlü 2 boyutlu uzayda köşelerin düğümleri, kenarların ise kenarları temsil ettiği 20 düğümlü, 30 kenarlı bir ağ oluşmuş olur. Hamilton, problemi tanımladıktan sonra, düzgün 12 yüzlüye düzlemde karşılık gelen graf üzerinde bir çözüm bulmuş ve böylece Hamilton turu kavramı ortaya çıkmıştır. Buna göre Hamilton turu, bir başlangıçtan yola çıkıp graf üzerinde bütün düğümlerden yalnız bir kez geçerek başlangıç noktasına dönen yoldur. Aslında Hamilton turu ilk defa 1759'da Euler, 1771'de Vandermonde tarafından, satrançtaki atın bütün kareleri gezerek başladığı kareye geri gelecek şekilde nasıl bir yol izlemesi gerektiğini bulmak için yapılan çalışmalar sonucunda kısmen ortaya konmuştur. Ancak problemi açıkça tanımlayan kişi Hamilton olmuştur (Gondran ve Minoux, 1984).

Hamilton turu ile Euler turu arasındaki en önemli fark, problemin tanımlandığı ağlardır. Hamilton turu; uğranacak yerlerin düğüm, düğümleri birbirine bağlayan yolların da kenar olarak tanımlandığı ağlarda kullanılırken; Euler turu ise; kavşakların düğüm, uğranılması düşünülen yerlerin ise kenarlar üzerinde tanımlandığı graflarda kullanılır (Sipahioğlu, 1996). Diğer bir ifadeyle, Euler turunda uğranacak yerler kenarlar üzerinde tanımlanır ve istenilen yere uğrayabilmek için graftaki her kenardan mutlaka ve yalnızca bir kez geçilmesi istenir. Hamilton turunda ise, uğranılması düşünülen yerler düğümler düğümlerde tanımlanır ve graftaki her düğümden mutlaka ve yalnızca bir kez geçilmesi istenir. Bu sebeple, Euler turlu problemlere, kenarları

gezecek gezgin için tur belirleme problemi; Hamilton turlu problemlere de, düğümleri gezecek gezgin için tur belirleme problemi demek doğru bir ifade olur.

2.8 Hamilton Turlu Problemler (Node Routing Problems)

Bu tür problemlerde amaç, graf üzerindeki tüm düğümlere en düşük maliyetle hizmet götürmeyi sağlayan turu bulmaktır. Bu kategoride iki tür problem vardır. Birincisi, kapasite kısıtlaması olmayan Gezgin Satıcı Problemi (Travelling Salesman Problem-GSP), diğeri de kapasite kısıtlaması olan Araç Rotalama Problemidir (ARP-Vehicle Routing Problem). GSP, yöneylem araştırmalarının fazla ilgi görmüş problemidir ve bu ismin ilk kez kimin kullandığı bilinmemektedir. Bu konuya ilgi Euler ile başlamış olup, birçok ilim adamı tarafından da incelenmiştir. Fakat problemle ilgili ilk ciddi çalışma 1954 yılında Dantzig, Fulkerson ve Johnson tarafından yayınlanan "Solution of a large scale travelling salesman problem" isimli çalışmadır.

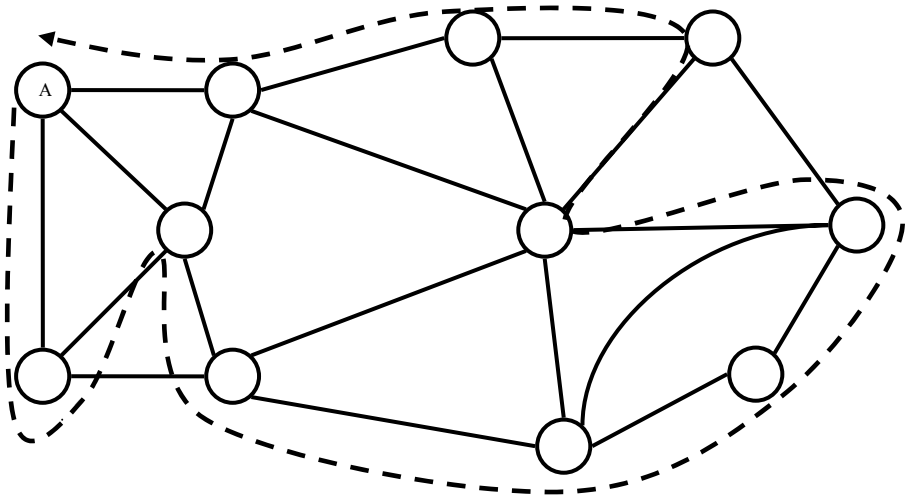
Hamilton türlü problemlerin tamamı aslında GSP'den türemiş problemlerdir. M-Gezgin Satıcı Problemi ve Araç Turu Belirleme Problemi (ATBP) üzerinde en çok çalışılmış diğer Hamilton turlu problemlerdir.

Hamilton turlu problem türlerinde amaç, graf üzerindeki tüm düğümlere en az maliyetle hizmet götürmeyi sağlayan turu bulmaktır. Bu kategoride değerlendirilecek ve en çok bilinen iki tür problem vardır. Bunlardan ilki, kapasite kısıtlaması olmayan Gezgin Satıcı Problemi (Travelling Salesman Problem-GSP), diğeri de kapasite kısıtlaması olan Araç Rotalama Problemidir (ARP-Vehicle Routing Problem)

2.8.1 Gezgin Satıcı Problemi (GSP)

Kapasite kısıtlamasının olmadığı tek bir araçlık klasik Hamilton turlu bir problem Gezgin Satıcı Problemidir (GSP). Birçok metot, araç rotalama problemini çözmek için alt çözüm yöntemi olarak GSP'yi kullanır. Buradaki metotlardan kasıt, daha önceki bölümlerde anlatılan kısa yol algoritmalarında kullanılan yöntemlerdir. Bu problem, dolaşılacak yerlerin düğüm ve aradaki yolların kenar olarak gösterildiği uygulamalarda,

düğüm tekrarı yapılmadan, tüm düğümlerin en az maliyetle dolaşılıp başlanan yere geri dönülmesi ihtiyacından ortaya çıkmıştır. GSP'de amaç, gerçekleştirilecek olan herhangi bir hizmet için en az maliyetli düğüm ziyaret sırası belirlemeye çalışmak ve gidilen yol ya da toplam maliyetin en aza indirilmesidir. Şekil 2.21'de GSP'nin grafiksel gösterimi görülmektedir.



Şekil 2.21: Gezgin Satıcı Problemin grafiksel gösterimi

Geleneksel GSP'de, gezginin uyması gereken kapasite, zaman vb. gibi sınırlama söz konusu değildir. Bu tür bir ek kısıtlamanın olması, problemin farklı isimli problemler olarak anılmasına sebep olur. Örneğin zaman kısıtlaması varsa, zaman tabanlı GSP adını alır.

2.8.2 Araç Rotalama Problemi (ARP)

Belirli kapasitelere sahip bir veya daha fazla aracın, bir veya daha fazla sayıda depodan talep miktarı belli olan bir veya daha fazla sayıda müşteriye hizmet vermek için en uygun güzergâhları kullanacakları biçimde yönlendirilmeleridir. Bir şebeke içinde, depolar araçların bulunduğu ve taleplerin karşılanmak için kullanıldığı yerlerdir. Depolar dışındaki düğümler hizmet götürülecek müşterileri temsil etmektedir. Problemin şekline göre müşteri ve hizmet kavramı farklı ifadeler alabilir. Örneğin mutfak tüpü dağıtımı yapan bir tüp dağıtım probleminde müşteri, tüp talebinde bulunan

kişinin bulunduğu düğüm veya düğüme en yakın nokta, hizmet ise tüpün bu noktaya ulaştırılması işidir.

ARP'yi, sadece ürünlerin müşterilere dağıtıldığı bir yapı olarak düşünmemek gerekir. Ayrıca dağıtım işinin tek bir araçla yapılıyor olması da şart değildir. Birden fazla araç ta kullanılabilir. Araçlar için belirlenen kapasite değerleri farklı anlamlar da taşıyor olabilir. ARP; araçların fiziksel olarak yerleri bilinen noktalara (müşterilere), tanımlanan araç kapasitesini aşmayacak biçimde ve belirlenen amaçlar doğrultusunda, hangi sırayla uğramaları gerektiğinin belirlenmesi problemidir.

Bu tür bir problemde her bir aracın belirli bir kapasitesinin olduğunu kabul edersek, herhangi bir düğümden gelen belli miktardaki talebin ve her bir kenarın ağırlık olarak bir değere sahip olması gerekir. Gelen talep miktarı araç kapasitesinden az ya da fazla olabilir. Talep miktarı araç kapasitesinden az olursa tek araç kullanılarak, fazla olursa birden fazla araç kullanılarak talep karşılanabilir. Sonuç olarak bu koşullar altında ARP, her müşteriye her bir aracın en az maliyetle hizmet vermesini sağlayan turun belirlenmesidir.

ARP ilk kez Dantzig ve Ramser tarafından 1959 yılında gerçekleştirilen "The truck dispatching problem" adlı çalışma ile tanımlanmış ve bir tamsayılı programlama modeli ile çözüm önerilmiştir. Problem 1995 yılında Canen ve Scott tarafından, kamyonla dağıtım, tedarik problemi ve kapasiteli tur belirleme problemi gibi farklı isimlerle de anılmıştır.

ARP ve GSP tipik "NP-Complete" problemleridir. Bir problemin "NP-Complete" olması demek, bu problemin boyutunun (kenar ve düğüm sayısının) artması ile o probleme ait en uygun sonucu bulmanın zor olacağı ve güvenilir sonuçlara ulaşmanın geç olması anlamına gelir. Bu sebeple bu tür problemlerin çözümünde en uygun sonucun bulunmasını garanti etmeyen ancak, çözüm zamanı çözüm bölgesi içinde kalan sezgisel yöntemlerden faydalanılır. Son on yıldan beri araştırmacılar "NP-Complete" problemlere ve çözüm yöntemlerine ilgi göstermişlerdir. ARP ve GSP problemleri

birbirleriyle ilişkilidir ve birini çözmek için geliştirilen algoritma, diğer problemlere de uygulanmaktadır.

Literatürde GSP ve ARP dışında Hamilton turlu problemler olarak şu problemleri de saymak mümkündür:

- Zaman Tabanlı Araç Turu Belirleme Problemi (Vehicle Routing Problem With Time Windows)
- Kısmi Taşımalı Araç Turu Belirleme Problemi (Split Delivery Vehicle Routing Problem)
- Geri Taşımalı Araç Turu Belirleme Problemi(Vehicle Routing Problem with Backhauling)
- Rasgele Araç Turu Belirleme Problemi (Stochastic Vehicle Routing Problem)
- Dönemlik Araç Turu Belirleme Problemi (Period Vehicle Rouing Problem)
- Stoğa Göre Araç Turu Belirleme Problemi (İnventory Routing Problem)
- Gerçek Zamanlı Ürün Dağıtım Problemi (Delivery Dispatching Problem) (Sipahioğlu, 1996).

2.9 Euler Turlu Problemler (Arc Routing Problem)

Euler turu, 1962 yılında Mei-Ko Kwan tarafından tanımlanan Çinli Postacı Probleminin temelini oluşturmaktadır (Bodin ve diğ., 1983).

Oluşturulan bir grafta Euler tur yapılabiliyorsa, tüm yollardan bir kez geçilmesi yeterli olacağından, gidilecek toplam yol, kenarların toplamı olur. Bazı durumlarda bütün yerlere uğrayabilmek için bazı kenarlardan birkaç kez geçilmesi gerekebilir. Bu durumda, olabilecek en kısa yolu kat ederek takip edilecek yolun bulunması problemi ortaya çıkar.

2.9.1 Çinli Postacı Problemi (Chinese Postman Problem)

Çinli Postacı Problemi (ÇPP), ilk olarak 1962 yılında Guan tarafından, bir postacının postaneye dönmeden önce görevlendirildiği bölgede uğraması gereken her yere uğrayacağı en kısa yürüme yolunu bulmak için tanımlanmıştır.

Postacı probleminde amaç, bir postacının bir merkezden başlayarak gidilmesi düşünülen tüm sokakları (kenar) dolaşıp bütün postaları dağıttıktan sonra tekrar başladığı yere yani merkeze dönmesi için izlemesi gereken en kısa turun belirlenmesidir. Bu problemin gerçeklenmesi için, uygulamanın yapılacağı bölgedeki tüm kavşakların birer düğüm, takip edilecek yolların birer kenar olarak belirtildiği bir graf çizilir. Postacının bir yoldan (kenar) geçmesi, o yol üzerindeki tüm yerlere uğraması anlamına gelmektedir.

Genel olarak, postacı problemi, bir yol boyunca arka arkaya dizilmiş noktalara hizmet veren servis araçlarının karşılaştığı en temel problemlerden biridir. Sonradan bu probleme bazı eklemeler yapılarak kapasiteli postacı problemi, kırsal alan postacı problemi, yönlü postacı problemi, hiyerarşik postacı problemi diye türevleri ortaya çıkmış ve değişik çözüm yaklaşımları önerilmiştir (Eiselt ve diğ., 1955).

ÇPP'nin iki farklı probleminin tanımı şu şekildedir. Birincisi, yönsüz bir grafta, yoldaki gidiş yönüne bağlı olarak maliyetlerin değiştiği Rüzgârlı Postacı Problemi; diğeri ise, grafta ziyaret edilmesi planlanan kenarların öncelik sırasına göre ziyaret edildiği Hiyerarşik Postacı Problemidir.

2.9.2 Kapasiteli Postacı Problemi (Capacitated Postman Problem)

Kapasiteli Postacı Problemi (KPP), Araç Rotalama Problemi (ARP) gibi düşünülebilen ve gerçek hayatta daha fazla uygulama alanı bulan bir rotalama problemidir. Problem, kenarlardan gelen belirli miktardaki herhangi bir talebin, belli bir kapasiteyi aşmadan karşılanmasını sağlayan en az maliyetli tur olarak tanımlanmaktadır.

2.9.3 Kırsal Alan Postacı Problemi (Rural Postman Problem)

ÇPP probleminde olduğu gibi, Kırsal Alan Postacı Problemi (KPP) de, kırsal alanda yapılan en az maliyetli posta dağıtım işlemidir. Bir sokakta postacının ziyaret edeceği yerler vardır. Fakat bu sokaklardan bazıları ziyaret zorunluluğu olmayan fakat diğer

bazı yollara geçişte kullanılan sokaklardır. Problemin amacı, bu kenarları ziyaret eden en az maliyetli rotayı bulmaktır.

2.9.4 Genel Araç Rotalama Problemi (GARP)

Genel Araç Rotalama Problemi (GARP), ARP ve Euler turlu problemin genişletilmiş şeklidir. Problem, düğümlerin N, kenarların A ve maliyetlerin M olduğu bir G(N, A, M) şebekesinde i ε N düğümleri R $\subseteq$ A kenarlarını en az maliyetle ziyaret eden turu bulmayı amaçlar. Bu tanım, Euler turlu problem (ETP) ve Hamilton turlu problemlerin (HTP), GARP'IN özel durumları olduğunu ortaya koyar. Bu özel durumlar;

- Eğer i = ∅ ve R $\subset$ A ise, GARP, Kırsal Alan Postacı Problemidir. (∅:boş küme)
- Eğer i = ∅ ve R=A ise, GARP, Çinli postacı problemidir.
- Eğer i = N ve R = ∅ ise, GARP, Araç Rotalama Problemi veya Kapasiteli Gezgin Satıcı Problemi'dir.

Bunların dışında, kapasite kısıtlamasının bulunduğu Genel kapasiteli Rotalama isminde ARP ve KARP'ın genel hali olan farklı bir problem sınıfı da vardır (Slavenko, 2001).

2.10 ARP'nin ETP'ye ve ETP'nin ARP'ye Dönüştürülmesi

Golden ve Wong (1981) Araç Rotalama Problemini, kapasiteli kenarları gezecek tur problemine dönüştürmek için, bir kenarı iki düğümle iki kenara bölmüşler ve bu kenarları daha önceki orijinal düğüm talebini yazarak ağırlıklandırmışlardır.

Kapasiteli Kenarları Gezecek Tur Belirleme Problemini, Kapasiteli Düğümleri Gezecek Tur Problemine dönüştürülmesi ilk defa Peran ve diğ. (1987) tarafından önerilmiştir. Bu dönüştürmeyi, her bir kenarı üç eşit parçaya bölüp, üç adet düğümle birleştirerek gerçekleştirmişlerdir. Buna göre bir düğümün talebi, daha önceki kenarın pozitif talebinin üçte biri kadardır. Bu yaklaşımda, düğüm çiftleri arasındaki mesafeler, ARP'de tek bir araç tarafından ziyaret edilen bir düğümün kenar ziyaretinde de tek bir

araç tarafından ziyaretini sağlamak ve üç kenarın aynı rota üzerinde art arda gelmesini garanti edecek biçimde tanımlanmıştır. Laporte (1997), Kırsal Alan Postacı Problemi gibi, birçok Kenar Rotalama Problemini Araç Rotalama Problemi'ne dönüştürmüştür.

2.11 Bölüm Özeti

Bu bölümde graf teorisi ile ilgili kavramlar üzerinde durulmuş ve bazı tanımlamalar ile bunları açıklanmaya çalışılmıştır. ARP probleminde en kısa yolun bulunması için kullanılan algoritmalar ve davranışları irdelenmiştir. Graf veri modeline uyarlanabilecek farklı türlerdeki problemler hakkında bilgi verilmiştir. Graflardaki dolaşma yöntemleri anlatılmıştır.

Bir sonraki bölümde ARP ile ilgili literatürdeki çalışmalar hakkında bilgi verilecek ve problemin çözümü için yapılan bilimsel çalışmalar irdelenecektir.

ÜÇÜNCÜ BÖLÜM
ARAÇ ROTALAMA PROBLEMİ

3. ARAÇ ROTALAMA PROBLEMİ

Bu tez çalışmasında, Araç Rotalama Problemi (ARP) ve türevleri incelenmiştir. ARP problemi, Graf Teorisinde geçen en kısa yol algoritmalarından olan Dijkstra algoritması kullanılarak C# programlama dilinde kodlanan simülasyon ortamında ele alınmıştır. Dağıtım veya pazarlama ağına sahip şirketler için araçların en uygun güzergâhları kullanacak biçimde yönlendirilmeleri, şehir içi veya şehir dışı müşteri taleplerini karşılamada nakliyat maliyetlerinin azaltılması ya da en aza indirilmesi noktasında çok önemlidir.

Bir şirketin başarısı hizmet verdiği müşteri sayısındaki değişim ile yakından ilişkilidir. Eğer zaman içerisinde müşteri kaybediyorsa, müşteri memnuniyetsizliği söz konusudur. Bir müşteri başka özel sebepleri yoksa çalışmakta olduğu bir şirketi, ya taleplerinin zamanında karşılanmamasından ya da şirket ürünlerinin aynı işi yapan diğer şirketlere göre daha yüksek fiyatlı olmasından dolayı bırakır. Bu yüzden nakliyat maliyetleri müşteri memnuniyeti açısından bir şirket için önemli bir giderdir. Bu giderlerin azaltılması, müşteri taleplerini karşılamak için şirketin kullandığı belirli kapasite ve hıza sahip araçların en uygun güzergâhları kullanacak şekilde müşteri talep noktalarına gönderilmesi ile sağlanabilir. En uygun güzergah belirli başarım gereksinimlerini optimum şekilde sağlayan güzergahtır. Örneğin, başarım sadece harcanan benzin olarak düşünülürse en uygun güzergâh, aracın en az yol alarak talep noktasına ulaştığı yoldur.

ARP'nin çözümü için bu güne kadar farklı yöntemler önerilmiş ve bir şekilde nakliyat maliyetlerinin azaltılması için en uygun varyasyonlar bulunmaya çalışılmıştır. Farklı yaklaşımlar ARP'nin farklı türevlerinin ortaya çıkmasına neden olmuştur. Bu türevleri;

- Sınırlı kapasiteye sahip araçların kullanıldığı Kapasiteli Araç Rotalama Problemi (KARP)
- Müşteri taleplerinin birden çok depo kullanılarak karşılanması sonucu ortaya çıkan Çoklu Depolu Araç Rotalama Problemi (ÇDARP)

- Müşterilerin yalnız belirli zaman dilimlerinde servis kabul ettiği Zaman Kısıtlamalı Araç Rotalama Problemi (ZKARP)
- Bazı bileşenlerin (talep zamanı, talep miktarı, talep noktası) rasgele bir davranış sergilediği durumlarda ortaya çıkan Olasılıklı Araç Rotalama Problemi (OARP)
- Müşteriye teslimatların belirli günlerde yapılması gibi bir durum olduğunda ortaya çıkan Süreli Araç Rotalama Problemi (SARP)
- Parçalı teslimatların yapıldığı ve bir müşteriye birkaç aracın servis yaptığı bir durum söz konusu olduğunda ortaya çıkan Parçalı Araç Rotalama Problemi (PARP)
- Müşteri teslimatlarının yapılmasından sonra müşterilerden bir şeyler toplanması söz konusu olduğunda ortaya çıkan Geri Taşımalı Araç Rotalama Problemi (GTARP)
- Aracın bir şeyler toplamasının ve müşterilere bir şeyler teslim etmesinin söz konusu olduğu durumlarda ortaya çıkan Toplamalı ve Teslimatlı Araç Rotalama Problemi (TTARP) şeklinde sıralamak mümkündür.

Bir sonraki bölümde ağırlıklı olarak ARP ve türevleri konusunda bugüne kadar yapılan çalışma ve yaklaşımlar irdelenecektir.

3.1 Literatür Araştırması

Klasik Araç Rotalama Problemi (ARP), aynı tür özelliklere sahip bir araç filosu için toplam maliyeti en az yapan yolların bulunmasını ve tasarlanmasını içerir. ARP'de bir coğrafi alana dağılmış müşteriler, depo ve aynı özelliklere sahip araçlar vardır. Problem, depodan başlayıp tekrar depoda sonlanan belirli miktarlardaki müşteri taleplerini araç kapasitesi ve uygun diğer kısıtlamaları dikkate alarak en az maliyete sahip araç rotalarını belirlemeyi amaçlar.

Literatüre bakıldığında ARP üzerinde geniş ölçüde çalışmalar yapıldığı açık bir şekilde görülmektedir. Problemi çözmek için kesin ve yaklaşık olarak birçok algoritma geliştirilmiş ve farklı yaklaşımlarda bulunulmuştur. Kesin algoritmalar sadece küçük büyüklükteki problemleri nispeten çözebilirken, birkaç sezgisel algoritmanın da (Heuristic Algorithm) çok başarılı olduğu ispatlanmıştır. Cordeau ve diğ.(2002) ve Laporte ve diğ. (2000) ARP için Tabu Arama (Tabu Search -TS)'nın en iyi temel sezgisel yöntem (meta-heuristic) olduğunu

açıklamışlardır. TS bilgisayar bellek yapısını kullanan, yerel minimum (local minima) ve sınır döngüler (limit cycles) gibi olağanüstü durumları ortadan kaldırmak amacıyla kullanılan bir meta stratejidir.

Osman (1993) çözüm olabilecek listeyi oluşturmak için genelleştirilmiş bir yol içinde değiş-tokuş ve ekleme hareket şekillerini kullanan standart bir TS algoritması geliştirmiştir. Bu TS işlemi geri çevrilmeleri önlemek için bir harekete sahip düğümleri kullanır.

Gendreau ve diğ. (1992), TS uygulamasının oldukça karmaşık olmasından dolayı 1-ekleme türü hareketleri hesaba katan genelleştirilmiş ekleme yöntemi (Generalised Insertion Procedeure-GENI) algoritmasını geliştirmiştir. Bu yaklaşım, yapım ve ilerleme sezgisel sınıfının arasında bir yerde bulunur ve araç rotasını düğüm düğüm düzenler. Bir düğüm eklendiğinde rota üzerindeki yerel değişiklikleri düzeltme işlemini gerçekleştirir. Bu yaklaşımın görülebilen diğer bir özelliği de p-komşuluk fikrini kullanarak aday listenin boyutunu indirgemeyi içermesi ve cezalar (penalties) ile geçici olumsuzlukları bir çözüme kavuşturmasıdır.

Taillard (1993), büyük çaplı problemleri birkaç alt probleme bölmüştür. Bunların her birini güçlü TS düzenli 1-değiş tokuş komşuluğu kullanarak ve tabu imtiyaz değerini 0.4 ile 0.6 arasında rasgele değişen değerler alarak çözmeye çalışmıştır. Bu yöntem, alt problemlerin ne kadar sıklıkta gerçekleştirildiğini dikkate alarak çeşitliliği sağlamak amacıyla hareketleri cezalandırmaya dayanmaktadır

Rochat ve Taillard (1995) olasılık tabanlı teknik (probabilistic technique) olarak adlandırılan bir uygulama geliştirmiştir. Bu teknik, gerçekte Taillard'ın (1993) benzer arama şemasını kullanmaktaydı. Bununla beraber, bazı arama tekniklerinin etkin kullanımı ile ekstra çeşitlilik ve kuvvetlendirme stratejileri geliştirilmiştir. Bu sebepten, arama işlemi süresince etkin çözümler elde edilmek suretiyle seçkin bir çözüm havuzu oluşturulmuştur.

Xu ve Kelly (1996), şebeke akış temelli TS modeli olarak adlandırılan zincir çıkarım modeli isminde bir algoritma geliştirmişlerdir. Bu algoritma, uzaklık ve olurluluk için eş zamanlı olarak kabul edilen rotalar arasındaki müşteri değişikliği için bir işleyiş sağlamaktaydı.

Blasum ve Hochstattler (2002), gezgin satışçı probleminin kapasite özelliğine sahip bir benzeri olan simetrik kapasiteli araç rotalama probleminin özelliklerine dikkat çeken gezgin satışçı problemi için dallandırma kesme metodunun başarılı bir uygulamasını gerçekleştirmişlerdir. Araç rotalama problemi için dallandırma kesme yaklaşımının geçerli olduğu üç eşitsizlik sınıfını incelemişler. Önce, yönsüz araç rotalama problemindeki çok yıldızlı eşitsizliklerin ihlal edilmesi için ve kapasiteli araç rotalama problemi için yeni, yoğun ve güçlü bir doğrusal sağlayan tam tanımlı bir yöntem geliştirmişlerdir.

Clarke ve Wright (1964), araç rotalama probleminin farklı türlerine uyarlanabilen bir yaklaşımda bulunmuştur. Bu yaklaşımda, bir depolu Q kapasitesine sahip m araçlı kapasiteli gezgin satışçı problemi gibi bir durumun olduğu durumda Clarke ve Wright sezgiseli bazı sabitleri sağlayan R gibi bir tur kümesini gerçekleştirmektedir. Bu sabitler; her tur R kümesinde gerçekleştirilir. R turlarında çakışmalar sadece depoda olabilir, aksi durumda bunlar birbirinden bağımsızdır. R kümesindeki turlar birbirine bağlı tüm bölgeleri içine almış biçimdedir.

Haghani ve Jung (2005), zamana bağlı olarak seyahat zamanlarının formülasyonunu dinamik araç rotalama problemi için yapmıştır. Araştırmada, devamlı olarak seyahat süreleri ile ilgili bilgiler araç rotalama probleminin içine sokulmuş ve ilk giren ilk çıkar (FIFO) için bekleme süresini ortadan kaldırılmıştır. Yol çalışması, trafik yoğunluğu ve hava şartları gibi nedenlerden seyahat zamanı değişkenliği hesaba katılmış ve yeni taleplere gerçek zamanda karşılık verildiği varsayılmıştır. Problemin çözümünde genetik algoritma kullanılmış ve diğer çözüm yöntemleriyle karşılaştırılarak verimliliği ölçülmüştür.

Ichoua ve diğ. (2000), ilk giren ilk çıkar prensibi şartlarını sağlayan zamana bağlı araç rotalama problemi için çözüm yöntemi geliştirmiştir. Jung ve diğ. (2001) araç rotalama problemini çözmek için genetik algoritma yaklaşımını kullanmışlardır. Goldberg ve Lingle (1985), gezgin satışçı problemine çözüm bulmak için genetik algoritmayı kullanmış. Bu çalışma daha sonraki araştırmacılar için bir başvurulacak bir uygulama olmuştur. Uchimura ve Sakaguchi (1995), geleneksel gezgin satışçı problemine farklı bir yaklaşımla yaklaşmış ve diğer çalışmalardan % 1.9 daha uygun sonuçlar elde etmiştir. Chatterjee ve diğ. (1996), genetik algoritma yaklaşımın kullanarak gezgin satışçı problemine benzeri çalışmalara göre % 2.6 daha uygun sonuç bulmuştur.

Thangiah ve Petrovic (1998), zaman kısıtlamalı ve çoklu depolu karmaşık kısıtlamalara sahip araç rotalama probleminin çözümünde genişletilmiş genetik algoritmayı kullanmışlardır. Tan ve diğ. (2001), hibrit genetik algoritma yaklaşımını zamana bağlı araç rotalama probleminin çözümünde kullanmışlardır. Aynı şekilde Berger ve diğ. (2003), zamana bağlı araç rotalama problemini hibrit genetik algoritma yaklaşımını kullanarak çözmeye çalışmışlardır.

Baker and Ayechew (2003), temel araç rotalama problemini çözmek için genetik algoritmayı kullanmışlardır. Çalışmada, talepleri bilinmekte olan müşterilerin talepleri tek bir depodan karşılanmaktadır. Kullanılan araçların taşıyabileceği yük miktarı ve araçların gidebileceği mesafe kısıtlandırılmıştır. Tek aracın tüm müşterilerin taleplerini karşıladığı varsayılmış ve sonuçta komşuluk arama yöntemi ile hibritleştirilmiş genetik algoritmanın, TS algoritmasına ve diğer gerçeğe yakın simülasyon yöntemlerine göre çözüm zamanı ve kalite açısından onlara yakın sonuçlar verebileceğini ortaya koymuşlardır. Yapılan çalışmayla diğer çalışmalara oranla % 0.5 daha iyi sonuç elde etmişlerdir.

Golden ve diğ. (1998), araç rotalama problemine çözüm bulmak için içerisinde genetik algoritmanın da olduğu meta sezgisel çalışmalarda bulunmuştur. Temel sezgisel algoritmaların performanslarını geleneksel araç rotalama test problemlerinde elde edilen sonuçlarla karşılaştırmış ve tabu arama tekniklerinin diğer meta sezgisel tekniklerle elde edilen çözümlerden çözüm kalitesi yönünden daha iyi sonuçlar verdiğini ortaya koymuşlardır. Fakat bu yöntemin karşılaştırma yapılan diğer yöntemlere göre daha çok parametre kullandığını belirtmişlerdir.

Gendreau ve diğ. (1998), köşeler ile ilgili pozitif kazançların ve kenarlarla ilgili uzaklıkların söz konusu olduğu bir graftaki problem olarak tanımladıkları yönsüz seçici gezgin satıcı problemini çözmek için bir algoritma geliştirmişler. Önceden belirlenmiş olan limiti aşmayan köşe alt küme uzunluğu üzerindeki maksimum kazançlı Hamilton çevrimi hesaplamayı içeren bu problemi çözmek için bir tabu arama sezgiseli tanımlamışlardır. Tanımladıkları bu sezgisel, köşe kümlerini yinelemeli olarak o anki tura ekleyen ya da köşe zincirini kaldıran bir yapı içermektedir. Rasgele üretilen ve 300 köşeye kadar çıkan graflar üzerinde yapılan testler geliştirilen yaklaşımın sürekli olarak optimuma yakın verimli sonuçlar verdiğini ortaya koymuştur.

Chu ve Beasley (1997), araç rotalama problemiyle benzer özellikler taşıyan genel görev problemine farklı bir yaklaşım getirmiştir. Bu çalışmada, kaynaklardaki kısıtlamaların göz önünde bulundurulduğu ve her bir görevin minimum maliyetle bir acenteye verildiği bir problem olarak tanımlanan genel görev problemini çözmek için, genetik algoritmadan farklı olarak her bir kromozom için ikili olmayan ondalık sayı dizisi kullanılmıştır. Görevlerin sırası indekste tutulmuş ve her acente numarası ile bu acentenin hangi işle görevlendirildiği sayı dizisinde kodlanmıştır. Araçlar numaralarıyla birlikte müşterilere tahsis edilmiştir. Bu sebeple verilen *n* müşterili *m* araçlı bir durumda, bir çözüm için kromozom n uzunluğunda bir biçime sahiptir ve kromozomdaki her gen [1, *m*] aralığında bulunmaktadır. Kullanılan bu gösterim araç rotalama probleminden biraz farklıdır ve açık bir şekilde araçların takip edecekleri tam rotayı belirtmez. Bununla birlikte müşterileri tahsis edildiği bilinen araçlarla, toplam seyahat mesafesi ve bazı kısıtlama ihlal düzeylerini içeren her rota açık bir şekilde belirtilmiştir.

Gillet ve Miller (1974), sweep tabanlı yaklaşım yöntemini kullanmışlardır. Bu yöntemde müşteriler merkezi bir depo etrafında dağıtılmıştır. Müşteriler önceden bilinen artan kutup açı mantığına göre gruplandırılmışlardır. Sonra uygulama için gerekli olan nüfus üyelerini üretmek ve uygulamaya başlamak için araçlardan bir tanesi bir müşteri için rasgele seçilmiştir. Gruplandırılan müşteriler, kısıtlamaların ihlal edilmeye başlandığı ana kadar belli bir sıra ile rasgele seçilen o anki araca tahsis edilmiştir. Kısıtlamalar ihlal edilmeye başlandığı anda bir sonraki aracın rasgele seçme işlemine başlamadan önce ilk seçilen aracın listesindeki en son müşteri o araçtan kaldırılır ya da kaldırılamaz. Problem için sadece araç kapasite kısıtlaması söz konusudur.

Fisher and Jaikumar (1981), araç rotalama problemine benzerliği olan az kısıtlama temelli genele görevlendirme problemi için farklı bir çözüm metodu geliştirmişlerdir. Bu metotla kısıtlamalar az tutulmuştur. Her araç için başlangıç noktası üreten bir yapıya dayanan bu metotta, tüm müşterilerin bulundukları noktaya göre gidilecek mesafe göz önünde tutularak bir genel görevlendirme işlemi yapılmıştır. Bu görevlendirme işlemi, yük şartını sağlayan araçların söz konusu müşterilere tahsis edilmesiyle gerçekleştirilmiştir.

Braysy and Gendreau (2001), zaman kısıtlı araç rotalama problemini çözmek için tabu arama sezgiselini kullanmışlardır. Belli bir coğrafi alana rasgele dağılmış olan müşteri taleplerini merkezi tek bir depodan karşılamak için kullanılacak en az maliyetli rotaları tespit etmeye çalışmışlar. Araçların verilen zaman aralığında her noktayı yalnız bir kez ziyaret

ettiklerini, araç hareketlerinin depoda başlayıp yine depoda bittiğini ve herhangi bir rota üzerindeki tüm noktalardaki toplam taleplerin kullanılan araçların kapasitesini aşmayacak şekilde olduğu varsayımında bulunarak probleme çözüm bulmaya çalışmışlar. Bunlara ek olarak kullandıkları her metodun temel özelliklerini tanımlamışlar ve elde ettikleri sonuçları literatürdeki benzer çalışmalardan elde edilen sonuçlarla karşılaştırmışlardır.

Lau ve diğ. (2003), ARP'nin limitli sayıda araçların kullanıldığı zaman kısıtlı problemi olan zaman kısıtlı araç rotalama problemine araç ve zaman kısıtı varsayımları altında servis yapılmayan müşterileri ya da zaman kısıtı durumunun biraz daha hafifletildiği durumu kapsayan olası çözümü bulmak için TS yaklaşımını kullanmışlardır. Problem için hesaplanabilir üst sınırlar belirlemişler. Çözüm için, bir güzergâhta hareket eden aracın doldurma yoğunluğunu güçlendirmek için o sistem ve eldeki verileri kullanan TS yaklaşımından faydalanmışlar. Zamanı hafifletmek için de gecikmelere karşın bir cezalandırma notasyonu tanımlamışlar. Yaklaşımlarında, müşterilerin işleri çoklu amaçları içeren hiyerarşik amaç fonksiyonu temelli bir fonksiyon içine eklenmiştir.

Baker ve diğ. (2003), ARP problemini çözmek için rasgele uyarlamalı arama yöntemi tabanlı ve bazı kombinasyonlarda çalışan görsel etkileşimli bir ara yüze sahip yöntem geliştirmişlerdir. Bilinen müşteri taleplerini, ağırlık limiti ile kısıtlandırılmış ve tek bir depoda bulunan araçların karşıladığını varsaymışlar. Bazı durumlarda da araçlara seyahat limiti de getirilmiştir. Bir aracın sadece bir müşterinin talebini karşılamasına izin verilmiştir.

Chu (2005), merkezi bir depodan müşteri taleplerini karşılamak belirli kapasiteye sahip sabit sayıdaki araçların en uygun rotalarını belirlemek ve eldeki araçların toplam talebi karşılamaması durumunda dışarıdan çağrılacak olan araç miktarının belirlenmesinde maliyet fonksiyonunu en aza indirgeyecek sezgisel bir algoritma önermiştir. Her iki durum için de geçerli olan matematiksel ve sezgisel algoritma geliştirmiştir.

Campbell ve Hardin (2005), stoklarını sabit bir süre içinde tükettiği bilinen müşterilere belirli zaman aralıklarında gün boyu teslimat yapan araç sayısını en aza indirgeme problemini problemin karmaşıklığını ve genel özelliklerini göz önünde bulundurarak incelemiştir. Özel durumlar için de problemin en uygunluğunu gösteren bir algoritma önermişlerdir.

3.2 Bölüm Özeti

Bu bölümde ARP'nin çözümünde literatürde önerilen yaklaşımlar irdelenmiştir. İrdelenen bu yaklaşımlardan yola çıkarak ARP'nin çözülmesi zor bir problem olduğu, problemin çözümünde tabu arama ve sezgisel algoritmaların oldukça fazla kullanıldığı ve iyi sonuçlar elde edildiği ve ayrıca problemin çözümünde genetik algoritmanın, yapay sinir ağlarının az da olsa kullanıldığı görülmüştür.

Bir sonraki bölümde C# programlama dili irdelenecek ve C# programlama dilinin tercih edilmesinin sebepleri anlatılacaktır. Ayrıca C# programlama dilindeki önemli bazı kavramlar açıklanacaktır.

DÖRDÜNCÜ BÖLÜM
NESNE YÖNELİMLİ PROGRAMLAMA
VE
C# PROGRAMLAMA DİLİ

4. NESNE YÖNELİMLİ PROGRAMLAMA VE C# PROGRAMLAMA DİLİ

Bu bölümde Nesne Yönelimli Programlama (NYP) hakkında ve NYP'nin dört temel yapısı olan veri soyutlama, veri kapsülleme, kalıtım ve çok biçimlilik hakkında bilgi verilecektir. Ayrıca NYP'nin avantajları ve simülasyonun gerçekleştirildiği .NET platformu tanıtılacak. Araç Rotalama Probleminin modellenmesinde ve bilgisayar ortamına aktarılmasında kullanılan C# programlama dilinden bahsedilecektir.

Günümüzde bilgisayar teknolojisindeki hızlı gelişmeler, bilgisayar tabanlı modellemelerin artmasına sebep olmuştur. Bilgisayar teknolojisinin hızlı gelişimi berberinde, matematiksel olarak modellenebilir birçok problemin modellenmesine imkân sağlayan yeniliklerin ortaya çıkmasını sağlamıştır (Arora G, 2002).

Her geçen gün yazılımların daha karmaşık bir hal alması, yazılım kodlarının kurumsal uygulama projelerinde on binlerce satırı bulması ve yazılım maliyetlerinin çok fazla artması bilim adamlarını yazılım noktasında yeni arayışlara yöneltmiştir. Bilim adamlarının yazılımcılara nesneleri kullanarak yeni bir yaklaşımın kullanılabileceğini göstermeleri çok yeni bir yazılım dilinin ortaya çıkmasına sebep olmuştur. İşte bu yeni yaklaşım ve yazılım dili Nesne Yönelimli Programlama (Object Oriented Programming)'dır.

Nesne yönelimli programlama (NYP), bir yazılımı nesneleri kullanarak tasarlama ve uygulama şeklidir. Nesne yönelimli programlama dillerinin birçoğu, bir dil sözdizimi ve derleyicisi olmakla beraber, uygun ve tam bir geliştirme ortamı sunar. Bu geliştirme ortamı iyi tasarlanmış ve aynı zamanda kullanımı kolay nesneleri bünyesinde barındıran bir kütüphane içerir.

Nesne yönelimli programlama yaklaşımında temel olan prensiplerden birisi bilgi gizlemedir (information hiding). Nesne yönelimli uygulamalar ve bilgisayar ortamında modellenmiş sistemler, kullanıcılara grafiksel olarak zengin bir şekilde bilgi sunar. Gerçek problemlerde var olan karmaşıklık probleminin çözülmesine katkı sağlarlar. Bunu gerçekleştirmek için de, nesnelerin içinde bulunan bazı bilgileri gizlerler. Nesne yönelimli programlama, öncelikli olarak kullanmakta olduğu veri üzerinde yoğunlaşır. Daha sonra da veriyi işleyen ve üzerinde işlemler yapan fonksiyonları (yöntemleri), verinin bir parçası olarak meydana getirir.

Nesne yönelimli programlamayı herhangi bir programlama dili ile ilişkilendirmek yanlış olur. Fakat nesne yönelimli programlamayı destekleyen bir programlama dili yaklaşımı daha doğru olan bir yaklaşımdır ve böyle bir programlama dilinde nesne yönelimli yaklaşımın kullanılması daha kolay ve pratik olur.

Nesne yönelimli programlama yaklaşımı ve tekniği, diğer yaklaşımlara oranla, yazılım geliştiren kişilere büyük avantajlar sağlamaktadır. Bu avantajların ilki, karmaşık yazılım projelerinin geliştirilmesi ve bakımını kolaylaştırıyor olması, diğeri de yazılım kodunun tekrar kullanılabilmesine (code-reusability) imkân sağlıyor olmasıdır. Bu noktada program kodunun tekrar kullanılabiliyor olması profesyonel yazılım şirketlerinin yazılım maliyetlerinin düşmesini sağlamıştır. Bu durum yazılımların lisans ücretlerinin düşmesine, sektörün sürekli olarak canlı kalmasına ve rekabet içinde gelişmesine yardımcı olmuştur. Yani nesne tabanlı yaklaşım, yazılım sistemlerinin geliştirilmesi evresini iyileştirmekle birlikte fonksiyonelliğin ve bu yaklaşımda kullanılan verinin ortaklığını geliştirir. Aslında nesne yönelimli programlama, bilgisayar sistemlerinde problem çözümlerine yeni bir bakış açısı ve açılımı getiren yeni bir yaklaşımdır. Bu yaklaşım probleme en iyi ve aynı zamanda tam bir çözüm aramayı ve bulmayı amaçlamaktadır. Bir programcı gerçek hayattaki olaylara yakın senaryolarla çalışır. Bunun için yapılması gereken ilk şey bilgisayarın gerçek hayattaki nesnelerle ilişki kurmasını sağlamaktır.

Bilgisayara bilgi verirseniz bu bilginin karşılığını alırsınız. Nesne yönelimli programlama bu noktadan sonra devreye girer ve gerçek hayatta karşılaştığımız gerçek varlıkları, bilgisayar ortamında gerçek hayattakine benzer varlıklar olarak modellememize imkân sağlar. Nesne yönelimli bir yazılım geliştirmedeki amaçlanan şey gerçek sistem ile uygulama arasındaki uyuşmazlığı azaltmaktır.

Nesne yönelimli programlamada asıl olan şey, gerçek hayatta var olan olguların programlamaya aktarılmasındaki yeni yaklaşımdır. Prosedürel programlamalarda veriler ve fonksiyonlar vardır. Her şey veri ve bu veriyi işleyen metodlar etrafında döner. Esasında nesne yönelimli programlamada da veri ve veriyi işleyip mantıklı sonuçlar üreten metodlar olmak üzere iki önemli birim bulunmaktadır. Fakat buradaki fark, gerçek hayattaki olguların daha iyi gözlemlenerek yazılım dünyasına aktarılmasındadır.

Nesne yönelimli programlama, veriyi ve bu veri üzerinde yapılması düşünülen işlemleri (yöntem) tek bir nesne olarak görür. Veriyi kendine özgü kimliği ve özellikleri olan bir varlık olarak görür. Bunun içindir ki nesne yönelimli programlamada ‘nesne’, ‘veri’ ve ‘yöntem’ çok önemli kavramlardır.

Nesne kavramı, gerçek hayatın her alanına uygulanabilme özelliğine sahiptir. Geliştirilen herhangi bir uygulama, insanın düşünce seyrini en iyi taklit etmek için varlıklar ve nesnelerden faydalanır.

Nesne yönelimli programlama dört temel yapı üzerine kurulmuştur. Bu dört temel yapı şu şekilde sıralanmaktadır:

- Veri Soyutlama
- Veri Kapsülleme
- Kalıtım
- Çok Biçimlilik

4.1 Veri Soyutlama

Gerçek hayatta soyutlama herhangi bir nesnenin teknik olarak nasıl çalıştığını değil, sadece ne olduğunun üzerinde yoğunlaşır. Örneğin çoğu kişinin arabadan anlaması soyutlama sayesindedir. Birçok kişi arabanın motorunun ne işe yaradığını bilir fakat nasıl çalıştığını ve çalışma prensibini bilmez. Çünkü araba motorunun çalışması için birçok parçası ve bu parçaların her birinin farklı görevleri vardır. Fakat soyutlamada insanların, arabanın nasıl çalıştığını en ince ayrıntısına kadar bilmeseler de, arabalar hakkında konuşabilmelerini, arabalar hakkında düşünebilmelerini ve araba kullanabilmelerini sağlar.

Program tasarımı ile uğraşanlar, soyutlamanın çok karmaşık program projelerinin yürütülmesinde mutlaka gerekli bir unsur olduğunun farkındadırlar. Küçük programlarda programların satır sayısı fazla değildir ve bir kişi tarafından en ince ayrıntısına kadar anlaşılabilir. Fakat büyük ticari programlar binlerce hatta milyonlarca kod satırından oluşurlar ve soyutlama olmadan bu kodları anlamak, tasarlamak ve düzeltmek imkânsızdır. İşte bu tür projeler kontrol soyutlama ve veri soyutlama yardımı ile gerçekleştirilirler.

Veri soyutlama, programın algoritmik yapısı ile değil veriler ve bu verilerle yapılan işlemler ile ilgilidir. Veri soyutlama programlama dilinde olmayan, kullanıcı tarafından tasarlanmış veri yapıları ile kullanılır. Her veri yapısı birtakım değişkenlerden, bazen sabitlerden ve bazen de bu verilerle yapılan işlemlerden olur. Bu tip veri yapıları soyut veri yapıları olarak ta adlandırılır.

Veriyi soyutlamanın bir takım avantajları vardır ve bunları şu şekilde sıralamak mümkündür:

- Problemin üzerine odaklanır
- Gerekli işlemleri ve özellikleri belirler
- Gereksiz ayrıntılardan kurtulmayı sağlar

Verinin soyutlanması gereklidir, çünkü bir nesnenin tüm özelliklerini ve işlemlerini kullanmak pratik olmayabilir. Pratik olan şey asıl odaklanılması gereken noktalara odaklanmak ve uygulamalarda bunları gerçekleştirmek önemlidir. Gereksiz şeylerle uğraşmak zaman kaybıdır ve programcı ancak veri soyutlamayla bundan kurtulabilir.

4.2 Veri Kapsülleme

Her sınıf özellik ve metot gibi bir takım üyelerden meydana gelir. Genellikle tüm sınıflar başka bir sınıf tarafından kullanılmak amacıyla oluşturulduğundan, bazı üyeler sadece dışarıdan kullanılmak üzere hazırlanmışlardır. Buna karşın bazı özellik ve metotlar, diğerlerine yardımcı olmak, sadece onlar tarafından sınıfın iç işlerinde kullanılmak amacıyla yazılır. Belli bir sınıfı kullanan sınıfın bunları görmesi ya da bilmesi gerekmez. Hatta bazı durumlarda özellik ve metotların sadece az bir kısmı dışarısı tarafından kullanılır. Bir kişi belli bir sınıfa baktığında üyelerden hangisinin işine yaradığını, hangisinin bazı metotlar

tarafından kullanılmak için yapıldığını ve bir sınıfa özel olduğunu anlaması oldukça zor olabilir. Bazen güvenlik sağlamak amacıyla bir sınıfı kullanan başka bir sınıfın, bazı özelliklerinin değiştirilmesiyle birkaç metoda erişmesi engellenmek istenebilir. Bazı özellik ve metotların ait oldukları sınıfın dışında erişimini sınırlama özelliğine kapsülleme adı verilir.

Kapsüllemenin avantajları; bir nesnenin çalışma ayrıntılarını kullanıcısından saklama işlevini görür. Yönettiği kod ve veriyi birbirine bağlar ve bu ikisini de dış kaynaklı karıştırma ve yanlış kullanımdan korur. Bu şekliyle veriyi dış ortamdan koruyan bir ambalaj vazifesi görür.

4.3 Kalıtım

Kalıtım, yeni bir nesne türetmek yerine, var olan başka bir nesnenin özelliklerinin alınarak bu özelliklere yenilerinin eklenmesi ile başka bir nesnenin türetilmesine verilen isimdir. Nesneler bu şekilde hiyerarşik bir grup oluşturabilirler. Programlama dilinde nesnelere sınıfta denilebilir. Kalıtım kavramı programlama dilinde sınıf türetme olarak geçer. Kendisinden türetme yapılan sınıfa üst sınıf, türeyen sınıfa da alt sınıf denilir. Bir üst sınıftan birden çok alt sınıf türetilebilir ve her alt sınıf, kendinden hiyerarşik olarak daha alt düzeyde olan alt sınıfların üst sınıfı olabilir. Bu şekilde kalıtım; bir sınıfa, diğer sınıflardan tanımlanmış öznitelikleri ve işlemleri kendi içinde kullanma imkânı verir. Başka bir sınıftan miras alan sınıf, alt sınıftır. Kalıtım yoluyla diğer sınıflara davranışlarını aktaran da üst sınıftır. Tekli ve Çoklu olmak üzere iki çeşit kalıtım vardır.

Tekli kalıtım, alt sınıflar sadece tek bir üst sınıftan türetilmiş ise böyle bir kalıtımdan bahsedilir. Bir sınıf, sınıf hiyerarşisinin ne kadar aşağısında ise gerçek hayattakinin tersine o kadar özellik ve yetenek taşır.

Çoklu kalıtım, alt sınıfların birden çok üst sınıftan oluşması durumunda çoklu kalıtım söz konusudur. Çoklu kalıtımlar tekli kalıtımlara göre çok daha karmaşıktır ve programcıların çoğu tarafından kullanılmaz.

4.4 Çok Biçimlilik

Çok biçimlilik, birbirinden farklı veri tipleri arasında işlem yapılabilmesidir ve bir işlemin farklı nesnelerde farklı biçimde davranmasını sağlar. Çok biçimlilik ile birbirinde farklı sınıflara benzer işlem uygulanırsa, işlem farklı sonuçlar ortaya çıkarır. Benzer işlemden kasıt, sadece aynı sayıda parametre aldığı anlamındadır. Çok biçimlilik, nesne yönelimli sistemin çok önemli bir özelliğidir ve kapsüllemeyi destekler.

4.5 Nesne Yönelimli Yaklaşımın Avantajları

Nesne yönelimli programlama, programcıların programlamaya yaklaşımının değişmesini gerektiren bir programlama yaklaşımıdır. Bu yaklaşım, programların geliştirilmesi zamanını kısaltır ve iyi kullanılması durumunda da bakıma, tekrar kullanılabilirliğe katkıda bulunur. Nesne yönelimli yaklaşımın avantajlarını şu şekilde sıralamak mümkündür:

- Bir problemin inceleme, tasarım ve gerçekleştirme aşamalarına, uygulama alanı ve terimleri ile yaklaşarak uygulama ve gerçek problem arasında çok yakın bir benzerlik sunar.
- Yeni uygulamaların geliştirilmek istenmesi durumunda, önceden var olan nesnelerin yeniden kullanılması imkânını sağlar. Bu durum uzun vadede yazılım geliştirme maliyetlerinde önemli bir tasarruf sağlar.
- Hatalar ve bakım sorunlarında azalmayı sağlar.
- Nesnelerin yeniden kullanılmasının sonucu olarak; tasarım ve yazılım geliştirme süreçlerini hızlandırır.
- Uygulama içindeki paylaşımı destekler.

4.6 .NET Platformu

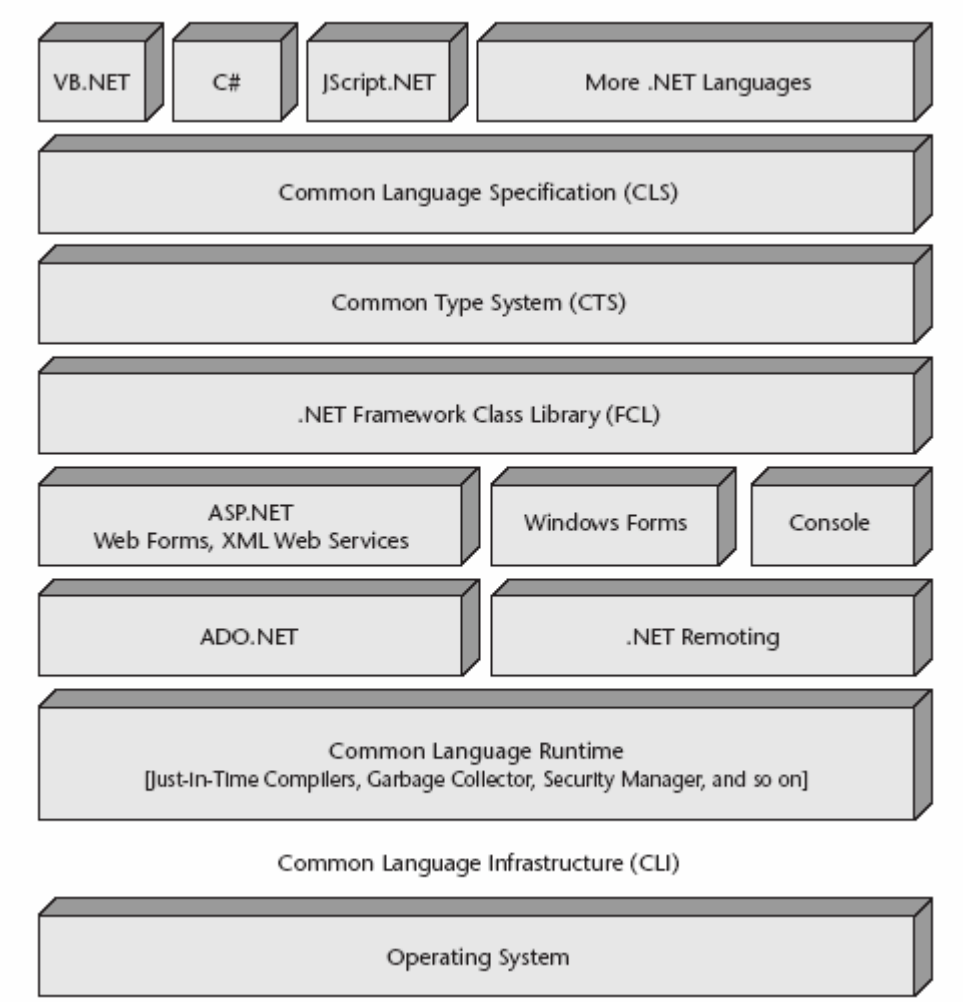
.NET Platformu, Internet'i en üst seviyede dağıtık bilgi işleme platformuna dönüştürmek amacıyla tasarlanmış teknolojiler kümesi olarak tanımlanabilir. Bu platform web servisleri koleksiyonlarından uygulamalar oluşturmak için yeni yollar sağlamaktadır. .NET Platformunun amacı dağıtık web uygulamaları için ihtiyaç duyulabilecek araç ve teknolojileri kullanarak web geliştiriciliğini kolaylaştırmaktır. .NET Platformunun sağladığı imkânları şu şekilde sıralamak mümkündür;

- Bir uygulamanın tüm katmanları üzerinde dilden bağımsız ve tutarlı programlama modeli sunar
- Teknolojiler arasında görünmeyen fakat alt yapıda var olan birlikte çalışabilirlik özelliğine sahiptir.
- Var olan teknolojiler arasında kolay geçiş imkânı sağlar. Bir .NET sınıfının işlevselliği her .NET uyumlu dil ya da programlama modeli tarafından kullanılabilir.
- HTTP, XML, SOAP gibi iletişim standartlarına uygun, platformdan bağımsız ve standartları temel alan internet protokollerine tam destek verir.

.NET platformu için kullanılan ana teknolojilerden birisi, uygulama geliştirmeyi kolay hale getiren ve verimliliğin en üst seviyeye çıkmasını sağlayan .NET Framework teknolojisidir.

.NET Framework teknolojisi, web tabanlı uygulamaların, web servislerinin ve hemen hemen her türlü uygulamanın geliştirilip çalıştırılabildiği, programlama dilinden ve platformdan bağımsız ve nesne yönelimli programlamayı destekleyen uygulama geliştirme platformudur. Bu platformun üç anahtar kelimesi "Build-Deploy-Run" kelimeleridir. Yani önce programı inşa et, sonra yerleştir ve sonra da çalıştır anlamına gelmektedir. .NET Framework verimlilik açısından yüksek, standartlara uygun ve çok sayıda dil desteğine sahip bir platformdur. İnternet uygulamalarının geliştirilmesinde ve yayımlanmasında ortaya çıkan zorlukların ortadan kalkmasını sağlayan servisleri içerisinde barındırır.

İşletim sistemi ile geliştirilen uygulamanın arasında yer alır ve geliştirilen uygulamanın derlenerek işletim sistemine uyumlu hale getirilip çalıştırılmasını sağlar. Uygulama ve işletim sistemi arasındaki bu arabirim, herhangi bir dilde yazılan uygulamanın derlenmesinden sorumludur. Birçok hazır sınıf kütüphanesini bünyesinde barındırarak derlenen uygulamanın çalıştırılması görevini yerine getirir. İşletim sistemlerinden bağımsızdır ve geliştirilen uygulamaları bulunduğu işletim sistemine uyarlar. Geliştirilen programın türü önemli değildir çünkü .NET Framework web, windows, mobil uygulamalar başta olmak üzere birçok uygulamayı destekler. .NET Framework birçok programlama dili desteğine sahip olduğundan programcının herhangi bir programlama dilinde uygulamanın geliştirmesine imkân sağlar. .NET Framework'ün mimarisi Şekil 4.1'deki gibidir.



Şekil 4.1: .NET Framework Mimarisi

.NET Framework, yirmiden fazla programlama diline destek vermekte olup, uygulama geliştiren programcıların iş mantığı içeren kod kısmına yoğunlaşmalarını sağlamaktadır. Bunun sonucu olarak güvenli, sağlam, üstün performanslı uygulamaların geliştirilmesine imkân vermektedir. Diğer yöntemlere göre; program geliştirme, yayımlama ve yönetimini çok kolaylaştırmaktadır. .NET Framework'ün bazı özelliklerini şu şekilde sıralamak mümkündür.

- **Platformdan Bağımsızdır**

Yeni bir gelişme ortamı için platform bağımsızlığı önceliklidir. Çünkü önceden var olan ortamlarda geliştirilmiş olan uygulamalar platformdan bağımsız değildi. Örneğin, Windows platformu için geliştirilmiş olan bir uygulama Unix ortamına uyumlu değildi. .NET Framework bu durumu ortadan kaldırmıştır. .NET Framework'ün geliştirilmesi ile Internet

ortamında ve değişik platformlarda çalıştırılabilecek uygulamalar geliştirmek mümkün hale gelmiştir. .NET desteğinin bir sonucu olarak .NET uygulamalarının en güçlü özelliği olan ortamlar arası geçişteki birlikte işlevsellik olarak adlandırılan MSIL (Microsoft Intermediate Language) ortaya çıkmıştır. MSIL, tüm dillerden çevrilmiş ortak bir dil olarak karşımıza çıkmaktadır. Bu dilde aslında müdahale edilebilir ve üzerinde değişiklik yapılabilen bir tür programlama dilidir. Derleme zamanında .NET platformu için yazılan tüm kod işlemciden bağımsız bir ara kod olan MSIL'e dönüştürülür. Windows platformu için yazılan bir kod herhangi bir platformda çalıştırılmak istendiğinde derleyici MSIL kodunu çalıştırılan platformun anlayacağı hale getirir.

- **Dilden Bağımsız Uygulamalar Geliştirmeyi Destekler**

Farklı ortamlardakilerle birlikte çalışabilen, dilden bağımsız uygulamalar geliştirmeyi sağlayan .NET, .NET ailesinden herhangi bir dili kullanarak geliştirilen bir uygulamanın kodunu diğer .NET diline kolayca çevirebilme imkânı sunar.

- **Web Uygulamalarını Destekler**

ASP gibi Script dillerini kullanarak web sayfası oluşturmak web programcıları için kolay olmamaktadır. Bu sebeple .NET programcılar için basit ve anlaşılabilir kodlar üretmek amacıyla ASP.NET teknolojisini sunmaktadır. ASP.NET teknolojisi kullanılarak oluşturulan web sayfaları web uygulamaları olarak adlandırılmaktadır. ASP.NET kullanarak yeni ASP.NET sayfaları oluşturulabileceği gibi var olan ASP sayfalarını ASP.NET sayfalarına dönüştürülebilmektedir. ASP.NET programcıya her hangi bir .NET programlama dilinde oluşturduğu çok ileri seviyedeki fonksiyonel sayfaları, web sayfasına eklemesine imkan tanımaktadır.

- **Web Servislerini Destekler**

.NET Framework, internet üzerinden bilgilere ulaşabileceğimiz farklı platformlar için uygulama geliştirmek amacıyla kullanabileceğimiz web servisleri oluşturmamızı sağlar. Bunu gerçekleştirmek için, bir sınıftan üretilen nesneye ait metodlar, internet üzerinden çağrılabilir ve farklı platformlarda çalışabilen uygulamalar tarafından kullanılabilir. Buna ek olarak web servisleri, uzaktaki sunuculara erişebilmeye yardım eder, sunucudan bir metodu çağırabilir, sunucu üzerinde bir sınıf örneği oluşturabilmeyi mümkün kılar ve sunucu üzerinde işlemler gerçekleştirebilmeyi sağlar. Web servisleri kullanıcının bir sunucuya erişim işlevini

basitleştirmek amacıyla HTTP'yi kullanır. HTTP XML kullanılarak yazılan mesajların sunucu ile istemci arasında transfer edilmesine yardım eder.

Dilden bağımsız olan .NET platformu programların yürütülebilmesi için gerekli olan tüm ortak servisleri .NET Framework mimarisi vasıtasıyla sağlar. .NET Framework sadece Microsoft Visaul Basic.NET, Microsoft Visual C#, C++.NET, Microsoft Jscript.NET dillerini değil üçüncü parti birçok programlama dilini (APL, COBOL, Pascal, Eiffel,Haksel,ML, Oberon, Perl, Python, Scheme, Fortran, Curriculum ve SmalTalk) destekler.

.NET Framework .NET Platformunun bütünleşik parçası olan teknolojiler kümesidir. .NET Framework, ASP.NET kullanımıyla web uygulamaları ve web servislerinin geliştirilmesi için gereken temel yapı taşlarını sağlar. Bu yap taşı servisleri, hem çevrim içi hem de çevrim dışı kullanıma hazır olan, dağıtık programlanabilir servislerdir. Bir servis, internet'e bağlı olmayan, şirket içinde çalışan yerel bir sunucudan sağlanan tek bilgisayar tarafından çağrılabilir ya da internet üzerinden erişilebilir. .NET yapı taşı servisleri SOAP'ı destekleyen herhangi bir platform tarafından kullanılabilir. Servisler, kimlik, uyarı ve mesajlaşma, klasör, arama ve yazılım teslimini içerir.

4.7 C# Programlama Dili

C#, güçlü, modern, nesne yönelimli ve aynı zamanda tip-güvenli (type-safe) bir programlama dilidir. Aynı zamanda C# , C/C++ dilinin güçlülüğünü ve Visual Basic'in ise kolaylığını sağlamaktadır. Java programlama dilinin ortaya çıkmasından bu zamana kadar programcılık alanında ortaya çıkan en büyük gelişmedir. C#, C++'ın gücünden, Java'nın da özelliklerinden faydalanarak tasarlanmış bir nesne yönelimli programlama dilidir. Şunu da ilave etmek gerekir ki şu an itibariyle C# Delphi ve C++ Builder'daki bazı özellikleri içerisinde barındırmaktadır. Fakat Delphi ya da C++ Builder hiçbir zaman Visual C++ ya da Visual Basic'in popülaritesini yakalayamamıştır.

C ve C++ programcıları için en önemli ve en büyük sorun, hızlı geliştirememe olarak düşünülmektedir. Çünkü C ve C++ program geliştiricileri çok alt seviye ile ilgilenirler ve üst seviyeye çıkmak istediklerinde zorluk çekerler. Ama C# ile artık bu durum ortadan kalkmıştır. Aynı ortamı kullanarak ister alt seviyede isterseniz üst seviyede program geliştirme imkânı

bulunmaktadır. C# programlama dili Microsoft tarafından geliştirilen .NET platformunun en temel ve resmi dili olarak sunulmuştur.

C# dili Turbo Pascal derleyicisini ve Delphi'yi oluşturan takımın lider Anders Heljberg ve Microsoft'ta Visual J++ takımında çalışan Scott Wiltamuth trafından geliştirilmiştir. C# bazı özellikleri sebebiyle, JAVA, VB ya da C++ benziyor gibi görünse de bunlar gibi değildir. C, C++ ve JAVA'nın güzel özelliklerini içinde barındıran yeni bir programlama dilidir.

Nesne yönelimli programlamanın günümüzde ne kadar yaygın olduğu bilinen bir gerçektir. Nesne yönelimli programlama yaklaşımının temel prensiplerinden birisi bilgi gizleme (information hiding)'dir. Bu prensip C#'ın sunduğu en önemli araçlardan birisi olan sınıf özellikleridir (class properties). Bir sınıf, nesneleri oluşturmak için kullanılan veri yapısı olarak tanımlanır. Bir nesne verileri içeren ve birbirleriyle ilişkilendirilmiş metotlara sahiptir. Veri ve metotlar sınıfın üyeleridir.

4.8 Neden C#

Araç Rotalama Simülasyonumuzu C#'ta yapmamızın temel sebeplerini şu şekilde sıralayabiliriz. C# programlama dili;

- C++'ı Temel alan, yenilikçi bir dil olması
- Nesne yönelimli programlamaya tam destek vermesi
- Birlikte çalışmada güvenilirlik sağlaması
- Modern ve bileşen dayalı bir dil olması
- Güçlü hata ayıklama ve test araçlarına sahip olması
- Mükemmel bir uygulama geliştirici araç seti sunması
- Windows tabanlı uygulama geliştirme konusunda mükemmel ortam sunması
- Zengin bir sınıf kütüphanesi sunması
- Güçlü web uygulamaları geliştirme ortamı sunması
- Hızla büyüyen uygulama geliştirici topluluğunun olması
- Gelecekte gelişmeye açık bir programlama dili olması gibi nedenler

C# programlama dilini seçmemizdeki önemli etkenlerdir.

4.9 Bölüm Özeti

Bu bölümde NYP'nin öneminden ve özelliklerinden bahsederek, C# programlama dilinin seçilişinin sebepleri üzerinde durulmuştur.

Bir sonraki bölümde Araç Rotalama Probleminin C# ortamında geliştirilen simülasyonu hakkında detaylı bilgi verilecektir.

BEŞİNCİ BÖLÜM
ARAÇ ROTALAMA PROBLEMİ SİMÜLASYONU

5. ARAÇ ROTALAMA PROBLEMİ SİMÜLASYONU

Araç rotalama probleminin söz konusu olabilmesi için, problemin uygulanacağı bir coğrafi alanın, bu coğrafi alan içerisinde rasgele dağılmış müşterilerin, bu müşterilere hizmet götürecek araçların ve bu araçların yüklerini yükleyip boşalttıkları depoların olması gerekmektedir.

Klasik olarak araç rotalama problemi (ARP), bir coğrafi alan içinde merkezi bir depo etrafında rasgele dağılmış olan müşterilerden gelen talepleri belirli kapasite ve hıza sahip araçları kullanarak en az maliyetle karşılama problemidir. Taşıma maliyetinin en aza indirgenmesi, araçların takip edebilecekleri en uygun rotaların belirlenmesine bağlıdır. Bir şirket herhangi bir coğrafi alan içerisinde, müşterilerine teslimat yapmak için kullandıkları depoların konumlarını belirlerken konumun uygunluğunu göz önünde bulundurması gerekmektedir.

Bir coğrafi alan içindeki yollar, müşteri teslimatlarını gerçekleştirmek için kullanılan vazgeçilemeyecek bir unsurdur. Bu yollar, kavşaklar gibi bağlantı noktalarıyla birbirine bağlanmaktadır. Dolayısıyla bu durum, bir noktadan farklı bir noktaya giderken çok sayıda yol seçeneğinin karşımıza çıkmasına sebep olmaktadır. Bu durumda ürün teslimatı yapan şirket, zaman ve uzaklık gibi iki temel şeyi dikkate almak zorundadır. Eğer şirket için müşteri memnuniyeti önemli ise-ki bir şirket için en önemli şeylerden biri budur- zaman kavramını mutlaka dikkate almalı ve müşteri taleplerini zamanında karşılamalıdır. Bir talebin zamanında karşılanması veya en kısa zaman içinde karşılanması, araçların kullandıklar yolların gidilecek noktaya uzaklığına bağlıdır.

Kullanılan yollarda araçların talep noktasına zamanında ulaşmasını etkileyen sebepler bulunmaktadır. O andaki yolun durumu, hava şartları, coğrafi konum ve trafik yoğunluğunu

bu sebepler arasında saymak mümkündür. Dolayısıyla bir rotalama problemi çözülürken bunların dikkate alınması en uygun yolun belirlenmesi açısından çok önemlidir.

ARP ve türevleri, çözümleri zor olan problemlerdir. Bu durum ARP ve türevleri problemlerin çözümüne farklı yaklaşımların ortaya çıkmasına sebep olmuştur. Bu yaklaşımlardan bazılarını genetik algoritma, tabu arama algoritmaları, en kısa yol algoritmaları, dinamik programlama, sezgisel ve meta sezgisel metotlar olarak saymak mümkündür. Tüm bu yaklaşımların genelinde belirli kısıtlamaların olduğu görülmektedir. Doğrusal olarak çözülmesi mümkün olmayan problemlerde böyle kısıtlamaların olması çok doğaldır. Bu algoritmaların ortaya çıkma sebebi de, belirli bazı kısıtlamalar yapmak suretiyle problemin çözümüne yakın en uygun sonucu elde etmektir. Zaten birebir kesin bir sonucun elde edilmesi de beklenmez.

5.1 Simülasyonun Tanıtımı

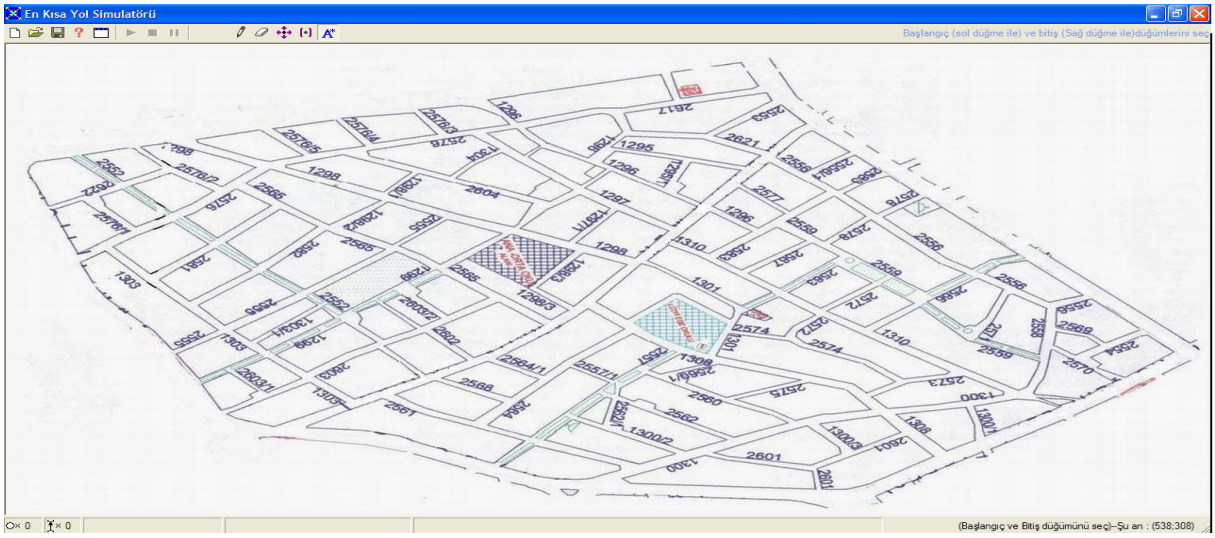
Bir coğrafi alandaki yollar ve kavşaklar graf kuramında tanımlanan kenar ve köşelerle ifade edilebilir. Nitekim 1736'da Leonhard Euler Seven Bridges of the Konigsberg problemini çözerken köprüleri kenar, köprülerin birbirine bağladığı kara parçalarını da köşe olarak ifade etmiş ve problemi bu şekilde çözmüştür.

Bu çalışmada graf kuramının bu özelliklerini kullanarak herhangi bir coğrafi alanı graf kuramındaki köşeler ve kenarlar şeklide ifade edebildiğimiz görsel bir arayüz geliştirilmiştir. C# programlama dilinde geliştirilen bu arayüze, ARP probleminin uygulamasının yapılacağı coğrafi alanının resmi arka plan resmi olarak eklenmiştir. Sonra coğrafi alanımızdaki yollar tek yönlü ya da çift yönlü olarak kenar şeklinde, kavşaklar da köşe olarak görsel bir biçimde ifade edilmiştir. Coğrafi alanın durumuna göre kıvrımı fazla olan yollardaki kıvrımlar da ölçümlerin gerçeğe daha yakın olması açısından köşe olarak gösterilmiştir. Her köşenin koordinatları dizilerde tutulmuştur. Kullanılan algoritmanın içine, birbirine doğrudan bağlı olan köşeler arasındaki uzaklığı, oklidean mesafe (Euclidean Distance) cinsinden bulan formülü yerleştirilerek kenarların ağırlığı uzaklık türünden hesaplanmıştır.

Simülasyonun arayüzünün üst kısmına, kenar ve köşe ekleme, kenar ve köşe silme, kenar ve köşe taşıma, kenar ve köşeyi aktifleştirip pasifleştirme ve ARP problemi için herhangi bir köşeye depo yerleştirmek amacıyla kullanılan düğmeler bulunduğu araç çubuğu

bulunmaktadır. Ayrıca, ARP problemi için oluşturulan graf, problem için oluşturduğumuz farklı senaryoları daha sonra kullanılmak üzere kaydetmek için kaydetme düğmesi, sonradan açmada kullanılan açma düğmesi, ARP için uygulanan senaryoyu başlatmak ve istenildiğinde durdurmak için kullanılacak düğmeler de aynı araç çubuğu üzerinde bulunmaktadır.

Simülasyonun arayüzünün alt kısmında, ARP probleminde aldıkları toplam mesafeyi, geçen toplam süreyi ve bu süre içinde karşılanılan talep miktarını gösteren durum çubuğu vardır. Ayrıca, coğrafi bölgeyi kaç kenar ve kaç köşe ile ifade etiğimizi gösteren köşe ve kenar sayısı da aynı durum çubuğunun en solunda bulunmaktadır. Şekil 5.1 ARP probleminin uygulandığı coğrafi alanın programın arayüzündeki görünümü görülmektedir.

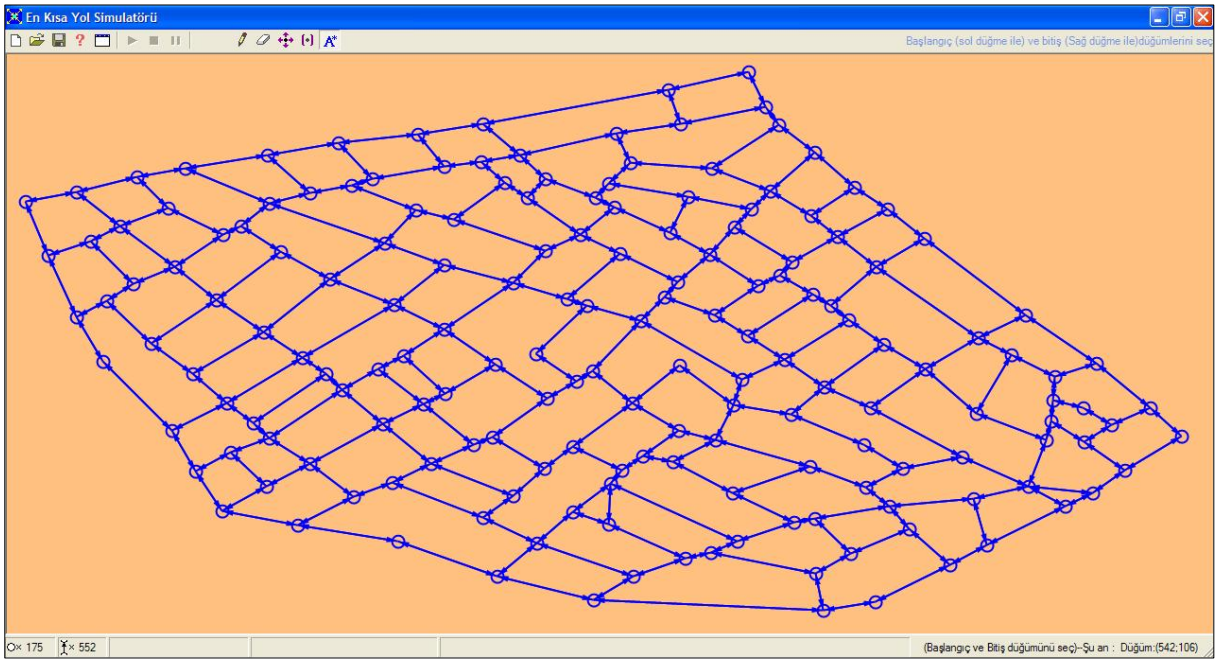


Şekil 5.1: Coğrafi alanın program arayüzünde görünümü

Şekilde 5.1'de görülen coğrafi alandaki yolları, caddelerin kesiştiği noktaları ve kıvrımı fazla olan yerlerdeki kıvrımların, köşe ve kenarlar olarak ifade edilmiş hali Şekil 5.2'deki gibidir.

Şekil 5.2'de görüldüğü gibi tüm caddelerin kesiştiği noktalar, kavşaklar köşe olarak ve bu kavşaklar ve kesişme noktalarını birbirine bağlayan yollar da kenar olacak şekilde gösterildi. Burada yerleşimden dolayı bazı caddeler ve sokakların kıvrımları bulunmaktadır. Bu kıvrım noktalarını da köşe olarak ifade ettik. Eğer iki köşe arasındaki yol üzerinde kıvrımlar varsa, bu köşelerin dik bir çizgiyle birleştirilmesi gerçek hayattaki durumla çelişkili bir durumun ortaya çıkmasına sebep olur. Çünkü bir noktadan diğer noktaya kuşbakışı olarak çizilerek

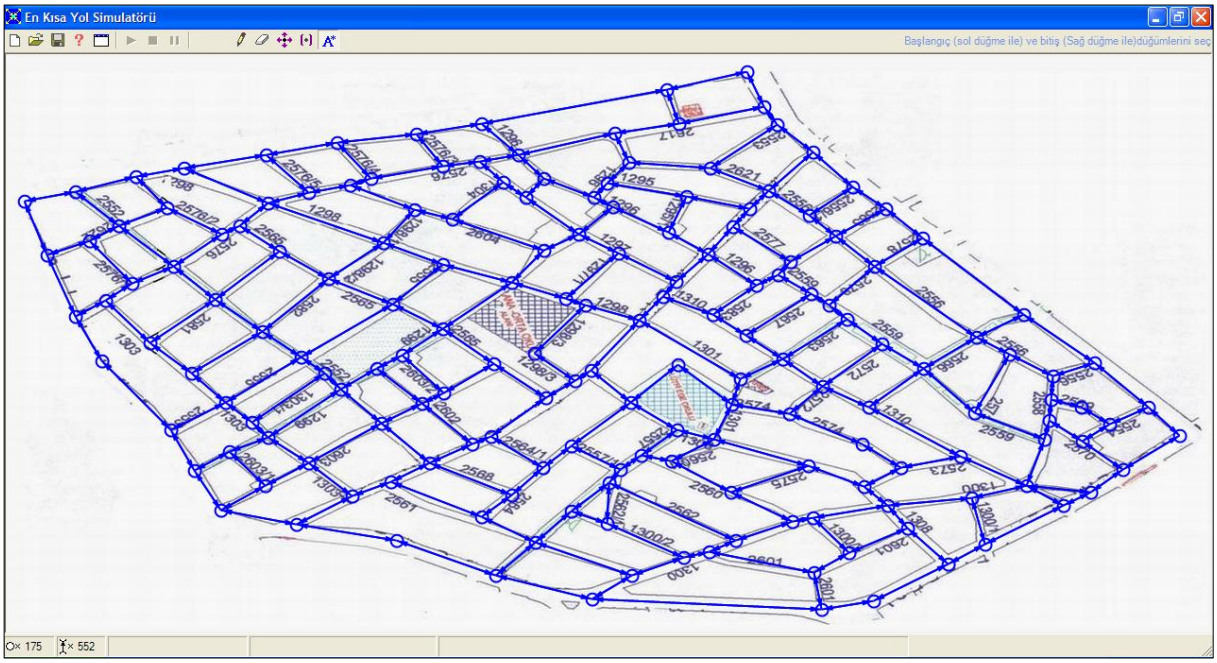
ifade edilmiş bir yolu takip ederek değil, aradaki kıvrımlara da uğrayarak gidilir. Burada coğrafi alan graf veri modeline uyarlanırken buna dikkat edildi.



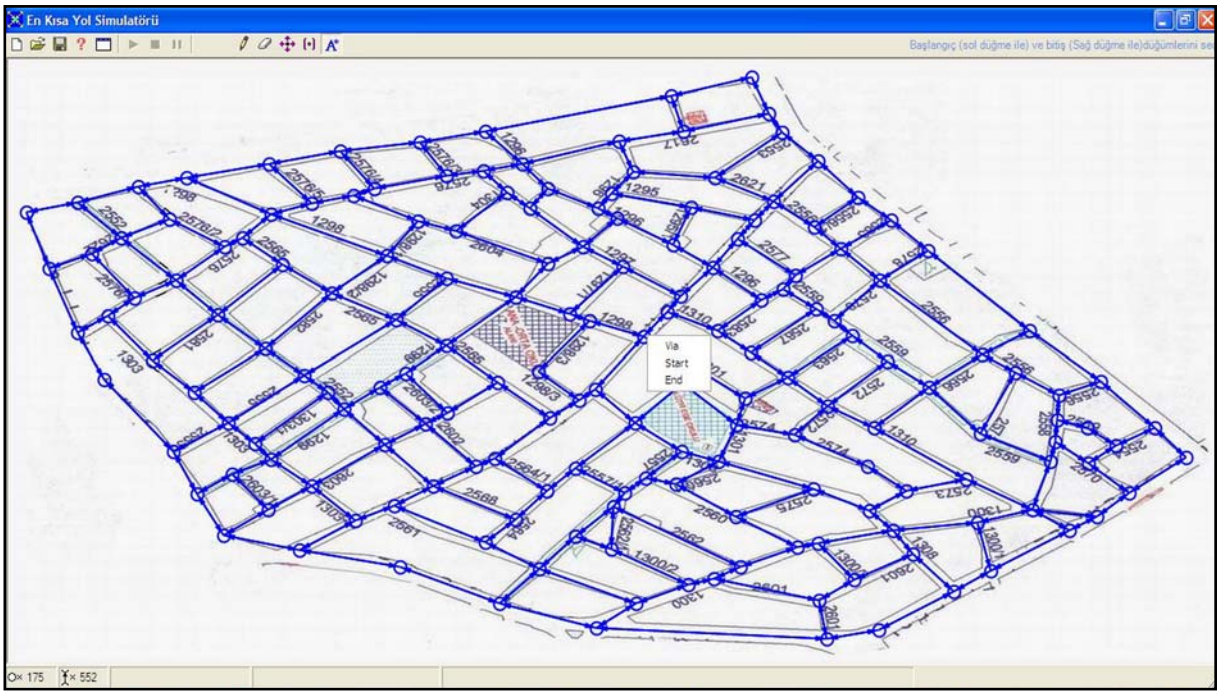
Şekil 5.2: Coğrafi alanın graf veri modeline uyarlanması

Bundan sonraki aşamada, ARP uygulamasını gerçekleştirmek için, coğrafi alanın şekli ile coğrafi alanın graf veri modeline uyarlanmış şekli program arayüzünde birleştirildi. Böylece bir anlamda deponun yerleştirebileceği nokta ve müşteri talep noktaları belirlenmiş oldu. Coğrafi alan ve bu alana ait graf veri modelinin birleştirilmiş hali Şekil 5.3'deki gibidir.

ARP probleminin uygulanabilmesi için bu coğrafi alanında köşe olarak ifade edilen noktalardan birine veya daha fazlasına müşterilere servis yapılacak depoların yerleştirilmesi gerekir. Depo ya da depolar dışındaki tüm köşeler müşteri talep noktaları olarak kabul edilmektedir. Depoyu eklemek için, depoyu ekleyeceğiniz köşe üzerinde farenin sağ düğmesini tıklatıp açılan listeden start seçeneği seçilmektedir. Bunu Şekil 5.4'te görmek mümkündür.



Şekil 5.3: Coğrafi alanın ve graf veri modeline uyarlanmış halinin birlikte gösterimi

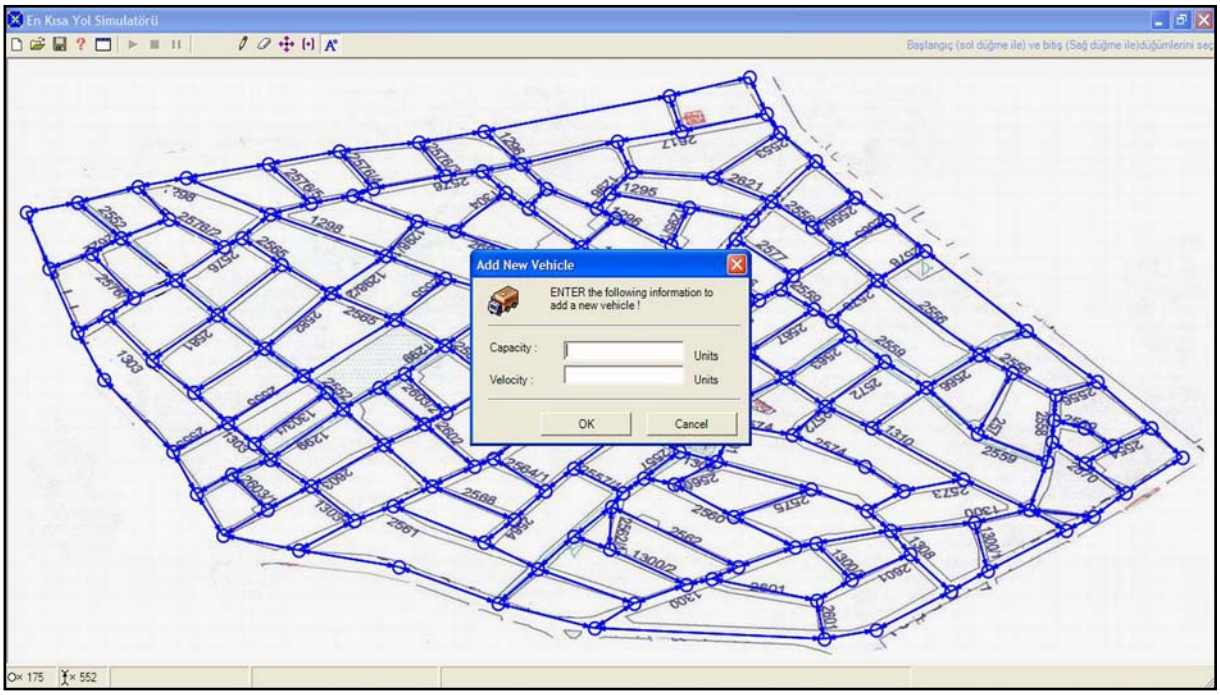


Şekil 5.4: Graf veri modelindeki herhangi bir köşeye depo ekleme

Depoyu ekledikten sonra, deponun resmi bu köşede belirir. Sonraki aşamada eklenen depoya ARP probleminde kullanılacak araçların eklenmesi işlemi gerçekleştirilir. Bu işlem de deponun eklenmesinde takip edilen yöntemle birbirine benzer. Depo eklenirken nasıl köşe üzerinde sağ fare düğmesini tıklatılıp açılan listeden *start* seçeneğini seçiliyorsa, araç

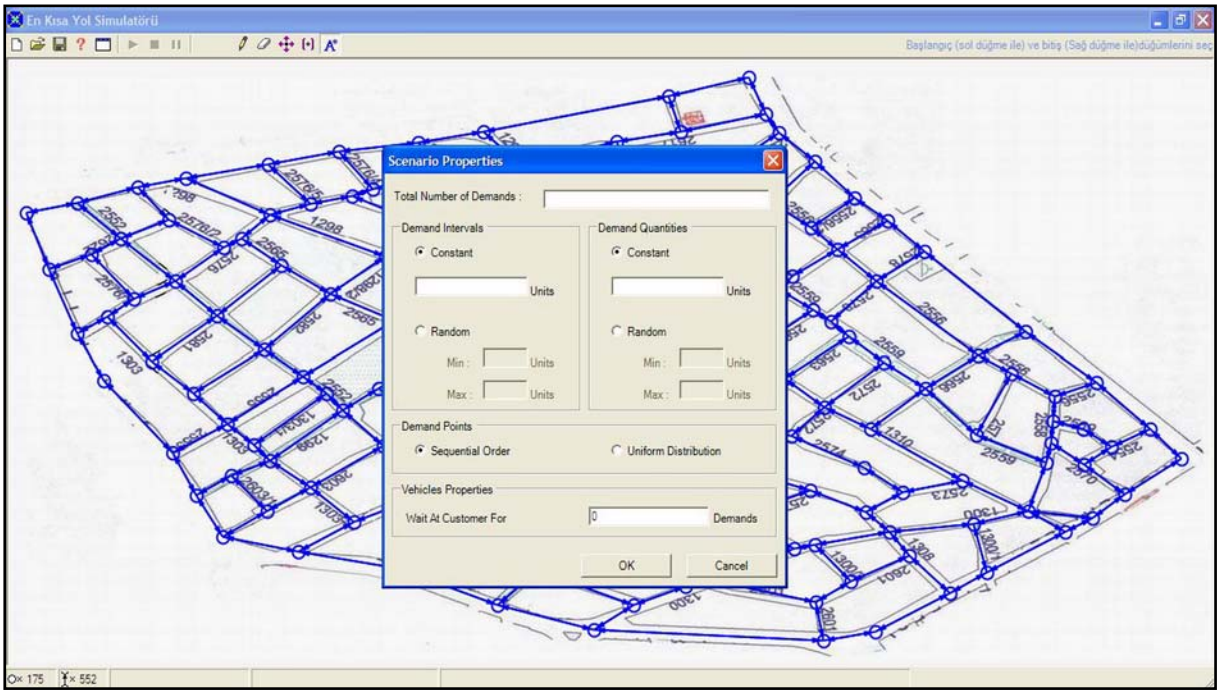
eklerken de yerleştirilen deponun kenarında sağ fare düğmesi kullanılarak açılan listeden *Add Vehicle* seçeneğini seçmek gerekir.

ARP probleminde kullanacak araçlar belirli bir kapasite ve hıza sahip olduğundan araç ekleme esnasında açılan formu kullanarak aracın hızının ve kapasitesinin belirlenmesi işlemi yapılır. Şekil 5.5 eklenmekte olan aracın kapasitesi ve hızının belirlendiği ekran görülmektedir.



Şekil 5.5: Depoya eklenen aracın hız ve kapasitesinin belirlenmesi

Depoya araçların eklenme işi tamamlandıktan sonra ARP probleminin uygulanması için senaryoların oluşturulması gerekir. Senaryonun oluşturulduğu formun görünümü Şekil 5.6'daki gibidir.



Şekil 5.6: Senaryo ekleme ekranı

Bu formu kullanarak oluşturulacak senaryoların özelliklerini aşağıdaki gibi sıralamak mümkündür:

1. Toplam talep sayısı kullanıcı tarafından belirlenir. Bu formdaki Total **Number of Demands** bölümü kullanılarak belirlenir.
2. Talebin geliş zaman aralığı ya sabittir ya da kullanıcını belirleyeceği zaman aralığındadır. Bu **Demand Intervals** kısmı kullanılarak gerçekleştirilir.
3. Gelen her bir talep sabit miktarlarda olabileceği gibi rasgele miktarlarda da olabilir. Bunu belirlemek için de **Demand Quantities** alanı kullanılır.
4. Talepler sıralı bir şekilde gelebileceği gibi dağınık bir şekilde, yani rasgele noktalardan gelebilir. Bunu belirlemek için de **Demand Points** alanı kullanılır.
5. Bir aracın talep noktasına varıp talebi karşıladıktan sonra, karşılanacak diğer talepler için bu noktadaki bekleme süresinin belirlenmesidir. Bir araç, talebi karşıladıktan sonra bir sonraki talebi bekler. Araç, gelen talebe yakınsa bu talebi karşılar yakın değilse en yakın depoya döner. Buradaki bekleme süresi, aracın talebi karşıladığı andan sonra gelecek kaç talep kadar beklemesi gerektiğini belirten sayıdır. Buradaki bekleme süresi, aracın depoya dönme şartlarından biridir. Yani bekleme süresinin 1 olması, bir talep gelene kadar beklesin, gelen talebe yakınsa o talebe gitsin, değilse en yakın depoya dönsün. Bekleme süresinin 2 olması, talep

gelene kadar beklesin, gelen talep için aracın kullanılması uygun değilse bir sonra gelecek talep için beklesin. Bunda da kullanılmıyorsa depoya dönsün anlamındadır.

Senaryo oluşturduktan sonra, daha sonra farklı durumlara uygulanması için kaydedilebilir. Senaryoyu kaydederken dosya uzantısının *.scn olmasına dikkat edilmesi gerekir. Aynı şekilde graf veri modelini de *.grf olarak kaydetmek mümkündür. Depoları ekledikten sonraki durum da *.gfo olarak kaydedilebilir. Böylece aynı alan içinde farklı durumlar için aynı senaryo çalıştırılıp elde edilecek sonuçların karşılaştırılması sağlanacaktır.

5.2 Senaryoların Çalıştırılması

ARP problemi için gerekli alan ve yukarıda belirtilen işlemler yapıldıktan sonra senaryoyu başlatmak gerekir. ARP simülasyonunda çalıştırılan senaryolar için iki algoritma önerilmiştir. Önerilen bu algoritmalar ve çalışma prensipleri şu şekildedir.

1. Gelen talep o anda kendisine en yakın aracın listesine eklensin (Always Closest)
2. Gelen talep aracın gitmekte olduğu noktaya yakın ise listesine eklensin (Closest to Target)

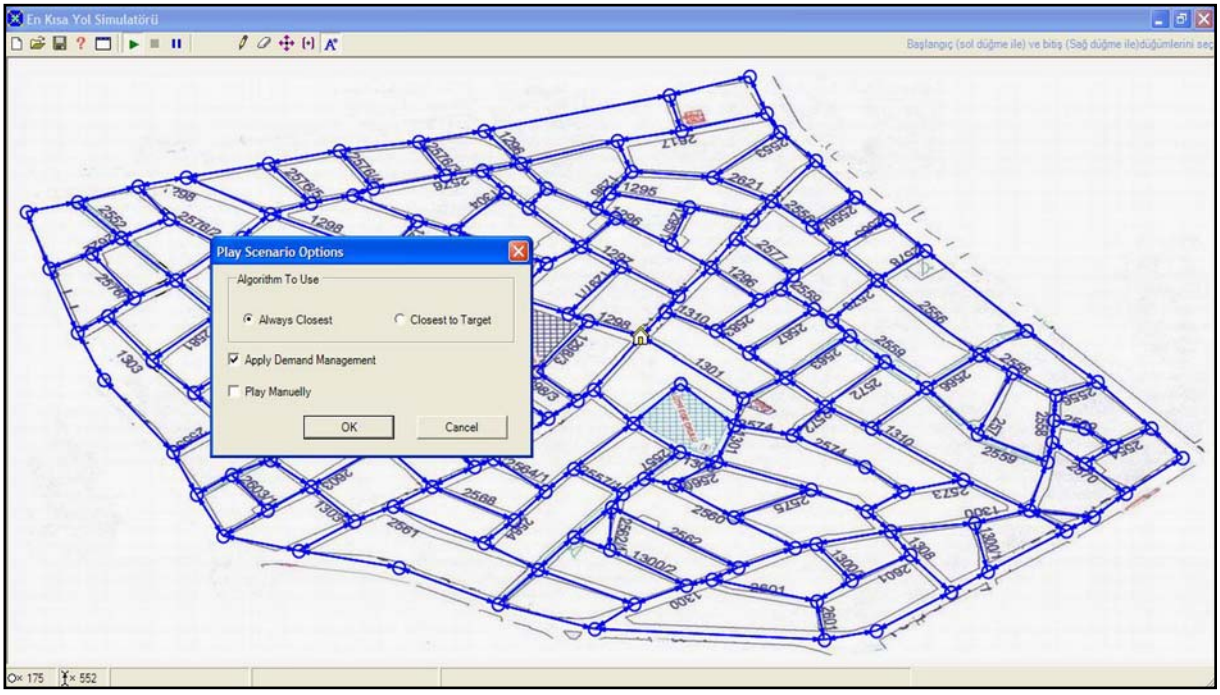
İlk durumda; bir talep geldiği anda hangi araç talebe daha yakınsa onun listesine girer. Gelen her talep için bu durum geçerlidir. Bu algoritmaya göre talep geldiği anda araçların talep noktasına uzaklığı önemlidir.

İkinci durumda; bir araç herhangi bir talebi karşılamaya giderken talep geldiğinde aracın gitmekte olduğu noktanın gelen talebe olan uzaklığına bakılır. Hangi aracın gittiği nokta gelen talebe yakınsa bu talep o aracın listesine eklenir. Bu algoritmaya göre talep geldiği anda araçların gitmekte olduğu noktanın talep noktasına uzaklığı önemlidir.

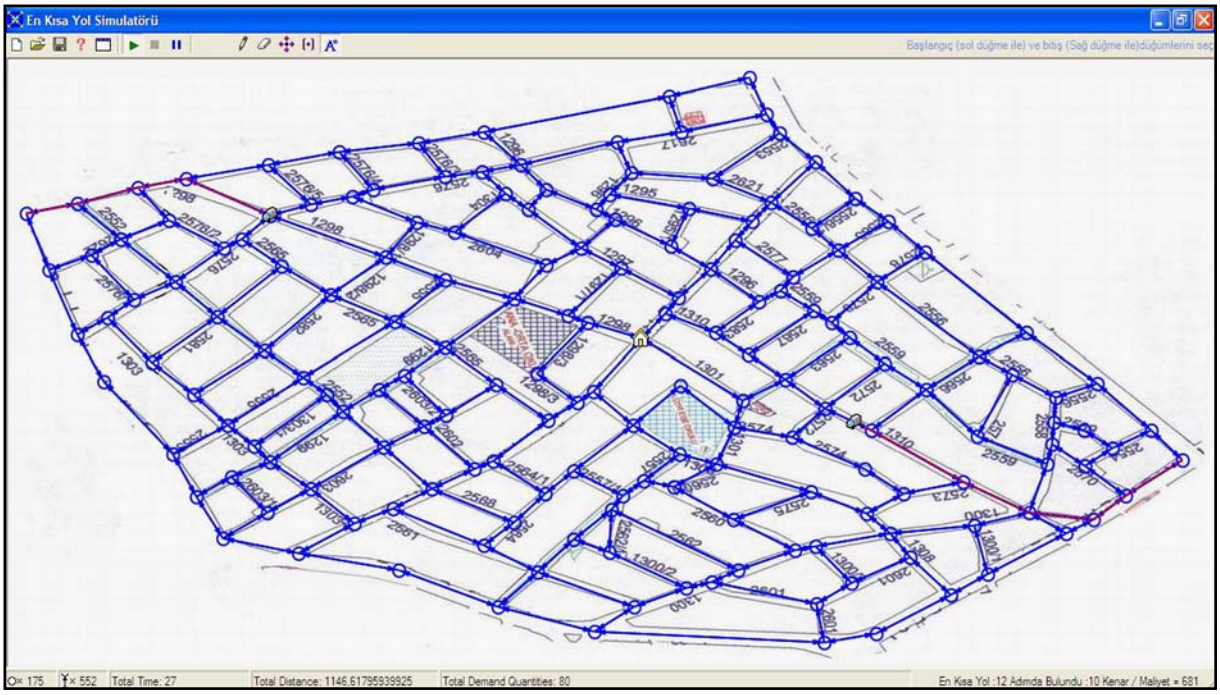
Önerilen bu iki algoritma, aynı senaryonun farklı durumları için incelenmiştir. Her iki durum için alınan toplam yol ve toplam süre hesaplanarak hangi algoritmanın daha iyi sonuçlar verdiği tespit edilmeye çalışılmıştır. Her iki durum için de talep yönetimi uygula seçeneğinin seçili olması gerekmektedir. Aksi taktirde simülasyon araçların her gelen talebi karşıladıktan sonra depoya dönmelerini gerektirir. Bu durum gerçek hayatta uygulanmayan ve optimum olmayan bir durumdur. Zaten elde edilen sonuçlar da bunu göstermektedir.

Senaryolardan elde edilen sonuçlar ve her bir talebin her iki algoritmada ne kadar sürede karşılandığı bilgisi metin dosyasına yazdırılmıştır.

Simülasyonu her iki algoritmaya göre manüel olarak çalıştırmak mümkündür. Bu durum en kısa yol yaklaşımın kullanarak gezgin satışçı problemine bir çözüm sunmaktadır. Depo dışındaki her noktadan talep geldiği varsayılır ve bu durum manüel olarak ayarlandıktan sonra senaryo başlatılırsa, araçlar en kısa mesafeli talepten başlayarak bütün talepleri karşılar. Eğer coğrafi alanda bir araçlı depo ile işlem yapılırsa gezgin satışçı problemi gerçeklenmiş olur. Eğer birden çok araçlı tek depo varsa bu durum çoklu gezgin satıcı problemine bir örnek olmuş olur. Senaryonun başlangıcında uygulanmak istenen algoritmanın seçimi Şekil 5.7'de, bu algoritmaya göre araçların hareketi de Şekil 5.8'de gösterilmiştir.



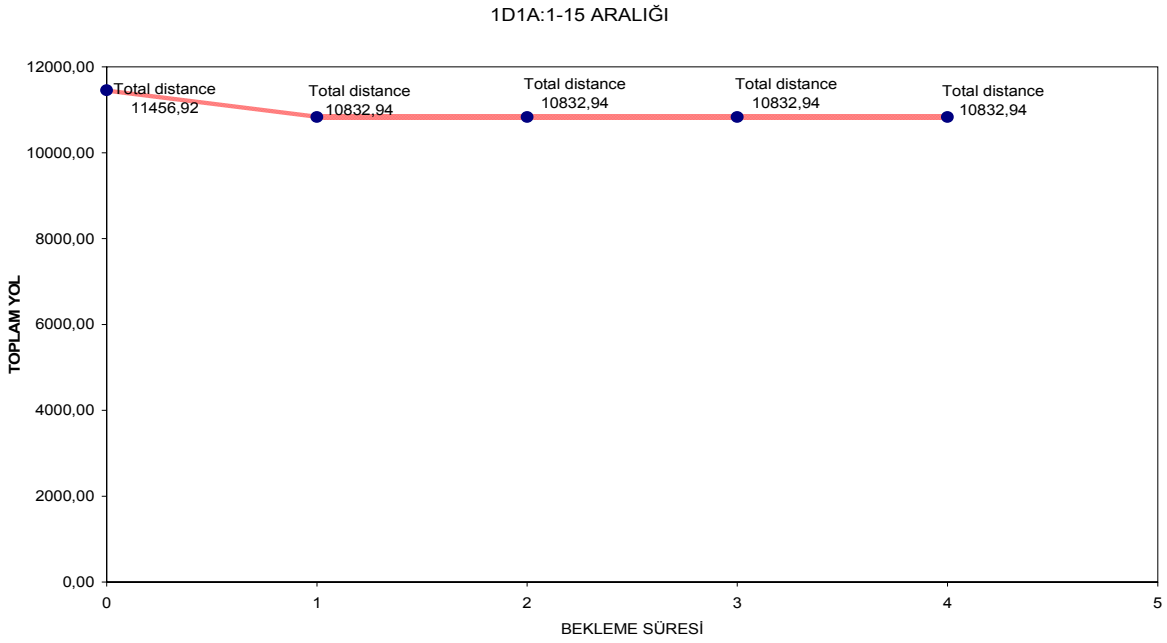
Şekil 5.7: Senaryo için algoritma seçimi



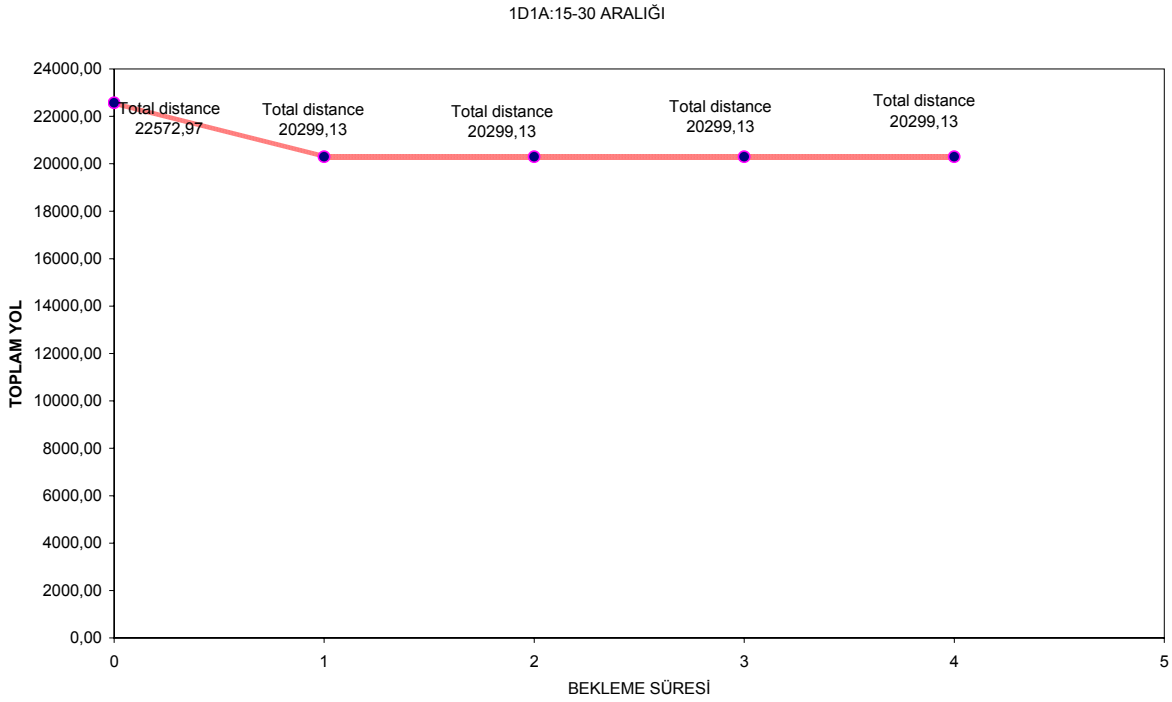
Şekil 5.8: Araçların talep noktalarına hareketi

5.3 Senaryoların Uygulanması ve Sonuçlar

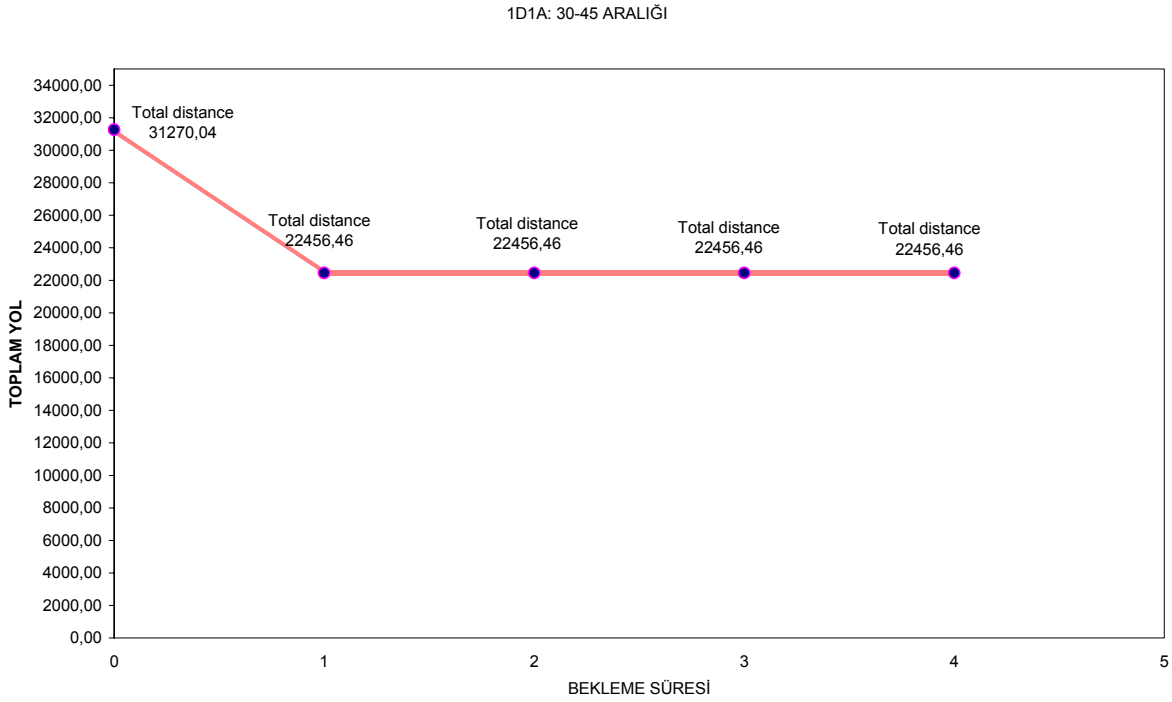
Bu aşamada aynı senaryolar içerisindeki farklı değişkenler değiştirilerek elde edilen sonuçlarda araçların talep noktalarında belli bir süre beklemesinin alınan toplam yola ve dolayısıyla maliyete etkisi incelenmiştir. Tek depo tek araç için 1-15, 15-30, 30-45 ve 45-60 zaman aralığındaki toplam yolun değişim grafikleri Şekil 5.9, Şekil 5.10, Şekil 5.11 ve Şekil 5.12'de gösterilmiştir.



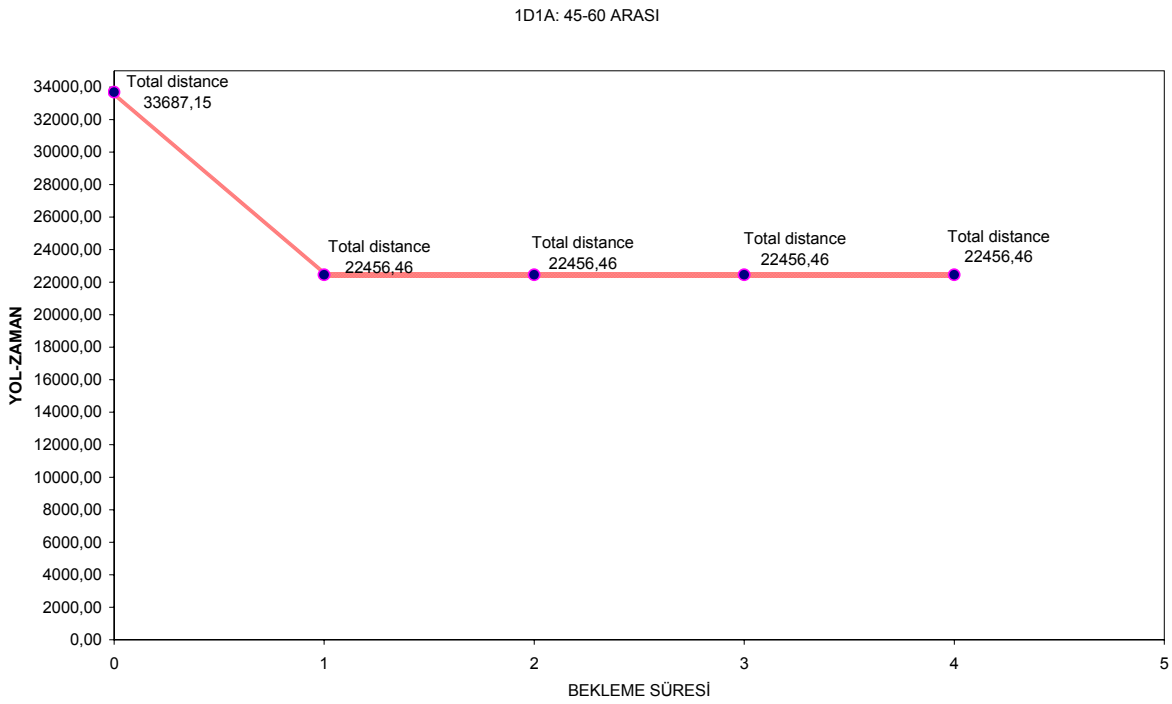
Şekil 5.9: Tek Depo Tek Araç İçin 1-15 Aralığında Toplam Yol-Bekleme Süresi Grafiği



Şekil 5.10: Tek Depo Tek Araç İçin 15-30 Aralığında Toplam Yol-Bekleme Süresi Grafiği

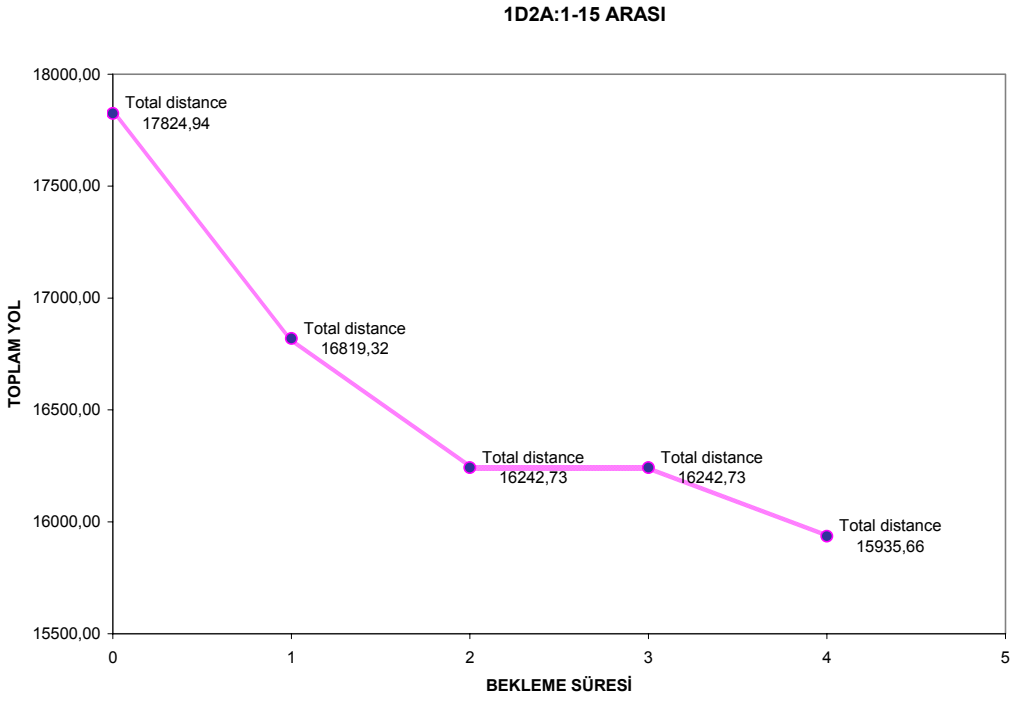


Şekil 5.11: Tek Depo Tek Araç İçin 30-45 Aralığında Toplam Yol-Bekleme Süresi Grafiği

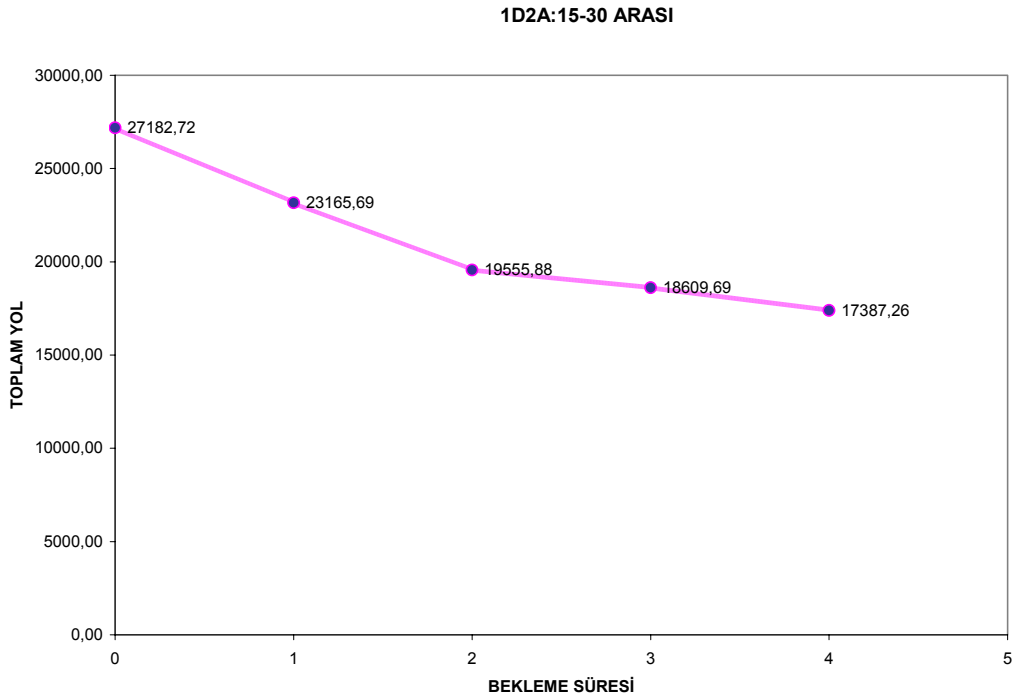


Şekil 5.12: Tek Depo Tek Araç İçin 45-60Aralığında Toplam Yol-Bekleme Süresi Grafiği

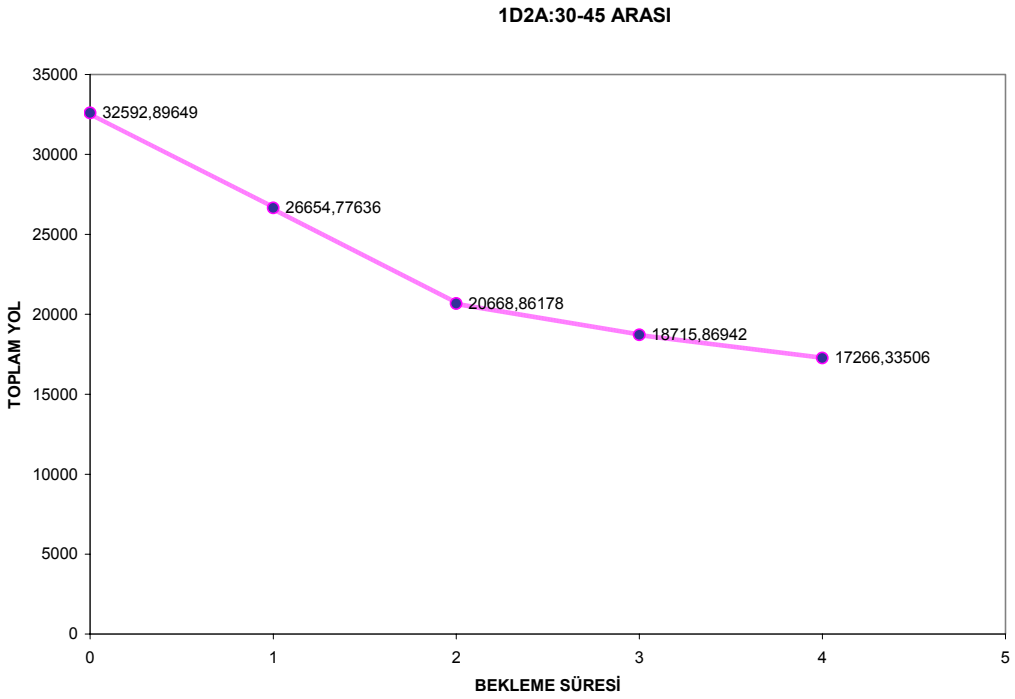
Tek depo tek araç kullanılarak elde edilen sonuçlara ek olarak, tek depo 2 araç kullanılarak elde edilen toplam yol-bekleme süresi grafikleri de Şekil 5.13, Şekil 5.14, Şekil 5.15 ve Şekil 5.16'da gösterilmektedir.



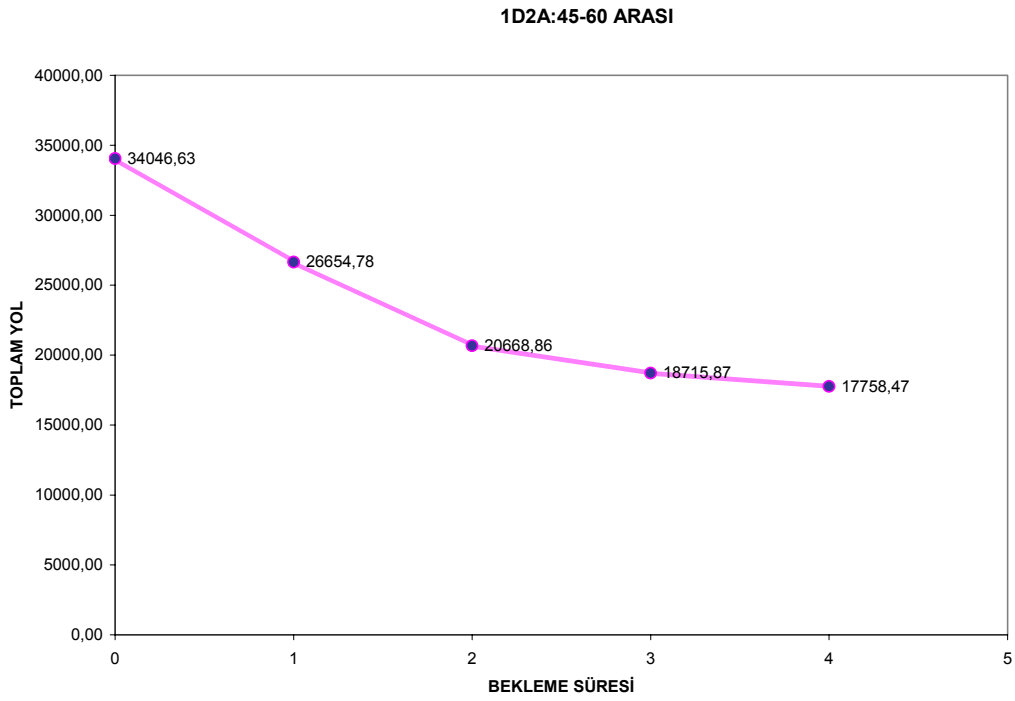
Şekil 5.13: Tek Depo İki Araç İçin 1-15 Aralığında Toplam Yol-Bekleme Süresi Grafiği



Şekil 5.14: Tek Depo İki Araç İçin 15-30 Aralığında Toplam Yol-Bekleme Süresi Grafiği



Şekil 5.15: Tek Depo İki Araç İçin 30-45 Aralığında Toplam Yol-Bekleme Süresi Grafiği



Şekil 5.16: Tek Depo İki Araç İçin 45-60 Aralığında Toplam Yol-Bekleme Süresi Grafiği

Talepler, belli bir zaman aralığında ve belli miktar aralığında rasgele noktalardan gelmektedir. Araçların belli bir hızı ve kapasitesi vardır. Aynı senaryo, bir depo bir araç, bir depo iki araç olan coğrafi alanda bekleme süresi sıfır, bir, iki ve üç olması durumunda çalıştırılmış ve elde edilen sonuçlar karşılaştırılmıştır. Elde edilen sonuçların tabloları Ek-A'dadır. Bu sonuçlara göre;

- Çok seri gelen talepleri karşılamada araçların talep noktasındaki bekleme süresinin toplam zaman etkisinin yok denecek kadar az olduğu görülmüştür. Taleplerin çok sık aralıklarla gelmesi, bekleme süresinin anlamını yitirmesine neden olmuştur.
- Çok seyrek aralıklarla gelen talepler için bekleme süresinin toplam alınan yolun azalmasına neden olmuştur. Talep noktasında bir veya daha fazla bir süre gelecek talebin beklenmesi aracın depoya dönmesi durumunda kat edeceği yolun ortadan kalmasına sebep olmuştur. Bekleme süresi, bir sonra gelecek talebin beklemekte olan araca yakın olması durumu, aracın bu talebi depoya gitmeden karşılamasına sebep olduğundan alınan yolun azalmasını sağlamıştır.
- Araç sayısının artması taleplerin karşılanma süresini azaltmıştır. Fakat bu durum, alınan toplam yolun bazı durumlarda artmasına sebep olmuştur. Araç sayısının

fazlalığı ve bekleme süresinin fazlalığı alınan toplam yolun önemli ölçüde azalmasını sağlamıştır.

- Bekleme süresinin fazla olması, az olması durumunda aracın depoya dönmesi için kat edeceği yolun ortadan kalkmasını sağlamıştır. Gelen bir sonraki taleplerin depoya dönmeden karşılanması sağlanmıştır. Bu durum aracın daha verimli bir şekilde ve az maliyetle kullanılmasını sağlamıştır.

Bir sonraki bölümde elde edilen sonuçlar verilecek ve gelecekte neler yapılabileceği hakkında bilgi verilecektir.

ALTINCI BÖLÜM
SONUÇ VE ÖNERİLER

6. SONUÇLAR VE ÖNERİLER

Bu bölümde yapılan çalışmanın sonuçları ve devamında yapılabilecek çalışmalar irdelenmiştir. İleride yapılabilecek çalışma önerileri sunulmuştur.

6.1 Sonuçlar

ARP problemi C# programlama dilinde graf veri modeline uyarlanarak simule edilmiştir. Aynı senaryo farklı durumlara uygulanarak, alınan toplam yolun, aracın bekleme süresi ile ilişkisini irdelenmiş ve aşağıdaki sonuçları elde edilmiştir.

- Çok seri gelen talepleri karşılamada araçların talep noktasındaki bekleme süresinin toplam zaman etkisinin yok denecek kadar az olduğu görülmüştür. Taleplerin çok sık aralıklarla gelmesi, bekleme süresinin anlamını yitirmesine neden olmuştur.

- Çok seyrek aralıklarla gelen talepler için bekleme süresinin toplam alınan yolun azalmasına neden olmuştur. Talep noktasında bir veya daha fazla bir süre gelecek talebin beklenmesi aracın depoya dönmesi durumunda kat edeceği yolun ortadan kalmasına sebep olmuştur. Bekleme süresi bir sonra gelecek talebin beklemekte olan araca yakın olması durumu aracın bu talebi depoya gitmeden karşılamasına sebep olduğundan alınan yolun azalmasını sağlamıştır.

- Araç sayısının artması taleplerin karşılanma süresini azaltmıştır. Fakat bu durum, alınan toplam yolun bazı durumlarda artmasına sebep olmuştur. Araç sayısının fazlalığı ve bekleme süresinin fazlalığı alınan toplam yolun önemli ölçüde azalmasını sağlamıştır.

- Bekleme süresinin fazla olması, az olması durumunda aracın depoya dönmesi için kat edeceği yolun ortadan kalkmasını sağlamıştır. Gelen bir sonraki taleplerin

depoya dönmeden karşılanması sağlanmıştır. Bu durum aracın daha verimli bir şekilde ve az maliyetle kullanmasını sağlamıştır.

6.2 Öneriler

ARP'nin içerisine yolun durumu, tek ya da çift yönlülüğü, trafik durumu ve hava şartları da dahil edilebilir ve bu durum için bir çözüm aranabilir.

Yollardaki trafik yoğunluğu zaman ve zamana bağlı bulanık mantık terimleri ile ifade edilerek belirli üyelik dereceleri ile derecelendirilip çözüm yaklaşımında bulunulabilir.

ARP'nin uygulandığı coğrafi alanın sayısal haritası kullanılarak gerçeğe daha yakın çözüm yaklaşımları geliştirilebilir

Taleplerin geliş sıklığına ve yoğun geldiği bölgelere göre hangi araçların ve hangi depoların kullanılması gerektiğine karar verebilecek akıllı karar verme sistemi geliştirilebilir.

Öğrenme algoritmaları kullanılarak çok geniş bir coğrafi alan için, daha sonra gelecek taleplerin nerelerden gelebileceğini tahmin eden ve öğrenen bir algoritma geliştirilebilir.

KAYNAKLAR

Akman İ., C ile Veri Yapıları, Sas Bilişim Yayınları, 418 Sayfa, İstanbul, 2002.

Arora G., Aiaswamy B., Pandey N., Microsoft C# Professional Projects, Premimer Press İnc., 897 Sayfa, U.S.A., 2002.

Baker B.M, Ayechew M.A., A Genetic Algorithm for the Vehicle Routing Problem, Computers and Operations Research, Vol. 30, pp. 787-800, 2003.

Baker B.M.,Carlos A., Carreto C., A Visual Interactive Approach To Vehicle Routing, Computers & Operations Research, Volume 30, Issue 3, pp. 321-337, 2003.

Berger J., Barkaoui M., Braysy O., A Route-Directed Hybrid Genetic Approach For Vehicle Routing Problem With Time Windows, INFOR 2003, Vol. 41, pp. 179-94, 2003.

Blasum U., Hochstattler W., Application of the branch-and-cut method to the vehicle routing problem, Technical report, Universitat zu Koln, 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, Vol. 10, pp. 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, 2001.

Campbell A. M., Hardin J. R., Vehicle Minimization For Periodic Deliveries, European Journal of Operational Research, Vol. 165, Issue 3, pp. 668–684, 2005.

Chatterjee S., Carrera C., Lynch L.A., Genetic Algorithms and Traveling Salesman Problems, European Journal of Operational Research, Vol. 93, pp. 490–510, 1996.

Chu Ching W., A Heuristic Algorithm for the Truckload and Less-Than-Truckload Problem, European Journal of Operational Research, Volume 165, Issue 3, pp. 657-667, 2005.

Chu P.C., Beasley J.E., A Genetic Algorithm for the Generalised Assignment Problem, Computers & Operations Research, Vol. 24, pp. 17–23, 1997.

Clarke G. , Wright J.W., Scheduling Of Vehicles From A Central Depot To A Number Of Delivery Points, Oper. Research, Vol.12, pp. 568-581, 1964.

Cordeau J. F., Gendreau M., Laporte G, A tabu search heuristic for periodic and multi-depot vehicle routing problems, Networks, Vol. 30, pp.105-119, 1997.

Cordeau J.F., Gendreau M., Laporte G., Potvin J.Y., Semet F., A guide to vehicle routing heuristics, Journal of the Operational Research Society, Vol. 53, pp. 512-522, 2002.

Çölkesen R., Veri Yapıları ve Algoritmalar, Papatya yayıncılık, 424 Sayfa, İstanbul 2002.

Dantzig G.B., Ramser R.H., "The truck dispatching problem", Management Science, Vol. 6, pp. 80-91, 1959.

Eiselt H.A., Gendrau M., Laporte G., Arc routing problems: part I: The chinese postman problem, operations research, Vol 43, pp. 231-242, 1995.

Fisher M.L, Jaikumar R., Ageneralized Assignment Heuristic for Vehicle Routing, Networks, Vol. 11, pp. 109-24, 1981.

Gendreau M., Hertz A., Laporte G., New insertion and postoptimization procedures for the travelling salesman problem, Operations Research, Vol. 40, pp. 1086-1094, 1992.

Gendreau M., Laporte G., Frederic S., A Tabu Search Heuristic for the Undirected Selective Travelling Salesman Problem, European Journal of Operational Research, Vol. 106, Iss. 2-3, pp. 539-545, 1998.

Gendreau M., Laporte G., Potvin J.Y., Vehicle routing: modern heuristics, in Local Search in Combinatorial Optimizations, Edited by E. Aarts and J.K. Lenstra, Wiley, pp. 311--336, 1997.

Gillett B.E., Miller L.R., Aheuristic Algorithm for the Vehicle Dispatch Problem, Operations Research, Vol. 22, pp.340-349, 1974.

Goldberg D., Lingle R., Alleles, Loci and The Traveling Salesman Problem, Proceedings of the 9rst International Conference on Genetic Algorithms, 1985.

Golden B. L., Wong R. T., Capacited arc routing problem, networks, vol.11, pp. 305-315, 1981.

Golden B.L., Wasil E.A., Kelly J.P, Chao I-M., The Impact Of Metaheuristics On Solving The Vehicle Routing Problem:Algorithms, Problem Sets and Computational Results. In: Crainic TG, Laporte G, editors., Fleet management and logistics. Boston: Kluwer Academic Publishers, pp. 33-56, 1998.

Gondran M., Minoux M., Graphs and Algorithms, John Wiley and Sons Inc., 650 Sayfa, New York, 1984.

Gross J., Yellen J., Graph Theory and Its Application, CRC Pres, 576 Sayfa, Washington D.C., 1999.

Haghani A., Jung S., A dynamic vehicle routing problem with time-dependent travel times, Computers & Operations Research, Vol. 32, Issue 11, pp. 2959-2986, 2005.

Ichoua S., Gendreau M., Potvin J.Y., Diversion Issues in Real-Time Vehicle Dispatching, Transportation Science, Vol. 34, pp. 426-438, 2000.

Ichoua S., Gendreau M., Potvin J.Y., Vehicle Dispatching With Time Dependent Travel Times, European Journal of Operations Research, Vol. 144, pp. 379–396, 2003.

Jung S., Haghani A., A Genetic Algorithm for the Time Dependent Vehicle Routing Problem, Journal of Transportation Research Board, Vol. 1771, pp. 161–171, 2001.

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

Lau Hoong C., Sim M., Teo Kwong M., Vehicle Routing Problem With Time Windows and A Limited Number Of Vehicles, European Journal of Operational Research, Vol. 148, Iss. 3, pp. 559-569, 2003.

Minieka E., Optimization Algorithms for Networks and Graphs, New York: Marcel Dekker Inc., 356pp., 1978.

Osman I. H., Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem, Annals of Operations Research, Vol. 41, pp. 421-451, 1993.

Rochat Y., Taillard E. D., Probabilistic diversification and intensification in local search vehicle routing, Journal of Heuristics, Vol. 1, pp. 147-167, 1995.

Sipahioğlu A, Gezgin Satıcı ve Araç Turu Belirleme Problemleri İçin Yeni Alt Tur Engelleme Kısıtları, Osmangazi Üniversitesi, Fen Bilimleri Enstitüsü, Doktora Tezi, 80 sayfa, 1996.

Slavenko C., Application of The Chinese postman Problem Model To The Toronto Transportation Network Within GIS-Based Software, University of Toronto, Applied Science, Thesis of Master, 103 sayfa, 2001.

Taillard E. D., Parallel Iterative Search Methods for Vehicle Routing Problems, Networks 23, pp. 661-673,1993.

Tan K.C., Lee L.H., Ou K., Hybrid Genetic Algorithm In Solving Vehicle Routing Problems With Time Window Constraints, Asia-Paci9c Journal of Operational Research, Vol.18, pp.121-130, 2001.

Thangiah S.R., Petrovic P., Introduction To Genetic Heuristics And Vehicle Routing Problems With Complex Constraints, In: Woodru1 D.L., editor, Advanced in computational and stochastic optimization, logic programming, and heuristic search, Boston: Kluwer Academic Publishers, pp. 253-286, 1998.

Uchimura K., Sakaguchi H., Vehicle Routing Problem Using Genetic Algorithms Based On Adjacency Relations, In:Proceedings of the sixth Vehicle Navigation and Information Systems Conference, pp. 214–217, 1995.

Xu J., Kelly J., A Network Flow-Based Tabu Search Heuristic for the Vehicle Routing Problem, Transportation Science 30, pp. 379-393, 1996.

Yetkin B., Evsel Katı Atık Toplama Araçlarının Rotalanması: Denizli İlinde Bir Pilot Çalışma, Pamukkale Üniversitesi Fen Bilimleri Enstitüsü Yüksek Lisans Tezi, 2004.

EK-A

EKLER

Tek depo tek araç için 1-15 zaman aralığındaki sonuçlar tablosu

Yaklaşım	waiting time	Total time	Total distance	Total Demand
1. YAKLAŞIM	0	541	11456,92	373
	1	515	10832,94	373
	2	515	10832,94	373
	3	515	10832,94	373
	4	515	10832,94	373
2. YAKLAŞIM	0	541	11456,92	373
	1	515	10832,94	373
	2	515	10832,94	373
	3	515	10832,94	373
	4	515	10832,94	373

Tek depo tek araç için 15-30 zaman aralığındaki sonuçlar tablosu

Yaklaşım	waiting time	Total time	Total distance	Total Demand
1. YAKLAŞIM	0	1140	22572,97	373
	1	1144	20299,13	373
	2	1144	20299,13	373
	3	1144	20299,13	373
	4	1144	20299,13	373
2. YAKLAŞIM	0	1140	22572,97	373
	1	1144	20299,13	373
	2	1144	20299,13	373
	3	1144	20299,13	373
	4	1144	20299,13	373

Tek depo tek araç için 30-45 zaman aralığındaki sonuçlar tablosu

Yaklaşım	waiting time	Total time	Total distance	Total Demand
1. YAKLAŞIM	0	1824	31270,04	373
	1	1804	22456,46	373
	2	1804	22456,46	373
	3	1804	22456,46	373
	4	1804	22456,46	373
2. YAKLAŞIM	0	1824	31270,04	373
	1	1804	22456,46	373
	2	1804	22456,46	373
	3	1804	22456,46	373
	4	1804	22456,46	373

Tek depo tek araç için 45-60 zaman aralığındaki sonuçlar tablosu

Yaklaşım	waiting time	Total time	Total distance	Total Demand
1. YAKLAŞIM	0	2530	33687,15	373
	1	2510	22456,46	373
	2	2510	22456,46	373
	3	2510	22456,46	373
	4	2510	22456,46	373
2. YAKLAŞIM	0	2530	33687,15	373
	1	2510	22456,46	373
	2	2510	22456,46	373
	3	2510	22456,46	373
	4	2510	22456,46	373

Tek depo iki araç için 1-15 zaman aralığındaki sonuçlar tablosu

Yaklaşım	waiting time	Total time	Total distance	Total Demand
1. YAKLAŞIM	0	459	17824,94	373
	1	454	16819,32	373
	2	461	16242,73	373
	3	461	16242,73	373
	4	461	15935,66	373
2. YAKLAŞIM	0	455	18474,81	373
	1	454	16994,34	373
	2	436	15699,47	373
	3	436	15448,16	373
	4	475	15504,42	373

Tek depo iki araç için 15-30 zaman aralığındaki sonuçlar tablosu

Yaklaşım	waiting time	Total time	Total distance	Total Demand
1. YAKLAŞIM	0	1824	32592,90	373
	1	1804	26654,78	373
	2	1804	20668,86	373
	3	1804	18715,87	373
	4	1804	17266,34	373
2. YAKLAŞIM	0	1824	32592,90	373
	1	1804	26654,78	373
	2	1804	20668,86	373
	3	1804	18715,87	373
	4	1804	17758,47	373

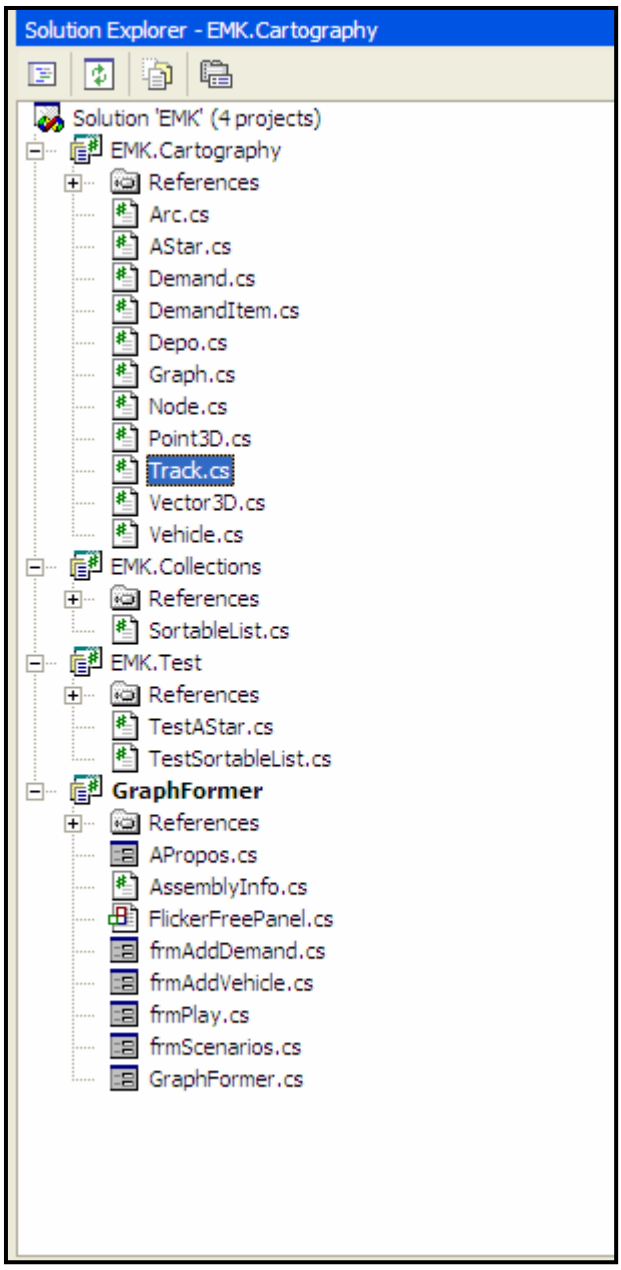
Tek depo iki araç için 30-45 zaman aralığındaki sonuçlar tablosu

Yaklaşım	waiting time	Total time	Total distance	Total Demand
1. YAKLAŞIM	0	1136	27182,72	373
	1	1136	23165,69	373
	2	1136	19555,88	373
	3	1141	18609,69	373
	4	1141	17387,26	373
2. YAKLAŞIM	0	1136	28602,46	373
	1	1136	23744,61	373
	2	1136	19555,88	373
	3	1141	18488,77	373
	4	1141	17266,34	373

Tek depo iki araç için 30-45 zaman aralığındaki sonuçlar tablosu

Yaklaşım	waiting time	Total time	Total distance	Total Demand
1. YAKLAŞIM	0	2530	34046,63	373
	1	2510	26654,78	373
	2	2510	20668,86	373
	3	2510	18715,87	373
	4	2510	17758,47	373
2. YAKLAŞIM	0	2530	34046,63	373
	1	2510	26654,78	373
	2	2510	20668,86	373
	3	2510	18715,87	373
	4	2510	17758,47	373

Simülasyonda kullanılan nesnelere ait sınıflar



ÖZGEÇMİŞ

Adı, soyadı: Şahin BAYZAN

Ana adı: Gülüzar

Baba adı: Mehmet

Doğum yeri ve tarihi: Araklı/Trabzon, 25.02.1971

Lisans eğitimi ve mezuniyet tarihi: Yakın Doğu Üniversitesi Bilgisayar Mühendisliği Bölümü, 1998

Çalıştığı yer: Pamukkale Üniversitesi Bilgisayar Mühendisliği Bölümü

Bildiği yabancı dil, aldığı belgeler: İngilizce

Mesleki etkinlikleri: "Araç Rotalarının En Kısa Yol Algoritmaları Kullanılarak Belirlenmesi" isimli yüksek lisans seminer sunumu