

**T.C.
GEBZE TEKNİK ÜNİVERSİTESİ**

Bilgisayar Mühendisliği Bölümü

**UÇAK PİSTİ
OPTİMİZASYONU
PROJESİ RAPORU**

Fatih Selim YAKAR

**Danışman
Prof. Dr. Fatih Erdoğan SEVİLGEN**

**Ocak, 2021
Gebze, KOCAELİ**

**T.C.
GEBZE TEKNİK ÜNİVERSİTESİ**

Bilgisayar Mühendisliği Bölümü

**UÇAK PİSTİ
OPTİMİZASYONU
PROJESİ RAPORU**

Fatih Selim YAKAR

**Danışman
Prof. Dr. Fatih Erdoğan SEVİLGEN**

**Ocak, 2021
Gebze, KOCAELİ**

ÖNSÖZ

Bu projede emeği geçen ve beni her hafta yaptığı toplantılarıyla yönlendiren danışmanım Prof. Dr. Fatih Erdoğan SEVİLGEN hocama, gerektiğinde bana teorik olarak yardımcı olan Mehmet Burak KOCA hocama ve her izleme buluşmalarında beni dinleyen Prof. Dr. Erhan APTOULA hocama en içten teşekkürlerimi sunarım.

Ayrıca eğitimim süresince bana her konuda tam destek veren aileme ve bana bilgi katan, gelişmemde yardımcı olan tüm hocalarıma saygı ve sevgilerimi sunarım.

Ocak, 2021

Fatih Selim YAKAR

İÇİNDEKİLER

ÖNSÖZ	III
İÇİNDEKİLER	IV
ŞEKİL LİSTESİ	VII
TABLO LİSTESİ	IX
KISALTMA LİSTESİ	X
SEMBOL LİSTESİ	XI
ÖZET	12
SUMMARY	14
1. GİRİŞ	16
2. YÖNTEM VE MALZEME	18
 2.1. PROBLEMİN OPTİMİZASYON PROBLEMİ OLARAK	
 TANIMLANMASI	18
 2.1.1. İndeks Kümeleri Ve Notasyon	18
 2.1.2. Karar Değişkenleri	19
 2.1.3. Karma Tamsayı Programlama Formülasyonu	19
 2.2. PROBLEM İÇİN UYGULANAN ALGORİTMALAR VE	
 YAKLAŞIMLAR	19
 2.2.1. ERT Algoritması	20
 2.2.1.1. Algoritma İçin Çözüm Gösterimi	20
 2.2.1.2. Algoritma İçin Çözümün Değeri	21
 2.2.1.3. Algoritmanın Sözde Kodu	21
 2.2.2. AATCSR Algoritması	22
 2.2.2.1. Algoritma İçin Çözüm Gösterimi	22
 2.2.2.2. Algoritma İçin Çözümün Değeri	22
 2.2.2.3. Algoritmanın Sözde Kodu	22
 2.2.3. FPI Algoritması	23
 2.2.3.1. Algoritma İçin Çözüm Gösterimi	24
 2.2.3.2. Algoritma İçin Çözümün Değeri	24
 2.2.3.3. Algoritmanın Sözde Kodu	24

2.2.4. SA Algoritması	25
2.2.4.1. Algoritma İçin Çözüm Gösterimi	25
2.2.4.2. Algoritma İçin Çözümün Değeri	26
2.2.4.3. Algoritmada Kullanılan Başlangıç Çözümü	26
2.2.4.4. Algoritmada Kullanılan Komşuluk	26
2.2.4.5. Hiper Parametreler Ve İşlevleri	26
2.2.4.6. Algoritmanın Akış Şeması	28
2.2.4.7. Algoritmanın Sözde Kodu	28
2.2.5. Meta-Raps Algoritması	29
2.2.5.1. Algoritma İçin Çözüm Gösterimi	30
2.2.5.2. Algoritma İçin Çözümün Değeri	30
2.2.5.3. Algoritmada Kullanılan Diğer Yöntemler	30
2.2.5.4. Algoritmada Kullanılan Komşuluk	30
2.2.5.5. Hiper Parametreler Ve İşlevleri	31
2.2.5.6. Algoritmanın Akış Şeması	32
2.2.5.7. Algoritmanın Sözde Kodu	33
2.2.6. GA Algoritması	34
2.2.6.1. Algoritma İçin Çözüm Gösterimi	34
2.2.6.2. Algoritma İçin Çözümün Değeri	35
2.2.6.3. Fenotip Ve Genotip	35
2.2.6.4. Başlangıç Popülasyonu Oluşturma Mekanizması	35
2.2.6.5. Çaprazlama Mekanizması	36
2.2.6.6. Mutasyon Mekanizması	37
2.2.6.7. Çaprazlama Havuzu Seçimi	37
2.2.6.8. Hayatta Kalanların Seçimi	38
2.2.6.9. Üzerine Yapılacak Yerel Arama	38
2.2.6.10. Hiper Parametreler Ve İşlevleri	38
2.2.6.11. Algoritmanın Akış Şeması	40
2.2.6.12. Algoritmanın Sözde Kodu	40

2.3. PROBLEM İÇİN UYGULANAN DENEYLER VE HİPER PARAMETRE ARALIKLARI	41
2.3.1. GA Hiper Parametreleri İçin Deney Arahları	41
2.3.2. SA Hiper Parametreleri İçin Deney Arahları	41
2.3.3. Meta-Raps Hiper Parametreleri İçin Deney Arahları	42
2.4. PROBLEM İÇİN ARAYÜZ YAPILMASI	42
3. BULGULAR	44
3.1. GA İçin Yapılan Deneyler Ve Bulgular	44
3.1.1. Çaprazlama Tipi Deneyi	45
3.1.2. Popülasyon Boyutu Deneyi	46
3.1.3. Çaprazlama Olasılığı Deneyi	49
3.1.4. İterasyon Sayısı Deneyi	53
3.2. SA İçin Yapılan Deneyler Ve Bulgular	54
3.2.1. Soğuma Katsayısı Deneyi	55
3.2.2. İç Döngü Sayısı Deneyi	58
3.2.3. İterasyon Sayısı Veya En Soğuk Sıcaklık Değeri Deneyi	62
3.2.4. Tekrar Isıtma Sayısı Deneyi	64
3.3. Meta-Raps İçin Yapılan Deneyler Ve Bulgular	66
3.3.1. Sapma Sınırı Deneyi	67
3.3.2. İterasyon Sayısı Deneyi	69
4. TARTIŞMA VE SONUÇ	72
5. KAYNAKLAR	73
6. EKLER	75

ŞEKİL LİSTESİ

ŞEKİL 2.1. İndeks Değiştirme Komşuluğu

ŞEKİL 2.2. SA Algoritması Akış Şeması

ŞEKİL 2.3. Çoklu İndeks Değiştirme Komşuluğu

ŞEKİL 2.4. Meta-Raps Akış Şeması

ŞEKİL 2.5. Fenotip-Genotip Gösterimi

ŞEKİL 2.6. Başlangıç Popülasyonunu Oluşturma

ŞEKİL 2.7. PMX Çaprazlama Mekanizmasının Çalışması

ŞEKİL 2.8. O1 Çaprazlama Mekanizmasının Çalışması

ŞEKİL 2.9. Mutasyon Mekanizmasının Çalışması

ŞEKİL 2.10. Rulet Tekerisi Seçimi Gösterimi

ŞEKİL 2.11. GA Algoritmasının Akış Şeması

ŞEKİL 2.12. GA' nın Sözde Kodu

ŞEKİL 2.13. Program İçin tkinter Arayüzü

ŞEKİL 2.14. Programın bokeh Sunucusunun Çalışması

ŞEKİL 2.15. Programın Sıralama Bilgilerini Göstermesi

ŞEKİL 3.1. Çaprazlama Tipi Deney Sonucu Bar Grafiği

ŞEKİL 3.2. Popülasyon Boyutuna Göre Çalışma Zamanları Dağılımı Grafiği

ŞEKİL 3.3. Popülasyon Boyutuna Göre En İyi Olmayan Değerler Dağılımı Grafiği

ŞEKİL 3.4. Popülasyon Boyutuna Göre Ölçekli Hata Dağılımı Grafiği

ŞEKİL 3.5. Çaprazlama İhtimaline Göre Çalışma Zamanları Dağılımı Grafiği

ŞEKİL 3.6. Çaprazlama İhtimaline Göre En İyi Olmayan Çözüm Sayısı Dağılımı Grafiği

ŞEKİL 3.7. Çaprazlama İhtimaline Göre Ölçekli Hata Dağılımı Grafiği

ŞEKİL 3.8. İterasyona Göre Ölçekli Hata Dağılımı Grafiği

ŞEKİL 3.9. Soğuma Katsayısına Göre En İyi Olmayan Sayısı Dağılımı Grafiği

ŞEKİL 3.10. Soğuma Katsayısına Göre Ölçekli Hata Dağılımı Grafiği

ŞEKİL 3.11. İç Döngü Miktarına Göre Çalışma Zamanı Dağılımı Grafiği

ŞEKİL 3.12. İç Döngü Miktarına Göre En İyi Olmayan Çözüm Dağılımı Grafiği

ŞEKİL 3.13. İç Döngü Miktarına Göre Ölçekli Hata Dağılımı Grafiği

ŞEKİL 3.14. İterasyon Sayısına Göre Ölçekli Hata Dağılımı Grafiği

ŞEKİL 3.15. Tekrar Isıtma Sayısına Göre Ölçekli Hata Dağılımı Grafiği

ŞEKİL 3.16. Sapma Olasılığı Sınırına Göre En İyi Olmayan Çözüm Sayısı Dağılımı Grafiği

ŞEKİL 3.17. Sapma Olasılığı Sınırına Göre Ölçekli Hata Dağılımı Grafiği

ŞEKİL 3.18. İterasyona Göre Ölçekli Hata Dağılımı Grafiği

ŞEKİL 6.1. Tüm Deneylerin Çalıştırılması Sonucu Elde Edilen Ölçekli Hata Dağılımı Grafiği

ŞEKİL 6.2. Tüm Deneylerin Çalıştırılması Sonucu Elde Edilen Çalışma Zamanları Dağılımı Grafiği

ŞEKİL 6.3. Tüm Deneylerin Çalıştırılması Sonucu Elde Edilen En İyi Olmayan Çözüm Sayısı Dağılımı Grafiği

ŞEKİL 6.4. Arayüzün GA ve 59. Veri Örneği İle Çalıştırılması

ŞEKİL 6.5. Arayüzün Sıralama Bilgisini Göstermesi

TABLO LİSTESİ

TABLO 3.1. Çaprazlama Tipi Deney Sonucu Tablosu

TABLO 3.2. Popülasyon Boyutu Deney Sonucu Tablosu

TABLO 3.3. Çaprazlama Olasılığı Deney Sonucu Tablosu

TABLO 3.4. GA İterasyon Sayısı Deneyi Tablosu

TABLO 3.5. SA Soğuma Katsayısı Deneyi Tablosu

TABLO 3.6. SA İç Döngü Sayısı Deneyi Tablosu

TABLO 3.7. SA İterasyon Sayısı Deneyi Tablosu

TABLO 3.8. SA Tekrar Isıtma Sayısı Deneyi Tablosu

TABLO 3.9. Meta-Raps Sapma Sınırı Deneyi Tablosu

TABLO 3.10. İterasyon Sayısı Deneyi Tablosu

TABLO 6.1. Tüm Deneylerin Çalıştırılması Sonucu Elde Edilen Tablo

KISALTMA LİSTESİ

ALP	: Uçak İniş Problemi
ATP	: Uçak Kalkış Problemi
AATCSR	: Ayırma ve Hazır Süreleriyle Uyarlanmış Görünür Gecikme Maliyeti Algoritması
FPI	: Hızlı Öncelik İndeksi Algoritması
ERT	: En Erken Hazır Zamanları Algoritması
SA	: Benzetimli Tavlama Algoritması
Meta-Raps	: Rastgele Öncelikli Arama Algoritması İçin Meta-sezgisel Yöntem
GA	: Genetik Algoritma

SEMBOL LİSTESİ

T	: SA' deki Sıcaklık
α	: SA' deki Soğuma Katsayısı
J	: Uçak Kümesi
r_j	: Taksi süresi hesaba katılmadan, kalkış veya iniş için hazırlanma zamanı
δ_j	: Taksi süresi hesaba katılmadan, kalkış veya iniş için hedef zaman
d_j	: Taksi süresi hesaba katılmadan, kalkış veya iniş için son zaman
O_j	: Operasyon tipi (kalkış veya iniş)
C_j	: Ağırlık sınıfı (küçük, büyük, ağır)
W_j	: Ağırlık sınıfı ve operasyon tipi katılarak bulunan öncelik
S_{kj}	: k ve j uçağı arasındaki minimum ayrılma zamanı
t_j	: j uçağının başlama zamanı

ÖZET

Havaalanlarında pist verimini belirleyen ana faktörlerden biri, iniş ve kalkış sırasında uçaklar arasında gerekli ayırmadır. Uçak ayımı için önde gelen ve takip eden uçak tipine bağımlılık, sıralama ve programlamayı önemli bir sorun haline getirir. Uçak İniş Problemi (ALP) ve Havaalanı Kalkış Problemleri (ATP), çeşitli operasyonel faaliyetlere tabi olarak, belirli hedefleri optimize etmek amacıyla, havaalanlarında mevcut pistlere inen veya kalkış yapan uçakların sırasını belirlemeye çalışır. Uçağın iniş ve kalkışlarının en uygun şekilde sıralanması teoride pist kapasitesini istenen amaca bağlı olarak arttırmır. Bu bitirme projesinde ise konu olarak ALP ve ATP üzerinde bu problemi çözmek ele alındı.

Bu projedeki asıl amaç ALP ve ATP problemlerini literatürde halihazırda olan ve literatürde olmayan yöntemlerle çözmek ve verimliliği artırmaktır. Bu bağlamda bu problemlerin birer optimizasyon problemi olduğunu unutmamak gereklidir. Yani bu problemin en iyi çözümü yoktur. Bu projedeki amaçlardan biride bu problemi en iyiye yaklaştıracak şekilde çözmektir. Bunu ise farklı algoritmalar kullanıp algoritmalar içindedede hiper parametrelerle bir optimizasyon yapıp sağlamak amaçlandı. Öncelikle proje için bu konuya ilgili bir optimizasyon problemi tanımlandı. Tanımlanan optimizasyon problemi literatürde olan formüller ve algoritmaların devşirilen yöntemler kullanılarak çözüldü. Bu yöntemler açgözlü yöntem olan AATCSR, FPI, ERT ve meta-sezgisel yöntemler olan SA(kendi içerisinde 3 tip), Meta-Raps(kendi içerisinde 3 tip)' di. Bunlara ek olarak literatürde olmayan ve optimizasyon problemine yeni uygulanan GA ilede çözüldü. Bütün bunlardan sonra projedeki amaç verimliliği artırmak ve en iyiye yaklaşmak olduğundan ötürü meta-sezgisel algoritmalar özelinde ek olarak hiper parametre optimizasyonu için deneysel çalışmalar yapıldı. En sonunda ise bu bulunan sonuçları gerçek bir havalimanına uygulamak amaçlandı.

Sonuç olarak elde edilen deneysel hiper-parametre optimizasyonu testlerine göre algoritmaların kendi içindeki en iyi ayarları bulundu, tablolandı ve grafiklendi. Bu bağlamda en iyi sonuçları çikaran algoritma' nın SA algoritması olduğu, ardından GA ve onun ardından ise Meta-Raps, en son ise aç gözlu algoritmalar olduğu kanısına varıldı.

SUMMARY

One of the main factors determining runway efficiency at airports is the necessary separation between aircraft during landing and take-off. The dependence on the leading and following aircraft type for aircraft separation makes sequencing and programming a major problem. Airplane Landing Problem (ALP) and Airport Departure Problems (ATP) attempt to determine the sequence of aircraft landing or taking off on runways available at airports, in order to optimize specific targets, subject to various operational activities. The optimal arrangement of the landing and take-offs of the aircraft increases the runway capacity in theory depending on the desired purpose. In this graduation project, solving this problem on ALP and ATP was discussed.

The main purpose of this project is to solve ALP and ATP problems with methods that are available in the literature and not in the literature, and to increase efficiency. In this context, it should not be forgotten that these problems are optimization problems. So there is no best solution to this problem. One of the aims of this project is to solve this problem in a way that brings it closer to the best. It was aimed to achieve this by using different algorithms and making an optimization with hyper parameters within the algorithms. First of all, an optimization problem related to this subject was defined for the project. The defined optimization problem was solved by using formulas and methods derived from algorithms in the literature. These methods were the greedy method AATCSR, FPI, ERT and meta-heuristic methods SA (3 types in itself), Meta-Raps (3 types in itself). In addition to these, it was solved with GA which is not available in the literature and newly applied to the optimization problem. After all of these, since the aim of the project is to increase efficiency and approach the best, experimental studies were carried out for hyper parameter optimization in addition to meta-heuristic algorithms. Finally, it was aimed to apply these results to a real airport.

According to the experimental hyper-parameter optimization tests obtained as a result, the best settings within the algorithms were found, tabulated and plotted. In this context, it was concluded that the algorithm that produces the best results is the SA algorithm, followed by GA and then Meta-Raps, and lastly the greedy algorithms.

1. GİRİŞ

Günümüzde uçakla seyahat etmek herkes için normalleşmiş ve günlük hayatı yaygın olan bir olaydır. Fakat uçakla seyahatin arka planında uçakların pistte taksi alanları arasındaki en iyi rotanın bulunması, kalkış ve iniş yapan uçakların belirli kısıtlar dahilinde sıralanması ve çizelgelenmesi gibi işlemler mevcuttur. Bu işlemlerin en iyi şekilde yapılması ise bir havaalanı için kritik önem taşır. Bu sayede havaalanı aynı seviyedeki işleri daha erken bitirdiği için zamandan ve yoğunluktan kazanç sağlar. Bu kazanç ise pist kapasitesi, birim zaman başına yapılan işlem sayısı ve doğrudan maddi harcama gibi kaynakların daha uygun şekilde kullanılması yani verimliliğini sağlar. Bahsedilen işlemler Hava Trafik Kontrolörleri tarafından deneyim bazlı yapılabileceği gibi bazı yazılımlar tarafından da yapılabilir. Bu bağlamda yapılan projede uçakların iniş ve kalkışlarındaki sıralamalar belirli yöntemlerle ele alınacaktır.

Bütün bu anlatılanlar eşliğinde ele alınan problemi çözmenin en iyi yolu öncelikle literatür üzerinde yapılan benzer veya alakalı çalışmaları inceleyip problemi anlamaktan geçiyordu. Hava Trafik Kontrolörlüğü ve bu alandaki terminolojiyi öğrenmek ve hakim olmak amacıyla bazı yayın ve makaleler üzerinde araştırmalar yapıldı.[1][2][3][4][5][6][7][8][9][10][11]

Bağlam hakkında yeterli bilgi sağlandıktan sonra ele alınacak problem ve benzer problemleri çözmeyi amaçlayarak yazılmış yayınlar ve makaleler üzerinde araştırmalar yapıldı. Buradan elde edilen bilgi birikimi ve çeşitli bilgiler dahilinde ana bir makaleye bağlı kalınarak karar değişkenleri, objektif fonksiyon, problem gösterimi gibi önemli ve temel işlemler yapılarak yeni bir optimizasyon problemi devşirildi.[1]

Devşirilen bu optimizasyon probleminde öncelikle AATCSR, FPI, ERT ağözlü algoritmalar kullanılarak bazı temel çözümler yapıldı. ERT algoritması en basit aç gözlü çözümü içeriyordu. İlk gelen uçak ilk sırayı alır mantığıyla ağözlü seçimler yapıp bir

sıralama elde ediyordu. Bundan sonra gelen AATCSR ve FPI algoritmaları ise belirli formüller kullanarak bulunan öncelik indekslerine göre sıralama yapıyordu. Ayrıca projenin önemli kısmını içeren meta-sezgisel algoritmalar içeriklerinde bulunan sezgilere göre işlem yaparak yerel en iyiden çıkış evrensel en iyiye ulaşmaya çalışıyordu. Bu algoritmalar ise SA, GA ve Meta-Raps' dı. Buradaki ana amaç ise sadece bu algoritmaları tanımından kullanmak değil içerisindeki hiper parametreleri kullanarak kendi içlerinde de en iyi parametre ayarlarını bulmaktı.

Rapor tüm yukarıda bahsedilen ve kısaca bilgi verilen konular detaylıca açıklanması yani sırasıyla problemin optimizasyon problemi olarak tanımlanması, çözümlerin açıklanması, çözümlerin uygulanması, çözümler kendi içindeki hiper parametre optimizasyonunun yapılması, hiper parametre optimizasyonu gereğince yapılan deneyler ve sonuçları ardından ise yapılan arayüz açıklanacak şekilde devam edecktir.

2. YÖNTEM VE MALZEME

Bu projede öncelikle genel bir plan belirlendi. Öncelikle araştırılan literatürdeki makalelerden ana bilgi kaynağı olarak kullanılacak bir makale seçildi.[1] Bu makaleye dayanarak bir optimizasyon problemi tanımlandı. Optimizasyon problem literatürden farklı olarak çoklu pist olmasını desteklemeyecek şekilde yeniden düzenlenendi.

2.1. PROBLEMİN OPTİMİZASYON PROBLEMİ OLARAK TANIMLANMASI

Projedeki optimizasyon problemini tanımlarken literatürde geçen ve aslında çoklu pist için tanımlanmış bir problem tanımı baz alındı.[3]

2.1.1. İndeks Kümeleri Ve Notasyon

$J = \{1, 2, \dots, n\}$: n tane farklı uçak. (kalkış veya iniş)

r_j : taksi süresi hesaba katılmadan, kalkış veya iniş için hazırlanma zamanı

δ_j : taksi süresi hesaba katılmadan, kalkış veya iniş için hedef zaman

d_j : taksi süresi hesaba katılmadan, kalkış veya iniş için son zaman

O_j : operasyon tipi (kalkış veya iniş)

C_j : ağırlık sınıfı (küçük, büyük, ağır)

W_j : ağırlık sınıfı ve operasyon tipi katılarak bulunan öncelik

s_{kj} : k ve j uçağı arasındaki minimum ayrılma zamanı

2.1.2. Karar Değişkenleri

t_j : j uçağının başlama zamanı

2.1.3. Karma Tamsayı Programlama Formülasyonu

$$r_j \leq t_j \leq d_j, \quad \forall j \in J \quad (2.1)$$

$$t_j \geq t_k + s_{kj} - (d_k - r_j + s_{kj}), \quad \forall k, j \in J, \quad k \neq j \quad (2.2)$$

$$T_j \geq t_j - \delta_j, \quad \forall j \in J \quad (2.3)$$

$$0 \leq T_j \leq d_j - \delta_j, \quad \forall j \in J \quad (2.4)$$

$$\text{küçült } \sum_{j \in J} \max(r_j, t_{j-1} + s_{j-1j}) \quad (2.5)$$

İzin verilen zaman aralığı sınırını verir.(1) Herhangi iki uçak arasında uygun ayrımların tahsis edilmesini sağlar.(2) Hedef zamanlara göre uçak geç kalmasını ifade eder.(3) Negatif olmaması ve mantıklı sınırlar içinde olması için sınırları belirler(4). Objektif fonksiyonu (5) toplam uçakların pistten indirilmesi veya kaldırılması zamanını en aza indirmeyi amaçlar.

2.2. PROBLEM İÇİN UYGULANAN ALGORİTMALAR VE YAKLAŞIMLAR

Problemi çözmek amacıyla üç adet açgözlü algoritma ve üç ana meta-sezgisel algoritma uygulandı. Uygulanan algoritmalar raporun devamında ayrıntısıyla açıklanacaktır. Ayrıca bahsedilecek ve uygulanmış olan tüm algoritmalar piton3.7(python3.7) sürümünde, Ghoniem and Farhadi(2012) tarafından sağlanan veri kümesi üzerinde, Intel Core i5 2,7 GHz Dual-Core CPU, Intel Iris Graphics 6100 GPU ve 8GB 1867 MHz DDR3 donanım ile uygulanmış ve test edilmiştir.

2.2.1. ERT Algoritması (En Erken Hazır Zamanları Algoritması)

Bu kısımda en basit ve pratikte hava trafik kontrolörleri tarafından en çok kullanılan açgözlü sıralama algoritması uygulandı. Ayrıca ERT kuralı, üretim süresinin en azı indirilmesi ve maksimum gecikme gibi farklı nesnel işlevlere sahip sorunlara da uygulanmıştır (Larson ve Dessouky, 1978; Damodaran ve Gallego, 2010). Bu algoritmada uçaklar hazır olma zamanlarına göre en erken olan en öncelikli olacak biçimde bir sıralamaya tabi tutulur. Böyle bir yaklaşım literatürde (örneğin, Tsai ve Lee, 1996; Jeong ve Kim, 2008) başarılı bir başlangıç çözümü oluşturmak için kullanılmıştır. Diğer bir yandan, ERT kuralının çözüm kalitesi ve hesaplama süreleri, diğer yöntemlerin karşılaştırılabileceği bir temel olarak kullanılabilir.

2.2.1.1. *Algoritma İçin Çözüm Gösterimi*

Algoritmalar kurgulanırken belirli bir çözüm gösterimi gerekiyordu. Ve bu çözüm gösterimine göre algoritma işlem yapacak, komşuluklar buna göre belirlenecek ve sonuç bu şekilde gösterilecekti. Bu problemde, bundan sonraki algoritmalarada da kullanmak üzere permütasyon çözüm gösterimi seçildi. Bu gösterim bize meta-sezgisel algoritmalarla çözümümüzü yaparken belirli iskeletlere sokmamız konusunda faydalayacaktı. Beş adet uçak bulunan bir problemin çözüm gösterimi şu şekildeydi:

0, 3, 2, 1, 4

Buradaki gösterimde beş adet uçak sıralandığında sıralamanın ilkinin sıfırıncı uçak ve sonuncunun ise dördüncü uçak olacağı ifade edildi.

2.2.1.2. Algoritma İçin Çözümün Değeri

Çözümün değerini ifade ederken ise amaç fonksiyonunun programlamaya dökülmüş hali kullanıldı. Bu bağlamda iki uçaklı bir problemin çözümü olan 0, 1 permütasyonunun çözüm değeri şu şekilde bulunuyordu:

$$\begin{aligned} & \text{hazırlanma süresi}(0) + \max(\text{hazırlanma süresi}(1), \text{başlama zamanı}(0) \\ & + \min(\text{ayırılma süresi}(0,1)) \end{aligned} \quad (2.6)$$

2.2.1.3. Algoritmanın Sözde Kodu

ERT açgözlü algoritması:

*Uçak sayısı boyutunca uçak dizisini oluştur
Seçilen uçağı elemek için J dizisini oluştur
Başlama zamanı dizisi oluştur*

Veri kümesi içindeki en erken hazır zamanına sahip olan uçağı seç

Uçak dizisine seçilen uçağı ekle

Başlama zamanı dizisine başlama zamanını ekle

J dizisinden uçağı sil

Uçak dizisi boyutunca tekrarla

J dizisi boyunca tekrarla

En erken hazır zamanına sahip olan uçağı seç

Tekrar bitisi

J dizisinden uçağı sil

Başlama zamanını hesapla ve ekle

Uçak dizisine seçilen uçağı ekle

Tekrar bitisi

2.2.2. AATCSR Algoritması (Ayırma ve Hazır Süreleriyle Uyarlanmış Görünür Gecikme Maliyeti Algoritması)

Burada, Lee andutorso (1997) tarafından sunulan ve [1] makalesinde geliştirilen ATCS kuralının bir uzantısı olarak ASP için AATCSR bileşik açgözlü algoritması kullanıldı. Kullanılan AATCSR sezgisel yöntemi, her bir uçak seçildikten sonra açgözlülük gereği bir anlamda dinamiktir; kalan uçaklar, Denklemde verilen öncelik indeksine göre önceliklendirilir. (7). Heniz programlanmamış her uçak için öncelik indeksi hesaplanır ve en yüksek indeks değerine sahip olan uçak seçilerek sıralama yapılır.

$$\begin{aligned}\pi_j(t, k) = & \exp(-\text{maks}(r_j - t, 0)) x \exp(-s_{kj}) x \\ & \exp(-\text{maks}(\delta_j - t, 0)) x \exp(-\text{maks}(d_j - t, 0))\end{aligned}\quad (2.7)$$

2.2.2.1. Algoritma İçin Çözüm Gösterimi

Geçerli algoritmada önceki yöntemde de belirtilen 2.2.1.1. numaralı başlıkta açıklanan ve örneklenen permütasyon gösterimi kullanıldı

2.2.2.2. Algoritma İçin Çözümün Değeri

Geçerli algoritmada 2.2.1.2. numaralı başlıkta açıklanan ve örneklenen amaç fonksiyonu ile hesaplanan çözüm değeri kullanıldı.

2.2.2.3. Algoritmanın Sözde Kodu

AATCSR açgözlü algoritması:

*Uçak sayısı boyutunca uçak dizisini oluştur
Seçilen uçağı elemek için J dizisini oluştur
Başlama zamanı dizisi oluştur*

Veri kümesi içindeki en erken hazır zamanına sahip olan uçağı seç
Uçak dizisine seçilen uçağı ekle
Başlama zamanı dizisine başlama zamanını ekle
J dizisinden uçağı sil
Uçak dizisi boyutunca tekrarla
J dizisi boyunca tekrarla
En büyük $\pi_j(t, k)$ indeksine sahip uçağı seç
Tekrar bitisi
J dizisinden uçağı sil
Başlama zamanını hesapla ve ekle
Uçak dizisine seçilen uçağı ekle
Tekrar bitisi

2.2.3. FPI Algoritması (Hızlı Öncelik İndeksi Algoritması)

Kullanılan FPI sezgisel yöntem, her bir uçak seçildikten sonra açgözlülük gereği bir anlamda dinamiktir; kalan uçaklar, Denklemde verilen öncelik indeksine göre önceliklendirilir. (8). Heniz programlanmamış her uçak için öncelik indeksi hesaplanır ve en yüksek indeks değerine sahip olan uçak seçilerek sıralama yapılır. AATCSR'den farklı olarak, FPI' de uçağın programlanmasıın aciliyeti üstel olmaktan çok doğrusal bir şekilde ele alınır, bu da onu hesaplama süresi açısından çok daha hızlı hale getirir. Sıfıra bölünmekten kaçınmak için paydalara dikkat etmek gereklidir.

$$\begin{aligned}
 FPI_j(t, k) = & 1 / \text{maks}(r_j - t, 1) \times 1 / \text{maks}(\delta_j - t, 1) \times \\
 & 1 / (\text{maks}(\delta_j - t, 1)) \times 1 / s_{kj}
 \end{aligned} \tag{2.8}$$

2.2.3.1. *Algoritma İçin Çözüm Gösterimi*

Geçerli algoritmada önceki yöntemde de belirtilen 2.2.1.1. numaralı başlıkta açıklanan ve örneklenen permütasyon gösterimi kullanıldı

2.2.3.2. *Algoritma İçin Çözümün Değeri*

Geçerli algoritmada 2.2.1.2. numaralı başlıkta açıklanan ve örneklenen amaç fonksiyonu ile hesaplanan çözüm değeri kullanıldı.

2.2.3.3. *Algoritmanın Sözde Kodu*

FPI açgözlü algoritması:

*Uçak sayısının boyutunca uçak dizisini oluştur
Seçilen uçağı elemek için J dizisini oluştur
Başlama zamanı dizisi oluştur*

Veri kümesi içindeki en erken hazır zamanına sahip olan uçağı seç

Uçak dizisine seçilen uçağı ekle

Başlama zamanı dizisine başlama zamanını ekle

J dizisinden uçağı sil

Uçak dizisi boyutunca tekrarla

J dizisi boyunca tekrarla

En büyük $FPI_j(t, k)$ indeksine sahip uçağı seç

Tekrar bitışı

J dizisinden uçağı sil

Başlama zamanını hesapla ve ekle

Uçak dizisine seçilen uçağı ekle

Tekrar bitışı

2.2.4. SA Algoritması (Benzetimli Tavlama Algoritması)

SA, Metropolis vd. (1956)[1]'nin çalışmasına dayanan iyi bilinen meta sezgisel algoritmalarlardan biridir. Bir dizi fiziksel enerji üretecek soğutma katılarındaki enerji seviyelerini simüle eden durumlar. Kirkpatrick vd. (1983)[1] bu yaklaşımı, bir maliyet fonksiyonunun global optimumunu veya optimuma yakını bulmak için kombinatoryal optimizasyon problemlerini çözmek için uygulamıştır. Genel olarak iyi bir çözüm sağladığından ve istatistiksel olarak optimal bir çözüm bulmayı garanti ettiğinden, sağlam bir meta-sezgisel algoritma olarak kabul edilir. Bazen $e^{-\Delta/T}$ kabul olasılığı ile daha kötü bir komşuluk hareketini kabul ederek yerel optimadan kaçmak için bir mekanizmaya sahiptir, burada Δ , mevcut çözümün ve aday çözümün amaç fonksiyon değerlerindeki farktır, T bir sıcaklık kontrol parametresidir. fiziksel tavlama analojisindeki sıcaklığa karşılık gelir. T yüksek olduğunda, çoğu hareket (daha iyi ve daha kötü) kabul edilecek ve T düşükçe, daha kötü hareketler reddedilecektir. Bu nedenle, yerel minimumda sıkışıp kalmayı önlemek için, algoritmanın başlangıcında nispeten yüksek bir T değeri ayarlanır. Özellikle bu problemde, sabit bir değer yerine, başlangıç sıcaklığı, başlangıç sıcaklığının daha esnek olmasını ve makul değerler olmasını sağlayan mevcut açgözlü çözümün amaç fonksiyon değerinin bir fonksiyonu olarak tanımlanır. SA, $T_k = aT_{k-1}$ fonksiyonuna göre sıcaklıkta düşüş yaşarken, her sıcaklıkta mevcut çözümün çevresini belirli bir komşuluk ile araştırır.

2.2.4.1. *Algoritma İçin Çözüm Gösterimi*

Geçerli algoritmada önceki yöntemde de belirtilen 2.2.1.1. numaralı başlıkta açıklanan ve örneklenen permütasyon gösterimi kullanıldı

2.2.4.2. *Algoritma İçin Çözümün Değeri*

Geçerli algoritmada 2.2.1.2. numaralı başlıkta açıklanan ve örneklenen amaç fonksiyonu ile hesaplanan çözüm değeri kullanıldı.

2.2.4.3. *Algoritmada Kullanılan Başlangıç Çözümü*

Geçerli algoritmada daha önceki bölümlerde bahsedilen açgözlü çözümler burada geliştirilecek olan ilk çözüm olarak ele alındı.

2.2.4.4. *Algoritmada Kullanılan Komşuluk*

Algoritmada komşuluk olarak “İndeks Değiştirme” komşuluğu kullanıldı. Bu komşuluk bir çözümden diğer bir çözümü türetmek için permütasyon içerisindeki rastgele aynı olmayan iki farklı indeksteki uçakların yerini değiştirmeyi amaçlıyordu. Dört uçaaklı bir problemde rastgele seçilen indekslerin birinci ve üçüncü olması durumdaki değişim şu şekildeydi:



ŞEKİL 2.1. İndeks Değiştirme Komşuluğu

2.2.4.5. *Hiper Parametreler Ve İşlevleri*

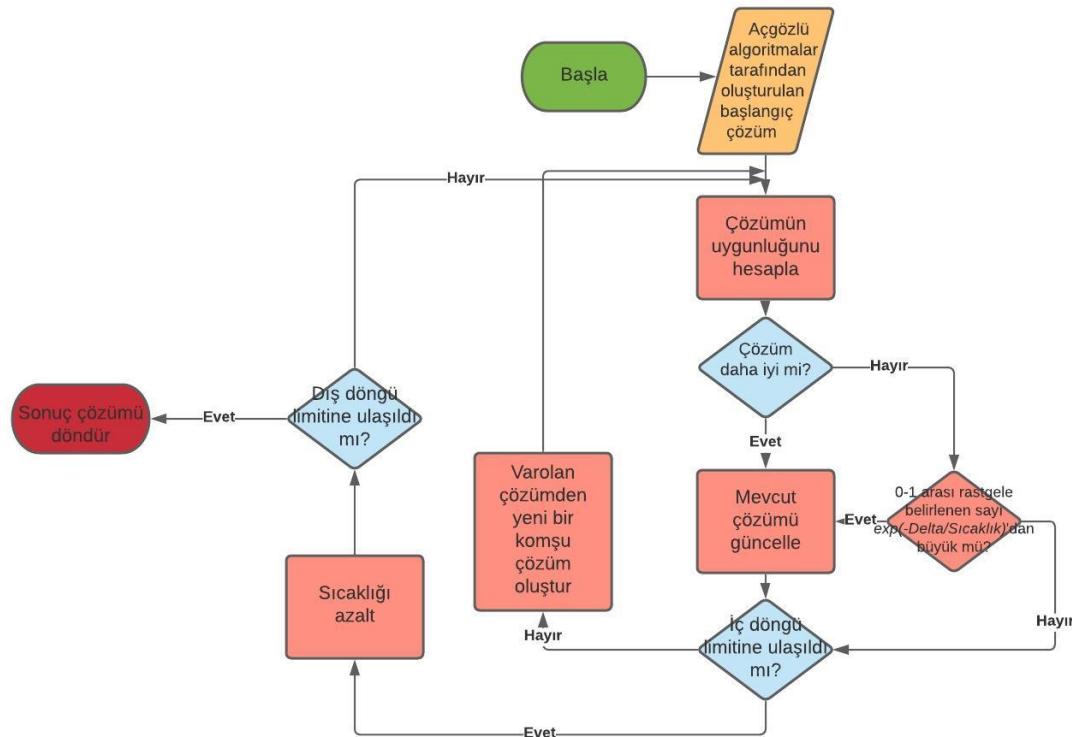
- Başlangıç sıcaklığı: Başlangıç sıcaklığı en baştaki kötü çözümü kabul etme ihtimalini belirler. Çok büyük olduğu durumda kontrol koşulu olan $e^{-\Delta/T}$ değeri küçüleceği için gelen her kötü çözümü kabul etme ihtimali çok artacaktır. Çok küçük olduğu durumda ise kötü çözüm kabul etmeyip basit bir yerel arama yapacaktır. Geçerli algoritmada başlangıç sıcaklığı raporun devamında ayrıntılı

şekilde bahsedilecek deneyler sonucunda açgözlü çözümün objektif fonksiyon değeri olarak belirlendi.

- Soğuma katsayısı: Soğuma katsayı her bir iterasyonda sıcaklığın ne kadar değişeceğini belirleyen bir katsayıdır. Sıfıra yakın olduğu değerlerde sıcaklığı çok fazla düşürecegi için kısa süre içinde yerel aramaya dönüp kötü çözümleri kabul edip mevcut çözümü farklılaştırma işlevini yerine getiremeyecektir. Bire yakın olduğu değerlerde ise çok daha az düşürecegi için fazla kötü çözüm kabul edip belirli bir iyi çözüme odaklanma işlevini yerine getiremeyecektir. Geçerli algoritmada soğuma katsayı raporun devamında ayrıntılı şekilde bahsedilecek deneyler sonucunda 0.95 olarak belirlendi.
- Bitirme kriteri: Bitirme kriteri SA algoritması için oldukça önemlidir. Buna göre algoritmanın ne kadar çalışacağı belirlenir. Bitirme kriteri bir süre, belirli bir iterasyon sayısı veya belirli bir en soğuk sıcaklık olabilir. . Geçerli algoritmada bitirme kriteri en soğuk sıcaklık olarak raporun devamında ayrıntılı şekilde bahsedilecek deneyler sonucunda 0.002 olarak belirlendi.
- Tekrar ısıtma sayısı: Tekrar ısıtma özelliği SA algoritmasının temel özelliklerinden değildir. Fakat uygulandığı taktirde farklı başlangıç sıcaklıklarıyla (bu algoritmada kendi ile 0.8'in çarpımı olarak belirlendi) algoritmayı tekrar tekrar çalıştırmayı sağlar ve daha iyi sonuç bulmak için yararlı olabilir. Geçerli algoritmada tekrar ısıtma sayısı raporun devamında ayrıntılı şekilde bahsedilecek deneyler sonucunda 4 olarak belirlendi.
- İç döngü sayısı: İç döngü sayısı belirli bir sıcaklık değerinde ne kadar komşuluk araması yapılacağını belirler. Bu bağlamda yüksek sıcaklıkta çözümleri kötüsüyle farklılaşımak, düşük sıcaklıkta belirli iyi çözüme odaklanmak için önemlidir.

Geçerli algoritmada tekrar iç döngü sayısı raporun devamında ayrıntılı şekilde bahsedilecek deneyler sonucunda 160 olarak belirlendi.

2.2.4.6. Algoritmanın Akış Şeması



ŞEKİL 2.2. SA Algoritması Akış Şeması

2.2.4.7. Algoritmanın Sözde Kodu

SA algoritması:

*Açgözlü çözümün objektif fonksiyonu kullanılarak tekrar ısıtma sıcaklığı oluştur
En iyi çözümü ve değeri tutacak değişken oluştur*

Tekrar ısıtma sayısı boyunca tekrarla

Sıcaklık=Tekrar ısıtma sıcaklığı

Sıcaklık 0.002'den büyükken tekrarla

İç döngü sayısı 160 olana kadar tekrarla

Komşulukla yeni çözüm oluştur

Yeni çözüm iyiye kabul et değilse formüle göre kabul edip etmeyeceğine karar ver.

En iyi çözümü güncelle

*Sıcaklık=0.95*sıcaklık*

*Tekrar ısıtma sıcaklığı=0.8*tekrar ısıtma sıcaklığı*

2.2.5. Meta-Raps Algoritması (Rastgele Öncelikli Arama Algoritması İçin Meta-sezgisel Yöntem)

Meta-RaPS, DePuy ve diğerleri tarafından yapılan çalışmaya dayanmaktadır. (2001), Arcus (1966) tarafından geliştirilen Montaj Hatları için Bilgisayar Dizileme İşlemlerinin Bilgisayar Yöntemi (COMSOAL) yaklaşımının değiştirilmiş bir versiyonunun uygulanması üzerine yapılan araştırmanın sonucu olarak. Meta-RaPS daha sonra, bunu öncelik kurallarını, rastgeleliği ve örneklemeyi bütünlüştiren rastgele bir ögenin eklenmesine dayanan açgözlü algoritmaları değiştirmek için kullanılan genel, yüksek seviyeli bir strateji olarak tanımlayan Moraga (2002) tarafından resmi olarak tanıtıldı. Meta-RaPS iki aşamadan oluşur: bir yapıçı aşama ve bir iyileştirme aşaması. Yapıçı aşamada, uygulanabilir çözümler, bir durdurma kriteri karşılanıncaya kadar rastgele hale getirilmiş öncelik kuralları aracılığıyla üretilir. İyileştirme aşamasında, yapıçı aşamada elde edilen çözümler belirli bir kriteri geçmeleri halinde iyileştirilebilir. Meta-RaPS, yerel optimada sıkışıp kalmayı önlemek için yapıçı algoritmayı, programlanacak bir sonraki uçak her zaman en iyi öncelik değerine sahip olmak zorunda olmayacağı şekilde değiştirir. Alternatif olarak, bir uçak bazen uygun uçak aday listesinden (CL) rastgele seçilir. Bu nedenle, SA algoritmasında kullanılanlara benzer olan komşuluk arama algoritmaları

kullanılarak yalnızca ümit verici değerlere sahip çözümler geliştirilir. Çözümleri seçmeli olarak iyileştirmenin mantığı, hesaplama süresini çok düşük çözümlere harcamak değil, küresel bir optimuma ulaşma potansiyeline sahip çözümleri geliştirmektir.

2.2.5.1. *Algoritma İçin Çözüm Gösterimi*

Geçerli algoritmada önceki yöntemde de belirtilen 2.2.1.1. numaralı başlıkta açıklanan ve örneklenen permütasyon gösterimi kullanıldı

2.2.5.2. *Algoritma İçin Çözümün Değeri*

Geçerli algoritmada 2.2.1.2. numaralı başlıkta açıklanan ve örneklenen amaç fonksiyonu ile hesaplanan çözüm değeri kullanıldı.

2.2.5.3. *Algoritmada Kullanılan Diğer Yöntemler*

Yapı olarak Meta-Raps açgözlü algoritmaların gelen seçimleri alıp farklılaştırmayı amaçladığından bu algoritmada da en başta tanımlanan üç açgözlü algoritma kullanıldı.

2.2.5.4. *Algoritmada Kullanılan Komşuluk*

Algoritmada komşuluk olarak “Çoklu İndeks Değiştirme” komşuluğu kullanıldı. Permütasyon içinden öncelikle kaç tane indeks değiştirileceğiyile ilgili rastgele bir sayı elde edilir. Ardından çözüm permütasyonunun ortasından ve başından başlayarak belirlenen rastgele sayı kadar uçak indeksi yer değiştirilir. Rastgele seçilen sayı iki olduğunda gerçekleşen değişim şu şekildedir:

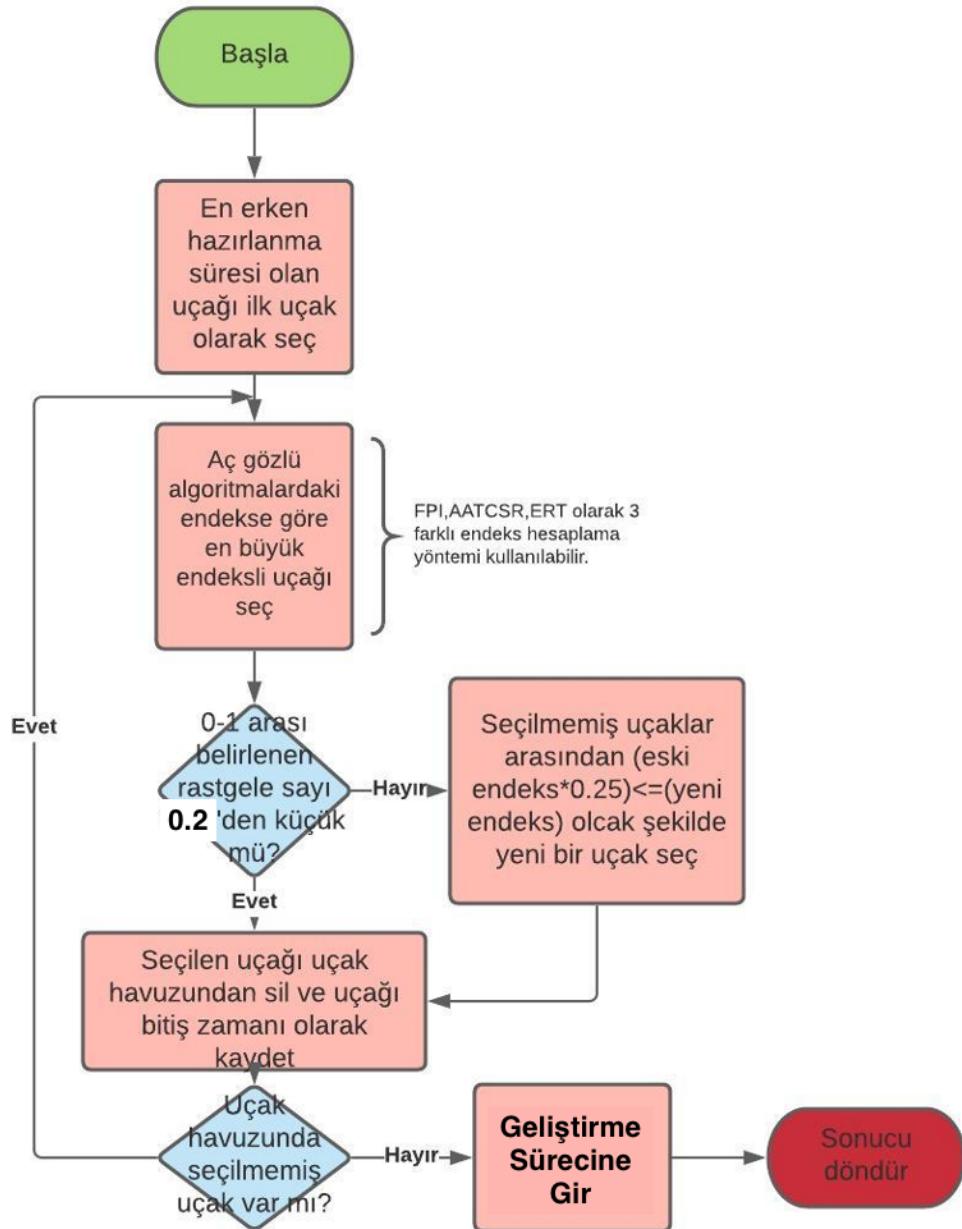


ŞEKİL 2.3. Çoklu İndeks Değiştirme Komşuluğu

2.2.5.5. *Hiper Parametreler Ve İşlevleri*

- İyileştirme döngüsü sayısı: Bu döngü SA algoritmasına benzer şekilde algoritmanın açgözlü kısmından çıkan çözümün komşu çözümleri arasında arama yapar. Arama sonucundaki çözümün kalitesi belli bir değerin üzerindeyse o çözümü kabul eder. Bu döngünün sayısının artması olası kabul edilecek çözüm sayısını artıracagından önemlidir. Bu döngünün değeri raporun sonraki kısımlarında bahsedilecek deneyler sonucu 5000 olarak belirlenmiştir.
- Sapma sınırı miktarı: Algoritmanın açgözlü kısımlarında çözüm 1-sapma sınırı değerinde değişime uğrar. Bu değerim çok büyük olması açgözlü çözümde uzaklaşmamaya sebep olur. Çok küçük olması ise çözümün kötüleşmesine sebep olabilir.

2.2.5.6. Algoritmanın Akış Şeması



ŞEKİL 2.4. Meta-Raps Akış Şeması

2.2.5.7. Algoritmanın Sözde Kodu

```
|  
  
fonksiyon meta_raps_aatcsr_döngü(sıralanacak_uçak_kümesi)  
    Gerekli değişkenleri ve dizileri ilklendir  
    İlk seçilen uçağı en az hazırlanma süresi olan olarak bul  
    for uçak kümesi sayısında dön  
        PI öncelik endeksine göre en büyük olan ve seçilmemiş bir uçak seç  
        0 ile 1 arası rastgele bir P ihtimali belirle  
        if p <= 0.2  
            s=seçilen uçak  
        else  
            seçilmemiş uçaklar arasından (eski PI*0.25 <= yeni PI) olacak şekilde bir  
            L uçağı seç  
  
            s=L uçağı  
            if bitisi  
                seçilen uçak=s  
                seçilmemiş uçak listesinden seçilen uçağı sil  
                seçilen uçağın başlama zamanını diziye kaydet  
                bitme zamanını güncelle  
            for bitisi  
                bitme zamanını döndür  
fonksiyon bitisi  
  
fonksiyon meta_raps_aatcsr_algoritması (sıralanacak_uçak_kümesi)  
    Gerekli değişkenleri ve dizileri ilklendir  
    for i=0'dan i=5000'e kadar  
        çözüm zaman=meta_raps_aatcsr_döngü(sıralanacak_uçak_kümesi)  
        if en kötü zaman<çözüm zaman  
            en kötü zaman=çözüm zaman  
        if bitisi  
            if en iyi zaman>çözüm zaman  
                en iyi zaman=çözüm zaman  
            if bitisi  
                if çözüm zaman<=(en iyi zaman+((en kötü zaman-en iyi zaman)*0.9))  
                    komşu araması ile komşu çözüm bul  
                    yeni çözüm zamanı=komşu araması çözümünün sonucu  
                    if yeni çözüm zamanı <= en iyi çözüm zamanı  
                        en iyi çözüm zamanı=yeni çözüm zamanı  
                    if bitisi  
                if bitisi  
            for bitisi  
                en iyi zamanı döndür
```

2.2.6. GA Algoritması (Genetik Algoritma)

Genetik algoritmalar, doğada gözlemlenen evrimsel süreçce benzer bir şekilde çalışan arama ve optimizasyon yöntemidir. Karmaşık çok boyutlu arama uzayında en iyinin hayatta kalması ilkesine göre bütünsel en iyi çözümü arar. Genetik algoritmaların temel ilkeleri ilk kez Michigan Üniversitesi'nde John Holland tarafından ortaya atılmıştır. Holland 1975 yılında yaptığı çalışmaları “Adaptation in Natural and Artificial Systems” adlı kitabında bir araya getirmiştir. İlk olarak Holland evrim yasalarını genetik algoritmalar içinde optimizasyon problemleri için kullanmıştır. Genetik algoritmalar problemlere tek bir çözüm üretmek yerine farklı çözümlerden oluşan bir çözüm kümesi üretir. Böylelikle, arama uzayında aynı anda birçok nokta değerlendirilmekte ve sonuçta bütünsel çözüme ulaşma olasılığı yükselmektedir. Çözüm kümesindeki çözümler birbirinden tamamen bağımsızdır. Her biri çok boyutlu uzay üzerinde bir vektördür. Genetik algoritmalar problemlerin çözümü için evrimsel süreci bilgisayar ortamında taklit ederler. Diğer eniyileme yöntemlerinde olduğu gibi çözüm için tek bir yapının geliştirilmesi yerine, böyle yapılardan meydana gelen bir küme oluştururlar. Problem için olası pek çok çözümü temsil eden bu küme genetik algoritma terminolojisinde popülasyon adını alır. Bu oluşturulan popülasyonda ise genetik algoritmaya özgü çaprazlama havuzu seçimi, çaprazlama, mutasyon, hayatta kalan seçimi işlemleri tekrarlanarak doğal seçilimle en iyi sonuç bulunmaya çalışılır.

2.2.6.1. *Algoritma İçin Çözüm Gösterimi*

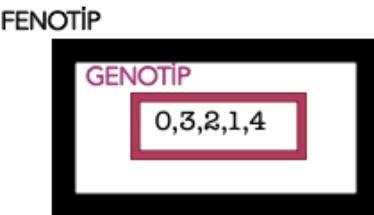
Geçerli algoritmada önceki yöntemde de belirtilen 2.2.1.1. numaralı başlıkta açıklanan ve örneklenen permütasyon gösterimi kullanıldı

2.2.6.2. *Algoritma İçin Çözümün Değeri*

Geçerli algoritmada 2.2.1.2. numaralı başlıkta açıklanan ve örneklenen amaç fonksiyonu ile hesaplanan çözüm değeri kullanıldı.

2.2.6.3. *Fenotip Ve Genotip*

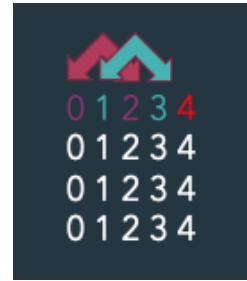
Normal koşullarda genetik algoritmanın temel versiyonunda fenotip uzayı problemin kendisiyle alakalı olurken genotip uzayı ikili düzenden oluşan bit dizini olarak gösterilir. Fakat bahsi geçen yöntem bu problem için uygun olmadığı için fenotip ve genotip uzayı aynı düşünüldü. Ve permütasyon çözüm gösterimi kullanıldı.



ŞEKİL 2.5. Fenotip-Genotip Gösterimi

2.2.6.4. *Başlangıç Popülasyonu Oluşturma Mekanizması*

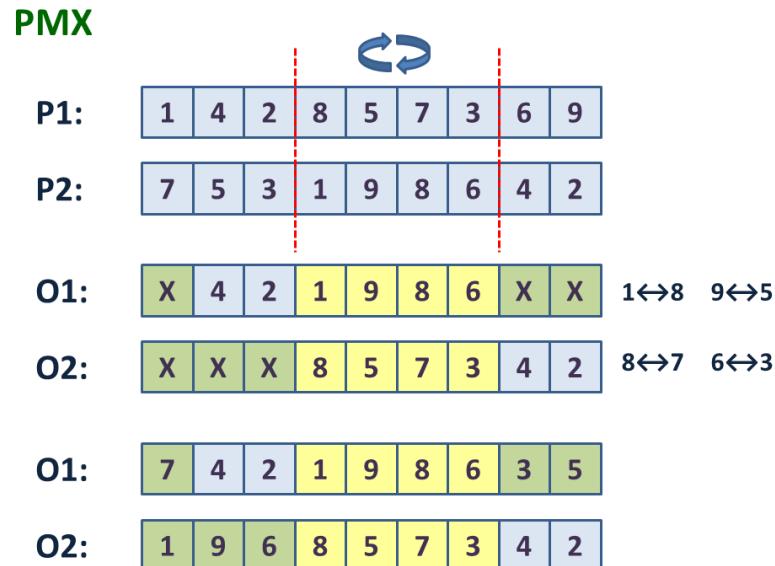
Başlangıç popülasyonu oluşturma amacıyla rastgele popülasyon oluşturma mekanizması kullanıldı. Bu mekanizmada beş uçaklı dört boyutlu bir popülasyon oluşturmak için öncelikle 5×4 'luk iki boyutlu bir dizi tanımlandı ve her hücresi sıfır ile dört arası değerler olacak şekilde dolduruldu. Ardından popülasyondaki her çözüm için tüm endekslerdeki sayılar rastgele başka bir endeks belirlenerek karşılıklı olarak değiştirildi.



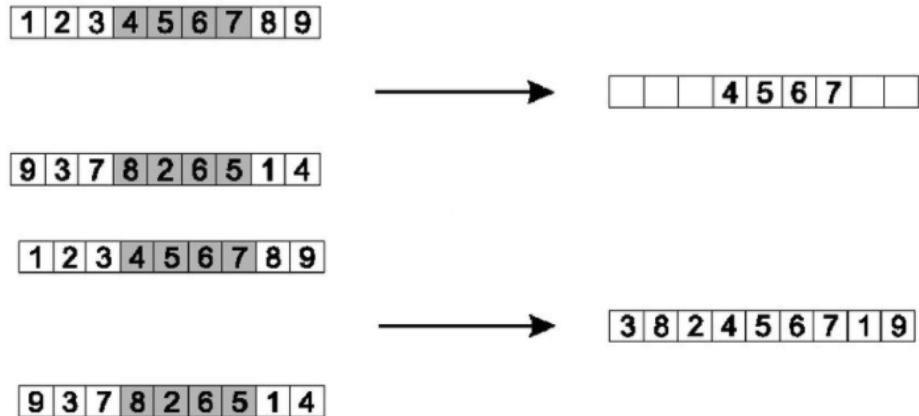
ŞEKİL 2.6. Başlangıç Popülasyonunu Oluşturma

2.2.6.5. Çaprazlama Mekanizması

Çaprazlama mekanizması için permütasyon düzeneindeki problemlere uygun olan iki adet çaprazlama mekanizması tanımlandı. Bu mekanizmalardan biri PMX diğeri ise O1'dı.



ŞEKİL 2.7. PMX Çaprazlama Mekanizmasının Çalışması



ŞEKİL 2.8. O1 Çaprazlama Mekanizmasının Çalışması

2.2.6.6. *Mutasyon Mekanizması*

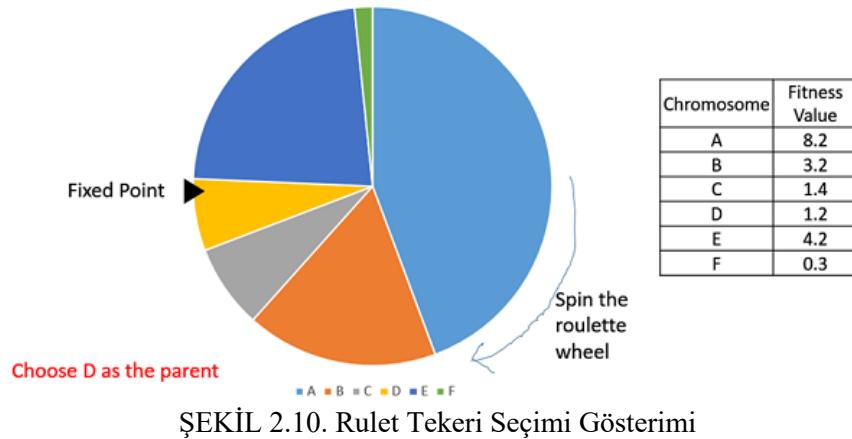
Bu problemde popülasyon içerisindeki her gende 1/kromozom sayısı ihtimalle mutasyon yapıldı. Bu mutasyonda indeks değiştirme komşuluğu uygulandı.



ŞEKİL 2.9. Mutasyon Mekanizmasının Çalışması

2.2.6.7. *Çaprazlama Havuzu Seçimi*

Çaprazlama havuzu seçimi için literatürde de kabul gören bir yöntem olan rulet tekeri seçimi kullanıldı. Burada çözümün uygunluğuna göre seçim olasılığı yüksek olacak şekilde seçimler yapıldı.



ŞEKİL 2.10. Rulet Tekerisi Seçimi Gösterimi

2.2.6.8. *Hayatta Kalanların Seçimi*

Hayatta kalanları seçmek için basit bir karşılaştırma yöntemi kullanıldı. Ebeveyn ve çocuk popülasyondaki her gen eşleşen indeksteki genlerlelaştırıldı. Daha iyi uygunluk değerine sahip olan gen bir sonraki iterasyona geçebildi. Böylece hep iki alternatif arasından iyisi seçildiği için ne fazla odaklanma ne fazla keşfetme yapmamış oldu.

2.2.6.9. *Üzerine Yapılacak Yerel Arama*

Meta-sezgisel yöntemleri geliştirmek amacıyla algoritmadan sonra yerel arama yapılması literatürde sıkça kullanılan bir yöntem, ve bu GA algoritması için ise yerel arama iterasyon sayısı 1000 olarak sabit tutulup bir yerel arama işlemi de uygulandı.

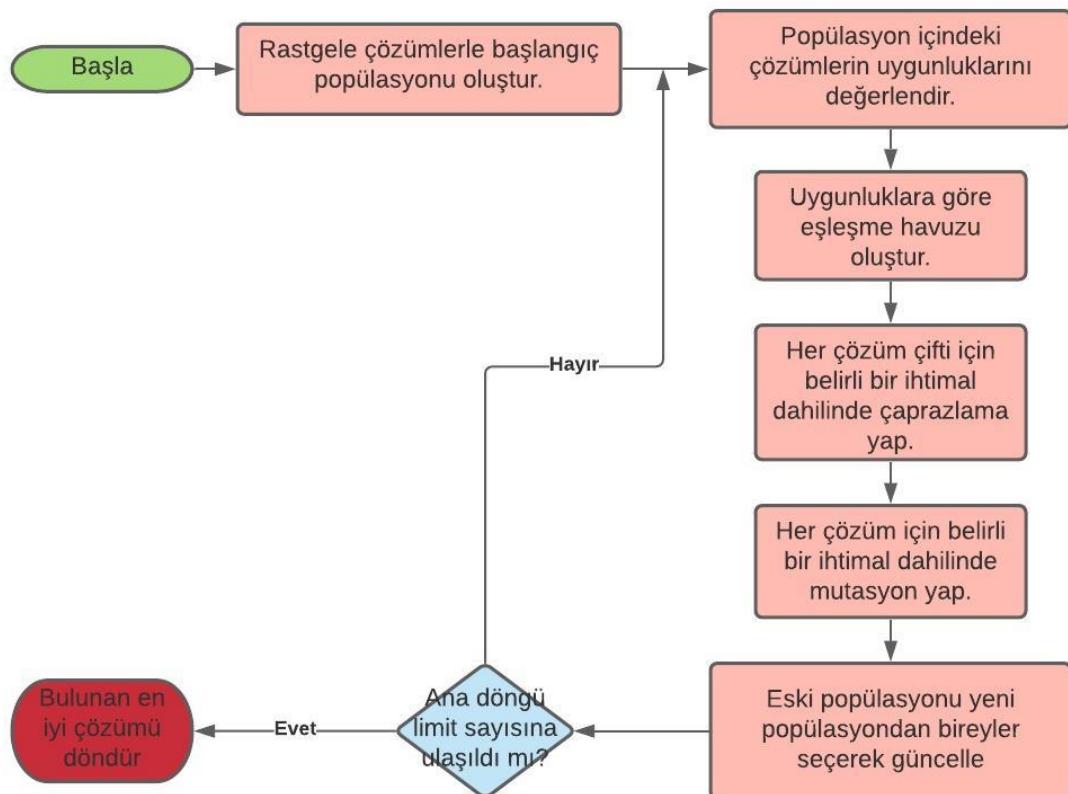
2.2.6.10. *Hiper Parametreler Ve İşlevleri*

- Popülasyon büyüklüğü: Popülasyon büyüklüğü başta oluşturulacak olan çözüm topluluğundaki çözüm sayısını ifade eder. Popülasyon büyüklüğü arttıkça alternatif çözümler ve işlem yapılacak çözümler artacağı için işe yarar hale gelir.

Fakat gereksiz büyüdüğü zaman ise fazla çeşitlilik ve işleme zorluğu yaratacağından çok büyük boyut seçmekte tercih olmaz. Bu raporda detaylıca bahsedilecek deneyler sonucu popülasyon miktarı 60 bulundu.

- Çaprazlama ihtimali: Çaprazlama sonucu iki çözümden yeni iki çözüm oluşturulacağı için iki çözümün değerlerine göre farklı amaçlara yarayabilir. Çaprazlama ihtimalinin çok yüksek olması önceki popülasyondan kalan ebeveynlerin tamamen yok olmasını sağlayacağı için problem olabilir. Tam tersi durumda ise yeni çocuk çözümlerin oluşma ihtimali azalacağı için bu da bir problem olabilir. Bu raporda detaylıca bahsedilecek deneyler sonucu çaprazlama ihtimali 0.8 bulundu.
- İterasyon sayısı: İterasyon sayısı doğal seçimimin yapılacağı tur sayısını ifade eder. Bu tur sayısı ne kadar fazla olursa o kadar iyiye gider. Fakat belli bir değerden sonra yaptığı işlem sayısı iyileşme miktarına göre gereksiz fazla olacağının belli bir değerde tutmak gereklidir. Bu raporda detaylıca bahsedilecek deneyler sonucu iterasyon sayısı 800 bulundu.

2.2.6.11. Algoritmanın Akış Şeması



ŞEKİL 2.11. GA Algoritmasının Akış Şeması

2.2.6.12. Algoritmanın Sözde Kodu

```

fonksiyon genetik_algoritma(sıralanacak uçağ kümesi)
    popülasyon boyutu*kromozom sayısı kadar rastgele düzende popülasyon oluştur
    for l_max sayısında dön
        popülasyondan üretilen bir eşleşme havuzu oluştur
        eşleşme havuzundan çaprazlama yaparak yeni çocuklar oluştur
        çocuk çözümlerde 1/kromozom sayısı ihtimalle mutasyon uygula
        ebeveyn ile çocuk popülasyonu karşılaştırarak hayatı kalanları yeni popülasyon
    for bitisi
        yap
        En iyi çözümü döndür
fonksiyon bitisi
  
```

ŞEKİL 2.12. GA' nın Sözde Kodu

2.3. PROBLEM İÇİN UYGULANAN DENEYLER VE HİPER PARAMETRE OPTİMİZASYONU ARALIKLARI

Bu bölümde daha önceki başlıklarda bahsedilen meta-sezgisel yöntemlerde yöntemin en iyi sonucu verebilmesi için belirli konseptlerce iyi uygulanmasının yanında hiper parametrelerinde kendi içindeki en iyi ayarlarda olması gerekmektedir. İşte bu kısım için hiper parametreler ve algoritmanın çalışma ayarlarıyla ilgili deneyler yapıldı. Bu deneyler daha detaylı şekilde proje raporunun bir sonraki bölümü olan “Bulgular” kısmında incelenecaktır. Deneyler yapılırken mümkün olduğunda harcanan süre sabit tutulup bahsi geçen parametrenin algoritma sonucundaki yaptığı değişikliği analiz etmek hedeflenmiştir. Yapılmış olan deneyler için algoritmalarla aşağıdaki parametre aralıkları ve deney sırası kullanıldı. Buradaki deneyleri tablolaştırmak ve görselleştirmek amacı ile Microsoft Excel Versiyon 16.44 kullanıldı.

2.3.1. GA Hiper Parametreleri İçin Deney Aralıkları

1. Çaprazlama tipi testi: O1-PMX
2. Popülasyon boyutu testi: 10-100
3. Çaprazlama ihtimali testi: 0.2-0.9
4. İterasyon testi: 100-3000

2.3.2. SA Hiper Parametreleri İçin Deney Aralıkları

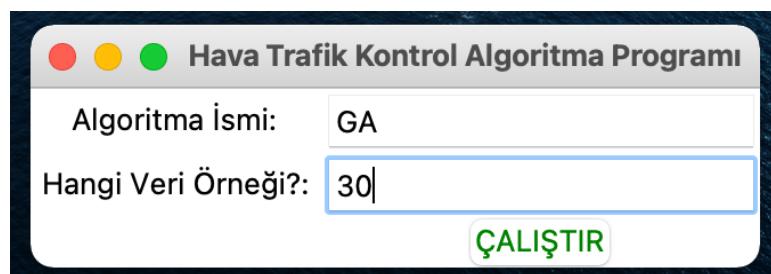
1. Soğutma katsayısı: 0.7-0.995
2. İç döngü sayısı: 20-200
3. En soğuk sıcaklık veya dış döngü iterasyon sayısı: 100-2000
4. Tekrar ısıtma sayısı: 1-10

2.3.3. Meta-Raps Hiper Parametreleri İçin Deney Aralıkları

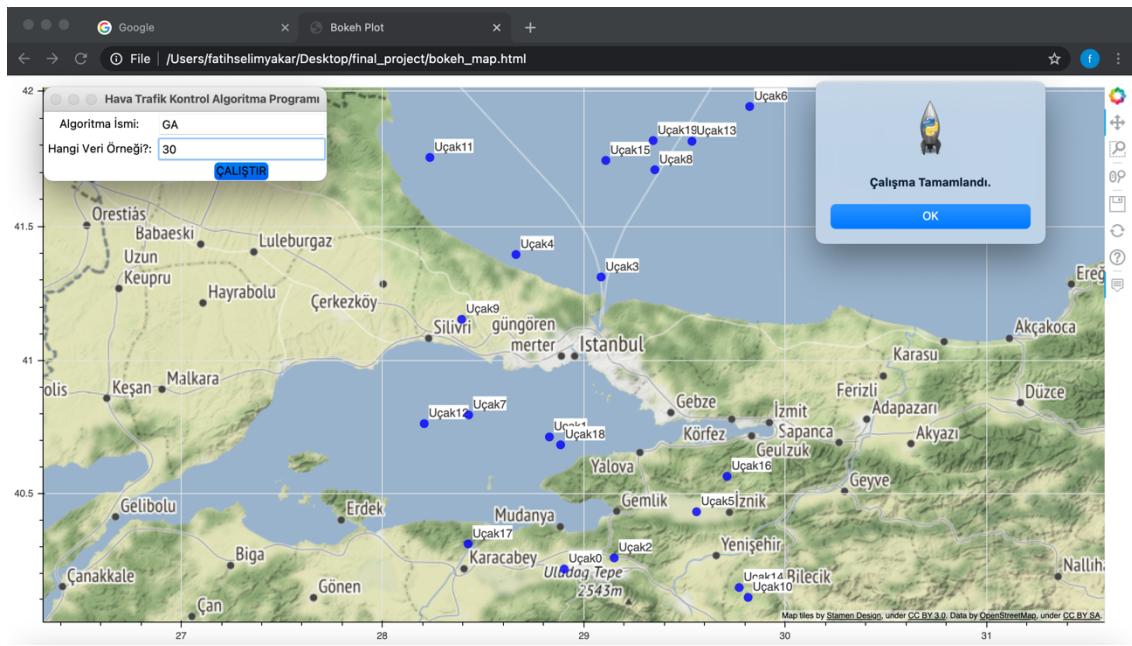
1. Sapma sınırı: 0.1-0.9
2. İterasyon sayısı: 500-5000

2.4. PROBLEM İÇİN ARAYÜZ YAPILMASI

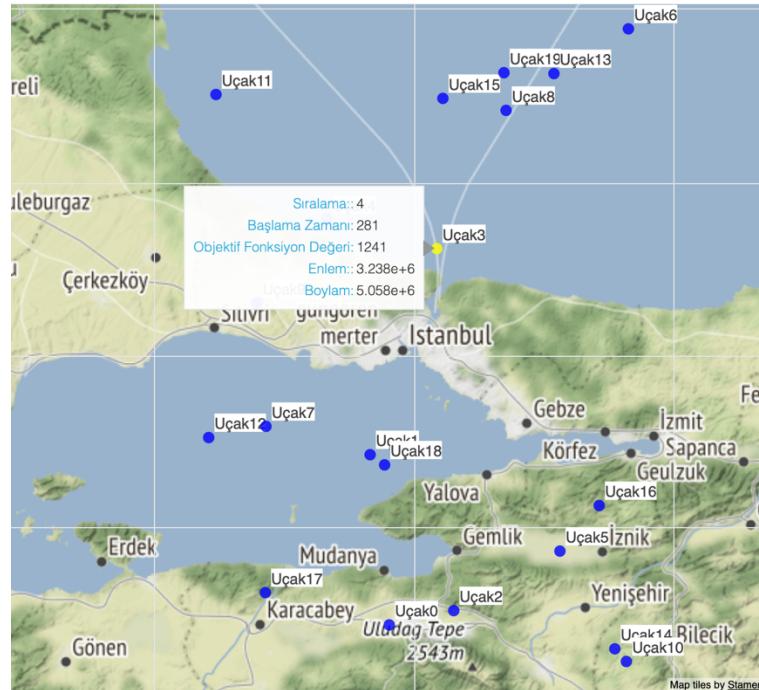
Her ne kadar problemin asıl amacında arayüz yapılması ve görselleştirmesi olmasa da göze hitap etmesi açısından küçük bir arayüz yapıldı. Normal şartlarda bu arayüz gerçek havaalanı bilgilerini girdi olarak alıp algoritmaları çalıştırarak görselleştirecekti. Fakat algoritmanın çalışması için gereken girdiler spesifik (hazır olma zamanları ve ayrılma zamanlarının uçağın büyülüğüne dolaylı olarak marka ve modeline göre değişmesi gibi) girdiler olduğu için herhangi bir API ile elde edilemedi. Bu yüzden bu problemin testlerinde de kullanılan veri kümesi ve algoritmaları içerecek şekilde arayüz tekrardan düzenlenendi. Bu arayüz piton3.7(python3.7) sürümünün beraberinde gelen tkinter eklemesi ile yapıldı. Haritalama ve görselleştirme ise bokeh eklemesi ile yapıldı. Arayüz “Algoritma ismi” kısmına GA, SAaatcsr, SAfpi, SAert, MRAatcsr, MRfpi, MRert, AATCSR, FPI, ERT gibi anahtar kelimeler yazılıp “Hangi Veri Örneği?” kısmına ise 0-59 arası bir sayı yazıldıkten sonra “ÇALIŞTIR” butonuna basarak çalıştırılıyordu. Uçakların enlem ve boyamları gerçek veriler alınamadığı için belirli bir aralıkta rastgele olarak oluşturuldu.



ŞEKİL 2.13. Program İçin tkinter Arayüzü



ŞEKİL 2.14. Programın bokeh Sunucusunun Çalışması



ŞEKİL 2.15. Programın Sıralama Bilgilerini Göstermesi

3. BULGULAR

Bu kısımda 2.3. bölümünde de bahsedilen deney aralıkları ve sıraları dahilinde bir dizi deney yapıldı. Her algoritmanın kendi içindeki hiper-parametreleri adil şekilde test edilip her hiper-parametre için en iyi olduğu değerler kaydedilerek algoritmanın hiper-parametre optimizasyonu yapıldı.

3.1. GA İçin Yapılan Deneyler Ve Bulgular

Bölüm 2.3.'de belirtildiği üzere GA algoritması için belirli aralıklarda deneyler yapıldı. 2.3. kısmında belirtilen aralıklar ise GA için literatürde yapılan araştırmalara ve algoritmanın bu problem için çalışmasındaki davranışa göre incelenerek belirlendi. Bu bağlamda tüm testleri tüm ihtimallerle denemek mevcut süre içerisinde öngörelmez şekilde uzun olacağı için her bir parametre için söz konusu test edilen parametre değişken tutulup diğer parametrelerle de algoritmanın çalışma süresi sabit tutulacak şekilde bir ayar yapılıp sabit süreler eşliğinde test yapılması amaçlandı. Bu testleri yaparken toplam yirmi tane veri örneği kullanıldı. Veri örneklerden aşağı çıkan sonucun rastgelelige bağlı çok iyi veya çok kötü çıkışmasına karşın her veri örneği için her test onar kere çalıştırılıp sonuçların ortalaması alındı. Çözümlerin değerlendirilmesi için ise aşağıdaki formül geliştirildi. Bu formül yirmi tane veri örneği için uygulanıp toplanarak bir toplam hata miktarı elde edildi.

$$\begin{aligned} \text{Ölçekli hata değeri} &= \\ \frac{\text{objektif fonksiyon değeri} - \text{en iyi objektif fonksiyon değeri}}{\text{en iyi objektif fonksiyon değeri}} \end{aligned} \quad (3.1)$$

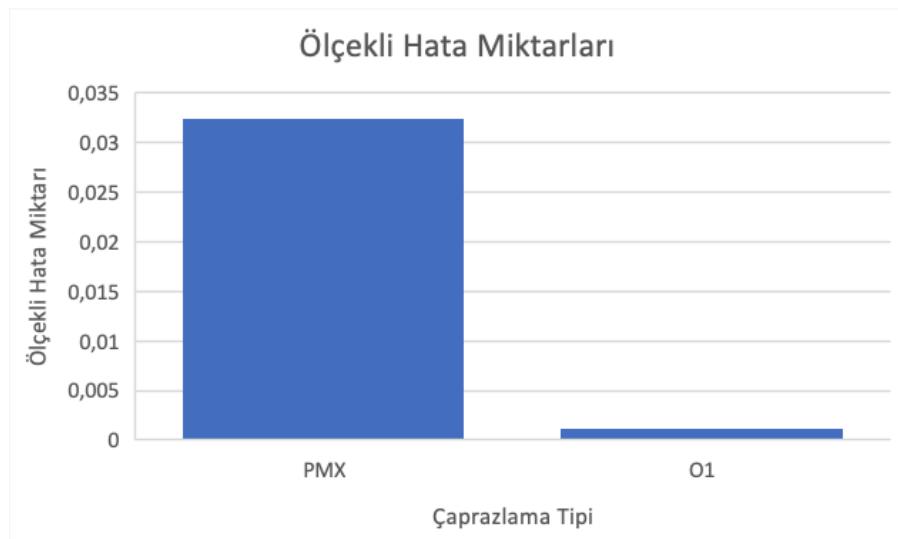
3.1.1. Çaprazlama Tipi Deneyi

Çaprazlama tipi için iki tane alternatif çaprazlama yöntemi belirlendi. Bu yöntemler raporun “Yöntem” kısmında açıklanan PMX ve O1 çaprazlamalarıydı. Çaprazlama tipi, GA için temel konseptlerden biri olduğu için ilk hiper-parametre testi ve belirlenmesi gereken hiper-parametrenin çaprazlama tipi olduğu kararına varıldı. Burada çaprazlama ihtimali 0,5 , iterasyon sayısı 830, popülasyon miktarı 60 olarak ayarlanıp çaprazlama tipi değişken tutularak test yapıldı. Tüm bunlar sonucunda yapılan testte aşağıdaki tablodaki veriler ölçüldü.

TABLO 3.1. Çaprazlama Tipi Deney Sonucu Tablosu

Veri Örneği	PMX	O1	En İyi Değerler
0	995,4	994	994
1	930	930	930
2	908	907	907
3	905,5	904	904
4	1056,3	1052	1052
5	931	931	931
6	898	898	898
7	879,6	879	879
8	898	898	898
9	927	927	927
10	943	943	943
11	943	943	943
12	957	957	957
13	899	899	899
14	911	911	911
15	1223,8	1222,9	1222,9
16	1387,3	1362,9	1362,9
17	1236	1237,5	1236
18	1230,4	1224,4	1224,4
19	1210,5	1210,5	1210,5
En İyi			
Olmayan Sayısı	8	1	1
Çalışma Zamanı	704	978	704
Ölçekli Hata	0,03247964	0,00121359	0,00121359
Döngü Sayısı	830	830	830

Yapılan testlerin sonuçlarında görüldüğü gibi PMX çaprazlama yöntemi 0, 2, 3, 4, 15, 16 ve 18. veri örneklerinde en iyi sonucu bulamamasına karşın O1 çaprazlama yöntemi sadece 17. de en iyi sonucu bulmadı. Aynı şekilde daha önceki bölümde açıklanan ölçekli hata hesabına göre de O1 çaprazlama yöntemi bir hayli daha küçük sonuç verdi. Buradaki O1'in iyiliğinin sebebi genin elemanları arasındaki sırayı PMX' e göre daha fazla korumayı başarması ve böylece aslında ebeveyn olarak kullanılan iki genin özelliklerini çocuk genlere daha iyi aktarması sonucu daha iyi sonuçlar çıkarmış olabilir olduğu düşünüldü. Tüm bunların ışığında çaprazlama tipi olarak O1 seçildi.



ŞEKİL 3.1. Çaprazlama Tipi Deney Sonucu Bar Grafiği

3.1.2. Popülasyon Boyutu Deneyi

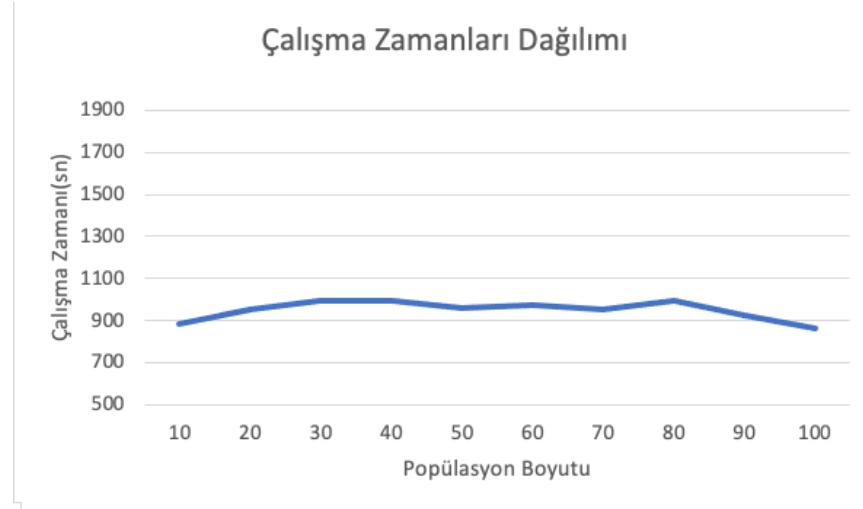
Popülasyon boyutu deneyi için literatürdeki genel GA algoritmalarına bakarak 10-100 deney aralığı belirlendi. Burada popülasyon boyutunun artması çalışma zamanını doğrudan artıracığı için test adilliği açısından bu süre artışını dengelemek amacıyla küçük popülasyon boyutları için iterasyon sayısı yüksek tutulurken büyük popülasyon boyutları için iterasyon sayısı düşük tutuldu. Deney için iterasyon sayısı ve popülasyon

boyutları parametreleri değişkenken çaprazlama ihtimali 0.5 ve çaprazlama tipi ise önceki testten bulunan O1 olarak belirlendi. Tüm bunlar sonucunda yapılan testte aşağıdaki tablodaki veriler ölçüldü.

TABLO 3.2. Popülasyon Boyutu Deney Sonucu Tablosu

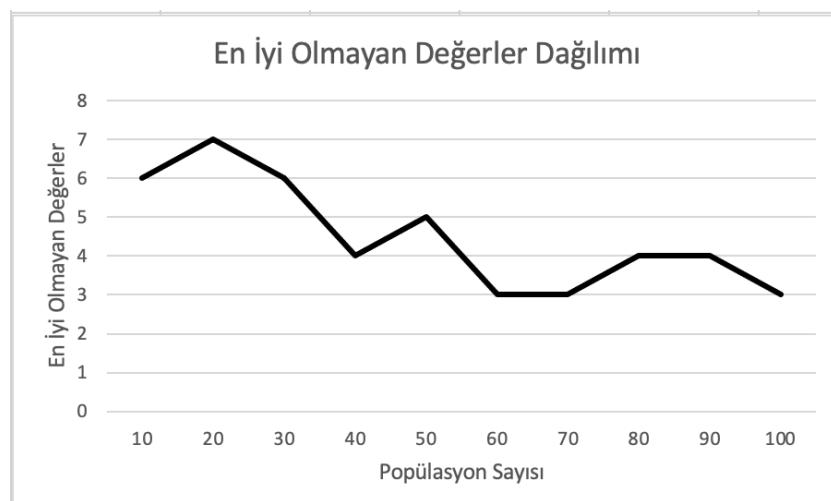
Veri Örneği	10	20	30	40	50	60	70	80	90	100	En İyi Değerle
0	994	994	994	994	994	994	994	994	994	994	994
1	930	930	930	930	930	930	930	930	930	930	930
2	907	907,7	908	907	907	907	907	907	907	907	907
3	904	907	904	904	904	904	904	904	904	904	904
4	1052,1	1052	1052	1052	1052	1052	1052	1052	1052	1052	1052
5	931	931	931	931	931	931	931	931	931	931	931
6	898	898	898	898	898	898	898	898	898	898	898
7	879	879	879	879	879	879	879	879	879	879	879
8	898	898	898	898	898	898	898	898	898	898	898
9	927	927	927	927	927	927	927	927	927	927	927
10	943	943	943	943	943	943	943	943	943	943	943
11	943	943	943	943	943	943	943	943	943	943	943
12	957	957,2	957	957	957	957	957	957,2	957	957	957
13	899	899	899	899	899	899	899	899	899	899	899
14	911	911	911	911	911	911	911	911	911	911	911
15	1224,4	1222,9	1223,8	1224,1	1222,9	1222,9	1222	1223,6	1222	1222	1222
16	1367,1	1367,9	1368,8	1375	1373,7	1362,9	1375,5	1372,8	1368,2	1366,8	1362,9
17	1243,5	1243,1	1240,1	1238	1241,1	1237,5	1236	1236	1237,5	1237,5	1236
18	1242,7	1232,5	1234,9	1228,6	1225,3	1224,4	1230,1	1235,8	1227,7	1226,8	1224,4
19	1209,8	1209	1210,2	1209	1210,5	1210,5	1210,5	1209	1212	1209	1209
En İyi Olmayan											
Sayı	6	7	6	4	5	3	3	4	4	3	3
Çalışma Süresi	883	956	996	993	963	978	951	995	925	862	862
Ölçekli Hata	0,0268165	0,0210643	0,0197899	0,015645	0,0147627	0,0031908	0,015141	0,0180929	0,0102789	0,0060353	0,0031908
Döngü Sayısı	5000	2500	1660	1250	1000	830	714	625	555	500	830

Ayrıca verilerden hareketli çizilen grafikler ise şu şekildeydi:



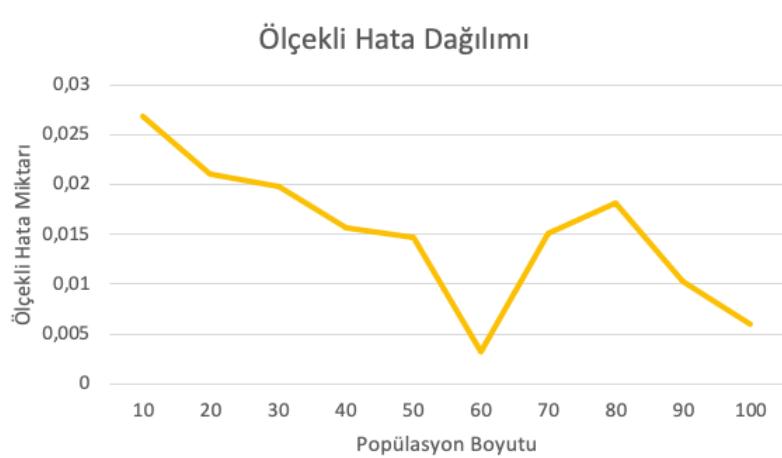
ŞEKİL 3.2. Popülasyon Boyutuna Göre Çalışma Zamanları Dağılımı Grafiği

Belirtilen grafikte elde edilen bilgiye göre başta amaçlanan çalışma süresinin değişmemesi koşulu küçük farklar olsa da karşılandığı gözükmemektedir.



ŞEKİL 3.3. Popülasyon Boyutuna Göre En İyi Olmayan Değerler Dağılımı Grafiği

En iyi olmayan çözüm sayısı görüldüğü gibi düşük sayılı popülasyonlar için daha fazla iken yüksek sayılı popülasyonlar için azaldığı gözüküyor. En iyi değerin ise 60-70 boyutlarında olduğu gözüküyor. Bunun nedeni aslında popülasyon sayısı ve iterasyon sayısı arasındaki dengenin o bölgelerde olmasından kaynaklanıyor.



ŞEKİL 3.4. Popülasyon Boyutuna Göre Ölçekli Hata Dağılımı Grafiği

Ölçekli hata dağılımı grafiğine bakıldığında ise aslında 60-70 civarında tavan yapan en iyi olmayan sayısına benzer olarak ölçekli hatanın da en iyi yani en düşük değerini 60 boyutunda aldığı gözükmüyor. Algoritmanın en iyi sonuç veren popülasyon boyutunun 60 olması ise bize aslında yoğunlaştırma ve çeşitlendirme işlemlerin burada bir dengeye ulaştığını gösteriyor. Yine yoğunlaştırma ve çeşitlendirme bağlamında popülasyon boyutu düşükken süreyi telafi etmek için daha fazla yineleme yapılması geçerli ayarlarda çok fazla çeşitlendirme yapması veya çok fazla yoğunlaştırma yapması sonucu fazla çeşitlendirme veya fazla odaklanma sorunu oluşturmuş olabilir. Popülasyon boyutu yüksekken süreyi telafi etmek için daha az yineleme yapılması ise geçerli ayarlarda çok daha az çeşitlendirme yapması veya çok daha az yoğunlaştırma yapmasına sebep olduğu için popülasyonun yeterince farklılaşmaması sorununa yol açmış olabilir. Tüm bunların sonucunda ise en az ölçekli hata sonucunu veren olarak popülasyon sayısı 60 seçildi.

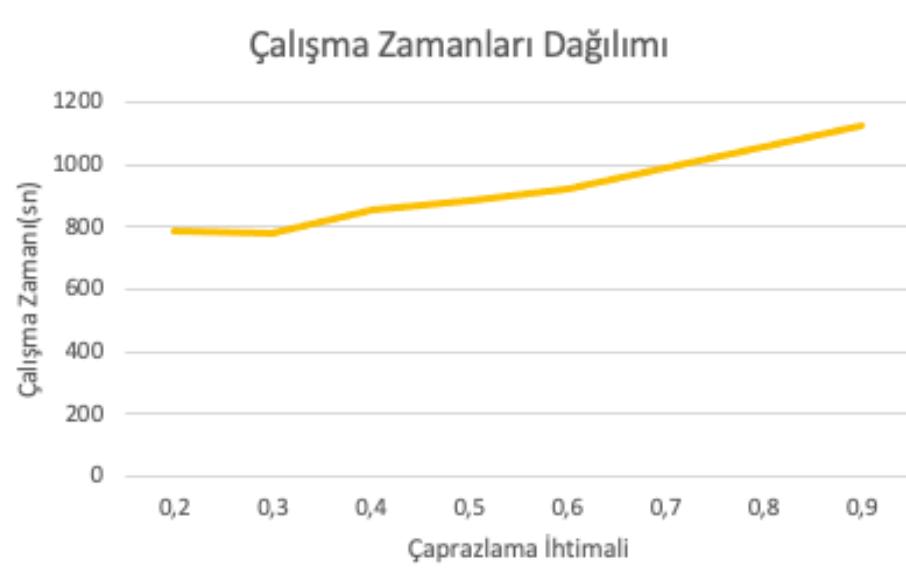
3.1.3. Çaprazlama Olasılığı Deneyi

Çaprazlama olasılığı deneyi için literatürdeki genel GA algoritmalarına bakarak ve verilerin davranışına bakarak 0.2-0.9 arası deney aralığı belirlendi. Burada çaprazlama ihtimali doğrultusunda çaprazlama sayısı artsa da çalışma süreleri arasında çok fazla fark olmadığı için süre telafisi için ek bir değişkenle dengeleme yapılmadı. Deney için çaprazlama ihtimali değeri değişkenken, iterasyon sayısı 800, çaprazlama tipi ve popülasyon sayısı ise önceki testlerde bulunan O1 ve 60 olarak belirlendi. Tüm bunlar sonucunda yapılan testte aşağıdaki tablodaki veriler ölçüldü.

TABLO 3.3. Çaprazlama Olasılığı Deney Sonucu Tablosu

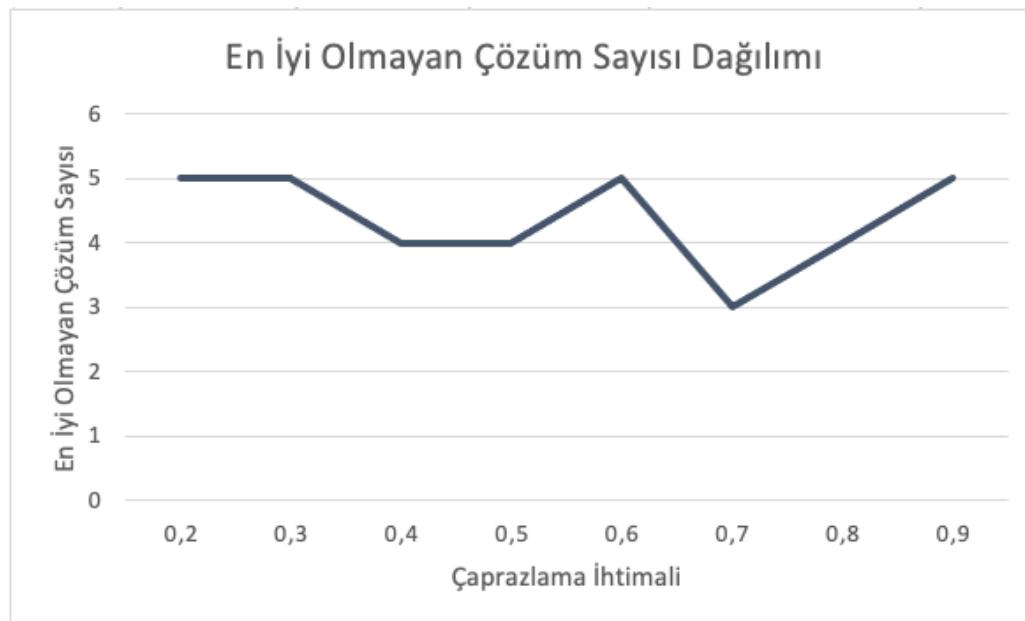
Veri Örneği	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	En İyi Değerler
0	994	994	994	994	994	994	994	994	994
1	930	930	930	930	930	930	930	930	930
2	907	907	907	907	908	907	907	907	907
3	904	904	904	904	904	904	904	904	904
4	1052	1052,1	1052	1052	1052	1052	1052	1052	1052
5	931	931	931	931	931	931	931	931	931
6	898	898	898	898	898	898	898	898	898
7	879	879	879	879	879	879	879	879	879
8	898	898	898	898	898	898	898	898	898
9	927	927	927	927	927	927	927	927	927
10	943	943	943	943	943	943	943	943	943
11	943	943	943	943	943	943	943	943	943
12	957	957	957	957	957	957	957	957	957
13	899	899	899	899	899	899	899	899	899
14	911	911	911	911	911	911	911	911	911
15	1223,8	1222	1224,1	1223,5	1223,8	1222	1222,9	1223,8	1222
16	1363,3	1368,6	1377,7	1366,4	1357,3	1369,4	1363,8	1368,6	1357,3
17	1240	1238	1238	1237,5	1240	1236	1238	1238	1236
18	1231	1234	1223,8	1227,1	1235,8	1231	1225,3	1225,3	1223,8
19	1210,5	1210,7	1210,5	1209	1209,4	1209,8	1210,5	1214,4	1209
En İyi Olmayan									
Sayısı	5	5	4	4	5	3	4	5	3
Çalışma Süresi	784	781	858	883	921	989	1054	1127	781
Ölçekli Hata	0,016254	0,0197793	0,019607	0,0118421	0,0159482	0,0154598	0,00961	0,0171087	0,00960992

Ayrıca verilerden hareketli çizilen grafikler ise şu şekildeydi:



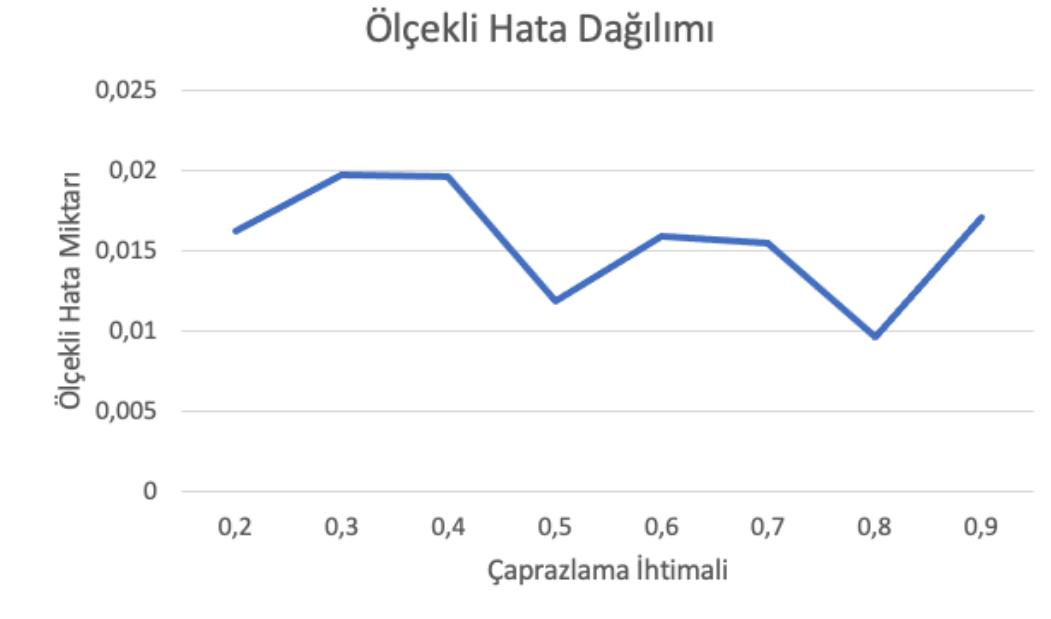
ŞEKİL 3.5. Çaprazlama İhtimaline Göre Çalışma Zamanları Dağılımı Grafiği

Belirtilen grafikte elde edilen bilgiye göre başta amaçlanan çalışma süresinin değişmemesi koşulu küçük farklar olsa da karşılandığı gözükmektedir. Buradaki küçük farklar ise çaprazlama ihtimalinin artması sonucu çaprazlama sayısı artması ve bununda bir işlem yoğunluğu oluşturmalarından dolayıdır.



ŞEKİL 3.6. Çaprazlama İhtimaline Göre En İyi Olmayan Çözüm Sayısı Dağılımı Grafiği

En iyi olmayan çözüm sayıları grafiğine bakıldığında çaprazlama ihtimalinin 0.7 olduğu alanda en iyi olduğu gözükmektedir. En iyi olmayan çözüm sayısı çözüm kalitesi konusunda grafiğin hangi taraflarında iyileştiğini gösterse de net olarak bilgi vermediği için en iyi çözüm 0.7 diyemeyiz. Fakat 0.7 civarlarında iyi çözümler var şeklinde bir yorum yapılabilir. En net kararı vermek için ölçekli hatalara bakmak çok daha net sonuçlar verecektir.



ŞEKİL 3.7. Çaprazlama İhtimaline Göre Ölçekli Hata Dağılımı Grafiği

Ölçekli hata dağılımı grafiğine bakılacak olursa çaprazlama ihtimali azken nispeten daha yüksek ölçekli hata oranı oluştururken yükseldikçe çaprazlama ihtimalinin 0.8 olduğu noktaya kadar ölçekli hata azalmakta ve en son radde olan 0.9 da ise tekrar artmaktadır. Yine bu durum parametreler arasındaki dengenin üç değerlerde değil de ara değerlerde sağlandığının bir göstergesidir. Çaprazlama ihtimali düşük değerlerdeyken ki hata oranının fazla olması popülasyondaki çaprazlama miktarı azaldığı için farklılaşmanın yeterli gelmeyip çeşitlendirme konusunda yetersiz kalmasıyla açıklanabilir. Öte yandan çaprazlama ihtimali yüksek değerdeyken ki hata oranının fazla olması ise tam tersi şekilde çaprazlama miktarının artması sonucu ebeveyn çözümlerde iyi olan çözümlerin çok fazla farklılaşmasıyla açıklanabilir. Tüm bunların sonucunda ise en az ölçekli hata sonucunu veren olarak çaprazlama ihtimali 0.8 seçildi.

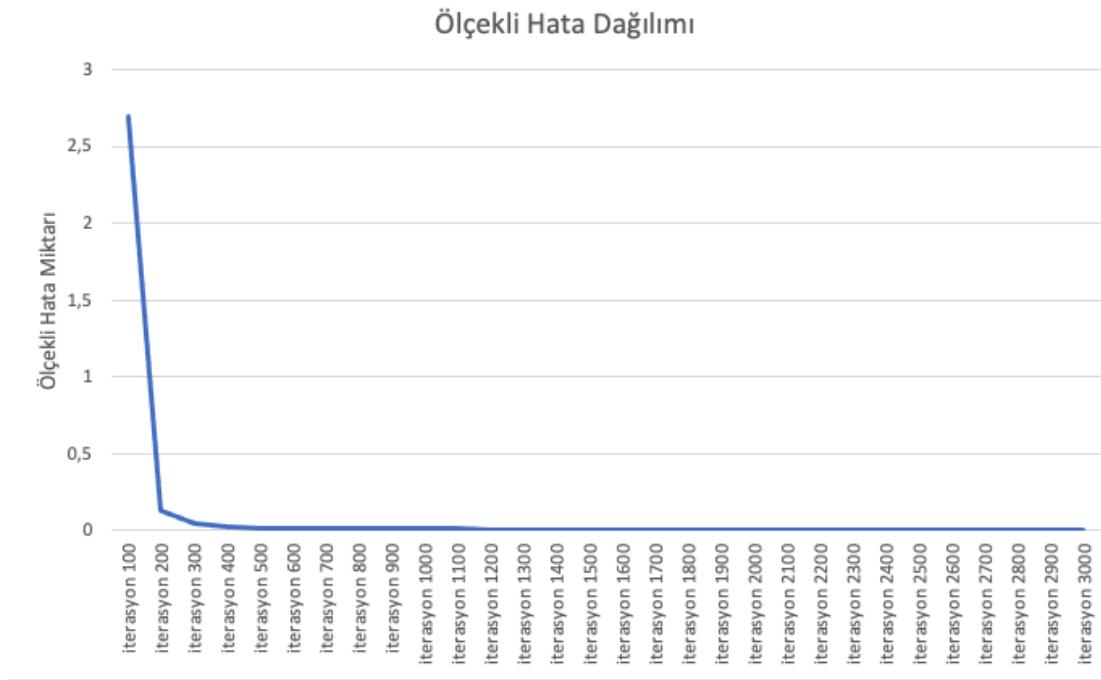
3.1.4. İterasyon Sayısı Deneyi

İterasyon sayısı aralığı için algoritmanın davranışlarına ve problemi çözme süresinin mantıklı sınırlar içerisinde olmasına bakılarak 100-3000 belirlendi. Önceki testlerden elde edilen veriler olan çaprazlama ihtimali 0.8, popülasyon boyutu 60 ve çaprazlama tipi O1 sabit olarak belirlenip iterasyon sayısı 100-3000 iken testler yapıldı. Testler sonucu aşağıdaki tablo ortaya çıktı.

TABLO 3.4. GA İterasyon Sayısı Deneyi Tablosu

Veri Örneği	iterasyon 100	iterasyon 200	iterasyon 300	iterasyon 400	iterasyon 500	iterasyon 600	iterasyon 700	iterasyon 800	iterasyon 900	iterasyon 1000
0	1139	994,1	994	994	994	994	994	994	994	994
1	1030,5	930	930	930	930	930	930	930	930	930
2	965,5	907	907	907	907	907	907	907	907	907
3	1018,7	904	904	904	904	904	904	904	904	904
4	1161,3	1052,1	1052,1	1052,1	1052,1	1052,1	1052	1052	1052	1052
5	1003,1	931	931	931	931	931	931	931	931	931
6	940,1	898	898	898	898	898	898	898	898	898
7	968,8	879	879	879	879	879	879	879	879	879
8	920,3	898	898	898	898	898	898	898	898	898
9	968	927	927	927	927	927	927	927	927	927
10	1065,9	943	943	943	943	943	943	943	943	943
11	1027,1	943	943	943	943	943	943	943	943	943
12	1009,2	957,2	957,2	957,2	957	957	957	957	957	957
13	945,1	899	899	899	899	899	899	899	899	899
14	993,9	911	911	911	911	911	911	911	911	911
15	1614,1	1247,9	1225,6	1223,5	1223,5	1223,5	1222,9	1222,9	1222,9	1222,9
16	1772,7	1402,4	1380,4	1373,9	1371	1370,9	1370,9	1368	1368	1367
17	1550,6	1260,7	1244,3	1241,1	1239,5	1239	1238,4	1237,5	1237,5	1237,5
18	1537,4	1256,8	1238,8	1231	1230,1	1230,1	1230,1	1230,1	1230,1	1230,1
19	1570,7	1254,1	1222,6	1213,3	1211,8	1210,9	1210,9	1210,5	1210,5	1210,5
Ölçekli Hata	2,690551943	0,133077237	0,044587885	0,021457288	0,015845439	0,014624191	0,013553648	0,010358236	0,010358236	0,009621099

Tabloda belirtilen objektif fonksiyon değerleri eğer en iyi objektif fonksiyonunu içeriyorsa kırmızı olacak şekilde işaretlendi. Bu bağlamda 800. İterasyondan sonra gelişmenin çok yavaşladığı görüldü. Bu bir GA olduğundan ötürü iterasyon sayısı ne kadar fazla olursa o kadar iyi çalışacaktır. Fakat belli bir değerden sonra iterasyon sayısının getirdiği yükün altında gelişme miktarı çok azalacaktır. Bu bağlamda bahsedilen sınır değer, tablodan oluşturulan aşağıdaki grafikten de hareketle 800 olarak belirlendi.



ŞEKİL 3.8. İterasyona Göre Ölçekli Hata Dağılımı Grafiği

3.2. SA İçin Yapılan Deneyler Ve Bulgular

Bölüm 2.3.' de belirtildiği üzere SA algoritması için belirli aralıklarda deneyler yapıldı. 2.3. kısmında belirtilen aralıklar ise SA için literatürde yapılan araştırmalara ve algoritmanın bu problem için çalışmasındaki davranışa göre incelenerek belirlendi. Bu bağlamda tüm testleri tüm ihtimallerle denemek mevcut süre içerisinde öngörelmez şekilde uzun olacağı için her bir parametre için söz konusu test edilen parametre değişken tutulup diğer parametrelerle de algoritmanın çalışma süresi sabit tutulacak şekilde bir ayar yapılip sabit süreler eşliğinde test yapılması amaçlandı. Bu testleri yaparken toplam yirmi tane veri örneği kullanıldı. Veri örneklerden aşağı çıkan sonucun rastgelelige bağlı çok iyi veya çok kötü çıkışmasına karşın her veri örneği için her test onar kere çalıştırılıp sonuçların ortalaması alındı. Çözümlerin değerlendirilmesi için ise aşağıdaki formül

geliştirildi. Bu formül yirmi tane veri örneği için uygulanıp toplanarak bir toplam hata miktarı elde edildi.

$$\begin{aligned} & \text{Ölçekli hata değeri} = \\ & \frac{(\text{objektif fonksiyon değeri} - \text{en iyi objektif fonksiyon değeri})}{\text{en iyi objektif fonksiyon değeri}} \end{aligned} \quad (3.2)$$

3.2.1. Soğuma Katsayısı Deneyi

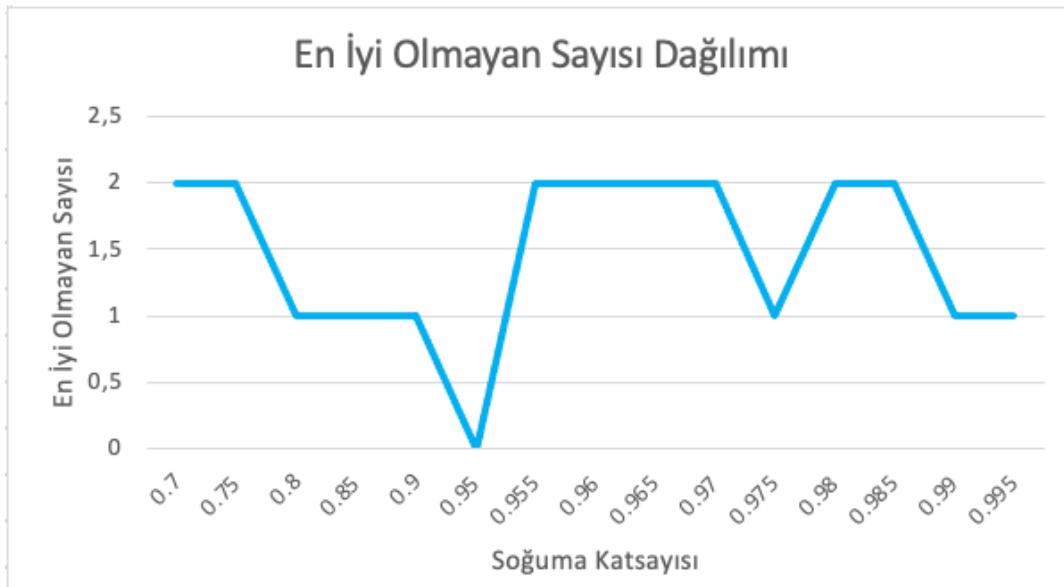
Soğuma katsayısı deneyi için literatürdeki genel SA algoritmalarına ve algoritmanın kendi davranışına bakarak 0.7-0.995 deney aralığı belirlendi. Burada soğuma katsayısının artması belli bir en soğuk sıcaklık belirlendiğinde bu en soğuk sıcaklığa gelene kadarki döngü sayısını ciddi ölçüde artıracak için test adilliği açısından bahsi geçen süre artışını dengelemek amacıyla küçük soğuma katsayısı değerleri için iterasyon sayısı küçük olduğundan ötürü tekrardan ısıtma sayısı artırıldı. Küçük soğuma katsayısında daha az iterasyon olmasının sebebi ise belli bir en soğuk sıcaklığa kadar her döngü değişiminde daha fazla düşme yaşamasından kaynaklanıyordu. Büyyük soğuma katsayısı değerleri için tekrardan ısıtma sayısı düşük tutuldu. Deney için tekrar ısıtma sayısı ve soğuma katsayısı parametreleri değişkenken, iç döngü sayısı 100, en soğuk sıcaklık 0.9 ve tekrar ısıtma katsayısı 0.8 olarak belirlendi. Tüm bunlar sonucunda yapılan testte aşağıdaki tablodaki veriler ölçüldü.

TABLO 3.5. SA Soğuma Katsayısı Deneyi Tablosu

Veri Örneği	0.7	0.75	0.8	0.85	0.9	0.95	0.955	0.96
0	994	994	994	994	994	994	994	994
1	930	930	930	930	930	930	930	930
2	907	907	907	907	907	907	907	907
3	904	904	904	904	904	904	904	904
4	1052	1052	1052	1052	1052	1052	1052	1052
5	931	931	931	931	931	931	931	931
6	898	898	898	898	898	898	898	898
7	879	879	879	879	879	879	879	879
8	898	898	898	898	898	898	898	898
9	927	927	927	927	927	927	927	927
10	943	943	943	943	943	943	943	943
11	943	943	943	943	943	943	943	943
12	957	957	957	957	957	957	957	957
13	899	899	899	899	899	899	899	899
14	911	911	911	911	911	911	911	911
15	1222	1222	1222	1222	1222	1222	1222	1222
16	1360,5	1358,9	1356,7	1355,9	1356,6	1355	1358	1361,1
17	1236	1236	1236	1236	1236	1236	1236	1236
18	1222,9	1222,9	1222	1222	1222	1222	1224,4	1222,9
19	1209	1209	1209	1209	1209	1209	1209	1209
Yineleme #	27	43	55	82	121	257	291	329
Tekrar İşitme	350	250	200	150	80	9	8	6
En İyi Olmayan	2	2	1	1	1	0	2	2
Çalışma zamanı	398,1	382,4	403,9	441,6	408,9	413	401	441
Ölçekli Hata	0,005532	0,0043512	0,0012546	0,0006642	0,0011808	0	0,004178	0,0052383

0.965	0.97	0.975	0.98	0.985	0.99	0.995	En İyi Değer
994	994	994	994	994	994	994	994
930	930	930	930	930	930	930	930
907	907	907	907	907	907	907	907
904	904	904	904	904	904	904	904
1052	1052	1052	1052	1052	1052	1052	1052
931	931	931	931	931	931	931	931
898	898	898	898	898	898	898	898
879	879	879	879	879	879	879	879
898	898	898	898	898	898	898	898
927	927	927	927	927	927	927	927
943	943	943	943	943	943	943	943
943	943	943	943	943	943	943	943
957	957	957	957	957	957	957	957
899	899	899	899	899	899	899	899
911	911	911	911	911	911	911	911
1222	1222	1222	1222	1222	1222	1222	1222
1357,4	1362,7	1358,6	1361,8	1361,8	1363,1	1360,4	1355
1236	1236	1236	1236	1236	1236	1236	1236
1222,9	1222,9	1222	1223,8	1223,8	1222	1222	1222
1209	1209	1209	1209	1209	1209	1209	1209
370	440	521	663	887	1333	2673	257
5	5	4	3	2	1	1	?
2	2	1	2	2	1	1	1
402,6	436,11	470,5	452,6	476	478	484	1119
0,0025077	0,0064192	0,00265683	0,0064914	0,0064914	0,0059779	0,0039852	21

Bahsi geçen tablodan hareketle aşağıdaki grafikler çizildi.



ŞEKİL 3.9. Soğuma Katsayısına Göre En İyi Olmayan Sayısı Dağılımı Grafiği



ŞEKİL 3.10. Soğuma Katsayısına Göre Ölçekli Hata Dağılımı Grafiği

Ölçekli hata dağılımı ve en iyi olmayan sayısı dağılımı grafiklerine baktığımızda ikisinde de algoritmanın kabaca aynı davranışını gösterdiğini açıkça görebiliyoruz. Ayrıca soğuma katsayısının düşük ve yüksek olduğu kısımlarda ölçekli hata ve en iyi olmayan sayısı bazında kötüleşmeler olduğunu görüyoruz. Düşük soğuma katsayısında bu şekilde kötü sonuçlar olmasının nedeni bu katsayının çok hızlı sıcaklık düşürmesi ve yapılan iterasyon sayısının yetmemesi sonucu çeşitlendirmedeki odaklanmaları kısıtlanması olabilir. Öte yapılan tekrar ısıtma işleminin değerlerinin bu soğuma katsayısına uymaması durumu da söz konusu olabilir. Aynı şekilde yüksek soğuma katsayılarındaki kötüleşmelerin nedeni ise aynı sıcaklık aralığındaki iterasyon sayısının fazlaca artması sonucu iç döngünün yeterince arama yapamaması olabilir. Yani asılina bakılacak olursa bu iki durumda önceki algoritmalarla da bahsettiğimiz çeşitlendirme ve yoğunlaştırma işlemlerinin dengesizliğinden kaynaklanabilir. Tüm bunların sonucunda ise en az ölçekli hata sonucunu veren olarak soğuma katsayısı 0.95 seçildi.

3.2.2. İç Döngü Sayısı Deneyi

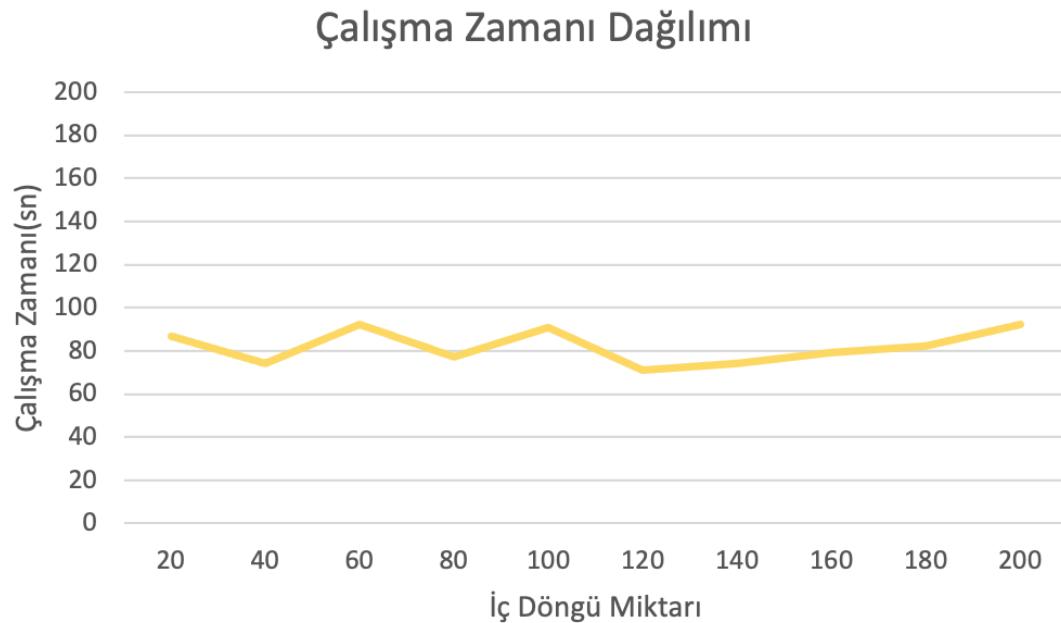
İç döngü sayısı deneyi için literatürdeki genel SA algoritmalarına ve algoritmanın kendi davranışına bakarak 20-200 deney aralığı belirlendi. Burada iç döngü sayısının artması algoritmanın çalışma süresini artttıracağı için test adilliği açısından bahsi geçen süre artışını dengelemek amacıyla küçük iç döngü sayısı değerleri için tekrardan ısıtma sayısı arttırıldı. Büyuk iç döngü sayısı için tekrardan ısıtma sayısı düşük tutuldu. Deney için tekrar ısıtma sayısı ve iç döngü sayısı parametreleri değişkenken, en soğuk sıcaklık 0.9, tekrar ısıtma katsayısı 0.8 ve soğuma katsayısı ise yapılan testlerden elde edilen 0.95 olarak belirlendi. Tüm bunlar sonucunda yapılan testte aşağıdaki tablodaki veriler ölçüldü.

TABLO 3.6. SA İç Döngü Sayısı Deneyi Tablosu

Veri Örneği	20	40	60	80	100	
0	994	994	994	994,2	994	994
1	930	930	930	930	930	930
2	907	907	907	907	907	907
3	904	904	904	904	904	904
4	1052	1052,2	1052	1052,1	1052,1	
5	931	931	931	931	931	931
6	898	898	898	898	898	898
7	879	879	879	879	879	879
8	898	898	898	898	898	898
9	927	927	927	927	927	927
10	943	943	943	943	943	943
11	943	943	943	943	943	943
12	957	957	957	957	957	957
13	899	899	899	899	899	899
14	911	911	911	911	911	911
15	1222	1222	1222,6	1222	1222	
16	1372,2	1372,6	1375,8	1375,1	1374,2	
17	1236	1237,5	1237,5	1237,8	1238,4	
18	1237,2	1236,6	1229,2	1230,7	1232,5	
19	1214,2	1213,5	1210,7	1210,5	1209	
Tekrar Isıtma #	9	4	3	2	2	
En İyi Olmayan	3	5	4	6	4	
Çalışma Süresi	87	74	92	77	91	
Ölçekli Hata	0,0146868	0,015316	0,0096218	0,0102126	0,010062	

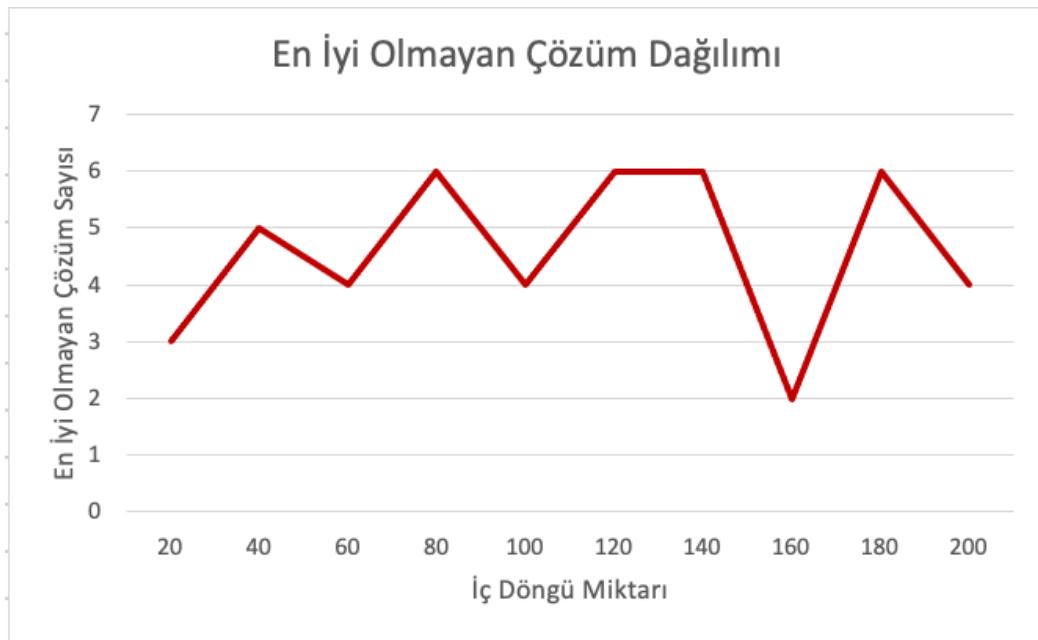
120	140	160	180	200	En İyi Değerler	
994,1	994	994	994,1	994	994	994
930	930	930	930	930	930	930
907	907	907	907	907	907	907
904	904,9	904	904	904	904	904
1052,2	1052,1	1052	1052,1	1052,1	1052	
931	931	931	931	931	931	931
898	898	898	898	898	898	898
879,6	879	879	879	879	879	879
898	898	898	898	898	898	898
927	927	927	927	927	927	927
943	943	943	943	943	943	943
943	943	943	943	943	943	943
957	957	957	957	957	957	957
899	899	899	899	899	899	899
911	911	911	911	911	911	911
1222	1222	1222	1223,3	1222	1222	
1385,4	1376,8	1366,9	1381,7	1373,6	1366,9	
1236	1237,5	1236	1236	1236	1236	
1230,5	1233,1	1235,5	1236,4	1233,9	1229,2	
1212	1210,5	1210,7	1210,5	1210,9	1209	
1	1	1	1	1	1 ?	
6	6	2	6	4	2	
71	74	79	82	92	71	
0,0180466	0,0139604	0,0065314	0,0191851	0,0103918	0,00653141	

Bahsi geçen tablodan hareketle aşağıdaki grafikler çizildi.



ŞEKİL 3.11. İç Dönüş Miktarına Göre Çalışma Zamanı Dağılımı Grafiği

Çalışma zamanı dağılımı grafiğinde tekrar ısratma sayısı ile amaçlanan süre dengelemesi sonucu adil bir test yapıldığı görülmektedir.



ŞEKİL 3.12. İç Döngü Miktarına Göre En İyi Olmayan Çözüm Dağılımı Grafiği



ŞEKİL 3.13. İç Döngü Miktarına Göre Ölçekli Hata Dağılımı Grafiği

Ölçekli hata dağılımı ve en iyi olmayan çözüm dağılımı grafiklerinden de kabaca anlaşılacağı üzere en iyi iç döngü miktarı 160 değerinde çıkmıştır. İç döngü miktarı belirli bir sıcaklıktaki komşuluk aramasının yapılmama miktarını belirtir. Az olması durumunda yeterli arama yapılmadığı yeteri kadar çözüm bulamaz. Çok fazla olması durumunda da diğer birçok parametre işin içine girecektir. Sıcaklığın yüksek olduğu durumda çok fazla çeşitlendirme ve sıcaklığın düşük olduğu durumda ise çok fazla yoğunlaştırma yapacaktır. Tüm bunların sonucunda SA için iç döngü miktarı 160 seçilmiştir.

3.2.3. İterasyon Sayısı Veya En Soğuk Sıcaklık Değeri Deneyi

SA algoritmasında iterasyonu arttırmanın bir sınırı vardır. Çünkü bir yerden sonra iterasyona bağlı olarak çok fazla küçülen sıcaklık sonucu SA’ın en önemli işlevi olan ve çeşitlendirmeye yarayan kötü sonuç kabul etme mekanizması yok olarak yerini basit bir yerel aramaya bırakacaktır. Bu şekilde olunca da en son bulduğu yerel en iyiye takılı kalacaktır. Bu testteki amaç ise gelişmeyi bıraktığı iterasyonu bulup bu iterasyonun sıcaklık değerini en soğuk sıcaklık olarak belirlemektir. Bu bağlamda tekrar ısıtma sayısı 1, tekrar ısıtma katsayısı 0.8 ve önceki testlerden seçilen iç döngü miktarı 160, soğuma katsayısı 0.95 olarak belirlendi. Ve iterasyon sayısı ise 100-2000 arası değişken olarak belirlendi. Bu test sonucu aşağıdaki tablo ortaya çıktı.

TABLO 3.7. SA İterasyon Sayısı Deneyi Tablosu

Veri Örneği	İterasyon 100	İterasyon 200	İterasyon 300	İterasyon 400	İterasyon 500
0	1090,8	994,1	994	994	994
1	1023,8	931,7	930	930	930
2	967	907	907	907	907
3	1020,4	905,9	904	904	904
4	1158,9	1053,2	1053	1053	1053
5	1018,3	932,2	931	931	931
6	920,4	898	898	898	898
7	963,1	879,6	879	879	879
8	919	898	898	898	898
9	965,8	927	927	927	927
10	1049,2	943	943	943	943
11	1039,9	943	943	943	943
12	1035,2	957	957	957	957
13	957,9	899	899	899	899
14	974,4	911	911	911	911
15	1479,3	1227,2	1222	1222	1222
16	1654,7	1390	1376,9	1376,9	1376,9
17	1521,3	1250	1237,5	1237,5	1237,5
18	1425,5	1242,3	1224,4	1224,4	1224,4
19	1483,6	1232,9	1209,9	1209,9	1209,9
Ölçekli Hata	2,236624828	0,062742544	0	0	0

Tablodan aşağıdaki grafik elde edildi.



ŞEKİL 3.14. İterasyon Sayısına Göre Ölçekli Hata Dağılımı Grafiği

Ölçekli hata dağılımı grafiğinde ve tabloda da görülebileceği üzere başta bahsedilen gelişmemeye sınırı 300. İterasyon olarak belirlendi. Bu iterasyona karşılık gelen sıcaklık ise 0.002 olduğu için en soğuk sıcaklık 0.002 olarak belirlendi.

3.2.4. Tekrar Isıtma Sayısı Deneyi

Tekrar ısıtma sayısı aslında basitçe SA algoritmasının kaç kere çalışacağını belirler. Fakat tekrar çalışma sırasında en başta belirlenen başlangıç sıcaklığını belli bir miktar azaltır. Böylece daha iyi sonuçlar bulmayı hedefler. Yine SA'ın normal iterasyon sayısında olduğu gibi belli bir değerden sonra sıcaklığı çok fazla düşürecek için işe yaramaz hale gelir. Benzer şekilde burda da gelişmenin durduğu veya gelişmenin çok yavaşladığı iterasyonu bulmak amaçlandı. Bu bağlamda iterasyon sayısı değişken 1-10 ve en soğuk sıcaklık 0.002, soğuma katsayısı 0.95 ve iç döngü sayısı 160 sabit olarak ayarlandı. Yapılan testler sonucu aşağıdaki tablo elde edildi.

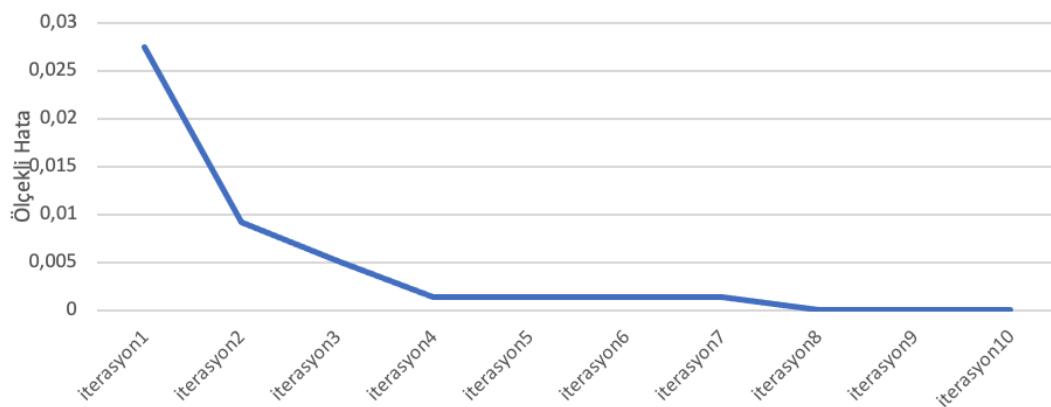
TABLO 3.8. SA Tekrar Isıtma Sayısı Deneyi Tablosu

Veri Örneği	İterasyon1	İterasyon2	İterasyon3	İterasyon4	İterasyon5
0	994	994	994	994	994
1	930	930	930	930	930
2	907	907	907	907	907
3	904	904	904	904	904
4	1052,2	1052,1	1052	1052	1052
5	931	931	931	931	931
6	898	898	898	898	898
7	879,6	879	879	879	879
8	898	898	898	898	898
9	927	927	927	927	927
10	943	943	943	943	943
11	943	943	943	943	943
12	957	957	957	957	957
13	899	899	899	899	899
14	911	911	911	911	911
15	1222,7	1222	1222	1222	1222
16	1384,2	1368,5	1364,2	1359	1359
17	1236	1236	1236	1236	1236
18	1228,6	1222,9	1222	1222	1222
19	1209,9	1209	1209	1209	1209
Ölçekli Hata	0,02748484	0,00915752	0,00515768	0,00132626	0,00132626

iterasyon6	iterasyon7	iterasyon8	iterasyon9	iterasyon10	En İyi Değer
994	994	994	994	994	994
930	930	930	930	930	930
907	907	907	907	907	907
904	904	904	904	904	904
1052	1052	1052	1052	1052	1052
931	931	931	931	931	931
898	898	898	898	898	898
879	879	879	879	879	879
898	898	898	898	898	898
927	927	927	927	927	927
943	943	943	943	943	943
943	943	943	943	943	943
957	957	957	957	957	957
899	899	899	899	899	899
911	911	911	911	911	911
1222	1222	1222	1222	1222	1222
1359	1359	1357,3	1357,2	1357,2	1357,2
1236	1236	1236	1236	1236	1236
1222	1222	1222	1222	1222	1222
1209	1209	1209	1209	1209	1209
0,00132626	0,00132626	7,3681E-05	0	0	0

Ayrıca tablodan elde edilen ölçekli hata grafiği aşağıdaki gibidir.

Tekrar Isıtma Sayısına Göre Hata Dağılımı



ŞEKİL 3.15. Tekrar Isıtma Sayısına Göre Ölçekli Hata Dağılımı Grafiği

Hata dağılımı ve tabloya bakıldığından iterasyon 9' dan sonra artık gelişmenin sağlanmadığı gözükmektedir. Fakat iterasyon 4' den sonra ise gelişmenin çok azaldığı gözükmektedir. Tüm bunlar neticesinde iterasyonların 4'den 9'a çıkması ile sürenin iki katına çıkacağı da göz önüne alındığı için burada 4, tekrar ısıtma sayısı olarak belirlendi.

3.3. Meta-Raps İçin Yapılan Deneyler Ve Bulgular

Bölüm 2.3.' de belirtildiği üzere Meta-Raps algoritması için belirli aralıklarda deneyler yapıldı. 2.3. kısmında belirtilen aralıklar ise Meta-Raps için bu problem üzerinde çalışmasındaki davranışa göre incelenerek belirlendi. Bu bağlamda tüm testleri tüm ihtimallerle denemek mevcut süre içerisinde öngörelmez şekilde uzun olacağı için her bir parametre için söz konusu test edilen parametre değişken tutulup diğer parametrelerle de algoritmanın çalışma süresi sabit tutulacak şekilde bir ayar yapılip sabit süreler eşliğinde test yapılması amaçlandı. Bu testleri yaparken toplam yirmi tane veri örneği kullanıldı. Veri örneklerden aşağı çıkan sonucun rastgelelige bağlı çok iyi veya çok kötü çıkışına karşın her veri örneği için her test onar kere çalıştırılıp sonuçların ortalaması alındı. Çözümlerin değerlendirilmesi için ise aşağıdaki formül geliştirildi. Bu formül yirmi tane veri örneği için uygulanıp toplanarak bir toplam hata miktarı elde edildi.

$$\begin{aligned} \text{Ölçekli hata değeri} = \\ \frac{(\text{objektif fonksiyon değeri} - \text{en iyi objektif fonksiyon değeri})}{\text{en iyi objektif fonksiyon değeri}} \end{aligned} \quad (3.3)$$

3.3.1. Sapma Sınırı Deneyi

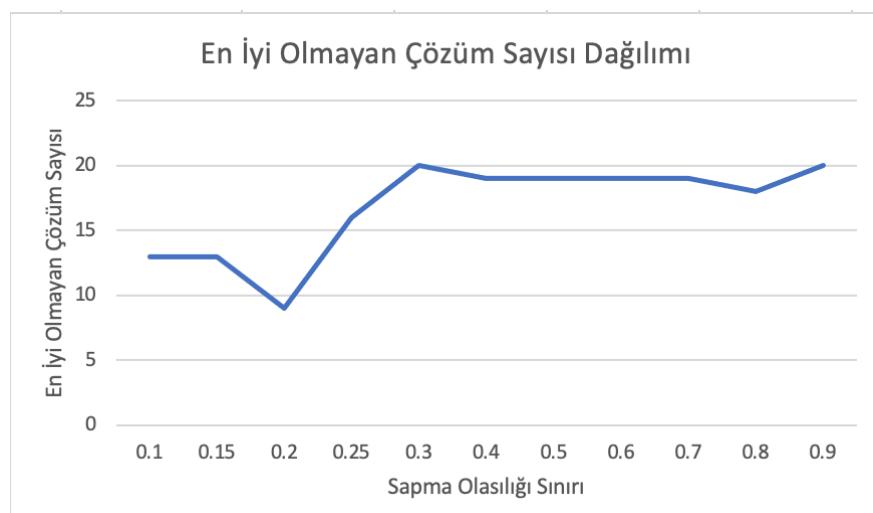
Meta-Raps önceki kısımlarda da anlatıldığı üzere bir açgözlü ve bir geliştirici kısımdan oluşan bir algoritmaydı. Sapma sınırı değişkeni ise açgözlü çözümle seçilen uçaktan sapması için ihtimal belirtiyordu. Bu bağlamda sapma sınırı 0.2 olduğunda (1-0.2) yani 0.8 ihtimalle açgözlü çözümden farklı bir uçak seçiliyordu. Tüm bunların eşliğinde sapma sınırı için 0.1–0.9 değerleri arası değerler denendi ve aşağıdaki tablo elde edildi.

TABLO 3.9. Meta-Raps Sapma Sınırı Deneyi Tablosu

Veri Örneği	0.1	0.15	0.2	0.25	0.3	0.4
0	1008	1011,2	1015,5	1012,9	1012,1	1018,9
1	947,4	941,1	941,1	948,1	946,8	953,2
2	926,8	926,2	925,9	928,3	929,8	930,3
3	917,8	919,1	921	920,7	922,4	926,9
4	1059,9	1059,9	1056,3	1058,3	1059	1059,2
5	939,7	941,3	938,2	940,6	941,5	941,8
6	898	898	898	898	899,8	899,4
7	896	896,5	898,8	895,9	898,4	903,9
8	898	898	898	899,2	898,6	898
9	927	927	927	927	927,6	928,2
10	948,3	946,9	950,4	950,8	950,8	955,1
11	947,7	943,3	944,9	947,9	950,4	950,3
12	958,4	957	959,1	958,4	957,7	959,1
13	912,5	912,5	911	912,5	914	914,4
14	921,6	918,5	915,5	915,5	917,1	928,6
15	1340,4	1328,6	1335,3	1321,7	1305	1307,6
16	1505,7	1506,2	1502,4	1489,8	1497,6	1479,7
17	1348,3	1331,1	1323,7	1335,2	1336,2	1335,4
18	1299,1	1301	1286,3	1279,8	1288,4	1268,5
19	1296,4	1303,6	1292,7	1300	1287,2	1281,8
En İyi Olmayan	13	13	9	16	20	19
Çözüm süresi	921	913	1103	1075	1116	1042
Ölçekli Hata	0,166093937	0,13999552	0,12127795	0,12569065	0,12841374	0,14173205

0.5	0.6	0.7	0.8	0.9	En İyi Değerler
1017,4	1016,8	1016,9	1018,9	1031,2	1008
949,4	957,6	965,8	983	1003,5	941,1
932,6	934,5	938,8	939,8	948,9	925,9
933,3	930,6	942,3	967,2	1000	917,8
1059,2	1060,1	1062	1062	1069	1056,3
940,1	942,5	944,5	948,4	958,6	938,2
903,8	910,8	912,4	913	913	898
904,1	907,3	913,4	918,6	927	895,9
902,4	906,8	912,1	913	913	898
928,2	932,3	936,2	945,6	951,5	927
965,7	958,3	976,8	991,5	1022,5	946,9
955	955,8	966,4	993,5	1019,3	943,3
957,7	957	960,7	961,7	969,9	957
915,2	917	922,4	928,8	946	911
927,8	931,5	938	944,9	950	915,5
1298,4	1289,1	1285,8	1294,9	1304,6	1285,8
1481,3	1486,7	1481,1	1478,4	1497,9	1478,4
1345,6	1341,8	1365,5	1383,4	1416,7	1323,7
1274,8	1261,5	1264	1261	1275,1	1261
1261,5	1266,5	1261,9	1273,6	1286,9	1261,5
19	19	19	18	20	9
1068	889	690	672	572	572
0,15912948	0,17574176	0,28213072	0,43790282	0,71141754	0,121278

Ayrıca yukarıdaki tablodan hareketle aşağıdaki grafikler elde edildi.



ŞEKİL 3.16. Sapma Olasılığı Sınırına Göre En İyi Olmayan Çözüm Sayısı Dağılımı Grafiği



ŞEKİL 3.17. Sapma Olasılığı Sınırına Göre Ölçekli Hata Dağılımı Grafiği

Ölçekli hata ve en iyi olmayan çözüm sayısı grafiklerine bakıldığında net bir davranış sonucu 0.2 sapma oranının en iyi olduğu görülmektedir. Bunun nedeni ise açıkça 0.9 gibi büyük sapma olasılıklarında geliştirilecek çözümün çok az farklılaşması olarak gösterilebilir. Aksi yandan 0.2 den küçük sapma olasılıklarında daha kötü sonuç vermesi ise ağzolu çözümün her ne kadar en iyi olmasa da belli bir iyilikte sonuç vermesi ve ondan çok fazla farklılaşmasıyla alakalı olabilir. Tüm bunların sonucunda ise en az ölçekli hatayı çeken sapma olasılığı 0.2 seçilmiştir.

3.3.2. İterasyon Sayısı Deneyi

İterasyon sayısı deneyinde algoritmanın davranışına ve harcadığı süreye göre 100-5000 aralığı seçildi. Bu algoritmada iterasyon sayısı iyileşmeyle paralel olarak arttığı için yine SA ve GA deki gibi belli değerlerden sonra iyileşme çok az olacaktır. Burada ki amaçta iyileşmenin azaldığı sayıyı bulmaktır. Tüm bunların eşliğinde iterasyon sayısı için 0.1–0.9 değerleri arası değerler denendi ve aşağıdaki tablo elde edildi.

TABLO 3.10. İterasyon Sayısı Deneyi Tablosu

Veri Örneği	iterasyon 500	iterasyon 1000	iterasyon 1500	iterasyon 2000	iterasyon 2500
0	1274,9	1033	1028,4	1026,3	1022,7
1	1183,5	970	955,5	950,1	949
2	1065,3	933,2	932,6	930,2	929,8
3	1098,8	935,4	931,1	927,8	925,8
4	1292,5	1100	1072,5	1063,6	1062,2
5	1168,1	951,7	943	943	942,4
6	996,5	902,3	900,4	899,8	899,2
7	1085,2	913,3	905,1	904,2	903,9
8	993,6	900,7	899,2	899,2	898
9	1066,2	929,7	929	927,7	927,7
10	1157	977,6	962,3	955,2	953
11	1186,9	972,9	959,8	954,6	951
12	1182,8	969,5	963,4	962,7	962,6
13	1084	924,8	917	912,5	911
14	1070,4	931,4	922,4	921,8	920,9
15	1723,7	1389,7	1378,3	1368,3	1360
16	1941,6	1575,6	1533,5	1529,9	1526
17	1743,4	1367,5	1340,2	1335,9	1329,9
18	1664	1327,6	1319,9	1318,3	1311,6
19	1762,1	1354,9	1340,1	1328,6	1318,9
Ölçekli Hata	4,678907513	0,584334255	0,379411306	0,310718709	0,263560854
iterasyon 3000	iterasyon 3500	iterasyon 4000	iterasyon 4500	iterasyon 5000	
1020,5	1019,5	1019	1015,5	1015,5	
949	948,3	947,9	946,3	941,1	
927,3	927,3	926	925,9	925,9	
925,5	925,5	924,2	923,1	921	
1061,3	1060,1	1058,3	1056,3	1056,3	
942,4	940,9	940	939,1	938,2	
898	898	898	898	898	
903,7	902,1	899,6	898,8	898,8	
898	898	898	898	898	
927,6	927,6	927,6	927,6	927	
951,7	951,7	951,7	950,4	950,4	
949,6	949,6	948,6	946,1	944,9	
961,2	960,5	960,5	960,5	959,1	
911	911	911	911	911	
920	920	918,5	917,6	915,5	
1354,5	1343,4	1343,4	1342,8	1335,3	
1519,9	1518,6	1515,2	1504,7	1502,4	
1327,7	1327,7	1324,6	1323,7	1323,7	
1300	1300	1293	1286,9	1286,3	
1302	1299,1	1298,4	1292,7	1292,7	
0,217867514	0,19906849	0,176426813	0,143591367	0,121277948	

Bu tablolar kullanılarak aşağıdaki grafik elde edildi.



ŞEKİL 3.18. İterasyona Göre Ölçekli Hata Dağılımı Grafiği

Açıkça görüldüğü üzere verilen değerler içinde 5000. iterasyonda en iyi değerler bulundu. Bu değerlerin iterasyon sayısı ileri gittiğinde iyileşebileceği öngörülse de belirlenen iterasyon sınırlarından sonra algoritmanın görelî çalışma süresi çok artacağı için tercih edilmedi. Sonuç olarak iterasyon sayısı 5000 olarak seçildi.

4. TARTIŞMA VE SONUÇ

Bu bitirme projesinde hali hazırda literatürde de bulunan Hava Trafik Kontrol işlemlerinden biri olan Uçak İniş Problemi ve Uçak Kalkış Problemi üzerinde duruldu. Bu probleme ilişkin literatürdeki yayınlar incelenerek bir optimizasyon problemi devşirildi. Devşirilen bu problem literatürde olan üç açgözlü yöntem ve iki meta-sezgisel yöntemle çözüldü. Buna ek olarak literatürde olmayan ve devşirilen bu probleme GA' da uygulandı.

Uygulanan bu algoritmalar belirli eniyilemeleri gerçekleştirse de daha da iyileştirmek amacıyla her algoritma özeline deneyler planlayıp yaparak ardından tablolaştırip görselleştirerek her algoritmanın kendi içerisinde en iyi halleri bulundu yani hiper parametre optimizasyonu yapıldı. Hiper parametre optimizasyonu sonucu tüm algoritmalar kullanılan veri setinde denendi.[EK A.] Bu denemeler neticesinde en iyi sonucu AATCSR açgözlü çözümüyle ilklendirilmiş SA algoritması verdi.

En iyi halleri bulunan ve son hallerine getirilen algoritmaların çalışabileceği bir arayüz ve harita üzerinde uçakların sıralamasını ve başlama zamanlarını gösteren bir araç yapıldı.[EK B.]

Bu proje ALP veya ATP problemi üzerinde çalışacak ve kullanılan algoritmalar hakkında bilgi ve geliştirme fikirleri almak için giriş niteliğinde bir çalışma olabilir.

5. KAYNAKLAR

- [1] HANÇERLİOĞULLARI,G. ,RABADİ,G., AL-SALEEM,A., KHARBECHE,M., Greedy algorithms and metaheuristics for a multiple runway combined arrival-departure aircraft sequencing problem, *Journal of Air Transport Management Volume 32*, 2013, Sayfa 39-48.
- [2] POTTS,C., MESGARPOUR,M., BENNELL,J., A Review Of Airport Runway Optimization, *University of Southampton, School of Mathematics, Doktora Tezi*, 2009.
- [3] SEVİLGEN,F.E.(2020). Optimization(CSE424), *Ders sunumları*, 2021
- [4] LEE,H., BALAKRISHNAN,H., A Comparison Of Two Optimization Approaches For Airport Taxiway And Runway Scheduling, *2012 IEEE/AIAA 31st Digital Avionics Systems Conference (DASC)*, 2012.
- [5] CLARE,G., RICHARDS,A., Optimization of Taxiway Routing and Runway Scheduling, *IEEE Transactions on Intelligent Transportation System* , 2011.
- [6] RAO,K., SELLADURAI,V., SARAVANAN,R., TRIZ Tool for Optimization of Airport Runway, *Part of the IFIP Advances in Information and Communication Technology book series (IFIP AICT, volume 304)*, 2009.
- [7] BENLIC,U., BROWNLEE,A., BURKE,E. Heuristic search for the coupled runway sequencing and taxiway routing problem, *Transportation Research Part C: Emerging Technologies Volume 71*, 2016, Sayfa 333-355
- [8] DURAND,N., ALLIOT,J. Ant Colony Optimization for Air Traffic Conflict Resolution, *Eighth USA/Europe Air Traffic Management Research and Development Seminar*, 2009
- [9] OpenSky REST API - The OpenSky Network API 1.4.0 documentation, <https://opensky-network.org/apidoc/rest.html>, [Kasım 2020].

- [10] Geodose, <https://www.geodose.com/2020/08/create-flight-tracking-apps-using-python-open-data.html>, [Kasım 2020]
- [11] RapidAPI, <https://rapidapi.com/category/Travel%2C%20Transportation>, [Kasım 2020].

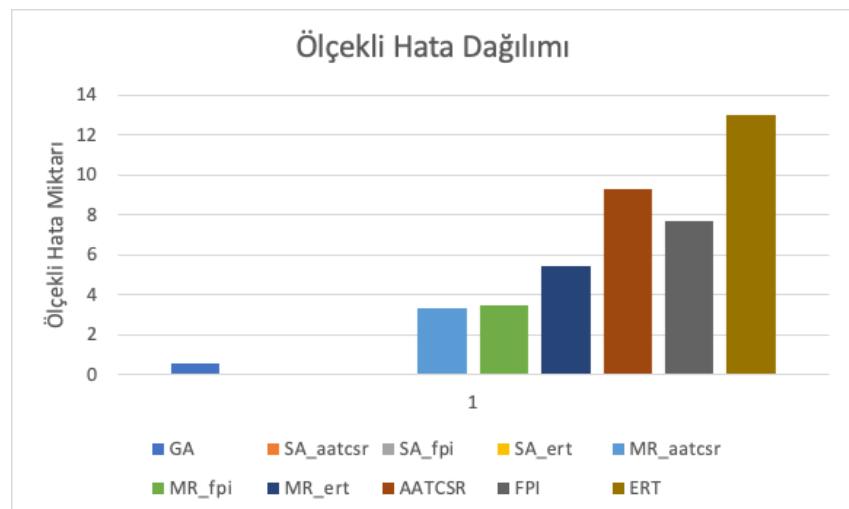
6. EKLER

A. Eldeki Tüm Veriler İle Algoritmaların Çalıştırılması

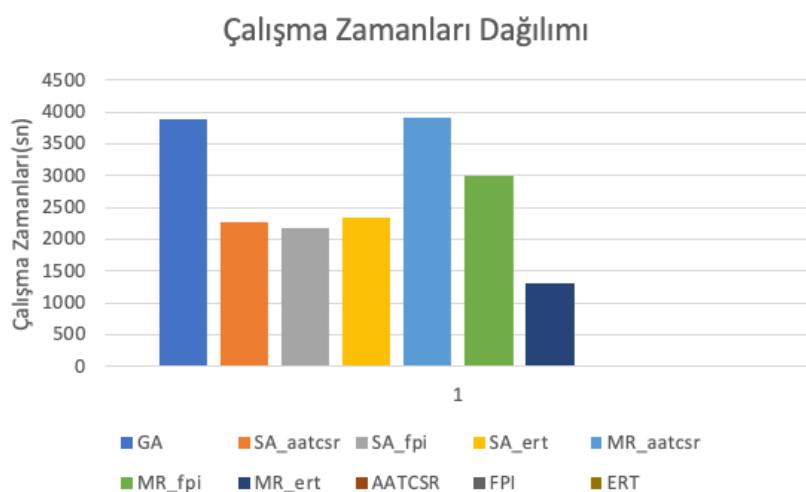
TABLO 6.1. Tüm Deneylerin Çalıştırılması Sonucu Elde Edilen Tablo

Veri Örn	GA	SA_aatcsi	SA_fpi	SA_ert	MR_aatcsi	MR_fpi	MR_ert	AATCSR	FPI	ERT	En İyi Değerler
0	994	994	994	994	1003	1004,9	1005,3	1.145	1044	1216	994
1	930	930	930	930	942	943,6	949,4	1105	1209	1036	930
2	907	907	907	907	923,4	923,5	931,2	1089	963	1104	907
3	904	904	904	904	925,8	914,8	930,1	1272	1039	1332	904
4	1052	1052	1052	1052	1065,2	1058,1	1061,8	1277	1179	1241	1052
5	931	931	931	931	935,5	935	942,4	1167	952	1229	931
6	898	898	898	898	898	898	898	980	973	1040	898
7	879	879	879	879	889,3	887,7	891,3	968	918	1104	879
8	898	898	898	898	898	898	898	980	913	1040	898
9	927	927	927	927	927	927	927	1023	947	1008	927
10	943	943	943	943	944,3	943	948,9	1163	1014	1178	943
11	943	943	943	943	943	944	944,6	1163	1014	1178	943
12	957	957	957	957	958,1	957	960	1090	957	1152	957
13	899	899	899	899	904,7	899	906,5	1095	1035	989	899
14	911	911	911	911	914,6	911	911	1031	1016	1076	911
15	1222	1222	1222	1222	1311,5	1299,7	1359,1	1511	1382	1586	1222
16	1356,1	1364,2	1366,3	1365,7	1482,1	1467	1554,1	1.610	1517	1817	1356,1
17	1237,5	1236	1236	1236	1312,1	1290,1	1401,5	1529	1408	1574	1236
18	1231,6	1225,3	1223,8	1222	1279,7	1275	1337,7	1312	1276	1312	1222
19	1210,5	1209	1209	1209	1274,6	1295,8	1354,8	1461	1395	1582	1209
20	1186	1186	1186	1186	1212,2	1206,8	1214,1	1291	1284	1336	1186
21	1253,6	1253,5	1253,4	1253,1	1269,4	1266,6	1282,9	1505	1321	1593	1253,1
22	1223	1223	1223	1223	1238,3	1233,5	1282,8	1483	1423	1552	1223
23	1263	1263	1263	1263	1293,8	1293	1313,9	1522	1507	1654	1263
24	1250,6	1250	1250	1250,1	1267,1	1261	1278,1	1.502	1318	1591	1250
25	1243	1243	1243	1243	1253,8	1249,3	1261,3	1342	1249	1410	1243
26	1239	1239	1239	1239	1251,2	1244,7	1254,6	1344	1260	1406	1239
27	1254	1254	1254	1254	1279	1277,4	1294,5	1532	1396	1517	1254
28	1192,4	1192,7	1192,6	1193,6	1203,6	1203	1210,2	1215	1275	1277	1192,4
29	1190	1190,5	1190	1190,2	1209,6	1208,1	1217,4	1248	1302	1519	1190
30	1242	1244	1242	1243,4	1273,6	1263,5	1291,7	1256	1256	1460	1242
31	1217	1217	1217	1217	1217,7	1217	1220,2	1413	1247	1465	1217
32	1266	1266	1266	1266	1276,9	1271,9	1279	1.407	1437	1405	1266
33	1269	1269	1269	1269	1274,5	1277,8	1288,1	1425	1455	1408	1269
34	1246	1246	1246	1246	1262,4	1257,3	1277,2	1.359	1291	1502	1246
35	1628,3	1626,5	1625,4	1626,2	1835,2	1831,4	1881,2	2137	2137	1907	1625,4
36	1561,3	1556	1556	1556	1730,3	1744,1	1875,8	1646	1661	1646	1556
37	1584,4	1580	1580	1580	1812,7	1801,1	1963,3	1895	1835	2111	1580
38	1614	1592,6	1591,8	1592,6	1809,8	1848,3	2005,4	1949	2180	2249	1591,8
39	1616,1	1598,4	1598	1598,1	1879,9	1848,5	1922,1	1936	2338	1823	1598
40	1579,3	1565,2	1568,4	1568,7	1696	1724,1	1766,4	1.746	1711	2028	1565,2
41	1594,4	1592	1592	1592	1708,5	1714,7	1775,6	1703	2049	2004	1592
42	1529,4	1527,2	1527	1527,3	1579,4	1575,2	1636,2	1703	1736	1943	1527
43	1546,2	1545,4	1545,1	1544,6	1682,9	1676,1	1845,5	1756	1688	1837	1544,6
44	1541	1533	1533	1533	1623,1	1654,4	1692,5	1763	1791	1851	1533
45	1549	1549	1549	1549	1578,1	1578,6	1632,4	1646	1583	1812	1549
46	1577	1577	1577	1577	1610,8	1616,7	1680,8	1.794	1592	1944	1577
47	1548,9	1547	1547	1547	1588,8	1596,5	1605,4	1.761	1851	1962	1547
48	1627,1	1627	1627	1627	1717,4	1701,9	1739,4	1.790	1828	1999	1627
49	1548	1548	1548	1548	1583,1	1581,7	1611,6	1.721	1813	1781	1548
50	1571,4	1566	1566	1566	1612,3	1612,2	1648,9	1.796	1617	2139	1566
51	1527,3	1527	1527	1527	1557,1	1549,7	1556,3	1.811	1630	1974	1527
52	1564	1564,2	1564	1564	1603,9	1599,8	1644	1.794	1630	2136	1564
53	1542,2	1539	1539,5	1539	1581,6	1580,5	1618,9	1584	1584	1896	1539
54	1559	1559	1559	1559	1601	1602,4	1638	1647	1610	1845	1559
55	3602,6	3230,6	3233,5	3232,4	3991,7	4197,4	4755,4	4.738	4748	3780	3230,6
56	3807,9	3427,6	3427,2	3426,6	4617,2	4653,9	4872,1	3825	4399	3904	3426,6
57	3574,9	3318,2	3319	3319,3	3826,4	3976,4	4608,2	3.534	3819	4084	3318,2
58	3475,1	3239,4	3240	3240,6	4216,4	4238,5	4478,5	4.176	4028	3610	3239,4
59	3544,3	3165,4	3169	3166,1	4349,1	4435,3	4804,3	3461	4954	3766	3165,4

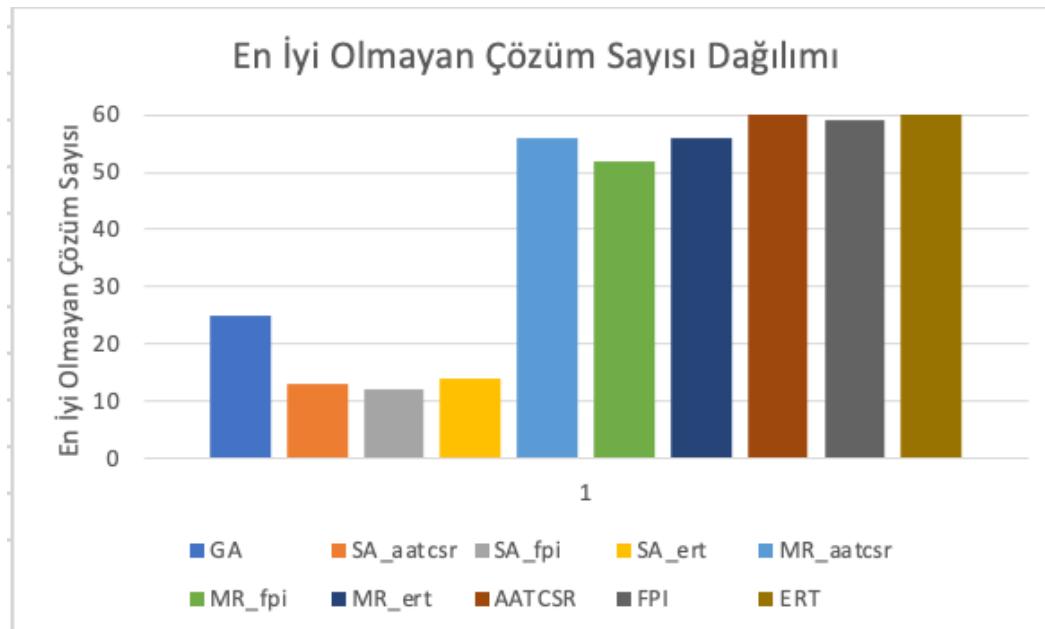
Ölçekli Hata	0,5660408	0,01377304	0,0147311	0,014431	3,36515298	3,4524208	5,458556	9,3025057	7,6689489	13,022092	0,013773
Çalışma Süresi(sn)	3882	2277	2177	2328	3918	3003	1314	2	2	2	2
En İyi Olmayan Sayısı	25	13	12	14	56	52	56	60	59	60	12



ŞEKİL 6.1. Tüm Deneylerin Çalıştırılması Sonucu Elde Edilen Ölçekli Hata Dağılımı Grafiği



ŞEKİL 6.2. Tüm Deneylerin Çalıştırılması Sonucu Elde Edilen Çalışma Zamanları Dağılımı Grafiği



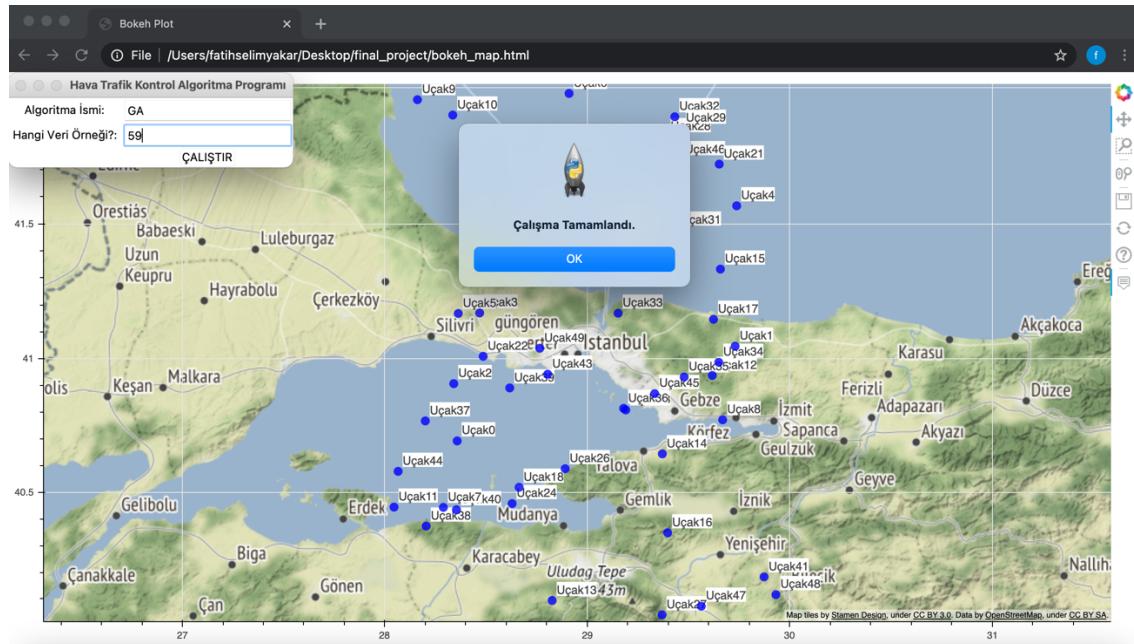
ŞEKİL 6.3. Tüm Deneylerin Çalıştırılması Sonucu Elde Edilen En İyi Olmayan Çözüm Sayısı Dağılımı Grafiği

Tüm bu deneylerin çalıştırılması sonucunda en az hatalı çalışan algoritma AATCSR ile ilklendirilmiş SA algoritması oldu. Bu algoritmanın ardından sırayla GA,Meta-Raps ve açgözlü algoritmalar geldi.

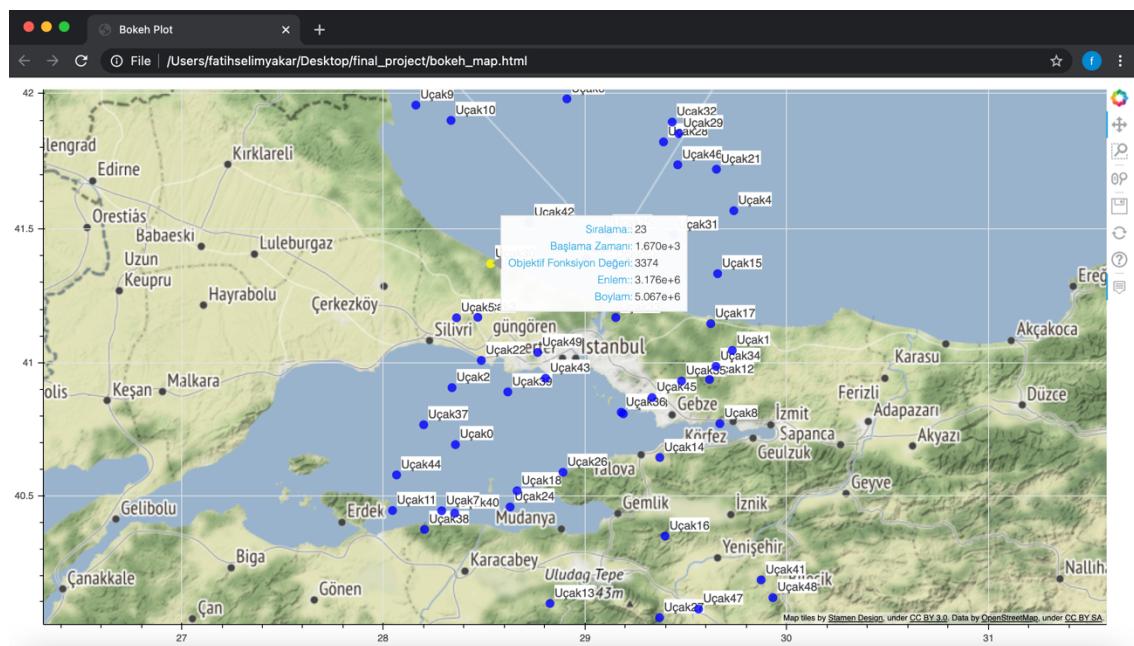
En yavaş çalışan algoritma ise bekleniği üzere GA oldu. Bunun nedeni ise GA'ının popülasyon tabanlı bir algoritma olmasından ötürü her bir iterasyonda popülasyon sayısı kadar objektif fonksiyon hesaplamasından dolayıydı. Buna karşın SA algoritması yönlüğe tabanlı (trajectory based) bir algoritma olduğundan dolayı her iterasyonda geliştirmesi gereken tek bir çözümü hesapladığı için çok daha kısa sürdü.

En iyi olmayan çözüm sayısı kısmında ise bazı farklar olsa da ölçekli hata miktarına paralel şekilde sonuçlar çıktı.

B. Algoritmaları Çalıştırın Arayüz



ŞEKİL 6.4. Arayüzün GA ve 59. Veri Örneği İle Çalıştırılması



ŞEKİL 6.5. Arayüzün Sıralama Bilgisini Göstermesi

Bu çalışma/..../2021 tarihinde aşağıdaki jüri tarafından Bilgisayar Mühendisliği Bölümü’nde Lisans Bitirme Projesi olarak kabul edilmiştir.

Bitirme Projesi Jürisi

Danışman Adı	Prof. Dr. Fatih Erdoğan SEVİLGEN	
Üniversite	Gebze Teknik Üniversitesi	
Fakülte	Mühendislik Fakültesi	

Jüri Adı	Prof. Dr. Fatih Erdoğan SEVİLGEN	
Üniversite	Gebze Teknik Üniversitesi	
Fakülte	Mühendislik Fakültesi	

Jüri Adı	Prof. Dr. Erchan APTOULA	
Üniversite	Gebze Teknik Üniversitesi	
Fakülte	Mühendislik Fakültesi	