OLUMSUZ HAVA KOŞULLARINI DİKKATE ALAN UÇUŞ ÇİZELGELEME PROBLEMİ İÇİN METASEZGİSEL YAKLAŞIMLAR

Ebru ERDEM

Yüksek Lisans Tezi
Bilgisayar Mühendisliği Anabilim Dalı
Bilgisayar Mühendisliği Bilim Dalı
Dr. Öğr. Üyesi Tolga AYDIN
2019
Her hakkı saklıdır

ATATÜRK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

OLUMSUZ HAVA KOŞULLARINI DİKKATE ALAN UÇUŞ ÇİZELGELEME PROBLEMİ İÇİN METASEZGİSEL YAKLAŞIMLAR

Ebru ERDEM

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
Bilgisayar Mühendisliği Bilim Dalı

ERZURUM
2019

Her hakkı saklıdır

T.C.
ATATÜRK ÜNİVERSİTESİ
Fen Bilimleri Enstitüsü Müdürlüğü

TEZ ONAY FORMU

OLUMSUZ HAVA KOŞULLARINI DİKKATE ALAN UÇUŞ ÇİZELGELEME PROBLEMİ İÇİN METASEZGİZEL YAKLAŞIMLAR

Dr. Öğr. Üyesi Tolga AYDIN danışmanlığında, Doç. Dr. Burak ERKAYMAN ortak danışmanlığında, Ebru ERDEM tarafından hazırlanan bu çalışma, 17/06/2019 tarihinde aşağıdaki jüri tarafından Bilgisayar Mühendisliği Anabilim Dalı Bilgisayar Mühendisliği Bilim Dalı'nda yüksek lisans tezi olarak **oybirliği** / ~~**oy çokluğu**~~ **(.../...)** ile kabul edilmiştir.

Başkan: Dr. Öğr. Üyesi Tolga AYDIN İmza :

Üye : Doç. Dr. Burak ERKAYMAN İmza : b.erkayman

Üye : Dr. Öğr. Üyesi Deniz DAL İmza :

Üye : Dr. Öğr. Üyesi Saffet Gökçen Şen İmza : Gökçen

Üye : Dr. Öğr. Üyesi Ferhat BOZKURT İmza :

Yukarıdaki sonuç;

Enstitü Yönetim Kurulu'nun 18/07/2019 tarih ve ...29.../..63..... nolu kararı ile onaylanmıştır.

Prof. Dr. Mehmet KARAKAN
Enstitü Müdürü

Not: Bu tezde kullanılan özgün ve başka kaynaklardan yapılan bildiriş, çizelge, şekil ve fotoğrafların kaynak olarak kullanımı, 5846 sayılı Fikir ve Sanat Eserleri Kanunundaki hükümlere tabidir.

ÖZET

Yüksek Lisans Tezi

OLUMSUZ HAVA KOŞULLARINI DİKKATE ALAN UÇUŞ ÇİZELGELEME PROBLEMİ İÇİN METASEZGİSEL YAKLAŞIMLAR

Ebru ERDEM

Atatürk Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı
Bilgisayar Mühendisliği Bilim Dalı

Danışman: Dr. Öğr. Üyesi Tolga AYDIN

Nüfusun artması, teknolojinin ilerlemesi, çalışma hayatının yoğunlaşması insanları zamanı verimli kullanmaya yönlendirmiştir. İnsanların zamanı verimli kullanma açısından faydalandığı en önemli ulaşım türü havayolu ulaşımıdır. Bu yüzden havayolu ulaşımına olan talep her geçen gün artmaktadır. Dolayısıyla havayolu şirketleri için yolcuların istek ve beklentilerini karşılamak ve havaalanında verilen hizmet kalitesini artırmak son derece önemlidir. Havayolu şirketleri bir yandan yolcuların isteklerini ve ihtiyaçlarını en ucuz ve hızlı yolla karşılamalı, bir yandan da zorlu rekabet şartlarında ayakta kalabilmek için verdiği hizmetlerin maliyetini düşürmeli ve etkinliğini artırmalıdır.

Havayolu trafiği performansını belirleyen en önemli faktörlerden biri, şirketlerin oluşturdukları tarifelerdir. Tarifeler yapılırken daha çok belirlenen güzergâha olan talep, hava araçlarının kullanılabilirliği, uçuş ekiplerinin elde edilebilirliği gibi parametreler kullanılmaktadır. Gidilecek güzergâhın, bilinen iklim şartları genellikle ihmal edilmektedir.

Bu tez çalışmasında; uçuş tarifelerinin yeniden hazırlanması işlemi olan uçuş çizelgeleme problemi çözülürken, hava koşulları dikkate alınarak uçuş gecikmesinin olabileceği ay, gün, saat saptanmıştır. Saptama işleminde, 1987-2018 tarihleri arasındaki 32 yıllık uçuş verileri temel alınmıştır. Bu kısıtların tespit edilebilmesi için, büyük veri teknolojilerinden olan Apache Spark kullanılmıştır. Elde edilen kısıtlar ile tarife optimize edilip, uçak sefer iptal ve gecikmelerinin olabildiğince önüne geçilmeye çalışılmıştır. Saptanan şans kısıtları optimizasyon gösterimine dahil edilmiştir. Yapılan deneyler sonucunda tarifenin optimize edilmesinde kullanılan metasezgisel algoritmalardan (üstsezgisel algoritmalar) benzetilmiş tavlamanın, genetik algoritma ve yapay arı koloni algoritmasına göre en optimum sonucu verdiği gözlemlenmiştir. Taranılan uzayın boyutu arttıkça, metasezgisel yaklaşımlar ile problemi çözmenin, klasik yöntem ile çözmeye göre zaman açısından sağladığı avantaj net olarak görülmüştür.

2019, 116 sayfa

Anahtar Kelimeler: Uçuş çizelgeleme problemi, benzetilmiş tavlama, genetik algoritma, yapay arı koloni algoritması

ABSTRACT

MS Thesis

METAHEURISTIC APPROACHES TO FLIGHT SCHEDULING PROBLEM CONSIDERING NEGATIVE WEATHER CONDITIONS

Ebru ERDEM

Atatürk University
Graduate School of Natural and Applied Sciences
Department of Computer Engineering
Computer Engineering

Supervisor: Assist. Prof. Dr. Tolga AYDIN

Population increase, advances in the technology, heavy workload of human beings have led people to use time more efficently. Transportation by air is the most important transport type that helps people to use their times efficiently. Therefore, demand for airline transportation is increasing day by day. It is extremely important for airline companies to meet the wishes and expectations of passengers and to improve the quality of service provided at the airports. Airline companies need to meet the demands and needs of the passengers at the cheapest and fastest way, on the other hand they should reduce the cost of their services and increase their efficiency in order to survive in challenging competition conditions.

One of the most important factors determining the performance of air traffic is the schedules the companies form. While arranging the schedules, the parameters such as the demand for the designated route, the availability of aircrafts and the availability of flight crews are used. The known climatic conditions of the route to be traveled are often neglected.

In this thesis; while solving the flight scheduling problem, weather conditions are taken into account and possible month, day and hours for possible delays are determined. This process is based on 32-year flight data collected between 1987-2018. In order to determine these constraints, Apache Spark which is one of the big data technologies is used. With the constraints obtained, the schedule has been optimized and the flight cancellations and delays have been avoided as much as possible. The chance constraints identified are included in the optimization demonstration. As a result of the experiments, it has been observed that among the metaheuristic algorithms, simulated annealing gives the optimum results according to genetic algorithm and artificial bee colony algorithm. As the size of the scanned space increases, it is seen that the advantage of solving the problem with metaheuristic approaches compared to classical method is clear, in terms of time considerations.

2019, 116 pages

Keywords: Flight scheduling problem, simulated annealing, genetic algorithm, artificial bee colony algorithm

TEŞEKKÜR

Tezimin konusunun belirlenmesinde, araştırma aşamasında, yönlendirilmesinde ve tamamlanmasında destek olan değerli hocalarım, tez danışmanım ve tez ortak danışmanım, Sayın Dr. Öğr. Üyesi Tolga AYDIN ve Sayın Doç. Dr. Burak ERKAYMAN'a bana ayırdıkları değerli zamanları ve sağladıkları destek için minnettarım.

Tezimin başlangıcından bitimine kadar bana inanan, benden yardımlarını esirgemeyen, her zaman yanımda olan, bildiklerini paylaşan ve bildiklerimi paylaşmamı öğreten annem, babam ve kardeşlerime,

Son olarak da gösterdikleri sabır ve verdikleri her türlü destek için arkadaşlarım Seda Erden Dertli'ye, Tuba Şahin'e, Nursen Keleş'e teşekkür ederim.

Ebru ERDEM
Temmuz, 2019

İÇİNDEKİLER

SİMGELER ve KISALTMALAR DİZİNİ

Kısaltmalar

A.N.	Ay No
ACO	Karınca Koloni Algoritması
AFSA	Yapay Balık Sürü Algoritması
API	Uygulama Programlama Arayüzü
BT	Benzetilmiş Tavlama
BTS	Ulaştırma İstatistikleri Bürosu
DAG	Yönlü Çevrimsiz Ağaç (Graf-Çizge)
E.K.D.	En Kötü Değer
FCFS	İlk Gelen İlk Hizmeti Alır Algoritması
G.N.	Gün No
GA	Genetik Algoritma
H.D.K.G.S.	Hava Durumundan Kaynaklı Gecikme Süresi
HDFS	Hadoop Dağıtılmış Dosya Sistemi
IMDB	Bellek İçi Veri Tabanları Sistemi
İ.S.	İterasyon Sayısı
JVM	Java Sanal Makinesi
K.Z.	Kalkış Zamanı
L.	Limit
MİB	Merkezi İşlem Birimi
P.B.	Popülasyon Büyüklüğü
PiCloud	Basitleştirilmiş Bulut Bilişim
pySpark	Spark Python Uygulama Programlama Arayüzü
RDD	Esnek Dağıtık Veri Kümeleri
RMB	Çin Yuanı
t.v.s	Tarife Veri Seti
V.G.S.	Varış Gecikme Süresi
YAKA	Yapay Arı Koloni Algoritması

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

1. GİRİŞ

Ülkeler arasındaki ilişkiler geliştikçe, hava yolu ulaşımına olan ihtiyaç artmaktadır. Havayollarında ticari kaygı nedeni ile gecikmeler son derece önemlidir. Havaalanları bir ağa bağlandığından bir havaalanında oluşan her gecikme dakikası, ağa bağlı diğer havaalanlarını ve dolayısıyla bağlı bütün seferleri, bu seferlerde görev alan tüm hava aracı ve uçuş personelini etkiler. İptal edilen ya da gecikmeye konu olan seferin havayolu şirketine getirdiği ekonomik maliyetin yanı sıra en kıt kaynaklar olan hava araçları ve uçuş ekiplerinin çalışma programlarının bozulması, giderilmesi çok daha menfi durumlara sebep olmaktadır. Meydana gelen gecikmelerin temel nedenleri; mürettebat eksikliği, kötü hava koşulları, mekanik bozukluklardır. Mürettebat eksikliği ve mekanik bozukluklar daha çok operasyonel şartlara bağlı olarak değişirken, kötü hava koşulları mevsime bağlı olarak çoğunlukla benzer davranış göstermektedir (Feo and Bard 1989).

Uçuş aksaklıklarının yaşanmasını önlemek için çeşitli alternatifler vardır. Klasik yaklaşım olarak aksaklıklar ile baş edebilmek için aksaklığın olduğu gün tarifeler yeniden düzenlenir ve çözüm üretilir. Fakat bu durum yolcu memnuniyetinin azalmasına, ekstra zaman kaybına ve hatta havayolu şirketleri için ekstra masrafa neden olur. Bu durumun önüne geçebilmek için, hava yolu trafik performansında uçuş gecikmelerini en aza indirecek bir sistem anlayışı gerekmektedir. Bu sistem anlayışında, uçuş çizelgeleme probleminin çözülmesi amacıyla yeni model geliştirilir, bu model ile meydana gelecek gecikmeler eski verilere dayanarak daha önceden belirlenip yeni tarifelerin oluşturulması sağlanır. Oluşturulacak tarifenin etkinliği, operasyonel süreçlerle uyumluluğunun yanı sıra tüm kısıtların en uygun düzeyde karşılanmasını sağlayan modelin doğruluğuna da bağlıdır. Bu nedenle kurulan model, performansın artırılması açısından, optimizasyon işleminde ayrıca önemlidir. Problemin optimizasyon ile çözülmesinin en önemli nedeni, zamandan tasarruf sağlamaktır. Çözüm uzayı çok büyük olduğu için bu uzayda etraflı arama yapmak çok zordur. Arama uzayının tamamı dolaşılmak istenmemiştir. Etkili bir şekilde dolaşılması amaçlanmıştır. Optimizasyonda en iyi olanı bulma işlemi, problemin çözümü için kullanılan algoritmanın amacına yönelik, uygunluk fonksiyonunu minimum ya da maksimum yapma işlemine göre değişmektedir. Birçok optimizasyon yöntemi

mevcuttur. Optimizasyon problemlerinin birçoğunun matematiksel model geliştirilerek çözülmesi ya da klasik yöntemler geliştirilerek çözülmesi çok zaman alabilmektedir. Bu yüzden bu çalışmada metasezgisel yöntemler geliştirilerek optimum sonuca ulaşılmaya çalışılmıştır.

1.1. Tezin Araştırma Kapsamı ve Amaçları

Uçuş çizelgeleme problemi, havayolu sektörünün yanı sıra demiryolu, karayolu ve denizyolu ulaşım sektörlerine de genişletebilecek tabiattadır. Şans kısıtları vasıtasıyla karar verme tarım sektöründe de uygulanabilir. Mesela hava şartlarına bakılarak hasat, sulama ve ekim zamanları en uygun şekilde belirlenebilir.

Bu tez çalışmasında, olumsuz hava koşulları dikkate alınarak, metasezgisel optimizasyon algoritmalarından benzetilmiş tavlama, genetik algoritma ve yapay arı koloni algoritması aracılığı ile problem çözülmüştür. Böylece, minimum maliyet verecek şekilde, gecikmelerin minimum düzeyde olması sağlanarak, uçuş çizelgeleme probleminin çözülmesi hedeflenmiştir. Ele alınan uçuş tarifelerinde, her bir tarife için en optimum zaman aralığı saptanmıştır.

1.2. Organizasyon

Bu çalışmanın devamındaki akış düzeni aşağıdaki gibi yapılacaktır.

- Bölüm 2'de kaynak özetleri başlığı altında, uçuş çizelgeleme problemi ile ilgili yapılan çalışmalardan bahsedilmektedir.
- Bölüm 3'de materyal ve yöntem başlığı altında, üzerinde çalışılan ham veriye ait bilgiler, materyalde verilmiştir. Araştırmanın amacına ulaşmada kullanılan tekniklerden ise yöntemde bahsedilmiştir.
- Bölüm 4'de araştırma bulguları başlığı altında, çalışmadan elde edilen deney sonuçları verilmiştir.

- Bölüm 5'de tartışma ve sonuç başlığı altında, çalışmanın literatürdeki çalışmalar ile karşılaştırılması ve elde edilen bulguların yorumlanması yapılmıştır.

1.3. Büyük Veri Açısından Uçuş Verilerinin Önemi

Büyük veri, klasik yöntemler ile işlenmesi zor olan, yapısal ve yapısal olmayan global veriyi temsil etmektedir. Uçuş verileri, büyük verilerin en etkili inceleme alanlarından biri olarak gösterilebilir. Bu veriler, yığın ya da akan veri şeklinde olabilir. Bu çalışmada, yığın şeklinde olan büyük veriler ile ilgilenilmiştir. Hızlı büyüyen verinin işlem hızının da aynı oranda artıp hızlı çözüm sunması gerekmektedir. Bu ihtiyacı karşılamak için, normal analiz yöntemleri yeterli olmamaktadır. Bu sebeple, büyük veri analiz etme teknolojileri günümüzde kullanılmaktadır. Bu çalışma kapsamında, Apache Spark kullanılarak analiz işlemi gerçekleştirilmiştir. Bu araç ile hava durumundan kaynaklı uçuş gecikmesinin yaşandığı zaman dilimi tespit edilmiştir. Oluşturulacak tarifelerde bu dilimler dikkate alınmıştır.

1.4. Uçuş Çizelgeleme Problemi

Havayolu sektöründe temel çizelgeleme problemlerinden biri uçuş çizelgeleme problemidir. Çizelgeleme, sınırlı kaynakların belirli gaye doğrultusunda, belirli zaman aralığında ilgili işlere atanması ile ilgili süreçtir. Uçuş çizelgelemesi, hangi uçuş noktalarına, ne kadar sıklıkla hizmet verileceğini tanımlayan bir problemdir (Orhan *et al.* 2010). Uçuş çizelgeleme probleminde, mevcut tarifeler, sert ve esnek kısıtlar altında yeniden hazırlanarak havayolu işletmesinin planlaması yapılır. Kesin çözüm veren algoritmalar (doğrusal programlama vb.) ve yaklaşık çözüm veren algoritmalar (sezgisel, metasezgisel) olmak üzere 2 çizelgeleme çözüm yöntemi vardır. Bu yöntemler ile çizelgeleme problemlerinin optimizasyonu yapılmaya çalışılır.

1.5. Sezgisel Algoritmalar-Metasezgisel Algoritmalar

Gündelik yaşamda, insanlar ve doğadaki diğer canlılar farkında olmadan bazı özellikleri (parametreleri) dikkate alarak sezgisel yaklaşımlarla iç içe yaşamaktadır. Canlıların sezgisel davranışlarının bilgisayarlara nasıl aktarılacağı, problemlerin çözümünde bu yaklaşımların nasıl temsil edileceği önemlidir. Sezgisel yaklaşım algoritmik yaklaşımlarda olduğu gibi amaca ulaşırken kesin olarak izlenmesi gereken yolu vermemektedir. Sezgisel algoritmalar, ilk olarak durum uzayında, mümkün olabilecek çözümler içerisinden rastgele bir çözüm kümesi ele alır. Bu çözüm kümesine mümkün yöntemler uygulanarak mevcut çözüm değiştirilir. Mevcut durumun değerlendirilmesi yapılır. Eğer gereksiz durum varsa atılır, aksi halde çözüm değerlendirmeye alınır ve işlemler tekrarlanır.

Sezgisel yöntemlerin her zaman çalışmadığı konusunda çeşitli tartışmalar vardır. Aslında bu tartışmaların temel sebebi problem alanının genişliği ile ilgilidir. Bir problem için çalışabilen bir sezgisel algoritma diğer problem için çalışmayabilir. Sezgisel programlamanın anlaşılması için problem çeşidinin net olarak anlaşılması gerekir.

Sezgisel algoritmalar, herhangi bir amaca ulaşabilmek için, elde mevcut bulunan alternatif çözümleri kullanarak özel geliştirilmiş yöntemler uygular. Bu algoritmalar, bulduğu sonuçların doğruluğunun ispat edilemeyebileceğini ama iyi çözümler verebileceğini vadeder.

Metasezgisel algoritmalar, sezgisel algoritmaların üstünde çalışır. Uygun parametreler ayarlanarak birden fazla problem için çözüm geliştirme imkanı sunar. Problemden bağımsız çalışan algoritmalardır. Metasezgisel algoritmalar rastgele bir çözüm ile başlar. Her adımda bir çözüm iyileştirilmeye çalışılır. Metasezgisel algoritmalarda, bulunan çözüm yerel optimum ise bu çözümün global olmadığını bildirecek yöntem bulunur. Metasezgisel algoritmaların arama uzayı, problemin çözüm kümesinden oluşmaktadır.

1.6. Karar Problemleri

1.6.1. Hesaplama teorisi

Hesaplama Teorisi, bilgisayar dünyasında bir problemin çözümünün belirli bir algoritma ile çözülme durumunu ve çözülebilirse ne kadar zamanda çözülebileceğini incelemektedir.

1.6.2. Karmaşıklık teorisi

Bu teori, hesaplama teorisinin alt dalı olarak düşünülebilir. Algoritma problem üzerinde çalışırken ne kadar zaman harcandığı ile ilgilenir.

Hesaplama teorisi ve karmaşıklık teorisi karar problemlerinin önemli bir kısmını oluşturur. Karar problemleri, NP Zor (deterministik olmayan) problemlerin tanımında kullanılan, çözümü evet ya da hayır olan problemlerdir. Optimizasyon problemi karar verme problemine indirgenebilir. Bu sayede problem çözülmüş olur. Karar verme problemleri P (polinomsal zaman) ve NP (deterministik olmayan polinomsal zaman) olmak üzere iki ana başlık altında toplanabilir.

1.6.2.a. P

Polinomsal zamanda problemin çözümünün “evet” ya da “hayır” a dönüştürülebildiği sınıftır. Bu problem sınıfına giren problemler iyi biçimlendirilmiş problemlerdir. Problemi üstten sınırlamak ve durum uzayını belirlemek mümkünse benzer problemlerin çözümü de algoritmik yaklaşımla çözülebilmektedir. Örneğin, $O(n^2)$, kabarcık sıralama algoritmasının zaman karmaşıklığıdır. Kabarcık sıralama algoritması P sınıfı algoritmaya girmektedir. Lineer zamanda bu algoritmanın çözümü gerçekleşebilmektedir. Bu problemler deterministik turing makinesinde çözülebilmektedir. Deterministik Turing

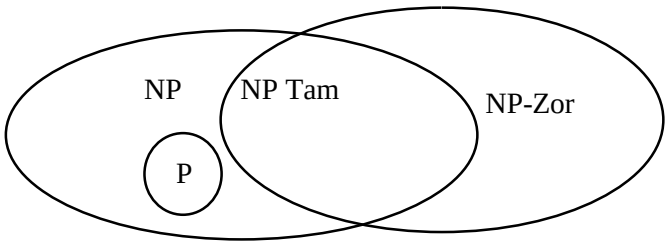
makinesinde, verilmiş bir anda, otomat bir durumda bulunmaktadır. Bazı işlemler sonucunda bir sonraki durumun ne olacağı bellidir.

1.6.2.b. NP

Problemi çözmek için elde yeterli bilgi yoktur. Polinomsal zamanda problem çözülememektedir ama çözüm getirilirse bu çözümün karşılığı bulunabilmektedir. O çözümün amaç fonksiyonundaki değeri söylenebilir. Örneğin, çoktan seçmeli sorularda şıklardan giderek doğru sonuca ulaşma NP sınıfı algoritmaya girer.

NP sınıfına giren problemler kötü biçimlendirilmiş problemlerdir. Bu problemler, polinomsal zaman içinde deterministik olmayan makinalarda çözülebilmektedir. Deterministik olmayan Turing makinesinde, belirli bir anda otomatın geçiş yapacağı durum belli değildir, deneme yanılma yolu ile bir sonraki duruma geçiş yapılır.

P ve NP için alt küme ilişkisi bulunmaktadır. Şekil 1.1'de görüldüğü gibi her P problem aslında NP problemdir.



Şekil 1.1 P-NP ilişkisi

1.6.2.c. NP zor

Polinomsal zamanda, bir çözümü olduğu ispatlanamayan problem çeşitlerini içeren sınıftır. Eğer NP sınıfında yer alan bir "E" karar problemi, polinomsal zamandaki "B" ye indirgenebiliyorsa, o zaman NP Zor problem olur. Bu problem sınıfı en az bir NP Tam kadar zordur. NP Tam'da yer alan problem, NP Zor sınıfına indirgenebilir. X problemi

NP Tam ise ve bu problemin polinomsal zamanda çözümü bilinirse, diğer problemler de bu çözüm ile çözülebilir.

Bu tezdeki problemin çözümü polinomsal zamana indirgenebiliyor. Bu yüzden, NP Zor sınıfı bir problemdir. Çözümünde metasezgisel yöntemler kullanılmıştır. Metasezgisel algoritmalar, sonucun doğruluğunun kanıtlanabilir olup olmadığını önemsemeden, belli süre zarfında en iyi çözüm ya da en iyi çözüme yakın sonuçlar vereceğini vadeden algoritmalardır.

1.6.2.d. NP tam

NP sınıfında yer alan bazı problemlerin polinomsal zamanda çözülmesi çok zordur. Bu problemler, NP Tam'a girmektedir. Bir "X" problemi, NP sınıfında ve NP sınıfındaki her problem polinomsal zamanda X'e indirgenebiliyorsa, bu problem NP Tam sınıfına girer. X problemi NP Tam olup, bu problemin polinomsal zamanda çözümü biliniyorsa, diğer problemler de bu çözüm ile çözülebilir. Polinomsal zamanda doğrulanabilen karar problemi, hem NP hem de NP Zor olursa, bu problem NP Tam karmaşıklık sınıfına girer. Bu nedenle NP Tam sınıfına giren problemler, en zor sınıf kategorisine girmektedir.

2. KAYNAK ÖZETLERİ

Havayollarında meydana gelen uçuş iptali ya da gecikmelerini etkileyen temel faktörler olarak; hava koşulları, mekanik problemler, gelen uçuş programlarındaki gecikmeler gösterilmektedir. Havayolu şirketlerinin bu faktörlere gerçek zamanlı, en uygun çözümü bulması gerekmektedir. Jarrah *et al.* (1993) geliştirdikleri karar destek sistemi ile havayolu şirketlerine yardımcı olmaya çalışmışlardır. Bu sistemde iki ağ modeli oluşturmuşlardır. Wu and Caves (2003) tarafından yapılan çalışmada ise, uçuş tarifesi oluştururken, uçuş gecikmelerini minimum düzeye getirecek çözüm yöntemi araştırılmıştır. Bu amaçla, uçuş kalkış saatlerini modelleyip, stokastik tabanlı bir yaklaşım sunulmuştur. Model sonuçlarını incelediklerinde, kalkış zamanının planlamasına ek olarak uçak yer hizmetlerinin operasyonel verimliliğinin gecikmeleri önemli ölçüde etkilediğini tespit etmişlerdir. Uçuş gecikmelerinin sebebi, havaalanlarındaki alt yapı kapasitesinin yetersizliği olarak gösterilebilir. Havayollarında uçuşlar için harcanan maliyet önemli ölçüdedir. Maliyetleri kontrol altına almak için, potansiyel bir kaynak olan, bakım işlemlerine önemli ölçüde odaklanılır. Ancak iç güvenlik politikaları ve federal düzenlemeler maliyetten yapılacak tasarrufları, iyileştirmeleri önemli ölçüde sınırlayabilmektedir. Feo and Bard (1989) oluşturdukları model ile bakım istasyonlarının yerleştirilmesini sağlayacak ve ihtiyaç duyulan talebi karşılayacak, uçuş çizelgelemesi oluşturmuşlardır. Problemi minimum maliyetli, bir ağ ile temsil etmişlerdir. İki fazlı bir sezgisel yardımı ile problem çözülmüştür. Boeing 727, Super 80, DC-10 filolarına ait, Amerika havayolları tarafından temin edilen veriler kullanılmıştır. Elde ettikleri sonuçlarda, sezgisel yöntemin, mevcut tekniklere göre maliyeti önemli ölçüde düşürdüğü gözlemlenmiştir. Adachi *et al.* (2004) tarafından uçak uçuş programlarını geliştiren ve bu uçuşlar için gereken toplam uçak sayısını minimuma indiren bir sistem geliştirilmiştir. Bu sistemdeki çözüm yaklaşımı olarak genetik algoritma kullanılmıştır. Elde edilen sonuçlarda 201'den 193'e ve 94'den 92'ye kadar uçak sayısını düşürebildiklerini gözlemlemişlerdir. Sezgisel algoritmalar uçaklar için, yeni uçuş çizelgeleri bulmak amacıyla, ağaç arama algoritmasını kullanır. Yapısal olarak farklı çözümler arasındaki en optimum sonuca ulaşmak için farklı stratejik yollar uygular. Andersson (2006) tarafından, İsveç havayolundan alınan uçuş verileri üzerinde, tabu arama ve benzetilmiş tavlama

algoritmaları kullanılmıştır. Uçuş pertürbasyon problemi üzerine, çözüm yaklaşımı geliştirilerek sonuçlar test edilmiştir. Elde edilen sonuçlarda, tabu algoritmasının, benzetilmiş tavlama algoritmasına göre daha iyi sonuçlar verdiği gözlemlenmiştir. GAO *et al.* (2009), metasezgisel algoritmalardan, açgözlü rastgele adaptif arama ve benzetilmiş tavlama algoritmalarının özelliklerini bütünleştiren yeni bir algoritma geliştirmiştir. Algoritma, oluşturulan uçuş çizelgeleme modeli üzerinde uygulanmıştır. Düzensiz (gecikmeye uğramış vb.) gerçekleşen 50 uçak ve 200 uçuş için %14 ve %19 aralığında toplam maliyetin azaldığı ve yaklaşık 90 milyon RMB (Çin Yuanı) kurtarılabileceği görülmüştür. Liu *et al.* (2010), uçak çizelgeleme probleminin hibrit çok amaçlı genetik algoritma ile nasıl optimize edilebileceği gösterilmiştir. Jungai and Hongjun (2012) tarafından, uçuş gecikmelerini optimize etmek amacıyla, benzetilmiş tavlama algoritması kullanılarak, bir model oluşturulmuştur. Modelin doğruluğu, Çin'de bulunan bir havaalanına ait uçuş bilgilerinde yer alan, gecikme maliyetindeki toplam değişim üzerinde test edilmiştir. Test sonucunda, maliyetin düştüğü gösterilmiştir. Xiang and Tang (2014) yaptıkları çalışmada, bir çizelgeleme optimizasyon modeli oluşturmuşlardır. Bu modele, uygun çözümlerin elde edilebilmesi için, bir genetik algoritma stratejisi geliştirmişlerdir. Bu strateji ile, uçuşlardan maksimum kazanç elde edilebileceğini ifade etmişlerdir. Uçuş planlaması (çizelgeleme, tarife oluşturma) uçuş gecikmelerini ve maliyetlerini azaltan teknolojilerin kilit noktasıdır. Verimli uçuş çizelgeleme, sadece havalimanının etkin kullanımını iyileştirmekle kalmaz aynı zamanda ani kazaların görülme ihtimalini azaltır. Uçuş emniyetini sağlar. Liang and Li (2014) tarafından yapılan çalışmada, yeni uçuş sıralama metodu olarak, karınca koloni algoritması ve genetik algoritmanın kombinasyonu olan ACO (Karınca Koloni Algoritması) – GA (Genetik Algoritma) algoritması önerilmiştir. Yapılan çalışmada, elde edilen simülasyon sonuçlarına göre, ACO-GA algoritmasının, FCFS (İlk Gelen İlk Hizmeti Alır Algoritması) ve ACO algoritmalarına göre daha iyi sonuçlar verdiği gözlemlenmiştir. Uçuş gecikmeleri önemli ölçüde azaltılmıştır. Endonezya Ulaştırma Bakanlığı tarafından yapılan açıklamaya göre, 2013 yılında Endonezya'da %20'den fazla uçuş gecikmesi olmuştur. Bu durum, havayolları için ciddi bir problem oluşturmuştur. Havayolu şirketi kârı ve yolcu memnuniyetinin artırılabilmesi için, bu gecikmelerin azaltılması gerekmektedir. Bu sebeple, havayolu şirketleri muhtemel gecikmeleri önceden tahmin etmelidir. Uçuş gecikme durumları modellenerek bu durum gerçekleştirilebilir. Uçuşlar

için, kalkış gecikme süresinin ve kalkış gecikme zamanının dağılımı, optimizasyon modellerinin uygun yoğunluk parametrelerinin tahmin edilmesi ile modellenebilir. Bu modelleme için, Garuda Endonezya havayolu şirketinden alınan 2012 yılına ait geçmiş kalkış gecikme verileri üzerinde, uçuş kalkış gecikme durumları için bir olasılık dağılımı oluşturulmuştur. Daha sonra iki aşamalı genetik algoritma uygulanarak, en uygun model seçilmiştir. İlk aşamada, maksimum ihtimaliyet fonksiyonunun logaritması çalıştırılmıştır. İkinci aşamada ise, en küçük kareler yöntemi optimize modeli ile işlem yapılarak, ilk aşamada yerel minimumda sıkışıp kalma durumu ortadan kaldırılmaya çalışılmıştır (Novianingsih and Hadianti 2014). Uçuş gecikmelerinin ciddi olduğu, büyük havaalanlarındaki iki pistin, uçuş çizelgelemelerini optimize etmek için, genetik algoritma ve yapay balık sürü algoritmanın füzyonu olan bir algoritma önerilebilir. Önerilen bu algoritmanın, FCFS, AFSA (Yapay Balık Sürü Algoritması) ve GA ya göre daha yüksek düzeyde verim sağladığı simülasyon sonuçlarında gösterilebilmektedir (Li *et al.* 2014).

Son yıllarda yapılan ölçümlere göre, uçuş gecikmeleri önemli boyutlara ulaşmıştır. Sternberg *et al.* (2016) tarafından bildirildiğine göre, Brezilyada ortalama 30 dakikadan fazla süre içerisinde, yurtiçinde %19,1 uçuş gecikmesi olmaktadır (Sternberg *et al.* 2016). Yolcuların gecikmeler, iptaller konusundaki şikayetleri ABD kongresinin daha güçlü önlemler almasını sağlamıştır (Wong and Tsai 2012). Bir havaalanında, olumsuz hava koşulları beklendiğinde, ABD'deki Federal Havacılık İdaresi, bu havaalanı için zemin gecikme programı yayınlar. Zemin gecikme programının amacı, olumsuz hava koşullarında art arda olan uçuş inişleri arasındaki zaman aralığını artırmaktır. Zemin gecikme programı ile uçuş gecikmelerinin önüne tam olarak geçilememektedir. Hazırlanan çoğu zemin gecikme programında, bir uçuş yakınlardaki bir havaalanına yönlendirilmekte ya da iptal edilmektedir. Bunun temel sebebi olarak, uçuş inişleri için mevcut yer sayısının, planlanan yer sayısından daha az olması gösterilmektedir. Abdelghany *et al.* (2008) yaptıkları çalışmada, varsayımsal olarak 8 istasyon ve 51 uçuş hava ağında DSTAR adında bir karar destek sistemi geliştirmişlerdir. Bu sistem açgözlü optimizasyon yaklaşımını uygulamakta ve yönlendirilmiş graf ile temsil edilmektedir. Bu grafın düğümleri, planlanan uçuş kalkış zamanı, uçuş varış zamanı, uçuş bakım zamanı

gibi kavramları temsil ederken, grafın kenarları ise, bu olaylar arasındaki zamanlaşmış faaliyetleri temsil etmektedir. Bu model ile gecikmelerin önüne geçilebildiği gösterilmiştir.

Bu tez çalışmasının, incelenen çalışmalar ile ortak özelliği uçuş gecikmesini önleyecek, uçuş çizelgelemesini yapmak hedeflenmiştir. Ama en temel farklılık olarak, gidilecek güzergâhın hava koşullarını da dikkate alarak bu işlem yapılmıştır. İncelenen çalışmalarda, genel olarak tek bir havaalanına ait bir yıllık veriler, klasik yöntemler ile dikkate alınmıştır. Bu tez çalışmasında ise büyük veri teknolojisi yardımı ile 32 yıllık veri, metasezgisel yöntemler ile değerlendirilmeye alınmıştır.

3. MATERYAL ve YÖNTEM

Bu tez çalışmasındaki üzerinde çalışılan veriden, veri analizi için kullanılan araçtan, problem için oluşturulan modelden ve optimizasyon için kullanılan algoritmalardan bahsedilmiştir.

3.1. Veri

Araştırma materyali olarak, uçuş gecikmesi ihtimalinin belirlenen üst sınırların altında olmasını temin eden doğrusal kısıtların girilmesi gerekmektedir. Bir uçuşun, beklenen kalkış zamanı ile gerçek kalkış zamanı arasında ya da beklenen varış zamanı ile gerçek varış zamanı arasında farkının olması, gecikmeli bir durumun olduğunu gösterir. Bu çalışmada, beklenen varış zamanı ile gerçek varış zamanı arasında farkı olan gecikmeli uçuşlar dikkate alınmıştır. Gecikmeli inme ihtimalleri için kabul edilebilir azami üst sınırlar modele önceden parametre olarak girilmelidir. Bu amaçla, kısıt veri seti oluşturulurken, kullanılacak olan, 1987-2018 yılları arasındaki uçuş verileri BTS'den (Ulaştırma İstatistikleri Bürosu) (Anonymous 1992) alınmıştır. Tedarik edilen bu veriler, herkes tarafından kullanılabilen, düzenlenebilen açık verilerdir. BTS, Amerika Birleşik Devletleri Ulaştırma Bakanlığı'nın bir parçasıdır. Kullanıcılara istatistiksel bir yapı sunarak, ülkenin ulaştırma sistemleri hakkında ayrıntılı analiz sonuçları verir. Temin edilen uçuş veri setleri içerisinde yer alan özelliklerden çalışma içerisinde kullanılan değişkenlerin genel bilgileri Çizelge 3.1'de verilmiştir.

Çizelge 3.1. Uçuş veri setine ait genel bilgiler

Yıl	Ay	Gün	Hava Durumu Gecikmesi	Varış Gecikmesi	Kalkış Saati	Veri Seti Büyüklüğü	Kayıt Sayısı
1987	1-12	1-31	(dakika cinsinden)	(dakika cinsinden)	(yerel)	121MB	1 311 826
1988	1-12	1-31	(dakika cinsinden)	(dakika cinsinden)	(yerel)	477MB	5 202 096
1989	1-12	1-31	(dakika cinsinden)	(dakika cinsinden)	(yerel)	463MB	5 041 200

Çizelge 3.1. (devam)

1990	1-12	1-31	(dakika cinsinden)	(dakika cinsinden)	(yerel)	485MB	5 270 893
1991	1-12	1-31	(dakika cinsinden)	(dakika cinsinden)	(yerel)	468MB	5 076 925
1992	1-12	1-31	(dakika cinsinden)	(dakika cinsinden)	(yerel)	469MB	5 092 157
1993	1-12	1-31	(dakika cinsinden)	(dakika cinsinden)	(yerel)	468MB	5 070 501
1994	1-12	1-31	(dakika cinsinden)	(dakika cinsinden)	(yerel)	478MB	5 180 048
1995	1-12	1-31	(dakika cinsinden)	(dakika cinsinden)	(yerel)	506MB	5 327 435
1996	1-12	1-31	(dakika cinsinden)	(dakika cinsinden)	(yerel)	509MB	5 351 983
1997	1-12	1-31	(dakika cinsinden)	(dakika cinsinden)	(yerel)	515MB	5 411 843
1998	1-12	1-31	(dakika cinsinden)	(dakika cinsinden)	(yerel)	513MB	5 384 721
1999	1-12	1-31	(dakika cinsinden)	(dakika cinsinden)	(yerel)	527MB	5 527 884
2000	1-12	1-31	(dakika cinsinden)	(dakika cinsinden)	(yerel)	543MB	5 683 047
2001	1-12	1-31	(dakika cinsinden)	(dakika cinsinden)	(yerel)	572MB	5 967 780
2002	1-12	1-31	(dakika cinsinden)	(dakika cinsinden)	(yerel)	505MB	5 271 359
2003	1-12	1-31	(dakika cinsinden)	(dakika cinsinden)	(yerel)	597MB	6 488 540
2004	1-12	1-31	(dakika cinsinden)	(dakika cinsinden)	(yerel)	638MB	7 129 270
2005	1-12	1-31	(dakika cinsinden)	(dakika cinsinden)	(yerel)	639MB	7 140 596
2006	1-12	1-31	(dakika cinsinden)	(dakika cinsinden)	(yerel)	640MB	7 141 922
2007	1-12	1-31	(dakika cinsinden)	(dakika cinsinden)	(yerel)	670MB	7 453 215
2008	1-12	1-31	(dakika cinsinden)	(dakika cinsinden)	(yerel)	657MB	7 009 728

Çizelge 3.1. (devam)

2009	1-12	1-31	(dakika cinsinden)	(dakika cinsinden)	(yerel)	468MB	6 450 296
2010	1-12	1-31	(dakika cinsinden)	(dakika cinsinden)	(yerel)	468MB	6 450 128
2011	1-12	1-31	(dakika cinsinden)	(dakika cinsinden)	(yerel)	440MB	6 085 292
2012	1-12	1-31	(dakika cinsinden)	(dakika cinsinden)	(yerel)	406MB	5 602 554
2013	1-12	1-31	(dakika cinsinden)	(dakika cinsinden)	(yerel)	462MB	6 369 493
2014	1-12	1-31	(dakika cinsinden)	(dakika cinsinden)	(yerel)	422MB	5 819 822
2015	1-12	1-31	(dakika cinsinden)	(dakika cinsinden)	(yerel)	422MB	5 819 090
2016	1-12	1-31	(dakika cinsinden)	(dakika cinsinden)	(yerel)	408MB	5 617 669
2017	1-12	1-31	(dakika cinsinden)	(dakika cinsinden)	(yerel)	417MB	5 674 632
2018	1-5	1-31	(dakika cinsinden)	(dakika cinsinden)	(yerel)	211MB	2 915 416

Çizelge 3.2'de verilen havaalanlarına ait, 6 aylık (Aralık-Mayıs) mevcut uçuş tarife bilgileri, Passrider'dan (Anonymous 2002) alınmıştır. Passrider, havayolu tarifelerinin oluşturulduğu, çeşitli seyahat bilgilerini içeren uçuşlara ait, tarife bilgilerini almamızı sağlayan kuruluştur. Tedarik edilen bu veriler, herkes tarafından kullanılabilen, düzenlenebilen açık verilerdir.

Çizelge 3.2. Tarife bilgisi alınan havaalanı isimleri (Anonymous 2015)

YVR – Vancouver Havaalanı
SFO – San Francisco Havaalanı
JFK – New York Havaalanı

Temin edilen uçuş tarife bilgisine ait kısa bir kesit örnek olarak aşağıda verilmiştir (Çizelge 3.3). Çalışma içerisinde, 3 tarife çeşidi kullanılmıştır. Birinci tarife, kalkış yeri "SFO", varış yeri "JFK" olan uçuşları temsil etmektedir. İkinci tarife, kalkış yeri "JFK", varış yeri "SFO" olan uçuşları temsil etmektedir. Üçüncü tarife, kalkış yeri "SFO", varış yeri "YVR" olan uçuşları temsil etmektedir. Birinci tarife, ikinci tarife ve üçüncü tarife **EK 1**'de verilmiştir.

Çizelge 3.3. Örnek uçuş tarife bilgileri

Gün	**Ay**	**Kalkış Yeri**	**Kalkış Zamanı**	**Varış Yeri**
13	12	SFO	19:07	YVR
…	…	…	…	…
1	5	JFK	21:50	YVR

Bu çalışma içerisinde optimizasyon ile her bir tarife bilgisi yeniden şekillendirilmiştir. Bu işlem yapılırken, 2 farklı uçak tipinde uçuş gerçekleştirilebileceği kabul edilmiştir. Bu uçak tipi bilgileri aşağıda verilmiştir (Çizelge 3.4).

Çizelge 3.4. Uçak tipi bilgileri (Anonymous 2015)

1– The Boeing 787-9(789)(Dreamliner)
2 – The Airbus A320-200(320)

3.2. Veri Analizi İçin Kullanılan Araçlar

Büyük veriyi analiz etmek için kullanılan teknolojiden, verinin bilgisayarda depo edilme şeklinden, problemin ifade edildiği matematiksel modelden ve metasezgisel algoritmalardan bahsedilmiştir.

3.2.1. Apache spark

Apache Spark, büyük veriyi kolay ve hızlı şekilde işleyen, dağıtılmış ve yüksek oranda ölçeklenebilir, bellekte verileri analiz edebilen, her türlü hesaplama için kullanılabilen genel amaçlı en son teknolojilerden biridir (Anonymous 2015).

Spark kendi içinde Mllib, Graphx, SparkSQL, Spark Stream gibi bileşenler barındırdığı için bu bileşenlerin yönetimi kolaydır. Spark; MLLib kütüphanesi sunarak, makine öğrenmesi uygulamaları geliştirmeyi, Graphx sunarak, graph tabanlı uygulamalar yapmayı, Spark Stream sunarak, gerçek zamanlı akan veri işlemeyi, SparkSQL sunarak, SQL sorgularını çalıştırıp Spark'ı veri ambarı gibi kullanmamıza olanak sağlar.

Apache Spark, "bellek içi", "ölçeklenebilir" ve "dağıtılmış" olmak üzere 3 temel özelliğe sahiptir.

3.2.1.a. Bellek içi

Spark veritabanı değildir. Veri bellekte geçici olarak depolanabilir ya da SQL veritabanı, bulut depolama, NoSQL veritabanı, HDFS (Hadoop Dağıtılmış Dosya Sistemi) gibi yapılarda depolanabilir.

Spark bellek içi dağıtık işlem yapabilen bir sistem olduğu için, bu çalışma kapsamındaki veriler, bilgisayarın hafızasından yararlanarak IMDB (Bellek İçi Veri Tabanları Sistemi) içerisinde saklanmıştır (Anonymous 2018). Spark bu veri tabanı üzerinde çalışarak, analiz işlemlerini gerçekleştirir. Bu veri tabanında bütün veriler diske yazılmadan Ram'de tutulur. Analiz yapılırken, fiziksel diskte depolanan veriler üzerinde sorgulama yapmak yerine, hafızada tutulan veriler üzerinde sorgulama işlemleri yapılır. IMDB'nin avantajı; genelde çok hızlı sorgu yanıt süreleri sağlayarak, bellek içi analitik işlemlerde zaman açısından tasarruf sağlamasıdır (Rouse 2015).

3.2.1.b. Ölçeklenebilir

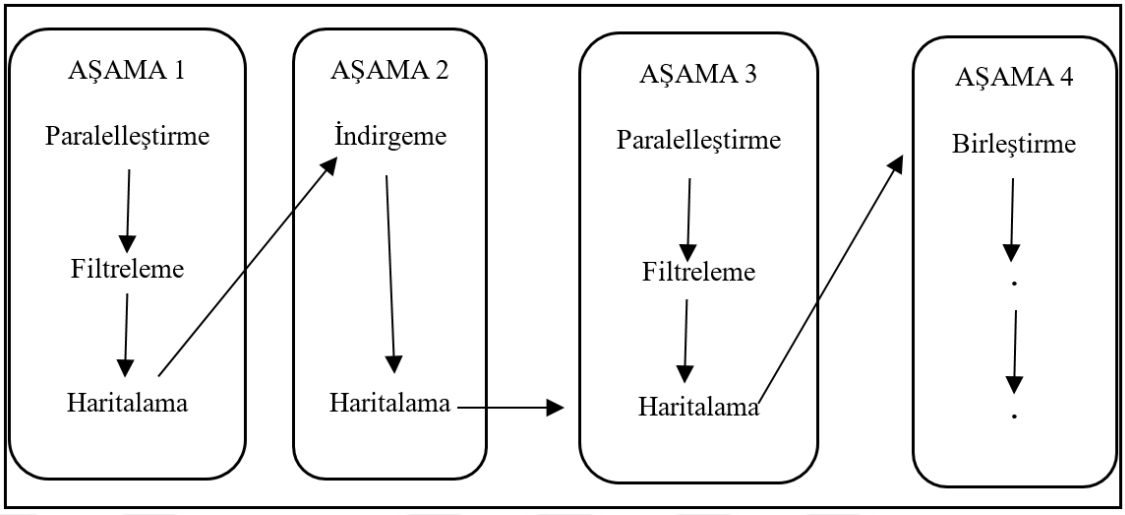
Spark ölçeklenebilir yapıdadır. Bir sistemin artan kapasite kullanımıyla birlikte performanstan kayıp vermeden yapacağı işlemleri sorunsuz gerçekleştirebilen teknolojidir.

3.2.1.c. Dağıtılmış

Daha fazla bellek ya da daha hızlı işlemci, bazen ihtiyaçlara yeterli yanıt verememektedir. Bu yüzden bazen dağıtılmış sistemlere ihtiyaç duyulur. Bu sistemlerde birçok makinaya ihtiyaç vardır. Hadoop gibi sistemler bu makinalarda verileri saklamak için kullanılmaktadır. Tutulan veriler, verinin işleneceği yerde tutulmaktadır. Veriler diskler üzerinde dağıtılır. Spark ise bu anlayışın daha ileri seviyesi olarak, veriyi bellekte dağıtmaktadır. Yerel makine üzerinde çalışırken, eğer bellekte yer yoksa disk üzerine yazmaya başlar. Veri bellekte dağıtıldığı için, yapılan işlemler çok daha hızlı gerçekleşmektedir. Ayrıca Spark'ın performansının hızlı olmasının nedeni, DAG (Yönlü Çevrimsiz Ağaç) yapısını kullanmasıdır (Kane 2017).

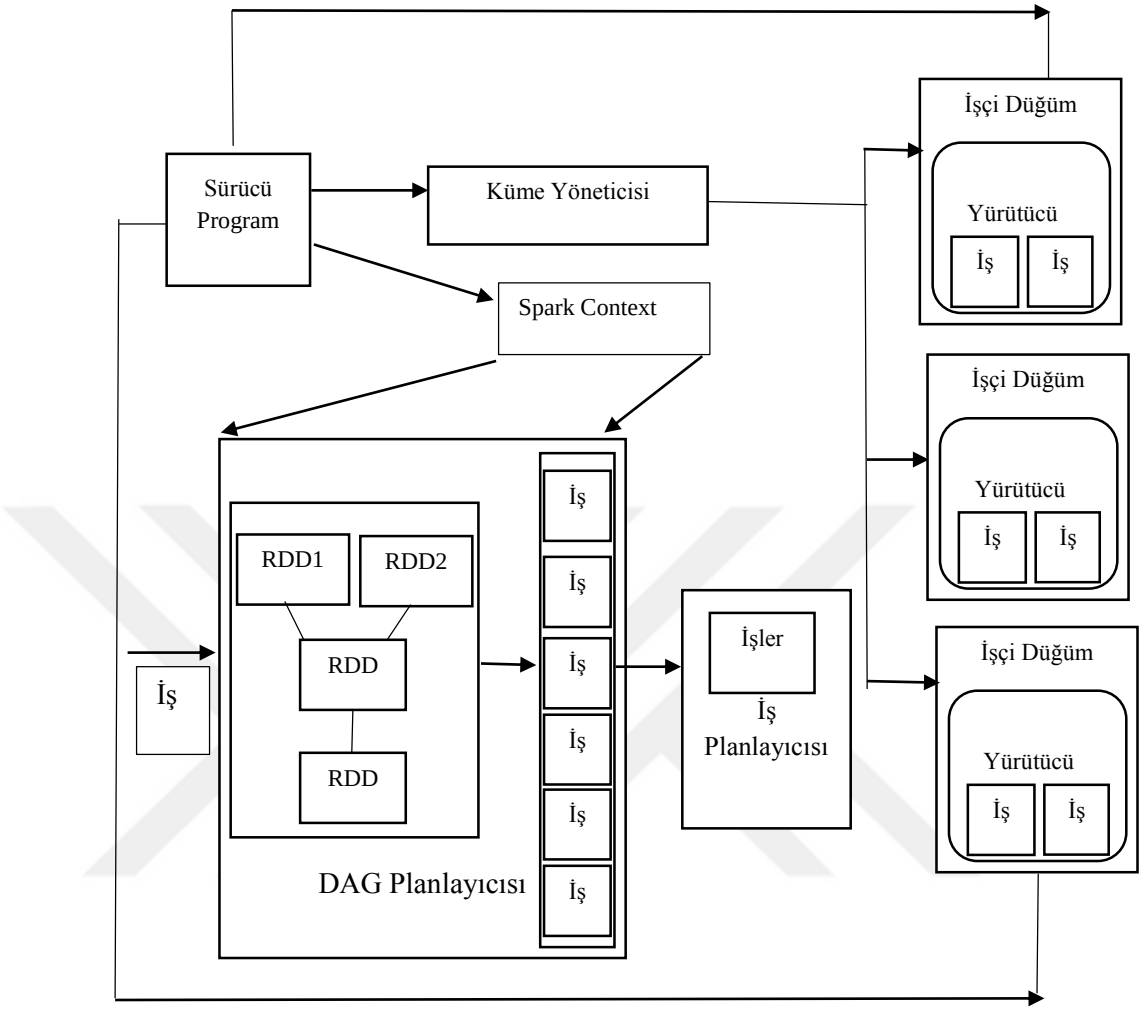
DAG, içerisinde döngü içermeyen yönlendirilmiş sonlu graftır. Her bir kenarın bir köşeden diğerine yönlendirildiği köşe ve kenarlardan oluşur. Yönlü olması, her bir kenarın bir yönü olduğu anlamına gelir. Döngü bulundurmaması ise, bir noktadan birden fazla geçme imkânı bulunmaması demektir. Bir düğümden, diğer düğüme geçerken gösterilen yönde ilerlemek mümkün iken, tersi yönde ilerlemek mümkün değildir.

Spark içinde DAG'ın çalışma şekli incelenecek olursa, köşe noktaları RDD (Esnek Dağıtık Veri Kümeleri) ile temsil edilir. Zamanlayıcı, Spark RDD'yi, uygulanan çeşitli işlemlere dayanarak aşamalara ayırır. Her aşama, aynı hesaplamayı paralel olarak gerçekleştirebilecek olan, RDD bölümlerine dayanan, genişletilmiş detaylara sahip görevlerden oluşur. Her bir görev ise kenarları temsil eder. Grafın burada yönlendirilmesi ve bu işlemin yapılma şekli DAG olarak adlandırılır (Şekil 3.1).



Şekil 3.1. DAG çalışma mimarisi (Anonymous 2017)

Spark içerisinde DAG'a ihtiyaç duyulmasının nedeni, büyük veri teknolojilerinden olan Hadoop üzerinden açıklanabilir. Hadoop, verileri saklamak için HDFS, veri işlemek için ise Mapreduce kullanır. Mapreduce sırası ile HDFS'den veri okur, map ve reduce işlemlerini uygular, sonuçları HDFS'ye yazar. Her bir mapreduce işlemi birbirinden bağımsızdır. Hadoop'un sırada hangi mapreduce işlemi olduğuna dair herhangi bilgisi yoktur. Spark'da ise bir DAG ardışık hesaplama aşaması oluşturulur. Yürütme planı, graf şeklinde temsil edilir. Zaman ve bellek kullanımı açısından, en optimum şekilde sonuca ulaşma hedeflenir. Spark DAG kullanımı ile Hadoop Mapreduce'dan gelen bazı kısıtlamalar ortadan kaldırılmak istenmiştir.



Şekil 3.2. Apache Spark çalışma şeması (Anonymous 2017)

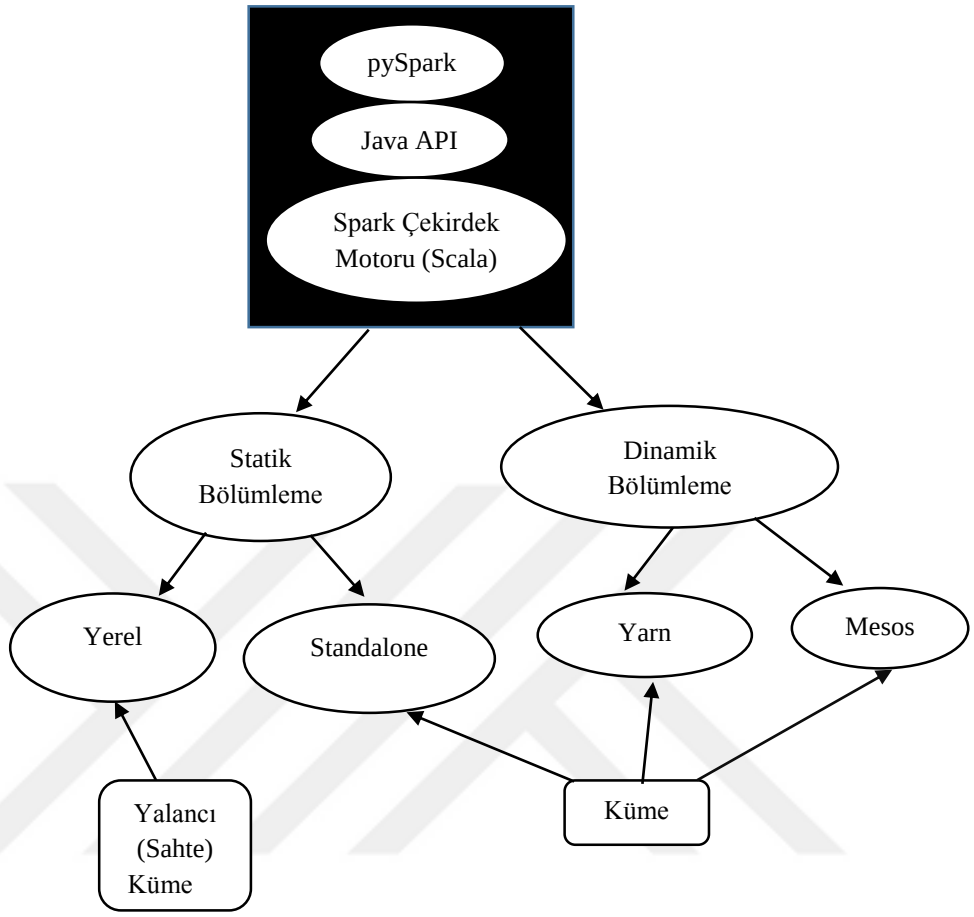
Apache Spark'ın temel çalışma mantığı Şekil 3.2'de incelenecek olursa (Anonymous 2017):

Sürücü program, tek makinada ya da bir kümede ana düğüm olarak bulunan, Spark Context'e sahip, uygulamanın başladığı ana programa sahip süreçtir. Spark, sürücü programın, aynı makine farklı çekirdekte mi yoksa bir kümede farklı bilgisayarlarda mı yerleştirildiğini anlar. Spark'ın çalışmasına yön verir.

Spark'da veriyi işleyebilmek için öncelikli olarak Spark Contex ya da Spark Session (Spark 2.0'dan itibaren) başlatılır. Programlama yaparken, kullanılan "sc" yapısı, Spark

Context anlamına gelir. Başlangıç noktası olarak kabul edilir. Spark Session'da ise, bu yapının karşılığı "spark" şeklindedir. Bu sc nesnesi, bir RDD oluşturmak için gereken yöntemleri sunar. Spark Context, java, python gibi programlama dillerinde geliştirilebilmektedir. Spark Context nesnesi, çalıştırılacak görevlerin işçi düğümlere yönlendirilmesini sağlar.

Küme Yöneticisi, kümeler arasındaki koordineyi sağlayıp, işçilere, işlerin dağıtımını yapar. İşlerin, işçi düğümde çalıştırılmasına izin verir. İşçi düğümden geriye dönen iş sonuçlarını raporlar. İdeal olarak her bir çekirdek başına 1 tane yürütücü dağıtılır. Spark, küme yöneticisi ve sürücü programını kullanarak tüm koordinasyonu yapabilir, hata toleransı verebilir. Spark, Standalone, Yarn, Mesos olmak üzere 3 farklı küme yöneticisinde çalışabilmektedir (Şekil 3.3). Ayrıca Spark yalancı küme yöneticisine sahip, yerel kipde çalışabilir. Spark ilk yüklendiğinde, varsayılan olarak kendi küme yönetici ile karşımıza çıkar (Standalone kipi). Eğer Yarn adında bileşene sahip Hadoop kümesi varsa, bu küme üzerinde dağıtık olarak çalışabilir. Mesos kipinde ise ihtiyaç durumuna göre, kaynak miktarı artırılıp azaltılabilmektedir. Bu sebeple, Standalone kipine göre daha kullanışlı olabilmektedir. Yerel kipin, diğer kiplerden en önemli farkı, Spark'ın tüm bileşenlerinin işlemlerini aynı JVM (Java Sanal Makinesi) içerisinde gerçekleştirmesidir. Tek bir JVM ile program çalıştırılır. Bu, SparkContext nesnelerinin oluşturulduğu, Spark kütüphanelerinin import edildiği, Java, Scala veya Python programlama dillerinden herhangi biri ile olabilir. Anlaşıldığı üzere, yerel kipte dağıtım yani paralelleştirme yoktur. Yerel kipte, paralelleştirme yapılmak istenirse, iş parçacığı sayısı ile istenilen durum gerçekleştirilebilir.



Şekil 3.3. Spark çalışma kipleri

Yürütücü, bir veya daha fazla görevi yürüten süreç olarak, Spark içerisinde yer almaktadır.

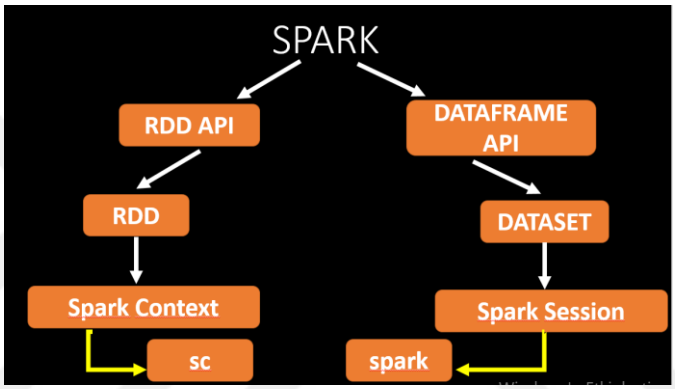
İşçi düğüm, kendi yürütücülerini yöneten süreçdir.

İş, yapılan iş bölümlerinin her biridir.

RDD, Spark etrafında, çekirdek nesnesi olarak bulunmaktadır. Veri kümesi için yapılan soyutlamadır. Herhangi bir kaynaktan veri yükleyerek RDD oluşturulabilir. RDD nesneleri üzerinde, bu verilerin dağıtılması için çeşitli yöntemler araştırılır. Her bir RDD parçalara ayrılarak farklı hesaplama düğümleri tarafından işlenecek şekilde otomatik

olarak dağıtılır. Daha sonra her bir RDD, dönüşümler veya aksiyonlar ile işlenir. RDD'ler yerel olarak çalışan ya da çalışmayan tüm bilgisayar kümesine yayılabilir. RDD kullanarak tekrar eden işlemleri yapmak kolay ve hızlıdır (Kane 2017).

RDD veri tabanındaki bir tablo olarak düşünülebilir. Bu tablo bünyesinde her tipte veriyi tutabilmektedir. Apache Spark, RDD'nin farklı bölümlerinde veri depolayabilme özelliğine sahiptir.



Şekil 3.4. Spark kütüphaneleri

Spark'ın iki kütüphanesi vardır. Bunlar: RDD tabanlı, geliştirilmesine son verilen kütüphane ve Dataframe tabanlı, geliştirilmeye devam edilen kütüphanedir (Şekil 3.4). Spark 2.0'dan önce, RDD API'nin (Uygulama Programlama Arayüzü) alt kümesi olan RDD, Spark'ın soyutlamasıydı. Daha sonra Dataframe API'nin alt kümesi olan Dataset, Spark'ın yeni soyutlaması olmuştur. RDD API, temelde hala var olmasına rağmen, düşük seviyede API'dir (Anonymous 2015). Dataframe'e dayalı API öncelikli API'dir. RDD tabanlı API'e göre, söz dizimi daha kolaydır. Spark dataframe yapısından dolayı, veriyi işlemek için tabular fonksiyonlar kullanabilir (Anonymous 2015). Bu çalışma kapsamında, Dataframe API kullanılmıştır. Dolayısıyla, Spark'da dataframe ile tüm işlemleri yapılabilmek için öncelikli olarak "Spark Session" başlatılmıştır.

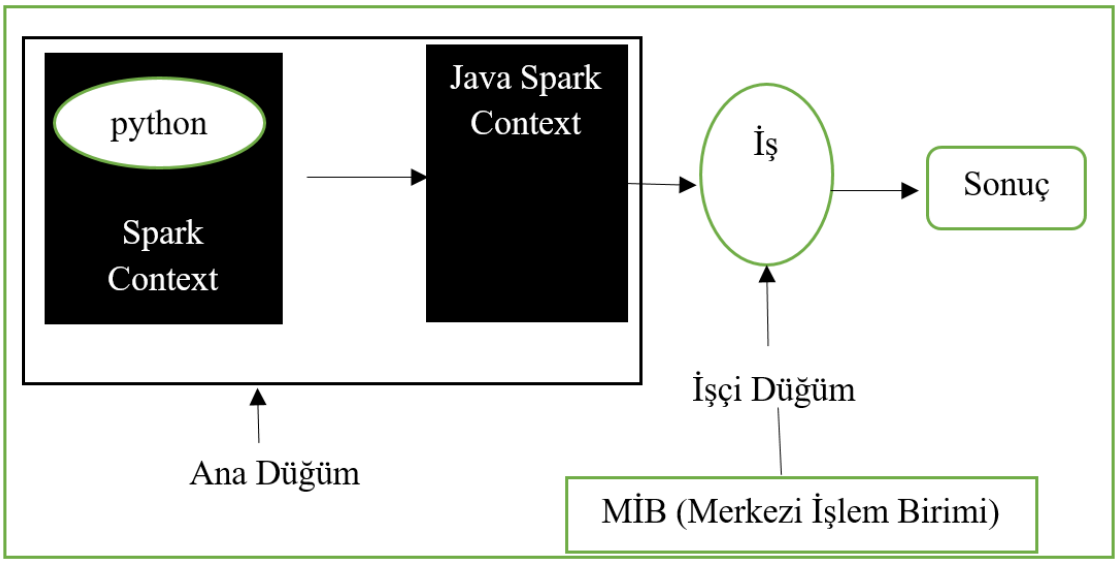
Ana URL, Spark makinesine (motoruna) bağlanacağımız parametre ile kip durumumuzu belirtmemizi sağlayan yapıdır. Ana URL'nin aldığı parametreler, Spark'ın çalışma kipine göre değişmektedir. Bu parametrelerden birkaç tanesine Çizelge 3.5'de yer verilmiştir.

Çizelge 3.5. Spark çalışma ortamları

Ana URL Parametresi	**Kip Durumu**
local	Yerel Kip
local[K]	
local[*]	
….	…
mesos://HOST:PORT	Mesos Kip

Spark Session, yerel ortamda, "seri" olarak "local" parametresi ile "paralel" olarak ise "local[*]" parametresi ile işleme alınabilir (Şekil 3.5, Şekil 3.6). Bu çalışmada, "local" ve "local[*]" durumları üzerinde çalışılmıştır.

local: Spark Session, "local" olarak başlatılır. Paralellik olmadan tek bir iş parçacığı ile Spark makinesine bağlanılıp tüm işlemler gerçekleştirilir.

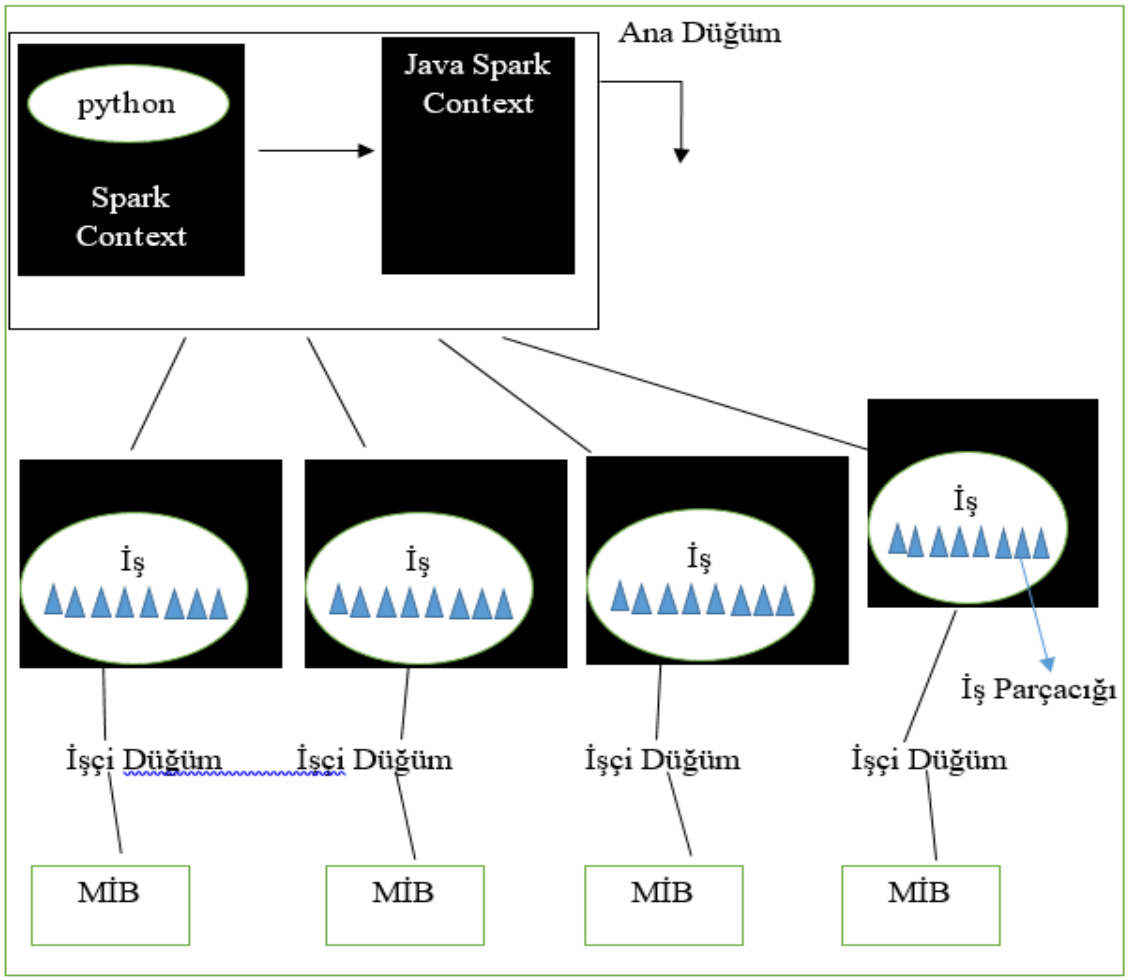


Şekil 3.5. Spark “local” çalışma yapısı

local[*]: Spark Session, “local[*]” olarak başlatılır. Paralellik sağlanarak birden fazla iş parçacığı ile tüm işlemler gerçekleştirilir. Bir işin eş zamanlı olarak işlenen her bir parçası, paralel olarak dağıtılır. Paralelleştirme işlemi yürütücü için kullanılan sürücüde yapılır. Bu ortamda çalışan iş parçacığı sayısı, bilgisayarın mantıksal çekirdek sayısı kadar olabilmektedir (Penchikala 2018).

local[*] durumu ile Spark konfigürasyon ayarları yapılıp, tüm işlemler gerçekleştirildiğinde, zaman açısından kazanç sağlandığı görülmüştür. Görev paralelliği sağlanıp iş dağılımı yapıldığı için, performans artışı gözlemlenmiştir. Apache Spark, programlama dili seviyesinde Python’a olan API’si ile “local[*]” ortamda paralel hesaplama için geliştirilen birçok yazılım sistemlerinden biri olarak gösterilebilir. Apache Spark kendi mimarisinde bulunan sistem sayesinde, programın parçalara bölünüp, işlemcilere atanmasını sağlayan mekanizmayı bünyesinde barındırır. Bu mekanizma, dolaylı paralellik yapmaktadır. Programcının, problemi nasıl parçalara bölmesi gerektiğini makinaya bildirmesine gerek kalmadan, Spark’ın kendi bünyesindeki yazılımı sayesinde paralellik sağlanır. Bu paralellikte yük dengeleme yapılarak, az görev yükü bulunan işlemcideki işlere, fazla görev yükü bulunan işlemcideki işler tahsis edilerek görev dağılımı dengelenir. Çalışma esnasında, kullanılan çekirdeklerin aynı yoğunlukta işlem yapıp, verimli sonuç vermeleri sağlanır. Her bir çekirdek ortak hafızayı kullanarak, eş zamanlı işlemleri gerçekleştirir.

Çalışma süresince, local[*]’ın sağladığı avantajın görülmesi sebebi ile, local[*] ile Spark Session başlatılıp, tüm analizler yapılmıştır. Bu çalışmada, üzerinde çalışılan bilgisayarın, çekirdek sayısı ve mantıksal çekirdek sayısı incelendiğinde, 4 çekirdek ve 8 mantıksal çekirdeğe sahip olduğu görülmektedir. O zaman, çalışma süresince 8 tane iş parçacığı çalışabilmektedir.



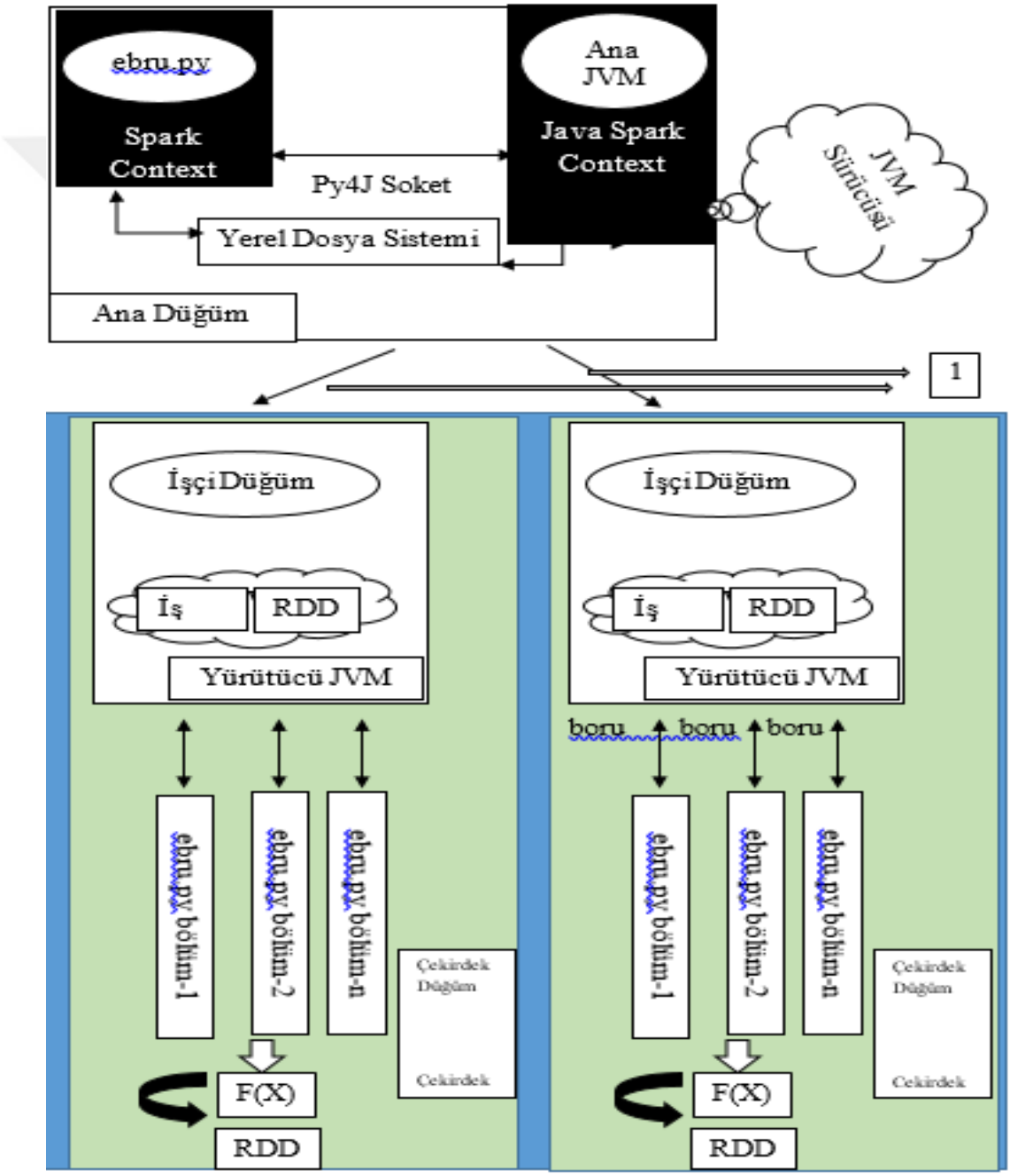
Şekil 3.6. Spark "local[*]" çalışma yapısı

3.2.2. Apache spark-python

Apache Spark; Java, Scala, R, Python Spark Python Uygulama Programlama Arayüzü ile geliştirilebilir. Her bir programlama dili için, API'si mevcuttur. Bu sayede, standart bir arayüz ile Spark uygulamalarının geliştirilmesi sağlanmış olur. Python, kütüphane zenginliğinden dolayı çokça tercih edilen dildir. Bu çalışmada, Apache Spark ve python kullanarak büyük veri analiz edilmiştir.

Çalışma içerisinde, Python kullanılırken, temel kütüphanelerden olan "scikit learn", "pandas" gibi kütüphaneler kullanılmamıştır. Verinin büyüklüğü arttıkça bu

kütüphanelerin ihtiyacı karşılayamadığı görülmüştür. Dolayısıyla, Spark ile entegre işlemi gerçekleştirilip, "pyspark (Spark Python Uygulama Programlama Arayüzü)" isimli kütüphaneden gelen, "Spark Python API" ile çalışılmıştır. PySpark, Spark'ın Java API'si üzerine kuruludur. Python sürücü programında, Spark Context bir Java Spark Context oluşturmak için Py4J kullanır. PySpark iç yapısı (mimarisi) Şekil 3.7'de gösterilmiştir.



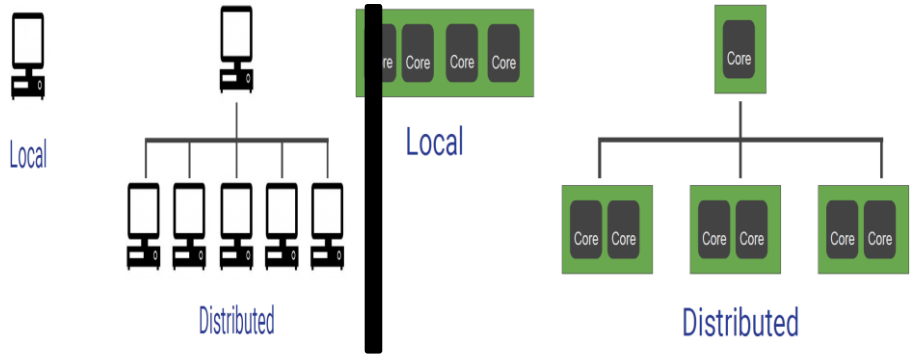
Şekil 3.7. pySpark iç yapısı (Rosen 2016)

Bu mimari incelendiğinde, pySpark'ın python yorumlayıcısı çalışmaya başladığında, Py4J soketi aracılığıyla bir JVM çalışmaya başlar. JVM gerçek Spark sürücüsü olarak çalışmaya başlayarak, Spark'ı işletim sistemine tanıtır. Spark, işçi makinesindeki yürütücüler ile iletişim kurmak için, "Java Spark Context" yükler. Python API çağrıları Spark Context nesnesine, Java API çağrıları ise Java Spark Context nesnesine dönüştürülür. Örneğin; bir pySpark kodunda "spark. read.csv()" implemantasyonu yapılırken, Java Spark Context'in "spark.read.csv()" methoduna bir çağrı gönderilir. Daha sonra, Java Spark Context, Spark yürütücüsünün JVM'leri ile iletişim kurar. Gerekli dosyaların yüklemesi yapılır. Dosya yüklenirken veri, Java string nesnesi olarak yüklenir. Bunun sebebi, RDD ile ilişkili verilerin aslında Spark JVM'nin içerisinde Java nesnesi olarak bulunmasıdır.

Spark'ın yerel ortamında çalışan, pysparkda yer alan bir python RDD nesnesi, yerel JVM'deki python RDD nesnesine karşılık gelir. Python RDD'de yer alan API çağrıldığında, PiCloud (Basitleştirilmiş Bulut Bilişim) tarafından geliştirilen ve cloud paketinden gelen, "cloudpick" modülü ile kod serileştirilir ve yürütücülere paralel olarak dağıtılır. Bu işlem Şekil 3.7'de 1 numara ile gösterilen yerde gerçekleşmektedir. Daha sonra, veri Java nesnelerinden Python ile uyumlu nesnelere dönüştürülür. Bir boru aracılığı ile veri alışverişinin gerçekleştiği yürütücü ile iletişimi olan, python yorumlayıcılarına aktarılır. Bu yorumlayıcılarda herhangi bir python işlemi yürütülür. Elde edilen sonuçlar, JVM içerisinde bir RDD nesnesi olarak depolanır.

3.3. Bilgisayarda Analiz Edilecek Verinin Tutulma Şekli

Bilgisayarın hafızasına bağlı olarak, veriler yerel bilgisayarda tutulabilir. Hafızanın karşılayamayacağı büyüklükteki veriler, birden fazla makinalara dağıtılabilir. Dağıtılmış veriler, yerel sisteme göre karışıktır. Bu iki sistemin karşılaştırılması Şekil 3.8'de verilmiştir.

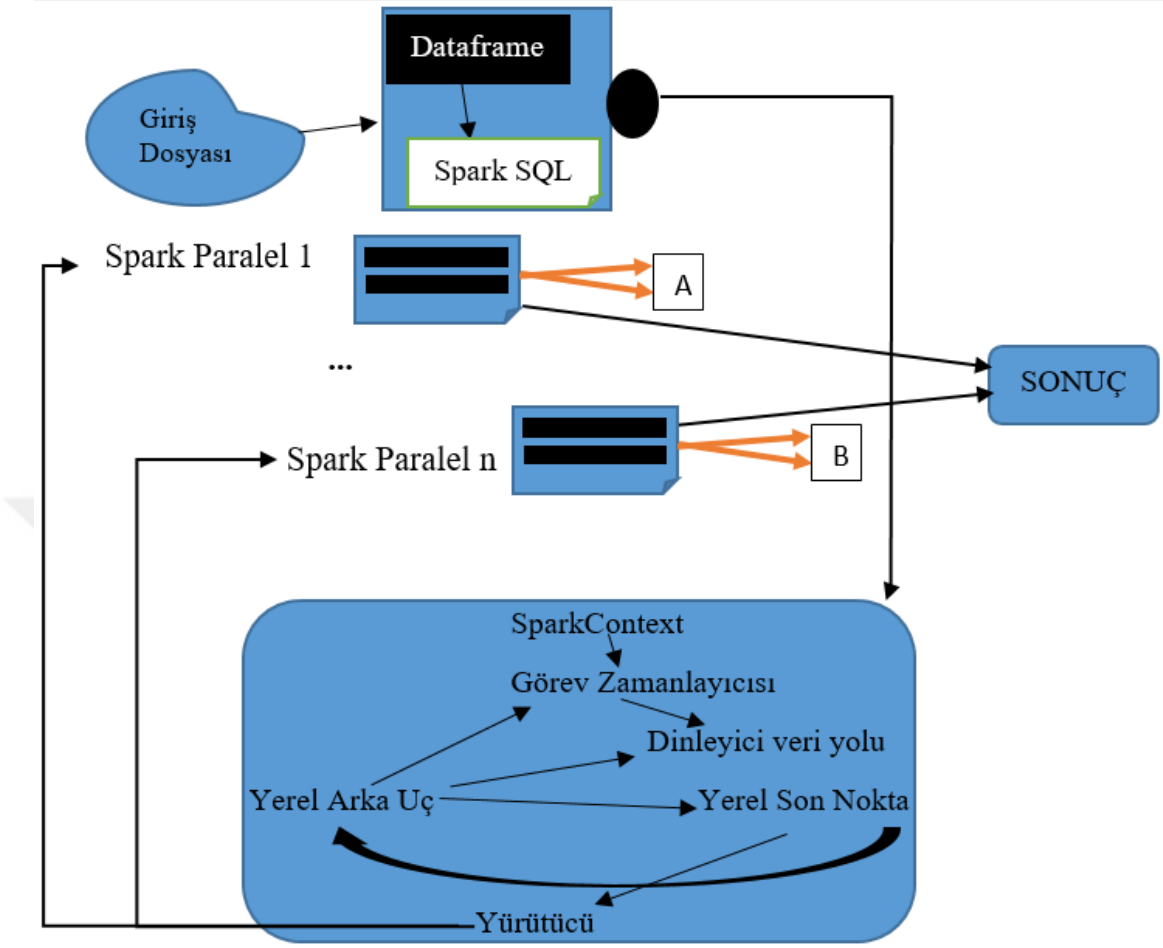


Şekil 3.8. Yerel sistem ve dağıtılmış sistem (Portilla Anonymous)

Spark, bellek içi dağıtık işlem yapabilen bir sistem olduğu için, bu çalışma kapsamındaki veriler, yerel ortamda, bilgisayarın hafızasından yararlanarak IMDB içerisinde saklanmıştır. Spark bu veri tabanı üzerinde çalışarak, analiz işlemlerini gerçekleştirmiştir.

Spark yerel ortamda çalışırken, tüm yürütme bileşenlerini tek bir JVM'de toplar. Varsayılan paralelleştirme işlemi, Master URL içerisinde özelleştirilmiş iş parçacığı sayısı kadar olmaktadır. Daha önce de değinildiği gibi, kullanılan iş parçacığı sayısına göre yerel ortam, "local" ya da "local[*]" olarak ele alınabilmektedir.

Dosya kaynağının yerel ortamda Spark'a verilme şekli Şekil 3.9'da ayrıntılı olarak verilmiştir. Bu mimari "yalancı (sahte) küme" olarak tanımlanmaktadır.

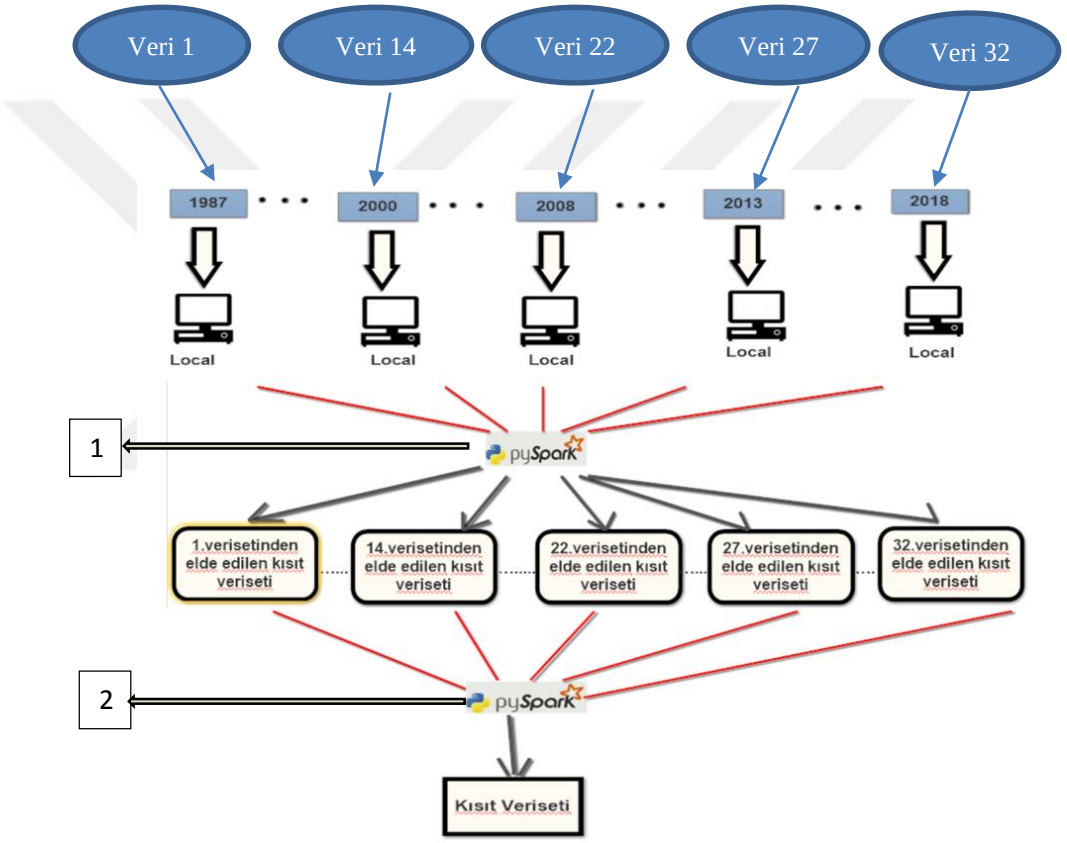


Şekil 3.9. Spark yerel kip çekirdek yazılım kısmı (Laskowski Anonymous)

Şekil 3.9, ayrıntılı incelenecek olursa, Spark Context ile Spark'a giriş noktası oluşturulur. Kip durumu belirtilir. Gelen işler kabul edilerek, görev zamanlayıcısına gönderilir. Buradan yapılan çıkarıma göre, yeni bir durum alındığında ya da mevcut durum güncellendiğinde, görev zamanlayıcısı çalışmaya başlar. Dinleyici veri yolu ile ortam dinlenilerek sırada bekleyen yeni bir iş olup olmadığı kontrol edilir. Burada yer alan, yerel arka uç, Spark'ın arka planda yazılım kısmı ile gerçekleşen işlemlerinden oluşmaktadır. "Yerel arka uç" ve "görev zamanlayıcısı" işlemlerinin birbiri içerindeki akış şeması "yerel zamanlayıcı arka uç" yapısını ortaya çıkarmaktadır. "Yerel zamanlayıcı arka uç", küme yöneticisi gibi çalışır. İşçilere kaynakların (işlerin) nasıl dağıtılacağını belirler. Yerel son nokta ise, işlemlerin sonlandığı kısımdır. Bu noktadan sonra, yürütücülere dağıtılması belirlenen işler gönderilir. Şekil 3.9'da A ve B noktalarında görüldüğü gibi, kullanıcın belirlediği kip durumuna göre, her bir iş kendi içinde paralelleştirilerek iş

parçacıklarına ayrılır. Her bir iş, kendisi için belirlenen çekirdek üzerinde çalışarak sonucunu verir.

Bu tez kapsamında, tek makina üzerinde Spark 2.0.2 kurulumu gerçekleştirilerek, BTS verisi yerel bilgisayarda tutulmuştur. Şekil 3.10'da gösterildiği gibi veriler "birleştirme yapısı" kullanılarak analiz edilmiş ve kısıt veri seti oluşturulmuştur.



Şekil 3.10. Analiz için kullanılan birleştirme yapısı

Bu kısıt veri seti oluşturulurken, Şekil 3.10'da 1 numara ile gösterilen işlemin, pySpark sahte kodu Şekil 3.11'de verilmiştir.

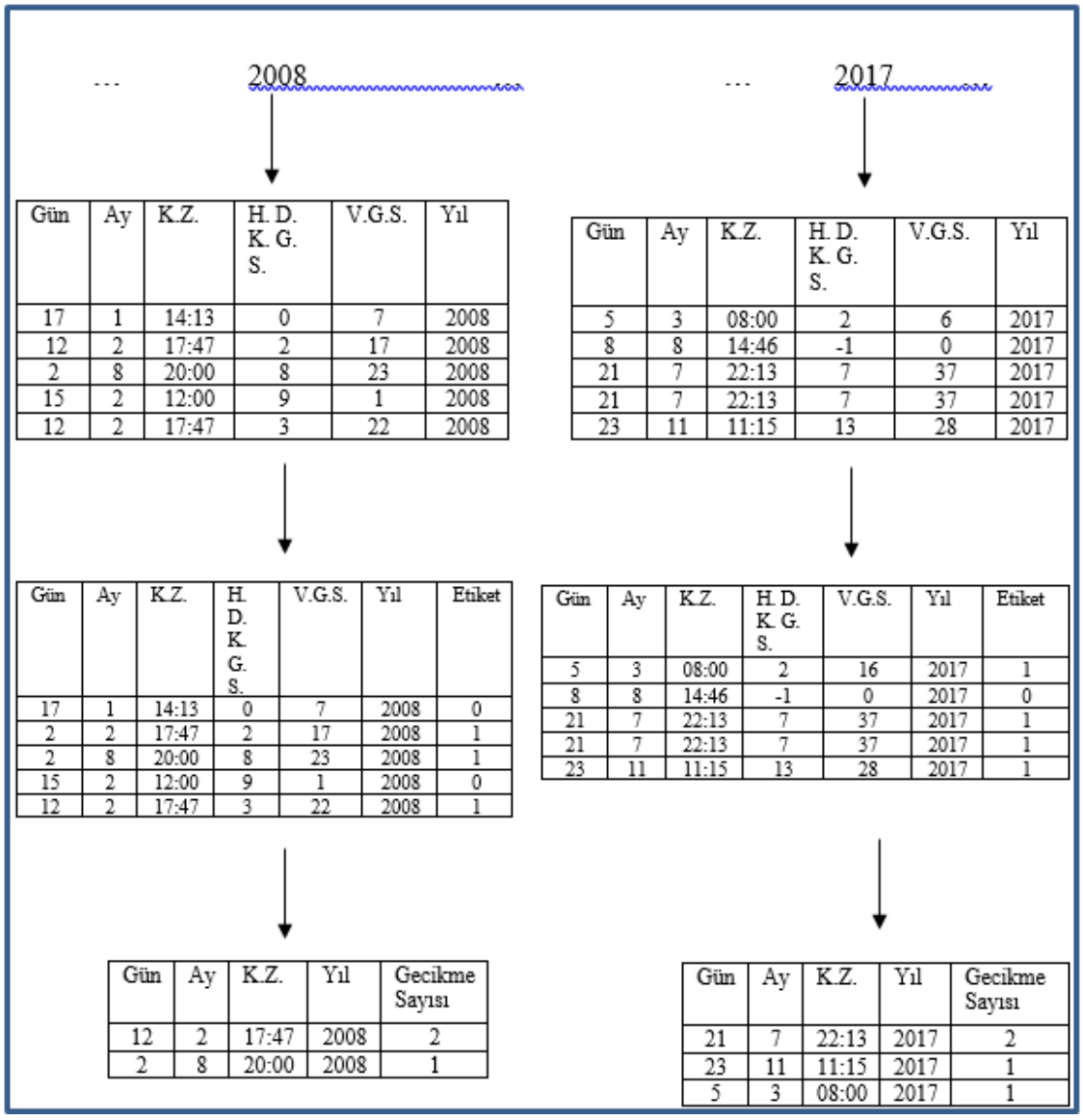
```
1 Başla
2 import(findspark, pyspark)
3 spark=new SparkSession("local[*]")
4 veriseti=spark.read("dosya adı.csv")
5 veriseti=int("Varış Gecikmesi", "Hava Durumu
Gecikmesi", "Kalkış Zamanı")
6 veriseti=veriseti.filter(Varış Gecikmesi & Hava
Durumu Gecikmesi !=NA)
7 if(veriseti.Varış Gecikmesi>15)
8     gecikme etiketi=1
9 end if
10 if(veriseti.Varış Gecikmesi<=15)
11     gecikme etiketi=0
12 end if
13 if(veriseti.Hava Durumu Gecikmesi>0)
14     hava durumu kaynaklı gecikme etiketi=1
15 else
16     hava durumu kaynaklı gecikme etiketi=0
17 end if
18 veriseti=if(hava durumu kaynaklı gecikme etiketi=1
&&  gecikme etiketi=1)
19 veriseti=veriseti.groupBy(gün,ay,kalkış
saati,yıl).count()
```

Şekil 3.11. pySpark sahte kodu

Bu sahte kod incelendiğinde, 2 nolu satırda "pyspark" etkileşimli kullanımı için gerekli olan "findspark, pyspark" modülleri yüklenir. Ayrıca "pyspark" kütüphanesi içerisinden, koşullu sütun ekleyebilmek için gerekli modüller yüklenmiştir. 3 nolu satırda Spark'ın dataframe tabanlı kütüphanesi kullanılmıştır. Dataframe ile işlemler yapabilmek için "Spark Session" başlatılmıştır. SparkSession "builder" yapısını kullanarak, birden fazla iş parçacığının paralel olarak çalışmasını sağlayan, yerel kipdeki "local[*]" ortam ile "spark" ismindeki oturum açılmıştır. Apache Spark başlatılmıştır. 4 nolu satırda dosya okunarak, Spark'a giriş noktası oluşturulmuştur. Okunan dosya ile veri çerçevesi oluşturulmuştur. Bu veri çerçevesi üzerinde, SparkSQL kütüphanesi ile işlemler

yapabilmek için gerekli ortam hazırlanmaya başlanmıştır. 5 ve 6 nolu satırlarda ön işleme yapılmıştır. “string” veri tipinde olan “Varış Gecikmesi”, “Hava Durumu Gecikmesi”, “Kalkış Zamanı” üzerinde ön işleme yapılarak “integer” veri tipine çevrilmiştir. Filtreleme işlemi yapılarak hava durumundan kaynaklı gecikme durumunun ve varış gecikmesinin belirtildiği kayıtlar dikkate alınmıştır. Eğer bir uçuşun varış gecikme süresi 15 dk süreden fazla zaman dilimini kapsamışsa, ilgili uçuşa “gecikme etiketi” olarak “1” verilmiştir. Aksi bir durumda ise, gecikme etiketi “0” olarak verilmiştir (7,8,9,10,11,12 nolu satırlar). Eğer bir uçuş hava durumundan kaynaklı gecikme yapmışsa yani “0” dan farklı dakikada gecikme durumu belirtilmişse, ilgili uçuşa “hava durumu kaynaklı gecikme etiketi” olarak “1” verilmiştir. Aksi durumda ise “0” verilmiştir (13,14,15,6,17 nolu satırlar). 18 nolu satırda, hava durumundan kaynaklı olarak gecikmeli varış yapan tüm uçuşlar biraraya toplanmıştır. Bu uçuşlar “gün, ay, saat, yıl, gecikme sayısı” olarak gruplandırılmıştır (19 nolu satır).

Şekil 3.10’da birleştirme yapısındaki, 1 nolu işlem, Şekil 3.12’de verilen örnek üzerinde özetlenmiştir. BTS verisindeki, “Gün, Ay, K.Z. (Kalkış Zamanı), H.D.K.G.S. (Hava Durumundan Kaynaklı Gecikme Süresi), V.G.S. (Varış Gecikme Süresi), Yıl” dikkate alınmıştır.



Gün	Ay	K.Z.	H. D. K. G. S.	V.G.S.	Yıl
17	1	14:13	0	7	2008
12	2	17:47	2	17	2008
2	8	20:00	8	23	2008
15	2	12:00	9	1	2008
12	2	17:47	3	22	2008

Gün	Ay	K.Z.	H. D. K. G. S.	V.G.S.	Yıl
5	3	08:00	2	6	2017
8	8	14:46	-1	0	2017
21	7	22:13	7	37	2017
21	7	22:13	7	37	2017
23	11	11:15	13	28	2017

Gün	Ay	K.Z.	H. D. K. G. S.	V.G.S.	Yıl	Etiket
17	1	14:13	0	7	2008	0
2	2	17:47	2	17	2008	1
2	8	20:00	8	23	2008	1
15	2	12:00	9	1	2008	0
12	2	17:47	3	22	2008	1

Gün	Ay	K.Z.	H. D. K. G. S.	V.G.S.	Yıl	Etiket
5	3	08:00	2	16	2017	1
8	8	14:46	-1	0	2017	0
21	7	22:13	7	37	2017	1
21	7	22:13	7	37	2017	1
23	11	11:15	13	28	2017	1

Gün	Ay	K.Z.	Yıl	Gecikme Sayısı
12	2	17:47	2008	2
2	8	20:00	2008	1

Gün	Ay	K.Z.	Yıl	Gecikme Sayısı
21	7	22:13	2017	2
23	11	11:15	2017	1
5	3	08:00	2017	1

Şekil 3.12. 1 nolu işlemin özeti

Bu kısıt veri seti oluşturulurken, Şekil 3.10'da 2 numara ile gösterilen işlemin, pySpark sahte kodu Şekil 3.13'de verilmiştir.

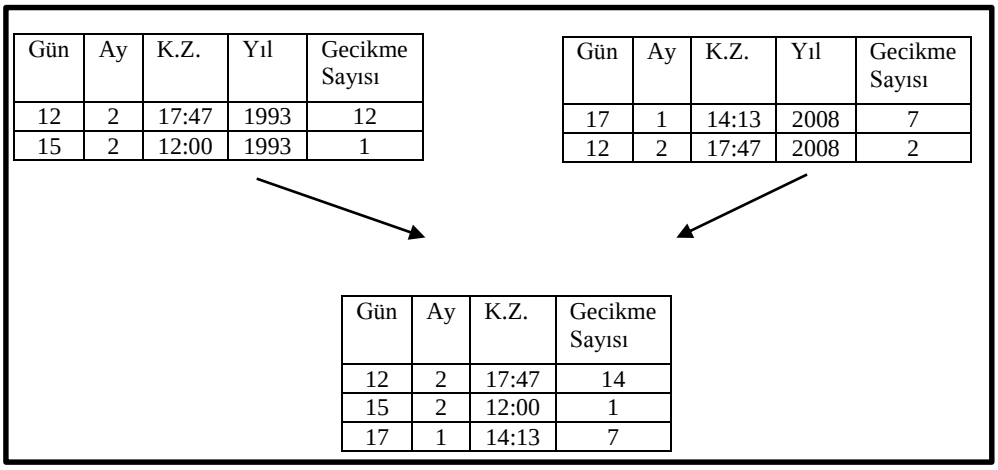
```
1 Başla
2  kısıtveriseti=veriseti.groupBy(gün, ay, kalkış
saati).sum("gecikme sayısı")
```

Şekil 3.13. 2. pySpark sahte kodu

1 nolu yalancı koddan elde edilen, 32 veri setinden çıkarımı yapılan, her bir kısıt veri seti tek bir dosyada birleştirilmiştir. Şekil 3.10'da yer alan 2 numaralı işlem için belirtilen sahte kod uygulanmıştır. Şekil 3.13'de yer alan sahte kod incelendiğinde; 2 nolu satırda 32 adet veri setinin kısıtlarını içeren dosya gün, ay, saat olarak gruplandırılmıştır. Gecikme sayısı sütunundaki tüm elemanlar toplanıp net kısıt veri seti elde edilmiştir.

İşlemlerde, dataframe API kullanılarak, pySpark dataframe nesnesi oluşturulmuştur. SparkSQL kütüphanesi kullanılarak SQL tabanlı analizler yapılmıştır.

Şekil 3.10'da birleştirme yapısındaki, 2 nolu işlem, Şekil 3.14'de verilen örnek üzerinden özetlenmiştir:



Şekil 3.14. 2 nolu işlemin özeti

Birleştirme olarak yukarıda bahsi geçen yapının, 4 ve 32 işlemcili bilgisayarlarda verdiği sonuçların aynı olduğu bizzat test edilip görülmüştür. Buradan çıkarılan sonuç, birleştirme çalışma şemasına göre, herhangi bir veri kaybı yaşanmadığıdır. Elde edilen sonuçta, 353 472 kısıt bulunmuştur. Bu kısıt veri setine ait kısa bir kesit Çizelge 3.6'da gösterilmiştir.

Çizelge 3.6. Kısıt veri seti

353 472 kısıt

Gün	Ay	K.Z.	Gecikme Sayısı
26	7	22:45	6
…	…	…	…
22	12	14:12	9
…	…	…	…
5	8	20:20	15
…	…	…	…

3.4. Model

Problemin yapısına göre, her bir problem matematiksel model olarak ifade edilebilir, bu model algoritma içerisinde uygunluk formülü olarak belirtilir.

3.4.1. İndeksler

i: Kalkış Yeri
j: Varış Yeri

$i,j \in \{YVR, SFO, JFK\}$

k: (G.N. (Gün No)) U (A.N. (Ay No)) U (K.Z.) da gerçekleşen uçuş
w: Uçak Tipi
$w \in \{1,2\}$

3.4.2. Parametreler

1) C_{kw}: k. uçuşta w tipi uçak ile uçuş gerçekleştirildiğinde, uçuş saati başına düşen maliyeti temsil etmektedir (Çizelge 3.7).

$$C_{kw} = \{ C_{k1} \mid C_{k2} \}$$

Çizelge 3.7. Uçuş tipi ve maliyet bilgileri (Anonymous 2015)

C_{kw}	C_{k1}	C_{k2}
Uçak Tipi	Boeing 787	Airbus A-320
Uçuş saati başına düşen maliyet	$5,303	$2,845

2) A_{ij}: i kalkış yeri ile j varış yeri arasında saat cinsinden oluşan mesafeyi temsil etmektedir (Çizelge 3.8).

$$A_{ij} = \{ A_{12} \mid A_{21} \mid A_{13} \mid A_{31} \mid A_{23} \mid A_{32} \}$$

Çizelge 3.8. Havaalanları arasındaki mesafe bilgileri (Anonymous 2015)

$A_{12} = A_{21}$ = YVR ile SFO arasında saat cinsinden oluşan mesafe = 2h25m
$A_{13} = A_{31}$ = YVR ile JFK arasında saat cinsinden oluşan mesafe = 5h25m
$A_{23} = A_{32}$ = SFO ile JFK arasında saat cinsinden oluşan mesafe = 6h00m

3) σ: Her bir k uçuşu için, kısıta verilen ağırlık değerini temsil etmektedir.

Uçuş çizelgelemesi yapılırken, optimum sonuca ulaşmak için programa bazı kısıtlamalar verilmelidir. Böylece algoritma, arama uzayında optimum sonucu bulmaya çalışırken, bu kısıtlamalara göre çalışmasına yön verir. Kısıtlar, sert kısıt ve yumuşak kısıt olabilir. Bulundukları kısıt çeşidine göre kısıtlara ağırlık değeri verilir. Bu ağırlık değeri, amaç fonksiyonunda kullanılır. Eğer program içerisindeki çözüm sert kısıt ya da yumuşak kısıt değilse, σ değeri 0,1 olarak alınmıştır. Eğer sert kısıt ise, ağırlık değeri artırılmış; yumuşak kısıt ise ağırlık değeri azaltılmıştır. Çalışma içerisinde yer alan sert ve yumuşak kısıtlar tanımlanacak olursa;

Sert Kısıtlar:

1. 23 den fazla uçuş gecikmesine tahammül yoktur, geçmiş uçuş verilerinden elde edilen kısıt veri setinde, bu şartı sağlayan tarife bilgisi varsa, algoritma içerisinde σ değeri 1 olarak alınmıştır.
2. Kalkış zamanı 01:00 ile 03:00 arasında değişen ve bu zaman aralığında yaşanan uçuş gecikmelerine tahammül yoktur. Eğer bu şart altında gecikme yaşandıysa, algoritma içerisinde σ değeri 1 olarak alınmıştır.

Yumuşak Kısıtlar:

1. Geçmiş uçuş verilerinden elde edilen kısıt veri setinde, 10 ile 23 arasında uçuş gecikmesinin yaşandığı tarife bilgisi var ise, algoritma içerisinde σ değeri 0,50 olarak alınmıştır.
2. Geçmiş uçuş verilerinden elde edilen kısıt veri setinde, 9 dan az uçuş gecikmesinin yaşandığı tarife bilgisi var ise, algoritma içerisinde σ değeri 0,25 olarak alınmıştır.

4) h(V.G.S. (Varış Gecikme Süresi) & H.D.K.G.S.| G.N. U A.N. U K.Z.) : G.N., A.N. ve K.Z. da hava durumundan kaynaklı oluşan gecikme durumunu temsil etmektedir.

3.4.3. Karar değişkenleri

$$X_{ijk_{G.N.\ U\ A.N.\ U\ K.Z.}} : \begin{cases} 1 \text{ , gecikme varsa} \\ 0 \text{ , gecikme yoksa} \end{cases}$$

$\varphi\left(X_{ijk_{G.\ N.\ U\ A.N.\ U\ K.Z.}}\right)$: i kalkış yerinden j varış yerine G.N., A.N. ve K.Z. da k uçuşunda gerçekleşen gecikme sayısıdır.

3.4.4. Amaç fonksiyonu

Amaç fonksiyonu sezgisel algoritmaların beynini oluşturur. Çözümün kalitesini gösteren fonksiyondur. Bu çalışmada, amaç fonksiyon, minimum maliyeti bulmak üzere özelleştirilecektir. Sezgisel algoritmaların başarısı bu fonksiyonun verimli, hassas olmasına bağlıdır.

Amaç Fonksiyonu =

$$\sum_{i,j,k} ((\varphi(X_{ijk\ G.N.\ U\ A.N.\ U\ K.Z.}) \mid h(V.G.S.\ \&\ H.D.K.G.S. \mid G.\ N.\ U\ A.N.\ U\ K.Z.))$$

$$* A_{ij} * C_{kw} * X_{ijk_{G.N.\ U\ A.N.\ U\ K.Z.}}) * \sigma$$

3.5. Optimizasyon İçin Kullanılan Algoritmalar

Bu çalışmada, minimum maliyette uçuş tarifesi oluşturmak için 3 tane sezgisel algoritma kullanılmıştır: Yapay arı koloni algoritması, genetik algoritma, benzetilmiş tavlama.

3.5.1. Yapay arı koloni algoritması

YAKA (Yapay Arı Koloni Algoritması), Derviş Karaboğa tarafından geliştirilmiş, arıların besin ararken kendilerine özgü içgüdüsel olarak kullandıkları zeki davranışların modellenerek oluşturulduğu popülasyon tabanlı sezgisel optimizasyon algoritmasıdır (Karaboga and Akay 2007). YAKA'da 3 tip arı vardır: işçi arı, gözcü arı, kâşif arı. Kâşif arılar rastgele besin kaynağı bulurlar. İşçi arılar besin kaynağına gidip işledikten sonra, besin kaynağını diğer arılarla paylaşmak için kovana geri gelir. Kovanda yer alan dans alanında yaptıkları dans ile gözcü arılara besinin yeri hakkında gerekli bilgiyi verir. Gözcü arılar güneş ışığından faydalanarak işçi arıların verdiği konum bilgisine göre uygun açı ile besine ulaşıp, besin üzerinde gerekli işlemeyi yapar. Eğer besin kaynakları üzerinde daha fazla besin (nektar) üretilemiyorsa, (başarısızlık durum sayısı > limit) görevli arı kâşif arı olur. Yeni rastgele besin kaynağı üretilir. Arılar bu kaynaklara yönlendirilerek işlem sırası en baştan tekrar yapılır (iterasyon sayısı kadar).

Bir besin kaynağındaki nektar miktarı o kaynağın ne kadar kaliteli olduğunu belirtir. O besinin ne kadar kaliteli olduğu ise algoritmada uygunluk değeri hesaplanarak bulunur. YAKA en fazla nektara sahip besin kaynağının yerini bulmaya çalışarak arama uzayındaki çözümlerden problemin minimumunu ya da maksimumunu veren noktayı bulmaya çalışır (Karaboga and Akay 2009). Bu çalışmada, oluşabilecek gecikmeleri önleyerek, minimum maliyette uçuş tarifesi oluşturulmaya çalışıldığı için, arama uzayındaki çözümlerden problemin minimum noktası bulunmaya çalışılmıştır.

YAKA sahte kodu Şekil 3.15'de verilmiştir.

1 Başla
2 başlangıç değerlerinin atanması (populasyon boyutu, iterasyon sayısı, limit, başarısızlık durum sayısı)
3 iterasyon sayısı =1
4 repeat
5 rastgele çözüm kümeleri oluştur
6 x_i çözüm kümesinin iyileştirme var mı kontrolü
7 uygunluk değeri hesapla, açgözlü yaklaşımla daha iyi olanı seç
8 işçi arılar için her bir x_i çözüm kümesi için geliştirme formülü uygula
9 uygunluk değeri hesapla, açgözlü yaklaşımla daha iyi olanı seç
10 x_i çözüm kümesi gelişmemişse; başarısızlık durum sayısı ++, gelişmişse başarısızlık durum sayısı =0
11 gözcü arıların kullanacakları olasılık değeri o_i hesapla, [0-1] arasında rastgele sayı üret
12 if rastgele sayı < o_i then
13 her bir x_i çözüm kümesi için geliştirme formülü uygula
14 uygunluk değeri hesapla, açgözlü yaklaşımla daha iyi olanı seç
15 x_i çözüm kümesi gelişmemişse; başarısızlık durum sayısı ++, gelişmişse başarısızlık durum sayısı =0
16 end if
17 if başarısızlık durum sayısı > limit then
18 adım 5'e gidilir
19 iterasyon sayısı ++
20 end if
21 en iyi çözümü hafızada tut
22 iterasyon sayısı ++
23 Until iterasyon sayısı

Şekil 3.15. YAKA sahte kodu

Şekil 3.15'de yer alan sahte kod incelendiğinde, 2 nolu satırda, besin sayısı kadar popülasyon boyutu, algoritmanın adımlarının kaç kere çalıştırılacağını belirten iterasyon sayısı, nektar miktarının ne zaman tükendiğinin sınırını belirleyecek olan limit değeri ve daha iyi bir besin kaynağı bulunamadığında başarısızlık durumunu betimleyecek olan değişken tanımlamaları ve ilk değer atamaları yapılmıştır. 3 ve 4 nolu satırlarda, ilk iterasyon ile işlemler yapılmaya başlanmıştır. 5 nolu satırda, kâşif arıların tespit ettiği besin kaynakları yani problemin başlangıç çözüm kümeleri rastgele oluşturulmuştur. 6 ve 7 nolu satırlarda, her bir çözüm kümesinde iyileştirme yapılıp yapılamayacağı, uygunluk

fonksiyonundaki ölçülen değeri ile karşılaştırılır. Eğer iyileştirme varsa, mevcut çözümün iyileştirme formülündeki sonuç değerinin uygunluk fonsiyonundaki değeri, eski uygunluk değerinden daha uygun ise (daha iyi bir besin kaynağı) mevcut durum güncellenmiştir. Aksi durumda, mevcut durum muhafaza edilmiştir. 8 ve 9 nolu satırlarda, işçi arıların fazı başlamıştır. Her bir işçi arı, kâşif arıların tespit ettiği besinlere geliştirme formülü uygulayarak daha iyi besin kaynağı yani çözüm olup olmadığını, uygunluk fonksiyonundaki karşılaştırma ile kontrol eder. Eğer daha iyi çözüm yoksa, başarısızlık durumu 1 artırılır. Aksi durumda başarısızlık durumu sıfırlanır (10 numaralı satır). 11 ve 12 ile numaralandırılan satırlarda, gözcü arı safhası başlamıştır. Gözcü arılar, "Rulet Çemberi" yöntemini kullanarak bir olasılık değeri belirler. Bu yöntemin temel mantığında, uygunluk değeri yüksek olan besinlerin seçilme ihtimali daha yüksek olduğu için bu oran ile 0 ve 1 arasında rastgele üretilen sayı arasında karşılaştırma yapılır. Eğer olasılık değeri daha büyükse, her bir çözüm için geliştirme formülü uygulanarak daha iyi besin kaynağı yani çözüm olup olmadığı, uygunluk fonksiyonundaki karşılaştırma ile kontrol edilir. Eğer daha iyi çözüm yoksa, başarısızlık durumu 1 artırılır. Aksi durumda, başarısızlık durumu sıfırlanır (13,14,15,16 numaralı satırlar). Başarısızlık durumu, limit değeri ile mukayese edilerek besin kaynağından nektar üretilip üretilemeyeceği kontrol edilir. Eğer üretilemiyorsa iterasyon sayısı 1 artırılır ve yeni besin kaynakları oluşturulur (17,18,19,20 numaralı satırlar). Yapılan aşamalar sonucunda ve gerekli görüldüğünde yapılan güncellemelerle, en iyi çözüm hafızada tutularak, iterasyon sayısı 1 artırılıp iterasyon sayısı tamamlanıncaya kadar program çalıştırılmaya devam etmiştir (21,22,23 numaralı satırlar).

YAKA önemli algoritma adımları:

1) Kâşif arı aşaması - besin kaynaklarının bulunması

Başlangıç besin kaynakları bulunurken, kâşif arılar rastgele besin kaynağı ararlar. Algoritmada, arıların yaptığı bu eyleme karşılık gelen işlem, rastgele çözüm kümesi oluşturmaktır. Her bir çözüm kümesi, parametre sayısı kadar eleman içerir ve her eleman

minimum ve maksimum değer aralığında rastgele oluşturulan değere göre x_{lm} besin değerini alır (Karaboga and Akay 2009).

YAKA çözüm kümesinde yer alan her bir parametreye değer atama formülü, Formül 1'de verilmiştir.

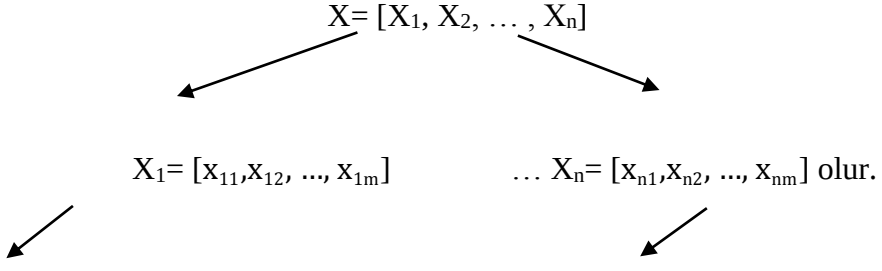
$$x_{lm}=\text{pmin}+\text{rand}(0,1)(\text{pmax-pmin})$$

Formül 1 Değer atama formülü

x_{lm}: Seçilen çözüm kümesinin geçerli parametre numarası
l: Mevcut çözüm kümesinin index numarası
m: l. Seçilen çözüm kümesinin index numarası
pmin: Parametrenin alacağı minimum değer
pmax: Parametrenin alacağı maksimum değer
n tane rastgele çözüm kümesi oluşturulmuş olsun (n tane besin kaynağı olsun.).

Parametre sayısı ise m olsun (her çözüm kümesinin m tane parametresi olsun).

O zaman, çözüm kümesi, X_1= 1. çözüm kümesi, X_2= 2. çözüm kümesi, **… ,** X_n= n. çözüm kümesi olur. Formül 2'de işlem özetlenmiştir:



Formül 2

Bu çalışma kapsamında, rastgele oluşturulan çözüm kümelerinin mantığı, t.v.s'den (tarife veri seti) alınan uçuş bilgileri ile rastgele uçuşlardan oluşan çözüm kümesinin oluşturulması şeklindedir (Çizelge 3.9).

Çizelge 3.9. YAKA çözüm kümesi oluşturma

n = popülasyon boyutu rastgele indis = 1 + rand() % (tarife veri seti kayıt sayısı-1) tarife veri seti kayıt sayısı=380 rastgele indis = 1 + rand() %(379)			
Uçuş[0]	t.v.s[rastgele indis][0]	t.v.s[rastgele indis][1]	t.v.s[rastgele indis][3]
Uçuş[1]	t.v.s[rastgele indis][0]	t.v.s[rastgele indis][1]	t.v.s[rastgele indis][3]
…	…	…	…
Uçuş[n-1]	t.v.s[rastgele indis][0]	t.v.s[rastgele indis][1]	t.v.s[rastgele indis][3]
Uçuş[n]	t.v.s[rastgele indis][0]	t.v.s[rastgele indis][1]	t.v.s[rastgele indis][3]

Rastgele oluşturulan çözüm kümesine Çizelge 3.10'da örnek verilmiştir.

Çizelge 3.10. YAKA örnek çözüm kümesi

Gün	Ay	Kalkış Saati
13	12	19:07

2) İyileştirme formülü

Kâşif arıların tespit ettiği besinlerde iyileştirme var mı diye kontrol yapılabilmesi için, öncelikle çözüm kümesine iyileştirme formülü uygulanır. Bu formül problemin yapısına

göre istenilen şekilde belirlenebilir. Aşağıda program içerisinde kullanılan iyileştirme formülünün temsili anlatımı verilmiştir (Şekil 3.16, Şekil 3.17):

Mevcut Çözüm;

G.N.	A.N.	K.Z.

Şekil 3.16. YAKA mevcut çözüm

İyileştirme Formülü Uygulanmış Çözüm;

G.N.	A.N.	K.Z. + 15

Şekil 3.17. YAKA iyileştirme formülü uygulanmış çözüm

Örnek çözüm kümesi üzerinde iyileştirme formülü uygulanacak olursa, Çizelge 3.11'deki gibi olur.

Çizelge 3.11. YAKA mevcut çözüm – İyileştirilmiş çözüm

	Gün	Ay	Kalkış Saati
Mevcut Çözüm	13	12	19:07
İyileştirme Formülü Uygulanmış Çözüm	13	12	19:22

3) Besin kaynaklarının uygunluk değerlerinin ölçülmesi

Besin kaynakları oluşturulduktan sonra her bir besin kaynağının kalitesi yani nektar miktarı ölçülmelidir. Algoritmada, arıların yaptığı bu eyleme karşılık gelen işlem, Formül 3'de verilmiştir (Karaboga and Akay 2009). Bu formülde "f", nektar miktarını, "c", kısıtları, "a", kısıtlar için tanımlanan ağırlık değerini temsil etmektedir.

$$f=\sum_{i=1}^{n} c_j a_j$$

Formül 3 Nektar miktarı

Besin kaynağının kalitesini hesaplama, problemin yapısına ve tanımlanan kısıtlamalara göre değişmektedir. Dolayısıyla uygunluk formülü her problemin yapısına göre değişmektedir. Besin kaynağının kalitesi (f) hesaplandıktan sonra besin kaynağına uygunluk değeri verilir (Karaboga and Akay 2009).

$$(\text{uygunluk değeri})_i = \frac{1}{1+f_i}, \; f_i \geq 0$$

$$(\text{uygunluk değeri})_i = 1 + \text{abs}(f_i), \; f_i < 0$$

Formül 4 Uygunluk değeri hesaplama

Problemin amaç fonksiyonu nektar miktarının bulunduğu denklemdir (f). Bu çalışmadaki amaç, amaç fonksiyonunu minimum yapmaktır. Bu değer ne kadar küçük olursa oluşturulan çözüm kümesi, besin açısından o kadar zengin demektir. O zaman f değeri, kadar küçük olursa uygunluk değeri o kadar uygun olur.

Her yeni komşuda bulunan uygunluk değeri, eski uygunluk değeri ile kontrol edilerek açgözlü yaklaşıma göre seçim yapılır. Yani, iyileştirme formülü sonucu oluşan çözümün uygunluk değeri ile mevcut çözümün uygunluk değeri karşılaştırılır. Eğer yeni uygunluk değeri daha büyükse, mevcut çözüm kümesi yeni çözüm kümesi olarak güncellenir ve eski çözüm kümesi hafızadan silinir.

Aşağıda yukarıda bahsedilen işlem adımlarının, temsili anlatımı verilmiştir:

Amaç fonksiyonunun bu problem içerisindeki temsili Formül 5'deki gibi özelleştirilmiştir.

$$\sum_{i,j,k}((\varphi(x_{ijk_{G.N.\ U\ A.N.\ U\ K.Z.}}) \mid h(V.G.S.\ \&\ H.D.K.G.S. \mid G.N.\ U\ A.N.\ U\ K.Z.)) * A_{ij} *$$

$$C_{kw} * x_{ijk_{G.N.\ U\ A.N.\ U\ K.Z.}}) * \sigma$$

Formül 5 Amaç fonksiyonu

İlk olarak mevcut çözümün uygunluk değeri, aşağıda belirtilen Adım 1 ve Adım 4 arasındaki ara basamaklarda hesaplanmıştır.

Adım1: Mevcut çözümün ağırlık değeri bulunur.

Mevcut çözüm kısıt veri setinde yer almakta ve yumuşak kısıtların 2. maddesindeki kısıt tanımlamasına uymaktadır. O zaman bu çözüm için σ değeri 0,25'dir.

Adım2: Mevcut çözümün tarife veri setindeki kalkış ve varış yerine göre saat cinsinden oluşan mesafesi bulunur.

Bu çözüm için,

i=SFO ve j=YVR dir. O zaman Çizelge 3.8'e göre, A_{ij}=2h25m=2,25'dir.

Adım3: Mevcut çözümün uçuş saati başına düşen maliyeti amaç fonksiyonuna dahil edilir.

$$C_{kw} = \{ C_{k1} \mid C_{k2} \}$$

C_{k1}=\$5,303 ve C_{k2}=\$2,845

Adım4: Mevcut çözümün temsil ettiği gün, ay, saat'de hava durumundan kaynaklı gerçekleşen gecikme sayısı alınır.

Kısıt veri setinde bu çözümün,

$\varphi(x_{ijk\ G.N.\ U\ A.N.\ U\ K.Z.}) \mid h(V.G.Z.\ \&\ H.D.K.G.Z. \mid G.N.\ U\ A.N.\ U\ K.Z.)$ değeri 6 olarak gözükmektedir.

Yukarıda her bir adımda bulunan değerlere göre;

Amaç fonksiyonu=f=(((6*5303*2,25)+(6*2845*2,25))*0,25)=(71590,5+38407,5)*0,25 =27499,5

27499,5>0 olduğu için; uygunluk değeri = $\frac{1}{1+f} = \frac{1}{1+27499,5} = \frac{1}{27500,5}$ = 3636297*10^{-11} olur.

İkinci olarak iyileştirme formülü uygulanmış çözümün uygunluk değeri, aşağıda belirtilen Adım 1 ve Adım 4 arasındaki ara basamaklarda hesaplanmıştır.

Adım1: Mevcut çözümün ağırlık değeri bulunur.

Mevcut çözüm kısıt veri setinde yer almakta ve yumuşak kısıtların 2. maddesindeki kısıt tanımlamasına uymaktadır. O zaman bu çözüm için σ değeri 0,25'dir.

Adım2: Mevcut çözümün tarife veri setindeki kalkış ve varış yerine göre saat cinsinden oluşan mesafesi bulunur. Ama iyileştirme formülü sonucu elde edilen çözüm tarife veri setinde yer almıyorsa, bu çalışma süresince yer almayan bu çözümlerin amaç fonksiyonu hesaplanırken;

i=SFO ve j=YVR ya da i=YVR ve j=SFO olarak düşünülür. Çizelge 3.8'e göre, A_{ij}=2h25m=2,25'dir.

Adım3: Mevcut çözümün uçuş saati başına düşen maliyeti amaç fonksiyonuna dahil edilir.

C_{kw} = { C_{k1} | C_{k2} }
C_{k1}=$5,303 ve C_{k2}=$2,845

Adım4: Mevcut çözümün temsil ettiği gün, ay, saat'de hava durumundan kaynaklı gecikme sayısı alınır.

Kısıt veri setinde bu çözümün, $\varphi(x_{ijk_{G.N.\,U\,A.N.\,U\,K.Z.}})\,|\,h(V.G.Z.\,\&\,H.D.K.G.Z.|\,G.N.\,U\,A.N.\,U\,K.Z.)$ değeri 2 olarak gözükmektedir.

Yukarıda her bir adımda bulunan değerlere göre;

Amaç fonksiyonu=f=(((2*5303*2,25)+(2*2845*2,25))*0,25)=(23863,5+12802,5)*0,25 = 36666*0,25=9166,5

9166,5>0 olduğu için; uygunluk değeri = $\frac{1}{1+f} = \frac{1}{1+9166,5} = \frac{1}{9167,5}$ =10908099*10^{-11} olur.

Çizelge 3.12'de görüldüğü gibi, mevcut çözüm ve iyileştirme formülü uygulanmış çözümün uygunluk değerleri verilmiştir. İyileştirme formülü uygulanmış çözümün uygunluk değeri daha büyük olduğu için çözüm hafızaya alınır, eski çözüm silinir.

Çizelge 3.12. Mevcut ve iyileştirilmiş çözümün uygunluk değerleri

	Gün	Ay	Kalkış Saati	Uygunluk Değeri
Mevcut Çözüm	13	12	19:07	$3636297*10^{-11}$
İyileştirme Formülü Uygulanmış Çözüm	13	12	19:22	$10908099*10^{-11}$

Çalışma içerisinde, eğer iyileştirilmiş çözüm tarife veri setinde ve kısıt veri setinde yer almazsa amaç fonksiyonu aşağıdaki gibi özelleştirilir:

Amaç fonksiyonu=f=(((0,0000001*5303*2,25)+(0,0000001*2845*2,25))*0,1)

σ değeri 0,1 olarak alınır. Çözümü oluşturan uçuş bilgileri, kısıt veri setinde yer almadığı için, daha önce belirtilen (varsayılan) ağırlık değeri işleme alınır.

$\varphi(x_{ijk_{G.N.\ U\ A.N.U\ K.Z.}}) \mid h(V.G.Z.\ \&\ H.D.K.G.Z. \mid G.N.\ U\ A.N.\ U\ K.Z.)$ değerinin 0,0000001 olarak alınmasının sebebi ise çözümü oluşturan uçuş bilgilerinin kısıt veri setinde yer almamasıdır. Demek ki, hava durumundan kaynaklı bir gecikme mevcut değildir. Tarifenin bu uçuş bilgileriyle oluşturulması hedeflenen bir sonuçtur. Çözüm kümesinin maliyeti minimum olacak şekilde özelleştirilmesi istendiği için, "0,0000001" gibi çok küçük bir değer işleme alınmıştır.

4) İşçi arı safhası

Bu aşamada, işçi arılar besin kaynaklarına giderek besin kaynağını işler ve gözcü arılara dans hareketi ile besin kaynakları hakkında bilgi vermek için kovana geri döner. Algoritmada, işçi arıların yaptığı bu eyleme karşılık gelen işlemde, işçi arıların ziyaret ettiği her bir çözüm kümesine geliştirme formülü uygulanır. Geliştirme Formülü uygulanırken, ilk olarak hafızada yer alan mevcut çözüm kümesinin rastgele üretilen komşusuna gidilir. Daha sonra Formül 6'da yer alan geliştirme formülü ile besin kaynağının yeni parametre değeri bulunmuş olur.

Bu çalışmada rastgele üretilen komşu çözüm 2 farklı yaklaşımla ele alınmıştır.

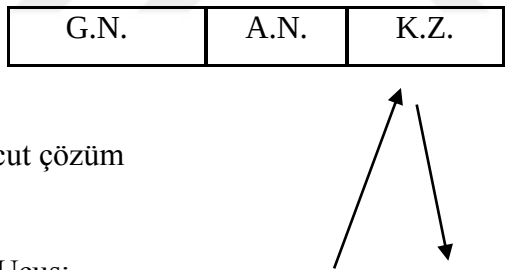
1. Yaklaşım

Rastgele lokasyon seçilir ve yer değiştirme işlemi gerçekleştirilir. Yer değiştirme yöntemi ile mevcut çözümden rastgele komşu çözüm üretilmiş olur. Çözüm kümesinde yer alan rastgele uçuş bilgisindeki kalkış zamanı ile mevcut çözümün kalkış zamanı yer değiştirilir. Bu işlemin gerçekleştirilme şekli Şekil 3.18, Şekil 3.19, Şekil 3.20, Şekil 3.21'de verilmiştir.

Rastgele indis= i olsun.

i=rand()%380

Mevcut Çözüm;



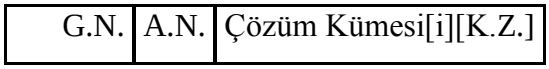
Şekil 3.18. YAKA mevcut çözüm

Rastgele Uçuş;

Şekil 3.19. Rastgele uçuş

Yer değiştirme işlemi sonucunda;

Komşu Çözüm;



Şekil 3.20. Komşu çözüm

Rastgele Uçuş ;

G.N.	A.N.	K.Z.

Şekil 3.21. Rastgele uçuş

Örnek çözüm kümesi üzerinde komşu çözüm üretilecek olursa, Çizelge 3.13'deki gibi olur.

Çizelge 3.13. YAKA yerdeğiştirme işlemi ile elde edilen komşu çözüm

		Gün	Ay	Kalkış Saati	Rastgele Üretilen İndis
1	Mevcut Çözüm	13	12	19:07	
	Rastgele Uçuş	20	1	10:20	i=11 olsun.
	Yerdeğiştirme Sonucu				
2	Komşu Çözüm	13	12	10:20	
	Rastgele Uçuş	20	1	19:07	

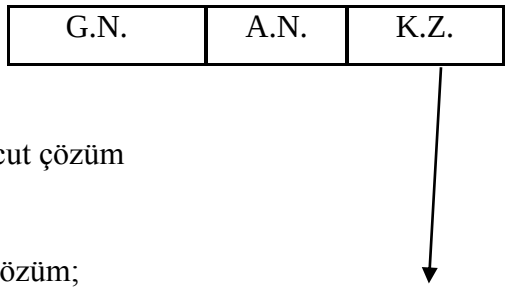
2. Yaklaşım

Rastgele bir lokasyon belirlenir ve belirlenen global histograma (ölçüme) göre yeni değerler verilir. Belirlenen uçuşlar için, global sınır olan "5 dk ve 15 dk" kriterine göre kalkış zamanına ekleme yapılıp mevcut çözüm güncellenir. Bu işlemin detayları, Şekil 3.22, Şekil 3.23, Şekil 3.24 ve Şekil 3.25'de verilmiştir.

[1-20] arasında rastgele bir sayı üretilir.

→Eğer üretilen sayı 1 ile 10 arasında ise;

Mevcut Çözüm;



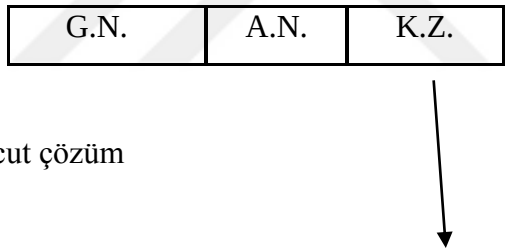
Şekil 3.22. YAKA mevcut çözüm

Komşu Çözüm;

Şekil 3.23. YAKA komşu çözüm

→ Eğer üretilen sayı 10 ile 20 arasında ise;

Mevcut Çözüm;



Şekil 3.24. YAKA mevcut çözüm

Komşu Çözüm;

Şekil 3.25. YAKA komşu çözüm

Komşu çözümün 5 ya da 15 dk eklenerek elde edilmesinin sebebi, literatürde yapılan araştırmalar sonucunda bu sürelerdeki geç kalkış ya da varışların bir sonraki seferlerdeki gecikmelere sebep olduğu bilgisine ulaşılmasından kaynaklanmaktadır.

Örnek çözüm kümesi üzerinde komşu çözüm üretilecek olursa, Çizelge 3.14'deki gibi olur:

Çizelge 3.14. YAKA global histograma göre yeni değerle elde edilen komşu çözüm

		Gün	Ay	Kalkış Saati	Rastgele Üretilen Sayı
1	Mevcut Çözüm	13	12	19:07	1 ≤ Rastgele Üretilen Sayı ≤10
	Komşu Çözüm	13	12	19:12	
2	Mevcut Çözüm	13	12	19:07	10 ≤ Rastgele Üretilen Sayı ≤20
	Komşu Çözüm	13	12	19:22	

$$p_{ik} = X_{ik} + Q * (X_{ik} - X_{jk})$$

Formül 6 (Karaboga and Akay 2009)

p_{ik}: Yeni parametre değeri

i: Mevcut çözüm kümesinin indeks numarası

k: Rastgele seçilen parametre indeks numarası

j: Rastgele seçilen komşu çözüm kümesi indeks numarası

X_{ik}: Mevcut çözüm kümesinin k. parametre değeri

X_{jk}: Rastgele çözüm kümesinin k. parametre değeri

Q: Çözüm kümelerinin parametreleri arasındaki değişim mesafe değeri

YAKA'da, Q= (random(0,1)-0.5)*2=[-1,1] olarak alındığı için bu çalışmada da, Q değeri [-1,1] arasında rastgele üretilen bir sayı olarak dikkate alınmıştır.

Formül 6'ya göre, bu çalışma kapsamında geliştirme formülü uygulanmıştır. Çizelge 3.14'de 2 numaralı örnek komşu çözüm kümesi üzerinde geliştirme formülü uygulanacak olursa;

p=mevcut çözüm[2] + ((rand()%2-0,5)*2)*(mevcut çözüm[2]-komşu çözüm[2]))

(rand()%2-0,5)*2)=0,4 olsun.

Çizelge 3.15'de görüldüğü gibi geliştirme formülü sonucu yeni çözüm kümesi elde edilmiş olur.

Çizelge 3.15. Geliştirme formülü ile elde edilen çözüm

	Gün	Ay	Kalkış Saati
Mevcut Çözüm	13	12	19:07
Komşu Çözüm	13	12	19:22
Geliştirme Formülü Sonucu Yeni Çözüm	13	12	19:01

Geliştirme formülü sonucunda, çözüm kümesinde iyileşme olup olmadığını incelemek için, uygunluk değerleri kontrol edilir. Eğer eski çözüm kümesine göre, daha iyi bir çözüm elde edildiyse, yeni çözüm mevcut çözüm olarak güncellenir ve hafızaya alınır, eski çözüm ise hafızadan silinir.

Aşağıda yukarıda bahsedilen işlem adımlarının, temsili anlatımı verilmiştir:

Geliştirme formülü sonucu oluşan çözümün uygunluk değeri, aşağıda belirtilen Adım 1 ve Adım 4 arasındaki ara basamaklarda hesaplanmıştır.

Adım1: Mevcut çözümün ağırlık değeri bulunur.

Mevcut çözüm kısıt veri setinde yer almakta ve yumuşak kısıtların 2. maddesindeki kısıt tanımlamasına uymaktadır. O zaman bu çözüm için σ değeri 0,25'dir.

Adım2: Mevcut çözümün tarife veri setindeki kalkış ve varış yerine göre saat cinsinden oluşan mesafesi bulunur. Ama geliştirme formülü sonucu elde edilen çözüm, tarife veri setinde yer almıyorsa, bu çalışma süresince amaç fonksiyonu hesaplanırken;

i=SFO ve j=YVR ya da i=YVR ve j=SFO olarak alınır. Çizelge 3.8'e göre, A_{ij}=2h25m=2,25'dir.

Adım3: Mevcut çözümün uçuş saati başına düşen maliyeti amaç fonksiyonuna dahil edilir.

C_{kw} = { C_{k1} | C_{k2} }

C_{k1}=$5,303 ve C_{k2}=$2,845

Adım4: Mevcut çözümün temsil ettiği gün, ay, saat'de hava durumundan kaynaklı gecikme sayısı alınır.

Kısıt veri setinde bu çözümün $\varphi(x_{ijk_{G.N.\,U\,A.N.\,U\,K.Z.}}) \,|\, h(V.G.Z. \,\&\, H.D.K.G.Z. \,|\, G.N.\,U\,A.N.\,U\,K.Z.)$ değeri 1 olarak gözükmektedir.

Yukarıda her bir adımda bulunan değerlere göre;

Amaç fonksiyonu=f=(((1*5303*2,25)+(1*2845*2,25))*0,25)=(11931,75+6401,25)*0,25= 18333*0,25=4583,25

4583,25>0 olduğu için; uygunluk değeri = $\frac{1}{1+f} = \frac{1}{1+4583,25} = \frac{1}{4584,25}$ =21813819*10^{-11} olur.

Çizelge 3.16'da görüldüğü gibi mevcut çözüm ve geliştirme formülü uygulanmış çözümün uygunluk değerleri verilmiştir. Geliştirme formülü uygulanmış çözümün uygunluk değeri daha büyük olduğu için çözüm hafızaya alınır, eski çözüm silinir.

Çizelge 3.16. Mevcut ve geliştirilmiş çözümün uygunluk değerleri

	Gün	Ay	Kalkış Saati	Uygunluk Değeri
Mevcut Çözüm	13	12	19:07	3636297*10^{-11}
Geliştirme Formülü Sonucu Yeni Çözüm	13	12	19:01	21813819*10^{-11}

5) Gözcü arı safhası

Bu aşamada, gözcü arılar diğer arıların topladığı besin kaynakları hakkındaki bilgiyi alırlar. Kovanda yer alan dans bölümünde, işçi arıların yaptığı dans hareketlerinden besin kaynaklarının konum bilgisini öğrenerek bu besinleri işlemek üzere besin kaynaklarına giderler. Çözüm kümesinde uygunluk değeri yüksek olan çözümün olasılık değeri yüksek olacak şekilde, mevcut uygunluk değerlerine göre seçim şansı oluşturulur. Bu olasılık değerleri Formül 7'ye göre ayarlanır (Karaboga and Akay 2009; Küçüksille ve Tokmak 2011).

$$o_i = 0{,}6 + \frac{(\text{uygunluk değeri})_i}{\sum_{i=1}^{n} (\text{uygunluk değeri})_i}$$

Formül 7

Formül 7'de 0,6 eklenmesinin nedeni, literatürde [0,6-0,9] arasında bir değerin formüle eklenmesi ile daha iyi bir çözüm alınabildiğinin belirtilmesinden dolayıdır. YAKA algoritmasında [0,1] arasında rastgele sayı üretilir, bu sayı o_i değeri ile karşılaştırılır. Eğer rastgele üretilen sayı, o_i değerinden küçükse gözcü arı besin kaynaklarına gönderilir ve her bir çözüme geliştirme formülü uygulanır. Daha sonra uygunluk formülü uygulanır. Bulunan uygunluk değeri, eski uygunluk değeri ile kontrol edilir. Eğer yeni uygunluk değeri daha büyükse, mevcut çözüm kümesi yeni çözüm kümesi olarak güncellenir. Eski çözüm kümesi hafızadan silinir.

3.5.2. Benzetilmiş tavlama

BT (Benzetilmiş Tavlama), ismini katı metali ısıtmak anlamına gelen tavlama kelimesinden alan, katıların fiziksel tavlama süreci ile olan benzerliğinden esinlenerek oluşturulan NP Zor sınıfı problemlerin çözümünde kullanılan metasezgisel algoritmalardan biridir. Katı metali ısıtıp mükemmel kristale dönüştürmek için hangi adımlardan geçmesi gerektiğini gösteren stokastik arama yöntemi sunar. Bu yöntemde,

katı cisim (metal) önce ısıtılır daha sonra özel bir yöntemle soğutulur. Sadece çözümü iyileştiren durumlar değil, bazen çözümü kötüleştiren durumlar da belirtilen ihtimallere göre kabul edilir. Böylece yerel optimum tuzaklardan kurtularak global optimuma erişebilmek amaçlanır (Kirkpatrick *et al.* 1983).

BT sahte kodu Şekil 3.26'da verilmiştir.

```
1 Başla
2 başlangıç değerlerinin atanması(sıcaklık, çevrim sayısı, soğutma oranı,
durdurma kriteri)
3 s0 = rastgele çözüm kümesi oluştur
4 en iyi çözüm= s0
5 maliyet(en iyi çözüm)= maliyet(s0)
6 while (sıcaklık>durdurma kriteri)
7 çevrim sayısı = 1
8 repeat
9 s1 = pertürbe yaparak komşu bul(s0)
10 ΔC = maliyet(s1) - maliyet(s0)
11 if(ΔC < 0)
12 s0 = s1
13 if(maliyet(s0) < maliyet(en iyi çözüm))
14 en iyi çözüm= s0
15 maliyet(en iyi çözüm)= maliyet(s0)
16 end if
17 end if
18 else if(random(0,1) < e-(ΔC/sıcaklık))
19 s0 = s1
20 end if
21 çevrim sayısı ++
22 Until çevrim sayısı
23 sıcaklık=sıcaklık*soğutma oranı
24 end while
```

Şekil 3.26. BT sahte kodu

Şekil 3.26'da yer alan sahte kod incelendiğinde, 2 nolu satırda, başlangıç sıcaklık, çevrim sayısı yani her sıcaklıkta kaç kere pertürbe işleminin yapılacağı, her iterasyonda ne kadar soğutma uygulanacağı, durdurma kriteri yani donma sıcaklığı belirlenmiştir. Rastgele oluşturulan çözüm kümesi ile işleme başlanmıştır (3 numaralı satır). En başta ilk çözüm,

en iyi çözüm olarak ve ilk çözümün maliyeti de, en iyi çözümün maliyeti olarak kabul edilmiştir (4 ve 5 nolu satırlar). Mevcut sıcaklık değeri, donma sıcaklığından büyük olduğu müddetçe işlemler yapılmaya başlanmıştır (6 nolu satır). Benzetilmiş tavlama için belirtilen pertürbe sayısı kadar içerdeki döngü gerçekleştirilir (7 ve 8 nolu satırlar). Çözüm kümesine pertürbasyon işlemi yapılarak yeni komşu çözüm elde edilir, yeni çözümün maliyeti ile eski çözümün maliyet farkı hesaplanarak ΔC bulunur. Eğer 0'dan küçükse, daha optimum sonucu bulduğumuz için mevcut çözüm güncellenir ve bu çözüm en iyi çözüm ile karşılaştırılarak en iyi çözüm ve en iyi çözümün maliyet güncellenmesi yapılır (9,10,11,12,13,14,15,17 nolu satırlar). 18 nolu satırda yapılan işlemde ΔC, 0'dan küçük değilse hemen yeni çözümün değerlendirilmesi reddedilmez. Bazen "hill-climb" da olduğu gibi çözümün yerel minimumda takılma ihtimali olabilir. Benzetilmiş tavlamada bu duruma bir çözüm getirilerek, bir ihtimale göre kötü durumu da kabul edip, çözümü yerel minimumdan kurtarıp global minimuma ulaştırmak hedeflenir. Benzetilmiş tavlamanın kötü durumları kabul etme ihtimali açıklanacak olursa, 0 ve 1 arasında rastgele üretilen sayı e-(ΔC/sıcaklık) değerinden küçükse kötü durumdur ama kabul edilir (19,20 nolu satırlar). Pertürbasyon sayısının tamamlanıp tamamlanmadığı kontrol edilir (21 ve 22 nolu satırlar). Her döngüde soğutma oranı kadar sıcaklık azaltılır (23 ve 24 nolu satırlar).

BT önemli algoritma adımları:

1) Çözüm uzayı belirlenmesi - Başlangıç çözümünün belirlenmesi

Tarife veri setinden alınan uçuş bilgileri ile popülasyon boyutu kadar rastgele uçuşlardan oluşan çözüm kümesi oluşturulur. Rastgele üretilen uçuşların oluşturulma şekli Çizelge 3.17'de verilmiştir.

Çizelge 3.17. BT çözüm kümesi oluşturma

n = popülasyon boyutu rastgele indis = 1 + rand() % (tarife veri seti kayıt sayısı-1) tarife veri seti kayıt sayısı=380 rastgele indis = 1 + rand() %(379)			
Uçuş[0]	t.v.s[rastgele indis][0]	t.v.s[rastgele indis][1]	t.v.s[rastgele indis][3]
Uçuş[1]	t.v.s[rastgele indis][0]	t.v.s[rastgele indis][1]	t.v.s[rastgele indis][3]
…	…	…	…
Uçuş[n-1]	t.v.s[rastgele indis][0]	t.v.s[rastgele indis][1]	t.v.s[rastgele indis][3]
Uçuş[n]	t.v.s[rastgele indis][0]	t.v.s[rastgele indis][1]	t.v.s[rastgele indis][3]

Rastgele oluşturulan çözüm kümesinden, rastgele oluşturulan herhangi bir çözüme örnek verilecek olursa (Çizelge 3.18);

Çizelge 3.18. BT örnek çözüm kümesi

Gün	Ay	Kalkış Saati
13	12	19:07

2) Komşuluk yapısı belirlenmesi

İşlemci hızını önemli ölçüde belirleyen kısım, pertürbe işleminin yapıldığı yani komşuluk yapısının belirlendiği kısımdır. Bu çalışmada, rastgele üretilen komşu çözüm 2 farklı yaklaşımla ele alınmıştır.

1. <u>Yaklaşım</u>

Rastgele oluşturulan mevcut çözüm kümesinden, rastgele 2 lokasyon seçilir ve yer değiştirme işlemi gerçekleştirilir. Yer değiştirme yöntemi ile mevcut çözüm kümesinden

rastgele komşu çözüm üretilmiş olur. Bu işlemin gerçekleştirilme şekli Şekil 3.27, Şekil 3.28, Şekil 3.29, Şekil 3.30'da verilmiştir.

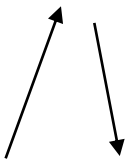
İlk rastgele indis i olsun.
İkinci rastgele indis j olsun.
i=rand()%380
j=rand()%380

Mevcut çözüm kümesindeki "i ve j" indisinde yer alan uçuşlar arasında yer değiştirme işlemi yapılır.

Mevcut Çözüm Kümesinde;

i.çözüm;

G.N.	A.N.	K.Z.



Şekil 3.27. i. Çözüm

j.çözüm;

G.N.	A.N.	t.v.s[i][K.Z.]

Şekil 3.28. j.Çözüm

Yer değiştirme işlemi sonucunda;

i.çözüm;

G.N.	A.N.	t.v.s[i][K.Z.]

Şekil 3.29. Yerdeğiştirme işlemi sonucunda i. çözüm

j.çözüm;

G.N.	A.N.	K.Z.

Şekil 3.30. Yerdeğiştirme işlemi sonucunda j. çözüm

Örnek çözüm kümesi üzerinde komşu çözüm üretilecek olursa, Çizelge 3.19'daki gibi olur.

Çizelge 3.19. BT yerdeğiştirme işlemi ile elde edilen komşu çözüm

		Gün	**Ay**	**Kalkış Saati**	**Rastgele Üretilen İndis**
1	Mevcut Çözüm	13	12	19:07	i=11 olsun.
	Rastgele Uçuş	20	1	10:20	
	Yerdeğiştirme Sonucu				
2	Komşu Çözüm	13	12	10:20	
	Rastgele Uçuş	20	1	19:07	

2. <u>Yaklaşım</u>

Rastgele bir lokasyon belirlenir ve belirlenen global histograma göre yeni değerler verilir. Yani, bizim belirlediğimiz uçuşlar için, global sınır olan "5 dk ve 15 dk" kriterine göre kalkış zamanına ekleme yapılıp mevcut çözüm güncellenir. Bu işlemin gerçekleştirilme şekli, Şekil 3.31, Şekil 3.32, Şekil 3.33, Şekil 3.34'de verilmiştir.

[1-20] arası rastgele bir sayı üretilir:

→Eğer üretilen sayı 1 ile 10 arasında ise;

Mevcut Çözüm;

G.N.	A.N.	K.Z.

Şekil 3.31. BT mevcut çözüm

Komşu Çözüm;

G.N.	A.N.	K.Z. + 5

Şekil 3.32. BT komşu çözüm

→ Eğer üretilen sayı 10 ile 20 arasında ise;

Mevcut Çözüm;

G.N.	A.N.	K.Z.

Şekil 3.33. BT mevcut çözüm

Komşu Çözüm;

G.N.	A.N.	K.Z. + 15

Şekil 3.34. BT komşu çözüm

Örnek çözüm kümesi üzerinde komşu çözüm üretilme aşaması, Çizelge 3.20'de verilmiştir.

Çizelge 3.20. BT global histograma göre yeni değerle elde edilen komşu çözüm

		Gün	Ay	Kalkış Saati	Rastgele Üretilen Sayı
1	Mevcut Çözüm	13	12	19:07	1 ≤ Rastgele Üretilen Sayı ≤10
	Komşu Çözüm	13	12	19:12	
2	Mevcut Çözüm	13	12	19:07	10 ≤ Rastgele Üretilen Sayı ≤20
	Komşu Çözüm	13	12	19:22	

3) ΔC kontrolü

Çizelge 3.21'de 2 numaralı örnek komşu çözüm kümesi ele alınırsa,

Çizelge 3.21. BT komşu çözüm kümesi

Gün	Ay	Kalkış Saati
13	12	19:22

Komşu çözümün maliyet değeri, aşağıda belirtilen Adım 1 ve Adım 4 arasındaki ara basamaklarda hesaplanır:

Adım1: Komşu çözümün ağırlık değeri bulunur. Mevcut çözüm kısıt veri setinde yer almakta ve yumuşak kısıtların 2. maddesindeki kısıt tanımlamasına uymaktadır. O zaman bu çözüm için σ değeri 0,25'dir.

Adım2: Mevcut çözümün tarife veri setindeki kalkış ve varış yerine göre saat cinsinden oluşan mesafesi bulunur. Ama geliştirme formülü sonucu elde edilen çözüm tarife veri setinde yer almıyorsa bu çalışma süresince tarife veri setinde yer almayan yeni çözümlerin amaç fonksiyonu hesaplanırken;

i=SFO ve j=YVR ya da i=YVR ve j=SFO olarak düşünülür. Çizelge 3.8'e göre, A_{ij}=2h25m=2,25 olarak bulunur.

Adım3: Mevcut çözümün uçuş saati başına düşen maliyeti amaç fonksiyonuna dahil edilir.

$C_{kw} = \{ C_{k1} \mid C_{k2} \}$
C_{k1}=\$5,303 ve C_{k2}=\$2,845

Adım4: Mevcut çözümün temsil ettiği gün, ay, saat'de hava durumundan kaynaklı gecikme sayısı alınır.

Kısıt veri setinde bu çözümün,

$\varphi(x_{ijk_{G.N.\,U\,A.N.\,U\,K.Z.}}) \mid h(V.G.Z. \,\&\, H.D.K.G.Z. \mid G.N.\,U\,A.N.\,U\,K.Z.)$ değeri 2 olarak gözükmektedir.

Yukarıda her bir adımda bulunan değerlere göre;

Amaç fonksiyonu

f=(((2*5303*2,25)+(2*2845*2,25))*0,25)=(23863,5+12802,5)*0,25=36666
*0,25=9166,5

Bulunan bu değer katılarak mevcut çözüm kümesinin yeni toplam maliyet değeri hesaplanır. Böylece komşu çözüm kümesinin maliyet değeri hesaplanmış olur. Daha sonra aşağıdaki formüle göre ΔC hesaplanır.

ΔC = maliyet(komşu çözüm kümesi) - maliyet(mevcut çözüm kümesi)

ΔC < 0 ise, komşu çözüm, mevcut çözüm ve en iyi çözüm olarak güncellenir. En iyi çözümün maliyeti de komşu çözümün maliyeti olarak güncellenir. Çevrim sayısı ve durdurma kriteri sağlanıncaya kadar sıra ile algoritmanın adımları tekrar işletilir.

3.5.3. Genetik algoritma

GA, popülasyondaki bireylerden doğaya uyum sağlayanların yaşama olasılığı daha yüksek olduğu için bu bireylerin genlerinin yeni kuşaklara aktarılması amacıyla bir dizi işlemin yapıldığı, stokastik algoritma yönteminin sunulduğu metasezgisel algoritmalardan biridir (Holland 1992).

GA sahte kodu Şekil 3.35'de verilmiştir.

```
1 Başla
2 başlangıç değerlerinin atanması(popülasyon boyutu,
çaprazlama oranı, mutasyon oranı, kuşak sayısı)
3 P0 = Rastgele ilk popülasyon havuzunu oluştur
4 en iyi çözüm = P0.first()
5 maliyet(en iyi çözüm)=maliyet(P0.first())
6 while(!kuşak sayısı)
7 uygunluk değeri bul(P0)
8 Pyeni = yeniden üretim(turnuva seçimi(P0))
9 çaprazlama(Pyeni , pç) // pç = Çaprazlama olasılığı
10 mutasyon uygula(Pyeni , pm) // pm = Mutasyon olasılığı
11 if(uygunluk değeri bul(Pyeni) < uygunluk değeri
bul(P0))
12 P0= Pyeni
13 maliyet(P0)=maliyet(Pyeni)
14 end while
```

Şekil 3.35. GA sahte kodu

Şekil 3.35'de yer alan sahte kod incelendiğinde, 2 nolu satırda, popülasyon boyutu, iki ebeveynin yeni birey oluşturmak üzere çaprazlanma ihtimalini belirleyen çaprazlama oranı, bir müddet sonra benzer özellikte birey oluşumunu önleyip çeşitlilik sağlamak için yapılacak olan mutasyon yapma ihtimalini belirleyen mutasyon oranı, kaç kuşak

popülasyon oluşturulacağını belirlemek için gerekli olan ilk değer atamaları yapılmıştır. Popülasyonda yer alan her bir kromozom rastgele oluşturulur. Her bir çözümün (kromozomun) maliyeti belirlenir (3 nolu satır). 4 ve 5 nolu satırda, popülasyon havuzunun ilk çözümü en iyi çözüm ve ilk çözümün maliyeti en iyi çözümün maliyeti olarak kabul edilir. Üretilecek kuşak sayısı kadar oluşturulacak popülasyonda genetik algoritma operatörleri uygulanır (6 nolu satır). 7 nolu satırda eğer popülasyon havuzundaki herhangi bir çözümün maliyeti en iyi çözümün maliyetinden daha küçük ise en iyi çözüm ve en iyi çözüm maliyeti üzerinde gerekli güncelleme işlemi yapılır. Turnuva seçim yöntemi ile eşleşmek üzere iki ebeveyn seçimi yapılır. Geçici olarak oluşturulan P_{yeni} popülasyona seçilen ebeveynler ve maliyetleri aktarılır (8 nolu satır). Çaprazlama ve mutasyon olasılığına göre P_{yeni} de kromozomlar üzerinde çaprazlama ve mutasyon işlemleri yapılır (9,10 nolu satırlar). P_{yeni}'de maliyet değerleri P_0'daki çözümlerin maliyet değerlerinden daha küçük bir değere sahip ise, P_0'daki çözümler ve P_0'daki maliyet değerleri güncellenir. Kuşak sayısı tamamlanınca, program sona erer (11,12,13,14 nolu satırlar).

GA önemli algoritma adımları:

1) Popülasyon havuzunun oluşturulması

Popülasyon havuzu içerisinde yer alan her bir birey (kromozom) rastgele oluşturulur. Popülasyon havuzu oluşturulurken C++ Standart Şablon Kütüphanesinde yer alan "make pair" veri yapısı kullanılmıştır. Bu yapı sayesinde, iki nesne bir arada tutulabilmektedir. Oluşturulan yapıda ilk nesne; kromozomu, ikinci nesne; kromozomun maliyetini tutacak şekilde özelleştirilmiştir.

Popülasyonda yer alan toplam kromozom sayısı, belirlenen popülasyon boyutu kadardır. Amaç fonksiyonu ile popülasyon içerisinde yer alan her bir bireyin ne kadar uygun olduğu tespit edilmiştir. N, popülasyon boyutunu temsil etmektedir. Her bir kromozom tek boyutlu vektör ile temsil edilmiştir. 1. kromozomun yani rastgele üretilen ilk uçuşun oluşturulma şekli Çizelge 3.22'de verilmiştir.

Çizelge 3.22. GA örnek kromozom yapısı

1. Kromozom	Uçuş[0] = t.v.s[rastgele indis][0]
	Uçuş[1] = t.v.s[rastgele indis][1]
	Uçuş[2] = t.v.s[rastgele indis][3]

Rastgele oluşturulan çözüm kümesine Çizelge 3.23'de örnek verilmiştir.

Çizelge 3.23. GA örnek çözüm kümesi

Gün	Ay	Kalkış Saati
13	12	19:07

Rastgele oluşturulan popülasyon havuzu Çizelge 3.24'de verilmiştir.

Çizelge 3.24. Popülasyon havuzu

İndis No	Kromozom No	Kromozom	Kromozomun Uygunluk Değeri (Maliyeti)
0	1	13 12 1907	27499,5
1	2	11 12 629	36666
…	…	…	…
…	…	7 4 1345	…
…	…	10 3 1125	…
N-1	N	…	…

Çizelge 3.24'de verilen kromozomlardan "11 12 629" un uygunluk değeri, aşağıda belirtilen Adım 1 ve Adım 4 arasındaki ara basamaklarda hesaplanmıştır.

Adım1: Mevcut çözümün ağırlık değeri bulunur. Mevcut çözüm kısıt veri setinde yer almakta ve yumuşak kısıtların 2. maddesindeki kısıt tanımlamasına uymaktadır. O zaman bu çözüm için σ değeri 0,25'dir.

Adım2: Mevcut çözümün tarife veri setindeki kalkış ve varış yerine göre saat cinsinden oluşan mesafesi bulunur.

i=SFO ve j=JFK'dır. O zaman Çizelge 3.8'e göre, A_{ij}=6h00m=6'dır.

Adım3: Mevcut çözümün uçuş saati başına düşen maliyeti amaç fonksiyonuna dahil edilir.

$C_{kw} = \{ C_{k1} \mid C_{k2} \}$
C_{k1}=$5,303 ve C_{k2}=$2,845

Adım4: Mevcut çözümün temsil ettiği gün, ay, saat'de hava durumundan kaynaklı gecikme sayısı alınır.

Kısıt veri setinde bu çözümün

$\varphi(x_{ijk_{G.N.\,U\,A.N.\,U\,K.Z.}}) \mid h(V.G.Z. \,\&\, H.D.K.G.Z. \mid G.N.\,U\,A.N.\,U\,K.Z.)$ değeri 3 olarak gözükmektedir.

Yukarıda her bir adımda bulunan değerlere göre;

Amaç fonksiyonu=f=(((3*5303*6)+(3*2845*6))*0,25)=(95454+51210)*0,25= 146664*0,25=36666'dır.

2)Yeniden üretim operatörü

Popülasyon havuzu içerisinden eşleşmek üzere iki tane ebeveyn seçilir. Kromozomlar arasında seçim yaparken turnuva seçim yöntemi uygulanmıştır. Bu işlemin yapılış şekli, aşağıda belirtilen Adım 1 ve Adım 2 arasındaki ara basamaklarda hesaplanmıştır.

Adım 1:

Rastgele Oluşturulan 1.Ebeveyn = rand() % popülasyon büyüklüğü
Rastgele Oluşturulan 2.Ebeveyn = rand() % popülasyon büyüklüğü

Böylece popülasyon havuzunun ilk elemanından, son elemanına kadar yer alan tüm indisler içerisinden rastgele indis seçilerek, rastgele kromozom seçimi yapılmıştır.

Rastgele Oluşturulan 1.Ebeveyn=0 ve Rastgele Oluşturulan 2.Ebeveyn=1 olsun.

Adım 2:

Minimum(27499,5, 36666)= 27499,5'dir.

Seçilen iki kromozomdan minimum maliyette olan kromozom ele alınır ve minimum maliyete sahip ebeveyn, mevcut popülasyon havuzu ile aynı boyutta olan "yeni popülasyon havuzuna" aktarılır. İlk kromozomun maliyeti, ikinci kromozomun maliyetinden küçük olduğu için yeni popülasyona aktarılır.

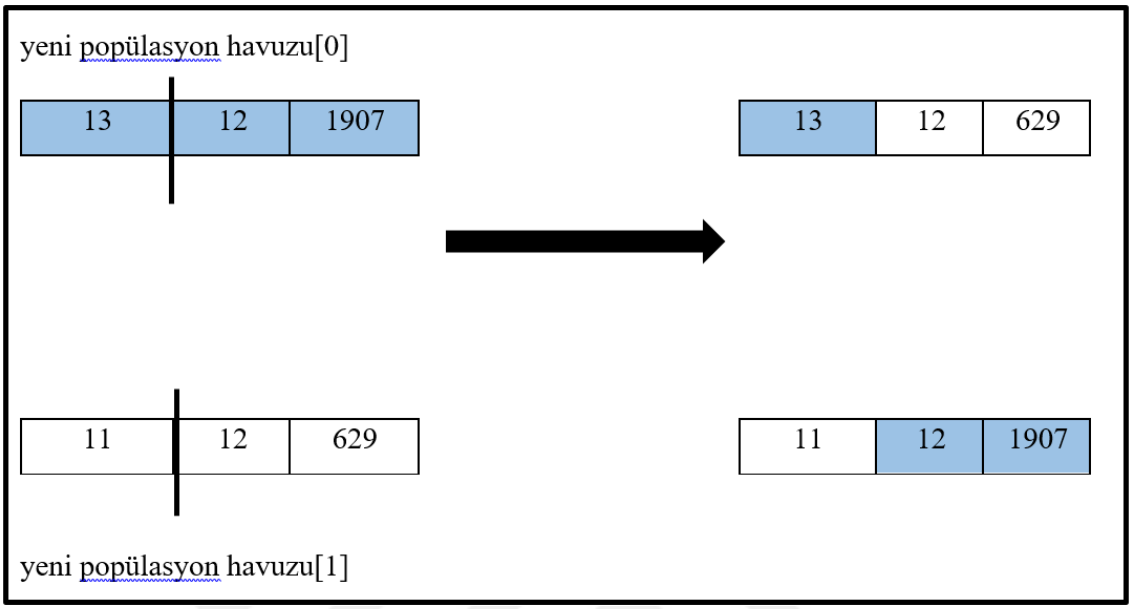
yeni popülasyon havuzu[0].first()= 13 12 1907
yeni popülasyon havuzu[0].second()=27499,5

3) Çaprazlama operatörü

Çaprazlama ile iyi kromozomların genleri birleştirilerek daha uygun kromozomlar oluşturulmaya çalışılır. 0 ile 1 arasında rastgele üretilen sayı, çaprazlama oranından küçük ise ebeveyn kromozomlar üzerinde tek noktalı çaprazlama yapılarak çocuk kromozomlar oluşturulur. Bu çalışmada çaprazlama noktası aşağıdaki yaklaşımla ele alınmıştır:

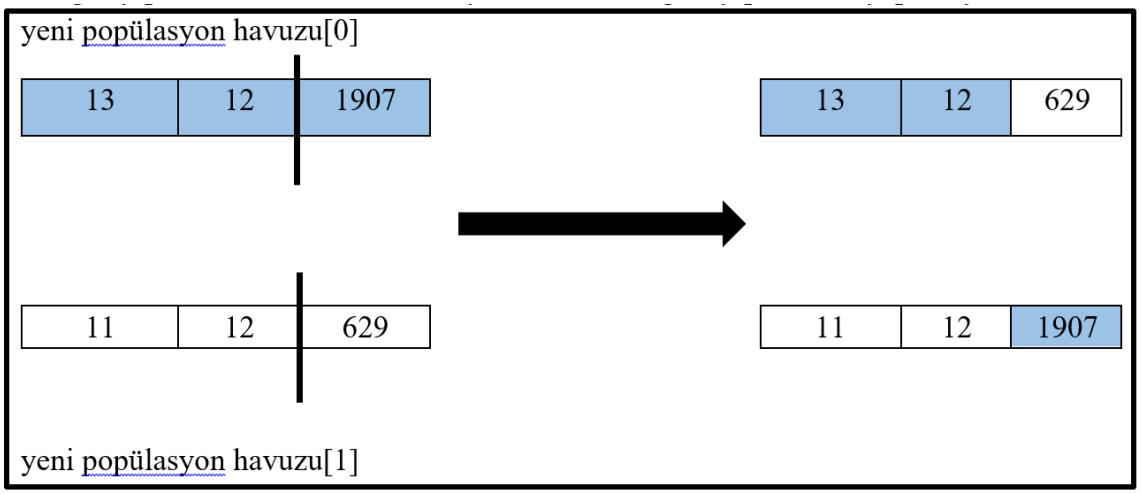
çaprazlama noktası = rand()%2

→Eğer çaprazlama noktası = 0 ise, Şekil 3.36'daki gibi çaprazlama yapılır:



Şekil 3.36. Mevcut kromozom durumları

→Eğer çaprazlama noktası = 1 ise, Şekil 3.37'daki gibi çaprazlama yapılır:



Şekil 3.37. Çaprazlama operatörü sonucu kromozom durumları

4) Mutasyon operatörü

Kromozomlar üzerinde mutasyon uygulanarak çeşitlilik sağlanıp bir müddet sonra tamamen aynı kromozomların oluşması önlenmiş olur. 0 ile 1 arasında rastgele üretilen sayı, mutasyon oranından küçük ise, kromozom üzerindeki gende mutasyon uygulanır. Bu çalışmada uygulanan mutasyon 2 farklı yaklaşımla ele alınmıştır. Bu yaklaşımlar aşağıda verildiği gibi ele alınacak olursa;

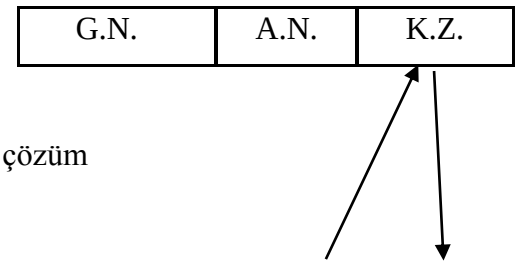
1. <u>Yaklaşım</u>

Popülasyonda rastgele seçilen kromozom ile mevcut kromozom üzerinde yer değiştirme işlemi yapılır. Yani, rastgele uçuş bilgisinde yer alan kalkış zamanı ile mevcut çözümün kalkış zamanı yer değiştirilir. Bu işlemin gerçekleştirilme şekli, Şekil 3.38, Şekil 3.39, Şekil 3.40, Şekil 3.41'de verilmiştir.

Rastgele indis= i olsun.

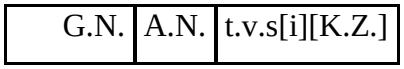
i=rand()%380

Mevcut Kromozom;



Şekil 3.38. GA mevcut çözüm

Rastgele Uçuş;



Şekil 3.39. Rastgele uçuş

Yer değiştirme işlemi sonucunda;

Mutasyona Uğrayan Kromozom;

G.N.	A.N.	t.v.s[i][K.Z.]

Şekil 3.40. Mutasyona uğrayan kromozom

Rastgele Uçuş;

G.N.	A.N.	K.Z.

Şekil 3.41. Rastgele uçuş

Örnek çözüm kümesi üzerinde komşu çözüm üretilecek olursa, Çizelge 3.25'deki gibi olur.

Çizelge 3.25. GA yerdeğiştirme işlemi ile mutasyona uğrayan çözüm

		Gün	Ay	Kalkış Saati	Rastgele Üretilen İndis
1	Mevcut Çözüm	13	12	19:07	i=11 olsun.
	Rastgele Uçuş	20	1	10.20	
	Yerdeğiştirme Sonucu				
2	Mutasyona Uğrayan Çözüm	13	12	10:20	
	Rastgele Uçuş	20	1	19:07	

2. Yaklaşım

Rastgele bir lokasyon belirlenir ve belirlenen global histograma göre yeni değerler verilir. Yani, bizim belirlediğimiz uçuşlar için, global sınır olan "5 dk ve 15 dk" kriterine göre kalkış zamanına ekleme yapılıp mevcut çözüm güncellenmiştir. Bu işlemin gerçekleştirilme şekli Şekil 3.42, Şekil 3.43, Şekil 3.44, Şekil 3.45'de verilmiştir.

[1-20] arası rastgele bir sayı üretilir.

→Eğer üretilen sayı 1 ile 10 arasında ise;

Mevcut Çözüm;

G.N.	A.N.	K.Z.

Şekil 3.42. GA mevcut çözüm

Mutasyona Uğramış Çözüm;

G.N.	A.N.	K.Z. + 5

Şekil 3.43. GA mutasyona uğramış çözüm

→ Eğer üretilen sayı 10 ile 20 arasında ise;

Mevcut Çözüm;

G.N.	A.N.	K.Z.

Şekil 3.44. GA mevcut çözüm

Mutasyona Uğramış Çözüm;

G.N.	A.N.	K.Z. + 15

Şekil 3.45. GA mutasyona uğramış çözüm

Örnek çözüm kümesi üzerinde komşu çözüm üretilecek olursa, Çizelge 3.26'daki gibi olur.

Çizelge 3.26. GA global histograma göre yeni değerle mutasyona uğrayan çözüm

		Gün	Ay	Kalkış Saati	Rastgele Üretilen Sayı
1	Mevcut Çözüm	13	12	19:07	1 ≤ Rastgele Üretilen Sayı ≤10
	Mutasyona Uğramış Çözüm	13	12	19:12	
2	Mevcut Çözüm	13	12	19:07	10 ≤ Rastgele Üretilen Sayı ≤20
	Mutasyona Uğramış Çözüm	13	12	19:22	

5) Yeni popülasyonda havuzundaki her bir bireyin(kromozomun) maliyet değeri hesaplanır.

6) Başlangıçtaki popülasyon havuzu ile yeni popülasyon havuzundaki her bir bireyin maliyet değeri karşılaştırılarak başlangıçtaki popülasyon havuzu güncellenir.

4. ARAŞTIRMA BULGULARI

Bulgular elde edilirken; uygulama aşaması, C++ dilinde kodlanarak 2.60 GHz CPU, 8.00 GB RAM özelliklere sahip Intel(R) Core(TM) i7-6700HQ özelliklere sahip bilgisayarda gerçekleştirilmiştir.

Parametrelerin uygun seçimi, optimizasyon algoritmalarının performansı için çok önemlidir. YAKA performans analizi için, gerçekleştirilen deney sonuçları Çizelge 4.1 ve Çizelge 4.2'de verilmiştir. BT performans analizi için, gerçekleştirilen deney sonuçları Çizelge 4.3'de verilmiştir. GA performans analizi için, gerçekleştirilen deney sonuçları Çizelge 4.4'de verilmiştir.

Çizelge 4.1, Çizelge 4.2, Çizelge 4.3 ve Çizelge 4.4'de yapılan deneyler, **EK 1**'de verilen birinci tarife bilgilerini içeren uçuşlar için yapılmıştır.

4.1. YAKA Performans Analizi

YAKA Algoritmasının Parametreleri:

- İterasyon sayısının belirlenmesi
- Limit değerinin belirlenmesi
- Popülasyon büyüklüğünün belirlenmesi

Çizelge 4.1 ve Çizelge 4.2'de herbir deneydeki, P.B. (popülasyon büyüklüğü), İ.S. (iterasyon sayısı), L. (limit) parametrelerinin aldığı değerlere göre, ortalama değer, optimum değer, E.K.D. (En Kötü Değer) sonuçları verilmiştir.

Çizelge 4.1. YAKA parametreleri ile 1. yaklaşıma göre yapılan deneyler

Komşuluk Yapısı → ***1.Yaklaşım***						
Parametreler				Popülasyon Büyüklüğü Deneyi		
Deney	P.B.	İ.S.	L.	Ortalama Değer	Optimum Değer	E.K.D.
1	80	40	150	7,20905e+008	6,27483e+008	7,49238e+008
2	**380**	**40**	**150**	8,21625e+008	**7,72579e+008**	8,73381e+008
				İ.S. Deneyi		
1	380	5	20	5,19017e+008	3,53776e+008	6,79904e+008
2	380	20	20	1,49779e+009	1,24506e+009	1,69582e+009
3	380	50	20	3,39082e+009	3,21685e+009	3,53685e+009
4	380	150	20	9,55626e+009	9,03765e+009	1,02055e+010
5	380	30	150	6,13280e+008	4,21683e+008	7,72064e+008
6	380	40	150	9,08458e+008	6,64243e+008	1,09456e+009
7	380	50	150	9,40877e+008	7,10937e+008	1,14183e+009
				Limit Deneyi		
1	380	20	10	2,07228e+009	1,75698e+009	2,35304e+009
2	380	20	100	5,16992e+008	3,62511e+008	7,02773e+008
3	380	20	150	4,34545e+008	3,46915e+008	5,99986e+008
4	380	20	190	4,87108e+008	3,66353e+008	7,33971e+008
5	380	50	10	4,61998e+009	4,49947e+009	4,82450e+009
6	380	50	20	3,79746e+009	2,80504e+009	4,45215e+009
7	380	50	40	1,65199e+009	1,07994e+009	2,67728e+009

Çizelge 4.2. YAKA parametreleri ile 2. yaklaşıma göre yapılan deneyler

Komşuluk Yapısı → ***2.Yaklaşım***						
Parametreler				Popülasyon Büyüklüğü Deneyi		
Deney	P.B.	İ.S.	L.	Ortalama Değer	Optimum Değer	E.K.D
1	80	40	150	1,41698e+008	9,61857e+007	1,83633e+008
2	**380**	**40**	**150**	1,49463e+008	**1,06676e+008**	1,75179e+008
				İ.S. Deneyi		
1	380	5	20	4,72265e+007	2,28304e+007	7,86542e+007
2	380	20	20	1,73846e+008	1,18614e+008	1,92647e+008
3	380	50	20	3,08523e+008	2,67827e+008	3,97249e+008
4	380	150	20	1,47155e+009	2,63098e+008	4,32956e+009
5	380	30	150	1,45059e+009	1,05581e+009	1,73348e+009
6	380	40	150	1,92594e+009	1,77182e+009	2,03304e+009
7	380	50	150	2,07151e+009	1,90137e+009	2,25157e+009
				Limit Deneyi		
1	380	20	10	2,05028e+009	1,70617e+009	2,26069e+009
2	380	20	100	1,10546e+009	1,00897e+009	1,17151e+009
3	380	20	150	9,62306e+008	6,91716e+008	1,27711e+009
4	380	20	190	9,83105e+008	8,01554e+008	1,15858e+009
5	380	50	10	5,27539e+009	4,77198e+009	5,70149e+009
6	380	50	20	4,10872e+009	3,80777e+009	4,42690e+009
7	380	50	40	3,19232e+009	2,93319e+009	3,50683e+009

4.2. BT Performans Analizi

BT Algoritmasının Parametreleri:

*Başlangıç sıcaklığının belirlenmesi

Bu çalışmada T=10000.0 alınmıştır.

*Soğutma Oranı
Soğutma(tavlama) oranı =0.98 alınmıştır.

*Sıcaklık Değiştirme Kuralının Belirlenmesi
T = T * Soğutma(tavlama) Oranı

*Her Sıcaklıkta Gerçekleştirilecek İterasyon Sayısının Belirlenmesi (Her sıcaklıkta kaç kere pertürbasyon işleminin yapılacağı)
S = 100 alınmıştır.

*Durdurma Kriterinin Belirlenmesi(Donma Sıcaklığı)
T_d = 0.2 alınmıştır.

Çizelge 4.3. BT parametreleri ile yapılan deneyler

BT ALGORİTMASI				
Komşuluk Yapısı	Parametreler	Ortalama Değer	Optimum Değer	E.K.D.
1.Yaklaşım	T=10000.0 Soğutma(tavlama) Oranı =0.98 S = 100 T_d = 0.2	4,17992e+007	**4,17992e+007**	4,17992e+007
2.Yaklaşım		4,17992e+007	**4,17992e+007**	4,17992e+007

4.3. GA Performans Analizi

GA Algoritmasının Parametreleri:

- Popülasyon Büyüklüğü
- Çaprazlama Oranı
- Mutasyon Oranı
- Durdurma Kriteri (Kuşak Sayısı)

Çizelge 4.4. GA parametreleri ile yapılan deneyler

GA ALGORİTMASI				
Mutasyon	Parametreler	Ortalama Değer	Optimum Değer	E.K.D.
1.Yaklaşım	Popülasyon Büyüklüğü: 380 Çaprazlama Oranı: 0.75 Mutasyon Oranı: 0.01 Durdurma Kriteri:100	8,14467e+008	**5,64519e+008**	1,06473e+009
2.Yaklaşım		8,78256e+008	**8,15650e+008**	9,56662e+008

4.4. YAKA, BT ve GA Performans Analiz Karşılaştırmaları

Çizelge 4.5'de, 3 tarife için de metasezgisel algoritmaların çalışma sürelerinin karşılaştırması verilmiştir. Bu sonuçlar elde edilirken;

1) YAKA algoritması için, Çizelge 4.1 ve Çizelge 4.2'de optimum ya da optimuma yakın sonuca ulaşılmasını sağlayan parametre değerleri dikkate alınmıştır. "P.B." değeri 380, "İ.S." değeri 40 ve "L." değeri 150 olarak alınmıştır.

2) BT algoritması için, Çizelge 4.3'de optimum ya da optimuma yakın sonuca ulaşılmasını sağlayan parametre değerleri dikkate alınmıştır. "T" değeri 10000.0, "soğutma oranı" 0.98, "S" değeri 100 ve "T_d" 0.2 olarak alınmıştır.

3) GA algoritması için, Çizelge 4.4'de optimum ya da optimuma yakın sonuca ulaşılmasını sağlayan parametre değerleri dikkate alınmıştır. "kuşak sayısı" 100, "çaprazlama oranı" 0,75, "mutasyon oranı" 0,01 ve "durdurma kriteri" 100 olarak alınmıştır.

Çizelge 4.5. BT, YAKA ve GA çalışma süreleri

BT Algoritması			
	1.Tarife Optimum Sonuca Ulaşma Süresi	**2.Tarife Optimum Sonuca Ulaşma Süresi**	**3.Tarife Optimum Sonuca Ulaşma Süresi**
1.Yaklaşım	5 saat 8 dakika	7 saat 8 dakika	5 saat 2 dakika
2.Yaklaşım	2 saat 44 dakika	2 saat 21 dakika	**2 saat 4 dakika**
GA Algoritması			
	1.Tarife Optimum Sonuca Ulaşma Süresi	**2.Tarife Optimum Sonuca Ulaşma Süresi**	**3.Tarife Optimum Sonuca Ulaşma Süresi**
1.Yaklaşım	2 saat 57 dakika	2 saat 49 dakika	**2 saat 31 dakika**
2.Yaklaşım	3 saat 26 dakika	2 saat 25 dakika	1 saat 9 dakika
YAKA Algoritması			
	1.Tarife Optimum Sonuca Ulaşma Süresi	**2.Tarife Optimum Sonuca Ulaşma Süresi**	**3.Tarife Optimum Sonuca Ulaşma Süresi**
1.Yaklaşım	12 saat 1 dakika	12 saat 19 dakika	11 saat 53 dakika
2.Yaklaşım	**10 saat 47 dakika**	11 saat 3 dakika	12 saat 10 dakika

Çizelge 4.6'da, 3 tarife için de metasezgisel algoritmaların bulduğu, optimum ya da optimuma yakın sonuçlar verilmiştir.

Çizelge 4.6. 3 tarife için elde edilen optimum sonuçlar

BT Algoritması			
	1.Tarife Optimum Değer	**2.Tarife Optimum Değer**	**3.Tarife Optimum Değer**
1.Yaklaşım	4,17992e+007	4,17992e+007	4,17992e+007
2.Yaklaşım	4,17992e+007	4,17992e+007	4,17992e+007
GA Algoritması			
	1.Tarife Optimum Değer	**2.Tarife Optimum Değer**	**3.Tarife Optimum Değer**
1.Yaklaşım	7.34137e+008	1.36446e+009	1.0624e+009
2.Yaklaşım	9.80166e+008	1.3894e+009	9.49191e+008
YAKA Algoritması			
	1.Tarife Optimum Değer	**2.Tarife Optimum Değer**	**3.Tarife Optimum Değer**
1.Yaklaşım	3.52326e+009	3.26832e+009	3.40262e+009
2.Yaklaşım	2.61343e+009	2.89292e+009	6.6762e+009

4.5. Klasik Yöntem İle Elde Edilen Analiz Sonucu

Çizelge 4.7'de, 1. tarife için klasik yöntem ile optimum sonuca ulaşılma süresi ve optimum maliyet bulunmuştur.

Çizelge 4.7. Klasik yöntem optimum sonuç ve çalışma süresi

Klasik Yöntem		
	Optimum Sonuca Ulaşma Süresi	**Optimum Değer**
1. Tarife	57 saat 2 dakika	4,17992e+007

5. TARTIŞMA ve SONUÇ

5.1. Tartışma

Bu çalışma ile büyük veri teknolojisi olan Apache Spark kullanarak, 32 yıllık uçuş verisini hava koşulları ile beraber dikkate alan; YAKA, GA, BT sezgisel algoritmaları ile uçuş çizelgeleme probleminin optimizasyonu literatürde (bildiğimiz kadarıyla) ilk defa yapılmış oldu.

5.2. Sonuç

Bu çalışmada, 1987-2018 yılları arasındaki geçmiş uçuş verileri üzerinde, büyük veri analiz etme teknolojilerinden olan Apache Spark kullanılarak kısıtlar bulunmuş ve bu şans kısıtları yardımıyla BT, YAKA ve GA ile uçuş çizelgeleme problemi optimize edilmeye çalışılmıştır.

Spark'ın bellek-içi veri işleme açısından yerel yöntemlere göre sağladığı avantaj, MySQL veri tabanını yönetmek için kurulan, MySQL Workbench ve PhpMyAdmin veri tabanı ara yüzlerinde herhangi bir yıla ait veri seti yüklenmeye çalışıldığında bellek hatası alınması sonucu çok net görülmüştür. 32 farklı büyük veriseti üzerinde Apache Spark, SparkSQL kütüphanesinin esnek, hızlı şekilde güçlü bir araç olarak kullanılabildiği gözlemlenmiştir. Spark'ın zaman ve performans açısından güçlü bir teknoloji olduğu tespit edilmiştir. Bu güçlü teknolojinin python API'si olan pySpark aracı kullanılarak, Spark makinesine (motoruna) yani ana URL'ye bağlanabilmek için; "local" kip ortamında, "local[*]" parametresi seçilmiş ve tüm çalışmalar sürdürülmüştür.

Sezgisel algoritmalar rastgele çözüm kümesi ile optimizasyon sürecini başlatır ve bu rastgele çözüm her adımda iyileştirilmeye çalışılır. Çalışmada kullanılan algoritmalar stokastik tabanlı optimizasyon algoritmaları olduğu için tek adımda en iyi çözümü bulamama ihtimali yüksektir. Uygun girilen parametre değerlerine göre algoritmalar

çalıştırıldığında optimum sonuca ya da optimum sonuca yakın değerlere ulaşma ihtimali yükselir. Bu nedenle, Çizelge 4.1, Çizelge 4.2, Çizelge 4.3, Çizelge 4.4 deneyleri yapılırken, her bir deney 5 kez çalıştırılmıştır.

Bu deney sonuçlarına bakılacak olursa;

Çizelge 4.1 ve Çizelge 4.2'deki popülasyon büyüklüğü deneyi incelendiğinde;

1) Popülasyon büyüklüğü çok küçük olursa; çözümün kalitesi bozulmuştur, çok büyük olursa da arama uzayında taranan alan büyük olduğu için sürenin uzamaya başladığı programın çalışması esnasında görülmüştür.
2) Genel olarak iterasyon sayısı artırıldığında optimum sonuca ulaşma süresinin uzadığı çok net görülmüştür.

Çizelge 4.1'de, komşuluk yapısının 1.yaklaşıma göre ele alındığı deneyde (yerdeğiştirme işlemine göre komşuluk yapısının belirlendiği deneyde):

1) İ.S. deneyine bakıldığında;

Deney 1-4 arasında, (limit değeri 20 alınmıştır), iterasyon sayısının artırılmasının maliyetin daha optimum olması konusunda etkisi olmamıştır. 5-7 arasında, (limit değeri 150 alınmıştır), iterasyon sayısının artırılmasının yine maliyetin daha optimum sonuca ulaşması konusunda etkisi olmamıştır. 1-7 arasındaki sonuçlar incelendiğinde, limit 150 iken iterasyon sayısındaki artış ile elde edilen sonuçların, limit 20 iken iterasyon sayısındaki artış ile elde edilen sonuçlara göre daha az maliyette sonuç verdiği görülmektedir. Buradan çıkarılabilecek yorum, limit değeri arttıkça çözüm kümesi üzerindeki başarısızlık oranının limit değerine ulaşma ihtimalinin daha düşük olacağı şeklindedir. Dolayısıyla yeni çözüm kümesi oluşturma ihtimali daha az olur, mevcut çözüm kümesi üzerindeki iyileştirme durumu artar ve maliyet minimuma yaklaşır. Bu yaklaşma durumu iterasyon sayısına bağlı olarak değiştiği için iterasyon sayısının sürekli artırılması bir müddet sonra daha iyi çözüm kümesi bulunmasına mani olabilir. Bunun

nedeni limit 150 iken düşük iterasyon değerleri için; başarısızlık durumunun limit değerine ulaşması güçleşir, ama iterasyon değeri artınca daha iyi bir çözüm bulunamadıkça başarısızlık ihtimali de artar, böylelikle yeni bir çözüm kümesi oluşturulmuş olur ve belli bir noktadan sonra ise daha optimum sonuç bulunamayabilir.

Çizelge 4.2'de, komşuluk yapısının 2.yaklaşıma göre ele alındığı deneyde (global histograma göre elde edilen yeni değerlerle komşuluk yapısının belirlendiği deney):

1) 2.yaklaşıma göre İ.S. deneyine bakıldığında;

Deney 1-4 arasında, (limit değeri 20 alınmıştır), iterasyon sayısının artırılmasının maliyetin daha optimum olması konusunda etkisi olmamıştır. 5-7 arasında, (limit değeri 150 alınmıştır), iterasyon sayısının artırılmasının yine maliyetin daha optimum sonuca ulaşması konusunda etkisi olmamıştır. 1-7 arasındaki sonuçlar incelendiğinde, limit 20 iken iterasyon sayısındaki artış ile elde edilen sonuçların, limit 150 iken iterasyon sayısındaki artış ile elde edilen sonuçlara göre daha az maliyette sonuç verdiği görülmektedir. 1. yaklaşıma göre buradaki sonucun farklı olmasının nedeni, rastgele komşu üretme durumu değiştiği için başarısızlık durumuna düşme ihtimali kaynaklı olduğu düşünülmektedir.

Çizelge 4.1 ve Çizelge 4.2 için;

1) 1. ve 2. yaklaşıma göre "limit" deneyine bakıldığında; limit, iterasyon sayısından çok küçük olunca maliyet olumsuz etkilenmektedir (1 ve 5 numaralı deneyler). Limit, iterasyon sayısından ne kadar küçük olursa maliyet o kadar olumsuz etkilenir. Bu duruma deney sonuçları üzerinden örnek verilecek olursa; 5 numaralı deneyin sonuçları 1 numaralı deneye göre daha kötüdür. Limit değeri, iterasyon değerinden yaklaşık %20 daha az olunca daha iyi sonuçlar alındığı gözlemlenmiştir (7 numaralı deney). 6 ve 7 numaralı deneylerde görüldüğü gibi limit değeri iterasyon değerine yaklaştıkça maliyette iyileşme görülmüştür. En optimum sonuçlara, 2-4 arası deney sonuçlarından da görülebileceği üzere, limit değeri iterasyon değerinden 5 ya da 7 kat büyük olunca

ulaşılabileceği görülmüştür. 2-4 arası deney sonuçları kendi içerisinde incelendiğinde ise, 2 numaralıdan 3 numaralıya geçerken maliyetin düştüğü, ama 3 numaralıdan 4 numaralıya geçerken maliyetin arttığı gözlemlenmiştir. Buradan çıkarılabilecek sonuç, limit iterasyon sayısından en fazla 7 kat büyük olunca optimum sonuca daha çok yaklaşılmaktadır. 4 numaralı deney sonuçlarında da görülebileceği gibi, limit iterasyon sayısından 7 kattan daha büyük olunca maliyet 3 numaralı deney sonucuna göre olumsuz etkilenmiştir. Bu durumda optimum sonuçtan yavaş yavaş uzaklaşılmaya başlandığı gözlemlenmiştir.

1.yaklaşım ve 2.yaklaşım sonuçları genel olarak incelendiğinde, 2.yaklaşımda daha optimum sonuca ulaşıldığı görülmüştür. Bunun sebebi, rastgele komşu üretme durumu değiştiği için en iyi çözüme ulaşma ihtimalinin artmasıdır. Şöyle ki; uçak kalkış saatlerine 5 ya da 15 dk eklenerek yeni tarife oluşturulur. Bu tarifenin kısıt verisetinde olma ihtimali daha düşük olur ve maliyeti minimum yapacak çözüm üretilmiş olur.

BT performans analizi için; Çizelge 4.3 incelendiğinde, başlangıç sıcaklığı ne kadar yüksek olursa arama uzayındaki taranacak genişlik o kadar büyük olduğundan global optimuma daha çok yaklaşmamız mümkün olur. Bu sebeple çalışma içerisinde, bu sıcaklık değeri 10000 °C alınmıştır, BT'de her seferinde sıcaklık genel olarak 0,8 °C ve 0,99 °C arasında değişen değerler oranında azaltılır. Her bir sıcaklık için belirlenen döngü miktarına ulaşıldığında, sıcaklık değiştirme kuralı uygulanarak sıcaklığın düşürülmesi sağlanır, iterasyon sayısı 100 ve donma noktası 0.2 °C alındığında çalışmanın daha iyi sonuçlar verdiği gözlemlenmiştir. Elde edilen sonuçlarda hem 1. hem de 2. komşuluk yapısı ile "4.17992e+007" sonucu elde edilmiş ve optimum değer olarak saptanmıştır. BT'de rastgele oluşturulan çözüm kümesi "100, 380" gibi farklı boyut değerleri için test edilmiştir. Elde edilen sonuçlarda optimum sonuca ulaşılabildiği net olarak görülmüştür.

GA performans analizi için; Çizelge 4.4 incelendiğinde, bu algoritmada çaprazlama oranı %75 olarak seçilmiştir. Genelde mutasyon oranı %1 ile %5 arasında değişmektedir. Ancak, bu çalışmada %1 olarak belirlenmiştir. Kuşak sayısı çok küçük seçilirse, sonuç yerel minimuma düşebilir, böyle bir durumda global minimuma ulaşamama ihtimali

vardır. Bu çalışmada iyileşmenin görüldüğü kuşak sayısı 100 olarak tespit edilmiştir. Elde edilen sonuçlarda, 1. yaklaşıma göre mutasyon uygulanması sonucu elde edilen bulguların, 2. yaklaşıma göre daha optimum maliyette sonuç verdiği gözlemlenmiştir.

Çizelge 4.3'de elde edilen sonuçlara göre, BT sezgisel algoritmasının diğer iki metasezgisel optimizasyon algoritmasına göre daha başarılı sonuçlar verdiği ve GA algoritmasının da YAKA algoritmasına göre daha optimum sonuç verdiği gözlemlenmiştir.

Çizelge 4.5'de elde edilen sonuçlara göre, BT'nin optimum sonuca, en kısa sürede ulaşan metasezgisel algoritma olduğu görülmüştür. En uzun sürede çalışan algoritmanın ise, YAKA olduğu tespit edilmiştir.

Çizelge 4.6'da bulunan optimum değerlerin, Çizelge 4.1, Çizelge 4.2 ve Çizelge 4.3'deki sonuçlarla uyumlu olduğu görülmüştür.

Çizelge 4.7'deki sonuç incelendiğinde, tarifenin klasik yöntem ile optimum sonuca göre şekillendirilme süresinin ne kadar uzun olabileceği görülmüştür. Klasik yöntem ile tek bir tarifenin optimum sonuca ulaşma süresi 9 dakika olarak ölçülmüştür. Çizelge 4.5'e göre, BT'nin ikinci yaklaşımına göre, tek bir tarifenin optimum sonuca ulaşma süresi 19,5 saniyedir. GA'nın birinci yaklaşıma göre, tek bir tarifenin optimum sonuca ulaşma süresi 23,8 saniyedir. YAKA ikinci yaklaşımına göre, tek bir tarifenin optimum sonuca ulaşma süresi 1,7 dakikadır. Buradan elde edilen sonuçlara göre metasezgisel yaklaşımlar ile elde edilen değerlerin, Çizelge 4.7'de bulunan değere göre, çalışma süresini önemli ölçüde düşürdüğü çok net görülmüştür.

Çalışma kapsamında büyük veri teknolojilerinden Apache Spark yardımı ile hava durumundan kaynaklı yüksek olasılıklı gecikmelerin olabileceği kısıtlar bulunmuş ve bu kısıtların uçuş çizelgeleme probleminin çözümü için modele nasıl eklendiği gösterilmiştir. Bu modelden, kullanılacak metasezgisel algoritmanın uygunluk formülü içerisinde faydalanılmıştır. Hangi metasezgisel algoritmanın bu problemin çözümü için

en uygun sonucu verdiği ilgili deneyler yardımı ile tespit edilmiştir. Benzetilmiş tavlama yöntemi ile minimum maliyet sağlanacak şekilde uçuş tarifelerinin yeniden şekillendirilebileceği net olarak görülmüştür.

KAYNAKLAR

Abdelghany, K.F., Abdelghany, A.F. and Ekollu, G., 2008. An integrated decision support tool for airlines schedule recovery during irregular operations. European Journal of Operational Research, 185 (2), 825-848.

Adachi, N., Sato, M. and Kobayashi, S., 2004. Application of Genetic Algorithm to Flight Schedule Planning. Systems and Computers in Japan, 35 (12), 83-92.

Andersson, T., 2006. Solving the flight perturbation problem with meta heuristics. Journal of Heuristics, 12 (1-2), 37-53.

Anonymous., 1992. Bureau of Transportation Statistics. https://www.transtats.bts.gov/DL_SelectFields.asp?Table_ID=236 (16.12.2017).

Anonymous., 2015. Apache Spark. https://spark.apache.org/ (1.09.2018).

Anonymous., 2015. Flight routes and linear programming 1. https://wiki.ubc.ca/Flight_routes_and_linear_programming_1#Flight_Time (1.09.2018).

Anonymous., 2017. Directed Acyclic Graph DAG in Apache Spark. https://data-flair.training/blogs/dag-in-apache-spark/ (29.11.2018).

Anonymous., 2018. In-memory database. https://en.wikipedia.org/wiki/In-memory_database (5.12.2018).

Feo, T. A. and Bard, J. F., 1989. Flight scheduling and maintenance base planning. Management Science, 35(12), 1415-1432.

GAO, Q., TANG, X.-W. and ZHU, J.-F., 2009. Research on Greedy Simulated Annealing Algorithm for Irregular Flight Schedule Recovery Model. IEEE International Conference on Grey Systems and Intelligent Services, Nanjing, China.

Holland, J.H., 1992. Genetic algorithms. Scientific american, 267 (1), 66-73.

Jarrah, A. I., Yu, G., Krishnamurthy, N. and Rakshit, A., 1993. A decision support framework for airline flight cancellations and delays. Transportation Science, 27(3), 266-280.

Jungai, T. and Hongjun, X., 2012. Optimizing Arrival Flight Delay Scheduling Based on Simulated Annealing Algorithm. 2012 International Conference on Medical Physics and Biomedical Engineering, China.

Kane, F., 2017. Spark Basics and Spark Examples. Frank Kane's Taming Big Data with Apache Spark and Python, Packt Publishing. 59-69.

Kane, F., 2017. Spark Basics and Spark Examples. Frank Kane's Taming Big Data with Apache Spark and Python, Packt Publishing. 70-79.

Karaboga, D. and Akay, B., 2007. Artificial bee colony (ABC) algorithm on training artificial neural networks. IEEE 15th Signal Processing and Communications Applications Conf., Eskisehir, Turkey.

Karaboga, D. and Akay, B., 2009. A comparative study of Artificial Bee Colony algorithm. Applied Mathematics and Computation, 214 (1), 108-132.

Kirkpatrick, S., Gelatt, C.D. and Vecchi, M.P.,1983. Optimization by simulated annealing. Science, 220 (4598), 671-680.

Küçüksille, E.U. ve Tokmak, M., 2011. Yapay arı kolonisi algoritması kullanarak otomatik ders çizelgeleme. Süleyman Demirel Üniversitesi Fen Bilimleri Enstitüsü Dergisi, 15 (3), 203-210.

Laskowski, J., Anonymous. Spark local (pseudo-cluster). https://jaceklaskowski.gitbooks.io/mastering-apache-spark/spark-local.html (15.12.2018).

Li, J., Liang, W. and Li, Y., 2014. Research on Flight Scheduling of Two Runways Based on the Fusion of Artificial Fish School Algorithm and Genetic Algorithm. IEEE International Conference on Software Engineering and Service Sciences, Beijing, China.

Liang, W. and Li, Y., 2014. Research on Optimization of Flight Scheduling Problem Based on the Combination of Ant Colony Optimization and Genetic Algorithm. IEEE International Conference on Software Engineering and Service Sciences, Beijing, China.

Liu, T.-K., Chen, C.-H. and Chou, J.-H., 2010. Optimization of short-haul aircraft schedule recovery problems using a hybrid multiobjective genetic algorithm. Expert Systems with Applications, 37 (3), 2307-2315.

Mckenzie, K., 2002. Flight Schedules For Airlines Worldwide. https://www.passrider.com/reservations/advanced-search/ (11.09.2018).

Novianingsih, K. and Hadianti, R., 2014. Modeling Flight Departure Delay Distributions. International Conference on Computer, Control, Informatics and Its Applications, Bandung, Indonesia.

Orhan, İ., Kapanoğlu, M. and Karakoç, T. H., 2010. Havayolu Operasyonlarında Planlama ve Çizelgeleme. Pamukkale University Journal of Engineering Sciences, 16(2).

Penchikala, S., 2018. Big data processing with apache spark. Lulu. Com. 10-12. Portilla, J., Anonymous. Spark and Python for Big Data with Pyspark. https://www.udemy.com/spark-and-python-for-big-data-with-pyspark (5.10.2018).

Rosen, J., 2016. PySpark Internals. https://cwiki.apache.org/confluence/display/SPARK/PySpark+Internals (7.12.2018).

Rouse, M., 2015. in-memory analytics. https://searchbusinessanalytics.techtarget.com/definition/in-memory-analytics (4.12.2018).

Sternberg, A., Carvalho, D., Murta, L., Soares, J. and Ogasawara, E., 2016. An analysis of Brazilian flight delays based on frequent patterns. Transportation Research Part E: Logistics and Transportation Review, 95, 282-298.

Wong, J.-T. and Tsai, S.-C., 2012. A survival model for flight delay propagation. Journal of Air Transport Management, 23, 5-11.

Wu, C. L. and Caves, R. E., 2003. Flight schedule punctuality control and management: a stochastic approach. Transportation planning and technology, 26(4), 313-330.

Xiang, Z. and Tang, Y., 2014. Optimization for Scheduling Arrival Flights at Airport Based on Airline's Profit. IEEE Workshop on Advanced Research and Technology in Industry Applications, Ottawa, ON, Canada.

Yao, X., Liu, Y., and Lin, G., 1999. Evolutionary programming made faster. IEEE Transactions on Evolutionary computation, 3(2), 82-102.

ÖZGEÇMİŞ

Ebru ERDEM 1994 yılında Erzurum'da doğdu. İlk, orta ve lise öğrenimini Erzurum'da tamamladı. 2011-2015 yılları arasında Atatürk Üniversitesi Bilgisayar Mühendisliği bölümünde lisans derecesini aldı. Mezuniyetinin ardından 2015 yılında Atatürk Üniversitesi'nde yüksek lisans eğitimine başlamıştır.