



T.C.
KONYA TEKNİK ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ



**İMAR UYGULAMALARINDA DAĞITIM VE
PARSELASYON İŞLEMLERİNİN YAPAY ZEKA
OPTİMİZASYON ALGORİTMALARI KULLANILARAK
GERÇEKLEŞTİRİLMESİ**

İsmail KOÇ

DOKTORA TEZİ

Bilgisayar Mühendisliği Anabilim Dalı

Kasım-2020
KONYA
Her Hakkı Saklıdır

TEZ KABUL VE ONAYI

İsmail KOÇ tarafından hazırlanan “İmar Uygulamalarında Dağıtım ve Parselasyon İşlemlerinin Yapay Zeka Optimizasyon Algoritmaları Kullanılarak Gerçekleştirilmesi” adlı tez çalışması 12/11/2020 tarihinde aşağıdaki jüri tarafından oy birliği ile Konya Teknik Üniversitesi Lisansüstü Eğitim Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı’nda DOKTORA TEZİ olarak kabul edilmiştir.

Jüri Üyeleri

İmza

Başkan (Danışman)

Doç. Dr. İsmail BABAOĞLU

.....

Üye

Prof. Dr. Tayfun ÇAY

.....

Üye

Prof. Dr. Oğuz FINDIK

.....

Üye

Doç. Dr. Mustafa Servet KIRAN

.....

Üye

Dr. Öğr. Üyesi Yüksel ÇELİK

.....

Yukarıdaki sonucu onaylarım.

Prof. Dr. Saadettin Erhan KESEN
Enstitü Müdürü

TEZ BİLDİRİMİ

Bu tezdeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edildiğini ve tez yazım kurallarına uygun olarak hazırlanan bu çalışmada bana ait olmayan her türlü ifade ve bilginin kaynağına eksiksiz atıf yapıldığını bildiririm.

DECLARATION PAGE

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.



İsmail KOÇ

Tarih:
24/11/2020

ÖZET

DOKTORA TEZİ

İMAR UYGULAMALARINDA DAĞITIM VE PARSELASYON İŞLEMLERİNİN YAPAY ZEKA OPTİMİZASYON ALGORİTMALARI KULLANILARAK GERÇEKLEŞTİRİLMESİ

İsmail KOÇ

Konya Teknik Üniversitesi
Lisansüstü Eğitim Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Doç. Dr. İsmail BABAOĞLU

2020, 134 sayfa

Jüri

Doç. Dr. İsmail BABAOĞLU

Prof. Dr. Tayfun ÇAY

Prof. Dr. Oğuz FINDIK

Doç. Dr. Mustafa Servet KIRAN

Dr. Öğr. Üyesi Yüksel ÇELİK

İmar uygulamalarındaki dağıtım ve parselasyon (LR – *Land redistribution and readjustment*) problemleri, birçok farklı kriterleri içerisinde barındıran karmaşık ve zor gerçek dünya problemleridir. Bu problemlerin çözülebilmesi için öncelikle ön parselasyon işlemlerinin başarıyla gerçekleştirilmesi gerekmektedir. Problemin çözümü için oluşturulan başlangıç popülasyonunda rastgele aday çözüm üretilirken veya çözümler üzerinde değişiklik yapılırken her aşamada parsellerin alanı güncellenmekte ve buna bağlı olarak da her bir parselin konumunun bir başka deyişle sınır noktalarının yeniden güncellenmesi gerekmektedir. Bu tez çalışmasında, imar mevzuatı ve örnek bir imar planı dikkate alınarak, Konya'nın bir bölgesi üzerinde ön parselasyon işlemi üç farklı şekilde otomatik olarak gerçekleştirilmektedir. Birincisi kenar uzunluklarına göre parselasyon, ikincisi alan değerlerine göre parselasyon ve üçüncüsü hem kenar hem de alan değerleri birlikte kullanılarak gerçekleştirilen parselasyon. Tüm bu parselasyon işlemleri için parselin alanı Gauss yöntemi kullanılarak hesaplanmaktadır. Ayrıca parselasyon işlemlerinde her bir parselin kenar noktalarını doğru bir şekilde belirlemek ve alanını hesaplamak için tüm yöntemlerde ikili arama tekniği kullanılmaktadır. Deneysel sonuçlar, ön parselasyon işlemlerinin çok hızlı ve başarılı bir şekilde gerçekleştirildiğini göstermektedir.

LR problemleri yapısı itibarıyla çizelgeleme ve gezgin satıcı problemlerine benzerlik gösteren ayrık optimizasyon problemleridir. Ayrıca bu problemler eş zamanlı olarak çözülmesi gereken çok fazla kriterle sahiptir. Bundan dolayı bu problemlerin çözümü klasik yöntemler yerine yapay zeka optimizasyon algoritmaları gerektirmektedir. Bu problemleri çözmek için ilk olarak, algoritmaların doğru karar vermesini ve aday çözümler arasında çözümleri objektif olarak değerlendirilmesini sağlayacak bir amaç fonksiyona ihtiyaç duyulmaktadır. LR problemlerinin çözümünde temel kıstas olacak bu amaç fonksiyon sayesinde, dağıtım ve parselasyon planlarının kalitesi herhangi bir uzmana ihtiyaç duyulmadan değerlendirilebilmesi ve karşılaştırılabilmesi sağlanmaktadır. Bu tez çalışmasında tüm kriterleri dikkate alan bir amaç fonksiyon önerilmektedir. Ayrıca bu çalışmada, literatürdeki çaprazlama tekniklerinden farklı olarak, klasik ve zeki Parsel Tabanlı çaprazlama operatörleri olarak adlandırılan iki farklı harita tabanlı çaprazlama operatörü önerilmektedir. Önerilen amaç fonksiyonu ve çaprazlama operatörleri yardımıyla bu tez çalışmasında önerilen ayrık yapay arı koloni (ABC – *artificial bee colony*), diferansiyel gelişim (DE – *differential evolution*), genetik (GA – *genetic algorithm*), parçacık sürü optimizasyonu (PSO – *particle swarm optimization*) ve ağaç tohum (TSA – *tree seed algorithm*) algoritmaları gerçek bir proje alanı üzerinde uygulanmaktadır. Deneysel çalışmalarda sonuçlar manuel olarak elde edilen resmi sonuçlarla karşılaştırılmaktadır. Bunların yanı sıra, geliştirilen uzman sistem sayesinde imar adalarındaki boşluk ve

taşan alanlar tamamen ortadan kaldırılarak algoritmalarından elde edilen çözümler gerçek dünyada doğrudan kullanılabilir hale getirilmektedir. Deneysel sonuçlar incelendiğinde, yapay zeka algoritmaların elde ettiği tüm sonuçların hem çözüm kalitesi hem de hız açısından manuel olarak elde edilen resmi sonuçlardan çok daha etkili olduğu açıkça görülmektedir. Ayrıca PSO algoritmasının diğer algoritmalarla göre çok daha etkili ve kararlı olduğu görülmektedir. Buna ilaveten önerilen zeki parsel tabanlı çaprazlama operatörünün klasik parsel tabanlı çaprazlama operatörüne göre çok daha etkin sonuçlar elde ettiği görülmektedir.

İmar uygulamalarında gerçek dünya problemlerinin çok karmaşık bir yapıya sahip olmasından dolayı problemin uygunluk değeri tam olarak bilinmemektedir. Bu yüzden LR problemlerinde test amaçlı kullanılmak üzere sentetik bir veri seti önerilmektedir. Önerilen veri setindeki problemlerin en iyi çözümü kesin olarak bilinmektedir. Bu veri seti parsel sayısına göre 20, 40, 60, 80 ve 100 olmak üzere 5 farklı problemten oluşmaktadır. Her bir problem seti parsel başına düşen malik sayısı bakımından 1, 2, 3 ve 4 olarak 4 farklı problem içermektedir. Bu nedenle, veri seti toplamda 20 farklı problemten oluşmaktadır. Sentetik veri seti kullanılarak gerçekleştirilen deneysel sonuçlar incelendiğinde, GA yönteminin hem hız hem de performans açısından en etkili algoritma olduğu görülmektedir. ABC birkaç problemde GA'dan daha iyi sonuçlara sahip olsa da ABC performans açısından GA yönteminden sonra en başarılı ikinci algoritmadır. Fakat zaman açısından ABC, neredeyse GA kadar başarılı bir algoritmadır. Diğer yandan, DE, PSO ve TSA algoritmalarının sonuçları, çözüm kalitesi açısından birbirine benzemektedir. Sonuç olarak, deneysel çalışmalar, GA yönteminin hız, performans ve kararlılık açısından LR problemlerinin çözümünde en etkili teknik olduğunu açıkça göstermektedir.

Anahtar Kelimeler: Dağıtım ve Parselasyon, Harita tabanlı Çaprazlama, İmar uygulamaları, Sentetik Veri Seti, Uzman Sistem, Yapay Zeka Optimizasyon Algoritmaları

ABSTRACT

PhD THESIS

IMPLEMENTATION OF LAND REDISTRIBUTION AND READJUSTMENT PROCESSES IN ZONING APPLICATIONS USING ARTIFICIAL INTELLIGENCE OPTIMIZATION ALGORITHMS

İsmail KOÇ

**Konya Technical University
Institute of Graduate Studies
Department of Computer Engineering**

Advisor: Assoc. Prof. Dr. İsmail BABAOĞLU

2020, 134 Pages

Jury

Assoc. Prof. Dr. İsmail BABAOĞLU

Prof. Dr. Tayfun ÇAY

Prof. Dr. Oğuz FINDIK

Assoc. Prof. Dr. Mustafa Servet KIRAN

Assist. Prof. Dr. Yüksel ÇELİK

Land redistribution and readjustment (LR) problems in urban areas are complex and difficult real-world problems involving many different criteria. In order to solve these problems, first of all, pre-parceling operations should be carried out successfully. While any random candidate solution in the initial population which is created for the solution of the problem is generating or it is updating, the area of the parcels is updated at each iteration and accordingly, the location of each parcel, in other words, its border points must be updated again. In this dissertation study, the pre-parceling process is automatically carried out in three different ways on a region of Konya considering the zoning legislation and a sample development plan: The first is the parceling process according to edge lengths, the second is parceling according to area values and third is parceling using both edge and area values together. In all these parceling operations, the areas of the parcels are calculated using the Gauss method. Moreover, the binary search technique is used in all methods to determine the boundaries of the parcel accurately and to calculate its area in the parceling process. Experimental results show that pre-parceling processes are carried out very quickly and successfully.

LR problems are discrete optimization problems that are similar to scheduling and traveling salesman problems in terms of their nature. In addition, these problems have too many criteria to be dealt with simultaneously. Therefore, solving these problems requires artificial intelligence optimization algorithms instead of classical methods. For solving these problems, first of all, an objective function is needed that will enable the algorithms to make the right decision and evaluate the solutions objectively among the candidate solutions. Thanks to this objective function, which is the basic criterion in solving LR problems, the quality of distribution and subdivision plans can be evaluated and compared without the need for any specialist. In this study, an objective function that takes all criteria into account is proposed. In addition, in this dissertation, two different map-based crossover operators called classical and intelligent parcel-based crossover operators are proposed different from crossover techniques in the literature. The discrete artificial bee colony (ABC), differential evolution (DE), genetic (GA), particle swarm optimization (PSO) and tree seed (TSA) algorithms which are proposed in this dissertation study are implemented on a real project area with the help of the proposed objective function and these operators. The results in the experimental studies are compared with the official results obtained manually. In addition to these, thanks to the expert system developed, the gaps and overflowing areas in the urban blocks are completely eliminated and the solutions obtained by algorithms are made available in the real world. When the experimental results are examined, it is clearly seen that all the results obtained by the artificial intelligence

algorithms are much more effective than the real results obtained manually in terms of both solution quality and speed. In addition, it is seen that the PSO algorithm is much more effective and robust than other algorithms. In addition, it is seen that the proposed intelligent parcel based crossover operator achieves much more effective results than the classical parcel based crossover operator.

Due to the very complex structure of real world problems in zoning applications, the optimal value is not known exactly. Therefore, a synthetic data set is proposed to be used as a benchmarking data set in LR applications whose optimal solution of the problems is certainly known. This data set consists of 5 different problems, 20, 40, 60, 80 and 100, depending on the number of parcels. Each problem set contains 4 different problems as 1, 2, 3 and 4 in terms of the number of land owners per parcel. Therefore, the data set consists of 20 different problems in total. When the experimental results using the synthetic data set are examined, it is seen that the GA method is the most effective algorithm in terms of both speed and performance. Although ABC has better results than the GA in a few problems, ABC is the second most successful algorithm after the GA method in terms of performance. However in terms of time, ABC is an algorithm almost as successful as the genetic algorithm. On the other hand, the results of DE, PSO and TSA algorithms are similar in terms of solution quality. As a result, the experimental results clearly show that the GA method is the most effective technique in solving LR problems in terms of speed, performance and robustness.

Keywords: Artificial Intelligence Optimization Algorithms, Expert Systems, Land Redistribution and Readjustment, Map-based Crossover, Synthetic Data Set, Zoning applications

ÖNSÖZ

Bu doktora tez çalışmasının başlangıcından itibaren teşvik edici fikirleriyle bana rehberlik eden, önerilerini ve katkılarını esirgemeyen tez danışmanım ve çok değerli hocam Doç. Dr. İsmail BABAOĞLU'na en içten teşekkürlerimi sunarım.

Doktora tez izleme komitesinde yer alan ve tavsiyeleriyle çalışmalarına büyük destek veren Prof. Dr. Tayfun ÇAY'a ve Doç. Dr. Mustafa Servet KIRAN'a çok teşekkür ederim.

Güzel bir çalışma ortamı sağlayan Konya Teknik Üniversitesi'ne ve bu süre boyunca fikirlerini paylaşmaktan çekinmeyen Bilgisayar Mühendisliği Bölümündeki öğretim elemanlarına teşekkür ederim.

Bugünlere gelmemde büyük emekleri olan saygıdeğer anneme ve babama en içten teşekkürlerimi sunarım.

Son olarak, tezimi yoğun çalışma sürecimde gösterdikleri büyük sabır ve anlayışla desteklerini esirgemeyen değerli eşim Şeyma KOÇ'a ve sevgili oğlum Ebubekir KOÇ'a ithaf ederim.

İsmail KOÇ
KONYA-2020

İÇİNDEKİLER

ÖZET	iv
ABSTRACT.....	vi
ÖNSÖZ	viii
İÇİNDEKİLER	ix
SİMGELER VE KISALTMALAR	xii
1. GİRİŞ	1
1.1. Tezin Amacı ve Önemi	2
1.2. Literatüre Katkısı	4
1.3. Tezin Organizasyonu	6
2. KAYNAK ARAŞTIRMASI	7
3. MATERYAL VE METOT	14
3.1. Materyal	14
3.1.1. Ön parselasyon problemi için proje sahası	14
3.1.1.1. Ön parselasyon için gerekli kavramlar	15
3.1.2. Dağıtım ve parselasyon problemi için proje sahası	16
3.1.3. Sentetik veri seti üretilmesi.....	18
3.1.3.1. Sentetik veri setinin önemi	18
3.1.3.2. Sentetik veri seti özellikleri	19
3.1.3.3. Sentetik veri setine ait örnek problemler	20
3.2. Metot.....	22
3.2.1. İkili Arama Yöntemi (BS)	22
3.2.3. Yapay arı koloni algoritması (ABC).....	24
3.2.3.1. Başlangıç evresi	25
3.2.3.2. İşçi arı evresi	26
3.2.3.3. Gözcü arı evresi	27
3.2.3.4. Kâşif arı evresi	27
3.2.3. Diferansiyel gelişim algoritması (DE)	27
3.2.3.1. Mutasyon	28
3.2.3.2. Çaprazlama	29
3.2.3.3. Seçim	29
3.2.4. Genetik algoritma (GA)	30
3.2.4.1. Seçim Operatörü	31
3.2.4.2. Çaprazlama Operatörü	32
3.2.4.3. Mutasyon Operatörü	33
3.2.5. Parçacık sürü optimizasyonu algoritması (PSO)	34
3.2.6. Ağaç tohum algoritması (TSA).....	36
4. İMAR UYGULAMALARI İÇİN ÖNERİLEN YÖNTEMLER.....	38

4.1. Ön Parselasyon İşlemi için Önerilen Yöntemler	38
4.1.1. BS ile Alan Hesabı.....	38
4.1.2. Kenar uzunluklarına göre ön parselasyon.....	39
4.1.3. Alana göre ön parselasyon	40
4.1.4. Kenar uzunluğu ve alana göre ön parselasyon.....	41
4.1.4.1. Parsel alanlarını dengeleme	41
4.2. Gerçek Bir Proje Sahası Kullanılarak Dağıtım ve Parselasyon Problemlerinin Çözümüne ait Önerilen Yöntemler	42
4.2.1. Dağıtım ve parselasyon problemlerinin tanımı ve amaç fonksiyonu	42
4.2.2. Başlangıç popülasyonunun oluşturulması	47
4.2.3. Başlangıç popülasyonunun değerlendirilmesi	48
4.2.4. Önerilen çaprazlama operatörü: Harita tabanlı çaprazlama.....	48
4.2.4.1. Klasik parsel tabanlı çaprazlama (CPC)	48
4.2.4.2. Zeki parsel tabanlı çaprazlama (IPC).....	50
4.2.5. Önerilen mutasyon operatörleri	53
4.2.5.1. Aynı ada içindeki parsellerin yerini değiştir	53
4.2.5.2. Farklı adalardaki parsellerin yerini değiştir	54
4.2.5. Uygun olmayan parsellerin birleştirilmesi.....	54
4.2.6. Parselasyon planının güncellenmesi	55
4.2.7. Yeni aday çözümün değerlendirilmesi	55
4.2.8. Algoritmanın sonlandırılması	55
4.2.9. Uzman sistem.....	62
4.3. Sentetik Veri Seti Kullanılarak Dağıtım ve Parselasyon Problemlerinin Çözümüne ait Önerilen Yöntemler	64
4.3.1. Amaç fonksiyonunun belirlenmesi ve örnek bir problem	64
4.3.2. Başlangıç popülasyonunun oluşturulması ve değerlendirilmesi.....	69
4.3.3. Önerilen çaprazlama operatörü	70
4.3.4. Önerilen mutasyon operatörleri	72
4.3.4.1. Parselleri yer değiştir	72
4.3.4.2. Malikleri yer değiştir	74
4.3.4.3. Malik ekle	76
4.3.5. Parselasyon planının güncellenmesi	78
4.3.6. Yeni aday çözümün değerlendirilmesi	78
4.3.7. Algoritmanın sonlandırılması	78
5. DENEYSEL ÇALIŞMALAR.....	80
5.1. Ön Parselasyon İşlemi için Deneysel Çalışmalar	80
5.1.1. Kenar uzunluğuna göre ön parselasyon sonuçları (PwEL).....	81
5.1.2. Alana göre ön parselasyon sonuçları (PwA).....	84
5.1.3. Kenar uzunluğu ve alana göre ön parselasyon sonuçları (PwELA)	85
5.1.3.1. PwELA ve PwA yönteminin birlikte kullanımı ile alan dengeleme.....	86
5.2. Gerçek bir Proje Sahası Kullanılarak Dağıtım ve Parselasyon Problemlerinin Çözümüne ait Deneysel Çalışmalar	88
5.3. Sentetik Veri Seti Kullanılarak Dağıtım ve Parselasyon Problemlerinin Çözümüne ait Deneysel Çalışmalar	98
5.3.1. Performans analizi	98
5.3.2. Zaman analizi.....	106
5.3.3. Genel analiz sonuçları.....	112
6. SONUÇLAR VE ÖNERİLER	120

6.1. Sonular	120
6.2. neriler	122
KAYNAKLAR	123
EKLER	128
ZGEMİŐ	132



SİMGELER VE KISALTMALAR

AAD	: Arsa ve Arazi Düzenlemesi
LR	: Dağıtım ve Parselasyon (Land Redistribution and Readjustment)
GIS	: Coğrafi Bilgi Sistemi (Geographic Information System)
AHP	: Analitik Hiyerarşik Süreç (Analytical Hierarchy Process)
IP	: Tamsayı Programlama (Integer Programming)
LP	: Doğrusal Programlama (Linear Programming)
MIP	: Karma Tamsayılı Programlama (Mixed Integer Programming)
CA	: Hücresel Otomata (Cellular Automata)
CPC	: Klasik Parsel Tabanlı Çaprazlama (Classical Parcel based Crossover)
IPC	: Zeki Parsel Tabanlı Çaprazlama (Intelligent Parcel based Crossover)
BS	: İkili Arama Yöntemi (Binary Search)
ABC	: Yapay Arı Koloni Algoritması (Artificial Bee Colony Algorithm)
GA	: Genetik Algoritmalar (Genetic Algorithm)
DE	: Diferansiyel Evrim Algoritması (Differential Evolution Algorithm)
PSO	: Parçacık Sürü Optimizasyonu (Particle Swarm Optimization)
TSA	: Ağaç Tohum Algoritması (Tree Seed Algorithm)
ST	: Arama Eğilimi (Search Tendency)
pbest	: Yerel En İyi (personal Best)
Gbest	: Global En İyi (Global Best)
FAR	: Toplam Alanın Binanın Zemin Alanına Oranı (Floor Area Ratio)
PwEL	: Kenar Uzunluğuna göre Ön Parselasyon (Parceling with Edge Length)
PwA	: Alana göre Ön Parselasyon (Parceling with Area)
PwELA	: Kenar Uzunluğu ve Alana göre Ön Parselasyon (Parceling with Edge Length and Area)
MaxFEs	: Maksimum Fonksiyon Değerlendirme Sayısı (Maximum Number of Function Evaluation)
FEs	: Fonksiyon Değerlendirme Sayısı (Number of Function Evaluation)
NLP	: Parsel Başına Düşen Malik Sayısı (Number of Landowner per Parcel)
TSP	: Gezgin Satıcı Problemi (Traveling Salesman Problem)

1. GİRİŞ

Kentsel arazi kullanımı, özellikle gelişmekte olan veya gelişmemiş ülkelerde artan nüfusa bağlı olarak sosyal, kültürel ve barınma gibi temel ihtiyaçlar için giderek önemi artan bir değer haline geldiğinden etkili yönetim sistemleri gerektirmektedir (Yomralioglu ve ark., 2007). Hızlı ve etkili bir kentleşme, büyük ölçüde kamu ve gerçek kişilerin talep ettiği imarlı arsalarla bağlıdır. Arsa ve arazi düzenlemesinin (AAD) en önemli faaliyetlerinden olan dağıtım ve parselasyon işlemleri (LR – *Land redistribution and readjustment*), kırsal alanları kentsel araziye dönüştürerek ve çeşitli ülkelerde kentsel altyapı sağlayarak insanların konut ve diğer sosyal ihtiyaçlarını karşılayan önemli bir araçtır (Yılmaz ve ark., 2015). LR, aynı zamanda düzgün bir şekilde sahip olmayan kadastro parsellerini daha etkin ve kolay kullanılabilir olacak uygun parsellere dönüştürmek için kullanılan etkili bir araçtır. Bu tanıma ek olarak, bu yöntem o bölgede yaşayan kişilerin yaşam kalitesini ve çevreyi iyileştiren temel kamu arsalarının oluşmasını sağlayan etkili bir tekniktir. Yöntemin temel avantajları ise, proje süresini en aza indirmesi ve geniş proje alanlarına uygulanabilmesidir (Doebele ve ark., 1986).

Kentsel arazi gelişimindeki temel sorunlardan biri eski alanların yenilenmesidir. Buradaki temel sorun, kentsel yenilenmenin eski binaları yenileriyle değiştirmekten daha fazlası olduğu gerçeğinde yatmaktadır (Li ve Li, 2007). Normalde kentsel yenileme ihtiyacının neden olduğu zorunlu satın alma veya özel mülkiyet, her zaman teknik avantajları teşvik eder. Çünkü bu sorunlar mülkiyet haklarının devri ve mevcut mülk sahiplerinin veya bir hisse talep edebilenlerin yeniden yerleşimi için tazminat içermektedir (Adams ve ark., 2001; Ng, 2002; Cypher ve Forgey, 2003).

Pek çok ülkede kentsel gelişim için zorunlu satın alma veya kamulaştırma gibi klasik araçlar kullanılmaktadır (De Wolff, 2002). Ancak bu araçlar, genellikle arazilerini vermek istemeyen arsa sahiplerinin itirazlarına neden olabilmektedir. Bu işlemler, tartışmalı müzakerelere bağlı olarak uzun vadeli mahkeme işlemlerinden dolayı sürdürülebilir kentsel gelişimin gerçekleştirilmesinde uzun süreli bir gecikmeye neden olabilmektir (Müller-Jökel, 2004). Kamu kuruluşlarına ait LR projeleri söz konusu olduğunda, bu durum genellikle yürürlükteki nazım ve uygulama planları ile sağlanmaktadır. Bu tür LR uygulamaları resen yapılmakta olup arazi sahiplerinin kabul ve onayına gerek duyulmamaktadır.

Klasik bir LR süreci, belediye tarafından yeniden düzenlenmesi planlanan herhangi bir bölgenin imar planlarının hazırlanmasıyla başlanılmaktadır. İmar planında sokakların oluşturduğu adalar (bloklar) içerisinde çok sayıda kamu veya özel mülkiyet alanları tahsis edilmektedir. Daha sonra sokak, park gibi kamusal alanlar hesaplanarak toplam proje alanı ile karşılaştırıldıktan sonra Düzenleme Ortaklık Payı (DOP) belirlenmektedir (Doebele, 1982). LR, Almanya, Hindistan, Malezya, Endonezya, Fransa, Güney Kore, Japonya, İtalya, İsveç, Nepal, Tayland, Batı Avustralya ve Kanada gibi dünyanın çeşitli bölgelerinde yaygın olarak kullanılan bir şehir planlama uygulama aracıdır (Larsson, 1997; Sorensen, 2000b; Sorensen, 2000a). Fakat LR kullanımı ülkeden ülkeye değişiklik gösterebilmektedir. Bir ülkede arazi sahiplerinin etkinliği daha fazla olabilirken, başka bir ülkede devlet ve kamu kurumlarının etkisi daha fazla olabilmektedir (Türk, 2009). Bununla birlikte, tüm ülkelerde DOP oranı, toplam kadastro parselleri alanından imar adalarının alanının çıkartılması ile bulunmaktadır. DOP kesintisinden sonra her arsa sahibi için kalan arazi, uygulama sahasındaki imar adalarına müstakil/hisseli olarak tahsis edilmektedir.

1.1. Tezin Amacı ve Önemi

Nüfus artışına bağlı olarak zamanla artan barınma, sosyo-ekonomik ve kültürel ihtiyaçların karşılanabilmesi için kentsel arazinin planlı ve en uygun şekilde kullanılması gerekmektedir. Yaşanılabilir ve kullanılabilir arazinin kısıtlı olması da kentsel arazi kullanımını daha da önemli hale getirmektedir. Sağlıklı ve yaşanılabilir bir kentleşme, bireylerin temel ihtiyaçlarını sağlayacak şekilde gerçekleştirilmiş imar uygulamalarıyla doğrudan ilişkilidir. Ülkemizde, imar uygulamaları farklı yöntemlerle yapılmaktadır. Fakat şehrin gelecekteki görünümünü oluşturan parselasyon işlemleri aynıdır. İmar planları içerisinde farklı büyüklükte ve genellikle düzgün şekle sahip olmayan kadastro parselleri bulunduğundan daha düzenli bir kentsel görünüm elde etmek için bunların imar planına göre yeniden düzenlenmesi gerekmektedir.

Türkiye’de dağıtım ve parselasyon işlemleri 3194 sayılı İmar Kanunun 18. Maddesine göre imara tahsis şeklinde yapılmaktadır. Bu işlemlerde kadastro parsellerindeki arsa sahiplerinin tahsisi, kadastro parsellerinin isabet ettiği imar planındaki imar parsellerinden yapılarak gerçekleştirilmektedir (Anonim, 2020).

Kadastro parsellerinin imar parselleriyle akıřtıęı durumlarda daęıtım iřlemi kolayca gerekleřtirilebilmektedir. Fakat bazı durumlarda kadaastro parselleri imar planlarında park, yeřil alan, otopark, saęlık ocaęı, eęitim kurumu, belediye hizmet alanı ve yol gibi kamusal alanlara isabet edebilmektedir. Byle bir durumda 3194 sayılı imar kanununa gre, kamusal alanlara isabet eden kadaastro parsellerindeki arsa sahipleri iin eski yerinden tahsisi teknik olarak mmkn olmadıęından, bulundukları eski kadaastro parsellerine en yakın imar parsellerinden tahsis iřlemleri yapılmaktadır. Bununla birlikte kadaastro ada ve parsellerin geometrik olarak dzgn řekle sahip olmaması, arsa sahiplerinin ok farklı hisse alanlarına sahip olması ve kadaastro ve imar planlarının akıřmamasından kaynaklı durumlar da daęıtım ve parselasyon iřlemlerini gerekleřtirmeyi zorlařtırmaktadır. Bu iřlemler hali hazırda manuel olarak yapıldıęından daęıtım ve parselasyon iřlemini gerekleřtiren mhendis veya teknisyenlere teknik aıdan ciddi zorluk oluřtırmaktadır. Aynı zamanda, manuel daęıtım ile teknisyen veya mhendis, hissi davranarak bazı arsa sahiplerine eski kadaastro parseline yakın bir imar parseline tahsis yaparken, bazılarına ise kadaastro parseline daha uzak bir imar parseline tahsis yaparak objektif olmayan bir daęıtım iřlemi gerekleřtirebilmektedir. Bu da uygulama sahasındaki sosyal huzurun bozulmasına neden olabilmektedir. Dięer bir problem ise, sz konusu kadaastro parselindeki maliklerin hisselerinin imar planına ne řekilde daęıtımının yapılacaęındaki zorluktur. Mlkiyet kullanımında olduęu gibi imar adalarının oluřmasında bireysel ve paylı mlkiyetli parseller bulunabilmektedir. Hisseli parsellerdeki arsa sahipleri (malikler), sz konusu parsellerin tasarrufu hakkında anlařamadıklarından, daęıtım sonucunda retilen hisseli parseller maliklerin istemedięi bir durumdur. rnek vermek gerekirse; kamusal alana isabet eden veya kadaastro ile imar planları akıřmayan bir proje sahasındaki herhangi bir parselde ait maliklerin daęıtımı iin, ilgili kadaastro parselinin yakınlarında DOP'tan bořalan herhangi bir imar parseli bazı durumlarda bulunmayabilir. Byle durumlarda daęıtımı yapılacak parseldeki maliklerin hisse alanlarının deęerleri dikkate alınarak hisseli parsel sayısının en az olacak řekilde daęıtım iřleminin gerekleřtirilmesi gerekmektedir. Hisselerin fazla olduęu durumlarda manuel olarak gerekleřtirilen daęıtım iřlemi teknik aıdan olduka zordur. Kadaastro parselinden uzaklařtıķa da arsa sahipleri bu durumu kabul etmemektedir. Manuel olarak gerekleřtirilen daęıtım ve parselasyon iřlemlerinde yařanan bu tr zorluklardan dolayı arsa sahipleri hali hazırdaki imar uygulamasına itiraz ederek/dava ederek iřlemlerin 3 ila

12 ay uzamasına neden olabilmektedir. Bu durum uygulama sahasında inşaat yapacak olan malik veya müteahhitlerin projelerine engel teşkil edebilmektedir. Sonuç olarak imar uygulamalarındaki bu tür problemlerden dolayı konut, sanayi, turizm, kamusal veya ticaret alanlarının inşaatı gibi yatırımlar gecikebilmektedir.

Bu tez çalışmasında imar uygulamalarındaki dağıtım ve parselasyon problemlerinin yapay zeka optimizasyon algoritmalarıyla çözümü hedeflenmektedir. Bu problemler için iki temel hedef gözetilmektedir. Bunlardan ilki, kadastro parselindeki maliklerin tahsisinin kadastro parseline en yakın imar parselinden yapılmasıdır. İkincisi ise, kadastro parseli içerisindeki maliklere müstakil imar parseli üretilmesi teknik açıdan mümkün ise bu işlemin gerçekleştirilmesidir. Bu işlemler sayesinde proje sahasında hisseli parsel sayısının en aza indirgenmesi hedeflenmektedir. Söz konusu bu iki hedefi en üst düzeyde karşılayacak olan çözüm, en uygun çözüm olarak kabul edilmektedir. Yapay zeka optimizasyon algoritmaları ile elde edilen en uygun çözüm sayesinde maliklerin uygulayıcılara ve projeye olan güvenlerinin artması hedeflenmektedir. Sonuç olarak, imar uygulamalarındaki dağıtım ve parselasyon işlemleri için önerilen optimizasyon algoritmaları yardımıyla manuel olarak gerçekleştirilen işlemler sonucunda oluşması muhtemel olumsuzlukların giderilmesi ve buna bağlı olarak projeden beklenen sosyal huzurun sağlanması hedeflenmektedir.

1.2. Literatüre Katkısı

İmar uygulamalarında dağıtım ve parselasyon problemleri, farklı kısıtları içerisinde barındıran gerçek dünya problemleridir. Bununla birlikte bu problemler kombinatoryal bir optimizasyon problemi olduğu için klasik yöntemlerle çözülmesi zor problemlerdir. Bundan dolayı bu problemlerin çözümü klasik yöntemler yerine yapay zeka optimizasyon algoritmaları gerektirmektedir. Bundan dolayı öncelikle algoritmanın doğru karar vermesini sağlayacak bir uygunluk fonksiyonuna ihtiyaç duyulmaktadır. LR problemlerini çözmeyi sağlayacak olan bu amaç fonksiyonu sayesinde dağıtım ve parselasyon planlarının uygunluğu bir uzmana ihtiyaç duyulmaksızın otomatik olarak değerlendirebilmekte ve mevcut çözümler birbirleriyle karşılaştırılabilmektedir. Önerilen amaç fonksiyonu ile literatürde en çok tercih edilen ve güçlü özellikleri olan optimizasyon algoritmaları kullanılarak bir sistem önerilmektedir. Bu işlemlerin hali hazırda manuel

olarak yapıldığı göz önüne alındığında önerilen yapay zeka tabanlı optimizasyon algoritmaları sayesinde tezin önemi net bir şekilde ortaya konulmaktadır. Geliştirilecek olan sürü zekası ve evrimsel tabanlı bu yöntemler sayesinde imar uygulamalarının daha hızlı sonuçlanmasını sağlamaktadır. Ayrıca daha adil bir dağıtım yapılarak sosyal huzurun oluşmasına katkı sağlaması hedeflenmektedir. Bununla birlikte LR ile ilgili literatür taraması incelendiğinde teknik açıdan bu konu ile ilgili çok az çalışma olduğu görülmektedir. Bu yüzden, bu tez çalışmasının literatürdeki yapay zeka temelli LR problemlerinin çözümüne ait boşluğu doldurması da hedeflenmektedir.

Bu tez çalışması üç temel aşamadan oluşmaktadır:

Birinci aşamada, ikili arama (BS – *binary search*) yöntemi ile Gauss yöntemi birlikte kullanılarak ön parselasyon işlemi gerçekleştirilmiştir. Önerilen ön parselasyon sayesinde LR problemlerinin çözümü için etkili bir alt yapı oluşturulmuştur.

İkinci aşamada ise, gerçek bir proje sahası üzerinde yapay arı koloni (ABC – *artificial bee colony*), diferansiyel gelişim (DE – *differential evolution*), genetik (GA – *genetic algorithm*), parçacık sürü optimizasyonu (PSO – *particle swarm optimization*) ve ağaç tohum (TSA – *tree seed algorithm*) algoritmaları kullanılarak sürü zekası ve evrimsel tabanlı yaklaşımlar sunulmuştur. Ayrıca dağıtım ve parselasyon problemlerinin çözümünde kullanılmak üzere iki farklı çaprazlama operatörü önerilmiştir: Klasik parsel tabanlı çaprazlama (CPC- *Classical parcel based crossover*), Zeki parsel tabanlı çaprazlama (IPC - *Intelligent parcel based crossover*). Ayrıca mutasyon işlemleri de imar uygulamalarına özgü tasarlanmıştır. Önerilen çaprazlama ve mutasyon işlemleri birlikte kullanılarak hibrit bir yaklaşım önerilmiştir. Uygulanan bu yaklaşım ile imar uygulamaları için alternatif ve etkili bir yöntem literatüre kazandırılmıştır. Ayrıca LR için geliştirilen uzman bir sistem yardımıyla elde edilen çözümler gerçek dünyada doğrudan kullanılabilir hale getirilmiştir.

Üçüncü aşamada ise LR için sentetik bir veri seti önerilmiştir. Gerçek dünyadaki bir problem, şekil ve boyut açısından çok karmaşık olabileceğinden, en iyi çözümün ne olduğunu bilmek genellikle çok zordur. Örneğin, bir proje sahasındaki kadastro ve imar adalarının çakışmaması olağan bir durumdur. Bu nedenle, amaç fonksiyonundaki mesafe ceza puanının en iyi değerinin ne olduğunu bilmek oldukça zordur. Diğer bir durum ise, maliklerin hisselerinin alan olarak çok farklı değerlere sahip olmasından dolayı maliklerin tahsisinin nasıl ve hangi parsel yapılabileceğini belirlemek oldukça zordur. Bu durum da

hisse ceza puanının en iyi değeri ne olduğunu bilmeyi güçleştirmektedir. Söz konusu bu durumlar, gerçek bir proje sahası üzerinde önerilen algoritmanın başarısını ifade etmesi zordur. Tüm bu dezavantajlardan dolayı, bu tez çalışmasında sentetik bir veri seti önerilmiştir. Çünkü sentetik bir veri seti kullanımı, yöntemin başarısının kabul edilebilir veya uygulanabilir olup olmadığına karar vermeyi ve mevcut çözümler üzerinde objektif değerlendirmeyi oldukça kolaylaştırmaktadır. Ayrıca literatür incelendiğinde LR çalışmaları için sentetik bir veri setinin olmadığı gözlemlenmiştir. Bu da tezin literatüre önemli bir katkısı olduğunu göstermektedir.

1.3. Tezin Organizasyonu

Bu tez çalışması altı bölümden oluşmaktadır.

Birinci bölümde, LR ile ilgili genel bilgiler verilmektedir. Ayrıca tezin amaç ve önemi kısaca anlatılmaktadır. Tezin literatüre katkısı ise üç aşamada anlatılmaktadır.

İkinci bölümde ise dağıtım ve parselasyon problemleriyle ilgili kaynak araştırması yapılarak çalışmalara ait kısa bilgiler sunulmaktadır.

Materyal ve metotları içeren üçüncü bölüm, tez çalışmasında kullanılacak olan verilere ait bilgileri ve kullanılan yöntemleri sunmaktadır. Ayrıca önerilen sentetik veri seti ile ilgili bilgiler de bu bölümde yer almaktadır.

Dördüncü bölümde imar uygulamalarında kullanılmak üzere önerilen yöntemler detaylıca açıklanmaktadır. İlk olarak ön parselasyon işlemi için önerilen yöntem sunulmaktadır. İkinci olarak LR problemleri için önerilen sürü zekâsı ve evrimsel algoritma temelli yaklaşımlar sunulmaktadır. Ayrıca imar problemleri için önerilen çaprazlama ve mutasyon operatörleri de bu bölümde anlatılmaktadır. Son olarak ise sentetik veri setindeki problemler için önerilen yaklaşımlar detaylı olarak verilmektedir.

Beşinci bölümde, gerçekleştirilen deneysel çalışmalar verilmektedir. İlk olarak, ön parselasyon aşamasında elde edilen sonuçlar harita üzerinde uzman kişinin sonucuyla karşılaştırmalı olarak sunulmaktadır. İkinci aşamada, gerçek bir proje sahası kullanılarak beş farklı algorithmadan edilen sonuçlar karşılaştırmalı olarak verilmektedir. Son olarak ise sentetik veri seti yardımıyla elde edilen sonuçlar sunulmaktadır.

Altıncı bölümde ise elde edilen sonuçlar genel anlamda değerlendirilerek gelecek çalışmalara yön vermesi açısından öneriler verilmektedir.

2. KAYNAK ARAŞTIRMASI

Türkiye ve uluslararası alandaki arsa ve arazi düzenlemesinin (AAD) en önemli faaliyetlerinden olan dağıtım ve parselasyon işlemleri (LR) ile ilgili literatür incelendiğinde, ağırlıklı olarak finans, yönetim ve çok taraflı katılım açısından çalışmalar bulunmaktadır. Bununla birlikte, arazi tahsis ve yeniden düzenlenmesinin teknik ayrıntıları ve bunların matematiksel modellemesi ve LR sürecinin çözümüyle ilgili sınırlı sayıda çalışma bulunmaktadır. Kaynak araştırma kısmında ilk olarak tez çalışmasıyla büyük ölçüde benzerlikler gösteren imar uygulamalarıyla ilgili teknik açıdan ele alınmış literatürdeki çalışmalar aşağıda verilmektedir. Bu çalışmalara ek olarak AAD ile ilgili genel bilgiler içeren önemli bazı çalışmalara ait özet bilgiler de bu bölümde sunulmaktadır.

Yomraliöglu (1992), yaptığı çalışmada AAD’de parsellerin yeniden dağıtım aşamasında değerlendirme konusunu ele almıştır. Arazi düzenleme çalışmalarında bir parselin değerini etkileyebilecek faktörlerin neler olduğunu ve bu faktörlerin nasıl işleme alınması gerektiğini incelemiştir. Ayrıca, AAD çalışmalarında parsellerin dağıtım için yeni bir çözüm tekniği önermiştir.

Uzun (1992), yaptığı tez çalışmasında kentsel alan düzenlemelerinde standart boyutlarda parsel üretme, tahsis alanlarına göre parsel üretimi, hissedar sayısına göre parsel üretimi gibi farklı imar parseli üretme yöntemleriyle ilgili çalışmalar yapmıştır.

Yomraliöglu ve Parker (1993) çalışmalarında, imar uygulamalarında LR sürecini ve bu sürecin kentsel gelişimdeki potansiyel kullanımından kısaca bahsetmişlerdir. Daha sonra, kentsel alanlarda daha iyi bir arazi yönetimi için Coğrafi Bilgi Sistemi (GIS - *Geographic Information System*) tabanlı bir arazi düzenlemesi modelini nasıl geliştirdiklerini anlatmışlardır.

Larsson (1997), çalışmasında AAD ile ilgili Almanya, Fransa, Japonya ve Avusturalya’da kullanılan yöntemlerin avantajlarının incelemiş ve bu yöntemlerin mevcut sorunlarını ve faydalı yönlerini araştırmıştır.

Aksay (1999), AAD hakkında genel bilgiler vermiştir. Çalışmada, 3194 sayılı imar Kanunu’nun 18. maddesi uygulaması, hukuki açıdan AAD’nin anlamı ve tanımı, uygulama aşamaları ve sonuçları ve yargı kararları ile ilgili mevzuatlar irdelenmiştir.

Sorensen (2000a), yaptığı çalışmada Güney Doğu Asya'da birkaç gelişmiş ülkenin AAD süreci ile ilgilenmiştir. Çalışmada Japon AAD uygulamasının nasıl yapıldığı incelenmiştir. Ayrıca, çalışmada AAD yöntemi, AAD ile ilgili mevcut literatür ve Tokyo bölgesinde yapılan bir proje incelenmiştir.

Aerts ve ark. (2003) çok sahalı arazi kullanım tahsis problemini, üçü doğrusal tam sayı programları olmak üzere dört farklı Tamsayı Programlama (IP – *Integer Programming*) kullanarak çözmüşlerdir. IP'ler, tarama tabanlı bir GIS ortamı için formüle edilmiştir ve geliştirme maliyetlerini en aza indirmek ve tahsis edilen arazi kullanımının yoğunluğunu en üst düzeye çıkarmak için tasarlanmıştır.

Turk (2005), yaptığı uygulama ile yeni arazilerin geliştirilmesini ve kentsel alanlarda inşa edilen yerlerin yeniden organize edilmesini, ayrıca hizmet verilen kentsel alanların ve kamu hizmetleri için gerekli alanların üretimini sağlamayı hedeflemiştir. Bu çalışmada, yöntemi etkileyen ana kısıtlar incelenmiş ayrıca yöntemin etkili bir şekilde çalışması için gerekli düzenlemeler tartışılmıştır.

Nişancı (2005) GIS sayesinde, nominal değerlendirme yöntemine dayalı piksel (raster) tabanlı arazi değerlendirme ve yeniden tahsis sistemi geliştirmiştir.

Turk (2007), çalışmasında başarılı ve verimli uygulama için gereken temel şartları uluslararası literatür çerçevesinde tanımlamıştır. Türkiye'deki LR yöntemi bu temel koşullar çerçevesinde analiz edilmiştir.

Yomraliöglü ve ark. (2007), çalışmalarında raster tabanlı arazi değerlendirme haritası yardımıyla nominal varlık tabanlı LR'ye ilişkin Trabzon ili için bir uygulama gerçekleştirmişlerdir. Bu çalışma için örnek olarak Karadeniz kıyısında yer alan Trabzon ili çalışma alanı olarak seçilmiştir.

Köktürk ve Köktürk (2009), yaptıkları çalışmada 3194 sayılı İmar Kanununun 18. maddesinde tanımlanan ve “eşit orantılı toprak kesintisi” ilkesine dayanan arsa düzenlemesinin, “eşdeğerlik” ilkesine dayalı arsa düzenlemesi ile değiştirilmesini önermişlerdir. Bu yeni ilkenin uygulanabilmesi için gerekli olan taşınmaz değerlendirme esasları incelemişler ve Türkiye için önerilerde bulunmuşlardır.

Uzun (2009) çalışmasında, öncelikle Türkiye'de kentsel arazi geliştirme planlarının uygulama yöntemleri hakkında bilgi vermiştir. Ayrıca, çalışmasında Türkiye'deki AAD yöntemini ayrıntılı olarak açıklamıştır.

Çete (2010) çalışmasında, Türkiye'de kalkınma planlarının uygulanmasında AAD'nin önemini tartışmıştır. Ayrıca, birincil ilkeleri ve uygulama aşamaları dâhil olmak üzere Türk AAD yöntemini tanıtmıştır. Ayrıca, yazar yöntemin güçlü ve zayıf yönleri ile gönüllü AAD uygulamalarını teşvik etmek, arazi sahiplerinin projelere katılımını sağlamak, projeyi alan yerine değer bazında yürütmek, arazi düzenleme uygulamalarından sonra gelişmeleri sağlamak için yasal yaptırımlar uygulamak gibi öneriler getirmiştir. Farklı projelerde kazanılan deneyimleri paylaşmak ve AAD projelerinin genel uygulamalarını iyileştirmek için GIS teknolojilerini kullanarak web tabanlı bir platform oluşturmayı önermiştir.

Turk ve Altes (2010), çalışmalarında kentsel düzenlemede belirli işlevlerin yerine getirilmesinde mevcut yaklaşımların tartışılmasına ve karşılaştırılmasına zemin hazırlamışlardır. Sonuçlar, toprak sahibi dernekler veya kooperatifler gibi özel bir önceliğin AAD modelinin Türkiye'de kentsel dönüşüm için uygun bir araç olabileceğini ortaya koyduğu göstermiştir.

Wickramasuriya ve ark. (2011), çalışmalarında vektör verilerini kullanarak tam otomatik bir arazi bölümlleme aracı sunmuşlar ve arazi bölümlleme için gerçekçi bir otomatik yöntem geliştirmişlerdir. Çalışmalar Güney Avustralya'da gerçekleştirilmiş olup vektör verilerini kullanan ve herhangi bir şekilde arazi parselleri için hem hisse hem de sokak düzenlemeleri içeren yerleşimler üretebilen tam otomatik bir arazi parselasyonunu gerçekleştirebilen bir yaklaşım sunmuşlardır.

Uygun (2011), yüksek lisans tezinde imar uygulamalarında değer esaslı kriterlerine göre dağıtım probleminin çözümünde bulanık mantık yöntemini kullanmıştır. Söz konusu modelin geçerliliğini doğrulamak için alan esasına göre yapılacak dağıtımla karşılaştırması yapılmış ve sonuçları analiz edilmiştir. Arsa düzenlemesi yapılmamış bir bölge üzerinde, ilk olarak mevcut kanuna uygun olarak "Alan Esasına" göre dağıtımının yapılması sağlanmıştır. Bunun ardından yapay zeka yöntemi kullanılarak "Değer Esasına" göre de kriterlerin belirlenerek dağıtım probleminin çözümü gerçekleştirilmiş ve iki farklı kritere göre elde edilen sonuçlar kıyaslanmıştır.

Feng ve ark. (2011), hücresel otomata (CA – *cellular automata*) ve PSO yöntemlerini kullanarak dinamik kentsel büyümenin modellenmesini gerçekleştirmişlerdir. PSO tabanlı CA modeli, Doğu Çin'deki Şanghay Belediyesi Fengxian bölgesinde, 1992'den 2008'e kadar 30 metrelik mekânsal çözünürlükte kentsel

büyümenin mekânsal-zamansal sürecini simüle etmek için uygulanmıştır. Hata matrisi ve simülasyon doğrulukları ile değerlendirilen simülasyon sonuçları, PSO temelli CA modelinin diğer mekânsal istatistik tabanlı CA modellerini geride bıraktığını göstermiştir.

Rabbani ve ark. (2012), hücresel yapı ve vektör kavramının bir kombinasyonu olarak hibrit CA modelini önermişlerdir. PSO ise hücrelerin gelişim parametrelerine olan uzaklıklarına göre şehirleşme olasılığını hesaplamak için kullanılmıştır. Önerilen model, 1988'den 2010'a kadar kentsel büyümeyi simüle etmek için 28.5 metrelik mekânsal çözünürlüğe sahip olan Tahran kentinin Landsat uydu görüntülerine uygulanmıştır.

Dahal ve Chow (2014) kentsel arazilerin otomatik olarak bölünmesi için *ArcGIS* isimli bir araç setini geliştirmişlerdir. Modellenmiş kentsel yerleşim planlarının gerçek dünyadaki şehirlerde gözlemlenen alt bölüm yapılandırmalarıyla uyumlu olduğu sonuçlardan gözlemlenmiştir. Ayrıca, geliştirilen yazılımın hem dinamik kentsel büyüme modellerine entegre edilebilir hem de bağımsız uygulamalar olarak kullanılabildiği görülmüştür.

Yılmaz ve ark. (2015), yaptıkları çalışmada ulusal LR süreçlerinin değerlendirilmesine ve karşılaştırılmasına yardımcı olacak bir çerçeve ve yöntem bilimi oluşturmak için LR hakkında çok çeşitli makaleleri incelemişlerdir. Çalışmanın temel katkısı, bir ülkenin LR performansını, literatürde mevcut olmayan, sistematik bir şekilde değerlendirmek için uluslararası olarak kabul görmüş bir metodolojinin oluşturulması için bir farkındalık ortaya koymak olmuştur.

Terzioğlu (2015), yaptığı çalışmada ülkemizde gerçekleştirilen imar uygulama yöntemlerini irdelleyerek Sultanbeyli ilçesinin tarihsel gelişimi ve mülkiyet yapısı hakkında bilgi vermiştir. Çalışma sonucunda sorunlar ve mevzuatta yapılabilecek muhtemel değişiklikleri içeren alternatif bir çözüm önerisi sunulmuştur. Çözüm önerilerine göre ilgili parseller için ayrı ayrı temsili imar uygulamaları yapılarak elde edilecek kazanımlar detaylı şekilde tartışılmıştır.

Kucukmehmetoglu ve Geymen (2016) LR için Karma Tamsayılı Programlama (MIP - *Mixed Integer Programming*) ve Doğrusal Programlama (LP - *Linear Programming*) tekniklerini kullanarak parselasyon ve dağıtım işlemlerini gerçekleştirmişlerdir. Çalışmalarında geliştirdikleri yöntem üç farklı özellikten oluşmaktadır. İlk özellik olarak operasyon araştırma tekniklerine dayanan orijinal matematiksel modeller sunulmuştur. Bu özellik çerçevesinde, tüm arsa sahiplerinin

kamusal alan kullanımından orantılı olarak düşülmesinden sonra kadastro parsellerini yüksek kaliteli imar parsellere ayıran iki adet LP ve bir MIP modeli kullanılmıştır. İkinci özellik, 18. Madde Yönetmeliği'ne bağlı olarak optimizasyon modellerinin arazinin yeniden tahsis edilmesinde mevcut uzman bazlı öznel uygulamaların üstünlüğünü ortaya koymak için geliştirilen değerlendirme kriterleri olmuştur. Bu kriterler, tahsis sonuçlarından yararlanan denklemler biçiminde tanımlanmıştır. Üçüncü özellik, kadastro parsel merkezlerinden imar parsellerindeki hisse merkezlerine kadar olan ışın sistemleri üzerindeki dağıtım sonuçlarının grafiksel sunumu olmuştur.

Alaybeyoğlu ve ark. (2016), kentsel dönüşümde mahalleler arası öncelik sıralamasını gerçekleştirmek için bulanık mantık yöntemi ile örnek bir karar destek sistemi geliştirmişlerdir. Karar destek sistemi kurulurken, öncelikle sistemin kamu kurumları tarafından kullanılabilmesi için olabildiğine basit mantıklı ve pratik olmasına dikkat etmişlerdir. Mülakatlar ışığında, en çok önem arz eden şu dört parametre üzerinde fikir birliği sağlamışlardır: Yapı Kalitesi, Altyapı, Sosyal Sorunlar ve Projenin Finanse Edilebilirliği. Tasarladıkları sistemde kuralların oluşturulması için, parametrelerin önem sırasını belirledikten sonra geliştirilen sistemde üç farklı test mahallesi için analizlerini yapmışlardır. Sonuç olarak böyle bir sistemin, belediye meclislerinde veya ilgili diğer kurumlarda, kentsel dönüşümde hangi bölgeden başlanması gerektiği konusunda verilecek kararlara katkı sağladıklarını belirtmişlerdir.

Ekiz (2017), çalışmasında ülkemizde AAD'de uygulanan alan bazlı dağıtımdan kaynaklanan sorunlara çözüm bulmaya çalışmıştır. Bunun için Analitik Hiyerarşik Süreç (AHP - *Analytical Hierarchy Process*) kullanarak değer esaslı bir yöntem önermiştir.

Ghavami ve ark. (2017) akıllı mekânsal arazi kullanım planlama destek sistemini önermişlerdir. Bu çalışmalarında, gerçek aktörler adına çalışan sosyal olarak rasyonel akıllı ajanlar tasarlamak için sosyal değer yönelim teorisinin yanı sıra hesaplama açısından etkili bir Bayes öğrenme tekniğini kullanmışlardır. Önerilen sistem, gerçek dünya kentsel arazi kullanım planlaması çalışmasına uygulanmıştır.

Maleki ve ark. (2017), İran'ın Tahran ilinde yaptıkları çalışmada kentsel arazi kullanımlarının sıralanması ve tahsisi için bir parsel seviye modeli sunmuşlardır. Önerilen model, GIS, bulanık hesaplamaların ve çok kriterli karar verme yöntemlerinin (*bulanık TOPSIS*) yeteneklerinden yararlanarak, mevcut kentsel arazi kullanımına ait planlama destek sistemlerinin yeteneklerini geliştirmeyi amaçlamıştır. Bu modelde ilk adım olarak

GIS'in bulanık hesaplamaları ve mekânsal analiz yetenekleri kullanılarak parsellerin ve mahallelerinin fiziksel özelliklerine göre nicel ve nitel değerlendirme kriterleri tahmin edilmiştir. İkinci adımda *bulanık TOPSIS* yöntemiyle kentsel arazi kullanımları kentsel arazi birimlerinin her biri için sıralanmıştır.

Demetriou (2017), kentsel düzenleme projelerinde parsellerin bölünmesinde, Kıbrıs'ta büyük bir çalışma alanından seçilen iki adada otomatik bölümlleme işlemi yapmıştır. GIS ve GA'nın bir araya getirildiği *LandParselS* isimli modül ile parsel bölümlmesi otomatik olarak yapılmıştır. Bu çalışmada parseller şekilsel, boyutsal ve değer açısından optimizasyon işlemine tabi tutulmuştur.

Güngör ve İnam (2019) yaptıkları çalışmada, LR kapsamında farklı dağıtım yöntemleri kullanılarak mevcut kadastro parsellerine tahsis edilen yeni imar parsellerinin alan-değer ilişkisi açısından incelenmesini hedeflemişlerdir.

Bozdağ ve Ertunç (2020) Niğde'nin belli bir bölgesinde seçmiş oldukları 30 dairenin taşınmaz değerini belirlemiş oldukları yapı ve yapının mekânsal özellikleri gibi kıstaslar ışığında açısından değerlendirmişlerdir. Çalışmalarında, AHP ve GIS yöntemi ile yapılan puanlama yardımıyla belirlenen 30 dairenin taşınmaz değerini belirlemişler ve bulunan bu değerleri taşınmazların piyasa değerleri ile karşılaştırmışlardır.

Koc ve ark. (2020) çalışmalarında Konya'nın bir bölgesi üzerinde üç farklı şekilde ön parselasyon işlemini gerçekleştirmişlerdir. Yaptıkları çalışmada parsel kenar uzunluğuna göre ön parselasyon, parsel alanına göre ve hem alan hem de kenar uzunluğu kriterlerine göre ön parselasyon işlemini gerçekleştirmişlerdir. Ön parselasyon işlemlerinde İkili Arama yöntemini ve Gauss yöntemini kullanmışlar ve sonuçları uzmanın elde ettiği sonuçlarla kıyaslamışlardır. Önerilen yöntemler ile ön parselasyon sonuçlarını oldukça kısa sürede elde etmişlerdir. Bu yüzden yaptıkları çalışma ile kesin parselasyon işlemleri için kullanılabilir etkili bir alt yapı sunmuşlardır.

Masoumi ve ark. (2020) çalışmalarında çok amaçlı evrimsel algoritmalar kullanarak dinamik kentsel arazi kullanımı değişim yönetimini önermişlerdir. Çalışma bölgesi olarak İran'ın Tahran şehrini belirlemişler ve problemi çözmek için iki adım önermişlerdir. Bunlardan ilki, bir veya daha fazla kentsel arazi kullanımında değişiklik olması durumunda çevredeki arazi kullanımlarının en uygun düzenlemesini elde etmek için çok amaçlı bir optimizasyon tekniğini önermişlerdir. İkinci adımda ise birinci adımın

çıktılarından karar vericiler için uygun çözümler üretmesi için kümeleme analizini kullanmışlardır.

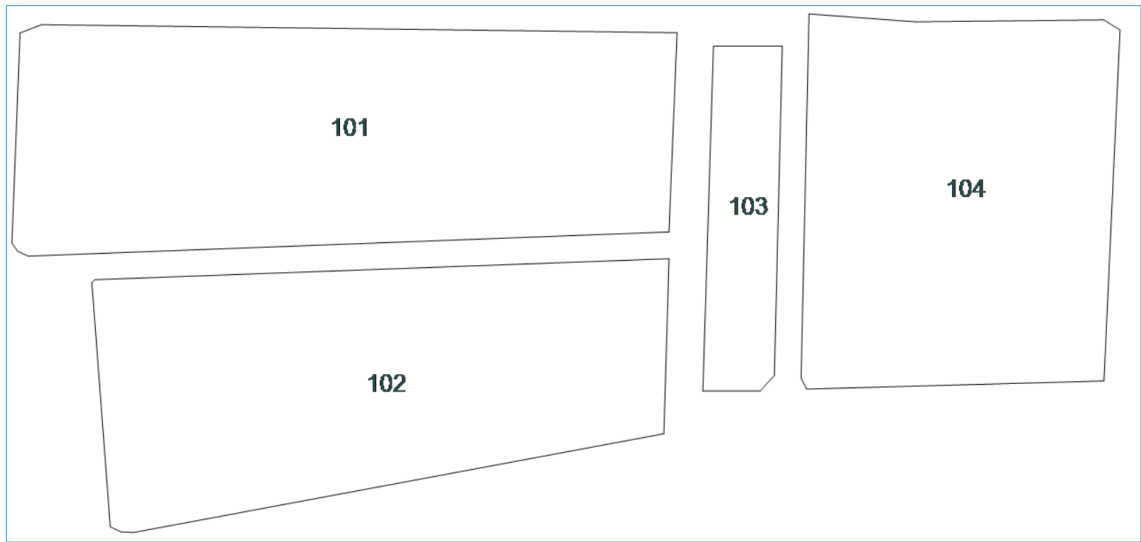
Koc ve Babaoglu (2020) çalışmalarında imar uygulamalarındaki dağıtım ve parselasyon planları için 20 farklı problem içeren sentetik bir veri seti önermişlerdir. Ayrıca dağıtım ve parselasyon problemlerinin çözümü için sürü zekası ve evrimsel tabanlı olmak üzere beş farklı optimizasyon algoritmasını kullanmışlardır. Buna ilaveten probleme özgü olarak harita tabanlı yeni bir çaprazlama ve üç farklı mutasyon operatörü önermişlerdir. Önerilen bu çaprazlama ve mutasyon operatörleri kullanılarak gerçekleştirilen hibrit bir yaklaşım yardımıyla optimizasyon algoritmaları sayesinde önerilen bu veri setindeki problemleri çözmeye çalışmışlardır. Elde edilen deneysel sonuçlarda özellikle GA yönteminin hem hız hem de performans olarak çok başarılı olduğu görülmüştür.

3. MATERYAL VE METOT

3.1. Materyal

3.1.1. Ön parselasyon problemi için proje sahası

Bu tez çalışmasında dağıtım ve parselasyon işlemlerini gerçekleştirmek için öncelikle ön parselasyon işleminin yapılması gerekmektedir. Bunun için Konya İli Meram İlçesindeki bir proje sahasının ön parselasyon işlemi yapılmıştır. Şekil 3.1 ön parselasyon için verilen bir proje sahasını göstermektedir.



Şekil 3.1. Ön parselasyon işlemi için proje sahası

Şekil 3.1’de 4 farklı imar adası bulunmaktadır. Bu adalardan 101 ve 102 nolu adalar yatay şekilde iken, 103 ve 104 nolu adalar ise imar planında dikey şekildedir. Ön parselasyon işleminde adaların yatay ya da dikey olmaları önem arz etmektedir. Bu işlemler Bölüm 4.1’de detaylıca anlatılmaktadır.

3.1.1.1. Ön parselasyon için gerekli kavramlar

a. Ayırık Nizam

Komşu parseldeki yapı ile birleşmeden inşa edilen yapının şekline ayırık nizam denir. Ayırık nizamda, adanın köşesinde yer alan parselin kenar uzunluğu imar yönetmeliğine göre en az 14 metre olmalıdır. Ara parsellerin kenar uzunluğu ise en az 12 m olmalıdır. Ayrıca kat adedi dördüncü kattan sonraki her bir kat için parselin her iki kenar uzunlukları 0.5 metre artırılmalıdır. Bu nedenle iç parsellerin kenar uzunlukları her kat için 1 m. artmaktadır. Denklem 3.1'e göre parselin minimum kenar uzunluğu hesaplanmaktadır. Ayrıca ayırık nizam örnekleri Şekil 3.2'de verilmektedir.

$$W = m + (f - 4) * (0.5 * 2) \quad (3.1)$$

Burada, W parselin minimum kenar uzunluğunu, m yönetmeliğe göre minimum 12m uzunluğunda olduğunu, f kat adedini ifade etmektedir. Bir parsel için her iki yan bahçe kenarı da dikkate alınmalıdır.

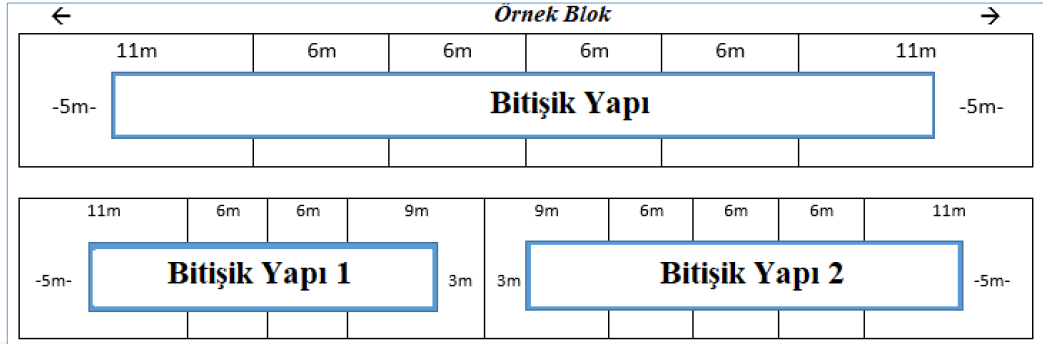
	←	Örnek imar adaları						→
A4 için								
Ara parsel kenar uzunluğu: 12m		14m	12m	12m	...	12m	...	14m
				A4				
A6 için								
Ara parsel kenar uzunluğu: 14m		14m	14m	14m	...	14m	...	14m
				A6				

Şekil 3.2. Ayırık nizamla ait örnekler

b. Blok Nizam

Blok nizam; cephe derinliği, cephe uzunluğu ve bina yüksekliği belirlenmiş olan yapının, birden fazla parsel üzerinde kurulması olarak tanımlanır. Bu nizamda, köşe parsellerin kenar uzunluğu en az 11 m olarak ve ara parsellerin kenar uzunluğu en az 6 m olarak belirlenmektedir. İki ayrı bitişik binanın birleştiği parsellerin kenar uzunlukları

3194 sayılı Kanunun 18. Maddesine göre en az 9 m olmalıdır. Bitişik bir nizamda bir adanın kenar uzunluğu (en uzun kenar) ada içerisinde maksimum 4 adet ara parseli geçmemek şartıyla 40 m'den fazla olmamalıdır. Blok nizam örnekleri Şekil 3.3'te verilmektedir.



Şekil 3.3. Blok nizamda ait örnekler

c. Kat Alan Oranı

Kat Alan Oranı (FAR - *Floor Area Ratio*) Denklem 3.2 yardımıyla hesaplanır. Bu değer imar planlarında verilen girdi parametresidir. Bu çalışmada FAR değeri 0.3 olarak alınmaktadır. Örneğin, bina alanı 150 m² olarak verilmiş ise buna göre binanın yer alacağı parselin alanı en az 500 m² olmalıdır.

$$FAR = \frac{\text{Bina zemin alanı}}{\text{Toplam parsel alanı}} \quad (3.2)$$

3.1.2. Dağıtım ve parselasyon problemi için proje sahası

Bu tez çalışmasında LR problemlerinin çözümü için ilk olarak proje sahasının belirlenmesi gerekmektedir. Uygulama sahası olarak Konya İli Meram İlçesi Harmancık Mahallesi seçilmiştir. Kadastro ve imar adalarına ait genel bilgiler sırasıyla Çizelge 3.1 ve 3.2'de verilmiştir. Ayrıca her bir adanın(bloğun) kaç adet parsel içerdiği de tablolarda verilmiştir.

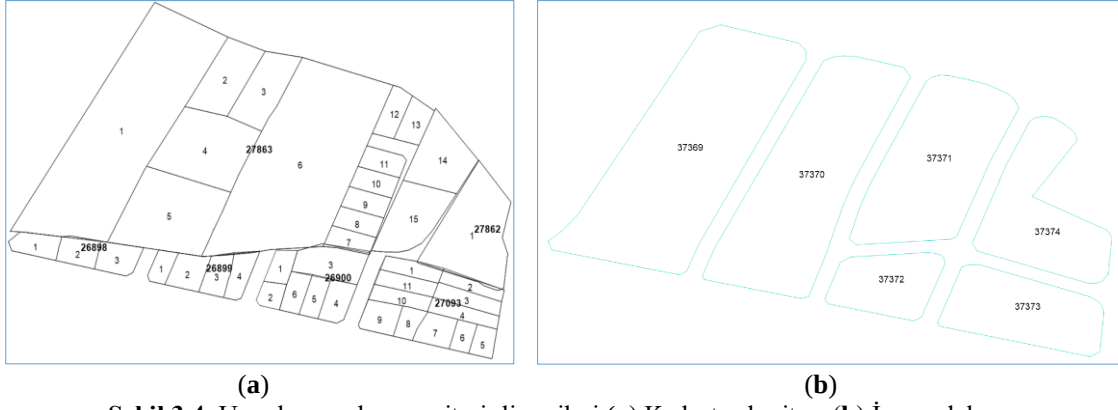
Çizelge 3.1. Kadastro adalarına ait genel bilgiler

Ada No	Ada Adı	Parsel Sayısı	Ada Alanı
1	26898	3	1,886.20
2	26899	4	2,138.58
3	26900	6	2,717.98
4	27093	11	5,431.98
5	27862	1	3,684.11
6	27863	15	40,053.02
Genel Toplam:		40	55,911.87 m ²

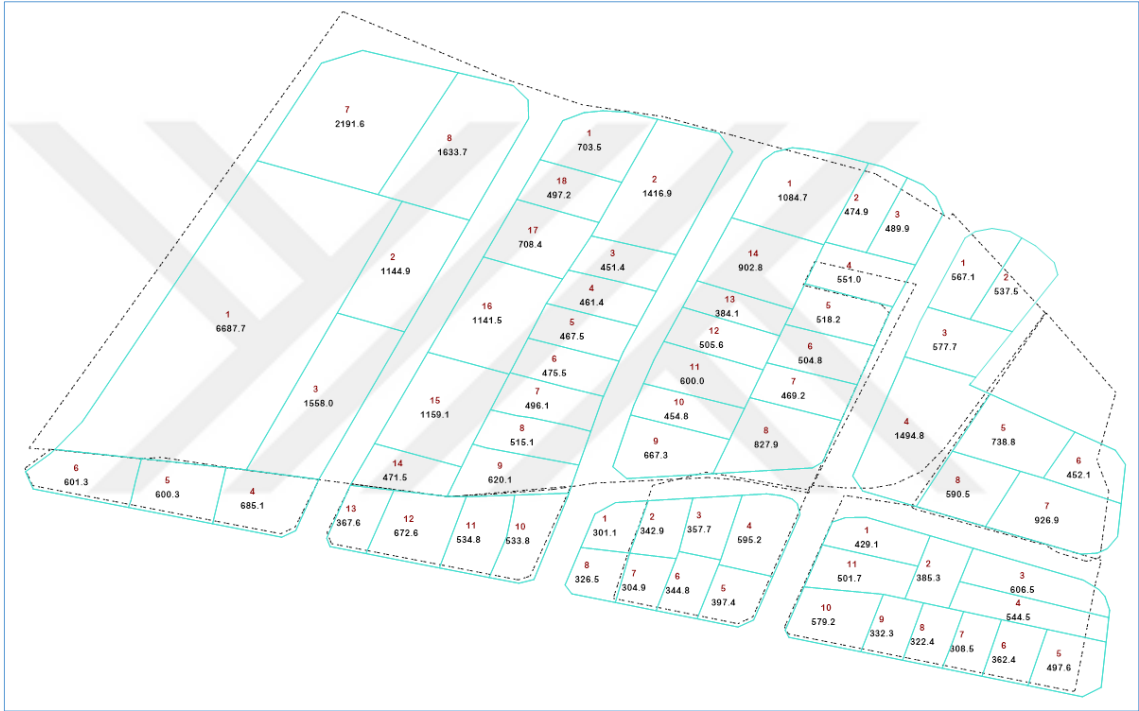
Çizelge 3.2. İmar adalarına ait genel bilgiler

Ada No	Ada Adı	Parsel Sayısı	Ada Alanı
1	37369	8	15,102.488
2	37370	18	11,693.967
3	37371	14	8,435.195
4	37372	8	2,970.537
5	37373	11	4,869.456
6	37374	8	5,885.585
Genel Toplam:		67	48,957.23 m ²

Çizelge 3.2’de yer alan imar ada bilgilerinin yanı sıra Harmancık mahallesinin resmi dağıtım sonuçlarını gösteren parsel bilgileri de yer almaktadır. Çizelge 3.2 incelendiğinde, Harmancık mahallesinin resmi dağıtım sonuçlarında 67 adet parsel olduğu görülmektedir. Kadastro parsellerinin toplam alanı 55911.87 m² iken DOP sonucu tahsis edilen toplam alan 48957.23 m² olarak hesaplanmıştır. Bu verilere göre proje alanında DOP oranı %26.25’tir. İmar Kanununa maksimum DOP oranı %45 olup proje sahasındaki DOP oranı bu oranın altındadır. Bölüm 4’te *ÖNERİLEN YÖNTEMLER* bölümünde, her bir malikin alanından bu oran kadar alan çıkarılarak tahsisleri gerçekleştirilmektedir. Ayrıca, çalışma sahasına ait kadastro parselasyon haritası ve imar adaları Şekil 3.4’te verilmiştir. Kadastro ve imar adalarının aynı harita üzerindeki çakışık durumu ise Şekil 3.5’te birlikte verilmiştir. Bu çalışmada toplam 113 malik bulunmaktadır. Maliklerin DOP kesintisinden sonraki kadastro alanlarına göre dağıtım işlemi gerçekleştirilmiştir.



Şekil 3.4. Uygulama sahasına ait girdi verileri (a) Kadastro haritası (b) İmar adaları



Şekil 3.5. Kadastro ve uzman tarafından gerçekleştirilmiş imar planlarının aynı harita üzerinde gösterimi

3.1.3. Sentetik veri seti üretilmesi

Bu tez çalışmasında gerçek verinin yanı sıra LR problemlerinde kullanılmak üzere sentetik bir veri seti önerilmiştir.

3.1.3.1. Sentetik veri setinin önemi

İmar uygulamalarındaki dağıtım ve parselasyon problemlerinde gerçek bir proje verisinin hazırlanması ve oluşturulması çok uzun zaman alan oldukça karmaşık bir

süreçtir. Kadastro adaları (blokları), parseller ve imar adalarının oluşturulması, tüm ada ve parsellerin konum ve alan verilerinin elde edilmesi ve arsa sahibi (malik) verilerinin oluşturulması bir uzman tarafından manuel olarak gerçekleştirildiği için gerçek proje verilerinin hazırlanması oldukça zordur. Çünkü tüm bu verilerin alan, konum ve malik açısından tamamen uyumlu olması gerekmektedir. Ayrıca, gerçek bir proje alanında genellikle düzgün şekle sahip olmayan adalar ve parseller bulunabilmekte ve kadastro ve imar adaları çok farklı bölgelerde olabilmektedir. Diğer bir deyişle kadastro ve imar alanları genellikle çakışmamaktadır. Ayrıca maliklerin alanları genellikle birbirinden çok farklı olabilmektedir. Bir malik çok küçük bir alana sahip olabilirken, diğeri çok büyük bir alana sahip olabilmektedir. Bu nedenle, önerilen yaklaşımın ne ölçüde başarılı olduğunu ifade etmek ve en uygun çözümün tam olarak ne olduğunu bilmek çok zordur. Buna ilaveten, algoritmanın önerdiği çözümün diğer çözümlerden daha iyi olduğunu söylemek de çok objektif bir karar olmamaktadır. Diğer bir açıdan çözümlerin kalitesi önerilen amaç fonksiyonuna bağlı olarak değerlendirilmektedir. Amaç fonksiyonundaki katsayı değerlerinin değişmesi veya farklı bir amaç fonksiyonunun kullanılması durumunda aday çözümlerin kalite sırası (maliyet sırası) değişebilmektedir. Sonuç olarak, gerçek proje sahasında, çözümlerin kalite sıralaması tamamen amaç fonksiyonuna bağlıdır ve bu durum çözümler arasındaki objektif değerlendirmeyi zorlaştırmaktadır. Bu tez çalışmasında, LR problemlerinin çözümünde kullanılacak gerçek verilerin değerlendirilmesindeki dezavantajlardan dolayı sentetik bir veri seti önerilmiştir. Bu veri seti oluşturulurken ideal çözümün net bir şekilde bilinmesine dikkat edilmiştir. Böylece, önerilen yaklaşımın başarısı çok net ve kolay bir şekilde değerlendirilebilmektedir.

3.1.3.2. Sentetik veri seti özellikleri

Bu tez çalışmasında önerilen veri seti 5 farklı temel problemten oluşmaktadır. Problemler parsel sayılarına göre 20, 40, 60, 80 ve 100 şeklindedir. Her bir problem seti parsel başına düşen arsa sahibi sayısına bağlı olarak 1, 2, 3 ve 4 malik olmak üzere 4 alt problemten oluşmaktadır. Böylece toplamda 20 farklı problem içeren bir veri seti elde edilmiş ve veri setinin özelliği Çizelge 3.3'te verilmiştir. Her bir problem *parsel sayısı* X *parsel başına malik sayısı* kadar farklı malik içermektedir. Örneğin bir problemde 40 parsel ve parsel başına 2 malik varsa, bu durumda bu problem toplam 80 farklı arsa sahibi

içermektedir. Arsa sahiplerinin hangi parsele tahsis edileceğini gözlemlemek için her arsa sahibine bir isim verilmesi gerekmektedir. Bu tez çalışmasında, maliklerin isimleri kolay anlaşılması için alfabetik olarak listelenmiştir. Malik adına ait format $\text{Önek}[x]-[y]$ ile ifade edilirse, Önek A'dan başlamak suretiyle parsel numarasına göre alfabetik olarak devam etmektedir. Parsel indeksi ya da numarası 1 ise isimleri $A[x]-[y]$, indeksi 2 ise adı $B[x]-[y]$ olur. Ayrıca, $[x]$ ifadesi, *parsel başına malik sayısı* kadar numaralandırılmaktadır. Örneğin, bir parselde 4 malik varsa, malik adları $\text{Önek}[1]-[y]$, $\text{Önek}[2]-[y]$, $\text{Önek}[3]-[y]$ ve $\text{Önek}[4]-[y]$ olarak adlandırılmaktadır. Son olarak, $[y]$ ifadesi *ada numarasını* belirtmektedir. Bu çalışmada 2 ada olduğundan, 1. adadaki tüm malik isimleri $\text{Önek}[x]-[1]$ şeklinde ve 2. adadaki tüm malik isimleri ise $\text{Önek}[x]-[2]$ şekilde olmaktadır. Örneğin her bir parselde 2 farklı malik varsa 1. adada 1. parseldeki 1. malik adı A1-1 şeklinde olmaktadır. 2. malik de A2-1 olarak adlandırılmaktadır. Malikler 2. adada ise birincisi A1-2, ikincisi ise A2-2 olarak adlandırılır.

Çizelge 3.3. Sentetik veri setine ait özellikler

Toplam Parsel Sayısı	Parsel Boyutu [Yükseklik x Genişlik]	Ada Boyutu [Yükseklik x Genişlik]
20	20mx15m	40mx5x15m =3000 m ²
40	20mx15m	40mx10x15m =6000 m ²
60	20mx15m	40mx15x15m =9000 m ²
80	20mx15m	40mx20x15m =12000 m ²
100	20mx15m	40mx25x15m =15000 m ²

3.1.3.3. Sentetik veri setine ait örnek problemler

Bu tez çalışmasında sentetik veri setinde toplam 20 problem bir başka deyişle 20 proje sahası yer almaktadır. Her proje alanında aynı büyüklükte iki kadastro adası bulunmaktadır. Proje sahasındaki adalardan aşağıdaki adanın ID No'su 1 iken yukarıdaki adanın ID No'su 2'dir. Her kadastro adasında problem büyüklüklerine göre sırasıyla 10, 20, 30, 40 ve 50 parsel bulunmaktadır. Her bir parsel 20x15m boyutların olup 300 m² alana sahiptir. Maliklerin sayısına göre, her bir malikin alanı her problemde değişmektedir. Malik alanları sırasıyla 1, 2, 3 ve 4 tür ve buna göre malik alanları 300, 150, 100 ve 75 m²'dir. Şekil 3.6 ve 3.7'te iki farklı örnek problem girdisi verilmektedir. Bu örneklerde, Şekil 3.6 20 parsel ve parsel başına 1 malik için verilen kadastro haritası

ve dağıtım listesini sunmaktadır. Şekil 3.7 ise 40 parsel ve parsel başına 2 malik için bir kadastro verilerini ve dağıtım listesini göstermektedir. Projenin kadastro ve malik verileri Şekil 3.6 ve 3.7'deki gibi iken, imar adalarının sınırları kadastro adasının sınırları ile birebir aynıdır. Bundan dolayı yeniden dağıtım listesi ve imar parsellerinden oluşan ideal bir çözümün, Şekil 3.6 ve 3.7'deki gibi kadastroda verilen parsel alanları ve malik sırası açısından birebir aynı olması gerekmektedir. Böylece elde edilen çözümün ceza değeri 0 olmalıdır.

1 A-2 300.0	2 B-2 300.0	3 C-2 300.0	4 D-2 300.0	5 E-2 300.0
10 J-2 300.0	9 I-2 300.0	8 H-2 300.0	7 G-2 300.0	6 F-2 300.0
1 A-1 300.0	2 B-1 300.0	3 C-1 300.0	4 D-1 300.0	5 E-1 300.0
10 J-1 300.0	9 I-1 300.0	8 H-1 300.0	7 G-1 300.0	6 F-1 300.0

Şekil 3.6. Örnek bir kadastro haritası (parsel sayısı =20, parsel başına malik sayısı =1)

1 A1-2 A2-2 300.0	2 B2-2 B1-2 300.0	3 C2-2 C1-2 300.0	4 D2-2 D1-2 300.0	5 E1-2 E2-2 300.0	6 F1-2 F2-2 300.0	7 G1-2 G2-2 300.0	8 H1-2 H2-2 300.0	9 I2-2 I1-2 300.0	10 J2-2 J1-2 300.0
20 T2-2 T1-2 300.0	19 S1-2 S2-2 300.0	18 R1-2 R2-2 300.0	17 Q2-2 Q1-2 300.0	16 P1-2 P2-2 300.0	15 O1-2 O2-2 300.0	14 N2-2 N1-2 300.0	13 M1-2 M2-2 300.0	12 L2-2 L1-2 300.0	11 K1-2 K2-2 300.0

1 A2-1 A1-1 300.0	2 B1-1 B2-1 300.0	3 C2-1 C1-1 300.0	4 D2-1 D1-1 300.0	5 E1-1 E2-1 300.0	6 F1-1 F2-1 300.0	7 G2-1 G1-1 300.0	8 H2-1 H1-1 300.0	9 I2-1 I1-1 300.0	10 J1-1 J2-1 300.0
20 T2-1 T1-1 300.0	19 S2-1 S1-1 300.0	18 R2-1 R1-1 300.0	17 Q1-1 Q2-1 300.0	16 P2-1 P1-1 300.0	15 O2-1 O1-1 300.0	14 N1-1 N2-1 300.0	13 M1-1 M2-1 300.0	12 L1-1 L2-1 300.0	11 K1-1 K2-1 300.0

Şekil 3.7. Örnek bir kadastr haritası (parsel sayısı =40, parsel başına malik sayısı =2)

3.2. Metot

3.2.1. İkili Arama Yöntemi (BS)

İkili arama yöntemi, bilgisayar bilimi uygulamalarında yaygın olarak kullanılan ve uygulanması hızlı ve kolay olan basit bir arama tekniğidir (Haklı ve ark., 2016; Ayop ve Tan, 2017). BS, standart bölme ve fethetme mantığına dayanmaktadır. Bu yöntemin önemli bir avantajı, arama süresini önemli ölçüde azaltmasıdır. BS'nin temel amacı, kısa sürede istenen veriler için aranan değerin bulunmasıdır. BS algoritması, özellikle arama alanında çok sayıda veri olduğunda, doğrusal arama yöntemine kıyasla çok hızlı bir arama yapma özelliğine sahiptir. Doğrusal bir arama tekniği, N adet sıralı elemanın olduğu bir veri seti için istenen elemana $O(N)$ adımda ulaşırken, aynı veri seti için BS $O(\log N)$ adımda istenen elemana ulaşmaktadır. BS için gereken yineleme sayısı (N_{BS}) Denklem 3.3 ile hesaplanır. Bu, yineleme sayısında büyük bir azalma anlamına gelmektedir. Örneğin 1000 veriden oluşan bir veri setinde istenilen elemana maksimum 10 yinelemeyle erişilebilmektedir. Bu sonuç, BS ile başarılı bir aramanın, çok daha büyük bir veri seti için nispeten çok daha kısa bir sürede gerçekleştirilebileceğini göstermektedir.

$$N_{BS} = \log_2 N \quad (3.3)$$

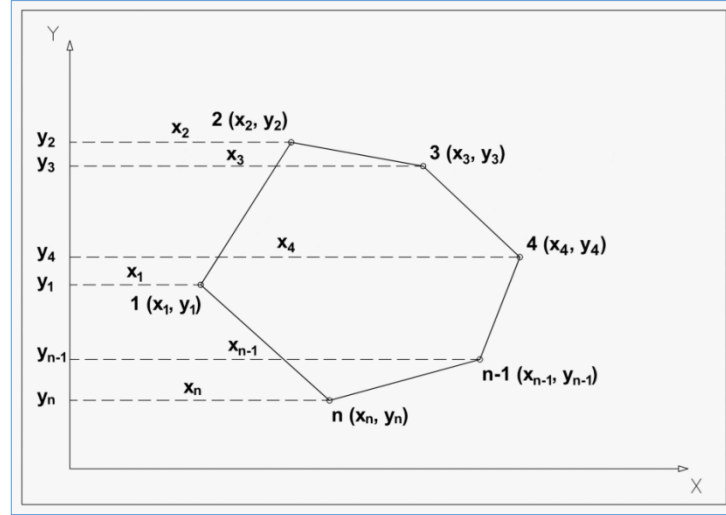
BS'nin çalışma mantığı oldukça basittir. İlk olarak, BS sıralı veriler üzerinde çalıştığı için, veri hali hazırda sıralı değilse doğru şekilde sıralanmalıdır. Sıralama işleminden sonra BS, dizinin ortasındaki elemanı dikkate alarak aranan elemanı (anahtar) karşılaştırır. Anahtar ortadaki elemandan daha küçükse, anahtar yalnızca veri kümesinin ilk yarısında aranır. Anahtar ortadaki elemana eşitse arama başarılı olur. Anahtar orta elemandan daha büyükse, anahtar yalnızca veri kümesinin ikinci yarısında aranır. Bu adımlar anahtar bulunana kadar devam eder.

3.2.2. Gauss yöntemi

Literatürde, düzgün olmayan geometrik şekillerin alanını hesaplamak için Delaunay üçgenlemesi (Delaunay, 1934) ve Gauss yöntemleri gibi farklı teknikler bulunmaktadır. Gauss yöntemi koordinat sistemi üzerinde çalıştığı için bu tez çalışmasında parsellerin alanlarını hesaplamak için bu yöntem tercih edilmiştir (Songu ve ark., 2009). Bu yönteme göre:

1. Her bir parselin kenarlarını tanımlayan n tane nokta $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ şeklinde art arda belirlenir.
2. Bu nokta kümesinden iki ardışık nokta belirlenir. İki ardışık nokta ve y eksenini tarafından oluşturulan yamuğun alanı hesaplanır. Örneğin, ilk iki nokta tarafından tanımlanan yamuk oluşturan noktaların koordinatları sırasıyla $(x_1, y_1), (x_2, y_2), (0, y_2)$ ve $(0, y_1)$ şeklindedir.
3. Bu işlem, parsel üzerindeki tüm sıralı nokta çiftleri dikkate alınarak arka arkaya gerçekleştirilir.
4. Elde edilen düşey yamukların alanları toplanarak parselin toplam alanı hesaplanır.

Basit bir çokgenin (parsel) alanı Şekil 3.8'de verilmektedir. Ayrıca çokgen alanının hesabı Denklem 3.4 ve 3.5'te verilmektedir.



Şekil 3.8. Basit bir poligonun alanı

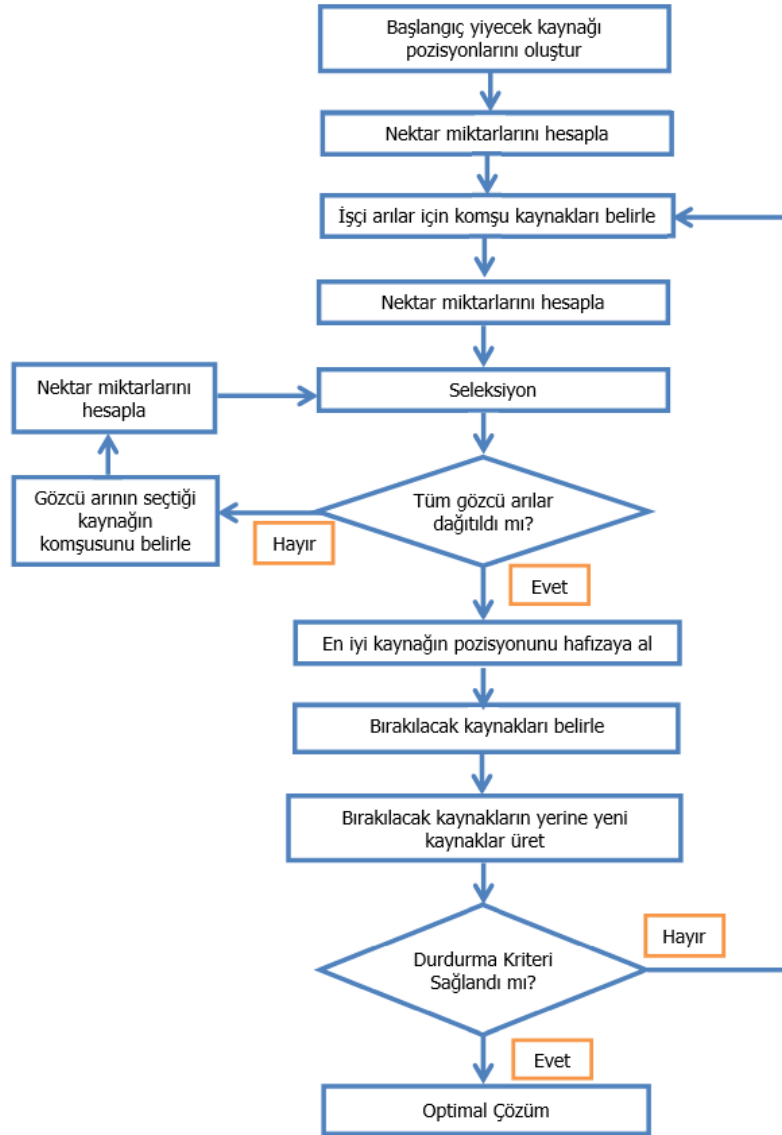
$$2A = (x_3 + x_2) \cdot (y_2 - y_3) + (x_4 + x_3) \cdot (y_3 - y_4) + (x_{n-1} + x_4) \cdot (y_4 - y_{n-1}) + (x_n + x_{n-1}) \cdot (y_{n-1} - y_n) - (x_1 + x_2) \cdot (y_2 - y_1) - (x_n + x_1) \cdot (y_1 - y_n) \quad (3.4)$$

Denklem 3.2'deki A , çokgenin alanını temsil etmektedir. Ayrıca, Denklem 3.4'ün genelleştirilmiş şekli Denklem 3.5'te verilmektedir.

$$2A = \sum_{i=1}^n [x_i(y_{i-1} - y_{i+1})] \text{ veya } 2A = \sum_{i=1}^n [y_i(x_{i-1} - x_{i+1})] \quad (3.5)$$

3.2.3. Yapay arı koloni algoritması (ABC)

Yapay arı koloni algoritması, bal arısı sürülerinin belirli bir akıllı davranışına dayanan bir optimizasyon algoritmasıdır (Karaboga ve Basturk, 2008). ABC algoritmasında, yapay arı kolonisi üç farklı arı grubundan oluşur: işçi arılar, gözcü arılar ve kâşif arılar. Dans alanında bir besin kaynağı seçmeye karar vermek için bekleyen arıya işçi arı, işçi arının paylaştığı bilgiler ışığında daha önce ziyaret ettiği besin kaynağına giden ikinci arı grubuna da gözcü arı adı verilmektedir (Karaboga, 2005). Rastgele besin kaynağı arayan üçüncü bir arı grubu da kâşif arı olarak isimlendirilmektedir. ABC algoritmasında, koloninin ilk yarısı çalışan arılardan oluşur ve sürünün ikinci yarısı gözcü arılardan oluşmaktadır. Diğer bir deyişle, işçi ve gözcü arıların sayısı, arı popülasyonundaki aday çözümlerin sayısına eşittir (Karaboga ve Basturk, 2007; Karaboga ve Akay, 2009). ABC algoritmasına ait akış şeması Şekil 3.9'da detaylıca verilmektedir.



Şekil 3.9. ABC algoritmasının akış şeması

3.2.3.1. Başlangıç evresi

ABC algoritması, mevcut çözüme karşılık gelen yiyecek kaynağının rastgele belirlenmesiyle başlar. Başlangıç popülasyonuna ait aday çözümler Denklem 3.6 kullanılarak üretilmektedir.

$$X_{ij}(t = 0) = x_j^{min} + (x_j^{max} - x_j^{min}) \times rand[0, 1] \quad (3.6)$$

for $i = 1, 2, \dots, N$ ve $j = 1, 2, \dots, D$

Denklem 3.6'da yer alan N sayısı, kolonideki arı sayısını, D arama uzayının boyutunu, x_j^{min} ve x_j^{max} arama uzayında sırasıyla j . boyutu için alt ve üst limitleri ifade etmektedir. $rand[0,1]$ ise 0 ve 1 arasında rastgele olarak seçilen ondalık bir değerdir.

3.2.3.2. İşçi arı evresi

Bu aşamada, her bir işçi arı için yeni bir aday çözüm üretilmektedir. İlk olarak, işçi arının çözümü yeni adaya ait çözüme kopyalanır ($V_i = X_i$). Daha sonra, çözümün bir parametresi Denklem 3.7 kullanılarak güncellenir.

$$V_{ij}(t) = x_{ij}(t-1) + \varphi \left(x_{kj}(t-1) - x_{ij}(t-1) \right) \quad (3.7)$$

for $j = 1, 2, \dots, D$

Buradaki $k \in (1, 2, \dots, N)$ ve $j \in (1, 2, \dots, D)$ verilen aralıklarda rastgele seçilen indekslerdir. Ayrıca k değeri i değerinden farklı olmalıdır. φ_{ij} ise $(-1, 1)$ aralığında rastgele üretilen bir sayıdır. Yeni aday çözüm bulunduğundan ve amaç fonksiyon değerinin hesaplanmasından sonra, aday çözümlerin uygunluk değeri ve işçi arılardaki çözümler Denklem 3.8'deki gibi hesaplanır.

$$fit_i = \begin{cases} \frac{1}{1 + f_i} & \text{if } f_i \geq 0 \\ 1 + abs(f_i) & \text{değilse} \end{cases} \quad (3.8)$$

Denklem 3.8'deki fit_i , i . aday çözümün uygunluk değeri, f_i ise i . işçi arının amaç fonksiyon değeridir. Güncellenmiş aday çözümün uygunluk değeri, mevcut çözümün uygunluk değerinden daha iyi ise, mevcut çözüm aday çözüm ile değiştirilir ve işçi arının limit değeri sıfırlanır, aksi halde limit değeri bir artırılır.

3.2.3.3. Gözcü arı evresi

ABC algoritmasında, çözümü iyileştirmek için her bir gözcü arı, bir işçi arıyı seçer. İşçi arının seçilmesi olasılığı, orijinal ABC algoritmasında Denklem 3.9'daki gibi rulet tekerleği seçimi kullanılarak hesaplanır.

$$P_i = \frac{fit_i}{\sum_{i=1}^{SN} fit_i} \quad (3.9)$$

Burada P_i , i . işçi arıyı seçme olasılığıdır. Seçilen işçi arının çözümü, Denklem 3.8 kullanılarak gözcü arılar tarafından iyileştirilir. Gözcü arı tarafından bulunan yeni çözümün uygunluk değeri işçi arının çözümünden daha iyi ise gözcü arı işçi arı ile yer değiştirir ve kullanılan arıya ait limit değeri sıfırlanır, aksi halde limit değeri bir artırılır.

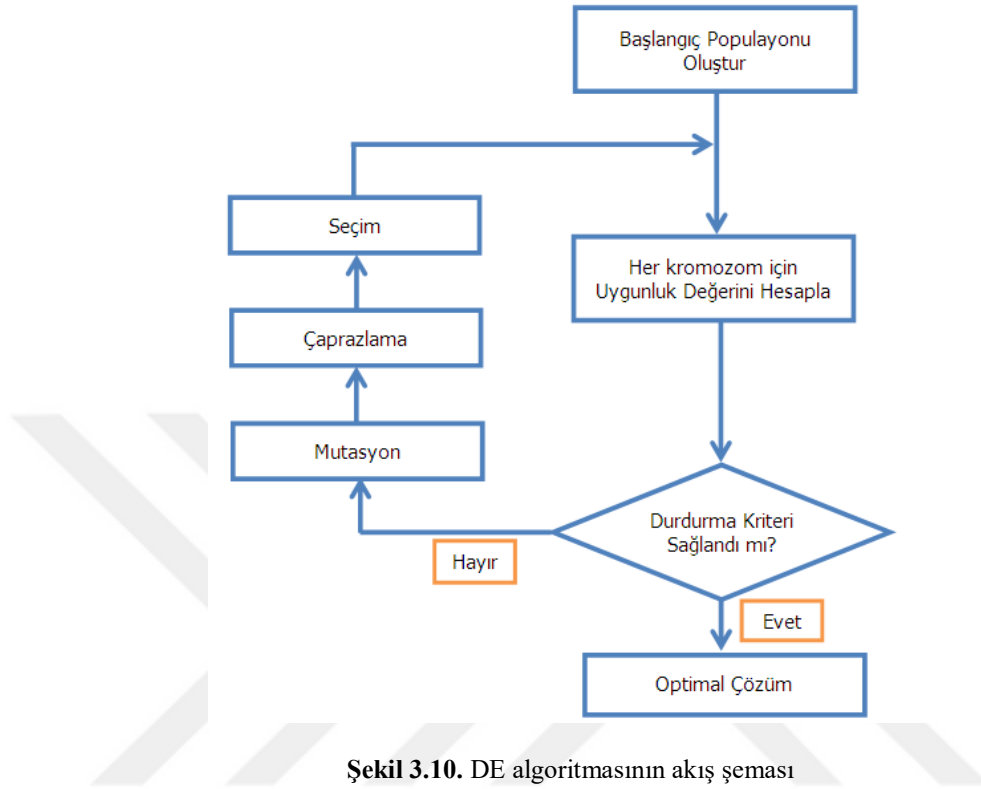
3.2.3.4. Kâşif arı evresi

Her bir işçi arının limit değeri önceden tanımlanmış bir limit değeri ile kontrol edilir. Limit değerine ulaşmış olan işçi arı, kâşif arıya dönüşür. Bundan sonra, Denklem 3.7 kâşif arı için çözüm üretmek için kullanılır ve limit değeri sıfırlanır. Kâşif arı sonra işçi arı olur. Bundan dolayı kâşif arı arı sürüsünün yerel en iyiye takılmasını engellemektedir.

3.2.3. Diferansiyel gelişim algoritması (DE)

Diferansiyel gelişim algoritması, özellikle sürekli problemlerde pek çok farklı optimizasyon problemini çözmek için etkili ve güçlü bir popülasyon tabanlı optimizasyon algoritmasıdır (Storn ve Price, 1997; Onwubolu ve Babu, 2004; Qin ve ark., 2008). İyi yakınsama hızının yanı sıra, DE'yi anlamak ve uygulamak çok basittir (Storn, 1996). DE, yakınsama ve kararlılık açısından diğer birçok optimizasyon algoritmasından daha iyi performans göstermektedir. Diğer optimizasyon algoritmaları gibi DE, rastgele oluşturulan bir başlangıç popülasyonu ile başlar (Rahnamayan ve ark., 2008). Ayrıca, DE'de operatörlerin çalışma prensibi GA'dan farklı olsa da, GA'ya benzer şekilde

çaprazlama, mutasyon ve seçim operatörleri bulunmaktadır. DE'nin akış şeması Şekil 3.10'da sunulmaktadır.



3.2.3.1. Mutasyon

Mutasyon operatörü, DE'nin diğer evrimsel algoritmalarından farklı olmasını sağlayan temel operatörüdür. Mutasyon, mevcut kromozomun bir kısım genleri üzerinde, rasgele belirlenmiş miktarlarda değişiklikler yapılmasını ifade etmektedir. Bu değişiklikler sayesinde kromozomun temsil ettiği çözüm noktası, çözüm uzayı içerisinde belli ölçüde hareket etmektedir. Bu işlemin hedefine ulaşabilmesi için, doğru yönde ve miktarda hareketi sağlayacak değişikliklerin belirlenmesi gerekmektedir. Her kuşaktaki mutasyon işlemi, popülasyondaki üç farklı kromozom rastgele seçilerek gerçekleştirilmektedir. DE'de en sık kullanılan mutasyon stratejileri Denklem 3.10 - 3.14'te verilmektedir.

$$DE/rand/1: \quad V_{i,g} = X_{r1,g} + F * (X_{r2,g} - X_{r3,g}) \quad (3.10)$$

$$DE/rand/2: \quad V_{i,g} = X_{r1,g} + F * (X_{r2,g} - X_{r3,g}) + F * (X_{r4,g} - X_{r5,g}) \quad (3.11)$$

$$DE/best/1: \quad V_{i,g} = X_{best,g} + F * (X_{r1,g} - X_{r2,g}) \quad (3.12)$$

$$DE/best/2: \quad V_{i,g} = X_{best,g} + F * (X_{r1,g} - X_{r2,g}) + F * (X_{r3,g} - X_{r4,g}) \quad (3.13)$$

$$DE/rand-to-best/1: \quad V_{i,g} = X_{x1,g} + F * (X_{best,g} - X_{r2,g}) + F * (X_{r3,g} - X_{r4,g}) \quad (3.14)$$

NP popülasyondaki kromozom sayısını ifade etmek üzere, Denklem 3.10-3.14'te yer alan $i = 1, \dots, NP$, $r1, r2, r3 \in [1, \dots, NP]$ olmak üzere bu değerler rastgele seçilir. Ayrıca $r1 \neq r2 \neq r3 \neq i$ şartının sağlanması gerekmektedir. Ayrıca $F \in [0, 1]$ olarak tercih edilen mutasyona ait bir ölçekleme parametresidir.

3.2.3.2. Çaprazlama

Çaprazlama yapılırken, mutasyon sonucu elde edilen fark kromozomu ($V_{i,G}$) ve mevcut kromozom ($X_{i,G}$) kullanılarak yeni jenerasyon için aday deneme kromozomu ($U_{i,G+1}$) üretilir. Aday deneme kromozomu oluşturulurken aday kromozomdaki her bir gen CR olasılıkla fark kromozomundan alınırken, $1 - CR$ olasılıkla ise mevcut kromozomdan alınır. Çaprazlama operatörü Denklem 3.15'te verilmektedir.

$$U_{j,i,G+1} = \begin{cases} V_{j,i,G+1} & \text{if } rand_j \leq CR \vee j = k \\ X_{j,i,G+1} & \text{değilse} \end{cases} \quad (3.15)$$

Denklem 3.15'te yer alan $j = 1, \dots, n$ ve $k \in [1, \dots, n]$ olmak üzere k her i için bir kez seçilen rastgele parametre indeksidir. Böylece en az bir genin yeni üretilen kromozomdan seçilmesi garanti edilir. Diğer taraftan $rand_j \in [0, 1]$ olmak üzere kontrol parametresi olan çaprazlama oranı $CR \in [0, 1]$ olup kullanıcı tarafından belirlenir.

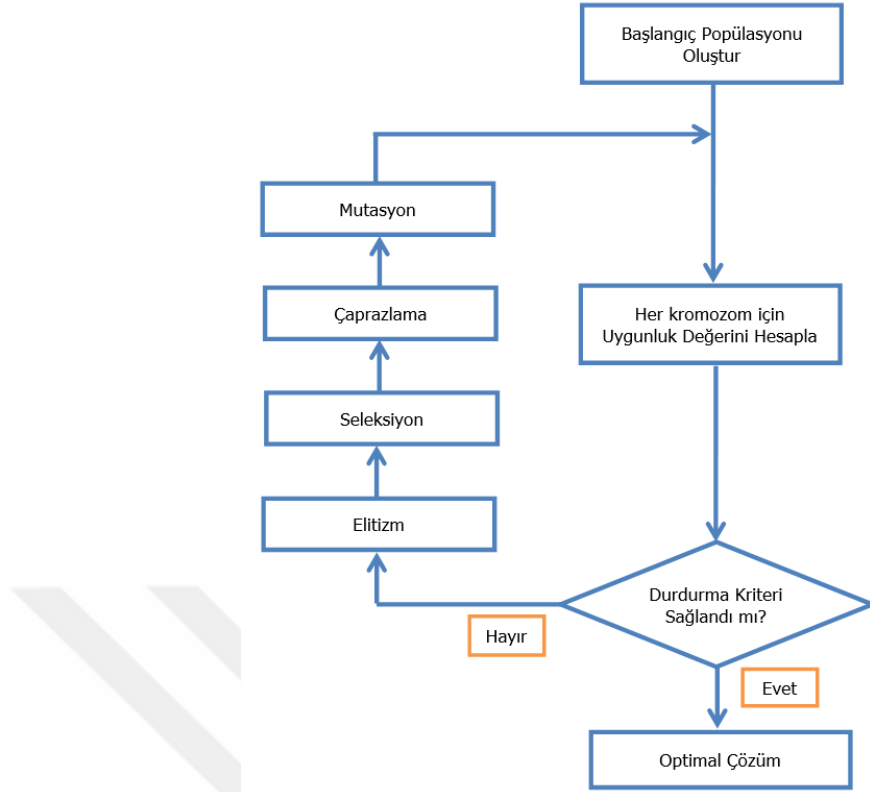
3.2.3.3. Seçim

DE algoritmasının seçim evresi diğer evrimsel algoritmalarından farklıdır. Bir sonraki jenerasyon için ilgili kromozom, Denklem 3.16'daki kurala göre aç gözlü bir yaklaşımla belirlenir. Yeni aday çözümün amaç fonksiyon değeri daha iyi ise kromozom yeni aday kromozomla yer değiştirilir aksi halde herhangi bir değişiklik yapılmaz. Seçim operatörü Denklem 3.16'da sunulmaktadır.

$$X_{i,G+1} = \begin{cases} U_{i,G+1} & \text{if } f(U_{i,G+1}) \leq f(X_{i,G}) \\ X_{i,G} & \text{değilse} \end{cases} \quad (3.16)$$

3.2.4. Genetik algoritma (GA)

Genetik algoritma, evrimsel biyolojide kromozomların gen yapısındaki rastgele değişikliklerle oluşan türler arasındaki en uygun türlerin hayatta kalmasını taklit etmeye dayanmaktadır. İlk olarak 1975 yılında Holland tarafından doğal adaptasyondan esinlenerek geliştirilmiş ve daha sonra Goldberg'in çalışmaları ile birçok alanda kullanılmıştır (Goldberg ve Holland, 1988; Goldberg, 1989). Basit bir GA, gelecek nesillere yeni ve daha kaliteli bireyler oluşturmak için, gen havuzu adı verilen bir başlangıç popülasyonu oluşturarak çalışır. Bu işlemde sonra GA, sırasıyla seçim, çaprazlama ve mutasyon adlı üç temel operatörü uygulamaktadır. İlk operatör, popülasyondaki kromozomların (dizelerin) amaç fonksiyon değerine göre elde edilen bir olasılık yardımıyla rastgele seçildiği seçim operatörüdür. İkinci operatör, seçim operatörü tarafından belirlenen her bir çift kromozomun çaprazlandığı ve yeni dizelerin oluşturulduğu çaprazlama operatörüdür. Üçüncü operatör ise, çaprazlama işleminden sonra bir kromozom üzerindeki herhangi bir değerin veya değerlerin belirli olasılık değerine dayalı olarak rastgele değişmesine izin veren mutasyon operatörüdür. Çaprazlama operatörü ve seçim operatörü GA'nın arama kabiliyetini güçlendiren en önemli operatörlerdir. Mutasyon ise, kromozomların arama uzayında farklı noktalara hareket etmesine izin vererek popülasyon içinde çeşitliliği sağlayan ve üreme ve çaprazlamanın neden olabileceği genetik materyal kaybını önlemeye yarayan operatördür. Bu nedenle mutasyon oranı çok düşük seçilirken, çaprazlama oranı yüksek seçilmektedir (Ahmed, 2010). Bu üç temel operatöre ek olarak, elitizm operatörü de popülasyondaki güçlü bireylerin hayatta kalmasını sağlamak için kullanılır. GA'nın akış diyagramı Şekil 3.11'de verilmektedir.



Şekil 3.11. GA'nın akış şeması

3.2.4.1. Seçim Operatörü

Üreme veya diğer deyişle seçim sürecinde, kromozomlar bir sonraki nesil çiftleşme havuzuna, onların uygunluk değerleri ile ilişkili bir olasılıkla kopyalanır. Yüksek uyum gösteren kromozomların daha yüksek bir kısmının yeni nesle tahsis edilmesiyle, GA'daki seçim evresi doğal dünyadaki hayatta kalma oranını taklit etmektedir. Bu aşamada yeni kromozom üretilmez ve bu işlem, algoritmanın her bir jenerasyonunda meydana gelir. Burada ne kadar kromozomun seçileceğine karar vermek biraz keyfidir. Ayrıca popülasyon içerisinde bütün kromozomların seçilmesinin gelecek nesle çok fazla katkısı olmamaktadır. Literatürde seçim operatörü olarak kullanılan birçok yöntem bulunmaktadır. Bu yöntemlerden en sık kullanılanları rulet tekerleği, rastgele seçme ve turnuva yöntemidir (Weile ve Michielssen, 1997). Bu yöntemler arasından turnuva seçiminin diğer seçim yöntemlerine göre en önemli avantajı ise dizi grupları arasındaki tercih sırasını gerektirmesidir. Bu durum seçim operatörünün uygun bir amaç fonksiyonu olmayan durumlarda bile çalışabilmesini sağlamaktadır (Aytuğ, 2013). Ayrıca turnuva yöntemi, uygunluk fonksiyonunun değerlendirilmesinin zor olduğu

durumlarda da oldukça kullanışlıdır (Reeves ve Rowe, 2002). Bundan dolayı GA ve ABC yöntemlerinde seçim işlemi için daha çok turnuva yöntemleri tercih edilmiştir. Sadece gerçek proje üzerinde yapılan deneysel çalışmalarda GA için rulet tekerleği turnuva yöntemine nazaran deneysel olarak daha başarılı sonuçlar ortaya koyduğu için bu yöntem tercih edilmiştir.

3.2.4.2. Çaprazlama Operatörü

Genetik algoritmada en önemli arama işlemi olan çaprazlama, popülasyonda bulunan iki farklı kromozomdan yeni kromozomlar oluşturularak gerçekleştirilir. İlk olarak, bir çift ebeveyn çiftleşme havuzundan seçim operatörü yardımıyla rastgele seçilir. Ardından, çaprazlama noktası adı verilen bir nokta rastgele seçilir ve iki ana dizinin çaprazlama noktasına göre işlem yapıldıktan sonra bilgi değişimi gerçekleştirilir ve böylece iki yeni çocuk oluşturulur (Abdoun ve Abouchabaka, 2012). Çaprazlama noktası olarak uygun bir nokta belirlendiğinde ebeveyn kromozomlardan uyum değeri daha yüksek çocuk kromozom veya kromozomlar üretilmektedir. Fakat uygun çaprazlama noktası önceden bilinemediğinden rastgele olarak bir çaprazlama noktası seçilmektedir. Literatürde temel çaprazlama yöntemleri olarak tek noktalı ve çift noktalı çaprazlama yöntemleri bulunmaktadır.

a. Tek noktalı çaprazlama yöntemi

Kromozom havuzundaki rastgele eşleşen her kromozom çifti için ilk ve son gen dışında aradaki genlerden rastgele bir nokta belirlenir. Burası çaprazlama noktasını göstermektedir. Bu noktadan sonra gelen genler her iki dizide karşılıklı olarak yer değiştirir ve böylece iki farklı çocuk kromozom elde edilir. Bu işlemin gerçekleşmesi için diziler aynı uzunlukta olmalıdır (Çayıroğlu, 2013). Tek noktalı çaprazlama yöntemi için örnek bir çaprazlama Şekil 3.12’de verilmektedir.

Ebeveyn Kromozom 1:	1	0	0	1	1	1	1	0
Ebeveyn Kromozom 2:	1	0	1	1	0	0	1	0
Çocuk Kromozom 1:	1	0	0	1	0	0	1	0
Çocuk Kromozom 2:	1	0	1	1	1	1	1	0

Şekil 3.12. Tek noktalı çaprazlama yöntemi için örnek bir çaprazlama

b. Çift noktalı çaprazlama yöntemi

Bu yöntemde kromozom yapısı üzerinde iki tane rastgele nokta seçilerek bu iki nokta arasındaki genlerin değişimi yapılarak çaprazlama işlemi gerçekleştirilir. Böylece iki farklı çocuk kromozom elde edilir. Çift noktalı çaprazlama yöntemi için örnek bir çaprazlama Şekil 3.13'te verilmektedir.

Ebeveyn Kromozom 1:	1	0	0	1	1	1	1	0
Ebeveyn Kromozom 2:	1	0	1	1	0	0	1	0
Çocuk Kromozom 1:	1	0	1	1	0	1	1	0
Çocuk Kromozom 2:	1	0	0	1	1	0	1	0

Şekil 3.13. Çift noktalı çaprazlama yöntemi için örnek bir çaprazlama

3.2.4.3. Mutasyon Operatörü

Her çaprazlama işleminden sonra oluşan iki yeni birey (çocuk) yeni neslin oluşturulması için son adımda mutasyon operatörüne tabi tutulur. Bu operatör rastgele seçilen noktalarda bir veya daha fazla gen değerini söz konusu kromozomda rastgele değiştirir. Mutasyon operatörü, GA'nın en iyi çözüme yakın çözüm bulmasını sağlamak için, popülasyondaki genetik çeşitliliği muhafaza ederek tüm çözüm uzayının en iyi aramada kullanıldığından emin olmak için gerekli olan arama yeteneğini geliştirir. Mutasyon problemin herhangi bir yerel en iyi noktaya takılmasını engellemektedir. Mutasyon oranları genellikle düşük tutulmaktadır. Bunun nedeni çaprazlama sonucu elde

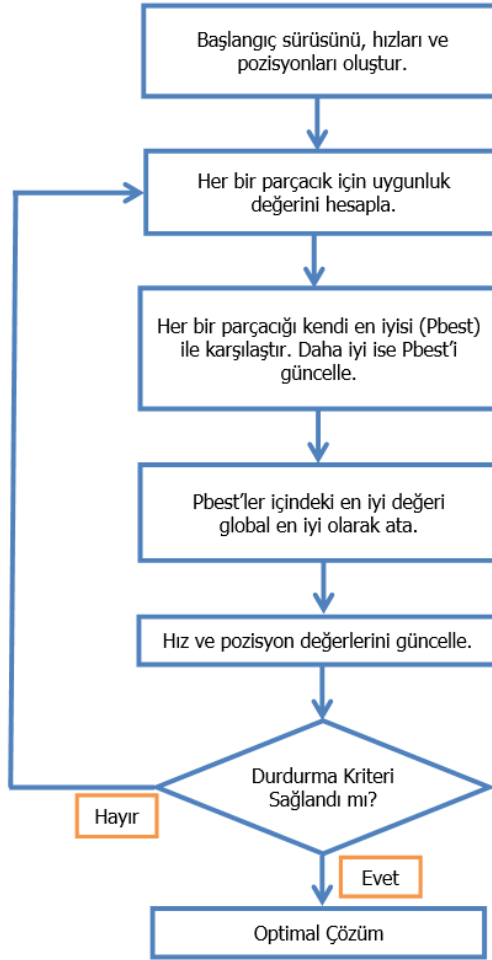
edilen uyum değeri yüksek dizileri kaybetmemektir (Abdoun ve Abouchabaka, 2012). Mutasyon operatörü için örnek bir mutasyon işlemi Şekil 3.14’te verilmektedir.

<i>Ebeveyn Kromozom 1:</i>	1	0	0	1	1	1	1	0
<i>Çocuk Kromozom 1:</i>	1	0	0	0	1	1	1	0

Şekil 3.14. Mutasyon operatörü için örnek bir mutasyon işlemi

3.2.5. Parçacık sürü optimizasyonu algoritması (PSO)

Parçacık sürü optimizasyonu algoritması, balık, kuş ve böcek sürülerinin iki boyutlu hareketlerinden esinlenerek geliştirilmiş, doğadan ilham alan sürü zekası tabanlı bir optimizasyon algoritmasıdır (Eberhart ve Kennedy, 1995a; Kennedy ve Eberhart, 1997). PSO, rastgele bir başlangıç popülasyonu ile başladığı için diğer popülasyon tabanlı algoritmalara benzemektedir. PSO algoritması, bireyler arasında sosyal bilgi paylaşımına dayanmakta ve algoritmada popülasyonu oluşturan bireyler, belirli sayıda nesil boyunca çözüm uzayında farklı konumlara geçerek arama yaparlar (Eberhart ve Kennedy, 1995b). Her bir birey, parçacık; parçacık popülasyonu ise sürü olarak adlandırılır. Her bir parçacık, sürüdeki en iyi konuma sahip parçacığa yakınsamak için önceki deneyimini kullanarak kendi konumunu günceller (Kennedy ve Eberhart, 1995). Sürüdeki en iyi değere sahip parçacık *Gbest* (*global best*) olarak adlandırılırken, mevcut nesle kadar elde edilen parçacığın en iyi değeri *Pbest* (*personel best*) olarak adlandırılır ve bu değerler algoritmanın hafızasında saklanır (Özçakar ve Bastı, 2012). PSO algoritması temel olarak, sürüdeki *Gbest* ve *Pbest* ışığında her parçacığın konumunun daha iyi bir konuma yakınsamasına dayanmaktadır. PSO'nun akış diyagramı Şekil 3.15'te verilmektedir.



Şekil 3.15. PSO algoritmanın akış şeması

PSO algoritmasında parçacıkların konum ve hızlarının hesaplanmasında Denklem 3.17 ve 3.18 kullanılır.

$$V_i^{t+1} = w * V_i^t + c_1 * rand_1 * (Pbest_i - P_i^t) + c_2 * rand_2 * (Gbest - P_i^t) \quad (3.17)$$

$$P_i^{t+1} = P_i^t + V_i^{t+1} \quad (3.18)$$

Burada V_i^{t+1} i. parçacığın $(t+1)$ anındaki hızını, V_i^t i. parçacığın t anındaki hızını, c_1 ve c_2 $[0, 2]$ aralığında değerler alan pozitif sabitleri, $rand_1$ ve $rand_2$ $[0, 1]$ arasında seçilen rastgele sayıları ifade etmektedir. $Pbest_i$ i. parçacığın en iyi konumunu, $Gbest$ ise kuş sürüşü içindeki en iyi konumu ifade etmektedir. Ayrıca, P_i^t i. parçacığın t anındaki konumunu ifade etmektedir. Her bir ajanın hızı Denklem 3.17'ye göre değiştirilmiş ardından ajanın pozisyonu Denklem 3.18 kullanılarak güncellenmiştir. w değeri ise

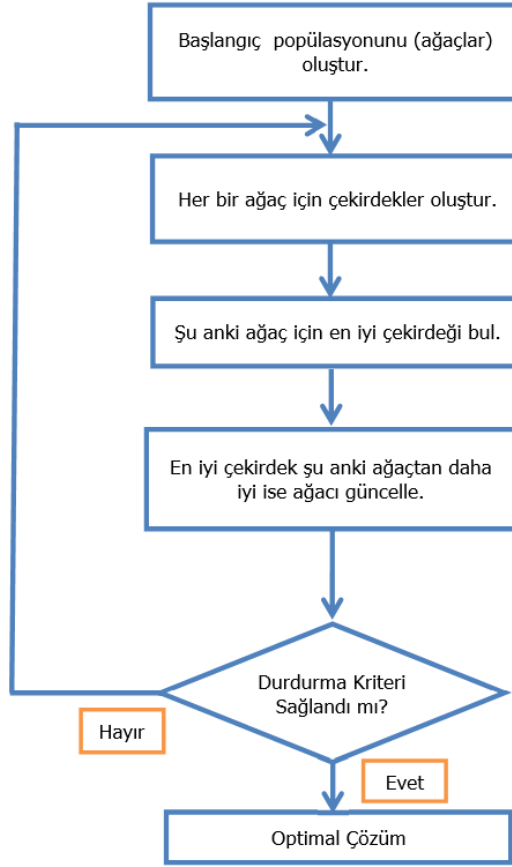
parçacığın hız güncellenmesini ve aynı zamanda yakınsamasını önemli derecede etkileyen bir parametre olup bunun için literatürde çok farklı teknikler önerilmiştir (Bansal ve ark., 2011). Algoritma temel olarak aşağıdaki basamaklardan oluşur (Özsağlam ve Çunkaş, 2008);

1. Başlangıç pozisyonları ve hızları rastgele üretilerek bir başlangıç sürüsü oluşturulur.
2. Sürü içindeki tüm parçacıklar için uygunluk değerleri hesaplanır.
3. Her bir parçacık için mevcut iterasyondaki yerel en iyi değer (*pbest*) bulunur.
4. Mevcut iterasyondaki yerel en iyiler içerisinde global en iyi değer (*gbest*) seçilir.

3.2.6. Ağaç tohum algoritması (TSA)

Ağaç tohum algoritması, ilk kez 2015 yılında sürekli optimizasyon problemlerini çözmek için geliştirilen yeni sürü zekası algoritmalarından biridir (Kiran, 2015; Babalık ve ark., 2018). Doğal TSA fenomeni, ağaçlar ve tohumlar arasındaki ilişkiye dayanır. Diğer optimizasyon algoritmalarına benzer şekilde, optimizasyon problemi için rastgele bir popülasyon oluşturulur. Bu algoritmadaki aday çözümler ağaç ve tohumlardan oluşmaktadır (Cinar ve Kiran, 2018). İterasyonlar boyunca ağaçlar ve tohumlar arasındaki ilişki dikkate alınarak yeni aday çözümler elde edilir. Olası bir aday çözüm olan yeni bir tohum oluştururken, TSA'nın güncelleme aşamasında en iyi çözüm veya mevcut ağaç kullanılır.

TSA'nın iki spesifik kontrol parametresi vardır: ST ve NS. NS, ağaç başına üretilcek tohum sayısını ifade etmektedir. ST (*search tendency*) ise TSA'nın güncelleme kuralındaki en iyi veya mevcut ağacın seçimini belirleyen kontrol parametresidir. ST $[0, 1]$ aralığında önceden tanımlanmış kontrol parametresidir ve bu parametre $[0, 1]$ aralığında üretilen rastgele bir sayı ile karşılaştırılır. Rastgele sayı ST'den küçükse, tohum üretiminde en iyi ağaç kullanılır; aksi takdirde rastgele seçilen bir ağaç ve mevcut ağaç kullanılarak yeni bir tohum üretilir (Kiran, 2017; Cinar ve Kiran, 2018). TSA'nın akış şeması Şekil 3.16'da verilmektedir.



Şekil 3.16. TSA algoritmanın akış şeması

TSA'nın çekirdek üretme aşaması için iki arama denklemi önerilmektedir. İlk denklem (Denklem 3.19), tohumun bu ağaç için üretileceği ağacın konumunu ve ağaç popülasyonundaki en iyi ağacı dikkate almaktadır. Bu arama denklemi ayrıca önerilen algoritmanın yerel arama veya yoğunlaştırma kapasitesini de geliştirmeyi hedeflemektedir. İkinci güncelleme kuralı Denklem 3.20'de sunulmuş olup bu denkleme göre ağaç için yeni bir tohum üretmek için iki farklı ağaç konumu kullanılır.

$$S_{i,j} = T_{i,j} + \alpha_{i,j} \times (B_j - T_{r,j}) \quad (3.19)$$

$$S_{i,j} = T_{i,j} + \alpha_{i,j} \times (T_{i,j} - T_{r,j}) \quad (3.20)$$

Burada, S_{ij} i . ağaç için üretilecek i . tohumun j . boyutu, T_{ij} i . ağacın j . boyutu, B_j , şimdiye kadar elde edilen en iyi ağaç konumunun j . boyutu, T_{rj} popülasyondan rastgele seçilen r . ağacın j . boyutu, $\alpha \in [-1, 1]$ aralığında rastgele üretilen ölçekleme faktörüdür ve i ve r farklı indekslerdir.

4. İMAR UYGULAMALARI İÇİN ÖNERİLEN YÖNTEMLER

4.1. Ön Parselasyon İşlemi için Önerilen Yöntemler

Bu tez çalışmasında, ilk olarak ön parselasyon işlemi LR problemlerinin çözümünde bir alt yapı olması açısından gerçekleştirilmiştir ve bunun için farklı yaklaşımlar önerilmiştir.

4.1.1. BS ile Alan Hesabı

Bu tez çalışmasında parselin sınırlarını bir başka ifadeyle parselin kenar uzunluklarını belirlemek için Gauss yöntemi ve BS birlikte kullanılmıştır. Bu çalışmada hata miktarı (durma kriteri) 0.001 m^2 olarak alınmıştır. Bu, parsel alanında en fazla 0.001 m^2 'ye kadar sapma olabileceği anlamına gelmektedir. İşlem adımları aşağıdaki gibidir:

1. Birinci parselin kenar uzunluğu belirlenir. İlk aşamada parselin kenar uzunluğu adanın uzun kenarlarının yarısı kadardır. Daha sonraki parseller için parselin kenar uzunluğu adanın kalan uzun kenarlarının yarısı kadardır. Başka bir deyişle parselasyon işlemi tamamlanan parselin sınır noktaları, üretilecek yeni parselin başlangıç noktası olarak kabul edilir.

2. Parselin alanı Gauss yöntemi ile hesaplanır.

3. Hesaplanan bu alan değeri parsel için istenen alan değeri ile karşılaştırılır.

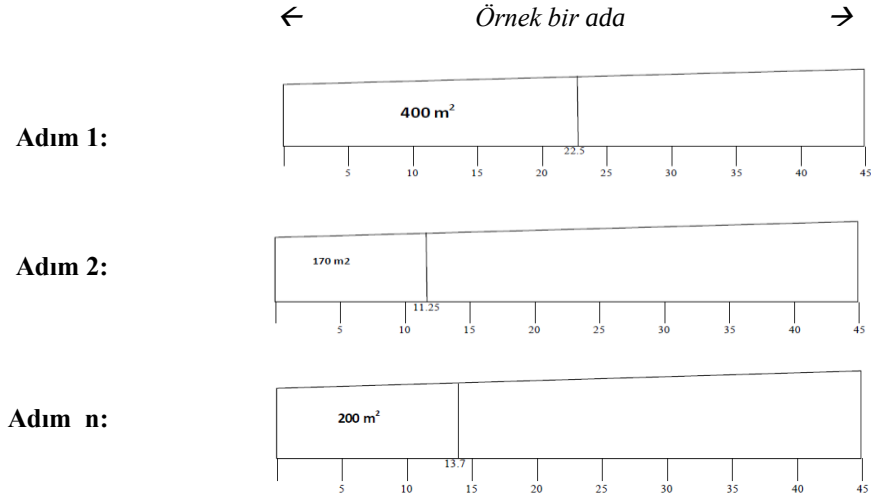
4. Hesaplanan alan değeri istenen alan değerinden küçük ise BS yöntemine göre üretilecek olan parselin kenar uzunluğu artırılır. Aksi takdirde parselin kenar uzunluğu azaltılır.

5. Durdurma kriteri sağlanmazsa, 2. adıma geri dönlür.

6. Durdurma kriteri sağlanırsa, 1. adımdan başlayarak bir sonraki parsel için benzer işlemler gerçekleştirilir.

7. Bu işlem adadaki tüm parsellerin alanları hesaplanıncaya kadar devam eder. Bu şekilde ilgili ada için ön parselasyon işlemi tamamlanmış olur. Adanın iki parçaya bölünmesi durumunda, adanın her bölümü için aynı işlemler gerçekleştirilir.

Bu yöntemin çalışma prensibine ait bir örnek Şekil 4.1'de kısaca verilmiştir.



Şekil 4.1. BS ile alan hesabına ait bir örnek

Parsel için gerekli alanın 200 m^2 ve adanın toplam kenar uzunluğunun 45 m olduğunu varsayalım. Bu durumda, BS başlangıçta ada kenarının orta noktasında (22.5 m) yer almaktadır. İstenilen alan hesaplanan alandan daha küçük olduğundan ($200 \text{ m}^2 < 400 \text{ m}^2$) arama sol tarafta yapılır. Daha sonra istenilen alan hesaplanan alandan ($200 \text{ m}^2 > 170 \text{ m}^2$) büyük olduğu için sağ kısımda arama yapılır. Bu işlem, hesaplanan yeni alan beklenen alana eşit olana kadar yinelemeli olarak devam eder. İşlem tamamlandığında BS tarafından her iki uzun kenarda belirlenen noktalar üretilecek parselin sınır noktalarını temsil etmektedir. Bu adımdan sonra üretilecek yeni bir parsel için belirlenen sınır noktaları başlangıç noktası olarak kabul edilir ve BS ile istenen alan değerine göre aynı işlemler uygulanır.

4.1.2. Kenar uzunluklarına göre ön parselasyon

Kenar uzunluklarına göre ön parselasyon (PwEL - *Parceling with edge lenght*) işleminde imar adası için daha önce verilen plan bilgileri dikkate alınarak aşağıdaki işlemler yapılmaktadır. Ada iki kısma bölünebiliyorsa, aşağıdaki işlemler (Adım 1, 2, 3 ve 4) iki farklı ada varmış gibi aynı şekilde uygulanır:

1. Adanın kenar uzunlukları (en uzun iki kenar) belirlenir.
2. İmar adası, ayırık nizamda mı yoksa blok nizamda mı?

3. (*Ayrık Nizam*) Ayrık nizamdaki adalar için köşe parsellere ait kenar uzunluğu en az 14 m olmalıdır. Ara parsellerin kenar uzunlukları en az 12 metre olmalı ve dört kattan sonra parsellerin kenar uzunlukları her kat için 1 m artırılmalıdır.

(*Blok Nizam*) Blok düzeni için, sonlu otomata modeli aşağıdaki gibi verilmiştir:

$\Sigma = [6, 9, 11]$ m olduğunu varsayalım. Bu nedenle;

$$= 11 ((6 + 6 \ 6 + 6 \ 6 \ 6 + 6 \ 6 \ 6 \ 6) (9 \ 9 (6 + 6 \ 6 + 6 \ 6 \ 6 + 6 \ 6 \ 6 \ 6)) *) 11$$

Yukarıda verilen kenar uzunluğu dikkate alınarak, bu çalışmada her bir adanın içindeki ara parsel sayısı rastgele üretilmektedir. Bu model, *Blok düzeni* için gerçekleştirilebilecek tüm çözümler için matematiksel bir sonuç üretmektedir. Bu model, ilk ve son parsellerin kenar uzunluklarının minimum 11 m, ara parsel sayısının maksimum 4 adet olması gerektiğini ifade etmektedir. Ada içerisine birden fazla bitişik blok inşa edilecek ise, bloklar arasına kenar uzunlukları en az 9 m olan iki parsel konulmalıdır.

4. Söz konusu kenar (uzun kenarlardan biri) tamamlandığında, diğer bir deyişle parselasyon işlemi tamamlandıktan sonra kalan alan tüm parsellere eşit olarak dâhil edilir ve tüm parsellerin alanları güncellenir. BS, parsellerin yeni alan değerlerine göre çalıştırılır. Böylece yeni parselasyon planı üretilir.

5. Tüm ada, parsellere bölündükten sonra, son olarak her bir parseli yönetmeliğe göre göre kuzeybatıdan başlayarak 1'den başlamak suretiyle saat yönünde numaralandırılarak parsel numarası (parsel kimliği) atanır.

4.1.3. Alana göre ön parselasyon

Alana göre ön parselasyon (PwA - *Parceling with area*) işlemi gerçekleştirilecek ada için daha önce verilen plan bilgileri dikkate alınarak aşağıdaki işlemler yapılmaktadır. Adanın ortasından ayırma çizgisi ile ada iki parçaya bölünebiliyorsa, aşağıdaki işlemler (Adım 7 hariç) iki farklı ada varmış gibi aynı şekilde uygulanır.

1. Her bir parselin minimum alanı belirlenir.
2. Adanın toplam alanı hesaplanır.
3. Hesaplanan alana göre kaç parsel yerleştirilebileceği belirlenir.

4. Tüm parsellerin alanları belirlendikten sonra adanın kalan alanı 100 m^2 'den büyük ise bu alan her parsel için eşit olarak tahsis edilir. Kalan alan 100 m^2 'den az ise bu alan (iki) köşe parsel için eşit olarak dağıtılır.

5. Parsellerin kesin alanları belirlendikten sonra en uzun kenarın köşe noktalarından herhangi birinden başlayarak parselasyon işlemi yapılır.

6. BS ile her bir parselin alanına göre kenar uzunluğu hesaplanır. Bu şekilde ilgili parselin sınırları belirlenmiş olur.

7. Tüm ada parsellere bölündükten sonra, son olarak her bir parsel için kuzeybatı yönünden itibaren 1'den başlamak suretiyle saat yönünde numaralandırılarak yönetmelik gereği parsel numarası (parsel kimliği) atanır.

4.1.4. Kenar uzunluğu ve alana göre ön parselasyon

Kenar uzunluğu ve alana göre ön parselasyon (PwELA - Parceling with edge length and area) işlemi yapılacak imar adası için daha önce verilen plan bilgileri dikkate alınarak aşağıdaki işlemler yapılmaktadır. Ada iki parçaya bölünebiliyorsa, aşağıdaki işlemler (Adım 1 ve 2), iki farklı ada varmış gibi aynı şekilde uygulanır.

1. Öncelikle alan ve kenara göre minimum parsel büyüklükleri hesaplanır.

2. İlgili parsel için gerekli minimum alan, minimum kenar uzunluğu kullanılarak karşılanıyor mu?

-Evet ise, bu parsel için minimum kenar uzunluğu yeterli kabul edilerek yeni parsel oluşturularak işlem sonlandırılır.

-Hayır ise, BS yardımıyla parsel için minimum alan değerini sağlayacak kenar uzunluğu hesaplanır ve işlem sonlandırılır.

3. Tüm ada parsellere bölündükten sonra son olarak her parsel için, yönetmelik gereği kuzeybatı yönünden başlamak suretiyle 1'den başlayarak saat yönünde numaralandırılarak parsel numarası (parsel kimliği) atanır.

4.1.4.1. Parsel alanlarını dengeleme

PwELA yönteminin 2. adımında görüldüğü gibi bir koşullu dallanma bulunmaktadır. Bundan dolayı, son parselin alanının ne kadar kalacağı tahmin

edilememektedir. Bu nedenle, son parselin kalan alanı diğer parsellerin alanından çok daha büyük olabilir. Hatta bu alan parsel üretmek için yetersiz olabilmektedir. Böyle bir durumda ise bu alan son parsel eklenmektedir. Bu tez çalışmasında son parselin fazlalık alanının diğer parsellere eşit şekilde eklenerek dengelenmesi için aşağıdaki adımlar gerçekleştirilmektedir.

1. *PwELA* yöntemine göre, her ada için tüm parseller üretilir.
2. Ada içerisindeki son parselin alanı belirlenir.
3. Son parselin fazlalık alanı her bir parselin alanına eşit olarak dağıtılır ve tüm parsellerin alan değerleri güncellenir.
4. Tüm parsellerin kesin alanları belirlendikten sonra, *PwA* yöntemine göre parselasyon işlemi yeniden başlatılır ve dengeleme işlemi tamamlanmış olur.

4.2. Gerçek Bir Proje Sahası Kullanılarak Dağıtım ve Parselasyon Problemlerinin Çözümüne ait Önerilen Yöntemler

Gerçek bir proje sahası üzerinde gerçekleştirilen LR problemlerin çözümü için aşağıdaki adımlar takip edilmektedir. Bunun için öncelikle önerilen algoritmalarından elde edilen çözümleri değerlendirmesi için öncelikle bir amaç fonksiyonunun belirlenmesi gerekmektedir. Bu yüzden ilk olarak gerçek proje sahası için çözümlerin kalitesini bir uzmana ihtiyaç duymadan belirleyecek bir amaç fonksiyonu önerilmektedir.

4.2.1. Dağıtım ve parselasyon problemlerinin tanımı ve amaç fonksiyonu

İmar uygulamalarındaki LR problemleri, kadastro parselinin kadastro adalarındaki konumu dikkate alınarak, tüm arazi sahiplerinin imar parsellerine en uygun yerleşimi olarak tanımlanmaktadır. LR, yapısal olarak gezgin satıcı (TSP – *traveling salesman problem*) ve tipik zamanlama problemleri gibi optimizasyon problemlerine benzediği için bir kombinatoriyal optimizasyon problemidir. Bu problemin çözümü 2 ana amaç doğrultusunda yapılmalıdır:

1. Her bir kadastro parselindeki malik ya da maliklere mümkün olduğunca en yakın imar parselinden tahsis yapılmalıdır.

2. Uygulama alanının teknik durumu uygun ise, malikler müşterek değil, müstakil imar parsellerine yerleştirilmelidir.

Problemin girdi ve çıktıları aşağıdaki gibi kısaca açıklanabilir.

Girdiler:

Kadastro Adaları: Proje alanındaki mevcut adalar

Kadastro Parselleri: Her kadastro adasındaki mevcut parseller

İmar Adaları: Uygulama alanında yollarla çevrili yeni bir kent görünümü veren parseller topluluğu.

Malikler: Her kadastro parselindeki arsa sahipleri/malikler

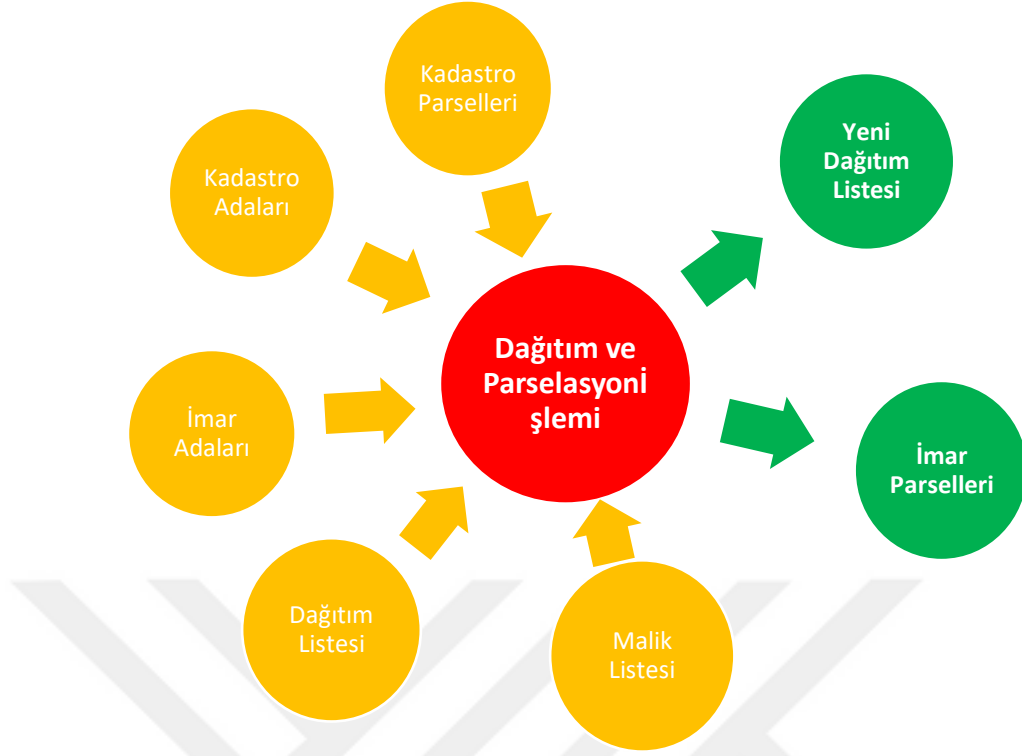
Dağıtım Listesi: Her bir arsa sahibinin kadastro parsel bilgilerini ve alanını gösteren mevcut dağıtım listesi

Çıktı:

İmar Parselleri: İmar adalarındaki parseller

Yeni Dağıtım Listesi: Kadastro parselindeki arsa sahiplerinin yeri, alanı ve hisse durumu dikkate alınarak elde edilen tüm arsa sahiplerinin parsel bilgilerini içeren yeni dağıtım(tahsis) sonucunu içeren liste

Şekil 4.2, LR işlemi için temel girdi ve çıktı değerlerini göstermektedir. LR süreci kadastro parsellerini ve imar adaları temel girdi olarak alırken, çıktı olarak ise yeni dağıtım listesi ve yeni imar parselleri üretmektedir. Ayrıca gerçek bir proje sahası için önerilen amaç fonksiyonuna ait girdi değerleri ve matematiksel model aşağıda verilmektedir.



Şekil 4.2. Dağıtım ve parselasyon işlemine ait girdi ve çıktı değerleri

a. Problem Girdileri

nO : Malik sayısı

nCB : Kadastro ada sayısı

nZB : İmar ada sayısı

$nPCB_i$: i . kadastro adasındaki parsel sayısı ($\forall i \in nCB$)

$nPZB_j$: j . imar adasındaki parsel sayısı ($\forall j \in nZB$)

nTP : Toplam parsel sayısı

PA_{ij} : j . imar adasındaki i . parselin alanı ($\forall i \in nPZB_j, \forall j \in nZB$)

BA_i : j . imar adasının alanı ($\forall j \in nZB$)

DL : Projenin en uzun köşegen uzunluğu

b. Karar Değişkenleri

D_{ijklm} : m . malik için i . Kadastro adasının k . parseli ile j . İmar adasının l . parseli arasındaki Öklid mesafesi ($\forall i \in nCB, \forall j \in nZB, \forall k \in nPCB_i, \forall l \in nPZB_j, \forall m \in M$)

$nJZP_m$: m . malik için, müstakil olarak üretilmesi uygun olmasına rağmen müşterek üretilen imar parsel sayısı ($\forall m \in M$)

$nZPS_m$: m . malik için, toplam imar parsel sayısı ($\forall m \in M$)

GR_i : i . İmar adasının taşma ve boşluk oranı ($0 < GR_i < 1$)

c. Kısıtlar

1. Her bir parselin en az bir kenarının yola cephesinin bulunması zorunludur.
2. Parseller verilen plan doğrultusunda minimum cephe ve kenar uzunlukları kriterlerini sağlamalıdır.

d. Önerilen amaç fonksiyonu için matematiksel model

$$TJ = \sum_{m=1}^M nJZP_m \quad (4.1)$$

$$D = \sum_{m=1}^M \sum_{l=1}^{nJZP_m} D_{m,l} = D_{ijklm} \quad (4.2)$$

$$GR_i = Abs(1 - \frac{\sum_{j=1}^{nJZB_i} P_{A_{ij}}}{BA_i}) \quad (4.3)$$

$$GR = \sum_{i=1}^{nZB} GR_i \quad (4.4)$$

$$Z = \min(k1 * \frac{100*TJ}{nO} + k2 * \frac{D}{DL} + k3 * \frac{100*GR}{nZB}) \quad (4.5)$$

$$0 < k1, k2, k3 < 1 \quad (4.6)$$

Denklem 4.1’de, TJ toplam müşterek hisse sayısını ifade etmektedir. TJ değeri hesaplanırken, müstakil bir parsel oluşturabileceği halde, müşterek dağıtımı yapılan her bir malik için bu değer 1 artırılır. Örneğin, müstakil bir parsel oluşturmak için 400 m²’lik bir alanın yeterli olduğunu varsayalım: Arsa sahibinin alanı 500 m² olmasına rağmen, bu malik müşterek bir parsele dağıtılsa, TJ değeri bu malik için 1 artırılır. Malik alanı 400 m²’nin altında ise, teknik açıdan malikin yalnız başına bir parsele tahsisi gerçekleşmeyeceği için malikin her hangi bir şekildeki tahsisinin TJ değeri üzerinde etkisi olmamaktadır. Tüm malikler için bu işlemler gerçekleştirilerek toplam TJ değeri üretilir. Daha sonra bu değer, Denklem 4.5’teki nO değerine bölünür. Böylece 0 ila 1 arasında bir

değer elde edilir. Son olarak Denklem 4.5'te bu değer 100 ile çarpılarak 0 ile 100 arasında bir değer üretilmektedir.

Denklem 4.2'deki, D projedeki tüm maliklerin kadastro parseli ile imar parselleri arasındaki mesafe değerlerinin toplamını ifade etmektedir. DL değeri, projeye özel bir girdi parametresidir ve bu değer, projedeki en uzak iki nokta arasındaki Öklid uzunluğunu ifade etmektedir. Böylelikle proje sahası ve proje ölçeği değişse bile amaç fonksiyonu projede etkin bir şekilde çalışmaktadır. D/DL yapılarak elde edilen bu değer Denklem 4.5'teki amaç fonksiyonundaki Z değerine yansıtılır. Bu proje sahası için D/DL işlemi ile elde edilen değerler 0 ile 100 arasında olduğu için herhangi bir katsayıya gerek duyulmamaktadır.

GR , Denklem 4.4'te imar adaları üzerindeki boşluk ve taşma oranlarının toplamını temsil etmektedir. Örneğin herhangi bir adanın alanı 3000 m^2 ve söz konusu adadaki tüm parsellerin alanlarının toplamı 3500 m^2 olduğunu varsayalım: Bu değerler ışığında Denklem 4.3'te verilen formüle göre GR_i hesaplanırken $Abs(1 - (3500/3000))$ işlemi yapılır. GR_i değeri normalde Abs fonksiyonu kullanılsaydı -0.2 olarak hesaplanırdı. Ancak Abs fonksiyonu sayesinde bu değer pozitif bir değere dönüştürülür. Bu nedenle, ada içerisindeki taşma ve boşluk durumları, amaç fonksiyonunda değer olarak eşit kabul edilmektedir. Bu durum hem taşma hem de boşluk alanlarını ceza puanını aynı şekilde etkileyen bir değer kabul ederek amaç fonksiyonunun doğru çalışmasını sağlamaktadır. Bu formüle göre sadece 0 ile 1 arası değerler elde edileceği için Denklem 4.5'te bu değer 100 ile çarpılarak 0 - 100 arası bir sayıya da dönüştürülür.

Amaç fonksiyonundaki hisse, mesafe ve boşluk ceza puan değerlerinin hepsi yapılan ara işlemler sonucunda bu problem için 0 ile 100 aralığında yer aldığı görülmüştür. Böylece ceza puanlarının birbirlerini olumsuz anlamda domino etmeden amaç fonksiyonunun etkili sonuçlar üretmesi sağlanmaktadır. Son olarak, bu değerler kullanıcı tarafından önceden tanımlanan ölçeklendirme parametresi ile çarpılır ve maliyet (Z) değeri bulunur. Denklem 4.6'daki $k1$, $k2$ ve $k3$ değerleri ölçekleme değerleridir. Bu çalışmada bu değerler eşit ve $1/3$ olarak alınmıştır.

4.2.2. Başlangıç popülasyonunun oluşturulması

Algoritmaların iterasyonlar boyunca çalışabilmesi için bir popülasyona ihtiyaç duyulmaktadır. Başlangıç popülasyonu oluşturmaya ait sözde kod Şekil 4.3'te verilmektedir. Bu koda göre, öncelikle seçilecek ada belirlenir. Ardından rastgele bir malik seçilir. Malikin alanı minimum alan değeri ile kıyaslanır. Alanı uygun ise parsel oluşturulur. Aksi halde başka bir malik daha seçilir. Bu işlem minimum alan kriteri sağlanıncaya kadar devam eder. Parsel oluştuktan sonra, tekrar bir malik seçilir. Bu işlem tüm maliklerin ataması gerçekleşene kadar devam eder. Bu çalışmada parselasyon işlemi güney doğu yönünden başlanarak gerçekleştirilmiş ve adanın ilk kısmı dolduğunda, diğer kısma geçilmiştir.

```

Prosedür Başlangıç_Populasyonu_Oluştur()
  Parselasyon işlemine (parsel oluşturmaya) güney doğu yönünden başla.
  Bir ada seç.
  i=1;

  do
    Bir imar adası seç (Blocki).
    do
      Rastgele bir malik seç.
      Seçilen maliklerin alanını topla.
      while (toplam malik alanı < minimum alan koşulu)

      if Blocki yatay bir ada ise
        Güney yönünden parsel oluşturmaya başla.
      else Blocki dikey bir ada ise,
        Doğu yönünden parsel oluşturmaya başla.
      end
    end

    BS kullanarak imar parseli oluştur.

    if Şimdi oluşturulacak parselde birden fazla malik varsa
      Müşterek bir parsel oluştur ve bu malikleri bu parselde tahsis et.
    else Sadece tek bir malik varsa
      Müstakil bir parsel oluştur ve bu maliki bu parselde tahsis et.
    end
  end

  if Blocki'nin ilk bölümü tamamlanmışsa
    Blocki'nin ikinci bölümüne geç.
  end

  if Blocki'nin parselasyon işlemi tamamlandı ise
    i=i+1;
  end

while imar parsellerinde tüm malikler tahsis edilmedi ise?
Ada Listesi döndür

```

Şekil 4.3. Başlangıç popülasyonu oluşturmaya ait sözde kodu

4.2.3. Başlangıç popülasyonunun değerlendirilmesi

Popülasyondaki her bireyin maliyet değeri, Denklem 4.5'teki amaç fonksiyonuna göre değerlendirilir. Bu işlem için her bir bireye ait adalar ve bu adalarda bulunan parseller ve her bir parseldeki malikler her bireyin çözüm kalitesini değerlendirmek için amaç fonksiyona gönderilir. Her bir bireyin uygunluk değeri (Z) bulunur.

4.2.4. Önerilen çaprazlama operatörü: Harita tabanlı çaprazlama

Çaprazlama işlemi, iki farklı bireyden belli özellikler alarak yeni bir çocuk oluşturan bir operatördür. Çaprazlama sonunda yeni yavrular ortaya çıkar. Bu çalışmada, literatürdeki çaprazlama operatörlerinden farklı olarak, mekânsal veri bazlı bir çaprazlama operatörü önerilmektedir. Önerilen çaprazlama işlemi, adalardaki parsellere karşılık gelen diğer bireydeki/ebeveyndeki aynı adadaki aynı indeksli parsellerin çaprazlanması ile gerçekleştirilir. Önerilen çaprazlama yönteminin bir örneği Şekil 4.5'te gösterilmektedir. Örnekte, iki farklı birey rastgele tanımlanmıştır. Her bir birey yatay ve dikey geometrik yapılara sahip iki adadan oluşmakta ve proje alanında farklı alanlara sahip 18 malik bulunmaktadır. Bu maliklerin parsellere rastgele atanmasıyla, Şekil 4.5'te gösterilen iki farklı bireye ait dağıtım ve parselasyon haritası üretilmiştir. Bu iki birey kullanılarak önerilen çaprazlama operatörü yardımıyla yeni bir çocuk birey elde edilmektedir. Ayrıca bu çalışmada iki farklı çaprazlama tekniği sunulmaktadır.

4.2.4.1. Klasik parsel tabanlı çaprazlama (CPC)

Klasik çaprazlama yöntemi, rastgele şekilde malikleri belirleyip tahsis işlemini gerçekleştirerek çalışmaktadır. Şekil 4.5'te verilen örneğe göre her iki kromozom için 1. adanın 4. parselleri çaprazlama işlemi için belirlenmiştir. Görüldüğü gibi *birey2*'de belirlenen parselde N ve O isimli malikler bulunmaktadır. N ve O maliklerinden sadece O isimli malik koşul durumunu (*random condition*) sağladığından, bu malik için çaprazlama işlemi uygulanır. Daha sonra *birey1*'deki O isimli malike ait parsel bulunur ve bu malik bu parselden çıkarılır. Çaprazlama işleminden sonra *birey1*'in 4. parselinde G , J ve O isimli malikler bulunmaktadır. Bu aşamada rastgele durumu sağlayan malikler

2. adanın 2. parseline gönderilir. Bu örneğe göre sadece malik *J* bu koşulu sağladığından, bu malik mevcut parselden çıkarılarak 2. parsele gönderilmektedir. Sonuç olarak bu iki parselin alanları ve malik listesi dağıtım işlemindeki değişikliğe (ekleme / silme) göre güncellenmektedir. Şekil 4.5'te gösterilen bu işlem sadece tek bir parsel için verilmektedir. Ayrıca *CPC*'ye ait sözde kod detaylı olarak Şekil 4.4'te sunulmaktadır.

```
Prosedür Klasik_Parsel_Tabanlı_Çaprazlama()

CR_Oran = Çaprazlama oranını belirle;
PS = birey1'in adalarındaki parsel sayısı;
CPS = PS * CR_Oran; // çaprazlama oranına göre çaprazlanacak parsel sayısı
parselList = İki farklı kişiden aynı indeksin aynı adasına sahip parselleri rastgele seç.
parselList1 = birey1'in parsel listesini al.
parselList2 = birey2'nin parsel listesini al.

for i = 1 to CPS
    parsel1 = parselList1[i]
    parsel2 = parselList2[i]
    malikListe1 = parsel1'in maliklerini al.
    malikListe2 = parsel2'nin maliklerini al.

    for each malik2 in malikListe2

        if rand < 0.5

            malikListe1.Ekle(malik2) // Dağıtım listesine malik ekleme
            parsel1.alan = parsel1.alan + malik2.alan // Parselin alanını güncelle
            parsel2_birey1 = birey1 adalarındaki malik2'nin parseline bul
            malikList2_birey1 = parsel2_birey1'in maliklerini bul.
            malikList2_birey1.Sil(malik2) // Maliki dağıtım listesinden kaldır
            malikList2_birey1.alan = malikList2_birey1.alan - malik2.alan // Parselin alanını güncelle

            for each malik1 in malikListe1
                if rand < 0.5

                    malikList2_birey1.Ekle(malik1) // Dağıtım listesine sahip ekleme ve sahibi parselde atama
                    parsel2_birey1.alan = parsel2_birey1.alan + malik1.alan // Parselin alanını güncelle
                    malikListe1.Sil(malik1) // Maliki dağıtım listesinden kaldır
                    parsel1.alan = parsel1.alan - malik1.alan // Parselin alanını güncelle

                end if
            end for
        end if
    end for
end for

Return yeni_çocuk_birey
```

Şekil 4.4. CPC'ye ait sözde kod

4.2.4.2. Zeki parsel tabanlı çaprazlama (IPC)

Zeki parsel tabanlı çaprazlama yöntemi, *CPC*'nin aksine, amaç fonksiyonundaki iki temel hedefe odaklanmaktadır. Bu amaçlardan ilki, malikin kadastrodaki konumuna en yakın imar parseline tahsis edilmesi ve bu mesafenin minimize edilmesidir. Bunun için IPC yönteminde, malikin mevcut imar parselinin konumu ve yeni tahsis yapılacağı parselin konumu dikkate alınmaktadır. Malikin yeni konumu mevcut konumundan daha yakınsa, çaprazlama gerçekleştirilir. Aksi halde işlem yapılmaz. İkinci amaç olan hisse sayısının azaltılması için çaprazlama işlemine maruz kalan parsel ya da parseller için teknik şartları sağlayan uygun bir malik alt kümesi varsa mevcut parsel bölünür ve yeni bir parsel üretilir. Böylece hisse sayısından kaynaklanan ceza puanının azaltılması sağlanır.

IPC yöntemi, *CPC*'de olduğu gibi Şekil 4.5'te açıklanmaktadır. Şekil 4.7'deki örneğe göre, her bir kromozomdaki 1. adanın 4. parselleri çaprazlama işlemi için rastgele belirlenmiştir. Bu şekle göre *birey2*'de belirlenen parselde *N* ve *O* isimli malikler bulunmaktadır. *N* ve *O* isimli maliklerden sadece *O* isimli malik mesafe koşulunu sağladığından, bu malik için çaprazlama işlemi uygulanır. Daha sonra *birey1*'deki *O* isimli malikin parseli bulunur ve bu malik söz konusu parselden çıkarılır. Çaprazlama işleminden sonra *birey1*'in 4. parselinde *G*, *J* ve *O* isimli malikler bulunmaktadır. Bu aşamada mesafe durumunu sağlayan malikler 2. adanın 2. parseline gönderilir. Bu örneğe göre sadece malik *J* bu koşulu sağladığından, bu malik mevcut parselden çıkarılarak 2. parselde gönderilmektedir. Sonuç olarak bu iki parselin alanları ve malik listesi dağıtım işlemindeki değişikliğe (ekleme / silme) göre güncellenmektedir. Bu aşamadan sonra güncellenen parseller için hisse sayısının azaltılması hedeflenmektedir. Parsel teknik olarak bölünebiliyorsa bir başka ifadeyle söz konusu parsel yeni bir parselin oluşması için yeterli alana sahipse mevcut parselin alan ve dağıtım listesi maliklerin bir alt kümesine göre güncellenir. Malik listesinde kalan ikinci alt kümedeki malik ya da malikler de yeni parsel oluşturmak için kullanılır. Yeni üretilen parsel, mevcut parselin komşusu olarak oluşturulur. Bu işlem Şekil 4.6'da sunulmaktadır. Şekil 4.8'de gösterilen bu işlem sadece tek bir parsel için anlatılmaktadır. Bu örneğe göre 2. adanın 2. parselinde *H* ve *J* isimli malikler bulunmaktadır. Parsel oluşturma işlemi sonrasında ise *H* ve *J* malikleri ayrı ayrı parsel oluşturmak için yeterli alana sahip olduğu varsayımından dolayı 2. adadaki 2.

parsel bölünerek yeni bir parsel üretilir. Üretilen yeni parsel yeşil ile gösterilmektedir. Dağıtım işlemi sonrasında ise parselasyon haritası güncellenir.

```

Prosedür Zeki_Parsel_Tabanlı_Çaprazlama()

kadastroListesi= Kadastro listesini al.
CRoran = Çaprazlama oranını belirle;
PS = birey1'in adalarındaki parsel sayısı;
CPS = PS * CRoran; // çaprazlama oranına göre çaprazlanacak parsel sayısı
parselList = İki farklı bireyden aynı indeksin aynı adasına sahip parselleri rastgele seç.
parselList1 = birey1'in parsel Listesini al.
parselList2 = birey2'nin parsel listesini al.

for i = 1 to CPS

    parsel1 = parselList1 [i]
    parsel2 = parselList2 [i]
    malikListe1 = parsel1'in maliklerini al.
    malikListe2 = parsel2'nin maliklerini al.

    for each malik2 in malikListe2

        malik2Konum = malik2'nin kadastro listesinden konumunu al.
        mevcutMesafe = malik2Konum ve parsel2 arasındaki mesafeyi hesapla.
        yeniMesafe = malik2Konum ile parsel1 arasındaki mesafeyi hesapla.

        if yeniMesafe < mevcutMesafe

            malikListe1.Ekle (malik2) // Dağıtım listesine malik ekleme
            parsel1.alan = parsel1.alan + malik2.alan // Parselin alanını güncelle
            parsel2_birey1 = birey1 adalarındaki malik2'nin parselini bul
            malikList2_birey1 = parsel2_birey1'in maliklerini bul.
            malikList2_birey1.Sil(malik2) // Maliki dağıtım listesinden kaldır
            malikList2_birey1.alan = malikList2_birey1.alan - malik2.alan // Parselin alanını güncelle

            for each malik1 in malikListe1

                malik1Konum = malik1'nin kadastro listesinden konumunu al;
                mevcutMesafe = malik1Konum ve parsel1 arasındaki mesafeyi hesapla.
                yeniMesafe = malik1Konum ile parsel2_birey1 arasındaki mesafeyi hesapla.

                if yeniMesafe < mevcutMesafe

                    malikList2_birey1.Ekle(malik1) // Dağıtım listesine malik ekleme ve maliki parselde atama
                    parsel2_birey1.alan = parsel2_birey1.alan + malik1.alan // Parselin alanını güncelle
                    malikListe1.Sil(malik1) // Maliki dağıtım listesinden kaldır
                    parsel1.alan = parsel1.alan- malik1.alan // Parselin alanını güncelle

                end if
            end for
        end if
    end for

    Yeni Parsel Oluştur (parcel1); // parsel1'den yeni parsel oluştur
    Yeni Parsel Oluştur (parsel2_birey1); // parsel2_birey1'den yeni parsel oluştur
end for

Return yeni_çocuk_birey

```

Şekil 4.5. IPC'ye ait sözde kod

Prosedür Yeni Parsel Oluşturma (parsel)

TumKume = İki ayrı parsel üretmek için parsel sahiplerinin tüm kombinasyonlarını al.

for each altKume **in** TumKume

malikListe1 = altKume'nin İlk grup Maliklerini Al

malikListe2 = altKume'nin İkinci grup Maliklerini Al

if hem malikListe1 hem de malikListe2 için iki ayrı parsel üretilebiliyorsa

parsel.alan = malikListe1'in alanlarının toplamı

parsel.MalikListesi = malikListe1

yeniParsel = Yeni Parsel Oluştur

yeniParsel.alan = malikListe2.alan

yeniParsel.MalikListesi = malikListe2

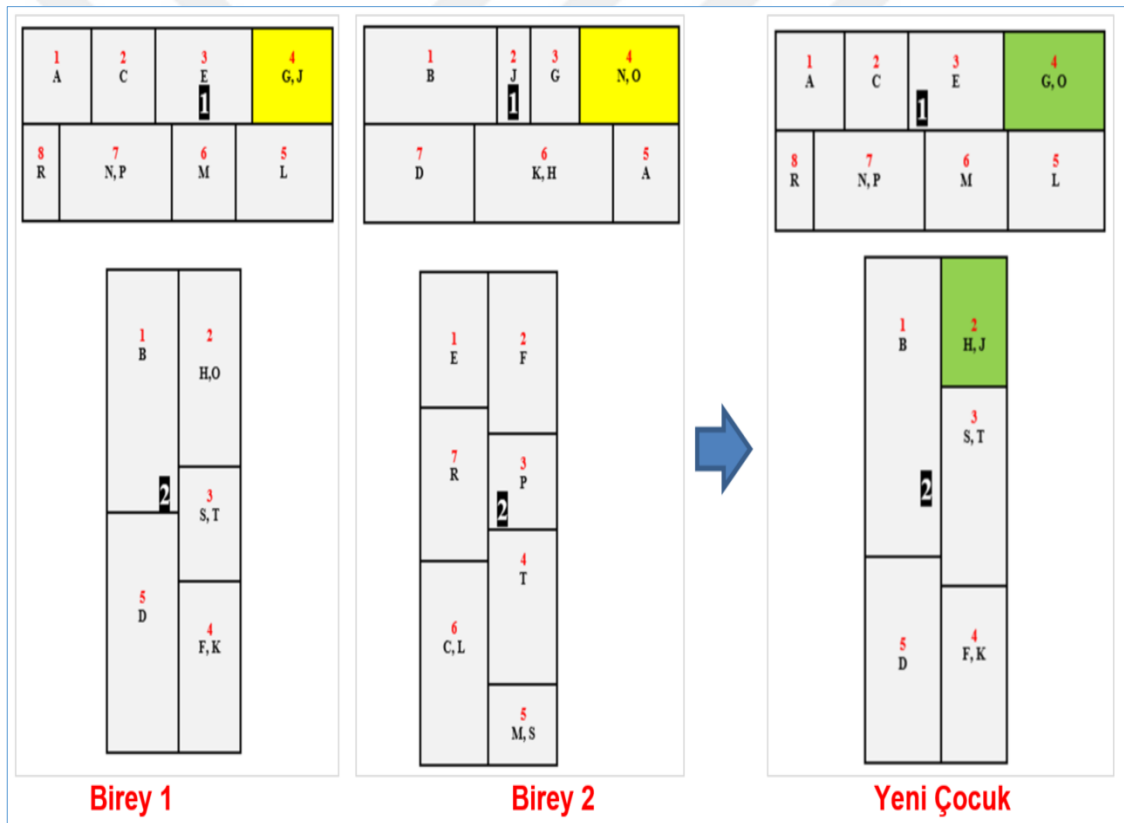
break;

end if

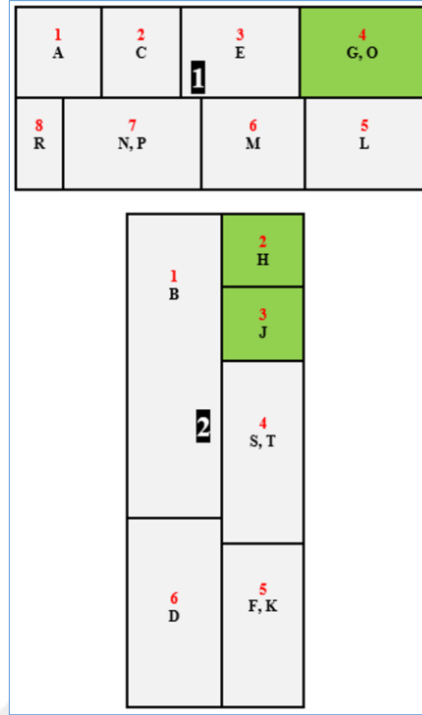
end for

Return yeniParsel

Şekil 4.6. IPC'nin yeni parsel oluşturma evresine ait sözde kod



Şekil 4.7. Parsel tabanlı çaprazlamaya ait bir örnek (Bir parsel için genel gösterim)



Şekil 4.8. IPC'deki parsel oluşturma evresine ait bir örnek

4.2.5. Önerilen mutasyon operatörleri

Çaprazlama işleminden sonra mevcut bireyde bazı değişiklikler oluşturmak için mutasyon işlemi uygulanır. Mutasyon operatörü, erken yakınsamayı önlemek ve bir bireyin belirli genlerini bazı olasılıklar yardımıyla değiştirerek yerel en iyiye takılmasını önlemek için kullanılır (Aggarwal ve ark., 2020). Bu çalışmada mekânsal verilere göre iki farklı mutasyon operatörü önerilmektedir.

4.2.5.1. Aynı ada içindeki parsellerin yerini değiştir

Bu mutasyon operatöründe amaç, alan açısından adaların her iki kısmına ait taşan ve boş alanları en aza indirmektir. Bu işlemde ilk olarak, adaların her iki kısmının taşma ve boşluk alanları hesaplanır. Adanın bir yönünde boş alan, diğer yönünde taşan alan varsa ve bu alanlar minimum parsel alanından fazla ise aşağıdaki adımlar uygulanır.

1. Ada indeks belirlenir. $i=1$.
2. Ada_i seçilir.

3. Ada_i için taşan alanlara (P1) sahip ada yönü belirlenir ve adanın boş alanına (P2) sahip ada yönü belirlenir.

4. P1'deki parseller alan açısından küçükten büyüğe doğru sıralanır.

5. Parsele indeks ataması yapılır. $j=1$;

6. P1.parsel_j P2'ye aktarılır. P1'deki parseller sırasıyla P2'ye aktarılır.

7. $j = j+1$;

8. P1'de boşluk olmadığı sürece ve aynı zamanda P2'de taşma olmadığı sürece

Adım 6'ya gidilir.

9. $i = i + 1$;

10. Şartı sağlayan tüm adalar tamamlandıncaya kadar Adım 2'ye gidilir.

4.2.5.2. Farklı adalardaki parsellerin yerini değiştir

Bu mutasyon operatöründe amaç, alan açısından iki farklı adayı dengelemektir. İşlem adımları aşağıdaki gibidir:

1. İlk olarak, adaların her iki kısmının taşma ve boş alanları hesaplanır.

2. Tüm adalar içinde en fazla taşmaya (B1) ve en fazla boşluğa (B2) sahip ada yönleri bulunur.

3. B1'deki parseller küçükten büyüğe doğru sıralanır.

4. Parsele indeks ataması yapılır. $j=1$;

5. Ardından B1.parsel_j B2'ye taşınır. $j=j+1$;

6. B1 boşluk alan olmadığı ve aynı zamanda B2'de taşma olmadığı sürece Adım 5'e gidilir. Aksi halde işlem sonlandırılır.

4.2.5. Uygun olmayan parsellerin birleştirilmesi

Bu aşamada teknik olarak uygun olmayan parseller birleştirilerek uygun parseller oluşturulur. Algoritmanın çaprazlama aşamasında parsellerin alanı ile ilgili bir kısıtlama bulunmamaktadır. Kısıtlamanın olmaması daha esnek bir çaprazlama işlemine olanak sunmakta ve buna bağlı olarak da genetik çeşitliliğin korunmasını sağlamaktadır. Bununla birlikte bu işlem, uygun olmayan parsellerin oluşmasına neden olabilmektedir. Bu sorunu çözebilmek için bu adım algoritmaya eklenmiştir. Bu şekilde uygun olmayan

parseller komşu parsel ya da parsellerle birleştirilerek yeni ve daha büyük bir parsel elde edilmektedir.

4.2.6. Parselasyon planının güncellenmesi

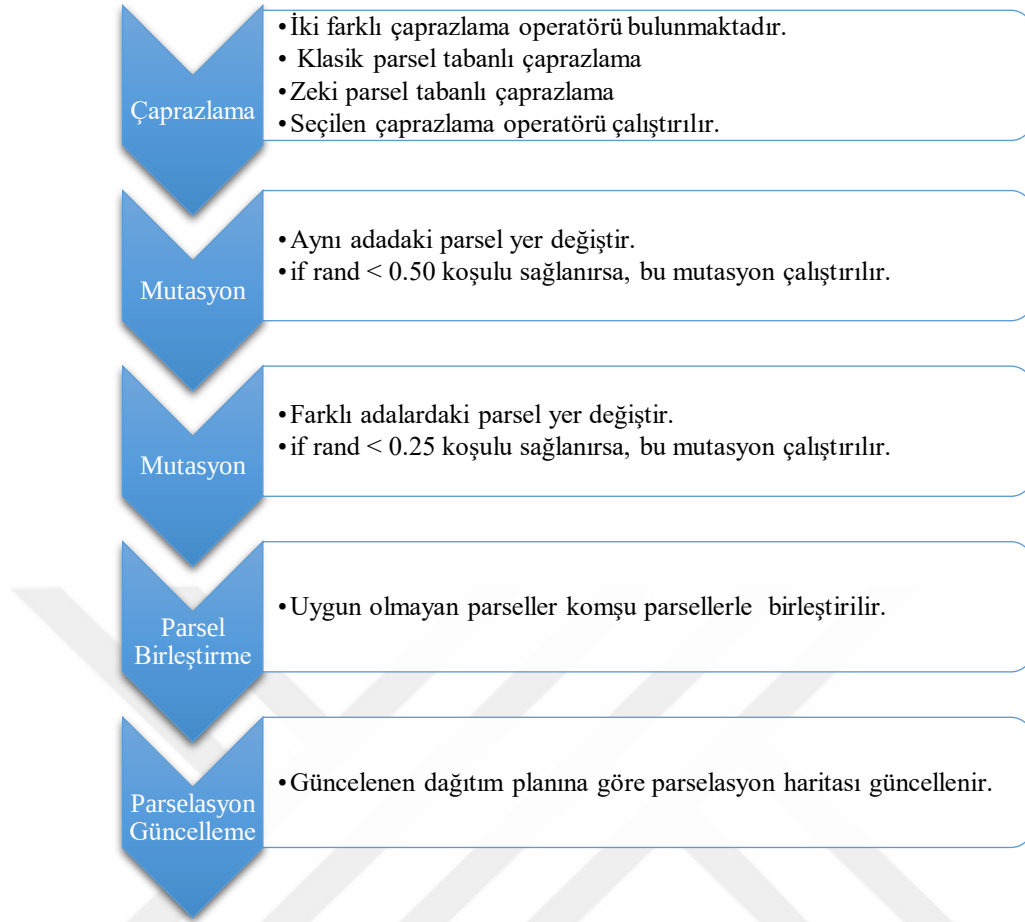
Bu aşamada, güncellenen dağıtım planına göre adalar içerisindeki parsellerin yeni konumlarının belirlenmesi gerekmektedir. Her bir parseldeki maliklerin toplam alanına göre öncelikle parsellerin sınır koordinatları belirlenir ardından da parselin merkez noktası belirlenir. Bu işlem için çok hızlı ve etkili sonuçlar üreten BS yöntemi kullanılmaktadır (Koc ve ark., 2020).

4.2.7. Yeni aday çözümün değerlendirilmesi

Güncellenen dağıtım ve parselasyon planları kullanılarak yeni aday çözüm amaç fonksiyona gönderilir ve yeni maliyet değeri hesaplanır.

4.2.8. Algoritmanın sonlandırılması

Algoritma maksimum yineleme sayısına ulaştığında sonlandırılır ve elde edilen en iyi çözüm kaydedilir. Ayrıca bu çalışmada, yeni birey elde etmek için kullanılan önerilen yöntemin akış şeması Şekil 4.9'da verilmektedir. Görüldüğü gibi iki birey arasında gerçekleştirilen çaprazlama işlemiyle birlikte tüm adımlar sırasıyla uygulanmaktadır. Algoritma başlatılmadan önce çaprazlama operatörlerinden yalnızca biri seçilir ve çaprazlama oranı kullanıcı tarafından belirlenir. ABC, DE, GA, PSO ve TSA algoritmalarında bahsedilen çaprazlama operatöründeki işlem adımları Şekil 4.9'da gösterildiği gibi uygulanmaktadır.



Şekil 4.9. Gerçek dünya problemine ait önerilen yaklaşımın akış şeması

ABC, DE, GA, PSO ve TSA'nın LR problemlerine uygulanması sırasıyla Şekil 4.10-4.14'te verilmiştir. İlk olarak, algoritmalar için N sayıda birey üretilir. Daha sonra bu bireyler önceden tanımlanmış çaprazlama oranlarına göre çaprazlama işlemine tabi tutulur. Ardından mutasyon işlemleri ve son olarak parsel birleştirme işlemlerine tabi tutulurlar. Elde edilen bu yeni dağıtım planına göre ise parselasyon haritası güncellenir.

ABC algoritması sürekli optimizasyon (*continuous optimization*) problemlerinin çözümü için önerilmiştir (Karaboga, 2005). Fakat LR problemleri ayrık bir yapıya sahip olduğundan ABC algoritmasının temel versiyonunun doğrudan probleme uyarlanması söz konusu değildir. Bundan dolayı orijinal ABC algoritmasındaki popülasyon oluşturmak için kullanılan Denklem 3.6 yerine önerilen ABC yaklaşımında Şekil 4.3'te verilen sözde kod kullanılmaktadır. Bununla birlikte sürekli algoritmaların hepsinde popülasyon oluşturma işlemleri aynı olup Denklem 3.6'daki gibidir. Bundan dolayı önerilen bu tez çalışmasında önerilen tüm ayrık algoritmalar için popülasyon oluşturmada Denklem 3.6 yerine Şekil 4.3'te verilen sözde kod kullanılmaktadır. Ayrıca yeni yiyecek

kaynağı oluşturmak için ABC’de kullanılan Denklem 3.7 yerine önerilen ayırık ABC yönteminde Şekil 4.9’da verilen çaprazlama ve mutasyon operatörlerini içeren hibrit yaklaşım önerilmektedir. Sonuç olarak LR problemlerinin yapısına özgü ayırık bir ABC yöntemi önerilmekte ve Şekil 4.10’da verilmektedir.

Prosedür ABC’nin LR’ye Uygulanması()

MaksIter, Çaprazlama oranı, yiyecek kaynağı sayısı, Limit belirle

Rastgele yiyecek kaynakları oluştur

N = besin kaynağı sayısı

Denklem 4.5 kullanarak yiyecek kaynaklarının maliyet değerini hesapla.

iter = 1

while iter < MaksIter

for i = 1 to N

Rastgele seçilen yiyecek kaynağı ile yiyecek kaynağı_i arasında çaprazlama uygula

*Mutasyon operatörleri de Şekil 4.9’daki gibi çaprazlamadan sonra uygulanır.

Denklem 4.5 kullanarak yeni yiyecek kaynağının maliyet değerini hesapla.

if yeni maliyet < yiyecek kaynağı_i maliyeti

Yeni yiyecek kaynağını yiyecek kaynağı_i ile değiştir.

Limit = 0

else

Limit = Limit + 1

end else

end for

for i = 1 to N

Turnuva tarafından seçilen yiyecek kaynağı ile yiyecek kaynağı_i arasında çaprazlama uygula

*Mutasyon operatörleri de Şekil 4.9’daki gibi çaprazlamadan sonra uygulanır.

Denklem 4.5 kullanarak yeni yiyecek kaynağının maliyet değerini hesapla.

if yeni maliyet < yiyecek kaynağı_i maliyeti

Yeni yiyecek kaynağını yiyecek kaynağı_i ile değiştir.

Limit = 0

else

Limit = Limit + 1

end else

end for

for i = 1 to N

if Yiyecek kaynağı_i limit değeri > Limit

Limit = 0

Yeni bir yiyecek kaynağı bul.

end if

end for

iter = iter + 1

end while

En iyi yiyecek kaynağını kaydet.

Şekil 4.10. ABC’nin LR problemlerine uygulanması

DE algoritması ilk olarak sürekli yapıya sahip optimizasyon problemleri için önerilmiştir (Storn ve Price, 1995). DE algoritması bu tür problemlerin çözümü için tasarlandığından dolayı LR problemlerinin çözümünde doğrudan kullanılamamaktadır. Bu yüzden yeni kromozomlar oluşturmak için sürekli DE algoritmasında kullanılan Denklem 3.10 ve 3.15 yerine önerilen ayırık DE yönteminde Şekil 4.9'da verilen çaprazlama ve mutasyon operatörlerinin hibrit yaklaşımı kullanılmaktadır. Diğer taraftan seçim işlemi için Denklem 3.16'da verilen eşitlik önerilen DE yönteminde de aynı şekilde kullanılmaktadır. Sonuç olarak bu tez çalışmasında LR problemlerini çözmek için geliştirilen ayırık DE algoritması Şekil 4.11'de sunulmaktadır.

```

Prosedür DE'nin LR'ye Uygulanması()
MaksIter, Çaprazlama oranı, kromozom sayısı
Rastgele kromozomları oluştur
N = kromozom sayısı
Denklem 4.5 kullanarak kromozomların maliyet değerini hesapla.
iter = 1

while iter < MaksIter

    for i = 1 to N
        [C1, C2, C3] = Mevcut kromozomdan farklı rastgele üç farklı kromozom seç.
        C2 ile C3 arasında çaprazlama uygula ve yeni bir kromozom oluştur (C23)
        C23 ile C1 arasında çaprazlama uygula ve yeni bir kromozom oluştur (C231)
        C231 ve kromozomi arasında çaprazlama uygula ve yeni bir kromozom (YeniC) oluştur.
        *Mutasyon operatörleri de Şekil 4.9'daki gibi çaprazlamadan sonra uygulanır.
        YeniC'nin maliyetini Denklem 4.5'e göre hesapla.

        if YeniC'nin maliyeti < kromozomi
            kromozomi'yi YeniC ile yer değiştir.
        end if

    end for

    iter = iter + 1
end while

En iyi kromozomu kaydet.

```

Şekil 4.11. DE'nin LR problemlerine uygulanması

GA literatürde yer alan gezgin satıcı, çizelgeleme gibi ayırık optimizasyon problemlerinin çözümü için doğrudan kullanılabilmektedir (Whitley, 1994). Benzer şekilde LR problemlerinin çözümünde algoritmanın yapısında herhangi bir değişiklik

yapılmadan GA doğrudan uygulanabilmektedir. Diğer taraftan, LR problemleri enlem ve boylam bilgileri gibi mekânsal veriler (*spatial data*) içermektedir. Bu yüzden literatürde GA yöntemi için önerilen tek noktalı veya çift noktalı çaprazlama operatörleri gibi çaprazlama operatörlerinin probleme doğrudan uygulanması söz konusu olmadığından harita tabanlı çaprazlama operatörleri tüm algoritmalar için önerilmektedir. Benzer şekilde literatürde ekleme (*Insertion Operator*), yer değiştirme (*Swap Operator*), ters çevirme (*Reverse Operator*) gibi mutasyon operatörleri de LR problemlerinde doğrudan kullanılamamaktadır. Bu mutasyon operatörleri yerine önerilen GA yönteminde harita tabanlı mutasyon operatörleri uygulanmaktadır. Böylelikle LR problemlerinin çözümü için önerilen GA yöntemi Şekil 4.12’de verilmektedir.

```

Prosedür GA'nın LR'ye Uygulanması()
  MaksIter, Çaprazlama oranı, kromozom sayısı, elitizm oranı
  P0= Rastgele kromozomları oluştur
  N = kromozom sayısı
  Denklem 4.5 kullanarak kromozomların maliyet değerini hesapla.
  iter = 1

  while iter < MaksIter
    P1=[] : yeni popülasyon boş durumda

    for i = 1 to N
      Turnuvadan iki kromozom seç.
      Seçilen iki kromozomu çaprazla.
      *Mutasyon operatörleri de Şekil 4.9'daki gibi çaprazlamadan sonra uygulanır.
      Yeni kromozomun (YeniC) maliyetini Denklem 4.5'e göre hesapla.
      P1= P1 + YeniC
    end for

    P0'a elitizm uygula
    P0= P1+P0; maliyetlerine göre birleştir.
  end while

  En iyi kromozomu kaydet.

```

Şekil 4.12. GA'nın LR problemlerine uygulanması

PSO algoritması ABC ve DE algoritmalarına benzer şekilde sürekli optimizasyon problemleri için önerilmiştir (Kennedy ve Eberhart, 1995). Bundan dolayı orijinal PSO algoritmasının LR problemlerinin çözümünde doğrudan kullanılması söz konusu değildir. Bu yüzden PSO'da yeni parçacıklar oluşturmak için kullanılan Denklem 3.17 ve 3.18

yerine ayrıık olarak geliştirilen PSO için Şekil 4.9'da verilen çaprazlama ve mutasyon operatörlerini içeren hibrit bir yaklaşımı kullanılmaktadır. Sonuç olarak bu tez çalışmasında LR problemlerini çözmek için geliştirilen ayrıık PSO yöntemi Şekil 4.13'te verilmektedir.

Prosedür PSO'nun LR'ye Uygulanması()
 MaksIter, Çaprazlama oranı, parçacık sayısı
 Rastgele parçacıkları oluştur
 $N = \text{parçacık sayısı}$
 Denklem 4.5 kullanarak parçacıkların maliyet değerini hesapla.
 iter = 1

while iter < MaksIter

for i = 1 to N

 pbest_i ile parçacık_i arasında çaprazlama uygula ve yeni bir parçacık (p1) oluştur.
 Gbest ile parçacık_i arasında çaprazlama uygula ve yeni bir parçacık (p2) oluştur.
 p1 ve p2 arasında çaprazlama uygula ve yeni bir parçacık (p3) oluştur.
 *Mutasyon operatörleri de Şekil 4.9'daki gibi çaprazlamadan sonra uygulanır.
 p3'ün maliyetini Denklem 4.5'e göre hesapla.

if p3'ün maliyeti < pbest_i
 pbest_i güncelle.
end

if p3'ün maliyeti < Gbest
 Gbest güncelle.
end

end for

 iter = iter + 1

end while

En iyi parçacığı kaydet.

Şekil 4.13. PSO'nun LR problemlerine uygulanması

Son olarak, TSA algoritmasının ilk versiyonu da ABC, DE ve PSO algoritmalarına benzer şekilde sürekli optimizasyon problemlerinin çözümünde kullanılmıştır (Kiran, 2015). Bu yüzden TSA yöntemi doğrudan ayrıık LR problemlerine uygulanamamaktadır. Bundan dolayı TSA'da yeni tohumlar oluşturmak için kullanılan Denklem 3.19 ve 3.20 yerine ayrıık olarak geliştirilen TSA için Şekil 4.9'da verilen çaprazlama ve mutasyon operatörlerini içeren hibrit bir yaklaşım önerilmektedir. Bununla birlikte TSA'nın tohum oluşturma evresi, tohum güncelleme stratejisi ve arama eğilimi

temel TSA yönteminde olduğu gibi birebir ayrık TSA yönteminde de kullanılmaktadır. Sonuç olarak bu tez çalışmasında LR problemlerini çözmek için ayrık bir TSA yöntemi Şekil 4.14'te verilmektedir.

Prosedür TSA'nın LR'ye Uygulanması()

MaksFes, Çaprazlama oranı, ağaç sayısı

Minimum ve maksimum çekirdek sayısı, arama eğilimi (ST)

Rastgele ağaçları oluştur

N = ağaç sayısı

Denklem 4.5 kullanarak ağaçların maliyet değerini hesapla.

FES = 1 (FES: Number of Function Evaluation)

while FES < MaksFes

for $i = 1$ to N

M = rastgele oluşturulacak çekirdek sayısı

for $s=1 : M$

En iyi ağaç i ile ağaç i arasında çaprazlama uygula ve yeni bir çekirdek ($s1$) oluştur.

$s1$ ile rastgele ağaç arasında çaprazlama uygula ve yeni bir çekirdek oluştur ($s2$).

*Mutasyon operatörleri de Şekil 4.9'daki gibi çaprazlamadan sonra uygulanır.

$s2$ 'nin maliyetini Denklem 4.5'e göre hesapla.

FES = FES + 1;

end for

Yeni oluşan çekirdeklerden en iyi çekirdeği bul.

if en iyi çekirdeğin maliyeti < ağaç i maliyeti

ağaç i güncelle.

end

if en iyi çekirdeğin maliyeti < en iyi ağacın maliyeti

En iyi ağacı güncelle.

end

end for

end while

En iyi ağacı kaydet.

Şekil 4.14. TSA'nın LR problemlerine uygulanması

4.2.9. Uzman sistem

Bu çalışmada, LR problemlerinin çözümü için optimizasyon algoritmaları önerilmektedir. Matematiksel modelde verildiği gibi, LR problemlerinde ele alınması gereken üç hedef vardır: 1. Her bir parsel için hisse sayısının en aza indirilmesi, 2. Her bir malik için kadastro ve imar parseli arasındaki mesafenin en aza indirilmesi ve 3. Her bir ada için ada içerisindeki taşma ve boşluk alanlarının en aza indirilmesi. Bu hedeflerden ilk ikisi, en aza indirilmesi gereken ana hedeflerdir. Üçüncü amaç, algoritmanın esnek bir şekilde çalışabilmesi ve her iterasyondan sonra hisse sayısının artışının engellenmesi için amaç fonksiyonuna eklenmiştir. Gerçek dünyada optimizasyon algoritmaları ile elde edilen dağıtım ve parselasyon haritalarının veya çıktıların doğrudan kullanılabilmesi için adalardaki taşma ve boşluk alanlarının tamamen yok edilmesi gerekmektedir. Bu çalışmada proje alanındaki tüm adaların doluluk oranını %100 olarak gerçekleştirmek için uzman bir sistem önerilmiştir. Şekil 4.15, ada alanlarını dengelemek için Uzman Sistemin sözde kodunu göstermektedir. Şekil 4.15'e göre en fazla taşma ve boşluk alanlarına sahip adalar belirlenir ve bu iki ada uzman sistemde 4 farklı duruma göre değerlendirilir. Her bir durumda, boşluk ve taşma alanları azaltılır veya mümkünse tamamen yok edilir. Bu çalışmada durma kriteri olarak tüm adalardaki taşma ve boşluk alanlarının toplamı maksimum 10 m^2 olarak belirlenmiştir. Proje alanının yaklaşık 5 hektar olduğu düşünüldüğünde, bu alanın kabul edilebilir olduğu oldukça açıktır.

Prosedür Uzman Sistem()**do***Her bir adanın doluluk oranını hesapla; // Her yön ayrı değerlendirilir.**Adaları doluluk oranlarına göre sırala.**B1 = En yüksek doluluk oranı olan adayı bul.**B2 = En yüksek boşluk oranına sahip adayı bul.**a1 = B1'nin taşma alanı**a2 = B2'nin boşluk alanı**P = B1'nin son parseli**Pa = P'nin alanı***if** $Pa < a1$ **then** // Durum A ve B : If $Pa (500 m^2) < a1 (1200 m^2)$ **if** $Pa < a2$ **then** // Durum A: If $Pa (500 m^2) < a2 (1500 m^2)$ o zaman $a2 1000 m^2$ ve $a1 700 m^2$ olacak.*P'yi B1'den tamamen kaldır.**P'yi tamamen B2'ye ekle.***else** // Durum B: If $Pa (500 m^2) > a2 (300 m^2)$ o zaman $a2 0$ olacak ve $a1 200 m^2$ olarak güncellenecek.*Dağıtım_ve_Parselasyonu_Güncelle (a2).**B2'nin doluluk oranı %100 olarak güncellenir.***end****else** // Durum C ve D : If $Pa (500 m^2) > a1 (400 m^2)$ **if** $a1 < a2$ **then** // Durum C: If $a1 (400 m^2) < a2 (1000 m^2)$ o zaman $a1 0$ olacak ve $a2 600 m^2$ olarak güncellenecek.*Dağıtım_ve_Parselasyonu_Güncelle (a1).**B1'nin doluluk oranı %100 olarak güncellenir.***else** // DURUM D: If $a1 (400 m^2) > a2 (300 m^2)$ then $a2 0$ olacak ve $a1 100 m^2$ olarak güncellenecek.*Dağıtım_ve_Parselasyonu_Güncelle (a2);**B2'nin doluluk oranı %100 olarak güncellenir.***end***a1 ve a2 güncelle.***while** tüm adalardaki taşma veya boşluk alan toplamı $> 10 m^2$ *Dağıtım ve parselasyon haritasını kaydet.***Şekil 4.15.** Uzman sistem (a)

```

Prosedür Dagıtım_ve_Parselasyonu_Güncelle (parcel_alanı)

a=parcel_alanı;
O1=Büyükten küçüğe sıralayarak P'nin maliklerini al.
O2=[];

while O1[0] < a
    O1[0] O2 listesine ekle.
    a= a - O1[0].alan;
    O1'den O1[0] çıkart.
end

O1[0].alan= a; // Kalan alan kadar malikten alan al.
O1[0]'ı O2'ye ekle.
if a > min_Alan then // min_Alan parsel oluşması için gerekli minimum alandır.
    B2'de yeni bir parsel üret.
    YP = yeni parsel;
else
    YP = B2'deki son parsel;
end

O2'yi YP'ye ekle.
YP'nin alanını güncelle.
P'nin alanını ve dağıtım listesini güncelle.

```

Şekil 4.15. Parselasyon ve dağıtım güncelleme aşaması (b)

4.3. Sentetik Veri Seti Kullanılarak Dağıtım ve Parselasyon Problemlerinin Çözümüne ait Önerilen Yöntemler

Bu bölümde tez çalışmasında önerilmiş olan sentetik veri setinde yer alan problemlerin çözümünde kullanılmak üzere önerilen yöntemler anlatılmaktadır. Bunun için ilk olarak amaç fonksiyonunun oluşturulması gerekmektedir.

4.3.1. Amaç fonksiyonunun belirlenmesi ve örnek bir problem

Algoritmaların çözümleri objektif olarak değerlendirebilmesi için bir amaç fonksiyonunun belirlenmesi gerekmektedir. İlk olarak gerçek dünya verisinde olduğu gibi problem girdileri, karar değişkenleri sunulmaktadır.

a. Problem girdileri

NL: Malik sayısı

NCB: Kadastro ada sayısı

NUB: İmar ada sayısı

NPCB_i: *i*. kadastro adasındaki parsel sayısı ($\forall i \in NCB$)

NPUB_j: *j*. imar adasındaki parsel sayısı ($\forall j \in NUB$)

NTP: Toplam parsel sayısı

PA_{ij}: *j*. imar adasının *i*. parselinin alanı ($\forall i \in NPCB_j, \forall j \in NUB$)

BA_i: *j*. imar adasının alanı ($\forall j \in NUB$)

DL: projedeki en uzun köşe uzunluğu

b. Karar değişkenleri

D_{ijklm} : *m*. malik için *i*. kadastro adasının *k*. parseli ile *j*. imar adasının *l*. parseli arasındaki Öklid mesafesi ($\forall i \in n NCB, \forall j \in NUB, \forall k \in NPCB_i, \forall l \in NPUB_j, \forall m \in NL$)

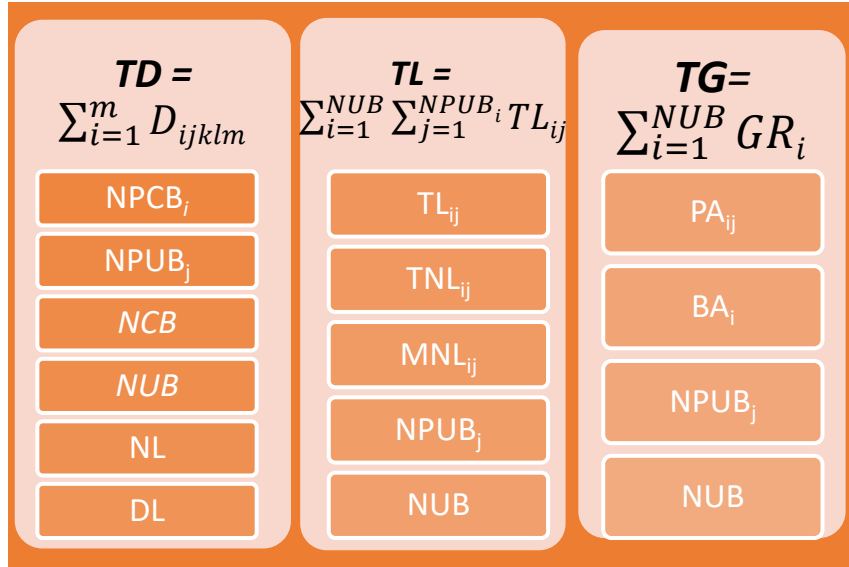
MNL_{ij}: *j*. imar adasının *i*. parseli için bir parselin oluşması için gerekli minimum malik sayısı

TNL_{ij}: *j*. imar adasının *i*. parselindeki toplam malik sayısı

TL_{ij}: TNL_{ij} ve MNL_{ij} arasındaki fark

GR_i: *i*. imar adasının taşıma veya boşluk oranı ($0 < GR_i < 1$)

LR problemlerinin girdi ve karar değişkenlerine göre hesaplanan çıktı değerleri bir başka ifadeyle ceza puanları Şekil 4.16'da verilmektedir. Bu şekilde göre ceza puanları, her birinde ayrı ayrı verilen alanlardaki girdi ve karar değişkenlerine göre hesaplanır. Örneğin, TL değeri hesaplanırken MNL_{ij}, TNL_{ij}, TL_{ij}, NPUB_i ve NUB giriş verileri ve karar değişkenleri kullanılır. Mesafe (TD) ve boşluk (TG) ceza puanları da benzer şekilde ilgili girdi ve karar değişkenleri kullanılarak hesaplanır.



Şekil 4.16. LR problemleri için girdi ve çıktı değerlerine ait görsel bir temsil

c. Kısıtlar

1. Her parselin yola en az bir cephesi olmalıdır.
2. Parseller verilen plana göre minimum cephe ve kenar uzunluğu kriterlerini karşılamalıdır. Bu çalışmada minimum kenar uzunluğu (bir parselin genişliği) olarak 15m alınmıştır.

$$TD = \sum_{m=1}^M D_{ijklm} \quad (4.7)$$

$$TL = \sum_{i=1}^{NUB} \sum_{j=1}^{NPUB_i} TL_{ij} = TNL_{ij} - MNL_{ij} \quad (4.8)$$

$$GR_i = Abs(1 - \frac{\sum_{j=1}^{NPUB_i} PA_{ij}}{BA_i}) \quad (4.9)$$

$$TG = \sum_{i=1}^{NUB} GR_i \quad (4.10)$$

$$Z = \min(\frac{100*TL}{NL} + \frac{TD}{DL} + \frac{100*TG}{NUB}) \quad (4.11)$$

Bu problemin amaç fonksiyonu, üç farklı ceza puanından oluşmaktadır. Denklem 4.7’de verilen ilk ceza puanı her arsa sahibi için imar ve kadastro parselleri arasındaki Öklid mesafelerinin toplamıdır. İkinci ceza puanı, Denklem 4.8’de sunulan malik sayısı

ile ilgili ceza puanıdır. Bir parsel oluşması için gerekli minimum malik sayısından fazla malik varsa bu fark kadar ceza puanı her bir parsel için uygulanır. Üçüncüsü; Denklem 4.9 ve 4.10'da verilen ve algoritmanın esnek bir şekilde çalışmasını sağlayan bir ceza puanıdır. İlgili adadaki taşma ve boş alanlar varsa bu alanları en aza indirmeyi, mümkünse bu uygunsuz alanların tamamen yok edilmesini hedefleyen bir ceza puanıdır. Tüm bu üç ceza puanı Denklem 4.11'de ölçeklenir. Böylece ceza puanlarının birbirini bastırması engellenir. Daha sonra elde edilen değerler toplanır ve uygunluk değeri olan Z değeri elde edilir.

Şekil 4.17, LR problemlerinde *parsel sayısı*= 20 ve *parsel başı malik sayısı* = 4 için örnek bir kadastro haritası ve dağıtım listesini göstermektedir. Bu örnekte kadastro haritasında aynı büyüklükte 2 ada bulunmaktadır. Problemde her adada 10 parsel olmak üzere toplam 20 parsel bulunmaktadır. Ayrıca her parselde 4 farklı hisse diğer bir deyişle malik bulunmaktadır. Dolayısıyla bu problem 80 farklı malikten oluşmaktadır. Bu problemde temel amaç, her malik için kadastro parseline en yakın imar parseline minimum hisseli parsel oluşturularak tahsis yapılmasıdır. Mümkünse aynı konumdan tahsis yapılarak ideal bir çözüm elde etmektir. Bu nedenle, imar planı ve yeni dağıtım planından oluşan ideal çözüm, girdi olarak verilen kadastro haritası ve dağıtım listesinin bire bir aynısı olmalıdır. Bu durumda maliyet değeri (Z) sıfırdır.

1 A4-2 A3-2 A2-2 A1-2 300.0	2 B1-2 B4-2 B2-2 B3-2 300.0	3 C2-2 C3-2 C4-2 C1-2 300.0	4 D1-2 D4-2 D3-2 D2-2 300.0	5 E3-2 E2-2 E1-2 E4-2 300.0
10 J1-2 J4-2 J3-2 J2-2 300.0	9 I2-2 I3-2 I4-2 I1-2 300.0	8 H3-2 H1-2 H4-2 H2-2 300.0	7 G4-2 G3-2 G2-2 G1-2 300.0	6 F4-2 F2-2 F1-2 F3-2 300.0
1 A4-1 A2-1 A1-1 A3-1 300.0	2 B1-1 B4-1 B3-1 B2-1 300.0	3 C1-1 C2-1 C3-1 C4-1 300.0	4 D3-1 D4-1 D1-1 D2-1 300.0	5 E4-1 E1-1 E3-1 E2-1 300.0
10 J2-1 J1-1 J3-1 J4-1 300.0	9 I4-1 I1-1 I2-1 I3-1 300.0	8 H1-1 H2-1 H4-1 H3-1 300.0	7 G3-1 G2-1 G1-1 G4-1 300.0	6 F2-1 F1-1 F4-1 F3-1 300.0

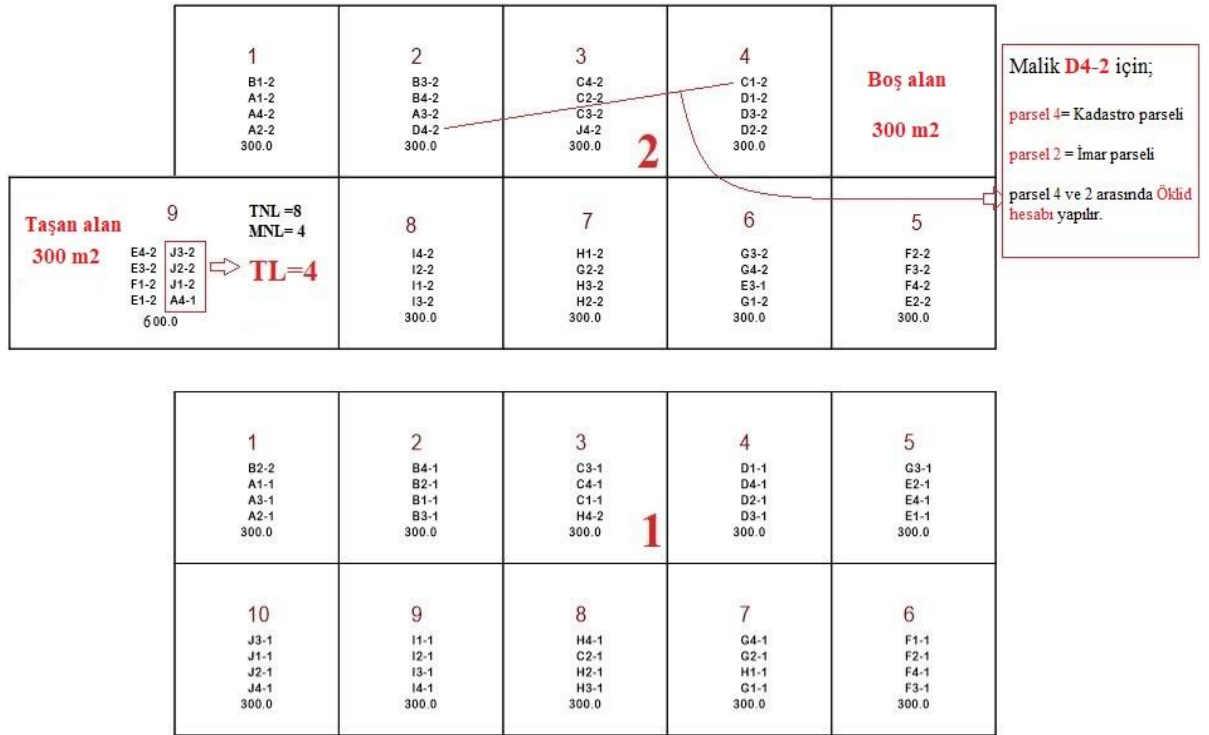
Şekil 4.17. Kadastro haritasını ve dağıtım listesini gösteren bir örnek (parsel sayısı= 20, malik sayısı =4)

Şekil 4.18 yeniden dağıtım listesinin bir örneğini ve rastgele oluşturulan yeni imar haritasını göstermektedir. Bu proje sahasında maliklerin parsellere tahsis durumlarına göre imar parselleri oluşturularak imar haritası elde edilmiştir. 1. adada (proje sahasındaki alttaki ada) taşma ve boşluk alanı olmadığı görülürken, 2. adada (üstteki ada) hem boşluk hem de taşma alanları bulunmaktadır. Bu durum belli bir ceza puanı gerektirmektedir. Şekil 4.18 incelendiğinde, imar planında 2. adadaki 5. parselin oluşması gereken alanda parsel oluşturulacak alan varken bu bölgede $300m^2$ 'lik bir boşluk alan oluşmuştur. Ayrıca 2. ada 9. parselin alanı $600m^2$ olduğundan $300m^2$ taşan alan görünmektedir. Bu durumda, toplam $600m^2$ 'lik bir alan, boşluk ceza puanlarını etkilemektedir. Denklem 4.9'daki toplam boşluk ve taşma alanının toplam ada alanına oranı GR_i değerini elde etmektedir. Bu durumda GR_1 değeri 0.2 ($600/3000$) olarak hesaplanır. Bu işlem Denklem 4.10'a göre tüm adalar için gerçekleştirilir. 1. ada için taşma veya boşluk olmadığı için GR_2 0'dır. Son olarak Denklem 4.11'e göre GR değeri 10.0 ($100 \times 0.2/2$) olarak hesaplanır ve elde edilen bu değer Z değerine eklenir.

MNL_{ij} karar değişkeni, her bir parselin alanına göre gerekli minimum malik sayısı ile ilgilidir. Örneğin bu çalışmada parsel alanı $300m^2$ ve her bir hissenin alanı $75m^2$ olduğundan MNL_{ij} 4 ($300/75$) olarak hesaplanmaktadır. Bu veri setindeki problemlerdeki tüm parseller boyut olarak birbirlerine eşit olduğundan MNL_{ij} değerleri birbirine eşittir. TNL_{ij} ise her bir parseldeki toplam hisse sayısıdır. Bu örnekte 2. adadaki 9. parsel 8 malikten oluşmaktadır. Bu durumda TNL_{ij} ile MNL_{ij} arasındaki farkı temsil eden TL_{ij} değeri bu parsel için 4 ($8 - 4 = 4$) olarak elde edilir. Tüm parseller için elde edilen değerler Denklem 4.8'e göre toplanır. Fakat diğer parseller 4 adet arsa sahibinden oluştuğu için bu parseller için TL_{ij} değerleri 0 ($4-4= 0$) olarak hesaplanmaktadır. Denklem 4.11 kullanılarak $TL \times 100/NL$ formülüne göre hisse ceza puanı 5.0 ($100 \times 4/80 = 5$) olarak hesaplanır ve bu değer Z değerine eklenir.

TD , her malik için kadastro parseli ile yeni imar parseli arasındaki mesafe değerini en aza indirmek için amaç fonksiyonuna dâhil edilmiştir. Her malik için kadastro parseli ile imar parseli arasındaki Öklid mesafesi ölçülmektedir. Bu çalışmada parsellerin orta noktası Öklid için referans alınmıştır. Şekil 4.17'deki dağıtım listesi incelendiğinde, malik D4-2 2. adanın 4. parselinde yer aldığı görülmektedir. Ancak Şekil 4.18'e bakıldığında D4-2 için yeniden dağıtımda 2. adadaki 4. parsel yerine 2. ada 2. parselden tahsisi yapılmıştır. Bu durumda D_{ijklm} değeri 2. ve 4. parsellerin orta noktaları arasındaki Öklid

mesafesi ölçülerek hesaplanır. Her bir parselin genişliği 15m olduğu ve bu parseller arasında 2 parsel bulunduğundan bu malik için ceza puanı 30 olarak hesaplanmaktadır. Başka bir örnek vermek gerekirse; malik A4-2, 2. ada 1. parselden tahsis edilmiştir. Kadastro parselinin bulunduğu imar parselinden tahsis yapıldığından bu malik için ceza değeri uygulanmaz. Ancak Öklid mesafesine göre A3-2'ye ceza puanı uygulanır. Bu şekilde tüm malikler için kadastro parseli ile yeni imar parseli arasındaki mesafeler hesaplanır ve Denklem 4.7'deki gibi TD değeri elde edilir. Toplam TD değerinin Denklem 4.11'deki gibi DL değerine bölünmesi ile elde edilen değer Z değerine eklenir.



Şekil 4.18. LR problemlerinin çözümüne ait bir örnek

4.3.2. Başlangıç popülasyonunun oluşturulması ve değerlendirilmesi

Yapay zeka optimizasyon algoritmaları, daha iyi veya en uygun çözümü bulmak için başlangıç popülasyonuna ihtiyaç duymaktadır. Önerilen yöntemde başlangıç popülasyonuna ait bir birey Şekil 4.19'da gösterilmektedir. Bu şekilde, her biri 10 parsel içeren iki imar adası bulunmaktadır. Her bir parsel, 300m² alana sahip olup her biri 100 m² alana sahip 3 arsa sahibinden oluşmaktadır. Bu örnekte, her bir parsel için rastgele üç arsa sahibi seçilmiştir. Bu işlem, tüm malikler herhangi bir parseli tahsis edilinceye kadar

devam eder. Elde edilen popülasyona ait aday çözümlerin uygunluk değerleri amaç fonksiyonuna göre değerlendirilir. Bu işlem Denklem 4.11'e göre yapılır.

1 A3-2 G3-2 I3-1 300.0	2 J3-1 C3-2 I3-2 300.0	3 I2-2 B2-2 E1-2 300.0	4 H3-2 F3-2 B1-2 300.0	5 E2-2 F2-2 D1-1 300.0
10 A2-2 H3-1 C2-1 300.0	9 A1-2 A2-1 J3-2 300.0	8 D3-2 D1-2 D2-2 300.0	7 G1-1 F1-2 J1-1 300.0	6 J1-2 G1-2 G2-2 300.0
1 I1-1 B3-2 J2-2 300.0	2 G3-1 H2-2 I1-2 300.0	3 E3-2 E3-1 B1-1 300.0	4 D2-1 E2-1 C1-2 300.0	5 I2-1 G2-1 C2-2 300.0
10 B3-1 H1-2 A3-1 300.0	9 C1-1 J2-1 H1-1 300.0	8 C3-1 A1-1 B2-1 300.0	7 H2-1 E1-1 D3-1 300.0	6 F2-1 F1-1 F3-1 300.0

Şekil 4.19. Rastgele oluşturulmuş örnek bir birey

4.3.3. Önerilen çaprazlama operatörü

Bu tez çalışmasında sentetik veriler için harita tabanlı bir çaprazlama operatörü önerilmektedir. Önerilen yaklaşımın çalışma prensibi gerçek dünya problemi için önerilen Şekil 4.4'te sunulan *CPC* ile aynıdır. Ayrıca, önerilen çaprazlama operatörüne ait bir örnek Şekil 4.20'de sentetik bir veri üzerinde gösterilmektedir. Bu örneğe göre, *parcel 4*, çaprazlama işlemi için seçilmiştir. Her parselde üç malik bulunmaktadır. Önerilen yaklaşımda rastgele duruma (*random condition*) göre, ilk önce malikler çaprazlama için belirlenir. Ardından, bu örnek için *F2-1* ve *D2-1* isimli malikler

çaprazlamaya tabi tutulur. *Birey 2*'deki malik *F2-1*, *Birey 1*'in *parcel 4*'üne tahsis edilmeden önce, malik *F2-1*, *Birey 1* içinde hangi parselde olduğu bulunur ve o parselden malik *F2-1* çıkartılır. Bunun yerine, rastgele operatöre bağlı olarak bir veya daha fazla malik seçildiyse, seçilen maliklerin tümü bu parselde tahsis edilecekti. Bu örnekte, malik *H3-2* bu parselde tahsis edilmiştir. Benzer şekilde, malik *D1-2* için çaprazlama işlemi yapılır ve Şekil 4.20'de verildiği gibi yeni bir birey elde edilir.

Birey 1 =

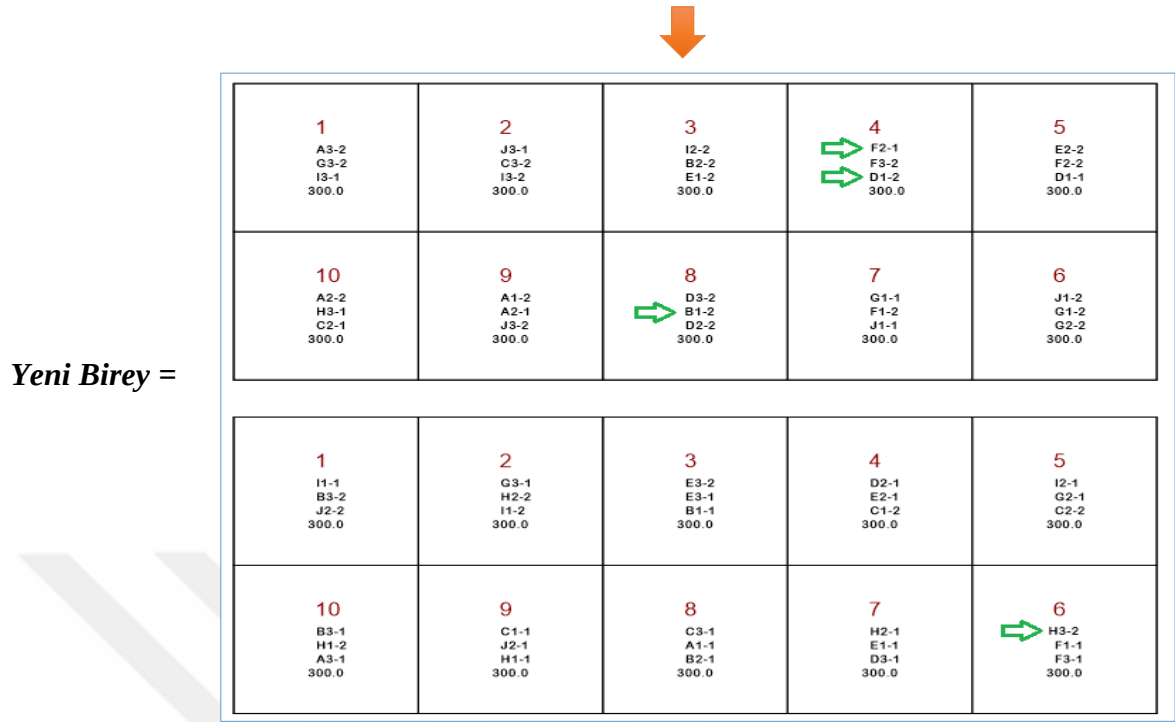
1 A3-2 G3-2 I3-1 300.0	2 J3-1 C3-2 I3-2 300.0	3 I2-2 B2-2 E1-2 300.0	4 H3-2 F3-2 B1-2 300.0	5 E2-2 F2-2 D1-1 300.0
10 A2-2 H3-1 C2-1 300.0	9 A1-2 A2-1 J3-2 300.0	8 D3-2 D1-2 D2-2 300.0	7 G1-1 F1-2 J1-1 300.0	6 J1-2 G1-2 G2-2 300.0

1 I1-1 B3-2 J2-2 300.0	2 G3-1 H2-2 I1-2 300.0	3 E3-2 E3-1 B1-1 300.0	4 D2-1 E2-1 C1-2 300.0	5 I2-1 G2-1 C2-2 300.0
10 B3-1 H1-2 A3-1 300.0	9 C1-1 J2-1 H1-1 300.0	8 C3-1 A1-1 B2-1 300.0	7 H2-1 E1-1 D3-1 300.0	6 F2-1 F1-1 F3-1 300.0

Birey 2 =

1 A3-2 G3-2 A2-2 300.0	2 A1-2 B2-2 C3-2 300.0	3 C1-1 E1-2 A1-1 300.0	4 F2-1 D2-2 D1-2 300.0	5 E2-2 G2-2 C1-2 300.0
10 D1-1 B1-2 J3-2 300.0	9 C3-1 J1-2 E2-1 300.0	8 B2-1 J2-2 H1-2 300.0	7 G3-1 I1-1 D3-2 300.0	6 E3-2 H3-2 B3-2 300.0

1 C2-1 I2-2 A2-1 300.0	2 C2-2 B1-1 J2-1 300.0	3 I3-1 H2-2 F1-2 300.0	4 F2-2 F3-2 G1-1 300.0	5 I2-1 D3-1 D2-1 300.0
10 G2-1 J1-1 I3-2 300.0	9 I1-2 A3-1 B3-1 300.0	8 H1-1 J3-1 H2-1 300.0	7 G1-2 H3-1 E3-1 300.0	6 F1-1 F3-1 E1-1 300.0



Şekil 4.20. Çaprazlama işlemine ait bir örnek

4.3.4. Önerilen mutasyon operatörleri

Bu çalışmada, önerilen çaprazlama operatörü yaklaşımla elde edilen sonuçları en iyi çözüme yaklaştırmak için probleme özgü üç farklı mutasyon operatörü önerilmiştir.

4.3.4.1. Parselleri yer değiştir

Bu mutasyon operatörünün sözde kodu Şekil 4.21’de verilmektedir. Bu çalışmada *Parselleri yer değiştir* operatörünün kullanım olasılığı 0.25 olarak alınmıştır. Bir başka ifadeyle if “ $rand < 0.25$ ” ifadesi karşılanırsa bu mutasyon uygulanır. Ek olarak, *Parselleri yer değiştir* mutasyon operatörüne ait bir örnek Şekil 4.22’de verilmiştir.

1. Rastgele iki farklı parsel seç.
2. Bu parsellerdeki tüm malikleri yer değiştir.
3. Parsel alanlarını malik alanlarının toplamına göre güncelle.

Şekil 4.21. Parselleri yer değiştir operatörüne ait sözde kod

1 I2-2 A2-2 J3-2 300.0	2 C1-2 B2-2 B1-2 300.0	3 G3-2 E2-1 D2-2 300.0	4 D3-1 F2-2 D1-1 300.0	5 G2-1 E3-2 E1-1 300.0
10 I3-1 J2-2 A3-2 300.0	9 I1-2 G3-2 C1-1 300.0	8 J1-2 H3-2 C2-2 300.0	7 E1-2 F1-2 D1-2 300.0	6 G1-2 E2-2 H3-1 300.0

1 A2-1 B1-1 G3-1 300.0	2 G2-2 G1-1 F2-1 300.0	3 A1-1 E3-1 D2-1 300.0	4 F3-1 F1-1 D3-2 300.0	5 F3-2 C2-1 A1-2 300.0
10 J1-1 C3-1 A3-1 300.0	9 J2-1 H1-1 H2-1 300.0	8 B3-1 I3-2 H2-2 300.0	7 J3-1 I2-1 H1-2 300.0	6 I1-1 B3-2 B2-1 300.0

Mutasyondan önce (a)



1 I2-2 A2-2 J3-2 300.0	2 C1-2 B2-2 B1-2 300.0	3 G3-2 E2-1 D2-2 300.0	4 B3-1 I3-2 H2-2 300.0	5 G2-1 E3-2 E1-1 300.0
10 I3-1 J2-2 A3-2 300.0	9 I1-2 G3-2 C1-1 300.0	8 J1-2 H3-2 C2-2 300.0	7 E1-2 F1-2 D1-2 300.0	6 G1-2 E2-2 H3-1 300.0

1 A2-1 B1-1 G3-1 300.0	2 G2-2 G1-1 F2-1 300.0	3 A1-1 E3-1 D2-1 300.0	4 F3-1 F1-1 D3-2 300.0	5 F3-2 C2-1 A1-2 300.0
10 J1-1 C3-1 A3-1 300.0	9 J2-1 H1-1 H2-1 300.0	8 D3-1 F2-2 D1-1 300.0	7 J3-1 I2-1 H1-2 300.0	6 I1-1 B3-2 B2-1 300.0

Mutasyondan sonra (b)

Şekil 4.22. Parselleri yer değiştir operatörüne ait bir örnek

4.3.4.2. Malikleri yer deęiřtir

Malikleri yer deęiřtir mutasyon operatörüne ait sözde kod Şekil 4.23'te verilmektedir. Bu çalışmada bu mutasyonun çalışma olasılığı 0.25 olarak alınmıştır. Ayrıca bu operatör malik sayısı 1'den fazla ve rastgele deęer 0.25'ten küçük olduęu durumda kullanılır. Söz konusu mutasyon operatörüne ait bir örnek ise Şekil 4.24'te verilmektedir.

1. Rastgele seçilen iki farklı parselden rastgele iki farklı malik seç.
2. Bu malikleri kendi aralarında yer deęiřtir.
3. Parsel alanlarını maliklerin alanlarının toplamına göre güncelle.

Şekil 4.23. *Malikleri yer deęiřtir* operatörüne ait sözde kod

1 I2-2 A2-2 J3-2 300.0	2 C1-2 B2-2 B1-2 300.0	3 G3-2 E2-1 D2-2 300.0	4 D3-1 F2-2 D1-1 300.0	5 G2-1 E3-2 E1-1 300.0
10 I3-1 J2-2 A3-2 300.0	9 I1-2 C3-2 C1-1 300.0	8 J1-2 H3-2 C2-2 300.0	7 E1-2 F1-2 D1-2 300.0	6 G1-2 E2-2 H3-1 300.0

1 A2-1 B1-1 G3-1 300.0	2 G2-2 G1-1 F2-1 300.0	3 A1-1 E3-1 D2-1 300.0	4 F3-1 F1-1 D3-2 300.0	5 F3-2 C2-1 A1-2 300.0
10 J1-1 C3-1 A3-1 300.0	9 J2-1 H1-1 H2-1 300.0	8 B3-1 I3-2 H2-2 300.0	7 J3-1 I2-1 H1-2 300.0	6 I1-1 B3-2 B2-1 300.0

Mutasyondan önce (a)



1 I2-2 A2-2 J3-2 300.0	2 C1-2 B2-2 B1-2 300.0	3 G3-2 E2-1 D2-2 300.0	4 D3-1 F2-2 I3-2 300.0	5 G2-1 E3-2 E1-1 300.0
10 I3-1 J2-2 A3-2 300.0	9 I1-2 C3-2 C1-1 300.0	8 J1-2 H3-2 C2-2 300.0	7 E1-2 F1-2 D1-2 300.0	6 G1-2 E2-2 H3-1 300.0

1 A2-1 B1-1 G3-1 300.0	2 G2-2 G1-1 F2-1 300.0	3 A1-1 E3-1 D2-1 300.0	4 F3-1 F1-1 D3-2 300.0	5 F3-2 C2-1 A1-2 300.0
10 J1-1 C3-1 A3-1 300.0	9 J2-1 H1-1 H2-1 300.0	8 B3-1 D1-1 H2-2 300.0	7 J3-1 I2-1 H1-2 300.0	6 I1-1 B3-2 B2-1 300.0

Mutasyondan sonra (b)

Şekil 4.24. Malikleri yer değiştir operatörüne ait bir örnek

4.3.4.3. Malik ekle

Malik ekle mutasyon operatörüne ait sözde kod Şekil 4.25’de sunulmaktadır. Bu çalışmada *Malik ekle* operatörün kullanım olasılığı 0.10 olarak belirlenmiştir. Bundan dolayı, bu operatör malik sayısı 1’den fazla ve rastgele değer 0.10’dan küçük olduğunda çalıştırılır. Ayrıca, söz konusu mutasyon operatörüne ait bir örnek Şekil 4.26’da verilmektedir.

1. Rastgele bir malik ve bir parsel seç.
2. Seçilen malik seçilen parsele aitse, Adım 1’e git.
3. Seçilen maliki seçili parsele taşı.
4. Hem seçilen parselin hem de seçilen malikin önceki parselinin parsel alanlarını güncelle.

Şekil 4.25. *Malik ekle* operatörüne ait sözde kod

1 I2-2 A2-2 J3-2 300.0	2 C1-2 B2-2 B1-2 300.0	3 G3-2 E2-1 D2-2 300.0	4 D3-1 F2-2 D1-1 300.0	5 G2-1 E3-2 E1-1 300.0
10 I3-1 J2-2 A3-2 300.0	9 I1-2 C3-2 C1-1 300.0	8 J1-2 H3-2 C2-2 300.0	7 E1-2 F1-2 D1-2 300.0	6 G1-2 E2-2 H3-1 300.0

1 A2-1 B1-1 G3-1 300.0	2 G2-2 G1-1 F2-1 300.0	3 A1-1 E3-1 D2-1 300.0	4 F3-1 F1-1 D3-2 300.0	5 F3-2 C2-1 A1-2 300.0
10 J1-1 C3-1 A3-1 300.0	9 J2-1 H1-1 H2-1 300.0	8 B3-1 I3-2 H2-2 300.0	7 J3-1 I2-1 H1-2 300.0	6 I1-1 B3-2 B2-1 300.0

Mutasyondan önce (a)



1 I2-2 A2-2 J3-2 300.0	2 C1-2 B2-2 B1-2 300.0	3 G3-2 E2-1 D2-2 300.0	4 D3-1 F2-2 200.0	5 G2-1 E3-2 E1-1 D1-1 400.0
10 I3-1 J2-2 A3-2 300.0	9 I1-2 C3-2 C1-1 300.0	8 J1-2 H3-2 C2-2 300.0	7 E1-2 F1-2 D1-2 300.0	6 G1-2 E2-2 H3-1 300.0

1 A2-1 B1-1 G3-1 300.0	2 G2-2 G1-1 F2-1 300.0	3 A1-1 E3-1 D2-1 300.0	4 F3-1 F1-1 D3-2 300.0	5 F3-2 C2-1 A1-2 300.0
10 J1-1 C3-1 A3-1 300.0	9 J2-1 H1-1 H2-1 300.0	8 B3-1 I3-2 H2-2 300.0	7 J3-1 I2-1 H1-2 300.0	6 I1-1 B3-2 B2-1 300.0

Mutasyondan sonra (b)

Şekil 4.26. Malik ekle operatörüne ait bir örnek

4.3.5. Parselasyon planının güncellenmesi

Dağıtım işlemi tamamlandıktan sonra parsel alanları dikkate alınarak parselasyon planları güncellenir. Her parselin köşe noktaları ve buna bağlı olarak merkez noktaları, amaç fonksiyonunda yer alan Öklid uzaklık değeri için hesaplanır.

4.3.6. Yeni aday çözümün değerlendirilmesi

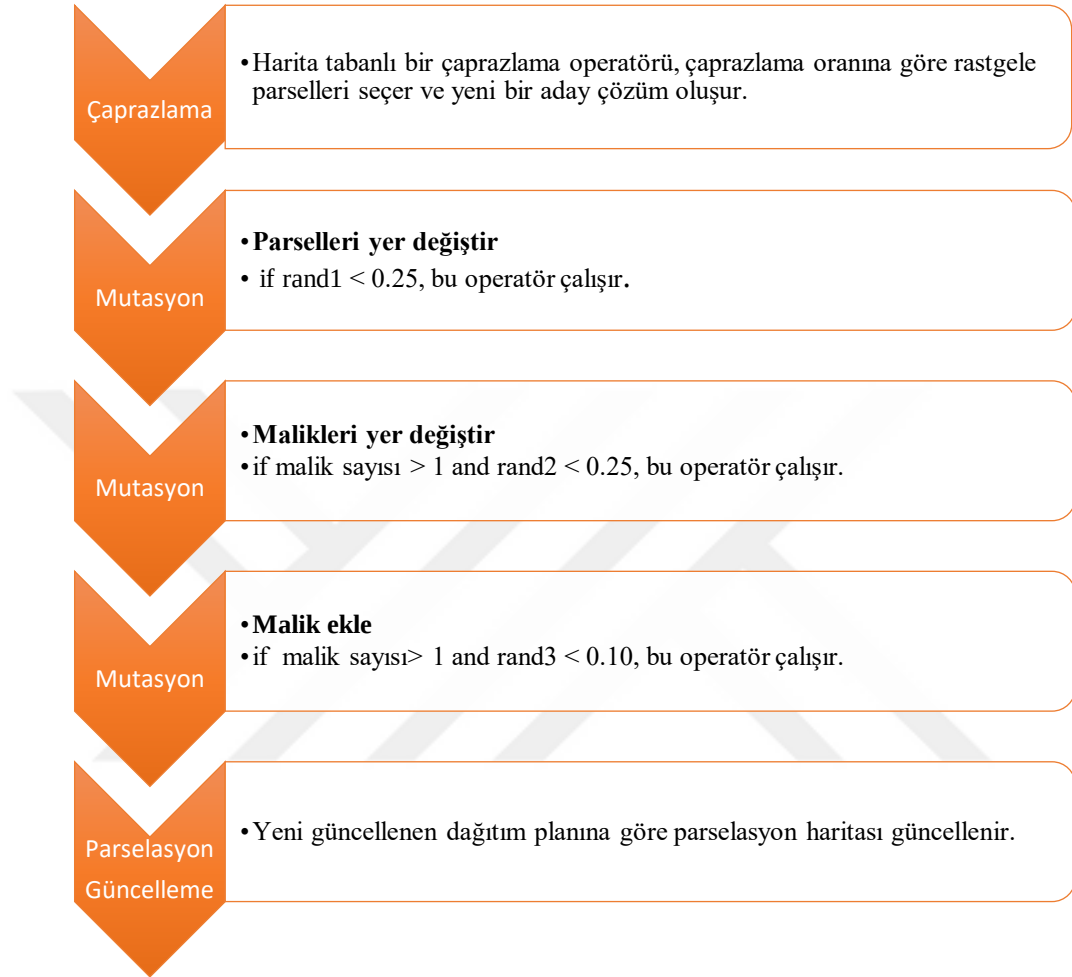
Her bir birey için dağıtım ve parselasyon planları güncellendikten sonra, bu değerler amaç fonksiyonuna gönderilir. Denklem 4.11'deki amaç fonksiyonunda, üç farklı ceza puanı dikkate alınarak her bir çözüm için maliyet değeri hesaplanır.

4.3.7. Algoritmanın sonlandırılması

Bu tez çalışmasında durdurma kriteri olarak en uygun çözümün bulunması belirlenmiştir. Önceden belirlenen iterasyon sayısı boyunca en uygun çözüm bulunamazsa, iterasyon sayısı durdurma kriteri olarak belirlenmiştir. Çaprazlama ve mutasyon operatörlerini içeren sentetik veri seti için önerilmiş olan hibrit yaklaşımın akış şeması Şekil 4.27'de verilmektedir. Herhangi iki birey çaprazlama işlemine maruz kaldığında, tüm işlemler sırasıyla bu şekilde uygulanır. Ayrıca, üç farklı mutasyon işlemi için üç farklı rasgele değer kullanılır. Her bir rastgele değer, söz konusu mutasyon operatörünün çalışıp çalışmayacağına karar verir. Ayrıca, malik sayısı 1'den fazla olduğunda *Malikleri yer değiştir* ve *Malik ekle* operatörleri kullanılır.

Bu tez çalışmasında sentetik veri setinde yer alan problemlerin çözümü için ABC, DE, GA, PSO ve TSA algoritmaları kullanılmaktadır. Algoritmalara ait gerçek proje verilerinin çözümü için önerilen akış diyagramları ise Şekil 4.10 - 4.14 verilmektedir. Sentetik veri seti için de bu akış diyagramları kullanılmaktadır. Fakat gerçek veriler için Şekil 4.10-4.14'te verilen her bireyin amaç fonksiyonunun hesaplanmasında kullanılan Denklem 4.5 yerine sentetik veri seti için Denklem 4.11 kullanılmaktadır. Ayrıca gerçek proje sahası için algoritmalara ait başlangıç popülasyonu Şekil 4.3'deki sözde koda göre oluşturulurken, sentetik veri seti için başlangıç popülasyonu ise Şekil 4.19'daki gibi

oluşturulmaktadır. Bununla birlikte gerçek veriler için yeni bireyler üretmek için hibrit yaklaşım olarak Şekil 4.9 kullanılırken sentetik veri seti için Şekil 4.27 kullanılmaktadır.



Şekil 4.27. Sentetik veri seti için önerilen hibrit yaklaşım

5. DENEYSEL ÇALIŞMALAR

5.1. Ön Parselasyon İşlemi için Deneysel Çalışmalar

Bu tez çalışmasında; ön parselasyon işleminin gerçekleştirilebilmesi için kenar değerleri, alan değerleri ve hem kenar hem de alan değerleri yardımıyla üç farklı yöntem önerilmiştir. Konya İli Meram İlçesi Harmancık Mahallesi uygulama alanı olarak belirlenmiştir. Tüm deneyler *Intel Core i5 3.10 GHz CPU, 4 GB RAM ve Windows 10* işletim sistemine sahip bir bilgisayarda Microsoft® Visual Studio®.NET 2015 platformu kullanılarak gerçekleştirilmiştir. Çalışma alanı farklı şekillerdeki dört adadan oluşmaktadır. Test amaçlı kullanılan adaların plan ve bina alan bilgileri Tablo 5.1'de verilmiştir. Adayı ayırma çizgisi ile ikiye bölebilmek için ada genişliğinin en az 40 metre olması gerekmektedir. Bu değer minimum parsel boyutları için önemlidir. Projenin girdileri bina alanı, plan tipi gibi veriler olup, çıktısı ise parselasyon haritasıdır.

Çizelge 5.1. Adalara ait girdi bilgileri

Ada No	Ada Alanı (m ²)	Yapı Nizamı	Bina Alanı (m ²) (Kuzey veya Batı Yönü)	Bina Alanı (m ²) (Güney veya Doğu Yönü)
101	9936.440	A6	100 (Kuzey)	125 (Güney)
102	8590.187	A4	150 (Kuzey)	100 (Güney)
103	1687.430	B	0 *bilgi verilmemiştir	0 *bilgi verilmemiştir
104	7869.079	A5	125 (Batı)	175 (Doğu)

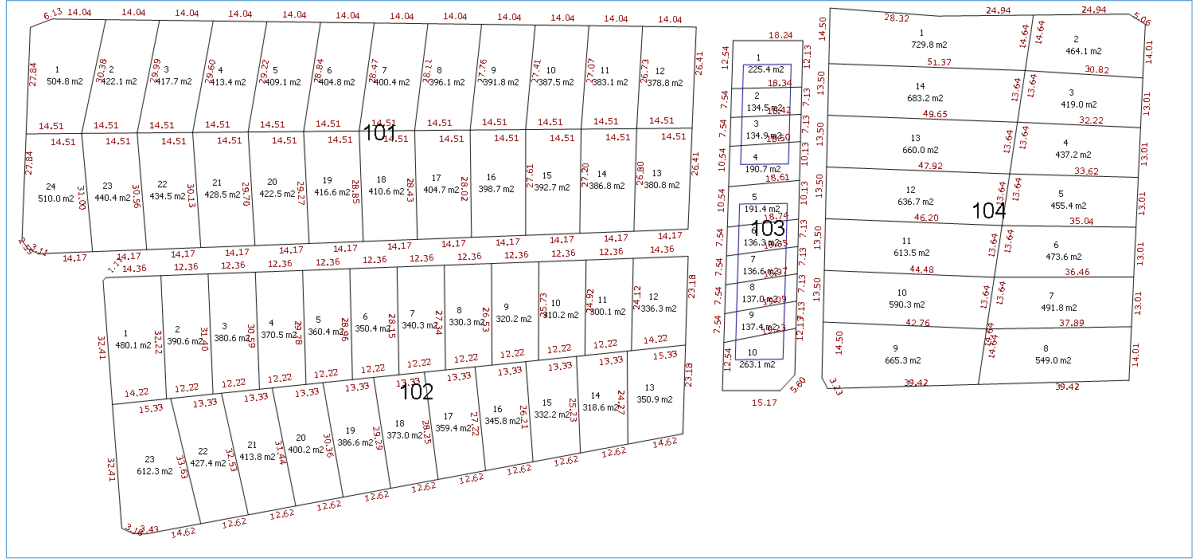
Çizelge 5.1'deki plan türleri, *PwEL* yöntemine göre ön parselasyon yapılırken kullanılacak parametrelerdir. Örneğin, 103 numaralı ada için *B* verilmiştir. Bu, 103 numaralı adanın *PwEL*'e göre parselasyona tabi olacağını göstermektedir. 101, 102 ve 104 numaralı adalar, Denklem 3.1'deki formüle göre minimum kenar uzunlukları hesaplanır ve bu bilgilere göre parselasyon işlemine tabi tutulur.

Çizelge 5.1'de verilen alan bilgisi adanın iki yönü için ayrı ayrı verilmektedir. Ada ayırma çizgisi ile ikiye bölündüğünde, adanın her bölümü için farklı alanlara sahip binalar inşa etmek mümkündür. Örneğin 102 numaralı ada için kuzey yönünde 150 m² alana sahip binalar, güney yönünde 100 m² alana sahip binalar yapılması hedeflenmektedir. Parseller için gerekli minimum alanlar FAR bilgileri dikkate alınarak hesaplanmaktadır. Denklem 3.2'ye göre ve daha sonra *PwA* ve *PwELA* yöntemine göre

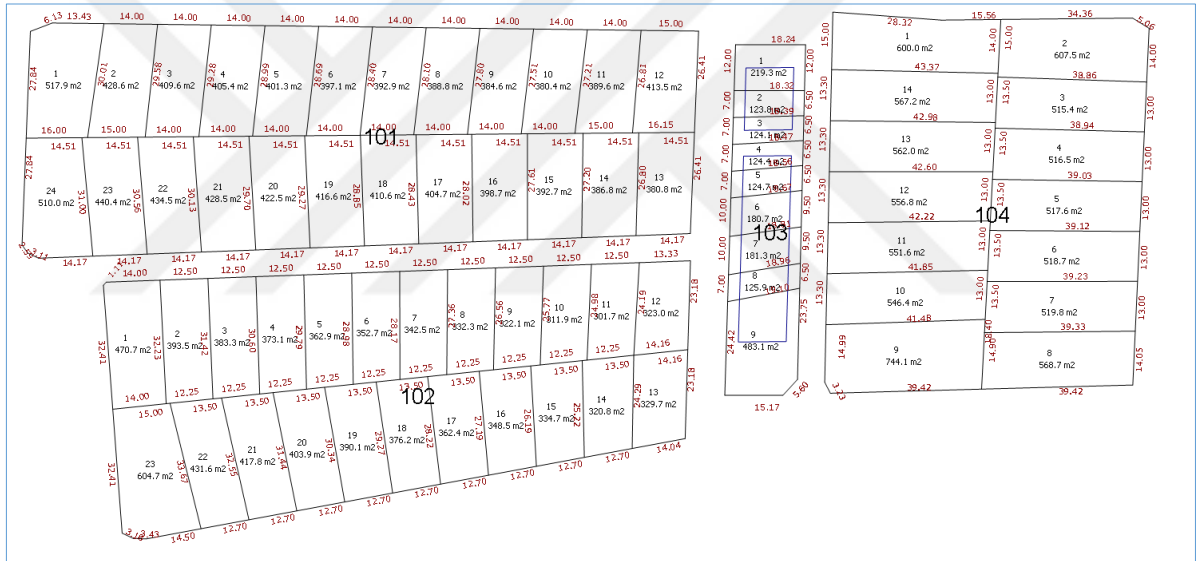
parselasyon işlemi gerçekleştirilir. İmar uygulamalarında genellikle 100 - 175m² büyüklüğünde konutlar tercih edildiğinden bu çalışmada konut büyüklükleri bu şekilde alınmıştır.

5.1.1. Kenar uzunluğuna göre ön parselasyon sonuçları (PwEL)

Kenar uzunluğuna göre ön parselasyon yöntemi (*PwEL*) ile elde edilen parselasyon sonucu Şekil 5.1'de gösterilmektedir. 101, 102 ve 104 numaralı adalar ayrık nizamdadır ve bu adalar Çizelge 5.1'de verilen yapı nizamına göre başarıyla parselasyon işlemine tabi tutulmaktadır. 103 numaralı ada, blok nizamdadır ve bu ada da blok nizam kurallarına göre başarıyla parsellenmiştir. Bu tez çalışmasında, her bir parsel otomatik olarak başarıyla numaralandırılmıştır. Ayrıca genişliği 40 metreden fazla olan adalar otomatik olarak ikiye bölünmektedir. CAD programından (NetCAD 5.1. versiyonu) girdi dosyasındaki alan değerlerine göre 1/1000 m² hassasiyetle her bir parsel için alan değerleri doğru bir şekilde *PwEL* yöntemi ile elde edilmektedir. Bir başka deyişle, her parsel için verilen alan değeri ve o adadaki tüm parsellerin alanları arasındaki maksimum fark 1/1000 m²'dir. *PwEL* yönteminde olduğu gibi, her bir parsel için alanların doğruluğu *PwA* ve *PwELA* yöntemleri için de aynı hassasiyet dikkate alınarak başarıyla gerçekleştirilmiştir. Çizelge 5.1'de verilen ada alanlarının her ada için üretilen parsellerin toplam alanına eşit olduğu görülmektedir. Ayrıca bir uzman tarafından elde edilen kenar uzunluğuna göre parselasyon sonucu da Şekil 5.2'de sunulmaktadır.



Şekil 5.1. PwEL yöntemiyle elde edilen parselasyon haritası



Şekil 5.2. Manuel yöntemle elde edilen parselasyon haritası

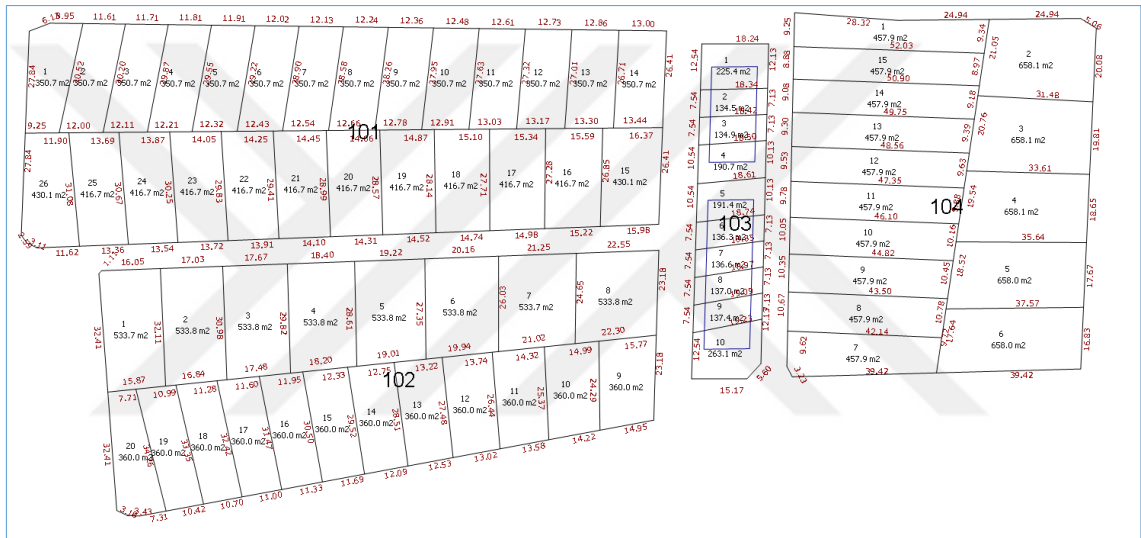
Şekil 5.1'te 103 numaralı ada için parselasyon sonucu incelendiğinde, bu sonuçlardan ada 103'ün Çizelge 1'de verilen bilgiler dikkate alınarak blok nizam kurallarına göre parselasyon işlemine tabi tutulduğu görülmektedir. Blok düzen kuralına göre ara parsellerin kenar uzunluğu en az 6 m, bitişik blokların bitişik parsellerinin kenar uzunlukları 9 m olmalıdır. Adanın köşe parselinin kenar uzunluğu ise en az 11 m olmalıdır. Şekil 5.1 incelendiğinde, parselasyon işleminin bu kurallara göre otomatik ve başarılı bir şekilde gerçekleştirildiği görülmektedir. Bu uygulamada önerilen yöntemle üstten başlayarak rastgele ara parsel sayısı 4 ve 3 olarak belirlenmiştir.

Ada 104 için durum şu şekildedir: A5 yapı nizamında ada 104 gösterildiği gibi, ara parsellerin kenar uzunlukları en az 12.5 m kuralını sağlamaktadır. Bu değer, Şekil 5.1'de görüldüğü gibi başarılı bir şekilde elde edilmiştir. Bu şekilde kısa kenarların her ikisi de 40 m'den uzun olduğu için, ada 104 iki kısma bölünmüş ve *PwEL* yöntemi adanın her iki kısmına da bağımsız olarak uygulanmıştır. Buradan da görüldüğü gibi, köşe parsellerin kenar uzunlukları minimum 14 m, ara parsellerin kenar uzunlukları ise minimum 13.0 metredir.

Şekil 5.1 ve 5.2 incelendiğinde, önerilen yöntemle elde edilen parselasyon planının uzman tarafından çizilen plana çok benzediği görülmektedir. Kenar uzunlukları açısından, her iki parselasyon haritası karşılaştırıldığında 101, 102 ve 103 adalarında sonuçların uzman görüşüne çok yakın olduğu açıkça görülmektedir. Bu nedenle parsel numarası, parsel şekli, parsel kenar uzunlukları ve parsel alanı gibi tüm kriterler açısından önerilen yöntemle elde edilen parselasyon planının uzman tarafından hazırlanan planla büyük ölçüde benzerlik gösterdiği görülmektedir. Ek olarak, ada 103, bina yerleşimi ve blok sırası açısından farklı parselasyon planları içerebilir. Mevzuata göre farklı parselasyon planlarının çözüm olarak kabul edilebileceği unutulmamalıdır. Bu durum tamamen ada içinde inşa edilmesi planlanan bitişik blokların sayısı ile ilgilidir. Uzman görüşüne göre önerilen yöntemle elde edilen plandaki en belirgin farklılık 104 numaralı adada görülmektedir. 104 numaralı adanın orta noktasının hesaplanmasındaki küçük bir hatadan dolayı *PwEL* sonucunun uzman görüşündeki sonuçtan biraz farklı olduğu görülmektedir. Ancak ideal çözüm uzman görüşünün de belirttiği gibi olmasına rağmen, önerilen yöntemle elde edilen sonuç da kabul edilebilir bir çözümdür. Sonuç olarak, *PwEL*'de tüm aşamaların baştan sona tamamen otomatik olduğu göz önüne alındığında bu farkın çok da önemli olmadığı söylenebilir.

5.1.2. Alana göre ön parselasyon sonuçları (PwA)

Alana göre ön parselasyon yöntemi (PwA) yöntemi ile elde edilen parselasyon sonucu Şekil 5.3'te gösterilmektedir. Denklem 3.2'deki gibi hesaplanan minimum parsel alanlarına göre 101, 102 ve 104 numaralı adalar için tüm parsellerin alanlarının başarıyla elde edildiği görülmektedir. Çizelge 5.1'de verilen bina alanları dikkate alınarak ada 103 için alan bilgisi verilmediğinden kenar uzunluk bilgileri dikkate alınarak parselasyon işlemi yapılmaktadır. Ayrıca uzman görüşüne göre yapılan parselasyon sonuçları önerilen yöntemin sonuçlarıyla aynı olduğu için uzman sonucu bu çalışmaya dâhil edilmemiştir.

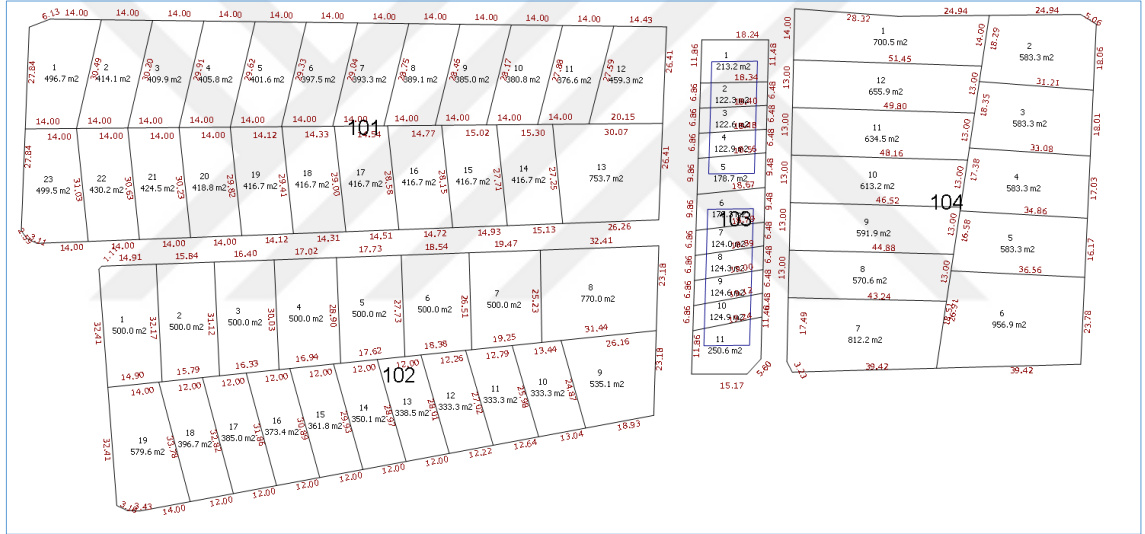


Şekil 5.3. PwA yöntemiyle elde edilen parselasyon haritası

101 numaralı ada için bina alanları Çizelge 5.1'de sırasıyla 100 m² ve 125 m² olarak verilmiş ve bu çalışmada FAR değeri 0.3 olarak alınmıştır. Bu değerler dikkate alınarak minimum parsel büyüklükleri Denklem 3.2'ye göre sırasıyla 333 m² ve 416.66 m² olmalıdır. 101 numaralı adada yer alan parsel alanlarının bu değerlerden büyük olduğu görülmektedir. Kuzey yönündeki tüm parseller 333 m²'den büyük olurken, güney yönündeki tüm parseller 417 m²'den büyüktür. Bu, önerilen yöntemin parselasyon işlemi başarıyla gerçekleştirdiğini göstermektedir. Ayrıca Şekil 5.3'te görüldüğü gibi parsel numaraları (parsel kimliği) kuzey yönünden başlayarak tüm parsellere başarıyla atanmaktadır.

5.1.3. Kenar uzunluğu ve alana göre ön parselasyon sonuçları (PwELA)

PwELA yöntemi ile elde edilen parselasyon sonucu Şekil 5.4'te gösterilmektedir. Diğer iki yöntemden farklı olarak *PwELA* yönteminde hem alan hem de kenar uzunluğu bilgileri dikkate alınarak parselasyon işlemi gerçekleştirilmiştir. Şekil 5.4'e bakıldığında 101, 102 ve 104 numaralı adalar için Çizelge 5.1'de verilen değerlere göre tüm alanların ve kenar uzunluklarının oldukça uygun olduğu görülmüş ve bunun sonucunda başarılı bir parselasyon işlemi yapılmıştır. Ayrıca, uzman tarafından oluşturulan parselasyon planları, önerilen yöntemin planları ile aynı olduğundan uzman sonucu bu çalışmaya dâhil edilmemiştir.



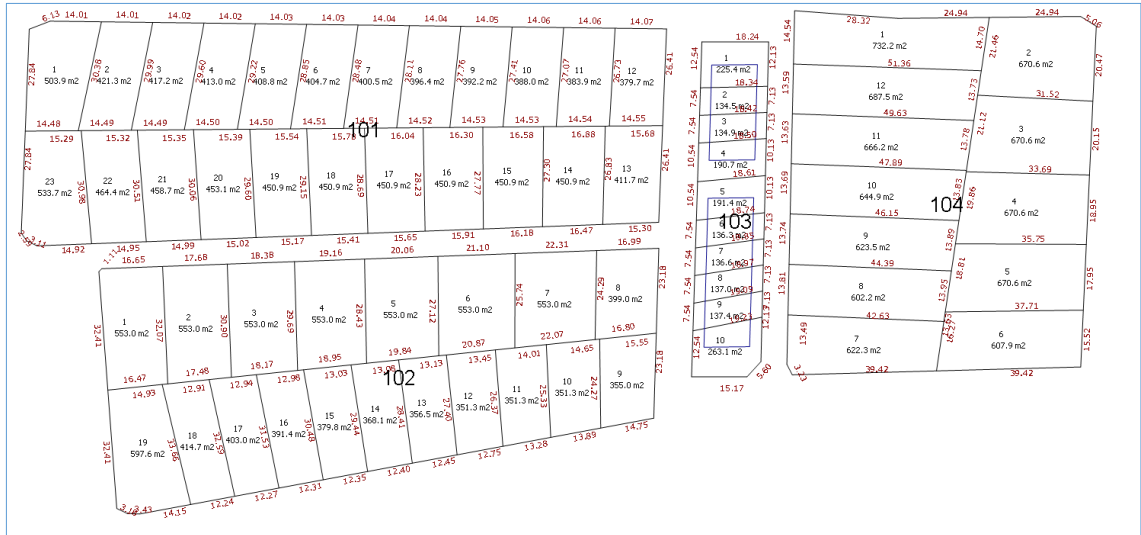
Şekil 5.4. PwELA yöntemiyle elde edilen parselasyon haritası

Şekil 5.4'te gösterildiği gibi, hem kenar uzunlukları hem de alan bilgisi birlikte alınarak başarılı bir parselasyon işlemi gerçekleştirilmiştir. Örneğin, ada 101 için parselasyon sonucu incelendiğinde kuzey yönünde yer alan parsellerin (101 Nolu adanın üst kısmında yer alan parseller) minimum alan ve kenar uzunluğu koşullarını sağladığı görülmektedir. Aynı şekilde güney yönünde yer alan parsellerin (alt kısmında yer alan parseller) minimum alan ve kenar uzunluğu şartlarını karşıladığı görülmektedir. 101 numaralı adaya bakıldığında 12 ve 13 numaralı parsellerin alan ve kenar uzunluklarının ada 101 içindeki diğer parsellerden daha büyük olduğu görülmektedir. Bunun nedeni

PwELA yönteminde hem kenar hem de alan bilgileri birlikte dikkate alındığından 2. adımdaki koşullu dallanmanın olmasıdır. Ayrıca Kuzeybatı köşesinden itibaren parsellere otomatik olarak parsel numaraları başarıyla atanmıştır.

5.1.3.1. *PwELA* ve *PwA* yönteminin birlikte kullanımı ile alan dengeleme

PwELA yöntemi ile elde edilen parselasyon sonucu Şekil 5.5'te gösterilmektedir. Bu işlemde, *PwELA* yöntemi çalıştırıldıktan sonra tekrar *PwA* yöntemi uygulanmış ve ardından her ada için parsel alanları dengelenmiştir. Bu sonuçlar Şekil 5.4 ile karşılaştırıldığında 101 numaralı adanın 12 ve 13 numaralı parselleri, 102 numaralı adanın 8 ve 9 numaralı parselleri ile 104 numaralı adanın ise 6 ve 7 numaralı parsellerinin alan ve kenar uzunluklarının dengelendiği açıkça görülmektedir. Bu yöntemde *PwELA* yönteminde olduğu gibi kenar ve alan değerleri dikkate alınarak parselasyon işlemi başarıyla gerçekleştirilmiştir. Ardından diğer yöntemlerde olduğu gibi bu yöntemde de parsel numaraları otomatik olarak parsellere atanmıştır. Bu yöntem sayesinde *PwELA* yönteminde ortaya çıkan son parsellerin alan ve cephe uzunluğundaki fazlalıklar giderilmiş ve parseller arasında alan ve cephe uzunluğu bakımından başarılı bir dengeleme sağlanmıştır.



Şekil 5.5. *PwELA* ve *PwA* yöntemi birlikte kullanılarak elde edilen parselasyon haritası

Bu tez çalışmasında imar uygulamaları için dört farklı parselasyon yöntemi önerilmiştir. *PwEL* yöntemi, parsellerin alanlarını hesaplamak için Gauss yöntemini kullanarak kenar uzunluğuna göre parselasyon işlemi yapmıştır. *PwA* yöntemi, parsellerin alanlarını hesaplamak için Gauss'u kullanmış ve BS yöntemi ile ilgili parsellerin sınır noktalarını hesaplamıştır. *PwELA* yöntemi, hem *PwEL* hem de *PwA* yöntemini birlikte kullanmıştır. Böylece bu yöntem, her bir parsel için hem kenar hem de alan kriterlerini sağlamıştır. Dördüncü yöntem ise *PwELA* ve *PwA*'nın hibrit versiyonudur ve bu yöntem ilk olarak *PwELA* yöntemini kullanmış ve ardından parsellerin eşit alana sahip olmasını sağlamak için *PwA* yöntemini kullanmıştır.

Çizelge 5.2, ön parselasyon için önerilen yöntemlerin çalışma süresi açısından karşılaştırmalı sonuçlarını göstermektedir. Tablodaki tüm sonuçlar mili saniye (ms) cinsindendir. Tabloda her bir yöntem için ortalama (Ort), minimum (Min), maksimum (Maks) ve standart sapma (SS) değerleri verilmiştir. Çizelge 5.2'de verilen değerlere göre her yöntem bağımsız olarak 30 defa çalıştırılmıştır. Sonuçlar incelendiğinde tüm yöntemlerin çok kısa sürede sonuç verdiği görülmüştür. Bu nedenle BS algoritmasının parselasyon işlemi için zaman ve performans açısından çok uygun bir yöntem olduğu açıktır. Bu ölçekte bir projenin uzman tarafından birkaç saat içinde tamamlanacağı göz önüne alındığında önerilen yöntemlerin ne kadar gerekli olduğu daha da iyi anlaşılmaktadır. Proje sahasının büyümesine bağlı olarak bu süre çok daha fazla uzamaktadır. Bu nedenle parselasyon sürecini otomatikleştirmenin avantajının zaman açısından daha net görülmesi beklenmektedir. Sonuç olarak, BS algoritmasının düzgün olmayan geometrik şekillere sahip alanlar üzerinde parselasyon işlemi için kullanılabileceği açıktır.

Çizelge 5.2. Ön parselasyon için önerilen yöntemlere ait çalışma süreleri

<i>Metot</i>	<i>Ort (ms)</i>	<i>Min (ms)</i>	<i>Maks (ms)</i>	<i>SS (ms)</i>
<i>PwEL</i>	34.567	33	38	1.257
<i>PwA</i>	37.667	34	78	7.683
<i>PwELA</i>	33.567	31	62	5.402
<i>PwELA & PwA</i>	35.500	32	94	10.920

5.2. Gerçek bir Proje Sahası Kullanılarak Dağıtım ve Parselasyon Problemlerinin Çözümüne ait Deneysel Çalışmalar

Bu tez çalışmasında gerçek proje sahasında LR problemlerini çözmek için ABC, DE, GA, PSO ve TSA algoritmaları uygulanmıştır. Tüm deneysel çalışmalar *Intel Core i7 2.80 GHz CPU, 16 GB RAM ve Windows 10 64 bit işletim sistemine* sahip bir makinede çalıştırılmış ve tüm kodlar *Visual Studio 2019* yazılım geliştirme ortamında C# programlama dili ile gerçekleştirilmiştir. Önerilen yöntem, LR süreci halihazırda tamamlanmış olan Konya İlinin Meram İlçesi Harmancık Mahallesiinde yer alan gerçek bir proje sahası üzerinde uygulanmıştır. Uygulama alanına ait veriler Çizelge 3.1 ve 3.2’de verilmektedir. Bu verilere göre dağıtım ve parselasyon işlemleri toplamda 6 farklı imar adasından oluşan 48957.23 m² alana sahip bir proje sahası üzerinde uygulanmıştır. Ayrıca proje sahasının kadastro ve imar haritaları Şekil 3.4 ve 3.5’te verilmektedir. Bu uygulamada ayrıca 113 malik bulunmaktadır.

Algoritmaları adil bir şekilde karşılaştırmak için parametre değerleri birbirine eşit seçilmiştir. Maksimum değerlendirme sayısı (MaxFEs - *Maximum Number of Function Evaluation*) değeri her algoritma için 20,000 (0.2×10^5) olarak belirlenmiş ve tüm algoritmalar 20 kez bağımsız olarak çalıştırılmıştır. Gerçek dünya problemine ait önerilen algoritmaların parametre değerleri EK-1’de verilmektedir.

Çizelge 5.3 bu çalışmada kullanılan optimizasyon algoritmalarının performans açısından karşılaştırmalı sonuçlarını göstermektedir. Elde edilen sonuçlar incelendiğinde PSO algoritmasının diğer algoritmalara göre çok önemli bir başarı gösterdiği görülmektedir. PSO algoritmasında IPC çaprazlama tekniği kullanıldığında, CPC yöntemi kullanılarak elde edilen sonuçlardan çok daha iyi performans gösterdiği de açıktır. Algoritmaların kararlılığı incelendiğinde PSO algoritmasının diğer algoritmalara göre daha sağlam olduğu görülmektedir. PSO algoritmasından sonra ise en iyi başarıyı DE ve ABC algoritmaları sergilemektedir. GA ve TSA algoritmaları ise en az başarı sergileyen algoritmalar olmuştur.

Çizelge 5.4’teki uzman sistem tarafından desteklenen algoritmaların sonuçları incelendiğinde, PSO yönteminin GA, TSA, ABC ve DE algoritmalarından daha etkili sonuçlar ürettiği açıkça görülmektedir. IPC ve CPC operatörleri algoritmalar bazında karşılaştırıldığında, algoritmada IPC tekniği kullanıldığında her algoritmanın CPC’den

daha etkili bir performans gösterdiği görülmektedir. Minimum maliyet değerleri incelendiğinde, algoritmaların performansının özellikle DE ve PSO yöntemlerinde yaklaşık 2 ila 3 kat arttığı görülmektedir. Ortalama maliyet değerleri incelendiğinde tüm algoritmaların performansının 2 kat arttığı görülmektedir. Sonuç olarak, algoritmalar arasında *IPC* tabanlı PSO ve uzman sistem tarafından desteklenen *IPC* tabanlı PSO, sırasıyla 3.929, 6.652 değeri ile en iyi performansa sahip olduğu görülmektedir.

Çizelge 5.3. Performans açısından algoritmaların karşılaştırılması

Algoritma	Çaprazlama Yöntemi	Maks Maliyet	Min Maliyet	Ort Maliyet	SS Maliyet
ABC	<i>CPC</i>	17.388	14.447	15.770	0.762
	<i>IPC</i>	9.634	7.460	8.654	0.615
DE	<i>CPC</i>	17.766	16.173	16.947	0.470
	<i>IPC</i>	8.428	6.479	7.406	0.494
PSO	<i>CPC</i>	11.717	9.432	10.375	0.573
	<i>IPC</i>	5.817	3.929	4.947	0.449
GA	<i>CPC</i>	21.582	15.678	18.685	1.346
	<i>IPC</i>	10.799	6.890	8.450	1.059
TSA	<i>CPC</i>	19.085	15.527	0.785	0.949
	<i>IPC</i>	9.586	6.048	7.717	1.023

Çizelge 5.4. Performans açısından uzman sistem destekli algoritmaların karşılaştırılması

Algoritma	Çaprazlama Yöntemi	Maks Maliyet	Min Maliyet	Ort Maliyet	SS Maliyet
ABCwES	<i>CPC</i>	20.203	17.640	18.563	0.683
	<i>IPC</i>	13.849	9.362	11.644	1.031
DEwES	<i>CPC</i>	21.230	17.750	19.359	0.741
	<i>IPC</i>	11.692	9.147	10.523	0.726
PSOwES	<i>CPC</i>	15.953	12.288	14.089	0.930
	<i>IPC</i>	9.687	6.652	8.050	0.841
GAwES	<i>CPC</i>	21.797	18.065	20.157	0.906
	<i>IPC</i>	12.743	9.305	11.178	1.040
TSAwES	<i>CPC</i>	21.967	17.207	0.890	1.171
	<i>IPC</i>	12.022	7.530	10.298	1.099

Önerilen algoritmaların zaman açısından karşılaştırmalı sonuçları Çizelge 5.5'te sunulmaktadır. Algoritmalar zaman olarak değerlendirildiğinde, GA algoritmasının diğerlerinden daha hızlı olduğu açıkça görülmektedir. GA'dan sonra ABC algoritması en hızlı algoritma olarak görülmektedir. Bu algoritmanın hızlı olmasının nedeni, popülasyonun yarısının işçi arılar, diğer yarısının da gözcü arılar olarak kullanılmasıdır.

Genel sonuçlar incelendiğinde algoritmaların 20,000 farklı aday çözümü maksimum 3 ila 13 dakika içerisinde değerlendirdiği görülmektedir. Ayrıca çaprazlama operatörleri açısından zamanlara bakıldığında, *IPC* yönteminin *CPC*'ye göre ek karar verme adımları nedeniyle biraz daha uzun sürede sonuç ürettiği görülmektedir. Sonuç olarak bu projenin bir uzman tarafından tamamlanmasının ortalama 3 gün sürdüğü düşünürse algoritmaların maksimum 15 dakika gibi kısa sürede etkili sonuçlar üretmesi çalışmanın literatüre katkısı ve gerçek dünyada alternatif bir araç olması açısından oldukça önemlidir.

Çizelge 5.5. Zaman açısından algoritmaların karşılaştırılması

Algoritma	Çaprazlama Yöntemi	Maks Süre	Min Süre	Ort Süre	SS Süre
ABC	<i>CPC</i>	4.56	4.22	4.33	0.09
	<i>IPC</i>	5.20	4.60	4.95	0.13
DE	<i>CPC</i>	10.44	8.91	9.45	0.42
	<i>IPC</i>	13.28	11.56	12.40	0.52
PSO	<i>CPC</i>	10.43	8.54	9.30	0.48
	<i>IPC</i>	13.88	11.94	13.03	0.51
GA	<i>CPC</i>	2.26	1.79	2.04	0.11
	<i>IPC</i>	4.47	3.41	3.79	0.28
TSA	<i>CPC</i>	6.33	4.61	0.42	0.48
	<i>IPC</i>	6.61	6.05	6.18	0.13

Farklı tekniklerin genel analizi karşılaştırmalı olarak Çizelge 5.6'da sunulmaktadır. Bu tablo, her algoritma için 20 bağımsız çalışmadan elde edilen en iyi sonuçları göstermektedir. Her algoritmanın maliyet değerlerine ek olarak, amaç fonksiyonunda yer alan ceza puanları, parsel sayısı ve malik sayısı da yer almaktadır. Uzman tarafından manuel olarak gerçekleştirilen resmi dağıtım ve parselasyon planları kullanılarak elde edilen sonuçlar da karşılaştırma amaçlı olarak tabloya eklenmiştir. Bu sonuçlara maliyet açısından bakıldığında, PSO yönteminin ABC, GA, TSA ve DE arasında en iyi sonucu verdiği görülmektedir. PSO algoritması, *Hisse* değerini 0 olarak bularak çok üstün bir performans göstermiştir. Buna ek olarak, PSO'nun hem *Boşluk* hem de *Mesafe* açısından ABC, GA, TSA ve DE'den daha iyi sonuçlar elde ettiği görülmektedir. *NS / NP* oranı incelendiğinde, PSO algoritması yine en yüksek başarıyı göstermiştir. Bu oran parsel başına düşen malik sayısını göstermektedir. Ayrıca bu tez çalışmasında, ABC, GA, TSA, DE ve PSO algoritmalarından elde edilen harita çıktılarının doğrudan gerçek dünyada kullanılabilir hale getirilmesi için, optimizasyon

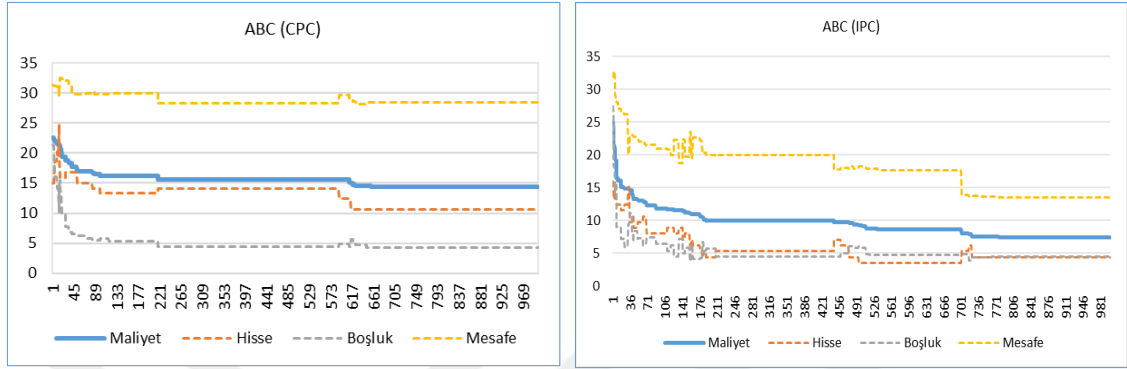
algoritmasının elde ettiği çözümler (haritalar) doğrudan bir uzman sisteme (ES) verilmiş ve kesin sonuçlar doğrudan ES tarafından elde edilmiştir. *ABCwES*, *GAwES*, *TSAwES*, *DEwES* ve *PSOwES* olarak adlandırılan algoritmalar incelendiğinde, bu algoritmalar ile elde edilen tüm sonuçların manuel olarak elde edilen 23.419 sonucundan çok daha iyi olduğu açıkça görülmektedir. *PSOwES* ise *Maliyet* değeri olarak 6.652 ile en iyi sonucu elde etmiştir ve teknisyenin elde ettiği değerin çok üzerinde bir başarı ortaya koymuştur. Ayrıca *PSOwES* yöntemi ile elde edilen sonuçlar incelendiğinde, bu yöntemin *Hisse* ve *Mesafe* ceza puanları açısından teknisyen sonuçlarından çok daha başarılı olduğu görülmektedir. Her ne kadar manuel olarak *Boşluk* değeri açısından daha iyi sonuç elde edilmiş olsa da, *PSOwES* ile elde edilen 0.014 ($\sim 48,000 \times 0.0014 = \sim 6.72 \text{ m}^2$) değerinin çok küçük ve kabul edilebilir bir alan değeri olduğu açıktır. Bu nedenle önerilen uzman sistem tabanlı algoritmalar sayesinde edilen çözümler, herhangi bir ara işlem veya araç olmaksızın doğrudan gerçek dünyada kullanılabilir durumdadır.

Çizelge 5.6. Farklı tekniklere ait karşılaştırmalı sonuçlar

Teknik	Maliyet	Hisse (%)	Boşluk (%)	Mesafe	Parsel Sayısı (NP)	Malik Sayısı (NS)	NS/NP
<i>ABC</i>	7.460	4.425	4.428	13.550	63	113	1.794
<i>ABCwES</i>	9.362	12.389	0.027	15.699	62	124	2.000
<i>DE</i>	6.479	3.540	2.258	13.659	62	113	1.823
<i>DEwES</i>	9.147	10.619	0.014	16.836	61	123	2.016
<i>PSO</i>	3.929	0.000	1.651	10.148	64	113	1.766
<i>PSOwES</i>	6.652	5.310	0.019	14.647	62	122	1.968
<i>GA</i>	6.890	0.885	5.383	14.424	63	113	1.794
<i>GAwES</i>	9.305	9.735	0.029	18.179	63	124	1.968
<i>TSA</i>	6.048	1.770	5.601	10.791	65	113	1.738
<i>TSAwES</i>	7.530	9.735	0.009	12.868	63	124	1.968
<i>Manuel</i>	23.419	43.363	0.002	26.892	67	195	2.910

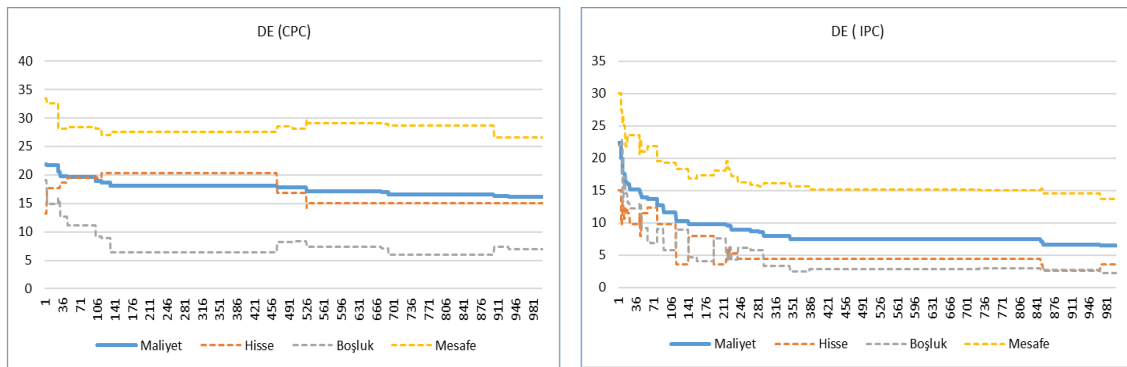
Algoritmaların yakınsama grafikleri Şekil 5.6-10'da verilmektedir. Şekil 5.6 incelendiğinde, *CPC* yönteminde yakınsama 600. iterasyon civarında dururken, *IPC* operatöründe yakınsamanın 700. iterasyona kadar devam ettiği görülmektedir. Bu iterasyonlardan sonra herhangi bir iyileşme görülmemektedir. Ayrıca, *IPC* yönteminin amaç fonksiyonunun genel çıktısı olan *Maliyet* açısından çok daha başarılı sonuçlar ürettiği görülmektedir. *CPC*'de *Boşluk* ceza değerlerinin *Mesafe* ve *Hisse* amaç değerlerine göre daha iyi yakınsadığı görülmektedir. Özellikle *CPC*'nin *Mesafe* ceza

değerlerinde ciddi bir iyileşme görülmemektedir. *IPC*'ye gelince, bu operatör kullanıldığında hem *Hisse* hem de *Boşluk* ceza değerlerinin *CPC*'ye benzer şekilde etkili yakınsamaya sahip olduğu görülmektedir. Son olarak, *IPC*'de, amaç fonksiyonundaki tüm değerlerin *CPC*'ye nazaran iyi bir yakınsama sergilediği görülmektedir.



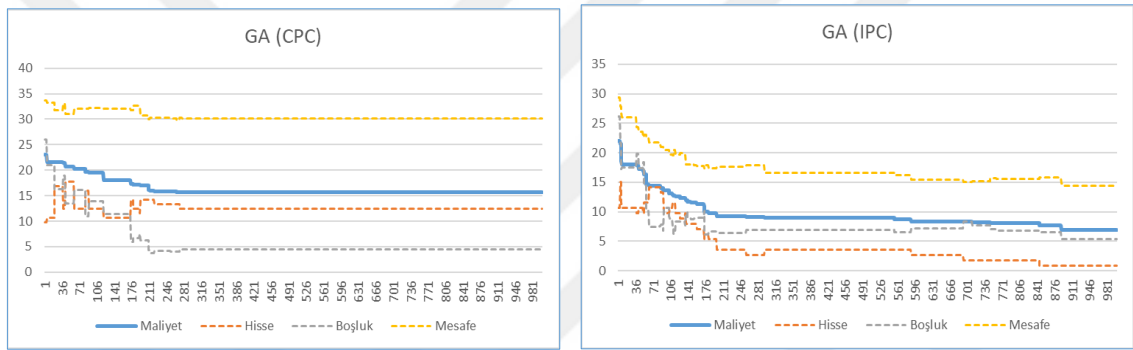
Şekil 5.6. ABC algoritmasına ait yakınsama grafikleri

Şekil 5.7'deki DE algoritmasına ait yakınsama grafikleri analiz edildiğinde, *CPC* operatöründeki yakınsama yaklaşık 900. iterasyona kadar devam ederken, *IPC* operatöründe yakınsamanın iterasyon boyunca devam ettiği açıkça görülmektedir. Ayrıca ABC'de olduğu gibi DE yönteminde de *IPC* yöntemi *Maliyet* açısından çok daha iyi sonuçlar vermiştir. Amaç değerlerine göre değerlendirildiğinde, *Boşluk* ceza değerlerinin her iki çaprazlama tekniğinde de en iyi yakınsamayı sergilediği, *Mesafe* ceza değerlerinin ise en zayıf yakınsama grafiği oluşturduğu görülmektedir. Ayrıca *IPC*'de *Hisse* değerlerinin *Boşluk* ceza değerine yakın sonuçlar elde edildiği görülmektedir. Sonuç olarak, *IPC* yönteminin üç farklı ceza puanı açısından *CPC*'den daha başarılı olduğu görülmektedir. Ayrıca iterasyon sayısı artırılırsa, DE'nin *IPC* kullanıldığında daha kaliteli çözümler elde edebileceği söylenebilir.



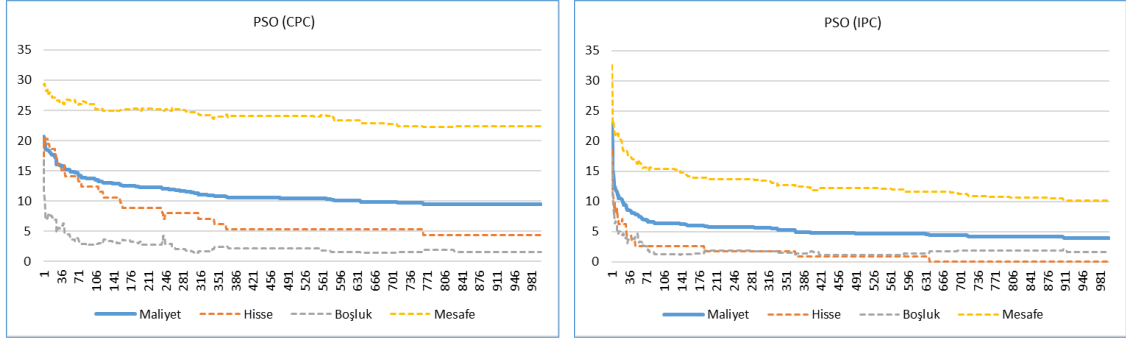
Şekil 5.7. DE algoritmasına ait yakınsama grafikleri

GA algoritmasına ait yakınsama grafikleri Şekil 5.8’de sunulmaktadır. Bu yakınsama grafikleri incelendiğinde, *CPC* yönteminde 200. iterasyonlarda yakınsamanın durduğu görülmektedir. *CPC* yönteminde en iyi yakınsamayı ise *Boşluk* ceza değeri üretmiştir. *IPC* operatörüne ait yakınsama grafiği incelendiğinde ise yakınsamanın iterasyon boyunca devam ettiği görülmektedir. Bu operatörde en iyi yakınsamayı *Hisse* değeri üretmiş ve *Hisse* değerini *Boşluk* değeri takip etmiştir. Sonuç olarak GA’da *CPC* operatörü kullanıldığında oluşan yakınsama sorununun *IPC* operatörü ile çözüme kavuşturulduğu görülmektedir. Bu da önerilen *IPC* tekniğinin ne ölçüde önemli olduğunu göstermektedir.



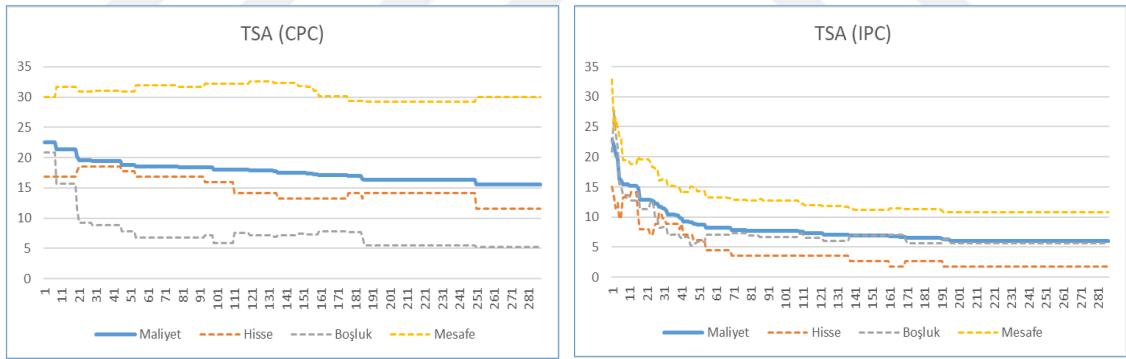
Şekil 5.8. GA algoritmasına ait yakınsama grafikleri

PSO algoritması için Şekil 5.9'daki iki farklı çaprazlama operatörüne ilişkin yakınsama grafikleri incelendiğinde, *CPC* ve *IPC* operatörünün yaklaşık 900. iterasyona kadar yakınsamaya devam ettiği gözlenmektedir. *CPC* operatöründe yakınsama başarısı açısından *Boşluk*, *Hisse* ve *Mesafe* ceza puanları olarak sıralandığı görülmektedir. *IPC* yönteminde de benzer durum olsa da, bu yöntemin *Hisse* ceza puanını en iyi çözüme kavuşturduğu görülmektedir. Ayrıca, *IPC* kullanan PSO'daki *Boşluk* değerlerine ait yakınsamanın diğer algoritmalarından çok daha başarılı olduğu da açıktır. Her iki teknik karşılaştırıldığında, ABC, DE, GA ve TSA'ya benzer şekilde, *IPC* operatörünün Maliyet sonuçları açısından *CPC*'ye göre çok iyi sonuçlar elde ettiği görülmektedir. Ayrıca, *IPC* operatörü sayesinde PSO'nun, ilk 100 yineleme içinde en hızlı yakınsamayı gösteren algoritma olduğu görülmektedir. Sonuç olarak, *IPC* tabanlı PSO'nun tüm algoritmalar arasında en başarılı yöntem olduğu açıktır.



Şekil 5.9. PSO algoritmasına ait yakınsama grafikleri

Şekil 5.10 TSA algoritmasına ait yakınsama grafiklerini sunmaktadır. İlk olarak *CPC* operatörüne ait yakınsama sonuçlarına bakıldığında, yakınsamanın 250. iterasyonlara kadar devam ettiği görülmektedir. *CPC* yönteminde en iyi yakınsamayı *Boşluk* değeri verirken, bu değeri *Hisse* değeri takip etmektedir. *IPC* yöntemi kullanıldığında ise yakınsama sonuçlarının *CPC*'den çok daha iyi olduğu görülmektedir. En iyi yakınsama sonucunu *Hisse* değeri verirken, bu değeri *Boşluk* değeri takip etmektedir. Sonuç olarak, tüm algoritmalar için çaprazlama operatörü olarak *CPC* yerine *IPC* yöntemi kullanmanın yakınsama hızını büyük ölçüde arttırdığı açıkça görülmektedir.



Şekil 5.10. TSA algoritmasına ait yakınsama grafikleri

Şekil 5.11 ve 5.12, sırasıyla PSO algoritması ve *PSOwES* ile elde edilen en iyi sonuçlara ait parselasyon haritalarını göstermektedir. 6 imar adasından oluşan bu haritada parsellerin bitişik cephelerinin açısı, her adanın kenar açısına paralel olarak otomatik olarak gerçekleştirilmiştir. Bu yaklaşım, herhangi bir ön işleme gerek kalmaksızın doğrudan kullanılabilir olacak düzenli bir parselasyon haritasının oluşmasını sağlamıştır. Şekil 5.11'e göre proje sahasındaki adalar taşma ve boşluk alanları içerdiği için bu çözüm, kesin çözüm yerine ara çözüm olarak kabul edilmektedir. Şekil 5.12 ise Şekil 5.11'de

optimizasyon algoritması ile elde edilen parselasyon ve dağıtım sonuçlarını girdi olarak alıp uzman sistem (ES) sayesinde elde edilmiş olan kesin parselasyon haritasını göstermektedir.

Şekil 5.6'daki ada alanındaki taşma alanlarına örnek olarak 37369 numaralı adadaki 7. parsel verilebilir. Bu parselin alanı PSO ile elde edilen parselasyon işlemi sonucunda 708.3 m^2 iken bu parselin alanı daha sonra ES sayesinde 616.2 m^2 olarak güncellenmiştir. Boşluk alanlarına örnek olarak 37373 adadaki 10 numaralı parsel de verilebilir. PSO sonucunda parselin alanı 430.6 m^2 iken bu alan daha sonra ES sayesinde 422.9 m^2 olarak güncellenmiştir. Şekil 5.11 ve Şekil 5.12 detaylıca incelendiğinde proje sahasında güncellenen tüm parseller şu şekildedir: 37369 numaralı adada 6 ve 7 numaralı parsel, 37370 numaralı adada 8 ve 9 numaralı parsel, 37371 numaralı adada 4 ve 5 numaralı parsel, 37373 numaralı adada 1 ve 10 numaralı parsel ve son olarak 37374 numaralı adada 3, 4, 5 ve 12 numaralı parseller. Ayrıca 37372 numaralı ada incelendiğinde toplam parsel sayısının değiştiği görülmektedir. PSO algoritmasına göre 7 parselden oluşan 37372 numaralı ada, daha sonra ES işlemi sayesinde 6 parsel düşmektedir. ES işlemleri Bölüm 4.2.9'da anlatıldığı gibi, ES ada alanlarındaki taşmaları ve boşlukları ortadan kaldırmak için kullanılmıştır. Sonuç olarak Şekil 5.11 ve 5.12 detaylı olarak incelendiğinde, alan olarak sadece son parsellerin değiştiği diğer parsellerde herhangi bir değişiklik olmadığı görülmektedir.

Bu çalışmada, amaç fonksiyonunda 3 farklı ceza puanı kullanılmıştır. Bu ceza puanlarından ikisi Madde 18'de belirtilmiştir (Anonim, 2020). Ek olarak, *Boşluk* değeri amaç fonksiyonuna dâhil edilmiştir. *Boşluk* ceza puanının hedef olarak belirlenmesinin avantajı, algoritmanın esnek bir şekilde çalışmasına olanak sağlanmasıdır. Aksi takdirde, algoritmanın her yinelemesinde boş alan kadar daha fazla hissedara veya belirli bir malik alanına bölmesi gerekmektedir. Bu durum da hisse sayısının artmasına neden olmaktadır. Bu nedenle, algoritmanın en iyi çözüme yakınsaması için bu ceza puanının amaç fonksiyonuna eklenmiştir. Sonuçları analiz ederken, PSO'nun ABC, GA, TSA ve DE arasında 3 hedefin tümü açısından en iyi algoritma olduğu açıkça görülmektedir. PSO ile elde edilen sonuçlar incelendiğinde *Hisse* değerinin 0 olduğu görülmektedir. Ancak tüm çözümlerde *Mesafe* değeri en başarısız ceza değeri olarak gözlemlenmektedir. Bunun nedeni, imar ve kadastro adaları Şekil 3.5'te gösterildiği gibi birbiriyle çakışmamaktadır. Ayrıca rastgele seçilen maliklerin herhangi bir alt kümesinin alanlarının toplamının bu

projedeki ada alanlarına tamamen eşit olmadığı da açıktır. Bir başka ifadeyle hiçbir şekilde herhangi bir malik grubunun alan olarak alt toplamı bir adanın alanıyla birebir eşit olmamaktadır. Bu yüzden, bu proje alanı için *Boşluk* ve *Mesafe* değerlerinin hiçbir zaman 0 olmayacağı açıktır. Sonuç olarak deneysel çalışmalar incelendiğinde PSO yönteminin diğer algoritmalarından daha kararlı ve daha başarılı olduğu görülmektedir. Ayrıca, *IPC* tekniğinin *CPC*'den çok daha başarılı olduğu açıktır. Buna ilaveten, geliştirilen uzman sistemin, algoritmalarla elde edilen mevcut çözümler üzerinde çok küçük değişiklikler yaparak etkili bir çözüm ürettiği deneysel sonuçlardan açıkça görülmektedir. Son olarak elde edilen çözümlerin manuel olarak üretilen çözüme göre daha kullanılabilir olduğu ve zaman açısından önerilen algoritmaların en iyi çözümü üretirken harcanan sürenin manuel çözüm için harcanan süreye göre çok daha kısa olduğu deneysel sonuçlarda açıkça görülmektedir.



Şekil 5.11. PSO algoritması tarafından elde edilen en iyi çözüme ait parselasyon haritası



Şekil 5.12. PSOWES tarafından elde edilen en iyi çözüme ait parselasyon haritası

5.3. Sentetik Veri Seti Kullanılarak Dağıtım ve Parselasyon Problemlerinin Çözümüne ait Deneysel Çalışmalar

Bu tez çalışmasında sentetik veri setindeki problemlerin çözümü için önerilen çaprazlama ve mutasyon operatörleri yardımıyla GE, ABC, DE, PSO ve TSA algoritmaları uygulanmıştır. Deneysel çalışmalar *Intel Core i7 2.80 GHz CPU, 16 GB RAM ve Windows 10 64 bit işletim sistemine* sahip bir makinede gerçekleştirilmiş olup, tüm kodlar C# programlama dili kullanılarak gerçekleştirilmiştir. Her problem bağımsız olarak 20 kez çalıştırılmış ve minimum (Min), maksimum (Maks), standart sapma (SS), ortalama (Ortalama), ortalama için sıra değerleri (Sıra (Ort)) ve standart sapma için sıra değerleri (Sıra (SS)) sonuçlar kaydedilmiştir. Ayrıca *Ortalama sıra, Final sırası, En iyi sıra* ve *En iyi sıra yüzdesi (%)* her bir sıra değeri dikkate alınarak sunulmuştur. Ayrıca algoritmaların hızlarını birbirleriyle karşılaştırmak için zaman sonuçları da saniye (s) cinsinden kaydedilmiştir. En iyi değere ulaşıldığında algoritma sonlandırılmış ve geçen süre raporlanmıştır. Bununla birlikte, algoritma en iyi değere ulaşmadığında ise toplam geçen süre kaydedilmiştir. En iyi değerler, tüm sonuçlar için kalın olarak verilmektedir. Algoritmaların giriş parametreleri de EK-2’de verilmektedir. Algoritmaların birbirleriyle adil bir şekilde karşılaştırılabilmesi için parametre değerleri eşit alınmıştır. Her algoritma, her bağımsız çalışmada 100,000 *MaxFEs* olarak çalıştırılmıştır. Ayrıca, algoritmaların belirli parametreleri mümkün olduğunca benzer alınmıştır. Sadece TSA algoritmasında, algoritmanın kendine has parametre değeri olan *ST* değeri, orijinal makaledeki gibi 0.3 olarak alınmıştır (Kiran, 2015).

5.3.1. Performans analizi

Çizelge 5.7 incelendiğinde GA'nın hem sağlamlık hem de performans açısından en iyi algoritma olduğu açıktır. Sıra değerlerine göre GA %75 başarıya ulaşırken PSO, GA'dan sonra %50 performansla ikinci en başarılı algoritma olmuştur. Diğer taraftan, DE ve ABC performanslarının birbirine yakın olduğu görülmektedir. Son olarak, TSA yalnızca parsel başına düşen malik sayısı (*NLP - number of landowner per parcel*) 1 olması durumunda çok başarılı sonuçlar elde etmiştir.

Çizelge 5.7. Parsel sayısı 20 için çözüm kalitesi açısından algoritmalara ait karşılaştırmalı sonuçlar

<i>Parsel başına düşen Malik sayısı</i>	ABC		DE		GA		PSO		TSA	
	<i>Min</i> ¹	<i>Maks</i> ²	<i>Min</i>	<i>Mak</i>	<i>Min</i>	<i>Maks</i>	<i>Min</i>	<i>Maks</i>	<i>Min</i>	<i>Maks</i>
	<i>Ort</i> ³	<i>SS</i> ⁴	<i>Ort</i>	<i>SS</i>	<i>Ort</i>	<i>SS</i>	<i>Ort</i>	<i>SS</i>	<i>Ort</i>	<i>SS</i>
	<i>Sıra(Ort)</i> ⁵	<i>Sıra(SS)</i> ⁶	<i>Sıra(Ort)</i>	<i>Sıra(SS)</i>	<i>Sıra(Ort)</i>	<i>Sıra(SS)</i>	<i>Sıra(Ort)</i>	<i>Sıra(SS)</i>	<i>Sıra(Ort)</i>	<i>Sıra(SS)</i>
1	0.000 ¹	0.000 ²	0.000	0.000	0.000	0.510	0.000	0.000	0.000	0.000
	0.000 ³	0.000 ⁴	0.000	0.000	0.090	0.147	0.000	0.000	0.000	0.000
	1 ⁵	1 ⁶	1	1	2	2	1	1	1	1
2	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.254
	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.597	0.506
	1	1	1	1	1	1	1	1	2	2
3	0.860	2.251	3.994	5.290	0.000	0.000	0.000	0.518	7.644	9.095
	1.752	0.385	4.743	0.378	0.000	0.000	0.238	0.157	8.347	0.385
	3	5	4	3	1	1	2	2	5	4
4	7.085	9.339	12.322	13.901	0.000	0.155	1.606	3.019	13.292	15.541
	8.182	0.520	13.181	0.464	0.008	0.034	2.226	0.370	14.258	0.471
	3	5	4	3	1	1	2	2	5	4
<i>Ortalama Sıra</i>	2.00	3.00	2.50	2.00	1.25	1.25	1.50	1.50	3.25	2.75
<i>Final Sırası</i>	3	5	4	3	1	1	2	2	5	4
<i>En İyi Sıra Sayısı</i>	2	2	2	2	3	3	2	2	1	1
<i>En İyi Sıra</i>										
<i>Yüzdesi (%)</i>	50	50	50	50	75	75	50	50	25	25

Çizelge 5.8, 40 parsel için karşılaştırmalı sonuçları vermektedir. Ortalama sıra değerlerine göre ABC GA'ya çok yakın olmasına rağmen, GA %75 en iyi sıra oranı ile en etkili yöntem olarak görülmektedir. PSO ise ABC'den sonraki üçüncü algoritma olmuştur. DE, bu problem setinde yalnızca bir kez en başarılı algoritma iken, TSA hiçbir problem setinde en başarılı algoritma olamamıştır. Sonuç olarak GA, hem kararlılık hem de performans açısından en etkili algoritma olmuştur.

Çizelge 5.8. Parsel sayısı 40 için çözüm kalitesi açısından algoritmalara ait karşılaştırmalı sonuçlar

Parsel başına düşen Malik sayısı	ABC		DE		GA		PSO		TSA	
	Min	Maks	Min	Maks	Min	Maks	Min	Maks	Min	Maks
	Ort	SS	Ort	SS	Ort	SS	Ort	SS	Ort	SS
	Sıra(Ort)	Sıra(SS)	Sıra(Ort)	Sıra(SS)	Sıra(Ort)	Sıra(SS)	Sıra(Ort)	Sıra(SS)	Sıra(Ort)	Sıra(SS)
1	0.000	0.000	0.000	0.000	0.000	1.174	0.000	0.510	0.000	0.516
	0.000	0.000	0.000	0.000	0.353	0.371	0.078	0.161	0.046	0.139
	1	1	1	1	4	4	3	3	2	2
2	0.000	0.840	9.120	10.495	0.000	0.415	0.929	2.249	11.646	14.428
	0.418	0.262	9.841	0.424	0.054	0.133	1.706	0.326	12.868	0.747
	2	2	4	4	1	1	3	3	5	5
3	14.235	17.227	23.990	26.891	0.155	1.672	11.336	14.587	31.649	36.061
	15.603	0.793	26.060	0.684	1.127	0.433	12.834	0.909	33.988	1.222
	3	3	4	2	1	1	2	4	5	5
4	26.263	30.730	47.054	49.507	2.335	3.920	33.258	40.005	46.952	50.245
	28.542	1.166	48.402	0.690	3.011	0.420	36.414	1.835	48.680	1.014
	2	4	4	2	1	1	3	5	5	3
Ortalama Sıra	2.00	2.50	3.25	2.25	1.75	1.75	2.75	3.75	4.25	3.75
Final Sırası	2	3	4	2	1	1	3	4	5	4
En İyi Sıra										
Sayısı	1	1	1	1	3	3	0	0	0	0
En İyi Sıra										
Yüzdesi (%)	25	25	25	25	75	75	0	0	0	0

Çizelge 5.9, 60 parsel için çözüm kalitesi açısından karşılaştırmalı sonuçları göstermektedir. Bu sonuçlara göre ortalama sıra değerleri göz önüne alındığında, GA, 1.25 değeriyle açıkça en başarılı algoritma iken, ABC 1.75 değeriyle GA'dan sonra etkili algoritmadır. Bu algoritmaları çözüm kalitesine göre sırasıyla PSO, DE ve TSA takip etmiştir. Ancak *En İyi Sıra Yüzdesi* değerlerine göre GA, 4 problemten 3'ünde en başarılı yöntem olup diğer algoritmalara göre büyük bir üstünlük elde etmiştir. Sonuç olarak, GA bu problem seti için de en güçlü algoritma olmuştur. 80 parsel için karşılaştırılmalı sonuçların sunulduğu Çizelge 5.10'daki sonuçlar da Çizelge 5.9'daki sonuçlara benzer şekildedir. Parsel sayısı 80 olan bu problem seti için GA yine en başarılı algoritmadır.

Çizelge 5.9. Parsel sayısı 60 için çözüm kalitesi açısından algoritmalara ait karşılaştırmalı sonuçlar

<i>Parsel başına düşen Malik sayısı</i>	ABC		DE		GA		PSO		TSA	
	<i>Min</i>	<i>Maks</i>	<i>Min</i>	<i>Maks</i>	<i>Min</i>	<i>Maks</i>	<i>Min</i>	<i>Maks</i>	<i>Min</i>	<i>Maks</i>
	<i>Ort</i>	<i>SS</i>	<i>Ort</i>	<i>SS</i>	<i>Ort</i>	<i>SS</i>	<i>Ort</i>	<i>SS</i>	<i>Ort</i>	<i>SS</i>
	<i>Sıra(Ort)</i>	<i>Sıra(SS)</i>	<i>Sıra(Ort)</i>	<i>Sıra(SS)</i>	<i>Sıra(Ort)</i>	<i>Sıra(SS)</i>	<i>Sıra(Ort)</i>	<i>Sıra(SS)</i>	<i>Sıra(Ort)</i>	<i>Sıra(SS)</i>
1	0.000	0.000	2.789	5.730	0.000	1.827	0.665	2.998	11.625	13.875
	0.000	0.000	4.403	0.793	0.500	0.496	1.442	0.592	12.307	0.466
	1	1	4	5	2	3	3	4	5	2
2	7.939	10.710	32.978	36.799	0.622	2.181	8.947	11.559	30.724	34.403
	9.079	0.866	34.975	0.933	1.471	0.398	10.378	0.770	32.720	1.181
	2	3	5	4	1	1	3	2	4	5
3	30.595	37.369	43.685	47.510	3.882	6.251	45.887	53.939	70.629	77.073
	35.363	1.893	46.022	0.867	5.254	0.732	49.251	2.162	73.948	1.543
	2	4	3	2	1	1	4	5	5	3
4	51.229	56.390	95.797	102.795	9.402	12.159	81.769	94.096	94.312	103.770
	54.411	1.498	100.202	1.953	10.537	0.721	88.064	2.988	100.720	2.044
	2	2	4	3	1	1	3	5	5	4
<i>Ortalama Sıra</i>	1.75	2.50	4.00	3.50	1.25	1.50	3.25	4.00	4.75	3.50
<i>Final Sırası</i>	2	2	4	3	1	1	3	4	5	3
<i>En İyi Sıra Sayısı</i>	1	1	0	0	3	3	0	0	0	0
<i>En İyi Sıra</i>										
<i>Yüzdesi (%)</i>	25	25	0	0	75	75	0	0	0	0

Çizelge 5.10. Parsel sayısı 80 için çözüm kalitesi açısından algoritmalara ait karşılaştırmalı sonuçlar

<i>Parsel başına düşen Malik sayısı</i>	ABC		DE		GA		PSO		TSA	
	<i>Min</i>	<i>Maks</i>	<i>Min</i>	<i>Maks</i>	<i>Min</i>	<i>Maks</i>	<i>Min</i>	<i>Maks</i>	<i>Min</i>	<i>Maks</i>
	<i>Ort</i>	<i>SS</i>	<i>Ort</i>	<i>SS</i>	<i>Ort</i>	<i>SS</i>	<i>Ort</i>	<i>SS</i>	<i>Ort</i>	<i>SS</i>
	<i>Sıra(Ort)</i>	<i>Sıra(SS)</i>	<i>Sıra(Ort)</i>	<i>Sıra(SS)</i>	<i>Sıra(Ort)</i>	<i>Sıra(SS)</i>	<i>Sıra(Ort)</i>	<i>Sıra(SS)</i>	<i>Sıra(Ort)</i>	<i>Sıra(SS)</i>
1	0.000	0.611	18.882	20.619	0.000	1.520	6.478	8.819	22.798	27.108
	0.070	0.139	19.819	0.441	0.645	0.434	7.975	0.537	24.956	1.243
	1	1	4	3	2	2	3	4	5	5
2	18.049	21.390	70.501	77.519	3.519	5.744	27.693	32.840	58.219	63.845
	19.615	0.920	73.572	1.956	4.756	0.566	30.221	1.487	60.916	1.577
	2	2	5	5	1	1	3	3	4	4
3	48.519	54.548	77.768	81.492	10.553	13.416	94.844	124.903	122.748	129.973
	52.052	1.607	79.357	0.891	11.951	0.765	104.122	6.651	126.474	2.236
	2	3	3	2	1	1	4	5	5	4
4	70.216	79.741	165.739	174.050	19.916	23.037	145.021	169.702	165.538	173.095
	75.823	2.234	170.725	2.243	21.457	0.917	155.822	5.987	170.008	2.301
	2	2	5	3	1	1	3	5	4	4
<i>Ortalama Sıra</i>	1.75	2.00	4.25	3.25	1.25	1.25	3.25	4.25	4.50	4.25
<i>Final Sırası</i>	2	2	4	3	1	1	3	4	5	4
<i>En İyi Sıra Sayısı</i>	1	1	0	0	3	3	0	0	0	0
<i>En İyi Sıra</i>										
<i>Yüzdesi (%)</i>	25	25	0	0	75	75	0	0	0	0

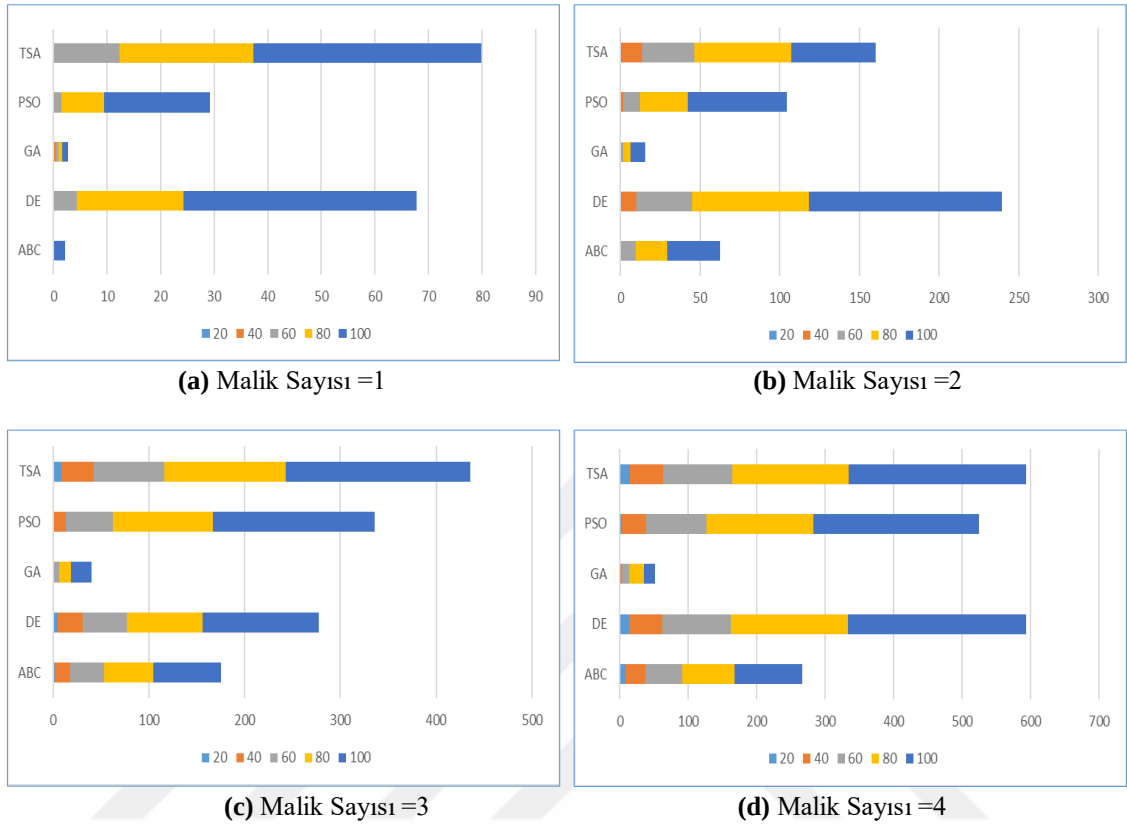
Çözüm kalitesi açısından 100 parsel problemi için karşılaştırmalı sonuçlar Çizelge 5.11'de sunulmaktadır. Bu sonuçlara göre GA, %100 ortalama sıra değeri ile tüm problemlerde en etkili yöntemdir. Daha sonra ABC, ortalama 2.00 sıralamasıyla GA'dan sonra 2. algoritmadır. Bu algoritmaları performans açısından PSO, TSA ve DE algoritmaları takip etmiştir. Sonuç olarak, bu problem seti için GA en güçlü ve kararlı algoritma olmuştur.

Çizelge 5.11. Parsel sayısı 100 için çözüm kalitesi açısından algoritmalara ait karşılaştırmalı sonuçlar

<i>Parsel başına düşen Malik sayısı</i>	ABC		DE		GA		PSO		TSA	
	<i>Min Ort</i>	<i>Maks SS</i>	<i>Min Ort</i>	<i>Maks SS</i>	<i>Min Ort</i>	<i>Maks SS</i>	<i>Min Ort</i>	<i>Maks SS</i>	<i>Min Ort</i>	<i>Maks SS</i>
	<i>Sıra(Ort)</i>	<i>Sıra(SS)</i>	<i>Sıra(Ort)</i>	<i>Sıra(SS)</i>	<i>Sıra(Ort)</i>	<i>Sıra(SS)</i>	<i>Sıra(Ort)</i>	<i>Sıra(SS)</i>	<i>Sıra(Ort)</i>	<i>Sıra(SS)</i>
1	0.642	4.222	41.051	46.086	0.259	1.890	17.823	22.290	38.321	47.323
	2.126	0.896	43.534	1.391	1.129	0.486	19.653	1.291	42.533	2.343
	2	2	5	4	1	1	3	3	4	5
2	30.743	36.750	118.241	126.715	7.976	10.290	56.027	66.223	49.319	55.964
	33.507	2.049	121.251	1.968	9.050	0.581	62.367	2.414	53.149	1.529
	2	4	5	3	1	1	4	5	3	2
3	63.952	77.259	119.031	124.964	19.136	23.739	156.615	179.406	177.846	199.007
	70.700	2.971	121.436	1.529	21.313	1.162	169.567	5.696	192.564	4.090
	2	3	3	2	1	1	4	5	5	4
4	89.928	104.242	253.310	266.865	15.266	17.699	226.180	254.037	253.910	265.342
	99.484	3.702	260.898	3.666	16.472	0.610	241.993	6.608	259.565	3.228
	2	4	5	3	1	1	3	5	4	2
<i>Ortalama Sıra</i>	2.00	3.25	4.50	3.00	1.000	1.000	3.50	4.50	4.00	3.25
<i>Final Sırası</i>	2	3	5	2	1	1	3	4	4	3
<i>En İyi Sıra Sayısı</i>	0	0	0	0	4	4	0	0	0	0
<i>En İyi Sıra</i>										
<i>Yüzdesi (%)</i>	0	0	0	0	100	100	0	0	0	0

Parsel başına düşen malik sayısı (*NLP*) açısından algoritma performansları karşılaştırmalı olarak Şekil 5.13'te verilmektedir. Sonuçlar ortalama değerlere göre analiz edildiğinde, *NLP*= 1 olduğunda, ABC GA'nin elde ettiği sonuçlara yakın sonuçlar elde etse de GA en etkili algoritma olmuştur. Buna ilaveten, ABC çözüm kalitesi açısından GA'ya en yakın yöntemdir. Diğer taraftan, PSO, DE ve TSA performans açısından GA ve ABC'den sonra sıralanmıştır. *NLP*= 2 için benzer bir durum olsa da, bu kez GA, ABC'den çok daha iyi bir başarı göstermiştir. Algoritmalar performans açısından GA ve ABC'den sonra PSO, TSA ve DE şeklinde sıralanmıştır. *NLP* değeri 3 olduğunda GA, *NLP* değeri 1 ve 2 olduğundaki başarısına kıyasla diğer algoritmalarından çok daha etkili olmuştur. ABC, bu problem setinde 2. en etkili algoritma olmasına rağmen, DE, ABC'ye yakın bir sonuç üretmiştir. Ayrıca çözüm kalitesi açısından DE'yi PSO ve TSA izlemiştir. Son olarak, *NLP* değeri 4 iken, GA tüm algoritmalarından çok daha üstün bir başarı yakalamıştır. Ayrıca ABC, GA'ya en yakın algoritma olmasına rağmen, GA, performans açısından çok etkili olmuştur. Diğer taraftan ABC'den sonra en etkili algoritma olan PSO'yu, birbirlerine yakın sonuçlar üreten TSA ve DE takip etmiştir. Bu sonuçlara göre parsel başına düşen malik sayısı arttıkça GA başarısını ve kararlılığını sürdürmüştür. Bu,

GA'nın büyük ölçekli problemlerde diğer algoritmalara nazaran çok daha etkili sonuçlar elde edebileceğini göstermektedir.



Şekil 5.13. Ortalama değerlere göre karşılaştırmalı performans analizi

GA tarafından deneysel çalışmalarda elde edilen LR sonuçları Şekil 5.14, 5.15, EK-3.1 ve EK-3.2'de verilmektedir. Şekil 5.14, parsel sayısı 20 ve *NLP* değeri 3 olduğunda GA tarafından elde edilen bir çözümü göstermektedir. Buna ilaveten Şekil 5.15 de GA ile elde edilen *NLP* değeri 4 için en uygun çözümü göstermektedir. Sonuçlar incelendiğinde girdi olarak verilen dağıtım ve parselasyon planları ile çıktıların birebir aynı olduğu açıktır. Bu, GA'nın her iki problem için de çok başarılı olduğunu göstermektedir. EK-3.1, parsel sayısı ve *NLP* sırasıyla 40 ve 2 olduğunda GA ile elde edilen sonucu göstermektedir. Sonuçların giriş verileriyle tamamen aynı olduğu görülmektedir. GA yönteminin bu problemler için en uygun sonucu bulduğu açıktır. EK-3.2 incelendiğinde 100 parselle sahip zor bir problem için GA ile 100,000 (10^5) FEs sonucunda 115 saniye gibi kısa bir sürede çok etkili bir sonuç elde edilmiştir. Bu sonuca göre 100 malikten sadece 2 malikin uygun parsellere tahsis edilmediği, diğer tüm maliklerin dağıtım ve parsel haritasına göre uygun şekilde tahsis edildiği görülmektedir.

Bu çözümde *Ada 2*'deki 25 ve 27 numaralı parsellerin basit bir takas işlemi ile maliklerin değişimi yapılarak en iyi çözümün elde edilebileceği açıktır. Sonuç olarak GA yönteminin sentetik verilerin çözümü için oldukça başarılı olduğu görülmektedir.

1 ● A2-2 ● A3-2 ● A1-2 300.0	2 ● B1-2 ● B2-2 ● B3-2 300.0	3 ● C1-2 ● C2-2 ● C3-2 300.0	4 ● D2-2 ● D3-2 ● D1-2 300.0	5 ● E3-2 ● E2-2 ● E1-2 300.0
10 ● J3-2 ● J2-2 ● J1-2 300.0	9 ● I3-2 ● I2-2 ● I1-2 300.0	8 ● H1-2 ● H3-2 ● H2-2 300.0	7 ● G2-2 ● G1-2 ● G3-2 300.0	6 ● F1-2 ● F2-2 ● F3-2 300.0
1 ● A3-1 ● A2-1 ● A1-1 300.0	2 ● B3-1 ● B2-1 ● B1-1 300.0	3 ● C1-1 ● C2-1 ● C3-1 300.0	4 ● D1-1 ● D2-1 ● D3-1 300.0	5 ● E1-1 ● E2-1 ● E3-1 300.0
10 ● J1-1 ● J3-1 ● J2-1 300.0	9 ● I3-1 ● I1-1 ● I2-1 300.0	8 ● H3-1 ● H1-1 ● H2-1 300.0	7 ● G2-1 ● G1-1 ● G3-1 300.0	6 ● F3-1 ● F2-1 ● F1-1 300.0

Şekil 5.14. Parsel sayısı 20, NLP değeri 3 olan probleme ait en iyi çözüme ait bir örnek

1 ● A4-2 ● A3-2 ● A2-2 ● A1-2 300.0	2 ● B1-2 ● B4-2 ● B2-2 ● B3-2 300.0	3 ● C2-2 ● C3-2 ● C4-2 ● C1-2 300.0	4 ● D1-2 ● D4-2 ● D3-2 ● D2-2 300.0	5 ● E3-2 ● E2-2 ● E1-2 ● E4-2 300.0
10 ● J1-2 ● J4-2 ● J3-2 ● J2-2 300.0	9 ● I2-2 ● I3-2 ● I4-2 ● I1-2 300.0	8 ● H3-2 ● H1-2 ● H4-2 ● H2-2 300.0	7 ● G4-2 ● G3-2 ● G2-2 ● G1-2 300.0	6 ● F4-2 ● F2-2 ● F1-2 ● F3-2 300.0
1 ● A4-1 ● A2-1 ● A1-1 ● A3-1 300.0	2 ● B1-1 ● B4-1 ● B3-1 ● B2-1 300.0	3 ● C1-1 ● C2-1 ● C3-1 ● C4-1 300.0	4 ● D3-1 ● D4-1 ● D1-1 ● D2-1 300.0	5 ● E4-1 ● E1-1 ● E3-1 ● E2-1 300.0
10 ● J2-1 ● J1-1 ● J3-1 ● J4-1 300.0	9 ● I4-1 ● I1-1 ● I2-1 ● I3-1 300.0	8 ● H1-1 ● H2-1 ● H4-1 ● H3-1 300.0	7 ● G3-1 ● G2-1 ● G1-1 ● G4-1 300.0	6 ● F2-1 ● F1-1 ● F4-1 ● F3-1 300.0

Şekil 5.15. Parsel sayısı 20, NLP değeri 4 olan probleme ait en iyi çözüme ait bir örnek

5.3.2. Zaman analizi

Çizelge 5.12'de parsel sayısı= 20 için algoritmaların zaman analizi verilmektedir. Tüm zaman değerleri saniye (s) cinsinden verilmektedir. Algoritmaların sonuçları zaman açısından incelendiğinde, ABC algoritması hem zaman performansı hem de kararlılık açısından sırasıyla 1.50 ve 1.75 ortalama sıra değeri ile en başarılı algoritmadır. GA, 2.25'lik bir performansla 2. algoritmadır. GA'yı sırasıyla *ortalama sıra* sıralaması açısından PSO, DE ve TSA takip etmiştir.

Çizelge 5.12. Parsel sayısı 20 için zaman açısından algoritmalara ait karşılaştırmalı sonuçlar

Parsel başına düşen Malik sayısı	ABC		DE		GA		PSO		TSA	
	Min ¹	Maks ²	Min	Mak	Min	Maks	Min	Maks	Min	Maks
	Ort ³	SS ⁴	Ort	SS	Ort	SS	Ort	SS	Ort	SS
	Sıra(Ort) ⁵	Sıra(SS) ⁶	Sıra(Ort)	Sıra(SS)	Sıra(Ort)	Sıra(SS)	Sıra(Ort)	Sıra(SS)	Sıra(Ort)	Sıra(SS)
1	0.46 ¹	0.85 ²	2.22	4.55	0.63	17.40	0.66	3.70	2.86	4.97
	0.65 ⁴	0.10 ⁴	3.62	0.62	5.62	7.19	1.49	0.76	4.02	0.67
	1 ⁵	1 ⁶	3	2	5	5	2	4	4	3
2	5.20	8.96	20.47	23.07	4.30	9.44	5.45	20.36	38.09	44.25
	6.30	0.80	21.93	0.72	6.32	1.36	11.80	4.71	41.74	1.39
	1	2	4	1	2	3	3	5	5	4
3	31.82	34.26	53.02	60.12	10.93	20.22	50.47	77.38	61.12	73.34
	32.40	0.55	57.79	2.28	14.74	2.59	66.80	5.85	64.65	3.43
	2	1	3	2	1	3	5	5	4	4
4	21.18	32.92	76.92	90.52	15.57	39.69	64.34	66.24	57.32	64.92
	29.42	3.46	81.19	3.77	27.85	5.92	65.10	0.45	62.58	2.50
	2	3	5	4	1	5	4	1	3	2
Ortalama Sıra	1.50	1.75	3.75	2.25	2.25	4.00	3.50	3.75	4.00	3.25
Final Sırası	1	1	4	2	2	5	3	4	5	3
En İyi Sıra Sayısı	2	2	0	1	2	0	0	1	0	0
En İyi Sıra										
Yüzdesi (%)	50	50	0	25	50	0	0	25	0	0

Çizelge 5.13, parsel sayısı = 40 için algoritmaların zaman analizini sunmaktadır. GA, zaman açısından en iyi performansı verirken, DE en kararlı algoritma olmuştur. ABC, ortalama sıra değeri 2.00 değeri ile zaman performansı açısından GA'yı takip etmiştir. Diğer taraftan, DE ve PSO performans açısından aynı değere sahip olmuştur. Son olarak TSA, bu problem setindeki en başarısız algoritma olmuştur.

Çizelge 5.13. Parsel sayısı 40 için zaman açısından algoritmalara ait karşılaştırmalı sonuçlar

<i>Parsel başına düşen Malik sayısı</i>	ABC		DE		GA		PSO		TSA	
	<i>Min</i>	<i>Maks</i>	<i>Min</i>	<i>Maks</i>	<i>Min</i>	<i>Maks</i>	<i>Min</i>	<i>Maks</i>	<i>Min</i>	<i>Maks</i>
	<i>Ort</i>	<i>SS</i>	<i>Ort</i>	<i>SS</i>	<i>Ort</i>	<i>SS</i>	<i>Ort</i>	<i>SS</i>	<i>Ort</i>	<i>SS</i>
	<i>Sıra(Ort)</i>	<i>Sıra(SS)</i>	<i>Sıra(Ort)</i>	<i>Sıra(SS)</i>	<i>Sıra(Ort)</i>	<i>Sıra(SS)</i>	<i>Sıra(Ort)</i>	<i>Sıra(SS)</i>	<i>Sıra(Ort)</i>	<i>Sıra(SS)</i>
1	5.24	16.49	28.05	40.91	5.61	56.42	16.12	110.99	55.32	97.23
	8.52	3.14	34.39	2.98	32.18	20.03	50.73	32.77	70.19	13.13
	1	2	3	1	2	4	4	5	5	3
2	62.32	105.54	106.92	109.37	26.62	54.08	107.24	110.12	119.13	122.28
	73.98	10.04	108.37	0.71	45.39	7.44	108.97	0.71	120.76	0.99
	2	5	3	1	1	4	4	2	5	3
3	80.30	189.01	133.03	147.11	61.93	63.94	140.11	168.95	170.02	201.53
	110.55	28.28	140.24	2.52	62.90	0.60	154.30	8.97	183.10	9.48
	2	5	3	2	1	1	4	3	5	4
4	125.73	214.24	185.71	219.87	73.03	80.09	145.47	151.43	166.08	171.97
	158.67	24.03	200.36	9.78	74.45	1.51	147.33	1.48	168.44	1.47
	3	5	5	4	1	3	2	2	4	1
<i>Ortalama Sıra</i>	2.00	4.25	3.50	2.00	1.25	3.00	3.50	3.00	4.75	2.75
<i>Final Sırası</i>	2	4	3	1	1	3	3	3	4	2
<i>En İyi Sıra Sayısı</i>	1	0	0	2	3	1	0	0	0	1
<i>En İyi Sıra</i>										
<i>Yüzdesi (%)</i>	25	0	0	50	75	25	0	0	0	25

Çizelge 5.14'teki sonuçlar incelendiğinde, ABC hem ortalama sıra hem de en iyi sıra sayısı açısından en etkili ve kararlı algoritmadır. Ayrıca GA, ABC'ye en yakın sonucu elde etmektedir. Bu problem setinde DE algoritması performans ve kararlılık açısından ABC ve GA'dan sonra gelirken, TSA ve PSO benzer sonuçlara sahiptir.

Çizelge 5.14. Parsel sayısı 60 için zaman açısından algoritmalara ait karşılaştırmalı sonuçlar

<i>Parsel başına düşen Malik sayısı</i>	ABC		DE		GA		PSO		TSA	
	<i>Min</i>	<i>Maks</i>	<i>Min</i>	<i>Maks</i>	<i>Min</i>	<i>Maks</i>	<i>Min</i>	<i>Maks</i>	<i>Min</i>	<i>Maks</i>
	<i>Ort</i>	<i>SS</i>	<i>Ort</i>	<i>SS</i>	<i>Ort</i>	<i>SS</i>	<i>Ort</i>	<i>SS</i>	<i>Ort</i>	<i>SS</i>
	<i>Sıra(Ort)</i>	<i>Sıra(SS)</i>	<i>Sıra(Ort)</i>	<i>Sıra(SS)</i>	<i>Sıra(Ort)</i>	<i>Sıra(SS)</i>	<i>Sıra(Ort)</i>	<i>Sıra(SS)</i>	<i>Sıra(Ort)</i>	<i>Sıra(SS)</i>
1	19.41	35.64	145.44	183.04	21.41	76.23	151.68	188.32	181.68	217.74
	26.96	5.18	164.27	10.64	58.01	20.57	169.96	10.76	197.11	11.67
	1	1	3	2	2	5	4	3	5	4
2	79.06	85.70	188.79	246.95	84.63	88.45	189.99	246.44	233.13	242.77
	80.96	1.61	197.79	14.17	85.92	0.87	200.71	14.21	236.48	2.25
	1	2	3	4	2	1	4	5	5	3
3	49.56	58.47	214.78	264.38	98.28	100.39	207.77	274.45	277.64	362.32
	55.99	1.71	228.73	15.86	99.44	0.61	243.09	18.02	307.57	23.76
	1	2	3	3	2	1	4	4	5	5
4	32.14	33.51	280.48	343.89	111.82	117.08	246.15	257.56	335.33	342.41
	32.75	0.37	305.65	19.84	113.29	1.18	250.39	3.20	339.08	2.05
	1	1	4	5	2	2	3	4	5	3
<i>Ortalama Sıra</i>	1.00	1.50	3.25	3.50	2.00	2.25	3.75	4.00	5.00	3.75
<i>Final Sırası</i>	1	1	3	3	2	2	4	5	5	4
<i>En İyi Sıra Sayısı</i>	4	2	0	0	0	2	0	0	0	0
<i>En İyi Sıra</i>										
<i>Yüzdesi (%)</i>	100	50	0	0	0	50	0	0	0	0

Çizelge 5.15'teki sonuçlar, GA'nın 1.25 ortalama sıra değerine ve %75'lik en iyi sıralama değerine göre en iyi algoritma olduğunu göstermektedir. Performans olarak ABC, GA'nın ardından 1.75 değeriyle 2. sırada yer almaktadır. Ayrıca, kararlılık açısından TSA, 2.50 değerine sahip ikinci algoritma olmuştur. Son olarak, DE ve PSO bu problem seti için benzer sonuçlar üretmiştir.

Çizelge 5.15. Parsel sayısı 80 için zaman açısından algoritmalara ait karşılaştırmalı sonuçlar

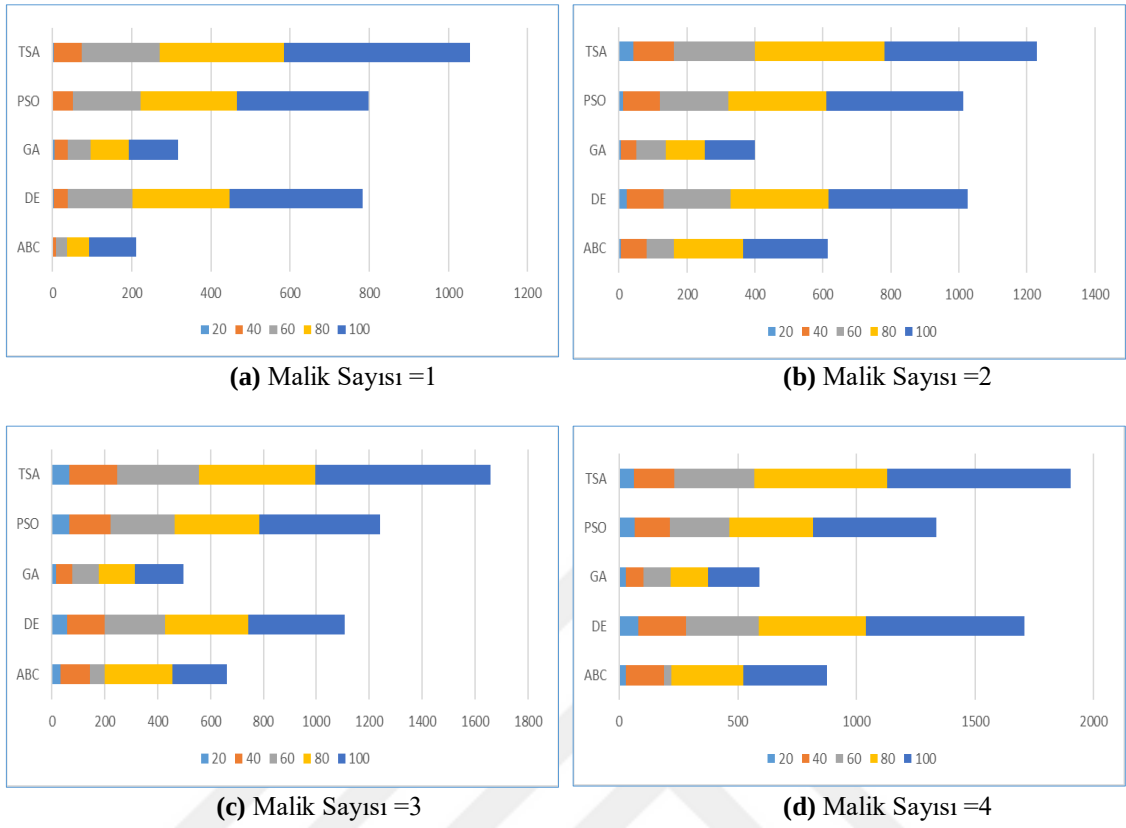
<i>Parsel başına düşen Malik sayısı</i>	ABC		DE		GA		PSO		TSA	
	<i>Min</i>	<i>Maks</i>	<i>Min</i>	<i>Maks</i>	<i>Min</i>	<i>Maks</i>	<i>Min</i>	<i>Maks</i>	<i>Min</i>	<i>Maks</i>
	<i>Ort</i>	<i>SS</i>	<i>Ort</i>	<i>SS</i>	<i>Ort</i>	<i>SS</i>	<i>Ort</i>	<i>SS</i>	<i>Ort</i>	<i>SS</i>
	<i>Sıra(Ort)</i>	<i>Sıra(SS)</i>	<i>Sıra(Ort)</i>	<i>Sıra(SS)</i>	<i>Sıra(Ort)</i>	<i>Sıra(SS)</i>	<i>Sıra(Ort)</i>	<i>Sıra(SS)</i>	<i>Sıra(Ort)</i>	<i>Sıra(SS)</i>
1	38.83	67.51	233.72	271.85	67.74	115.17	231.72	271.75	309.36	334.62
	56.49	10.12	244.64	13.62	97.20	10.01	242.77	12.56	313.26	5.21
	1	3	4	5	2	2	3	4	5	1
2	133.07	319.45	285.79	294.34	113.49	117.05	284.94	293.96	344.38	429.21
	203.14	46.56	288.79	1.97	114.49	0.84	289.40	2.75	382.64	24.37
	2	5	3	2	1	1	4	3	5	4
3	168.32	406.79	254.69	373.99	133.64	148.34	308.20	341.18	436.75	453.40
	256.33	76.67	315.40	37.74	136.66	4.30	319.09	6.75	442.04	4.06
	2	5	3	4	1	2	4	3	5	1
4	191.85	489.27	448.51	464.18	154.53	170.96	344.02	367.21	511.01	623.98
	301.13	96.79	452.70	4.15	160.04	5.81	354.05	6.50	559.07	30.72
	2	5	4	1	1	2	3	3	5	4
<i>Ortalama Sıra</i>	1.75	4.50	3.50	3.00	1.25	1.75	3.50	3.25	5.00	2.50
<i>Final Sırası</i>	2	5	3	3	1	1	3	4	4	2
<i>En İyi Sıra Sayısı</i>	1	0	0	1	3	1	0	0	0	2
<i>En İyi Sıra</i>										
<i>Yüzdesi (%)</i>	25	0	0	25	75	25	0	0	0	50

Çizelge 5.16'ya göre GA, zaman performansı açısından ortalama ve en iyi sıra değerlerine göre en iyi algoritma olarak açıkça görülmektedir. ABC, 1.75 ortalama sıra değeriyle GA'dan sonra en iyi algoritma olmuştur. Kararlılık açısından ise DE 2.00 ortalama sıra değeri ile en başarılı teknik olarak gözükmemektedir. Performans açısından, GA ve ABC'den sonra sırasıyla PSO, DE ve TSA algoritmaları sıralanmıştır.

Çizelge 5.16. Parsel sayısı 100 için zaman açısından algoritmalara ait karşılaştırmalı sonuçlar

<i>Parsel başına düşen Malik sayısı</i>	ABC		DE		GA		PSO		TSA	
	<i>Min</i>	<i>Maks</i>	<i>Min</i>	<i>Maks</i>	<i>Min</i>	<i>Maks</i>	<i>Min</i>	<i>Maks</i>	<i>Min</i>	<i>Maks</i>
	<i>Ort</i>	<i>SS</i>	<i>Ort</i>	<i>SS</i>	<i>Ort</i>	<i>SS</i>	<i>Ort</i>	<i>SS</i>	<i>Ort</i>	<i>SS</i>
	<i>Sıra(Ort)</i>	<i>Sıra(SS)</i>	<i>Sıra(Ort)</i>	<i>Sıra(SS)</i>	<i>Sıra(Ort)</i>	<i>Sıra(SS)</i>	<i>Sıra(Ort)</i>	<i>Sıra(SS)</i>	<i>Sıra(Ort)</i>	<i>Sıra(SS)</i>
1	99.54	127.91	333.10	339.60	115.75	136.10	327.34	339.69	458.97	564.04
	118.63	5.63	335.76	1.66	124.89	7.15	333.59	3.57	469.03	22.89
	1	3	4	1	2	4	3	2	5	5
2	168.56	400.55	361.68	456.94	146.30	150.93	355.98	447.57	369.90	548.93
	250.65	73.82	409.05	22.74	147.97	0.97	403.08	23.10	448.96	45.91
	2	5	4	2	1	1	3	3	5	4
3	198.23	224.17	361.44	366.90	176.99	222.42	443.84	513.81	650.96	699.09
	205.37	8.12	364.79	1.56	182.82	12.28	457.01	14.95	662.12	10.68
	2	2	3	1	1	4	4	5	5	3
4	220.83	712.25	661.72	728.62	203.92	235.11	502.10	551.46	766.29	779.08
	354.00	140.22	668.51	14.11	215.28	9.45	519.74	12.93	773.48	3.09
	2	5	4	4	1	2	3	3	5	1
<i>Ortalama Sıra</i>	1.75	3.75	3.75	2.00	1.25	2.75	3.25	3.25	5.00	3.25
<i>Final Sırası</i>	2	4	4	1	1	2	3	3	5	3
<i>En İyi Sıra Sayısı</i>	1	0	0	2	3	1	0	0	0	1
<i>En İyi Sıra</i>										
<i>Yüzdesi (%)</i>	25	0	0	50	75	25	0	0	0	25

Parsel başına düşen Malik sayısı (*NLP*) açısından ortalama değerlere göre karşılaştırmalı zaman analizi Şekil 5.16'da verilmektedir. Tüm sonuçlar saniye cinsinden verilmektedir. Bu sonuçlara göre *NLP* 1 iken ABC en hızlı algoritma olmuştur. GA, ABC'ye yakın sonuçlar üreterek ikinci sırada yer almıştır. DE ve PSO çok benzer zaman sonuçları gösterirken, TSA bu problem seti için en yavaş yöntemdir. *NLP* 2 olduğunda, bu kez GA en iyi algoritma olup onu ABC izlemiştir. Diğer taraftan, DE ve PSO benzer sonuçlar elde ederken, TSA yine en etkisiz yöntem olmuştur. GA, *NLP* değeri 3 olduğunda en hızlı yöntem olup ABC ise GA'dan sonra en etkili algoritma olmuştur. DE, hız açısından bu problem setinde PSO'dan daha etkili olurken, TSA son sırada yer almıştır. Son olarak, *NLP* 4 iken, GA'nın başarısı diğer *NLP* durumlarına kıyasla ABC'den çok daha üstün olmuştur. Bu problem setinde PSO, DE'den daha başarılı iken, TSA yine en yavaş algoritma olmuştur. Sonuçlar genel olarak incelendiğinde *NLP*'deki artışa rağmen GA'nın zaman açısından problem çözümü için harcadığı sürenin çok fazla artmadığı gözlemlenmiştir. Bu durum, LR'deki büyük ölçekli problemler için GA'nın diğer algoritmalara göre en iyi çözüme daha hızlı bir şekilde gidebileceğini göstermektedir.



Şekil 5.16. Ortalama değerlere göre karşılaştırmalı zaman analizi

5.3.3. Genel analiz sonuçları

Tablolardaki karşılaştırmalı sonuçlara göre GA, çözüm kalitesi açısından diğer yöntemlere göre çok daha başarılı performans göstermiştir. Bununla birlikte GA, problem büyüklüğündeki artışa bağlı olarak diğer yöntemlerle performans farkını kademeli olarak artırmıştır. Ayrıca, *NLP* sayısındaki artış, GA'yı diğer algoritmalarından daha üstün kılmıştır. Çünkü *NLP* 1 olduğunda ve problem boyutu 20, 40 ve 60 gibi daha küçük olduğunda ABC, GA'dan daha başarılı sonuçlar elde etmiştir. Fakat *NLP*'nin 2, 3 ve 4 olduğu tüm problem boyutlarında, GA her zaman en iyi algoritma olmuştur. GA, *Malikleri yer değiştir* ve *Malik ekle* adlı iki mutasyon operatörünü diğer algoritmalara göre daha başarılı bir şekilde kullandığından, *NLP*'nin 1'den büyük olduğu tüm durumlarda GA'nın en iyi çözüme en yakın sonuçları elde ettiği görülmüştür. Diğer yandan PSO algoritması GA ve ABC'den sonra en iyi başarıyı göstermiştir. Son olarak, TSA ve DE algoritmaları bu veri kümesi için daha az başarı göstermiştir.

Algoritmalar kararlılık açısından kıyaslandığında, GA'nın aynı şekilde veri kümesindeki çoğu problem için en kararlı algoritma olduğu açıkça görülmektedir. Buna ek olarak *NLP* değerinin 1 olduğu tüm problem boyutlarında, ABC yöntemi en kararlı algoritma olarak görülmektedir. Parsel sayısının 100 olduğu tüm problemlerde ise *NLP* değeri ne olursa olsun, GA en güçlü yöntem olarak görülmektedir. Zaman karşılaştırmasına gelince, zaman tablolarına göre, *NLP*'nin 1 olması durumunda ABC algoritması en hızlı algoritma iken GA ise *NLP*'nin 1'den büyük olduğu tüm problem boyutlarında en hızlı algoritma olarak görülmektedir.

Performans tablolarına dayanarak, ABC ve GA'nın LR problemlerini çözmede en iyi algoritmalar olduğu açıkça belirtilebilir. ABC ve GA algoritmalarının diğer algoritmalarından üstün olmasının nedeni şu şekilde açıklanabilir: PSO ve TSA gibi algoritmalarda yeni aday çözümler üretme sürecinde en iyi bireyler (parçacık veya ağaç) kullanılır. Bu durum çözümlerin yerel en iyiye sıkışıp kalmasına sebep olabilir. Bu veri kümesi için, TSA ve PSO, deneysel sonuçlara göre yerel en iyiye takıldığı için daha düşük performansa sahip olma eğiliminde olmuşlardır. Öte yandan DE, her jenerasyonda rastgele üç farklı kromozom kullanarak yeni kromozomlar üretmektedir. Bu durum aynı zamanda gelecek nesillerde daha etkili kromozomların üretilmesini de engelleyebilmektedir. Çünkü DE rastgele seçilen kromozomlar yerine daha güçlü kromozomlar kullanmamaktadır. Bu nedenle DE, bu çalışmada çok başarılı sonuçlar üretememiştir. Diğer taraftan, ABC ve GA'da kullanılan turnuva yöntemi sayesinde, elit bireyler alt kümesinden seçilen kaliteli çözümler, gelecek nesil için daha güçlü bireylerin ortaya çıkmasını sağladığı deneysel sonuçlara göre söylenebilir. Ayrıca GA'nın seçim operatöründe her iki kromozomun da turnuva yöntemi ile seçilmesinin onu ABC'den de daha başarılı sonuçlar üretmesine olanak sağladığı söylenebilir. Sonuç olarak GA, çözüm kalitesi, hızı ve kararlılığı açısından ABC, DE, PSO ve TSA yöntemlerinden çok daha başarılı sonuçlar elde etmiştir.

Bu çalışmada, LR problemlerini çözmek için üç farklı mutasyon operatörü önerilmiştir. Bu operatörler *NLP* değerine göre çalıştırılmıştır. *NLP* 1 ise, yalnızca *Parselleri yer değiştir* operatörü Şekil 4.27'deki koşullar altında çalıştırılmış, *NLP* 1'den büyükse, *Malikleri yer değiştir* ve *Malik ekle* operatörleri dâhil edilerek, rastgele koşullara göre üç mutasyon operatörü çalıştırılmıştır. Sonuçlar incelendiğinde, GA'nın bu iki operatörü etkin bir şekilde kullanması algoritmanın bir avantajı olarak söylenebilir.

Çünkü *NLP*'nin 2, 3 ve 4 olduğu tüm problemlerde GA en başarılı algoritma olarak görülmektedir. Bununla birlikte, *NLP* 1 olduğunda GA, bu durumda ABC kadar önemli bir başarı gösterememiştir. Bu durum GA için bir dezavantaj şeklinde ifade edilebilir. Diğer bir deyişle ABC, 20-1, 40-1, 60-1 ve 80-1 gibi problemlerde GA'dan daha iyi bir performans elde etmiştir. Sonuç olarak, düşük karmaşıklığa sahip problemlerde ABC'nin GA'ya göre daha başarılı olduğu söylenebilir. ABC'ye ek olarak, bu başarı DE, PSO ve TSA için de benzer şekildedir. Bu algoritmalar, 20-1 probleminde çok etkili sonuçlar elde ederek GA'dan üstün olduklarını göstermişlerdir. Bu sonuçlar, tüm algoritmaların GA'ya göre *Parsel yer değiştir* operatörünü kullanmada daha yetenekli olduğunu göstermiştir.

Çizelge 5.17. Sıra değerlerine göre genel performans analizi

<i>Parsel Sayısı</i>	<i>Sıra</i>	ABC		DE		GA		PSO		TSA	
		<i>Ort</i>	<i>SS</i>	<i>Ort</i>	<i>SS</i>	<i>Ort</i>	<i>SS</i>	<i>Ort</i>	<i>SS</i>	<i>Ort</i>	<i>SS</i>
20	<i>Ortalama Sıra</i>	2.00	3.00	2.50	2.00	1.25	1.25	1.50	1.50	3.25	2.75
	<i>En İyi Sıra Sayısı</i>	2	2	2	2	3	3	2	2	1	1
	<i>Ortalama Sıra</i>	2.00	2.50	3.25	2.25	1.75	1.75	2.75	3.75	4.25	3.75
40	<i>En İyi Sıra Sayısı</i>	1	1	1	1	3	3	0	0	0	0
	<i>Ortalama Sıra</i>	1.75	2.50	4.00	3.50	1.25	1.50	3.25	4.00	4.75	3.50
	<i>En İyi Sıra Sayısı</i>	1	1	0	0	3	3	0	0	0	0
60	<i>Ortalama Sıra</i>	1.75	2.00	4.25	3.25	1.25	1.25	3.25	4.25	4.50	4.25
	<i>En İyi Sıra Sayısı</i>	1	1	0	0	3	3	0	0	0	0
	<i>Ortalama Sıra</i>	2.00	3.25	4.50	3.00	1.00	1.00	3.50	4.50	4.00	3.25
80	<i>En İyi Sıra Sayısı</i>	0	0	0	0	4	4	0	0	0	0
	<i>Ortalama Sıra</i>	1.90	2.45	3.80	2.90	1.30	1.35	2.80	3.60	3.80	3.60
	<i>Final Sırası</i>	2	2	4	3	1	1	3	4	4	4
100	<i>Toplam En İyi Sıra Sayısı</i>	5	5	3	3	16	16	3	2	2	1
	<i>Toplam En İyi Sıra Sayısı</i>										
	<i>Yüzdesi (%)</i>	25	25	15	15	80	80	15	10	10	5

Çizelge 5.18. Sıra değerlerine göre genel performans analizi

Parsel Sayısı	Sıra	ABC		DE		GA		PSO		TSA	
		Ort	SS	Ort	SS	Ort	SS	Ort	SS	Ort	SS
20	Ortalama										
	Sıra	1.50	1.75	3.75	2.25	2.25	4.00	3.50	3.75	4.00	3.25
	En İyi Sıra Sayısı	2	2	0	1	2	0	0	1	0	0
40	Ortalama										
	Sıra	2.00	4.25	3.50	2.00	1.25	3.00	3.50	3.00	4.75	2.75
	En İyi Sıra Sayısı	1	0	0	2	3	1	0	0	0	1
60	Ortalama										
	Sıra	1.00	1.50	3.25	3.50	2.00	2.25	3.75	4.00	5.00	3.75
	En İyi Sıra Sayısı	4	2	0	0	0	2	0	0	0	0
80	Ortalama										
	Sıra	1.75	4.50	3.50	3.00	1.25	1.75	3.50	3.25	5.00	2.50
	En İyi Sıra Sayısı	1	0	0	1	3	1	0	0	0	2
100	Ortalama										
	Sıra	1.75	3.75	3.75	2.00	1.25	2.75	3.25	3.25	5.00	3.25
	En İyi Sıra Sayısı	1	0	0	2	3	1	0	0	0	1
	Genel Ortalama										
	Sıra	1.60	3.15	3.55	2.55	1.60	2.75	3.50	3.45	4.75	3.10
	Final Sırası	1	4	3	1	1	2	2	5	4	3
	Toplam										
	En İyi Sıra Sayısı	9	4	0	6	11	5	0	1	0	4
	Toplam										
	En İyi Sıra Sayısı										
	Yüzdesi (%)	45	20	0	30	55	25	10	10	5	25

Çizelge 5.17, sıra değerlerine göre genel performans analizini göstermektedir. Bu sonuçlara göre GA, 20 problemin 16'sında hem performans hem de kararlılık açısından en başarılı algoritma olmuştur. GA algoritmasını başarı sırasına göre ABC, DE, PSO ve TSA takip etmiştir. Ancak kararlılık açısından algoritmalar ABC, PSO, DE ve TSA olarak sıralanmıştır. Ek olarak, Çizelge 5.18, karşılaştırmalı olarak sıra değerlerine göre genel zaman analizini sunmaktadır. En iyi sıralama açısından GA, 20 problemin 11'inde en başarılı olarak %55 ile en hızlı algoritma olarak görülmektedir. ABC, GA'yı %45'lik en iyi sıra yüzdesi ile takip etmiştir. Kararlılık açısından ise ABC ve GA arasında büyük bir fark görülmemiştir. GA %25 ile en kararlı algoritma iken, ABC en iyi sıra yüzdesine göre %20 değeriyle 2. en kararlı algoritma olmuştur. Bununla birlikte, genel ortalama sıra değerlerine göre, ABC ve GA algoritması 1.60 (*Sıra Ortalama*) değeri ile her ikisi de

birinci sırada yer almıştır. Kararlılık açısından ise, DE 2.55 değeri ile ilk algoritma iken, GA 2.75 değerli ikinci algoritma olmuştur.

Çizelge 5.19. Friedman testine göre istatistiksel sonuçlar

<i>Problem</i>	<i>ABC</i>	<i>DE</i>	<i>GA</i>	<i>PSO</i>	<i>TSA</i>
<i>20-1</i>	1—(0.000)	1—(0.000)	2—(0.090)	1—(0.000)	1—(0.000)
<i>20-2</i>	1—(0.000)	1—(0.000)	1—(0.000)	1—(0.000)	2—(0.597)
<i>20-3</i>	3—(1.752)	4—(4.743)	1—(0.000)	2—(0.238)	5—(8.347)
<i>20-4</i>	3—(8.182)	4—(13.181)	1—(0.008)	2—(2.226)	5—(14.258)
<i>40-1</i>	1—(0.000)	1—(0.000)	4—(0.353)	3—(0.078)	2—(0.046)
<i>40-2</i>	2—(0.418)	4—(9.841)	1—(0.054)	3—(1.706)	5—(12.868)
<i>40-3</i>	3—(15.603)	4—(26.060)	1—(1.127)	2—(12.834)	5—(33.988)
<i>40-4</i>	2—(28.542)	4—(48.402)	1—(3.011)	3—(36.414)	5—(48.680)
<i>60-1</i>	1—(0.000)	4—(4.403)	2—(0.500)	3—(1.442)	5—(12.307)
<i>60-2</i>	2—(9.079)	5—(34.975)	1—(1.471)	3—(10.378)	4—(32.720)
<i>60-3</i>	2—(35.363)	3—(46.022)	1—(5.254)	4—(49.251)	5—(73.948)
<i>60-4</i>	2—(54.411)	4—(100.202)	1—(10.537)	3—(88.064)	5—(100.720)
<i>80-1</i>	1—(0.070)	4—(19.819)	2—(0.645)	3—(7.975)	5—(24.956)
<i>80-2</i>	2—(19.615)	5—(73.572)	1—(4.756)	3—(30.221)	4—(60.916)
<i>80-3</i>	2—(52.052)	3—(79.357)	1—(11.951)	4—(104.122)	5—(126.474)
<i>80-4</i>	2—(75.823)	5—(170.725)	1—(21.457)	3—(155.822)	4—(170.008)
<i>100-1</i>	2—(2.126)	5—(43.534)	1—(1.129)	3—(19.653)	4—(42.533)
<i>100-2</i>	2—(33.507)	5—(121.251)	1—(9.050)	4—(62.367)	3—(53.149)
<i>100-3</i>	2—(70.700)	3—(121.436)	1—(21.313)	4—(169.567)	5—(192.564)
<i>100-4</i>	2—(99.484)	5—(260.898)	1—(16.472)	3—(241.993)	4—(259.565)
Ortalama Sıra #	2.075	3.875	1.575	3.050	4.425
<i># ki kare testi = 48 df = 4 p-değeri = 9.4378e-10 (sıfır hipotezinin reddi)</i>					

Çizelge 5.19'da algoritmaların Friedman testi ile elde ettiği başarıya göre ortalama sıralamalar belirlenmiştir. Daha sonra Çizelge 5.20'de en başarılı algoritma ile diğer algoritmalar arasında anlamlı ve net bir fark olup olmadığını belirlemek için Wilcoxon işaret testi kullanılmıştır. Bunun için Wilcoxon testi diğer algoritmalar arasında bağımsız olarak uygulanmış ve her problem için Friedman testi ile belirlenen en iyi algoritma karşılaştırılmıştır.

Çizelge 5.19'un her hücresinde verilen tamsayı değerleri, tüm deneysel sonuçlara göre algoritmalar tarafından belirlenen nihai başarı sıralamalarıdır. Tabloda parantez içinde verilen ondalık değerler verilen problemler için algoritmalarından elde edilen 20 bağımsız çalışmanın ortalama değerleridir. Sonuçlara göre tüm algoritmalar 1'den 5'e kadar sıralanmaktadır. Örneğin son satırdaki ortalama 1.575 başarı sıralaması olan GA algoritmasının genel başarı sıralaması 1'dir. Böylece tüm algoritmaların başarı sıralaması listelenmiştir. Algoritmalar başarı sıralaması en yüksekten en düşüğe şu şekildedir: GA,

ABC, PSO, DE ve TSA. Tüm bu sonuçların önemi çok düşük ve geçerli bir $9.4378E-10$ p değeri ile doğrulanmıştır.

Çizelge 5.20. Wilcoxon işaret sayıları testi ile istatistiksel sonuçlar

GA vs	DE p (s)	ABC p (s)	PSO p (s)	TSA p (s)
20-1	0.94E-4 (+)	0.94E-4 (+)	0.94E-4 (+)	0.94E-4 (+)
20-2	# NaN (-)	# NaN (-)	# NaN (-)	9.43E-8 (+)
20-3	8.01E-12 (+)	8.01E-12 (+)	1.08E-8 (+)	8.01E-12 (+)
20-4	0.01E-8 (+)	0.01E-8 (+)	0.01E-8 (+)	0.01E-8 (+)
40-1	6.61E-8 (+)	6.61E-8 (+)	0.89E-4 (+)	0.16E-4 (+)
40-2	0.02E-8 (+)	1.26E-8 (+)	0.02E-8 (+)	0.02E-8 (+)
40-3	0.07E-8 (+)	0.07E-8 (+)	0.07E-8 (+)	0.07E-8 (+)
40-4	0.07E-8 (+)	0.07E-8 (+)	0.07E-8 (+)	0.07E-8 (+)
60-1	0.06E-8 (+)	2.55E-8 (+)	1.041E-8 (+)	0.06E-8 (+)
60-2	0.07E-8 (+)	0.07E-8 (+)	0.07E-8 (+)	0.07E-8 (+)
60-3	0.07E-8 (+)	0.07E-8 (+)	0.07E-8 (+)	0.07E-8 (+)
60-4	0.07E-8 (+)	0.07E-8 (+)	0.07E-8 (+)	0.07E-8 (+)
80-1	0.07E-8 (+)	4.72E-8 (+)	0.07E-8 (+)	0.07E-8 (+)
80-2	0.07E-8 (+)	0.07E-8 (+)	0.07E-8 (+)	0.07E-8 (+)
80-3	0.07E-8 (+)	0.07E-8 (+)	0.07E-8 (+)	0.07E-8 (+)
80-4	0.07E-8 (+)	0.07E-8 (+)	0.07E-8 (+)	0.07E-8 (+)
100-1	0.07E-8 (+)	0.03E-4 (+)	0.07E-8 (+)	0.07E-8 (+)
100-2	0.07E-8 (+)	0.07E-8 (+)	0.07E-8 (+)	0.07E-8 (+)
100-3	0.07E-8 (+)	0.07E-8 (+)	0.07E-8 (+)	0.07E-8 (+)
100-4	0.07E-8 (+)	0.07E-8 (+)	0.07E-8 (+)	0.07E-8 (+)

NaN (sayı değil): Veri setinde gerçekleştirilen testler için karşılaştırılabilir bir durum söz konusu değildir.

Yukarıdaki Friedman test sonucuna göre en başarılı sıralamaya sahip olan GA yaklaşımının diğer algoritmalarla karşılaştırılması Çizelge 5.20'de Wilcoxon istatistiksel anlamlılık testine göre sunulmaktadır. Çizelge 5.20'de verilen Wilcoxon istatistiksel test sonuçlarında, p-değer 0.01 alınmıştır ve bu değer eşik değerini göstermektedir. P değerinden (p) daha yüksek sonuçlar için, anlam(lar) '+' işaretiyle belirtilir. Buradaki eşik değeri, sonuçlar arasındaki istatistiksel olarak önemli farkın %99 (yüzde doksan dokuz) olasılık güven aralığında olduğunu göstermektedir. Başka bir deyişle, sonuçlar %1 anlamlılık düzeyinde değerlendirilmektedir. Ayrıca, algoritmaların sonuçları arasında istatistiksel bir fark ($p \geq 0.01$) olmadığı gerçeği Çizelge 5.20'de '-' işareti ile gösterilmektedir. Bu sonuçlara göre GA, ikinci problem (20-2) dışındaki tüm problemlerde önemli bir farka sahip olmuştur. Bu problemde GA ile TSA arasında önemli bir fark olduğu anlaşılmaktadır. Böylelikle GA yaklaşımının başarı sıralaması testine göre en iyi algoritma olduğu belirlenmiş ve bu algoritmanın test sonuçları ile diğer

algoritmaların sonuçları arasında %99 olasılık güven aralığında istatistiksel olarak anlamlı farklılıklar olduğu sonucuna varılmıştır.

Klasik TSP problemi gibi ayrık problemlerden farklı olarak LR problemlerini çözmek için yapılan işlemlerin büyük ölçüde zaman maliyeti gerektirdiği söylenebilir. LR çözümünün ilk adımında, çaprazlama işlemi sonucunda maliklerin parselleri değişir (tahsis işlemi). Yeniden dağıtım (tahsis) sonuçları, mutasyon süreci sonucunda yeniden değiştirilir. Bu yeniden dağıtım sonuçlarına bağlı olarak her bir parselin alanı güncellenir, parselin tüm koordinatları (X, Y) tekrar hesaplanır ve ardından merkez noktası belirlenir. Bu işlemlerin tüm adalarda her bir parsel için gerçekleştiği düşünüldüğünde, bu işlemler zaman açısından ciddi bir maliyet oluşturmaktadır. Algoritmalarındaki çaprazlama işlem sayılarına bakıldığında, GA'da yalnızca bir çaprazlama işlemi görülmektedir. Diğer bir deyişle, hibrit yaklaşım, her kromozom için yalnızca bir kez kullanılır. ABC algoritmasında, hibrit yaklaşım hem işçi arı hem de gözcü aşamalarında iki kez kullanılır. ABC algoritmasının yapısı incelendiğinde popülasyonun yarısı işçi ve diğer yarısı gözcü arılardan oluşmaktadır. Bu nedenle, popülasyondaki birey sayısı, başlangıçta verilen birey sayısının yarısıdır. Bu çalışmada popülasyondaki birey sayısı 50 olarak alındığından GA, DE, PSO ve TSA'ya ait popülasyonda 50 birey bulunurken, ABC'ye ait popülasyonda 25 birey bulunmaktadır. Böylece her çalışmada MaxFEs sayıları eşitlenmiş ve algoritmalar adil bir şekilde karşılaştırılmıştır. Bu değerlere göre, GA hibrit yaklaşımı 50 kromozom için 50 kez kullanırken, ABC hibrit yaklaşımı toplam 50 kez, işçi arıda 25 kez ve gözcü arı aşamalarında 25 kez kullanmıştır. Bundan dolayı ABC ve GA'nın zaman açısından benzer bir performans gösterdiği söylenebilir. TSA'ya gelince, hibrit yaklaşım her tohum üretimi için bu algortmada iki kez kullanılmıştır. Bu nedenle, TSA'nın zaman maliyetinin diğer algoritmalarından daha fazla olması beklenen bir durumdur. Son olarak, PSO ve DE incelendiğinde, bu hibrit yaklaşımı her birey (parçacık veya kromozom) için 3 kez kullanılmıştır. Bu nedenle, PSO ve DE'nin zaman açısından benzer performansa sahip olmasının bir nedeninin bu olduğu söylenebilir. Tüm çaprazlama ve mutasyon operatörlerini içeren hibrit yaklaşımlar sayıları göz önüne alındığında, ABC ve GA, zaman açısından en başarılı algoritmalar olarak görülmektedir. Ardından, PSO ve DE, çalışma süresine göre GA ve ABC'den sonra sıralanmıştır. Bu sonuçlar, LR problemlerini çözmede hızlı bir algoritma tasarlarken, yeni aday çözümler elde etmek için kullanılan çaprazlama işlem sayısını en aza indirmek gerektiğini açıkça göstermektedir. Böylelikle

problemin çözümünün daha kısa sürede gerçekleştirilmesi beklenmektedir. Ayrıca performans sonuçlarından açıkça görülmektedir ki, çaprazlama operatör sayısının fazla olması çözüm kalitesine çok fazla katkı sağlamamakta hatta çözüm kalitesini düşürmektedir. Sonuç olarak LR problemleri için çok hızlı ve etkili bir yöntemin daha az çaprazlama işlemi kullanan bir algoritma olduğu sonucuna varılmıştır.



6. SONUÇLAR VE ÖNERİLER

6.1. Sonuçlar

Bu tez çalışması üç ana bölümden oluştuğu için sonuçlar bu şekilde sunulmaktadır.

Bu tez çalışmasında, 3194 sayılı imar kanunu ve örnek bir imar planı dikkate alınarak, Konya İli Meram İlçesine ait bir bölgede ön parselasyon işlemi otomatik olarak gerçekleştirilmiştir. Bu çalışma alanında parselasyon işlemleri, kenar uzunluğu, alan ve hem kenar uzunluğu hem de alan bilgileri dikkate alınarak üç farklı şekilde gerçekleştirilmiştir. Tüm parselasyon işlemleri için parsel alanı hesabında Gauss yöntemi kullanılmıştır. Ayrıca parsellerin sınırlarının ve alanlarının verimli bir şekilde hesaplanması için $PwEL$, PwA ve $PwELA$ yöntemlerinde ikili arama tekniği başarıyla kullanılmıştır. Deneysel sonuçlar parselasyon sürecinin çok hızlı ve başarılı olduğunu ve önerilen yöntemlerle elde edilen parselasyon planlarının uzman tarafından gerçekleştirilen plana çok benzediğini açıkça göstermiştir. Ayrıca, önerilen ön parselasyon yönteminin, oluşturulması gereken kriterleri karşılayacak ideal bir parselasyon planı için bir alt yapı sağladığı açıkça söylenebilir.

Bu tez çalışmasında gerçek bir proje sahası kullanılarak LR problemlerinin çözümünde ABC, DE, GA, PSO ve TSA algoritmaları kullanılmıştır. Algoritmaların parselasyon işlemini gerçekleştirebilmesi için bu tez çalışmasında önerilen ön parselasyon yöntemi kullanılmıştır. Parselasyon işlemi genellikle maliklerin alanına göre gerçekleştirildiğinden parsel oluşturulurken PwA yöntemi kullanılmıştır. LR uygulamalarının gerçekleştirilmesi için 18. Maddeye göre iki temel hedef belirlenmiştir. Ayrıca, problemin yapısına uygun olarak, CPC ve IPC operatörleri olarak adlandırılan iki farklı çaprazlama operatörü önerilmiştir. Özellikle IPC sayesinde algoritmalar çok etkili sonuçlar elde etmiştir. Elde edilen sonuçlar geliştirilen ES'ye doğrudan verilerek adalardaki taşma ve boşluk alanları tamamen kaldırılmıştır. Böylelikle ES destekli optimizasyon algoritmaları ile üretilen dağıtım ve parselasyon planları doğrudan gerçek dünyada kullanılabilir hale getirilmiştir. ES tarafından desteklenen algoritmalar ile elde edilen kesin çözümler, manuel olarak üretilen çözüm ile karşılaştırıldığında önerilen yöntemlerin çok etkili olduğu açıkça görülmüştür. Ayrıca, performans açısından, PSO'nun diğer algoritmalara göre çok daha başarılı olduğu açıkça görülmüştür. Zaman

açısından değerlendirildiğinde, bu proje alanı için bir uzman ortalama 3 günde manuel olarak dağıtım ve parselasyon işlemini tamamlarken, önerilen yöntemler 5 ila 15 dakika arasında oldukça kaliteli çözümler sunmuştur. Proje ölçeğinin büyümesi dikkate alındığında ise önerilen yöntemlerin zaman açısından katkısı çok daha net ortaya çıkması beklenmektedir.

Bu tez çalışmasında son olarak, sentetik bir veri seti üzerinde GA, ABC, PSO, DE ve TSA isimli optimizasyon algoritmaları LR problemlerini çözmek için kullanılmıştır. Bu algoritmalar, LR problemleri için önerilen harita tabanlı çaprazlama operatörü ve mutasyon operatörleri hibrit uygulanarak kullanılmıştır. Ayrıca, LR uygulamalarında test verisi olarak kullanılmak üzere sentetik bir veri seti oluşturulmuştur. Veri seti hazırlanırken, problemlerin en iyi çözümlerinin kolayca tespit edilebilmesi ve ispatlanabilmesi dikkate alınarak tasarlanmıştır. Ayrıca parsel sayısı ve malik sayısı farklı alınarak üretilen problemler sayesinde algoritmaların başarısının daha net bir şekilde karşılaştırılması amaçlanmıştır. Buna ek olarak, çözümlerin kalitesini belirlemek için yeni bir amaç fonksiyonu önerilmiştir. Önerilen amaç fonksiyonu, LR süreçlerinde ana hedefler olan her bir malikin kadastro ve imar parselleri arasındaki mesafeyi ve her bir parseldeki hisse sayısını en aza indirmeye dayandırılmıştır. Bununla birlikte problemin ideal çözümünün sonucunun 0 olduğu bilindiği için önerilen veri seti aracılığıyla amaç fonksiyona bakılmaksızın algoritmaların karşılaştırılmasının objektif olarak gerçekleştirildiği açıktır. Bundan dolayı bu veri seti, bu çalışmanın literatüre önemli bir katkısı olmuştur. Deneysel sonuçlar incelendiğinde ise GA, 20 problemin %80'inde en başarılı yöntem olmuş ve aynı zamanda genel sonuçlara göre en kararlı algoritma olmuştur. Ayrıca, problemin boyutu büyüdükçe ABC, PSO, TSA ve DE'nin performansları çok daha fazla etkilenirken, GA ise bu durumdan çok az etkilenmiştir. Ayrıca zaman açısından ABC algoritması GA'ya çok yakın sonuçlar elde etse de GA zaman olarak da en iyi başarıyı göstermiştir. Tüm bu sonuçlara dayanarak, GA algoritması, problem boyutundaki artışa rağmen diğer algoritmalara nazaran en kısa sürede en iyi değeri elde etmiştir. Bu sonuçlar, LR problemleri için GA'nın çok güçlü ve hızlı bir alternatif yöntem olabileceğini açıkça göstermektedir.

6.2. Öneriler

Bu tez çalışmasında ön parselasyon işlemleri için BS yöntemi kullanılmış ve çok kısa sürede etkili sonuçlar elde edilmiştir. Önerilen BS tabanlı yöntem parselasyon işleminin gerekli olduğu farklı imar uygulamalarında veya arazi toplulaştırmada gerekli parselasyon işlemlerinde de kullanılabilir.

İleride yapılacak çalışmalarda, farklı sürü zekası veya evrimsel tabanlı algoritmalar kullanılarak daha etkili çözümler elde edilebilir. Özellikle karınca kolonisi optimizasyon algoritması bu probleme uyarlanarak daha etkili sonuçlar elde edilebilir. Bu çalışmada önerilen parsel tabanlı çaprazlama operatörü arazi toplulaştırma problemi gibi mekânsal veri içeren farklı problemlere de uygulanarak etkili sonuçlar elde edilebilir. Ayrıca geliştirilecek yeni çaprazlama ve mutasyon operatörleri sayesinde daha kısa sürede daha iyi çözümler üretilebilir. Buna ilaveten, çok amaçlı optimizasyon teknikleri kullanılarak daha etkin çözümler elde edilebilir. Ayrıca problemin analitiği detaylı olarak incelenebilir ve problemin yapısal analizine dair geniş ölçekli bir çalışma gerçekleştirilebilir.

Bu tez çalışmasında LR uygulamaları için yeni bir sentetik veri seti önerilmiştir. Üretilen sentetik veri seti kullanılarak LR problemleri için geliştirilen farklı algoritmaların başarıları objektif bir şekilde değerlendirilebilir. Ayrıca sentetik veri seti farklı problemler eklenerek zenginleştirilebilir.

KAYNAKLAR

- Abdoun, O. ve Abouchabaka, J., 2012, A comparative study of adaptive crossover operators for genetic algorithms to resolve the traveling salesman problem, *arXiv preprint arXiv:1203.3097*.
- Adams, D., Disberry, A., Hutchison, N. ve Munjoma, T., 2001, Managing urban land: the case for urban partnership zones, *Regional Studies*, 35 (2), 153-162.
- Aerts, J. C., Eisinger, E., Heuvelink, G. B. ve Stewart, T. J., 2003, Using linear integer programming for multi-site land-use allocation, *Geographical analysis*, 35 (2), 148-169.
- Aggarwal, D., Saxena, D. K., Back, T. ve Emmerich, M., 2020, Real-World Airline Crew Pairing Optimization: Customized Genetic Algorithm versus Column Generation Method, *arXiv preprint arXiv:2003.03792*.
- Ahmed, Z. H., 2010, Genetic algorithm for the traveling salesman problem using sequential constructive crossover operator, *International Journal of Biometrics & Bioinformatics (IJBB)*, 3 (6), 96.
- Aksay, B., 1999, Hukuki Açıdan Arazi ve Arsa Düzenlemesi, *Mahalli İdareler Derneği Yayını, Ankara*.
- Alaybeyoğlu, A., Kılıç, U., İregöl, A. ve Konbul, Y., 2016, Kentsel Dönüşümde Mahalleler Arası Öncelik Sıralaması: Örnek Bir Karar Destek Sistemi, *Uludağ University Journal of The Faculty of Engineering*, 21 (2), 377-402.
- Anonim, 2020, 3194 sayılı İmar Kanunu Madde 18. Maddesi, <http://www.mevzuat.gov.tr>, (Son Erişim: 29.04.2020).
- Ayop, R. ve Tan, C. W., 2017, Improved control strategy for photovoltaic emulator using resistance comparison method and binary search method, *Solar Energy*, 153, 83-95.
- Aytuğ, O., 2013, Metasezgisel Yöntemler ve Uygulama Alanları, *Çukurova Üniversitesi İktisadi ve İdari Bilimler Fakültesi Dergisi*, 17 (2).
- Babalik, A., Cinar, A. C. ve Kiran, M. S., 2018, A modification of tree-seed algorithm using Deb's rules for constrained optimization, *Applied soft computing*, 63, 289-305.
- Bansal, J. C., Singh, P. K., Saraswat, M., Verma, A., Jadon, S. S. ve Abraham, A., 2011, Inertia Weight strategies in Particle Swarm Optimization, *2011 Third World Congress on Nature and Biologically Inspired Computing*, 633-640.
- Bozdağ, A. ve Ertunç, E., 2020, CBS ve AHP Yöntemi Yardımıyla Niğde Kenti Örneğinde Taşınmaz Değerleme, *Geomatik*, 5 (3), 228-240.
- Cinar, A. C. ve Kiran, M. S., 2018, Similarity and logic gate-based tree-seed algorithms for binary optimization, *Computers & Industrial Engineering*, 115, 631-646.
- Cypher, M. L. ve Forgey, F. A., 2003, Eminent domain: An evaluation based on criteria relating to equity, effectiveness, and efficiency, *Urban Affairs Review*, 39 (2), 254-268.

- Çayiroğlu, İ., 2013, İleri Algoritma Analizi Ders Notları.
- Çete, M., 2010, Turkish land readjustment: Good practice in urban development, *Journal of urban planning and development*, 136 (4), 373-380.
- Dahal, K. R. ve Chow, T. E., 2014, A GIS toolset for automated partitioning of urban lands, *Environmental Modelling & Software*, 55, 222-234.
- De Wolff, H., 2002, Urban land readjustment: establishing joint ownership as a new tool for urban regeneration in The Netherlands', *32nd Int. Symp. of the European Faculty of Land Use and Development*.
- Delaunay, B., 1934, Sur la Sphere Vide, bull. Acad. Science USSR VII: Class, *Sci. Mat. Nat*, 793-800.
- Demetriou, D., 2017, A spatially based artificial neural network mass valuation model for land consolidation, *Environment and Planning B: Urban Analytics and City Science*, 44 (5), 864-883.
- Doebele, W., 1982, Land Readjustment: A Different Approach to Financing Urbanization, Health and Company, Lexington Books, Massachusetts, DC.
- Doebele, W., Matsubara, J. ve Nishiyama, Y., 1986, Conceptual models of land readjustment, *Land Readjustment: The Japanese System, A Lincoln Institute of Land Policy Book, Boston, USA*.
- Eberhart, R. ve Kennedy, J., 1995a, Particle swarm optimization, *Proceedings of the IEEE international conference on neural networks*, 1942-1948.
- Eberhart, R. ve Kennedy, J., 1995b, A new optimizer using particle swarm theory, *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, 39-43.
- Ekiz, M., 2017, Eşdeğerlilik Esaslı Arazi Düzenlemesinde Analitik Hiyerarşik Prosesin Kullanılması, *Selçuk Üniversitesi Fen Bilimleri Enstitüsü*, Yüksek Lisans Tezi, Konya.
- Feng, Y., Liu, Y., Tong, X., Liu, M. ve Deng, S., 2011, Modeling dynamic urban growth using cellular automata and particle swarm optimization rules, *Landscape and Urban Planning*, 102 (3), 188-196.
- Ghavami, S. M., Taleai, M. ve Arentze, T., 2017, An intelligent spatial land use planning support system using socially rational agents, *International Journal of Geographical Information Science*, 31 (5), 1022-1041.
- Goldberg, D. E. ve Holland, J. H., 1988, Genetic algorithms and machine learning, *Machine learning*, 3 (2), 95-99.
- Goldberg, D. E., 1989, Genetic Algorithms in Search, Optimization and Machine Learning, Addison Wesley Publishing Company, Reading, MA.
- Güngör, R. ve İnam, Ş., 2019, İmar Uygulamalarında Farklı Dağıtım Metotlarının Karşılaştırılması, *Geomatik*, 4 (3), 254-263.
- Hakli, H., Uğuz, H. ve Çay, T., 2016, A new approach for automating land partitioning using binary search and Delaunay triangulation, *Computers and Electronics in Agriculture*, 125, 129-136.

- Karaboga, D., 2005, An idea based on honey bee swarm for numerical optimization, *Technical report-tr06, Erciyes university, engineering faculty, computer*
- Karaboga, D. ve Basturk, B., 2007, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, *Journal of global optimization*, 39 (3), 459-471.
- Karaboga, D. ve Basturk, B., 2008, On the performance of artificial bee colony (ABC) algorithm, *Applied soft computing*, 8 (1), 687-697.
- Karaboga, D. ve Akay, B., 2009, A comparative study of artificial bee colony algorithm, *Applied mathematics and computation*, 214 (1), 108-132.
- Kennedy, J. ve Eberhart, R., 1995, Particle swarm optimization, *Proceedings of ICNN'95-International Conference on Neural Networks*, 1942-1948.
- Kennedy, J. ve Eberhart, R. C., 1997, A discrete binary version of the particle swarm algorithm, *1997 IEEE International conference on systems, man, and cybernetics. Computational cybernetics and simulation*, 4104-4108.
- Kiran, M. S., 2015, TSA: Tree-seed algorithm for continuous optimization, *Expert Systems with Applications*, 42 (19), 6686-6698.
- Kiran, M. S., 2017, Withering process for tree-seed algorithm, *Procedia computer science*, 111, 46-51.
- Koc, I. ve Babaoglu, I., 2020, A comparative study of swarm intelligence and evolutionary algorithms on urban land readjustment problem, *Applied soft computing*.
- Koc, I., Cay, T. ve Babaoglu, I., 2020, Different Approaches to Automated Land Subdivision using Binary Search Algorithm in Zoning Applications, *Proceedings of the Institution of Civil Engineers-Municipal Engineer*, 1-27.
- Köktürk, E. ve Köktürk, E., 2009, Eşdeğerlik İlkesine Dayalı Arsa Düzenlemesinde Taşınmaz Değerlerinin Belirlenmesi, *Jeodezi ve Jeoinformasyon Dergisi* (101), 10-17.
- Kucukmehmetoglu, M. ve Geymen, A., 2016, Optimization models for urban land readjustment practices in Turkey, *Habitat international*, 53, 517-533.
- Larsson, G., 1997, Land readjustment: A tool for urban development, *Habitat international*, 21 (2), 141-152.
- Li, L.-H. ve Li, X., 2007, Land readjustment: An innovative urban experiment in China, *Urban Studies*, 44 (1), 81-98.
- Maleki, J., Hakimpour, F. ve Masoumi, Z., 2017, A parcel-level model for ranking and allocating urban land-uses, *ISPRS International Journal of Geo-Information*, 6 (9), 273.
- Masoumi, Z., Coello, C. A. C. ve Mansourian, A., 2020, Dynamic urban land-use change management using multi-objective evolutionary algorithms, *Soft Computing*, 24 (6), 4165-4190.
- Müller-Jökel, R., 2004, Land readjustment—A win-win-strategy for sustainable urban development, *FIG Working Week*, 22-27.

- Ng, I., 2002, Compulsory purchase and compensation in Hong Kong, *Property Management*.
- Nişancı, R., 2005, CBS ile Nominal Değerleme Yöntemine Dayalı Piksel Tabanlı Kentsel Taşınmaz Değer Haritalarının Üretilmesi, Karadeniz Teknik Üniversitesi, *Fen Bilimleri Enstitüsü, Doktora Tezi, Trabzon*.
- Onwubolu, G. ve Babu, B., 2004, New optimization techniques in engineering. 2004, Springer, Berlin, New York.
- Özçakar, N. ve Bastı, M., 2012, P-Medyan kuruluş yeri seçim probleminin çözümünde parçacık sürü optimizasyonu algoritması yaklaşımı, *Istanbul University Journal of the School of Business Administration*, 41 (2).
- Özsağlam, M. Y. ve Çunkaş, M., 2008, Optimizasyon problemlerinin çözümü için parçacık sürü optimizasyonu algoritması, *Politeknik Dergisi*, 11 (4).
- Qin, A. K., Huang, V. L. ve Suganthan, P. N., 2008, Differential evolution algorithm with strategy adaptation for global numerical optimization, *IEEE Transactions on evolutionary computation*, 13 (2), 398-417.
- Rabbani, A., Aghababae, H. ve Rajabi, M. A., 2012, Modeling dynamic urban growth using hybrid cellular automata and particle swarm optimization, *Journal of Applied Remote Sensing*, 6 (1), 063582.
- Rahnamayan, S., Tizhoosh, H. R. ve Salama, M. M., 2008, Opposition-based differential evolution, *IEEE Transactions on evolutionary computation*, 12 (1), 64-79.
- Reeves, C. ve Rowe, J. E., 2002, Genetic algorithms: principles and perspectives: a guide to GA theory, Springer Science & Business Media, p.
- Songu, C., Şerbetçi, M. ve Gülal, E., 2009, Ölçme bilgisi, Birsan yayınevi, p.
- Sorensen, A., 2000a, Land readjustment and metropolitan growth: an examination of suburban land development and urban sprawl in the Tokyo metropolitan area, *Progress in planning*, 53 (4), 217-330.
- Sorensen, A., 2000b, Conflict, consensus or consent: implications of Japanese land readjustment practice for developing countries, *Habitat international*, 24 (1), 51-73.
- Storn, R. ve Price, K., 1995, Differential evolution—a simple and efficient adaptive scheme for global optimization over continuous spaces: technical report TR-95-012, *International Computer Science, Berkeley, California*.
- Storn, R., 1996, On the usage of differential evolution for function optimization, *Proceedings of North American Fuzzy Information Processing*, 519-523.
- Storn, R. ve Price, K., 1997, Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces, *Journal of global optimization*, 11 (4), 341-359.
- Terzioğlu, G., 2015, Çok Hisseli Parsellerde İmar Planı Uygulaması: Sultanbeyli Örneği. Gebze Teknik Üniversitesi, *Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi*.
- Turk, S. S., 2005, Land readjustment: an examination of its application in Turkey, *Cities*, 22 (1), 29-42.

- Türk, S. S., 2007, An analysis on the efficient applicability of the land readjustment (LR) method in Turkey, *Habitat international*, 31 (1), 53-64.
- Türk, S. S. ve Altes, W. K. K., 2010, How suitable is LR for renewal of inner city areas? An analysis for Turkey, *Cities*, 27 (5), 326-336.
- Türk, Ş. Ş., 2009, Arazi ve arsa düzenlemesi yöntemi ve uluslar arası çerçevede etkin uygulanabilirliği, *ITU Journal Series A: Architecture, Planning, Design*, 8 (1).
- Uygun, U., 2011, İmar Uygulamasında Değer Esası Kriterlerine Göre Dağıtım Probleminin Yapay Zekâ Yöntemiyle Çözümü, Gebze Yüksek Teknoloji Enstitüsü, *Yüksek Lisans Tezi*.
- Uzun, B., 1992, Kentsel Alan Düzenlemelerinde İmar Parseli Üretme Yöntemleri ve Sonuçlarının İrdelenmesi, *Trabzon: Karadeniz Technical University, MS Geodesy And Photometry Engineering*.
- Uzun, B., 2009, Using land readjustment method as an effective urban land development tool in Turkey, *Survey Review*, 41 (311), 57-70.
- Weile, D. S. ve Michielssen, E., 1997, Genetic algorithm optimization applied to electromagnetics: A review, *IEEE Transactions on Antennas and Propagation*, 45 (3), 343-353.
- Whitley, D., 1994, A genetic algorithm tutorial, *Statistics and computing*, 4 (2), 65-85.
- Wickramasuriya, R., Chisholm, L. A., Puotinen, M., Gill, N. ve Klepeis, P., 2011, An automated land subdivision tool for urban and regional planning: Concepts, implementation and testing, *Environmental Modelling & Software*, 26 (12), 1675-1684.
- Yilmaz, A., Çağdaş, V. ve Demir, H., 2015, An evaluation framework for land readjustment practices, *Land use policy*, 44, 153-168.
- Yomralioglu, T., 1992, Arsa ve Arazi Düzenlemesi için Yeni bir Uygulama Şekli.
- Yomralioglu, T. ve Parker, D., 1993, A GIS-based land readjustment system for urban development, *Proceedings of the 4th European Conference on geographical information system: EGIS '93 Conference', Genoa, 29th March–1st April, EGIS Foundation, Utrecht*, 372-379.
- Yomralioglu, T., Nisanci, R. ve Yildirim, V., 2007, An implementation of nominal asset based land readjustment, *FIG Working Week*, 13-17.

EKLER

EK-1 Gerçek dünya problemi için önerilen algoritmalara ait parametre değerleri

ABC		DE		GA		PSO		TSA	
Parametre Adı	Değeri	Parametre Adı	Değeri	Parametre Adı	Değeri	Parametre Adı	Değeri	Parametre Adı	Değeri
Popülasyon Boyutu	20	Popülasyon Boyutu	20	Popülasyon Boyutu	20	Popülasyon Boyutu	20	Popülasyon Boyutu	20
İterasyon	1000	İterasyon	1000	İterasyon	1000	İterasyon	1000	İterasyon	1000
MaxFEs	0.2×10^5	MaxFEs	0.2×10^5	MaxFEs	0.2×10^5	MaxFEs	0.2×10^5	MaxFEs	0.2×10^5
Limit Değeri	125	Çaprazlama Oranı (r2 ve r3)	0.80	Seçim Operatörü	Rulet tekerleği	Çaprazlama Oranı (Mevcut ve pBest)	0.80	ST	0.3
Seçim Operatörü	Turnuva	Çaprazlama Oranı (r1 ve r2)	0.90	Elitizm Oranı	0.10	Çaprazlama Oranı (Mevcut ve Gbest)	0.90	Maksimum çekirdek sayısı (%)	0.25
Alt küme boyutu	5	Çaprazlama Oranı (mevcut ve r1)	0.70	Çaprazlama Oranı (cr1 ve cr2)	0.75	Çaprazlama Oranı (pBest ve Gbest)	0.70	Minimum çekirdek sayısı (%)	0.10
Çaprazlama Oranı (İşçi arı evresi)	0.70							Çaprazlama Oranı (Mevcut ve en iyi ağaç)	ST
Çaprazlama Oranı (Gözcü arı evresi)	0.90							Çaprazlama Oranı (Mevcut ve rastgele ağaç)	1-ST

EK-2 Sentetik veri seti için önerilen algoritmalara ait parametre değerleri

ABC		DE		GA		PSO		TSA	
Parametre Adı	Değeri	Parametre Adı	Değeri	Parametre Adı	Değeri	Parametre Adı	Değeri	Parametre Adı	Değeri
Popülasyon Boyutu	50	Popülasyon Boyutu	50	Popülasyon Boyutu	50	Popülasyon Boyutu	50	Popülasyon Boyutu	50
İterasyon	2000	İterasyon	2000	İterasyon	2000	İterasyon	2000	İterasyon	2000
MaxFEs	10 ⁵	MaxFEs	10 ⁵	MaxFEs	10 ⁵	MaxFEs	10 ⁵	MaxFEs	10 ⁵
Limit Değeri	125	Çaprazlama Oranı (r2 ve r3)	0.80	Seçim Operatörü	Turnuva	Çaprazlama Oranı (Mevcut ve pBest)	0.80	ST	0.3
Seçim Operatörü	Turnuva	Çaprazlama Oranı (r1 ve r2)	0.90	Alt Küme Boyutu	5	Çaprazlama Oranı (Mevcut ve Gbest)	0.90	Maksimum çekirdek sayısı (%)	0.25
Alt küme boyutu	5	Çaprazlama Oranı (mevcut ve r1)	0.70	Elitizm Oranı	0.10	Çaprazlama Oranı (pBest ve Gbest)	0.70	Minimum çekirdek sayısı (%)	0.10
Çaprazlama Oranı (İşçi arı evresi)	0.70			Çaprazlama Oranı (cr1 ve cr2)	0.75			Çaprazlama Oranı (Mevcut ve en iyi ağaç)	ST
Çaprazlama Oranı (Gözcü arı evresi)	0.90							Çaprazlama Oranı (Mevcut ve rastgele ağaç)	1-ST

EK-3.1 Parsel sayısı 40, NLP değeri 3 olan probleme ait en iyi çözüme ait bir örnek

1 ● A1-2 ● A2-2 300.0	2 ● B2-2 ● B1-2 300.0	3 ● C2-2 ● C1-2 300.0	4 ● D2-2 ● D1-2 300.0	5 ● E1-2 ● E2-2 300.0	6 ● F1-2 ● F2-2 300.0	7 ● G1-2 ● G2-2 300.0	8 ● H1-2 ● H2-2 300.0	9 ● I2-2 ● I1-2 300.0	10 ● J2-2 ● J1-2 300.0
20 ● T2-2 ● T1-2 300.0	19 ● S1-2 ● S2-2 300.0	18 ● R1-2 ● R2-2 300.0	17 ● Q2-2 ● Q1-2 300.0	16 ● P1-2 ● P2-2 300.0	15 ● O1-2 ● O2-2 300.0	14 ● N2-2 ● N1-2 300.0	13 ● M1-2 ● M2-2 300.0	12 ● L2-2 ● L1-2 300.0	11 ● K1-2 ● K2-2 300.0
1 ● A2-1 ● A1-1 300.0	2 ● B1-1 ● B2-1 300.0	3 ● C2-1 ● C1-1 300.0	4 ● D2-1 ● D1-1 300.0	5 ● E1-1 ● E2-1 300.0	6 ● F1-1 ● F2-1 300.0	7 ● G2-1 ● G1-1 300.0	8 ● H2-1 ● H1-1 300.0	9 ● I2-1 ● I1-1 300.0	10 ● J1-1 ● J2-1 300.0
20 ● T2-1 ● T1-1 300.0	19 ● S2-1 ● S1-1 300.0	18 ● R2-1 ● R1-1 300.0	17 ● Q1-1 ● Q2-1 300.0	16 ● P2-1 ● P1-1 300.0	15 ● O2-1 ● O1-1 300.0	14 ● N1-1 ● N2-1 300.0	13 ● M1-1 ● M2-1 300.0	12 ● L1-1 ● L2-1 300.0	11 ● K1-1 ● K2-1 300.0

EK-3.2 Parsel sayısı 100, NLP değeri 1 olan probleme ait en ideal çözüme ait sonuç

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
●A-2 300.0	●B-2 300.0	●C-2 300.0	●D-2 300.0	●E-2 300.0	●F-2 300.0	●G-2 300.0	●H-2 300.0	●I-2 300.0	●J-2 300.0	●K-2 300.0	●L-2 300.0	●M-2 300.0	●N-2 300.0	●O-2 300.0	●P-2 300.0	●Q-2 300.0	●R-2 300.0	●S-2 300.0	●T-2 300.0	●U-2 300.0	●V-2 300.0	●W-2 300.0	●X-2 300.0	●AA-2 300.0
50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	31	30	29	28	27	26
●AX-2 300.0	●AW-2 300.0	●AV-2 300.0	●AU-2 300.0	●AT-2 300.0	●AS-2 300.0	●AR-2 300.0	●AQ-2 300.0	●AP-2 300.0	●AO-2 300.0	●AN-2 300.0	●AM-2 300.0	●AL-2 300.0	●AK-2 300.0	●AJ-2 300.0	●AI-2 300.0	●AH-2 300.0	●AG-2 300.0	●AF-2 300.0	●AE-2 300.0	●AD-2 300.0	●AC-2 300.0	●AB-2 300.0	●Y-2 300.0	●Z-2 300.0
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
●A-1 300.0	●B-1 300.0	●C-1 300.0	●D-1 300.0	●E-1 300.0	●F-1 300.0	●G-1 300.0	●H-1 300.0	●I-1 300.0	●J-1 300.0	●K-1 300.0	●L-1 300.0	●M-1 300.0	●N-1 300.0	●O-1 300.0	●P-1 300.0	●Q-1 300.0	●R-1 300.0	●S-1 300.0	●T-1 300.0	●U-1 300.0	●V-1 300.0	●W-1 300.0	●X-1 300.0	●Y-1 300.0
50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	31	30	29	28	27	26
●AX-1 300.0	●AW-1 300.0	●AV-1 300.0	●AU-1 300.0	●AT-1 300.0	●AS-1 300.0	●AR-1 300.0	●AQ-1 300.0	●AP-1 300.0	●AO-1 300.0	●AN-1 300.0	●AM-1 300.0	●AL-1 300.0	●AK-1 300.0	●AJ-1 300.0	●AI-1 300.0	●AH-1 300.0	●AG-1 300.0	●AF-1 300.0	●AE-1 300.0	●AD-1 300.0	●AC-1 300.0	●AB-1 300.0	●AA-1 300.0	●Z-1 300.0

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı : İsmail KOÇ
Uyruğu : T.C.
Doğum Yeri ve Tarihi : Konya - 1987
Telefon : 0 543 648 94 01
Faks : -
E-Posta : ismailkoc42@gmail.com, ismailkoc@ktun.edu.tr

EĞİTİM

Derece	Adı	İlçe	İl	Bitirme Yılı
Lise	: Dumlupınar Lisesi	Selçuklu	Konya	2004
Üniversite	: Selçuk Üniversitesi	Selçuklu	Konya	2010
Yüksek Lisans	: Selçuk Üniversitesi	Selçuklu	Konya	2016
Doktora	: Konya Teknik Üniversitesi	Selçuklu	Konya	-

İŞ DENEYİMLERİ

Yıl	Kurum	Görevi
2009-2010	Belya Yazılım	Yarı Zamanlı Yazılım Geliştirici
2011-2011	Elpo Otomasyon	Bilgisayar Mühendisi
2011-2012	Ayosis Yazılım	Bilgisayar Mühendisi
2012-2013	Hitit Üniversitesi	Araştırma Görevlisi
2013-2018	Selçuk Üniversitesi	Araştırma Görevlisi
2018-	Konya Teknik Üniversitesi	Araştırma Görevlisi

UZMANLIK ALANI

Yapay Zeka, Sürü Zekası, Evrimsel Algoritmalar, Mekânsal Veriler

YABANCI DİLLER

İngilizce
YÖKDİL: 83.75

YAYINLAR

Uluslararası SCI, SCI-Exp İndekslerinde Taranan Dergilerdeki Yayınlar

1. **Koc I.**, Babaoğlu I., 2020, *A Comparative Study of Swarm Intelligence and Evolutionary Algorithms on Urban Land Readjustment Problem*, Applied Soft Computing (Doktora tezinden üretilmiştir.)

2. **Koc I.**, Çay T., Babaoğlu I., 2020, *Different Approaches to Automated Land Subdivision using Binary Search Algorithm in Zoning Applications*, Proceedings Of The Institution of Civil Engineers-Municipal Engineer (Doktora tezinden üretilmiştir.)
3. Beskirli M., **Koc I.**, Kodaz H., 2019, *Optimal Placement of Wind Turbines Using Novel Binary Invasive Weed Optimization*, Tehnicki Vjesnik-Technical Gazette
4. Beskirli M., **Koc I.**, Hakli H., Kodaz H., 2018, *A new optimization algorithm for solving wind turbine placement problem: Binary artificial algae algorithm*, Renewable Energy
5. Atay Y., **Koc I.**, Babaoğlu I., Kodaz H., 2017, *Community Detection from Biological and Social Networks A Comparative Analysis of Metaheuristic Algorithms*, Applied Soft Computing
6. Inik O., Ulker E., **Koc I.**, 2020, *B-Spline Curve Approximation by Utilizing Big Bang-Big Crunch Method*, Computer Systems Science & Engineering

Uluslararası Diğer İndekslerde Taranan Dergilerdeki Yayınlar

1. **Koc I.**, Kivrak H., Babaoğlu I., 2019, *The Estimation Of The Energy Demand in Turkey Using Grey Wolf Optimizer Algorithm*, Annals of the Faculty of Engineering Hunedoara-International Journal of Engineering
2. **Koc I.**, Nureddin R., Kahramanli H., 2018, *Türkiye’de Enerji Talebini Tahmin Etmek İçin Doğrusal Form Kullanarak GSA ve IWO Tekniklerinin Uygulanması*, Selçuk Üniversitesi Mühendislik, Bilim ve Teknoloji Dergisi
3. **Koc I.**, Hakli H., 2018, *A New Mutation Operator for Differential Evolution Algorithm in Solving Continuous Problems*, Annals of Faculty Engineering Hunedoara – International Journal of Engineering
4. **Koc I.**, Baykan OK., Babaoğlu I., 2018, *Gri kurt optimizasyon algoritmasına dayanan çok seviyeli imge eşik seçimi*, Journal of Polytechnic
5. Atay, Y., **Koc I.**, and Beskirli M., 2017, *Detection of Cohesive Subgroups in Social Networks using Invasive Weed Optimization Algorithm*, The Eurasia Proceedings of Educational & Social Sciences
6. Inik O., **Koc I.**, 2017, *Gray Wolf Optimizer for Knot Placement in B-Spline Curve Fitting*, Gaziosmanpaşa Bilimsel Araştırma Dergisi (GBAD)
7. **Koc I.**, 2016, *Big Bang Big Crunch Optimization Algorithm for Solving the Uncapacitated Facility Location Problem*, International Journal of Intelligent Systems and Applications in Engineering (IJISAE)

Uluslararası Konferanslarda Sunulan Yayınlar

1. **Koc I.**, Inik O., Ulker E., Babaoglu I., 2019, *Effects of Data Augmentation in Melanoma Cancer Classification Using Convolution Neural Network*, 533rd International Conference on Science, Engineering Technology - ICSET 2019, Nov 20-21, 2019
2. Beskirli M., **Koc I.**, Kodaz H., 2017, *Parameter Tuning for The Artificial Algae Algorithm*, International Conference on Research in Education and Science, May 18-21, 2017
3. **Koc I.**, Nureddin R., Babaoglu I., Uymaz SA., 2017, *Binary Invasive Weed Optimization Algorithm Approaches for Binary Optimization*, 14th International Conference on Informatics and Information Technologies , Sept 07-09, 2017
4. **Koc I.**, Inik O., 2017, *Cuckoo Search Algorithm for Optimal Choice of Free Knots in B-Spline Data Fitting*, 1st International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT 2017), Nov 02-04, 2017
5. **Koc I.**, Beskirli M., 2017, *Forecasting Turkey's Energy Demand Using CuckooSearch Algorithm*, International Conference on Research in Education and Science (ICRES 2017), May 18-21, 2017
6. Beskirli M., **Koc I.**, Kodaz H., 2016, *Individual Parameter Selection Strategy for Artificial Algae Algorithm*, 9th International Conference on Computer and Electrical Engineering, Dec 07-12, 2016
7. **Koc I.**, Karadogan A., 2016, *S Shaped Transfer Functions For Binary Firefly Algorithm*, International Conference on Computer Science and Engineering, Oct 20-23, 2016
8. Beskirli M., **Koc I.**, 2015, *A Comparative Study of Improved Bat Algorithm and Bat Algorithm on Numerical Benchmarks*, Yayın Yeri: 4th International Conference on Advanced Computer Science Applications and Technologies (ACSAT), Dec 08-10, 2015
9. **Koc I.**, Babaoglu I., 2015, *A Novel Modified Binary Bat Algorithm For Feature Selection in Classification Problems*, International Conference on Advanced Technology and Sciences - ICAT 2015, Aug 04-07, 2015 (Yüksek lisans tezinden üretilmiştir.)

PROJELER

1. “Optimizasyon Tabanlı Arazi Toplulaştırma Yazılımı”, 01.01.2018 - 31.05.2019, (TÜBİTAK KOBİ Ar-Ge 1507 Başlangıç Destek Programı), Araştırmacı