T.C.
ONDOKUZ MAYIS ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ
İSTATİSTİK ANA BİLİM DALI

AĞ OPTİMİZASYONU MODELLERİ: PYTHON İLE UYGULAMALAR

Yüksek Lisans Tezi

Bagul OVLYAKULYYEVA

Danışman

Doç. Dr. Talat ŞENEL

.

SAMSUN
2023

TEZ KABUL VE ONAYI

Bagul OVLYAKULYYEVA tarafından, **Doç. Dr. Talat ŞENEL** danışmanlığında hazırlanan **"AĞ OPTİMİZASYONU MODELLERİ: PYTHON İLE UYGULAMALAR"** başlıklı bu çalışma, jürimiz tarafından 2.8.2023 tarihinde yapılan sınav sonucunda oy birliği ile başarılı bulunarak Yüksek Lisans Tezi olarak kabul edilmiştir.

	Unvanı Adı Soyadı **Üniversitesi** **Ana Bilim/Ana Sanat Dalı**	**Sonuç**
Başkan	Doç. Dr. Esin AVCI Giresun Üniversitesi Fen Edebiyat Fakültesi İstatistik Ana Bilim Dalı	☒ Kabul ☐ Ret
Üye	Doç. Dr. Talat ŞENEL (Danışman) Ondokuz Mayıs Üniversitesi Fen Fakültesi İstatistik Ana Bilim Dalı	☒ Kabul ☐ Ret
Üye	Doç. Dr. Erol TERZİ Ondokuz Mayıs Üniversitesi Fen Fakultesi İstatistik Ana Bilim Dalı	☒ Kabul ☐ Ret

Bu tez, Enstitü Yönetim Kurulunca belirlenen ve yukarıda adları yazılı jüri üyeleri tarafından uygun görülmüştür.

Prof. Dr. Ahmet TABAK
Enstitü Müdürü

BİLİMSEL ETİĞE UYGUNLUK BEYANI

Hazırladığım Yüksek Lisans tezinin bütün aşamalarında bilimsel etiğe ve akademik kurallara riayet ettiğimi, çalışmada doğrudan veya dolaylı olarak kullandığım her alıntıya kaynak gösterdiğimi ve yararlandığım eserlerin Kaynaklar'da gösterilenlerden oluştuğunu, her unsurun enstitü yazım kılavuzuna uygun yazıldığını ve TÜBİTAK Araştırma ve Yayın Etiği Kurulu Yönetmeliği'nin 3. bölüm 9. maddesinde belirtilen durumlara aykırı davranılmadığını taahhüt ve beyan ederim.

Etik Kurul Gerekli mi ?

Evet ☐ (Gerekli ise ekler kısmına ekleyiniz)

Hayır ☒

… /.../ 20...
Bagul OVLYAKULYYEVA

TEZ ÇALIŞMASI ÖZGÜNLÜK RAPORU BEYANI

Tez Başlığı: AĞ OPTİMİZASYONU MODELLERİ: PYTHON İLE UYGULAMALAR

Yukarıda başlığı belirtilen tez çalışması için şahsım tarafından 04/07/2023 tarihinde intihal tespit programından alınmış olan özgünlük raporu sonucunda;

Benzerlik oranı : % 16

Tek kaynak oranı : % 2 çıkmıştır.

04 /07 / 2023
Doç. Dr. Talat ŞENEL

ÖZET

AĞ OPTİMİZASYONU MODELLERİ: PYTHON İLE UYGULAMALAR

Bagul OVLYAKULYYEVA
Ondokuz Mayıs Üniversitesi
Lisansüstü Eğitim Enstitüsü
İstatistik Ana Bilim Dalı
Yüksek Lisans, Ağustos/2023
Danışman: Doç. Dr. Talat ŞENEL

Ağ, bir noktalar kümesini ve bu noktaları ikili olarak bağlayan çizgiler kümesini içerir. Ağlar, değişik ortamlarda ve çeşitli şekillerde ortaya çıkabilirler. Günümüzün rekabetçi iş ortamında, firmaların sınırlı kaynaklarını en verimli şekilde kullanmaları giderek daha da önem kazanmaktadır. Bu durum genellikle karar verme süreçlerinde karı maksimize etmek ya da maliyetleri minimize etmek amacıyla ortaya çıkar. Gemi, bina, uçak, araç yapımı; en kısa yol, kablo, hat, boru döşeme gibi problemler, ağ modellerine dönüştürülerek çözülebilecek türden modellere örnektir. Bu tür problemler doğrusal programlama kullanılarak da çözülebilir, ancak ağ modelleriyle çözüm sağlamak genellikle daha etkilidir. Ağ optimizasyonu modelleri, en düşük maliyetli akış problemi, en kısa yol problemi, maksimum akış problemi, ulaştırma problemi, atama problemi, kapsayan ağaç problemi ve CPM/PERT modelleri şeklinde sınıflandırılmaktadır. Optimizasyon modellerinin temel amacı, katılımcılara gerçek hayatta matematiksel modellemeye dayalı uygulamalar yapabilme becerisi kazandırmak için yöneylem araştırması tekniklerini kullanmaktır. Bu bağlamda, ulaştırma ve lojistik problemleri gibi yöneylem araştırması tekniklerinin bir parçası olan konular genellikle ağ akış problemleri olarak sınıflandırılmaktadır ve incelenmektedir. Bu çalışmanın "Literatür Araştırması" bölümünde literatürde yer alan ilgili çalışmalar özetlenmektedir. Sonraki bölüm olan "Materyal ve Metot" bölümünde tez kapsamında önerilen yaklaşımlar, kullanılan model ve algoritmalar açıklanmakta, takip eden "Uygulama ve Bulgular" bölümünde ise önerilen yaklaşımların uygulanmasıyla elde edilen sonuçlar sunulmaktadır. Son kısımda yer alan "Sonuç ve Öneriler" bölümü tez çalışmasının genel bir değerlendirmesini sunmaktadır. Bu çalışmada Ağ optimizasyonu modelleri incelenip, bunların değişik alanlarda uygulanabilirliği hakkında bilgiler verilecektir. Ayrıca ağ optimizasyonu konusunda literatürde mevcut olan çok sayıda uygulama ele alınmıştır. Bu uygulamalar, PYTHON programlama dili kullanılarak da çözümlenmiş ve sonuçlar karşılaştırılarak önerilerde bulunulmuştur.

Anahtar Sözcükler: Ağ optimizasyonu, Optimizasyon, Modelleme, Python

ABSTRACT

NETWORK OPTIMIZATION MODELS: APPLICATIONS WITH PYTHON

Bagul OVLYAKULYYEVA
Ondokuz Mayıs University
Institute of Graduate Studies
Department of Statistic
Master, August/2023
Supervisor: Assoc. Prof. Dr. Talat ŞENEL

A network contains a set of points and a set of lines that connect these points pairwise. Networks can emerge in different settings and in different ways. In today's competitive business environment, it is increasingly important for firms to make the most efficient use of their limited resources. This often occurs in decision-making processes to maximize profits or minimize costs. Problems such as ship, building, airplane, vehicle construction; shortest road, cable, line, pipe laying are examples of the kind of models that can be solved by converting them into network models. Such problems can also be solved using linear programming, but it is often more efficient to provide solutions with network models. Network optimization models are classified as minimum cost flow problem, shortest path problem, maximum flow problem, transportation problem, assignment problem, spanning tree problem and CPM/PERT models. The main objective of optimization models is to use operations research techniques to provide participants with the ability to make real-life applications based on mathematical modeling. In this context, topics that are part of operations research techniques, such as transportation and logistics problems, are often classified and studied as network flow problems. The "Literature Review" section of this study summarizes the relevant studies in the literature. The next section, "Materials and Methods", describes the approaches, models and algorithms proposed in this thesis, followed by "Application and Results", which presents the results obtained by applying the proposed approaches. The "Conclusion and Recommendations" section in the last part presents an overall evaluation of the thesis. In this study, Network optimization models will be examined and information about their applicability in different fields will be given. In addition, a large number of applications available in the literature on network optimization are discussed. These applications are also analyzed using the PYTHON programming language and the results are compared and recommendations are made.

Keywords: Network optimization, Optimization, Modelling, Python

ÖN SÖZ VE TEŞEKKÜR

Yüksek lisans eğitimim süresince beni destekleyen, bilgisiyle, yardımıyla ve güveniyle yanımda olan danışman hocam Doç. Dr. Talat ŞENEL'e sonsuz teşekkürlerimi sunuyorum.

Lisans ve Yüksek Lisans eğitim aşamalarımda benden yardımlarını esirgemeyen, yol gösteren değerli İstatistik Bölümü Öğretim Üyeleri hocalarıma teşekkürlerimi sunuyorum.

Anneme, babama eğitimim boyunca sağladığı destek ve güven için derin bir teşekkür etmek istiyorum. Sizlerin benim yanımda olduğunuzu bilmek, her adımda benim için büyük bir motivasyon oldu. Size minnettarlığımı ifade etmek için yeterli kelimeler bulmak zor. Sizin sevgi dolu destekleriniz ve güveniniz sayesinde büyüyüp gelişebildim. Sizin sayenizde bugün buradayım ve hayallerime doğru ilerliyorum. Sizlere sonsuz teşekkürlerimi sunuyorum.

Bu tez çalışmamı sevgili kardeşim Abdurahmana'a adıyorum.

Bagul OVLYAKULYYEVA

İÇİNDEKİLER

SİMGELER VE KISALTMALAR

DPM	: Doğrusal Programlama Modeli
MPM	: Matematiksel Programlama Modeli
LPM	: Lineer Programlama Modeli
EKY	: En Kısa Yol
MYO	: Kapsayan Ağaç Problemi
CPM	: Critical Path Mehod – Kritik Yol Yöntemi
ES	: En Erken Başlama Süresi
LS	: En Erken Tamamlanma Süresi
LS	: En Geç Başlama Süresi
LF	: En Geç Tamamlanma Süresi
PERT	: Program Değerlendirme ve Gözden Geçirme Tekniği
AB	: Ara Bolluk Süresi
BB	: Bağımsız Bolluk Süresi
SB	: Serbest Bolluk Süresi
TB	: Toplam Bolluk Süresi
Tij	: Faaliyet süresi
i,j	: Olaylar
mm	: masraf maili
M_y	: Hızlandırılmış Maliyet
M_n	: Normal Maliyet
T_n	: Normal Tamamlanma Süresi
T_y	: Yoğun Çalışma Süresi
t_a	: En erken tamamlanma süresi
t_b	: En geç tamamlanma süresi
t_m	: Mühtemel süre
t_e	: Beklenen Süre
Σ	: Standart Sapma
σ^2	: Varyans
$\bar{x}$	: Beklenen değer
Z	: Tablo Değeri
Y	: Projenin Tamamlanması İstenen Süre
μ	: Projenin Beklenen Tamamlanma Süresi

ŞEKİLLER DİZİNİ

TABLOLAR DİZİNİ

1.GİRİŞ

Ağ, temel olarak çeşitli cihazları birbirine bağlayan ve veri paylaşımını sağlayan bir teknolojidir. En güzel örneklerinden biri ise internettir. Matematiksel programlama (MP), bir kişinin veya bir organizasyonun hedeflerine ulaşmak için kısıtlı kaynakları en optimal veya en etkin şekilde kullanmasına karar vermesini sağlayan bir yönetim bilimidir. Bu nedenle matematiksel programlama genellikle optimizasyon olarak da adlandırılır. Optimizasyon modellerinin amacı yöneylem araştırması tekniklerini kullanarak katılımcılara gerçek hayatta matematiksel modellemeye dayalı uygulamalar yapma becerisi kazandırmaktır. Yöneylem araştırması tekniklerinden olan ulaştırma ve lojistik problemleri genellikle literatürde ağ akış problemleri olarak sınıflandırılan kapsam içinde incelenir. Proje, birbiriyle ilişkili faaliyetlerin zaman ve kaynak harcama gereksinimleri dikkate alınarak bir araya getirildiği bir süreçtir. Projelerin uygun şekilde yürütülmesi ve tamamlanması sürecinde doğrusal programlama modelinin konusu olan CPM ve PERT metotların kullanılması projede en etkili sonuçlara ulaşmak için oldukça yararlıdır. CPM ve PERT metodlarının amacı, çizelgeleme faaliyetlerine analitik anlamlar kazandırmaktır. CPM, faaliyet sürelerini deterministik olarak yani, sürelerin önceden belirlenmiş ve kesin olarak kabul ederken, PERT ise bu sürelerin olasılıklı olduğunu kabul eder.

AR-GE projesi veya inşaat projesi gibi büyük projelerde, işlerin akışının grafik olarak gösterilmesinde ağlar oldukça yararlıdır. Yöneticiler çoğunlukla büyük bir projenin planlanması, iş plan çizelgesi hazırlanması, iş akışının kontrol edilmesi ve izlenmesi ile yükümlüdürler. Projeyi oluşturan aktivitelerin hepsi bitirildiğinde proje bitirilmiş olur. Büyük projelerde hangi aktivitelerin yer alacağı, her birinin hangi sırada tamamlanması gerektiği, tahmini bitirilme sürelerinin bilinmesi veya belirlenmesi gerekir. Bu tür bilgiler projeyi yürütecek ekipteki elemanlar tarafından belirlenir. Projelerin uygun şekilde yürütülmesi ve tamamlanması sürecinde PERT/CPM olarak adlandırılan yöntemlerin kullanılması proje yöneticisinin işlerini büyük ölçüde kolaylaştıracaktır.

Çalışmanın ilerleyen bölümlerinde, “Literatür Araştırması” bölümünde literatürde yer alan ilgili çalışmalar özetlenmektedir. Sonraki bölüm olan “Materyal ve Metot” bölümünde tez kapsamında önerilen yaklaşımlar, kullanılan model ve algoritmalar açıklanmakta, takip eden “Uygulama ve Bulgular” bölümünde ise

önerilen yaklaşımların uygulanmasıyla elde edilen sonuçlar sunulmaktadır. Son kısımda yer alan "Sonuç ve Öneriler" bölümü tez çalışmasının genel bir değerlendirmesini sunmaktadır. Bu çalışmada, literatürde mevcut olan günlük hayattaki problemlerden ağ yapısına uygun olan problemler ele alınıp çözümleri yapılacak ve yorumlanacaktır. Çözümlemeler özellikle PYTHON dilinde yapılmaya çalışılarak literatüre katkı da sağlanacaktır.

1.1.Literatur taraması

Ağ optimizasyonu modelleri konusu hakkında yapılmış çok sayıda bilimsel çalışma mevcuttur.Bunlardan bazıları aşağıda sunulmuştur.

Duraid Haqi AL-MOMAYEZ.D (2023) çalışmasında Türkiye Taşkömürü Kurumu internet sitesindeki bilgilerden derlenen veri setini kullanılarak, ulaştırma modelini kurmuştur ve Python ve R programlama dillerinde çözümler yapmıştır

Ulu.F (2023) çalışmasında karınca kolonisi optimizasyonu (KKO) algoritması ile depo rota planlaması konusu ele almıştır. Bu çalışmada, en kısa yolların ve rotanın belirlenmesi amacıyla Floyd-Warshall ve KKO algoritmaları kullanılmıştır.

Cingöz.K, ve Gürgen.E (2023) çalışmalarında; hızlı yiyecek sektöründe hizmet veren bir firmanın ürün üretim sürecini detaylı bir şekilde ele almışlardır.

Gürsoy.H, ve Duman.E (2022) çalışmalarında; iki amaçlı en kısa yol problemi ve bu problemi çözmek için literatürde geliştirilen yöntemleri incelemişler ve karşılaştırmışlar.

Bülbül.E (2022) çalışmasında, 64 düğümlü bir çizgeyi 3 boyutlu ortama uyarlamıştır. Bu ortamda, Dijkstra algoritması ve A* algoritmasını, çeşitli fiziksel faktörlerin etkisi altında simüle edilerek, bu faktörlerin negatif etkileri gözlemlemiş ve algoritmanın adaptasyonunu sağlayacak modifikasyonlar yapmıştır.

Arman.K, ve Tuş.A (2022) çalışmalarında; lojistik alanında faaliyet gösteren bir firmanın Düzce-Artvin arası dağıtım faaliyetlerinde toplam mesafe ve süreyi minimum kılan rota belirleme problemini ele almışlardır.

Akay.M ve Tuş.A (2022) araştırma makalesi çalışmasında; Minimum Yayılan Ağaç problemini Denizli şehir içi ulaşımda diğer taşıma sistemlerinin arasındaki etkileşimlerin sağlanması amacı ile uygulamıştır.

Keskin.B, ve Özcan.E (2022) çalışmalarında; lojistik merkezlere trenlerin en kısa sürede ve en kısa yoldan ulaşımını sağlama amacı güdülmüştür. Bu problem, Floyd-Warshall algoritması kullanılarak Python programlama diliyle çözülmüştür.

Taşkıran.M (2022) çalışmasında; okulun ders programı hazırlanmasında ders çizelgeleme-atama problemi kullanmıştır.

Dhulkefl.E, Durdu.A ve Terzioğlu.H (2020) çalışmalarında; İHA'lar için engellerden kaçınarak ve hedef noktasına en kısa yolu belirlemek amacıyla Dijkstra algoritması kullanılmıştır. İHA'nın belirli yüksekliklerde uçtuğu varsayılarak 2D ve 3D modellemeler gerçekleştirmişler. Deneyde Dijkstra algoritmasının sonuçları 3D modelleme daha iyi sonuca ulaşılmıştır.

Bayzan.Ş, Çetin.M ve Alper Uğur.A(2020) çalışmalarında; bir aracın başlangıç noktasından hedef noktasına en etkili şekilde seyahat etmesi için yapılan planlamada en kısa yol yaklaşımı kullanılmış ve taleplerin karşılanması simülasyonu gerçekleştirmişlerdir. Model için bir noktadan başka bir noktaya gitmek için en kısa yolun tespit edilerek, en uygun güzergahın belirlenebileceğini göstermiştir.

Bakışkan.E (2019) yüksek lisans tez çalışmasında; proje kapsamında, bir Karadeniz tipi balıkçı gemisinin inşa sürecindeki tüm faaliyetleri kontrol altına alınmıştır. Kritik ve kritik olmayan faaliyetleri belirleyerek, projeyi yöneten yetkilinin dar boğaz oluşturan faaliyetleri ve bu faaliyetlere yönelik alınan önlemlerin görebilmesini sağlamıştır.

Akşehir.K (2019) çalışmasında; (GSP) gezgin satıcı probleminin çözümünde kullanılan sezgisel bir yöntem olan karınca kolonisi algoritmasının çözüm performansına etki eden parametre değerlerini belirlemeyi amaçlamıştır. Yapılan deneyler sonucunda, uygun parametre değerlerini kullanılarak karınca kolonisi algoritmasıyla Türkiye'nin bütün illeri arasındaki en kısa mesafe değeri belirlenebilmektedir.

Aydın.S (2017) çalışmasında; Türkiye'de enerji kaynaklarının etkinliği ölçülmüştür. Çalışma sonuçlarına göre, elektrik dağıtım bölgelerini ve şehirlerin elektrik enerji etkinliklerini belirlemiştir.

Mazlum.M (2014) yüksek lisans tez çalışmasında; proje yönetiminin önemli bir basamağını oluşturan klasik PERT ve CPM teknikleri ile bulanık proje yönetiminde kullanılan bulanık PERT ve CPM tekniklerini, online internet şubesi düşüncesiyle projesinin planlanması için kullanmış ve sonuçları karşılaştırmıştır.

Berberler.M, Uğurlu.O ve Kızılateş.G (2012) araştırma makalesi çalışmalarında; Macar algoritması olarak bilinen bir yöntem kullanılarak atama problemlerine çözüm getirmiştirler.

Yetgin,M (2011) çalışmasında; Türkiye'deki yük taşımacılığında en kısa turu belirleme konusunda, karayolu ve demiryolu arasındaki etkileşimlerin sağlanması üzerinde durmuştur. Çalışmada, karayolları ve demiryollarının mevcut durumları değerlendirerek, demiryolu ile ulaşımın mümkün olduğu bölgelere demiryolu kullanılarak, demiryolu ulaşımının mümkün olmadığı bölgelere ise karayolu kullanılarak tüm ülkede en kısa tur belirlemiştir. Bu çalışma, yük taşımacılığındaki ulaşım yöntemlerinin optimal bir şekilde entegre edilerek en etkili rota planlamasının yapılmasına katkı sağlamaktadır.

Dener.M, Akcayol.M, Toklu.S ve Bay.Ö (2011) çalışmalarında, en kısa yol problemi için genetik algoritma tabanlı bir yöntem geliştirmiştirler. Bu yöntem, zamana bağlı olarak değişen yol maliyetlerinin sahip olduğu 43 düğümlü bir graf üzerinde en kısa yolun bulunmasında kullanılmıştır. Geliştirilen algoritma, yol maliyetlerinin dinamik olarak değiştiği senaryolarda etkili bir çözüm sunmaktadır.

Mercangöz.B (2010) çalışmasında; merkezi Slovenya'da bulunan Türkiye'de beyaz eşya satışı yapan uluslararası bir firmanın Türkiye'deki dağıtım ağının tasarlanması gerçekleştirilmiştir.

Karslı.N (2010) Doktora tezi çalışmasında maliyeti ve yönü olan toplam beş farklı ağ üzerinde üç farklı Genetik Algoritma ve Yönlendirilmiş Yerel Arama algoritması kullanılarak En Kısa Yol problemi çözülmüştür.

İstanbullu.H (2009) Doktora tezi çalışmasında; büyük hacimli görüntü dosyalarının bağımsız bir bilgisayar ağında taşınması, saklanması ve geri çağrılması için uygulama yapmıştır.

Karadeniz.C.Ö (2007) yüksek lisans çalışmasında; Kritik Yol Metodu ve Program Değerlendirme ve Gözden Geçirme Tekniğini kullanarak, Halkalı-Gebze arasındaki mevcut demiryolu hattının iyileştirilmesi ve kapasitesinin artırılması yönünde çalışma yapmıştır.

Duran.C (2007), matbaa üretim aşamalarında ilk kez CPM-PERT tekniklerini uygulayan bir çalışma yapmıştır.

Sarıca.İ (2006) yüksek lisans tez çalışmasında; Buz Pateni İnşaatı Projesi için PERT yönteminin kullanılabilirliğini ve etkinliğini değerlendirmiştir.

Demirkol.Ö, ve Demirkol,A (2003) çalışmalarında; Örnek Ağ Uygulamaları üzerine araştırmalar yapılmıştır. Bu araştırmalar kapsamında, en kısa yol hesaplamalarında Dijkstra Algoritması ile Bellman-Ford Algoritması

karşılaştırılmıştır. Elde edilen sonuçlara göre, Dijkstra Algoritması'nın kesinlikle daha iyi sonuçlar verdiği görülmüştür.

Analı.İ (1999) yüksek lisans tez çalışmasında; başlangıç çözümünün bulunması için V.A.M. (Vogel's Approximation Method) yöntemi kullanmış ve optimallik kontrolü yapılarak optimum ihracat modelini tespit etmiştir.

Literatürde başka Ağ optimizasyonu modelleri çalışmaları da bulunmaktadır.

2.MATERYAL VE METOT

2.1.Ağ Optimizasyonu Modelleri

Ağ optimizasyonu modelleri; en düşük maliyetli akış problemi, maksimum akış problemi, en kısa yol problemi, ulaştırma problemi, atama problemi, kapsayan ağaç problemi, CPM/PERT modelleri şeklinde sınıflandırılmaktadır. Bu bölümde Ağ Optimizasyonu Modelleri incelenip, bunların değişik alanlarda uygulanabilirliği hakkında bilgiler verilecektir.

2.1.1. Temel Kavramlar

Ağ optimizasyonu modelleri çalışmalarında ve başarılı çözüm değerine ulaşımında, algoritma yapısında kullanılan kavramların ve bu kavramların doğru değerlerinin iyi belirlenmesi büyük önem taşır. Bu kavramlar, modelin başarılı bir şekilde optimize edilmesi için gereklidir. Aşağıda bu kavramlar hakkında bilgi verilmektedir.

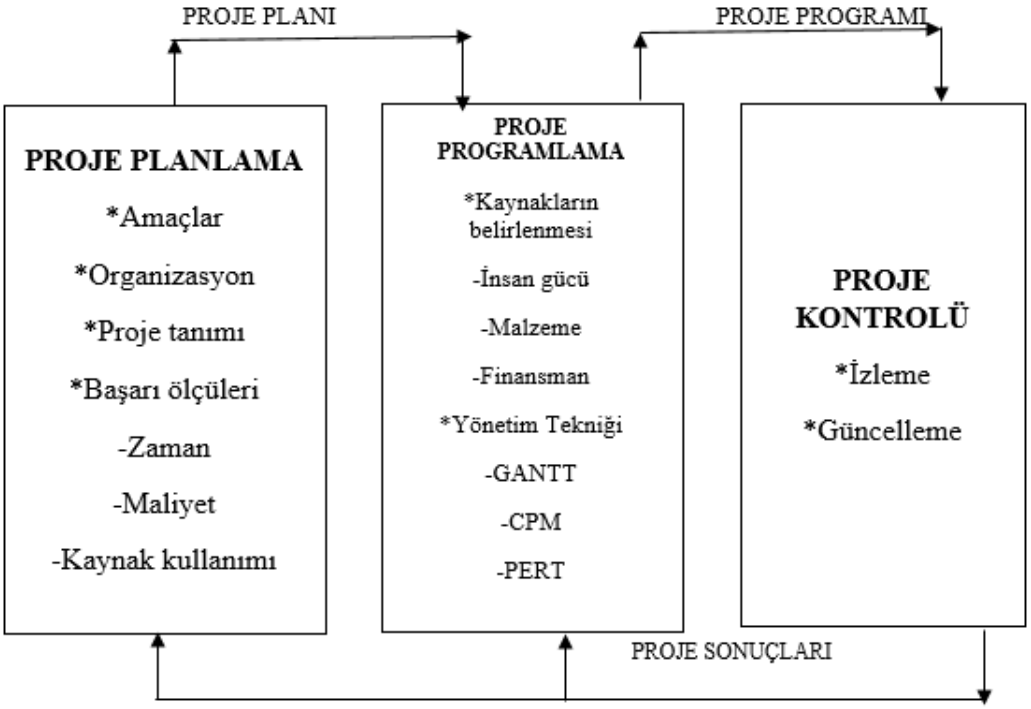
Karar Değişkeni: Ağ optimizasyonu probleminde, alınacak kararları temsil eden değişkenler karar değişkenleri olarak adlandırılır ve x, y, z gibi sembollerle gösterilir.

Kısıtlar: Karar değişkenleri veya parametreler ile karar değişkenleri arasındaki zorunlu ilişkilerin her birine kısıt denir. Amaca ulaşabilmek için uyulması gereken kuralları yada kısıtlayıcı durumları belirtir.

Amaç Fonksiyonu: Karar değişkenlerinin toplam getirisinin miktarını tanımlayan fonksiyondur. Maksimizasyon yada minimizasyon şeklinde olur. (Altunkaynak, M. 2003)

2.1.2.Proje Yönetimi Kavramları

Proje tanımı: Projede ne yapılacağını, neden yapılacağını ve nasıl yapılacağını açık bir şekilde ifade eder. Bu tanım, projenin planlanması, yürütülmesi ve kontrol edilmesi süreçlerinde yol göstericidir.



Şekil 2. 1 Proje Yönetimi Aşamaları (G.J.Monks, 1996)

Şekil 2.1'de gösterildiği gibi; Proje yönetimi aşamaları, proje planlama, proje programlama ve proje kontrolünü içermektedir. Bu aşamalar, bir projenin başarılı bir şekilde yönetilmesi ve hedeflerine ulaşması için izlenen adımları ifade eder.

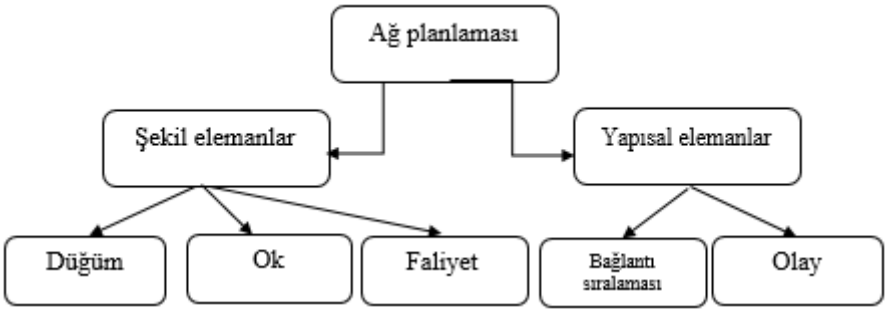
Proje planlama: Proje yönetimi sürecinin başlangıcında gerçekleştirilen aşamadır. Bu aşamada, projenin hedefleri, kapsamı, süresi, maliyetleri ve kaynakları detaylı bir şekilde belirlenir.

Proje programlama: Proje planının hayata geçirilmesi için kullanılan aşamadır. Bu aşamada, projede yer alan farklı faaliyetlerin başlangıç ve bitiş süreleri belirlenir, faaliyetler arasındaki bağımlılıklar ve ilişkiler tanımlanır. Proje programlama, proje takviminin oluşturulması ve kaynakların etkin bir şekilde yönetilmesi için kullanılan teknikler ve araçlar içerir.

Proje kontrolü: Projenin ilerlemesinin izlenmesi ve kontrol edilmesi aşamasıdır. Bu aşamada, gerçekleşen faaliyetlerin planlanan sürelerle karşılaştırılması, kaynakların kullanımının takibi, maliyetlerin kontrolü ve risklerin yönetimi gerçekleştirilir. Proje kontrolü, projenin hedeflerine uygun ilerlemesini sağlamak ve gerektiğinde düzeltici önlemler almak için kullanılır

Proje yönetimi aşamaları, projenin başarılı bir şekilde yönetilmesi ve tamamlanması için önemlidir. Bu aşamalar, projenin planlı ve disiplinli bir şekilde

ilerlemesini sağlar, kaynakların etkin kullanımını destekler, risklerin yönetilmesini sağlar ve projenin hedeflere uygun olarak tamamlanmasını sağlar.(Bakışkan.E, 2019)

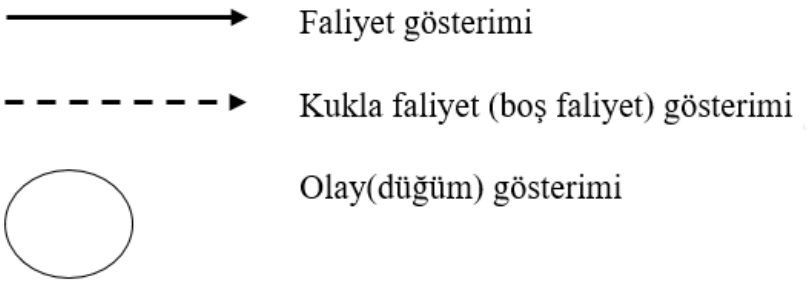


Şekil 2. 2 Ağ Planlaması Gösterimi

Olay (düğüm): Zaman diliminde meydana gelen, bir faaliyetin bitmiş ve diğer faaliyetlerin başlamaya hazır olduğunu ifade eder. Ağ modellerinde tüm olaylar veya düğümler "O" daire içinde gösterilir.

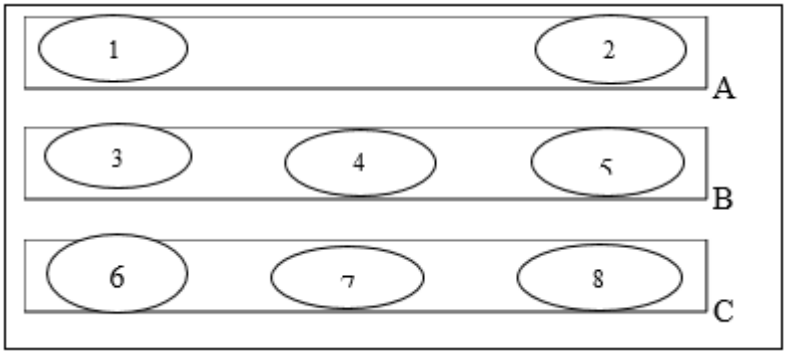
Oklar: Ağ modelinde her bir etkinlik yönlü bir okla gösterilmektedir. Oklar faaliyetler arasındaki mantıksal ilişkiyi göstermek için kullanılır.

Ağ planlanmasında kullanılan şekil elemanları aşağıdaki gibidir.



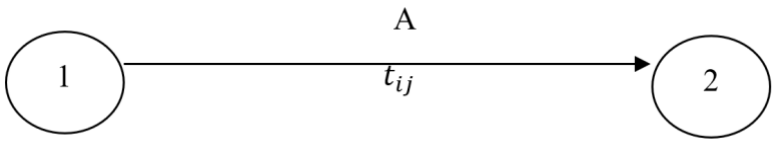
Şekil 2. 3 Ağ Planlanmasında Şekil Elemanları Gösterimi

Gantt şeması: Gantt şemaları, proje yönetiminde önemli bir araçtır. Bu şemalar sayesinde projedeki faaliyetlerin sıralaması, süreleri ve bağımlılıkları daha iyi anlaşılır ve projenin zaman çizelgesi görsel olarak temsil edilir. Bu şekilde proje yöneticileri ve ekip üyeleri, projenin ilerlemesini takip edebilir, faaliyetlerin zamanlamasını ayarlayabilir ve proje hedeflerini daha iyi yönetebilir. Ancak, Gantt şemaları, daha karmaşık ilişkileri, paralel faaliyetleri veya döngüsel bağımlılıkları ifade etmek için yetersiz kalabilir (Sarıca.İ, 2006).



Şekil 2. 4 Gantt Cetveli (Mazlum. M, 2014)

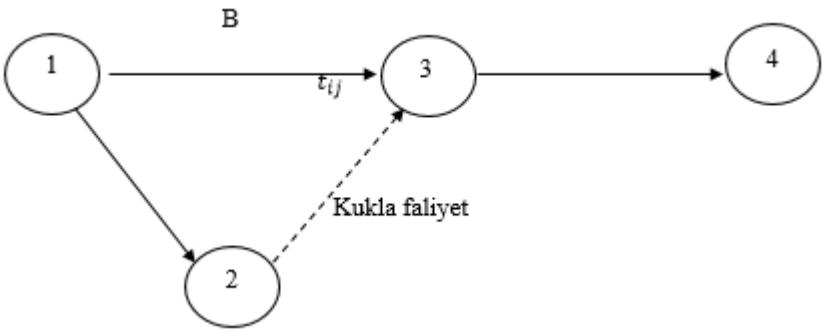
Faliyet: bir projenin veya işin tamamlanması için gerçekleştirilen belirli bir görev veya eylemi ifade eder. Her faaliyet, başlangıç ve bitiş noktaları ile belirli bir süre içinde tamamlanması gereken görevleri içerir. Faaliyetler genellikle, birbirleriyle mantıksal bir sıra içinde bağlantılıdır ve belirli bir bağımlılık ilişkisi vardır. Bir faaliyetin başlaması genellikle önceki faaliyetlerin tamamlanmasına bağlıdır.



Şekil 2. 5 Faaliyetlerin Grafik Gösterimi

Şekil 2.5'de faaliyetlerin grafik gösterimi verilmiştir. Ağ diyagramlarında faaliyetlerin adları okların üstüne, süreleri ise okların altına yazılır. Oklar genellikle (i,j) gibi iki düğüm noktası arasında yer alır ve faaliyet akışını gösterir. Okların yönü, bir faaliyetin diğer bir faaliyeti takip ettiğini ve hangi yönde ilerlendiğini belirtir.

Kukla faaliyet: Gerçekleşmesi için zaman ve kaynak kullanımı gerektirmeyen ve sadece bir değişkenin başka bir değişkene bağımlılığını gösteren faaliyetlerdir. Bundan dolayı "bağımlılık oku" olarak adlandırılırlar.



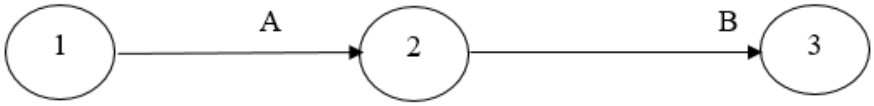
Şekil 2. 6 Kukla Faaliyetlerin Grafik Gösterimi

Şekil 2.6'daki 2 ile 3 arasındaki faaliyet kukla faaliyetidir. Görüldüğü gibi kukla faaliyetinin süresi sıfırdır.

2.1.4. Faaliyetler Arasındaki Bağıntılar

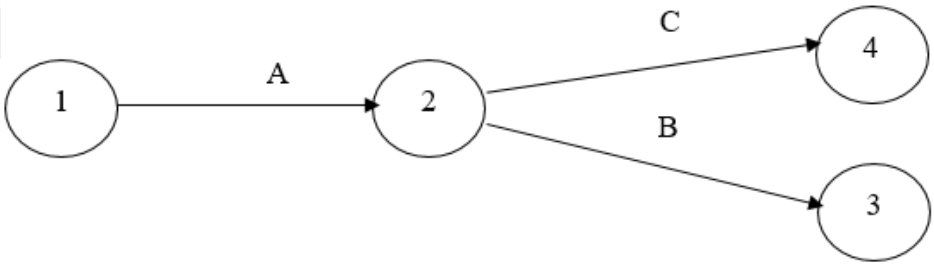
"Faaliyetler arası bağlantı" genellikle "faaliyetler arası iletişim" veya "faaliyetler arası koordinasyon" olarak da adlandırılır ve bir organizasyon içindeki farklı faaliyetler arasındaki etkileşimi ifade eder. Birbiriyle mantıksal bağlantıları olan çok sayıda faaliyetten proje oluşturulmaktadır. Bu bağlantıların ağ şebekesinin hatasız oluşturulması gerekmektedir. Bu bağlantılara ait genel örnekler aşağıdaki gibidir.

1) A faaliyetinden sonra B faaliyeti gelir.



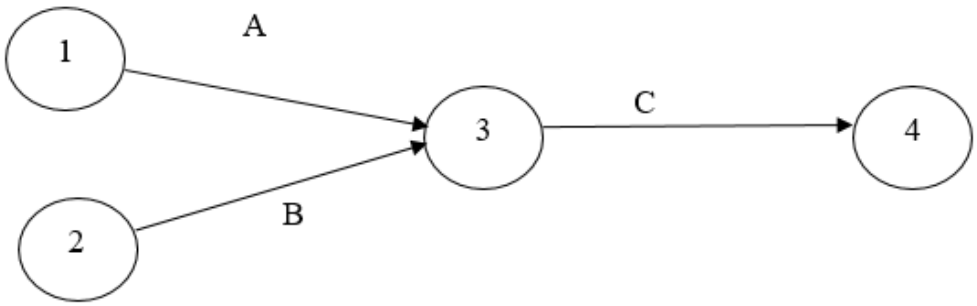
Şekil 2. 7 İki Faaliyet Arasındaki Bağlantı Gösterimi

2) B ve C faaliyeti A faaliyetinden sonra başlar.



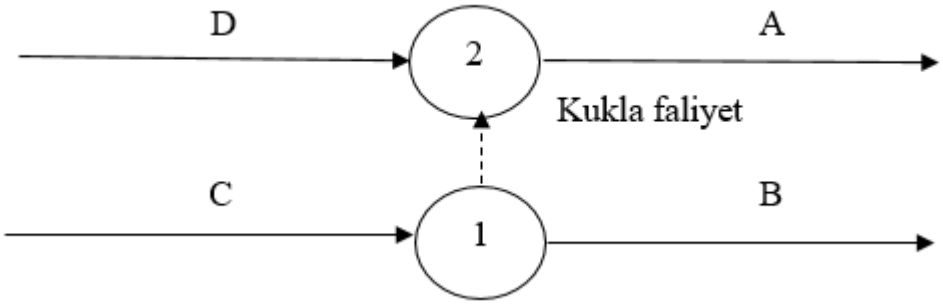
Şekil 2. 8 Üç Farklı Faaliyetin Arasındaki Bağlantı Gösterimi

3) C faaliyeti A ve B faaliyetinden sonra başlar.



Şekil 2. 9 Üç Farklı Faaliyetin Arasındaki Bağlantı Gösterimi

4) A faaliyeti C faaliyeti ve D faaliyetinin bitimine bağlı, fakat B faaliyeti yalnızca C faaliyetinin bitimi ile başlar.



Şekil 2. 10 Kukla Faaliyetinin Farklı Gösterimi

Projeyi oluşturan faaliyetler arasındaki mantıksal ilişkiler, şebeke diyagramı modellerinin temelini oluşturur. (Sarıca.İ, 2006)

2.2.En Düşük Maliyetli Akış Problemi

En düşük maliyetli akış problemi, bir ağda bir kaynaktan bir hedefe en düşük maliyetle ne kadar malzeme veya birim akış gönderilebileceğini belirlemeyi amaçlayan bir matematiksel problemdir. En düşük maliyetli akış problemi; ulaştırma, atama, en kısa yol, maksimum akış algoritmalarının özel durumlarıdır. Bu problemler doğrusal programlama problemi olarak formüle edilebildiğinden etkince çözülebilmektedir.

En düşük maliyetli akış problem için matematiksel model denklemi :

Amaç fonksiyonu: $\min Z = \sum_{i=1}^{n} \sum_{j:1}^{n} c_{ij} x_{ij}$ **(2.1)**

x_{ij} i. tedarikçiden j. müşteriye gönderilen ürün miktarını belirtmektedir.

c_{ij} Bir birim ürünün gönderilme maliyetidir.

Kısıtlar:

$$\sum_i x_{ij} - \sum_i x_{ij} = b_i \quad \textbf{(2.2)}$$

$$\text{xij} \geq 0 \text{ ve tamsayı} \quad \textbf{(2.3)}$$

$$\sum_i b_i = 0 \quad \textbf{(2.4)}$$

Burada:

(2.1) numaralı denklem amaç fonksiyonunu temsil etmektedir. Amaç fonksiyonu, düğümler arası toplam hareket maliyetini en aza indirmeyi hedefler. Amaç fonksiyonun minimum olabilmesi için 2 adet kısıtlayıcı denklemlerin sağlanması gerekmektedir.

(2.2) numaralı denklemde denklemdeki ilk toplam i düğümden gönderilen toplam akışı, ikinci toplam düğüme gelen toplam akışı ifade etmek ile birlikte, düğümün net akış farkını göstermektedir.
(2.3) numaralı işaret gösteren kısıtlayıcı denklemde ise karar değişkenlerin her zaman pozitif ve tamsayı olması gerektiğini göstermektedir.
(2.4) numaralı denklemde problemin uygun çözüm olabilmesi için eşitliğin sağlanması gerekmektedir. (Özkan. Ş, 2012)

2.3.En Kısa Yol Problemi

En kısa yol problemi (EKY), bir başlangıç noktasından hedeflenen bir noktaya gitmek için en kısa mesafeyi bulmayı amaçlayan bir matematiksel problemidir. Bu problem, özellikle yolculuk planlaması, taşımacılık, mühendislik ve bilgisayar grafikleri gibi alanlarda önemlidir. En kısa yol problemi, 1959 yılında Hollandalı bilgisayar bilimcisi Edsger Wybe Dijkstra tarafından Dijkstra algoritması için geliştirilmiştir. EKY problemleri, alternatiflere göre en düşük maliyetli yolları bulmayı hedefler. Bu amaç doğrultusunda kullanılan algoritmalar arasında A*arama algoritması, Dijkstra algoritması, Bellman-Ford algoritması, Floyd-Warshall algoritması ve Johnson's algoritması yer almaktadır.

En kısa yol problemi, maliyetleri (cij) dikkate alarak, m düğümden oluşan bir ağ yapısı üzerinde iki düğüm arasında, xi,j adında ikili değişken kullanılarak, en düşük maliyetli yolun bulunmasını amaçlar.
En kısa yol problemi için matematiksel model denklemi:

Amaç fonksiyonu: $$\min Z = \sum_{i=1}^{m} \sum_{j:1}^{m} c_{ij} x_{ij} \quad \textbf{(2.5)}$$

Kısıtlar:

$$\sum_{j=1}^{m} x_{ij} - \sum_{k=1}^{m} x_{ki} = \begin{cases} 1 & eğer\ i = 1 \\ 0 & eğer\ i \neq 1\ ya\ da\ i \neq m \\ -1 & eğer\ i = m \end{cases} \quad \textbf{(2.6)}$$

$$xij \in \{0,1\}\ \forall i,j \quad \textbf{(2.7)}$$

Biçimindedir (ULU. F, 2023)

En kısa yol problemi, problemin durumuna göre çeşitli alt varyasyonlara ayrılmaktadır. İşte en yaygın olarak karşılaşılan bazı en kısa yol problemi varyasyonları:

1. Tek-eşli EKY problemi
2. Tek-kaynaklı EKY problemi
3. Tek-hedefli EKY problemi

4. Tüm eşler arası EKY problemi

2.3.1. A* arama algoritması

A*arama algoritması Tek-eşli EKY probleminde etkili algoritmadır. A*algoritması, sezgisel bir fonksiyonu kullanarak düğümleri öncelikli sıraya göre keşfeder ve en düşük maliyetli yolu bulur. A* arama algoritması aşağıdaki formül ile hesaplanır.

$$f_{(n)} = g_{(n)} + h_{(n)} \quad \textbf{(2.8)}$$

Burada:

$f_{(n)}$: mesafenin hesaplandığı sezgisel fonksiyonu

$g_{(n)}$: başlangıç düğümünden mevcut düğüme kadar gelmenin maliyetini

$h_{(n)}$: mevcut düğümden hedef düğüme ulaşmak için tahmini mesafe değerini

İfade etmektedir. $f_{(n)}$ Fonksiyonunun sezgisel olmasının nedeni, içindeki $h_{(n)}$ fonksiyonunun tahmine dayalı sezgisel bir fonksiyon olmasıdır.

2.3.2 Dijkstra Algoritması

Tek kaynak noktasından tüm diğer noktalara olan en kısa yolları hesaplamak için kullanılan bir algoritma Dijkstra algoritmasıdır. Bu algoritma, graf teorisi ve matematiksel programlama alanlarında kullanılır ve ilk olarak 1956 yılında Edsger Dijkstra tarafından geliştirilmiştir. Dijkstra algoritması, bir graf üzerinde çalışır ve grafın her bir düğümüne en kısa yol mesafesini hesaplar. Başlangıç düğümünden itibaren her bir düğüm için, o düğüme olan en kısa yolu hesaplamak için çevreleyen tüm düğümler arasından en düşük maliyetli olanı seçer. Bu işlem, başlangıç düğümden başlayarak diğer düğümlere olan en kısa yolları hesaplamak için tekrarlanır.

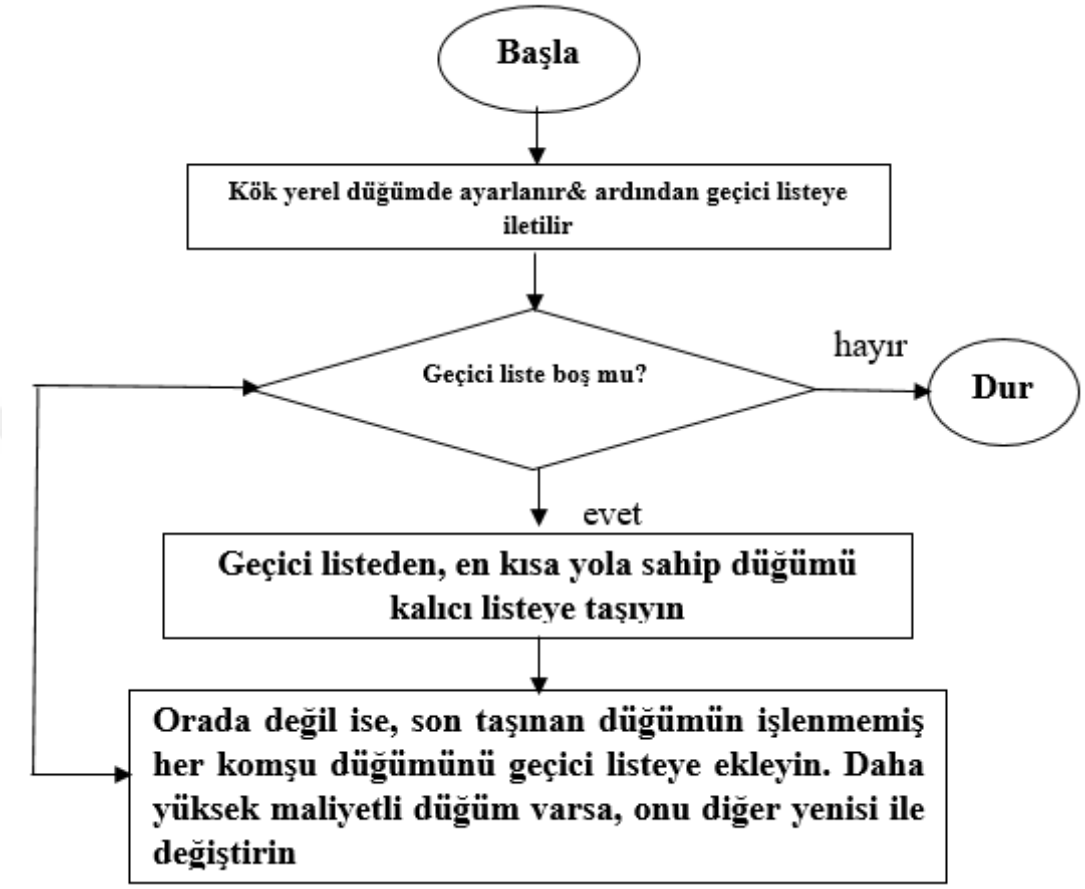
Dijkstra algoritması, öncelik kuyruğu yapısını kullanarak çalışır. Öncelik kuyruğu, düğümler arasındaki maliyetleri tutar ve maliyeti en düşük olan düğümü öncelikli olarak işler. Bu sayede algoritma, daha az sayıda düğüme erişmek için daha düşük maliyetli yolları öncelikli olarak hesaplar. Dijkstra algoritmasının karmaşıklığı, düğüm sayısı olan n için O(n^2) şeklindedir.

Dijkstra algoritmasını kullanmak için:

1.Ağırlıklı ve yönlü grafikler için geliştirilmiştir.

2.Kenar değerleri 0'dan küçükse, kenarların ağırlık değeri sıfır veya pozitif olmalıdır. Eğer kenar değerleri sıfır ise, Bellman-Ford algoritması kullanılabilir.

3.Negatif ağırlıklar için çalışmaz. Bu algoritma, negatif döngülerin olmadığı grafiklerde doğru sonuçlar üretebilir. (Dhulkefl. E. J., Durdu.A., &Terzioğlu.H, 2020)

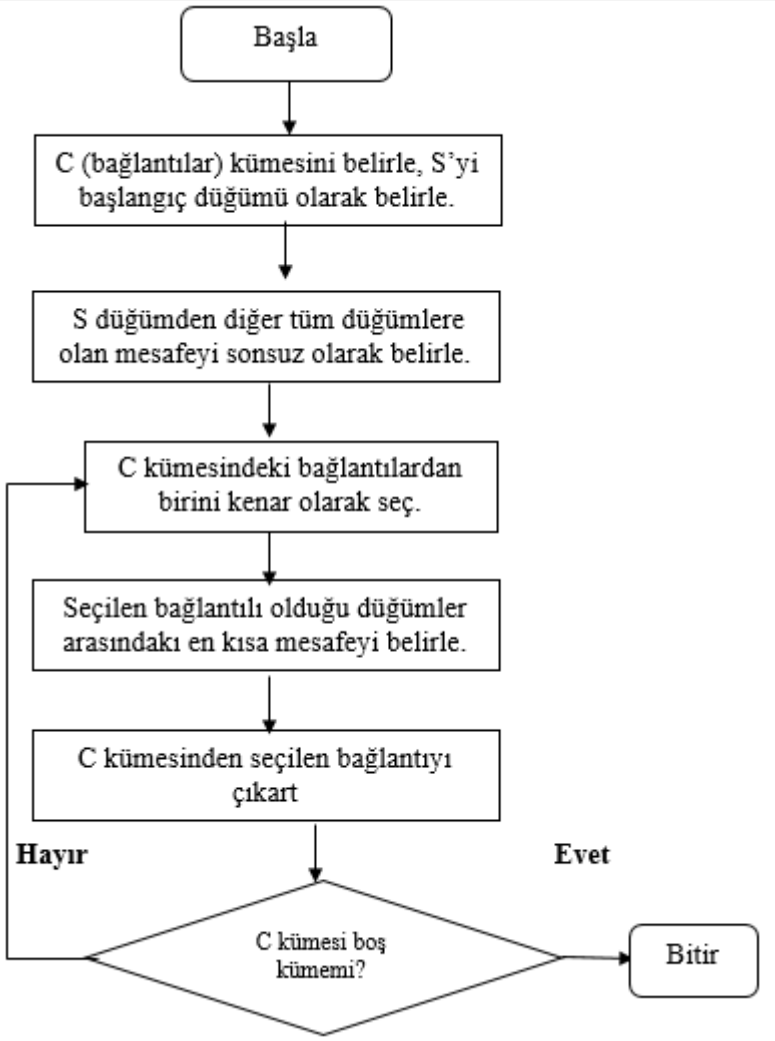


Şekil 2. 11 Dijkstra Algoritmasının Akış Diyagramı

2.3.3. Bellman-Ford Algoritması

Belman-Ford algoritmasının Dijkstra algoritmasından farkı, düğümler arası bağlantıların (kenarların) negatif olabilen ağ yapılarında da çalışabilmesidir. Dijkstra algoritması, sadece pozitif kenar ağırlıkları olan ağlarda doğru sonuçlar verirken, Belman-Ford algoritması negatif ağırlıklara da uyum sağlayabilir.

Dijkstra algoritması her adımda en kısa mesafeye sahip düğümü seçerek ilerlerken, Belman-Ford algoritması tüm düğümleri dikkate alır ve daha genel bir yaklaşım kullanır. Belman-Ford algoritmasının karmaşıklığı, düğüm sayısı olan n için O(n^2) şeklindedir.



Şekil 2. 12 Bellman-Ford Algoritmasının Akış Diyagramı

2.3.4. Floyd-Warshall Algoritması

Floyd-Warshall algoritması, kenar ağırlıklı yönlendirilmiş grafikte tüm köşe çiftleri arasındaki en kısa yolları hesaplamak için basit ve yaygın olarak kullanılan bir algoritmadır. Floyd-Warshall algoritması, tek kaynak noktasından tüm diğer noktalara olan en kısa yolları hesaplamak için kullanılan bir algoritmadır Negatif döngülerin varlığını tespit etmek için de kullanılabilir. Bu görev için, Floyd-Warshall algoritmasının birçok mevcut uygulamasının başarısız olacağını göstereceğiz, çünkü yürütme sırasında üstel olarak büyük sayılar görünebilir. Bu algoritmanın zaman karmaşıklığı 〖O(n〗^3) şeklindedir. Burada n, düğüm sayısını temsil eder. Sonuç olarak, bu algoritma, tüm çiftler arasındaki en kısa mesafeleri hesaplamakta kullanılır ve mesafeler matrisini döndürür. Bu algoritma negatif ağırlıklı ağlarda çalışabilir ve negatif döngülerin varlığını tespit edebilir. (Stefan. H 2010)

2.3.5. JOHNSON'S Algoritması

Johnson's algoritması, 1977 yılında Donald B. Johnson tarafından geliştirilmiş bir algoritmadır. Bu algoritma, tüm eşler arası en kısa yol problemini çözmek için kullanılır. Grafikler Floyd-Warshall algoritmasına benzer; ama Johnson'ın algoritması daha hızlı ve diğerinden daha seyrek grafikler için çalışıyor. Bunun için Dijkstra ve Bellman-Ford algoritmaları kullanılmaktadır. Algoritma.

e kenar sayısı ve n düğüm sayısı ise, çalışma süresi algoritma karmaşıklığı $O(en + n^2 logn)$ şeklindedir.

Yukarıdaki bilgilere göre, dört algoritmaların özet bir karşılaştırma tablosu:

Tablo 2. 1 En Kısa Yol Problemı Algoritmalarının Karşılaştırma Tablosu

Parametreler	Dijkstra Algoritması	Belman-Ford Algoritması	Floyd-Warshall Algoritması	Jhonson's Algoritması
Veri yapısı	Grafik	Grafik	Grafik	Grafik
Karmaşıklık	$O(n^2)$	$O(en)$	$O(n^3)$	$O(en + n^2 logn)$
Negatif ağırlıklı ağlarda çalışabilir	HAYIR	EVET	EVET	EVET
En kısa yol varyasyonu	Tek-kaynaklı	Tek-kaynaklı	Tüm eşler arası	Tüm eşler arası

*n düğüm sayısı ve *e kenar sayısı

2.4 Maksimum akış problemi

Maksimum akış problemi, bir ağda bir kaynak noktasından bir hedef noktasına bağlantı kapasitesi aşılmayacak şekilde en fazla akışı sağlamak amacı ile nasıl bir yol izleneceğini belirlemek için kullanılan bir optimizasyon problemidir. Bu problemde iki düğüm arasındaki gerçekleşecek olan akış kapasitesini aşmaması ve akışların korunumu ilkesi göz önüne alınarak model elde edilir.

Maksimum akış problemi için matematiksel model denklemi:

Amaç fonksiyonu: $max = x_0$ **(2.9)**

x_0: i. tedarikçiden j. müşteriye gönderilen toplam ürünün miktarını

d_{ij}: i.den j.ye kadar akış kapasitesini

x_{ij}: i.den j.ye kadar akış miktarını göstermektedir.

Kısıtlar:

$\sum_k x_{1k} - \sum_k x_{k1} = x_0$, *Başlangıç düğüm* **(2.10)**

$$\sum_k x_{jk} - \sum_k x_{kj} = 0 \qquad \text{,Ara düğümler} \tag{2.11}$$

$$\sum_k x_{nk} - \sum_k x_{kn} = -x_0 \qquad \text{,Bitiş düğüm} \tag{2.12}$$

$$0 \leq x_{ij} \leq d_{ij} \tag{2.13}$$

gibi elde edilir. (Kara.İ, 2000)

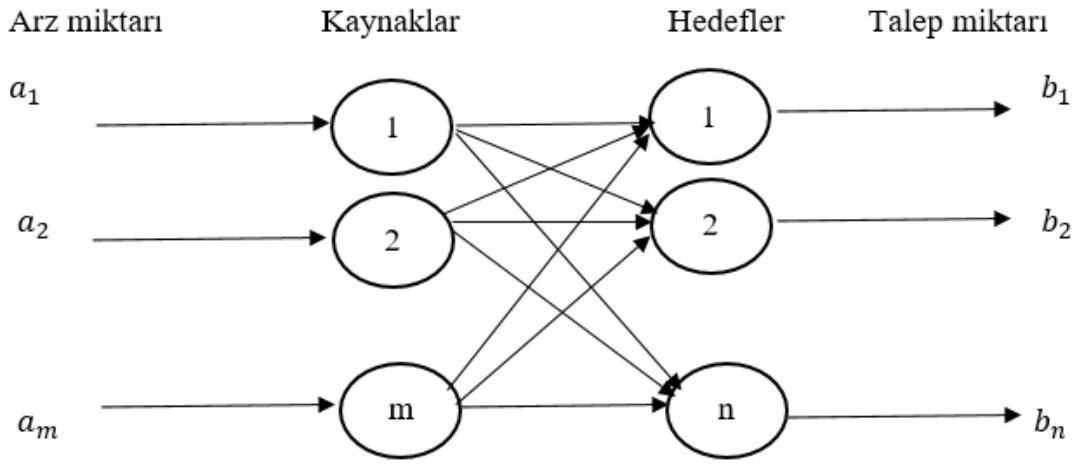
2.4.1 Ford-Fulkerson Algoritması

Ford-Fulkerson yöntemi veya Ford-Fulkerson algoritması, bir akış ağındaki maksimum akışı hesaplayan bir açgözlü algoritmadır. Algoritma, 1956 yılında LR Ford Jr. ve DR Fulkerson tarafından yayınlanmıştır. Ford-Fulkerson yöntemi, maksimum akış problemlerinin çözümünde etkili bir algoritmadır ve birçok uygulama alanında kullanılır. Örneğin, ağ tasarımı, trafik yönlendirme, veri iletimi ve enerji akışı problemleri gibi alanlarda kullanılabilir.

2.5 Ulaştırma problemi

Ulaştırma problemi taşıma sektöründe taşıma kapasitesi, zamanlama, rota planlaması, depolama ve diğer faktörlerin optimize edilmesi amacıyla kullanılan matematiksel bir yaklaşımdır. Ulaştırma problemi kavramı, 1930'larda George Dantzig tarafından geliştirilmiştir.

Günümüzde ulaştırma problemi taşıma sektöründe birçok farklı uygulama alanına sahiptir. Örneğin, nakliye şirketleri, lojistik şirketleri ve denizcilik şirketleri ulaştırma maliyetlerini azaltmak, teslimat sürelerini kısaltmak ve müşteri memnuniyetini arttırmak için ulaştırma problemi kullanılmaktadır. Ulaştırma probleminin genel hali aşağıda verilmiştir.



Şekil 2. 13 Ulaştırma Probleminin Genel Gösterimi

Ulaştırma probleminin genel gösteriminden yola çıkarak matematiksel modeli aşağıdaki gibi elde edebiliriz.

Amaç fonksiyonu : $Min\ Z = \sum_{i=1}^{m} \sum_{J=1}^{n} c_{ij} x_{ij}$ (**2.14**)

x_{ij} i. Tedarikçiden j. Müşteriye gönderilen ürün miktarını belirtmektedir.

c_{ij} Bir birim ürünün gönderilme maliyetidir.

Kısıtlar:

$$\sum_i x_{ij} = b_j \in \text{N} \quad \textbf{(2.15)}$$

$$\sum \text{j}\ x_{ij} = a_i \in \text{M} \quad \textbf{(2.16)}$$

$$\text{xij} \geq 0 \text{ ve tamsayı} \quad \textbf{(2.17)}$$

$$\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j \quad \textbf{(2.18)}$$

Burada

(2.14) numaralı denklem amaç fonksiyonu temsil etmekte olup toplam ulaştırma maliyetini minimum yapmaktır. Amaç fonksiyonun minimum olabilmesi için 3 adet kısıtlayıcı denklemlerin sağlanması gerekmektedir.

(2.15) numaralı kısıtlayıcı denklemde tedarikçi kapasitesinin tedarikçilerden yapılacak olan toplam taşımaya eşit olmasını göstermektedir.

(2.16) numaralı kısıtlayıcı denklemde müşteriye yapılması gereken toplam taşıma miktarının müşterinin talep ettiği miktarına eşit olmasını göstermektedir.

(2.17) numaralı işaret gösteren kısıtlayıcı denklemde ise karar değişkenlerin her zaman pozitif ve tamsayı olması gerektiğini göstermektedir.

(2.18) numaralı kısıtlayıcı denklem ise dengelenmiş ulaştırma probleminde kullanılan kısıtlayıcı denklemdir.

Ulaştırma yöntemi, arz ve talep faktörlerinden etkilenir. Arz, bir ürün veya hizmetin piyasada sunulabilecek miktarını ifade ederken, talep ise o ürün veya hizmete olan talebi ifade etmektedir. Ulaştırma probleminin dengede olabilmesi için, toplam arzın toplam talebe eşit olması gerekmektedir. Eğer bu eşitlik sağlanmazsa, ulaştırma problemi çözüme ulaşmadan önce denge durumuna getirilmelidir. Denge durumunu sağlamak için, yoğun talep gören bir ulaştırma yöntemi, arzın bu talebi karşılayacak şekilde artırılması gerektiği anlamına gelir. Bu şekilde, toplam arzın toplam talebe eşit olduğu bir denge noktası elde edilir ve ulaştırma probleminin çözümüne geçilebilir. Aynı şekilde, talebin azaldığı bir durumda arzın da buna göre düşürülmesi gerekebilir. Bu nedenle, ulaştırma yöntemleri için arz ve talep faktörleri,

ulaştırma şirketlerinin planlanması ve karar vermesi gereken önemli faktörlerdir. (Taha,2000)

2.5.1. Kuzey-Batı Köşe Algoritması

Bu yöntemde başlangıç tablosunun maliyet dikkate alınmaksızın kuzey-batı köşesinden başladığı için bu şekilde adlandırılmıştır.

Tablo 2. 2 Ulaşım Tablosu

	Tüketim Merkezleri				Arz
	x_{11}	x_{12}	...	x_{1n}	a_1
	x_{21}	x_{22}	...	x_{2n}	a_2
	...	...	...	...	...
Üretim Merkezi	...	...	...	...	...
	x_{m1}	x_{m2}	...	x_{mn}	a_m
Talep	b_1	b_2		b_n	

Bu tabloda, her bir hücre Xij ile temsil edilen i. üretim merkezinden j. tüketim merkezine taşınacak mal miktarını gösterir. Tablo, üretim merkezlerinin satırlarına ve tüketim merkezlerinin sütunlarına sahiptir. Üretim merkezlerini i=1, 2, 3, ..., m ve tüketim merkezlerini j=1, 2, 3, ..., n şeklinde temsil edebilirsiniz. Ulaştırma tablosu düzenlendikten sonra, kuzey-batı köşesinden bir hücre seçilir ve aşağıdaki algoritmalar izlenir:

Adım 1: Seçilen hücredeki miktar, talep ve arz miktarlarına göre karşılaştırılır.

Adım 2: Burada üç durum söz konusudur:

Eğer Talep miktarı> arz miktarından büyükse, hücredeki arz miktarı kadar ürün taşınır ve talep miktarı azaltılır.

Eğer arz miktarı> talep miktarından büyükse, hücredeki talep miktarı kadar ürün taşınır ve arz miktarı azaltılır.

Eğer Arz miktarı = talep miktarı eşitse, hücredeki mal taşınır ve hem arz hem de talep miktarları sıfıra indirilir.

Adım 3: Taşınan ürün miktarı kaydedilir ve ilgili hücre işaretlenir veya çizilir. İşlem, tüm hücrelerin kontrol edilene kadar tekrarlanır.

Bu şekilde, ilerleyen adımlarda sadece kullanılabilir talep ve kaynak miktarlarıyla çalışılır. Yöntem, en büyük atamaları yaparak başlaması ve sadece mevcut kaynakları kullanması nedeniyle basit ve hızlı bir yaklaşımdır. Kuzey batı-köşe yöntemi, basit ve anlaşılır bir yaklaşım olduğu için tercih edilir. Daha sonra, bu

başlangıç planını iyileştirmek için daha farklı yöntemler kullanılabilir. (Al-Momayez.D, 2023)

2.5.2. En Düşük Maliyetler Algoritması

En düşük maliyetler yönteminde en uygun çözüm hedeflendiğinden dolayı daha iyi başlangıç çözüm bulunmaktadır. Bundan dolayı en düşük maliyetler problemi diye adlandırılmaktadır. Bu yöntemde başlangıç tablodan yararlanarak, en düşük maliyetli kutuya mümkün olduğunca fazla atama yapılması ile ilk çözüme ulaşılır. Eğer maliyetlerin eşit olması durumu söz konusu ise rastgele atama yapılır. Bundan dolayı en düşük maliyetler problemi diye adlandırılmıştır.

Tablo 2.2'de her bir hücre Xij ile temsil edilen i. üretim merkezinden j. tüketim merkezine taşınacak mal miktarını gösterir. Tablo, üretim merkezlerinin satırlarına ve tüketim merkezlerinin sütunlarına sahiptir. Üretim merkezlerini i=1, 2, 3, ..., m ve tüketim merkezlerini j=1, 2, 3, ..., n şeklinde temsil edebilirsiniz. Ulaştırma tablosu düzenlendikten sonra, en düşük birim maliyetli kutudan bir hücre seçilir ve aşağıdaki algoritmalar izlenir:

Adım 1: Seçilen kutunun arz ve talep miktarlarına göre taşınacak ürün miktarı belirlenir. Arz ve talep miktarlarına bağlı olarak, seçilen kutudaki arz ve talep miktarlarına veya tablodaki diğer sınırlamalara uygun şekilde atama yapılır.

Adım 2: Atamanın tekrarlamasını önlemek amacı ile sıfır olan arz veya talep satırdan veya sütundan iptal edilir. Eğer satır ve sütun birlikte sıfır olursa, biri seçilir. İptal edilmeyen satır veya sütundaki sıfır arz (talep) miktarları dikkate alınmaz.

Adım 3: Sadece bir iptal edilmeyen satır veya sütun kaldığında durulur. Aksi takdirde, bir önceki adımda sütun iptal edilmişse sağdaki hücreye geçilir, satır iptal edilmişse bir alt hücreye geçilir ve Adım 1'e geri dönülür. (Taha.H, 2000)

2.5.3. Vogel Yaklaşım Algoritması

Ulaştırma probleminde daha iyi bir temel uygun çözüm bulmak için geliştirilen algoritmalardan biri Vogel's Approximation Method kelimesinin ilk harflerinden oluşan VAM olup, Vogel'in yaklaşım yöntemi olarak isimlendirilmektedir. Vogel yaklaşımı, diğer ulaştırma problemleri için geliştirilen yöntemlerden biridir ve genellikle ilk adımda oluşturulan uygunluk maliyet matrisinin doğru şekilde hazırlanmasını gerektirir. Bu yöntem, optimum bir sevkiyat planı sağlamaya çalışırken, bazen diğer yaklaşımlarla karşılaştırıldığında daha iyi sonuçlar verebilir.

Ancak, büyük ölçekli ulaştırma problemleri için hesaplama karmaşıklığı yüksek olabilir. .(Şule.Ö, 2012)

Adım 1: Ulaştırma tablosundaki en yüksek ceza maliyete sahip satır ve sütün belirlenir.

Adım 2: Belirlenen hücrelere, talep edilen miktar ve sağlanan miktar dikkate alınarak ceza maliyeti yüklenir. Gereksimini karşılanan satır ve sütün tablodan çıkartılır.

Adım 3: Geriye kalan hücrelerdeki satır ve sütünlerin ceza maliyetleri tekrar hesaplanır ve adım2 deki gibi hesaplamalar yapılır.

Adım 4: İşlem sadece 1 satır veya 1 sütun kalana kadar tekrarlanır.

Tüm dağıtımlar bu adımlar izlenerek yapılır ve başlangıç optimal çözüm elde edilir. (Taha.H, 2000)

2.6 Atama problemi

Atama problemi, bir ağ üzerindeki kaynakların en etkili şekilde kullanımını sağlamak için kaynakların hangi görevlere atanacağını belirleyen bir optimizasyon problemidir. Atama problemi temel bir kombinatoryal optimizasyon problemidir. Ağ optimizasyonu atama problemi, kaynakların zaman, maliyet ve kapasite kısıtlamaları altında atanması gereken bir dizi görevin bulunduğu durumlarda kullanılır. Bu problemi çözmek, işletmelerin verimliliğini artırmalarına ve kaynaklarını daha iyi yönetmelerine yardımcı olmaktadır. Atama problemleri matematiksel olarak modellenir ve optimizasyon algoritmaları kullanılarak çözülür. Bu problemi çözmek için kullanılan yaygın teknikler arasında lineer programlama, tam sayılı programlama, genetik algoritmalardır.

Atama probleminin matematiksel modeli aşağıdaki gibidir.

Amaç fonksiyonu $min\ Z$: $\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij}$ **(2.19)**

Kısıtlar:

$\sum_{i:1}^{m} x_{ij} = 1$ *(i=1,,,,,.n)* **(2.20)**

$\sum_{j:1}^{n} x_{ij} = 1$ *(j=1,,,,,.m)* **(2.21)**

$x_{ij} = 0$ veya 1 **(2.22)**

Burada

(2.19) numaralı denklem amaç fonksiyonu temsil etmekte olup, toplam atama maliyetini minimum yapmaktır. Amaç fonksiyonun minimum olabilmesi için 3 adet kısıtlayıcı denklemlerin sağlanması gerekmektedir.

(2.20) ve (2.21) numaralı kısıtlayıcı denklemler arz miktarı ve talep miktarı daima bire eşit olması gerektiğini göstermektedir.

(2.22) numaralı işaret gösteren kısıtlayıcı denklemde ise karar değişkenlerin her zaman pozitif ve tamsayı olması gerektiğini göstermektedir.

Atama problemini doğrusal amaç fonksiyona hem de tam sayılı pozitif kısıtlara sahip olduğundan dolayı simpleks yöntemi ile çözmek mümkündür. Atama problemlerinde yaygın kullanılan bir diğer yöntem ise Macar Algoritması yöntemidir. (Altunkaynak.B & Bakır.M.A 2003)

2.6.1 Macar Algoritması

Macar Algoritması, kural temelli bir algoritmadır ve en uygun eşleşme probleminin çözümü için kullanılır. Genellikle, atama problemleri maliyet matrisi kullanılarak çözülür. Örneğin, işçi atamaları, makine atamaları veya diğer kaynak atamaları gibi durumlarda kullanılabilir. Atama probleminde Macar Algoritması en yaygın kullanılan yöntemdir. Bu algoritmada, kaynakların sayısının hedef sayısına eşit olması gerekir. Yani problem dengeli olmalıdır.

Macar algoritmasının adımları aşağıdaki gibidir:

Adım 1: boyutlu orijinal maliyet matrisindeki her satırın minimumunu bul ve bu değeri, kendi satırındaki değerlerden çıkarak yeni matris oluştur. Yeni matristeki her sütunun minimumunu bul ve bu değeri, kendi sütunundaki değerlerden çıkarak yeni indirgenmiş maliyet matrisini oluştur.

Adım 2: İndirgenmiş maliyet matrisindeki tüm sıfır değerli hücrelerden geçecek şekilde en az sayıda yatay ve dikey çizgiler oluştur. Eğer tüm sıfır değerlerini örtecek şekilde sayıda çizgi çizilmişse, örtülmüş sıfırlar arasında optimum çözüm mevcuttur. Aksi durumda Adım 3'e git.

Adım 3: Sıfır olmayan en küçük elemanı bul. Bu değeri, indirgenmiş maliyet matrisindeki örtülmemiş elemanlardan çıkar ve iki çizgi ile örtülmüş elemanlara ekle. Daha sonra Adım 2'ye git.

Macar Algoritması, atama problemlerini hızlı ve etkili bir şekilde çözmek için kullanılan bir algoritmadır. Yüksek verimliliği ve doğruluğu nedeniyle tercih edilen bir yöntemdir. (Altunkaynak.B & Bakır.M.A 2003)

2.7 Kapsayan ağaç problemi

Minimum yayılan ağaç (MYO), grafik teorisinde, özellikle ağırlıklı grafikler için kullanılan bir terimdir. Kenarların toplam ağırlığını veya maliyetini en aza indirirken, orijinal grafiğin tüm köşelerini içeren bir alt grafiktir.

Kapsayan ağaç probleminin matematiksel modeli aşağıdaki gibidir.

Amaç fonksiyonu : $Min\ Z$: $\sum_{i=1}^{m}\sum_{j=1}^{n} d_{ij}x_{ij}$ **(2.23)**

d_{ij}: i ile j arasındaki uzaklıgı belirtmektedir.

x_{ij}: i.düğümden j. düğüm arasındaki bağlantıyı belirtmektedir.

Kısıtlar:

$$\sum_{i:1}^{m} x_{ij} \geq 1 \qquad (i=1,,,,,,m) \qquad \textbf{(2.24)}$$

$$\sum_{j:1}^{n} x_{ij} \geq 1 \qquad (j=1,,,,,,n) \qquad \textbf{(2.25)}$$

$$x_{ij} = 0 \text{ veya } 1 \qquad \textbf{(2.26)}$$

Burada

(2.23) numaralı denklem amaç fonksiyonu temsil etmekte olup, toplam atama maliyetini minimum yapmaktır. Amaç fonksiyonun minimum olabilmesi için 3 adet kısıtlayıcı denklemlerin sağlanması gerekmektedir.

(2.24) ve (2.25) numaralı kısıtlayıcı denklemler arz miktarı ve talep miktarı daima bire eşit veya büyük olması gerektiğini göstermektedir.

(2.26) numaralı işaret gösteren kısıtlayıcı denklemde ise karar değişkenlerin her zaman pozitif ve tamsayı olması gerektiğini göstermektedir.

Minimum yayılan ağaç bulma işlemi, olası tüm yayılan ağaçların dikkate alınmasını ve toplam ağırlığı en küçük olanın seçilmesini içerir. Bir grafiğin minimum kapsayan ağacını bulmak için Kruskal'ın algoritması ve Prim'in algoritması gibi çeşitli algoritmalar vardır. (Kara. İ, 2000)

2.7.1. Prim algoritması

Prim algoritması, Vojtěch Jarník tarafından 1930 yılında önerilmiş, daha sonra Robert C. Prim (1957) ve Edsger W. Dijkstra (1959) tarafından bağımsız olarak yeniden keşfedilmiştir. Prim algoritması, En Az Maliyetli Kapsayan ağacı bulmak için kullanılan bir algoritmadır.Prim algoritmasının adımları şu şekildedir:

Adım1: Başlangıç düğümü seçilir ve bu düğüm MYO'nın başlangıç düğümü olur.

Adım2: İşaretlenen düğümün komşuları arasından henüz Myo'ya dahil olmayan düğümler bulunur ve bu düğümlere komşu olan kenarlar incelenir.En düşük ağırlığa sahip olan bir kenar seçilir ve bu kenarın bağlı olduğu düğüm MYO'ya eklenir.

Adım3: Ağaçtaki kenar sayısı n, düğüm sayısının bir eksiği (n-1) olduğunda işlem sonlandırılır. Yani n-1 sayıda kenar ağaca eklenmişse En Az Maliyetli Kapsayan Ağaç yapısı tamamlanır. (Akay. M, Tuş. A, 2022)

2.7.2. Kruskal algoritması

Kapsayan ağaç probleminde minimum maliyetli ve tüm düğümlerden geçen yol ağının bulunabilmesi için birden çok algoritma geliştirilmiştir. Bunlardan biri Kruskal algoritmasıdır. Bu algoritma 1956 yılında Joseph Kruskal tarafından geliştirilmiştir. En Az Maliyetli Kapsayan Ağaç çözümünün Kruskal algoritmasının adımları şu şekildedir:

Adım1: Grafik üzerindeki tüm kenarlar ağırlıklarına göre küçükten başlayıp, büyüğe doğru sıralanır ve S kenar kümeyi oluştur

Adım2: S kümesinden henüz ağaçta olmayan en küçük ağırlıklı kenar seçilir. Bu kenar ağaca katıldığında döngü oluşturmuyorsa kenar ağaca dahil edilir. Eğer seçilen kenar bir düğüm oluşturmuyorsa, kenar ağaca dahil edilir ve S kümesinden çıkarılır.

Adım3: Ağaçtaki kenar sayısı n, düğüm sayısının bir eksiği (n-1) olduğunda işlem sonlandırılır. Yani n-1 sayıda kenar ağaca eklenmişse En Az Maliyetli Kapsayan Ağaç yapısı tamamlanır. (Akay. M, Tuş. A, 2022)

Prim ve Kruskal algoritmaları, En Az Maliyetli Kapsayan Ağaç yapısını bulmak için kullanılan iki farklı algoritmadır. Her iki algoritma da aynı MYO yapısını bulur, ancak yaklaşımları farklıdır. Prim algoritması, bir düğümle başlar ve ağacı adım adım büyütür. Başlangıç düğümünden başlayarak, ağaçtaki kenarlardan henüz eklenmemiş ve ağırlığı en küçük olan bir kenar seçilir. Bu seçilen kenar, ağaca bağlı olmalıdır, yani ağaçtaki bir düğüme bağlanmalıdır. Seçilen kenar, ağaca eklenir ve ağaç büyümeye devam eder. Bu işlem, ağaçtaki tüm düğümler birleşene kadar tekrarlanır. Prim algoritması, her adımda ağaç büyütme mantığıyla çalışır. Kruskal algoritması ise farklı bir yaklaşım kullanır. Başlangıçta her düğümü kendi kendine bağlı ayrı ağaçlara yerleştirir. Ardından, tüm kenarları ağırlıklarına göre küçükten büyüğe sıralar. En küçük ağırlıklı kenardan başlayarak, kenarı ağaca ekler. Eğer eklenen kenar, ağaçta bir döngü oluşturmuyorsa ve kenarın uç düğümleri farklı ağaçlarda yer alıyorsa, kenar ağaca eklenir. Kenar eklendikten sonra ağaçtaki düğümler birleşir ve işlem tekrarlanır. Kruskal algoritması, kenarların aralarında bağlantı olup olmadığına bakmaksızın ağaca kenarları ekleyerek MYO yapısını oluşturur. (Akay.M,. Tuş.A, 2022)

2.8 CPM

CPM (Critical Path Method) yöntemi, projenin tamamlanması için gerekli olan görevleri belirlemenizi sağlayan bir tekniktir. Proje yönetimindeki kritik yol, tüm projeyi tamamlamak için zamanında tamamlanması gereken en uzun faaliyetler dizisidir. Faaliyetlerin bir-birleri ile bağlantılarına lojit bağlantı denir. Bu bağlantıyı sağlayan metot kritik yörünge metodu denir. Aktivite sürelerinin kesinlikle bilindiği durumlarda kullanılan bir proje yönetimi yöntemidir. Bu yöntemin grafik uygulaması 1918 yılında Henry L. Gantt tarafından geliştirilmiştir. (Sarıca. İ, 2006)

2.8.1. Projenin Zaman Sınırlarının Belirlenmesi

Zaman sınırlarının belirlenmesi için kullanılan yöntem, matematik işlemi içerir. Bu işlemler, her olaya gelen tüm yollar boyunca faaliyetlerin tamamlanma sürelerinin toplanması ve ilgili olayın gerçekleşmesi için belirlenen zaman değerlerinin tespit edilmesini içerir. Bu zaman değerleri, "en erken olay zamanı" ve "en geç olay zamanı" olarak adlandırılır. Bu yöntem, projenin zaman sınırlarının belirlenmesi ve faaliyetlerin zamanlamasının yapılması için kullanılır. Bu sayede, projenin başlangıç ve bitiş tarihleri, faaliyetlerin ne zaman başlayıp bitmesi gerektiği ve projenin zamanlamasının nasıl yönetileceği konusunda bilgi sağlar.

2.8.2. Düğüm Noktalarının En Erken Olay Zamanlarının Belirlenmesi.

Düğüm noktalarının En erken olay zamanı (TE) kavramı, projenin zaman planlamasında önemli bir role sahiptir. Projedeki faaliyetlerin ve olayların düzgün bir şekilde sıralanması ve zamanlaması için kullanılır. Bu sayede, projenin tamamlanma süresini doğru bir şekilde tahmin edebilir ve projenin zamanında bitirilmesini sağlayabilirsiniz. En erken olay zamanı (TE_i), olayın başlangıcı olan tüm faaliyetlerin en erken başlama zamanlarına karşılık gelir ve projenin son faaliyetinin bitiş noktasını temsil eden olayın (TE_j)'si, projenin tamamlanması için gerekenen erken zamanı gösterir

En erken olay zamanı aşağıdaki gibi elde edebiliriz

$$(TE)_j = (TE)_i + \max\,(t_{ij})$$

Böylece, her olayın en erken olay zamanını belirleyebilirsiniz. Buradaki en önemli durum, şebekenin başlangıç noktasına karşılık gelen TE değeri"0" sıfır olmasıdır.

2.8.3. Düğüm Noktalarının En Geç Olay Zamanlarının belirlenmesi.

En geç olay zamanı (TL_{ij}) kavramı, projenin zaman planlamasında en erken olay zamanı gibi önemli bir role sahiptir ve projedeki faaliyetlerin ve olayların düzgün bir şekilde sıralanması ve zamanlaması için kullanılır. Bu sayede, projenin tamamlanma süresini doğru bir şekilde tahmin edebilir ve projenin zamanında bitirilmesini sağlayabilirsiniz. En geç olay zamanı (TL_i), olayın başlangıcı olan tüm faaliyetlerin mümkün olan en geç başlama zamanlarına karşılık gelir ve projenin son faaliyetinin bitiş noktasını temsil eden olayın (TL_j)'si, projenin tamamlanması için gereken en geç zamanı gösterir. En geç olay zamanında şebekenin sonundan başına doğru yanı sağından başlayarak soluna doğru ilerleyerek işlem gerçekleştirilmelidir. Her olay için, en geç olay zamanı, o olaydaki faaliyetin tamamlanma süresinin başlangıcı olan olay zamanından çıkarılmasıyla elde edilir. Eğer olayda birden fazla faaliyet başlangıcı varsa, faaliyetler arasından sadece tamamlanma süresinin en küçük olan faaliyeti seçilir.

En geç olay zamanı aşağıdaki gibi elde edebiliriz

$$(TL)_i = (TL)_j - min(t_{ij})$$

Böylece, her olayın en erken olay zamanını belirleyebilirsiniz. Burada önemli olan bir durum, şebekedeki olayların sonucu en erken olay zamanı değerinin en geç olay zamanı değerine eşit olduğudur.

2.8.4. Faliyetlerin En Erken Başlama Süresi

Bir faaliyetin en erken başlama süresi, faaliyetin başlayabileceği en erken zamana denk gelen süreyi ifade eder. Bu süre, faaliyetin başlama noktasını temsil eden olayın en erken olay zamanına karşılık gelir. İngilizce olarak "Early Starting Time" ifadesi ES olarak kısaltılır.

En Erken Başlama süresinin formülü aşağıdaki gibidir.

$$ES = (TE)_i \quad \textbf{(2.27)}$$

2.8.5. Faaliyetlerin En Erken Bitirme Süresi

En erken bitirme süresi bir faaliyetin tamamlanabileceği en erken süreyi ifade ederse faaliyetin başlama zamanına faaliyet süresinin eklenmesiyle elde edilir. İngilizce olarak “Early Finish Time” ifadesi EF olarak kısaltılır.

En Erken Bitirme Süresinin formülü aşağıdaki gibidir.

$$EF = ES + t_{ij} = (TE)_i + t_{ij} \quad \textbf{(2.28)}$$

2.8.6. Faaliyetlerin En Geç Başlama Süresi

En geç bitirme süresi, bir faaliyetin tamamlanması gereken son tarihi ifade eder. Bu süre, faaliyetin zamanında tamamlanması için projenin toplam süresi içinde son tarih olarak belirlenir ve faaliyetin gecikme süresini ifade edebilir. İngilizce olarak "Late Finish Time" ifadesi LF olarak kısaltılır.

En Geç Başlama Süresinin formülü aşağıdaki gibidir.

$$LF = (TL)_j \quad \textbf{(2.29)}$$

2.8.7. Faaliyetlerin En Geç Bitirme Süresi

En geç başlama süresi bir faaliyetin başlayabileceği en son zamana denk gelen süreyi ifade eder. Bu süre, bir faaliyetin bitirilmesi gereken en son tarihten faaliyet süresi çıkarılarak elde edilir. İnglizce olarak "Late Start Time " ifadesi LS olarak kısaltılır.

$$LS = LF - t_{ij} = (TL)_j - t_{ij} \quad \textbf{(2.30)}$$

2.8.8. Kritik Yolun Belirlenmesi

Herhangi bir projede, başlangıç noktasından bitiş noktasına gelen tüm yollar içerisinde yörünge süre toplamı en uzun olan yol, kritik yörünge olarak adlandırılır. Kritik yörünge üzerinde yer alan faaliyetler ise kritik faaliyet olarak tanımlanır. CPM yönteminde kritik yolun uzunluğu projenin minimum bitirilme süresine eşittir. Bu faaliyetlerin gecikmesi projenin gecikmesine neden olabilir. Ondan dolayı kritik yolun üzerindeki faaliyetlere daha fazla dikkat edilmelidir. Bu faaliyetlerin kritik faaliyet olabilmesi için aşağıdaki koşulların sağlanması gereklidir. (Cenk, D. 2007)

1. $(TE)_i$=$(TL)_i$
2. $(TE)_j$=$(TL)_j$
3. $(TE)_j$-$(TE)_i$=$(TL)_j$-$(TL)_i$=$(t)_{ij}$

Eğer yukarıdaki koşullar sağlanırsa, (i, j) faaliyeti kritik yol üzerinde demektir. Bu koşullardan faydalanarak, $(t)_{ij} = 0$ bulunması gerek ki, toplamı sıfır olan faliyetler kritik faliyetlerdir denir.

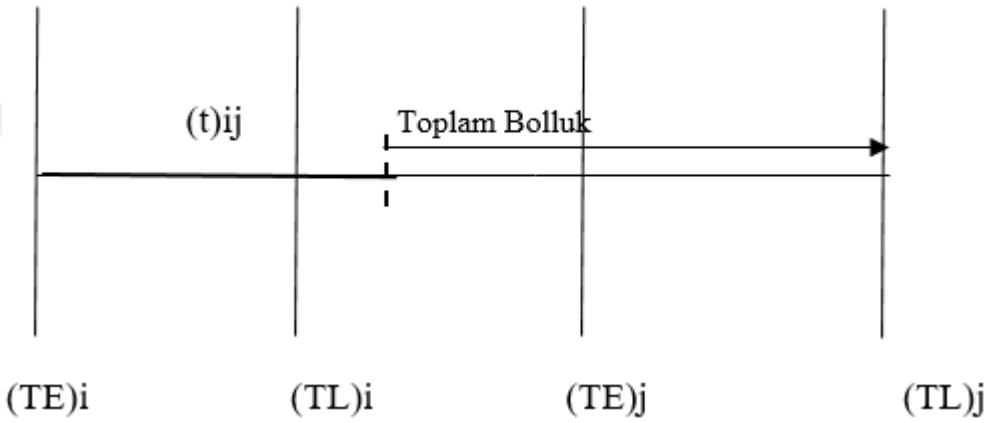
2.8.9. Faaliyetlerin bolluk sürelerinin hesabı

CPM kullanıldığında, projedeki bazı faaliyetler kritik olmayan ve geciktirilebilir faaliyetler olarak kabul edilir. Bu faaliyetler, projenin zaman sınırları içinde esneklik gösterir. Şebekenin programlama aşamasında, bu faaliyetler için zaman aralıkları belirlenir. Tamamlanma süresi $(t)_{ij}$ olan i-j düğüm noktaları arasındaki faaliyet için,

(TE)i, (TL)i,(TE)j ve (TL)j arasında bağlantı vardır. Bu bağıntılar dikkate alınarak bolluğun tanımlanmasına olanak sağlar. Faaliyetin toplam bolluk, serbest bolluk, bağımsız bolluk ve ara bolluk olmak üzere dört bolluk süreleri vardır.

Toplam Bolluk: Bir faaliyetin tamamlanma süresindeki gecikmeyi ifade eder. Bu, faaliyetin başlama zamanının ve bitiş zamanının esneklik aralığını gösterir. Toplam bolluk, faaliyetin bitiş süresinin projenin toplam süresine olan etkisini ölçer. Bir faaliyetin toplam bolluğu, TEi ve TLj değerleri arasındaki fark olarak hesaplanır:

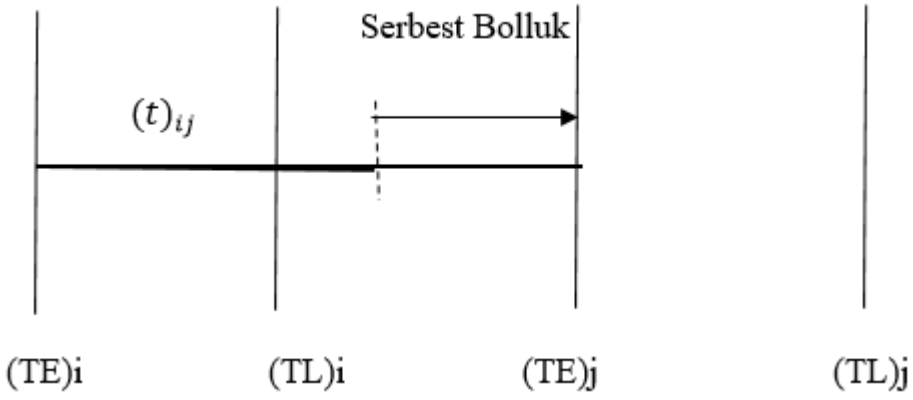
Toplam Bolluk = $(TL)_j$ - $[(TE)_i + (t)_{ij}]$



Şekil 2. 14 Toplam Bolluk Sürenin Grafik Gösterimi (İbrahim.S, 2006)

Serbest Bolluk: Bir faaliyetin başlangıç süresindeki gecikmeyi ifade eder. Serbest bolluk, bir faaliyetin başlama zamanının esneklik aralığını gösterir. Serbest bolluk, faaliyetin bitiş süresinin diğer bağımlı faaliyetlere olan etkisini ölçer. Bir faaliyetin serbest bolluğu, (TE)i ve (TE)j değerleri arasındaki fark olarak hesaplanır:

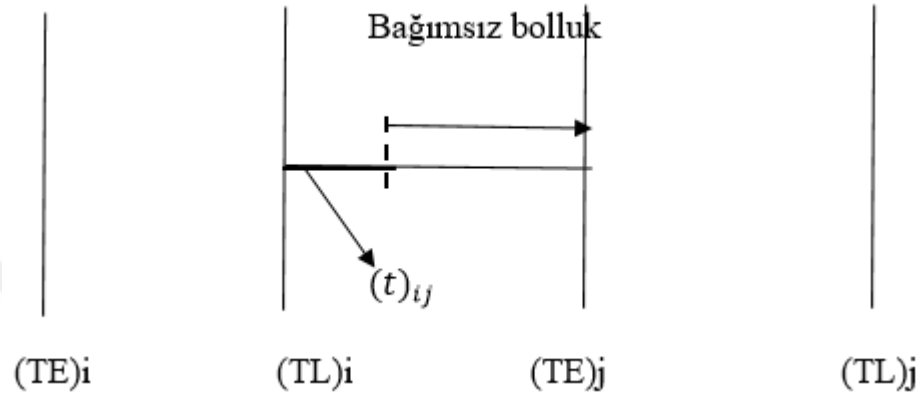
Serbest Bolluk =$(TE)_j - [(TE)_i + (t)_{ij}]$



Şekil 2. 15 Serbest Bolluk Sürenin Grafik Gösterimi (Sarıca. İ, 2006)

Bağımsız Bolluk: Bir faaliyetin başlama süresindeki gecikmeyi ifade eder, ancak bu gecikme diğer bağımlı faaliyetleri etkilemez. Bağımsız bolluk, bir faaliyetin başlama zamanının esneklik aralığını gösterir. Bağımsız bolluk, faaliyetin bitiş süresinin diğer bağımlı faaliyetlere olan etkisini dikkate almaz. Bir faaliyetin bağımsız bolluğu, (TL)i ve (TE)j değerleri arasındaki fark olarak hesaplanır:

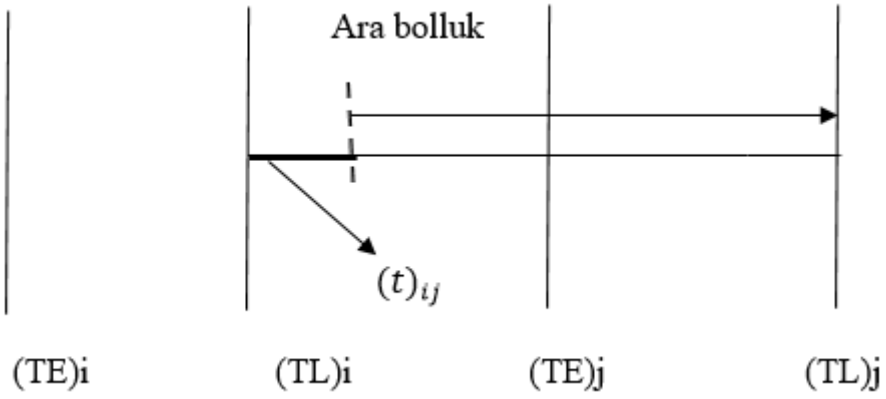
Bağımsız Bolluk = TEj - (TLi + $(t)_{ij}$)



Şekil 2. 16 Bağımsız Bolluk Süreninin Grafik Gösterimi (Sarıca. İ, 2006)

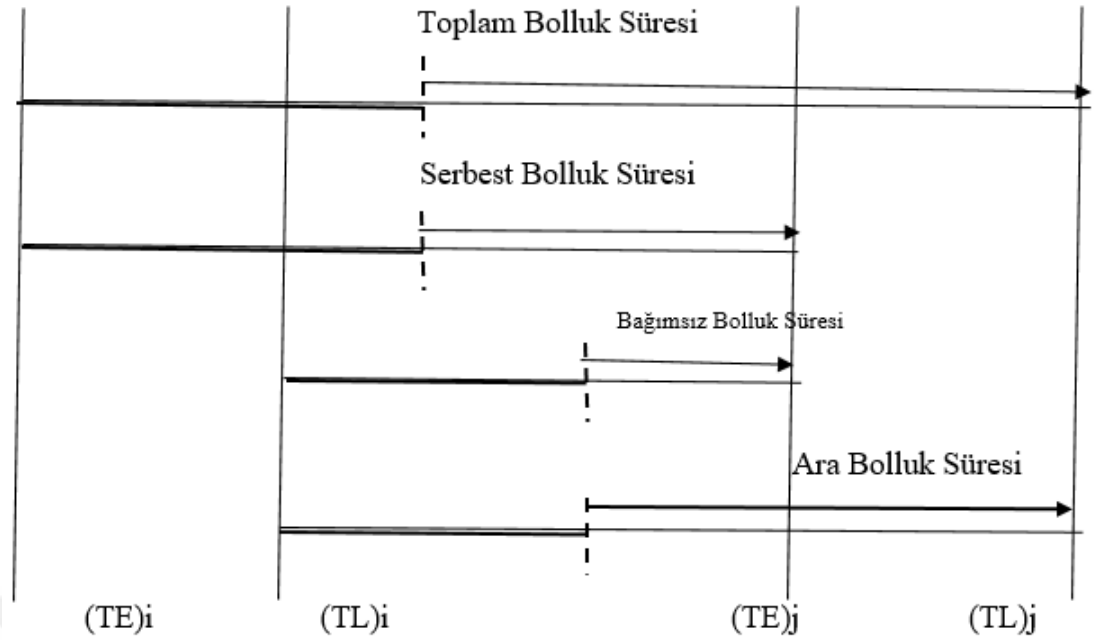
Ara Bolluk: İki ardışık faaliyet arasındaki esneklik zamanını ifade eder. Ara bolluk, iki ardışık faaliyetin bitiş ve başlama süreleri arasındaki farkı temsil eder. Bu bolluk, iki faaliyetin bitiş ve başlama sürelerinin esneklik aralığını gösterir. Ara bolluk, (TE)j ve (TL)i değerleri arasındaki fark olarak hesaplanır:

Ara Bolluk = TLi – TEj



Şekil 2. 17 Ara Bolluk Sürenin Grafik Gösterimi (Sarıca. İ, 2006)

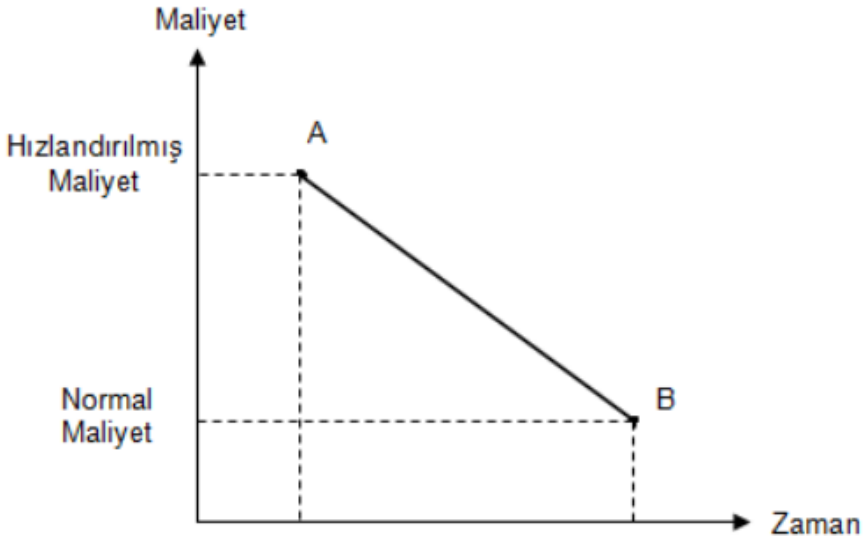
Dört farklı bolluklar incelendikten sonra, birbirlerinin arasındaki ilişkileri aşağıdaki gibi ifade edebiliriz.



Şekil 2. 18 Bolluklar Arası İlişkinin Grafik Gösterimi (Sarıca. İ, 2006)

2.8.10. Zaman ve Maliyet İlişkisi ve Yoğun Çalışma

Bir projenin tamamlanması süresi ve maliyeti arasında doğrudan bir ilişki vardır. Tamamlama süresinin sınırlandırılması için çeşitli yöntemler kullanılabilir. Zaman-maliyet analizi yapılırken, hedefler belirlenerek zamanla mı yoksa maliyetin mi daha uygun olduğuna genel karar verilir. Bazı projelerde zaman kısıtlıdır ve önceden yazılmış bir süre kritik öneme sahiptir. Proje süresinin kısaltılması için daha fazla kaynak kullanımı veya kaynak yüklemesi gibi yönlendirmeye başvurulduğunda, bu durum ek maliyetlere yol açabilir. Aşağıda zaman-maliyet ilişkisini gösterilmiştir.



Şekil 2. 19 Zaman-Maliyet İlişkisi Gösterimi (Bakışkan.E , 2019)

Projenin zaman-maliyet ilişkileri Şekil 2.19'da gösterildiği gibi grafiğin doğrusal ve sürekli olduğu varsayılarak analiz yapılmaktadır. Grafikte de görüldüğü gibi projenin tamamlanması süresi ve maliyeti arasında doğrudan bir ilişki vardır. Sonuç olarak, hızlandırma ve uzatma işlemlerinin belirli sınırları içinde yapılabildiği bir projede, kritik yol üzerindeki hedeflere odaklanarak hızlandırma işlemlerinin gerçekleştirilmesi önemlidir. Bu şekilde tamamlama süresi kısaltılabilir. (Bakışkan, E. 2019)

CPM'de, birden çok kritik yolu oluştuğu zaman, bu yollarda yoğun çalışma yapılmalıdır. Projenin bir faaliyeti üzerinde yoğun çalışma yapılırken günlük maliyet artışına masraf maili (mm) denir ve

$$mm = \frac{M_n - M_y}{T_n - T_y} \quad \textbf{(2.31)}$$

gibi hesaplanır. Burada

M_n Projenin normal maliyeti

M_y Projenin yoğun maliyeti

T_n Projenin normal tamamlanma süresi

T_y Projenin yoğun çalışma süresidir.

Yoğun çalışma işlemi projenin hedeflenen süresine ulaşılabilmesi için kullanılan bir yöntemdir. Yoğun çalışma işlemi, projenin kritik yolundaki faaliyetler üzerinde uygulanır ve bu faaliyetlerin süresini kısaltmayı amaçlar. Projede yoğun çalışma işlemi; sonucu, faaliyetlerin esneklik süreleri azalır, proje yönetimi zorlaşır.

2.9. PERT

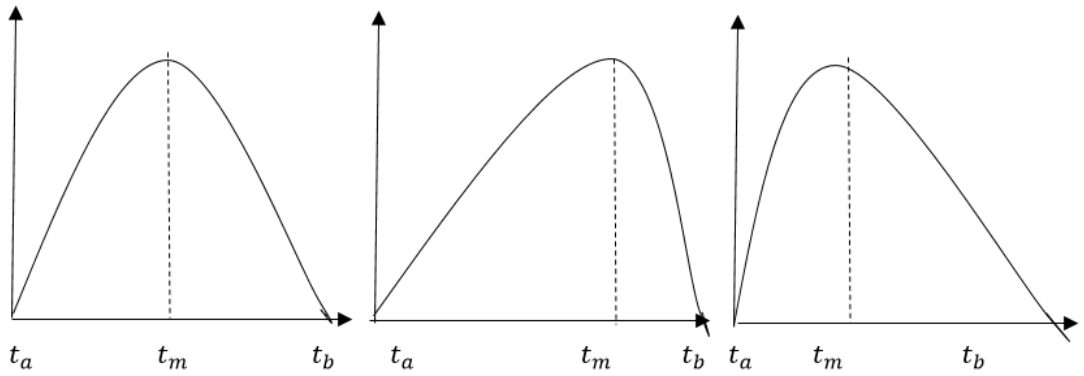
PERT (Program Evaluation and Review Technique), Program Değerlendirme ve Gözden Geçirme Tekniği anlamına gelmektedir. Bu yöntem projelerin tamamlanması için gerekli olan zaman, kaynak ve maliyetlerin hesaplanması, projelerin takip edilmesi ve kontrol edilmesi için kullanılan bir ağ planlama tekniğidir. CPM yönteminde faaliyetlerin zaman süreleri kesin olarak bilinmekte, fakat uygulama kısmında faaliyetlerin kesin olarak bilinmesi olanak dışıdır. Bu durumlarda olasılıklı zaman süreleri söz konusudur. İşte böyle durumlarda PERT analizi kullanılmaktadır. PERT, her bir görevin tamamlanması için gerekli olan zamanın, beklenen değeri, en erken tamamlanma süresi ve en geç tamamlanma süresi gibi çeşitli istatistiksel ölçümlerini hesaplamak için kullanılır. (Kısmet.C, Ender.G, 2023)

2.9.1 Süresi belirsiz faaliyetlerin tahmini

PERT yönteminde, bir faaliyet için aşağıda verilen 3 tahmin süresinin bilinmesi gerekir. Bunlar:

1. *Faaliyetin en erken tamamlanma süresi* (t_a): Projenin planlanmasında her şeyin yolunda gittiği durumlarda, faaliyetin en çabuk tamamlanacağı süredir. Faaliyetin en erken tamamlanma süresi öyle bir süredir ki, bu faaliyet %99'dan daha kısa sürede bitirilemez.
2. *Faaliyetin en geç tamamlanma süresi* (t_b): Projenin en kötü durumlarda en uzun sürede faaliyetin bitirilme süresidir. Faaliyetin en geç tamamlanma süresi öyle bir süredir ki, bu faaliyet %99'dan daha uzun sürede bitirilemez.
3. *En yüksek ihtimal ile tamamlanma süresi Muhtemel süre* (t_m): faaliyet sürelerinin tahmin edilmesiyle ilgili bir açıklamadır. Faaliyetin muhtemel süresi, normal koşullar altında gerçekleştiği ve geçmiş deneyimlere dayanarak tahmin edildiği varsayılan süredir.

Projede her faaliyetin olasılık dağılımı beta dağılımıdır. Faaliyetin en erken tamamlanma süresi ve en yüksek ihtimal ile tamamlanma süresi arasındaki mümkün olan tüm aktivite sürelerini kapsamaktadır. Projede faaliyetin mühtemel süresi, en erken tamamlanma süresi ile en geç tamamlanma süresinin tam ortasında olması şart değildir. Yanı beta dağılımı simetrik dağılımdır. Bazı durumlarda beta dağılımı sağa çarpık ya da sola çarpık dağılım olabilir.



Şekil 2. 20 Beta Dağılımın Üç Farklı Gösterimi

2.9.2. Faaliyetlerin Beklenen Tamamlanma Süreleri ve Varyanslarının hesaplanması

PERT yönteminde beklenen tamamlanma süresi, üç tahmini süre olan en erken tamamlanma süresi, en geç tamamlanma süresi ve muhtemel sürelerinin ağırlıklı

ortalaması alınarak elde edilir. Şekil:20'deki beta dağılımında tahmin edilen 1 faaliyetin beklenen tamamlanma süresi değeri olan t_e'yi aşağıdaki gibi

$$\bar{x} = t_e = \frac{t_a+4t_m+t_b}{6} \quad \textbf{(2.32)}$$

hesaplamak mümkündür.

(2.32) formülün yeterliliğinin ispatı yapılamamaktadır. Standart dağılımından esinlenerek, PERT tekniğinde faaliyetlerin tamamlanma sürelerinin dağılımı, (a,b) aralığında en geç tamamlanma süresi ile en erken tamamlanma süresinin farkı dağılımın 6 standart sapması (σ)'ya eşittir. . Bu durumda dağılımın standart sapması

$$6\sigma=t_b-t_a \quad \textbf{(2.33)}$$

Buradan yola çıkarak standart sapma formülü

$$\sigma_{t_e} = \frac{(t_b-t_a)}{6} \quad \textbf{(2.34)}$$

olarak hesaplanır.

Faaliyetlerin belirsizliğini ifade eden varyanslar dağılımın standart sapması varyansının karekökü olduğundan dolayı

$$\sigma_{t_e}{}^2 = (\frac{(t_b-t_a)}{6})^2 \quad \textbf{(2.35)}$$

(2.35)'deki gibi faaliyetin varyansı elde edilir.

(2.34) ve (2.35) deki formüller ile her faaliyet için beklenen süreler ve varyans hesaplanır

2.9.3. Proje Tamamlanma Süresinin Analizi

PERT yönteminde, projenin kritik yolunun belirlenmesi için öncelikle başlangıçtan bitişe kadar olan tüm yollardaki faaliyetin tamamlanma süresi bulunur. Bu yolların arasındaki en uzun toplam süre kritik yoldur. Kritik yol üzerindeki faaliyetlerin tamamlanma süreleri toplamı (μ), projenin beklenen tamamlanma süresini temsil ederken, kritik yol üzerindeki faaliyetlerin varyanslarının toplamı ($\sigma_{t_e}{}^2$), projenin varyansını temsil eder.

Projenin beklenen tamamlanma süresi:

$$\mu = \sum\bar{x} = \sum t_e \quad \textbf{(2.36)}$$

Projenin varyansı:

$$\sigma_{t_e}{}^2 = \sum \sigma_{t_e}{}^2 \quad \textbf{(2.37)}$$

Merkezi Limit Teoremi 'ne göre, dağılımın şekline bakılmaksızın, n tane ($n \geq 30$) değişkenin toplamı, normal dağılıma sahip değişken olarak kabul edilebilir. Bu teoreme göre, projenin tamamlanma süresi kritik yol üzerindeki faaliyetlerin

tamamlanma sürelerinin toplamıdır ve bu süre normal dağılım gösterir. Bu durumda, standart normal dağılım tablosunu kullanarak belirli bir hedeflenen süre için gerçekleşme olasılığını hesaplayabiliriz. Hedef süre olarak belirlenmiş bir değer (Y) için gerçekleşme olasılığını hesaplamak için, kritik yolun varyans değeri kullanılarak dağılımın standart sapması belirlenir. Standart sapmanın standart 'z' puanı kullanılarak hesaplanabilmesi için bir dönüşüm yapılır. Bu dönüşüm aşağıdaki formülle hesaplanır:

$$Z = \frac{Y-\mu}{\sigma_{t_e}} \qquad \textbf{(2.38)}$$

Burada

Y: projenin tamamlanması için belirlenen hedef süresi

μ: Projenin ortalama tamamlanma süresi

Z: Değeri, normal dağılım eğrisi üzerinde hedef süreye ulaşma ihtimalidir.

Ayrıca, kritik yolun tamamlanma süresinin istatistiksel bir yaklaşım olan normal dağılımı takip ettiğini ve ortalama süre değerinin normal dağılımın ortalamasıyla uyumlu olduğunu ifade eder. (Z = 0). Bu durumda, projenin tamamlanma süresinin ortalama değeri üzerindeki dağılımı dikkate alarak, projenin bu ortalama süre içinde tamamlanma olasılığı %50 olarak belirlenir. Yani, projenin kritik yolunun, ortalama süreden daha uzun bir sürede tamamlanma ihtimali %50'dir. (Sarıca.İ, 2006)

Ek:1'de Standart normal dağlım tablosu verilmiştir. Tablonun kullanım şekli sütunlarda tam kısım ve birinci ondalık kısmı, satırlarda virgülden sonraki ikinci basamağı göstermektedir. Eldeki bilgiye göre ilgili satıra ve sütüne karşılık gelen değer ise aradığımız Z tamamlanma olasılığı değeridir.

2.9.4 PERT Metodunda Beta Dağılımı

Beta dağılımı, özellikle sınırlı dağılım aralıkları olduğu projenin uygulamaları için sıklıkla kullanılan dağılımdır. Gerçek verilerin, birçok çalışmada beta dağılımı ile yeterli düzeyde ifade edilebildiği gözlemlenmiştir. İstatistikte, beta dağılımı [0,1] aralığında tanımlanır ve genellikle iki pozitif (α ve β) şekil parametresiyle ifade edilir.

$$f(x) = \frac{(\alpha+\beta+1)!}{\alpha!\beta!}x^{\alpha}(1-x)^{\beta} \qquad 0<x<1 \qquad \textbf{(2.39)}$$

α ve β parametreye sahip beta dağılımlımında X değişkeni için:

Beklenen değeri: $E(x)=\frac{\alpha}{\alpha+\beta}$ **(2.40)**

Varyansı: $Var(x)=\frac{\alpha\beta}{(\alpha+\beta^2(\alpha+\beta+1)}$ **(2.41)**

Çarpıklık değeri: $\frac{2(\beta-\alpha)\sqrt{\alpha+\beta+1}}{(\alpha+\beta+2)\sqrt{\alpha\beta}}$ **(2.42)**

Fazladan basıklık değeri: $6*\frac{\alpha^3-\alpha^2(2\beta-1)+\beta^2(\beta+1)-2\alpha\beta(\beta+2)}{\alpha\beta(\alpha+\beta+2)(\alpha+\beta+3)}$ **(2.43)**

formülleri yukarıdaki gibi elde edilir. Bu formüllerden yola çıkarak dağılımın ortalaması ve varyansı elde edilir.

$$\bar{x} = t_e = \frac{t_a + 4t_m + t_b}{6}$$

$$\sigma_{t_e}{}^2 = (\frac{(t_b - t_a)}{6})^2$$

2.9.5. PERT ve CPM Yöntemlerinin Karşılaştırılması

Her iki teknik de proje yönetiminde önemli araçlardır ve projenin gereksinimlerine ve özelliklerine bağlı olarak tercih edilebilir. PERT, belirsizliklerin ve değişkenliklerin yüksek olduğu projelerde daha etkilidir, CPM ise daha belirli ve kesin süre tahminlerine dayanan projelerde daha uygundur. (Bakışkan.E , 2019)

Gelişen teknoloji devrinde Python programlama dilinin önemi oldukça büyük. İşte Python'ın gelişen teknoloji devrindeki önemli noktalarından bazıları:

Kolay Okunabilirlik ve Öğrenme Kolaylığı: Python, sadece anlaşılır bir sözdizimine sahip olmakla kalmaz, aynı zamanda basit ve anlaşılır bir dil yapısına da sahiptir.

Geniş Kullanım Alanları: Python, geniş bir kullanım alanına sahiptir. Veri analizi, yapay zeka, makine öğrenimi, web geliştirme, bilimsel hesaplama, otomasyon, ağ programlama ve daha birçok alanda Python kullanılabilir. Bu çok yönlülük, Python'ın teknoloji devrindeki önemini artırır.

Veri Analizi ve Yapay Zeka: Python, veri analizi ve yapay zeka alanında çok popülerdir. Numpy, Pandas ve Scikit-learn gibi kütüphaneler, veri analizi ve makine öğrenimi için güçlü araçlar sunar.

Web Geliştirme: Python, web geliştirme alanında da etkili bir seçenektir.

Topluluk ve Kütüphaneler: Python, geniş bir topluluğa ve zengin bir kütüphane ekosistemine sahiptir. Bu, kullanıcıların birçok sorunun üstesinden gelmelerini ve projelerinde hızlı ilerlemelerini sağlar. Python topluluğu, sürekli olarak yeni kütüphaneler, araçlar ve çözümler üretir.

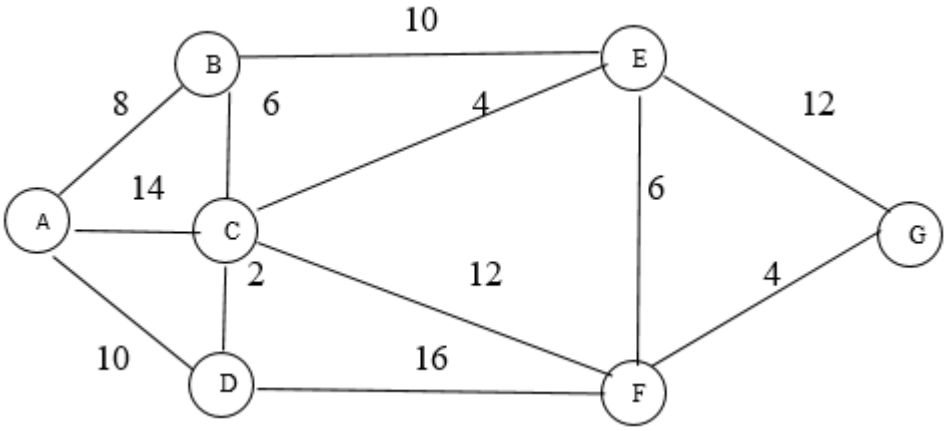
Bu noktalar, Python programlama dilinin gelişen teknoloji devrindeki önemini vurgular. Python'ın esneklik, kolay öğrenilebilirlik, geniş kullanım alanları ve güçlü kütüphaneleri sayesinde birçok projede tercih edilen bir dil haline gelmiştir.(Aydemir, E. 2021)

3.UYGULAMA VE BULGULAR

Çalışmanın bu bölümünde ağ optimizasyon modelleri olan en kısa yol problemi, maksimum akış problemi, ulaştırma problemi, atama problemi, kapsayan ağaç problemi ve CPM/PERT modelleri ile ilgili, literatürde mevcut olan uygulamalardan bazıları ele alınmıştır. Bu uygulamalar Python ile çözülecektir. Mevcut çözüm değerleri ile, Python kullanılarak elde edilen çözüm değerleri karşılaştırılacak ve önerilerde bulunulacaktır.

3.1. En Kısa Yol Problemi Örnek Çözümü

1.düğümden 7.düğüme olan en kısa yolu belirleyiniz?



Şekil 3. 1 En Kısa Yol Problemi Örneği

Şekil 3.1'den yararlanarak 1.düğüme kalıcı atama yapılarak, tüm düğümler için atamalar yapılmış ve 6.iterasyonda çözüme ulaşılmıştır. Çözüm tablosu aşağıdaki gibidir.

Tablo 3. 1 En kısa yol Rotası

Düğüm	En Kısa Yol	En kısa Yol Rotası
2	8	A-B
3	12	A-D-C
4	10	A-D
5	16	A-D-C-E
6	22	A-D-C-E-F
7	26	A-D-C-E-F-G *

Tablo 3.1'de görüldüğü gibi A.düğümden G.düğüme kadar olan en kısa yol A-D-C-E-F-G düğümlerden geçiyor. Bu problemin en kısa yolu A-D-C-E-F-G düğümlerdir. (Timor. M, 2020:401)

3.1.1.Örneğin Python ile çözümü çözümü ve karşılaştırılması

```
import heapq
def dijkstra(graph, start):
    distances = {node: float('inf') for node in graph}
    distances[start] = 0
    queue = [(0, start)]
    previous_nodes = {}
    while queue:
        current_distance, current_node = heapq.heappop(queue)

        if current_distance > distances[current_node]:
            continue
        for neighbor, weight in graph[current_node].items():
            distance = current_distance + weight
            if distance < distances[neighbor]:
                distances[neighbor] = distance
                previous_nodes[neighbor] = current_node
                heapq.heappush(queue, (distance, neighbor))
    return distances, previous_nodes
def shortest_path(previous_nodes, end):
    path = []
    current_node = end
    while current_node is not None:
        path.append(current_node)
        current_node = previous_nodes.get(current_node)
    return list(reversed(path))
# Örnek bir graf oluşturalım
graph = {
    'A':{'B':8,'C':14,'D':10},
     'B':{'A':8,'C':6,'E':10},
     'C':{'A':14,'B':6,'D':2,'E':4,'F':12},
     'D':{'A':10,'C':2,'F':16},
     'E':{'B':10,'C':4,'F':6,'G':12},
     'F':{'C':12,'D':16,'E':6,'G':4},
     'G':{'E':12,'F':4}
}
start_node = 'A'
end_node = 'G'
distances, previous_nodes = dijkstra(graph, start_node)
shortest_distance = distances[end_node]
shortest_path = shortest_path(previous_nodes, end_node)
print(f"En kısa yol: {shortest_path}")
print(f"Toplam maliyet: {shortest_distance}")    }
```

Şekil 3. 2 En Kısa yol problemi Python Sözde Kod Gösterimi

Python programlama dilinde bu problem Dijkstra algoritması kullanılarak, başlangıç düğümü 'A' ve hedef düğümü 'G' arasındaki en kısa yol ve toplam maliyeti hesaplanır. Sonuçlar ekrana yazdırılır.

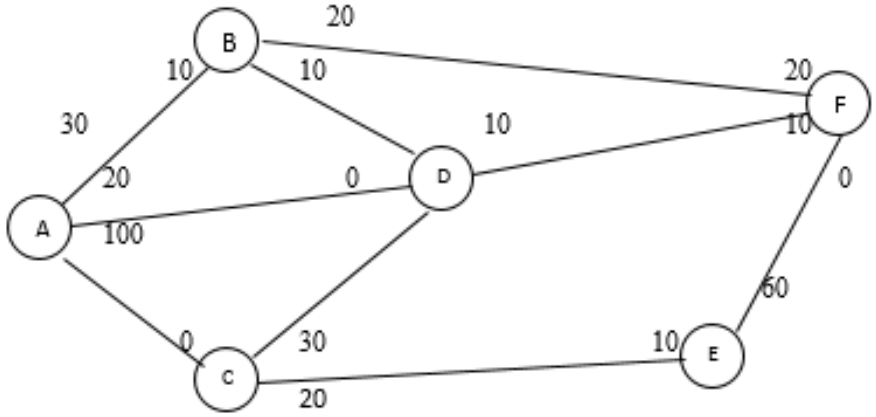
```
En kısa yol: ['A', 'D', 'C', 'E', 'F', 'G']
Toplam maliyet: 26
```

Şekil 3. 3 En Kısa Yol Problemi Python Programı Sonucu

Bu problemi Python programlama dilinde çözmek için heapq kütüphanesi kullanılmıştır. Bu problemi çözmek için Dijkstra algoritması kullanılmıştır. Çözüm sonucu en kısa yolu ['A', 'D', 'C', 'E', 'F', 'G']düğümlerdir.

3.2. Maksimum Akış Problemi Örnek Çözümü

Bir doğal park yöneticileri parkı gezmeye ve çevreyi görmeye gelen ziyaretçileri doğal ortamı bozmadan gezdirebilmek için park içinde kurdukları bir sistemle gezdirmeyi planlıyorlar. Burada amaç mümkün olduğu kadar çok sayıdaki ziyaretçinin en fazla görülecek yeri görerek en yüksek yere ulaşıp parkı en yüksek yerden görmelerini sağlamaktır. Aşağıdaki şekilde parkın giriş yerinden en yüksek tepeye ulaşılabilecek bağlantılar gösterilmiştir. Ağdaki düğümler park içindeki çeşitli dinlenme ve sosyal olanakların bulunduğu yerlerdir ve bağların yanlarindeki değerler de taşıma kapasitelerini belirtmektedir. Ziyaretçiler taşıma araçları ile ilerlerken bu düğümlerden geçmekte, inip binebilmektedirler. Bu bilgilere göre maksimum akış algoritması ile taşıma kapasitelerinin planı hazırlayınız?(Özkan. Ş, 2012)



Şekil 3. 4 Maksimum Akış Problemi Örneği

Problemin çözüm aşamaları aşağıdaki gibidir:

Adim:1 İlk önce başlangıç düğümden bitiş düğüme ulaşmak için herhangi bir akış yolu seçilir.

Adım:2 Bu yol üzerindeki en düşük kapasiteye sahip bağlantı bulunur. Bu miktar ilave edilecek maksimum kapasite anlamını taşır.

Adım:3 bu yol üzerinde yer alan herhangi bir düğüm için akış yönündeki kapasite azaltılır yada eklenir.

Adım:4 bu işlemler akış yönünde hiçbir işlem yapılmayınca kadar devam eder.

Yukarıdaki adımlar izlenerek maksimum akış kapasitesi 50 birim olarak bulunmuştur.

3.2.1.Örneğin Python ile çözümü çözümü ve karşılaştırılması

```
from collections import defaultdict
class Graph:
    def __init__(self, graph):
        self.graph = graph
        self.ROW = len(graph)

    def bfs(self, s, t, parent):
        visited = [False] * (self.ROW)
        queue = []
        queue.append(s)
        visited[s] = True
        while queue:
            u = queue.pop(0)
            for ind, val in enumerate(self.graph[u]):
                if visited[ind] == False and val > 0:
                    queue.append(ind)
                    visited[ind] = True
                    parent[ind] = u
        return True if visited[t] else False
    def fordFulkerson(self, source, sink):
        parent = [-1] * (self.ROW)
        max_flow = 0
        while self.bfs(source, sink, parent):
            path_flow = float("Inf")
            s = sink
            while s != source:
                path_flow = min(path_flow, self.graph[parent[s]][s])
                s = parent[s]
            max_flow += path_flow
            v = sink
            while v != source:
                u = parent[v]
                self.graph[u][v] -= path_flow
                self.graph[v][u] += path_flow
                v = parent[v]
        return max_flow
# Örnek kullanım
graph = [[0, 30, 100, 20, 0, 0],
         [10, 0, 0, 10, 0, 20],
         [0, 0, 0, 30, 20, 0],
         [0, 0, 10, 0, 0, 10],
         [0, 0, 10, 0, 0, 60],
         [0, 20, 0, 10, 0, 0]]

g = Graph(graph)
source = 0
sink = 5
print("Maksimum akış:", g.fordFulkerson(source, sink))
```

Şekil 3. 5 Maksimum Akış Maliyetli Problem Python Sözde Kod Gösterimi

```
Users\bagul\.vscode\extensions\ms-python.python-
2023.10.1\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher'  '50271'
'--' 'C:\Users\bagul\OneDrive\Masaüstü\enkısayol\maxflowdeneme.py'
Maksimum akış: 50
```

Şekil 3. 6 Maksimum Akış Problemi Python Programı Sonucu

Python programlama dilinde bu problem Ford-Fulkerson algoritması kullanılarak, başlangıç düğümü 'A' ve hedef düğümü 'F' arasındaki maksimum akış algoritması ile taşıma kapasitelerinin planı hesaplandı Sonuçlar ekrana yazdırıldı. Görüldüğü gibi Maksimum Akış kapasitesi 50 birim olarak bulunmuştur.

3.3. Ulaştırma Problemi Örnek Çözümü

Temel Mobilya. D, E ve F şehirlerindeki fabrikalarından A, B ve C şehirlerindeki mağazalarına ürettiği masalardan nakletmek istemektedir. Her bir fabrikanın aylık üretim kapasitesi ve mağazaların aylık talepleri bilinmekte ve bu kapasite ve taleplere uygun maliyetli taşıma yapılmak istenmektedir. Firma tarafından, üretim maliyetinin her üç fabrikada da yaklaşık olarak eşit olduğu; dolaysıyla taşımada önemli olan faktörün her kaynaktan-hedefe birim taşıma maliyeti olduğunu tespit edilmiştir. Birim taşıma maliyetleri, fabrikalara göre üretim kapasiteleri ile mağaza talepleri aşağıda verilmiştir. Ulaştırma probleminin uygun çözümünü Vogel yaklaşımı ile çözünüz?

Tablo 3. 2 Ulaştırma Problemi Örneği

Fb mğz	A	B	C	Fabrika üretimi(Arz)
D	5	4	3	100
E	8	4	3	300
F	9	7	5	300
Mağaza talebi	300	200	200	700 / 700

Tablo 3.2'den yararlanarak çözüm elde edilmiştir. İlk önce DP modeli ve kısıtlar aşağıdaki gibi oluşturulmuştur.

DP Modeli: x_{ij}=1 eğer $Fabrika_i$ grubu $Mağaza_i$ araştırmasına atanırsa; 0 aksi halde olsun

$$min\ Z = 5X_{1A} + 4X_{1B} + 3X_{1C} + 8X_{2A} + 4X_{2B} + 3X_{2C} + 9X_{3A} + 7X_{3B} + 5X_{3C}$$

Kısıtlar

$$X_{1A} + X_{1B} + X_{1C} \leq 100\,, \qquad Fabrika\ üretimi\ 1için$$
$$X_{2A} + X_{2B} + X_{2C} \leq 300, \qquad Fabrika\ üretimi 2\ için$$
$$X_{3A} + X_{3B} + X_{3C} \leq 300, \qquad Fabrika\ üretimi 3\ için$$

$$X_{1A} + X_{2A} + X_{3A} \geq 300, \qquad Mağaza\ talebi 1\ için$$
$$X_{1B} + X_{2B} + X_{3B} \geq 200\,, \qquad Mağaza\ talebi 2\ için$$
$$X_{1C} + X_{2C} + X_{3C} \geq 200, \qquad Mağaza\ talebi 3\ için$$

Çözüm sonucunda toplam maliyeti: (5*100)+(4*200)+3*100)+9*200)+5*200)=3900 bulunmuştur. İşlem sonunda çözümün optimal olup olmadığı kontrol edilmiştir. DB=2, DC=2, EA=1, FB=1 ≥ 0 ilerleme değerlerinin tümü sıfırdan büyük veya eşit olduğuna göre çözüm optimaldir denir. (Timor. M, 2020:172)

3.3.1.Örneğin Python ile çözümü çözümü ve karşılaştırılması

```
from pulp import LpProblem, LpVariable, LpStatus, LpMinimize, GLPK, value
M=3
N=3
a=range(1,M+1)
a1=range(M)
b=range(1,N+1)
b1=range(N)
xindx=[(a[i],b[j])for j in b1 for i in a1]
model=LpProblem("Transportation LP Problem", LpMinimize)
x=LpVariable.dicts("X", xindx,0,None)
model+=5.0*x[1,1]+4.0*x[1,2]+3,0*x[1,3]+8,0*x[2,1]+4,0*x[2,2]+3,0*x[2,3]+9,0*
x[3,1]+7,0*x[3,2]+5,0*x[3,3],
model+=x[1,1]+x[1,2]+x[1,3]<=100.0,
model+=x[2,1]+x[2,2]+x[2,3]<=300.0,
model+=x[3,1]+x[3,2]+x[3,3]<=300.0,

model+=x[1,1]+x[2,1]+x[3,1]>=300.0,
model+=x[1,2]+x[2,2]+x[3,2]>=200.0,
model+=x[1,3]+x[2,3]+x[3,3]>=200.0,
model.solve(GLPK())
print("Status:", LpStatus[model.status])
for v in model.variables():
print(v.name,"=",v.varValue)
print("objective Function", value(model.objective))
```

Şekil 3. 7 Ulaştırma problemi Python Sözde kod Gösterimi

```
runfile('C:/Users/bagul/.spyder-py3/temp.py', wdir='C:/Users/bagul/.spyder-py3')
[1] 3900
```

Şekil 3. 8 Ulaştırma Problemi Python Programı Sonucu

Bu mdelin çözümü için ilk önce model kuruldu ve kısıtlar belirlendi. Vogel yönteminde olduğu gibi minimum toplam maliyetli ulaştırma sağlanmıştır. Ulaştırma problemi Vogel yaklaşımı ile toplam maliyeti 3900 tl bulunmuştur. Yukarıdaki örnekte, 3 talep miktar ve 3 arz miktardan oluşan ulaştırma problemidir. Bu problemi Python programlama dilinde çözmek için pulp kütüphanesi kullanılmıştır. Bu problemi çözmek için ilk önce mdel deklemi ve kısıtlar oluşturulur. Vogel yöntemine benzer şekilde, minimum toplam maliyeti 3900 sağlayacak şekilde ulaştırma bulunmuştur.

3.4.Atama Problemi Örnek Çözümü

Aşağıdaki atama problemini DP problemi olarak modelleyip çözümünü bulalım. Üç müşteriden gelen talepler doğrultusunda istatistiksel araştırmalar yapılacaktır. Talepleri karşılayacak kapasitede 4 farklı araştırma grubumuz vardır. Bu grupların talep edilen araştırmalara atanmaları halinde oluşacak maliyetler (bin tl) aşağıdaki tabloda verildiği gibi öngörülmüştür. Hangi grup hangi araştırmayı yapmak üzere görevlendirilirse toplam maliyet minimum olur.

Tablo 3. 3 Atama Problemi Örneği

	Araştırma		
Grup	A1	A2	A3
G1	150	210	270
G2	170	230	220
G3	180	230	225
G4	160	240	230

Tablo 3.3'den yararlanarak, DP modeli ve kısıtlar aşağıdaki gibi oluşturulmuştur.

DP Modeli: x_{ij}=1 eğer G_i grubu A_i araştırmasına atanırsa; 0 aksi halde olsun

$$min\ Z = 150X_{11} + 210X_{12} + 270X_{13} + 170X_{21} + 230X_{22} + 220X_{23} + 180X_{31} + 230X_{32} + 225X_{33} + 160X_{41} + 240X_{42} + 230X_{43}$$

Kısıtlar

$$X_{11} + X_{12} + X_{13} \leq 1, G1 için$$
$$X_{21} + X_{22} + X_{23} \leq 1, G2 için$$
$$X_{31} + X_{32} + X_{33} \leq 1, G3 için$$
$$X_{41} + X_{42} + X_{43} \leq 1, G4 için$$

$$X_{11} + X_{21} + X_{31} + X_{41} = 1, A1 için$$
$$X_{12} + X_{22} + X_{32} + X_{42} = 1, A2 için$$
$$X_{13} + X_{23} + X_{33} + X_{43} = 1, A3 için$$

Daha sonra Macar algoritması ile toplam maliyet minimum 590 tl ile atama olacaktır. Bu örneğin WinQSB çözümüde 590 tl olarak bulunmuştur. Çözümlerin karşılaştırmasını yapmak amacı ile Python çözümleyelim ve karşılaştırma yapalım.(Erdem. İ , 2013)

3.4.1.Örneğin Python ile çözümü çözümü ve karşılaştırılması

```
import numpy as np
from scipy.optimize import linear_sum_assignment
cost_matrix=np.array([[150, 210, 270],
                      [170, 230, 220],
                      [180, 230, 225],
                      [160, 240, 230]])
row_ind, col_ind = linear_sum_assignment(cost_matrix)
opt_ass = col_ind
tc = cost_matrix[row_ind, col_ind].sum()
print(opt_ass)
print('Total assignment cost is % d'% tc)
```

Şekil 3. 9 Atama Problemi Python Programı Sözde Kod Gösterimi

Yukarıdaki örnekte, 3 kaynak ve 3 hedef arasında bir atama problemidir. "cost_matrix" değişkeni, kaynakların hedeflere atanması için olası maliyetleri içeren bir maliyet matrisidir. "linear_sum_assignment" fonksiyonu ise Kuzey-Batı yöntemine benzer şekilde, minimum toplam maliyeti sağlayacak atamaları bulur. Sonuçlar, atanmış kaynaklar ve hedeflerin indeksleriyle birlikte toplam maliyeti gösterir. Bu örnek, scipy.optimize kütüphanesini kullanarak basit bir atama probleminin çözümünü göstermektedir.

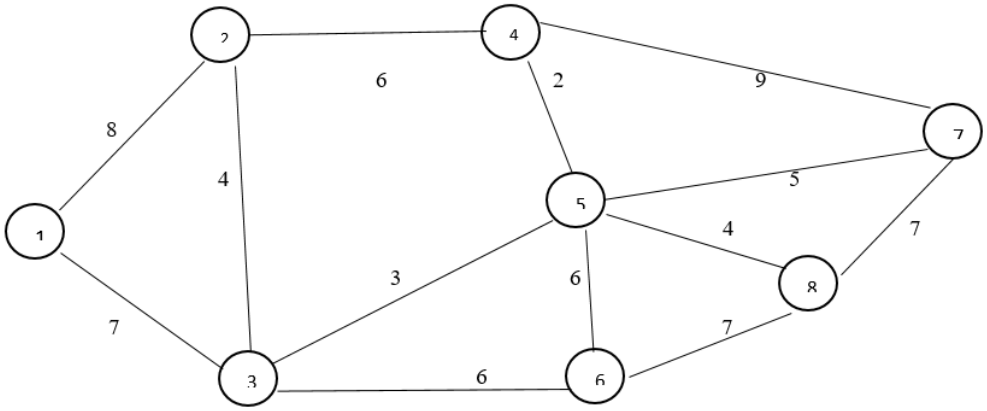
```
runfile('C:/Users/bagul/.spyder-py3/temp.py', wdir='C:/Users/bagul/.spyder-py3')
[1 2 0]
Total assignment cost is  590
```

Şekil 3. 10 Atama Problemi Python Programı Sonucu

Çözüm sonuçları Macar yöntemi ve WinQSB yönteminde oldugu gibi Python sonucu 590 tl bulunmuştur. Genişletilmiş problemler için daha gelişmiş kütüphaneleri veya kayıtlı programlama kütüphanelerini kullanmanız gerekebilir

3.5. Kapsayan Ağaç Problemi Örnek Çözümü

Bir Yerleşim yerindeki 8 tüketim merkezine, boru hattı ile doğal gaz verilecektir. Tüketim merkezleriyle, birbirlerine bağlanabilir olanların uzaklıkları ($10\boldsymbol{m^2}$). Aşağıdaki şebeke diyagramında verilen bilgilere göre, enaz kullanılarak gerçekleştirilmesi gereken bağlantıları bulunuz.(Kara. İ, 2000)



Şekil 3. 11 Kapsayan Ağaç Problemi Örneği

Bu model çözümü için Kapsayan Ağaç problemi kullanılmaktadır. Bağlantı yapılmış düüğümler, yani kapsayan ağaça alınmış düğümler birleştirilerek, yerleşim yerleriyle ilgili yayılma probleminin çözümlenmiştir.Kapsayan ağaçta yer alan ayrıt uzunlukları gözönüne alındığında, tüketim merkezlerini en az kablo kullanarak bir birine bağlamak için 3100 m kablo gerekliği olduğu anlaşılmaktadır. Şimdi ise Python Programlama dilinde çözümlerine bakalım ve karşılaştırma yapalım.

3.5.1.Örneğin Python ile çözümü çözümü ve karşılaştırılması

```
import sys
class Graph:
    def __init__(self, vertices):
        self.V = vertices
        self.graph = [[0 for _ in range(vertices)] for _ in range(vertices)]
    def printMST(self, parent):
        total_weight = 0
        print("Kapsayan Ağaç:")
        for i in range(1, self.V):
            total_weight += self.graph[i][parent[i]]
            print(f"{parent[i]} - {i}\t{self.graph[i][parent[i]]}")
        print("Toplam Ağırlık:", total_weight)
    def minKey(self, key, mstSet):
        min_val = sys.maxsize
        min_index = -1
        for v in range(self.V):
            if key[v] < min_val and not mstSet[v]:
                min_val = key[v]
                min_index = v
        return min_index
    def primMST(self):
        key = [sys.maxsize] * self.V
        parent = [None] * self.V
        key[0] = 0
        mstSet = [False] * self.V
        parent[0] = -1
        for _ in range(self.V):
            u = self.minKey(key, mstSet)
            mstSet[u] = True
            for v in range(self.V):
                if self.graph[u][v] > 0 and not mstSet[v] and key[v] > self.graph[u][v]:
                    key[v] = self.graph[u][v]
                    parent[v] = u
        self.printMST(parent)
# Örnek kullanım
g = Graph(8)
g.graph = [
    [0, 8, 7, 0, 0, 0, 0, 0],
    [8, 0, 4, 6, 0, 0, 0, 0],
    [7, 4, 0, 0, 3, 6, 0, 0],
    [0, 6, 0, 0, 2, 0, 9, 0],
    [0, 0, 3, 2, 0, 6, 5, 4],
    [0, 0, 6, 0, 6, 0, 0, 7],
    [0, 0, 0, 9, 5, 0, 0, 7],
    [0, 0, 0, 0, 4, 7, 7, 0]
]
g.primMST()
```

Şekil 3. 12 Kapsayan Ağaç Problemi Python Sözde Kod Gösterimi

```
Kapsayan Ağaç:
2 - 1        4
0 - 2        7
4 - 3        2
2 - 4        3
2 - 5        6
4 - 6        5
4 - 7        4
Toplam Ağırlık: 31
```

Şekil 3. 13 Kapsayan Ağaç Problemi Python sonucu

Bu örnekte,. primMST metodu, Prim algoritmasını kullanarak Kapsayan Ağaç Problemi'ni çözer ve sonucu ekrana yazdırır. Kod örneğinde, 8x8 boyutunda bir matris (g.graph) kullanılarak grafın ağırlıklı matrisi oluşturulmuştur. Bu matris, literatürdeki çalışmaların sağladığı verileri temsil etmektedir. Kapsayan ağaç problemini çözmek için g.primMST() çağrısı yapılır ve sonuç ekrana yazdırılır. Python programlama dilinde hem sonuç 31 ($\boldsymbol{m^2}$) kablo gerekliği anlaşılmaktadır. Bu şekilde, literatürdeki çalışmaların sağladığı verilerle Kapsayan Ağaç Problemi'ni çözebilir ve çözümü diğer çalışmalarla karşılaştırabilirsiniz.

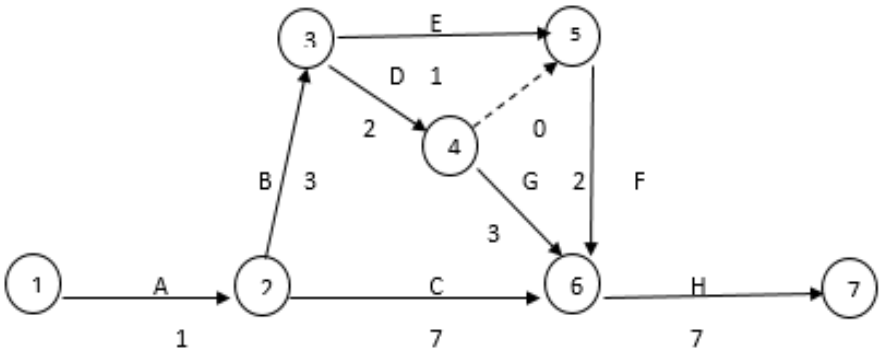
3.6. CPM modeli Örnek Çözümü

Yeni bir ürün üretmeyi amaçlayan bir şirket aşağıdaki verilen bilgilere dayanarak yeni ürünü için proje programlaması yapmak istemektedir. Bu bilgilere göre Proje süresini ve kritik yolu hesaplayınız.(Öztürk. A, 2016)

Tablo 3. 4 CPM model Örneği

Faaliyet	Faaliyetin Özü	Süre	Faaliyetlerin Önceki Sırası
A	Üretim için gerekli malzeme maliyetlerinin belirlenmesi	1	_
B	Üretim akış tablosunun hazırlanması	3	A
C	Malzemelerin siparişşi ve gönderilen malzemelerin beklenmesi	7	A
	Üretim hattını örgütleme		
D	Kontrol işlemlerinin belirleme	2	B
E	Kontrol hattını yerleştirme	1	B
F	İşçileri eğitme	2	D,E
G	Ürünleri biraraya toplama	3	D,E
H		7	C,F,G

CPM probleminin ağ diyagramı aşağıdaki gibidir.



Şekil 3. 14 CPM Problemi Ağ Gösterimi

Tablodaki bilgilerden yararlanarak, ağ diyagramı oluşturulur. Sonra CPM metoduna göre hesaplamalar yapılır. Hesaplamalar sonucu örneğin kritik değeri 16 bulunmuştur. Yani 16 günü geçmeden o faliyetin ne kadar geç başlayabileceğini gösterir.

3.6.1.Örneğin Python ile çözümü çözümü ve karşılaştırılması

```
def pert_analysis(duration, predecessors):
    num_tasks = len(duration)
    earliest_start = [0] * num_tasks
    earliest_finish = [0] * num_tasks
    for i in range(num_tasks):
        if not predecessors[i]:
            earliest_start[i] = 0
        else:
            earliest_start[i] = max(earliest_finish[pred] for pred in predecessors[i])
        earliest_finish[i] = earliest_start[i] + duration[i]
    latest_start = earliest_start.copy()
    latest_finish = earliest_finish.copy()
    slack_time = [latest_start[i] - earliest_start[i] for i in range(num_tasks)]
    critical_path = [i for i, slack in enumerate(slack_time) if slack == 0]
    project_duration = max(earliest_finish)
    return earliest_start, earliest_finish, latest_start, latest_finish, slack_time,
critical_path, project_duration
# Örnek kullanım
duration = [1, 3, 7, 0, 2, 1, 2, 3, 7]
predecessors = [[], [1], [1], [0], [2], [2], [5,6], [5], [3,7,8]]
earliest_start, earliest_finish, latest_start, latest_finish, slack_time, critical_path,
project_duration = pert_analysis(duration, predecessors)
# Sonuçları yazdırma
print("En Erken Başlama Zamanları:", earliest_start)
print("En Erken Bitiş Zamanları:", earliest_finish)
print("En Geç Başlama Zamanları:", latest_start)
print("En Geç Bitiş Zamanları:", latest_finish)
print("Slack Süreleri:", slack_time)
print("Kritik Yol:", critical_path)
print("Proje Süresi:", project_duration
```

Şekil 3. 15 CPM Problemi Python Sözde Kod Gösterimi

En Erken Başlama Zamanları: [0, 0, 3, 1, 10, 10, 11, 11, 14]
En Erken Bitiş Zamanları: [1, 3, 10, 1, 12, 11, 13, 14, 21]
En Geç Başlama Zamanları: [0, 0, 3, 1, 10, 10, 11, 11, 14]
En Geç Bitiş Zamanları: [1, 3, 10, 1, 12, 11, 13, 14, 21]
Slack Süreleri: [0, 0, 0, 0, 0, 0, 0, 0, 0]
Kritik Yol: [0, 1, 2, 3, 4, 5, 6, 7, 8]
Proje Süresi: 21

Şekil 3. 16 CPM problemi Python Sonucu

İlk olarak, pert_analysis adında bir fonksiyon tanımlanır. Bu fonksiyon, projenin PERT analizini yapacak ve gerekli hesaplamaları gerçekleştirecektir. Fonksiyon iki parametre: duration (süreler) ve predecessors (öncüller) listeleri alır. Bu kod, her görevin en erken başlama ve bitiş zamanlarını, en geç başlama ve bitiş zamanlarını, Slack zamanlarını, kritik yolu ve projenin süresini hesaplar. Bu bilgiler, projenin planlanması ve takibi için önemli olabilir. Hesaplama sonucu Proje süresi 21 gün olarak bulunmuştur.

3.7. PERT Modeli Örnek Çözümü

Dört olay ve beş faaliyetli bir proje için PERT analizinin faaliyetler için üç tahmin süresi aşağıdaki tabloda verilmiştir.

Tablo 3. 5 PERT Problemi Örneği

Faaliyetler	En erken tamamlama	Mühtemel süre	En geç tamamlama
1-2	3	4	6
1-3	3	6	8
2-3	1	3	5
2-4	6	8	10
3-4	3	7	9

Yukarıdaki bilgilerden yararlanarak, projenin en az 15 gunde bitirilmesi olasılığını hesaplayalım.

Önce faaliyetlerin beklenen zamanı ve varyanslarını bulunur. Daha sonra formülden yararlanarak ortalama , varyans ve standart sapma değerleri aşağıdaki gibi hesaplandı.

$$\mu = \sum \bar{x} = 13{,}84$$

$$\sigma_{t_e}{}^2 = \sum \sigma_{t_e}{}^2 = 1{,}69$$

$$\sigma = 1{,}3$$

$$Z = \frac{Y-\mu}{\sigma_{t_e}} = \frac{15-13,84}{1,3} = 0,89$$

Normal Dağılım Tablosundaki 0,89 değeri 0,8133 e eşittir. Projenin 15 günde bitirilme olasılığı %81,3 olarak bulunmuştur.

3.7.1.Örneğin Python ile çözümü çözümü ve karşılaştırılması

```
import numpy as np

def pert_analysis(duration, optimistic, pessimistic):
    num_tasks = len(duration)
    earliest_start = [0] * num_tasks
    earliest_finish = [0] * num_tasks
    latest_start = [0] * num_tasks
    latest_finish = [0] * num_tasks

    # Hesaplama işlemleri
    for i in range(num_tasks):
        earliest_start[i] = (optimistic[i] + 4 * duration[i] + pessimistic[i]) / 6
        earliest_finish[i] = earliest_start[i] + duration[i]

    latest_finish[num_tasks - 1] = earliest_finish[num_tasks - 1]
    latest_start[num_tasks - 1] = earliest_start[num_tasks - 1]

    for i in range(num_tasks - 2, -1, -1):
        latest_finish[i] = latest_start[i + 1]
        latest_start[i] = latest_finish[i] - duration[i]

    slack_time = [latest_start[i] - earliest_start[i] for i in range(num_tasks)]
    critical_path = [i for i, slack in enumerate(slack_time) if slack == 0]
    project_duration = max(earliest_finish)

    # Projenin ortalama ve varyansını hesaplama
    mean = sum(earliest_finish) / num_tasks
    variance = sum([(duration[i] - mean) ** 2 for i in range(num_tasks)]) / num_tasks

    # Projenin en az 15 günde tamamlanma olasılığını hesaplama
    probability = 1 - (variance / project_duration ** 2)

      # Projenin en az 15 günde tamamlanma olasılığını hesaplama
    probability = 1 - (variance / project_duration ** 2)

    return probability

# Örnek kullanım
duration = [4, 6, 3, 8, 7]
optimistic = [3, 3, 1, 6, 3]
pessimistic = [6, 8, 5, 10, 9]

probability = pert_analysis(duration, optimistic, pessimistic)

# Sonucu yazdırma
print("En az 15 günde tamamlanma olasılığı:", probability)
```

Şekil:3.14 PERT Problemi Python Sözde Kod Gösterimi

```
En az 15 günde tamamlanma olasılığı: 0.8669618055555555
```

Şekil 3.15 PERT Problemi Python Sonucu

Yukarıdaki kodda, duration, optimistic ve pessimistic listelerine örnek veriler atanır. Ardından, pert_analysis fonksiyonu bu verilerle çağrılır ve dönen değerler ilgili değişkenlere atanır. Sonrasında, hesaplanan değerler ekrana yazdırılır. Bu sayede, Pert analizi kullanılarak projenin erken başlama, erken bitiş, geç başlama, geç bitiş, boş zaman, kritik yol ve proje süresi gibi önemli değerlerini hesaplayabilir ve ekrana yazdırabilirsiniz. Çözüm sonucunda 15 günde projenin bitirilmesi olasılığı 0,86 bulunmuştur.

Python'un bu avantajları, projeleri hızlı ve verimli bir şekilde geliştirmenize yardımcı olur. Ayrıca, Python'un geniş kütüphane desteği, çözüm sağlamak için kullanılabilecek birçok önceden yazılmış işlev ve aracı içerir. Bu, projelerinizi daha verimli bir şekilde yönetmenizi sağlar ve tekrarlayan işleri otomatikleştirmenize olanak tanır.

Örneklerin Python çözümleri, Python'un bu avantajlarını kullanarak belirli bir probleme yönelik pratik ve etkili bir çözüm sunar. Ayrıca, Python örnekleri, dilin yapısını ve işlevselliğini daha iyi anlamanıza yardımcı olur ve daha karmaşık projelerde bu bilgileri uygulamaya geçirebilmenizi sağlar.

4.SONUÇ VE ÖNERİLER

Çalışmanın “Literatür Araştırması” bölümünde literatürde yer alan ilgili çalışmalar özetlenmiştir. Sonraki bölüm olan “Materyal ve Metot” bölümünde tez kapsamında önerilen yaklaşımlar, kullanılan model ve algoritmalar açıklanmakta, takip eden “Uygulama ve Bulgular” bölümünde ise önerilen yaklaşımların uygulanmasıyla elde edilen sonuçlar sunulmuştur. Son kısımda yer alan “Sonuç ve Öneriler” bölümü tez çalışmasının genel bir değerlendirmesini sunmaktadır. Bu çalışmada, literatürde mevcut olan günlük hayattaki problemlerden ağ yapısına uygun olan problemler ele alınıp çözümleri yapılmış ve Python ile karşılaştırılması yorumlanmıştır. Çözümlemeler özellikle PYTHON dilinde yapılmaya çalışılarak literatüre katkı da sağlanmıştır.

Python programının geniş kullanıcı tabanı ve aktif topluluğu, ağ optimizasyon problemleri için çözümler bulmanızı ve sorunlarınıza yönelik özelleştirilmiş çözümler geliştirmenizi kolaylaştırır.

Bu çalışma, ağ optimizasyon modellerinin pratik uygulamalarını göstermek ve Python programlama dilinde çözüm süreçlerini adım adım göstermek için bir rehber olacak. Örnekler ve karşılaştırmalar, ağ optimizasyonun önemini vurgulamak ve farklı çözüm yöntemlerinin etkinliğini anlamak için kullanılacaktır. Bu şekilde, okuyucuların ağ optimizasyon modellerine olan anlayışını derinleştirmek ve bu modelleri kendi projelerinde uygulamak için gerekli bilgi ve becerileri kazanmalarını sağlamak hedeflenmektedir.

Ağ optimizasyon modelleri, karmaşık problemleri çözmek ve iş süreçlerinde önemli iyileştirmeler sağlamak için güçlü bir araçtır. Python programlama dilinin esnekliği ve zengin kütüphane ekosistemi, bu modellerin uygulanmasını kolaylaştırır. Projenizin gereksinimlerine göre uygun optimizasyon yöntemlerini kullanarak, en iyi sonuçları elde edebilir ve işlemlerinizin verimliliğini artırabilirsiniz.

KAYNAKLAR

Akay, M. & Ayşegül.Tuş, A.,(2022) *Minimum Yayılan Ağaç (MYA) Problemi: Denizli İli Hafif Raylı Sistem Proje Önerisi İçin Minimum Mesafeli Hat Belirleme*. Pamukkale Üniversitesi, Sosyal Bilimler Enstitüsü. Araştırma makalesi. Denizli

Akşehir, K.(2019) "*Gezgin Satıcı Probleminin Karınca Kolonisi Algoritması İle Çözüm Performansının Arttırılmasında Parametre Optimizasyonu*". On dokuz Mayıs Üniversitesi, Fen Bilimleri Enstitüsü. Yüksek lisans. Samsun

Altaylı, B. (1996) *Yönetim kararlarında kantitatif yöntemler yöneylem araştırması*. Uytes,66-68,Ankara

Altunkaynak, B & Bakır. M.A. (2003) *Tamsayılı Programlama*. Nobel Yayın Dağıtım, Ankara

Analı, İ.,(1999) *Ulaştirma Modeli Ve Türk Tekstil Sektörindeki Diş Ticaret Sermaye Şirketlerinin Îhracatlarının Ulaştırma Modeli Yardamıyla Optimizasyonu*. Marmara Universitesi,Sosyal Bilimler Enstitüsü.Yüksek lisans. İstanbul

Arman, K., & Tuş, A.,(2022) "*Bir Lojistik Firmasının En Kısa Yol Problemine Düğüm Kombinasyonu Algoritmasının Uygulanması*" İstanbul Üniversitesi. Journal of Transportation and Logistics. Araştırma .İstanbul

Aydemir, E (2021) *Uygulamalar ile Python ve Yapay Zeka*. Nobel Yayın Dağıtım, Ankara

Bakışkan, E., (2019) *CPM-PERT Proje Yönetim Tekniklerinin Karadeniz Tipi Balıkçı Gemileri İnşa Sürecine Uygulanması* Karadeniz Teknik Üniversitesi, Fen Bilimleri Enstitüsü. Yüksek lisans. Trabzon

Berberler, M. E., Onur.U & Gözde.K,(2012) *Macar Algoritmasının Sıfırları Kapatma Alt Yordamı Üzerine*. Dokuz Eylül Üniversitesi, Fen Fakültesi, Bilgisayar Bilimleri Bölümü. Araştırma makalesi. İzmir

Bülbül, E.(2022) "*Değişen 3b Ortamda En Kısa Yol Algoritmaları*". Ankara Üniversitesi, Fen Bilimleri Enstitüsü. Yüksek Lisans Tezi. Ankara

Cingöz, K., & Gürgen, E., (2023) *Hızlı Yiyecek Sektöründe Üretim Sürecinin Planlanması: Pert Yöntemi Uygulaması*. Tarsus Üniversitesi Uygulamalı Bilimler Fakültesi Dergisi, Yıl: 2023, Cilt: 3, Sayı:1, ss. 36-46

Demirkol, Ö.E., & Aşkın Demirkol, A.,(2003) *Dıjkstra Ve Bellman-Ford En Kısa Yol Algoritmalarının Karşılaştırılması* Sakarya Üniversitesi, Fen Bilimleri Enstitüsü. Araştırma makalesi. Sakarya

Dener, M., Akcayol, M.A., Sinan Toklu, S., & Ömer Faruk Bay, Ö.F.,(2011) *Zamana Bağlı Dinamik En Kısa Yol Problemi İçin Genetik Algoritma Tabanlı Yeni Bir Algoritma* Gazi Üniversitesi, Mühendislik Fakültesi, Araştırma makalesi. Ankara

Dhulkefl. E. J., Durdu.A., &Terzioğlu.H (2020) *Dıjkstra Algorıthm Usıng Uav Path Plannıng*. Konya Teknik Üniversitesi, Mühendislik Bilimleri Dergisi. Araştırma Makalesi. Konya

Duran, C. (2007) *Cpm - Pert Modelleri Ve Uygulaması*. Marmara Üniversitesi, Sosyal Bilimler Enstitüsü. Yüksek lisans. İstanbul

Erdem, İ., (2013). *Yöneylem Araştırması ve WinQSB Uygulamaları*, 1.baaskı, Seçkin Kitabevi, Ankara

Gürsoy, H., & Duman, E.,(2022) "*İki Amaçlı En Kısa Yol Problemi Ve Bir Uygulaması*" Ankara

İstanbullu, H., (2009) *Maksimum Akış Problemi, Görüntü Taşınımı, Saklanması Ve Çağrımı İçin Bir Algoritma*. İstanbul Üniversitesi, Sosyal Bilimler Enstitüsü. Doktara. İstanbul

Karadeniz, C.Ö.,(2007) *PERT-CPM İle Proje Planlama, Değerlendirme Ve Bir İşletme Uygulaması*.Marmara Üniversitesi Sosyal Bilimler Enstitüsü. Yüksek lisans. İstanbul

Karslı, N.,(2010) *Akıllı Ulaşım Sistemleri İçin Yapay Bağışıklık Sistemleri Ve Genetik Algoritma İle Yeni Stokastik En Kısa Yol Algoritmalarının Geliştirilmesi* Atatürk Üniversitesi, Fen Bilimleri Enstitüsü. Doktora. Erzurum

Mazlum, M., (2014) *CPM-PERT Ve Bulanık Mantık Teknikleriyle Proje Yönetimi Ve Bir İşletmede Uygulanması*. Yıldız Teknik Üniversitesi, Fen Bilimleri Enstitüsü. Yüksek lisans. İstanbul

Mercangöz, B. A , *Ağ Modelleri Ve Tedarik Zincirinde Ağ Optimizasyonuna İlişkin Bir Uygulama*. İstanbul Üniversitesi, Sosyal Bilimler Enstitüsü. Doktara. İstanbul 2010

Monks Joseph G. *Schaum's Outline Of Theory And Problems Of Operation Management*; McGraw – Hill Inc, (1996)

Özkan, Ş., (2012). Yöneylem Araştırması nicel karar teknikleri, 3.baskı, Nobel Yayn Evi, Ankara

Öztürk, A. 2016. *Yöneylem Araştırması*. Ekin Basım Yayın Evi, Bursa.

Sarıca, İ. CPM *Ve PERT Teknikleriyle Proje Planlama Ve Bir İşletmede Uygulanması*. Uludağ Üniversitesi, Sosyal Bilimler Enstitüsü. Yüksek lisans. Bursa 2006

Stefan. H., (2010) "*Negatif Döngülere Sahip Grafiklerde Floyd-Warshall Algoritması*". University of Bonn, Research Institute for Discrete Mathematics. Germany

Sunny Baker, *Complete Idiot's Guide to Project Management*.Indianapolis, IN, USA: Alpha Books, 2000. s 15.

Şahin.B Meriç.Ç Alper.U *Araç Rotalama Probleminde Araç Rotalarının Tespitinde En Kısa Yol Yaklaşımı: Denizli Örneği*. Pamukkale Üniversitesi. Bildiri. Denizli.2020

Taha, H.A., (2010). Yöneylem Araştırması, 6.baskı, Literatür Yayınevi, İstanbul Doç.Dr. Metin Türkay. *Optimizasyon Modelleri Ve Çözüm Metodları*. Koç Üniversitesi, Endüstri Mühendisliği Bölümü, Rumelifeneri Yolu, Sarıyer 34450. İstanbul

Thierauf; R.C.Klekamp; *Decision Making Through Operation Research*; 2nd Edition Wisley Series; s.122 (1975)

Timor, M., (2020) *Yöneylem Araştırması*, 2.baskı, Türkmen Kitabevi, İstanbul

Taşkıran, M, (2022) *Ders Çizelgeleme-Atama Problemi Optimizasyonu*. Burdur Mehmet Akif Ersoy Üniversitesi, Sosyal Bilimler Enstitüsü. Yüksek Lisans. Burdur.

Ulu, F., (2023) *Karınca Kolonisi Optimizasyonu Algoritması İle Depo Rota Planlaması*. Bursa Uludağ Üniversitesi, Fen Bilimleri Enstitüsü. Yüksek lisans. Bursa

Yetgin, M.,(2011) *Türkiye' De Çok Modlu Taşımacılıkta En Kısa Yolların Belirlenmesi* Gazi Üniversitesi, Fen Bilimleri Enstitüsü. Yüksek Lisans. Ankara

EKLER

EK 1. Standart Z Normal Dağılım Tablosu

z	0.00	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09
0.0	0.0000	0.0040	0.0080	0.0120	0.0160	0.0199	0.0239	0.0279	0.0319	0.0359
0.1	0.0398	0.0438	0.0478	0.0517	0.0557	0.0596	0.0636	0.0675	0.0714	0.0753
0.2	0.0793	0.0832	0.0871	0.0910	0.0948	0.0987	0.1026	0.1064	0.1103	0.1141
0.3	0.1179	0.1217	0.1255	0.1293	0.1331	0.1368	0.1406	0.1443	0.1480	0.1517
0.4	0.1554	0.1591	0.1628	0.1664	0.1700	0.1736	0.1772	0.1808	0.1844	0.1879
0.5	0.1915	0.1950	0.1985	0.2019	0.2054	0.2088	0.2123	0.2157	0.2190	0.2224
0.6	0.2257	0.2291	0.2324	0.2357	0.2389	0.2422	0.2454	0.2486	0.2517	0.2549
0.7	0.2580	0.2611	0.2642	0.2673	0.2704	0.2734	0.2764	0.2794	0.2823	0.2852
0.8	0.2881	0.2910	0.2939	0.2967	0.2995	0.3023	0.3051	0.3078	0.3106	0.3133
0.9	0.3159	0.3186	0.3212	0.3238	0.3264	0.3289	0.3315	0.3340	0.3365	0.3389
1.0	0.3413	0.3438	0.3461	0.3485	0.3508	0.3531	0.3554	0.3577	0.3599	0.3621
1.1	0.3643	0.3665	0.3686	0.3708	0.3729	0.3749	0.3770	0.3790	0.3810	0.3830
1.2	0.3849	0.3869	0.3888	0.3907	0.3925	0.3944	0.3962	0.3980	0.3997	0.4015
1.3	0.4032	0.4049	0.4066	0.4082	0.4099	0.4115	0.4131	0.4147	0.4162	0.4177
1.4	0.4192	0.4207	0.4222	0.4236	0.4251	0.4265	0.4279	0.4292	0.4306	0.4319
1.5	0.4332	0.4345	0.4357	0.4370	0.4382	0.4394	0.4406	0.4418	0.4429	0.4441
1.6	0.4452	0.4463	0.4474	0.4484	0.4495	0.4505	0.4515	0.4525	0.4535	0.4545
1.7	0.4554	0.4564	0.4573	0.4582	0.4591	0.4599	0.4608	0.4616	0.4625	0.4633
1.8	0.4641	0.4649	0.4656	0.4664	0.4671	0.4678	0.4686	0.4693	0.4699	0.4706
1.9	0.4713	0.4719	0.4726	0.4732	0.4738	0.4744	0.4750	0.4756	0.4761	0.4767
2.0	0.4772	0.4778	0.4783	0.4788	0.4793	0.4798	0.4803	0.4808	0.4812	0.4817
2.1	0.4821	0.4826	0.4830	0.4834	0.4838	0.4842	0.4846	0.4850	0.4854	0.4857
2.2	0.4861	0.4864	0.4868	0.4871	0.4875	0.4878	0.4881	0.4884	0.4887	0.4890
2.3	0.4893	0.4896	0.4898	0.4901	0.4904	0.4906	0.4909	0.4911	0.4913	0.4916
2.4	0.4918	0.4920	0.4922	0.4925	0.4927	0.4929	0.4931	0.4932	0.4934	0.4936
2.5	0.4938	0.4940	0.4941	0.4943	0.4945	0.4946	0.4948	0.4949	0.4951	0.4952
2.6	0.4953	0.4955	0.4956	0.4957	0.4959	0.4960	0.4961	0.4962	0.4963	0.4964
2.7	0.4965	0.4966	0.4967	0.4968	0.4969	0.4970	0.4971	0.4972	0.4973	0.4974
2.8	0.4974	0.4975	0.4976	0.4977	0.4977	0.4978	0.4979	0.4979	0.4980	0.4981
2.9	0.4981	0.4982	0.4982	0.4983	0.4984	0.4984	0.4985	0.4985	0.4986	0.4986
3.0	0.4987	0.4987	0.4987	0.4988	0.4988	0.4989	0.4989	0.4989	0.4990	0.4990

ÖZ GEÇMİŞ

Bagul Ovlyakulyyeva, Aşkabat şehri Azatlık ilçesi fen bilimleri ve İngilizce dersleri ihtisas okulu Lisesi'ni bitirdikten sonra, Ondokuz Mayıs Üniversitesi Fen Edebiyat Fakültesi, İstatistik bölümünden 22.06.2020 tarihinde mezun oldu. 2020 yılında OMÜ LEE İstatistik Ana Bilim Dalı Yüksek Lisans programına girdi. Bagul Ovlyakulyyeva, iyi derecede Türkmence, İngilizce ve Rusça bilmektedir. Temel ilgi alanları, Python ve R programlama.

İletişim Bilgileri

ORCID ID : 0000-0002-2122-5693

Yayınlar:

1. Ovlyakulyyeva, B. & Şenel, T " SOLUTION OF ASSIGNMENT AND MAXIMUM FLOW PROBLEMS FROM NETWORK OPTIMIZATION MODELS WITH PYTHON" 4TH İNTERNATİONAL "ACHARAKA" CONGRESS ON LİFE, ENGINEERING, AND APPLIED SCIENCES LETTER OF ACCEPTANCE Türkiye, İzmir, 15-18 July. 2023, P.104-110

2. Ovlyakulyyeva, B. & Şenel, T " NETWORK OPTIMIZATION MODELS" 2ND İNTERNATİONAL "ARTEMİS" CONGRESS ON LIFE, SOCIAL, HEALTH, AND SPORTS SCIENCES LETTER OF ACCEPTANCE Türkiye, İzmir, 17-19 December. 2022, P.146