ONDOKUZ MAYIS ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

UÇAKLARA KAPILARIN ATANMASI VE YOLCU BAGAJ SIRALAMA İSTASYONU PROBLEMİ PROJE TASARIM RAPORU

ESRA BOZOK

ELİF HAZİNEDAR

ZEYNEP SARE PINAR

EMİNE İSKENDEROĞLU

Danışman

Dr. Öğr. DURMUŞ ÖZKAN ŞAHİN

OCAK, 2023

TEŞEKKÜR

Bu çalışmada bize yardımını esirgemeyen, öneri ve paylaşımlarıyla destek veren saygıdeğer Dr. Öğretim Üyesi Durmuş Özkan Şahin‘e ve aynı zamanda bize, bilgisayar biliminin mantığını oluşturan algoritmik düşünebilmeyi öğreten tüm hocalarımıza teşekkür ederiz.

ÖZET

Hava taşımacılığındaki artan trafik talebi, havaalanlarının sürekli gelişimini zorunlu hale getirmektedir. Bu nedenle, havaalanlarında işletme iyileştirme çalışmaları önem kazanmıştır. Havayolu işletmelerinin müşteri beklentilerini kapsamlı bir şekilde analiz etmeleri ve bu sonuçlara göre cevap verebilmeleri doğrudan havayolu işletmelerinin piyasadaki üstünlüğünü sağlamasına etki eder. Havaalanı performansını etkileyen iki önemli husus vardır. İlki, uçak kapı atama problemidir. Bu, uçakların hangi kapılardan iniş ve kalkış yapması gerektiğini belirlemek için çözülmesi gereken bir problemdir. Diğer önemli bir husus ise bagaj sıralama istasyonu atama problemidir. Yanlış işlenen bagajlar, itibar kaybına ve yolcu memnuniyetsizliğine neden olmaktadır. Bagaj sıralama istasyonu atama problemi, birden çok kısıt dikkate alınarak bagajların uçağa ulaştırılması işlemidir. Bu çalışmada, kapı ve bagaj atama problemlerine yönelik en uygun algoritma seçimi için gerekli literatür taraması yapılmış ve çözüm önerileri incelenmiştir. Ayrıca, uçak bilet alım web uygulaması için gerekli çalışmalara da başlanmıştır. Bu çalışma, hava taşımacılığı alanındaki işletmeler için önemli bir adım olarak kabul edilebilir çünkü kapı ve bagaj atama problemlerinin etkili bir şekilde çözülmesi, havaalanı performansının iyileştirilmesine ve müşteri memnuniyetinin artmasına katkıda bulunabilir.

Anahtar Kelimeler: Havaalanı yönetimi, uçak kapı ataması, bagaj sıralama istasyonu ataması, algoritmalar.

ABSTRACT

The increasing demand for air transportation has made the continuous development of airports necessary. As a result, improvement of management efforts in airports has become important. Airline companies analyzing customer expectations in a comprehensive manner and being able to respond accordingly directly affect the superiority of the airline companies in the market. There are two important factors that affect airport performance. The first one is the problem of airplane gate assignment, which is a problem that needs to be solved to determine which gates the airplanes should land and depart from. The other important factor is the baggage sorting station assignment problem. Incorrectly handled baggage can cause loss of money and reputation, as well as dissatisfaction among passengers. The baggage sorting station assignment problem is a process of transporting baggage to the airplane while considering multiple constraints. In this study, a necessary literature search was conducted to select the most appropriate algorithm for the gate and baggage assignment problems and a solution proposal was presented. Additionally, necessary studies have been started for an airplane ticket purchase web application. This study can be considered as an important step for companies in the field of air transportation, because resolving the gate and baggage assignment problems effectively can contribute to improving airport performance and increasing customer satisfaction.

Keywords: Airport management, airplane gate assignment, baggage sorting station assignment, algorithms.

İÇİNDEKİLER

ŞEKİLLER LİSTESİ

ÇİZELGELER LİSTESİ

BÖLÜM: I

GİRİŞ

1

AMAÇ

Bu proje kapsamında amaç, incelenen havaalanı kapı ve bagaj problemlerine yönelik çalışmalardan yararlanarak uçak bilet alımı, yolcu check-in'i, havaalanı uçak kapıları ve yolcu bagajlarının atanmasına yönelik bir web uygulaması simülasyonu oluşturmaktır. Uçakların havaalanı kapı ataması problemi için aprona atanan uçuşların ve toplam yürüme mesafesinin en aza indirilmesine yönelik bir çözüm yaklaşımı sunulması amaçlanmıştır. Bunun yanında bagajların yanlış işlenmesi müşteri memnuniyetini etkileyen en önemli faktörlerden biridir. Bagajların uçak kalkış öncesinde uçağa aktarılmasını aynı zamanda sıralama istasyonlarında çakışma olmamasını sağlamak ve gerçek zamanlı verilere dayanarak çıkan atama sonuçlarının gerçeğe yakın olmasının sağlanması önem arz etmektedir. Bu atama problemleri için en uygun algoritma tercihleri için gerekli incelemeler yapılmıştır. İncelemeler sonucunda müşteri memnuniyetinin arttırılması, maliyetin azaltılması amaçlanmıştır. Bu proje için motivasyonumuzu, problemin çözümüne yönelik çeşitli algoritmaları incelemek ve bu doğrultuda güncel olarak varlığını sürdüren uçak kapı ve bagaj atamaları problemine yönelik alternatif bir çözüm önerisi sunabilmemiz hedeflenmiştir.

2

LİTERATÜR ÖZETİ

Havaalanları, yolculuk etmek için kullanılan en yaygın yerlerdir ve bu nedenle, havaalanı işletmeleri, yolcuların güvenliğini ve memnuniyetini sağlamak için çeşitli önlemler alınmaktadır. Ancak, havaalanı işletmeleri aynı zamanda büyük miktarlarda yolcu ve bagaj taşır ve bu nedenle, kapı ve bagaj ataması gibi problemler ortaya çıkabilmektedir. Bu problemler, yolcuların uçuşlarını kaçırmalarına veya bagajlarının kaybolmasına neden olabilir. Literatürde, havaalanı uçak kapı ataması ve yolcu bagajlarının uçağa yönlendirilmesi problemlerinin çeşitli çözümleri hakkında çalışmalar bulunmaktadır. Bu çalışmalar, havaalanı işletmelerinin bu problemleri nasıl çözebileceğine dair öneriler sunmaktadır.

2.1 *İlgili Uçağa Kapı Ataması Literatür Özeti*

Babic vd. [4] uçak kapı atama problemine bir doğrusal 0-1 tamsayılı programlama modeli olarak yaklaşmışlar ve bu modele göre bir dal-sınır algoritması kullanarak optimal çözümler aramaya çalışmışlar. Bu algoritmada, transfer yolcularının dikkate alınmadığı düşünülmüş. Bu şekilde, uçak kapı atama problemine en uygun çözümlerin bulunması hedeflenmiştir.

Mangoubi ve Mathaisel [16], kapılar arasında yolcuları taşıma ile ilgili bir problemi çözmek

için doğrusal programlama gevşetmesi ve doyumsuz bir heuristik kullandılar. Yolcuların yürüme mesafesini kapıdan kapıya taşıma modeline göre değerlendirdiler. Ayrıca, problemi çok sayıda değişken ve kısıt içeren bir lineerleştirilmiş ikinci derece tamsayı programlama modeli olarak formüle ettiler. Ancak, lineerleştirilmiş ikinci derece tamsayı programlama modelini çözmenin zor ve zaman alıcı olacağını vurguladılar. Bu nedenle, önerilen model üzerinde daha fazla çalışmadılar.

Wirasinghe ve Bandara da [5] havalimanı terminali yapılarında yürüme mesafesini minimuma indirmek için bir simülasyon tekniği kullandılar. Bu sayede farklı terminal yapılarının yürüme mesafesi dağılımları hesaplanarak en uygun yapı belirlenmiştir. Hareketli yürüyüş yollarının kurulmasının yolcu yürüme mesafesini azaltma etkisi de araştırılmıştır.

Braaksma ve Shortreed [7], havaalanında uçuşların kapı atamasını yaparken, gereken zorunlu adımları belirleyerek, uçuşların bekleme süresini minimize etmeyi hedeflemiştir. Bu amaçla, uçuşların kapı atamasını yaparken dikkate alınması gereken çeşitli faktörler (uçuş türü, kapı türü, uçuşlar arasındaki zaman aralıkları vb.) belirlenmiş ve bu faktörler kritik yol yöntemiyle değerlendirilerek en uygun kapı atamasını bulmayı amaçlamıştır.

A.Bolat [6] havaalanı kapılarının etkin bir şekilde atanması için bir matematik modeli geliştirmiş ve bu modelin, uçuş saatlerinde yapılan küçük değişiklikleri özümseyebilen yeterli esnekliğe sahip olmasını amaçlamıştır. Büyük değişikliklerle başa çıkmak için ise, alt sınırları kullanan optimum ve heuristik yöntem kullanmıştır. Kullanılan yöntemlerin verimliliği, rastgele oluşturulan saatler üzerinde test edilmiş ve sezgisel yöntemin, en fazla 20 parçalı çözümü bir seviyede değerlendirdiğinde, optimal çözüme ulaştığı görülmüştür. Ayrıca, Riyad Uluslararası Havaalanı'ndan elde edilen verilere göre, sezgisel yöntem, mevcut uygulamayı aşmış ve ortalama olarak, uzak mesafede hizmet edilen uçakların sayısında %72.03 ve sürüklenen uçakların sayısında %54.28 oranında iyileştirme sağlamıştır.

Yu Cheng [8] havalimanı kapı atama sürecinde uçakların kapılara atanması için bir bilgi tabanlı sistem önermiştir. Bu sistem, matematik programlama teknikleriyle bütünleştirilmiştir. Statik ve dinamik durumlar için de çözümler sunmayı amaçlamaktadır. Önerilen sistem, bir grup uçağın atamasını yapmak için mevcut tüm kapıları inceleyerek çok amaçlı bir fonksiyonu optimize eder. Önerilen yaklaşımın doğruluğu, bir örnek vaka analizi ve mikrobilgisayar ortamında çeşitli fonksiyonları olan bir prototip sistem kullanılarak test edilmiştir.

Haghani ve Chen [13] havaalanı kapı atama problemini çözmek için matematiksel programlama yaklaşımını kullanmışlar. Bu yaklaşım, yolculardan kapı ve uçak durak arasında olan toplam yürüme mesafesini minimize etme amacıyla problemi karışık tam sayı doğrusal programlama modeli olarak formüle etmiştir.Haghani ve Chen, bu modeli kommeryal optimize yazılımını kullanarak çözmüş ve Los Angeles Uluslararası Havaalanı'nda yaptıkları bir örnek çalışmada uygulamışlardır. Ayrıca, matematiksel programlama yaklaşımından daha hızlı çözümler bulabilecek bir heuristik yöntem de önermişlerdir. Bu yöntem, neredeyse optimal çözümleri bulabilmektedir.

Bu çalışmaların yanı sıra bazı araştırmacılar da statik kapı atamalarının performansını arttırmak amacıyla stokastik uçuş gecikmelerini dikkate alarak çalışmalar yapmışlardır. Örneğin, Yan ve diğerleri [19, 20] aynı kapıya atanan iki ardışık uçak arasında sabit tampon süresi kullanımışlardır. Bu tampon süresi, iki uçak arasındaki zaman boşluğu olarak düşünülebilir ve bu boşluk sayesinde, stokastik uçuş gecikmelerinin etkisi en aza indirgenmeye çalışılmıştır. Bu şekilde, atanan kapıların verimliliği arttırılmaya çalışılmıştır.

Yan ve Huo [20] büyük ölçekli problemleri çözmek için ağırlık metodu, sütun üretim yaklaşımı, simplex metodu ve dal ve sınır tekniği kullanmışlardır. Bu yöntemler havaalanı olarak Chiang Kai-Shek (CKS) Havaalanı'nın işletimi üzerinde test edilmiştir. Bu çalışma sonucunda, modelin gerçek işletmelerde kullanışlı olabileceği görülmüştür.

Yan ve Chang [19] Taipei Taoyuan Uluslararası Havalimanı'ndaki bir örnek çalışma ile gösterdiği bir çalışmada, stokastik kapı atama problemini, uçuş gecikmelerini de baza alarak, bir karışık tam sayı programlama modeli olarak formüle etti. Modeli çözmek için bir simülasyon tabanlı optimize yöntemi kullandı. Bu yöntem, farklı kapı atamalarının performansını değerlendirmeyi içeren uçuş gecikmelerinin çeşitli senaryoları oluşturuldu.

Ding ve diğer çalışma arkadaşları [9] Taipei Taoyuan Uluslararası Havalimanı'nda bir örnek çalışma ile gösterilen bir çalışmada, havalimanı kapı atama probleminde, kapılara atanamayan uçakları minimize etmek ve toplam yürüme mesafelerini ve böylelikle bağlantı sürelerini minimuma indirgemek amacıyla, greedy yaklaşımı, benzetilmiş tavlama ve tabu aramanın bir hibridi kullanıldı. Bu yöntemlerle elde edilen deneysel sonuçların daha öncekilere göre daha iyi sonuçlar verdiği belirlendi.

Ulrich Dorndorf [10] ve arkadaşları, kapı atama problemini ele alan literatürde birçok örneği olan genel bir uçak kapı zamanlama problemini çözmek için geliştirilen matematiksel modelleri ve güncel çözüm tekniklerini inceledi. Bu çalışmalar, bir zaman dilimine veya birden fazla zaman dilimine sahip modeller ve yolcu veya işletmeci amaçlarına göre modeller olarak sınıflandırmıştır. Donford vd. bu çalışmadaki amacı yerel işletim işlemlerinin zamanlamasına dokunmadan sadece uçak kapı yönetimine odaklanmaktır.

2.2 *Havaalanı Bagaj Ataması Literatür Özeti*

Literatürün bu kısmında, havaalanı işletmelerinin karşılaştığı bir diğer önemli problem olan bagaj sıralama istasyonlarına yönelik yapılan literatür çalışmaları incelenecektir.

Amadeo Ascó Signes [3], havaalanı bagaj sıralama istasyonu problemi için uygun çözümler bulmak için bir evrimsel algoritma ve farklı işleticiler kullanmıştır. İşleticiler, mevcut

kaynakların yetersiz olduğu durumlarda veya kaynakların talebi karşılamaya yeterli olduğu durumlarda kullanılmıştır. Farklı işleticilerin katkıları, diğer yaklaşımlara göre incelenmiş ve karşılaştırılmış, hangi seçimin problemdeki özel durumlara göre uygun olabileceği hakkında bilgi vermiştir.

Rijsenbrij ve Ottjes [17] bir simülasyon kullanarak, bagaj taşıma ve programlama yöntemi ile parçalı otomatik bagaj yükleme ve boşaltma aracı (bagaj kamyonu) prototipinin uygulaması incelenmiş ve rapor edilmiştir. Bu çalışmada programlama yöntemi ve yeni bagaj aracının kullanımı ile büyük tasarrufların mümkün olabileceği görülmüştür.

Edward Huang [14] ve arkadaşları havaalanı dış hat bagaj işleme sisteminde bagaj yükleme bölgelerinin nasıl atanacağı problemine yönelik bir stokastik vektör atama modeli sunmuş ve bu modelin gerçek bir havaalanı örneği üzerinde test etmişlerdir. Bu modelin performansı, mevcut yöntemlerle karşılaştırılmıştır.

BÖLÜM: II

MATERYAL

3

ALGORİTMALARA GENEL BAKIŞ

Havaalanlarında uçak kapı atama ve bagaj yönetimi gibi problemlerde, genellikle en iyi çözümleri bulmak için algoritmalar kullanılmaktadır. Bu algoritmalar, belirli bir kriter seti dikkate alarak, en iyi seçenekleri belirler. Örneğin uçak kapı atamada, en az yürüme mesafesi/süresi olan kapıyı seçmeyi hedefleyen bir algoritma yazılabilir. Bagaj yönetiminde ise, en az maliyeti olan bagaj yönetim sistemini seçmeyi hedefleyen bir algoritma yazılabilir. Algoritmaların veri girdisi olarak uçak tasarımı, yolculuk rotaları, mevcut kapı ve bagaj sistemleri gibi faktörleri dikkate alarak çalışması ve sonuç olarak en iyi seçeneklerin incelenmesi planlanmaktadır. Bu çalışmada problemlere matematiksel modeller ve algoritmalar yardımıyla çözüm getirilmesi hedeflenmektedir.

3.1 *Matematiksel Programlama Teknikleri*

Matematiksel programlama algoritmaları, havaalanı uçak kapı ve bagaj atamaları gibi problemleri çözmek için oldukça yararlı olabilir. Bu algoritmalar, çok sayıda değişkeni, kısıtlamaları ve optimize edilmesi gereken amaç fonksiyonlarını içeren bu problemleri

çözmek için kullanılır. Bu bölümde atama problemleri için uygun olabilecek optimizasyon algoritmaları incelenmiştir.

3.1.1 *Kesme Düzlemi Algoritması*

Kesme düzlem algoritması, lineer programlama problemlerini çözmek için kullanılan bir optimizasyon algoritmasıdır. Algoritma, kesme düzlemleri ekleyerek çözümü iyileştirip çalışır. Kesme düzlemleri, matematiksel kısıtlamalardır ve optimal çözümleri içermeyen mümkün bölge parçalarını kesmeye yarar.

Kesme düzlem algoritması, ilk olarak bir başlangıç mümkün çözümü bulur. Bu, tüm problem kısıtlamalarını karşılayan bir çözümdür. Daha sonra, geçerli mümkün çözümü tarafından karşılanmayan kısıtlamaları arar. Her ihlal edilen kısıtlaması için algoritma, mümkün olmayan çözümü içeren mümkün bölge parçasını kesen bir kesme düzlemi ekler. Bu işlem, tüm kısıtlamaları karşılayan bir mümkün çözüm bulunana kadar tekrarlanır.

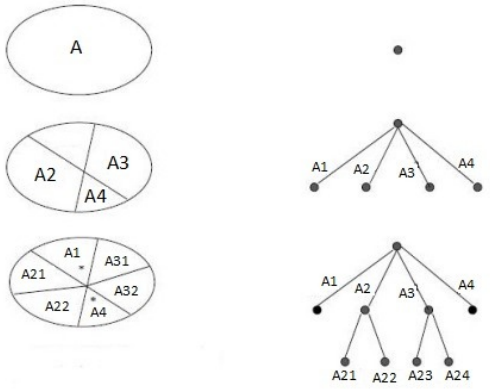
Kesme düzlem algoritmasının bir avantajı, çok sayıda değişken ve kısıtlamaya sahip problemleri ele alabilmesidir. Ayrıca, uygulaması oldukça kolaydır ve çeşitli tipte lineer programlama problemleriyle kullanılabilir. Ancak, optimal çözüme ulaşmada yavaş olabilir ve her zaman global optimumu bulamayabilir.

3.1.2 *Dal-Sınır Tekniği*

Dal ve sınır algoritması pek çok optimize etme probleminin çözümü için kullanılır, buna ek olarak tamsayı programlama ve tümleşik optimize etme problemleri için de uygulanabilir. Algoritma, tüm çözüm adaylarını sistematik bir şekilde numaralandırır ve olumsuz çözüme

yol açabilecek adayları üst ve alt sınırlar belirleyerek çözüm kümesi dışında tutar böylece sayılamaları azaltarak daha hızlı yakınlaşılır. Geleneksel dal ve sınır algoritması, her adımda pek çok alt problem dallandırılır ve dalların içindeki problemler çözülür. Şekil 1'de bunun bir örneği gösterilmiştir. Olumlu bir çözüm bulunduğunda, bulunan en olumlu çözüm bir sonraki dallandırma için üst sınır olarak belirlenir ve bir sonraki dallandırma, belirlenen üst sınır için yapılır. Bu işlem, en iyi çözüme ulaşana kadar tekrarlanır. Dal ve sınır algoritması, azaltma problemleri üzerine kurulmuştur ancak algoritmayı arttırma problemleri için de uyarlanabilir. Örnek Eşitlik 1'de verilmiştir.

$$(maksimize\, a(x) \equiv minimize\, b(x), b(x) = -a(x)) \tag{1}$$



Şekil 1: Dal Sınır Algoritması Metod Örneği

Literatüre bakıldığında çalışmalarda dal-sınır tekniği birçok araştırmacı tarafından atama problemleri için kullanılmış ve iyi sonuçlar elde edilmişti. Babic,Yan ve Huo [4, 20] gibi araştırmacılar bu tekniği kullanarak atama problemlerine çözüm getirmeye çalışmıştır. Yan ve Huo dal-sınır tekniğini ve başka metotlarla birlikte kullanmış ve bu modelin gerçek işletmelerde kullanışlı olabileceğini görmüştür.

Kesme Düzlem ve Dal Sınır algoritması Karşılaştırması yapılmıştır Çizelge 1.

Çizelge 1: Dal-Sınır ve Kesme Düzeyi Algoritmalarının Karşılaştırması

Dal-Sınır	Kesme Düzlem
Mümkün çözüm alanı içinde bulunan tüm tamsayılı çözüm çiftlerini incelemektir.	Algoritmanın temel fikri, lineer programlama probleminin en iyi çözümüne yakın olan en iyi tam sayı noktasını bulmaktır.
Kullanıcıya en iyi sonucu sunma açısından Kesme Düzlemini geçer.	Dal-Sınır yöntemine göre çözüme daha kısa sürede ulaşır.
Dal-Sınır algoritması, tamsayılı çözüm çiftlerinin elde edildiği düğümlerde Doğrusal Programlama yapılması gerektiğinden, büyük çaplı problemlerde zaman açısından avantaj sağlamaz. Her bir düğüm için ayrı ayrı işlem yapılması gerekir.	Bu, algoritmanın dezavantajı ise; yuvarlaklaştırma hatalarına karşı hassas olması nedeniyle, bazen optimum sonuca ulaşılamamasıdır.

3.1.3 *Doğrusal Programlama*

Doğrusal programlama problemi, doğrusal kısıtlar adı verilen eşitlik veya eşitsizlik grubuyla amaç fonksiyonun değerini optimize etmeyi hedefler. Doğrusal kelimesinin kullanılması, girdiyle çıktı arasında doğrusal bir ilişkinin olması mantığına dayanır. Karar değişkenlerinin doğrusal formatta, kantitatif(sayısal) aynı zamanda 0 veya pozitif olması şartlarının gerçekleştirilmesi gerekmektedir. Bu değişkenler arasında alternatif seçimler bulunmalıdır ki bu da seçim yapabilmeyi sağlar. Aynı zamanda kullanılacak kaynakların sınırlı olması şartı vardır. Doğrusal programlama kısa dönemli problemler için daha uygundur. Doğrusal programlamanın ilk aşamasında karar ortamı çok iyi bir şekilde anlaşılmalıdır ve karar ortamını etkileyebilecek tüm kısıtların belirlenmesi gerekmektedir Doğrusal programlamanın ikinci aşamasında ise bu karar değişkenlerinin anlaşılır bir şekilde ifadesi oluşturulmalı ve her kararın birer birer ifadeleri yapılmalıdır. Son aşamada ise karar değişkenleri kullanılarak kısıtların ve amaç fonksiyonunun ifadesi yazılmalıdır. Aşağıdaki Eşitlik 2, 3 ve 4'üncü formülasyonlarda, minimum maliyet ve maksimum kazancı sağlayan amaç fonksiyonlarının matematiksel ifadesi yapılmıştır. Karar değişkenleri olarak: $X_1, X_2 \ldots X_n$ kullanılmıştır [11].

$$Z_{max} = C_1 \cdot X_1 + C_2 \cdot X_2 + C_3 \cdot X_3 + \cdots + C_n \cdot X_n \tag{2}$$

$$Z_{max} = \sum_{j=1}^{\pi} C_j X_j \tag{3}$$

$$Z_{min} = \sum_{j=1}^{\pi} C_j X_j \tag{4}$$

3.1.4 *Dinamik Programlama*

Dinamik programlama tekniğinin temelinde karar kavramı ve karar süreçleri vardır. Genellikle işletmeler, amaç fonksiyonlarının birbirleriyle çelişmelerinden dolayı yöneticilerin belirlenen amaçlara en uygun çözüm araştırmaları yapmaları gerekir. Birbirlerini etkileyen bu amaç fonksiyonlarının matematiksel olarak ifade edilmesi dinamik programlama tekniğini oluşturur. Sıralı karar problemlerinin çözümünde kullanılabilecek bir yöntem olarak dinamik programlama tercih edilir. İlk alınan kararın, bir sonraki kararı etkilemesi sıralı karar problemini doğurur. Dinamik programlama yöntemiyle ele alınan problemler, daha az değişkeni olan daha küçük alt problemlere ayrılarak çözülmektedir. Kısaca problemin optimize edilerek daha az sayıda değişkenler içeren problemlere dönüşümü yapılır. Dinamik isminin verilmesinin nedeni zaman değişkeninin açık olarak ele alınmasıdır. Küçük sorunlar teker teker ele alınır ve en sonunda büyük resme bakılarak optimal çözüm önerisinde bulunulur. Dinamik programlamanın getirisi genellikle kazanç, maliyet ve kaynak tüketimi olmaktadır. Hesaplama kolaylığı sağlar. Dinamik programlamanın bir dezavantajı olarak, değişkenlerin değerlerinin çok fazla sayıda olması her adımda oluşturalacak çizelgelerin kapasitesini arttırmasına sebebiyet verecektir.

3.1.5 *Karınca Kolonisi Optimizasyonu*

Matematiksel modeller üzerine kurulmuş bir algoritmadır. Karıncaların yuva ve yiyecekleri arasındaki en kısa yolu fenemon adı verilen maddeyi salgılayarak belirledikleri görülmüştür. Yapay karınca kolonileri üzerinde oluşturulan optimizasyonun değerlendirilmesi için sunulan algoritmalar gerçek karınca davranışlarından daha farklı bir yapıdadır. Bu algoritma genel-

likle, uzun ve karmaşık optimizasyon problemlerini çözmek için kullanılır.

Karınca kolonisi optimizasyonun akış diyagram örneği Şekil 2 verilmiştir. [15] Akış diyagramı detaylı bir şekilde açıklanacak olursa:

-Başlangıç noktası: Problem ve özellikleri tanımlanır ve bir karınca kolonisi oluşturulur.

-Karınca görevlendirmesi: Kolonideki karıncalara görevler atanır.

-Karıncaların hareketi: Karıncalar problemi çözmek için hareket ederler ve çözüm yollarını keşfederler.

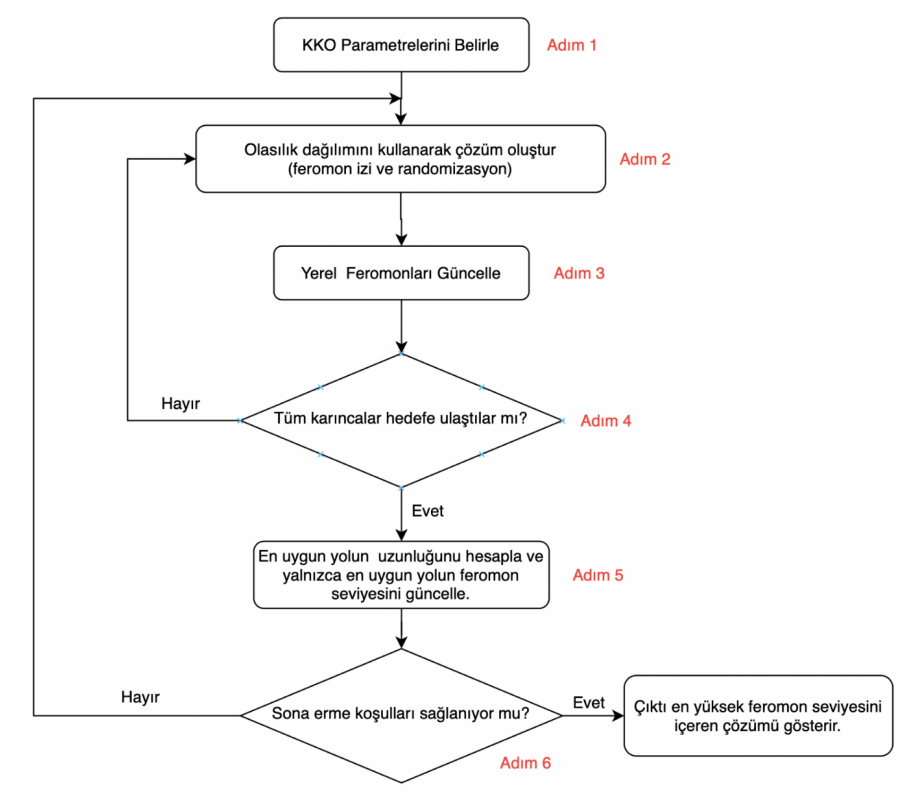
-Çözüm yollarının değerlendirilmesi: Karıncalar tarafından keşfedilen çözüm yolları değerlendirilir ve en iyi yollar belirlenir.

-Feromon güncellemesi: En iyi yollar belirlendikten sonra, karıncalar tarafından bırakılan feromonlar(diğer karıncalar tarafından takip edilen yolların değeri) güncellenir ve bu, diğer karıncaların benzer çözüm yollarını keşfetmelerine yardımcı olur.

-İterasyon: Şekil 2'de,adım 2 ile 4 arasındaki işlemler tekrar edilir ve karıncalar problemi çözmek için daha fazla çözüm yolunu keşfederler.

-Sonuç: Belirli bir döngü sayısına ulaştığında veya belirli bir çözüm kalitesine ulaştığında algoritma durdurulur ve en iyi çözüm bulunur.

Bu optimizasyon, probleme çözüm ararken çok fazla yolun denenmesini sağlar.Karıncalar tarafından takip edilen yolların seçilme olasılığı sürekli değiştiğinden en kısa yolu bulma problemi için etkili bir yöntem olabilir. Havaalanı uçak kapı atamalarında yolcu yürüme mesafesi minimize edilmek istendiğinden bu algoritma da problemin çözümünde kullanılabilecek seçenekler arasındadır.



Şekil 2: KKO akış diyagram örneği

[15]

3.1.6 *Benzetilmiş Tavlama (Simulated Annealing)*

Benzetilmiş tavlama, metallerin veya alaşımların ısıtılarak önceden belirlenen sıcaklıklarda tutulması ve daha sonra ani olarak soğutulması sırasında meydana gelen fiziksel ve kimyasal değişimleri ifade eder. Bunun amacı atomik düzenlemenin arttırılmasıdır. Benzetilmiş tavlama mevcut çözüm yerine yeni çözüm üretir. Bu şekilde yerel optima noktalarından kaçışı ve problemdeki en global çözümü bulmayı hedefler. Tipik olarak, yeterince iyi bir çözüm belirlenene kadar veya verilen bir zaman limitine ulaşılana kadar bu işlem tekrarlanır.

Bu benzerlikten yararlanılarak benzetilmiş tavlama algoritması kapı ve bagaj atama problemlerine uygulanabilmektedir. Bu algoritma var olan çözümlere daha iyi alternatifler oluşturmak için ideal bir yöntemdir. Bu yöntem üç ana aşamadan oluşur:

1. Başlangıç Durumuna Getirme: Başlangıç çözümü seçilir ve benzetilmiş tavlama parametreleri belirlenir. Var olan çözüm değeri optimal çözüm olarak atanır.

2. Seçim ve Bitirme Aşaması: Ulaşılan çözümler başlangıçta belirlenen parametrelerle kıyaslanır. Eğer istenilen kriterler sağlanmışsa algoritma durdurulur. Sağlanmamışsa yeni bir çözüm seçilir.

3. Güncelleme: Elde edilen yeni çözüm en iyi çözüm olarak atanır. Seçim ve bitirme aşamasına geri dönülür.

Benzetilmiş tavlama ile uçakların terminal kapılarına atanması problemi, maddelerin kararsızlığının minimize edilmesinin sağlanması ile yolcu yürüme ve bagaj taşıma süresinin minimuma indirgenmesi açısından benzerlik gösterir.

3.1.7 *Tabu Arama*

Mümkün olan çözümler arasından en iyi olanı belirlemek için iteratif çalışan bir arama, araştırma algoritmasıdır. Ulaştığımız son aşamadan bir önceki aşamada dairesel bir yapının içerisine girilmemesi amacıyla diğer döngüde tekrardan kaçınılması mantığına dayanır. Meydana gelen çözümler arasında kıyaslama yapan bu algoritma sahip olduğu alanın dışındaki çözümleri inceleyerek en optimal çözümü vermektedir. Metasezgisel bir yaklaşım sunmaktadır. Algoritmanın, çözümleri taradığı alanı genişletmesi her tekrarda değer çıktısı veren bir fonkiyonun daha ideal değere ulaştığında bu değere bağlı olan hareketin sonraki çözüm olarak alınması ile ilgilidir.

Bu algoritmanın işleyişi başlangıç çözümününün belirlenmesi ile başlar. Bu adımda bu çözüm rastgele olarak seçilebildiği gibi sorunu çözebilecek alternatif algoritma yardımıyla da seçilebilmektedir. Hareket mekanizması ise elimizdeki çözüm üzerinde gerçekleşen de-

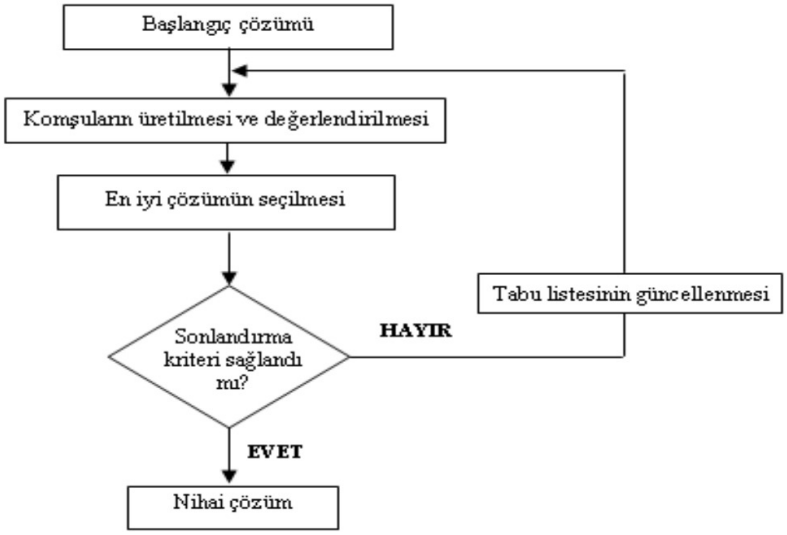
ğişimin yeni çözümü oluşturmasıdır. Bu yapı sağladığı hareketler ile komşuluk kavramını türetir. Komşu kavramında en iyi sonuç veren hareket seçilmelidir. Elde edilen arama kayıtları bellekte tutulmaktadır. Engellenen hareketler tabu olarak nitelendirilir ve esneyebilen bellek içerisinde tabu listesi başlığı altında saklanmaktadır. İlerleyen zamanda engellenmenin kalktığı durumda listeden alınmaktadır ve hareketin gerçekleştirilmesi sağlanmaktadır.

İki tip bellek kullanımı vardır. Bunlar yakın süreçte gerçekleşen hareketlerin doğrudan veya sistematik olarak belirlenen bir zamanda tabu listesine eklendiğinde kullanılan bellek yapısı ve hareketlerin belirli parametrelere göre tekrar sayılarını tutan ikincil bellek yapısıdır. Bu bağlamda hafızalar kısa ve uzun süreli olarak değerlendirilmektedir.

Tabu durumunun kalkması için en yaygın kullanım, bulunan durumdan daha iyi sonuç verebilecek bir hareketin engelinin kalkmasıyla sağlanmasıdır. Başka bir kullanım ise hareket ihtiyacı nedeni ile zaman karşılaştırılması yapılarak en az zamanı kalan hareketin engellenme durumunun kalkmasıdır.

Bazı şartların sağlanması ile algoritmanın işleyişi durdurulmaktadır. Bu şartlara komşuluğun olmaması, tekrar sayısının dışına çıkılamaması, çözümün bulunması örnekleri verilebilmektedir.

Bu algoritma ilk çözüm ile işleyişini başlatmaktadır. İzne sahip olan bir hareket bu çözümün bir komşusunu seçmektedir. Bu seçimin kıyaslanması gerçekleştirilerek daha iyi bir sonuç sağlanması durumunda kıyaslanan bu yeni çözüm üzerinden işleyişe devam edilmektedir. Hareket eğer tabu durumunun kalkmasını sağlıyorsa yeni çözüm oluşumu için kullanılmaktadır. İlerlemenin devamı için bazı hareketlerin listede kaydı sağlanarak izni kaldırılmaktadır. İşleyin durdurulması için şartların sağlanması gerekmektedir. Bu işleyiş Şekil 3'te gösterilmiştir. Çizelge 2'de gelen uçakların kapılara atanmasının maliyetleri gösterilmek-



Şekil 3: Tabu Arama Algoritması Akış Diyagramı

[12]

tedir. Bu örnek, uçakların kapılara atanması tabu arama algoritması kullanılarak minimum ya da minimuma yakın maliyetteki yerleştirilmelerin yapılmasıdır.

Çizelge 2: Örnek Uçak Kapı Atama Maliyet Gösterimi

	A1D	B13	D14	F9A
TK01	15n	2n	6n	9n
TK02	8n	4n	13n	14n
TK03	7n	16n	n	12n
TK04	3n	11n	10n	5n

Tablodaki verilerden rastsal bir yerleştirme seçilir. Bu bağlamda uçakların kapılara atanması $''TK04, TK03, TK02, TK01''$ olarak belirlenmişitir. Bu gösterim $TK04$'ün $A1D$, $TK03$'ün $B13$, $TK02$'nin $D14$, $TK01$'in $F9A$ kapılarına atandıklarını göstermektedir. Bu gösterim biçiminin maliyeti tablodaki veriler kullanılarak

$TK04, TK03, TK02, TK01 = 3n + 16n + 13n + 9n = 41n$ olarak hesaplanır.

En iyi maliyet $= 41n$

İterasyon sayısı $= 3$

Tabu listesi uzunuluğu $= (), (), ()$

KOMŞULUKLAR: 1. İTERASYON

En başarılı kabul edilen atama $= TK04, TK03, TK02, TK01$

$$TK03, TK04, TK02, TK01 = 40n \tag{5}$$

$$TK02, TK03, TK04, TK01 = 43n \tag{6}$$

$$TK01, TK03, TK02, TK04 = 49n \tag{7}$$

$40n < 41n$ olduğu için:

En iyi kapı atama= $TK03, TK04, TK02, TK01$

En iyi maliyet: $40n$

Tabu listesi:$(TK03, TK04), (), ()$

KOMŞULUKLAR: 2. İTERASYON

En başarılı kabul edilen atama $= TK03, TK04, TK02, TK01$

$$TK04, TK03, TK02, TK01 = 41n \tag{8}$$

$$TK02, TK04, TK03, TK01 = 29n \tag{9}$$

$$TK01, TK04, TK02, TK03 = 42n \tag{10}$$

$29n < 40n$ olduğu için:

En iyi kapı atama $= TK02, TK04, TK03, TK01$

En iyi maliyet $= 29n$

Tabu listesi $= (TK02, TK03), (TK03, TK04), ()$

KOMŞULUKLAR: 3. İTERASYON

En başarılı kabul edilen atama $= TK02, TK04, TK03, TK01$

$$TK04, TK02, TK03, TK01 = 17n \tag{11}$$

$$TK03, TK04, TK02, TK01 = 40n \tag{12}$$

$$TK01, TK04, TK03, TK02 = 41n \tag{13}$$

$17n < 29n$ olduğu için:

En iyi kapı atama $= TK04, TK02, TK03, TK01$

En iyi maliyet $= 17n$

Tabu listesi $= (TK04, TK02), (TK02, TK03), (TK03, TK04)$

Tabu arama algoritması kullanılarak uçakların kapılara atanması işleminde en uygun kapı atama sıralamasının "$TK04, TK02, TK03, TK01$" olduğuna karar verilmiştir. Hesaplamalara göre 3 iterasyon sonucunda maliyet $17n$ bulunmuştur.

3.1.8 *Açgözlü Yaklaşım (Greedy Approach)*

Bir problemin çözümünde gelecekte neden olabileceği sonuçlar hesaba katılmadan var olan koşullar altında en uygun olan seçimin uygulanmasıdır. Greedy adını almasının bir sebebi olarak çözüm yöntemini uygularken ilk adım olarak en iyi alternatifi seçmesi, problemin tamamına bakıldığında verilebilecek en doğru kararı kaçırmasına neden olur. Yani açgözlü bir yaklaşımla en iyi alternatif, çözümün ilk adımında uygulanır. Bir problem için açgözlü yaklaşım kullanıldığında çözüm bulmak ve bu algoritmalar için çalışma süresinin hesaplanması diğer algoritmalara göre daha kolaydır fakat her zaman doğru sonuçları vermeyebilir. Bazı bilinen Greedy algoritmaları şunlardır: Dijkstra, Knapsack, Huffman, Prims ve Kruksal algoritmalarıdır.

Açgözlü yaklaşımı ile para üstü verme işlemini bir örnekle ifade edelim. Elimizde 25, 15, 10, 1 olmak üzere 4 farklı bozuk para bulunmaktadır. Hedef değerin 32 lira olması istenmektedir. Açgözlü yaklaşımda, her adımda en büyük değer seçilir.

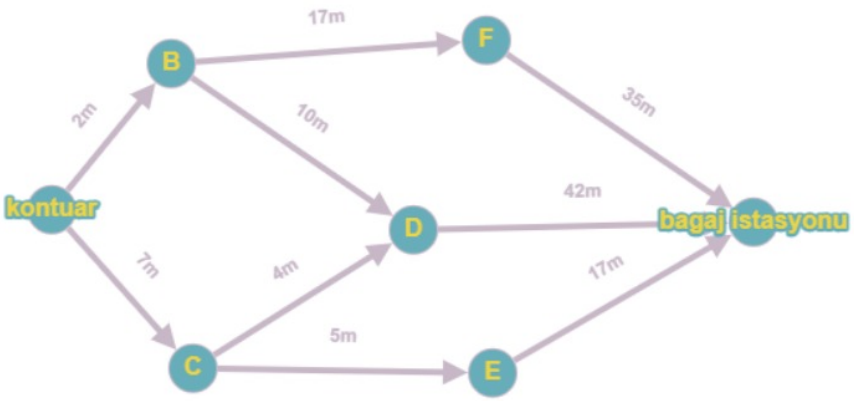
Hedeflenen değer 32 lira olduğunda:

- 25 lira ile çözüme başlar. Hedefe ulaşılamadığı için ekleme yapılmaya devam edilir.
- 25 eklendiğinde, hedef değer 32 lirayı aşacağı için ikinci en büyük parayla devam edilir.
- 15 eklendiğinde, hedef değer istenilen değerden büyük olduğu için üçüncü büyük para seçilir.
- 10 eklendiğinde, hedef değer istenilen değerden büyük olduğu için 1 liralık bozuk paralardan 7 kere ekleme yapılır.

Toplamda Açgözlü yaklaşımı kullanılarak 8 adet para üstü verilmiştir.

Oysa ki en optimal çözüm “15, 15, 1, 1” olacaktı. Toplamda 4 adet para üstü verilebilirdi.

Açgözlü yaklaşımı, graf üzerinde belirli bir konuda en iyi sonucu bulabilmek amacıyla dolaşma yapılırken bir sonraki düğümü belirlemek için kullanılan bir karar verme, seçme yöntemidir. Örnek alınarak oluşturulan açgözlü yaklaşımla en kısa yol problemi Şekil 4’de verilmiştir. Kontuara verilen bir bagajın iletimi gerçekleştirilirken açgözlü yaklaşımıyla hangi



Şekil 4: Açgözlü Yaklaşımla En Kısa Yol Problemi

yolu izleyeceğine dair karar verilir. Açgözlü yaklaşımı bütüne bakılmadan mevcut durumdaki en kısa yolu seçer. Şekil 4'deki bagajın kontuardan çıktıktan sonra bagaj istasyonuna varmak için başta en iyi çözüm olarak 2m mesafesini kullanır. Sonrasında karşısına çıkan yollardan en kısa olan 10m yolu ile transferine devam eder. Son olarak bagaj istasyonuna iletimi için karşısına çıkan yol 42m uzunluğundadır.

Verilen örnek için graf tanımları:

$G = (D, K)$

$D = \{A, B, C, D, E, F, G\}$

Buradaki A noktası kontuarı ifade eder. G noktası ise bagajların varış noktası olan istasyonu ifade eder.

|AG|, yollara bağlı olarak değişen kontuar ve bagaj istasyonu arası mesafe olarak belirlenmiştir [18].

$K = \{(A, B), (A, C), (B, F), (B, D), (F, G), (D, G), (C, D), (C, E), (E, G)\}$

$$(A, B) + (B, D) + (D, G) = |AG|$$

$$2m + 10m + 42m = 54m$$

olarak hesaplanır. Oysaki elde edilen sonuç problemin bütününe bakıldığında en iyi alternatif değildir.

$$(A, C) + (C, E) + (E, G) = |AG|$$

$$7m + 5m + 17m = 29m$$

Aslında başlangıçta kötü bir sonuç olarak görülen, bagajları C noktasına götürebilecek 7m mesafesinin problemin global çözümünün içerisinde olduğu ve takip edildiğinde gidilen C ve E noktalarının bizi bagaj istasyonuna ulaştıran en etkili yol olduğu atlanır.

Açgözlü ve Dinamik Programlama karşılaştırıldığında açgözlü yaklaşımda bir seçim elemanı tutulurken dinamik programlamada tutulmaz. Dinamik programlamadaki gibi bir karar verme yapısı açgözlü yaklaşımda yoktur. Açgözlü yaklaşımında bu durum ilerideki adımların seçilmesine veya çözümlerine bağlı değildir.

Çizelge 3: Örnek Uçuş Çizelgesi

Uçuşlar	Geliş Zamanı	Gidiş Zamanı	Geliş-Gidiş Süresi
1	3	7	4
2	4	8	4
3	3	12	9
4	7	10	3
5	7	16	9
6	8	14	6
7	11	18	7
8	9	14	5

Çizelge 3'de örnek uçuşların geliş, gidiş zamanları ve kapılarda geçirdikleri süreler verilmiştir. Greedy yaklaşımı kullanılarak uçuşların gidiş zamanları dikkate alınıp en uygun uçuşlar kapılara şu şekilde atanmıştır [2].

Çizelge 5'de First Come First Served (FCFS) yaklaşımı kullanıldığında, greedy yaklaşımına göre daha az uçuşun kapılara atandığı görülmüştür. Greedy yaklaşımı kullanılarak uçuşların kapılara atanması Çizelge 4'te verilmiştir. Uçuşlar karşılaştırıldığında Greedy yaklaşımının FCFS yönteminden avantajlı oldığu görülmüştür. Bu yüzden FCFS yöntemi yerine Greedy yaklaşımı kullanılmıştır.[2]

Çizelge 4: Greedy Yaklaşımına Göre Uçakların Kapılara Atanması

	Zaman																	
Kapı	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
A1D				TK 0101				TK 0404				TK 0707						
B2C					TK 0202					TK 0808								

Çizelge 5: FCFS Yaklaşımına Göre Uçakların Atanması

	Zaman																	
Kapı	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
A1D				TK 0101				TK 0404				TK 0707						
B2C				TK 0303														

3.1.9 *Genetik Algoritma*

Genetik algoritma, evrime dayanan bir algoritma çeşididir. Bu algoritma bir başlangıç popülasyonu kullanır. Evrimdeki esas mekanizmalardan biri doğal seçilimdir. Genetik algoritma, popülasyondaki canlıların çeşitli farklılıklarından dolayı güçlü olanların yaşama ihtimallerinin daha yüksek olması mantığına dayanır.

Bir arama algoritması olan genetik algoritma belirlenen bir probleme çözüm bulmak için kullanılır. Gerçek hayatta karşımıza çıkan bir problem eğer bir arama problemine dönüştürülebilirse genetik algoritma bir çözüm üretmek için kullanılabilir. Arama algoritmasının içinde listelenir ve içinde rassallık barındırır. Bu rassallık her işleyişinde farklı şekillerde rol almasından kaynaklanmaktadır. Evrimsel algoritmanın bir parçasıdır.

1) Uygun problemin belirlenmesi

Bir uçağın kapı ataması yapılırken yolcuların toplam yürüme mesafesinin en kısa olduğu durumun bulunması problem seçimine örnek olarak verilebilmektedir.

2) Problemin algoritmaya uyarlanması

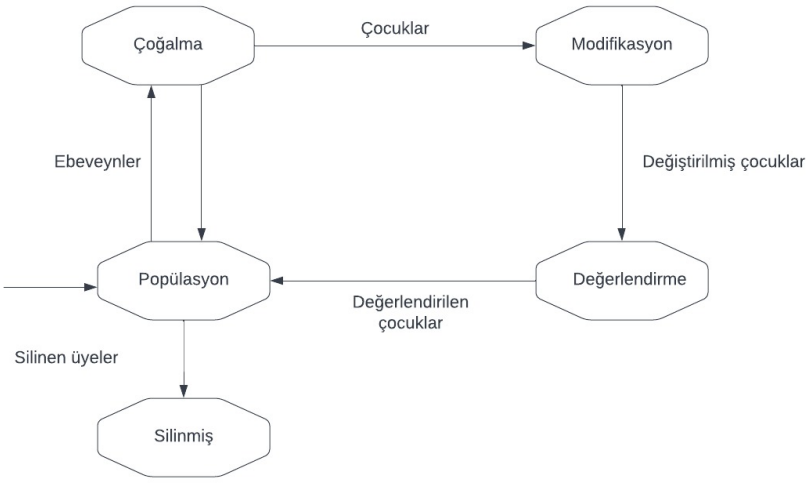
Belirlenen problemin genler ve kromozomlar ile bu dile ait bileşenlerle ifade edilmesi sağlanır.

3) İlk popülasyonun oluşturulması

İlk popülasyon için kendi kromozom dizilimlerine sahip örnek çocuklar oluşturulur.

4) Oluşturulan çocukların değerlendirilmesi

İstenilen çocukların oluşturulduğu bir hedef fonksiyonu ve diğer çocukların, istenilen duruma yakın ve uzaklığı ile alakalı değerlendirme fonksiyonu durumunun tanımı mevcuttur. Değerlendirme fonksiyonu, gelecek nesiller hakkında çıkarım yapmamızı sağlamaktadır.



Şekil 5: Genetik Algoritma'nın Çoğalma Döngüsü

Başlangıç durumundan başlanılarak popülasyon üretilir. Bu popülasyon gen havuzuna karşılık gelir. Çoğalma durumunda seçilmiş olan atalar, çift yönlü ok tanımlanması ile tekrardan topluluğa dahil edilmesi gerçekleştirilebilir. Bu sayede ataların tekrardan üretime katılması ve daha çok çeşitliliğin sağlanması mümkün olmaktadır. Bu durum istenilmediği takdirde atanın kullanıldıktan sonra silinerek, çocuklarının değerlendirilmesi ile popülasyona katılıp, katılmayacağına karar verilir. Genel mantığında, doğal seçilim ile zayıf olan üyelerin, işleyişten çıkarılması vardır. Şekil 5'da bu mantık çoğalma döngüsu halinde gösterilmiştir.

Genetik algoritmayla kodlama yapabilmek için temel fonksiyonların bilinmesi gerekir. Bu fonksiyonlar şunlardır [1]:

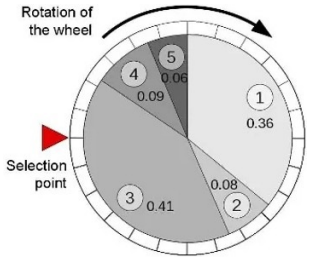
Fitness Fonksiyonu (Fitness Function) : Fitness fonksiyonu genetik algoritmanın çözümleri nasıl değerlendireceğini belirlediği ve optimize etmek için kullandığı bir ölçüttür. Örnek olarak, maliyeti minimize etmek için kullanılan bir fonksiyon olabilir.

Seçim Fonksiyonu (Selection) : Bu fonksiyon, genetik algoritmanın hangi çözümleri devam ettireceğini ve hangilerini atlayacağını belirler. Genellikle en iyi değerlere sahip çözümler seçilir. Örnek olarak, elitist seçim yöntemi kullanılabilir. Literatüre bakıldığında bilinen seçim algoritmaları şunlardır:

-Rulet Tekerleği Seçimi : Genetik algoritmalarda kullanılan bir seçim yöntemidir. Bu yöntem her kromozomun fitness değerine göre bir olasılık değeri atar ve bu olasılıklar arasından rastgele bir sayı üreterek seçilen kromozomu belirler. Bu olasılık değerleri her kromozom için fitness değerleri ile orantılıdır, dolayısıyla daha yüksek fitness değerine sahip olanlar daha yüksek bir olasılıkla seçilir. Seçilen kromozomların uygunluk fonksiyon hesaplamalarını yapar ve Eşitlik 14'deki formüle göre rulet tekerleğine yerleştirir (Şekil 6). Bu yöntem genetik

çeşitliliği arttırmak ve daha iyi çözümler elde etmek için kullanılır.

$$Pi = \frac{f_i}{\sum_{j=1} f_j} \tag{14}$$



Şekil 6: Rulet Tekerleği Seçimi

[1]

-Rastgele Seçim : Bu yöntemde, populasyon içindeki herhangi bir kromozom rastgele bir şekilde seçilir.Kromozomların fitness değerlerine göre bir olasılık değeri atanmaz ve herhangibir öncelik yoktur. Bu yöntem genetik çeşitliliği arttırmak ve çözümler arasındaki farklılıkları azaltmak için kullanılabilir.

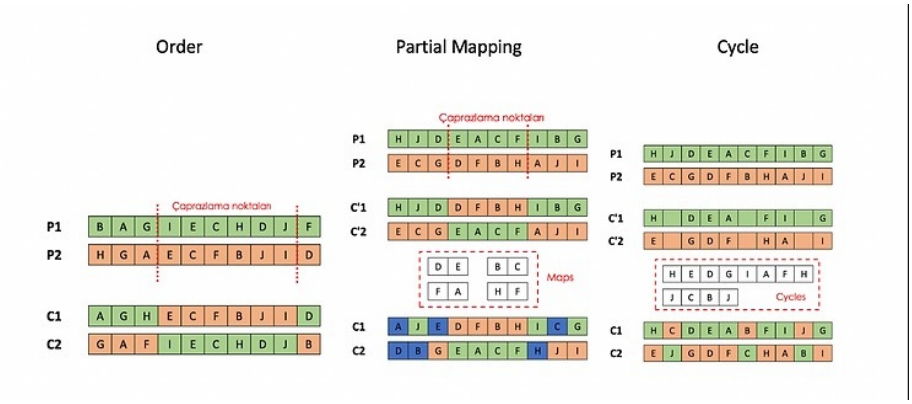
-Turnuva Seçimi : Bu yöntemde, populasyon içinden rastgele seçilmiş birkaç kromozom arasından, en yüksek fitness değerine sahip olan seçilir. Bu yöntem, genetik algoritmanın çözümler arasındaki genetik çeşitliliği koruma imkanı tanır ve aynı zamanda daha iyi çözümleri bulmak için çalışır. Kararlı ve performanslı çözümler elde etmek için kullanılır.

Çaprazlama Fonksiyonu (Crossover) : Genetik algoritmanın çözümler arasında nasıl çaprazlama yapacağını belirler. Şekil 7' deki fonksiyon parent1 ve parent2 adlı iki çözüm alır ve rastgele bir çaprazlama noktası seçer. Bu noktadan sonra, parent1 dizisi ile parent2 dizisi arasında çaprazlama yapar ve yeni bir çocuk dizisi oluşur. Bu çocuk dizisi, önceki nesil çözümlerinin genetik bilgisini içerir ve genetik çeşitliliği arttırır.

```
>> function offspring = crossover(parent1, parent2)
    n = length(parent1);
    crossoverPoint = randi([1, n-1]);
    offspring = [parent1(1:crossoverPoint) parent2(crossoverPoint+1:n)];
end
```

Şekil 7: Genetik Algoritma Crossover Örnek Matlab Kodu

Genetik algoritmada çaprazlama fonksiyonu için; Şekil 8'de gösterilen order crossover,partial mapping crossover,cycle crossover gibi yöntemler kullanılır.



Şekil 8: Çaprazlama Fonksiyonunda Kullanılan Yöntemler

[1]

Mutasyon Fonksiyonu (Mutation) : Mutasyon, genetik algoritmanın çözümler arasında rastgele değişiklikler yapmasına olanak tanır ve çözümler arasındaki genetik çeşitliliği arttırır. Elde edilen çözümün komşularının taranmasıyla gerçekleşir ve daha iyi bir çözüm olup olmadığı araştırılır. Bu yöntem, heuristic ve meta-heuristic algoritmaların sıklıkla kullandığı bir yöntemdir.

Döngü Fonksiyonu : Genetik algoritmanın kaç nesil boyunca çalışacağını veya belirli bir kriterlere ulaştığında ne zaman durdurulacağını belirler. Örneğin, belirli bir fitness değerine ulaşıldığında durdurulabilir.

Çözüm Fonksiyonu : Genetik algoritmada optimize edilen problemlerin sonuçlarını üreten fonksiyondur.

3.2 *Simülasyon ve Kural Tabanlı Sistemler*

Simülasyon ve kural tabanlı sistemler, uçak kapı ve bagaj sıralama istasyonu atama gibi karmaşık sorunları çözmek için kullanabileceğimiz algoritmaların iki çeşididir.

Simülasyon algoritmaları, bir sistemin bir modelini oluşturmak ve bilgisayar yazılımını kullanarak sistemin farklı koşullar altında davranışını simüle etmeyi amaçlar. Bu, çeşitli senaryoları test etme, analiz etme ve sonuçları tahmin etme olanağı sağlar. Örneğin, bir havaalanı işletmesi, farklı kapı atamalarının havaalanı kapasitesi ve verimliliği üzerindeki etkilerini modellemek için simülasyon algoritması kullanabilir.

Kural tabanlı sistemler, alanında uzman olanlar tarafından tanımlanmış bir kural setine dayanır. Bu sistemler sadece sınırlı sayıda olası sonuçların olduğu ve karar verme sürecinin basit kurallara ayrılabildiği durumlarda kullanılabilir. Örneğin, bir havaalanı bagaj sıralama sistemi, hedefe ve uçuş numarasına dayalı olarak doğru uçuşa yönlendirme kuralları verilen bir kural tabanlı algoritma kullanılarak tasarlanabilir.

Simülasyon ve kural tabanlı algoritmalar, taşımacılık ve lojistik, imalat ve sağlık gibi çeşitli alanlarda karmaşık sorunları çözmek için güçlü araçlar olarak kabul edilir.

4

ÇALIŞMA ORTAMI

4.1 *Kullanılan Yazılım Dilleri ve Programlar*

Havayolu sektöründe, müşterilerin uçuşlarını planlamaları ve rezervasyonlarını kolayca yapabilmelerini sağlamak için web uygulamalarının tasarımı ve geliştirilmesi önemlidir. Bu kısımda havayolu projesinin web uygulamasının tasarımında ve bagaj yönetimi, kapı atama gibi problemlerinin çözüm algoritmalarının geliştirilmesinde en doğru sonucu elde etmek için projede kullanılan yazılım dilleri ve programlardan bahsedilmiştir.

4.1.1 *Visual Studio*

Visual Studio, bir integrated development environment (IDE)'dir. Bu yazılım, birçok dillerde program yazmak için araçlar sunar ve Windows, Linux ve MacOS gibi farklı işletim sistemlerinde çalışabilir. Visual Studio, web uygulamaları geliştirirken tercih edilir çünkü kullanımı kolaydır ve birçok araç içerir. Örneğin, Visual Studio, kod editörü, depolama alanı yönetimi, hata ayıklama ve test araçları gibi özellikleri bir arada sunar. Bu, web uygulamalarının geliştirilmesini hızlandırır ve daha kolay hale getirir. Ayrıca, Visual Studio,

.NET framework ile uyumludur ve bu framework, çok sayıda dil ve platform için geniş destek sunar. Bu, web uygulamalarının farklı platformlarda çalıştırılmasını kolaylaştırır.

4.1.2 *ASP.NET MVC*

ASP.NET MVC, bir web uygulama framework'üdür ve MVC tasarım şablonunu kullanarak web uygulamalarının geliştirilmesine yardımcı olur. Bu framework, uygulamayı Model, View ve Controller bölümlerine ayırır ve bu sayede uygulamanın veri yapısı, iş mantığı ve kullanıcı arayüzü ayrı ayrı yönetilebilir. .NET framework ile uyumludur ve bu sayede .NET dilleriyle (örneğin C#) yazılmış kodları kullanabilir.

4.1.3 *C#*

C#, bir nesne yönelimli programlama dili olarak bilinir. C#'un syntax'ı C, C++ ve Java gibi dillerden etkilenmiştir, ancak aynı zamanda benzersiz özellikleri de içermektedir. C#'un amacı, uygulama geliştirmek için kullanıcı dostu ve esnek bir dildir. Bu dilde öncelikle Microsoft'un .NET framework üzerinde çalışan yazılımlar yazılır. Bu framework, Windows işletim sistemi ve web uygulamaları gibi çeşitli platformlar için yazılımların yazılmasını kolaylaştıran bir kütüphane ve araç seti içerir. C#, birçok farklı türde projeler için kullanılabilir. Bunlar arasında Windows masaüstü uygulamaları, web uygulamaları, mobil uygulamalar, oyunlar ve sanal gerçeklik uygulamaları gibi çeşitli seçenekler yer alır. C#, aynı zamanda Visual Studio gibi geliştirme ortamları için de desteklenir. Bu ortamlar, yazarlarına kod yazma, depolama, test etme ve yayınlama süreçlerini kolaylaştırmak için araçlar sağlar.

C#'un öne çıkan özelliklerinden bazıları; çoklu dil desteği, oluşturulmuş olan birçok kütüphane, kullanıcı dostu syntax, güçlü ve esnek bir tipte sistemi, Garbage Collection gibi özelliklerdir.

4.1.4 *Razor*

Razor, ASP.NET web uygulamaları için tasarlanan Microsoft tarafından geliştirilen Web sayfası oluşturma ve tasarım dilidir. C# veya Visual Basic dillerini kullanarak web sayfalarını oluşturmanıza ve düzenlemenize olanak sağlar.

Razor, HTML ve C# kodunu karıştırmak yerine, C# kodunun HTML içinde nasıl çalıştığını göstererek, web sayfası tasarımını daha anlaşılır ve okunaklı hale getirir. Razor ile web sayfaları oluştururken, C# kodunun arasına gömülmüş olarak değil, ayrı olarak yazılmasına olanak sağlar. Bu sayede, HTML ve C# kodunun arasındaki ayrım daha belirgin hale gelir ve kodun daha okunaklı hale gelir.

Razor, MVC(Model-View-Controller) yapısının bir parçası olarak kullanılabilir. Bu yapı, web uygulamalarının tasarımını, görünümünü ve veri işleme katmanlarını ayrı ayrı yönetmenizi sağlar. Bu sayede, kodunuz daha organize ve anlaşılır hale gelir.

Son olarak, Razor ile oluşturulmuş web sayfaları, tarayıcılar tarafından direkt olarak çalıştırılabilir ve sunucu tarafından işlenmeden önce herhangi bir ara adımda işlenmez. Bu sayede web sayfalarının yüklenmesi daha hızlı olur.

4.1.5 *CSS*

CSS(Cascading Style Sheets), web sayfasının görünümünü tasarlayan bir dil olarak bilinir. Bu dil, HTML veya XML dilleriyle yazılmış bir web sayfasının içeriğini ve yapısını tanımlarken, sayfanın görünümünün nasıl olacağını tasarlamak için kullanılır. Örneğin, bir sayfanın yazı tipi, arka planı, metin boyutu veya paragraf aralıkları gibi özellikleri CSS kodları ile tanımlanabilir. CSS kodları HTML veya XML dosyalarına eklenir ve tarayıcıların bu kodları okuyarak sayfayı görüntülemesi sağlanır. Bu kodlar sayfanın tasarımını değiştirir böylece bir web sayfası daha estetik ve kullanıcı dostu hale getirilebilir. Bu dil sayesinde eşsiz ve özgün tasarımlar oluşturmak mümkündür. Bu tarz tasarımlar oluşturmak için css kodlarının kullanımını doğru şekilde ve yeterli bilgi ile kullanmak gereklidir.

4.1.6 *JavaScript*

JavaScript, web sayfasının dinamik hale getirilmesi için kullanılan bir programlama dili olarak bilinir. JavaScript ile web sayfalarının içeriği ve görünümü daha canlı hale getirilebilir ve kullanıcı etkileşimi sağlanabilir. Örneğin, bir web formunda veri doğrulama, bir butona tıklandığında bir pencere açma, bir menünün açılıp kapatılması gibi işlemler JavaScript kodları ile kolaylıkla gerçekleştirilir. JavaScript, HTML ve CSS ile birlikte kullanıldığında web sayfalarının daha etkileyici bir hale getirilmesi mümkündür. Özellikle, HTML ve CSS ile yapılmış bir web sayfasının içeriği ve görünümünün JavaScript ile dinamik hale getirilmesi eşsiz ve özgün tasarımlar oluşturmak için mükemmel bir yol olarak kabul edilir.

4.1.7 *Microsoft SQL Server*

Microsoft SQL Server, Microsoft tarafından geliştirilen ve yönetilen bir veritabanı yönetim sistemidir. Kuruluşların ihtiyaçları olan veri depolama, yönetme, erişme ve işleme gibi ihtiyaçlarını karşılamak üzere tasarlanmıştır. SQL Server, birçok farklı platform ve işletim sistemi üzerinde çalışabilen bir veritabanı yönetim sistemi olup, veri tabanlarının yönetimi, depolanması, yedeklenmesi ve korunması gibi konularda kullanıcılara avantajlar sunar.

Microsoft SQL Server, veri tabanı oluşturma, yönetme ve sorgulama gibi temel işlemleri gerçekleştirmek için SQL (Structured Query Language) adı verilen bir veritabanı sorgu dili kullanır. Ayrıca, Microsoft tarafından geliştirilen Business Intelligence ve Advanced Analytics özellikleri ile birlikte, kuruluşlar için önemli analitik ve raporlama fırsatları sunar.

4.1.8 *MATLAB*

Matlab, matematik ve bilim alanlarındaki problemlerin çözümü için tasarlanmış bir programdır. Program, birçok çeşit algoritmalar sunar ve bu algoritmaları kullanarak çeşitli matematiksel ve bilimsel problemleri çözmenizi sağlar. Matlab'ta, lineer cebir, optimizasyon, sinyal işleme ve veri analizi gibi algoritmalar bulunur. Bu algoritmalar, kullanıcılar tarafından özelleştirilerek ve bir araya getirilerek, çeşitli problemlerin çözümü için kullanılabilir. Matlab algoritmalarının kullanımı, matematik ve bilimsel problemlerin çözümünde önemli bir rol oynar.

BÖLÜM: III

YÖNTEM

5

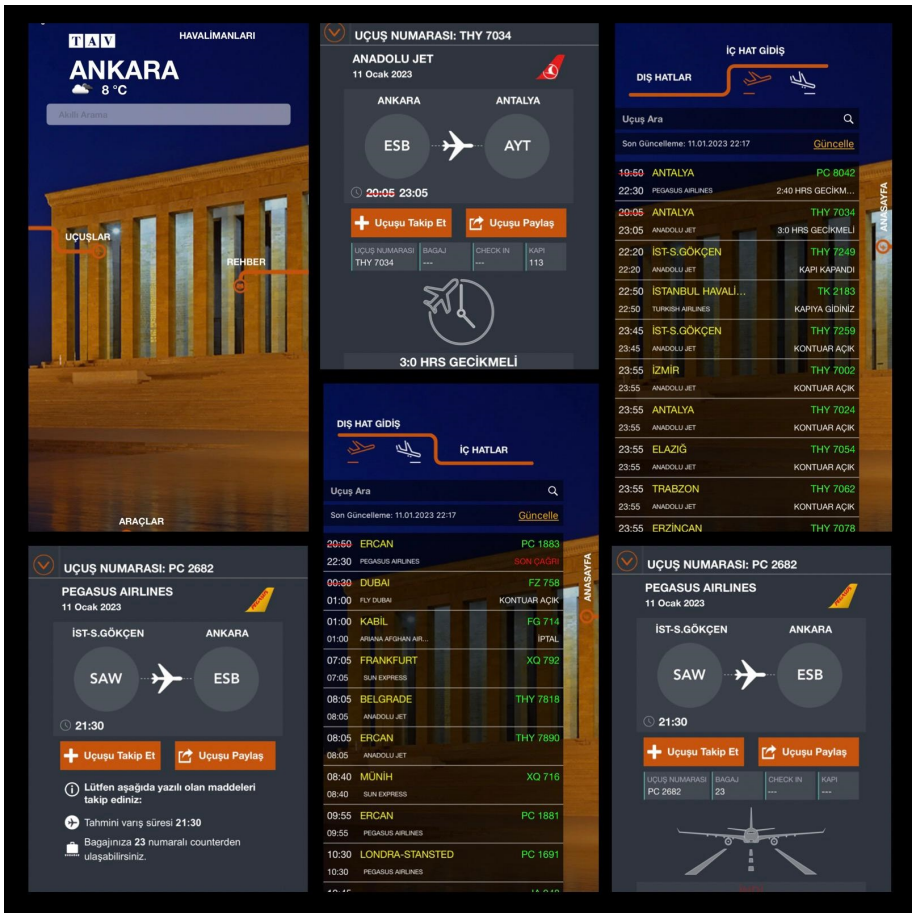
YÖNTEM

5.1 *Olası Senaryoları Gerçekleştirmek İçin Kullanılabilecek Veri Kümesi*

Uçak kapı atama ve bagaj sıralama işlemlerinde algoritmalar kullanarak problemleri çözmek için gerekli veri kümeleri Türkiye Havalimanları İşletmeleri A.Ş. (TAV) mobil ve web uygulamasından sağlanacaktır. TAV mobil uygulamasından elde edilebilecek veriler; anlık uçuş takibi, tahmin edilen ve gerçekleşen uçak iniş ve kalkış saati,uçuşlar arasındaki süre farkı,uçakların atandığı kapı numarası, uçakların kapı işgal süresi,uçakların tipi, şirket adı gibi bilgilerdir. Şekil 9' da mobil uygulamanın ekran görüntüleri verilmiştir.

5.2 *Çalışma Takvimi*

Çizelge 6'da çalışmamızın aylara göre planlaması yapılmıştır.



Şekil 9: TAV Mobil Uygulaması Ekran Görüntüleri

Çizelge 6: Aylara Göre Çalışma Planlama Tablosu

ZAMAN ARALIĞI	YAPILAN ÇALIŞMA
KASIM	Proje danışmanı ile iletişime geçildi. Proje ile ilgili literatür çalışmaları araştırıldı. Bu çalışma konu başlıkları şunlardır: "Havaalanı Uçak Kapı Atamaları", "Havaalanı Bagaj Atama İstasyonları" vb. başlıklı çalışmalara bakıldı.

Çizelge 6: Aylara Göre Çalışma Planlama Tablosu

ZAMAN ARALIĞI	YAPILAN ÇALIŞMA
ARALIK	Kasım ayı süresince yapılan araştırmalar daha detaylandırılarak rapor yazımına başlandı. Bununla eş zamanlı olarak atama problemleri dışında istenen uçak bilet alım sitesinin önyüz geliştirmelerine başlandı.
OCAK	Raporun teslim tarihinden önce rapor üzerinde son düzenlemeler gerçekleştirildi. Rapora incelenen literatür özetleri ve önyüz görselleri aktarıldı.Yapılan literatür çalışmaları doğrultusunda grup arkadaşlarıyla atama problemleri çözümü için hangi algoritmanın daha uygun olabileceği tartışıldı.
ŞUBAT	Uçak bilet alım web ekranı üzerinde istenen geliştirmelere göre ekran üzerinde eklemeler yapılacaktır. Bunun yanı sıra öncelik olarak kapı atama problemi ele alınıp uygun algoritmalarla çözümlere başlanacaktır.

Çizelge 6: Aylara Göre Çalışma Planlama Tablosu

ZAMAN ARALIĞI	YAPILAN ÇALIŞMA
MART	Bilet alım sitesinde check-in ve ödeme sistemi işlemleri ele alınacaktır. Eş zamanlı olarak kapı atama problemi için uygun havaalanı verileriyle işlemler gerçekleştirilecektir. Bunun yanı sıra bagaj atama problemi içinde çözümlere başlanacaktır.
NİSAN	Ekran üzerinde geliştirmelere devam edilirken kapı ve bagaj atama problemlerine uygun seçilen algoritmalar üzerinde çalışılarak seçilen havaalanı verileri kullanılıp matematiksel işlemler yapılacaktır.
MAYIS	Nisan ayı süresince yapılan çalışmalar sürdürülüp sonuçlar raporlanacaktır.
HAZİRAN	Projeye ait son kontroller yapılacaktır. Belirtilen tarihe göre rapor düzenlenip teslimi ve sunumu yapılacaktır.

BÖLÜM: IV

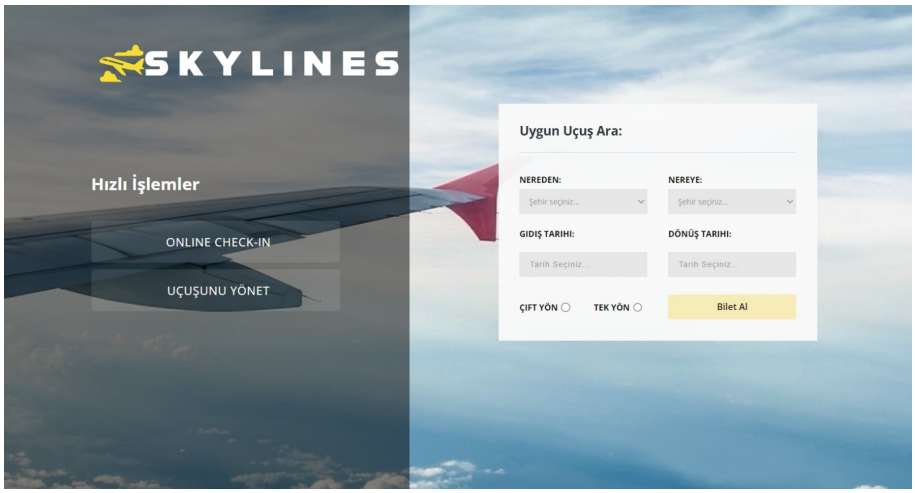
SONUÇ

6

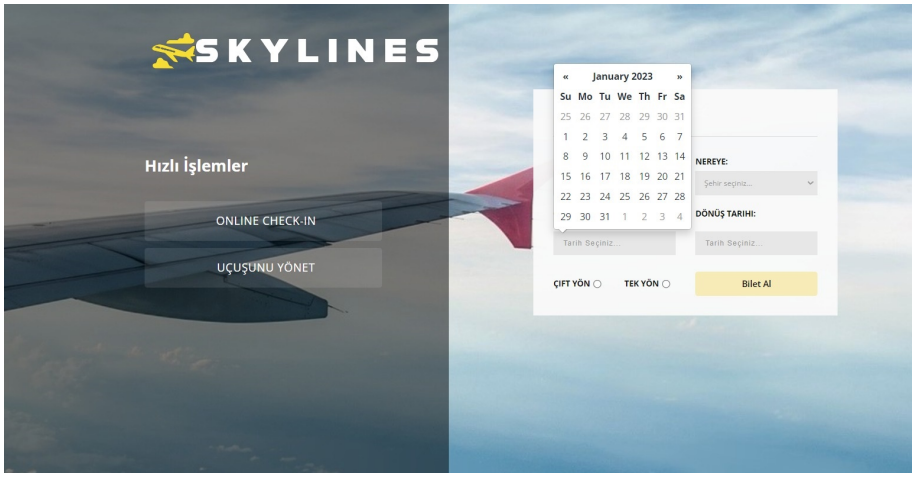
SONUÇ

6.1 *Web Uygulama Ekran Görüntüleri*

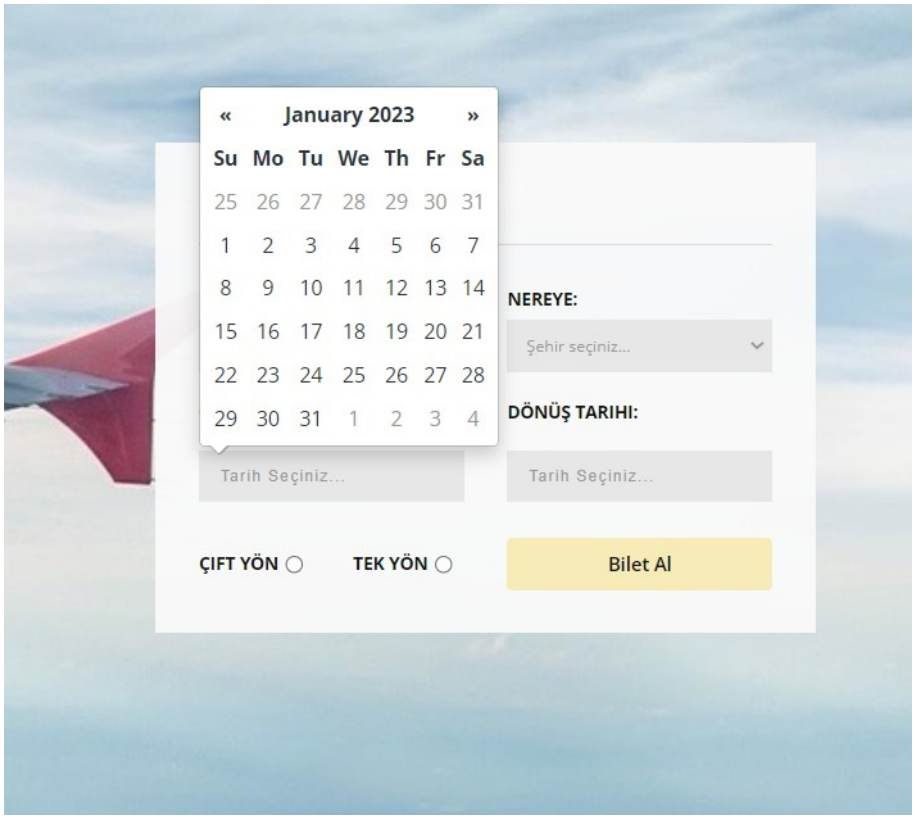
Müşterilerin uçuşlarını planlamaları ve kolayca işlemlerini halledebilmeleri için web uygulamalarının geliştirilmesi gerekli ve önemlidir. Bu kısımda geliştirilmekte olan ve ASP.NET MVC yapısı kullanılarak tasarlanmış web kullanıcı arayüzleri eklenmiştir. Index kısmında bilet alma, uçuş arama gibi işlemler yapılırken qr kod üretmesi sağlanacak olan check-in sayfasına da yönlendirilmesi eklenmiştir.



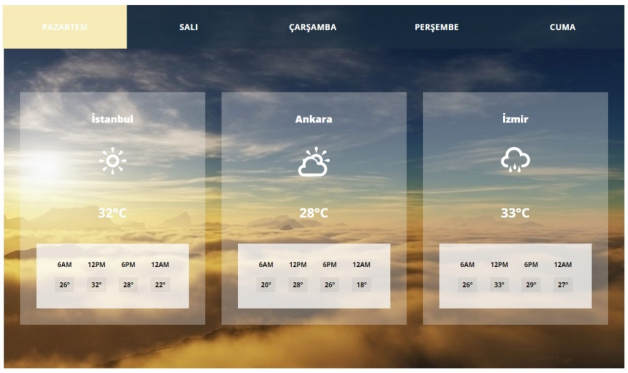
Şekil 10: index.cshtml



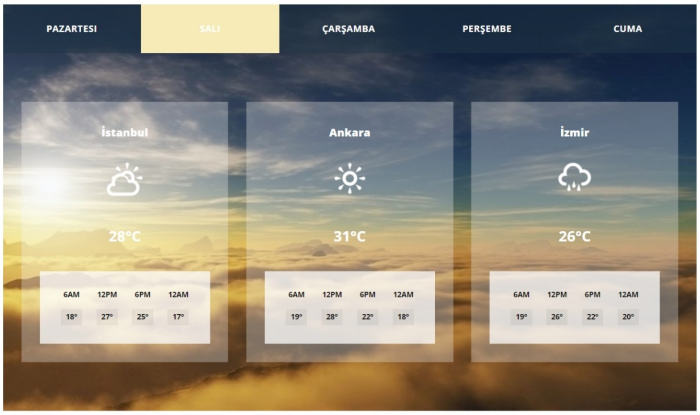
Şekil 11: index.cshtml Tarih Seçici



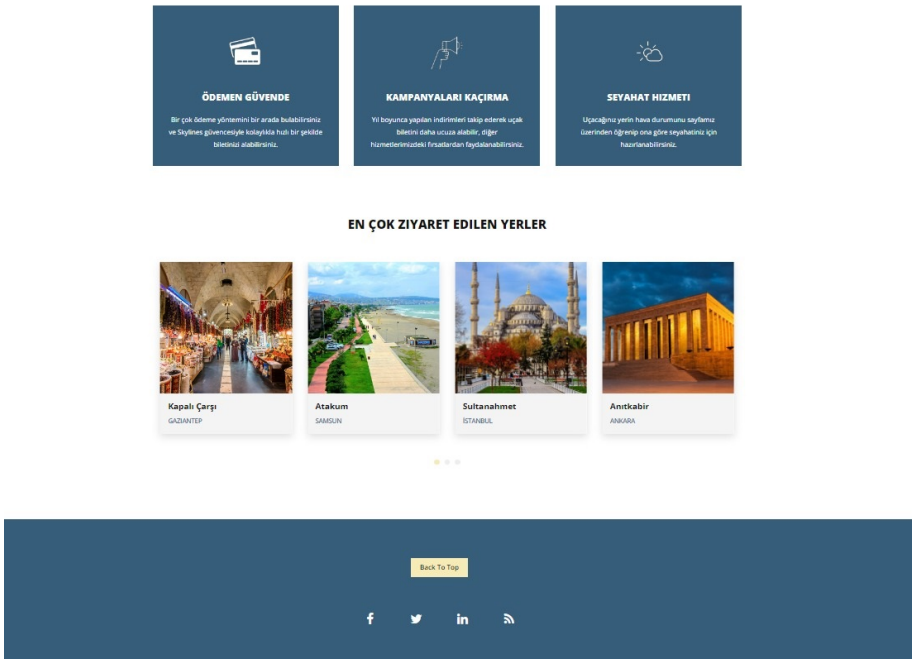
Şekil 12: index.cshtml Tarih Seçici



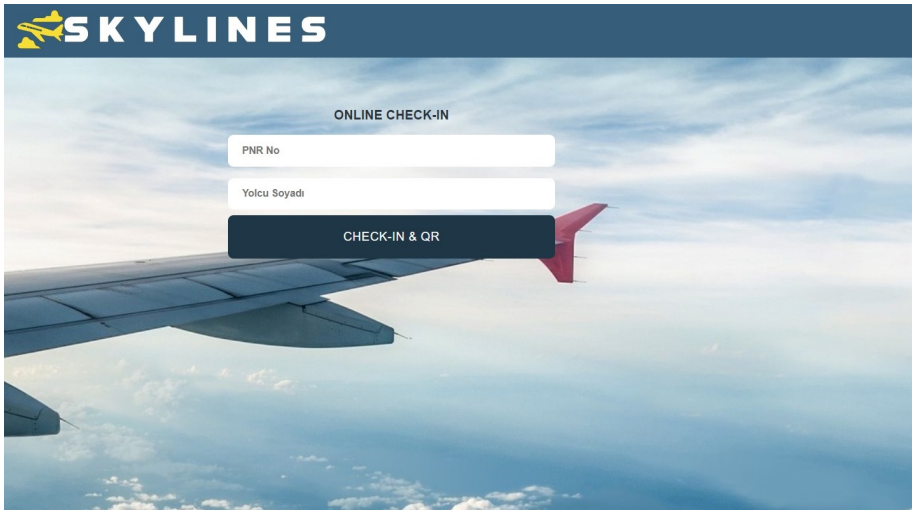
Şekil 13: index.cshtml Hava Durumu



Şekil 14: index.cshtml Hava Durumu



Şekil 15: index.cshtml Ziyaret Seçenekleri



Şekil 16: checkin.cshtml

KAYNAKÇA

[1] ALGORITMA, G. Umut Kapucu. https://www.hypiostech.com/post/genetik-algoritma.

[2] ARSLAN, Ş. Uçakların terminal kapılarına atanması probleminin farklı yöntemlerle çözümü ve uygulaması.

[3] ASCÓ SIGNES, A. An evolutionary algorithm and operators for the airport baggage sorting station problem. *Soft Computing 23* (10 2019), 10055–10083.

[4] BABIC, O., TEODOROVIĆ, D., AND TOSIC, V. Aircraft stand assignment to minimize walking. *Journal of Transportation Engineering-asce - J TRANSP ENG-ASCE 110* (01 1984).

[5] BANDARA, S., AND WIRASINGHE, S. Walking distance minimization for airport terminal configurations. *Transportation Research Part A: Policy and Practice 26* (01 1992), 59–74.

[6] BOLAT, A. Assigning arriving flights at an airport to the available gates. *Journal of The Operational Research Society - J OPER RES SOC 50* (02 1999), 23–34.

[7] BRAAKSMA, J. P., AND SHORTREED, J. H. Improving airport gate usage with critical path. *Transportation Engineering Journal of ASCE 97* (1971), 187–203.

[8] CHENG, Y. A knowledge-based airport gate assignment system integrated with mathematical programming. *Computers & Industrial Engineering 32* (1997), 837–852.

[9] DING, H., LIM, A., RODRIGUES, B., AND ZHU, Y. The over-constrained airport gate assignment problem. *Computers OR 32* (07 2005), 1867–1880.

[10] DORNDORF, U., DREXL, A., NIKULIN, Y., AND PESCH, E. Flight gate scheduling: State-of-the-art and recent developments. *Omega 35* (06 2007), 326–334.

[11] ELEVLİ, P. B. YÖNEYLEM ARAŞTIRMASI . https://avys.omu.edu.tr/storage/app/public/birol.elevli/133461/Ders2_DP_Modelleme.pdf.

[12] ERDOĞMUŞ, P. TÜRKİYEDEKİ GÜÇ SİSTEMİNDE TAVLAMA BENZETİMİ, GENETİK ALGORİTMA ve TABU ARAŞTIRMA ALGORİTMALARI KULLANILARAK EKONOMİK DAĞITIM . https://dergipark.org.tr/tr/download/article-file/186346.

[13] HAGHANI, A., AND CHEN, M. Optimizing gate assignments at airport terminals. *Transportation Research Part A-policy and Practice 32* (1998), 437–454.

[14] HUANG, E., MITAL, P., GOETSCHALCKX, M., AND WU, K. Optimal assignment of airport baggage unloading zones to outgoing flights. *Transportation Research Part E: Logistics and Transportation Review 94* (10 2016), 110–122.

[15] KESER, B. Karınca Kolonisi Optimizasyonu. https://medium.com/@berkekeser/karÄśnca-kolonisi-optimizasyon-algoritmasÄś-4da0b37cb393 Jul 22, 2020.

[16] MANGOUBI, R., AND MATHAISEL, D. Optimizing gate assignment at airport terminals. *Transportation Science 19* (05 1985), 173–188.

[17] RIJSENBRIJ, J., AND OTTJES, J. New developments in airport baggage handling systems. *Transportation Planning and Technology 30* (08 2007), 417–430.

[18] WAYAHDI, M., GINTING, S., AND SYAHPUTRA, D. Greedy, a-star, and dijkstra's algorithms in finding shortest path. *International Journal of Advances in Data and Information Systems 2* (02 2021), 45–52.

[19] YAN, S., AND CHANG, C.-M. A network model for gate assignment. *Journal of Advanced Transportation 32* (1998), 176–189.

[20] YAN, S., AND HUO, C.-M. Optimization of multiple objective gate assignments. *Transportation Research Part A-policy and Practice 35* (2001), 413–432.