Gezgin Kargo Problemi

Traveling Cargo Problem

Cumali TOPRAK
Mühendislik Fakültesi, Bilgisayar Mühendisliği
Kocaeli Üniversitesi
Kocaeli, Türkiye
cumalitoprakk@gmail.com

Berkay Efe ÖZCAN
Mühendislik Fakültesi, Bilgisayar Mühendisliği
Kocaeli Üniversitesi
Kocaeli, Türkiye
berkayefeozcan@gmail.com

***Özetçe*—Bu projede gezgin kargo probleminin çözümlenmesi amaçlanmaktadır. Amaç kullanıcının tasarladığımız kullanıcı arayüzü ile girdiği şehirleri dolaşması ve başladığı noktaya mümkün olan en kısa yollardan geri dönmesidir. Biz de bu noktada mümkün olan en verimli algoritmalar üzerinde çalışmalar yaparak bu probleme çözüm bulduk. Kullandığımız algoritmalar ve detayları ilerleyen bölümlerde detaylıca ele alınacaktır. Tasarladığımız yazılım oldukça kullanıcı dostu bir arayüze sahip olup oldukça kolay anlaşılabilir. Ayrıca bu proje Java programlama dili kullanılmıştır.**

Anahtar Kelimeler—Gezgin Kargo Problemi, Kısa Yol Algoritmaları, Rota, Java.

***Abstract* — In this project, it is aimed to solve the traveler cargo problem. The aim is to navigate the cities that the user enters with the user interface we designed and return to the starting point in the shortest possible way. At this point, we found solutions to this problem by working on the most efficient algorithms possible. The algorithms we use and their details will be discussed in detail in the following sections. The software we designed has a very user-friendly interface and is easy to understand.**

Keywords— Traveling Cargo Problem, Shortest Path Algoritm, Route, Java.

I. GİRİŞ

Öncelikle belirtmek gerekir ki projemiz java dili kullanılarak gerçekleştirilmiş olup kullanıcı arayüzü kısmında javaFX kütüphanesi kullanılmıştır. Proje amaçlandığı gibi Kocaeli şehrinden başlayıp kullanıcının gidilmesini istediği şehirlere uğrayıp başlangıç şehri olan Kocaeli şehrine mümkün olan en kısa yollardan dönmektedir. Program varsa en kısa beş güzergahı[(1)], eğer bu kadar sayıda güzergah yoksa olan güzergahları listeleyecek şekilde programlanmıştır. Projemizi gerçekleştirirken birçok algoritma üzerinde çalıştık. Örneğin djikstra algoritması ile denediğimiz zaman bu algoritmanın maksimum şehir sayısı olan 10 şehrin güzergahlarını bulması yaklaşık olarak 20 saniye civarında oluyordu. Fakat performans açısından daha iyi bir algoritma olan Floyd-Warshall algoritmasını denediğimizde bunun 10 şehrin güzergahlarının yaklaşık 1-2 saniye içerisinde gerçekleştirdiğini gördük. Bu yüzden projemizde Floyd-Warshall algoritmasını kullandık. Projenin ve algoritmanın incelenmesi ilerleyen bölümlerde detaylıca ele alınacaktır.

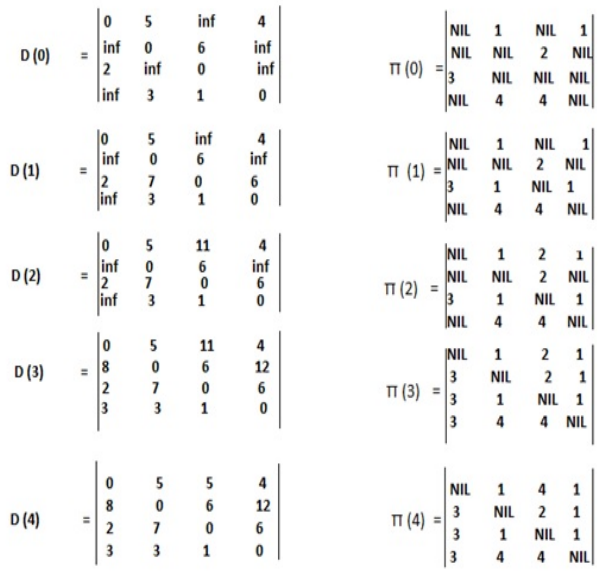
II. Kullanılan Yöntemler

A. Floyd-Warshall Algoritması

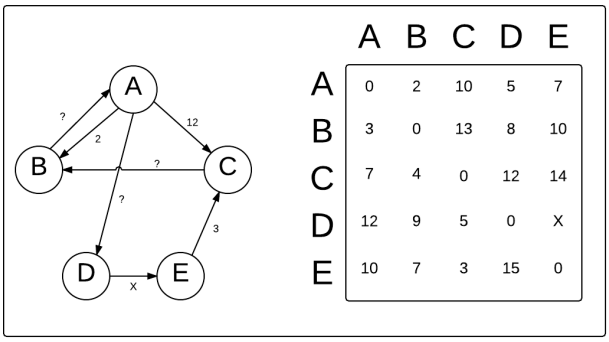
Algoritma basitçe bir grafta gidilebilecek düğümlerin **komşuluk listesini (adjecency list)** çıkararak bu düğümlere olan mesafeyi tutan bir masfuf (matris) üzerinden çalışır. Algoritmanın her adımında matris üzerinde çeşitli işlemler yapılarak en kısa yol bulunmaya çalışılır. Algoritma matris üzerinde çalışan ve düğümler arası mesafeleri her adımda güncelleyen bir algoritma olduğu için iteratif yazılabilir. Aşağıda bu işlemi gerçekleştiren bir kod parçası ve görseller verilmiştir.[1]

```
int i, j, k;
// Input data into dist, where dist[i][j] is the distance from i to j.
int dist[N][N];  // For some N

calculate_all_pairs_shortest_path() {
        int i, j, k;
    for ( k = 0; k < N; k++ )
        for ( i = 0; i < N; i++ )
            for ( j = 0; j < N; j++ )
    dist[i][j] = min( dist[i][j], dist[i][k] +
        dist[k][j] );
            }

int get_distance(int u, int v) {
        return dist[u][v];
            }
```

Şekil-1: Algoritmanın uygulanışını gösteren kod parçası.



Şekil-2: Floyd-Warshall algoritmasının işleyişinin aşamaları.

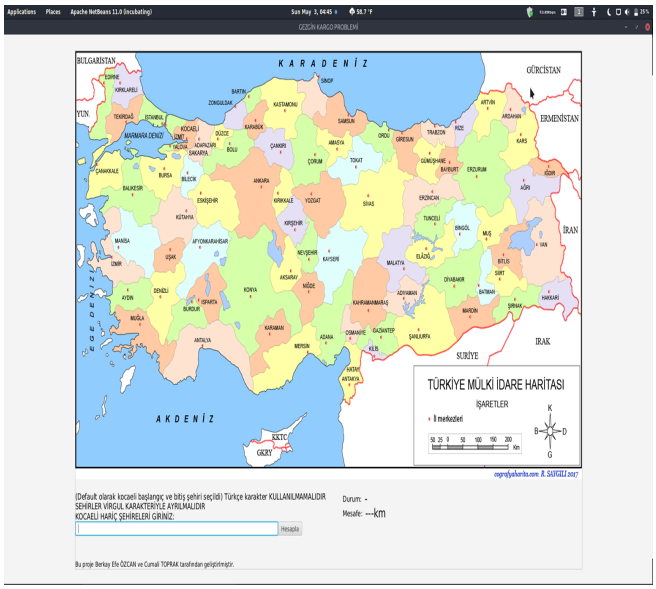


Şekil-3: Floyd-Warshall algoritmasının tablo üzerinde gösterimi.

Biz de bu algoritmayı projemize uyarlarken öncelikle 81*81 matris tanımladık. Bu matrisin değerlerini öncelikle her şehrin sahip olduğu komşulara olan mesafeleri matrise geçirdik. Gidilmeyen komşularına ise maksimum değerini atadık. Matrise bu değerler atandıktan sonra algoritma çalıştığı anda var olan komşuluklar üzerinden gidilemeyen diğer tüm komşulara gidilebilmesi sağlandı(all pairs shortest path). Matris değerleri dinamik olarak değişirken, matrisin değerleri güncellendikçe şehir-komşuluk verilerimizi de dinamik olarak güncelledik. Algoritma işlemini tamamlandığı zaman tanımladığımız veri yapılarında her şehirden diğer 80 şehre gidecek en kısa rotalar ve mesafeleri bulunmuş oldu.

B. Permutasyon

Java teknolojilerinden faydalanarak programcıların masaüstü ve web uygulamaları geliştirmesine olanak sağlayan ileri seviyeli yazılım teknolojisidir. JavaFX de ileri seviye içerik, ses, grafik video içeren, etkileşimli modern uygulamalar oluşturulmasına ve dağıtılmasına olanak sağlayan bir yapıdır. [2] Projede bulunan güzergahların harita üzerinde oklarla gösterilmesi gerektiğinden biz de bu amaçla çok daha kullanışlı olan javaFX'i kullandık. Aşağıda program arayüzü gösterilmektedir.



Şekil-4: JavaFX kullanılarak gerçekleştirilen grafik arayüzü.

C. Permutasyon Metodu

Bu fonksiyon yardımı ile kullanıcı gidilecek şehirleri girdiği zaman ideal güzergahları bulmak için kaç farklı yol denenmesi gerektiği bulunur. Bulunan bu yollar sırası ilgili metod tarafından denenip ideal yollar bulunur. Bu metod recursif olarak yapılmıştır.

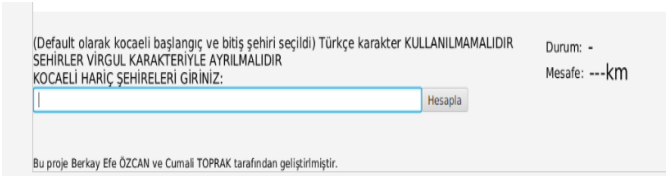
D. Dosya Bilgileri

Bulunan güzergahların harita üzerinde gösterilmesi gerektiğinden dolayı dosyada bazı değişlikleri gerçekleştirdik. Örneğin her şehir için kullancağımız haritadan konum bilgisini alıp dosyaya manuel olarak ekledik. Bu sayede güzergahlar bulunduğu zaman her şehri oklarla harita üzerinde kolayca gösterebildik. Ayrıca dosyadan verileri çekerken BufferedReader objesini kullandık.

III. Programın Çalıştırılması

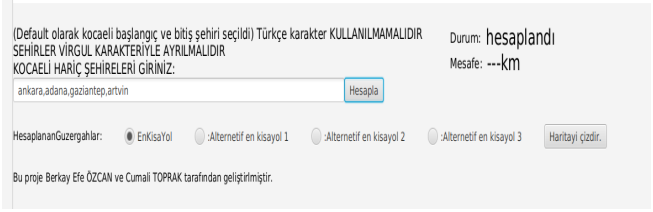
Uygulamayı çalıştırdığımız zaman aşağıda da resmini gösterdiğimiz bir arayüz ile karşılaşacaksınız. Burada bir Türkiye haritası ve gidilecek şehirleri girebileceğiniz textbox bulunmaktadır. Textbox'a şehirleri sırası ile virgüllerle ayırarak girmeniz gerekmektedir. Ayrıca Kocaeli şehrinden başlayacağı bilindiği için ayrıyeten Kocaeli şehrin başa yazmanıza gerek yoktur. Kocaeli hariç en fazla şehir 9 şehir girebilirsiniz. Bundan fazla da şehir girilebilir fakat bu biraz zaman alacaktır. Proje değerlendirilmesinde de belirtildiği üzere en fazla on şehir girilmesi durumunda program bu işlemi 1-2 saniye gibi kısa bir süre içerisinde

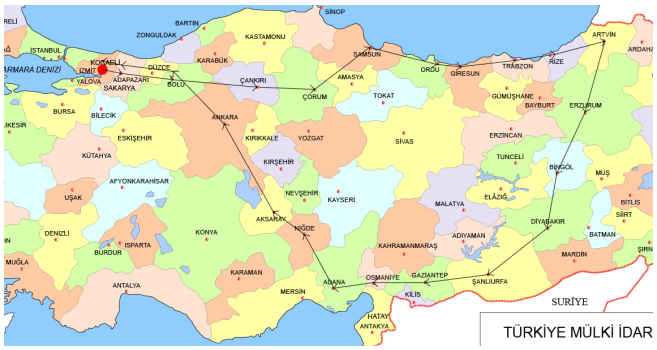
gerçekleştirmektedir. -Bu işlem tamamlandıktan sonra sonuçlar ayrıca output.txt adlı dosyaya yazdırılmaktadır.-



Şekil-5: Gidilecek şehirlerin girileceği textbox.

Gidilmesi istenen şehirler kullanıcı tarafından textboxa girildikten sonra butona basıldığı anda algoritma çalışmaya başlar ve en az 1 en fazla 5 güzergah gösterilir. Ve kullanıcı uzaklıklarına göre sıralanmış bu yollardan hangisini seçerse bu yol harita üzerinde oklarla gösterilir. Bu yolları seçme işlemi radio button'ları ile gerçekleştirilmiştir. Aşağıya programı test aşamasının görselleri eklenmiştir.





Şekil-6:Alternatif yollardan herhangi birini gösteren harita.

Yukarıdaki şekilde kullanıcı girdiği şehirlere göre 5 alternatif yol oluşturulmuştur. Kullanıcı alternatif yollardan birini seçtiği anda yukarıda da gösterildiği gibi harita üzerinde oklarla bu güzergah gösterilmiştir. Kullanıcı dilediği taktirde diğer tüm yolları seçebilir.

IV. HARİTANIN BOYUT UNUN AYARLANMASI VE AYARIN DOSYADAN OKUNMASI

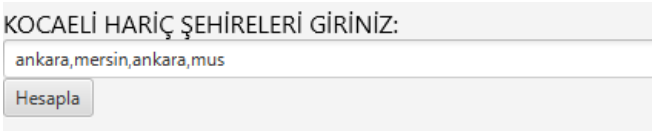
Projemizde 1,920 x 1,080 ve 1,280 x 720 piksel çözünürlüğe sahip bilgisayarlar için haritayı boyutlandırdık. Bu boyutlama işlemini 'cozunurlukAyar.txt' nin içindeki bilgiye göre yapılmaktadır. Eğer dosyada '**hd**' yazıyorsa harita, 1,280 x 720 piksel çözünürlükteki bilgisayar ekranına sığacak şekilde kullanıcı arayüzüne eklenir. İlgili dosyada **hd** kelimesi dışında bir şey yazıyorsa harita büyük boyutta kullanıcı arayüzüne eklenir.

V. KARŞILAŞILMASI MUHTEMEL HATALAR

E. A.Kullanıcı Arayüzündeki Hatalar:

Kullandığımız floyd-warshall algoritmasından dolayı kullanıcı şehirlerin girileceği textbox a iki tane aynı şehir ismini girerse program hatalı çıktı üretmektedir. Örneğin;

Kullanıcı text boxa aşağıdaki girdileri girmiş olsun.



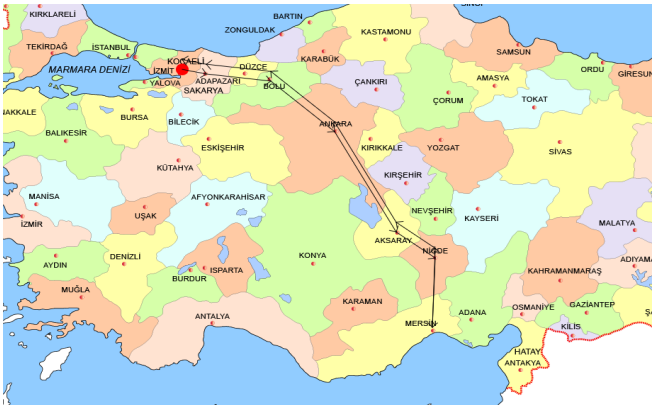
Şekil-7:Yanlış şehir girdisi.

Şekil-8:Yanlış giriş sonucu oluşan güzergah.

Şekil 7 de iki kere aynı şehir tekrar etmiş olup şekil-8 de de görüldüğü gibi hatalı bir sonuç ortaya çıkmıştır. Ayrıca şehir isimlerinin Türkçe karakterler kullanılarak[(2)] girilmesi taktirde kullanıcıya hata mesajı verilmektedir.

Harita çizdirilirken bir şehire iki kere uğranmışsa y pikselleri birbirine oldukça yakın oldursa tek bir ok gibi gözükebilir. Örneğin Kocaeli,Ankara,Mersin programa girdi olarak girildiğinde sonuç:

[KOCAELI, SAKARYA, BOLU, ANKARA, AKSARAY, **NIGDE, MERSIN,NIGDE** ,AKSARAY,ANKARA ,BOLU,SAKARYA,KOCAELI] olmaktadır. Niğdeye iki kere uğrandığından dolayı aşağdaki gibi bir çıktı oluşur.



Şekil-9: Bir şehre iki kere uğranmasın sonucu oluşabilecek çıktı.

F. B.Jar Dosyası Çalıştırılacağında Oluşabilecek Hatalar:

Eğer javaFX kütüphanesi çalıştırma ortamında kurulu değilse java projenin main klasını bulamadığını söyler. Projenin jar dosyası üzerinden çalışabilmesi için jar dosyasının içinde bulunduğu dizinde sehirlerVeMesafeler.txt, harita.png, icon.png ve cozunurlukAyarı.txt dosyalarının olması gerekmektedir. Aksi halde program hata verecektir.

VI. GERÇEKLEŞTİRİLEN TESTLER VE DEĞERLENDİRME

Projeyi test ederken proje değerlendirmesinde de bahsedildiği gibi en az üç en fazla şehir girilip test edilmiş olup bunu toplam 1-2 saniye içerisinde gerçekleştirmektedir. Fakat daha fazla şehir girildiği zaman bu süre artmaktadır. Ayrıca 11-12 tane şehir girildiğinde programda java collectors limit exceeded hatası oluşabilmektedir. İlerleyen dönemde threadlerle proje refactor edildiği taktirde 1-2 saniyenin altında da programın çalışabileceğini düşünüyoruz.

VII. ÖZET VE SONUÇLAR

Özetle, proje amaçlandığı gibi Kocaeli şehrinden başlayıp istenilen şehirlere uğranıp tekrar Kocaeli şehrine gelecek şekilde programlanmış olup gayet makul sürede bu işlemi gerçekleştirmektedir. Projede Floyd-Warshall algoritması kullanılmıştır. Proje tamamlandıktan sonra belirli sayıda şehirler girilip ayrı ayrı program test edilmiştir. Program sade bir kullanıcı arayüzüne sahip olup kolayca anlaşılabilmektedir. Bu proje hem veri yapılarını anlamamıza hem de java dilini doğru etkili kullanılmasına katkı sağlamıştır. Bu projeyi github üzerinden yayımlacağız. Katkı yapmak veyahut kaynak kodları kullanmak isteyenler dilediği şekilde kullanabilirler.

Kaynakça

[1]http://bilgisayarkavramlari.sadievrenseker.com/2009/05/29/floyd-warshall-algoritmasi/

[2] http://sercancetin.com/java-fxe-genel-bakis-nedir-ne-degildir/

[3] https://www.youtube.com/watch?v=oNI0rf2P9gE

[4] https://www.baeldung.com/java-array-permutations

[5]https://stackoverflow.com/questions/41353685/how-to-draw-arrow-javafx-pane

[6]https://www.javatpoint.com/javafx-tutorial

[7]https://www.kgm.gov.tr/Sayfalar/KGM/SiteTr/Uzakliklar/illerArasiMesafe.aspx

DİPNOTLAR

1-İller arası mesafeler verisi Karayolu Genel Müdürlüğü'nün sitesinden çekilmiştir.

2-Türkçe karakterler, şehir isimleri dosyadan okutulduğu ve dosyadaki şehirlerin isimleri Türkçe karakter kullanılmadan yazıldığı için kullanılamamaktadır.