OCPI STANDARTLARI KULLANILARAK ELEKTRİKLİ ARAÇ ŞARJ İSTASYONLARININ OPTİMİZASYONU

Ahmet Çağrı ŞEN

T.C.
BURSA ULUDAĞ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

OCPI STANDARTLARI KULLANILARAK ELEKTRİKLİ ARAÇ ŞARJ İSTASYONLARININ OPTİMİZASYONU

Ahmet Çağrı ŞEN
0009-0001-6915-4812

Doç. Dr. Arif BAŞGÜMÜŞ
0000-0002-0611-3220
(Danışman)

Doç. Dr. Mustafa NAMDAR
0000-0002-3522-4608
(2.Danışman)

YÜKSEK LİSANS TEZİ
ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

BURSA – 2025
Her Hakkı Saklıdır

ÖZET

Yüksek Lisans Tezi

OCPI STANDARTLARI KULLANILARAK ELEKTRİKLİ ARAÇ ŞARJ İSTASYONLARININ OPTİMİZASYONU
Ahmet Çağrı ŞEN

Bursa Uludağ Üniversitesi
Fen Bilimleri Enstitüsü
Elektronik Mühendisliği Anabilim Dalı

Danışman: Doç. Dr. Arif BAŞGÜMÜŞ
İkinci Danışman: Doç. Dr. Mustafa NAMDAR (Kütahya Dumlupınar Üniversitesi)

Elektrikli araçların (EV) giderek artan şekilde benimsenmesi, verimli, ölçeklenebilir ve birlikte çalışabilir EV şarj istasyonu altyapılarına duyulan ihtiyacı artırmaktadır. EV kullanımının yaygınlaşmasıyla birlikte, istasyon seçiminin optimizasyonu, gerçek zamanlı veri entegrasyonu ve şarj noktası operatörleri (CPO'lar) ile e-mobilite hizmet sağlayıcıları (EMSP'ler) arasındaki birlikte çalışabilirlik gibi kritik sorunlar ortaya çıkmaktadır. Bu çalışma, Açık Şarj Noktası Arayüzü (OCPI) standardı ile Doğrusal Programlama (LP), Benzetilmiş Tavlama (SA), Parçacık Sürü Optimizasyonu (PSO) ve Pekiştirmeli Öğrenme (RL) gibi ileri hesaplamalı optimizasyon yöntemlerini bütünleştiren kapsamlı bir optimizasyon çerçevesi önermektedir.

Bu bağlamda, Flutter ve Dart kullanılarak geliştirilen bir prototip mobil uygulama tanıtılmaktadır. Uygulama, güç erişilebilirliği, birim şarj maliyeti, mesafe ve aracın şarj durumu (SoC) gibi temel parametreler doğrultusunda çevredeki sekiz EV şarj istasyonunu değerlendirmek ve sıralamak üzere tasarlanmıştır. Her bir optimizasyon yöntemi için 0 ile 1 arasında değişen istasyon puanları hesaplanmakta; kullanıcı dostu bir arayüz aracılığıyla kullanıcı konumu, istasyon konumları ve ilgili ölçütler görsel olarak sunulmaktadır.

Önerilen çözüm, kullanıcıların karar verme süreçlerini optimize etmekle kalmayıp, her bir optimizasyon algoritmasının pratik koşullar altında etkinliğine ve uygunluğuna ilişkin değerli bilgiler de sunmaktadır. Ayrıca uygulama, sonuçların CSV dosyaları olarak dışa aktarılmasına olanak sağlayarak, ileri düzey analizler veya daha geniş sistem entegrasyonları için kullanım imkanı sunmaktadır.

Bu çalışma, EV şarj altyapılarında bütünleşmiş optimizasyon çerçevelerine ilişkin mevcut boşlukları ele almakta ve birlikte çalışabilirlik standartlarının ileri düzey optimizasyon teknikleriyle pratik uygulamasını göstererek alana önemli katkılar sağlamaktadır.

Anahtar Kelimeler: Elektrikli Araç Şarjı, Şarj İstasyonu Optimizasyonu, OCPI (Açık Şarj Noktası Arayüzü), Birlikte Çalışabilirlik, Optimizasyon Algoritmaları

2025, xvi + 90 sayfa.

ABSTRACT

MSc Thesis

OPTIMIZATION OF ELECTRIC VEHICLE CHARGING STATIONS USING OCPI STANDARDS

Ahmet Çağrı ŞEN

Bursa Uludağ University
Graduate School of Natural and Applied Sciences
Department of Electronics Engineering

Supervisor: Assoc. Prof. Dr. Arif BAŞGÜMÜŞ
Second Supervisor: Assoc. Prof. Dr. Mustafa NAMDAR (Kütahya Dumlupınar University)

The increasing adoption of electric vehicles (EVs) highlights the critical need for efficient, scalable, and interoperable EV charging station infrastructure. As EV utilization expands, significant challenges arise in terms of optimal station selection, real-time data integration, and interoperability among charging point operators (CPOs) and e-mobility service providers (EMSPs). This study proposes a comprehensive optimization framework integrating the Open Charge Point Interface (OCPI) standard with advanced computational optimization methods, including Linear Programming (LP), Simulated Annealing (SA), Particle Swarm Optimization (PSO), and Reinforcement Learning (RL).

A mobile application was developed using Flutter and Dart to dynamically evaluate and rank eight nearby EV charging stations based on essential parameters such as power availability, charging cost, distance, and vehicle state of charge (SoC). The application computes station scores ranging from 0 to 1 for each optimization method, providing a comparative analysis through an intuitive interface that visually presents the user's location, station locations, and relevant metrics.

This proposed solution optimizes user decision-making processes and provides insights into the performance and applicability of each optimization algorithm under real-world conditions. Additionally, the application enables exporting results to CSV files, supporting further analysis and integration with broader systems.

This research significantly contributes to addressing existing gaps in integrated optimization frameworks within EV charging infrastructures, illustrating the practical implementation of interoperability standards in conjunction with advanced optimization techniques.

Key words: Electric Vehicles (EVs), Charging Station Optimisation, OCPI (Open Charge Point Interface), Interoperability, Optimisation Algorithms

2025, xvi + 90 pages.

TEŞEKKÜR

Yüksek lisans tez çalışmam süresince bilgi ve tecrübesiyle beni yönlendiren, sabrı, anlayışı ve desteğiyle her zaman yanımda olan; motivasyonumu artırarak bu çalışmanın başarıyla tamamlanmasında büyük katkı sağlayan değerli danışmanım Sayın Doç. Dr. Arif BAŞGÜMÜŞ'e en içten teşekkürlerimi sunarım.

Çalışmamın her aşamasında kıymetli görüş ve katkılarıyla beni destekleyen, yol göstericiliğiyle gelişimime katkı sağlayan ikinci tez danışmanım Sayın Doç. Dr. Mustafa NAMDAR'a gönülden teşekkürlerimi sunarım.

Bu zorlu süreçte sabrı, anlayışı ve varlığıyla bana güç katan sevgili eşim Elif ŞEN'e sonsuz teşekkürlerimi sunuyorum.

Güçlü kalmam için bana ilham olan, gülüşüyle yorgunluğumu unutturan canım oğlum Kemal ŞEN'e minnettarım.

Beni yetiştiren ve bugün aramızda olmasa da desteğini her zaman hissettiğim sevgili annemi rahmet ve özlemle anıyorum.

Destek ve ilgisiyle her zaman yanımda hissettiren kıymetli babama teşekkür ederim.

Ve son olarak, fikirleriyle bakış açımı zenginleştiren ve düşünsel açıdan ilham veren değerli kuzenim Çağan KOYUN'a teşekkür ederim.

Ahmet Çağrı ŞEN
10/06/2025

İÇİNDEKİLER

Sayfa

ÖZET....viii
ABSTRACT....ix
TEŞEKKÜR....x
SİMGELER ve KISALTMALAR DİZİNİ....xi
ŞEKİLLER DİZİNİ....xiv
ÇİZELGELER DİZİNİ....xvi
1. GİRİŞ....1
2. KURAMSAL TEMELLER ve KAYNAK ARAŞTIRMASI....4
2.1. Elektrikli Araçlar: Temel Kavramlar ve Eğilimler....4
2.2. Elektrikli Araç Şarj Altyapısı....6
2.3. Açık Şarj Noktası Arayüzü (OCPI) Standardı....9
2.3.1. OCPI'ye Genel Bakış ve Tarihsel Gelişim....9
2.3.2. OCPI Mimarisi ve İletişim Protokolü....10
2.3.3. OCPI Veri Yapıları ve İşlevsellikleri....12
2.3.4. OCPI ve Diğer EV Şarj Protokollerinin Karşılaştırmalı Analizi....14
2.4. Elektrikli Araç Şarj İstasyonları İçin Optimizasyon Kuramı ve Teknikleri....16
2.4.1. Doğrusal Programlama (LP)....16
2.4.2. Doğrusal Olmayan Programlama (NLP)....17
2.4.3. Karma Tamsayılı Doğrusal Programlama (MILP)....17
2.4.4. Dinamik Programlama (DP)....18
2.4.5. Metasezgisel Optimizasyon Teknikleri....19
2.4.6. Pekiştirmeli Öğrenme Teknikleri....24
2.5. Elektrikli Araç Şarj İstasyonlarında Optimizasyon Parametreleri....25
2.5.1. Talep Tarafı Parametreleri....25
2.5.2. Arz Tarafı Parametreleri....26
2.6. Araştırma Boşluğu ve Tezin Motivasyonu....27
2.6.1. Mevcut Literatürde Belirlenen Boşluklar....27
2.6.2. OCPI Standardının Kullanım Gerekçesi....28
2.6.3. Önerilen Araştırmanın Katkısı ve Önemi....29
3. MATERYAL ve YÖNTEM....31
3.1. Materyal....31
3.2. Yöntem....38
3.2.1. Sistem Mimarisi....38
3.2.2. Mobil Uygulama Geliştirme....39
3.2.3. Optimizasyon Algoritmaları....44
3.3. Simülasyon ve Değerlendirme Ortamı....53
4. BULGULAR ve TARTIŞMA....56
5. SONUÇ....84
KAYNAKLAR....87
ÖZGEÇMİŞ....90

SİMGELER ve KISALTMALAR DİZİNİ

Simgeler	Açıklama
a	Ağırlıklandırma katsayısı
A	Kısıt katsayılar matrisi
α	Feromon bilgisinin göreceli önem katsayısı
β	Sezgisel bilginin göreceli önem katsayısı
b	Kısıtların sağ tarafı vektörü
c	Amaç fonksiyon katsayıları
c_1	Bireysel öğrenme katsayısı
c_2	Sosyal öğrenme katsayısı
δ	Delta
e	Doğal logaritma tabanı
E	Enerji farkı
E_i	Enerji tüketimi miktarı (kWh)
ε	Epsilon
F	Fiyat
$f(x)$	Doğrusal olmayan amaç fonksiyonu
G	Güç
g_i	Eşitsizlik kısıtları
$g_i(x)$	i. eşitsizlik türü doğrusal olmayan kısıt fonksiyonu
h_j	Eşitlik kısıtları
$h_j(x)$	j. eşitlik türü doğrusal olmayan kısıt fonksiyonu
k	Duyarlılık katsayısı
m	Toplam eşitsizlik kısıtı sayısı
n	Toplam eşitlik kısıtı sayısı
N_i	i konumundan erişilebilen komşu konumlar kümesi
η_{ij}	i'den j'ye geçişin çekiciliğini belirleyen sezgisel bilgi
p	Toplam eşitlik kısıtı sayısı
ρ	Feromon buharlaşma oranı
r_1, r_2	[0,1] aralığında rastgele üretilen sayılar
S	Çözüm uzayı
s	Optimal maliyet fonksiyonu
T	Sıcaklık parametresi
$tanh$	Hiperbolik tanjant
τ_{ij}	Feromon Yolu
U	Uzaklık
ω	Atalet katsayısı
x	Karar değişkenleri vektörü
$x_i^{(k)}$	i. parçacığın k. iterasyondaki konumu
$v_i^{(k)}$	i. parçacığın k. iterasyondaki hızı
y	Ayrık (ikili) karar değişkenleri vektörü
Z	Zaman
γ	İndirim katsayısı

Kısaltmalar	Açıklama
AC	Alternative Current (Alternatif Akım)
API	Application Programming Interface (Uygulama Programlama Arayüzü)
BEV	Battery Electric Vehicle (Bataryalı Elektrikli Araç)
CPO	Charge Pointer Operator (Şarj Noktası Operatörü)
CSV	Comma-Separated Values (Virgül ile Ayrılmış Değerler)
DC	Direct Current (Doğrusal Akım)
EV	Electric Vehicle (Elektrikli Araç)
EVSE	Electrical Vehicle Supply Equipment (Elektrikli Araç Tedarik Ekipmanı)
eMSP	eMobility Service Provider (eMobilite Hizmet Sağlayıcısı)
FCEV	Full Cell Electric Vehicles (Yakıt Hücreli Elektrikli Araçlar)
HEV	Hybrid Electric Vehicle (Hibrit Elektrikli Araç)
HTTP	Hypertext Transfer Protocol (Bağlantılı Metin Aktarım Protokolü)
HTTPS	Hypertext Transfer Protocol Secure (Güvenli Metin Aktarım Protokolü)
IEA	International Energy Agency (Uluslararası Enerji Ajansı)
ISO	International Organization for Standardization (Uluslararası Standardizasyon Örgütü)
JSON	JavaScript Object Notation (JavaScript Nesne Gösterimi)
OCA	Open Charge Alliance (Açık Şarj Birliği)
OCPI	Open Charge Point Interface (Açık Şarj Noktası Arayüzü)
OCPP	Open Charge Point Protocol (Açık Şarj Noktası Protokolü)
OSCP	Open Smart Charging Protocol (Açık Akıllı Şarj Protokolü)
PHEV	Plug-in Hybrid Electric Vehicle (Şarj Edilebilir Hibrit Elektrikli Araç)
SoC	State of Charge (Şarj Durumu)
TCP/IP	Transmission Control Protocol / Internet Protocol (Aktarım Kontrol Protokolü / İnternet Protokolü)
V2G	Vehicle-to-Grid (Araçtan Şebekeye)

ŞEKİLLER DİZİNİ

		Sayfa
Şekil 2.2.1.	Türkiye toplam şarj noktası sayısı.....	9
Şekil 2.3.2.1.	Protokollerin ve rollerinin genel görünümü (ElaadNL, 2017).....	11
Şekil 3.1.1.	Android Studio uygulama geliştirme platformu flutter proje oluşturma ekranı	33
Şekil 3.1.2.	Uygulama geliştirmede kullanılan kütüphaneler.....	34
Şekil 3.2.2.1.	Kullanıcı konumunu alma fonksiyonu.....	39
Şekil 3.2.2.2.	İstasyon bilgilerini Assets'ten yükleme fonksiyonu.....	40
Şekil 3.2.2.3.	İstasyon bilgilerinin bulunduğu JSON dosyası….....	40
Şekil 3.2.2.4.	Coğrafi mesafe hesaplama fonksiyonu.....	41
Şekil 3.2.2.5.	Kullanıcı konumu ve istasyon işaretleyicileri fonksiyonu.....	41
Şekil 3.2.2.6.	İstasyon listesi fonksiyonu.....	42
Şekil 3.2.2.7.	Mevcut şarj seviyesi giriş alanı fonksiyonu.....	42
Şekil 3.2.2.8.	Hesapla butonu ve sonuçlar sayfasına geçiş fonksiyonu.	43
Şekil 3.2.2.9.	Sonuçlar sayfası fonksiyonu.....	43
Şekil 3.2.2.10.	CSV'ye aktar butonu ve CSV'ye dönüştürme fonksiyonu.....	44
Şekil 3.2.2.11.	CSV verisini oluşturma ve kaydetme fonksiyonu.....	44
Şekil 3.2.3.1.	Normalizasyon için kullanılacak değerlerin tanımlandığı fonksiyon.....	44
Şekil 3.2.3.2.	LP optimizasyonu için karar ağırlıklarının tanımlandığı fonksiyon.....	45
Şekil 3.2.3.3.	LP optimizasyonu için hesaplama fonksiyonu.....	46
Şekil 3.2.3.4.	SA optimizasyonu için sıcaklık sabitinin tanımlandığı fonksiyon.....	47
Şekil 3.2.3.5.	SA optimizasyonu için hesaplama fonksiyonu.....	48
Şekil 3.2.3.6.	PSO optimizasyonu için karar ağırlıklarının tanımlandığı fonksiyon.....	49
Şekil 3.2.3.7.	PSO optimizasyonu için hesaplama fonksiyonu.....	50
Şekil 3.2.3.8.	RL optimizasyonu için karar ağırlıkları ve Tanh sabitlerinin tanımlandığı fonksiyon.....	51
Şekil 3.2.3.9.	RL optimizasyonu için hesaplama fonksiyonu.....	52
Şekil 3.3.1.	Uygulama ana ekranı.....	55
Şekil 3.3.2.	Uygulama sonuç ekranı.....	55
Şekil 4.1.	Şehir merkezi senaryosu için seçilen kullanıcı konumu ve istasyonların konumu.....	56
Şekil 4.2.	Şehir merkezi senaryosu için "istasyon_konumlari.json" dosyası.....	57
Şekil 4.3.	Şehir merkezi senaryosu için seçilen konumun simülasyon cihazında aktifleştirilmesi.....	58
Şekil 4.4.	Şehir merkezi senaryosu için uygulama ana ekranı.....	59
Şekil 4.5.	Şehir merkezi senaryosu için optimizasyon sonuçları ekranı (SoC %20 iken).....	59
Şekil 4.6.	Cihaz belleğindeki optimizasyon sonuçları kayıt alanı...	60

Şekil 4.7. Şehir merkezi senaryosu için SoC bazlı seçilen ilk istasyonların Güç-Fiyat dağılımı..... 62
Şekil 4.8. Şehir merkezi senaryosunda LP yöntemi için SoC skor eğrileri..... 62
Şekil 4.9. Şehir merkezi senaryosunda SA yöntemi için SoC skor eğrileri..... 64
Şekil 4.10. Şehir merkezi senaryosunda PSO yöntemi için SoC skor eğrileri..... 65
Şekil 4.11. Şehir merkezi senaryosunda RL yöntemi için SoC skor eğrileri..... 66
Şekil 4.12. Şehir dışı senaryosu için seçilen kullanıcı konumu ve istasyonların konumu..... 67
Şekil 4.13. Şehir dışı senaryosu için uygulama ana ekranı..... 68
Şekil 4.14. Şehir dışı senaryosu için optimizasyon sonuçları ekranı (SoC %20 iken) 69
Şekil 4.15. Şehir dışı senaryosu için SoC bazlı seçilen ilk istasyonların Güç-Fiyat dağılımı..... 71
Şekil 4.16. Şehir dışı senaryosunda LP yöntemi için SoC skor eğrileri..... 71
Şekil 4.17. Şehir dışı senaryosunda SA yöntemi için SoC skor eğrileri..... 72
Şekil 4.18. Şehir dışı senaryosunda PSO yöntemi için SoC skor eğrileri..... 73
Şekil 4.19. Şehir dışı senaryosunda RL yöntemi için SoC skor eğrileri..... 74
Şekil 4.20. Uzun yol senaryosu için rota üzerinde seçilen istasyonların konumu..... 76
Şekil 4.21. Uzun yol senaryosu için uygulama ana ekranı..... 77
Şekil 4.22. Uzun yol senaryosu için optimizasyon sonuçları ekranı (SoC %20 iken) 77
Şekil 4.23. Uzun yol senaryosu için SoC bazlı seçilen ilk istasyonların Güç-Fiyat dağılımı..... 79
Şekil 4.24. Uzun yol senaryosunda LP yöntemi için SoC skor eğrileri..... 79
Şekil 4.25. Uzun yol senaryosunda SA yöntemi için SoC skor eğrileri..... 80
Şekil 4.26. Uzun yol senaryosunda PSO yöntemi için SoC skor eğrileri..... 81
Şekil 4.27. Uzun yol senaryosunda RL yöntemi için SoC skor eğrileri..... 82

ÇİZELGELER DİZİNİ

			Sayfa
Çizelge	2.3.4.1.	EV Şarj Altyapısında Kullanılan Haberleşme Protokollerinin Karşılaştırılması.....	15
Çizelge	3.3.1.	OCPI EV Şarj İstasyonu Optimizasyonu uygulamasının test edilebilmesi için belirlenen konumlar, koordinatlar, senaryolar ve fiyat/güç profilleri.....	53
Çizelge	4.1.	Şehir merkezi senaryosu için belirlenen istasyonların bilgileri	56
Çizelge	4.2.	Şehir merkezi senaryosunda %20 SoC için sonuçlar.....	60
Çizelge	4.3.	Şehir merkezi senaryosunda %50 SoC için sonuçlar.....	61
Çizelge	4.4.	Şehir merkezi senaryosunda %80 SoC için sonuçlar.....	61
Çizelge	4.5.	Şehir dışı senaryosu için belirlenen istasyonların bilgileri.....	68
Çizelge	4.6.	Şehir dışı senaryosunda %20 SoC için sonuçlar.....	69
Çizelge	4.7.	Şehir dışı senaryosunda %50 SoC için sonuçlar.....	70
Çizelge	4.8.	Şehir dışı senaryosunda %80 SoC için sonuçlar.....	71
Çizelge	4.9.	Uzun yol senaryosu için belirlenen istasyonların bilgileri.....	76
Çizelge	4.10.	Uzun yol senaryosunda %20 SoC için sonuçlar.....	78
Çizelge	4.11.	Uzun yol senaryosunda %50 SoC için sonuçlar.....	78
Çizelge	4.12.	Uzun yol senaryosunda %80 SoC için sonuçlar.....	78

1. GİRİŞ

EV'lerin giderek artan şekilde benimsenmesi, verimli, güvenilir ve ölçeklenebilir bir şarj altyapısına olan ihtiyacı artırmıştır. Küresel ölçekte sera gazı emisyonlarını azaltma ve fosil yakıtlara olan bağımlılığı azaltma çabaları hız kazanırken, hükümetler ve sanayi kuruluşları EV şarj ağlarının yaygınlaştırılmasını hızlandırmaktadır. Ancak, EV kullanıcı sayısının artmasıyla birlikte mevcut elektrikli araç şarj istasyonları (EVCS) altyapısı, özellikle istasyon seçiminin optimizasyonu, gerçek zamanlı veri entegrasyonu ve farklı şarj noktası operatörleri (CPO) ile e-mobilite hizmet sağlayıcıları (EMSP) arasındaki birlikte çalışabilirlik sorunları gibi kullanıcı odaklı karar verme süreçleri bakımından çeşitli zorluklar barındırmaktadır. Bu zorlukların üstesinden gelebilmek için, şarj istasyonlarının verimli kullanımını optimize eden ve EV ekosistemindeki farklı paydaşlar arasında kesintisiz iletişimi sağlayan gelişmiş ağ yönetim çözümlerine ihtiyaç duyulmaktadır.

Kullanıcı taleplerinin artması ve şarj istasyonlarının mülkiyet yapısının merkeziyetsiz bir nitelik taşıması, birlikte çalışabilirliği mümkün kılan standartlaştırılmış iletişim protokollerine olan ihtiyacı artırmaktadır. Bu bağlamda, açık şarj noktası arayüzü (OCPI) protokolü önemli bir gelişme olarak öne çıkmaktadır. OCPI, farklı EV şarj ağı bileşenleri arasında birlikte çalışabilirliği kolaylaştırmak amacıyla geliştirilmiş yeni bir protokoldür. OCPI, EV şarj ağları arasında standart, güvenli ve merkeziyetsiz iletişimi mümkün kılmakta; tarifelerin gerçek zamanlı güncellenmesi, istasyon uygunluk bilgileri ve uzaktan erişim kontrolü gibi işlevleri desteklemektedir.

Birlikte çalışabilirliğe yönelik bu tür gelişmelere rağmen, fiyat, güç, uygunluk ve mesafe gibi çok sayıda dinamik ve heterojen faktörün dikkate alındığı optimal EVCS seçimi sorunu hâlen karmaşık bir problem olarak varlığını sürdürmektedir. Geleneksel yönlendirme algoritmaları ya da en yakın istasyonu seçme yaklaşımları, özellikle aracın mevcut şarj durumu (SoC) gibi kişiselleştirilmiş veriler dikkate alındığında, çok kriterli karar verme gerekliliğini yeterince karşılayamamaktadır. Bu noktada, optimizasyon tabanlı yaklaşımlar sorunun çözümünde umut vadetmektedir. Farklı hesaplamalı tekniklerin kullanımı, daha akıllı ve bağlama duyarlı kararlar alınmasını mümkün

kılmakta; bu sayede kullanıcı memnuniyeti artırılmakta, menzil kaygısı azaltılmakta ve şarj ağı genelinde verimlilik optimize edilmektedir.

Bu çalışma, söz konusu araştırma problemini çok algoritmalı bir optimizasyon çerçevesi önererek ele almaktadır. Geliştirilen çerçeve, veri entegrasyonu amacıyla OCPI protokolünden faydalanmakta ve Flutter çatı sistemi ile Dart programlama dili kullanılarak geliştirilmiş mobil bir uygulama aracılığıyla gerçek zamanlı istasyon değerlendirmesi gerçekleştirmektedir. Uygulama; istasyon güç kapasitesi, şarj birim maliyeti, kullanıcının istasyona olan uzaklığı gibi gerçek zamanlı bilgiler ve kullanıcı girdisi (%SoC) temelinde yakın istasyonları dinamik biçimde analiz etmekte; ardından dört farklı optimizasyon algoritması kullanılarak istasyonlara ilişkin optimal sıralamaları hesaplamaktadır. Bu çalışmada farklı matematiksel optimizasyon yöntemleri teorik açıdan incelenmiş olup, uygulama için Doğrusal Programlama (LP), Benzetilmiş Tavlama (SA), Parçacık Sürüsü Optimizasyonu (PSO) ve Pekiştirmeli Öğrenme (RL) yöntemleri seçilmiştir.

Uygulamanın arayüzü, kullanıcının mevcut konumunu ve kullanıcının çevresindeki 8 adet istasyonu harita üzerinde görüntülemekte, her istasyon için ilgili bilgileri sunmaktadır. Kullanıcı mevcut SoC bilgisini girdikten sonra algoritmalar her bir istasyon için puanlama yapmakta ve karşılaştırmalı bir sıralama sunmaktadır. Ayrıca, sonuçlar virgül ile ayrılmış değerler (CSV) formatında dışa aktarılabilmekte, bu da ileri analiz ya da başka sistemlerle entegrasyon açısından olanaklar sağlamaktadır. Bu çok yöntemli yaklaşım, her bir algoritmanın gerçek dünya koşulları altındaki görece güçlü ve zayıf yönlerini ortaya koymanın yanı sıra, kullanıcı merkezli karar verme süreçlerine katkı sağlayan özgün bir araç sunmaktadır.

Kuramsal açıdan değerlendirildiğinde, araştırma optimizasyon kuramı, akıllı mobilite sistemleri ve iletişim protokolleri gibi alanlardaki mevcut bilgi birikimini temel almaktadır. Literatürde münferit optimizasyon yöntemleri ve şarj protokolleri üzerine önemli ilerlemeler kaydedilmiş olsa da OCPI gibi birlikte çalışabilirlik standartlarını gelişmiş optimizasyon teknikleriyle bir araya getiren entegre çözümler eksiktir. Bu durum, önerilen çalışmanın hem yenilikçi hem de gerekli olduğunu göstermektedir.

Çalışma kapsamında ayrıca, OCPI standardının yapısal ve işlevsel boyutları ayrıntılı biçimde incelenmekte, diğer protokollerle karşılaştırmalı analiz yapılmakta ve OCPI seçiminin gerekçesi sunulmaktadır. Geliştirilen çerçevenin uyarlanabilir yapısı, gelecekte diğer standartlara genişletilmesine ya da talep tahminine dayalı algoritmalarla entegrasyonuna olanak tanımaktadır. Optimizasyon cephesinde ise, kesin yöntemlerin (örneğin LP) sezgisel ve yapay zekâ tabanlı tekniklerle (örneğin RL) birlikte ele alınması; çözüm kalitesi, yakınsama hızı ve hesaplama maliyeti gibi gerçek zamanlı mobil uygulamalar için kritik öneme sahip unsurların kapsamlı biçimde analiz edilmesini sağlamaktadır.

Özetle, bu çalışma, akıllı EV şarj altyapısının gelişimine aşağıdaki katkılarda bulunmaktadır:

- Standartlaştırılmış veri alışverişi için OCPI protokolünün entegrasyonu,
- EVCS seçimi için kullanıcı odaklı bir mobil uygulamanın geliştirilmesi,
- Karar verme süreçlerini desteklemek amacıyla dört farklı optimizasyon yönteminin uygulanması ve karşılaştırılması,
- Ek ölçütlerin dahil edilebildiği ve akıllı şehir sistemlerine genişletilebilecek esnek bir çerçevenin oluşturulması.

Bu çalışmanın geri kalan bölümleri şu şekilde yapılandırılmıştır: Bölüm 2, teorik ve araştırma arka planı sunularak EV altyapısı, OCPI standartları ve optimizasyon yöntemleri ele alınmaktadır. Bölüm 3, sistemin geliştirilmesi için kullanılan yöntemleri, mimari tasarımı, veri entegrasyonunu ve optimizasyon algoritmalarını açıklamaktadır. Bölüm 4, her bir optimizasyon yönteminin çeşitli simülasyon senaryoları kapsamında performans değerlendirmesi yapılmaktadır. Bölüm 5 ise çalışmanın temel bulgularını özetleyerek, olası etkilerini ve gelecekteki araştırma yönelimlerine ilişkin önerileri sunmaktadır.

2. KURAMSAL TEMELLER ve KAYNAK ARAŞTIRMASI

2.1. Elektrikli Araçlar: Temel Kavramlar ve Eğilimler

EV'lerin geçmişi 19. yüzyılın başlarına kadar uzanmaktadır. İlk pratik EV'ler 1880'lerde ortaya çıkmış ve sessiz çalışmaları, kullanım kolaylıkları ve içten yanmalı motorlu araçlara kıyasla daha düşük emisyon seviyeleri nedeniyle popülerlik kazanmıştır. 20. yüzyılın başlarında, ABD yollarındaki araçların yaklaşık üçte biri elektrikliydi (Anderson & Anderson, 2010).

Ancak, Henry Ford'un seri üretim yöntemlerini geliştirmesi ve petrol yakıtlarının daha erişilebilir hale gelmesi, içten yanmalı motorlu araçların maliyetini önemli ölçüde düşürerek EV'lere olan ilgiyi azaltmıştır (Kirsch, 2021).

20. yüzyılın sonlarında, çevresel kaygılar, artan yakıt maliyetleri ve batarya teknolojilerindeki ilerlemeler, EV'lere olan ilgiyi yeniden canlandırmıştır. Bu dönemde, özellikle 1990'larda, fosil yakıtların çevresel etkileri ve enerji güvenliği konularındaki endişeler, EV'lerin geliştirilmesini ve benimsenmesini teşvik etmiştir. Ayrıca, lityum-iyon batarya teknolojisindeki gelişmeler, EV'lerin menzilini artırmış ve performanslarını iyileştirmiştir. Bu faktörler, EV'lerin yeniden popülerlik kazanmasına ve otomotiv endüstrisinde önemli bir dönüşümün başlamasına katkı sağlamıştır. (ABD Enerji Departmanı, 2014)

EV'ler, tahrik sistemlerine göre birkaç temel kategoriye ayrılmaktadır:

Bataryalı Elektrikli Araçlar (BEV): Tamamen elektrikli bataryalar ve motorlar ile çalışan bu araçlar, egzoz emisyonu üretmez ve yalnızca şarj altyapısına bağımlıdır (Kusuma vd., 2025). Togg ve Tesla gibi modeller bu gruba girmektedir.

Hibrit Elektrikli Araçlar (HEV): İçten yanmalı motor ile elektrik motoru ve bataryayı bir araya getirerek daha yüksek yakıt verimliliği ve daha düşük emisyon sunan, ancak harici şarj gerektirmeyen araçlardır (Hu vd., 2018). Fiat Egea Hybrid bu gruba örnektir.

Şarj Edilebilir Hibrit Elektrikli Araçlar (PHEV): HEV'lere kıyasla daha büyük bataryalara sahiptir ve harici kaynaklardan şarj edilebilir, böylece hem BEV'lerin hem de içten yanmalı motorlu araçların avantajlarını birleştirir (Kusuma vd., 2025). Bu araçlar yaklaşık 50 km menzili sadece batarya ile kat edebilir.

Yakıt Hücreli Elektrikli Araçlar (FCEV): Hidrojen yakıt hücreleri kullanarak araç içinde elektrik üreten, yüksek verimlilik sağlayan ve sıfır emisyonla çalışan araçlardır, ancak özel hidrojen dolum altyapısı gerektirirler (Ehsani vd., 2018).

Teknolojik olarak EV'ler birkaç kritik bileşenden oluşmaktadır:

- Elektrik motorları (genellikle kalıcı mıknatıslı senkron motorlar veya endüksiyon motorları)
- Güç elektroniği ve motor kontrol sistemleri (DC-AC invertörler, dönüştürücüler)
- Enerji depolama sistemleri (lityum-iyon ve katı hal bataryalar gibi)
- Şarj arayüzleri ve altyapı uyumluluğu

EV pazarının yaygınlaşması, son yıllarda hükümet politikaları, teknolojik ilerlemeler ve artan çevresel farkındalık gibi faktörlerin etkisiyle üstel bir büyüme göstermektedir. Uluslararası Enerji Ajansı'na (IEA) göre, 2023 yılı itibarıyla küresel EV satışları yaklaşık 14 milyon adede ulaşarak, toplam otomobil satışlarının %18'ini oluşturmuştur. Bu oran, 2022 yılındaki %14'lük paya kıyasla kayda değer bir artışa işaret etmektedir. Bu yükseliş eğiliminin 2024 yılında da devam etmesi beklenmekte olup, satışların yaklaşık 17 milyon adede ulaşacağı ve bu sayede küresel otomobil satışlarının %20'sinden fazlasını oluşturacağı öngörülmektedir. Çin, %60'lık payla en büyük pazar konumunu korurken, onu %25 ile Avrupa ve %10 ile Amerika Birleşik Devletleri izlemektedir. Vietnam ve Tayland gibi gelişmekte olan ekonomilerde de EV'lere yönelik benimseme oranlarının hızla arttığı görülmekte, bu da küresel ölçekte elektrikli mobiliteye yönelik daha geniş kapsamlı bir geçişin işaretlerini vermektedir (IEA, 2024).

EV yayılım oranlarını şekillendiren başlıca faktörler arasında politika ve teşvikler, batarya teknolojilerindeki gelişmeler ve şarj altyapısının genişletilmesi yer almaktadır (Gnann vd., 2018).

EV'lerin hızla yaygınlaşması, elektrik şebekeleri ve enerji altyapısı üzerinde önemli etkilere sahiptir. Özellikle yoğun talep saatlerinde EV şarj yüklerinin artması, tepe talep seviyelerini yükselterek şebeke kararsızlığına neden olabilmekte ve altyapı yatırımlarını gerektirmektedir (Hajimiragha vd., 2010).

Bu nedenle, akıllı şarj stratejileri ve gelişmiş enerji yönetim sistemleri, olumsuz etkilerin azaltılması açısından kritik önem taşımaktadır.

Bununla birlikte, EV'lerin entegrasyonu şebeke üzerinde olumlu etkiler de yaratabilir. Araçtan şebekeye (V2G) teknolojileri sayesinde, EV'ler depoladıkları enerjiyi yüksek talep dönemlerinde şebekeye geri besleyerek şebeke kararlılığına katkı sağlayabilir, tepe yüklerini dengeleyebilir ve yenilenebilir enerji kaynaklarının entegrasyonunu destekleyebilir (Desai vd., 2024).

Mevcut Eğilimler ve Öngörüler:

- EV teknolojisi, özellikle batarya kapasitesi, menzil, verimlilik ve şarj süreleri açısından hızla gelişmektedir.
- Küresel pazar yayılımı, çevresel politikalar ve ekonomik teşvikler sayesinde ivme kazanmaktadır.
- Artan EV benimsenmesi, akıllı şarj yönetimini ve altyapı optimizasyonunu zorunlu kılmaktadır.
- Şebeke üzerindeki etkilerin etkin bir şekilde yönetilebilmesi için akıllı şebeke teknolojileri, talep yönetimi mekanizmaları ve V2G stratejilerinin uygulanması gerekmektedir.

2.2. Elektrikli Araç Şarj Altyapısı

EV şarj altyapısının geliştirilmesi ve etkin bir şekilde işletilmesi, EV'lerin benimsenmesi, güvenilirliği, kullanıcı memnuniyeti ve mevcut enerji şebekelerine entegrasyonu açısından kritik bir faktördür. EV şarj altyapısı; şarj teknolojileri, enerji tedarik ağları,

yönetim sistemleri ve birlikte çalışabilirlik standartları gibi çeşitli unsurları içermektedir (Ehsani vd., 2018).

EV şarj istasyonları, güç seviyeleri, şarj hızları ve teknoloji standartlarına göre farklı kategorilere ayrılmaktadır.

Temel sınıflandırmalar:

AC Şarj Seviye 1 (Yavaş Şarj)
Tipik Gerilim ve Akım: 120 V AC, en fazla 16 A
Güç Aralığı: ~1.4–1.9 kW
Şarj Süresi: 20–40 kWh kapasiteli tipik bir batarya için 8–20 saat
Uygulama Alanı: Başlıca konut kullanımı
Avantaj ve Dezavantajlar: Basit altyapı gerektirir ancak düşük şarj hızı nedeniyle pratik kullanım sınırlıdır.

AC Şarj Seviye 2 (Orta Hızlı Şarj)
Tipik Gerilim ve Akım: 208–240 V AC, en fazla 80 A
Güç Aralığı: ~3.3–19.2 kW
Şarj Süresi: 40–80 kWh kapasiteli tipik bir batarya için 4–8 saat
Uygulama Alanı: Konutlar, ticari alanlar, halka açık alanlar ve iş yerleri
Avantaj ve Dezavantajlar: Pratik kullanım imkânı sunması ve kurulum maliyetlerinin makul olması nedeniyle yaygın olarak tercih edilmektedir.

DC Şarj Seviye 3 (Hızlı Şarj)
Tipik Gerilim ve Akım: 400–1000 V DC, en fazla 500 A
Güç Aralığı: 50–350 kW veya daha yüksek
Şarj Süresi: %80 şarj seviyesi (SoC) için genellikle 20–40 dakika
Uygulama Alanı: Otoyol şarj koridorları, halka açık şarj istasyonları, ticari filolar
Avantaj ve Dezavantajlar: Uzun mesafe seyahatleri için hızlı şarj olanağı sağlar, ancak kurulum ve işletme maliyetleri yüksektir (Acharige, 2023).

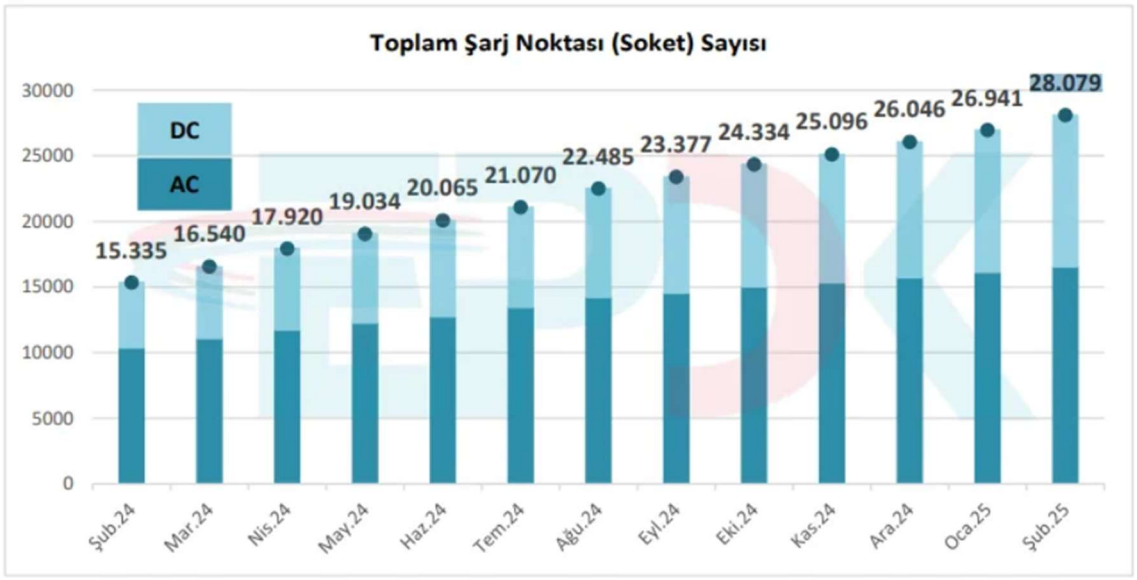
Kablosuz Şarj (İndüktif)

Gelişmekte olan bu sistemde manyetik alan üzerinden güç aktarımı sağlanır. Fiş kullanımı gerektirmediği için kullanıcı dostudur; ancak genellikle 3–11 kW arası güç sunarlar ve kablolu sistemlere göre daha düşük verime sahiptirler. Park pozisyonunun hassas olması gerektiğinden henüz niş uygulama alanına sahiptirler.

Küresel Düzeyde Karşılaşılan Temel Sorunlar: Dünya genelinde karşılaşılan temel zorluklardan biri, istasyonların yeterli coğrafi kapsama alanına sahip olmaması ve farklı şarj ağı operatörleri arasında karşılıklı çalışabilirliğin sağlanamamasıdır. EV kullanım oranı arttıkça hem devlet kurumları hem de özel sektör paydaşları şarj noktalarının yaygınlaştırılması ve altyapının standartlaştırılması konularına daha fazla odaklanmaktadır.

Türkiye'de Şarj Altyapısının Büyümesi: Türkiye, EV kullanımında uzun süre düşük seyirli bir grafik sergilemiş ve bu durum şarj altyapısına da yansımıştır. 2019 yılı itibarıyla ülke genelinde kamusal kullanıma açık toplam şarj noktası sayısı 600'ün altında kalmıştır.

Ancak, 2020 sonrası dönemde Türkiye'de EV şarj altyapısında kayda değer bir ivme yaşanmıştır. 2022 yılında devlet destekleri ve özel sektör yatırımlarıyla 300 milyon Türk Lirası değerinde kaynak aktarımı yapılmış, bu da hızlı şarj istasyonlarının kurulmasını hızlandırmıştır. 2020 yılında yaklaşık 1.000 olan şarj istasyonu sayısı, 2024 yılı itibarıyla 21.000'e ulaşmış ve yirmi katlık dramatik bir artış kaydedilmiştir. Şekil 2.2.1'de görüleceğe üzere, mevcut verilere göre 2025 yılı Şubat ayı itibarıyla Türkiye'de toplam 28.079 adet EV şarj soketi bulunmaktadır. Bunların 16.533 adedi AC şarj noktası, 11.546 adedi ise DC şarj noktası olarak hizmet vermerktedir (Enerji Ajansı, 2025).



Şekil 2.2.1. Türkiye toplam şarj noktası sayısı

2.3. Açık Şarj Noktası Arayüzü (OCPI) Standardı

2.3.1. OCPI'ye Genel Bakış ve Tarihsel Gelişim

OCPI, farklı CPO'lar, eMSP'ler ve diğer paydaşlar arasındaki birlikte çalışabilirliği artırmak amacıyla geliştirilmiş, açık kaynaklı ve standartlaştırılmış bir iletişim protokolüdür. OCPI, veri alışverişini ve iletişim süreçlerini standart hale getirerek kesintisiz gezici şarj ve farklı hizmet sağlayıcıları arasında şeffaf veri paylaşımı sağlamaktadır (Guillemin vd., 2024).

OCPI girişimi, 2014 yılı civarında, Avrupa'da parçalı yapıdaki EV şarj ağları nedeniyle ortaya çıkan birlikte çalışabilirlik sorunlarını çözme ihtiyacı doğrultusunda başlatılmıştır. İlk olarak Hollanda Bilgi Platformu Şarj Altyapısı (NKL) gibi kuruluşlar tarafından desteklenen OCPI, hızla gelişerek uluslararası düzeyde tanınan bir protokol haline gelmiştir ve günümüzde EVRoaming Vakfı tarafından yönetilmektedir. OCPI'nin geliştirilmesi sürdürülmekte olup, en güncel sürüm olan OCPI 2.2.1, gelişmiş işlevsellikler ve veri alışverişi yapıları sunmaktadır (EVRoaming Foundation, 2023).

OCPI, zamanla yalnızca Avrupa'da değil, küresel çapta yaygın olarak benimsenen bir roaming standardına dönüşmüştür. Temmuz 2021 itibarıyla, Avrupa'da 27 farklı CPO

tarafından işletilen yaklaşık 50.000 şarj noktası OCPI üzerinden entegre edilmiştir. Bu standart daha sonra Kuzey Amerika ve Asya'daki şarj ağlarına da yayılmış ve küresel bir norm haline gelmiştir.

OCPI'nin başarısının ardında, protokolün açık kaynaklı, ücretsiz ve topluluk katılımına açık bir yapıda olması yatmaktadır. Geliştirme süreci kolektif olarak yürütüldüğünden, enerji şirketlerinden donanım üreticilerine, mobilite hizmet sağlayıcılarından yazılım geliştiricilerine kadar pek çok farklı aktör protokolün evrimine katkı sunabilmektedir. Bu kapsayıcı yaklaşım sayesinde OCPI, esnekliği ve kapsamlı özellik seti ile Avrupa'da en çok tercih edilen roaming protokolü haline gelmiştir.

2.3.2. OCPI Mimarisi ve İletişim Protokolü

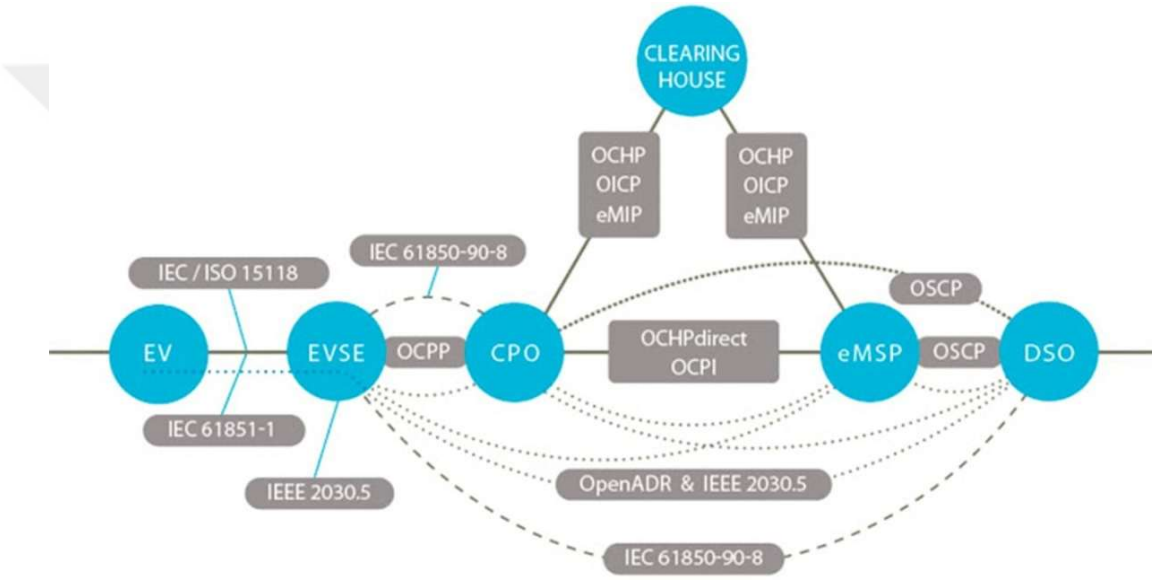
OCPI, CPO ile eMSP arasında, arka uç sistemlerin doğrudan iletişimini mümkün kılan, merkezi olmayan ve web tabanlı bir uygulama programlama arayüzü (API) olarak tasarlanmıştır. OCPI mimarisi temelde eşler arası bir yapıya sahiptir. Bu yapı kapsamında, her bir taraf bir dizi OCPI uç noktası sunar ve bilgiyi doğrudan veya merkezi olmayan bir merkez aracılığıyla diğer tarafla paylaşır. Bu mimari, merkezi bir sunucuya olan bağımlılığı ortadan kaldırarak sistemin esnekliğini ve dayanıklılığını artırır (EVRoaming Foundation, 2023).

OCPI mesajları JSON formatında yapılandırılmış olup, standart HTTP/HTTPS protokolleri üzerinden iletilmektedir. Bu da protokolün web teknolojileri ile kolaylıkla entegre edilebilmesini sağlar. RESTful yaklaşım sayesinde, OCPI hem eş zamanlı hem de gecikmeli işlemleri desteklemektedir. Bu özellik, kullanım senaryosuna bağlı olarak gerçek zamanlı sorguların yapılmasına veya daha sonra gerçekleştirilecek veri alışverişlerine olanak tanır (Guillemin vd., 2024).

Roller ve Topoloji: OCPI sistemi içinde, CPO (bir veya daha fazla şarj istasyonu işleten taraf) ile eMSP (sürücülere hizmet sunan taraf) arasında bir bağlantı, API kimlik bilgilerinin ve uç nokta URL'lerinin paylaşılması yoluyla kurulur. Bu bağlantı doğrudan (ikili eşleşme) olabilir; bu durumda her bir eMSP, her bir CPO ile ayrı ayrı entegre olur.

Alternatif olarak, taraflar yalnızca bir merkezi roaming merkezine bağlanarak dolaylı olarak birbirlerine ulaşabilirler. OCPI her iki modeli de doğal olarak desteklemektedir: başlangıçta eşler arası bağlantılar için geliştirilmiş olsa da, ölçeklenebilirlik amacıyla merkezli yapıları da desteklemektedir.

Önemli bir fark olarak, OCPI protokolü şarj cihazı donanımı ile doğrudan iletişim kurmaz; bunun yerine, işletmelerin arka uç sistemleri arasında bilgi alışverişini sağlar. Örneğin, bir CPO kendi şarj cihazlarıyla OCPP protokolü üzerinden iletişim kurarken, OCPI aracılığıyla bu bilgiyi dış paydaşlarla (eMSP vb.) paylaşabilir (Greenflux, 2025).



Şekil 2.3.2.1. Protokollerin ve rollerinin genel görünümü (ElaadNL, 2017)

İletişim Akışı: OCPI, çift yönlü bir iletişim mekanizması sunar; yani hem CPO hem de eMSP hem istemci hem sunucu gibi davranabilir. Örneğin, bir CPO, bir eMSP'den istasyon bilgisi için GET isteği alabilirken, eMSP bir şarj oturumu tamamlandığında CPO'ya POST isteği ile oturum detaylarını (örneğin enerji tüketimi) iletebilir. Her taraf, sahip olduğu veriye karşılık gelen OCPI modülünü kendi sisteminde barındırır. Başlangıçta taraflar arasında API kimlik bilgileriyle gerçekleştirilen bir doğrulama süreci sonrasında, taraflar birbirlerinin uç noktalarına doğrudan istekte bulunabilir.

Bu mimari, merkezi bir uzlaştırma platformuna bağımlılığı ortadan kaldırarak hem karmaşıklığı azaltır hem de tekil hata noktalarını önler (IDACS Consortium, 2022).

2.3.3. OCPI Veri Yapıları ve İşlevsellikleri

OCPI'nin en güçlü yönlerinden biri, EV şarj ağları arasında ağlar arası geçiş hizmetini mümkün kılacak kapsamlı ve modüler veri yapıları sunmasıdır. Her bir modül, şarj hizmetinin belirli bir yönünü temsil eden standart veri nesnelerini ve bunlara ilişkin işlemleri tanımlar. Bu yapı hem CPO'lar hem de eMSP'ler arasında tutarlı ve uyumlu bir veri alışverişi sağlar.

Konum ve Kullanılabilirlik Verileri:

OCPI; şarj istasyonlarının konum bilgilerini, EVSE, bağlantı tipleri ve çalışma saatleri gibi detaylı bilgileri tanımlar. Bu statik veri modülü, eMSP'lerin sürücülere uygun şarj noktalarını listeleyebilmesi için gereklidir. AC Tip 2, DC CCS gibi farklı şarj konnektör türleri, kapasiteler, engelli erişimi ve açık/kapalı saatler gibi bilgiler sayesinde sürücüler doğru ve erişilebilir istasyonlara yönlendirilebilir (EVRoaming Foundation, 2023).

Konum verileri matematiksel olarak şu şekilde temsil edilebilir;

$$Konum = (Enlem, Boylam) \tag{2.1}$$

Gerçek Zamanlı Durum Güncellemeleri:

OCPI, şarj istasyonlarının gerçek zamanlı doluluk durumlarını (uygun, meşgul, hizmet dışı vb.) ve aktif şarj oturumlarının verilerini destekler. Bu sayede CPO'lar, şarj noktalarının güncel durumlarını eMSP'lerle paylaşır. Şarj oturumu başladığında, CPO oturumun ilerleyişine dair verileri (aktarılan enerji, süre vb.) OCPI üzerinden iletebilir. Bu dinamik veri akışı, sürücülere harita üzerinden gerçek zamanlı bilgi sunulmasını ve şarj noktalarının verimli kullanılmasını sağlar.

Kullanılabilirlik durumu ise gerçek zamanlı statülerle belirtilir;

$$Müsaitlik = (Mevcut, Dolu, Rezerve, Sipariş\ Dışı, Bilinmeyen) \tag{2.2}$$

Yetkilendirme ve Jetonlar:

OCPI, RFID kartlar veya mobil uygulama kimlikleri gibi araç kullanıcılarının tanımlandığı "jeton" bilgilerini işler. eMSP'ler tarafından verilen bu jetonlar, OCPI

üzerinden CPO'lara aktarılır ve böylece şarj işlemi başlatılırken yetkilendirme sağlanır. OCPI, hem çevrim içi yetkilendirme (her işlemde eMSP sistemine sorgu) hem de çevrim dışı beyaz liste (önceden yüklenmiş izinli jetonlar) yöntemlerini destekler. Bu yapı, farklı ağlar arasında sorunsuz şarj oturumu başlatılmasına olanak tanır.

Tarife ve Fiyatlandırma Verileri:

OCPI'nin Tarife modülü, şarj hizmetlerinin fiyatlandırma politikalarını (kWh başına ücret, dakika bazlı ücret, sabit ücretler vb.) standart formatta iletmek için kullanılır. CPO'lar, fiyatlandırma yapılarını şeffaf şekilde paylaşabilir; böylece eMSP'ler bu bilgileri kullanıcıya tutarlı biçimde sunabilir. Kullanıcıların fiyat hakkında önceden bilgilendirilmesi, hizmet deneyimini iyileştirir ve ağlar arası geçişte sürpriz maliyetleri engeller (EVRoaming Foundation, 2023).

Genel fiyatlandırma modeli şu şekilde ifade edilir;

$$Maliyet_{toplam} = \sum_{i=1}^{n} \left(E_i \; x \; Fiyat_{enerji,i}\right) + \sum_{j=1}^{m} \left(Zaman_j \; x \; Fiyat_{zaman,j}\right) + SabitÜcret \quad (2.3)$$

Denklem (2.3), toplam enerjiye dayalı maliyet, zaman temelli maliyet ve sabit ücretin toplamından oluşmaktadır.

Burada;

E_i: i.zaman aralığında tüketilen enerji (kWh)

$Fiyat_{enerji,i}$: $i.$ zaman aralığında kWh başına enerji fiyatı

$Zaman_j$: $j.$ zaman aralığında şarj süresi

$Fiyat_{zaman,j}$: $j.$ zaman aralığında zaman bazlı tarife

$SabitÜcret$: sabit oturum ücreti

Bu denklem, özellikle OCPI 2.2.1 protokolünde tanımlanmış, EV şarj istasyonlarında müşterilere doğru ve şeffaf ücretlendirme yapmak amacıyla standartlaştırılmıştır.

Şarj Oturumu Yönetimi:

OCPI oturum modülü, gerçek zamanlı şarj oturumlarının takibini ve yönetimini sağlar. Bu sistem, oturum başlatma, tüketilen enerji, şarj süresi, kullanıcı kimlik doğrulaması ve durum değişiklikleri gibi unsurları içerir. Oturum yönetimi, doğru faturalandırma, şeffaf izleme ve operasyonel yönetimi sağlar (Guillemin vd., 2024).

Standart bir oturum verisi şu şekilde temsil edilmektedir;

$$Oturum_{verisi} = \left\{ \begin{matrix} OturumID, KullanıcıID, BaşlangıçZamanı, BitişZamanı, \\ TüketilenEnerji(kWh), Durum \end{matrix} \right\} \quad (2.4)$$

OCPI, EV sürücülerinin abonelik yaptıkları hizmet sağlayıcılar dışındaki şarj istasyonlarını kullanabilmesini sağlayarak birlikte çalışabilirliği artırır. Gezici hizmetler (roaming), farklı CPO'lar ve eMSP'ler arasında kimlik doğrulama jetonlarının, kullanıcı bilgileri ve oturum verilerinin güvenli bir şekilde değişimini içerir. Böylece kullanıcılar, birden fazla hesap veya abonelik gereksinimi olmadan farklı hizmet sağlayıcıların şarj istasyonlarına erişebilirler (Ampeco, 2025).

Gezici hizmetler matematiksel olarak aşağıdaki gibi temsil edilmektedir;

$$Dolaşım_{yetkilendirme} = f(Jeton_{Kullanıcı}, KimlikBilgisi_{eMSP}, KimlikBilgisi_{CPO}) \quad (2.5)$$

2.3.4. OCPI ve Diğer EV Şarj Protokollerinin Karşılaştırmalı Analizi

Çizelge 2.3.4.1, OCPI'nin diğer EV şarj protokolleriyle ilişkisini ve benzersiz yönlerini özetlemektedir.

Çizelge 2.3.4.1. EV Şarj Altyapısında Kullanılan Haberleşme Protokollerinin Karşılaştırılması

Protokol	OCPI	OCPP	OSCP	ISO 15118
Kullanım Alanı	Gezici hizmetler, sağlayıcılar arası iletişim	Şarj istasyonu yönetimi	Şebeke kapasite yönetimi	EV-Şarj Cihazı iletişimi
İletişim Katmanı	RESTful API (HTTP/JSON)	WebSocket/HTTP (JSON/SOAP)	Web tabanlı (HTTP/JSON)	TCP/IP, Güç Hattı İletişimi
Odak Noktası	İşletmeler arası birlikte çalışabilirlik	Şarj noktası yönetimi ve izleme	Şebeke desteği, yük yönetimi	Araç kimlik doğrulama, otomatik şarj
Geliştirici Kuruluş	EVRoaming Vakfı	Open Charge Alliance (OCA)	Open Charge Alliance (OCA)	Uluslararası Standardizasyon Örgütü (ISO)

OCPI ve OCPP EV şarj ekosisteminde sıklıkla birlikte anılmalarına rağmen, farklı katmanlarda hizmet verirler ve birbirlerini tamamlayıcı niteliktedirler. OCPP, esasen şarj cihazı (donanım) ile şarj istasyonu yönetim sistemi (genellikle CPO'nun merkezi yazılımı) arasındaki iletişimi standartlaştırır. Bu iletişim, şarj başlatma/durdurma komutları, hata bildirimi, ölçüm verileri gibi düşük seviyeli işlemleri kapsar. Her iki protokol de açık kaynaklıdır ve kendi alanlarında geniş çapta benimsenmiştir. OCPP (2010 yılında, yine Hollanda'da geliştirilmiştir), farklı donanım markalarının aynı platformda çalışabilmesini sağlayarak donanım-yazılım uyumluluğunu mümkün kılmıştır. OCPI ise kullanıcıların farklı şarj ağlarında tek hesapla hizmet alabilmesini sağlayarak hizmet düzeyinde birlikte çalışabilirliği tesis eder. Temel fark, OCPP'nin şarj donanımına yönelik düşük seviyeli operasyonel kontrolü sağlaması, OCPI'nin ise şarj hizmetlerinin bulunabilirliği, fiyatlandırılması, kimlik doğrulaması ve faturalandırma gibi üst seviye işlevleri yönetmesidir. Pratikte sağlam ve kullanıcı dostu bir EV şarj ağı için her iki protokolün birlikte kullanılması gereklidir: OCPP ile teknik yönetim, OCPI ile şebekeler arası kullanıcı deneyimi sağlanır.

ISO 15118 standardı, EV ile şarj istasyonu arasındaki iletişimi (fiziksel ve veri bağlantı katmanında) düzenleyen uluslararası bir protokoldür. Bu standardın en bilinen özelliği, araçla istasyon arasında sürücünün manuel etkileşimine gerek kalmadan otomatik kimlik doğrulama sağlayan "Tak ve Şarj Et" fonksiyonudur. Bu sayede araç, şarj cihazına

bağlandığında kendini otomatik olarak tanıtır; güvenli bir el sıkışma süreciyle şarj işlemi başlatılır ve ücretlendirme yapılır. ISO 15118 ayrıca, araçla şebeke arasında çift yönlü iletişimi destekler (araçtan şebekeye enerji aktarımı - V2G dahil).

ISO 15118, OCPI'den farklı olarak, donanım seviyesinde çalışır: EV ile şarj cihazı arasındaki fiziksel protokolleri yönetir. Öte yandan OCPI, bu fiziksel katmanın üstünde, işletmelerin yazılımsal sistemleri arasında veri alışverişini sağlar. Örneğin, bir ISO 15118 mesajı, şarj cihazından CPO sistemine OCPP aracılığıyla ulaşabilir ve buradan OCPI yoluyla eMSP'ye aktarılabilir. Bu bağlamda ISO 15118, OCPP ve OCPI protokolleriyle birlikte çalışabilir ve EV şarj ekosisteminde farklı görevleri yerine getirirler. OCPI, OCPP ve OSCP'yi tamamlayarak EV ekosistemindeki genel birlikte çalışabilirliği artırırken, ISO 15118 daha çok araç-şarj cihazı iletişimine odaklanmaktadır (Desai vd., 2024).

2.4. Elektrikli Araç Şarj İstasyonları İçin Optimizasyon Kuramı ve Teknikleri

Optimizasyon, tanımlanmış kısıtlar ve hedefler altında mümkün olan alternatifler arasından en iyi çözümün belirlenmesi süreci olarak tanımlanır. EV şarj istasyonlarının operasyonlarının optimize edilmesi, şarj cihazlarının yer seçimi, şarj oturumlarının zamanlaması, enerji yönetimi, fiyatlandırma stratejileri gibi çok boyutlu karar problemlerini kapsayan karmaşık bir süreçtir. EV şarj yönetiminde kaynak tahsisi, maliyet azaltımı ve enerji verimliliğinin sağlanması açısından kritik bir rol oynar (Boyd & Vandenberghe, 2004).

2.4.1. Doğrusal Programlama (LP)

Doğrusal Programlama amaç fonksiyonunun ve kısıtların doğrusal olduğu matematiksel optimizasyon problemlerinde en uygun çözümü bulmak amacıyla kullanılan bir yöntemdir. Standart formu şu şekilde ifade edilir (Boyd & Vandenberghe, 2004):

$$min\ c^T x \tag{2.6}$$

$Ax \leq b,\ x \geq 0$ koşuluna tabidir.

Denklem (2.6)'da x karar değişkenleri vektörünü, c amaç fonksiyon (maliyet veya ödül) katsayılarını temsil eden vektör, A kısıtları temsil eden katsayılar matrisini, b ise kısıt limitlerini gösteren vektörü ifade eder.

EV şarj uygulamalarında doğrusal programlama, toplam şarj maliyetinin minimize edilmesi, şarj cihazlarının verimli zamanlaması ve pik güç talebinin azaltılması gibi problemleri çözmek amacıyla kullanılmaktadır (Hu vd., 2016).

2.4.2. Doğrusal Olmayan Programlama (NLP)

Doğrusal Olmayan Programlama, doğrusal olmayan amaç fonksiyonları veya kısıtlar içeren problemlerdir. Genel formu şu şekildedir (Bazaraa vd., 2013):

$$\min f(x) \tag{2.7}$$

$g_i(x) \leq 0, \quad i = 1, 2, \dots, m$ $\qquad$ $h_j(x) = 0, \quad j = 1, 2, \dots, p$ koşuluna tabidir.

Denklem (2.7)'de $f(x)$ doğrusal olmayan amaç fonksiyonunu, $g_i(x)$ eşitsizlik kısıtlarını, $h_j(x)$ eşitlik kısıtlarını, x karar değişkenleri vektörünü ifade eder.
Doğrusal olmayan programlama, batarya şarj eğrilerinin hassas modellenmesi, doğrusal olmayan fiyatlandırma yapıları, gerilim kararlılığı ve reaktif güç optimizasyonu gibi şebeke istikrarı analizleri veya enerji kayıpları gibi karmaşık etkilerin göz önüne alındığı senaryolarda uygulanmaktadır (Tan vd., 2017).

2.4.3. Karma Tamsayılı Doğrusal Programlama (MILP)

Karma Tamsayılı Doğrusal Programlama, doğrusal programlamayı tamsayılı (özellikle ikili) değişkenlerle birleştiren matematiksel bir optimizasyon yaklaşımıdır. Bu teknik, sürekli ve ayrık kararları içeren optimizasyon problemlerinde geniş ölçüde kullanılmaktadır.
Şarj noktalarının açma/kapama kararları gibi ayrık kararları içeren planlama ve tahsis problemleri için uygundur.

$$min\ \ c^T x + d^T y \tag{2.8}$$

$Ax + By \leq b,\ \ x \geq 0,\ \ y \in \{0,1\}$ koşuluna tabidir.

Denklem (2.8)'de, x sürekli karar değişkenleri vektörünü, y tamsayı (genellikle ikili) karar değişkenleri vektörünü, A ve B kısıt matrislerini, b kısıt sınırları vektörünü, c ve d ise sırasıyla sürekli ve tamsayı değişkenlerle ilişkili maliyet vektörleridir.

Şarj cihazlarının programlanması (açık/kapalı kararları), enerji tahsisi (şarj zaman dilimlerinin ayrık seçimi), EV şarjının akıllı şebeke ve yenilenebilir enerji kaynakları ile entegrasyonu gibi alanlarlarda kullanılmaktadır (Gan vd.., 2013).

2.4.4. Dinamik Programlama (DP)

Dinamik programlama, Richard Bellman tarafından geliştirilen ve çok aşamalı karar verme problemlerini çözmekte yaygın olarak kullanılan güçlü bir optimizasyon yöntemidir. Karmaşık problemleri daha basit alt problemlere bölerek sistematik olarak çözer ve her aşamada en iyi kararları alarak önceden çözülen alt problemlerin sonuçlarını verimli biçimde kullanır (Bellman, 1958).

Dinmik Programlamanın temel prensibi aşağıdaki özyinelemeli Bellman denklemi ile ifade edilir:

$$V_t(s) = \min_{u_t}\{C_t(s,u_t) + V_{t+1}\big(f_t(s,u_t)\big)\} \tag{2.9}$$

Denklem (2.9)'da, $V_t(s)$ t aşamasındaki durum s için optimal maliyet fonksiyonunu, $C_t(s,u_t)$ durum s'de alınan u_t kararının doğrudan maliyetini, $f_t(s,u_t)$ mevcut durumdan bir sonraki duruma geçiş fonksiyonunu, u_t karar değişkenini (kontrol eylemi), s durum değikenini ve t ayrık zaman dilimini ifade eder.

Dinamik programlama EV şarj süreçlerinde, özellikle şarj durumu (SOC) gibi zamana bağlı kısıtlar için etkilidir. Zamana bağlı optimal şarj/deşarj programları, zamanla değişen elektrik tarifeleri altında maliyet minimizasyonu veya değişken yenilenebilir enerji

kaynakları altında çok zaman dilimli şarj planlaması için uygun bir optimizasyon yöntemidir (Wu vd., 2023).

2.4.5. Metasezgisel Optimizasyon Teknikleri

Metasezgisel optimizasyon teknikleri, özellikle karmaşık, büyük ölçekli ve doğrusal olmayan problemlerin çözümünde etkilidir.

Genetik Algoritmalar (GA)

Genetik Algoritmalar, biyolojik evrim süreçlerinden esinlenen algoritmalardır. Seçilim, çaprazlama ve mutasyon işlemleriyle çalışır. İlk olarak John Holland tarafından 1960'larda önerilmiş ve 1975 yılında yayınlanan seminal çalışmasında resmen tanıtılmıştır (Holland, 1975).

Genetik algoritmalar, doğal seçilim ve genetik mekanizmaları taklit ederek, büyük ve karmaşık arama uzaylarında etkili çözümler bulmayı amaçlar. Goldberg, genetik algoritmaların, geleneksel yöntemlerle çözülmesi zor olan problemlerde bile yüksek performans gösterebileceğini belirtir (Goldberg, 1989).

Genetik algoritmalar, doğası gereği sezgisel olduğundan klasik doğrusal ya da doğrusal olmayan programlama yöntemlerindeki gibi kapalı formda matematiksel ifadelere sahip değildir. Ancak temel formülasyonu şu şekildedir:

Amaç Fonksiyonu (Uygunluk Fonksiyonu): Problemin amacı uygunluk fonksiyonunun maksimum veya minimum değerini bulmaktır:

$$\max f(x) \text{ veya } \min f(x) \tag{2.10}$$

Kromozomlar (Çözüm Temsili): Her çözüm adayı (örneğin şarj programı, güç tahsisi veya şarj istasyonu konumu) tipik olarak ikili (binary) ya da reel değerli formatlarda kodlanmış kromozomlar olarak temsil edilir.

Popülasyon: Çözüm adaylarından (kromozomlardan) oluşan grup.

Daha iyi çözümler elde etmek üzere popülasyona iteratif olarak seçim, çaprazlama, mutasyon işlemleri uygulanır.

Seçim:Yüksek uygunluğa sahip çözümlerin yeni nesilleri oluşturmak için seçilmesidir.

Çaprazlama:İki ebeveyn çözümün özelliklerini birleştirerek yeni bireyler oluşturmasıdır.

Mutasyon:Çeşitliliği sağlamak amacıyla çözümlere rastgele değişikliklerin yapılmasıdır.

Temel adımlar:

1. Başlangıç popülasyonunun oluşturulması
2. Uygunluk değerlendirmesi
3. Seçilim, çaprazlama ve mutasyon
4. Yakınsama kriteri sağlanana kadar tekrar

Uygunluk fonksiyonu örneği:

$$Uygunluk(x) = \frac{1}{Maliyet(x) + \alpha . BeklemeZamanı(x)} \tag{2.11}$$

Denklem (2.11)'de, $Uygunluk(x)$ çözüm x'in kalitesini temsil eden uygunluk değerini ifade eder. $Maliyet(x)$ çözüm x ile ilişki maliyettir; genellikle şarj maliyeti, enerji gideri ve operasyonel harcamaları ifade eder. $BeklemeZamanı(x)$ çözüm x için bekleme süresini temsil eder; EV kullanıcılarının uygun şarj noktası için bekleme sürelerini yansıtır. α ise maliyet ve bekleme süresi bileşenleri arasındaki önem dengesini sağlayan ağırlıklandırma katsayısıdır.

Parçacık Sürü Optimizasyonu (PSO)

Parçacık sürü optimizasyonu, kuş sürüsü veya balık topluluğu gibi doğal sistemlerin kolektif hareketini taklit eder. İlk olarak 1995 yılında Kennedy ve Eberhart tarafından önerilen PSO, karmaşık ve çok boyutlu arama uzaylarında etkin biçimde gezerek optimum veya optimuma yakın çözümler üretir. Parçacık sürü optimizasyonu, aday çözümleri (parçacıklar) uygunluk fonksiyonuna göre iteratif olarak iyileştirir. Her parçacık arama uzayında konum ve hız bilgilerini tutar ve bunları bireysel deneyimler ve sürünün ortak bilgisi doğrultusunda günceller. Parçacıklar, hız (v_i) ve konumlarını (x_i) şu denklemlerle güncellerler (Kennedy & Eberhart, 1995).

Konum güncelleme:

$$x_i^{k+1} = x_i^k + v_i^{k+1} \tag{2.12}$$

Hız güncelleme:

$$v_i^{k+1} = \omega . v_i^k + c_1 r_1 \left(pbest_i - x_i^k\right) + c_2 r_2 \left(gbest - x_i^k\right) \tag{2.13}$$

Denklem (2.12) ve (2.13)'de:

v_i^k: Parçacık i'nin k.iterasyondaki hız vektörüdür.

x_i^k: Parçacık i'nin k.iterasyondaki konum vektörüdür.

ω: Atalet katsayısıdır; önceki hızın mevcut hıza etkisini kontrol eder.

c_1, c_2: Bilişsel ve sosyal hızlanma katsayılarıdır; sırasıyla bireysel en iyi çözüm (pbest) ve küresel en iyi çözüm (gbest) üzerindeki etkileri belirler.

r_1, r_2: [0,1]aralığında eş dağılımlı rastgele sayılardır ve algoritmaya stokastik keşif yeteneği kazandırır.

$pbest_i$: Parçacık i'nin kendi geçmişindeki en iyi bulduğu çözümdür.

$gbest$: Tüm parçacıklar ve iterasyonlar boyunca elde edilen küresel en iyi çözümdür.

Benzetilmiş Tavlama (SA)

Benzetilmiş Tavlama, metallerin fiziksel tavlama sürecinden esinlenerek geliştirilmiş, olasılıksal tabanlı bir optimizasyon tekniğidir. İlk olarak Metropolis ve arkaşları tarafından 1953 yılında önerilmiş ve daha sonra Kirkpatrick, Gelatt ve Vecchi tarafından 1983 yılında optimizasyon algoritması olarak sistematik biçimde geliştirilmiştir. Benzetilmiş Tavlama algoritması, erken aşamalarda daha kötü çözümleri belirli bir olasılıkla kabul ederek yerel minimumlardan kaçınmakta ve böylece küresel veya küresele yakın optimum çözümleri bulmayı amaçlamaktadır. Benzetilmiş tavlamanın temel prensibi, algoritma ilerledikçe kötü çözümlerin kabul edilmesidir (Angelini vd., 2023).

Benzetilmiş Tavlama yöntemi, genellikle aşağıdaki optimizasyon problemi formunda kullanılır:

$$\min_{x \in X} f(x) \tag{2.14}$$

Denklem (2.14)'te $f(x)$ minimize edilmesi gereken maliyet veya kayıp türünden amaç fonksiyonunu, $x \in X$ ise uygun çözümler kümesini ifade eder.

Benzetilmiş Tavlama'nın temel özelliği, Metropolis kriterine dayalı kabul olasılığıdır:

$$P(\Delta E, T) = \begin{cases} e^{-\frac{\Delta E}{t}}, & eğer\ \Delta E > 0 \\ 1, & diğer\ durumda \end{cases} \tag{2.15}$$

Denklem (2.15)'te:

$P(\Delta E)$: Daha kötü bir çözümün kabul edilme olasılığıdır; algoritmanın yerel minimumlardan kaçabilmesini sağlar.

$\Delta E = f(x_{yeni}) - f(x_{mevcut})$: Yeni (daha kötü) çözüm ile mevcut çözüm arasındaki amaç fonksiyonu (enerji seviyesi) farkıdır.

T: Sıcaklık parametresidir; bu değer iterasyonlarla birlikte kademeli olarak azaltılır ve daha kötü çözümlerin kabul edilme olasılığını kontrol eder.

Karınca Kolonisi Optimizasyonu (ACO)

Karınca Kolonisi Optimizasyonu, doğadaki gerçek karınca kolonilerinin besin arama davranışlarından esinlenen, doğa temelli bir metasezgisel algoritmadır. İlk olarak Marco Dorigo tarafından 1992 yılında doktora tezinde önerilmiş ve daha sonra çeşitli yayınlarla sistematik hale getirilmiştir (Dorigo vd., 1996). Karınca Kolonisi Optimizasyonu, özellikle yol planlama, çizelgeleme ve tesis yer seçimi gibi kombinatoryal optimizasyon problemlerinde oldukça etkilidir.

Algoritmanın temel ilkesi stigmerjidir. Yani bireyler arasında çevresel izler (feromonlar) aracılığıyla dolaylı iletişim kurulan bu mekanizma sayesinde karıncalar kolektif olarak en kısa yolları keşfeder. Karınca Kolonisi Optimizasyonunda yapay karıncalar, olasılıksal olarak çözümler üretir ve daha iyi çözümleri teşvik etmek amacıyla sanal feromonlar bırakırlar.

Karınca Kolonisi Optimizasyonunun genel yapısı şu temel bileşenlere dayanır:
Çözüm Uzayı (S): Uygun tüm çözüm konfigürasyonları (örneğin, şarj istasyonu konumları veya EV şarj çizelgeleri).
Feromon Yolu (τ_{ij}): i ve j bileşenleri arasındaki feromon yoğunluğu; çözüm kalitesini temsil eder.
Heuristik Bilgi (η_{ij}): Probleme özgü olarak, i konumundan j'ye geçişin ne kadar tercih edilir olduğunu gösterir.

Geçiş Olasılığı:

$$P_{ij}(t) = \frac{\left[\tau_{ij}(t)\right]^{\alpha} . \left[\eta_{ij}\right]^{\beta}}{\sum_{k \in N_i} [\tau_{ik}(t)]^{\alpha} . [\eta_{ik}]^{\beta}} \tag{2.16}$$

Denklem (2.16)'da, α ve β feromon ve sezgisel bilginin etkisini kontrol eden parametreler, η_i ise mevcut konum i'den erişilebilecek bileşenler kümesidir.

Feromon Güncelleme Kuralı:

$$T_{ij}(t+1) = (1-p)\tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}, \qquad p \in (0,1) \tag{2.17}$$

Denklem (2.17)'de, p feromon buharlaşma oranı, $\Delta\tau_{ij}^{k}$ k'ncı karıncanın (i,j) yoluna bıraktığı feromon miktarı, m ise toplam karınca sayısını ifade etmektedir.

2.4.6. Pekiştirmeli Öğrenme Teknikleri

Pekiştirmeli öğrenme, bir sistemin çevresiyle etkileşimi sonucu geri bildirimler aracılığıyla karar alma sürecini öğrendiği bir makine öğrenmesi alt alanıdır. Pekiştirmeli öğrenme, belirsizlik altında sıralı karar alma süreçlerini modelleyen Markov Karar Süreçleri kuramsal çerçevesine dayanır. Açık sistem modellerine ihtiyaç duymadan optimal kontrol stratejileri öğrenebilme yeteneği sayesinde RL, enerji ve ulaşım sistemleri gibi karmaşık alanlarda büyük ilgi görmektedir (Sutton & Barto, 2018).

Temel denklemi Bellman beklenti denklemi olarak bilinir:

$$V(s) = \mathbb{E}[R_{t+1} + \gamma V(s_{t+1}) \mid s_t = s\,] \tag{2.18}$$

Denklem (2.18)'de:

$V(s)$: Durum-değer fonksiyonu; belirli bir durumdan başlayarak izlenecek bir politika sonucunda beklenen toplam ödülü gösterir.

$\mathbb{E}[\cdot]$: Olasılıksal olaylar dikkate alınarak ortalama (beklenen) değeri ifade eder.

R_{t+1}: Mevcut durumdan s_t,bir eylem gerçeleştirdikten sonra ulaşılan s_{t+1} durumunda alınan anlık ödüldür.

γ: Gelecek ödüllerin önemini belirleyen indirim katsayısıdır. ($0 \leq \gamma < 1$)

s_t, s_{t+1}: Sırasıyla mevcut ve bir sonraki durumu ifade eder; ardışık karar süreçlerinde kullanılır.

Q-Öğrenme, Derin Q-Ağları ve Politika Gradyanı gibi pekiştirmeli öğrenme teknikleri, şebeke koşulları, fiyatlandırma ve kullanıcı davranışları gibi değişkenleri dikkate alarak dinamik kararlar üretir. Böylece çok amaçlı optimizasyon hedefleri arasında dengeli çözümler elde edilir (Zhao vd., 2025).

2.5. Elektrikli Araç Şarj İstasyonlarında Optimizasyon Parametreleri

EV şarj istasyonlarında gerek işletme gerekse planlama amaçlı gerçekleştirilen optimizasyon çalışmaları, hem talep tarafındaki (EV kullanıcılarının ihtiyaçları) hem de arz tarafındaki (şarj hizmeti sağlayıcıları, şebeke operatörleri ve istasyon sahipleri) parametreleri dikkate almalıdır. Bu parametreler, optimizasyon probleminin amaç fonksiyonunu ve kısıtlarını tanımlar. Dengeleyici ve bütüncül bir yaklaşım hem kullanıcı memnuniyetini hem de altyapı verimliliğini sağlamayı hedefler (Tan vd., 2017).

2.5.1. Talep Tarafı Parametreleri

Talep tarafı parametreleri, doğrudan EV kullanıcılarının davranışları, beklentileri ve ihtiyaçları ile ilgilidir. Optimizasyon modellerinde bu unsurların dikkate alınması, kullanıcı dostu ve etkili şarj hizmetlerinin sunulabilmesini sağlar. Bu parametreler, EV'lerin zamanında, uygun maliyetle ve mümkün olan en az beklemeyle şarj edilmesine yönelik kullanıcı odaklı çözümlerin oluşturulmasına katkı sunar. Temel parametreler şunlardır:

Şarj Talebi (Enerji İhtiyacı): Her EV, şarj istasyonuna geldiğinde belirli bir batarya doluluk seviyesine (SoC) sahiptir ve hedeflenen seviyeye ulaşmak için belirli miktarda enerji (kWh) gerektirir. Bu, optimizasyonun temel girdisidir. Kullanıcı i için gerekli toplam enerji $E_{ihtiyaç,i}$ şu şekilde tanımlanır:

$$E_{ihtiyaç,i} = C_{bat,i}\, x\, \left(SOC_{hedef,i} - SOC_{mevcut,i}\right) \tag{2.19}$$

Denklem (2.19)'da:

$C_{bat,i}$: EV i'nin batarya kapasitesi (kWh),

$SOC_{hedef,i}$: Hedef şarj seviyesi (genellikle %80-100),
$SOC_{mevcut,i}$: Mevcut şarj seviyesi (%) ifade etmektedir.

Şarj Zaman Aralığı: EV kullanıcılarının genellikle ayrılmak istedikleri bir zaman veya maksimum bekleme toleransları vardır. Bu durum, zaman penceresi olarak modellenir. Kullanıcı memnuniyeti açısından, toplam şarj süresinin ve kuyrukta bekleme süresinin minimize edilmesi kritik bir hedeftir. Aracın şarj için uygun olduğu zaman dilimi $(t_{var,i}, t_{ayr,i})$ aşağıdaki eşitsizlikle ifade edilir:

$$t_{var,i} \leq t_{şarj,i} \leq t_{ayr,i} \tag{2.20}$$

Denklem (2.20)'de:
$t_{var,i}$ (varış zamanı): Kullanıcı i'nin aracını şarj istasyonuna getirdiği zaman. Bu zaman noktasından önce şarj işlemi başlatılamaz.
$t_{şarj,i}$(şarj zamanı): Gerçek şarj işleminin başladığı zaman.
$t_{ayr,i}$ (ayrılış zamanı): Kullanıcı i'nin aracını şarj istasyonundan almayı planladığı zamandır. Bu zamandan sonra şarj işlemi sonlanmış olmalıdır.

Tercih ve Konfor Faktörleri: Kullanıcıların belirli şarj istasyonlarını, şarj başlıklarını veya enerji kaynaklarını (örneğin yenilenebilir) tercih etme gibi yumuşak parametreleri olabilir. Bazı kullanıcılar, daha düşük ücret karşılığında şarjlarının ertelenmesini kabul edebilirken, bazıları hızlı şarjı tercih eder. Bu esneklik, optimizasyon modeline kullanıcı tanımlı kısıtlar veya tercih ağırlıkları biçiminde dahil edilebilir.

2.5.2. Arz Tarafı Parametreleri

Arz tarafı parametreleri, EV şarj hizmeti sağlayıcıları, istasyon işletmecileri ve şebeke operatörleri gibi hizmet sunucularının bakış açısıyla ilgilidir. Bu parametreler, şarj altyapısının verimli işletilmesi, maliyetlerin optimize edilmesi ve teknik sınırlamalara uyulması açısından belirleyicidir.

Dinamik Fiyatlandırma ve Tarife Yapısı (P_t): zaman t'deki enerji fiyatını temsil eder. Genellikle anlık piyasa fiyatlarına veya zaman-of-kullanım tarifelerine dayanır.

$$P_{toplam} = \sum_{t=1}^{T} (E_t . P_t) \tag{2.21}$$

Şebeke Kapasite Kısıtı ($P_{şebeke}^{maks}$): Aşağıdaki şekilde ifade edilen kısıt, şebekeden çekilen maksimum gücü sınırlar.

$$\sum_{i=1}^{N} P_{şarj,i}(t) \leq P_{şebeke}^{maks}, \qquad \forall t \tag{2.23}$$

İstasyon Kullanılabilirliği ($S_{boş,j}$): Şarj cihazı j'nin zaman t'de müsait (1) ya da dolu/kullanılamaz (0) olduğunu belirtir.

$$S_{boş,j}(t) \in \{0,1\}, \qquad \forall j,t \tag{2.24}$$

2.6. Araştırma Boşluğu ve Tezin Motivasyonu

2.6.1. Mevcut Literatürde Belirlenen Boşluklar

EV şarj altyapılarına yönelik optimizasyon teknikleri üzerine araştırma yapılmış olsa da, özellikle OCPI gibi standartlaştırılmış iletişim protokollerinin kapsamlı entegrasyonu ve kullanımı bağlamında önemli boşluklar devam etmektedir. Kapsamlı bir literatür taraması sonucunda aşağıdaki temel araştırma boşlukları tespit edilmiştir:

Standartlaştırılmış Kontrol ve Veri Entegrasyonunun Eksikliği: EV şarj optimizasyonuna dair birçok akademik çalışma, tek bir operatör veya dikey bütünleşmiş sistem varsayımı altında geliştirilmiştir. Ancak gerçek dünyada birden çok CPO ve eMSP mevcuttur ve bu çok aktörlü yapı, birlikte çalışabilirliği ve merkezi olmayan kontrol mekanizmalarını gerekli kılmaktadır. Bu bağlamda, mevcut optimizasyon modellerinin

çoğu, OCPI gibi açık standartları entegre etmeksizin tekil sistemlere odaklanmakta ve bu durum çok ağlı yapılarda uygulanabilirliği kısıtlamaktadır.

OCPI'nin Akademik Optimizasyon Literatürüne Entegrasyonu Yetersizliği: OCPI, endüstride yaygın olarak kullanılan bir ağlar arası geçiş protokolüdür, ancak akademik çalışmalarda daha çok fonksiyonel ve teknik tanımlamalar üzerinden ele alınmakta, bu protokolün sağladığı zengin veri altyapısının operasyonel optimizasyona katkısı yeterince incelenmemektedir. Özellikle OCPI ile sağlanan durum, tarife ve kullanıcı yetkilendirme bilgilerinin, şarj planlaması, yük dengeleme ve fiyat optimizasyonu gibi konularda nasıl kullanılabileceğine dair uygulamalı modeller eksiktir.

Türkiye Özelinde Literatür Eksikliği: Türkiye'de EV şarj altyapısı hızla gelişmektedir. 2021/2022 yıllarında yürürlüğe giren şarj istasyonları lisans düzenlemeleri, OCPI tabanlı sistemlerin teşviki ve istasyon sayısındaki hızlı artış gibi gelişmeler, yeni dinamiklerin dikkate alındığı çalışmalara ihtiyaç olduğunu göstermektedir. Artan yerli üretim (örneğin Togg) gibi faktörler, Türkiye'ye özgü veri ve senaryolarla analiz yapılmasını gerekli kılmaktadır.

Sınırlı Ampirik Doğrulama ve Pratik Uygulama: Mevcut optimizasyon algoritmalarının çoğu, OCPI veya benzeri protokollere dayalı gerçek dünya veri setleri ile test edilmemiştir. Bu da çözüm önerilerinin pratik geçerlilik ve ölçeklenebilirlik açısından sınırlı kalmasına neden olmaktadır (Guillemin vd., 2024).

2.6.2. OCPI Standardının Kullanım Gerekçesi

Bu tez kapsamında OCPI standardının kullanımı, bu iletişim protokolünün sunduğu benzersiz olanaklar ve mevcut literatürdeki eksiklikleri doğrudan hedef alması sebebiyle tercih edilmiştir:

Birlikte Çalışabilirlik ve Standardizasyon: OCPI, farklı şarj platformları ve işletmecileri arasında kesintisiz veri paylaşımı ve sistem entegrasyonu sağlar. Bu sayede,

gelişmiş optimizasyon algoritmalarının uygulanmasında gereken veri erişimi önemli ölçüde kolaylaşır (EVRoaming Foundation, 2023).

Gerçek Zamanlı Veri Yetkinliği: OCPI, fiyatlandırma, konum, müsaitlik ve oturum yönetimi gibi konularda gerçek zamanlı veri paylaşımını destekler. Bu da, enerji fiyatları, şebeke durumu ve kullanıcı taleplerindeki değişimlere dinamik şekilde yanıt verebilen optimizasyon stratejileri için temel teşkil eder (Guillemin vd., 2024).

Ayrıntılı ve Şeffaf Tarife Yapısı: OCPI, enerji bazlı, zaman bazlı ve oturum bazlı olmak üzere detaylı fiyatlandırma bilgileri sağlar. Bu özellik, hem kullanıcılar için şeffaf fiyatlama sunmak, hem de hassas maliyet minimizasyonu için gereklidir (Desai vd., 2024).

Ölçeklenebilirlik ve Genişletilebilirlik: OCPI'nin modüler yapısı, farklı coğrafi ve operasyonel bağlamlara uyarlanabilirliği ve ölçeklenebilirliği destekler. Bu durum, araştırma çıktılarının gelecekteki şarj altyapılarına uygulanabilirliğini artırır (Ampeco, 2025).

Bu gerekçeler doğrultusunda, OCPI standardının merkezde olduğu bir optimizasyon çalışması yapmak hem günümüzün ihtiyaçlarına hem de geleceğin akıllı ve bütünleşmiş şarj altyapılarına uyumlu çözümler geliştirilmesine katkı sunacaktır. Bu çalışma, OCPI'nin sunduğu birlikte çalışabilirlik ve veri zenginliğini, gelişmiş optimizasyon algoritmalarıyla birleştirerek, çok aktörlü şarj ekosistemlerinde verimliliği ve kullanıcı memnuniyetini artırmayı hedeflemektedir.

2.6.3. Önerilen Araştırmanın Katkısı ve Önemi

Bu çalışma hem akademik literatüre hem de pratik EV şarj altyapısının gelişimine önemli katkılar sunmayı hedeflemektedir. OCPI standardı ile ileri düzey optimizasyon tekniklerinin bütünleştirilmesi, EV şarj ağlarının çok aktörlü yapısı içinde daha verimli ve akıllı çalışmasını sağlayacak yeni bir yaklaşım sunar.

OCPI Tabanlı Kapsamlı Bir Optimizasyon Çerçevesi Geliştirilmesi: Mevcut çalışmaların çoğu sadeleştirilmiş veri kümeleri ya da özel iletişim protokolleri kullanırken, bu çalışma OCPI uyumlu veri yapıları ve gerçek zamanlı iletişim kabiliyetlerine dayalı detaylı bir optimizasyon modeli sunacaktır.

Türkiye Odaklı ve Küresel Uygulanabilirlikte İçerik: Bu çalışmada, 2021 sonrası lisanslama yapısı, Togg gibi yerli girişimlerin yarattığı yeni kullanıcı profili gibi Türkiye'ye özgü faktörler dikkate alınacaktır. Ancak geliştirilecek modellerin genel yapısı, benzer gelişmekte olan pazarlar için de kolaylıkla uyarlanabilir nitelikte olacaktır. Bu da araştırmanın hem yerel hem küresel anlamda etki potansiyelini artırmaktadır.

Gerçekçi Simülasyonlarla Ampirik Doğrulama: Geliştirilen algoritmalar ve stratejiler, OCPI verilerine dayalı gerçekçi simülasyon senaryoları veya gerçek veri setleri kullanılarak test edilecektir. Bu sayede, çözüm önerilerinin pratik geçerliliği ve sektörel uygulanabilirliği desteklenecektir.

Bu araştırmanın önemi, şarj istasyonu operasyonlarının iyileştirilmesi, enerji ve altyapı maliyetlerinin azaltılması, şebeke güvenilirliğinin artırılması ve kullanıcı deneyiminin geliştirilmesi gibi çok boyutlu faydaları sayesinde, sürdürülebilir mobilite çözümlerine katkı sağlama potansiyelinde yatmaktadır.

3. MATERYAL ve YÖNTEM

Bu çalışmada, OCPI protokolü kullanılarak EV şarj operasyonlarının incelenmesi ve optimize edilmesine yönelik bir mobil uygulamanın mimarisi ve geliştirme süreci, OCPI uyumlu veri entegrasyonu, çeşitli optimizasyon algoritmalarının uygulanması ve önerilen stratejilerin performansının değerlendirilmesine yönelik genel sistem tasarımı ayrıntılı biçimde açıklanmıştır.

3.1. Materyal

Dart

Bu çalışmanın uygulama geliştirme kısmında kodlama dili olarak Dart kullanılmıştır. Dart, Google tarafından istemci taraflı geliştirme amacıyla tasarlanmış, nesne yönelimli ve sınıf tabanlı bir programlama dilidir. 2011 yılında tanıtılan Dart, özellikle Flutter uygulamaları için birincil programlama dili olarak kullanılmaktadır.

Dart, hem derleme öncesi (ahead-of-time, AOT) hem de çalışma anı (just-in-time, JIT) derleme modlarını desteklemektedir. AOT derleme, yüksek düzeyde optimize edilmiş yerel kod üretimini mümkün kılarak Flutter uygulamalarının mobil ve masaüstü platformlarda yüksek performans göstermesini sağlar. Geliştirme sürecinde ise JIT derleme kullanılarak sıcak yeniden yükleme gibi hızlı yineleme imkânları sunulur.

Sözdizimi bakımından “C” tabanlı programlama dillerinden (örneğin Java, JavaScript) etkilenmiş bir yazılım dilidir.

Mühendislik uygulamaları bağlamında değerlendirildiğinde; Dart’ın performans odaklı yapısı ve eşzamanlılık modeli, gerçek zamanlı veri görselleştirme, sinyal izleme panelleri ve gömülü sistemlerde grafiksel kullanıcı arayüzü (GUI) geliştirme gibi alanlar için uygun bir çözüm sunmaktadır (Google, 2025).

Flutter

Bu çalışmada, dart dili kullanılarak platformlar arası kullanıcı arayüzleri geliştirmeyi mümkün kılan Flutter kullanılmıştır. Flutter, Google tarafından geliştirilen açık kaynaklı bir kullanıcı arayüzü (UI) yazılım geliştirme setidir (SDK). Tek bir kod tabanı kullanarak çoklu platformlara yönelik uygulamaların geliştirilmesini mümkün kılan bu framework, ilk olarak Mayıs 2017'de yayımlanmıştır. Kısa sürede gelişerek, mobil (iOS, Android), web, masaüstü ve gömülü sistemler için yerel olarak derlenen uygulamalar oluşturmak amacıyla kullanılan önde gelen yapılardan biri haline gelmiştir.
Flutter'ın mimarisi, reaktif programlama modeline dayanmaktadır. Bu yapı, geliştiricilerin kullanıcı arayüzlerini widget ağaçlarının bileşimi aracılığıyla esnek ve modüler bir şekilde oluşturabilmesine olanak tanır. Flutter'daki her bileşen bir "widget" olarak tanımlanır; bu bileşenler yapısal öğeleri (örneğin butonlar, menüler), stil öğelerini (örneğin yazı tipleri, renkler) ve düzen modellerini (örneğin satırlar, sütunlar) kapsar.

Mimari açıdan öne çıkan bir diğer unsur, Flutter'ın Skia grafik motorunu kullanmasıdır. Bu motor, kullanıcı arayüzü bileşenlerini GPU hızlandırmasıyla doğrudan işler. Bu tasarım yaklaşımı, yüksek performanslı ve platformlar arası tutarlı görsel çıktılar sunarak yerel OEM widget'larına olan bağımlılığı azaltır ve taşınabilirliği ile performansı artırır.

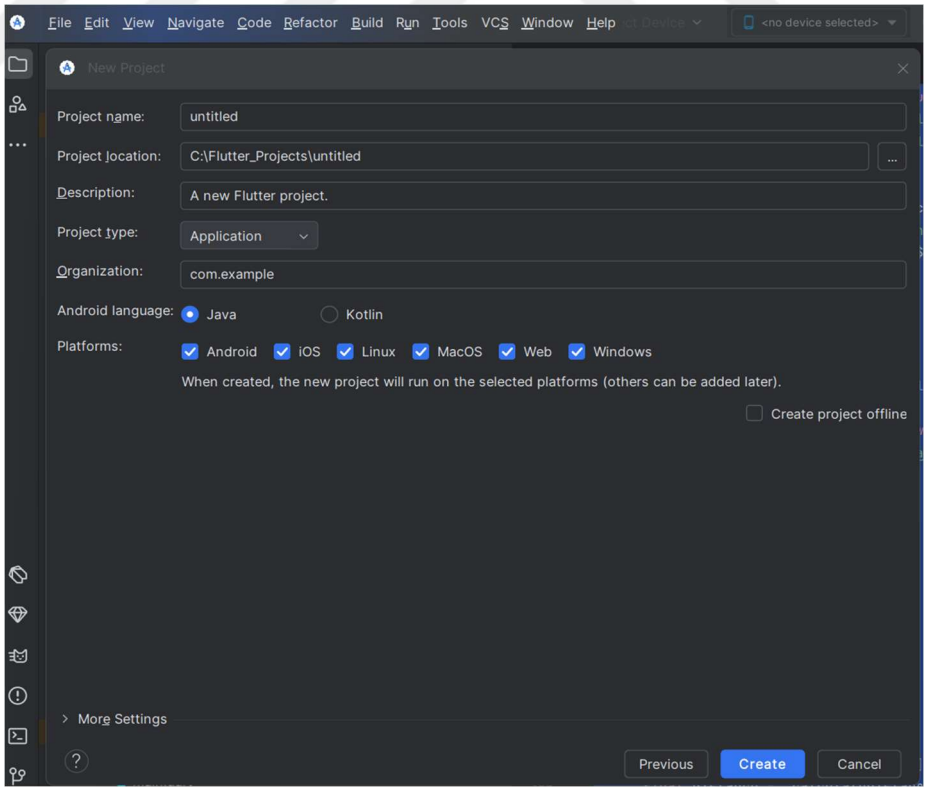
Flutter, geliştirme sürecinde "hot reload" (sıcak yeniden yükleme) özelliğiyle öne çıkar; bu özellik, uygulama üzerinde yapılan değişikliklerin anında görülebilmesini sağlar ve tam bir yeniden başlatma gerektirmeksizin gerçek zamanlı güncellemeye imkân tanıyarak geliştirici verimliliğini artırır. Ayrıca, Android'e özgü Material Design ve iOS'a özgü Cupertino widget setlerini destekleyerek her iki platforma özgü kullanıcı arayüzlerinin oluşturulmasına olanak sunar.

Mühendislik açısından değerlendirildiğinde, Flutter çoklu platform uygulama geliştirme sürecinde maliyetin, zamanın ve kaynak tekrarının azaltılmasında önemli avantajlar sağlar (Google, 2025).

Android Studio

Bu çalışmanın uygulama geliştirme ortamı olarak Android Auto seçilmiştir. 2013 yılında Google tarafından tanıtılan Android Studio, Android uygulama geliştirme için resmi entegre geliştirme ortamıdır. JetBrains'in IntelliJ IDEA platformu üzerine inşa edilen Android Studio, Java, Kotlin ve Flutter ile entegre edildiğinde Dart gibi programlama dillerini destekleyerek yerel Android geliştirme süreçleri için güçlü bir ortam sunmaktadır. Android Studio, sanal Android cihazlar üzerinde gerçek zamanlı test ve dağıtım için entegre emülatör desteği vermektedir. Ayrıca gerçek zamanlı CPU, bellek ve ağ kullanımı izleme için profil oluşturma araçlarını desteklemektedir.

Android Studio, özellikle Elektrik ve Elektronik Mühendisliği alanında; cihaz kontrolü, sensör verilerinin alınması ya da mikrodenetleyici tabanlı sistemlerden telemetri verilerinin görselleştirilmesi gibi işlevleri yerine getiren Android tabanlı uygulamaların geliştirilmesinde önemli bir yere sahiptir (Google, 2025).



Şekil 3.1.1. Android Studio uygulama geliştirme platformu flutter proje oluşturma ekranı

Kullanılan Kütüphaneler

Uygulamada kullanılan kütüphaneler Şekil 3.1.2'de verilmiştir.

```
import 'dart:convert';
import 'dart:io';
import 'dart:math';
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:google_maps_flutter/google_maps_flutter.dart';
import 'package:geolocator/geolocator.dart';
import 'package:path_provider/path_provider.dart';
import 'package:http/http.dart' as http;
```

Şekil 3.1.2. Uygulama geliştirmede kullanılan kütüphaneler

dart:convert: Bu temel Dart kütüphanesi, veri kodlama ve çözümleme (encoding/decoding) işlemleri için kullanılmaktadır. Özellikle; JSON (JavaScript Object Notation) işlemleri (jsonEncode, jsonDecode), UTF-8, ASCII ve Base64 dönüşümleri bunların en önemlileridir.

Veri serileştirme süreçlerinde kritik bir rol oynamaktadır. Bu, özellikle web servisleri, API'ler veya MQTT/HTTP üzerinden haberleşen gömülü cihazlarla etkileşimde önemlidir. Mühendislik bağlamında, sensör verilerinin, yapılandırma parametrelerinin veya günlük kayıtlarının (log) etkin biçimde saklanması ve iletilmesini sağlar (Dart Team, 2025).
Bu çalışmada OCPI formatındaki veriyi listeye dönüştürmek için kullanılmaktadır.

dart:io: Bu kütüphane, dosya okuma/yazma işlemleri (File, Directory), ağ iletişimi (Socket, HttpClient) gibi düşük seviyeli giriş/çıkış işlemlerinin yapılmasını sağlar.

Ana işletim sistemiyle doğrudan etkileşim sağladığı için yerel dosya sistemine veri okuma/yazma işlemleri, sistem tanılama kayıtlarının tutulması için oldukça kritiktir:

Bu yönüyle dart:io, IoT sistemlerinde, veri kaydedicilerde ve geçit uygulamalarında mühendislik projeleri kapsamında yaygın olarak kullanılmaktadır.

Bu çalışmada, CSV dosyasını cihazda ilgili konuma yazmak için kullanılmaktadır.

dart:math: Bu çekirdek Dart kütüphanesi, trigonometrik fonksiyonlar (sin, cos, tan), matematiksel işlemler (sqrt, pow, log), rastgale sayı üretimi gibi matematiksel sabitleri ve fonksiyonları sağlamaktadır.
Bu kütüphane, mühendislik uygulamalarında algoritmik hesaplamalar, sinyal işleme, veri analizi ve simülasyon işlemleri için sıkça kullanılmaktadır (Dart Team, 2025).

Bu çalışmada, mesafe hesaplama ve optimizasyon işlemleri için kullanılmaktadır.

material.dart: Google'ın kullanıcı arayüzleri için görsel tasarım dili olan Material Design bileşenlerine erişim sağlayan birincil Flutter paketi material.dart'tır. Scaffold, AppBar, TextField, ElevatedButton gibi çok çeşitli ön tanımlı bileşenler ile Row, Column, Container gibi yerleşim yapıları içermekte olup, geliştiricilerin çoklu platformlarda estetik açıdan tutarlı ve duyarlı kullanıcı arayüzleri oluşturmasına imkan tanır (Google, 2025).

services.dart (rootBundle): rootBundle, Flutter'ın hizmetler (services) kütüphanesinin bir parçası olup, uygulama ile birlikte paketlenmiş varlıklara (assets) erişim sağlar. Bu varlıklar, metin dosyaları (örneğin .csv, .json), yapılandırma dosyaları, görsel veri kaynaklarını içerebilir.

Geliştiriciler, rootBundle.loadString() fonksiyonunu kullanarak bu dosyaları çalışma zamanında okuyabilir. Bu özellik; sistem parametrelerinin başlatılması, çevrimdışı veri kümelerinin yüklenmesi veya gömülü sistemlerde kalibrasyon verilerinin entegrasyonu gibi durumlarda oldukça yararlıdır (Google, 2025).

google_maps_flutter.dart: Bu kütüphane, Flutter ekibi tarafından resmi olarak sağlanan bir eklentidir ve Flutter tabanlı uygulamalarda Google Haritalar'ın yerleştirilmesi ve

kontrol edilmesini mümkün kılar. Bu eklenti, Android ve iOS platformları için yerel harita bileşenleri sunar; haritalar, platforma özgü Google Haritalar SDK'ları aracılığıyla işlenirken, kullanıcı etkileşimi Dart programlama dili üzerinden gerçekleştirilir.

Harita İşleme: Normal, uydu, hibrit ve arazi olmak üzere özelleştirilebilir harita türlerini destekleyen, etkileşimli Google Haritalar'ın gösterimini sağlar.

İşaretçi Yönetimi: Harita üzerinde ilgi noktaları, takip edilen varlıklar veya sensör konumlarını temsil eden özel işaretçilerin eklenmesi, kaldırılması ve güncellenmesine olanak tanır.

Kamera Kontrolü: Kamera konumu, yakınlaştırma düzeyi, eğim ve yön gibi parametrelerin programatik olarak kontrolünü mümkün kılar; bu özellik, dinamik harita gezintisi ve görselleştirme açısından önemlidir.

Kullanıcı Konumu Entegrasyonu: geolocator gibi paketlerle entegre edilerek, kullanıcının mevcut konumunun gerçek zamanlı olarak izlenmesi ve harita üzerinde gösterilmesi sağlanabilir.

Dokunmatik Hareket Desteği: Kaydırma, yakınlaştırma ve döndürme gibi dokunmatik hareketlerle kullanıcı etkileşimini destekler (Google, 2025).

Bu çalışmada kullanıcı ve şarj istasyonları konumlarının harita üzerinde gösterilmesi ve işaretlenmesi için kullanılmaktadır.

geolocator.dart: Bu kütüphane, konum belirleme işlevleri için yüksek seviyeli soyutlamalar sunmaktadır.

Mevcut konumun alınması (getCurrentPosition), konum değişikliklerinin dinlenmesi (getPositionStream), konum izni yönetimi gibi işlevleri içerir.
Platform kanalları üzerinden yerel konum servislerine köprü işlevi görür ve varlık takibi, navigasyon sistemleri veya endüstriyel ve Nesnelerin İnterneti (IoT) bağlamında coğrafi

sınırlamalı kontrol sistemleri gibi konum bilgisi gerektiren uygulamaların geliştirilmesinde kritik rol oynar (Baseflow, 2025).

Elektrik ve Elektronik Mühendisliğinde, GPS tabanlı izleme sistemleri, mobil robotik uygulamaları ve bağlamsal farkındalık içeren otomasyon sistemlerinde yaygın olarak kullanılmaktadır.

Bu çalışmada, kullanıcının mevcut konumunun okunmasını ve kullanıcın şarj istasyonuna olan mesafenin hesaplanması için kullanılmaktadır.

path_provider.dart: Bu kütüphane, cihaz dosya sistemine platformdan bağımsız olarak erişim sağlayan bir arabirim sunmaktadır. Geçici dizin (getTemporaryDirectory), uygulama belgeleri dizini (getApplicationDocumentsDirectory), harici depolama alanı (destekleniyorsa) dizinlerine erişim sağlar.
Gömülü/mobil mühendislik sistemlerinde, yerel veri kaydı, çevrimdışı mod desteği veya ürün yazılımı güncellemeleri gibi işlemler için kritik öneme sahiptir (Flutter Community, 2025).

Bu çalışmada, geliştirilen uygulamanın cihaz belleğine veri yazması için kullanılmaktadır.

http.dart: Bu kütüphane, ağ üzerinden (internetten) veri almak ya da veri göndermek için kullanılır. Genellikle REST API'lerine istek yapmak için kullanılır. Örneğin: kullanıcı bilgileri çekme, konum verisi gönderme, hava durumu API'sinden veri alma vb. işlemler için kullanılabilir.

Bu çalışmada, Google Directions API'den rota ve mesafe verisi çekmek için kullanılmaktadır.

3.2. Yöntem

3.2.1. Sistem Mimarisi

Önerilen sistem mimarisi, standartlaştırılmış OCPI veri alışveriş mekanizmalarını kullanarak gerçek zamanlı EV şarj senaryolarını simüle etmek ve optimize etmek üzere tasarlanmıştır. Bu mimari aşağıdaki temel bileşenlerden oluşmaktadır:

Mobil Uygulama Katmanı: Flutter çatısı ve Dart programlama dili kullanılarak geliştirilmiş olup, çapraz platform desteği ve kullanıcı etkileşimi sağlamaktadır. Uygulama, sürücülerin:

- İstasyonları harita üzerinden görmesini,
- İstasyon ayrıntılarını (güç, fiyat) incelemesini,
- Şarj gereksinimlerini (SoC) girmesini,
- Optimum istasyon önerilerini almasını sağlar.

Flutter tercih edilmesinin nedeni, tek kod tabanıyla hem Android hem de iOS üzerinde yüksek performanslı ve zengin kullanıcı arayüzüne sahip bir uygulama geliştirme imkânı sunmasıdır. Bu katman, kullanıcının konum bilgisini (GPS üzerinden) ve araç bilgilerini toplayarak sistemle etkileşim sağlar.

Veri Entegrasyon Katmanı: Bu katman, sistemin harici veri kaynaklarıyla olan iletişimini yönetir. Özellikle OCPI ile uyumlu arayüzler aracılığıyla şarj istasyonu verilerini alır. Prototip sistemde bu veriler, OCPI 2.2.1 sürümüne uygun biçimde JSON veri dosyası veya bir API olarak yapılandırılmıştır. Alınan bilgiler şunları içerir:

- İstasyon kimlikleri,
- Coğrafi konum (enlem/boylam),
- Konnektör güç seviyeleri,
- Fiyatlandırma ve tarife bilgileri.

Optimizasyon Motoru: Dört farklı algoritmik model [Doğrusal Programlama(LP), Benzetilmiş Tavlama(SA), Parçacık Sürü Optimizasyon(PSO) ve Pekiştirmeli

Öğrenme(RL)] içeren çok kriterli istasyon sıralama mekanizması; mesafe, şarj hızı (güç), fiyat ve kullanıcının mevcut SoC bilgisi gibi kriterleri göz önünde bulundurarak istasyonları puanlar ve sıralama yapar.

Çıktı ve Dışa Aktarım Arayüzü: Optimum istasyon sıralamaları ve skorlarını içeren sonuçları CSV formatında dışa aktarır ve analiz için kullanılabilir hale getirmektedir. Bu yapı, gerçek zamanlı veri ediniminden algoritmik karar alma sürecine ve sonuçların dağıtımına kadar uçtan uca bir etkileşim sunmaktadır.

3.2.2. Mobil Uygulama Geliştirme

Uygulamada hayata geçirilen başlıca işlevler şunlardır:

Kullanıcı Konumunun Alınması:

Cihazın GPS sistemi kullanılarak kullanıcının mevcut konumu belirlenir; bu, yakınlık tabanlı istasyon seçimi için temel gerekliliktir.

```
Future<void> _kullaniciKonumunuAlma() async {
  bool servisEtkin = await Geolocator.isLocationServiceEnabled();
  if (!servisEtkin) {
    return;
  }
  LocationPermission izin = await Geolocator.checkPermission();
  if (izin == LocationPermission.denied) {
    izin = await Geolocator.requestPermission();
    if (izin == LocationPermission.denied) {
      return;
    }
  }
  if (izin == LocationPermission.deniedForever) {
    return;
  }
  Position pozisyon = await Geolocator.getCurrentPosition(
    locationSettings: const LocationSettings(
      accuracy: LocationAccuracy.high,
    ),
  );

  setState(() {
    _kullaniciKonumu = LatLng(pozisyon.latitude, pozisyon.longitude);
  });
}
```

Şekil 3.2.2.1. Kullanıcı konumunu alma fonksiyonu

OCPI Veri Edinimi:

Uygulama ile EV hizmet sağlayıcıları arasındaki birlikte çalışabilirliği sağlamak üzere OCPI 2.2.1 sürümü iletişim protokolü olarak benimsenmiştir. Şarj istasyonu verilerini almak amacıyla OCPI ile uyumlu bir JSON dosyasına bağlanır. Elde edilen bilgiler şunları içerir:

- İstasyonların tanımlayıcı bilgileri
- İstasyonların coğrafi koordinatları
- İstasyonların güç ve tarife fiyatlandırma bilgileri

```dart
Future<void> _OCPIIstasyonlariniAssetstenYukleme() async {
  final jsonVerisi = await rootBundle.loadString('assets/istasyon_konumlari.json');
  final veriler = jsonDecode(jsonVerisi);
  setState(() {
    _istasyonlar = List<Map<String, dynamic>>.from(veriler);
  });
}
```

Şekil 3.2.2.2. İstasyon bilgilerini Assets'ten yükleme fonksiyonu

dart × {} istasyon_konumlari.json ×


```json
[
  {
    "kimlik": "TR-001",
    "isim": "A-İstasyon",
    "konum": {"enlem": 40.216793, "boylam": 28.899339},
    "ozellikler": [{"guc": 22.0, "tarife": 8.5}]
  },
  {
    "kimlik": "TR-002",
    "isim": "B-İstasyon",
    "konum": {"enlem": 40.219250, "boylam": 28.899833},
    "ozellikler": [{"guc": 22.0, "tarife": 7.0}]
  },
  {
    "kimlik": "TR-003",
    "isim": "C-İstasyon",
    "konum": {"enlem": 40.219889, "boylam": 28.901120},
    "ozellikler": [{"guc": 22.0, "tarife": 7.5}]
  },
  {
    "kimlik": "TR-004",
    "isim": "D-İstasyon",
    "konum": {"enlem": 40.220545, "boylam": 28.902279},
```


Şekil 3.2.2.3. İstasyon bilgilerinin bulunduğu JSON dosyası

Coğrafi Mesafe Hesaplama:

```
Future<double?> _yolMesafesiniGetir(LatLng baslangic, Map<String, dynamic> hedefKonum) async {
  const apiKey = 'AIzaSyDJ1uEkmwSk0CR87ZCnRYffBlKzDTuRGRc'; // API key
  final url = 'https://maps.googleapis.com/maps/api/directions/json'
      '?origin=${baslangic.latitude},${baslangic.longitude}'
      '&destination=${hedefKonum['konum']['enlem']},${hedefKonum['konum']['boylam']}'
      '&mode=driving'
      '&key=$apiKey';
  final response = await http.get(Uri.parse(url));

  if (response.statusCode == 200) {
    final data = jsonDecode(response.body);
    if (data['routes'] != null && data['routes'].isNotEmpty) {
      final distanceInMeters = data['routes'][0]['legs'][0]['distance']['value'];
      return distanceInMeters / 1000; // km
    } else {
      print('Rota bulunamadı');
    }""
  } else {
    print('Google Directions API Hata Kodu: ${response.statusCode}');
  }
  return null;
}
```

Şekil 3.2.2.4. Coğrafi mesafe hesaplama fonksiyonu

Ana Sayfa Arayüzü:

Uygulama ana sayfası aşağıdaki alanlardan oluşmaktadır:

- Harita ve İşaretçiler

```
Set<Marker> _isaretleyicileriOlustur() {
  Set<Marker> isaretleyiciler = {};
  if (_kullaniciKonumu != null) {
    isaretleyiciler.add(
      Marker(
        markerId: MarkerId("kullanici"),
        position: _kullaniciKonumu!,
        infoWindow: InfoWindow(title: "Buradasınız"),
        icon: BitmapDescriptor.defaultMarkerWithHue(BitmapDescriptor.hueAzure),
      ),
    );
  }
  for (int i = 0; i < _istasyonlar.length; i++) {
    isaretleyiciler.add(
      Marker(
        markerId: MarkerId(_istasyonlar[i]['kimlik']),
        position: LatLng(_istasyonlar[i]['konum']['enlem'],
            _istasyonlar[i]['konum']['boylam']),
        infoWindow: InfoWindow(title: '${i + 1}. ${_istasyonlar[i]['isim']}'),
      ),
    );
  }
  return isaretleyiciler;
}
```

Şekil 3.2.2.5. Kullanıcı konumu ve istasyon işaretleyicileri fonksiyonu

- İstasyon Listesi

```
Expanded(
  flex: 2,
  child: Row(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: [
      Expanded(
        child: ListView.builder(
          itemCount: (_istasyonlar.length / 2).ceil(),
          itemBuilder: (context, index) {
            var istasyon = _istasyonlar[index];

            final mesafe = istasyon['uzaklik'] as double? ?? 0.0;
            return ListTile(
              title: Text(istasyon['isim']),
              subtitle: Column(
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
                  Text('Güç: ${istasyon['ozellikler'][0]['guc']} kW | Fiyat: ₺${istasyon['ozellikler'][0]['tarife']}'),
                  SizedBox(height: 4),
                  Text('Uzaklık: ${mesafe.toStringAsFixed(2)} km'),
                ],
              ),
            );
          },
        )
```

Şekil 3.2.2.6. İstasyon listesi fonksiyonu

- Mevcut Şarj Seviyesi(%) giriş alanı

```
Padding(
  padding: const EdgeInsets.all(8.0),
  child: TextFormField(
    controller: _socDenetleyicisi,
    keyboardType:
    TextInputType.numberWithOptions(decimal: true),
    inputFormatters: [
      FilteringTextInputFormatter.allow(
          RegExp(r'^(100(\.0)?|\d{1,2}(\.\d)?)')),
    ],
    decoration: InputDecoration(
      labelText: 'Mevcut Şarj Seviyesi (%)',
      border: OutlineInputBorder(),
    ),
  ),
),
```

Şekil 3.2.2.7. Mevcut şarj seviyesi giriş alanı fonksiyonu

- Hesapla butonu

```
Padding(
  padding: const EdgeInsets.only(bottom: 12.0),
  child: ElevatedButton(
    onPressed: () {
      double soc = double.tryParse(_socDenetleyicisi.text) ?? 0.0;
      Navigator.push(
        context,
        MaterialPageRoute(
          builder: (context) => SonuclarSayfasi(
              kullaniciKonumu: _kullaniciKonumu!,
              istasyonlar: _istasyonlar,
              soc: soc,
          ),
        ),
      );
    },
    child: Text('HESAPLA'),
  ),
```

Şekil 3.2.2.8. Hesapla butonu ve sonuçlar sayfasına geçiş fonksiyonu

Sonuçlar Sayfası:

```
child: SingleChildScrollView(
  scrollDirection: Axis.horizontal,
  child: DataTable(
    columnSpacing: 16.0,
    dataRowMinHeight: 16.0,
    dataRowMaxHeight: 24.0,
    headingRowHeight: 28.0,
    columns: [
      DataColumn(label: Text("Tercih", style: TextStyle(fontSize: 12))),
      DataColumn(label: Text("LP", style: TextStyle(fontSize: 12))),
      DataColumn(label: Text("LP Skor", style: TextStyle(fontSize: 12))),
      DataColumn(label: Text("SA", style: TextStyle(fontSize: 12))),
      DataColumn(label: Text("SA Skor", style: TextStyle(fontSize: 12))),
      DataColumn(label: Text("PSO", style: TextStyle(fontSize: 12))),
      DataColumn(label: Text("PSO Skor", style: TextStyle(fontSize: 12))),
      DataColumn(label: Text("RL", style: TextStyle(fontSize: 12))),
      DataColumn(label: Text("RL Skor", style: TextStyle(fontSize: 12))),
    ],
    rows: List.generate(widget.istasyonlar.length, (index) {
      return DataRow(cells: [
        DataCell(Text((index + 1).toString())),
        DataCell(Text(lpSiralama[index]['isim'].toString().substring(0, 1))),
        DataCell(Text(lpSiralama[index]['skor'].toStringAsFixed(4))),
        DataCell(Text(saSiralama[index]['isim'].toString().substring(0, 1))),
        DataCell(Text(saSiralama[index]['skor'].toStringAsFixed(4))),
        DataCell(Text(psoSiralama[index]['isim'].toString().substring(0, 1))),
        DataCell(Text(psoSiralama[index]['skor'].toStringAsFixed(4))),
        DataCell(Text(rlSiralama[index]['isim'].toString().substring(0, 1))),
        DataCell(Text(rlSiralama[index]['skor'].toStringAsFixed(4))),
      ]);
    }),
```

Şekil 3.2.2.9. Sonuçlar sayfası fonksiyonu

CSV Dışa Aktarım Özelliği:

Sıralı liste ve ilgili skorların cihazın "Belgeler" dizinine CSV formatında kaydeden yerel dosya sistemi uygulanmıştır.

```
Align(
  alignment: Alignment.bottomRight,
  child: Padding(
    padding: const EdgeInsets.all(16.0),
    child: ElevatedButton(
      onPressed: _csvyeAktar,
      child: Text("CSV'ye aktar"),
    ),
  ),
)
```

Şekil 3.2.2.10. CSV'ye aktar butonu ve CSV'ye dönüştürme fonksiyonu

```
class ListeyiCsvyeDonustur {
  const ListeyiCsvyeDonustur();

  String donustur(List<List<dynamic>> satirlar, {String alanAyraci = ',', String metinAyraci = '"'}) {
    return satirlar.map((satir) {
      return satir.map((alan) {
        return '$metinAyraci$alan$metinAyraci';
      }).join(alanAyraci);
    }).join("\n");
  }
}
```

Şekil 3.2.2.11. CSV verisini oluşturma ve kaydetme fonksiyonu

3.2.3. Optimizasyon Algoritmaları

Dinamik koşullar altında EV şarj istasyonlarının sıralanması ve önerilmesi amacıyla beş farklı optimizasyon yöntemi uygulanmış ve değerlendirilmiştir. Her bir yöntem için değerler normalize edilmiştir.

```
class _SonuclarSayfasiDurum extends State<SonuclarSayfasi> {
  late double Gmaks, Fmaks, Umaks;
```

Şekil 3.2.3.1. Normalizasyon için kullanılacak değerlerin tanımlandığı fonksiyon

Doğrusal Programlama (LP)

Bu çalışmada, çok kriterli karar verme problemi olarak modellenen şarj istasyonu seçimi, normalize edilmiş kriterler ve ağırlıklı bir fayda fonksiyonu üzerinden aşağıdaki gibi hesaplanmıştır:

$$Gnorm = \frac{G}{G_{maks}}, Fnorm = \frac{F}{F_{maks}}, Unorm = \frac{U}{U_{maks}}, SoC_{norm} = \frac{SoC}{100} \quad (3.1)$$

$$LP_{Skor} = a_g . G_{norm} + a_f . (1 - F_{norm}) + a_u . (1 - U_{norm}) + a_{soc} . SoC_{norm} \quad (3.2)$$

Denklem (3.1) ve (3.2)'de kullanıcı için en uygun istasyonu seçerken aşağıdaki kriterleri maksimize veya minimize etmek amaçlanmıştır:

- Güç (G): Ne kadar yüksekse o kadar iyi (maksimize).
- Şarj durumu (SoC): Kullanıcının mevcut şarj seviyesi (% olarak, ortak katsayı).
- Fiyat (F): Ne kadar düşükse o kadar iyi (minimize).
- Uzaklık (U): Ne kadar yakınsa o kadar iyi (minimize).

a_g, a_{soc}, a_f, a_u her bir kriterin ağırlığını ifade etmektedir. Bu ağırlıkların toplamı $a_g + a_{soc} + a_f + a_u = 1$ olacak şekilde normalize edilmiştir (Keeney & Raiffa, 1993) (Triantaphyllou, 2000).

Faydalı kriterlerin doğrudan, olumsuz kriterlerin ise ters normalize edilerek katkı sağladığı çok kriterli bir fayda fonksiyonu kullanılması amaçlanmıştır.

```
static const double lp_aGuc = 0.4;
static const double lp_aSoc   = 0.3;
static const double lp_aFiyat = 0.2;
static const double lp_aUzaklik  = 0.1;
```

Şekil 3.2.3.2. LP optimizasyonu için karar ağırlıklarının tanımlandığı fonksiyon

```
double LPSkorHesaplama(Map<String, dynamic> istasyon) {
  // LP_Adım_1
  double guc     = (istasyon['ozellikler'][0]['guc'] as num).toDouble();
  double fiyat   = (istasyon['ozellikler'][0]['tarife'] as num).toDouble();
  double uzaklik = istasyon['uzaklik'] as double? ?? 0.0;
  // LP_Adım_2
  double maksMesafe = widget.soc / km_basina_soc;
  if (uzaklik > maksMesafe) return 0.0;
  // LP_Adım_3
  double normGuc     = (Gmaks > 0) ? guc     / Gmaks : 0.0;
  double normFiyat   = (Fmaks > 0) ? fiyat   / Fmaks : 0.0;
  double normUzaklik = (Umaks > 0) ? uzaklik / Umaks : 0.0;
  double normSoc     = widget.soc / 100.0;
  // LP_Adım_4
  double dinamikGuc     = normSoc * lp_aGuc;
  double dinamikFiyat   = (1 - normSoc) * lp_aFiyat;
  double dinamikUzaklik = (1 - normSoc) * lp_aUzaklik;
  double dinamikSoc     = (1 - normSoc) * lp_aSoc;
  // LP_Adım_5
  double dinamikToplam    = dinamikGuc + dinamikFiyat + dinamikUzaklik + dinamikSoc;
  double agirlikliGuc     = dinamikGuc     / dinamikToplam;
  double agirlikliFiyat   = dinamikFiyat   / dinamikToplam;
  double agirlikliUzaklik = dinamikUzaklik / dinamikToplam;
  double agirlikliSoc     = dinamikSoc     / dinamikToplam;
  // LP_Adım_6
  double lp_skor = agirlikliGuc * normGuc
                 + agirlikliFiyat * (1 - normFiyat)
                 + agirlikliUzaklik * (1 - normUzaklik)
                 + agirlikliSoc * normSoc;
  // LP_Adım_7
  return lp_skor.clamp(0.0, 1.0);
}
```

Şekil 3.2.3.3. LP optimizasyonu için hesaplama fonksiyonu

LPSkorHesaplama fonksiyonu; Adım_1'de istasyon verilerinden güç, fiyat ve uzaklık değişkenleri alıp, normalize etmek için hazırlık (sayısal biçime dönüştürme) yapar. Adım_2'de kullanıcının mevcut SoC düzeyine göre menzil kısıtı belirlenir ve bu menzil dışında kalan istasyonlara sıfır skor atanarak erişilemeyen istasyonlar elenir. Adım_3'te güç, fiyat ve uzaklık değerleri, kendi maksimum değerlerine göre denklem (3.1) doğrultusunda normalize edilir. Bu sayede farklı birimlerdeki veriler aynı ölçekte değerlendirilmiş olur. Adım_4'te normalize edilmiş SoC değeri dinamik ağırlık hesaplamasında kullanılır. Düşük SoC seviyelerinde fiyat ve uzaklık gibi faktörlere daha fazla ağırlık verilirken, yüksek SoC seviyelerinde güç parametresi öncelik kazanır.

Böylece bağlama duyarlı ağırlıklar hesaplanır ve karar kriterlerine adaptif önem düzeyleri atanır. Adım_5'te, hesaplanan ağırlıkların toplamı 1'e normalize edilir ve ağırlıklı fayda fonksiyonu oluşturulur. Bu fonksiyonda güç ve SoC kriterleri doğrudan katkı sağlarken, fiyat ve uzaklık kriterleri ters normalize edilerek (1 - normalizasyon) dezavantajlı etkileri modele dahil edilir. Adım_6'da, tüm bu ağırlıklı kriterler birleştirilir ve denklem (3.2)'ye göre LP tabanlı istasyon skoru hesaplanır. Böylece, her istasyon için bağlamsal olarak değerlendirilmiş bir uygunluk skoru elde edilir. Son olarak Adım_7'de hesaplanan skorlar 0 ile 1 aralığında sınırlandırılır ve tüm skorların geçerli bir aralıkta kalması sağlanır.

Benzetilmiş Tavlama (SA)

Bu çalışmada SA'nın kabul olasılığı modeli şu şekilde skorlaştırılmıştır:

$$E = (1 - G_{norm}) + F_{norm} + U_{norm} + (1 - SoC_{norm}) \tag{3.3}$$

$$SA_{skor} = e^{(-\frac{E}{T})} \tag{3.4}$$

Burada:

E: enerji değeri, sistemin kalitesizliğini temsil eder.
T: sıcaklık katsayısı, bu çalışmada sabit olarak 0,95 alınmıştır.

```
static const double sa_aGuc = 0.25;
static const double sa_aSoc   = 0.25;
static const double sa_aFiyat = 0.25;
static const double sa_aUzaklik  = 0.25;
static const double sa_sicaklik = 0.95;
```

Şekil 3.2.3.4. SA optimizasyonu için sıcaklık sabitinin tanımlandığı fonksiyon

```
double SASkorHesaplama(Map<String, dynamic> istasyon) {
  // SA_Adım_1
  double guc     = (istasyon['ozellikler'][0]['guc'] as num).toDouble();
  double fiyat   = (istasyon['ozellikler'][0]['tarife'] as num).toDouble();
  double uzaklik = istasyon['uzaklik'] as double? ?? 0.0;
  // SA_Adım_2
  double maksMesafe = widget.soc / km_basina_soc;
  if (uzaklik > maksMesafe) return 0.0;
  // SA_Adım_3
  double normGuc     = (Gmaks > 0) ? guc     / Gmaks : 0.0;
  double normFiyat   = (Fmaks > 0) ? fiyat   / Fmaks : 0.0;
  double normUzaklik = (Umaks > 0) ? uzaklik / Umaks : 0.0;
  double normSoc     = widget.soc / 100.0;
  // SA_Adım_4
  double dinamikGuc     = normSoc * sa_aGuc;
  double dinamikFiyat   = normSoc * sa_aFiyat;
  double dinamikUzaklik = (1 - normSoc) * sa_aUzaklik;
  double dinamikSoc     = (1 - normSoc) * sa_aSoc;
  // SA_Adım_5
  double dinamikToplam    = dinamikGuc + dinamikFiyat + dinamikUzaklik + dinamikSoc;
  double agirlikliGuc     = dinamikGuc     / dinamikToplam;
  double agirlikliFiyat   = dinamikFiyat   / dinamikToplam;
  double agirlikliUzaklik = dinamikUzaklik / dinamikToplam;
  double agirlikliSoc     = dinamikSoc     / dinamikToplam;
  // SA_Adım_6
  double enerji = ((1- normGuc) * agirlikliGuc)
                + (normFiyat * agirlikliFiyat)
                + (normUzaklik * agirlikliUzaklik)
                + ((1- normSoc) * agirlikliSoc);
  // SA_Adım_7
  double kabulOlasiligi = exp(-enerji / sa_sicaklik);
  // SA_Adım_8
  return kabulOlasiligi.clamp(0.0, 1.0);
}
```

Şekil 3.2.3.5. SA optimizasyonu için hesaplama fonksiyonu

SASkorHesaplama fonksiyonu; ilk 3 adımda LPSkorHesaplama fonksiyonu ile aynı işlemleri yapar. Adım_4'te normalize edilmiş SoC değerine bağlı olarak kriter ağırlıkları dinamik şekilde ayarlanır. Bu bağlamda, yüksek SoC durumunda güç ve fiyat gibi performans parametreleri öne çıkarken, düşük SoC düzeylerinde uzaklık ve aciliyet kriterleri daha fazla önem kazanır. Bu durum, algoritmaya adaptif bir karar verme süreci sağlar. Adım_5'te dinamik ağırlıkların toplamı normalize edilerek her bir kriterin göreli etkisi belirlenir. Bu ağırlıklar, SA algoritmasındaki enerji fonksiyonunun bileşenlerine

yapılan ağırlıklı katkıyı ifade eder. Adım_6'da SA'nın temelinde yer alan enerji fonksiyonu denklem (3.3)'e göre hesaplanır. Adım_7'de hesaplanan enerji değeri SA algoritmasında kabul olasılığına dönüştürülür. Bu işlem, denklem (3.4)'te görüleceği üzere, Metropolis kriteriyle tanımlanan üstel bir fonksiyon aracılığıyla gerçekleştirilir. Son olarak Adım_8'de hesaplanan skorlar 0 ile 1 aralığında sınırlandırılır ve tüm skorların geçerli bir aralıkta kalması sağlanır.

Parçacık Sürü Optimizasyonu (PSO)

Bu çalışmada her istasyon, bir parçacık gibi ele alınmış ve üç temel PSO bileşeniyle skor üretilmesi amaçlanmıştır.

$$Atalet = 0.5 \,.\, G_{norm} + \; 0.5 \,.\, SoC_{norm} \tag{3.5}$$

$$Bilişsel = G_{norm}^{2} \tag{3.6}$$

$$Sosyal = (1 - F_{norm}) + (1 - U_{norm}) \tag{3.7}$$

$$PSO_{skor} = a_{Atalet} \,.\, Atalet + \; a_{Bilissel} \,.\, Bilissel + \; a_{Sosyal} \,.\, \left(\frac{Sosyal}{2}\right) \tag{3.8}$$

```
static const double pso_aAtalet = 0.3;
static const double pso_aBilissel = 0.3;
static const double pso_aSosyal = 0.2;
static const double pso_aSoc = 0.2;
```

Şekil 3.2.3.6. PSO optimizasyonu için karar ağırlıklarının tanımlandığı fonksiyon

```
double PSOSkorHesaplama(Map<String, dynamic> istasyon) {
  // PSO_Adım_1
  double guc     = (istasyon['ozellikler'][0]['guc'] as num).toDouble();
  double fiyat   = (istasyon['ozellikler'][0]['tarife'] as num).toDouble();
  double uzaklik = istasyon['uzaklik'] as double? ?? 0.0;
  // PSO_Adım_2
  double maksMesafe = widget.soc / km_basina_soc;
  if (uzaklik > maksMesafe) return 0.0;
  // PSO_Adım_3
  double normGuc     = (Gmaks > 0) ? guc / Gmaks : 0.0;
  double normFiyat   = (Fmaks > 0) ? fiyat / Fmaks : 0.0;
  double normUzaklik = (Umaks > 0) ? uzaklik / Umaks : 0.0;
  double normSoc     = widget.soc / 100.0;
  // PSO_Adım_4
  double dinamikAtalet   = normSoc * pso_aAtalet;
  double dinamikBilissel = normSoc * pso_aBilissel;
  double dinamikSosyal   = (1 - normSoc) * pso_aSosyal;
  double dinamikSoc      = (1 - normSoc) * pso_aSoc;
   // PSO_Adım_5
  double dinamikToplam    = dinamikAtalet + dinamikBilissel + dinamikSosyal + dinamikSoc;
  double agirlikliAtalet    = dinamikAtalet   / dinamikToplam;
  double agirlikliBilissel  = dinamikBilissel / dinamikToplam;
  double agirlikliSosyal    = dinamikSosyal   / dinamikToplam;
  double agirlikliSoc       = dinamikSoc      / dinamikToplam;
  // PSO_Adım_6
  double atalet = normGuc * 0.5 + normSoc * 0.5;
  double bilissel = pow(normGuc, 2).toDouble();
  double sosyal = (1 - normUzaklik) + (1 - normFiyat);
  // PSO_Adım_7
  double pso_skor = agirlikliAtalet * atalet + agirlikliBilissel * bilissel
                  + agirlikliSosyal * (sosyal / 2) + agirlikliSoc * normSoc;
  // PSO_Adım_8
  return pso_skor.clamp(0.0, 1.0);
}
```

Şekil 3.2.3.7. PSO optimizasyonu için hesaplama fonksiyonu

PSOSkorHesaplama fonksiyonu; ilk 3 adımda LPSkorHesaplama fonksiyonu ile aynı işlemleri yapmaktadır. Adım_4'te kullanıcının mevcut SoC düzeyine göre, PSO bileşenlerinin ağırlıkları dinamik şekilde ayarlanır. Atalet ve bilişsel bileşenler, yüksek SoC düzeylerinde artırılır. Sosyal bileşen ve SoC hassasiyeti, düşük SoC düzeylerinde öne çıkar. Bu sayede algoritma, kullanıcının aciliyet düzeyine uygun bir karar stratejisi benimser. Adım_5'te toplam dinamik ağırlık hesaplanarak her bileşenin göreli etkisi belirlenir. Bu değerler daha sonra bileşen skorlarını ağırlıklandırmak için kullanılır.

Adım_6'da denklem (3.5), (3.6) ve (3.7)'ye göre PSO algoritmasına özgü üç karar bileşeni hesaplanır. Adım_7'de denklem (3.8)'e göre üç karar bileşeni ve SoC katkısı, dinamik olarak belirlenen ağırlıklarla birleştirilerek PSO tabanlı nihai skor hesaplanır. Bu bileşenlerin her biri kullanıcı bağlamına göre optimize edilmiş dinamik etkiler barındırmaktadır. Son olarak Adım_8'de hesaplanan skorlar 0 ile 1 aralığında sınırlandırılır ve tüm skorların geçerli bir aralıkta kalması sağlanır.

Pekiştirmeli Öğrenme (RL)

RL, ajanların bir ortamdan gelen ödül geri bildirimine göre en uygun stratejiyi öğrenmesini amaçlar (Sutton & Barto, 2018).
Bu çalışmada, doğrudan öğrenen bir ajan modeli kullanılmamakla birlikte, kriterler üzerinden oluşturulan ödül fonksiyonu tanımlanmıştır:

$$RL_{\text{ödül}} = a_g.G_{norm} + a_f.(1 - F_{norm}) + a_u.(1 - U_{norm}) + a_{soc}.(1 - SoC_{norm}) \quad (3.9)$$

$$x = k.(RL_{\text{ödül}} - RL_{merkez}) \quad (3.10)$$

$$RL_{skor} = \frac{\tanh(x) + 1}{2} \quad (3.11)$$

Ödül fonksiyonu, yüksek güç, düşük fiyat ve yakın mesafe gibi olumlu kriterlerden maksimum fayda sağlayacak şekilde tasarlanmıştır.

```
static const double rl_aGuc = 0.25;
static const double rl_aSoc   = 0.25;
static const double rl_aFiyat = 0.25;
static const double rl_aUzaklik  = 0.25;
static const double rl_k       = 2.0;
static const double rl_merkezleme  = 0.5;
```

Şekil 3.2.3.8. RL optimizasyonu için karar ağırlıkları ve Tanh sabitlerinin tanımlandığı fonksiyon

```dart
double RLSkorHesaplama(Map<String, dynamic> istasyon) {
  // RL_Adım_1
  double guc     = (istasyon['ozellikler'][0]['guc'] as num).toDouble();
  double fiyat   = (istasyon['ozellikler'][0]['tarife'] as num).toDouble();
  double uzaklik = istasyon['uzaklik'] as double? ?? 0.0;
  // RL_Adım_2
  double maksMesafe = widget.soc / km_basina_soc;
  if (uzaklik > maksMesafe) return 0.0;
  // RL_Adım_3
  double normGuc     = (Gmaks > 0) ? guc / Gmaks : 0.0;
  double normFiyat   = (Fmaks > 0) ? fiyat / Fmaks : 0.0;
  double normUzaklik = (Umaks > 0) ? uzaklik / Umaks : 0.0;
  double normSoc     = widget.soc / 100.0;
  // RL_Adım_4
  double dinamikGuc     = normSoc * rl_aGuc;
  double dinamikFiyat   = normSoc * rl_aFiyat;
  double dinamikUzaklik = (1 - normSoc) * rl_aUzaklik;
  double dinamikSoc     = (1 - normSoc) * rl_aSoc;
  // RL_Adım_5
  double dinamikToplam    = dinamikGuc + dinamikFiyat + dinamikUzaklik + dinamikSoc
  double agirlikliGuc     = dinamikGuc     / dinamikToplam;
  double agirlikliFiyat   = dinamikFiyat   / dinamikToplam;
  double agirlikliUzaklik = dinamikUzaklik / dinamikToplam;
  double agirlikliSoc     = dinamikSoc     / dinamikToplam;
  // RL_Adım_6
  double odulGuc     = normGuc;
  double odulFiyat   = 1- normFiyat;
  double odulUzaklik = 1- normUzaklik;
  double odulSoc     = 1- normSoc;
  // RL_Adım_7
  double toplamOdul = agirlikliGuc * odulGuc
                    + agirlikliFiyat * odulFiyat
                    + agirlikliUzaklik * odulUzaklik
                    + agirlikliSoc * odulSoc;
  // RL_Adım_8
  double x = rl_k  * (toplamOdul - rl_merkezleme);
  double tanhVal = (exp(x) - exp(-x)) / (exp(x) + exp(-x));
  double skor = (tanhVal + 1) / 2;
  // RL_Adım_9
  return skor.clamp(0.0, 1.0);
}
```

Şekil 3.2.3.9. RL optimizasyonu için hesaplama fonksiyonu

RLSkorHesaplama fonksiyonu; ilk 3 adımda LPSkorHesaplama fonksiyonu ile aynı işlemleri yapmaktadır. Adım_4'te her kriter için ağırlıklar, kullanıcının aciliyetine göre dinamik olarak hesaplanır. Yüksek SoC durumlarında, güç ve fiyat daha önemlidir. Düşük SoC durumlarında, uzaklık ve aciliyet ön plana çıkar. Bu dinamik yapı sayesinde, algoritma bağlama duyarlı şekilde davranır. Adım_5'te toplam ağırlıklar normalize edilerek her kriterin nihai etkisi belirlenir. Adım_6'da her kriter için pozitif ödül katkısı

hesaplanır. Adım_7'de denklem (3.9)'a göre her ödül bileşeni kendi ağırlığıyla çarpılarak bileşik ödül fonksiyonu oluşturulur. Adım_8'de denklem (3.10) ve (3.11)'e göre toplam ödül değeri, tanh tabanlı bir aktivasyon fonksiyonuyla ölçeklenir. Bu aktivasyon fonksiyonu, skorlarda sigmoidal bir geçiş davranışı tanımlar ve sistemin daha yumuşak kararlar almasını sağlar. Son olarak Adım_9'da hesaplanan skorlar 0 ile 1 aralığında sınırlandırılır ve tüm skorların geçerli bir aralıkta kalması sağlanır.

3.3. Simülasyon ve Değerlendirme Ortamı

Önerilen sistemin etkinliğini değerlendirmek amacıyla, gerçek dünya kullanım senaryolarını mümkün olduğunca yakından taklit eden bir simülasyon ortamı oluşturulmuştur. Bu ortam, mobil uygulamanın, veri entegrasyonunun ve optimizasyon motorunun farklı koşullar altında (kullanıcı konumları, batarya doluluk seviyeleri vb.) nasıl performans gösterdiğini test etmeye olanak tanımış; aynı zamanda, şarj istasyonlarının ve kullanıcıların tam entegre edildiği bir ağ kurmaya gerek kalmadan analiz için sonuç verisi toplamayı mümkün kılmıştır. Simülasyon ortamı, optimizasyon motorunu gerçek zamanlı verilerle çalışıyormuş gibi besleyerek, neredeyse gerçek zamanlı şarj senaryolarını taklit etmiştir. Bu doğrultuda gerçek coğrafi ve senaryo verileri kullanılmıştır.

Değerlendirme, Türkiye'nin Bursa ilinde gerçekleştirilen bir vaka çalışmasına odaklanmış; farklı kullanıcı senaryolarını temsil etmesi amacıyla şehir içinde üç farklı konum ve bu konumlara ait farklı senaryolar seçilerek simülasyon yapılmıştır.

Çizelge 3.3.1. OCPI EV Şarj İstasyonu Optimizasyonu uygulamasının test edilebilmesi için belirlenen konumlar, koordinatlar, senaryolar ve fiyat/güç profilleri

Konum	Koordinat	Senaryo Adı	Fiyat/Güç Profili
Tofaş Türk Otomobil Fabrikası	40,259017, 29,061945	Şehir merkezi	Orta ve yüksek fiyat, orta güç (60 kW ve 75 kW)
Bursa Uludağ Üniversitesi	40,226485, 28,876042	Şehir dışı	Düşük ve orta fiyat, düşük güç (22 kW ve 30 kW)

Togg Teknoloji Kampüsü	40,395748, 29,082898	Uzun yol	Yüksek fiyat, yüksek güç (180 kW ve 250 kW)

Bu konumlar, farklı bağlamsal senaryoları temsil edecek şekilde seçilmiştir:

Şehir Merkezi Senaryosu:

Örnek olarak, Bursa şehir merkezinde yer alan Tofaş Türk Otomobil Fabrikası çevresi seçilmiştir. Bu senaryoda orta-yüksek fiyatlandırma düzeyi ve orta seviye şarj gücüne sahip istasyonlar (örneğin 60 kW ve 75 kW) hedef alınmıştır. Amaç, şehir içi yüksek talep ve konfor için daha fazla ödeme isteği olabilecek yoğun kentsel ortamları taklit etmektir.

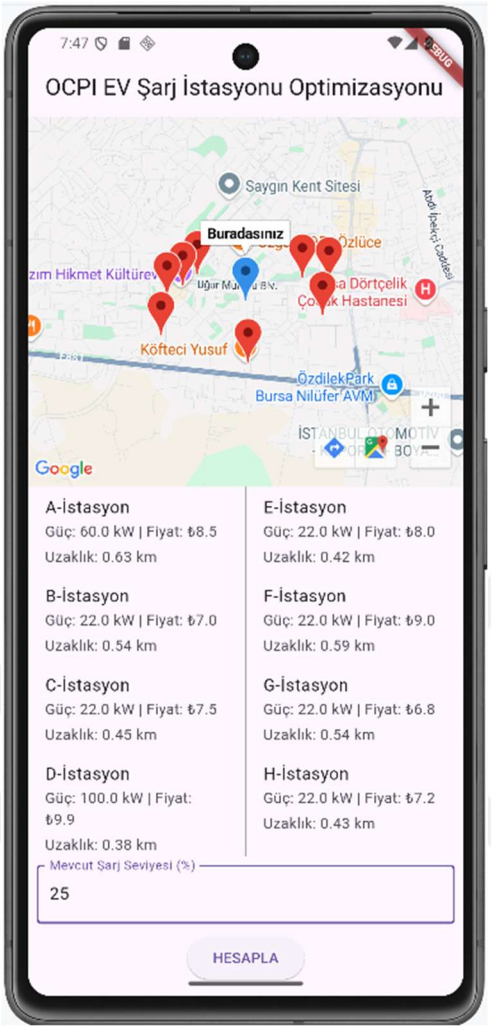
Şehir Dışı Senaryosu:

Örnek olarak, şehir merkezinin biraz dışında yer alan Uludağ Üniversitesi çevresi seçilmiştir. Bu bölgede düşük-orta düzeyde fiyatlandırma mevcutken, yalnızca düşük güçlü şarj cihazları (22 kW ve 30 kW) hedef alınmıştır. Bu senaryo, daha düşük maliyetli ancak daha yavaş şarj imkânı sunan ve yoğunluğun daha az olduğu yerleşim bölgelerini temsil etmektedir.

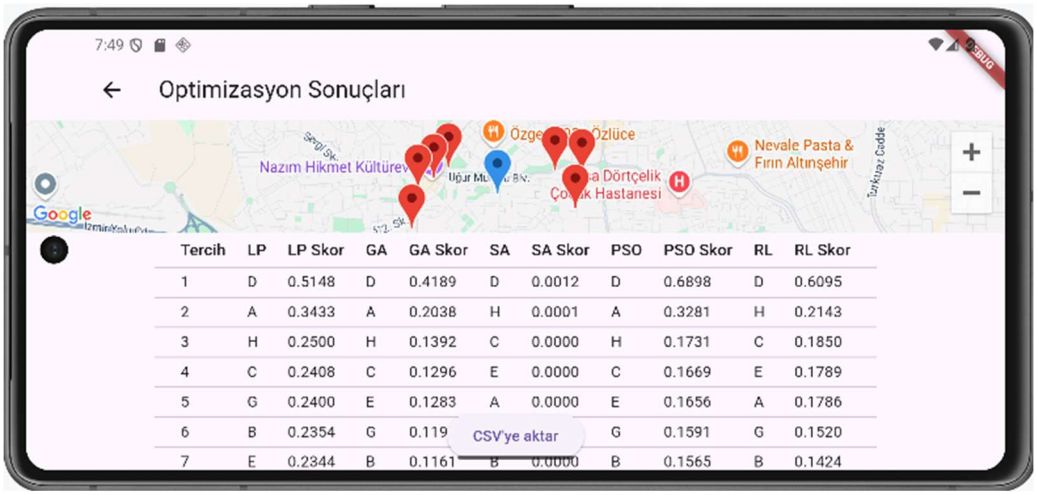
Uzun Yol Senaryosu:

Örnek olarak, Togg Teknoloji Kampüsü başlangıç noktalı uzun mesafeli bir seyahatte rota üzerindeki istasyonlar ele alınmıştır. Bu senaryoda yüksek fiyatlı ancak çok yüksek güçlü şarj cihazları (180 kW ve 250 kW DC hızlı şarj) yer almaktadır. Amaç, zamanın kritik olduğu otoyol şarj noktalarını simüle etmek ve bu bağlamda hızın öncelikli olduğu durumları test etmektir.

Simülasyon ve test işlemleri, Android Studio üzerinde Pixel 7 sanal cihazı kullanılarak gerçekleştirilmiştir. Şekil 3.3.1'de uygulama ekranının ve Şekil 3.3.2'de uygulama sonuç ekranının, simülasyon cihazı üzerindeki görüntüsü görülmektedir.



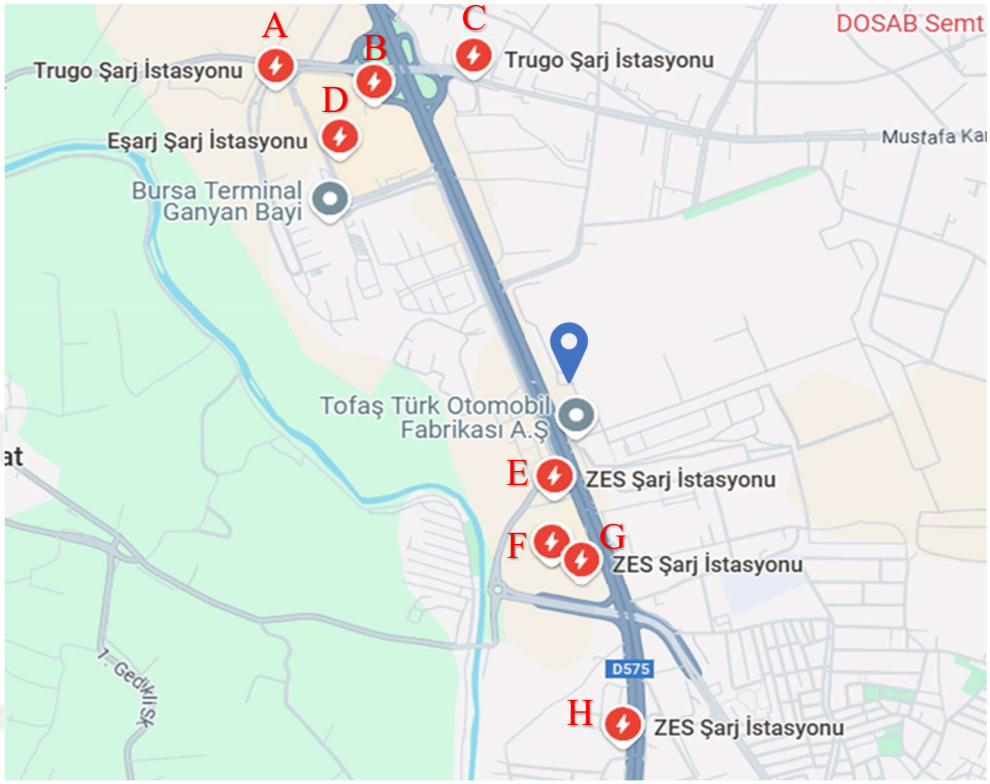
Şekil 3.3.1. Uygulama ana ekranı



Şekil 3.3.2. Uygulama sonuç ekranı

4. BULGULAR ve TARTIŞMA

Şehir Merkezi Senaryosu



Şekil 4.1. Şehir merkezi senaryosu için seçilen kullanıcı konumu ve istasyonların konumu

Şekil 4.1.'de gösterildiği üzere, belirlenen konumlardan Tofaş Türk Otomobil Fabrikasının etrafından 8 adet istasyon Google Maps uygulaması aracılığıyla belirlenmiştir.

Çizelge 4.1. Şehir merkezi senaryosu için belirlenen istasyonların bilgileri

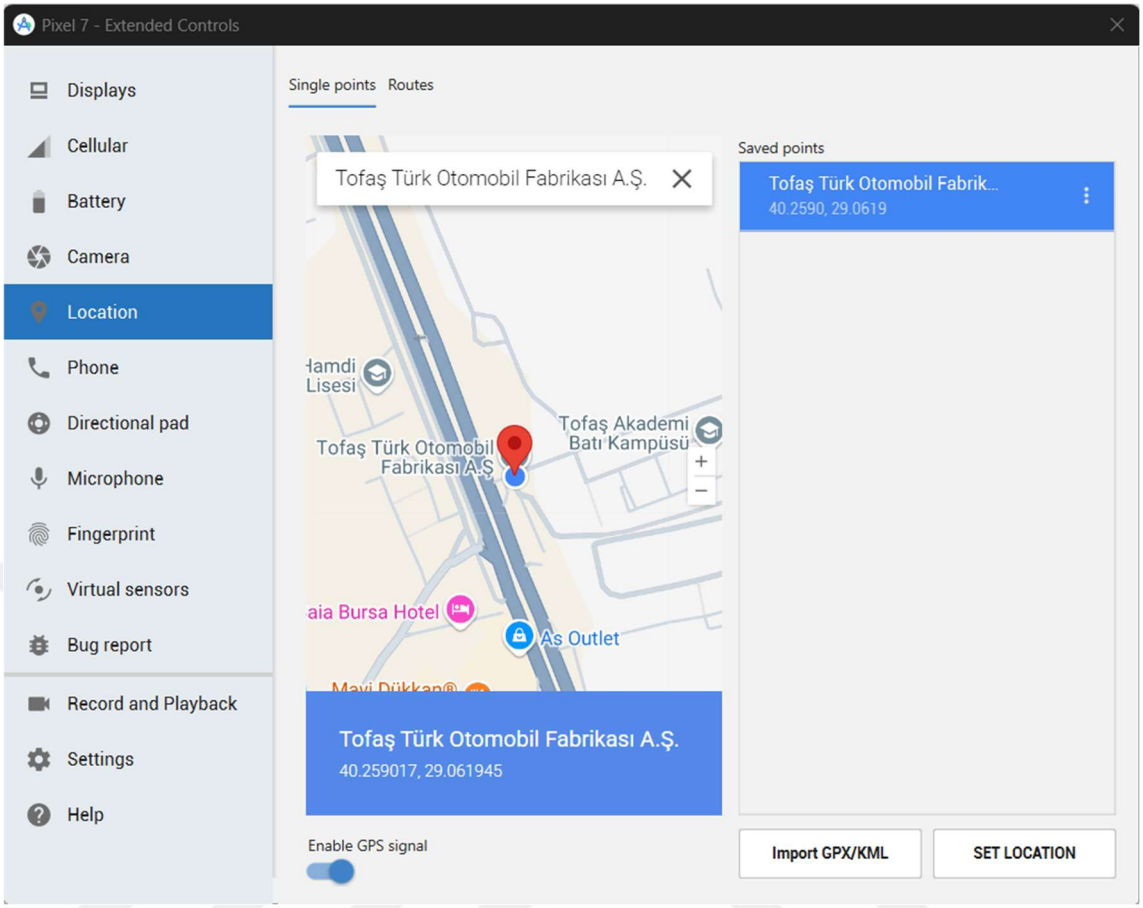
İstasyon Adı	Enlem	Boylam	Güç (kw)	Fiyat (₺)
A	40,269382	29,045385	75	9,7
B	40,269012	29,055035	75	9,6
C	40,269799	29,058434	75	9,5
D	40,267530	29,053821	60	7,2
E	40,258128	29,061044	75	9,9
F	40,256137	29,061104	60	7,7
G	40,255859	29,061954	60	7,5
H	40,251088	29,063532	60	7,2

Çizelge 4.1'de belirlenen 8 istasyonun enlem, boylam, güç ve tarife bilgileri Şekil 4.2'deki görüleceği gibi "istasyon_konumlari.json" dosyasına girilmiştir.

main.dart {} istasyon_konumlari.json

```
[
  {
    "kimlik": "TR-001",
    "isim": "A-İstasyon",
    "konum": {"enlem": 40.269382, "boylam": 29.045385},
    "ozellikler": [{"guc": 75, "tarife": 9.7}]
  },
  {
    "kimlik": "TR-002",
    "isim": "B-İstasyon",
    "konum": {"enlem": 40.269012, "boylam": 29.055035},
    "ozellikler": [{"guc": 75, "tarife": 9.6}]
  },
  {
    "kimlik": "TR-003",
    "isim": "C-İstasyon",
    "konum": {"enlem": 40.269799, "boylam": 29.058434},
    "ozellikler": [{"guc": 75, "tarife": 9.5}]
  },
  {
    "kimlik": "TR-004",
    "isim": "D-İstasyon",
    "konum": {"enlem": 40.267530, "boylam": 29.053821},
    "ozellikler": [{"guc": 60, "tarife": 7.2}]
  },
```

Şekil 4.2. Şehir merkezi senaryosu için "istasyon_konumlari.json" dosyası

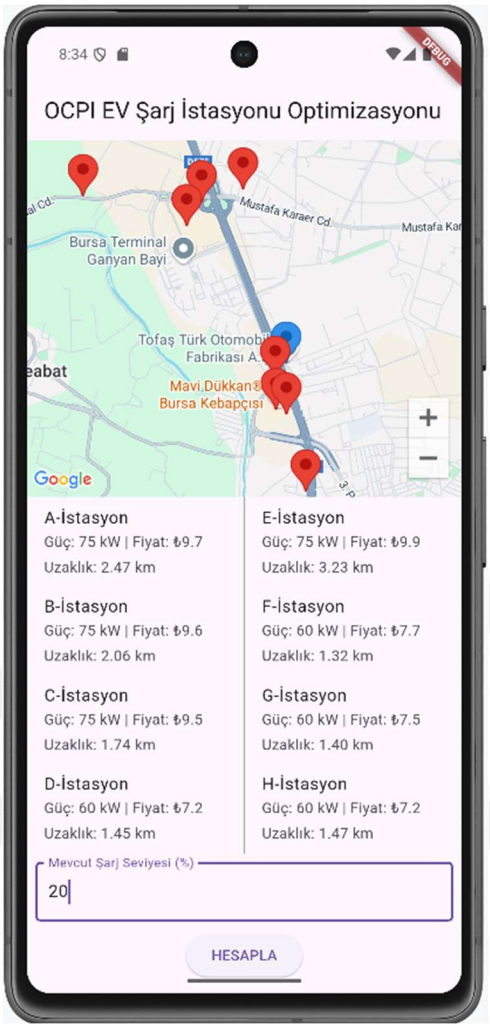


Şekil 4.3. Şehir merkezi senaryosu için seçilen konumun simülasyon cihazında aktifleştirilmesi

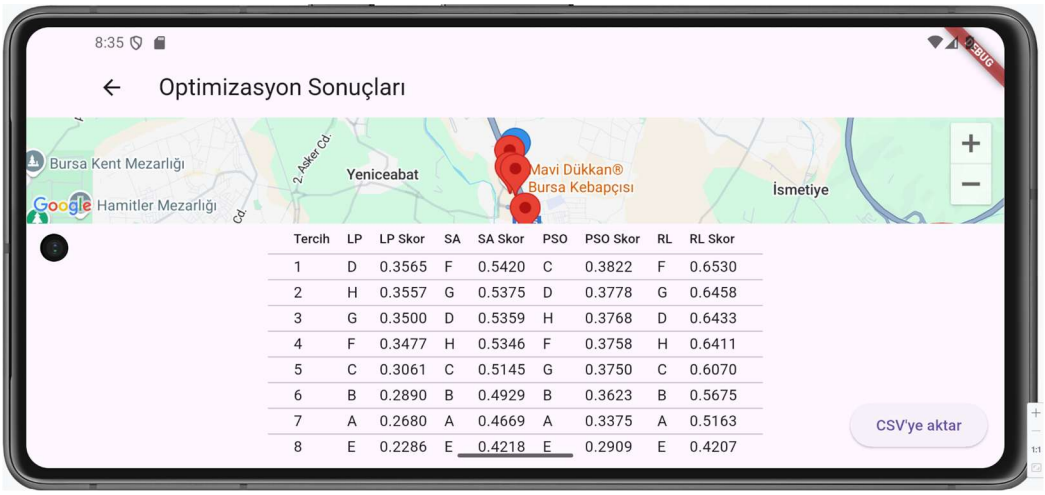
Şekil 4.3'te şehir merkezi senaryosu kullanıcı konumu için seçilen Tofaş Türk Otomobil Fabrikası konumu simülasyon cihazında aktifleştirilmiştir, yani cihazın mevcut konum olarak bu konumu alması sağlanmıştır.

Bu işlemlerin ardından, Android Studio Terminal arayüzünden sırasıyla "flutter clean" ile proje temizlemiş, "flutter pub get" komutu ile projenin "pubspec.yaml" dosyasında belirtilen bağımlılıkları indirip projeye eklenmesi sağlanmış ve "flutter run" komutu ile uygulama kodunun derlenip simülasyon cihazına yüklenmesi sağlanmıştır.

Şekil 4.4'te uygulama ana ekranı aracılığıyla kullanıcı SoC'si bilgisi %20 girilerek Şekil 4.5'te ki sonuçlar elde edilmiştir.

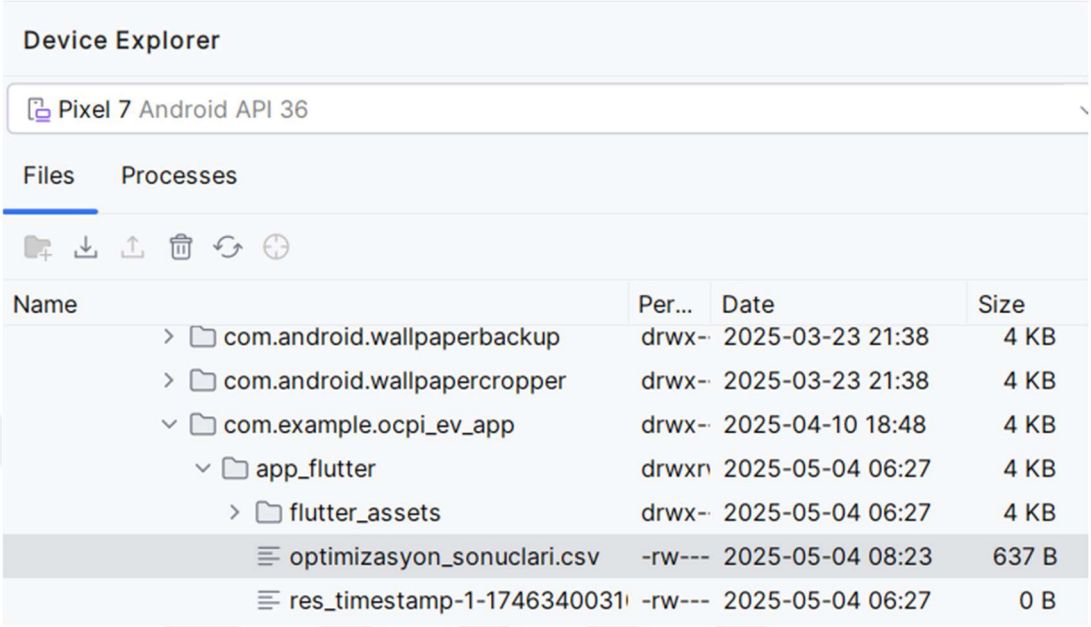


Şekil 4.4. Şehir merkezi senaryosu için uygulama ana ekranı



Şekil 4.5. Şehir merkezi senaryosu için optimizasyon sonuçları ekranı (SoC %20 iken)

Sonuçlar, Şekil 4.5 ekranında bulunan CCV'ye aktar butonu aracılığı ile Şekil 4.6'da ki gibi simülasyon cihazının iç belleğine kaydedilmiştir.



Şekil 4.6. Cihaz belleğindeki optimizasyon sonuçları kayıt alanı

%20 SoC değeri için elde edilen sonuçlar Çizelge 4.2'de paylaşılmaktadır.

Çizelge 4.2. Şehir merkezi senaryosunda %20 SoC için sonuçlar

Tercih	LP	LP Skor	SA	SA Skor	PSO	PSO Skor	RL	RL Skor
1	D	0,3565	F	0,5420	C	0,3822	F	0,6530
2	H	0,3557	G	0,5375	D	0,3778	G	0,6458
3	G	0,3500	D	0,5359	H	0,3768	D	0,6433
4	F	0,3477	H	0,5346	F	0,3758	H	0,6411
5	C	0,3061	C	0,5145	G	0,3750	C	0,6070
6	B	0,2890	B	0,4929	B	0,3623	B	0,5675
7	A	0,2680	A	0,4669	A	0,3375	A	0,5163
8	E	0,2286	E	0,4218	E	0,2909	E	0,4207

Testler %50 ve %80 SoC değerleri için aynı senaryoda tekrarlanmıştır.

%50 SoC değeri için elde edilen sonuçlar Çizelge 4.3'te, %80 SoC değeri için elde edilen sonuçlar Çizelge 4.4'te paylaşılmaktadır.

Çizelge 4.3. Şehir merkezi senaryosunda %50 SoC için sonuçlar

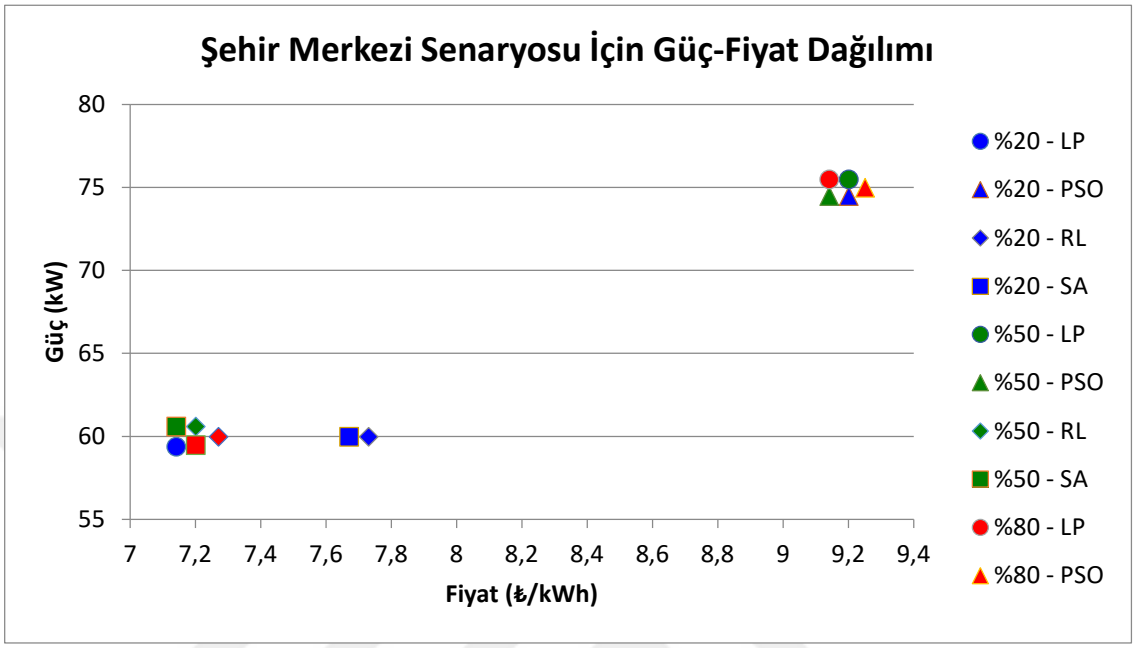
Tercih	LP	LP Skor	SA	SA Skor	PSO	PSO Skor	RL	RL Skor
1	C	0,6043	D	0,6102	C	0,6752	D	0,5307
2	B	0,5923	H	0,6093	B	0,6643	H	0,5293
3	D	0,5796	F	0,6084	A	0,6506	F	0,5280
4	H	0,5790	G	0,6077	E	0,6250	G	0,5268
5	A	0,5776	C	0,5911	D	0,5693	C	0,5005
6	G	0,5750	B	0,5743	H	0,5687	B	0,4732
7	F	0,5734	A	0,5540	F	0,5682	A	0,4393
8	E	0,5500	E	0,5179	G	0,5677	E	0,3775

Çizelge 4.4. Şehir merkezi senaryosunda %80 SoC için sonuçlar

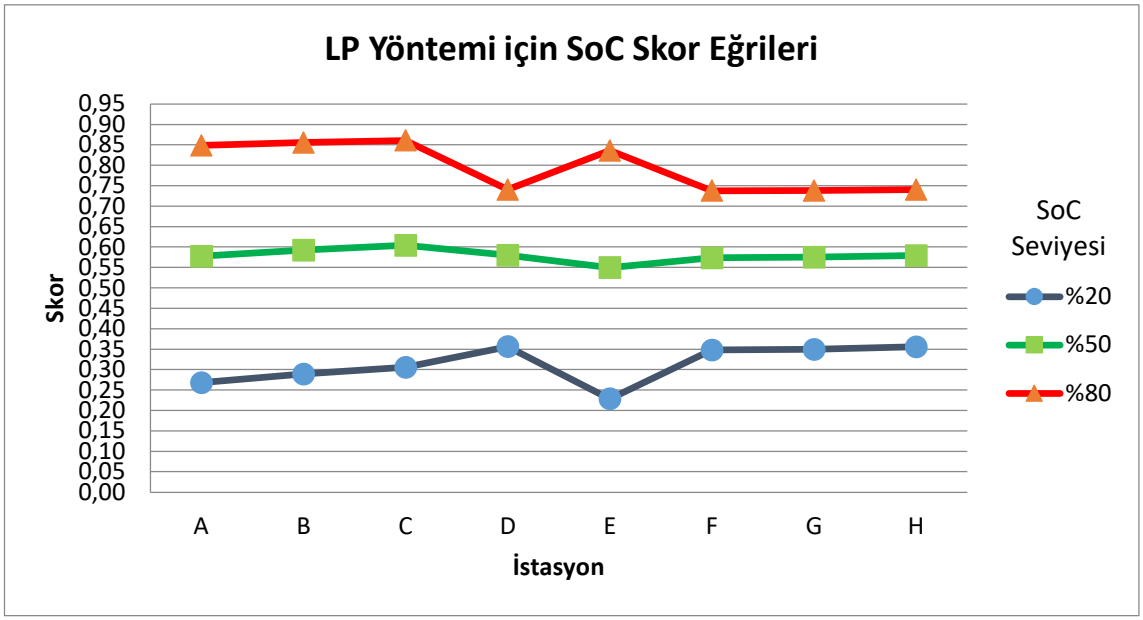
Tercih	LP	LP Skor	SA	SA Skor	PSO	PSO Skor	RL	RL Skor
1	C	0,8610	D	0,6320	C	0,8894	D	0,5041
2	B	0,8556	H	0,6316	B	0,8854	H	0,5035
3	A	0,8489	G	0,6250	A	0,8806	G	0,4935
4	E	0,8364	F	0,6213	E	0,8714	F	0,4879
5	D	0,7407	C	0,6177	D	0,7037	C	0,4823
6	H	0,7405	B	0,6087	H	0,7035	B	0,4684
7	G	0,7386	A	0,5981	F	0,7033	A	0,4518
8	F	0,7379	E	0,5785	G	0,7031	E	0,4207

Çalışmada kullanılan dört optimizasyon yöntemi (LP, SA, PSO, RL), farklı SOC seviyelerinde (20%, 50%, 80%) kullanıcıya önerilen şarj istasyonlarını seçerken güç, fiyat ve kullanıcının istasyona olan mesafesi değişkenlerini dikkate alarak farklı karar stratejileri sergilemiştir.

Şekil 4.7'de her yöntemin şehir merkezi senaryosunda SoC bazlı seçtiği ilk istasyonların Güç-Fiyat dağılımı paylaşılmaktadır.



Şekil 4.7. Şehir merkezi senaryosu için SoC bazlı seçilen ilk istasyonların Güç-Fiyat dağılımı



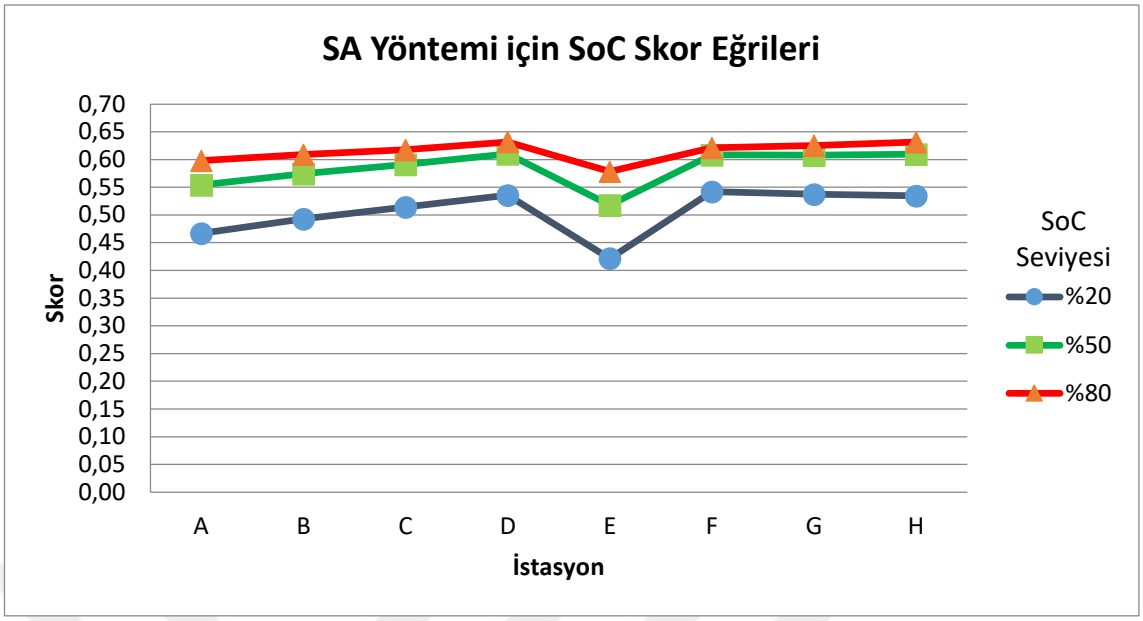
Şekil 4.8. Şehir merkezi senaryosunda LP yöntemi için SoC skor eğrileri

Şekil 4.8’de şehir merkezi senaryosunda LP yöntemi için SoC skor eğrileri paylaşılmaktadır.

%20 SoC seviyesinde LP algoritması, öncelikli olarak kısa mesafeleri ve orta düzeyde güç-fiyat oranlarını temel alan bir strateji benimsemektedir. Bu bağlamda İstasyon D en yüksek skorla öne çıkmakta; bu da yakınlık ve makul tarife seviyelerine verilen önemi yansıtmaktadır. İstasyonlar H, G ve F de yüksek sıralamalarda yer almakta olup, LP'nin acil şarj senaryolarında hızlı erişilebilirlik ve orta düzeyde maliyet etkinliği temelinde optimizasyon yaptığı görülmektedir.

%50 SoC seviyesinde LP algoritması, tercihini daha yüksek güce sahip olan (75 kW) ancak maliyeti biraz daha yüksek olan İstasyon C’ye kaydırmakta; bu da yakınlık odaklı yaklaşımdan verimlilik ve maliyet dengesine yönelen stratejik bir değişimi göstermektedir. İstasyonlar B ve D de üst sıralarda yer almakta, aciliyetin azalmasıyla birlikte daha dengeli kararların devreye girdiğine işaret etmektedir.

%80 SoC seviyesinde LP algoritması, yüksek güç değerine sahip istasyonları, daha yüksek tarifelerine rağmen, belirgin şekilde önceliklendirmektedir. İstasyon C bu seviyede en yüksek skorla öne çıkarken, B ve A istasyonları da yüksek sıralarda yer almaktadır. Bu eğilim, yüksek SoC seviyelerinde düşük aciliyetle birlikte verimlilik ve şarj hızının maksimize edilmesine yönelik stratejik bir kaymayı göstermektedir.



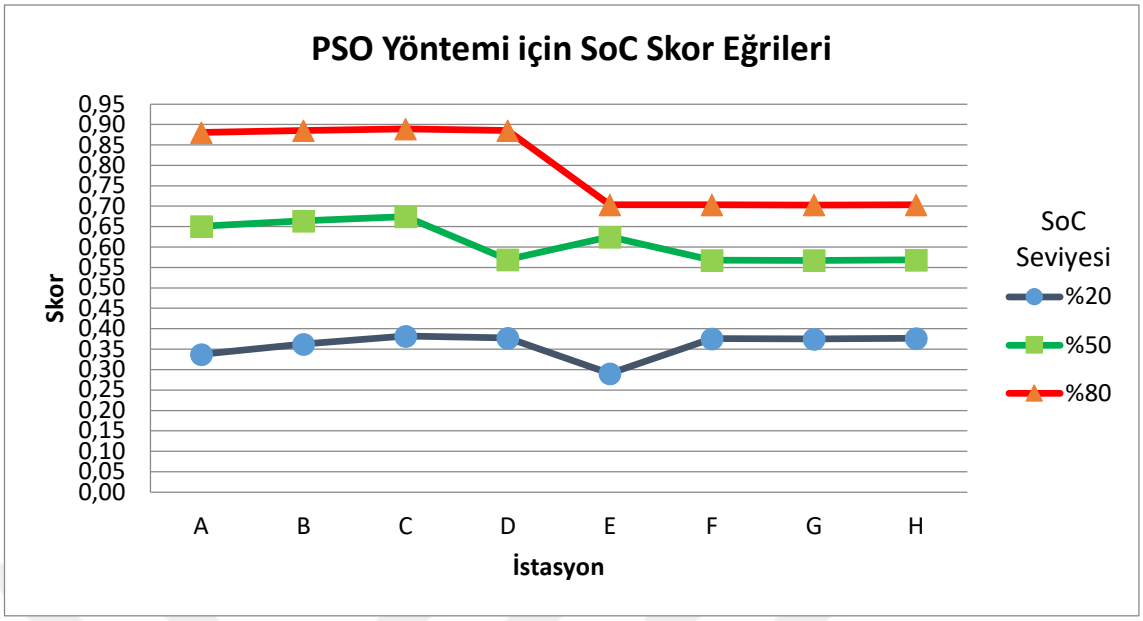
Şekil 4.9. Şehir merkezi senaryosunda SA yöntemi için SoC skor eğrileri

Şekil 4.9'da şehir merkezi senaryosunda SA yöntemi için SoC skor eğrileri paylaşılmaktadır.

%20 SoC seviyesinde SA algoritması, güce kıyasla yakınlığı belirgin şekilde önceliklendirerek İstasyon F'yi en yüksek skorla seçmektedir. G, D ve H istasyonları da yakın sıralarda yer almakta; bu durum SA'nın acil ihtiyaçlara yönelik, erişilebilir en yakın istasyonlara güçlü biçimde öncelik veren yaklaşımını ortaya koymaktadır.

%50 SoC seviyesinde SA algoritması, tercihini hafifçe değiştirerek İstasyon D'yi ön plana çıkarmaktadır. Bu, yakınlık ile orta-yüksek güç erişimi arasında optimal denge kurulduğunu göstermektedir. H ve F istasyonlarının yüksek sıralamaları, kısa mesafe önceliğinin makul güçle birlikte sürdüğünü göstermektedir.

%80 SoC seviyesinde SA algoritması, artan SoC seviyesine rağmen İstasyon D'yi tercih etmeye devam etmektedir. Bu, SA'nın orta düzeyde güç, makul fiyat ve kısa mesafeyi istikrarlı şekilde önceliklendirdiğini ortaya koymaktadır. H istasyonu da benzer şekilde yüksek sıralarda yer almakta; bu durum, SA'nın enerji seviyesi artsa dahi tercih tutarlılığı gösterdiğini kanıtlamaktadır.



Şekil 4.10. Şehir merkezi senaryosunda PSO yöntemi için SoC skor eğrileri

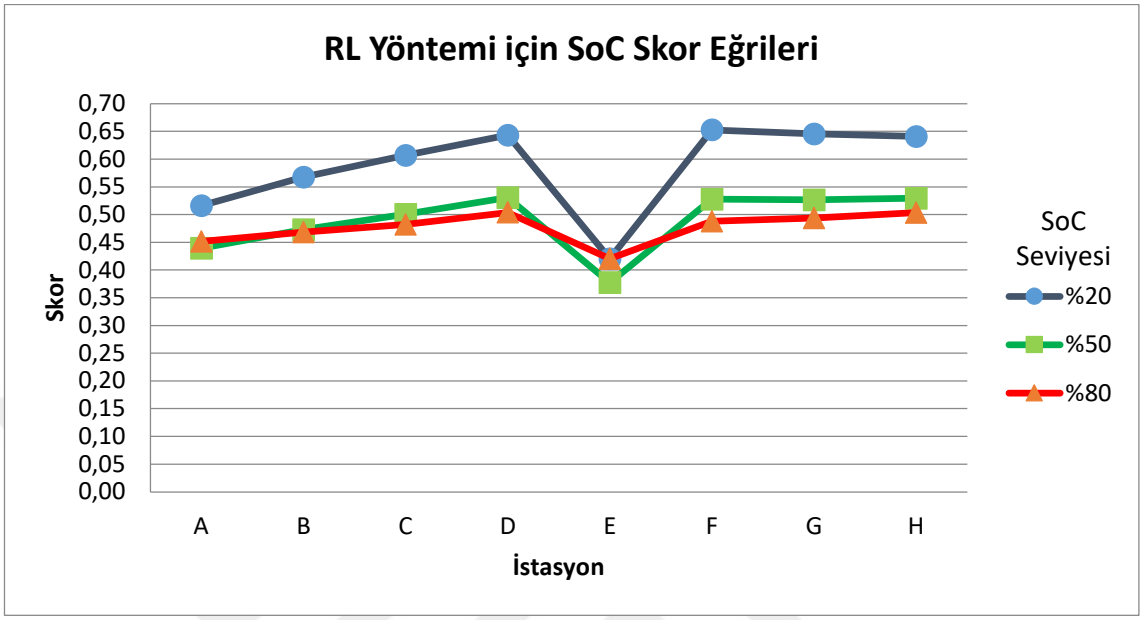
Şekil 4.10'da şehir merkezi senaryosunda PSO yöntemi için SoC skor eğrileri paylaşılmaktadır.

%20 SoC seviyesinde PSO algoritması, görece uzak ancak güçlü istasyonları tercih ederek İstasyon C'yi önceliklendirmektedir. Bu, PSO'nun yüksek güç değerlerine olan ilgisini, acil şarj koşullarında dahi koruduğunu göstermektedir. D ve H istasyonları da tercihler arasında yer almakta, bu da mesafe ile güç arasında dengeli bir yaklaşım izlendiğini göstermektedir.

%50 SoC seviyesinde PSO algoritması, yüksek güç kapasiteli istasyonları belirgin biçimde önceliklendirmekte; İstasyon C ve B en yüksek skorları almaktadır. Bu durum, PSO'nun verimlilik ve hız odaklı stratejisini, maliyet ve mesafe açısından daha esnek bir tutumla sürdürdüğünü göstermektedir.

%80 SoC seviyesinde PSO algoritması, yüksek güç ve yüksek tarife değerlerine sahip istasyonları net şekilde ön plana çıkarmaktadır. İstasyon C en yüksek skorla tercih edilmekte, onu B ve A izlemektedir. Bu sonuç, PSO'nun aciliyetin düşük olduğu

koşullarda verimlilik ve hızlı şarjı agresif biçimde optimize eden özgün stratejisini açıkça ortaya koymaktadır.



Şekil 4.11. Şehir merkezi senaryosunda RL yöntemi için SoC skor eğrileri

Şekil 4.11'de şehir merkezi senaryosunda RL yöntemi için SoC skor eğrileri paylaşılmaktadır.

%20 SoC seviyesinde RL algoritması, belirgin biçimde yakın istasyonlara öncelik vermekte ve en yüksek skorları F, G ve D istasyonlarına atfetmektedir. Bu tercih, RL'nin uyarlanabilir doğasını ve acil şarj senaryolarında zaman kaybını minimize etmeye yönelik stratejik karar alma yetisini göstermektedir.

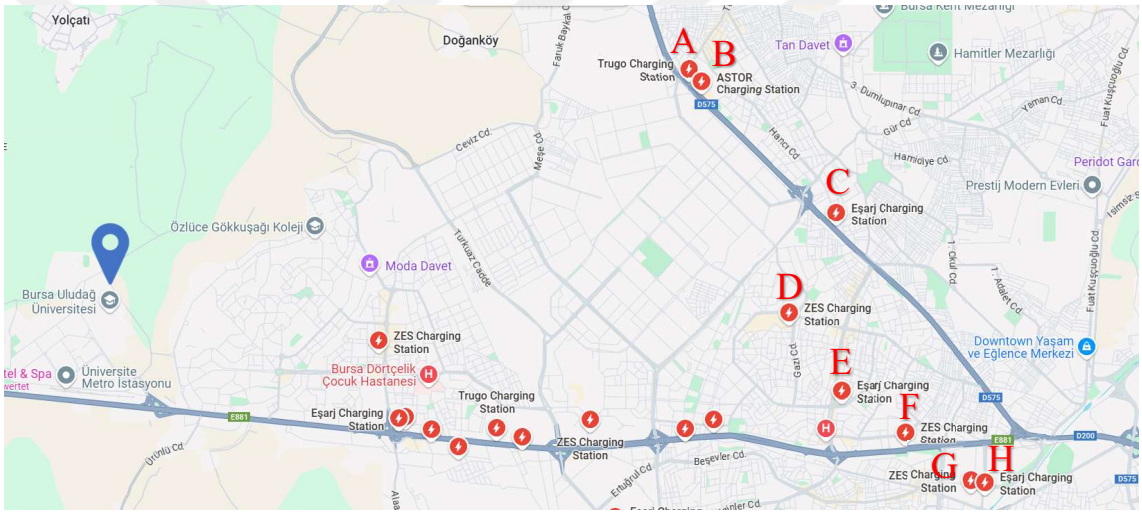
%50 SoC seviyesinde RL algoritması, yine yakınlık ve orta düzey güç kriterlerine göre tercihler yapmakta, ancak skorlar %20 SoC seviyesine göre düşmektedir. En yüksek skor, D istasyonuna aittir. Bu durum, RL'nin önlemli bir strateji izlediğini ortaya koymaktadır.

%80 SoC seviyesinde RL algoritması, yine orta derecede yakınlığa ve düşük tarifeye sahip istasyonları tercih etmeye devam etmekte ve D istasyonunu en üst sıraya yerleştirmektedir. Bu sürekli eğilim, LP ve PSO'nun yüksek güç istasyonlarına yönelik

agresif optimizasyon yaklaşımından ayrışmakta; RL'nin geleneksel optimizasyon felsefesini yansıtmaktadır.

Genel olarak değerlendirildiğinde, şehir merkezi senaryosunda, PSO yöntemi sistematik olarak yüksek performanslı ancak maliyetli çözümleri tercih ederken; SA ve RL yöntemleri daha kararlı, daha ekonomik ve erişilebilir istasyonlara yönelmiştir. LP yöntemi ise bu iki uç eğilim arasında denge kurarak SoC seviyesine bağlı olarak esnek bir karar mekanizması sergilemiştir.

Şehir Dışı Senaryosu



Şekil 4.12. Şehir dışı senaryosu için seçilen kullanıcı konumu ve istasyonların konumu

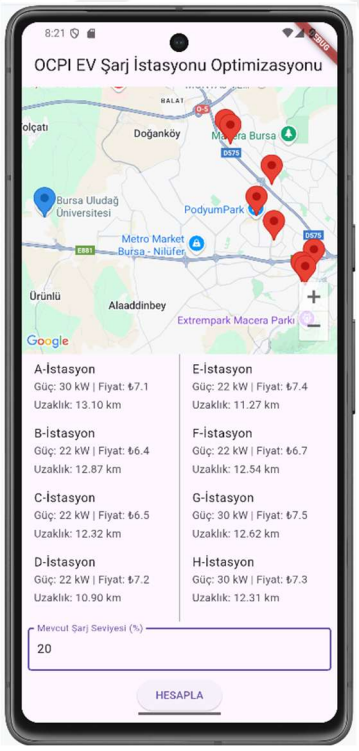
Şekil 4.12'de görüleceği üzere, şehir dışı senaryosu için Bursa Uludağ Üniversitesi'nin konumu etrafında 10 km'nin üzerinde uzaklıkta bulunan 8 adet istasyon Google Maps uygulaması aracılığıyla belirlenmiştir.

Çizelge 4.5'te belirlenen 8 istasyonun enlem, boylam, güç ve tarife bilgileri şehir merkezi senaryosunda da yapıldığı gibi "istasyon_konumlari.json" dosyasına girilmiştir.

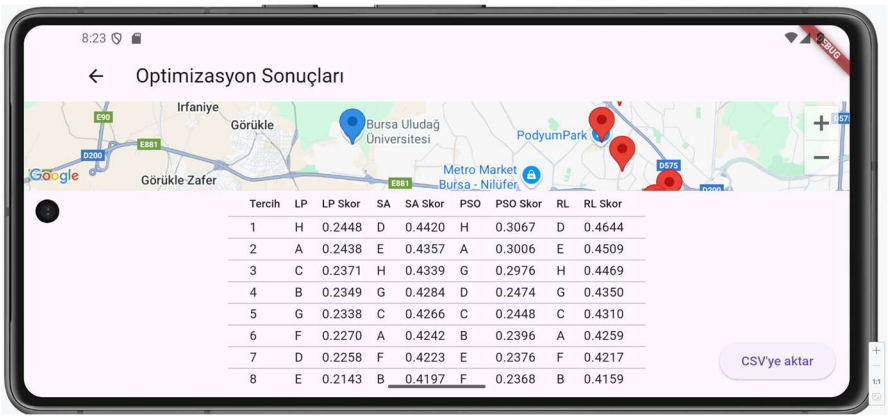
Çizelge 4.5. Şehir dışı senaryosu için belirlenen istasyonların bilgileri

İstasyon Adı	Enlem	Boylam	Güç (kw)	Fiyat (₺)
A	40,253219	28,957063	30	7,1
B	40,251566	28,959228	22	6,4
C	40,237683	28,977705	22	6,5
D	40,226994	28,970920	22	7,2
E	40,218617	28,978715	22	7,4
F	40,208917	28,996614	22	6,7
G	40,204507	28,991129	30	7,5
H	40,203184	28,992717	30	7,3

Bu işlemlerin ardından, Android Studio Terminal arayüzünden yazılıma Hot Restart (Shift + R) işlemi yapılarak, uygulama durumunun sıfırlanıp yeniden başlatılması sağlanmıştır. Bu işlem ile uygulama main() fonksiyonundan başlayarak tüm kodu baştan çalıştırır. Bellekteki tüm değişkenleri temizler ve yeni yapılan kod değişikliklerini uygular. Ama bu işlem tam derlemeye göre oldukça hızlı olduğu için tercih edilmiştir.



Şekil 4.13. Şehir dışı senaryosu için uygulama ana ekranı



Tercih	LP	LP Skor	SA	SA Skor	PSO	PSO Skor	RL	RL Skor
1	H	0.2448	D	0.4420	H	0.3067	D	0.4644
2	A	0.2438	E	0.4357	A	0.3006	E	0.4509
3	C	0.2371	H	0.4339	G	0.2976	H	0.4469
4	B	0.2349	G	0.4284	D	0.2474	G	0.4350
5	G	0.2338	C	0.4266	C	0.2448	C	0.4310
6	F	0.2270	A	0.4242	B	0.2396	A	0.4259
7	D	0.2258	F	0.4223	E	0.2376	F	0.4217
8	E	0.2143	B	0.4197	F	0.2368	B	0.4159

Şekil 4.14. Şehir dışı senaryosu için optimizasyon sonuçları ekranı (SoC %20 iken)

Şehir dışı senaryosunda elde edilen sonuçlar Çizelge 4.6, Çizelge 4.7 ve Çizelge 4.8'de paylaşılmaktadır.

Çizelge 4.6. Şehir dışı senaryosunda %20 SoC için sonuçlar

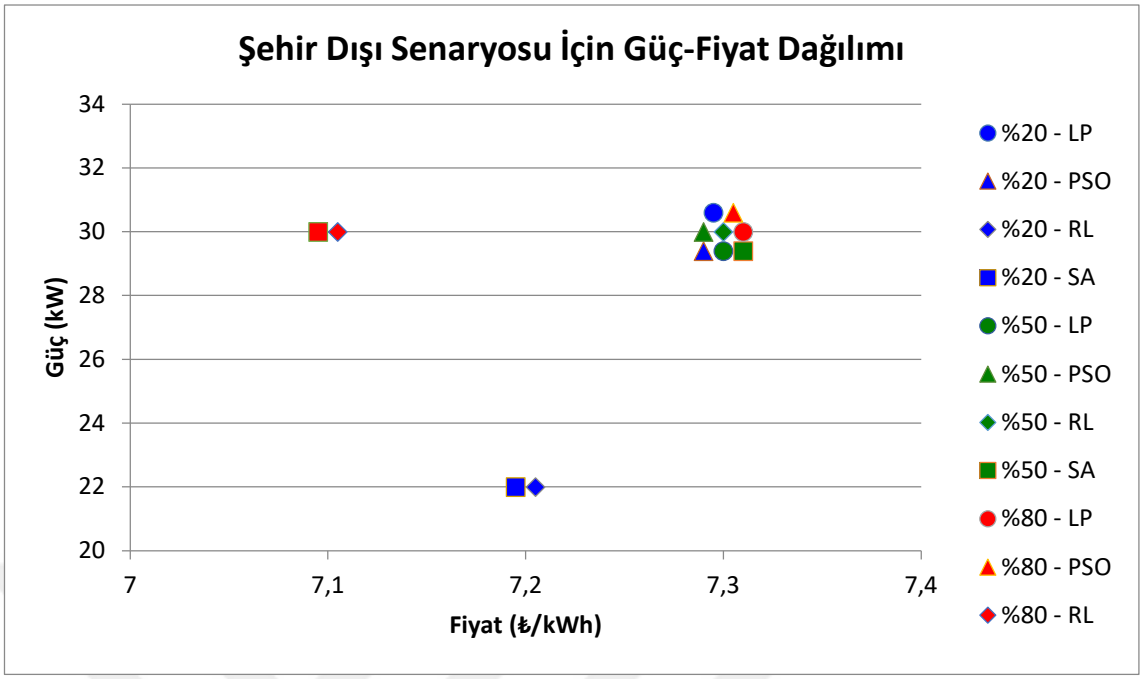
Tercih	**LP**	**LP Skor**	**SA**	**SA Skor**	**PSO**	**PSO Skor**	**RL**	**RL Skor**
1	H	0,2448	D	0,4420	H	0,3067	D	0,4644
2	A	0,2438	E	0,4357	A	0,3006	E	0,4509
3	C	0,2371	H	0,4339	G	0,2976	H	0,4469
4	B	0,2349	G	0,4284	D	0,2474	G	0,4350
5	G	0,2338	C	0,4266	C	0,2448	C	0,4310
6	F	0,2270	A	0,4242	B	0,2396	A	0,4259
7	D	0,2258	F	0,4223	E	0,2376	F	0,4217
8	E	0,2143	B	0,4197	F	0,2368	B	0,4159

Çizelge 4.7. Şehir dışı senaryosunda %50 SoC için sonuçlar

Tercih	LP	LP Skor	SA	SA Skor	PSO	PSO Skor	RL	RL Skor
1	H	0,5613	H	0,5299	H	0,6337	H	0,3981
2	A	0,5607	A	0,5253	A	0,6303	A	0,3902
3	G	0,5537	G	0,5230	G	0,6287	G	0,3862
4	C	0,4760	D	0,5099	D	0,4671	D	0,3638
5	B	0,4744	C	0,5080	C	0,4657	C	0,3604
6	F	0,4689	B	0,5042	B	0,4628	B	0,3538
7	D	0,4681	E	0,5027	E	0,4617	E	0,3513
8	E	0,4600	F	0,5022	F	0,4613	F	0,3504

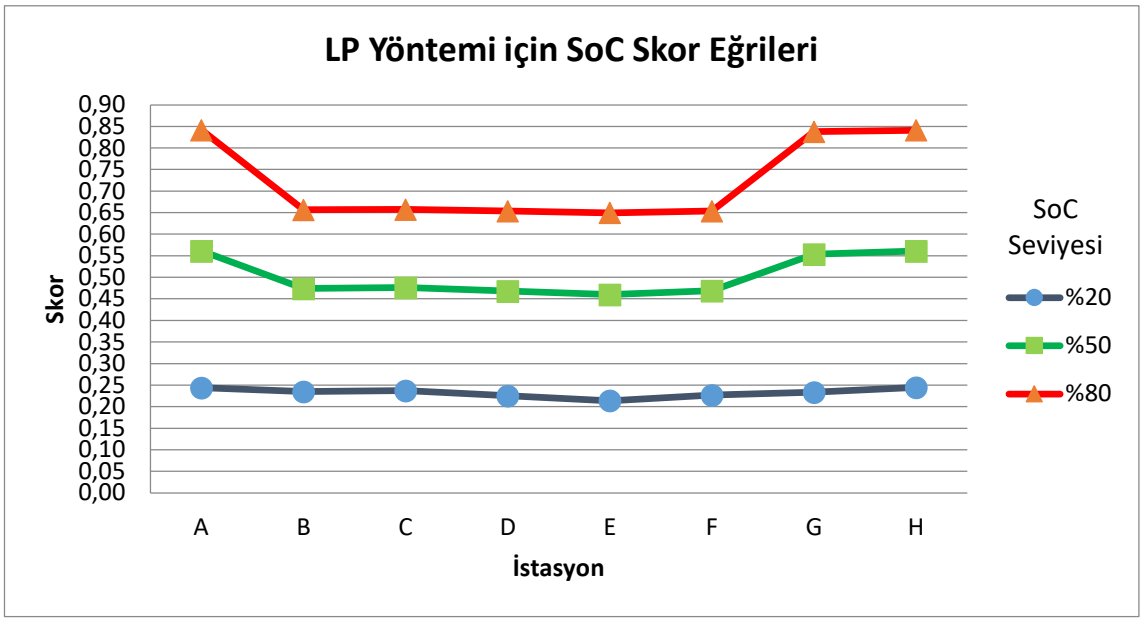
Çizelge 4.8. Şehir merkezi senaryosunda %80 SoC için sonuçlar

Tercih	LP	LP Skor	SA	SA Skor	PSO	PSO Skor	RL	RL Skor
1	H	0,8415	A	0,5916	H	0,8745	A	0,4416
2	A	0,8412	H	0,5887	A	0,8733	H	0,4370
3	G	0,8380	G	0,5807	G	0,8727	G	0,4243
4	C	0,6573	B	0,5510	D	0,6236	B	0,3764
5	B	0,6566	C	0,5503	C	0,6231	C	0,3753
6	F	0,6541	F	0,5432	B	0,6221	F	0,3638
7	D	0,6537	D	0,5352	E	0,6217	D	0,3508
8	E	0,6500	E	0,5277	F	0,6215	E	0,3387



Şekil 4.15. Şehir dışı senaryosu için SoC bazlı seçilen ilk istasyonların Güç-Fiyat dağılımı

Şekil 4.15'te her yöntemin şehir dışı senaryosunda SoC bazlı seçtiği ilk istasyonların Güç-Fiyat dağılımı paylaşılmaktadır.



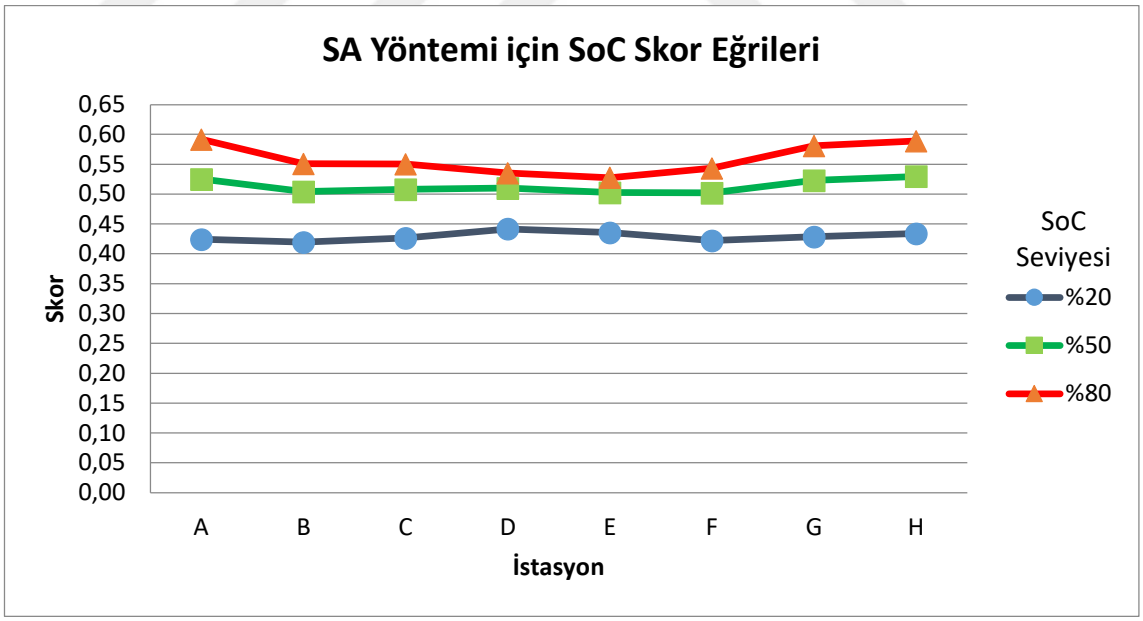
Şekil 4.16. Şehir dışı senaryosunda LP yöntemi için SoC skor eğrileri

Şekil 4.16'da şehir dışı senaryosunda LP yöntemi için SoC skor eğrileri paylaşılmaktadır.

%20 SoC seviyesinde LP algoritması, orta derecede yakınlık ve göreli olarak daha yüksek güç verimliliğine sahip istasyonları önceliklendirmektedir. Bu bağlamda, İstasyon H en yüksek skorla ilk sırada yer almakta; onu A ve C istasyonları takip etmektedir. Bu durum, LP'nin acil şarj ihtiyacı durumlarında mesafe, maliyet ve güç erişimi arasında dengeli bir değerlendirme yaptığına işaret etmektedir.

%50 SoC seviyesinde LP algoritması, istasyon tercihi açısından tutarlılığını korumakta; yine İstasyon H en yüksek skoru elde etmiş ve bu durum, aciliyetin azaldığı ve seçim kriterlerinin daha esnek hale geldiği bu seviyede LP'nin değerlendirme etkinliğinin arttığını göstermektedir.

%80 SoC seviyesinde LP algoritması, yüksek SoC seviyelerinde düşük aciliyet nedeniyle LP'nin optimal güç-fiyat oranına sahip istasyonları güçlü biçimde tercih etme eğilimini yeniden teyit etmektedir. A ve G istasyonlarının da yüksek skorlar almaya devam etmesi, farklı SoC seviyelerinde sıralama eğilimlerinin istikrarlı olduğunu göstermektedir.



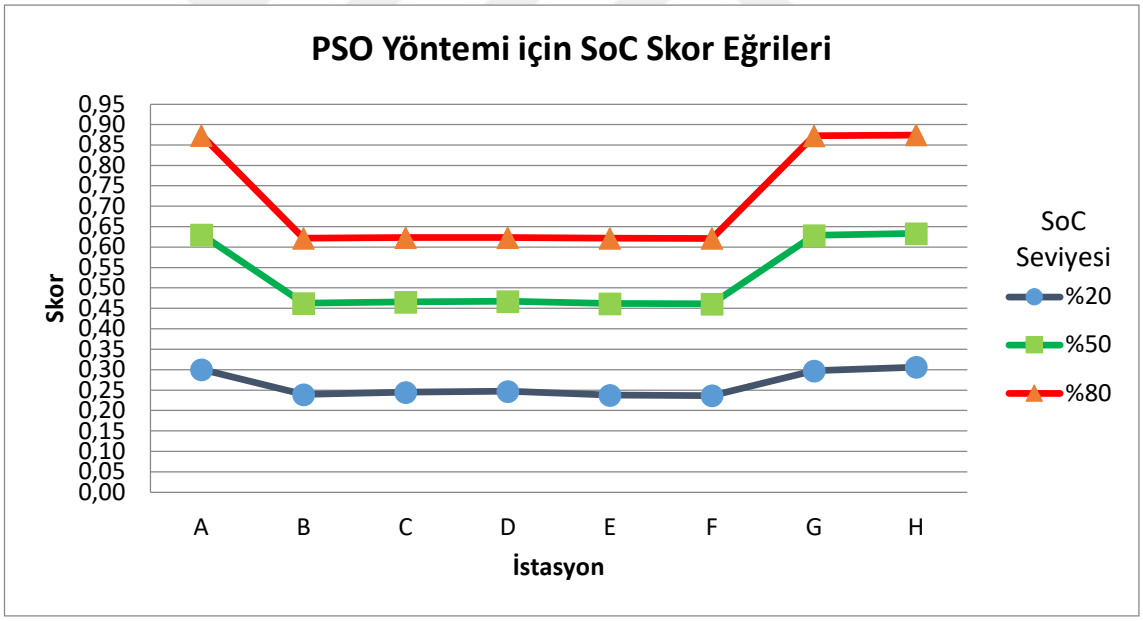
Şekil 4.17. Şehir dışı senaryosunda SA yöntemi için SoC skor eğrileri

Şekil 4.17'de şehir dışı senaryosunda SA yöntemi için SoC skor eğrileri paylaşılmaktadır.

%20 SoC seviyesinde SA algoritması, yakınlığı güçlü şekilde önceliklendirmekte ve İstasyon D'yi birinci istasyon olarak belirlemektedir. Bu durum, SA'nın acil şarj gereksinimlerinde mesafe faktörünü ön planda tuttuğunu ve bunun sonucunda güç veya maliyet faktörlerinde kısmi ödünler verebildiğini göstermektedir.

%50 SoC seviyesinde SA algoritması, tercihini daha yüksek güçlü istasyonlara (örneğin İstasyon H) kaydırmakta ve bu yönüyle LP sonuçlarıyla daha fazla örtüşmektedir. Bu eğilim, SA'nın aciliyetin azaldığı durumlarda verimlilik ile öncelik arasında denge kurabildiğini ortaya koymaktadır.

%80 SoC seviyesinde SA algoritması, dikkat çekici bir biçimde İstasyon A'yı öncelikli olarak seçmekte ve bu tercih sürekli olarak İstasyon H'yi tercih eden diğer yöntemlerden ayrışmaktadır. Bu farklılık SA'nın daha geniş çözüm alanlarını keşfetme ve yüksek SoC düzeylerinde alternatif istasyonları değerlendirme potansiyelini ortaya koymaktadır.



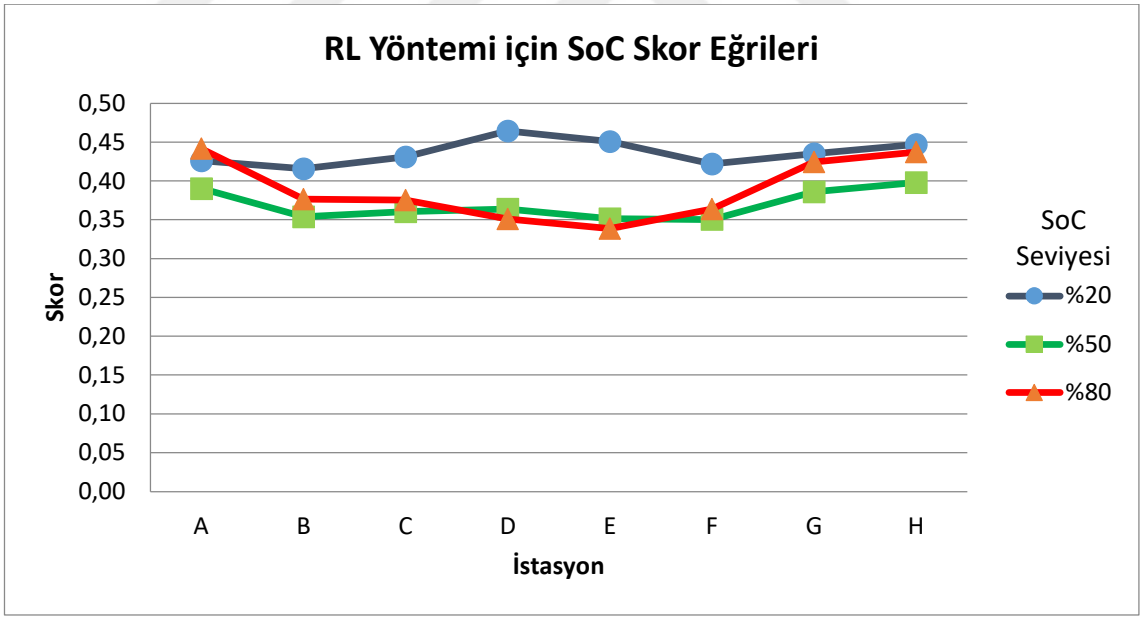
Şekil 4.18. Şehir dışı senaryosunda PSO yöntemi için SoC skor eğrileri

Şekil 4.18'de şehir dışı senaryosunda PSO yöntemi için SoC skor eğrileri paylaşılmaktadır.

%20 SoC seviyesinde PSO algoritması, İstasyon H'yi birinci istasyon olarak tercih etmekte ve bu durum, LP'ye benzer şekilde dengeli bir yaklaşımı yansıtmaktadır; ancak PSO'nun acil şarj erişimini bir nebze daha önceliklendirdiği gözlemlenmektedir. A ve G istasyonları da yüksek skorlarla yakın tercih seçenekleri olarak öne çıkmaktadır; bu da PSO'nun orta düzeyde güçlü ve erişilebilir istasyonlara yönelimini desteklemektedir.

%50 SoC seviyesinde PSO algoritması, İstasyon H'yi en yüksek skorla belirgin şekilde tercih etmekte ve bu söz konusu SoC düzeyinde tüm yöntemler arasında en yüksek göreli skoru temsil etmektedir. Bu durum, PSO'nun verimli güç-fiyat konfigürasyonlarına olan tutarlı eğilimini ortaya koymaktadır.

%80 SoC seviyesinde PSO algoritması, İstasyon H'ye yönelik tercihini daha da belirginleştirmekte ve bu durum aciliyetin azaldığı koşullarda maliyet etkinliği ile istasyon gücünün optimum kombinasyonunu hedeflediğini göstermektedir.



Şekil 4.19. Şehir dışı senaryosunda RL yöntemi için SoC skor eğrileri

Şekil 4.19'da şehir dışı senaryosunda RL yöntemi için SoC skor eğrileri paylaşılmaktadır.

%20 SoC seviyesinde RL algoritması, öncelikli olarak yakın istasyonları tercih etmekte ve özellikle İstasyon D'yi en yüksek skorla ön plana çıkarmaktadır. Bu tercih, RL'nin kısa vadeli ödülleri maksimize etme stratejisini yansıtmakta; güç veya fiyat farklılıklarına rağmen daha hızlı erişim potansiyelini önceliklendirdiğini göstermektedir.

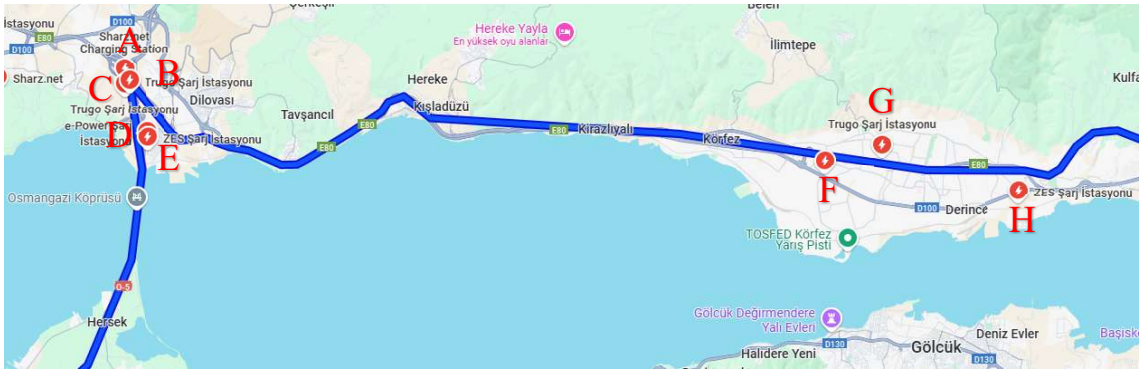
%50 SoC seviyesinde RL algoritması, diğer yöntemlerden belirgin şekilde ayrışarak İstasyon H, A ve G'yi tercih etmekte; ancak bu istasyonlara verdiği skorlar görece daha düşüktür. Bu durum, RL'nin daha temkinli ve artımlı bir değerlendirme stratejisi izlediğini ortaya koymaktadır.

%80 SoC seviyesinde RL algoritması, beklenmedik şekilde İstasyon A'yı en iyi seçenek olarak sıralamakta ve LP ile PSO'dan ayrışmaktadır.

Genel olarak değerlendirildiğinde, şehir dışı senaryosunda, LP ve PSO yöntemleri, tüm SoC seviyelerinde yüksek güç ve dengeli fiyatlandırmaya sahip istasyonları sistematik şekilde tercih ederek küresel ölçekte optimal çözümlere yönelik yapılandırılmış bir yaklaşım sergilemektedir.

SA, esnekliğini ortaya koyarak düşük SoC düzeylerinde yakınlığı önceliklendirmekte, ancak SoC yükseldikçe LP ve PSO tercihleriyle daha fazla örtüşmektedir.
RL ise, ödül dayalı kriterleri sürekli olarak uygulamakta ve istasyon sıralamalarında kendine özgü sonuçlar üretmektedir.

Uzun Yol Senaryosu



Şekil 4.20. Uzun yol senaryosu için rota üzerinde seçilen istasyonların konumu

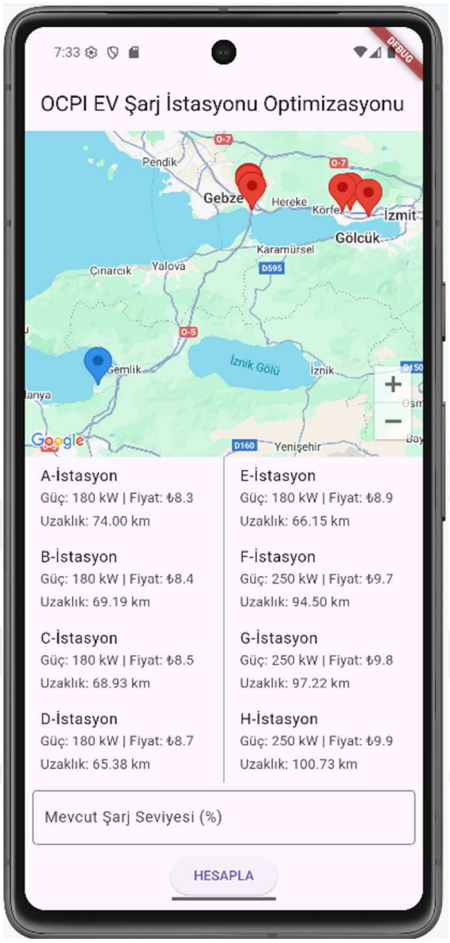
Uzun yol senaryosu için Google Maps programı aracılığıyla Togg Teknoloji Kampüsü ve Ankara arasında bir rota oluşturulmuş ve bu rota üzerinden 8 adet EV şarj istasyonu belirlenmiştir.

Çizelge 4.9. Uzun yol senaryosu için belirlenen istasyonların bilgileri

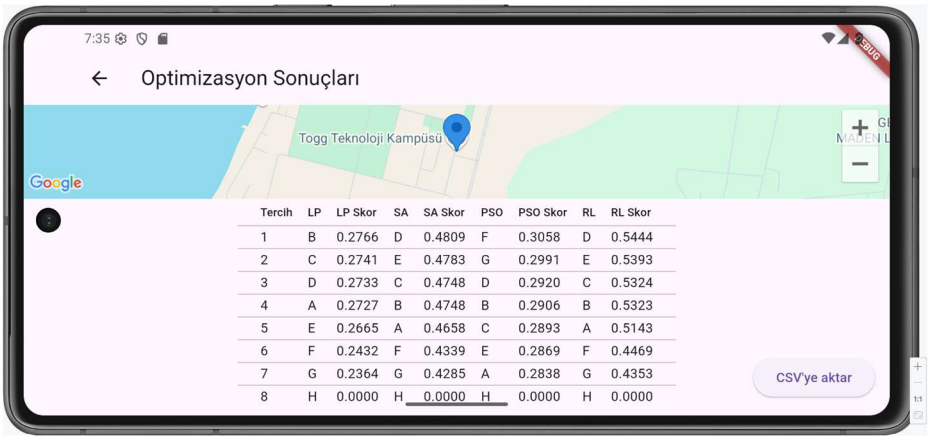
İstasyon Adı	Enlem	Boylam	Güç (kw)	Fiyat (₺)
A	40,793308	29,511025	180	8,3
B	40,790576	29,512613	180	8,4
C	40,789046	29,510881	180	8,5
D	40,775110	29,519541	180	8,7
E	40,773853	29,520047	180	8,9
F	40,770563	29,776587	250	9,7
G	40,772995	29,798656	250	9,8
H	40,759290	29,850612	250	9,9

Çizelge 4.9'da belirlenen 8 istasyonun enlem, boylam, güç ve tarife bilgileri diğer senaryolarda da yapıldığı gibi "istasyon_konumlari.json" dosyasına girilmiştir. Uzun yol senaryosu kullanıcı konumu için seçilen Togg Teknoloji Kampüsü konumu simülasyon cihazında aktifleştirilmiştir. Bu işlemlerin ardından, Android Studio

Terminal arayüzünden yazılıma şehir dışı senaryosunda da detaylı olarak anlatılan Hot Restart işlemi yapılmıştır.



Şekil 4.21. Uzun yol senaryosu için uygulama ana ekranı



Şekil 4.22. Uzun yol senaryosu için optimizasyon sonuçları ekranı (SoC %20 iken)

Uzun yol senaryosunda elde edilen sonuçlar Çizelge 4.10, Çizelge 4.11 ve Çizelge 4.12'de paylaşılmaktadır.

Çizelge 4.10. Uzun yol senaryosunda %20 SoC için sonuçlar

Tercih	LP	LP Skor	SA	SA Skor	PSO	PSO Skor	RL	RL Skor
1	B	0,2766	D	0,4809	F	0,3058	D	0,5444
2	C	0,2741	E	0,4783	G	0,2991	E	0,5393
3	D	0,2733	C	0,4748	D	0,2920	C	0,5324
4	A	0,2727	B	0,4748	B	0,2906	B	0,5323
5	E	0,2665	A	0,4658	C	0,2893	A	0,5143
6	F	0,2432	F	0,4339	E	0,2869	F	0,4469
7	G	0,2364	G	0,4285	A	0,2838	G	0,4353
8	H	0,0000	H	0,0000	H	0,0000	H	0,0000

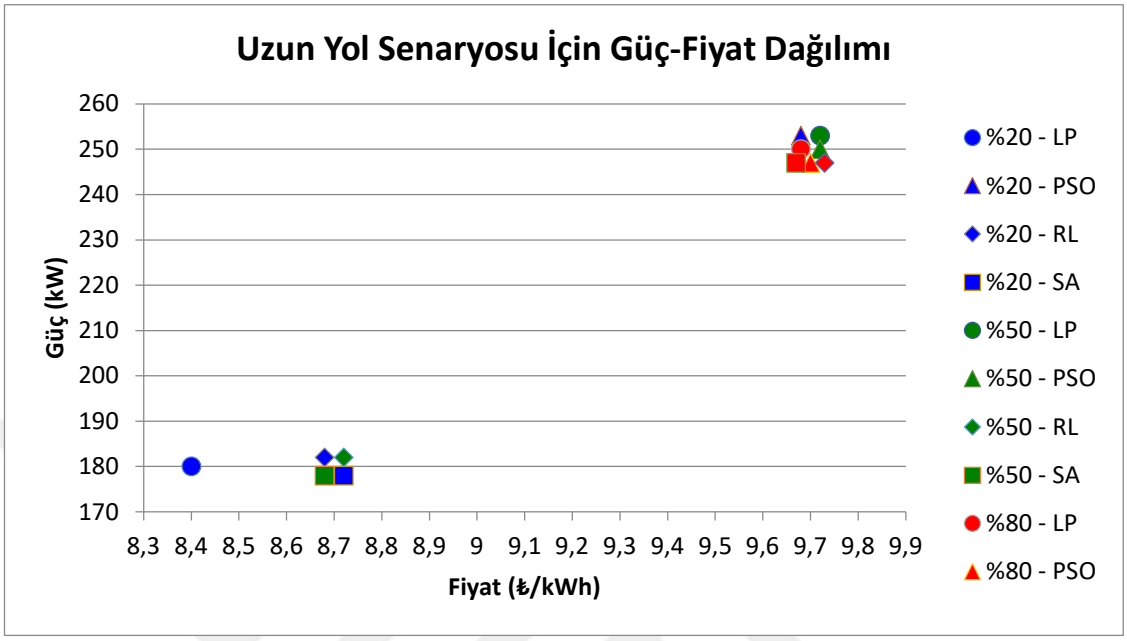
Çizelge 4.11. Uzun yol senaryosunda %50 SoC için sonuçlar

Tercih	LP	LP Skor	SA	SA Skor	PSO	PSO Skor	RL	RL Skor
1	F	0,5602	D	0,5448	F	0,6332	D	0,4236
2	G	0,5555	B	0,5437	G	0,6295	B	0,4218
3	H	0,5500	C	0,5427	H	0,6250	C	0,4200
4	B	0,4996	E	0,5408	D	0,4857	E	0,4169
5	C	0,4979	A	0,5384	B	0,4850	A	0,4126
6	D	0,4973	F	0,5292	C	0,4842	F	0,3970
7	A	0,4969	G	0,5241	E	0,4830	G	0,3881
8	E	0,4925	H	0,5179	A	0,4812	H	0,3775

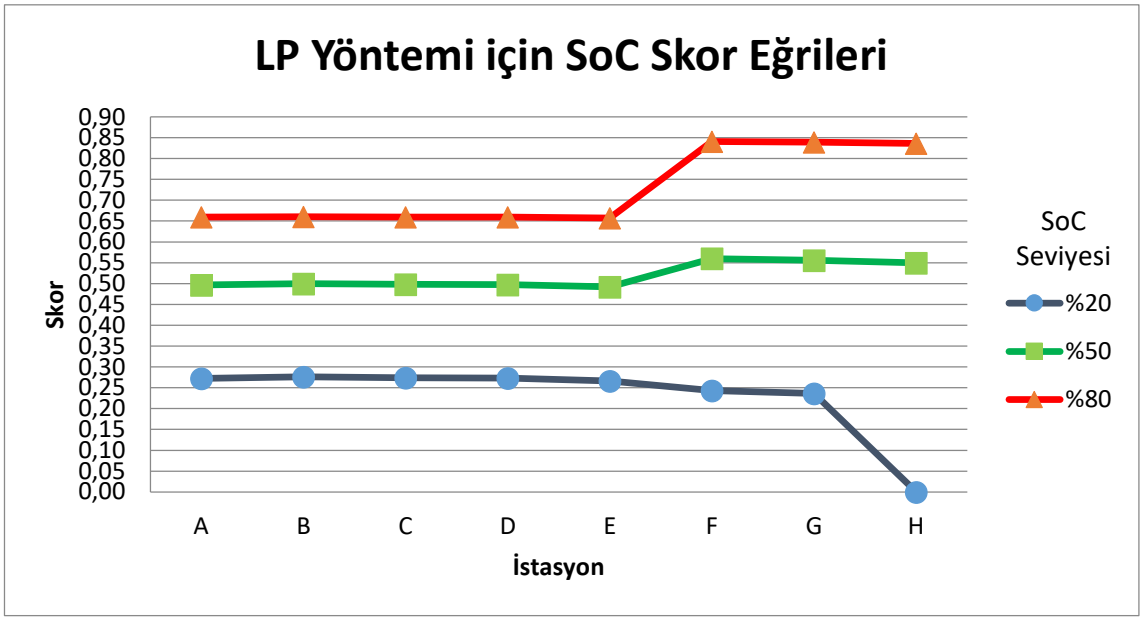
Çizelge 4.12. Uzun yol senaryosunda %80 SoC için sonuçlar

Tercih	LP	LP Skor	SA	SA Skor	PSO	PSO Skor	RL	RL Skor
1	F	0,8410	F	0,5872	F	0,8744	F	0,4346
2	G	0,8389	G	0,5831	G	0,8730	G	0,4280
3	H	0,8364	H	0,5785	H	0,8714	H	0,4207
4	B	0,6607	B	0,5664	D	0,6219	B	0,4012
5	C	0,6599	A	0,5659	B	0,6216	A	0,4005
6	D	0,6597	C	0,5641	C	0,6214	C	0,3976

7	A	0,6595	D	0,5614	E	0,6209	D	0,3932
8	E	0,6575	E	0,5562	A	0,6203	E	0,3848



Şekil 4.23. Uzun yol senaryosu için SoC bazlı seçilen ilk istasyonların Güç-Fiyat dağılımı



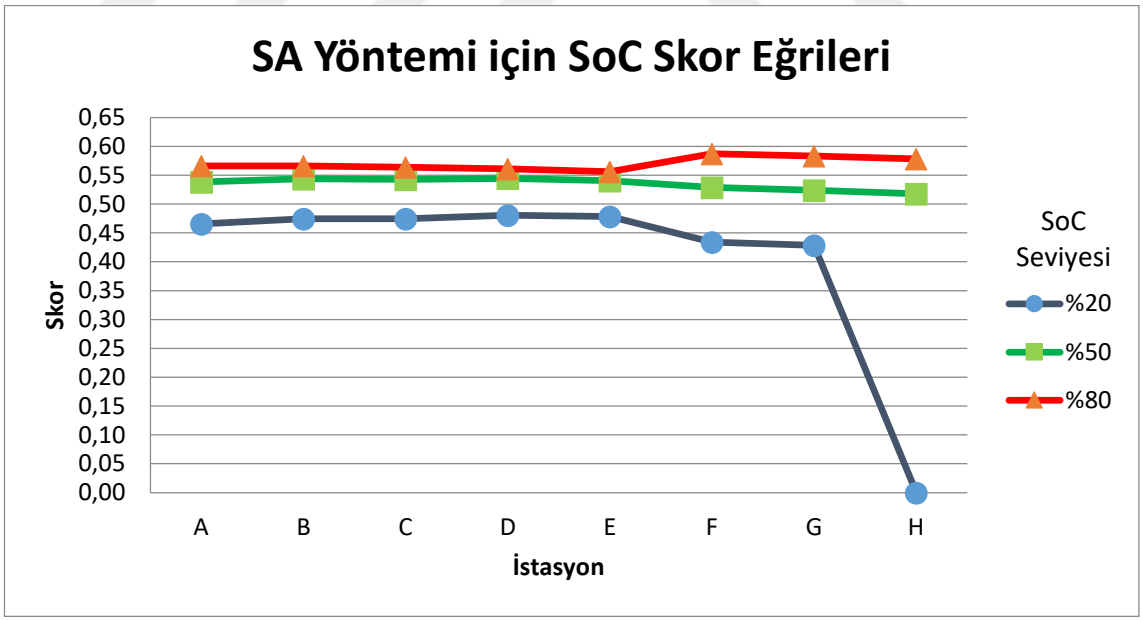
Şekil 4.24. Uzun yol senaryosunda LP yöntemi için SoC skor eğrileri

Şekil 4.24'te uzun yol senaryosunda LP yöntemi için SoC skor eğrileri paylaşılmaktadır.

%20 SoC seviyesinde LP algoritması, orta mesafeli ve maliyet-etkin istasyonları önceliklendirmekte; İstasyon B en yüksek skorla tercih edilmektedir. Bu durum, LP'nin uzun yolculuklardaki acil şarj ihtiyaçlarında yakınlık, yüksek güç seviyesi (180 kW) ve makul tarife arasında denge kurma eğilimini göstermektedir.

%50 SoC seviyesinde LP algoritması, mesafeye rağmen daha yüksek güce sahip istasyonlara yönelmekte ve İstasyon F (250 kW) en yüksek skorla ön plana çıkmaktadır. Bu durum, uzun mesafeli seyahatlerde duraklama sürelerini minimize etmek için hızlı şarjı önceliklendiren stratejik bir eğilimi yansıtmaktadır.

%80 SoC seviyesinde LP algoritması, daha uzak mesafelerde yer alan yüksek güçlü istasyonlara güçlü biçimde yönelmekte; İstasyon F en yüksek skoru almaktadır. G ve H istasyonları da yüksek sıralarda yer almakta olup, bu durum LP'nin azalan aciliyet düzeyinde şarj verimliliği ve bekleme süresini azaltma yönündeki yaklaşımını açıkça göstermektedir.



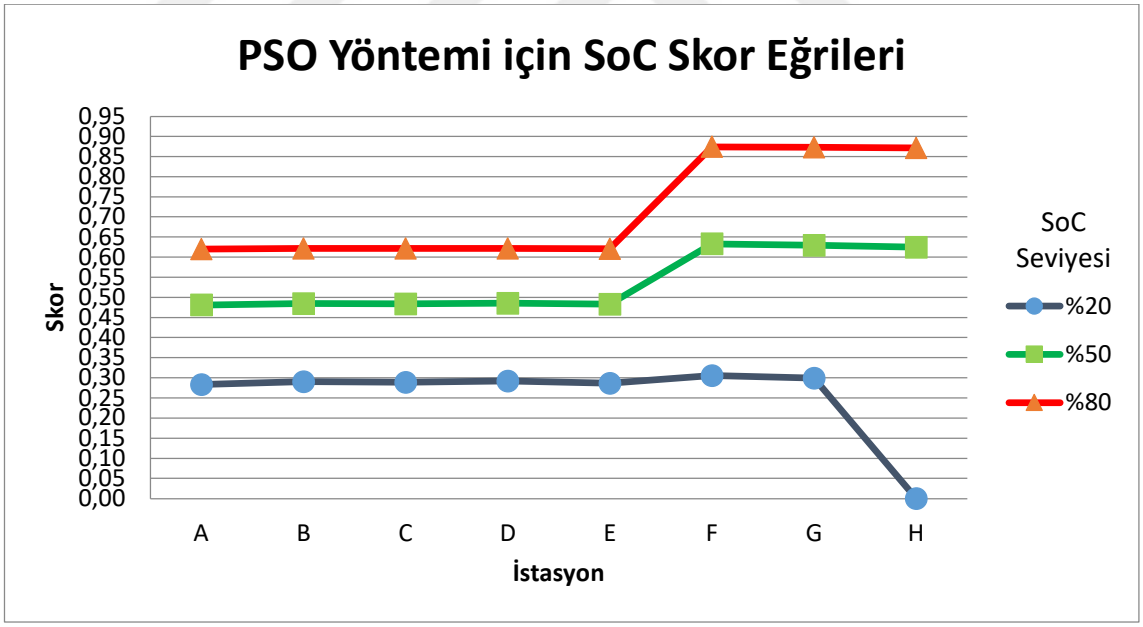
Şekil 4.25. Uzun yol senaryosunda SA yöntemi için SoC skor eğrileri

Şekil 4.25'te uzun yol senaryosunda SA yöntemi için SoC skor eğrileri paylaşılmaktadır.

%20 SoC seviyesinde SA algoritması, yeterli güce sahip yakın istasyonları belirgin şekilde önceliklendirerek İstasyon D'yi seçmektedir. Bu yaklaşım, yön sapmalarını en aza indirerek acil şarj ihtiyacını karşılamayı ve aynı zamanda yeterli güç seviyesini sağlamayı amaçlamaktadır.

%50 SoC seviyesinde SA algoritması, yine yüksek güç verimliliğine sahip yakın istasyonlara yönelmekte ve İstasyon D'yi en yüksek skorla tercih etmektedir. Bu da SA'nın tutarlı biçimde kısa mesafe önceliğini sürdürdüğünü ve kabul edilebilir şarj hızını esas aldığını göstermektedir.

%80 SoC seviyesinde SA algoritmasının tercihi dikkat çekici biçimde değişmekte ve daha yüksek güce sahip, ancak daha uzakta bulunan İstasyon F'ye yönelmektedir. Bu değişim, SA'nın aciliyetin azaldığı durumlarda daha yüksek şarj kapasitelerini önceliklendiren uyarlanabilir bir strateji izlediğini göstermektedir.



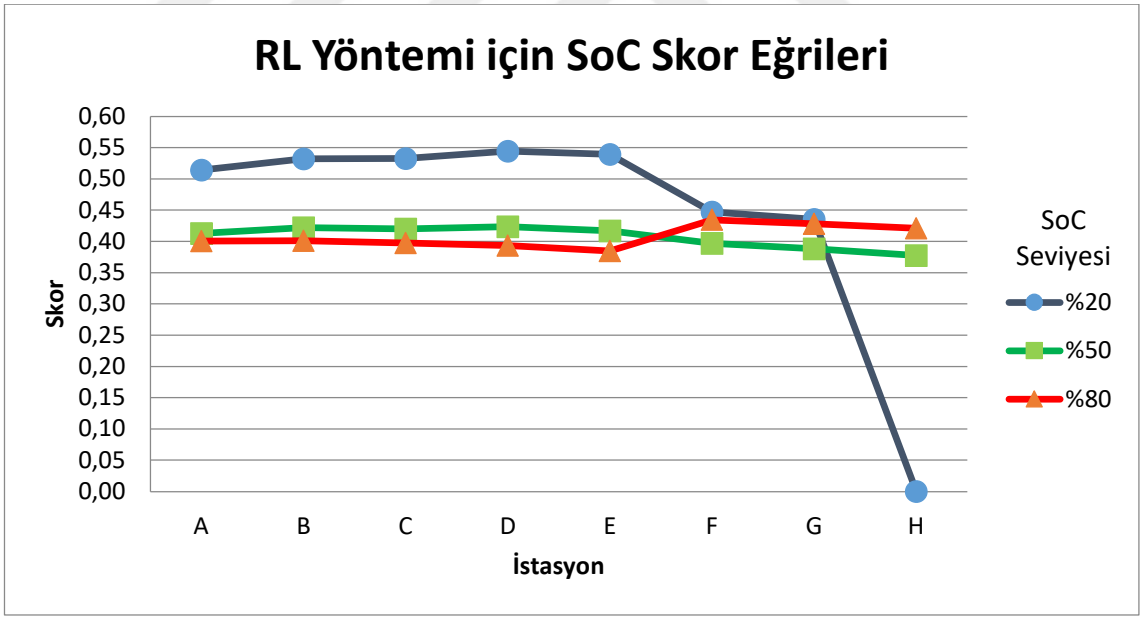
Şekil 4.26. Uzun yol senaryosunda PSO yöntemi için SoC skor eğrileri

Şekil 4.26'da uzun yol senaryosunda PSO yöntemi için SoC skor eğrileri paylaşılmaktadır.

%20 SoC seviyesinde PSO algoritması, uzak mesafedeki güçlü istasyonlara net biçimde yönelerek İstasyon F'yi en yüksek skorla seçmektedir. Bu durum, PSO'nun düşük batarya seviyelerinde dahi genel seyahat süresi ve şarj zamanını minimize etmeye yönelik erken stratejik vurgusunu yansıtmaktadır.

%50 SoC seviyesinde PSO algoritması, yüksek güç kapasitesine sahip istasyonları açık biçimde önceliklendirmekte; İstasyon F, ardından G ve H istasyonları sıralanmaktadır. Bu durum, PSO'nun mesafe ve tarife artışlarına rağmen uzun yolculuklar için şarj hızını maksimize etmeye devam ettiğini göstermektedir.

%80 SoC seviyesinde PSO algoritması, yüksek tarifeli yüksek güçlü istasyonlara güçlü bir şekilde yönelmekte ve İstasyon F yine en yüksek skoru almaktadır. Bu durum, PSO'nun daha az acil senaryolarda bile şarj verimliliğini agresif biçimde optimize etmeye yönelik tutarlı stratejisini ortaya koymaktadır.



Şekil 4.27. Uzun yol senaryosunda RL yöntemi için SoC skor eğrileri

Şekil 4.27'de uzun yol senaryosunda RL yöntemi için SoC skor eğrileri paylaşılmaktadır.

%20 SoC seviyesinde RL algoritması, güzergaha daha yakın ve yeterli güç seviyesine sahip istasyonları belirgin şekilde tercih etmekte; İstasyon D en yüksek skorla seçilmektedir. Bu tercih, RL'nin kısa vadeli ödül odaklı karar alma sürecini ve düşük batarya seviyelerinde yön sapmalarını en aza indirme stratejisini göstermektedir.

%50 SoC seviyesinde RL algoritması, temkinli tercihlerini sürdürerek makul mesafedeki ve orta-yüksek güce sahip istasyonlara yönelmekte; İstasyon D yine en yüksek skoru almaktadır. Bu eğilim, RL'nin verimlilik ile yakınlık arasında denge arayışını yansıtmaktadır.

%80 SoC seviyesinde RL algoritması, yeterli güce ve orta uzaklığa sahip istasyonlara yönelmekte; mesafeye rağmen İstasyon F'ye kısmi bir öncelik tanımaktadır.

Genel olarak değerlendirildiğinde, uzun yol senaryosunda, LP ve PSO yöntemleri, başlangıçta yakın istasyonları önceliklendirirken, SoC düzeyi arttıkça yüksek güçlü istasyonlara yönelerek uzun yolculuklar için seyahat ve şarj süresini etkin biçimde optimize etmektedir. SA yöntemi, başlangıçta yakınlık odaklı bir strateji izlerken, aciliyetin azalmasıyla birlikte yüksek güçlü ve daha uzak istasyonları değerlendiren orta düzeyde uyarlanabilir bir yaklaşım sergilemektedir. RL yöntemi ise, tüm SoC seviyelerinde yakınlık ve yeterli güç dengesini koruyarak geleneksel bir strateji benimsemektedir.

Ayrıca geliştirilen yazılımda, belirli bir SoC seviyesinde EV'nin ulaşabileceği menzili aşan şarj istasyonlarına algoritmik olarak bilinçli şekilde sıfır skor atanmasını sağlayan bir kısıtlama uygulanmaktadır. Bu mantık, uzun yol senaryosu düşük SoC (%20) için özellikle H istasyonu özelinde açık biçimde doğrulanmıştır. H istasyonunun hesaplanan mesafesi, aracın bu SOC seviyesindeki ulaşılabilir menzilini aştığı için, tüm optimizasyon yöntemlerinde (LP, SA, PSO, RL) tutarlı şekilde sıfır skor almıştır.

5. SONUÇ

Bu tez çalışmasında, EV şarj istasyonu seçiminin optimize edilmesi konusundaki kritik sorunu, ileri düzey optimizasyon tekniklerinin uygulanması ve OCPI standartlarının benimsenmesi yoluyla ele almıştır. Çalışmanın temel motivasyonu, konuma duyarlı, veriye dayalı ve standartlara uyumlu bir karar verme çerçevesi tasarlayarak EV şarj süreçlerinde kullanıcı deneyimini ve verimliliği artırmaktır. Bu doğrultuda, Android platformları için Flutter kullanılarak Dart dilinde geliştirilen özgün bir yazılım uygulaması ortaya konmuştur. Bu yazılım uygulaması, kullanıcıya ait şarj durumu, istasyonlara ait meta veriler ve çeşitli optimizasyon algoritmalarını entegre ederek dinamik ve stratejik şarj istasyonu önerileri sunmaktadır.

Çalışmanın kuramsal temeller bölümünde, EV'ler, EV altyapısı ve OCPI protokolüne ilişkin temel kavramlar ayrıntılı şekilde incelenmiştir. OCPI standardı; mimarisi, veri yapısı ve birlikte çalışabilirlik özellikleri açısından analiz edilmiş, bu standardın yaygın kullanılan diğer şarj protokolleri üzerindeki avantajlarına vurgu yapılmıştır. Ayrıca, çalışmanın algoritmik temeline katkı sağlamak amacıyla doğrusal programlama (LP), benzetilmiş tavlama (SA), parçacık sürü optimizasyonu (PSO) ve pekiştirmeli öğrenme (RL) gibi optimizasyon tekniklerine dair kapsamlı bir literatür taraması gerçekleştirilmiştir. Bu tekniklerin, özellikle SoC düzeyine bağlı olarak değişen aciliyet durumları doğrultusunda mesafe, güç, fiyat ve kullanıcı öncelikleri gibi çeşitli operasyonel parametreleri nasıl ele aldığına özel önem verilmiştir.

Tez kapsamında tasarlanan yazılım uygulaması sistemi, yakındaki sekiz şarj istasyonundan veri toplayarak bunları dört farklı optimizasyon stratejisiyle değerlendirmektedir. Algoritmalar, her istasyon için 0 ile 1 arasında bir puan hesaplamakta ve seçenekleri sıralayarak kullanıcının bilinçli kararlar almasına olanak tanımaktadır. "Şehir Merkezi", "Şehir Dışı," ve "Uzun Yol" olmak üzere tanımlanan senaryolar sayesinde algoritmaların farklı gerçek dünya senaryolarında nasıl performans gösterdiği sistematik biçimde değerlendirilmiştir. Her senaryo ayrıca üç farklı SoC düzeyi (%20, %50 ve %80) altında analiz edilerek, değişen aciliyet durumlarının ve stratejik tercihlerinin istasyon seçiminde nasıl etkili olduğu gösterilmiştir.

Elde edilen sonuçlar, her bir optimizasyon algoritmasının kendine özgü davranış kalıplarını ortaya koymaktadır. LP yöntemi, düşük SoC düzeylerinde yakınlığa odaklı kararlar alırken, yüksek SoC seviyelerinde yüksek güçlü istasyonları önceliklendiren verimlilik odaklı tercihlere yönelerek son derece uyarlanabilir bir karar verme yapısı sergilemiştir. Bu durum, LP yönteminin çok ölçütlü karar ortamlarında bağlama duyarlı ve esnek bir yapı sunduğunu göstermektedir.

SA algoritması ise özellikle düşük SoC seviyelerinde yakın istasyonları tercih eden temkinli bir strateji izlemiş, aciliyetin azalmasıyla birlikte yüksek güçlü istasyonları da karar kümesine kademeli olarak dahil etmiştir. Rastlantısal yapısı sayesinde daha geniş çözüm uzaylarını keşfetme yeteneği kazanırken, kullanıcı odaklı önceliklerini korumayı başarmıştır.

PSO algoritması, dört yöntem arasında en saldırgan yaklaşımı benimseyerek, özellikle orta ve yüksek SoC düzeylerinde mesafe veya ücret dikkate almaksızın yüksek güçlü istasyonlara sistematik şekilde öncelik tanımıştır. Bu eğilim, PSO'nun küresel arama kapasitesi ve genel yolculuk verimliliğini en üst düzeye çıkarma amacına uygundur. Ancak, bu strateji acil durumlar veya maliyete duyarlı kullanıcılar için her zaman uygun olmayabilir.

RL algoritması ise ödül temelli öğrenme yaklaşımıyla istikrarlı ve tutarlı kararlar vermeyi hedeflemiştir. Tüm senaryolarda ve SoC seviyelerinde yakın ve orta güçlü istasyonlara yönelme eğilimi göstermiştir. RL'nin bu tutucu fakat dengeli sıralama sonuçları, anlık kazançlar yerine uzun vadeli faydayı maksimize etmeye yönelik tasarımıyla örtüşmektedir. Bu özellik, enerji maliyetlerini ve zaman kullanımını uzun vadede azaltmayı amaçlayan uygulamalarda önemli bir avantaj sunmaktadır.

Karşılaştırmalı olarak değerlendirildiğinde, hiçbir algoritmanın tüm koşullarda diğerlerinden mutlak üstünlük göstermediği ortaya konmuştur. Bunun yerine, her bir yöntemin belirli kullanıcı önceliklerine ve durumsal gerekliliklere uygun özgün avantajlar sunduğu sonucuna varılmıştır. LP ve PSO, özellikle yüksek kapasiteli şehir

merkezinde hız ve verimlilik arayan kullanıcılarla örtüşürken; SA ve RL, şehir dışı bölgelerde veya düşük altyapılı alanlarda güvenilirlik, tutarlılık ve erişilebilirliğin öncelikli olduğu senaryolarda tercih edilebilir (All Sciences Academy, 2025).

OCPI standardının entegrasyonu, önerilen çözümün birlikte çalışabilirlik ve ölçeklenebilirlik özelliklerini güvence altına alarak yazılımın şarj ağlarıyla sorunsuz şekilde iletişim kurabileceği gözlemlenmiştir. Bu uyumluluk, uygulamanın gerçek dünya EV ekosistemlerinde uygulanabilirliğini ve gelecekteki genişletilebilirliğini önemli ölçüde artırmaktadır.

Bu tez akademik açıdan, EV şarj altyapısı bağlamında farklı optimizasyon tekniklerini karşılaştırmalı olarak değerlendiren bir çerçeve ortaya koymuştur. Yöntemsel olarak, çok senaryolu ve çok seviyeli bir değerlendirme yaklaşımı önererek geniş kapsamlı gerçek dünya dinamiklerini yakalamayı başarmıştır. Pratik düzeyde ise, EV kullanıcılarına akıllı ve optimize edilmiş şarj kararları vermelerinde yardımcı olabilecek işlevsel ve taşınabilir bir Android uygulaması geliştirilmiştir.

Gelecekte bu çalışmanın çeşitli yönlerden genişletilmesi mümkündür. İlk olarak, optimizasyon çerçevesine gerçek zamanlı trafik koşulları, istasyonlardaki kuyruk uzunlukları ve mevcutluk bilgileri eklenerek karar doğruluğu artırılabilir. İkinci olarak, kullanıcı tercihleri (örneğin maliyet hassasiyeti, zaman önceliği, tercih edilen şebekeler) sisteme dahil edilerek kişiselleştirilmiş öneriler sunulabilir. Üçüncü olarak, çevrimiçi pekiştirmeli öğrenme yöntemleriyle kullanıcı etkileşimlerinden sürekli öğrenme sağlanabilirken gizlilik korunabilir. Ayrıca, ulusal ve uluslararası OCPI ağlarıyla daha geniş çaplı entegrasyon, bu uygulamanın akıllı mobilite sistemleri için güçlü bir karar destek aracına dönüşmesini sağlayabilir. Son olarak, yazılımın iOS platformuna taşınması ve araç bilgi-eğlence sistemlerine entegre edilmesi, kullanıcı erişimini ve pratik uygulanabilirliği önemli ölçüde artırabilir.

KAYNAKLAR

ABD Enerji Departmanı. (2014). *The History of the Electric Car.* Energy.gov; Department of Energy.https://www.energy.gov/articles/history-electric-car

All Sciences Academy. (2025). *Optimization of Electric Vehicle Charging Stations Using OCPI Standards*. Zenodo. https://doi.org/10.5281/zenodo.15548999

Ampeco. (2025). *The Complete OCPI Guide*. https://www.ampeco.com/guides/the-complete-ocpi-guide/

Anderson, C. D., & Anderson, J. (2010). Electric and hybrid cars: A history. McFarland.

Angelini, M. C., & Ricci-Tersenghi, F. (2023). Limits and performances of algorithms based on simulated annealing in solving sparse hard inference problems. *Physical Review X*, *13*(2), 021011. https://doi.org/10.1103/PhysRevX.13.021011

Acharige, S. S., Haque, M. E., Arif, M. T., Hosseinzadeh, N., Hasan, K. N., & Oo, A. M. T. (2023). Review of electric vehicle charging technologies, standards, architectures, and converter configurations. *IEEE access*, *11*, 41218-41255. 10.1109/ACCESS.2023.3267164

Bazaraa, M. S., Sherali, H. D., & Shetty, C. M. (2013). *Nonlinear Programming: Theory and Algorithms*. John Wiley & Sons.

Baseflow (2025). *Geolocator for Flutter.* https://pub.dev/packages/geolocator

Bellman, R. (1958). Dynamic programming and stochastic control processes. *Information and control*, *1*(3), 228-239. https://doi.org/10.1016/S0019-9958(58)80003-0

Boyd, S. P., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

Dart Team (2025). *dart:convert Library.* https://api.dart.dev/stable/dart-convert/dart-convert-library.html

Dart Team (2025). *dart:math Library.* https://api.dart.dev/stable/dart-math/dart-math-library.html

Desai, M., & Shah, S. (2024). Energy management through V2G concept with EV charging/discharging strategy. Journal of Electrical Systems, 20(11s), 4774–4785. 10.52783/jes.8736

Dorigo, M., Maniezzo, V., & Colorni, A. (1996). Ant system: optimization by a colony of cooperating agents. *IEEE transactions on systems, man, and cybernetics, part b (cybernetics)*, *26*(1), 29-41. 10.1109/3477.484436

Ehsani, M., Gao, Y., Longo, S., & Ebrahimi, K. (2018). *Modern electric, hybrid electric, and fuel cell vehicles*. CRC press.https://doi.org/10.1201/9780429504884

ElaadNL (2017) EV Related Protocol Study v1.1. Technical report, ElaadNL.

Enerji Ajansı. (2025). *Türkiye'de şarj istasyonu sayısı*. https://enerjiajansi.com.tr/turkiyede-sarj-istasyonu-sayisi/

EVRoaming Foundation. (2023). *Open charge point interface protocol (OCPI 2.2.1)*. https://evroaming.org/app/uploads/2023/04/OCPI-2.2.1-specification-final.pdf

Flutter Community. (2025). *path_provider: Flutter package*. https://pub.dev/packages/path_provider

Gan, L., Topcu, U., & Low, S. H. (2013). Optimal decentralized protocol for electric vehicle charging. *IEEE Transactions on Power Systems*, *28*(2), 940-951. 10.1109/TPWRS.2012.2210288

Gnann, T., Stephens, T. S., Lin, Z., Plötz, P., Liu, C., & Brokate, J. (2018). What drives the market for plug-in electric vehicles?-A review of international PEV market

diffusion models. *Renewable and Sustainable Energy Reviews*, *93*, 158-164. https://doi.org/10.1016/j.rser.2018.03.055

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley.

Google. (2025). *Android Studio user guide*. https://developer.android.com/studio

Google. (2025). *Dart programming language overview*. https://dart.dev

Google. (2025). *Flutter documentation*. https://flutter.dev

Google. (2025). *Google Maps Flutter: Flutter package*. https://pub.dev/packages/google_maps_flutter

Google. (2025). *Material widgets – Flutter*. https://api.flutter.dev/flutter/material/material-library.html

Greenflux. (2025). *Roaming protocols: OCPI, OICP, OCHP, and eMIP*. https://www.greenflux.com/expertise/blogs/roaming-protocols-ocpi-oicp-ochp-and-emip/

Guillemin, S., Choulet, R., Guyot, G., & Hing, S. (2024). Electrical Vehicle Smart Charging Using the Open Charge Point Interface (OCPI) Protocol. *Energies*, *17*(12), 2873. https://doi.org/10.3390/en17122873

Hajimiragha, A., Canizares, C. A., Fowler, M. W., & Elkamel, A. (2010). Optimal transition to plug-in hybrid electric vehicles in Ontario, Canada, considering the electricity-grid limitations. *IEEE Transactions on Industrial Electronics*, *57*(2), 690-701. 10.1109/TIE.2009.2025711

Holland, J. H. (1975). *Adaptation in natural and artificial systems*. MIT Press.

Hu, J., Jiang, X., & Zheng, L. (2018). Design and analysis of hybrid electric vehicle powertrain configurations considering energy transformation. *International Journal of Energy Research*, *42*(15), 4719-4729. https://doi.org/10.1002/er.4225

Hu, J., Morais, H., Sousa, T., & Lind, M. (2016). Electric vehicle fleet management in smart grids: A review of services, optimization and control aspects. *Renewable and Sustainable Energy Reviews*, *56*, 1207-1226. https://doi.org/10.1016/j.rser.2015.12.014

IDACS Consortium. (2022). *IDACS report: Deliverable 2.2.5 – Recommendations on data access for public charging infrastructure*. https://english.rvo.nl/sites/default/files/2023-07/20220630_IDACSActivity_2_Deliverable_2.2.5.pdf

International Energy Agency. (2024). *Global EV outlook 2024*. https://www.iea.org/reports/global-ev-outlook-2024/

Keeney, R. L., & Raiffa, H. (1993). *Decisions with multiple objectives: preferences and value trade-offs*. Cambridge university press.

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of ICNN'95-international conference on neural networks* (Vol. 4, pp. 1942-1948). 10.1109/ICNN.1995.488968

Kirsch, D. A. (2021). *The Electric Vehicle and the Burden of History*. Rutgers University Press. 0-8135-2809-7

Kusuma, I., Ruliyanta, Kusumoputro, R. A. S., & Iswadi, A. (2025). Electric Vehicle Review: BEV, PHEV, HEV, or FCEV?. *Jurnal Konversi Energi Dan Manufaktur*, *10*(1), 70–83. https://doi.org/10.21009/JKEM.10.1.8

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction* (2nd ed.). MIT Press.

Tan, K. M., Ramachandaramurthy, V. K., & Yong, J. Y. (2017). Integration of electric vehicles in smart grid: A review on vehicle to grid technologies and optimization techniques. *Renewable and Sustainable Energy Reviews*, *53*, 720-732. https://doi.org/10.1016/j.rser.2015.09.012

Triantaphyllou, E., & Triantaphyllou, E. (2000). *Multi-criteria decision making methods* (pp. 5-21). Springer Us.

Wu, S., & Pang, A. (2023). Optimal scheduling strategy for orderly charging and discharging of electric vehicles based on spatio-temporal characteristics. *Journal of Cleaner Production*, *392*, 136318. https://doi.org/10.1016/j.jclepro.2023.136318

Zhao, Q., Wu, W., Cheng, N., Guo, G., & Han, Y. (2025). A Deep Reinforcement Learning-Based Multi-Objective Charging Scheduling Method for Electric Vehicle Charging Station. *IEEE Internet of Things Journal*. 10.1109/JIOT.2025.3567097