

ÇOK BÜYÜK KONUMSAL VERİNİN KULLANICI ODAKLI GÖRSELLEŞTİRİLMESİ

USER ORIENTED VISUALIZATION OF VERY LARGE SPATIAL DATA

MUHAMMED TEKİN ERTEKİN

DR. ÖĞR. Ü. BURKAY GENÇ

Tez Danışmanı

Hacettepe Üniversitesi

Lisansüstü Eğitim – Öğretim ve Sınav Yönetmeliğinin

Bilgisayar Grafiği Anabilim Dalı İçin Öngördüğü

YÜKSEK LİSANS TEZİ

olarak hazırlanmıştır.

2020

Eşime ve Aileme...

ÖZET

ÇOK BÜYÜK KONUMSAL VERİNİN KULLANICI ODAKLI GÖRSELLEŞTİRİLMESİ

Ertekin, Muhammed Tekin

Yüksek Lisans, Bilgisayar Animasyonu ve Oyun Teknolojileri Bölümü

Tez Yöneticisi : BURKAY GENÇ

Eylül 2020, 102 sayfa

Günümüzde konum servis sağlayıcıların ve konum hizmeti veren donanımların gelişmesiyle birlikte birçok farklı cihazdan konum verileri alınabilmekte ve bu verilerin çeşitli uygulamalarda kullanılabilmesi gerçekleşmektedir. Toplanan veriler gerek coğrafik gerek beşeri faktörlerden etkilenmekte ve homojen ya da heterojen dağılımlar gösterebilmektedir. Bu kadar çok verinin kullanıcıların algılayabileceği şekilde gösterilmesi özel yöntemler gerektirmektedir. Yazılacak bu tez çok sayıda konumsal verinin kullanıcılara verimli şekilde ifade edilmesini amaçlamaktadır. Bu çalışmanın sonucunda çok fazla sayıda (örneğin: milyonlarca) noktadan oluşan konumsal verinin, küresel ya da bölgesel olarak görselleştirilmesi hedeflenmektedir.

Anahtar Kelimeler: görselleştirme, konum verisi, Voronoi Diyagramları

ABSTRACT

USER ORIENTED VISUALIZATION OF VERY LARGE SPATIAL DATA

Ertekin, Muhammed Tekin

M.Sc., Department of Computer Graphics

Supervisor : BURKAY GENÇ

September 2020, 102 pages

Today, with the development of location service providers and equipment providing location services, location data can be obtained from many different devices and this data can be used in various applications. The data collected are affected by both geographic and human factors, and sometimes they may show homogeneous and sometimes heterogeneous distributions. Displaying so much data in a way that users can perceive requires special methods. This thesis to be written aims at expressing many spatial data to the users in the most efficient way. As a result of this study, it is aimed to visualize spatial data consisting of many (e.g. millions) points globally or regionally and to do this in real time.

Keywords: visualization, spatial data, rasterization, Voronoi Diagrams

TEŞEKKÜR

Tezli Yüksek Lisans eğitim sürecinde gerek akademik gerekse hayat tecrübesiyle yapmış olduğum bilimsel çalışma ve araştırmalara destek veren, tez yazım sürecinin sadece basit uygulamalarla ilgili olmadığını ve akademik yönünün olduğunu gösteren, bu konulardaki bilgi ve tecrübelerini benimle paylaşan kıymetli insan değerli hocam Dr. Öğr. Üyesi Burkey GENÇ'e,

Tezimin jüri üyeliğinde yer alan ve kıymetli fikirleriyle daha kaliteli bir çalışma oluşturulmasına yardımcı olan Prof. Dr. Haşmet GÜRÇAY, Dr. Öğr. Üyesi Zümra KAVAFOĞLU, Dr. Öğr. Üyesi Ufuk ÇELİKCAN ve Dr. Öğr. Üyesi Fatih SAĞLAM'a,

Tüm bu çalışmalarım esnasında yanımda olan ve beni bilimin ışığında yetiştiren aileme,

Tez çalışmam ve akademik kariyerime olan katkıları sebebiyle, hayatımın her anında yanımda olan ve desteğini hissettiğim hayat arkadaşım, eşim, Zeynep ERTEKİN'e,

Teşekkürlerimle...

Muhammed Tekin ERTEKİN

Eylül 2020, Ankara

İÇİNDEKİLER

ÖZET	i
ABSTRACT	ii
İÇİNDEKİLER	v
TABLO LİSTESİ	vii
ŞEKİL LİSTESİ	ix

BÖLÜMLER

1	GİRİŞ	1
1.1	Tanımlamalar ve Alt Yapı	3
2	LİTERATÜR ARAŞTIRMALARI	9
3	ÇALIŞMADA KULLANILAN VERİ SETLERİ	31
3.1	Dünya Düzleminde Oluşturulan Sentetik Veri	31
3.2	Türkiye Düzleminde Oluşturulan Sentetik Veri	34
3.3	Dünya Düzlemindeki Gerçek Veri	35
4	METODOLOJİ	37
4.1	Mevcut Metodolojiler	37
4.1.1	Metod 1 - Sabit Kare Kullanılarak Bölütleme	38
4.1.2	Metod 2 - Adaptif Kare Kullanılarak Bölütleme	42
4.1.3	Metod 3 - Adaptif Dikdörtgen Kullanılarak Bölüt- leme	45
4.2	Önerilen Metodoloji	48
4.2.1	Önerilen Metodolojinin Girdileri	48
4.2.2	Önerilen Metodolojinin Çıktıları	50
4.2.3	İlk Karelere Bölütleme	51
4.2.4	İlk Karelerin Daha Küçük Karelere Bölünmesi	53
4.2.5	Kareler İçerisinde Rastgele Noktalar Seçilmesi ve Voronoi Diyagramlarının Oluşturulması	56

4.2.6	Ball-tree Algoritması Kullanılarak Voronoi Diyagramlarının Renklendirilmesi	57
4.2.7	Taban Haritası ve Renklendirilmiş Voronoi Diyagramları ile Nihai Görüntünün Elde Edilmesi	59
5	TESTLER VE SONUÇLAR	61
5.1	Performans Analizleri ve Kıyaslamalar	61
5.1.1	Görsel Analizler	61
5.1.2	Sayısal Analizler	72
5.1.3	Kıyaslamalar	80
5.2	Test ve Çıktı Yorumları	86
6	SONUÇ	87
	KAYNAKLAR	89
	EKLER	

TABLO LİSTESİ

Tablo 3.1	Dünya düzleminde oluşturulan sentetik verinin dağılımı	31
Tablo 3.2	Türkiye düzleminde oluşturulan sentetik verinin dağılımı	34
Tablo 3.3	Dünya düzleminde hava kalitesi verisinin dağılımı	36
Tablo 4.1	Veri setlerine ve kare boyutlarına göre toplam entropi değerleri[31]	39
Tablo 4.2	Metod 2 içerisinde farklı veri setlerine ve kare boyutlarına göre toplam entropi değerleri[31]	44
Tablo 4.3	Metod 3 tüm veri setleri için minimum boyutlara göre entropi değerleri[31]	47
Tablo 4.4	Noktaların karelere bölünmesi	52
Tablo 4.5	Karelerin daha küçük karelere bölünmesi	54
Tablo 4.6	Kareleri dörde bölme işlemi	55
Tablo 4.7	Rastgele noktaların seçilmesi ve Voronoi diyagramlarının oluşturulması	56
Tablo 4.8	Ball-tree algoritması ile Voronoi diyagramının renklendirilmesi . . .	58
Tablo 4.9	Tüm fonksiyonların bulunduğu algoritma işleyişi	60
Tablo 5.1	Veri seti içerisindeki nokta sayısına göre algoritma çalışma zamanının değişimi	74
Tablo 5.2	Merkez nokta sayısına göre algoritma çalışma zamanının değişimi .	75
Tablo 5.3	Veri seti içerisindeki nokta sayısına göre algoritma hafıza kullanımı (megabayt)	76
Tablo 5.4	Merkez nokta sayısına göre algoritma hafıza miktarının değişimi . .	77
Tablo 5.5	Veri seti içerisindeki nokta sayısına göre entropi değerleri	78
Tablo 5.6	Merkez nokta sayısına göre entropi değerleri	80
Tablo 5.7	Metod 1, Metod 2, Metod 3 ve önerilen metodun çalışma zamanları (sn)	82
Tablo 5.8	Tüm metodların hücre sayısı	83
Tablo 5.9	Tüm metodların veri setlerine göre entropi değerleri	85

ŞEKİL LİSTESİ

Şekil 1.1	Strateji oyunlarında harita örnekleri (a) Age of Empires II (b) Command and Conquer: Generals 2	2
Şekil 1.2	Askeri hareket merkezi harita kullanımı	3
Şekil 1.3	Renklendirilmiş Voronoi diyagramı örneği	5
Şekil 1.4	Ekstra veri içermeyen konum veri seti örneği	6
Şekil 2.1	Öklid uzaklık ölçüsüne göre Voronoi diyagramı[8]	9
Şekil 2.2	p boyutlu uzayda Minkowski uzaklığı hesaplaması [9]	10
Şekil 2.3	Manhattan uzaklık ölçüsüne göre Voronoi diyagramı[8]	10
Şekil 2.4	Zürafa üzerinde yer alan Voronoi diyagramı benzeri şekiller[8]	11
Şekil 2.5	Yaprak üzerindeki dokuların oluşturduğu Voronoi diyagramı benzeri şekiller[8]	11
Şekil 2.6	Kurumuş toprağın oluşturduğu Voronoi diyagramı benzeri şekiller[8] .	12
Şekil 2.7	KNN algoritması çalışma mantığı[12]	13
Şekil 2.8	Ball-tree yapısı (a) Çemberler (b) İlişkili ağaç yapısı[17]	15
Şekil 2.9	Ball-tree algoritması[16]	15
Şekil 2.10	Kd-tree yapısı [22]	16
Şekil 2.11	K-means algoritması ile kümeleme aşamaları[26]	17
Şekil 2.12	Pikselleştirme örneği	18
Şekil 2.13	9 hücreli basit bir pikselleştirme örneği	19
Şekil 2.14	Türkiye veri seti pikselleştirme örneği	20
Şekil 2.15	B5 veri seti pikselleştirme örneği	20
Şekil 2.16	Hava kalitesi veri seti pikselleştirme örneği	21
Şekil 2.17	Amerikan iç göç haritası (tüm vektörlerin gösterimi)[32]	22
Şekil 2.18	Amerikan iç göç haritası (birleştirilen düğümler)[32]	22
Şekil 2.19	Amerikan iç göç haritası (tüm vektörlerin güncellenme sonrası gösterimi)[32]	23
Şekil 2.20	HPGCL algoritması girdileri ve çıktıları[33]	24
Şekil 2.21	Jiri Komzak ve Marc Eisenstadt tarafından oluşturulan öğrenci verilerini konumsal olarak gösteren uygulama [34]	24
Şekil 2.22	SNN algoritmasıyla birleştirilen vektörel veriler [35]	25
Şekil 2.23	Taksi verilerinin karmaşıklığı [35]	25
Şekil 2.24	Xi Zhu ve Diansheng Guo tarafından oluşturulan nihai görüntü [35] .	26
Şekil 2.25	2 boyutlu noktaların oluşturduğu Voronoi diyagramları ve Delaunay üçgenlemeleri [36]	26
Şekil 2.26	Voronoi diyagramları ve Delaunay üçgenlemeleri kullanılarak hazırlanan topolojik harita [36]	27
Şekil 2.27	Seviyelere göre hatalar ve entropi grafiği[37]	27
Şekil 2.28	Sıcaklık verisine göre oluşturulan çok renkli Voronoi diyagramları[37] .	28
Şekil 2.29	Topolojik harita ve ilgili Voronoi diyagramı[38]	28
Şekil 2.30	Voronoi diyagramı kullanılarak elde edilen komşuluk ilişkisi[38] . .	29
Şekil 2.31	Voronoi diyagramı kullanılarak elde edilen ağaç yapısı[38]	30

Şekil 3.1	Veri seti B1 , 7160 nokta (3612 kırmızı - 3548 yeşil)	32
Şekil 3.2	Veri seti B2 , 14323 nokta (7208 kırmızı - 7115 yeşil)	32
Şekil 3.3	Veri seti B3 , 28350 nokta (14284 kırmızı - 14066 yeşil)	33
Şekil 3.4	Veri seti B4 , 68090 nokta (34245 kırmızı - 33845 yeşil)	33
Şekil 3.5	Veri seti B5 , 127672 nokta (64036 kırmızı - 63636 yeşil)	33
Şekil 3.6	Türkiye düzleminde oluşturulan sentetik veri temel haritası	34
Şekil 3.7	Türkiye düzlemi sentetik veriler, kırmızı 100000, yeşil 100000 nokta	35
Şekil 3.8	Dünya hava kalitesi endeksi haritası (https://waqi.info/)	35
Şekil 3.9	Dünya hava kalitesi endeksi noktasal dağılımı	36
Şekil 4.1	Çakmak tarafından uygulanan Metod 1 algoritmasının entropi hesaplaması [31]	38
Şekil 4.2	Metod 1 algoritmasının 20° kare büyüklüğüyle gösterimi[31]	39
Şekil 4.3	Metod 1 algoritmasının 10° kare büyüklüğüyle gösterimi [31]	40
Şekil 4.4	Metod 1 algoritmasının 5° kare büyüklüğüyle gösterimi [31]	40
Şekil 4.5	Metod 1 algoritmasının 2.5° kare büyüklüğüyle gösterimi [31]	40
Şekil 4.6	Metod 1 algoritmasının 1.25° kare büyüklüğüyle gösterimi [31]	41
Şekil 4.7	Metod 1 algoritmasının B5 veri seti kullanılarak oluşturulan entropi grafiği [31]	41
Şekil 4.8	Metod 2 algoritmasının B5 veri seti görsel sonucu entropi : 0.8326 , kare boyutu : 20° [31]	42
Şekil 4.9	Metod 2 algoritmasının B5 veri seti görsel sonucu entropi : 0.7224 , kare boyutu : 10° [31]	42
Şekil 4.10	Metod 2 algoritmasının B5 veri seti görsel sonucu entropi : 0.6647 , kare boyutu : 5° [31]	43
Şekil 4.11	Metod 2 algoritmasının B5 veri seti görsel sonucu entropi : 0.6194 , kare boyutu : 2.5° [31]	43
Şekil 4.12	Metod 2 algoritmasının B5 veri seti görsel sonucu entropi : 0.5485 , kare boyutu : 1.25° [31]	43
Şekil 4.13	Metod 2 algoritmasının B5 veri seti kullanılarak oluşturulan entropi değerleri[31]	44
Şekil 4.14	Metod 3 algoritmasının B5 veri seti görsel sonucu entropi : 0.9756 , minimum boyut : 20° [31]	45
Şekil 4.15	Metod 3 algoritmasının B5 veri seti görsel sonucu entropi : 0.8680 , minimum boyut : 10° [31]	45
Şekil 4.16	Metod 3 algoritmasının B5 veri seti görsel sonucu entropi : 0.7933 , minimum boyut : 5° [31]	46
Şekil 4.17	Metod 3 algoritmasının B5 veri seti görsel sonucu entropi : 0.7293 , minimum boyut : 2.5° [31]	46
Şekil 4.18	Metod 3 algoritmasının B5 veri seti görsel sonucu entropi : 0.6788 , minimum boyut : 1.25° [31]	46
Şekil 4.19	Metod 3 algoritmasının B5 veri seti entropi grafiği[31]	47
Şekil 4.20	Kullanılan kare sınıfı	52
Şekil 4.21	B5 veri seti için ilk karelerin oluşumu kare büyüklüğü = 30	53
Şekil 4.22	B5 veri seti için bölünen kareler yoğunluk eşiği = 5, baskınlık eşiği = 0.4, derinlik = 5	55
Şekil 4.23	B5 veri seti için seçilen rastgele noktalar yoğunluk eşiği = 5, baskınlık eşiği = 0.4, derinlik = 5	57

Şekil 4.24 B5 veri seti için oluşturulan Voronoi diyagramı yoğunluk eşiği = 5, baskınlık eşiği = 0.4, derinlik = 5	57
Şekil 4.25 B5 veri seti için oluşturulan Voronoi diyagramı (renkli) yoğunluk eşiği = 5, baskınlık eşiği = 0.4, derinlik = 5	59
Şekil 4.26 B5 veri seti için nihai harita yoğunluk eşiği = 5, baskınlık eşiği = 0.4	59
Şekil 5.1 Veri seti B1, 7160 nokta, yoğunluk eşiği = 0.25, hücre sayısı = 2486, algoritma zamanı = 0.1306164 saniye, algoritma ram miktarı = 3.827621 megabayt	62
Şekil 5.2 Veri seti B2, 14323 nokta, yoğunluk eşiği = 0.7, hücre sayısı = 5600, algoritma zamanı = 0.317150 saniye, algoritma ram miktarı = 8.403469 megabayt	62
Şekil 5.3 Veri seti B3, 28350 nokta, yoğunluk eşiği = 1.5, hücre sayısı = 7337, algoritma zamanı = 0.591450 saniye, algoritma ram miktarı = 12.320357 megabayt	63
Şekil 5.4 Veri seti B4, 68090 nokta, yoğunluk eşiği = 3.0, hücre sayısı = 6041, algoritma zamanı = 1.075125 saniye, algoritma ram miktarı = 16.266073 megabayt	63
Şekil 5.5 Veri seti B5, 127672 nokta, yoğunluk eşiği = 5.0, hücre sayısı = 5468, algoritma zamanı = 2.110353 saniye, algoritma ram miktarı = 23.769349 megabayt	63
Şekil 5.6 Veri seti Türkiye, 200000 nokta, yoğunluk eşiği = 0.8, hücre sayısı = 6724, algoritma zamanı = 3.203463 saniye, algoritma ram miktarı = 35.097085 megabayt	64
Şekil 5.7 Veri seti Hava Kalitesi, 6373 nokta, yoğunluk eşiği = 8.0, hücre sayısı = 5609, algoritma zamanı = 0.219413 saniye, algoritma ram miktarı = 7.315873 megabayt	64
Şekil 5.8 Derinlik = 2, hücre sayısı = 626, algoritma zamanı = 1.189814 saniye, algoritma ram miktarı = 18.147149 megabayt	65
Şekil 5.9 Derinlik = 3, hücre sayısı = 1349, algoritma zamanı = 1.380306 saniye, algoritma ram miktarı = 18.992249 megabayt	65
Şekil 5.10 Derinlik = 4, hücre sayısı = 2816, algoritma zamanı = 1.628641 saniye, algoritma ram miktarı = 20.695605 megabayt	66
Şekil 5.11 Derinlik = 5, hücre sayısı = 5468, algoritma zamanı = 2.023584 saniye, algoritma ram miktarı = 23.769349 megabayt	66
Şekil 5.12 Derinlik = 6, hücre sayısı = 8789, algoritma zamanı = 2.384620 saniye, algoritma ram miktarı = 27.588289 megabayt	67
Şekil 5.13 Yoğunluk eşiği = 1, hücre sayısı = 1607, algoritma zamanı = 1.400253 saniye, algoritma ram miktarı = 19.289733 megabayt	68
Şekil 5.14 Yoğunluk eşiği = 3, hücre sayısı = 3371, algoritma zamanı = 1.670530 saniye, algoritma ram miktarı = 21.341897 megabayt	68
Şekil 5.15 Yoğunluk eşiği = 5, hücre sayısı = 5468, algoritma zamanı = 1.983692 saniye, algoritma ram miktarı = 23.769349 megabayt	69
Şekil 5.16 Yoğunluk eşiği = 7, hücre sayısı = 8498, algoritma zamanı = 2.403601 saniye, algoritma ram miktarı = 27.274405 megabayt	69
Şekil 5.17 Yoğunluk eşiği = 9, hücre sayısı = 14585, algoritma zamanı = 3.104696 saniye, algoritma ram miktarı = 34.368145 megabayt	69
Şekil 5.18 Baskınlık eşiği = 0.3, hücre sayısı = 4883, algoritma zamanı = 2.008595 saniye, algoritma ram miktarı = 23.086893 megabayt	70

Şekil 5.19 Baskınlık eşiği = 0.4, hücre sayısı = 5468, algoritma zamanı = 2.007631 saniye, algoritma ram miktarı = 23.769445 megabayt	71
Şekil 5.20 Baskınlık eşiği = 0.5, hücre sayısı = 6101, algoritma zamanı = 2.021590 saniye, algoritma ram miktarı = 24.515693 megabayt	71
Şekil 5.21 Baskınlık eşiği = 0.6, hücre sayısı = 7343, algoritma zamanı = 2.379632 saniye, algoritma ram miktarı = 25.944469 megabayt	71
Şekil 5.22 Baskınlık eşiği = 0.7, hücre sayısı = 11654, algoritma zamanı = 2.701807 saniye, algoritma ram miktarı = 30.991777 megabayt	72
Şekil 5.23 Veri seti içerisindeki nokta sayısına göre algoritma çalışma zamanının değişimi	73
Şekil 5.24 Merkez nokta sayısına göre algoritma çalışma zamanının değişimi	74
Şekil 5.25 Veri seti içerisindeki nokta sayısına göre algoritmanın kullandığı hafıza miktarı (megabayt)	76
Şekil 5.26 Merkez nokta sayısına göre algoritma hafıza kullanımı	77
Şekil 5.27 Veri seti içerisindeki nokta sayısına göre entropi	78
Şekil 5.28 Merkez nokta sayısına göre entropi değişimi	79
Şekil 5.29 Metod 1 algoritmasının 1.25° kare büyüklüğüyle gösterimi	80
Şekil 5.30 Metod 2 algoritmasının 1.25° kare büyüklüğüyle gösterimi	81
Şekil 5.31 Metod 3 algoritmasının 1.25° kare büyüklüğüyle gösterimi	81
Şekil 5.32 Önerilen metodun gösterimi Derinlik = 5, hücre sayısı = 5468	81
Şekil 5.33 Tüm metodların çalışma zamanı karşılaştırması	83
Şekil 5.34 Tüm metodların hücre sayısına göre çalışma zamanı karşılaştırması	84
Şekil 5.35 Tüm metodların hücre sayısına göre entropi karşılaştırması	85

BÖLÜM 1

GİRİŞ

Konumsal veri gerçek ya da sanal dünyadaki bir objenin ait olduğu koordinat düzleminde ifade edilmesini sağlayan matematiksel veri setidir. Konumsal veriler geçmişten günümüze mühendislik, askeriye, endüstri, oyun teknolojileri, haritalandırma gibi birçok disiplin içerisinde kullanılmıştır. Konumsal verilerin son kullanıcıya açık bir şekilde aktarılması günümüzde birçok farklı yöntemle yapılmaktadır.

GPS gibi konumsal veri üreten cihazların dünya üzerinde yaygınlaşmasıyla birlikte elde edilen çok büyük sayıda verinin kullanıcılara sunulması bir problem haline gelmiştir. Ayrıca oyun sektöründe özellikle de strateji temalı oyunlarda oyun dünyasına ait veriler oyun içerisinde haritalarda sunulmaktadır. Gerçek ortamda ya da oyun gibi sanal ortamlarda oluşan bu veriler genellikle noktasal bazda gösterilmekte ve bu durum kullanıcıya kolaylık sağlamamaktadır. Örnekeleyecek olursak on binlerce nokta içeren bir haritanın kullanıcıya nokta bazlı gösterimi kullanıcı için herhangi bir anlam ifade etmeyecektir. Bunun yerine kullanıcıya bu verilerin işlenerek ve özetlenerek harita üzerinde bölgesel bir ifade tekniğiyle sunulması daha anlaşılır olacaktır. Biz bu tezde bu problem üzerine çalışmalar gerçekleştirdik.

Özet olarak bizim amacımız çok sayıda noktaya karşılık gelen verinin kullanıcıya daha hızlı, daha net ve daha anlaşılır bir şekilde gösterilmesini sağlamaktır. Bu tez içerisinde aşağıdaki konularla ilgilenildi:

- Çok büyük konumsal verilerin harita üzerinde görselleştirilmesi
- Konumsal verilerin Voronoi diyagramına dönüştürülmesi
- Alternatif metodların karşılaştırılması ve performans analizleri

Haritalar üzerinde çok büyük konumsal veriler oluşturmak üç ana nedenden dolayı zor bir çalışma konusudur. Her şeyden önce, sınırlı ekran çözünürlükleri nedeniyle görselleştirilmiş veri noktaları birbirleriyle çakışmaktadır. Bu çakışma neticesinde de noktalar birbirlerinin üzerine kapatılmaktadırlar. İkincisi, görüntülenen veriler genellikle farklı özelliklere sahip olması gerekir. Örneğin farklı renkler ya da işaretleyiciler farklı verileri göstermek için kullanılır. Son olarak, birçok çizim aracı ve görüntü oluşturunucuda, sınırlı miktarda veri oluşturulabilir, bu nedenle bu araçlar bunlardan daha büyük verileri görüntülemekte zorlanırlar. Bu durumda, verilerin sadece sınırlı bir kısmı temsil edilir. Ortaya çıkan görüntü ise bilgi kaybı olduğu için gerçek konum verilerini doğru bir şekilde temsil edemez.



(a)



(b)

Şekil 1.1: Strateji oyunlarında harita örnekleri (a) Age of Empires II (b) Command and Conquer: Generals 2

Konumsal noktaların gösterilmesi birçok alanda kullanılmaktadır. Örneğin strateji oyunlarında oyuncuların objelerinin oyun haritası olarak ekranın genelde sağ alt ya da sol alt tarafında kullanıcıya gösterilmesi yaygındır. Bu haritaların örnekleri Şekil 1.1 içerisinde gösterilmiştir. Askeriye, web uygulamaları, coğrafi bilgi sistemleri gibi birçok alanda da verilerin harita üzerinde gösterilmesi kullanıcılara daha anlamlı veri sunma açısından önemlidir. Askeriye içerisinde harita kullanımı Şekil 1.2 içerisinde gösterilmiştir. Askeriye içerisinde özellikle hava ve deniz kuvvetlerinde objelerin (savaş uçağı, gemi vs.) konumu haritalar üzerinde gösterilmekte ve bu verilere göre strateji belirlenmektedir. Teknolojinin gelişimiyle birlikte her araca hatta her askere konumsal veri paylaşımını anlık yapan cihazlar eklenmiş ve bu sayede koordinasyonun artırılması hedeflenmiştir. Coğrafi bilgi sistemleri içerisinde teknolojinin artmasıyla birlikte özellikle belediye ve Çevre Bakanlığı gibi kurumlarda konumsal verilerin gösterilmesi yaygınlaşmıştır. Örneğin tapu bilgileri, bina verileri, kamu kurumlarının

noktasal verileri gibi birçok harita gösterimi yaygın olarak kullanılmaktadır.



Şekil 1.2: Askeri hareket merkezi harita kullanımı

Biz bu tezde kullanıcılara sunulacak olan haritanın daha anlamlı olması adına bir takım çalışmalar gerçekleştirdik. Bu tezi oluştururken temel motivasyonumuz çok sayıda nokta içeren haritaların kullanıcıya doğru, anlaşılır, az veri kaybı yaşanan ve dinamik haritalarda yenilenme zamanının makul sürelerde kaldığı bir haritalandırma sunmaktır. Son kullanıcıya sunmak istediğimiz harita insan gözünün doğada aşına olduğu Voronoi diyagramlarının verilere göre renklendirilmesiyle oluşturulacaktır. Ayrıca bu harita oluşturulurken algoritma zamanı (algorithmic time) ve bilgi karmaşıklığı (entropy) kavramlarını da göz önünde bulunduruyor olacağız. Bu çalışmanın içerisinde bizden önce yapılmış yöntemlerle kendi yöntemlerimizi kıyaslayacağız. Bilgisayar biliminin temellerinden olan hafıza kullanımı da (memory usage) göz önüne alacağımız bir diğer kavram olacaktır. Temel olarak bu çalışmada en önemli motivasyonumuz çok sayıda konumsal verinin kullanıcıya doğru ve zamanında aktarılması olacaktır.

1.1 Tanımlamalar ve Alt Yapı

Nokta, geometride boyutsuz bir eleman olarak ifade edilir. Eni, boyu ve derinliği olmayan bir terimdir. N boyutlu bir düzlemde bir nokta N adet sayıyla ifade edilebilir. Biz bu çalışmada sadece 2 boyutlu düzlemlerde çalışacağımız için noktalarımız da 2

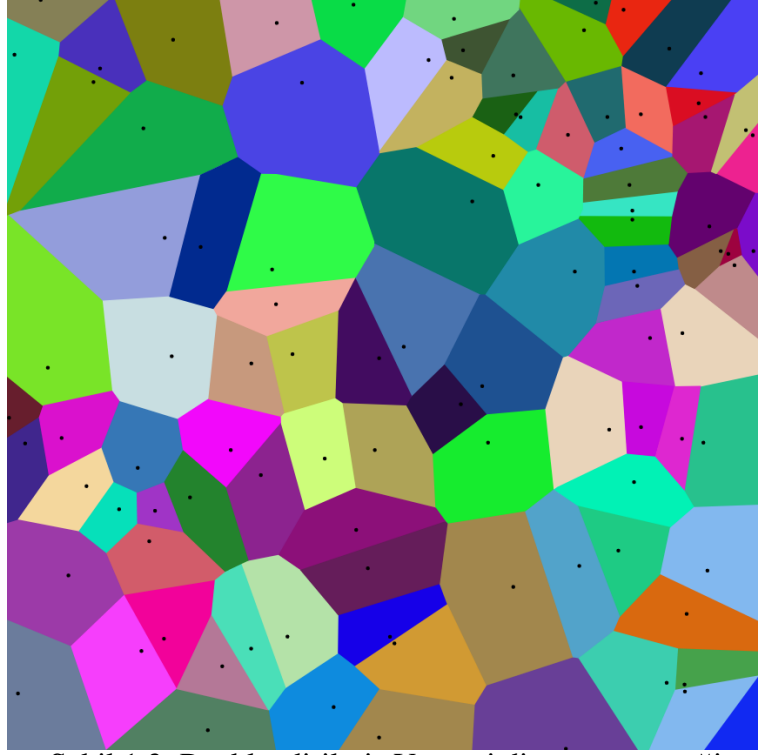
sayıyla ifade edilebilir (örn. x ve y). Nokta kümesi ise bir ya da daha çok noktanın oluşturduğu veri setine verdiğimiz isimdir. Çok sayıda nokta derken sayısı binlerle ya da milyonlarla ifade edilen nokta kümelerinden bahsetmekteyiz. Her bir noktanın ek olarak sayısal ya da renk kodu şeklinde ek verileri bulunabilmektedir. Binlerce noktanın verisiyle birlikte son kullanıcıya gösterilmesi nokta sayısı göz önünde bulundurulduğunda karmaşık gelebilmektedir. Bu sebeple bu noktaların tamamının gösterilmesi yerine belirli alanlarla ifade edilmesi yaygın kullanılan yöntemlerden biridir.

Alan matematikte sınırlı, kapalı yüzeylerin ölçüsü ya da bir şeklin düzlemde kapladığı yer miktarı olarak tanımlanır. Görselleştirmede noktalar yerine alanların kullanımı kullanıcıya daha anlamlı veriler sunmaktadır. Bu noktada asıl problem verilerin nasıl alanlara bölüneceği ve bu bölümlerin son kullanıcıya nasıl gösterileceğidir. Bölünen alanların renklendirilme tekniği ve bu tekniğin verimli bir şekilde gösterilmesi önemlidir. Ayrıca bu haritalandırma teknikleri her zaman statik haritalarda uygulanmaz. Dinamik haritalandırmada ise karşımıza bu haritanın son kullanıcıya gerçek zamanlı ya da makul sürelerde tazelenerek sunulması gibi bir problem ortaya çıkmaktadır. Bu problemle başa çıkabilmek için sunulan yöntemin makul bir sürede haritalandırmayı bitirip son kullanıcıya sunması beklenmektedir. Bu tezde çok büyük sayıdaki noktasal verilerin son kullanıcıya Voronoi diyagramı kullanılarak daha anlamlı ve yeterince hızlı bir şekilde gösterilmesi konusu ele alınmıştır.

Bölge genel olarak coğrafya içerisinde kullanılmaktadır. Matematikteki alan tanımına benzer. Coğrafyada bölgeler, fiziksel özellikler(coğrafi), insan etki özellikleri (beşeri) ve insanlık ile çevre etkileşimi vb. isimlerle anılan bölünmüş kapalı alanlardır. Coğrafi bölgeler ve alt bölgeler, ulusal sınırlar gibi yetki alanlarının hukukta tanımlandığı insan coğrafyası haricinde, çoğunlukla kesin olarak tanımlanmamış ve bazen geçici sınırlarla tanımlanmaktadır. Örneğin Voronoi diyagramı düzlemi verilen bir site kümesi için her bir siteye en yakın noktalardan oluşan bölgelere(alanlara) ayırır denilebilir.

Matematikte, bir **Voronoi diyagramı**, bir düzlemin belirli bir nokta kümesinin (merkez nokta) her birine yakın bölgelere bölünmesidir. Şekil 1.3 içerisinde bir Voronoi diyagramı örneği görülmektedir. En basit durumda, bu nesneler düzlemde (hücreler, siteler veya alanlar olarak adlandırılır) dışbükey alanları temsil eder. Şekildeki her bir

renkli alan bir hücreyi ifade etmektedir. Siyah noktalar ise diyagramın merkez noktalarını temsil etmektedir. Her hücre için, düzlemin diğer merkez noktalardan daha yakın olan tüm noktalarından oluşan karşılık gelen bir bölge vardır. Bu bölgelere Voronoi hücreleri denir.



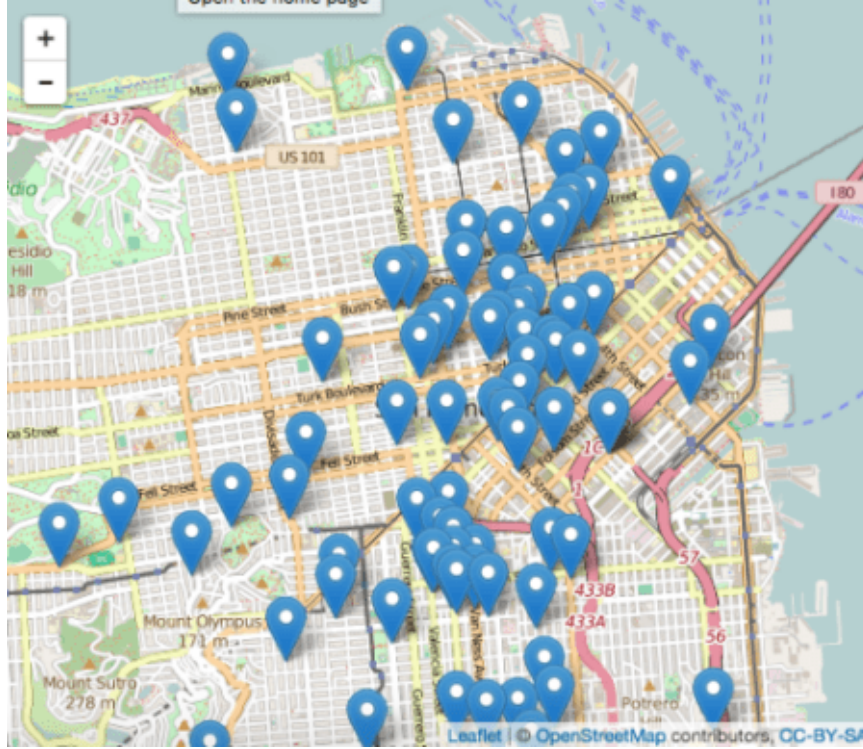
Şekil 1.3: Renklendirilmiş Voronoi diyagramı örneği

Voronoi diyagramı Georgy Voronoy'un adını taşır ve aynı zamanda Voronoi mozaikleri (tessellation), Voronoi ayrışması (decomposition), Voronoi kısımları (partition) veya Dirichlet mozaikleme (Peter Gustav Lejeune Dirichlet'e ithafen)[1] olarak da adlandırılır. Yine Voronoi hücreleri Thiessen çokgenleri olarak da bazı kaynaklarda bahsedilir[1]. Voronoi diyagramları başta bilim ve teknoloji olmak üzere görsel sanatta dahil birçok alanda pratik ve teorik uygulamalara sahiptir[2].

Veri, gözlem yapılarak toplanan ya da sentetik olarak üretilen, genellikle sayısal olan özellik ya da bilgilerdir. Daha teknik olarak, veriler bir veya daha fazla kişi, nesne ya da bilgisayar objesi hakkında nitel ya da nicel değişkenlerin bir değerler kümesidir [3].

Konumsal veri (spatial data) özel bir veri tipine verilen isimdir. Ait olduğu boyuta göre değişmekle birlikte genel olarak 2 ya da 3 boyutlu tanımlanabilir. Konum verileri

Şekil 1.4 içerisinde görüldüğü gibi ekstra bir veri içermeyebilir. Biz bu tezde ekstra veri içeren konumsal veri setleri üzerinde çalışmalar yürüteceğiz. Bu ekstra veriler iki tip olabileceği gibi (kırmızı,yeşil) çok farklı tiplerde de (örn. bir kaç farklı renk) olabilir. Ayrıca bu ekstra veriler sürekli (continuous) veriler şeklinde de görülebilmektedir. Bu tezde konumsal verilerin içerisindeki ekstra veriler iki tip olacak şekilde çalışma yapılmıştır.



Şekil 1.4: Ekstra veri içermeyen konum veri seti örneği

Küme analizi (clustering) veya kümeleme, bir grup nesnenin birbirine daha yakın nesneler aynı grupta olacak şekilde gruplanması olarak ifade edilebilir. Veri madenciliğinin ana konularından biridir. Örüntü tanıma, görüntü analizi, bilgi değerlendirme, biyoinformatik, veri sıkıştırma, bilgisayar grafikleri ve makine öğrenimi gibi birçok alanda kullanılan ortak bir tekniktir. İstatistiksel veri analizi için kullanılır[4].

Bölütleme (segmentasyon) çoğu zaman görüntü işlemenin temel aşamalarındandır. Resim bölütleme, resmi her bir alan içerisinde farklı özelliklerin barındırıldığı daha mantıklı alt kümelere bölmek olarak tanımlanabilir. Bölütleme her zaman resim verisi üzerinde uygulanmak zorunda değildir. Örneğin, resim içerisindeki bulunan benzer parlaklıklar ya da karanlık kısımlar bulunabilir, bu kısımlar aynı görüntünün deği-

şik alanlarındaki aynı objeyi ifade edebilirler. Örnekeleyecek olursak; uydu fotoğraflarında, hareketli araçlar ve arka planı yoldan ayırmak amacıyla bölütleme uygulanabilir. Bütün benzer problemlere uygulanabilecek evrensel bir bölütleme algoritması bulunmamaktadır ve çoğu bölütleme yöntemi her zaman istenen sonuçları veremez [5].

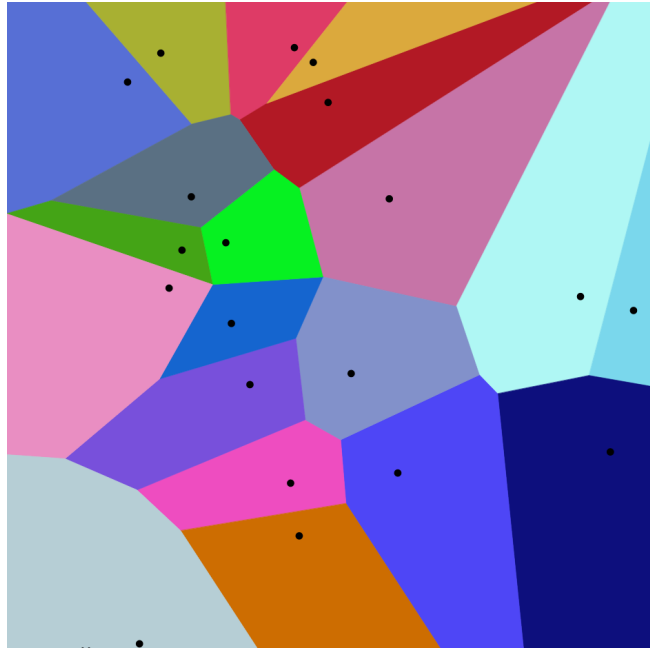
Haritalandırma (mapping) Dünyanın (ya da başka gezegenlerin) tümünün ya da belirli bir parçasının kısmi oranlarda küçültülüp (ölçeklendirilip) düzlem üzerinde ifade edilmesidir[6]. Genel olarak haritalandırma izdüşümü üzerinden yapılır. Haritalandırma sadece gerçek verileri değil oyun ya da veri tabanında yer alan kullanıcı ya da bilgisayar tarafından oluşturulan sanal verilerin görselleştirilmesiyle de yapılabilir. Genel olarak haritalandırma sayısal ya da sayısal olmayan verilerin son kullanıcıya bir resim şeklinde sunulması olarak tanımlanabilir. Ayrıca haritalandırma eski zamanlardan beri üzerinde çalışılan konulardan biridir. Askeriye, coğrafya, şehir planlaması, bilgisayar bilimi gibi birçok disiplin içerisinde önemli yer tutmaktadır.

Entropi genellikle karmaşıklık değeri olarak ifade edilir. Fizik, kimya, matematik, biyoloji gibi birçok disiplin içerisindeki hesaplamalarda kullanılabilir. Bir grup nesne içerisindeki karmaşıklık grubun homojenlik oranı olarak tarif edilebilir. Grupta bulunan nesneler ne kadar eş dağılırsa, grubun entropi miktarı da o oranda artmaktadır. Bazı alanlarda her ne kadar entropinin artması amaçlansa da bu çalışmada biz entropinin yani karmaşıklığın azaltılmasını ve kullanıcıya bu sayede daha anlamlı veri sunulmasını amaçlıyor olacağız.

BÖLÜM 2

LİTERATÜR ARAŞTIRMALARI

Voronoi diyagramları, kendisini tanımlayan ve üzerinde çalışmalar gerçekleştiren Georgy Feodosevich Voronoy'un adını almıştır. Kendisi 1908'de yaptığı çalışmada Voronoi diyagramlarını matematiksel bir şekilde ifade edebilmiştir[7]. Voronoi diyagramları sonsuz bir alanın daha önceden verilmiş noktalar etrafında eşit uzaklıklarla bölünmesi sonucu elde edilen geometrik bir tanımlamadır. Noktalardan elde edilen her bir bölüme hücre adı verilir. Bir hücre alanındaki tüm noktalar o hücreyi merkez alan noktaya diğer merkez noktalardan daha yakındır. Şekil 2.1 ve Şekil 2.3'de görüldüğü üzere siyah noktalar merkez noktaları temsil etmektedir. Renkli alanlar merkez noktaya karşılık gelen Voronoi hücreleridir.



Şekil 2.1: Öklid uzaklık ölçüsüne göre Voronoi diyagramı[8]

Minkowski uzaklığı ya da Minkowski metriği, hem Öklid uzaklığının hem Manhattan uzaklığının bir genellemesi olarak düşünülebilecek vektör uzayındaki bir metriktir. Adını Alman matematikçi Hermann Minkowski'den almıştır[9]. Buna göre p boyutlu bir uzayda X ve Y noktaları verildiğinde $D(x, y)$ yani uzaklık Şekil 2.2 içerisindeki formüle göre hesaplanabilir.

$$D(X, Y) = \left(\sum_{i=1}^n |x_i - y_i|^p \right)^{\frac{1}{p}}$$

Şekil 2.2: p boyutlu uzayda Minkowski uzaklığı hesaplaması [9]

Öklid uzaklığını matematiksel olarak ifade etmek gerekirse uzaklık formülü olarak $d = \sqrt{(|x_1 - x_2|)^2 + (|y_1 - y_2|)^2}$ ifadesini kullanabiliriz. Manhattan uzaklığı ise $d = |x_1 - x_2| + |y_1 - y_2|$ şeklinde ifade edilebilir.

Yaygın olarak, Voronoi diyagramları noktalar arasındaki Öklid mesafesini kullanır (Şekil 2.1), ancak Manhattan mesafesi gibi diğer mesafe ölçümleri de mümkündür (Şekil 2.3). Öklid mesafesi kullanıldığında bölgelerin kenarları merkez noktalarından gelen uzantıların dik açıortaylarıdır. Bu tezde, Voronoi diyagramlarını oluşturmak için Öklid uzaklık metriği kullanılmıştır.



Şekil 2.3: Manhattan uzaklık ölçüsüne göre Voronoi diyagramı[8]

Daha matematiksel ve genel olarak, P_k noktası ile tanımlanan Voronoi bölgesi R_k , P_k 'ye olan uzaklığı P_j 'ye olan uzaklığından daha büyük olmayan alanı ifade etmektedir. Burada j, k'dan farklı herhangi bir indekstir. Matematiksel olarak bu alanı aşağıdaki şekilde ifade edebiliriz:[8]

$$R_k = \{x \in X | d(x, P_k) \leq d(x, P_j)\}$$



Şekil 2.4: Zürafa üzerinde yer alan Voronoi diyagramı benzeri şekiller[8]

Voronoi diyagramları özellikle doğal hayatta birçok farklı alanda karşımıza çıkmaktadır. Örnek vermek gerekirse, zürafalar üzerindeki desenler (Şekil 2.4), dokuların yapraklarda oluşturduğu şekiller (Şekil 2.5) veya kurutulmuş toprağın çatlaması şeklinde (Şekil 2.6) doğadaki Voronoi benzeri şekilleri görebiliriz.



Şekil 2.5: Yaprak üzerindeki dokuların oluşturduğu Voronoi diyagramı benzeri şekiller[8]

Doğada gördüğümüz Voronoi benzeri oluşumlar her ne kadar mükemmel Voronoi diyagramları olmasa da, doğada hiçbir geometrik şeklin mükemmel bir yansıması olmayacağı aşıkardır. Doğada gördüğümüz bu şekillerde kenarlarda düzensizlikler olabilir ve hatta bazı fazladan çizgiler yer alabilirler. Ancak Voronoi diyagramlarının deseni doğa üzerinde yer alan birçok obje üzerinde bu hatalara rağmen açıkça görülebilir. Bir mağarada ya da kayanın kenarlarında da görülebilen bu şekiller insan gözünün aşına olduğu daha önceki görsel hafızasında yer alan şekillerdir. Temel olarak bu çalışmada Voronoi diyagramları kullanılarak son kullanıcıya haritalandırma gösterimi yapılmasının gerekçesi budur.



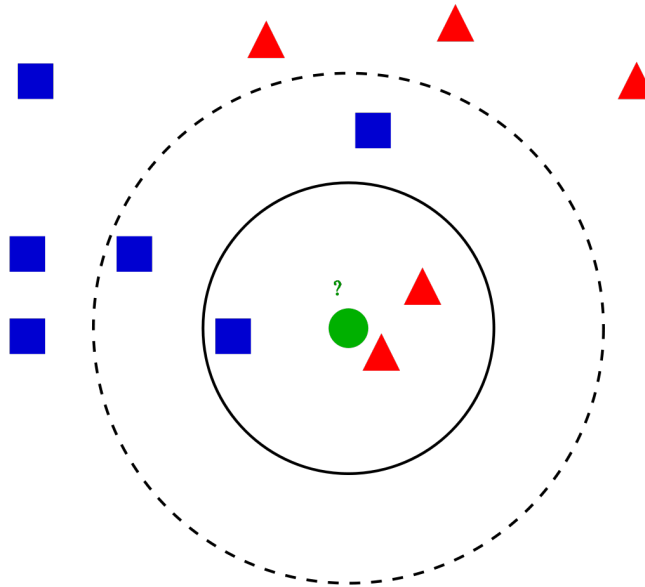
Şekil 2.6: Kurumuş toprağın oluşturduğu Voronoi diyagramı benzeri şekiller[8]

Fortune algoritması, $O(n \log(n))$ zamanı içerisinde çalışan ve Voronoi diyagramlarının n adet merkez nokta verildiğinde oluşturulmasını sağlayan bir algoritmadır. Steven Fortune tarafından 1987 yılında oluşturulmuştur. Fortune 1987 yılında “A Sweep-line Algorithm for Voronoi Diagrams” ismiyle bir de makale yayınlamıştır[10].

Çalışmamızda birçok en yakın noktayı bulma işlemi gerçekleştirilmektedir. Örnek vermek gerekirse verilen bir k noktasının, N adet nokta içeren bir küme içerisinde hangi noktaya en yakın olduğu sorusu çalışmamız içerisindeki en temel konulardan bir tanesidir. En temel düşünceyle N adet noktanın tamamına bakılıp, k noktasının en yakın olduğu nokta bulunabilir. Fakat küme içerisinde çok sayıda nokta olduğu göz önünde bulundurulduğunda ve bu sorgulamanın da çok sayıda yapılacağı düşünüldüğünde $O(N)$ zaman karmaşıklığına sahip bu yöntem bizim çalışmamız açısından verimli değildir. Daha önceki kısımlarda bahsettiğimiz zamanlama problemi (özellikle dinamik

veri içeren haritalandırmalarda) $O(N)$ zaman karmaşıklığına sahip bir algoritmayla çözülemeyecektir. Bu sebeple daha düşük zaman karmaşıklıklarında çözüm üreten bir “En yakın nokta bulma algoritması” ihtiyacımız mevcuttur. Bu sebeple daha az zaman karmaşıklığına sahip çalışmaları inceledik.

Görüntü işleme ve diğer benzer disiplinlerde, k-en yakın komşu (KNN) algoritması, sınıflandırma ya da kümeleme için yaygın kullanılan yöntemlerden birisidir. Thomas Cover 1992 yılında bu algoritma üzerinde parametrik olmayan çalışmalar gerçekleştirmiştir[11]. Bu yöntemde girdi olarak daha önceden belirlenmiş noktalar kullanılır. Bu noktalar farklı veri tiplerine (kümelerine) ait noktalar olarak tanımlanabilir. Yeni bir nokta verildiğinde bu noktaya en yakın k adet noktaya bakılır ve yeni noktanın hangi sınıfa ait olduğu belirlenmiş olur. Yeni nokta eğer istenirse sisteme eklenebilir. Şekil 2.7 içerisinde KNN algoritmasının çalışma mantığı gösterilmiştir[12].



Şekil 2.7: KNN algoritması çalışma mantığı[12]

KNN algoritması için çıktı bir noktanın ait olduğu sınıftır. Bir nokta, komşularının yakınlığına göre sınıflandırılır. Buna göre k (k , pozitif bir tam sayıdır ve genel olarak küçük bir sayı seçilir) adet en yakın nokta içerisinde hangi sınıf ağırlıklıysa o nokta artık o sınıf ait olmaktadır. Bu işlem sonucunda yeni nokta o sınıfa atanmış olur. Eğer $k = 1$ seçilirse, nokta en yakın komşu noktanın sınıfına dahil edilir[13]. KNN algoritmasının zaman karmaşıklığı $O(nm)$ olarak ifade edilir. Bu ifadede n KNN içerisinde yer alan noktaların sayısı, m ise KNN’de sorgulanan noktaların sayısını göstermekte-

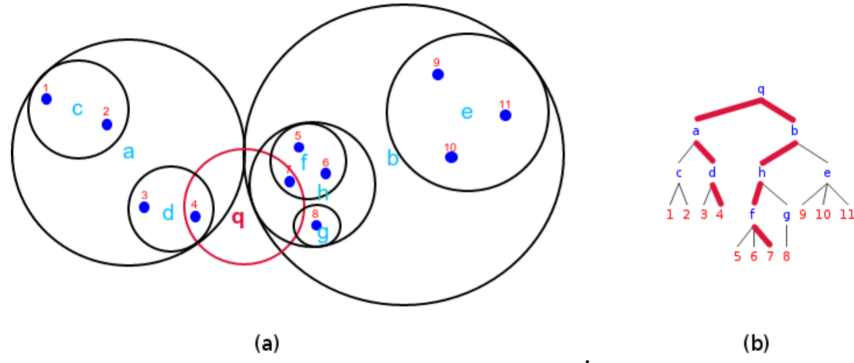
dir. Bu zaman karmaşıklığı bir kaç strateji ile düşürülebilir.

KNN algoritması, sonucun sadece yerel olarak belirlendiği bir tahminleme algoritması olarak kabul edilebilir. Bu algoritma sınıflandırma için sadece mesafeye dayandığından, eğitim verilerini düzenlemek (yani gürültülerden arındırmak), doğruluğunu önemli ölçüde artırabilir[14]. Ayrıca sadece yerel dinamiklere bakarak bir sonuca varmaya çalışması bu algoritmanın zayıf yanlarından birisidir.

KNN algoritması bizim problemimiz için bir çözüm oluşturabilir. Örneğin veri seti içerisindeki noktalar KNN'e eklenebilir. Daha sonrasında haritada gösterilecek olan noktalar da (piksel olarak düşünülebilir) KNN içerisinde sorgulanarak renklendirme sağlanabilir. Fakat bu yöntem hem yerel değerlerle çalıştığı için hem de yüksek zaman karmaşıklığına sahip olduğu için tercih edilmemiştir.

KNN hem sınıflandırma hem de regresyon için yararlı bir algoritma olarak kabul edilebilir. Bu algoritma üzerinde çok sayıda ek çalışma yapılmış ve bu algoritma temel alınarak yeni algoritmalar oluşturulmuştur. Komşuların sınıflandırmaya olan katkıları geliştirilebilir. Böylece daha yakın komşular, sınıflandırmaya daha uzak olanlardan daha fazla katkıda bulunabilir. Örneğin, noktaların mesafesine göre bir ağırlıklandırma metodu eklenebilir. Her komşuya $1/d$ (d iki nokta arasındaki mesafe) kadar bir ağırlık verilmesi daha sağlıklı sonuçlar elde edilmesini sağlayabilir[15].

Ball-tree algoritması 1989 yılında Stephen Omohundro tarafından çalışılmış bir yakınlık bulma algoritmasıdır. Bu algoritmaya göre daha önceden verilen noktalar belirli çemberlerin içerisine yerleştirilmektedir[16]. Bu yerleşmenin sonucunda ortaya çıkan hiyerarşik ağaçla birlikte yeni verilecek olan bir noktanın hangi noktaya en yakın olduğu $O(\log n)$ zaman karmaşıklığı içerisinde bulunabilmektedir. Şekil 2.8 içerisinde çemberler ve ortaya çıkan ağaç görülebilir[17].



Şekil 2.8: Ball-tree yapısı (a) Çemberler (b) İlişkili ağaç yapısı[17]

Çok sayıda ball-tree oluşturma algoritması daha önceden çalışılmıştır[16]. Bu algoritmanın amacı verilen bir noktaya en yakın n adet noktanın hızlı bir şekilde bulunmasını sağlamaktır. Genel yöntemlerin aksine bu yöntem arama algoritmaları arasında $O(\log n)$ zamanında çalışan algoritmalarından bir tanesidir. Bu aramanın verimli olması için oluşturulan ağacın dengeli olması gerekmektedir. Bununla birlikte, verimli bir ağacın genel olarak tanımı en çok düğüm sayısı ile en az derinlik oluşturması şeklinde tanımlanabilir. Gerçek dünya veri setlerinin çeşitli varyasyonları göz önünde bulundurulduğunda, bu bir algoritma tasarımı için zor bir süreçtir. Ancak verileri uygulamada iyi bir şekilde düğümlere ayıran bir algoritma olarak ball-tree öne çıkmaktadır. Genel olarak, bir ağaç oluşturma zamanı ile arama yapma zamanı arasında bu algoritma içerisinde elde edilen bir denge mevcuttur[18]. Bu denge de bu yapıyı bizim için kullanılabilir bir hale getirmiştir. Voronoi diyagramlarının renklendirilmesi konusunda bu çalışma içerisinde ball-tree algoritması kullanılmıştır.

```
function construct_balltree is
  input:  $D$ , an array of data points.
  output:  $B$ , the root of a constructed ball tree.

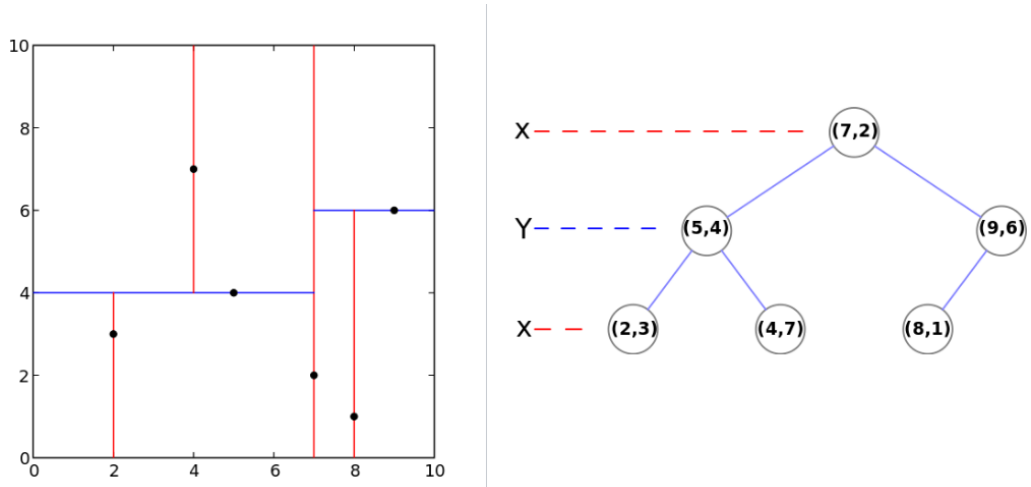
  if a single point remains then
    create a leaf  $B$  containing the single point in  $D$ 
    return  $B$ 
  else
    let  $c$  be the dimension of greatest spread
    let  $p$  be the central point selected considering  $c$ 
    let  $L, R$  be the sets of points lying to the left and right of the median along dimension  $c$ 
    create  $B$  with two children:
       $B.pivot := p$ 
       $B.child1 := \text{construct\_balltree}(L)$ ,
       $B.child2 := \text{construct\_balltree}(R)$ ,
      let  $B.radius$  be maximum distance from  $p$  among children
    return  $B$ 
  end if
end function
```

Şekil 2.9: Ball-tree algoritması[16]

Ball-tree algoritması Şekil 2.9 içerisinde görüldüğü gibi özetlenebilir.

Diğer birkaç veri yapısı ile karşılaştırıldığında, Ball-tree yapılarının, özellikle boyutlarının sayısı arttıkça, en yakın komşu arama probleminde oldukça iyi performans gösterdikleri gösterilmiştir[19]. Bununla birlikte, belirli bir uygulama için en yakın komşu algoritmasının veri yapısı, verilerin kaç boyutlu olduğuna, veri noktalarının toplam büyüklüğüne ve temeldeki veri yapısına bağlı olarak değişkenlik gösterebilir[20]. Bu durumda problemin ve problemde kullanılan veri setlerinin özellikleri göz önünde bulundurulmalıdır. En yakın komşu arama algoritması bu özelliklere göre seçilmelidir.

Kd-tree algoritması 1975 yılında Jon Louis Bentley tarafından oluşturulmuş bir yakınlık bulma algoritmasıdır[21]. Her ne kadar farklı kullanım alanları da olsa bu yapı bu araştırma içerisinde en yakın nokta bulma problemi çerçevesinden incelenmiştir. Bu yapıda, her düğüm k-boyutlu bir noktanın içerisinde olduğu bir elemanı barındırır. Yapı ikili bir ağaç formasyonunda organize edilmiştir. Şekil 2.10 içerisinde bu yapının nasıl oluşturulduğu gösterilmektedir[22].

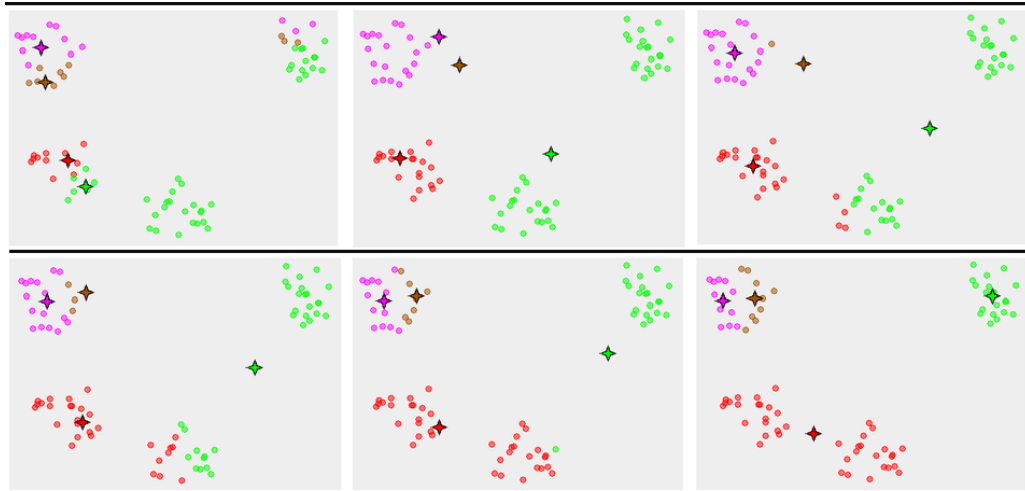


Şekil 2.10: Kd-tree yapısı [22]

Kd-tree algoritması her elemanında bir nokta barındıran ikili bir ağaçtır. Yaprak olmayan her düğüm bir bölmeyle alanı ikiye ayırır. Bu alanlara yarı boşluk denir. Bu bölmenin sol tarafında yer alan noktalar, o düğümün sol alt ağacıyla temsil edilir ve bu doğrunun sağındaki noktalar ise sağ alt ağaçla temsil edilir. Bu doğruların boyutu şu şekilde seçilir: ağaçtaki her düğüm, k boyutlarından biriyle ilişkilendirilir (eğer 2

boyut varsa derinlik x ve y şeklinde ilerler bkz. Şekil 2.10) ve diğer doğru bu doğruya dik olacak bir şekilde ayarlanmış olur. Dolayısıyla, örneğin belirli bir bölünme için " x " eksenini seçilirse, alt ağaçtaki düğümden daha küçük " x " değerine sahip tüm noktalar sol alt ağaçta görünecek ve daha büyük " x " değerine sahip tüm noktalar sağ alt ağaçta bulunabilecektir. Kısaca ağaç yapısında arama yapmak istendiğinde her seviyede sadece tek boyut için karşılaştırma yapılacaktır. Böyle bir arama içerisinde, noktanın x ve y değerleri karşılaştırılarak en yakın nokta ağaç içerisinde bulunabilecektir[23].

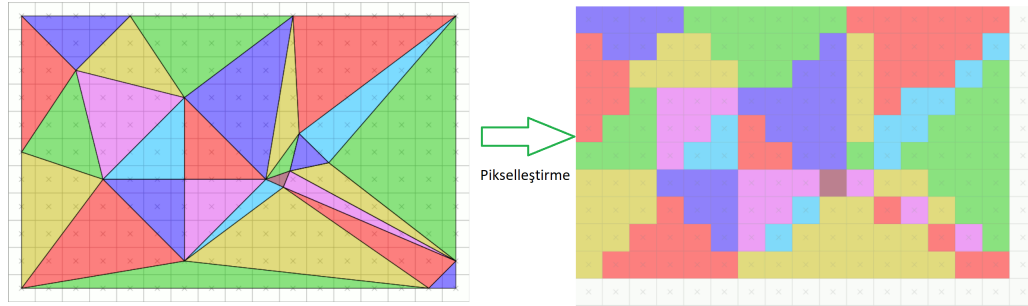
Geleneksel K-means kümeleme ve yakınlık algoritması MacQueen tarafından ilk olarak 1967 yılında çalışılmıştır[24]. K-means, literatürde ve endüstriyel hayatta en çok karşımıza çıkan ve yaygın olarak kullanılan kümeleme algoritmalarından birisidir. Algoritma, kümeleri sayısal veriler olarak ele alır ve her kümenin ortalama da denilen bir merkezini oluşturur[25]. İlk olarak, K sayısının nokta başlangıçtaki merkezleri oluşturur ve küme sayısı belirlenmiş olur. Kalan diğer noktalar, mesafe ölçümüne bağlı bir hata fonksiyonunun kullanılması sayesinde kendisine en yakın kümeye eklenir. Ardından, tüm bu yapı tekrar oluşturularak güncellenir. Algoritma tekrarlı bir şekilde bu işleme devam eder. En son hata oranı belirli bir oranın altına düştüğünde bu işlem sonlandırılır ve yapı nihai halini almış olur.



Şekil 2.11: K-means algoritması ile kümeleme aşamaları[26]

K-means yapısı iki adımda özetlenebilir: başlangıç durumu ve yineleme durumu. K-means algoritmasındaki kümelerin oluşturulması aşaması ve tekrar güncelleştirilmesi aşamasında tüm yapının tekrar hesaplanması gerekmektedir. Her ne kadar küçük veri

setleri için kullanılsa da K-means algoritması büyük veri kümeleri için de etkilidir ve kabul edilebilir zamanlarda çalışabilir. Fakat K-means algoritmasının bazı eksik yanları mevcuttur. Algoritmanın çalışma performansı (zaman ve verimliliği) başlangıçta seçilen noktalarla ilişkilidir. İlk seçilen ağırlık merkezleri (yani noktalar) iyi değilse, nesneler verimsiz şekilde oluşabilmektedir. Yanlış noktaların seçilmesi sonucunda noktalar her bir yinelemede (iterasyon) tekrar tekrar taşınacaktır[27]. Bu tezde her ne kadar K-means kümeleme yöntemi kullanılabilir olsa da yöntem verimsizliği ve algoritma üzerinde çok fazla değişiklik oluşturulması gerekliliğinden (bizim problemimize uygun hale gelmesi için), bu tez içerisinde bu algoritma kullanılmamıştır. Şekil 2.11 içerisinde K-means algoritması kullanılarak oluşturulmuş kümeler aşama aşama görülebilir[26].



Şekil 2.12: Pikselleştirme örneği

Pikselleştirme (rasterisation) bu çalışmada ele alınan problem ve benzeri problemler için ilk akla gelen yöntemlerden birisidir. Pikselleştirme, vektörel bir resim objesi üzerinde ya da konumsal veri içeren veri setleri üzerinde uygulanır. İlgili veri seti öncelikle belirlenen sayıda ve büyüklükte karelere bölünür. Daha sonra bu bölümlendirme içerisinde çoğunluk olan verinin (ya da vektörel resimlerde alanın) rengi o karenin rengi olarak atanmış olur. Şekil 2.12 içerisinde bir pikselleştirme örneği görülmektedir.

Michael F. Worboys, 1994 yılında pikselleştirme konusunda bazı çalışmalar gerçekleştirmiştir[28]. Bu çalışmaya göre grafik (genellikle vektörel) formatında şekillerle açıklanan bir görüntüyü alıp bir tarama görüntüsüne (birlikte görüntülendiğinde, temsil edilen görüntüyü oluşturan bir dizi piksel, nokta veya çizgi) dönüştürme pikselleştirme olarak adlandırılmıştır. Pikselleştirilmiş görüntü daha sonra bir bilgisayar ekranında, video ekranında veya yazıcıda görüntülenebilir[29]. Oluşan bu harita (ya da

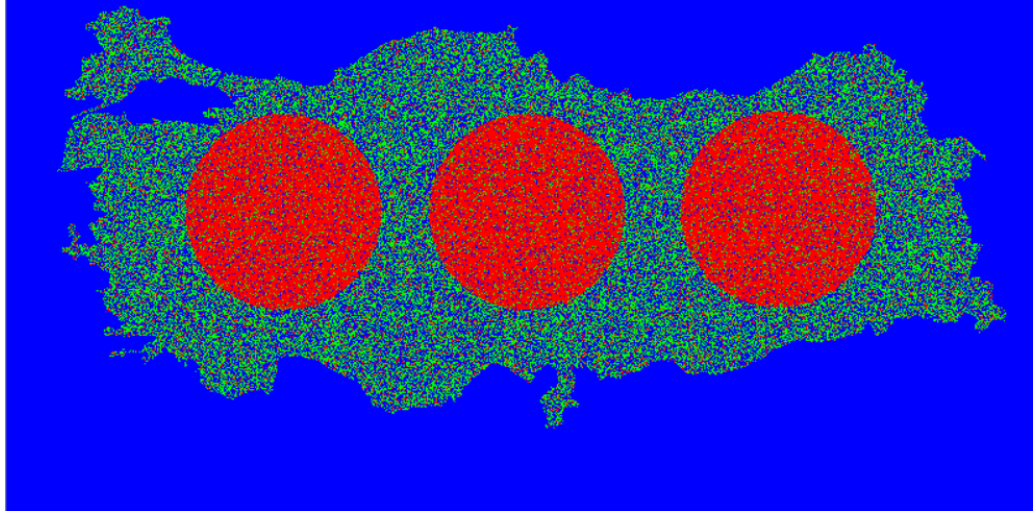
bitmap) belirli bir dosya formatında (png,bitmap vb.) saklanabilir. Pikselleştirme, 3 boyutlu model çizme uygulamalarında ya da basit 2 boyutlu resim gösterme uygulamalarında sıkça kullanılmaktadır.

3 boyutlu modellemelerin genellikle kullandığı yapı poligonaldır. Bu poligonlar genellikle 3 boyutlu üçgenler şeklinde temsil edilirler. 3 boyutlu objelerin 2 boyutlu ekranlar üzerinde izdüşümünün oluşturulması ise bilgisayar grafiği alanının yaygın bir araştırma konusudur. Pikselleştirmeden önce 3 boyutlu poligonlar birtakım tekniklerle 2 boyutlu alanlara aktarılır. Daha sonrasında ekran çözünürlüğü göz önünde bulundurularak pikselleştirme yapılır[30].

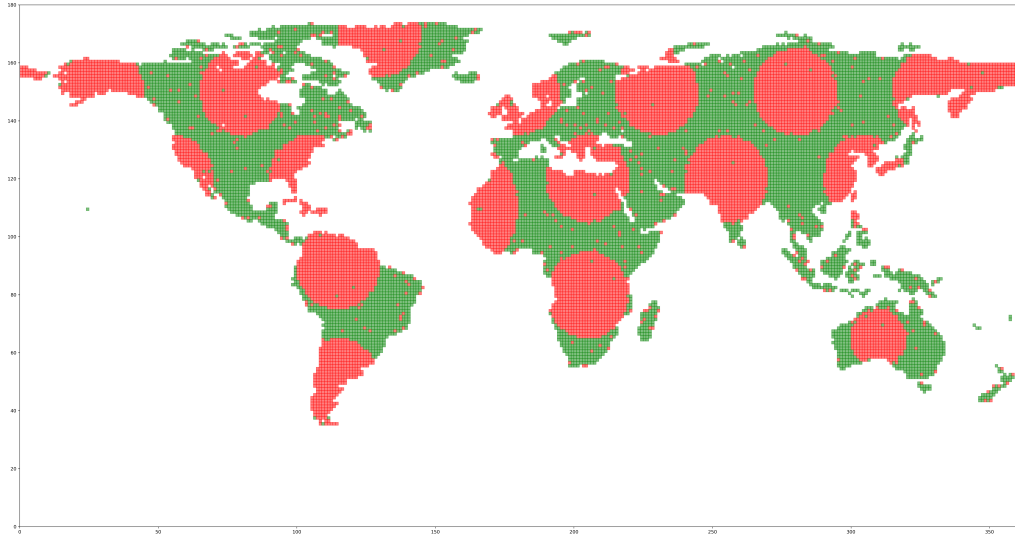
Kırmızı:51 Yeşil:49	Kırmızı:51 Yeşil:49	Kırmızı:51 Yeşil:49
Kırmızı:51 Yeşil:49	Kırmızı:1 Yeşil:99	Kırmızı:51 Yeşil:49
Kırmızı:51 Yeşil:49	Kırmızı:51 Yeşil:49	Kırmızı:51 Yeşil:49

Şekil 2.13: 9 hücreli basit bir pikselleştirme örneği

Pikselleştirme her ne kadar basit ve hızlı bir çözüm sunsa da bazı problemleri de beraberinde getirmektedir. Öncelikle pikselleştirme veri kaybına sebep olmaktadır. Şekil 2.13 içerisinde bir pikselleştirme örneği görülmektedir. Bu örneğe göre veri seti içerisinde yer alan toplam 900 veri 9 farklı hücreye bölünmüştür. Veriler incelendiğinde 409 kırmızı 491 yeşil veri olduğu görülmektedir. Söz konusu verilerde yeşil verilerin baskınlığı kırmızı verilerden daha fazla olduğu halde son kullanıcı 8 kırmızı ve 1 yeşil alan görmektedir. Bu durum ise kullanıcıya hatalı bir gösterim sunmaktadır. Bu sebeple bu çalışmada pikselleştirme metodu kullanılmamıştır.

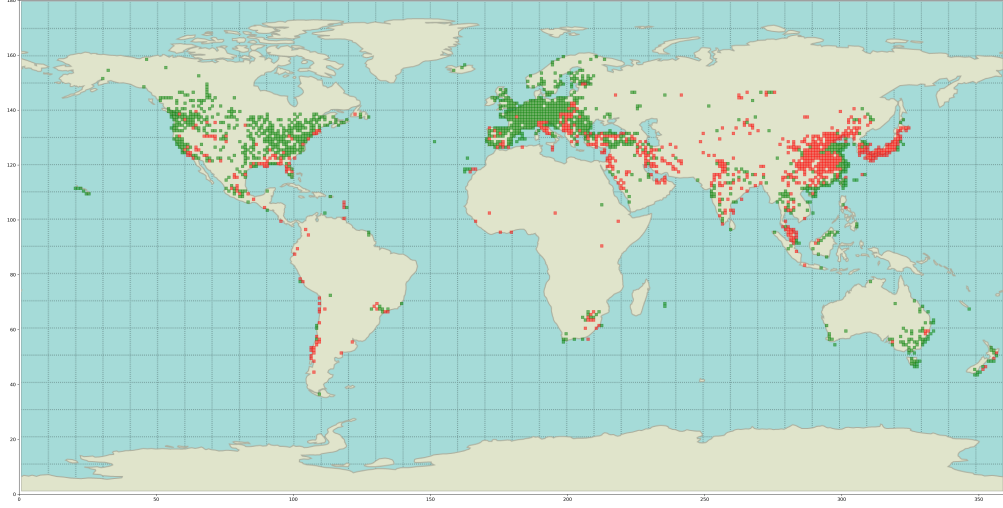


Şekil 2.14: Türkiye veri seti pikselleştirme örneği



Şekil 2.15: B5 veri seti pikselleştirme örneği

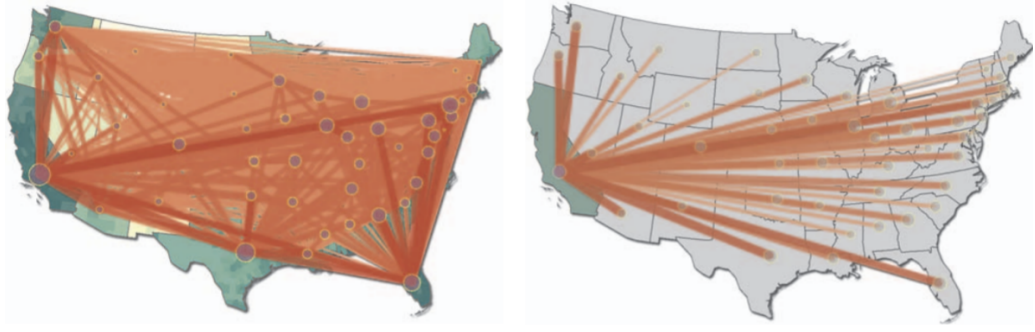
Bu çalışmada kullanılan veri setlerinin pikselleştirme sonuçları Şekil 2.14,2.15 ve 2.16 içerisinde gösterilmektedir. Çalışmada kare boyutu $1 * 1$ olarak seçilmiştir. Bu gösterimler bu çalışmanın nihai sonucuyla verilmiş olan bu görseller karşılaştırılabilir. Ayrıca pikselleştirme sonucu son kullanıcıya sunulan veri insan doğasına Voronoi diyagramlarında olduğu kadar da uygun değildir. Bunun yanında pikselleştirmede çalışma süresinin yüksekliği ve her bir görüntüleme seviyesinde tekrar hesaplama zorluğu da pikselleştirmeyi bu problem için etkin bir çözüm yöntemi kılmamaktadır.



Şekil 2.16: Hava kalitesi veri seti pikselleştirme örneği

Yapılan bu çalışma, 2017 yılında Hacettepe Üniversitesinde Bilgehan Çakmak tarafından hazırlanan “ÇOK BÜYÜK KONUMSAL VERİNİN GÖRSELLEŞTİRİLMESİNDE KARMAŞIKLIK YÖNETİMİ” isimli tez çalışması baz alınarak oluşturulmuştur[31]. Çakmak bu çalışmada 5 adet veri seti hazırlamıştır. Bu veri setleri bizim çalışmalarımızda da kullanılmıştır. Çakmak’ın tez çalışmasında K-Means Sınıflandırma, K-Medoids Sınıflandırma, K-Medians Sınıflandırma SLINK, CLINK, BIRCH, DBSCAN gibi birçok çalışma incelenmiştir.

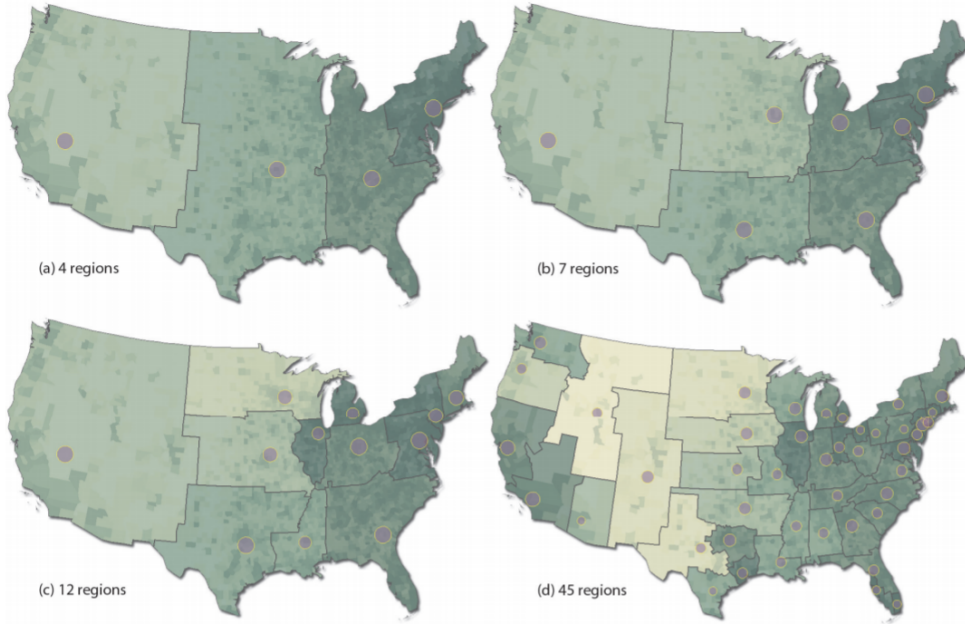
Çakmak tarafından oluşturulan çalışmada bizim problemimizin çözümüne yönelik 3 adet çözüm yöntemi sunulmuştur. İlk yöntem pikselleştirme yöntemiyle benzerlik göstermektedir. Buna göre sabit kare boyutlarında veri seti bölünmüş ve bu boyutlara göre bir pikselleştirme uygulanmıştır. Çakmak’ın çalışması içerisinde oluşturulan ikinci yöntemde göre kare boyutları daha adaptif bir şekilde oluşturulmuştur. Buna göre bir karenin içerisinde eğer belirli bir entropi değeri sağlanamıyorsa o kare daha küçük karelere bölünmektedir. Çakmak’ın çalışması içerisinde oluşturulan üçüncü yöntemde göre ise kareler bölünmelerini dört farklı kare ile değil de belirli dikdörtgenlere göre oluşturmaktadır. Buna göre bir karenin içerisinde eğer belirli bir entropi değeri sağlanamıyorsa o kare daha küçük karelere değil entropi hesaplamalarına göre daha küçük dikdörtgenlere bölünmektedir. Eğer belirli bir entropi değeri sağlandıysa o kare (ya da dikdörtgen) daha küçük dikdörtgenlere bölünmeden bırakılmaktadır.



Şekil 2.17: Amerikan iç göç haritası (tüm vektörlerin gösterimi)[32]

Diansheng Guo tarafından 2009 yılında yapılan çalışmada Amerikan iç göç verileri incelenmiş ve bu verilerin son kullanıcıya nasıl sunulabileceği üzerinde çözüm yöntemleri çalışılmıştır[32]. Bu çalışmada çözülmeye çalışılan problem sekiz yüz bin vektörel verinin (bir şehirden diğerine göç) kullanıcıya anlamlı bir şekilde sunulamasıdır. Şekil 2.17 içerisinde görüldüğü üzere bu verinin tamamının son kullanıcıya sunulması ciddi bir karmaşa meydana getirmektedir.

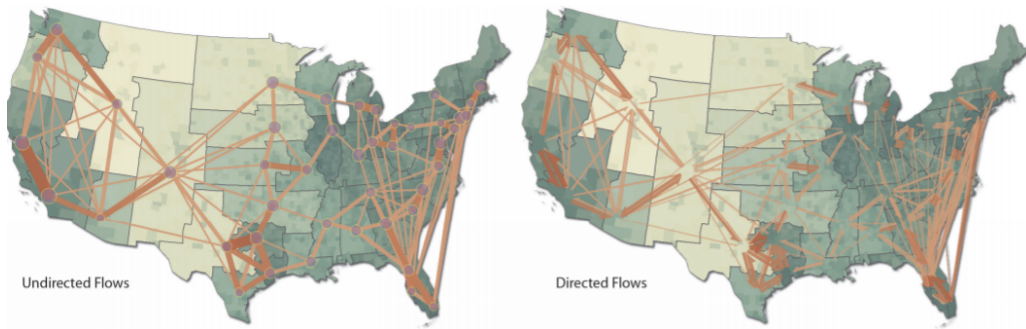
GUO: FLOW MAPPING AND MULTIVARIATE VISUALIZATION OF LARGE SPATIAL INTERACTION DATA



Şekil 2.18: Amerikan iç göç haritası (birleştirilen düğümler)[32]

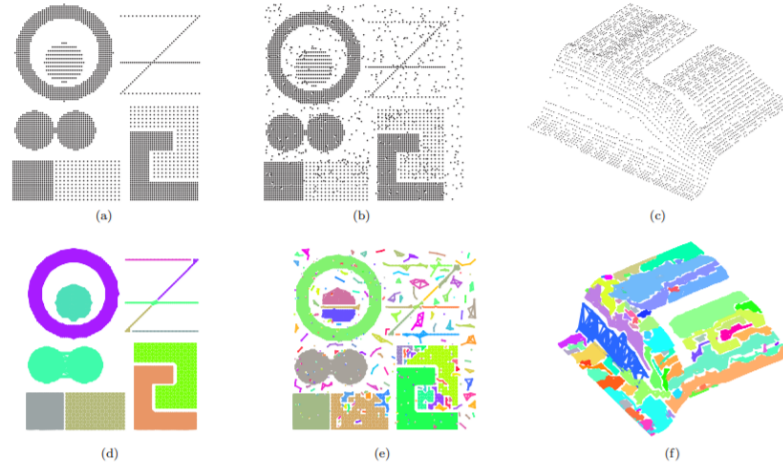
Diansheng Guo tarafından yapılan çalışmada bu veriler daha büyük düğümler elde edilerek daha anlamlı hale getirilmiştir. Şekil 2.18 içerisinde bu çözüm yönteminde kullanılan düğümler gösterilmiştir. Daha sonrasında vektörel veriler bu düğümlere

göre birleştirilmiş ve son kullanıcıya daha anlamlı bir görsel sunulması amaçlanmıştır. Bu birleştirme sırasında bir araya getirilen düğümlerdeki vektörlerin başlangıç ve bitiş noktaları güncellenerek tekrar ayarlanmıştır. Şekil 2.19 içerisinde vektörler düğümlerin birleştirilmesi sonucu tekrar gösterilmiştir. Şekillerden anlaşılacağı kadarıyla bu yöntemle oluşturulan gösterim son kullanıcıya daha anlaşılır bir gösterim sunmaktadır. Ayrıca araştırmacılar görselin filtrelenmesi (yaş, cinsiyet, meslek vb.) ve düğümlerin bir araya getirilebilmesi için bir araç geliştirmişlerdir. Bu çalışma sonucunda Amerika iç göç durumu incelendiğinde merkezden kıyılara doğru bir iç göç yaşandığı görülmektedir.



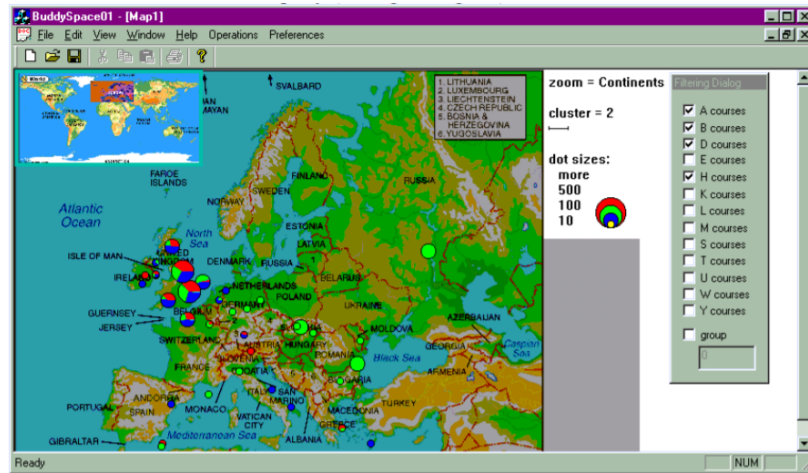
Şekil 2.19: Amerikan iç göç haritası (tüm vektörlerin güncellenme sonrası gösterimi)[32]

Karl-Heinrich Anders 2003 yılında yapmış olduğu çalışmada noktasal olarak elde edilen veri setinin ilişkili bir biçimde kümelenmesi üzerine çalışmalar yürütmüştür[33]. Kendisi tarafından oluşturulan HPGCL (hierarchical parameter-free graph-clustering) algoritmasına göre noktalar yoğunluk ve geometrik şekiller baz alınarak gruplandırılmaktadır. Kümeler ve küme dışı kalan noktaların uzaklıkları da temel belirleyici etmenlerdendir. Bu işlemlerden sonra veriler kümelenmektedir. Ayrıca yöntem özellikle 3 boyutlu objelerin 2 boyutlu düzlemde noktasal olarak (tarama verileri sonucu) ifade edilmesini de sağlamaktadır. Şekil 2.20 içerisinde görüldüğü üzere HPGCL algoritması girdi olarak noktaları alıp çıktı olarak noktaların ait olduğu kümeleri vermektedir. İlgili şeklin (c) ve (f) bölümünde 3 boyutlu noktaların 2 boyutlu izdüşümü temel alınmıştır.

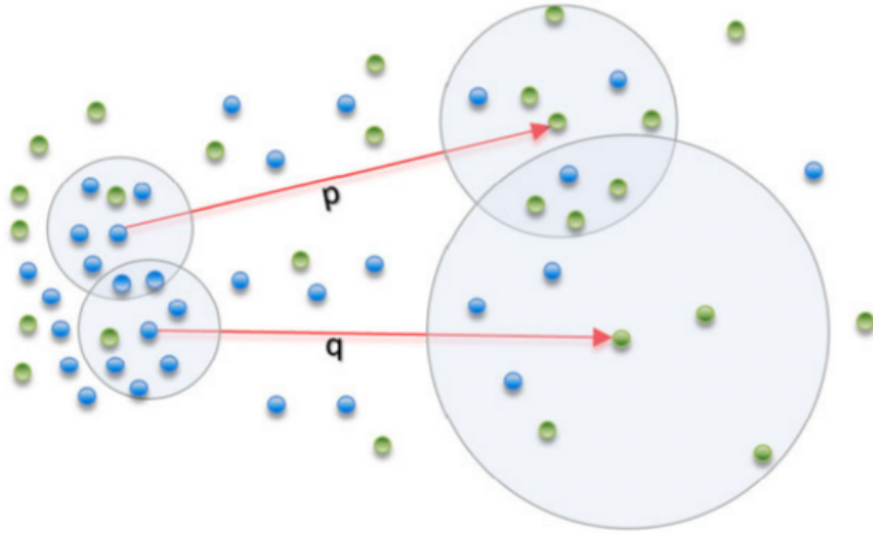


Şekil 2.20: HPGCL algoritması girdileri ve çıktıları[33]

Jiri Komzak ve Marc Eisenstadt, 2001 yılında yapmış oldukları çalışmada konumsal verileri bulunan Dünya üzerindeki öğrenci bilgilerini bir harita üzerinde sunmaya çalışmışlardır[34]. Bu çalışma ölçeklenebilir olup, çalışmaya göre öğrencilerin çalıştıkları alanlar (lisans, yüksek lisans, doktora vb.) harita üzerinde gösterilmeye çalışılmıştır. Kıta, ülke ve şehir gibi ölçeklerin bulunduğu sistemde öğrencilerin sayısı çemberin büyüklüğüyle orantılı olarak ifade edilmiştir. Çemberin içerisindeki renkler ise çalışılan alanların ağırlığını göstermektedir. Çalışmanın sonucunda bir uygulama geliştirilmiştir. Uygulamaya göre verisi bulunan (milyonlarca öğrenci verisi) üniversiteler harita üzerinde ölçeklenebilir olarak gösterilmiştir. Anlık veriye dayalı olmayan sistemde görüntüler arka planda belirli bir sürede oluşturulmuş ve son kullanıcıya statik olarak sunulmaktadır. Şekil 2.21 içerisinde uygulama gösterilmiştir.

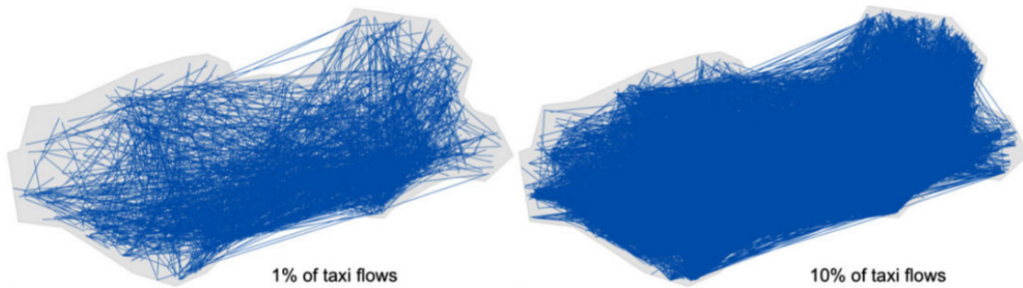


Şekil 2.21: Jiri Komzak ve Marc Eisenstadt tarafından oluşturulan öğrenci verilerini konumsal olarak gösteren uygulama [34]

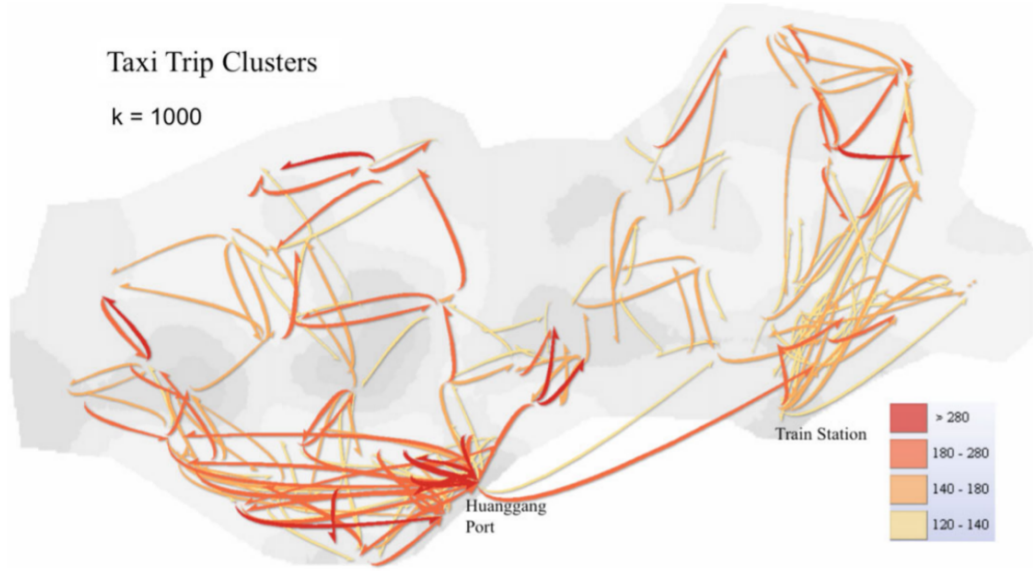


Şekil 2.22: SNN algoritmasıyla birleştirilen vektörel veriler [35]

Xi Zhu ve Diansheng Guo 2014 yılında yapmış oldukları çalışmada[35] Diansheng Guo'nun 2009 yılında tek başına yaptığı çalışmaya[32] benzer bir şekilde vektörel verilerin daha iyi bir şekilde gösterilmesi için bir algoritma üzerinde çalışmışlardır. Yapmış oldukları çalışma taksilerin yolcuları aldığı ve bıraktığı noktalar baz alınarak oluşturulmuştur. SNN (shared nearest neighbors) algoritması kullanılarak yapılan çalışmaya göre taksilerin yolcuyu aldığı ve bıraktığı noktalar kümelenmektedir. Uygulanan işlem Şekil 2.22 içerisinde gösterilmektedir. Bu kümeleme neticesinde Şekil 2.23 içerisinde sadece yüzde 1 ve yüzde 10'u gösterilen veri karmaşasının önüne geçilmiş daha sade bir görünüm elde edilmiştir. Uygulamanın nihai durumu Şekil 2.24 içerisinde gösterilmiştir.



Şekil 2.23: Taksi verilerinin karmaşıklığı [35]



Şekil 2.24: Xi Zhu ve Diansheng Guo tarafından oluşturulan nihai görüntü [35]

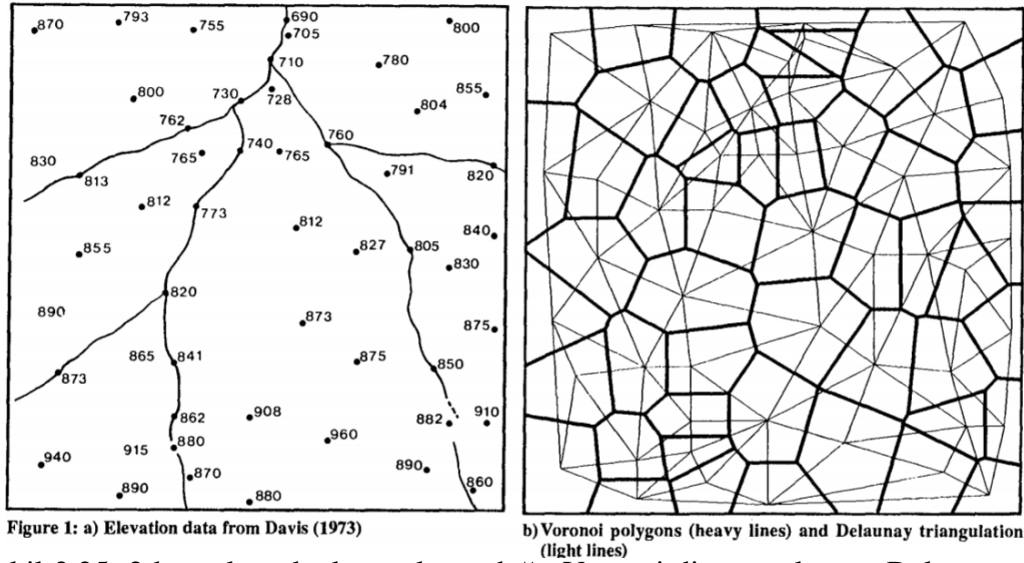


Figure 1: a) Elevation data from Davis (1973)

b) Voronoi polygons (heavy lines) and Delaunay triangulation (light lines)

Şekil 2.25: 2 boyutlu noktaların oluşturduğu Voronoi diyagramları ve Delaunay üçgenlemeleri [36]

Christopher Gold tarafından 1991 yılında yapılan araştırmada, araştırmacı yükseklik verisi de içeren konumsal verilerin Voronoi diyagramları kullanılarak gösterilebileceği bir çalışma gerçekleştirmiştir[36]. Bu çalışmada her bir nokta Voronoi diyagramı içerisinde merkez nokta olarak kabul edilmiştir. Çalışma neticesinde Voronoi diyagramlarının kesen Delaunay üçgenlemelerinin eğimi hesaplanmıştır. Delaunay üçgenlemelerinin yükseklikleri bağlı bulundukları noktaların ağırlıklı ortalaması kullanılarak

hesaplanmıştır. Bu hesaplamalar neticesinde noktaları ve yükseklikleri verilen bir alanın eğimleri hesaplanarak topolojik haritaların oluşturulması sağlanmıştır. Şekil 2.25 içerisinde verilen noktaların Voronoi diyagramları ve Delaunay üçgenlemeleri gösterilmiştir. Ayrıca daha sonra bu veriler kullanılarak oluşturulan topolojik harita da Şekil 2.26 içerisinde gösterilmiştir.

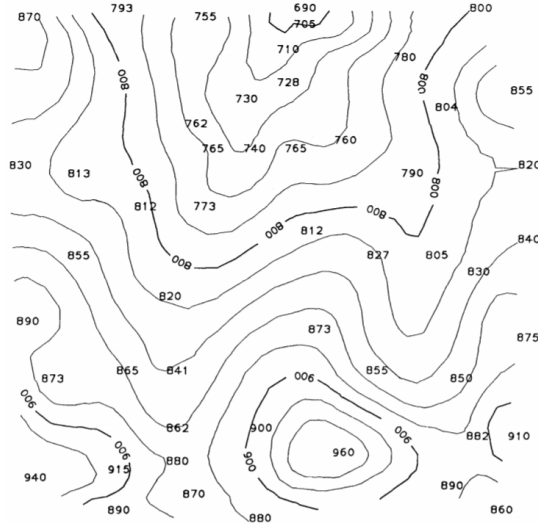
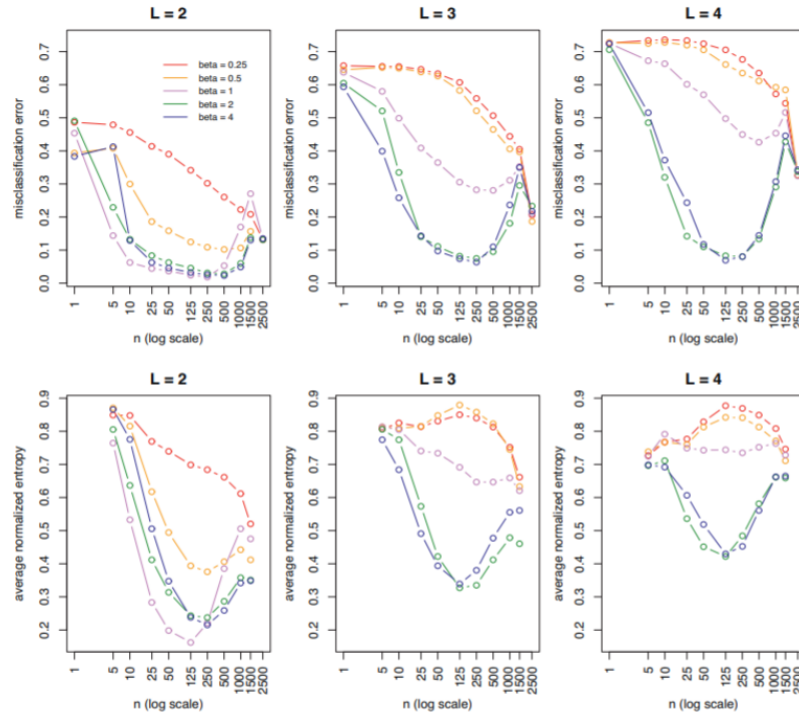


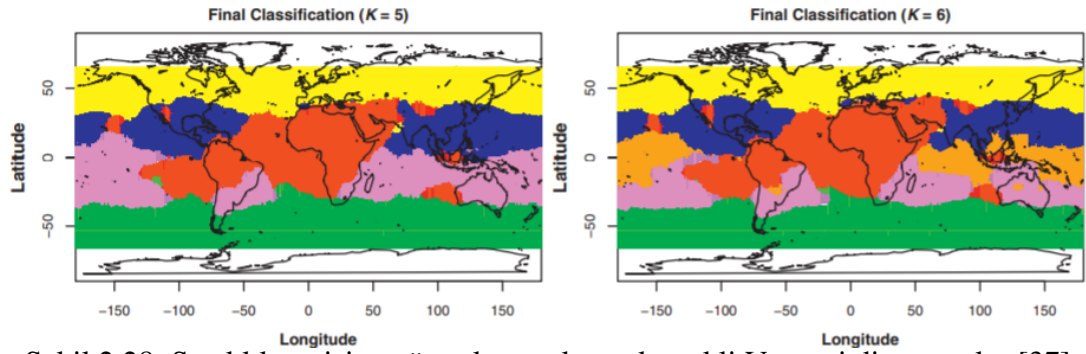
Figure 7: Area stealing interpolation of Davis data

Şekil 2.26: Voronoi diyagramları ve Delaunay üçgenlemeleri kullanılarak hazırlanan topolojik harita [36]

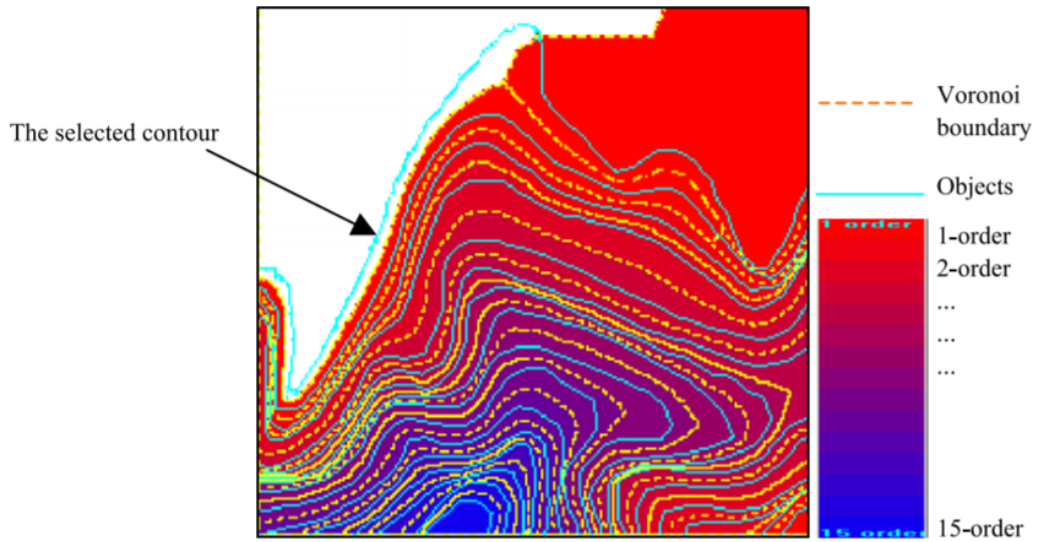


Şekil 2.27: Seviyelere göre hatalar ve entropi grafiği[37]

Piercesare Secchi, Simone Vantini ve Valeria Vitelli tarafından 2013 yılında yapılan araştırmada Dünya üzerindeki sıcaklık verilerinin son kullanıcıya daha sağlıklı gösterilmesi amaçlanmıştır[37]. Çalışmada sıcaklık verileri birden çok aşama ele alınarak gruplanmış ve bu gruplama sonucunda oluşan yapı Voronoi diyagramları kullanılarak ifade edilmiştir. Çalışma içerisinde Voronoi hücrelerinin sayısı artırıldığında entropi değerlerinin düşüşü de gösterilmiştir. Şekil 2.27 içerisinde entropilerin değişimi gösterilmiştir. Şekil içerisindeki n değeri Voronoi hücrelerinin sayısını göstermektedir. Çalışmada aylara göre sıcaklık verileri kullanılmış ve dinamik bir haritalama yöntemi elde edilmiştir. Şekil 2.28 içerisinde nihai görünüm gösterilmiştir. Şekil 2.28 içerisinde gösterilen harita Voronoi diyagramlarının birleştirilmesiyle elde edilmiştir.



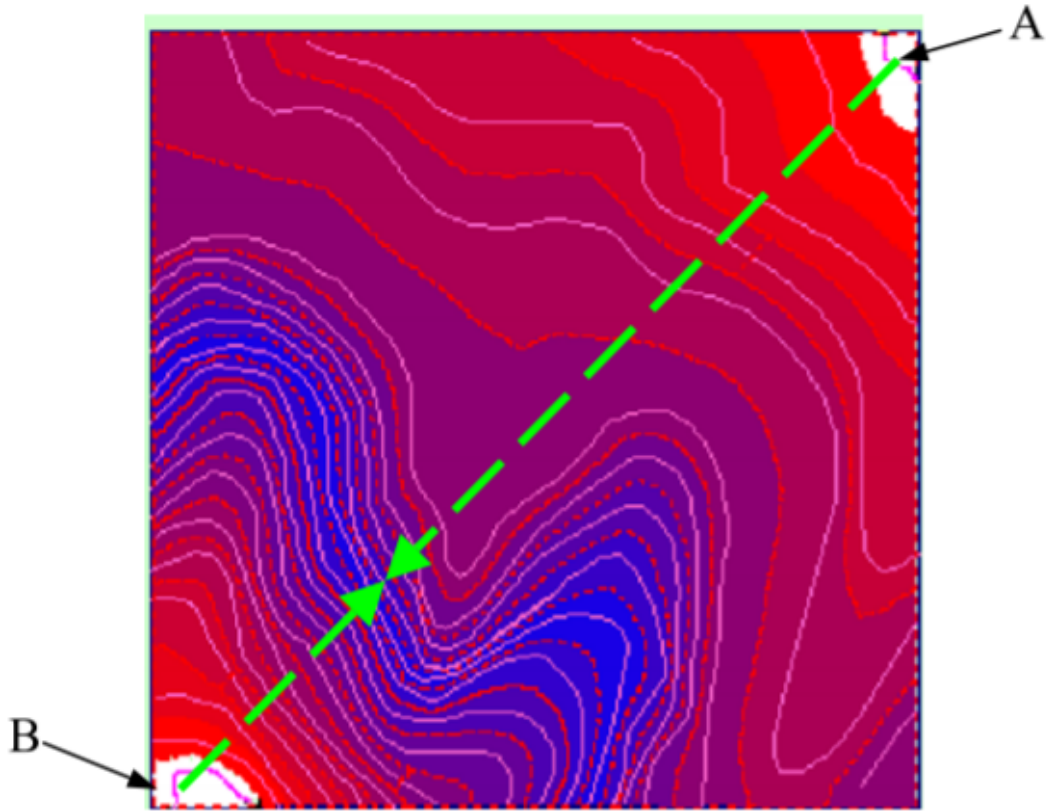
Şekil 2.28: Sıcaklık verisine göre oluşturulan çok renkli Voronoi diyagramları[37]



Şekil 2.29: Topolojik harita ve ilgili Voronoi diyagramı[38]

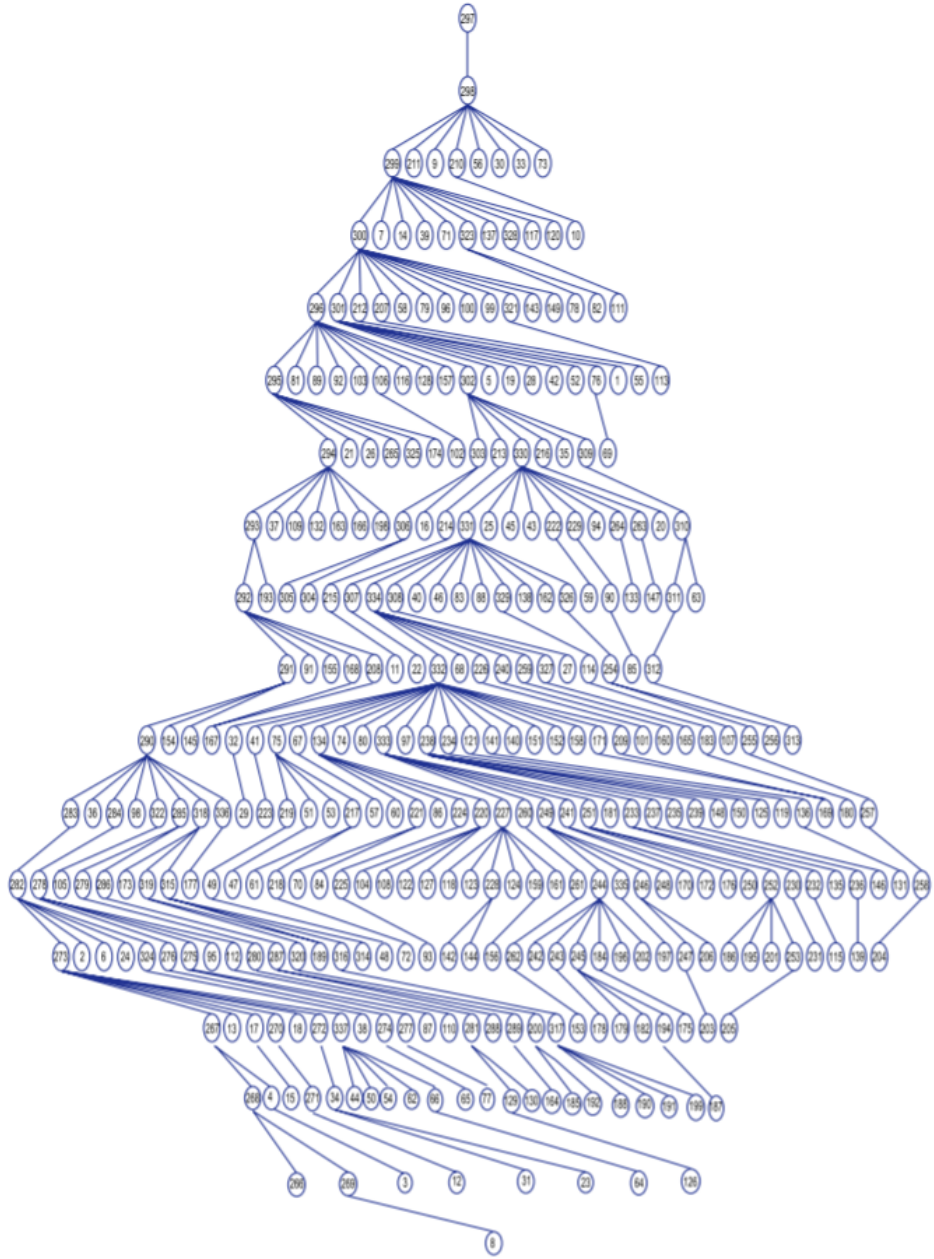
Jun Chen, Renliang Zhao ve Zhilin Li, 2004 yılında yaptıkları araştırmada topolojik

haritası verilen bir alanın hiyerarşik ağacını oluşturmaya çalışmışlardır[38]. Bu çalışmada Voronoi diyagramlarından da faydalanmışlardır. Buna göre sadece topolojik harita üzerinden komşuluk ilişkisi kurmak ve alanın hiyerarşik ağaç verisini çıkarmak bazı noktalarda hatalara sebep olmaktadır. Şekil 2.29 içerisinde topolojik harita ve Voronoi diyagramı gösterilmiştir.



Şekil 2.30: Voronoi diyagramı kullanılarak elde edilen komşuluk ilişkisi[38]

Jun Chen, Renliang Zhao ve Zhilin Li'nin yaptıkları araştırmada ise topolojik harita üzerine uygulanan Voronoi diyagramları neticesinde hiyerarşik ağaçlar oluşturulmuş ve daha anlamlı sonuçlar elde edilmiştir. Şekil 2.30 de ise bu veriler kullanılarak komşuluk ilişkileri oluşturulmuştur. Şekil 2.31 te ise oluşturulan hiyerarşik ağaç gösterilmektedir.



Şekil 2.31: Voronoi diyagramı kullanılarak elde edilen ağaç yapısı[38]

BÖLÜM 3

ÇALIŞMADA KULLANILAN VERİ SETLERİ

Bu çalışmanın gerçekleştirilmesi amacıyla 3 adet veri seti kullanılmıştır. Birinci veri seti daha önce benzer bir çalışma yürüten Bilgehan Çakmak'ın [31] oluşturduğu Dünya haritası düzleminde yer alan yapay bir veri setidir. İkinci veri seti Türkiye haritası baz alınarak bizim tarafımızdan oluşturulmuştur. Bu set Çakmak'a ait olan set gibi yapay olarak oluşturulmuştur. Üçüncü set ise “https://waqi.info/” adresinde yer alan Dünya düzleminde hava kirliliği verilerinin yine aynı sitenin uygulama programlama ara yüzü kullanılarak bizim tarafımızdan çekilmesiyle oluşturulan gerçek verilerdir.

3.1 Dünya Düzleminde Oluşturulan Sentetik Veri

Bu veri seti daha önce Çakmak tarafından yine Yüksek Lisans tezinde kullanılmak amacıyla Matplotlib Basemap aracıyla oluşturulmuştur. Set içerisinde 5 farklı veri dosyası bulunmaktadır. Bu 5 farklı veri dosyasının özeti Tablo 3.1’de sunulmaktadır.

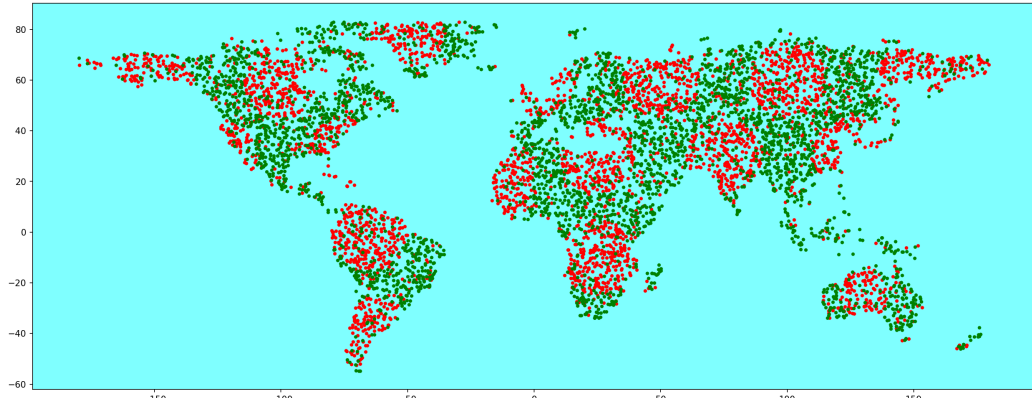
Bu tablodan anlaşıldığı üzere, kırmızı ve yeşil noktaların sayısı yaklaşık olarak eşit şekilde ayarlanmıştır. Fakat kırmızı noktalar bazı çemberlerin içerisinde baskın ola-

Tablo 3.1: Dünya düzleminde oluşturulan sentetik verinin dağılımı

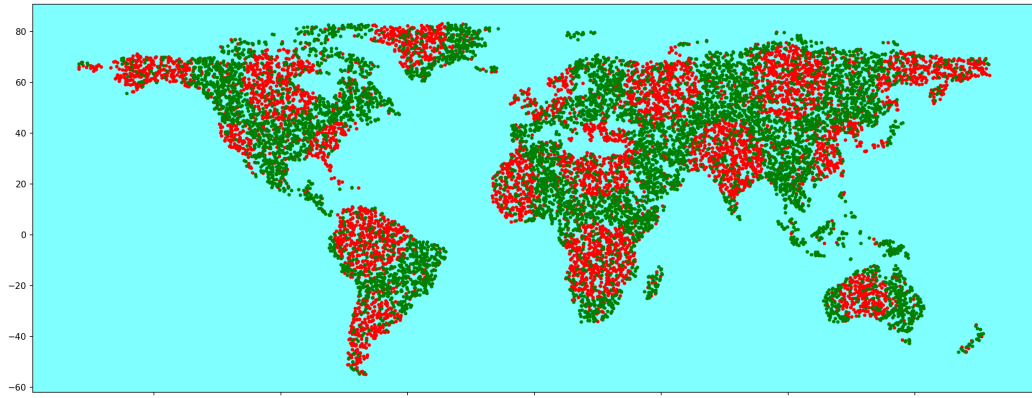
Veri Seti Adı	Toplam Nokta Sayısı	Tip 1 Nokta Sayısı (Kırmızı)	Tip 2 Nokta Sayısı (Yeşil)
B1	7160	3612	3548
B2	14323	7208	7115
B3	28350	14284	14066
B4	68090	34245	33845
B5	127672	64036	63636

cak şekilde ayarlanmış ama haritanın her yerinde bulunmaktadır. Yeşil noktalar ise bu çemberlerde seyrek olarak bulunmakta bu alanların dışındaki her alanda eşit dağılımla bulunmaktadır. Biz bu araştırmamızda baskın olan alanların diğer alanlardan görsel olarak ayrılmasını ve bu ayrılmanın matematiksel olarak nasıl ifade edilebileceğini araştırdığımızdan dolayı farklı veri setlerine ihtiyaç duyduk. Bu amaçla aynı doğrultuda hazırlanmış bu seti 5 farklı büyüklükte kullandık.

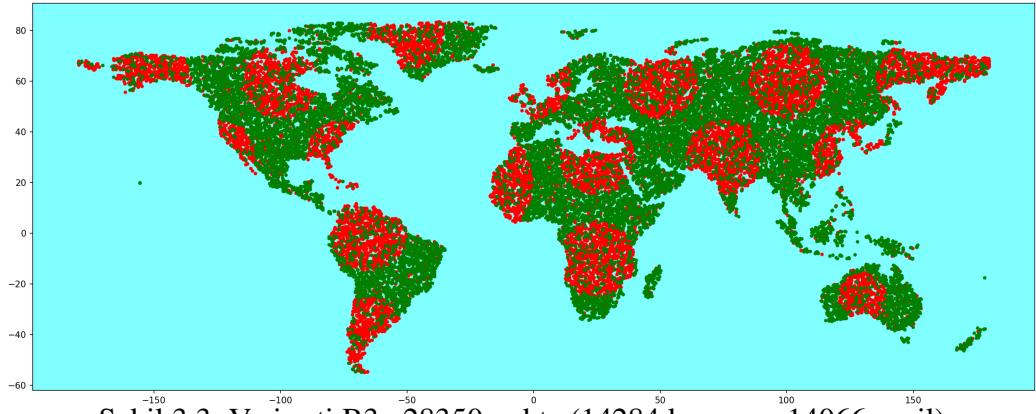
Şekil 3.1,3.2,3.3,3.4 ve 3.5 içerisinde sırasıyla 7160,14323,28350,68090 ve 127672 nokta sayısına sahip veri setlerinin dağılımı gösterilmektedir. Temel olarak, bu veriler 2 boyutlu düzlemde oluşturulmuştur. Yatay ekseninde (x) $[-180, 180]$ boylam ve düşey ekseninde (y) $[-65, 85]$ enlem arasında bu noktalar yerleştirilmiştir. Dünya düzleminin 2 boyutlu silindirik harita izdüşümünden faydalanılmıştır. Veri setinin daha gerçekçi olması amacıyla rastgele gürültüler eklenmiştir. Matematiksel hesaplamalarda ve karşılaştırmalarda daha çok bu veri seti üzerinde çalışmalar yapıyor olacağız.



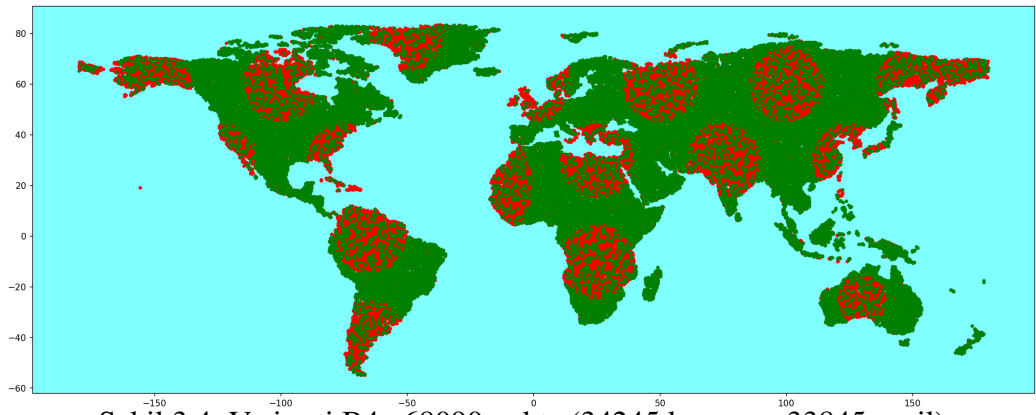
Şekil 3.1: Veri seti B1 , 7160 nokta (3612 kırmızı - 3548 yeşil)



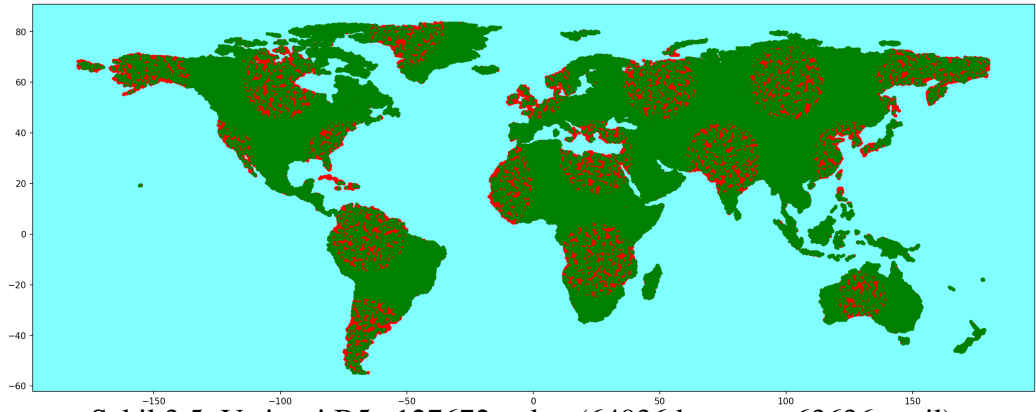
Şekil 3.2: Veri seti B2 , 14323 nokta (7208 kırmızı - 7115 yeşil)



Şekil 3.3: Veri seti B3 , 28350 nokta (14284 kırmızı - 14066 yeşil)



Şekil 3.4: Veri seti B4 , 68090 nokta (34245 kırmızı - 33845 yeşil)

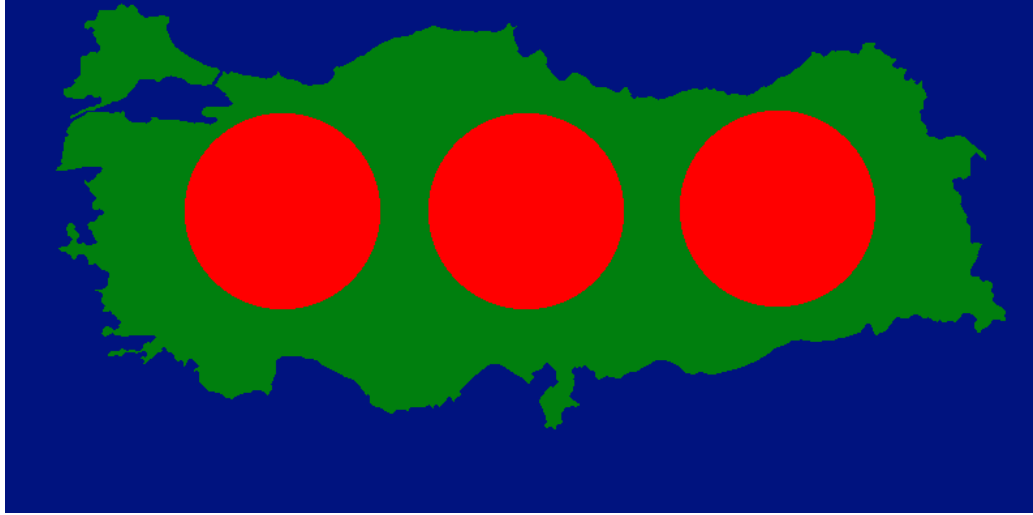


Şekil 3.5: Veri seti B5 , 127672 nokta (64036 kırmızı - 63636 yeşil)

Şekil 3.5 içerisinde açıkça görüldüğü üzere yeşil noktalar kırmızı noktaların üzerini örtmüştür. Bunun sebebi yeşil noktaların haritaya kırmızı noktalardan daha sonra aktarılmasıdır. Bu çalışmadaki temel motivasyonumuz bu gibi gösterim hatalarının önüne geçmektir.

3.2 Türkiye Düzleminde Oluşturulan Sentetik Veri

Türkiye düzleminde oluşturulan veri bizim tarafımızdan oluşturulmuştur. Temel harita (basemap) olarak baz aldığımız dosya Şekil 3.6 içerisinde gösterilmektedir.



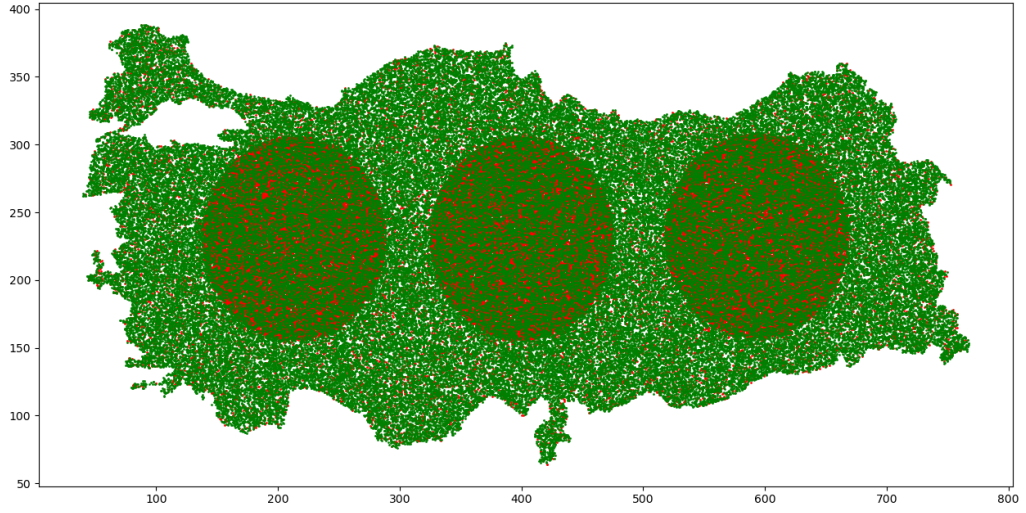
Şekil 3.6: Türkiye düzleminde oluşturulan sentetik veri temel haritası

Bu harita kullanılarak merkezde 3 adet çember seçilmiştir. Bu çemberlerin içerisinde Tip 1 Nokta (kırmızı) sayısı baskın olacak şekilde ayarlanmıştır fakat haritanın her alanında kırmızı nokta bulunmaktadır. Yeşil noktaların dağılımı ise haritanın her alanına eşit dağılacak şekilde yerleştirilmiştir. Tablo 3.2 içerisinde nokta sayıları belirtilmiştir. Şekil 3.6 içerisinde görülen kırmızı alanların içerisindeki kırmızı nokta oranı yaklaşık olarak yüzde 73'tür. Yeşil alanlar içerisindeki yeşil noktaların oranı ise yaklaşık olarak yüzde 78'dir.

Tablo 3.2: Türkiye düzleminde oluşturulan sentetik verinin dağılımı

Veri Seti Adı	Toplam Nokta Sayısı	Tip 1 Nokta Sayısı (Kırmızı)	Tip 2 Nokta Sayısı (Yeşil)
Türkiye	200000	100000	100000

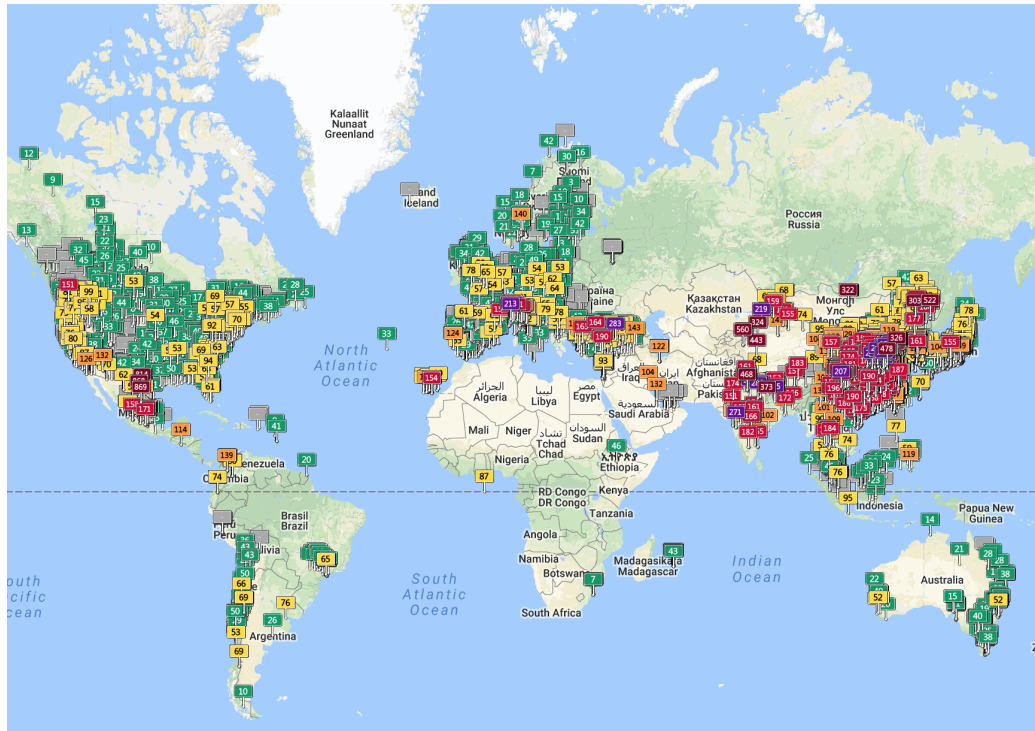
Bu veri setine ait noktaların dağılımı Şekil 3.7 içerisinde gösterilmektedir. Yatay ekseninde (x) [0, 800] ve dikey ekseninde (y) [0, 400] değerleri arasında bu noktalar yerleştirilmiştir. Verilerin dağılımı Şekil 3.6 baz alınarak oluşturulmuştur. Bu sebeple şekil boyutları eksen olarak kabul edilmiştir.



Şekil 3.7: Türkiye düzlemi sentetik veriler, kırmızı 100000, yeşil 100000 nokta

3.3 Dünya Düzlemindeki Gerçek Veri

Bu veri seti “<https://waqi.info/>” adresinde her gün canlı olarak yayınlanan Dünya hava kalitesi endeksi verilerini içermektedir. Aynı sitenin “<https://aqicn.org/api/>” adresindeki uygulama programlama ara yüzü kullanılarak 30 Temmuz 2020 tarihinde elde edilmiştir.



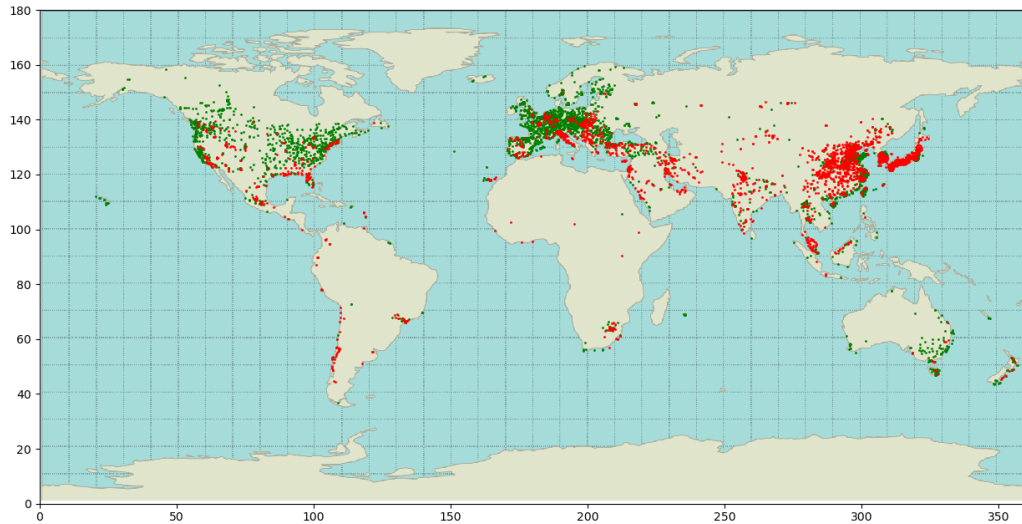
Şekil 3.8: Dünya hava kalitesi endeksi haritası (<https://waqi.info/>)

Bu veri Dünya üzerinde bulunan 6000’den fazla istasyondan elde edilen hava kalitesi verilerini içermektedir. Şekil 3.8 içerisinde görüldüğü üzere bu servis hava kalitesini harita üzerinde gösterirken kullanıcıya anlamlı veri sunamamaktadır.

Tablo 3.3: Dünya düzleminde hava kalitesi verisinin dağılımı

Veri Seti Adı	Toplam Nokta Sayısı	Tip 1 Nokta Sayısı (Kırmızı)	Tip 2 Nokta Sayısı (Yeşil)
Hava Kalitesi	6373	2858	3515

Bu veri ikiden fazla sınıfa sahiptir. Bu çalışmanın temel varsayımlarına uyabilmesi adına veriyi bir eşik değeri kullanarak iki sınıflı hale getirdik. Eşik değeri dengeli bir veri oluşturmak adına 40 seçilmiştir. Tablo 3.3 içerisinde çekilen bu verilerin dağılımı gösterilmektedir.



Şekil 3.9: Dünya hava kalitesi endeksi noktasal dağılımı

Şekil 3.9 içerisinde görüldüğü üzere bu noktalar x ekseninde $[-180,180]$ arasında y ekseninde ise $[-90,90]$ arasında doğal bir dağılım sergilemektedir.

BÖLÜM 4

METODOLOJİ

4.1 Mevcut Metodolojiler

Bu çalışmada temel olarak ele alınan çalışma Bilgehan Çakmak tarafından hazırlanan “Çok Büyük Konumsal Verinin Görselleştirilmesinde Karmaşıklık Yönetimi” isimli tez çalışmasıdır[31]. Bu çalışma içerisinde yer alan veri setleri bizim tarafımızdan da kullanılmıştır. Bu tezin içerisinde birçok sınıflandırma algoritması ve benzer çözüm yöntemleri incelendikten sonra veri setinin tanıtımına geçilmiştir. Veri setlerinin ayrıntıları Bölüm 3 içerisinde incelendiğinden bu kısımlar tekrar anlatılmamıştır.

Çakmak bizimle aynı problem üzerine bir çalışma gerçekleştirmiştir. Buna göre çok sayıda nokta içeren(her nokta kırmızı ya da yeşil olarak sınıflandırılmıştır) veri setlerinin son kullanıcıya hızlı, temiz ve anlaşılır bir şekilde sunulması amaçlanmıştır. Bu amaç doğrultusunda çalışmada üç adet çözüm önerisi sunulmuştur. Bu çözüm önerileri sırasıyla veri setinin eşit parçalara(karelere) bölünmesi, veri setinin eşit büyüklükte olmayan karelere adaptif bir şekilde bölünmesi ve veri setinin dikdörtgenlere adaptif bir şekilde bölünmesi olarak sıralanabilir. Bu çalışmada kullanılan yöntemler ve bu yöntemlerin sonuçları Bölüm 5 içerisinde karşılaştırmalı olarak incelenecektir.

Karelere ya da dikdörtgenlere bölme yöntemi her ne kadar kolay ve hızlı sonuçlar verse de insan göz yapısına uygun değildir. Çakmak tarafından uygulanan bu yöntemler genel olarak daha verimli bir bölme nasıl gerçekleştirilir sorusunun karşılığıdır. Biz de çalışmamızda aynı problemin çözümüne odaklandık. Verimli bölmenin matematiksel olarak karşılığı ise entropi hesaplamaları şeklinde gerçekleşmiştir. Sunulan 3 yöntem içerisinde her ne kadar farklı entropi hesaplamaları yer alsa da temel

olarak entropi alanların içerisindeki noktalara göre ağırlıklı bir hesaplama.

4.1.1 Metod 1 - Sabit Kare Kullanılarak Bölütme

Bu metoda göre veri seti içerisindeki noktalar eşit ölçekteki karelere bölünmekte ve daha sonrasında bu kareler renklendirilmektedir. Metod girdi olarak kareleri(x ve y değerleri ve bu noktalardaki renk bilgisi 0,1 gibi) ve kare boyutunu almaktadır. Alanın tümü bu karelere bölündükten sonra noktalar içerisinde bulundukları karelere bölünmektedir. Bu bölünmenin neticesinde ilgili kare renklendirilmektedir. Buna göre eğer kare içerisinde hiçbir nokta yoksa ya da nokta sayıları eşitse o kare açık mavi olarak renklendirilmektedir. Kare içerisinde herhangi bir grubun(kırmızı ya da yeşil) baskınlığı söz konusu ise kare o renkle temsil edilmektedir. Bu yönüyle metod genel olarak pikselleştirmeye benzemektedir[31].

Metodun algoritması Ek A içerisinde aktarılmıştır. Kare boyutları paralel ve meridyenler baz alınarak oluşturulmuştur. Buna göre $20^\circ \times 20^\circ$, $10^\circ \times 10^\circ$, $5^\circ \times 5^\circ$, $2.5^\circ \times 2.5^\circ$, $1.25^\circ \times 1.25^\circ$ gibi 5 adet kare boyutu belirlenmiştir.

$$H(X) = \sum_{i=1}^n P(x_i) I(x_i) = - \sum_{i=1}^n P(x_i) \log_2(P(x_i))$$

Kare içi hesaplama

$$H_{Grid}(X) = \frac{\sum_{i=1}^n H_{Cell_i} * X_{Cell_i}}{X_{Grid}}$$

Tüm alanın ağırlıklı entropi hesaplaması

Şekil 4.1: Çakmak tarafından uygulanan Metod 1 algoritmasının entropi hesaplaması [31]

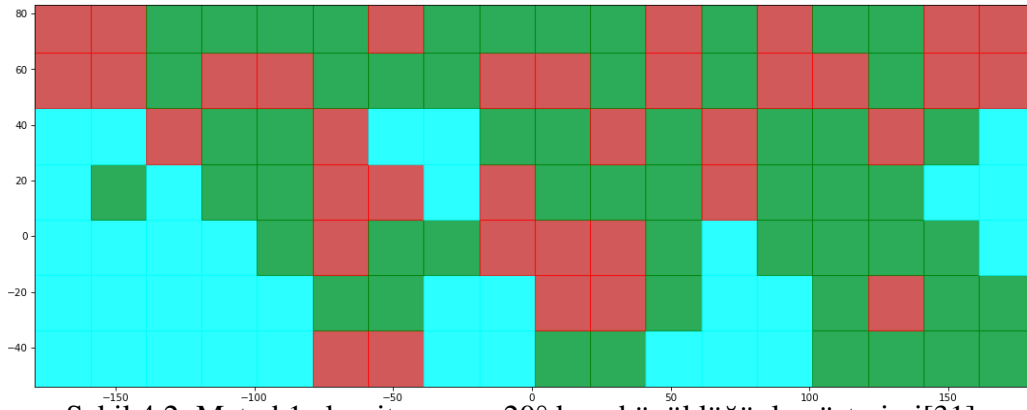
Bu metod içerisinde entropi hesabı ağırlıklı ortalama kullanılarak yapılmıştır. Entropi formülü olarak Shannon hesaplaması kullanılmıştır. Buna göre her karenin ayrı ayrı entropisi hesaplanmış daha sonra kare içerisindeki noktaların sayısı baz alınarak ağırlıklı bir toplam entropi hesabı yapılmıştır. Şekil 4.1 içerisinde yapılan entropi hesaplamaları görülmektedir.

Tablo 4.1: Veri setlerine ve kare boyutlarına göre toplam entropi değerleri[31]

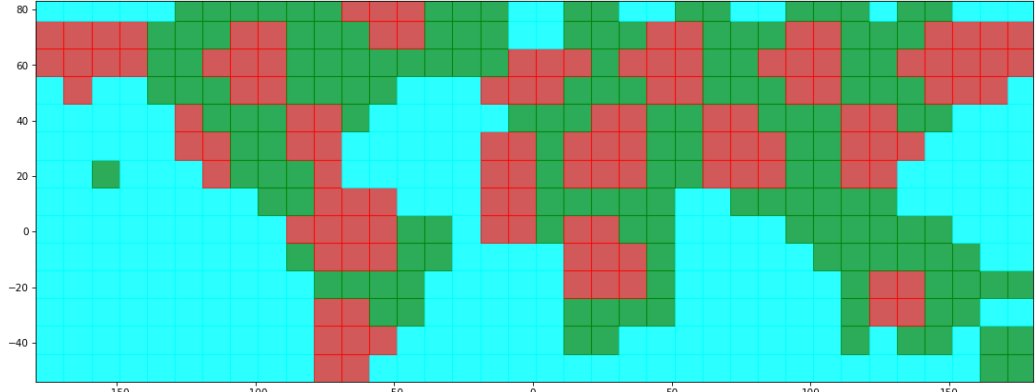
Kare Büyüküğü	B1 Veri Seti	B2 Veri Seti	B3 Veri Seti	B4 Veri Seti	B5 Veri Seti
20°	0.8374	0.8271	0.8348	0.8326	0.8327
10°	0.7023	0.7058	0.7174	0.7204	0.7227
5°	0.5897	0.6204	0.6454	0.6605	0.6650
2.5°	0.3766	0.4699	0.5540	0.6048	0.6197
1.25°	0.1461	0.2439	0.3644	0.4907	0.5475

Bu hesaplamalara göre entropi değerleri kare boyutları küçüldükçe azalmaktadır. Algoritmanın zamanı incelendiğinde ise nokta sayısı arttıkça geçen sürenin arttığı gözlemlenmektedir. Ayrıca kare boyutlarının küçülmesinin de zamanı arttırdığı gözlemlenmiştir. Tablo 4.1 içerisinde veri setlerinin kare boyutlarına göre toplam entropi değerleri gösterilmiştir.

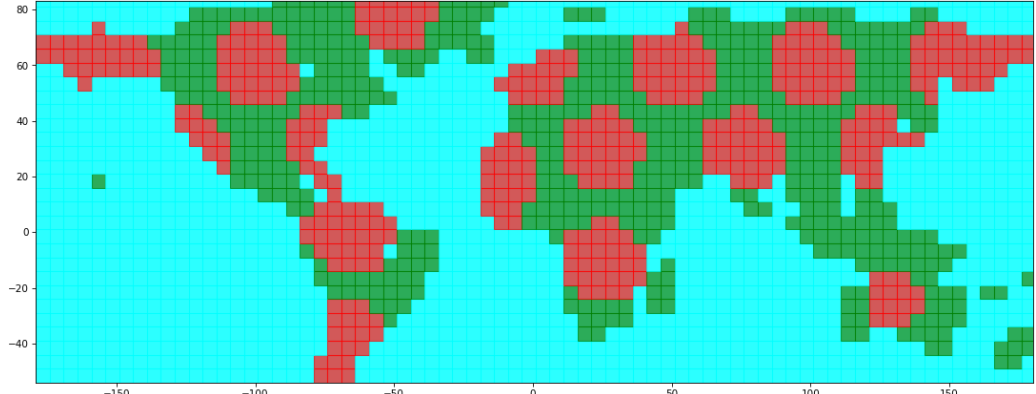
Karelerin renklendirilmesi sonucunda görsel haritalar elde edilmiştir. Bu haritalardan B5 veri setine ait görsel sonuçlar Şekil 4.2, 4.3, 4.4, 4.5 ve 4.6 içerisinde kare boyutlarına göre farklılık gösterecek şekilde ifade edilmiştir. Bu şekiller için entropi değerleri sırasıyla 0.8327, 0.7227, 0.6650, 0.6197 ve 0.5475 olarak hesaplanmıştır.



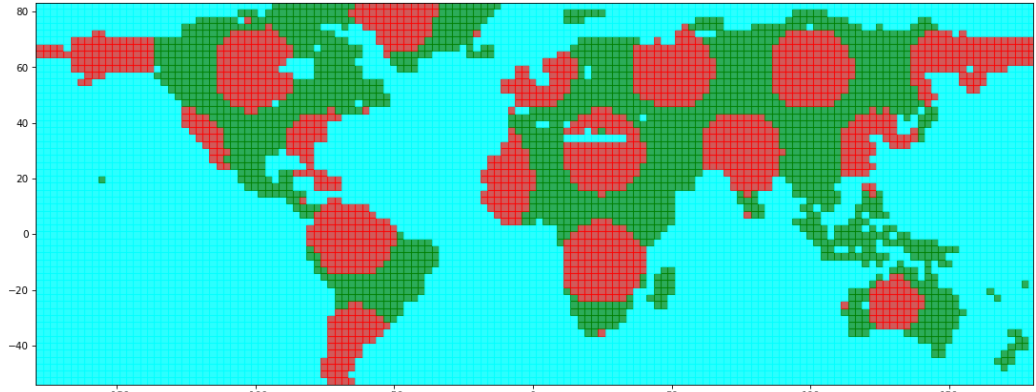
Şekil 4.2: Metod 1 algoritmasının 20° kare büyüklüğüyle gösterimi[31]



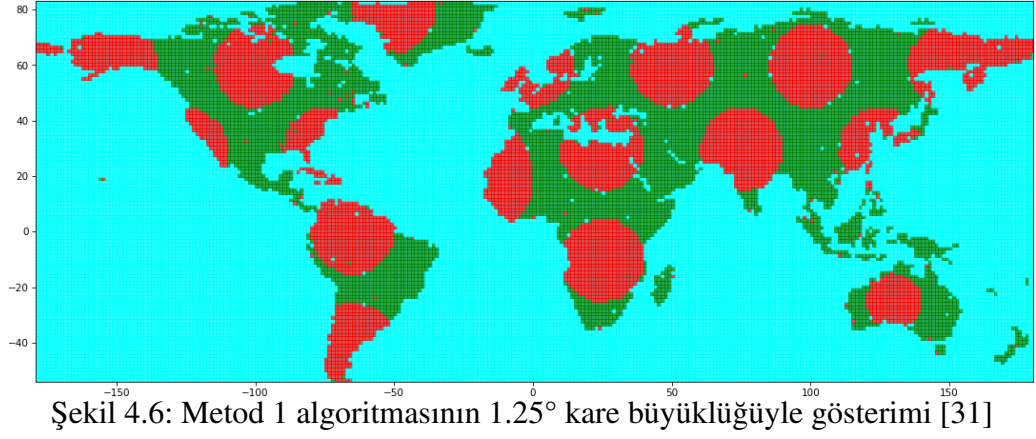
Şekil 4.3: Metod 1 algoritmasının 10° kare büyüklüğüyle gösterimi [31]



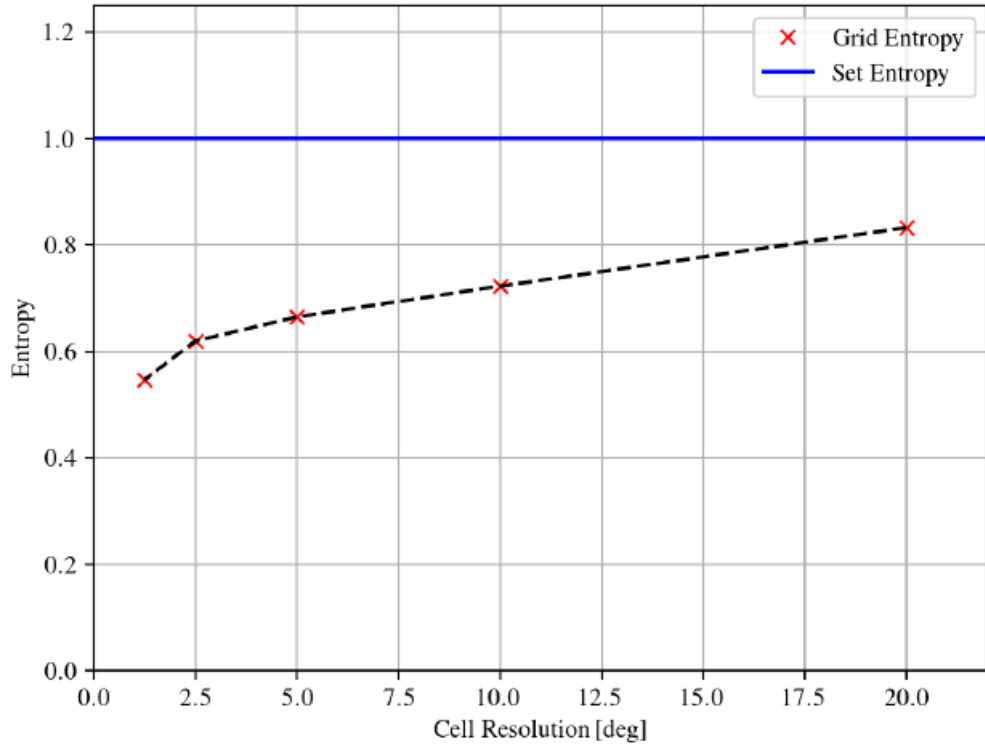
Şekil 4.4: Metod 1 algoritmasının 5° kare büyüklüğüyle gösterimi [31]



Şekil 4.5: Metod 1 algoritmasının 2.5° kare büyüklüğüyle gösterimi [31]



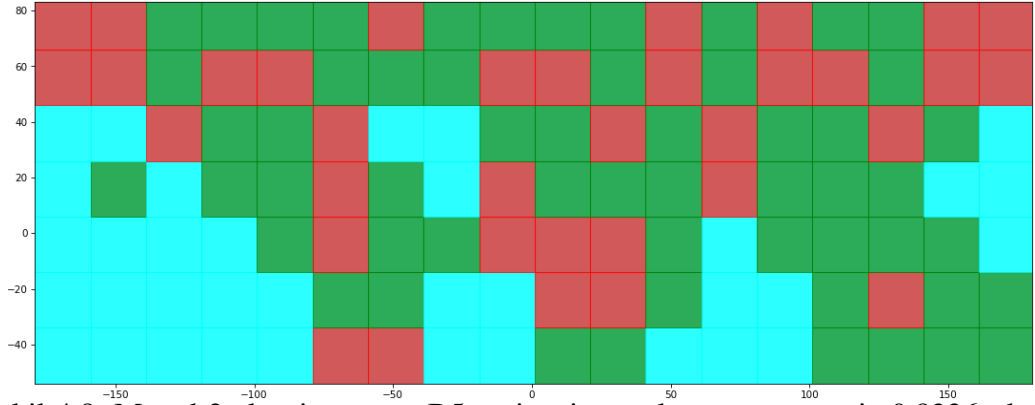
Metod içerisinde B5 veri seti kullanılarak oluşturulan görsellerin entropi değerleri ilgili formül kullanılarak hesaplanmıştır. Bu hesaplama göre entropi grafiği Şekil 4.7 içerisinde verilmiştir.



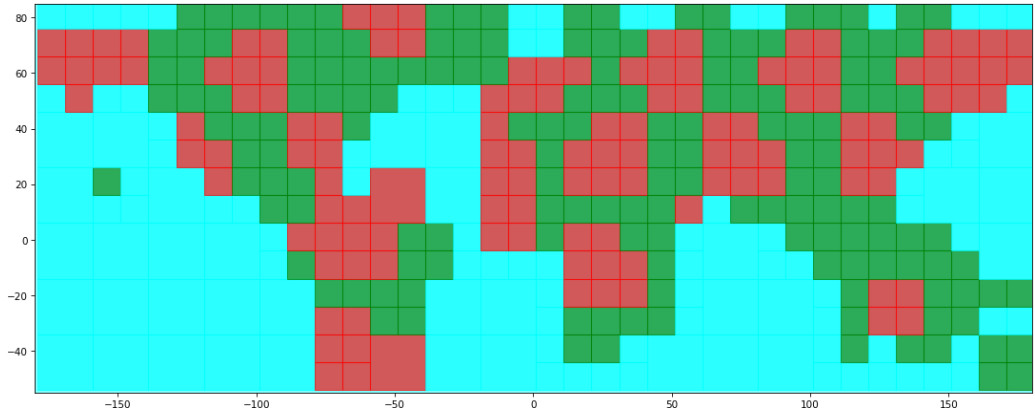
Şekil 4.7: Metod 1 algoritmasının B5 veri seti kullanılarak oluşturulan entropi grafiği [31]

4.1.2 Metod 2 - Adaptif Kare Kullanılarak Bölütleme

Çakmak tarafından uygulanan ikinci metoda göre kareler adaptif bir bölünmeye tabi tutulmuştur. Bu metodun uygulanmasındaki ilk motivasyon Metod 1’de oluşan neredeyse doğrusal zamanlamayı kısaltmaktır. Ayrıca, sonuç içerisindeki her küme farklı büyüklüklere sahiptir. Bu farklı büyüklüklerle birlikte veri daha iyi ifade edilebilmektedir. Farklı kare boyutları için uygulanan yöntem ise entropi hesabına dayanmaktadır. Buna göre bir karenin içerdiği karmaşıklık değeri bir eşik değere tabi tutulur. Bu sayede eğer yeterince az karmaşıklık elde edilemediyse kare daha küçük dört adet kareye bölünür. Azami bölünme seviyesine ulaşıldığında ise bölünme entropi hesabına bakmaksızın durdurulur.



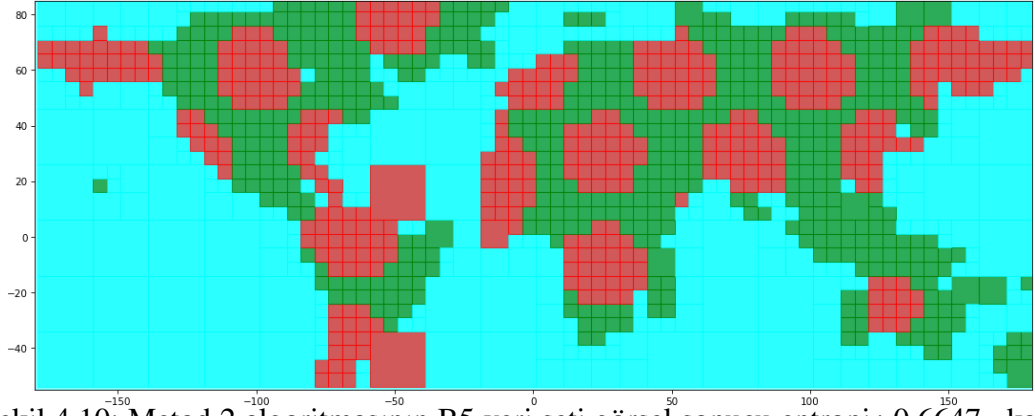
Şekil 4.8: Metod 2 algoritmasının B5 veri seti görsel sonucu entropi : 0.8326 , kare boyutu : 20° [31]



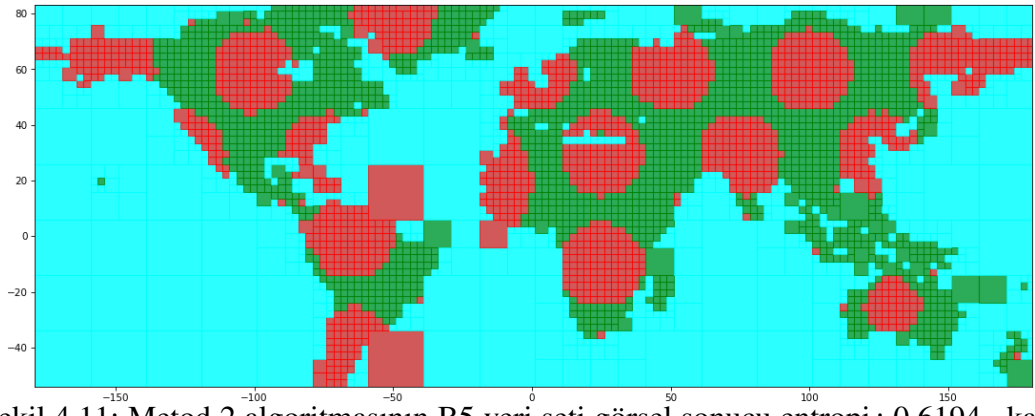
Şekil 4.9: Metod 2 algoritmasının B5 veri seti görsel sonucu entropi : 0.7224 , kare boyutu : 10° [31]

Şekil 4.8, 4.9, 4.10, 4.11 ve 4.12 içerisinde Metod 2’ye göre alınan görsel sonuçlar ve entropi değerleri gösterilmektedir. Bu görseller B5 veri seti kullanılarak oluşt-

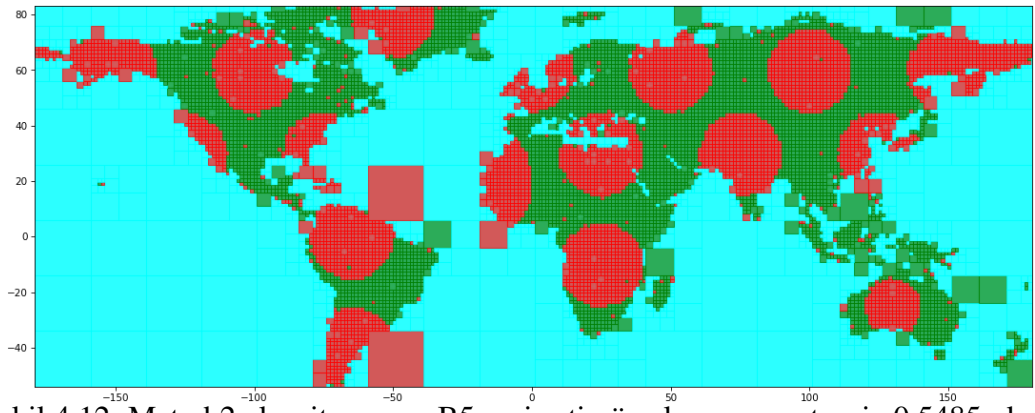
rulmuştur. Görsellerde belirtilen kare boyutu minimum boyutu ifade etmektedir. Minimum kare boyutuna ulaşıldığında entropiye bakılmaksızın daha küçük bölütleme yapılmamaktadır.



Şekil 4.10: Metod 2 algoritmasının B5 veri seti görsel sonucu entropi : 0.6647 , kare boyutu : 5° [31]



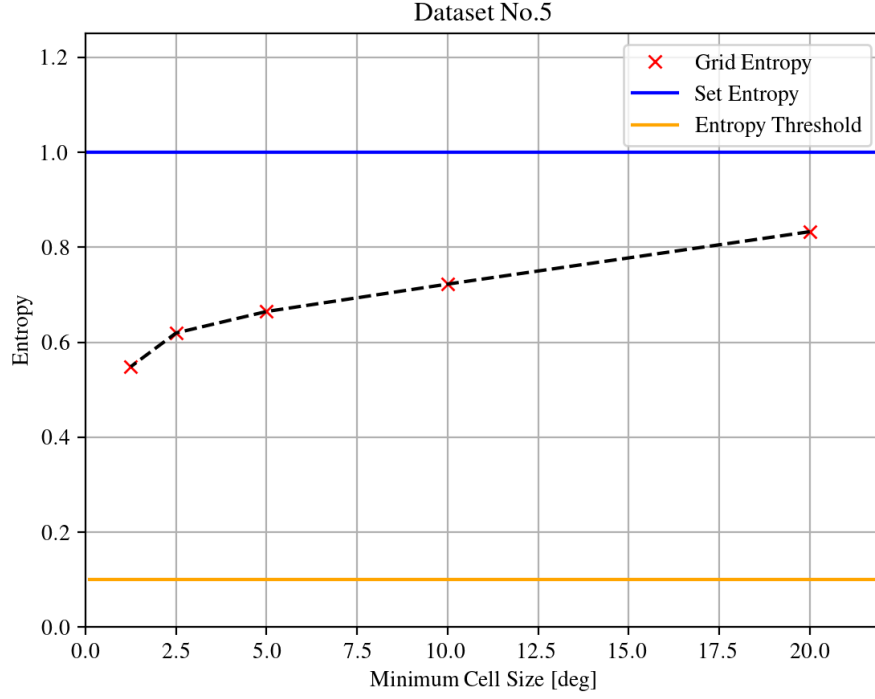
Şekil 4.11: Metod 2 algoritmasının B5 veri seti görsel sonucu entropi : 0.6194 , kare boyutu : 2.5° [31]



Şekil 4.12: Metod 2 algoritmasının B5 veri seti görsel sonucu entropi : 0.5485 , kare boyutu : 1.25° [31]

Metodun algoritması Ek B içerisinde aktarılmıştır.

B5 veri seti bu metod içerisinde kullanıldığında kare boyutlarının küçülmesiyle birlikte entropide bir azalma olduğu gözlemlenmektedir. Şekil 4.13 içerisinde B5 veri seti için kare boyutları ve entropi değerleri gösterilmektedir.



Şekil 4.13: Metod 2 algoritmasının B5 veri seti kullanılarak oluşturulan entropi değerleri[31]

Tablo 4.2 içerisinde farklı veri setleri ve kare büyüklüklerine göre entropi değerleri aktarılmıştır.

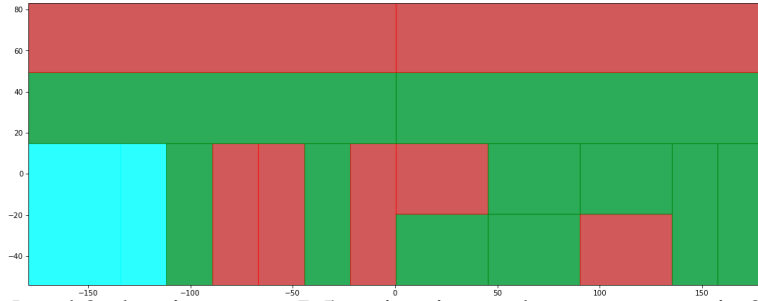
Tablo 4.2: Metod 2 içerisinde farklı veri setlerine ve kare boyutlarına göre toplam entropi değerleri[31]

Kare Büyüklüğü	B1 Veri Seti	B2 Veri Seti	B3 Veri Seti	B4 Veri Seti	B5 Veri Seti
20°	0.8374	0.8271	0.8348	0.8326	0.8327
10°	0.7011	0.7055	0.7168	0.7199	0.7224
5°	0.5894	0.6204	0.6450	0.6601	0.6647
2.5°	0.3779	0.4745	0.5559	0.6047	0.6194
1.25°	0.1484	0.2530	0.3697	0.4945	0.5485

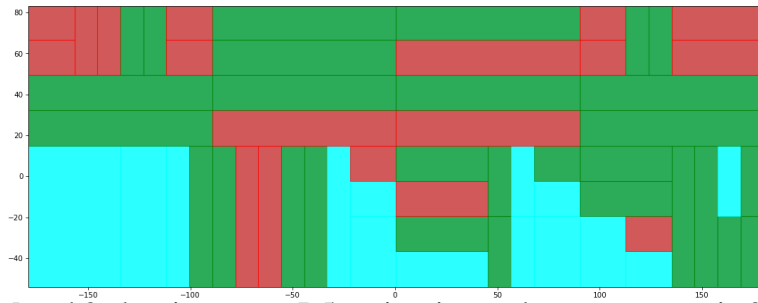
4.1.3 Metod 3 - Adaptif Dikdörtgen Kullanılarak Bölütleme

Çakmak tarafından uygulanan Metod 3'e göre bölütleme kareler üzerinden değil dikdörtgenler üzerinden yapılmaktadır. Bu yöntemin kullanılmasındaki amaç dikdörtgenlerin karelere göre daha esnek yapıda kullanılabilmesidir. Ayrıca dikdörtgen kareye göre daha anlamlı gösterimler sunabilir. Karelere göre bölütleme her ne kadar kolay uygulanabilir olsa da daha uzun çalışma zamanına sahiptir. Bunun sebebi kare sayısının çok yüksek olmasıdır. Bu metodun uygulanmasındaki temel motivasyon Metod 2'ye göre daha az bölütleme yaparak zaman karmaşıklığını azaltmaktır. Metodun algoritması Ek C içerisinde aktarılmıştır.

Metod 2'ye benzer şekilde bu metodun uygulanabilmesi için de başlangıç bölütlemesi yapmak gerekmektedir. Fakat Metod 2'den farklı olarak bu bölütleme bir dilim sayısı seçilerek yapılır. Buna göre algoritmanın başlangıcında seçilen bu dilimleme sayısına göre bir bölütleme gerçekleştirilir. Daha sonra her bir bölge belirli bir entropi değeri sağlanıncaya kadar özyinelemeli olarak tekrar bölünür. Asgari boyutlara ulaşıldığında bölütleme durdurulur.

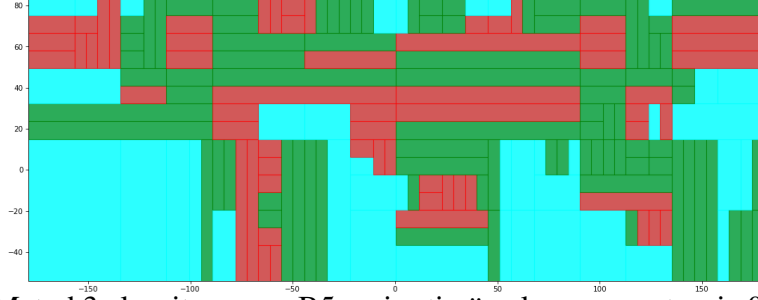


Şekil 4.14: Metod 3 algoritmasının B5 veri seti görsel sonucu entropi : 0.9756 , minimum boyut : 20° [31]

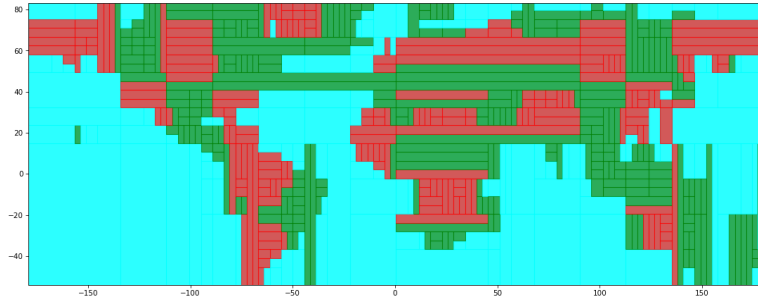


Şekil 4.15: Metod 3 algoritmasının B5 veri seti görsel sonucu entropi : 0.8680 , minimum boyut : 10° [31]

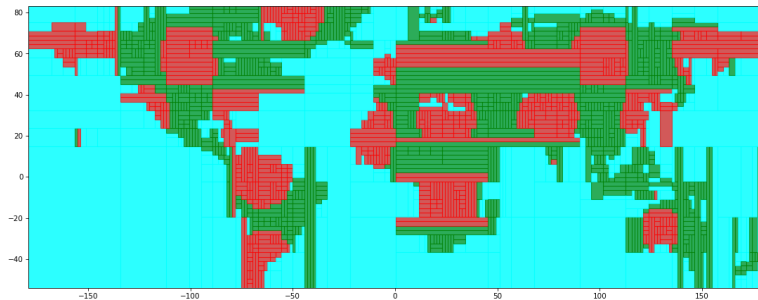
Şekil 4.14, 4.15, 4.16, 4.17 ve 4.18 içerisinde Metod 3'e göre alınan görsel sonuçlar ve entropi değerleri gösterilmektedir. Görsel sonuçlarda B5 veri seti kullanılmıştır. Minimum boyut dikdörtgenlerin kısa kenarının ulaşabileceği en küçük boyutu ifade etmektedir. Bu boyuta ulaşıldıktan sonra bölütleme entropiye bakılmaksızın sonlandırılır.



Şekil 4.16: Metod 3 algoritmasının B5 veri seti görsel sonucu entropi : 0.7933 , minimum boyut : 5° [31]

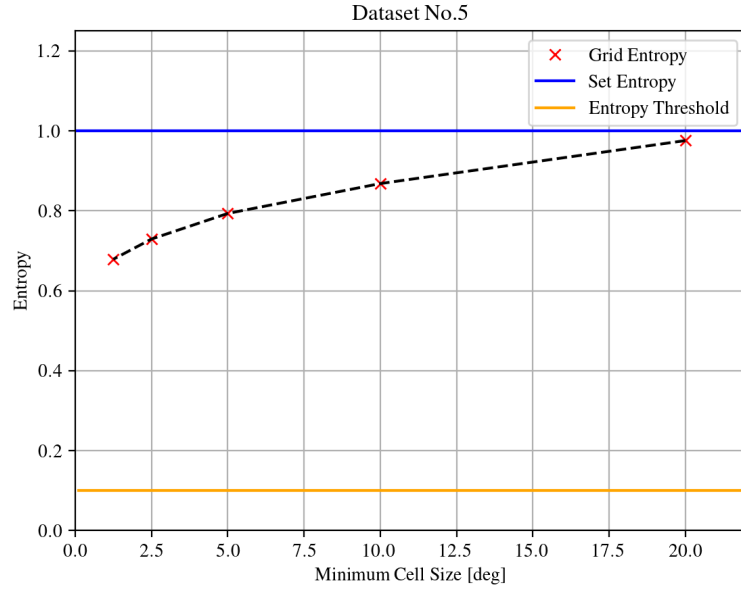


Şekil 4.17: Metod 3 algoritmasının B5 veri seti görsel sonucu entropi : 0.7293 , minimum boyut : 2.5° [31]



Şekil 4.18: Metod 3 algoritmasının B5 veri seti görsel sonucu entropi : 0.6788 , minimum boyut : 1.25° [31]

Metod 3 içerisinde uygulanan B5 veri seti ile minimum boyutun küçülmesiyle birlikte entropide bir azalma olduğu gözlemlenmektedir. Şekil 4.19 içerisinde B5 veri seti için minimum boyut ve entropi değerleri gösterilmektedir.



Şekil 4.19: Metod 3 algoritmasının B5 veri seti entropi grafiği[31]

Tüm veri setleri için entropi değerleri Tablo 4.3 içerisinde aktarılmıştır.

Tablo 4.3: Metod 3 tüm veri setleri için minimum boyutlara göre entropi değerleri[31]

Minimum boyut	B1 Veri Seti	B2 Veri Seti	B3 Veri Seti	B4 Veri Seti	B5 Veri Seti
20°	0.9738	0.9724	0.9766	0.9772	0.9756
10°	0.8830	0.8591	0.8669	0.8671	0.8680
5°	0.8118	0.7727	0.7771	0.8096	0.7933
2.5°	0.6806	0.7047	0.6937	0.7464	0.7293
1.25°	0.5271	0.5782	0.6272	0.6807	0.6788

Çakmak tarafından uygulanan metodların çıktıları incelendiğinde her ne kadar hızlı bir işlem uygulansa da görsel sonuçlar içerisinde istenmeyen durumların olduğu gözlemlenebilir. Örneğin Metod 2 işlemine göre oluşturulan görsellerde hatalı gösterimler sonucu bazı büyük kareler bulunmaktadır. Aynı şekilde Metod 3 içerisinde de bazı dikdörtgenlerin gereğinden fazla uzun ya da geniş olması gözlemlenmiştir. Kıyı kısımları her üç metod içerisinde de tam olarak algılanamamıştır.

4.2 Önerilen Metodoloji

Çok büyük sayıdaki konumsal verilerin görselleştirilmesi problemi bu çalışmanın temel aldığı problemdir. Bu problem çerçevesinde bizim çözümümüz Voronoi diyagramları kullanarak bir bölütleme gerçekleştirmektir. Buna göre öncelikle veri seti karelere bölünür. Daha sonrasında bu kareler içerdiği verilerin yoğunluğuna ve baskınlığına göre tekrar tekrar daha küçük karelere bölünür. Bu bölünme işlemi izin verilen seviyeye kadar devam eder. Daha sonrasında karelere bölme işlemi bitirilir. Bu sayede veriler farklı büyüklükteki karelerin içerisine yerleştirilmiş olur.

Karelere bölünme işlemi tamamlandıktan sonra her kare içerisinde rastgele bir nokta seçilir. Seçilen noktalar Voronoi diyagramına merkez nokta olarak verilir. Bu sayede Voronoi diyagramı elde edilmiş olur.

Voronoi diyagramının elde edilmesinden sonra Ball-tree algoritması kullanılarak veri seti içerisindeki noktaların hangi Voronoi merkez noktasına yakın olduğu tespit edilir. Bu sayede Voronoi hücreleri renklendirilir. Eğer Voronoi merkez noktasına yakın herhangi bir veri seti noktası bulunmuyorsa görsellerde mavi renkle ifade edilir. Bunun dışında hangi grubun (kırmızı ya da yeşil) çoğunluğu bulunuyorsa Voronoi hücresi o renkle renklendirilir. Eşitlik durumunda yeşil renk kullanılır. Görsellerde arka plana veri setinin ait olduğu bir taban haritası çizdirilmektedir. Bu haritanın üzerine Voronoi diyagramları şeffaf şekilde çizilerek son kullanıcıya uygun bir gösterim yapılmış olur.

4.2.1 Önerilen Metodolojinin Girdileri

Bu algoritmanın girdi olarak aldığı bilgiler şu şekilde listelenebilir:

- Veri seti
- Yükseklik
- Genişlik
- Yoğunluk eşiği

- Baskınlık eşiği
- İlk kare boyutu
- Derinlik

Bu parametreler içerisinde Yoğunluk Eşiği, Baskınlık Eşiği, İlk Kare Boyutu ve Derinlik bilgileri kullanılarak aynı veri seti için farklı haritalar oluşturmak mümkündür. Ayrıca bu parametreler kullanılarak çalışma zamanı ihtiyaca göre arttırılabilir ya da azaltılabilir. Bu parametreler sayesinde farklı kalitelere haritalandırma yapılabilir.

Veri seti, Bölüm 3 içerisinde bahsedildiği üzere x , y ve grup değeri (kırmızı için 0, yeşil için 1) olmak üzere 3 veri içeren noktalar kümesidir. Bu çalışmada B1, B2, B3, B4, B5, Türkiye ve hava kalitesi olmak üzere yedi tane veri seti kullanılmıştır. Türkiye ve hava kalitesi veri seti bizim tarafımızdan oluşturulmuştur. B1, B2, B3, B4 ve B5 veri seti ise Bilgehan Çakmak tarafından oluşturulmuştur[31]. Algoritmanın anlatımı ve görsel sonuçların gösterimi esnasında tüm veri setleri kullanılacaktır. Performans analizleri ve kıyaslamalar için ise sadece B1, B2, B3, B4 ve B5 veri seti kullanılacaktır.

Yükseklik, veri setinin ait olduğu düzlemdeki y değerlerinin alabileceği aralık olarak tanımlanabilir. Bu değer sayesinde haritanın sınırları oluşturulur. Özellikle üst ve alt tarafı veri içermeyen veri setlerinde bu parametrenin girilmesi doğru haritalandırma adına önemlidir.

Genişlik, yükseklik parametresi gibi x değerlerinin alabileceği aralık olarak tanımlanabilir. Sağ ve sol taraflarda veri içermeyen veri setleri bu parametre ile daha doğru görsellenebilir.

Yoğunluk eşiği, algoritma içerisindeki karelerin daha küçük karelere bölünüp bölünmeyeceği konusunda karar vermek için kullanılır. Buna göre eğer ilgili kare yeterli yoğunluğa sahip değilse daha küçük karelere bölünür. Özellikle ada ya da kıyı gibi bir kısmı deniz (ya da veri içermeyen) karelerin daha doğru gösterilmesi için bu parametre gereklidir.

Baskınlık eşiği, karelerin içerisindeki verilerin baskınlığına göre tekrar bölünmesini sağlamaktadır. Örneğin yüzde 70 kırmızı (tip 1) ve yüzde 30 yeşil (tip 2) veri içeren

bir karenin daha küçük dört kareye bölünüp bölünmeyeceği bu eşik kullanılarak hesaplanır. Bir karenin baskınlık değeri iki tip nokta arasındaki farkın yüzdesel ifadesi olarak tanımlanabilir. Örneğin sözü edilen yüzde 70'e yüzde 30'luk veri içeren bir karenin baskınlık değeri 0.4 olarak hesaplanır. Bu hesaplama

$$|kirmizi - yesil|/toplamnokta$$

formülü kullanılarak oluşur. Bu örneğe göre baskınlık eşiği 0.5 seçilirse bu kare daha küçük dört adet kareye bölünür. Eğer baskınlık eşiği 0.3 seçilirse bu kare bölünmez. Dolayısıyla baskınlık eşiğinin yükseltilmesi daha küçük karelerin elde edilmesini sağlar. Bu parametrenin değiştirilmesiyle farklı haritalar elde edilmesi mümkündür.

İlk kare boyutu, veri setinin yükseklik ve genişlik bilgilerine göre algoritmanın ilk aşamasında ne büyüklükteki karelere bölüneceğinin bilgisini içerir. Sadece bu parametrenin küçültülmesi daha detaylı haritanın oluşturulmasını sağlar.

Derinlik, ilk karelerin oluşturulmasından sonra bu karelerin kaç kere daha bölünebileceğinin sınırını ifade eden bir değerdir. Her bir karenin içerisinde kendisine ait derinlik değeri tutulmaktadır. İlk kareler için bu değer 0 olarak belirlenmiştir. Kare her bölündüğünde yeni oluşan dört kare için bu değer bir önceki karenin bir fazlası olarak revize edilir. Eğer derinlik değerine ulaşan herhangi bir kare olursa baskınlık ve yoğunluk gibi değerlere bakılmaksızın kare bölünmesi durdurulur. Bu parametre sayesinde farklı yakınlaştırma seviyelerinde farklı sonuçlar elde etmek mümkündür. Buna göre aynı veri seti için daha büyük derinlik parametreleri verilerek daha uzun zamanda daha ayrıntılı çıktılar almak mümkündür.

4.2.2 Önerilen Metodolojinin Çıktıları

Algoritmamızın çıktıları aşağıda sıralanmıştır.

- Nihai harita görseli
- Voronoi diyagramı içerisindeki hücre sayısı
- Algoritmanın çalışma süresi
- Algoritmanın hafıza kullanımı

- Toplam entropi değeri

Nihai harita görseli, veri setinin son kullanıcıya daha anlamlı sunulabilmesi için üretilen arka planında bir taban haritanın yer aldığı bir görseldir. Bu taban haritanın üzerine renklendirilmiş Voronoi diyagramı şeffaf bir şekilde yerleştirilmiştir. Nihai haritanın haricindeki çıktılar son kullanıcıya gösterilmesi gereken çıktılar değildir. Diğer çıktıların üretilme sebebi tez çalışmalarımızın performans analizini oluşturmak ve diğer metodlarla kıyaslama yapmaktır.

Voronoi diyagramı içerisindeki hücre sayısı, veri setinin kaç farklı alana bölündüğü olarak ifade edilebilir. Bu değer toplam kare sayısına da eşittir.

Algoritmanın çalışma süresi, okuma süresi ve çizim süresi hariç algoritmanın toplam çalışma süresini ifade etmektedir. Bu sürenin büyüklüğü özellikle dinamik haritalandırmalarda önemlidir.

Algoritmanın hafıza kullanımı, veri setinin okunması ve son çizim hariç algoritmanın çalışması için gereken toplam hafıza miktarıdır.

Toplam entropi değeri, haritanın algoritmanın çalışması neticesinde aldığı son şekil baz alınarak hesaplanan karmaşıklık değeridir.

4.2.3 İlk Karelere Bölütleme

Bu tez çalışmasında önerilen metodun başlangıcında girdi olarak verilen ilk kare büyüklüğüne göre bir başlangıç bölütlemesi yapılmaktadır. Buna göre öncelikle veriler okunur. Daha sonrasında haritanın yükseklik ve genişlik verileri göz önünde bulundurularak ilk kareler oluşturulur. Bu karelerin kenarının uzunluğu ilk kare büyüklüğü girdisine göre belirlenir. Sonrasında veri seti içerisindeki noktalar bu karelerin içerisine yerleştirilir. Oluşturulan bu ilk karelerin derinlik değeri 0 olarak atanır.

Her bir kare sınıfının içerisinde $x1$, $x2$, $y1$ ve $y2$ değerleri bulunur. Bu değerler karenin sınırlarını ifade eder. Ayrıca her bir karenin derinlik değeri ve noktalar listesi vardır. Şekil 4.20 içerisinde kare sınıfının yapısı gösterilmiştir.

```

class Kare:
    var x1, x2, y1, y2;
    var noktalar[];
    var derinlik;

```

Şekil 4.20: Kullanılan kare sınıfı

Okunan veri setindeki noktalar ilk kareler hesaplandıktan sonra bu karelerin içerisine aktarılır. Tablo 4.4 içerisinde aktarma fonksiyonu gösterilmiştir.

Tablo 4.4: Noktaların karelere bölünmesi

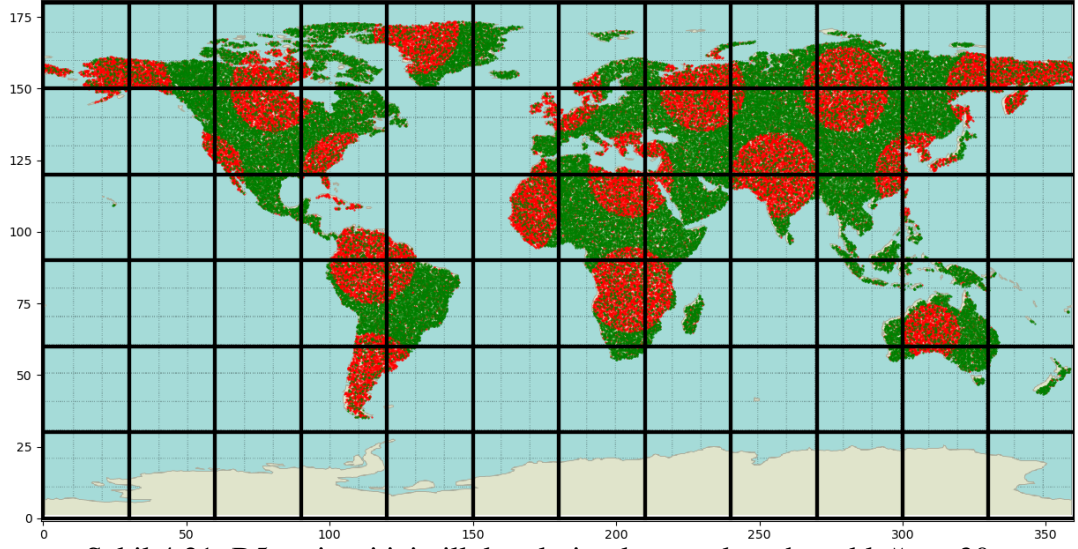
Fonksiyon 1: İlk karelerin oluşturulması

```

function ilkKareleriOlustur(var yukseklik,var genislik, var buyukluk,var noktalar[]):
    kareler = [];
    for x in range(math.ceil(yukseklik/buyukluk)):
        for y in range(math.ceil(genislik/buyukluk)):
            x1 = x * buyukluk;
            y1 = y * buyukluk;
            if(x + 1) * buyukluk > yukseklik :
                x2 = yukseklik;
            else:
                x2 = (x + 1) * buyukluk;
            if(y + 1) * buyukluk > genislik :
                y2 = genislik;
            else:
                y2 = (y + 1) * buyukluk;
            kareler.append(new Kare(x1,x2,y1,y2,0));
    w = math.ceil(genislik/buyukluk);
    for nokta in noktalar:
        indeks = int( nokta[x] / buyukluk ) * w;
        indeks += int( nokta[y] / buyukluk );
        kareler[indeks].noktalar.append(nokta);
    return kareler;

```

İlk karelerin oluşturulması sonucunda meydana gelen durum Şekil 4.21 içerisinde gösterilmiştir.



Şekil 4.21: B5 veri seti için ilk karelerin oluşumu kare büyüklüğü = 30

4.2.4 İlk Karelerin Daha Küçük Karelere Bölünmesi

İlk karelerin oluşturulmasından sonra bu kareler bir kuyruk yapısı içerisine atılır. Sırası gelen kare yoğunluk eşiği ve baskınlık eşiği girdilerine göre daha küçük dört kareye bölünür. Bölünen kareler tekrar kuyruğa eklenir. Yoğunluk eşiği ve baskınlık eşiğini geçen kareler yeni bir listeye alınır. Ayrıca derinlik limitini aşan kareler de aynı listeye eklenir. Son oluşan liste Voronoi diyagramlarının oluşturulması için bir sonraki fonksiyona gönderilir. Tablo 4.5 içerisinde bölünme fonksiyonu gösterilmiştir.

Tablo 4.5: Karelerin daha küçük karelere bölünmesi

Fonksiyon 2: Kareleri bölme işlemi**function kareleriBol(var kareler,var derinlik, var yogunluk, var baskinlik):***karetemp* = *kareler*;*sonkareler* = [];**while** len(*karetemp*) > 0:*kare* = *karetemp.pop()*;**if** len(*kare.noktalar*) == 0:*sonkareler.append(kare)*;**else:***kirmizi* = 0;*yesil* = 0;**for** *nokta* in *kare.noktalar*:**if** *nokta[renk]* == *kirmizi*:*kirmizi* = *kirmizi* + 1;**else:***yesil* = *yesil* + 1;**if** *kare.derinlik* >= *derinlik*:*sonkareler.append(kare)*;**else if** float(len(*kare.noktalar*)) /float((*kare.x2-kare.x1*)*(*kare.y2-kare.y1*))< *yogunluk* **or**float(math.fabs(*kirmizi-yesil*))/float(*kirmizi+yesil*) < *baskinlik*:*karetemp.push(dordeBol(kare))*;**else:***sonkareler.append(kare)*;**return** *sonKareler*;

Tablo 4.5 içerisinde gösterilen bölünme fonksiyonu içerisindeki dörde bölme işlemi Tablo 4.6 içerisinde gösterilmiştir.

Şekil 4.22 içerisinde karelerin bölünmesi sonrasındaki durum gösterilmiştir. Özellikle şeklin sol tarafında bulunan Hawaîi adasındaki bölümlendirme yoğunluk eşiği verisinin öneminin bir yansımasıdır. Buna göre ada yoğunluğu yeterli gelmediğinden bölütleme işlemi devam etmiştir. Kıtaların kıyı kesimlerinde ve Akdeniz bölgesinde de görüldüğü üzere veri bulunan kıyılar çok net şekilde bölütlenebilmiştir.

Tablo 4.6: Kareleri dörde bölme işlemi

Fonksiyon 3: Kare bölme fonksiyonu

function dordeBol(var kare):

merkezx = (kare.x1 + kare.x2)/2.0;

merkezy = (kare.y1 + kare.y2)/2.0;

kareler = [];

k1 = new Kare(kare.x1,merkezx,kare.y1,merkezy,kare.derinlik+1);

k2 = new Kare(kare.x1,merkezx,merkezy,kare.y2,kare.derinlik+1);

k3 = new Kare(merkezx,kare.x2,kare.y1,merkezy,kare.derinlik+1);

k4 = new Kare(merkezx,kare.x2,merkezy,kare.y2,kare.derinlik+1);

for nokta in kare.noktalar:

if nokta[x] < merkezx:

if nokta[y] < merkezy:

 k1.noktalar.append(nokta);

else:

 k2.noktalar.append(nokta);

else:

if int(nokta[y]) < merkezy:

 k3.noktalar.append(nokta);

else:

 k4.noktalar.append(nokta);

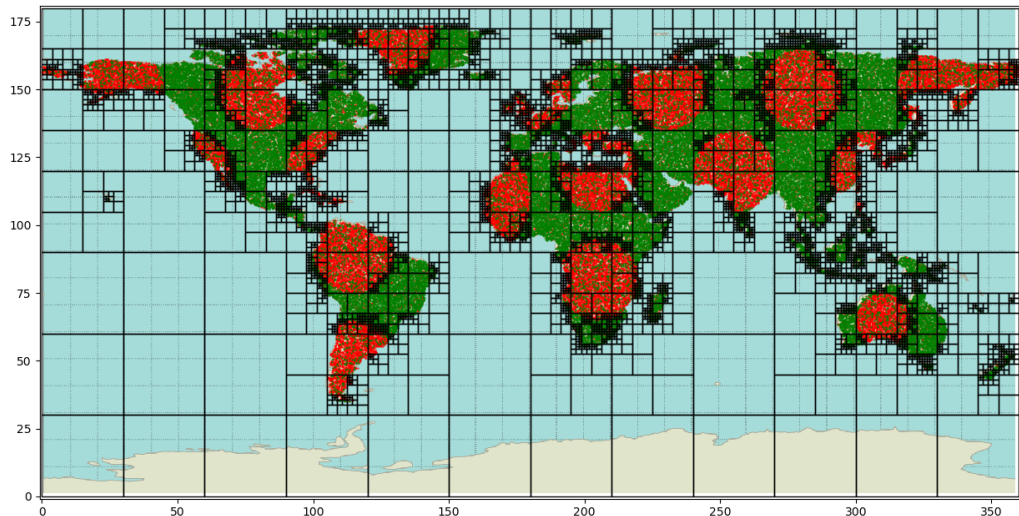
kareler.append(k1);

kareler.append(k2);

kareler.append(k3);

kareler.append(k4);

return kareler;



Şekil 4.22: B5 veri seti için bölünen kareler yoğunluk eşiği = 5, baskınlık eşiği = 0.4, derinlik = 5

4.2.5 Kareler İçerisinde Rastgele Noktalar Seçilmesi ve Voronoi Diyagramlarının Oluşturulması

Nihai karelerin oluşturulmasını takiben bu kareler içerisinde rastgele noktalar seçilmektedir. Karelerin orta noktalarının değil de rastgele noktaların seçilme sebebi Voronoi diyagramlarının problemsiz oluşturulmasıdır.

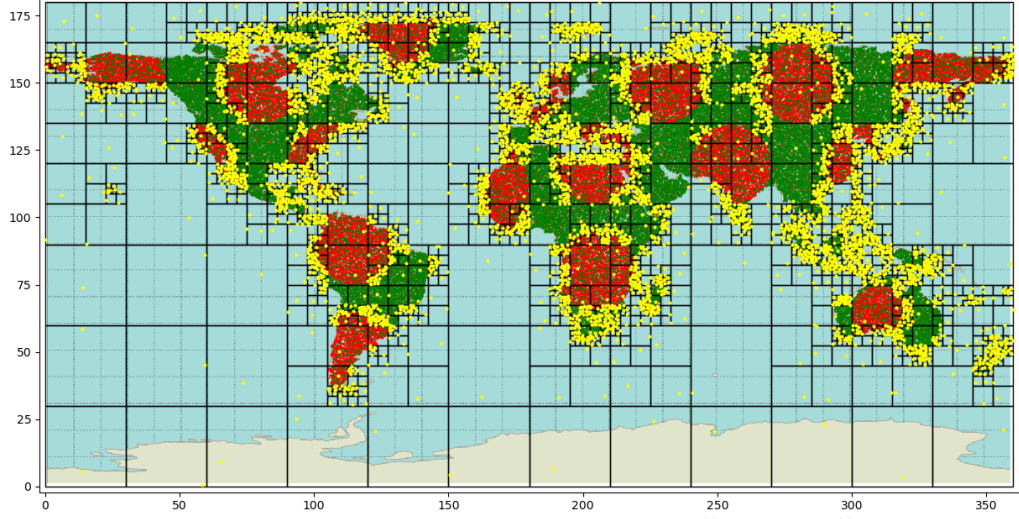
Voronoi diyagramlarında doğrusal (collinear) noktaların gösteriminde bazı problemler yaşanmaktadır [39]. Bu problemlerden kaçınmak amacıyla Voronoi diyagramında kullanılacak olan merkez noktalar karelerin içerisinde rastgele noktalar seçilmiştir. Tablo 4.7 içerisinde noktaların seçildiği ve Voronoi diyagramına bu noktaların gönderildiği fonksiyon ifade edilmiştir.

Tablo 4.7: Rastgele noktaların seçilmesi ve Voronoi diyagramlarının oluşturulması

Fonksiyon 4: Rastgele noktaların seçilmesi ve Voronoi diyagramlarının oluşturulması

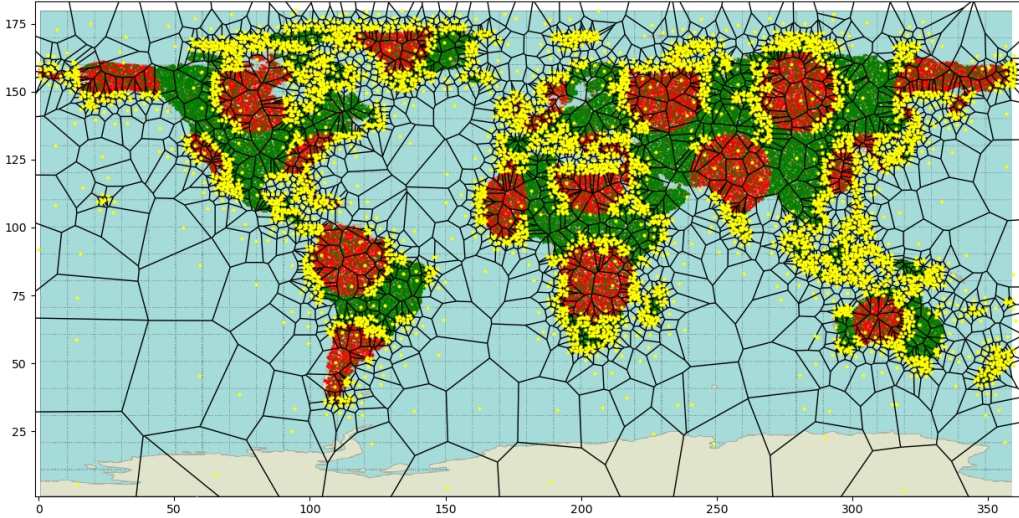
```
function voronoiOlustur(var kareler[]):  
    merkezNoktalar = [];  
    Voronoi voronoi;  
    for kare in kareler:  
        merkezNoktalar.append(RandomInside(kare));  
    voronoi = Voronoi(merkezNoktalar);  
    return voronoi;
```

Merkez noktalarının oluşturulması sonucunda oluşan durum Şekil 4.23 içerisinde gösterilmiştir. Seçilen merkez noktalar büyük sarı noktalarla ifade edilmiştir.



Şekil 4.23: B5 veri seti için seçilen rastgele noktalar yoğunluk eşiği = 5, baskınlık eşiği = 0.4, derinlik = 5

Oluşturulan bu noktalar merkez noktası olarak Voronoi diyagramı içerisine gönderilmektedir. Şekil 4.24 içerisinde bu merkez noktalar seçilerek oluşturulan Voronoi diyagramı gösterilmiştir.



Şekil 4.24: B5 veri seti için oluşturulan Voronoi diyagramı yoğunluk eşiği = 5, baskınlık eşiği = 0.4, derinlik = 5

4.2.6 Ball-tree Algoritması Kullanılarak Voronoi Diyagramlarının Renklendirilmesi

Voronoi diyagramlarının oluşturulması sonrasında bu diyagramın içerisindeki hücrelerin renklendirilmesi gerekmektedir. Bu amaçla Ball-tree algoritmasından faydala-

nılmıştır. Bu algoritma öncelikle n adet nokta verilerek kurulur. Algoritma bu noktalardan bir ağaç yapısı meydana getirir. Bu ağaç yapısı kullanılarak yeni bir noktanın bu n nokta içerisinde hangisine en yakın olduğu $O(\log n)$ zaman karmaşıklığı içerisinde bulunmuş olur. Tablo 4.8 içerisinde Ball-tree algoritması kullanılarak Voronoi diyagramlarının renklendirilmesi gösterilmiştir.

Tablo 4.8: Ball-tree algoritması ile Voronoi diyagramının renklendirilmesi

Fonksiyon 5: Ball-tree algoritması ile Voronoi diyagramının renklendirilmesi

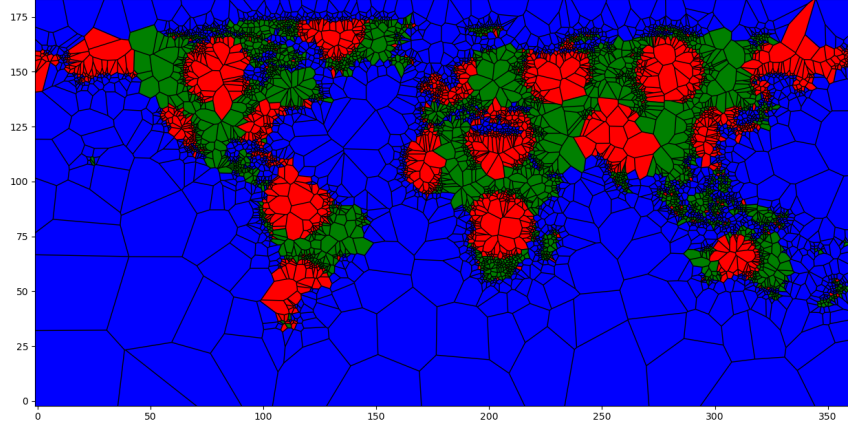
```

function voronoiRenklendir(var voronoi, var noktalar[]):
    renkListesi[] = null;
    BallTree btree = Baltree(voronoi.merkezNoktalar);
    for nokta in noktalar:
        indeks = btree.query(nokta);
        if list[indeks] == null:
            if nokta[renk] == kirmizi:
                renkListesi[indeks] = -1;
            else:
                renkListesi[indeks] = 1;
        else:
            if nokta[renk] == kirmizi:
                renkListesi[indeks] += -1;
            else:
                renkListesi[indeks] += 1;
    return renkListesi;

```

Renklendirme amacıyla öncelikle boş bir renk listesi oluşturulur. Renk listesinin büyüklüğü Voronoi hücre sayısına eşittir. Sonrasında Ball-tree algoritması içerisinde merkez noktalar gönderilir. Daha sonrasında veri seti içerisindeki her bir noktanın en yakın olduğu merkez nokta Ball-tree algoritmasından sorgulanır. Sorgulamanın sonucu renk listesine kaydedilir.

Her bir veri seti noktası için renk listesinde karşılık gelen değer arttırılır ya da azaltılır. Renk listesindeki çoğunluğa göre Voronoi diyagramı renklendirilmiş olur. Buna göre Voronoi hücresine karşılık gelen renk listesi ögesi boşsa o hücre şeffaf açık mavi renkle ifade edilir. Voronoi hücresine denk gelen renk listesindeki değer negatifse ya da 0 ise hücre kırmızıyla ifade edilir. Eğer denk gelen liste pozitifse yeşil renkle ifade edilir. İki grubun eşit çıkması durumunda oluşan 0 değerinin kırmızı renkle ifade edilme sebebi kırmızı noktaların toplamda yeşil noktalardan daha fazla sayıda olmasıdır. Şekil 4.25 içerisinde renklendirilmiş Voronoi diyagramları gösterilmiştir.

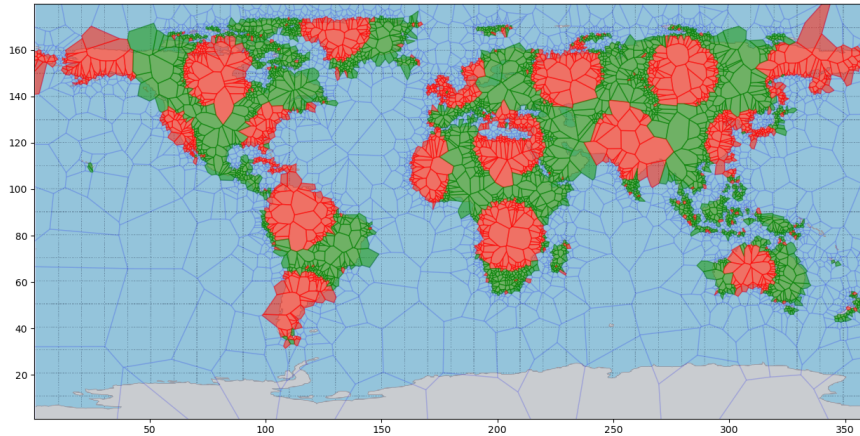


Şekil 4.25: B5 veri seti için oluşturulan Voronoi diyagramı (renkli) yoğunluk eşiği = 5, baskınlık eşiği = 0.4, derinlik = 5

Bu işlemler sonucunda çok sayıda konumsal verinin sınıflandırılması bitirilmiş olur.

4.2.7 Taban Haritası ve Renklendirilmiş Voronoi Diyagramları ile Nihai Görüntünün Elde Edilmesi

Son kullanıcıya çok sayıda konumsal verinin daha anlaşılır bir şekilde gösterilebilmesi amacıyla renklendirilmiş Voronoi diyagramının arkasına taban haritası yerleştirilmiştir. Renklendirilmiş Voronoi diyagramının taban haritasını kapatmaması adına diyagram şeffaf bir şekilde gösterilmiştir. Şekil 4.26 içerisinde tüm algoritmanın çalışması sonucunda oluşan harita gösterilmiştir. Bu gösterimde kırmızı ve yeşil alanların şeffaflığı 0.5, veri içermeyen hücrelerin alanları ise 0.1 şeffaflık oranında etki etmiştir.



Şekil 4.26: B5 veri seti için nihai harita yoğunluk eşiği = 5, baskınlık eşiği = 0.4

Tüm bu işlemlerin bulunduğu genel yapı Tablo 4.9 içerisinde gösterilmiştir.

Tablo 4.9: Tüm fonksiyonların bulunduğu algoritma işleyişi

Fonksiyon 6: Tüm fonksiyonların bulunduğu algoritma işleyişi

```
function Main(var noktalar[],var yukseklik,var genislik  
var buyukluk,var derinlik, var yogunluk, var baskinlik,var tabanHaritasi):  
    ilkKareler= ilkKareleriOlustur(yukseklik,genislik,buyukluk,noktalar[]);  
    sonKareler= kareleriBol(ilkKareler, derinlik, yogunluk, baskinlik);  
    Voronoi voronoi= voronoiOlustur(sonKareler);  
    renkListesi= voronoiRenklendir(voronoi, noktalar[]);  
    Goster(voronoi, renkListesi, tabanHaritasi);  
return;
```

BÖLÜM 5

TESTLER VE SONUÇLAR

Bu bölümde bizim önerdiğimiz metodun hangi parametrelerle hangi çıktıları ürettiğini ve üretilen çıktıların sayısal değerlerini anlatıyor olacağız. Parametrelerin çıktılarına görsel ve sayısal olarak nasıl etki ettiği hakkında bilgiler vereceğiz. Ayrıca Bilgehan Çakmak tarafından yapılan benzer çalışma ile de kendi yöntemimizi kıyaslayacağız.

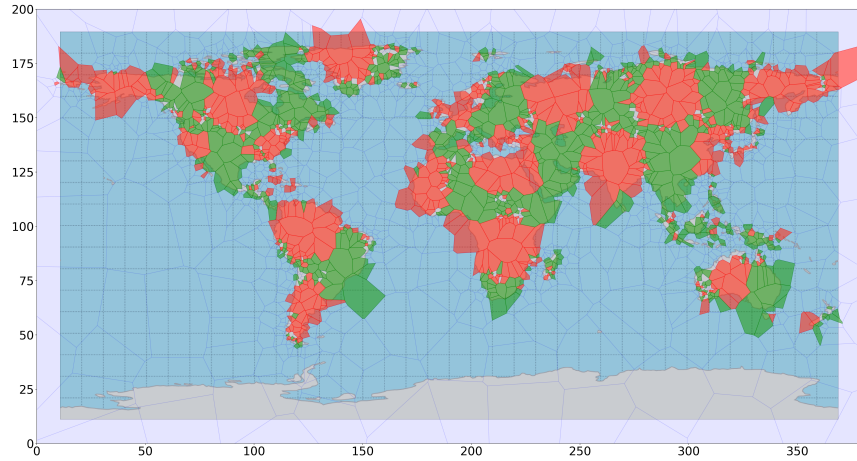
5.1 Performans Analizleri ve Kıyaslamalar

5.1.1 Görsel Analizler

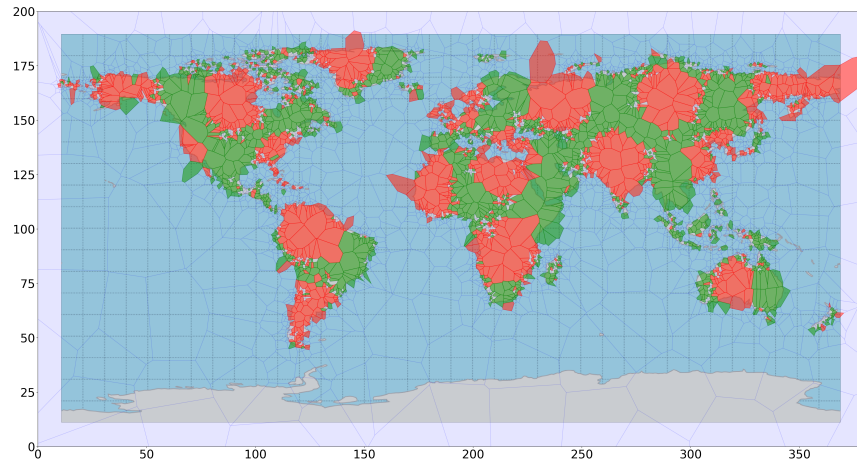
Yöntemimiz 7 farklı veri seti üzerinde test edilmiştir. Bunlar içerisinde en çok B1, B2, B3, B4 ve B5 adlı veri setlerine odaklanacağız. Metodumuzun nasıl sonuçlar verdiğini analiz edebilmemiz için farklı derinlikler kullandık. Buna göre her veri seti için başlangıç kare büyüklüğü 30 olarak seçilmiştir. Derinlik parametresinde ise 2, 3, 4, 5 ve 6 değerleri seçilmiştir. Yoğunluk eşiği her veri seti için ayrı ayrı belirlenmiştir. Her veri seti için 5 farklı yoğunluk eşiği belirledik. Baskınlık eşiğinde 0.3, 0.4, 0.5, 0.6 ve 0.7 değerleri belirlenmiştir. Tüm bu test durumları göz önünde bulundurulduğunda 625 adet sonuç elde edilmektedir. Karmaşaya sebep olmaması açısından bu sonuçlar veri setine, derinlik girdisine, yoğunluk eşiğine ve baskınlık eşiğine göre diğer parametreler sabit tutularak aktarılacaktır.

5.1.1.1 Veri Setlerine Göre Analizler

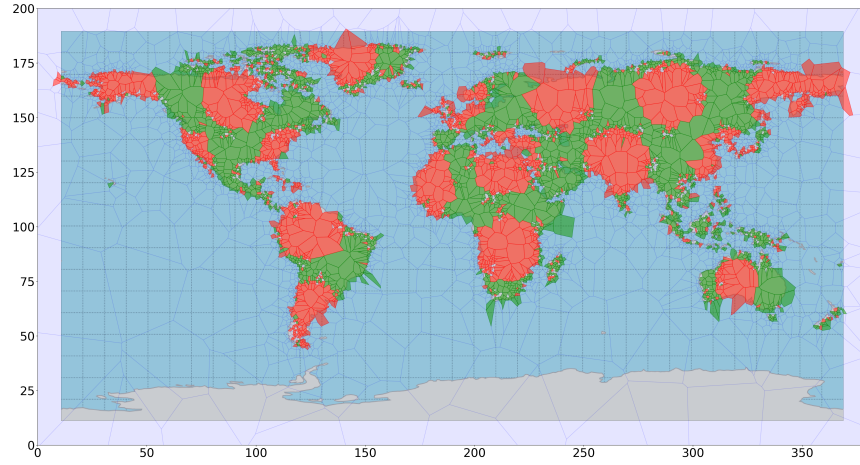
Çalışmamızda kullandığımız B1, B2, B3, B4, B5, Türkiye ve Hava Kalitesi isimli veri setlerimizde sırasıyla 7160, 14323, 28350, 68090, 127627, 200000 ve 6373 adet nokta bulunmaktadır. Farklı nokta sayılarına sahip bu setler için aynı yoğunluk eşiği değerinin kullanılması uygun olmayacaktır. Bunun dışında derinlik ve baskınlık eşiği gibi parametreler sabit tutulmuştur. Tüm setler için baskınlık değeri 0.4 olarak, derinlik değeri 5 olarak belirlenmiştir. Bu parametreler ışığında görsel sonuçlarımız sırasıyla Şekil 5.1, 5.2, 5.3, 5.4, 5.5, 5.6 ve 5.7 içerisinde aktarılmıştır.



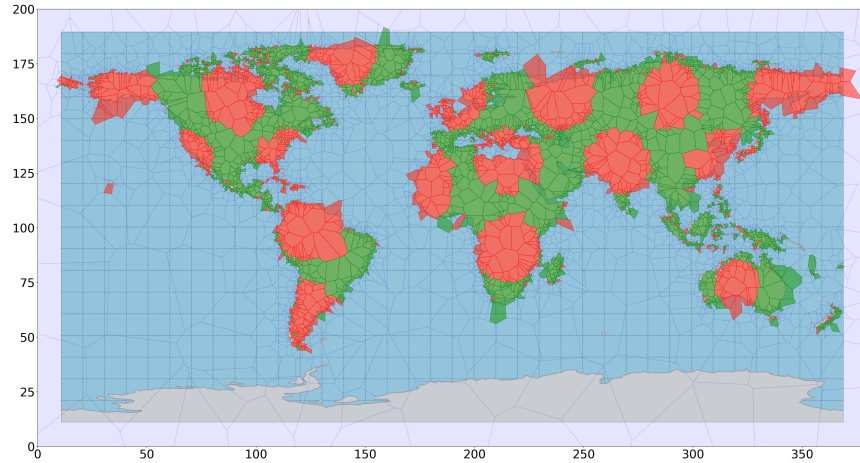
Şekil 5.1: Veri seti B1, 7160 nokta, yoğunluk eşiği = 0.25, hücre sayısı = 2486, algoritma zamanı = 0.1306164 saniye, algoritma ram miktarı = 3.827621 megabayt



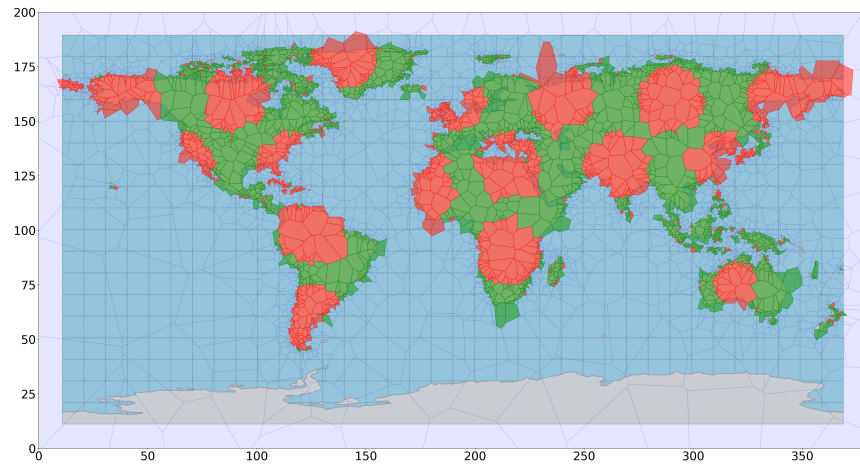
Şekil 5.2: Veri seti B2, 14323 nokta, yoğunluk eşiği = 0.7, hücre sayısı = 5600, algoritma zamanı = 0.317150 saniye, algoritma ram miktarı = 8.403469 megabayt



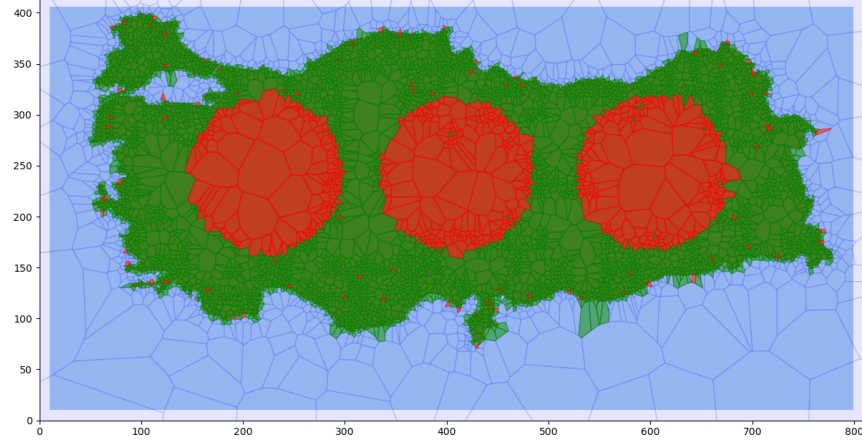
Şekil 5.3: Veri seti B3, 28350 nokta, yoğunluk eşiği = 1.5, hücre sayısı = 7337, algoritma zamanı = 0.591450 saniye, algoritma ram miktarı = 12.320357 megabayt



Şekil 5.4: Veri seti B4, 68090 nokta, yoğunluk eşiği = 3.0, hücre sayısı = 6041, algoritma zamanı = 1.075125 saniye, algoritma ram miktarı = 16.266073 megabayt

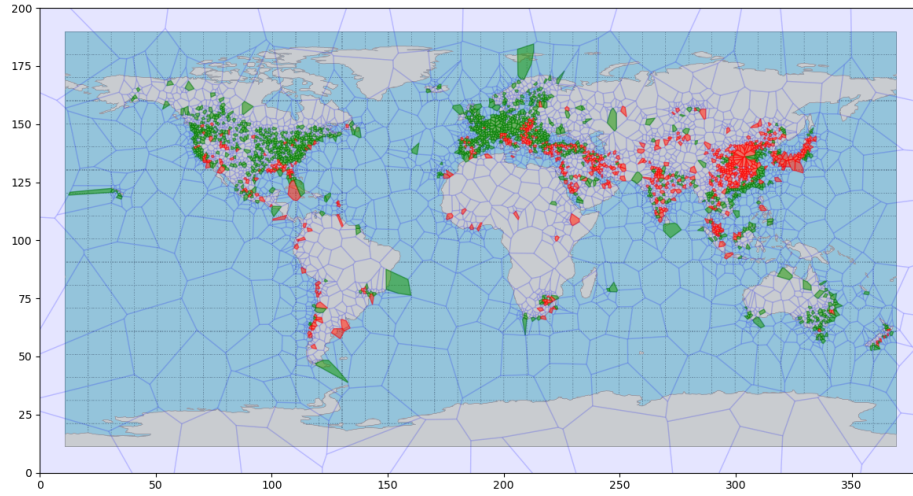


Şekil 5.5: Veri seti B5, 127672 nokta, yoğunluk eşiği = 5.0, hücre sayısı = 5468, algoritma zamanı = 2.110353 saniye, algoritma ram miktarı = 23.769349 megabayt



Şekil 5.6: Veri seti Türkiye, 200000 nokta, yoğunluk eşiği = 0.8, hücre sayısı = 6724, algoritma zamanı = 3.203463 saniye, algoritma ram miktarı = 35.097085 megabayt

Özellikle Şekil 5.1, 5.2, 5.3, 5.4 ve 5.5 incelendiğinde veri seti içerisindeki noktaların değişmesine karşın ortaya çıkan görsellerin çok büyük değişiklikler göstermediği gözlemlenmektedir. Uygun yoğunluk eşiği değeri verildiğinde aynı altyapı kullanılarak oluşturulan bu veri setlerinin benzer sonuçlar vermesi tutarlılık açısından önemlidir.

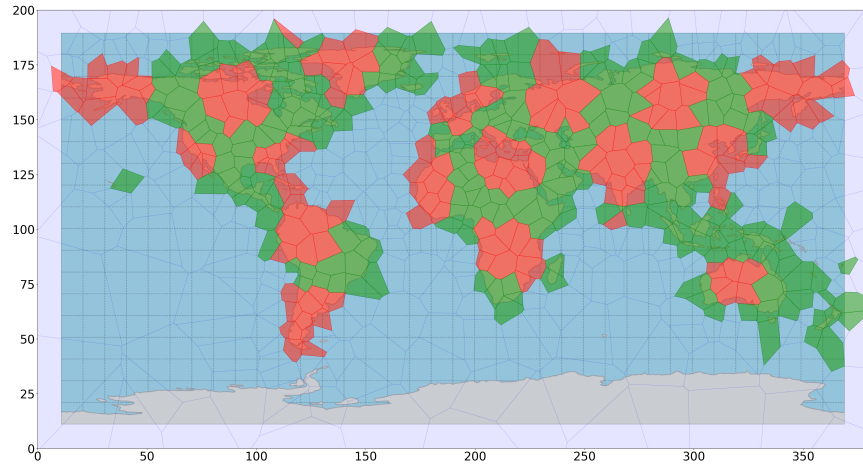


Şekil 5.7: Veri seti Hava Kalitesi, 6373 nokta, yoğunluk eşiği = 8.0, hücre sayısı = 5609, algoritma zamanı = 0.219413 saniye, algoritma ram miktarı = 7.315873 megabayt

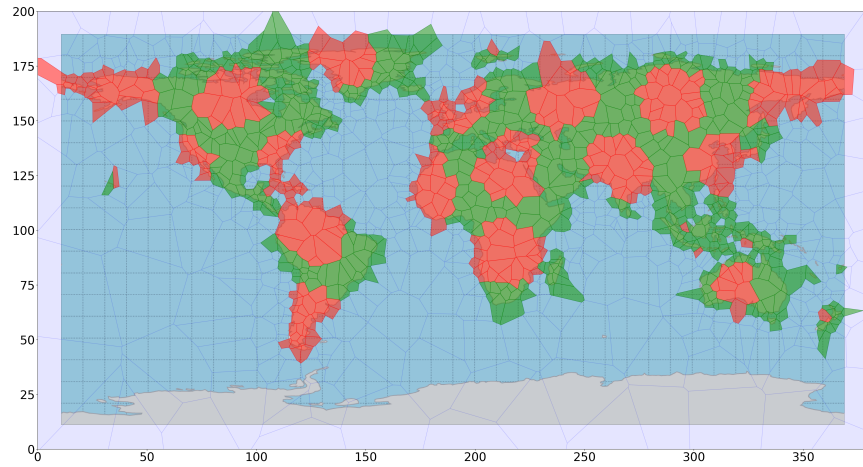
5.1.1.2 Derinlik Girdisine Göre Analizler

Derinlik parametresinin sonuçlara nasıl etki ettiğini gösterebilmek adına diğer girdiler sabit tutulmuştur. Bu amaçla B5 veri seti seçilmiştir. Yoğunluk eşiği değeri aynı veri

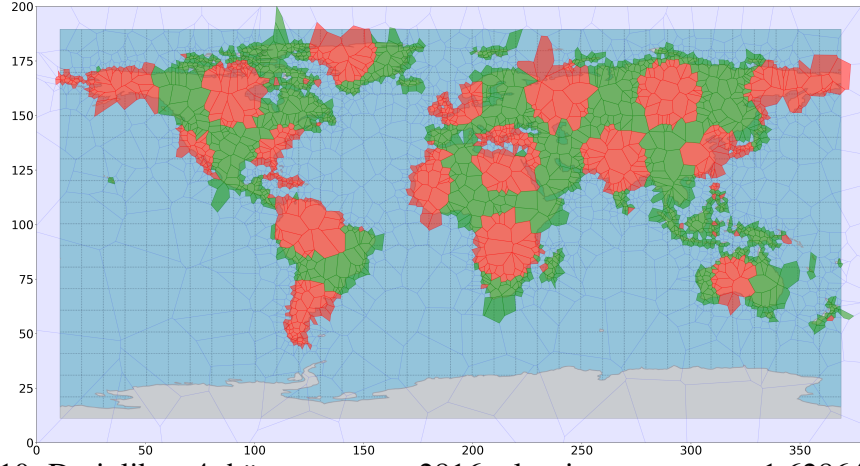
seti kullanıldığı için 5 olarak seçilmiştir. Baskınlık eşiği 0.4 olarak seçilmiştir. Derinlik parametresi kullanılarak daha ayrıntılı görseller elde etmek mümkündür. Örneğin derinlik parametresi 2 olarak seçildiğinde kareler çok fazla detaylandırılmamakta ve bunun sonucunda daha az detaya sahip görüntüler oluşmaktadır. Şekil 5.8, 5.9, 5.10, 5.11 ve 5.12 içerisinde derinlik parametresinin sırayla 2, 3, 4, 5 ve 6 olarak seçilmesi sonucu oluşan görseller aktarılmıştır.



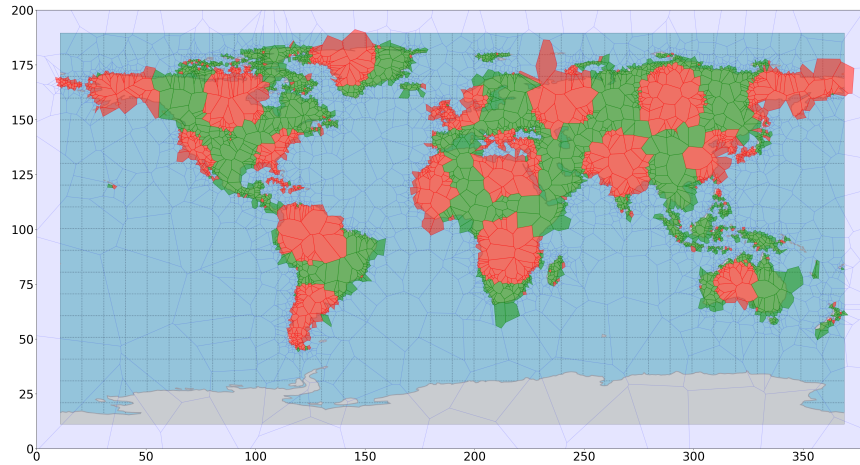
Şekil 5.8: Derinlik = 2, hücre sayısı = 626, algoritma zamanı = 1.189814 saniye, algoritma ram miktarı = 18.147149 megabayt



Şekil 5.9: Derinlik = 3, hücre sayısı = 1349, algoritma zamanı = 1.380306 saniye, algoritma ram miktarı = 18.992249 megabayt

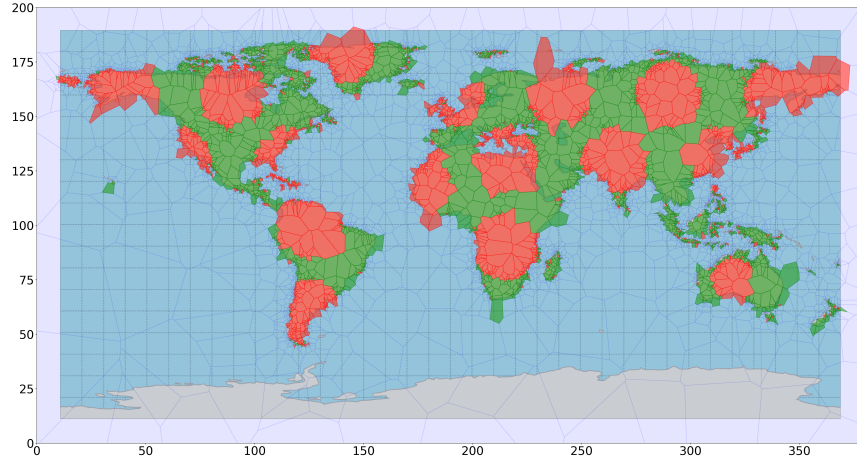


Şekil 5.10: Derinlik = 4, hücre sayısı = 2816, algoritma zamanı = 1.628641 saniye, algoritma ram miktarı = 20.695605 megabayt



Şekil 5.11: Derinlik = 5, hücre sayısı = 5468, algoritma zamanı = 2.023584 saniye, algoritma ram miktarı = 23.769349 megabayt

Derinlik parametresinin artırılması sonucunda oluşan Voronoi hücresi sayısı üstsel olarak artmaktadır. Fakat belirli bir derinlikten sonra yoğunluk eşiği ve baskınlık eşiği parametrelerine göre daha fazla bölümlendirmeye ihtiyaç duyulmamaktadır. Derinlik parametresi kullanılarak farklı yakınlaştırma seviyelerine göre daha az ya da daha çok ayrıntılı görseller elde edilebilir.



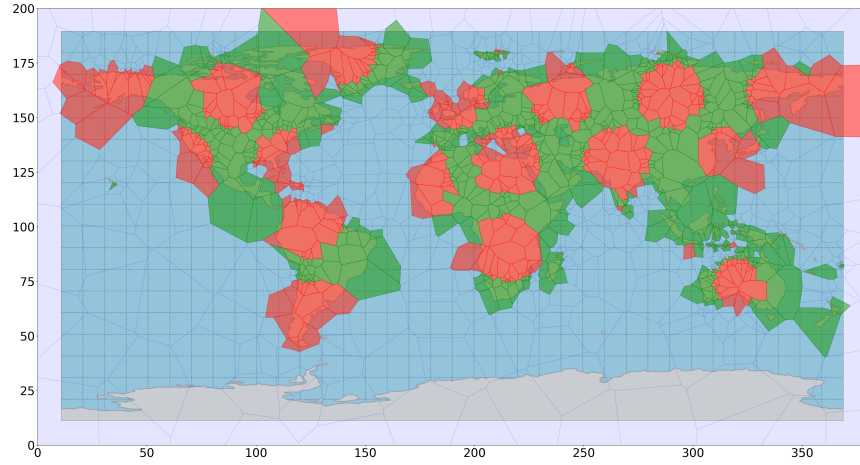
Şekil 5.12: Derinlik = 6, hücre sayısı = 8789, algoritma zamanı = 2.384620 saniye, algoritma ram miktarı = 27.588289 megabayt

5.1.1.3 Yoğunluk Eşiği Girdisine Göre Analizler

Yoğunluk eşiği özellikle kıyıların ve adaların daha iyi görselleştirilebilmesi adına önemlidir. Yoğunluk eşiği birim kare başına düşmesi beklenen nokta sayısı olarak ifade edilebilir. Buna göre eğer herhangi bir karenin yoğunluğu bu eşiğin altında kalırsa (örneğin ada ya da kıyı gibi bölgelerde) kare tekrar dört eşit parçaya bölünmektedir. Yoğunluk eşiğinin veri setine göre çok yüksek tutulması durumunda ise tamamı kara parçası içerisinde yer alan alanlarda bile eşik aşılamayacaktır. Bu durum da gereksiz bölütlemelere sebebiyet verebilir. Yoğunluk eşiği parametresi belirlenirken tamamı karalardan oluşan bir karenin yoğunluğunu tespit etmek daha doğru olacaktır.

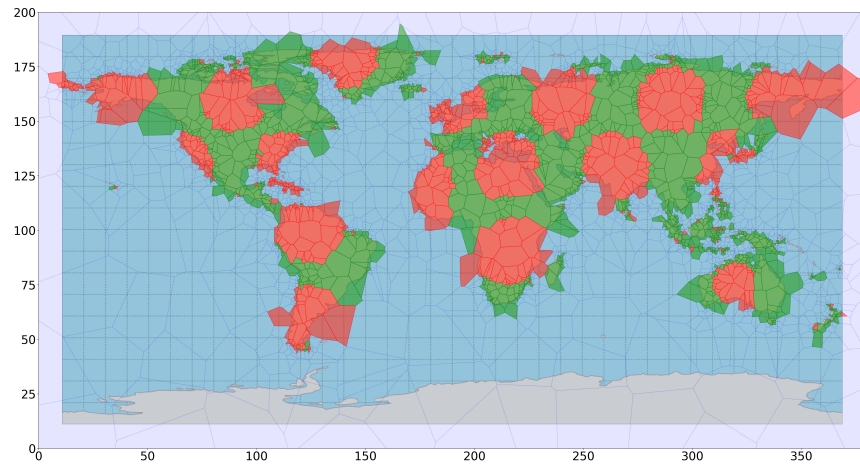
Yoğunluk eşiğinin oluşturulan görsellere nasıl bir etkisi olduğunun daha net anlaşılabilmesi için diğer parametreler sabit tutulmuştur. Bu amaçla sadece B5 veri seti kullanılmıştır. Derinlik 5 olarak belirlenmiştir. Baskınlık eşiği ise 0.4 olarak seçilmiştir.

Şekil 5.13, 5.14, 5.15, 5.16 ve 5.17 içerisinde yoğunluk eşiği parametresinin sırayla 1, 3, 5, 7 ve 9 olarak seçilmesi sonucu oluşan görseller aktarılmıştır.

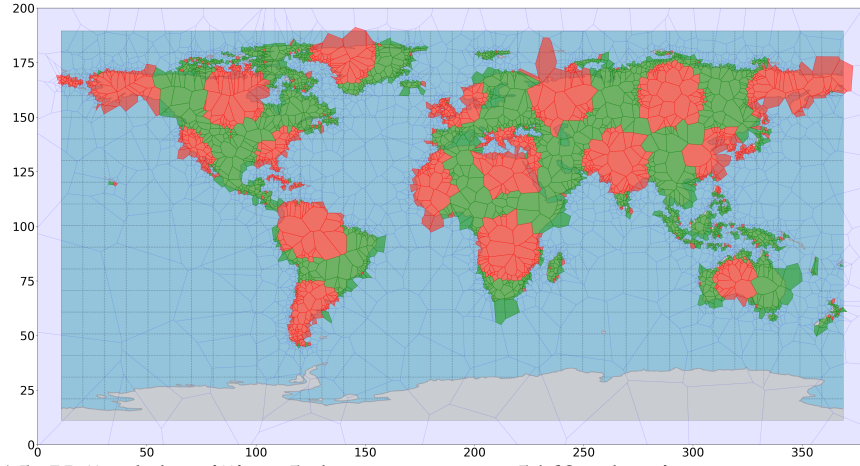


Şekil 5.13: Yoğunluk eşiği = 1, hücre sayısı = 1607, algoritma zamanı = 1.400253 saniye, algoritma ram miktarı = 19.289733 megabayt

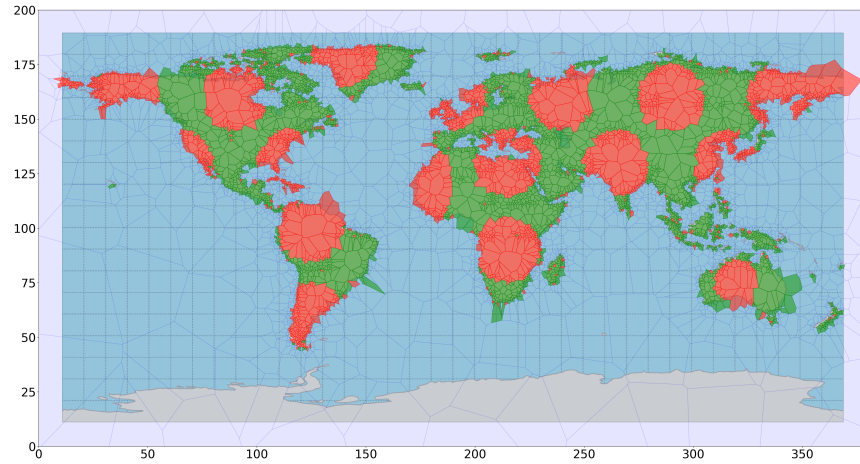
Özellikle Şekil 5.17 dikkate alındığında yoğunluk eşiğinin çok yüksek tutulduğu ve bu durum neticesinde yoğunluğu yeten bölgelerin dahi bölünmek zorunda kaldığı görülmektedir. Bu sebeple yoğunluk eşiğinin veri setine göre ayarlanması gerekmektedir.



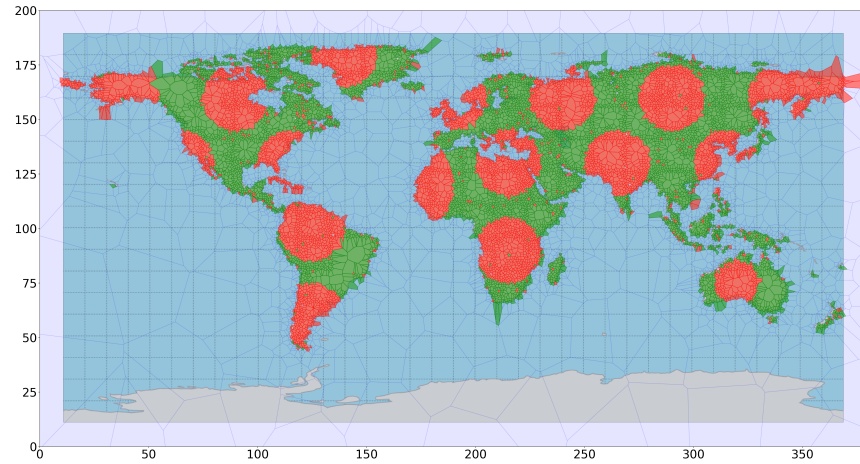
Şekil 5.14: Yoğunluk eşiği = 3, hücre sayısı = 3371, algoritma zamanı = 1.670530 saniye, algoritma ram miktarı = 21.341897 megabayt



Şekil 5.15: Yoğunluk eşiği = 5, hücre sayısı = 5468, algoritma zamanı = 1.983692 saniye, algoritma ram miktarı = 23.769349 megabayt



Şekil 5.16: Yoğunluk eşiği = 7, hücre sayısı = 8498, algoritma zamanı = 2.403601 saniye, algoritma ram miktarı = 27.274405 megabayt



Şekil 5.17: Yoğunluk eşiği = 9, hücre sayısı = 14585, algoritma zamanı = 3.104696 saniye, algoritma ram miktarı = 34.368145 megabayt

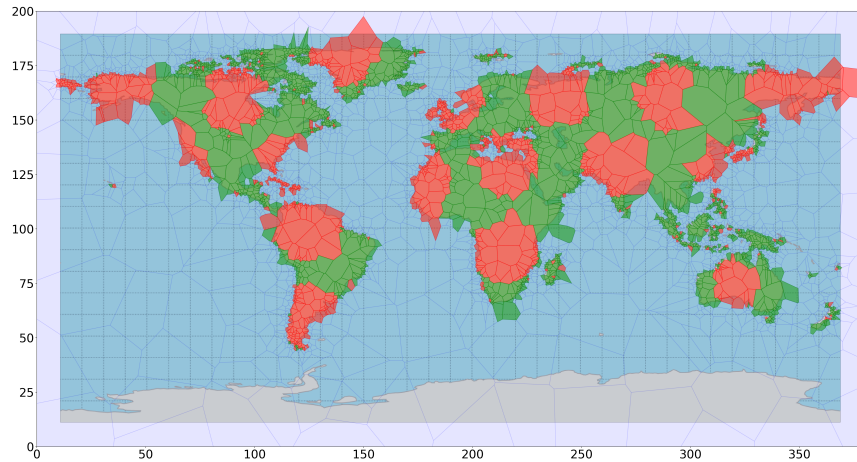
5.1.1.4 Baskınlık Eşiği Girdisine Göre Analizler

Baskınlık eşiği bir karenin içerisindeki noktaların renklendirme için yeterli oranda olup olmadığına karar vermek için kullanılır. Örneğin belirli bir kare içerisinde 70 kırmızı 30 yeşil nokta olması durumunda bu karenin baskınlık değeri 0.4 olarak hesaplanır. Bu oran $|kırmızı - yeşil|/toplamnokta$ şeklinde hesaplanır. Eğer baskınlık eşiği bu değerden küçük bir sayı (örneğin 0.3) olarak seçilirse bu kare tekrar bölünmez. Eğer baskınlık eşiği bu değerden daha büyük olursa (örneğin 0.5) bu kare baskınlık eşiğinin altına gelinceye kadar daha küçük karelere bölünür. Dolayısıyla baskınlık eşiğinin büyütülmesi sonucunda daha çok bölütleme yapılır.

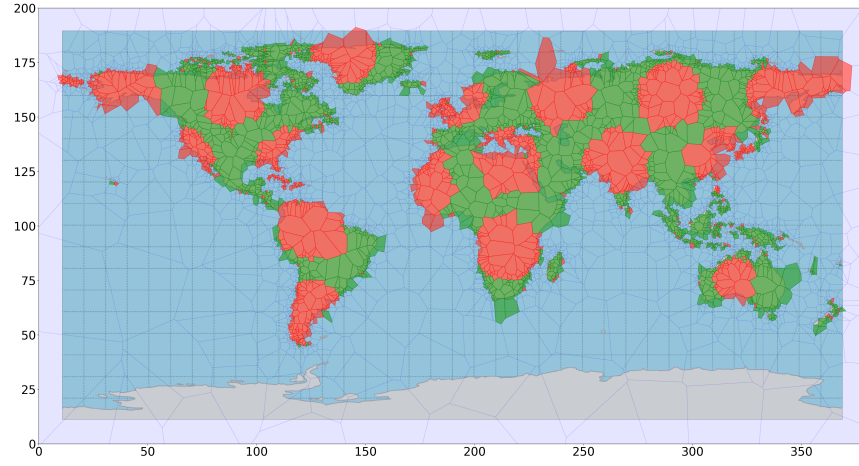
Baskınlık eşiği veri seti içerisindeki farklılaşma alanlarını ayırır. Örneğin B5 veri seti içerisindeki yeşil ve kırmızı çoğunluğun yakın olduğu daire şeklindeki baskınlık ayrımları bu parametrenin değiştirilmesiyle daha net ya da daha yumuşak şekilde görselleştirilebilir.

Baskınlık eşiği parametresindeki değişikliklerin daha iyi ifade edilebilmesi için diğer parametreler sabit tutulmuştur. Bu amaçla sadece B5 veri seti kullanılmıştır. Derinlik 5 olarak belirlenmiştir. Yoğunluk eşiği ise 5 olarak seçilmiştir.

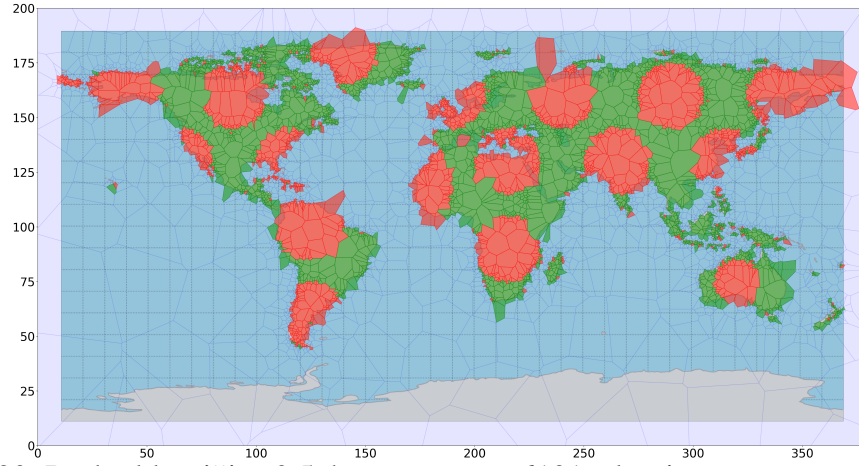
Şekil 5.18, 5.19, 5.20, 5.21 ve 5.22 içerisinde baskınlık eşiği parametresinin sırayla 0.3, 0.4, 0.5, 0.6 ve 0.7 olarak seçilmesi sonucu oluşan görseller aktarılmıştır.



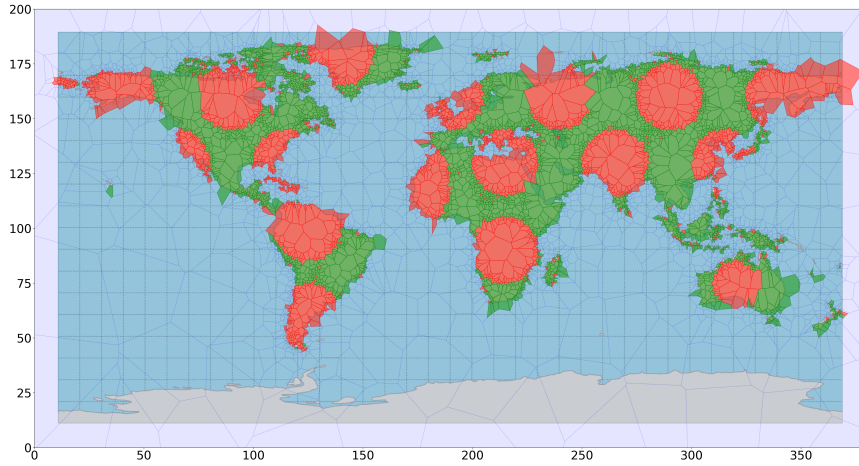
Şekil 5.18: Baskınlık eşiği = 0.3, hücre sayısı = 4883, algoritma zamanı = 2.008595 saniye, algoritma ram miktarı = 23.086893 megabayt



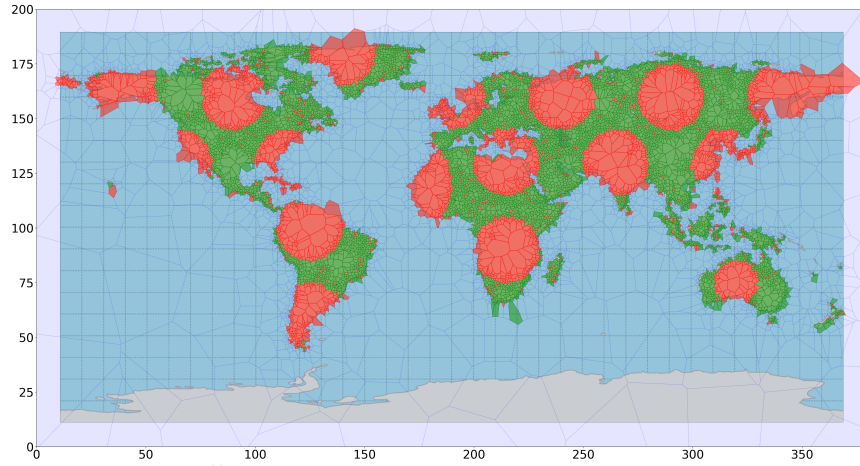
Şekil 5.19: Baskınlık eşiği = 0.4, hücre sayısı = 5468, algoritma zamanı = 2.007631 saniye, algoritma ram miktarı = 23.769445 megabayt



Şekil 5.20: Baskınlık eşiği = 0.5, hücre sayısı = 6101, algoritma zamanı = 2.021590 saniye, algoritma ram miktarı = 24.515693 megabayt



Şekil 5.21: Baskınlık eşiği = 0.6, hücre sayısı = 7343, algoritma zamanı = 2.379632 saniye, algoritma ram miktarı = 25.944469 megabayt



Şekil 5.22: Baskınlık eşiği = 0.7, hücre sayısı = 11654, algoritma zamanı = 2.701807 saniye, algoritma ram miktarı = 30.991777 megabayt

Şekil 5.22 dikkate alındığında baskınlık eşiği çok yüksek tutulmuş ve bunun sonucunda bölünmemesi gereken alanlar dahi daha küçük alanlara bölünmüştür. Yüzde 84 kırmızı yüzde 16 yeşil nokta içeren alanlar dahi bölünmüştür. Çünkü $|84 - 16|/100 < 0.7$ ifadesi gerçekleşmiştir. Dolayısıyla bu parametre veri setinin içerdiği karmaşık alanlar göz önünde bulundurularak belirlenmelidir.

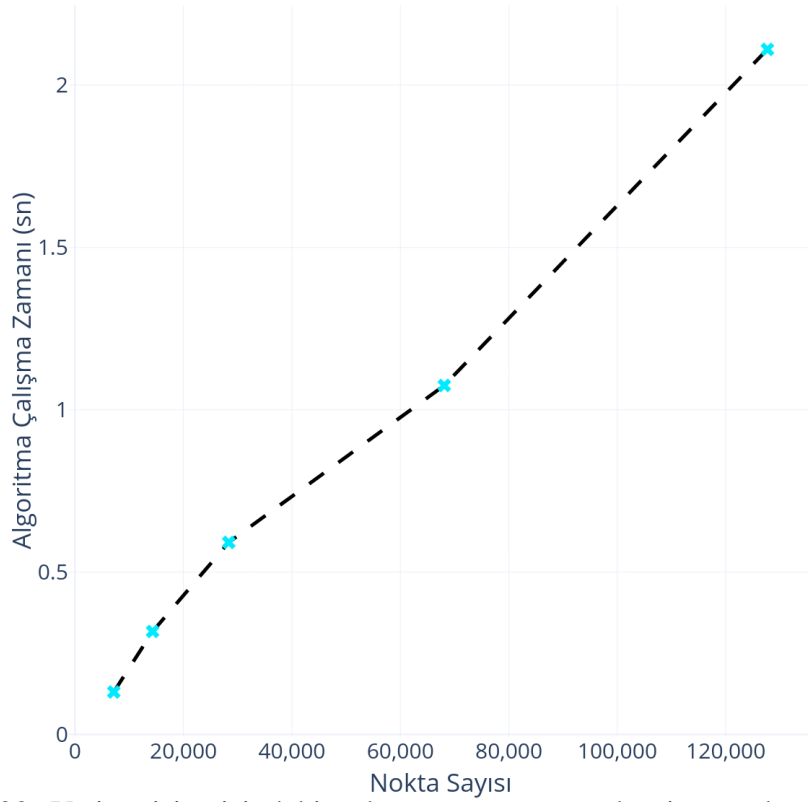
5.1.2 Sayısal Analizler

Çalışma zamanlaması bu tezin odaklandığı temel analizlerden birisidir. Bu tezin başlangıç aşamasında gerçek zamanlı ya da gerçek zamana yakın haritalandırma yapılmaya çalışılmıştır. Önerdiğimiz algoritma içerisinde en fazla zaman tüketen kısımlar karelere bölme ve Ball-tree algoritmasının çalıştırılmasıdır. Ball-tree algoritması karelere bölmekten daha fazla zaman tükettiği için algoritmanın zaman karmaşıklığı yaklaşık olarak Ball-tree algoritmasının Voronoi diyagramlarını renklendirmek için kullandığı kısım kabul edilebilir.

Ball-tree algoritması içerisinde arama yapmak, n merkez noktası için $O(\log n)$ zaman karmaşıklığına sahiptir. Burada n ifadesi toplam kare sayısına eşittir. Ayrıca toplam kare sayısı hem toplam merkez nokta sayısına hem de toplam Voronoi hücresi sayısına eşittir. Eğer sorgulanacak olan veriler tek tek Ball-tree algoritmasına gönderilirse zaman karmaşıklığı $O(m \log n)$ değerine eşit olur. Formüldeki m ifadesi veri seti içerisinde yer alan toplam noktaları ifade etmektedir. Dolayısıyla n adet merkez nokta

ve m adet veri seti noktası için zaman karmaşıklığı $O(m \log n)$ olarak ifade edilebilir.

Veri seti içerisindeki nokta sayısı m ile ifade edilmiştir. Bu sayıdaki değişim algoritma zamanını etkilemektedir. Buna göre veri setleri içerisinde bulunan noktaların artması algoritmamızın zaman karmaşıklığını arttırmaktadır. Şekil 5.23 içerisinde veri seti içerisindeki nokta sayısının değişimine göre algoritma zamanının değişimi gösterilmektedir. Buradan anlaşılacağı üzere nokta sayısı ile algoritmanın toplam çalışma süresi arasında doğrusal bir ilişki bulunmaktadır.



Şekil 5.23: Veri seti içerisindeki nokta sayısına göre algoritma çalışma zamanının değişimi

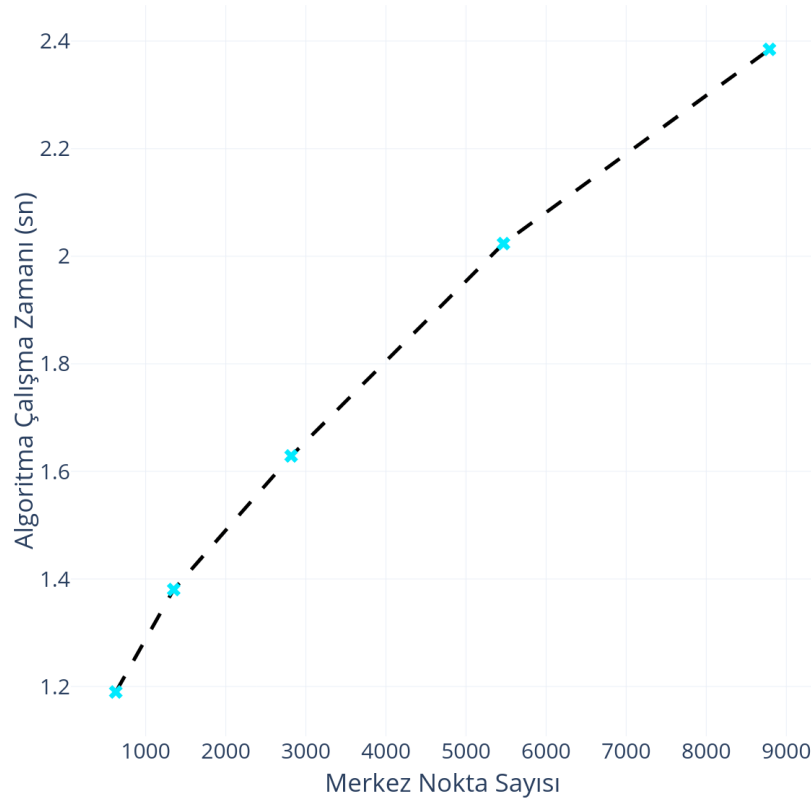
Tablo 5.1 içerisinde nokta sayısının değişimine göre algoritmanın çalışma zamanının değişimi gösterilmiştir. Ayrıca bu veriler elde edilirken diğer parametrelerin durumu da aktarılmıştır. Bu grafik oluşturulurken derinlik ve baskınlık eşiği gibi parametreler sabit tutulmuştur. Tüm setler için baskınlık değeri 0.4 olarak, derinlik değeri 5 olarak belirlenmiştir. Yoğunluk eşiği her veri seti için ayrı ayrı hesaplanmıştır. Bu hesaplamada yaklaşık kara alanları ve toplam nokta sayısı dikkate alınmıştır. Yoğunluk

değeri toplam nokta sayısının nokta içerebilecek alanlara bölünmesi sonucu hesaplanmıştır. Veri setlerindeki nokta sayısı arttıkça yoğunluk eşiği değeri de orantılı olarak arttırılmıştır.

Tablo 5.1: Veri seti içerisindeki nokta sayısına göre algoritma çalışma zamanının değişimi

Veri Seti Adı	Toplam Nokta Sayısı	Yoğunluk Eşiği	Voronoi Hücre Sayısı	Çalışma Zamanı
B1	7160	0.25	2486	0.130616 sn
B2	14323	0.7	5600	0.317150 sn
B3	28350	1.5	7337	0.591450 sn
B4	68090	3.0	6041	1.075125 sn
B5	127672	5.0	5468	2.110353 sn

Derinlik, yoğunluk eşiği, baskınlık eşiği gibi parametreler oluşturulan merkez nokta sayısını dolayısıyla n değerini etkilemektedir. Şekil 5.23 içerisinde hücre sayısının değişimine göre algoritma zamanının değişimi gösterilmektedir.



Şekil 5.24: Merkez nokta sayısına göre algoritma çalışma zamanının değişimi

Şekil 5.24 içerisinde anlaşılabacağı üzere merkez nokta sayısı ile algoritmanın ça-

alışma zamanı arasında logaritmik bir ilişki bulunmaktadır. Bunun en temel sebebi Ball-tree algoritmasının $O(\log(n))$ zaman karmaşıklığına sahip olmasıdır. Dolayısıyla sabit nokta sayısı için merkez noktalarının artması algoritmanın çalışma zamanını logaritmik olarak etkilemektedir.

Tablo 5.2 içerisinde merkez nokta sayısının değişimine göre algoritmanın çalışma zamanının değişimi gösterilmiştir. Ayrıca bu veriler elde edilirken diğer parametrelerin durumu da aktarılmıştır. Bu grafik oluşturulurken yoğunluk ve baskınlık eşiği gibi parametreler sabit tutulmuştur. Tüm setler için baskınlık değeri 0.4 olarak, yoğunluk eşiği değeri 5 olarak belirlenmiştir. Tablo oluşturulurken nokta sayısı sabit tutulmak amacıyla sadece B5 veri seti kullanılmıştır.

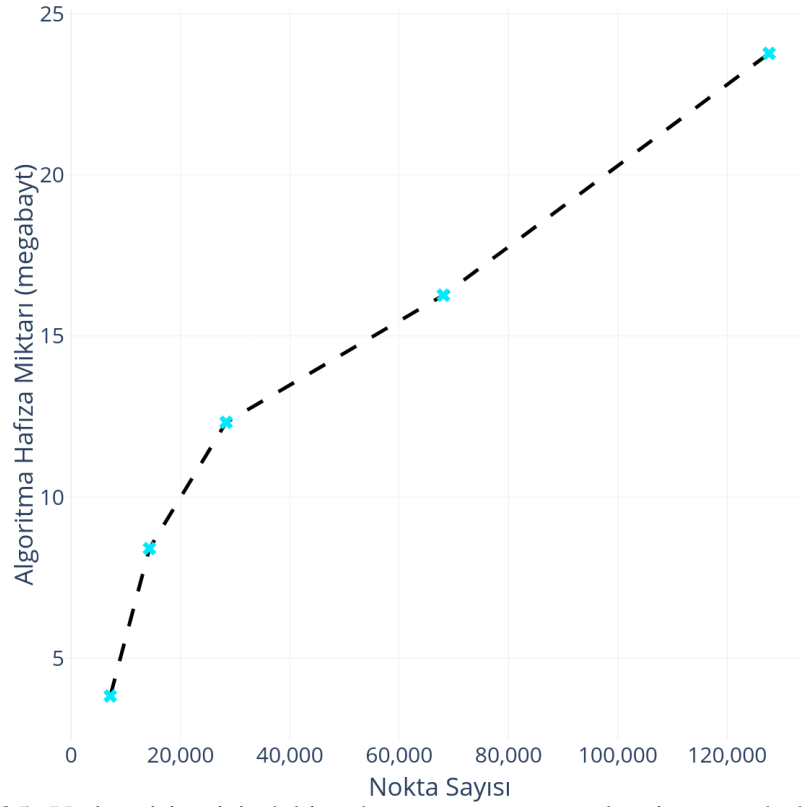
Tablo 5.2: Merkez nokta sayısına göre algoritma çalışma zamanının değişimi

Veri Seti Adı	Derinlik	Voronoi Hücre Sayısı	Çalışma Zamanı
B5	2	626	1.189814 sn
B5	3	1349	1.380306 sn
B5	4	2816	1.628641 sn
B5	5	5468	2.023584 sn
B5	6	8789	2.384620 sn

Algoritmanın hafıza kullanımı her ne kadar günümüzde donanımların gelişmesiyle birlikte çok önem arz etmese de askeri teknoloji ve gömülü sistemlerde yer alan donanımların bazı durumlarda zayıf olması sebebiyle incelenmiştir. Kullanılan hafıza miktarına etki eden temel faktör algoritma zamanına da etki eden veri setindeki toplam nokta sayısı ve merkez noktalarının sayısıdır.

Kullanılan hafıza miktarına ilk okunan veriler ve çizim amacıyla oluşturulan veriler dahil edilmemiş sadece algoritmanın kullandığı objelerin hafıza üzerinde kapladığı yer ifade edilmiştir. Kullanılan hafıza miktarı megabayt cinsinden ifade edilmiştir.

Şekil 5.25 içerisinde nokta sayısının kullanılan hafıza miktarına etkisi gösterilmiştir. Bu gösterim oluşturulurken derinlik ve baskınlık eşiği gibi parametreler sabit tutulmuştur. Tüm setler için baskınlık değeri 0.4 olarak, derinlik değeri 5 olarak belirlenmiştir.



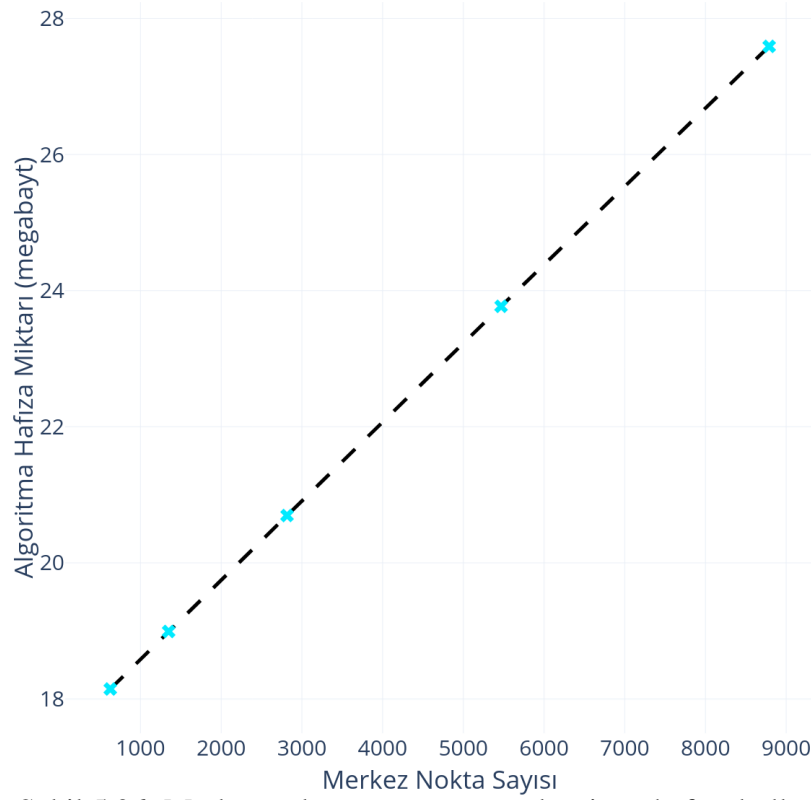
Şekil 5.25: Veri seti içerisindeki nokta sayısına göre algoritmanın kullandığı hafıza miktarı (megabayt)

Tablo 5.3 içerisinde veri seti içerisindeki nokta değişimine göre algoritma hafıza kullanımı değerleri aktarılmıştır.

Tablo 5.3: Veri seti içerisindeki nokta sayısına göre algoritma hafıza kullanımı (megabayt)

Veri Seti Adı	Toplam Nokta Sayısı	Yoğunluk Eşiği	Voronoi Hücre Sayısı	Hafıza miktarı
B1	7160	0.25	2486	3.82762 mb
B2	14323	0.7	5600	8.40346 mb
B3	28350	1.5	7337	12.32035 mb
B4	68090	3.0	6041	16.266073 mb
B5	127672	5.0	5468	23.76934 mb

Derinlik, yoğunluk eşiği, baskınlık eşiği gibi parametreler oluşturulan merkez nokta sayısını dolayısıyla kullanılan hafıza miktarını etkilemektedir. Şekil 5.26 içerisinde hücre sayısının değişimine göre algoritma hafıza miktarının değişimi gösterilmektedir.



Şekil 5.26: Merkez nokta sayısına göre algoritma hafıza kullanımı

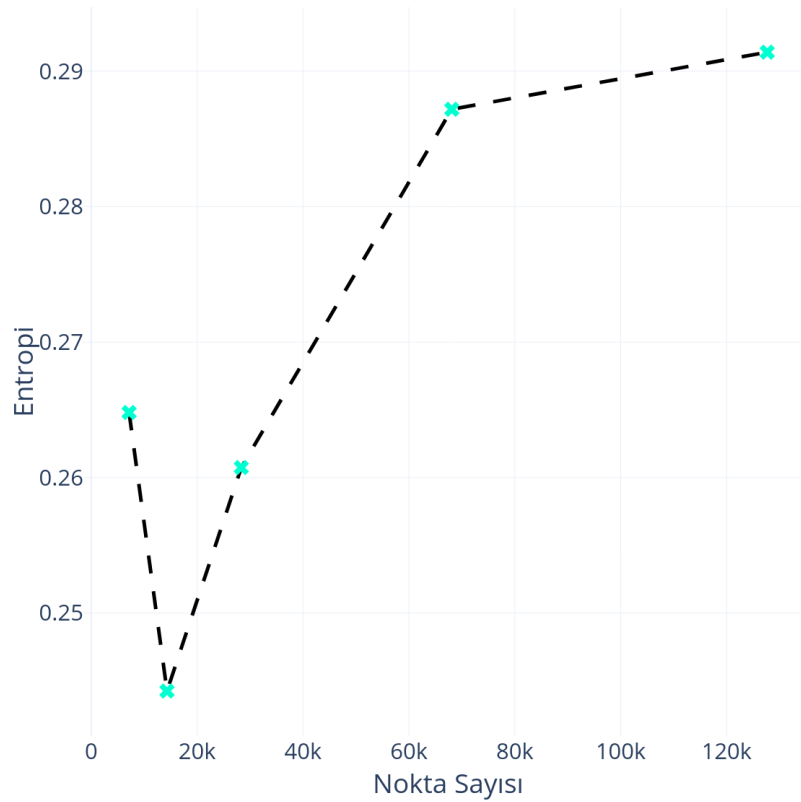
Tablo 5.4 içerisinde merkez nokta sayısının değişimine göre algoritmanın kullandığı hafıza miktarının değişimi gösterilmiştir. Ayrıca bu veriler elde edilirken diğer parametrelerin durumu da aktarılmıştır. Bu grafik oluşturulurken yoğunluk ve baskınlık eşiği gibi parametreler sabit tutulmuştur. Tüm setler için baskınlık değeri 0.4 olarak, yoğunluk eşiği değeri 5 olarak belirlenmiştir. Tablo oluşturulurken nokta sayısı sabit tutulmak amacıyla sadece B5 veri seti kullanılmıştır.

Tablo 5.4: Merkez nokta sayısına göre algoritma hafıza miktarının değişimi

Veri Seti Adı	Derinlik	Voronoi Hücre Sayısı	Hafıza miktarı
B5	2	626	18.147149 mb
B5	3	1349	18.992249 mb
B5	4	2816	20.695605 mb
B5	5	5468	23.769349 mb
B5	6	8789	27.588289 mb

Entropi algoritmamızın sonuçları arasında en önemli değerlerden bir tanesidir. Algoritmamızın çalıştırılması sonucunda oluşan haritanın entropi değeri bizim karma-

şıklığı ne kadar iyi yönettiğimizi ifade eder. Toplam entropi değerine nokta sayısı ve merkez nokta sayısı etki etmektedir. Nokta sayısı benzer merkez noktalar oluşturulduğu durumlarda ise etkisizdir. Şekil 5.27 içerisinde nokta sayısına bağlı olarak entropi değişimi gösterilmiştir. Bu gösterim oluşturulurken derinlik ve baskınlık eşiği gibi parametreler sabit tutulmuştur. Tüm setler için baskınlık değeri 0.4 olarak, derinlik değeri 5 olarak belirlenmiştir.



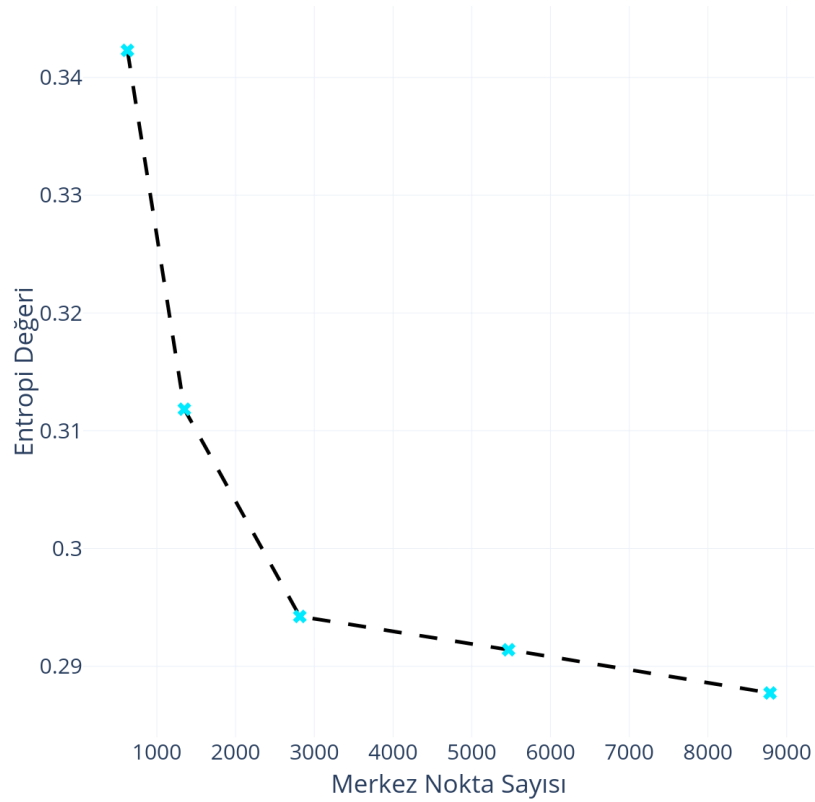
Şekil 5.27: Veri seti içerisindeki nokta sayısına göre entropi

Tablo 5.5 içerisindeki verilerden de anlaşılacağı üzere veri seti içerisindeki toplam nokta sayısının entropi üzerinde herhangi bir etkisi bulunmamaktadır.

Tablo 5.5: Veri seti içerisindeki nokta sayısına göre entropi değerleri

Veri Seti Adı	Toplam Nokta Sayısı	Yoğunluk Eşiği	Voronoi Hücre Sayısı	Entropi Değeri
B1	7160	0.25	2486	0.264807
B2	14323	0.7	5600	0.244246
B3	28350	1.5	7337	0.260741
B4	68090	3.0	6041	0.287186
B5	127672	5.0	5468	0.291400

Derinlik, yoğunluk eşiği, baskınlık eşiği gibi parametreler oluşturulan merkez nokta sayısını dolayısıyla entropi değerini etkilemektedir. Şekil 5.28 içerisinde hücre sayısının değişimine göre entropi değerinin değişimi gösterilmektedir. Şekilden de anlaşılacağı üzere merkez nokta sayısının artışı ile entropi değeri azalmış fakat belirli bir azalıktan sonra bir değere yakınsamıştır.



Şekil 5.28: Merkez nokta sayısına göre entropi değişimi

Tablo 5.6 içerisinde merkez nokta sayısının değişimine göre entropi değerinin değişimi gösterilmiştir. Ayrıca bu veriler elde edilirken diğer parametrelerin durumu da aktarılmıştır. Bu grafik oluşturulurken yoğunluk ve baskınlık eşiği gibi parametreler sabit tutulmuştur. Tüm setler için baskınlık değeri 0.4 olarak, yoğunluk eşiği değeri 5 olarak belirlenmiştir. Tablo oluşturulurken nokta sayısı sabit tutulmak amacıyla sadece B5 veri seti kullanılmıştır.

Tablo 5.6: Merkez nokta sayısına göre entropi deęerleri

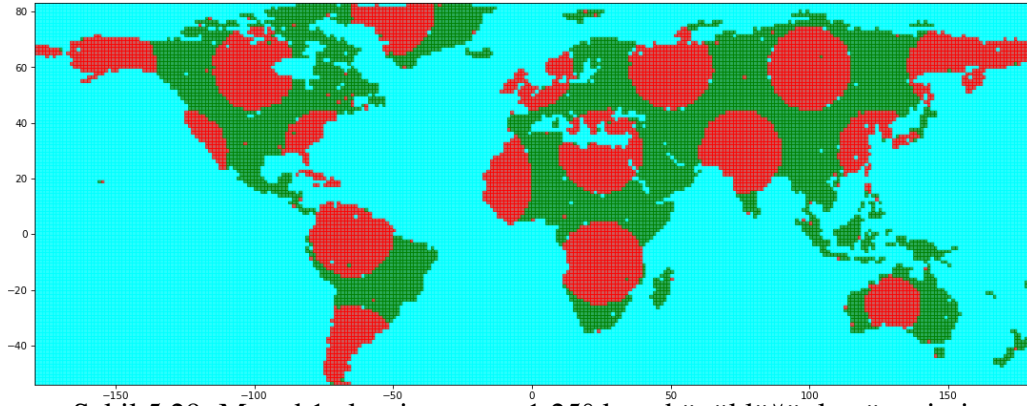
Veri Seti Adı	Derinlik	Voronoi Hcre Sayısı	Entropi Deęeri
B5	2	626	0.342300
B5	3	1349	0.311831
B5	4	2816	0.294231
B5	5	5468	0.291400
B5	6	8789	0.287730

5.1.3 Kıyaslamalar

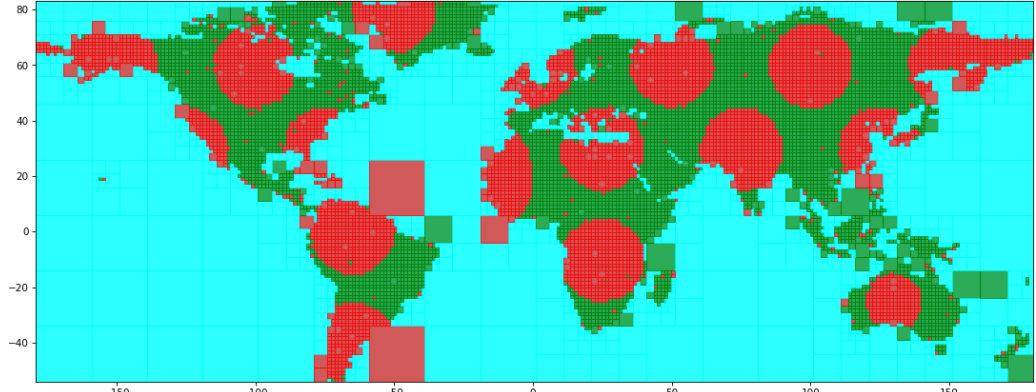
alıřmamıza en yakın alıřma akmak tarafından gerekleřtirilmiřtir. Bu sebeple her iki alıřmanın kıyaslamaları yapılmıřtır. akmak tarafından uygulanan Metod 1, Metod 2 ve Metod 3 bizim tarafımızdan onerilen metodla hem grsel hem de sayısal birtakım kıyaslamalara tabi tutulmuřtur. Tm kıyaslamalar iin B5 veri seti kullanılmıřtır. Ayrıca tm metodlar aynı donanım ve yazılım sistemleri kullanılarak kıyaslanmıřtır. akmak'ın algoritması iin gereken entropi deęeri ise 0.1 olarak belirlenmiřtir.

5.1.3.1 Grsel Kıyaslamalar

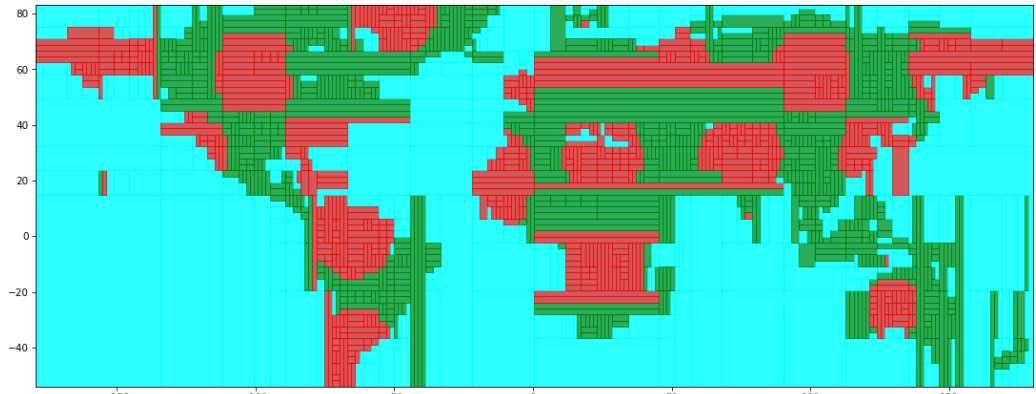
akmak tarafından 3 adet metod alıřılmıřtır. Buna gre Metod 1 ierisinde sabit karelere gre bltleme yapılmıřtır. Metod 2 ierisinde ise kareler daha alt karelere bltlenmiřtir. Metod 3 ierisinde kareler yerine dikdrtgenler bltleme iin kullanılmıřtır. Benzerlik oluřturması aısından grsel kıyaslamalar ierisinde Metod 1, Metod 2 ve Metod 3 iin 1.25° lık minimum boyut kullanılmıřtır. onerilen metod iinse derinlik deęeri 5 olarak belirlenmiřtir. Baskınlık eřięi deęeri 0.4 yoęunluk eřięi deęeri iin ise 5 deęeri seilmiřtir.



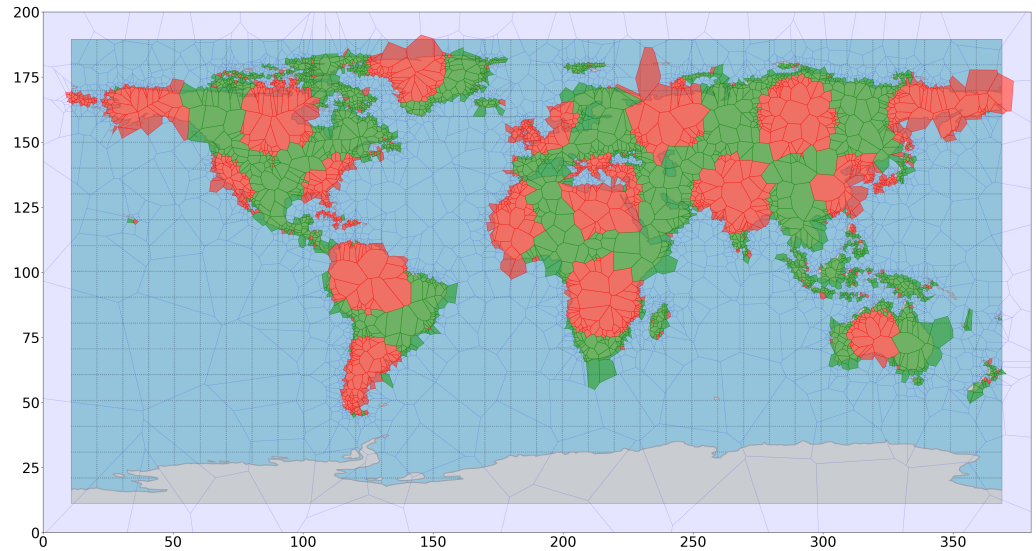
řekil 5.29: Metod 1 algoritmasının 1.25° kare byklęyle gsterimi



Şekil 5.30: Metod 2 algoritmasının 1.25° kare büyüklüğüyle gösterimi



Şekil 5.31: Metod 3 algoritmasının 1.25° kare büyüklüğüyle gösterimi



Şekil 5.32: Önerilen metodun gösterimi Derinlik = 5, hücre sayısı = 5468

Şekil 5.29, 5.30, 5.31 ve 5.32 içerisinde görüldüğü üzere önerdiğimiz metod diğer metotlara göre görsel olarak daha iyi sonuçlar elde etmiştir. Çakmak'ın çıktıları in-

celendiğinde ilginç kare ve dikdörtgenlere rastlanmaktadır. Bizim metodumuzun çıktılarında ise kıyılarıdaki taşmalar hariç bu tarz problemler yaşanmamaktadır.

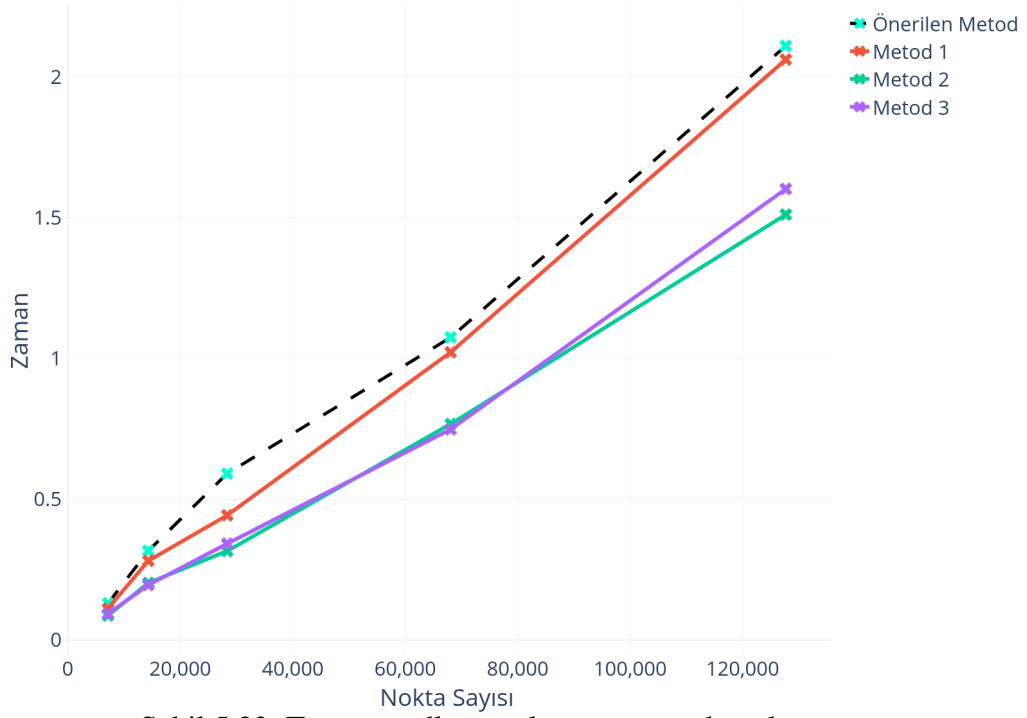
5.1.3.2 Çalışma Zamanına Göre Sayısal Kıyaslamalar

Çalışma zamanı kıyaslaması için aynı veri setleri kullanılmıştır. Metod 1, Metod 2, Metod 3 ve önerilen metodumuz için B1, B2, B3, B4 ve B5 veri setleri çalıştırılmıştır. Yapılan bu çalışmalar aynı donanım ve yazılım altyapısı kullanılarak eşit şartlarda oluşturulmuştur. Metod 1, Metod 2 ve Metod 3 için en küçük hücre boyutu 1.25° olarak belirlenmiştir. Önerdiğimiz metod için ise derinlik değeri 5 olarak belirlenmiştir.

Tablo 5.7: Metod 1, Metod 2, Metod 3 ve önerilen metodun çalışma zamanları (sn)

Veri Seti Adı	Nokta Sayısı	Metod 1	Metod 2	Metod 3	Önerilen metod
B1	7160	0.111934	0.087239	0.093367	0.130616
B2	14323	0.281653	0.202933	0.195630	0.317150
B3	28350	0.443289	0.316601	0.342300	0.591450
B4	68090	1.022307	0.767353	0.748391	1.075125
B5	127672	2.061759	1.511213	1.602207	2.110353

Tablo 5.7 içerisinde tüm metodların çalışma zamanı değerleri aktarılmıştır.



Şekil 5.33: Tüm metodların çalışma zamanı karşılaştırması

Şekil 5.33 içerisinde görüldüğü üzere önerdiğimiz metod Çakmak tarafından önerilen metod 1 ile yakın bir çalışma zamanı performansı göstermektedir.

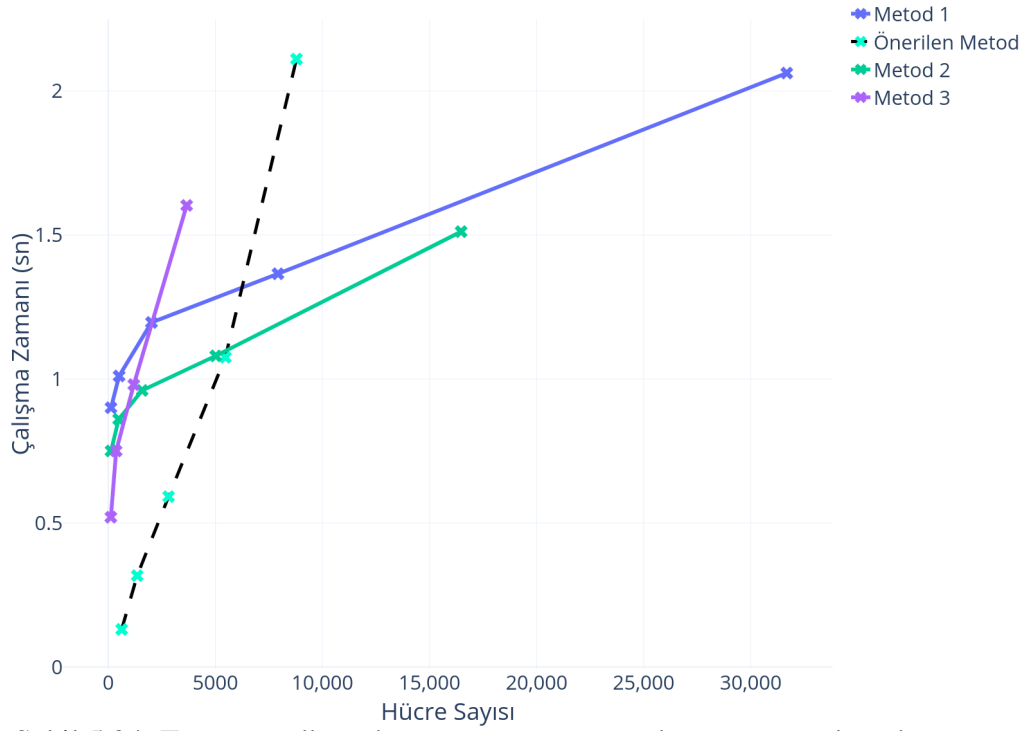
Hücre sayısına göre çalışma zamanı bu tez içerisinde önemli bir kıyas noktası oluşturmaktadır. Daha anlamlı karşılaştırma yapabilmek adına Çakmak'ın tezinde yer alan seviye 1, 2, 3, 4 ve 5 yani 20° , 10° , 5° , 2.5° ve 1.25° kare büyüklükleri bizim tezimizdeki derinlik değeriyle kıyaslanmıştır. Bizim tezimiz içerisinde yer alan 2, 3, 4, 5 ve 6 değerleri kullanılmıştır. Kıyaslamalarda B5 veri seti kullanılmıştır. Tablo 5.8 içerisinde bu seviyelere göre oluşturulan hücre sayısı aktarılmıştır.

Tablo 5.8: Tüm metodların hücre sayısı

Seviye	Önerilen metod	Metod 1	Metod 2	Metod 3
1	626	128	128	37
2	1349	506	488	121
3	2816	2018	1580	367
4	5468	7922	5016	1191
5	8789	31682	16476	3661

Şekil 5.34 içerisinde tüm metodların hücre sayısına göre çalışma zamanı grafiği ak-

tarılmıştır. Bu karşılaştırmaya göre önerdiğimiz metodumuz hücre sayısı baz alındığında Metod 1 ve Metod 2'ye göre daha yavaş bir bölütleme gerçekleştirmektedir. Fakat önerilen metod Metod 3'e göre daha hızlı bölütleme yapmıştır. Şekildeki grafiklerin birbirini kesme sebebi çok farklı sayılarda bölütleme yapılmasıyla ifade edilebilir.



Şekil 5.34: Tüm metodların hücre sayısına göre çalışma zamanı karşılaştırması

5.1.3.3 Entropiye Göre Sayısal Kıyaslamalar

Tüm metodlar incelendiğinde amacın entropiyi mümkün olduğunca düşürmek olduğu gözlemlenebilir. Bu amaçla Çakmak'ın tezindeki 1.25° kare büyüklüğüne göre metodların entropisi incelenmiştir. Bizim metodumuzda ise derinlik 5 olarak belirlenmiştir.

Tablo 5.9 içerisinde seviyelere göre entropi kıyaslamaları yapılmıştır. Buna göre önerdiğimiz metod diğer metodlara göre nokta sayısı arttıkça daha iyi entropi değerlerine sahiptir. Tüm veri setlerinde Metod 3'e göre daha iyi bir performans göstermiştir. Metod 1 ve Metod 2'ye göre ise nokta sayıları arttıkça daha iyi performanslar göstermektedir.

Tablo 5.9: Tüm metodların veri setlerine göre entropi değerleri

Veri Seti Adı	Nokta Sayısı	Metod 1	Metod 2	Metod 3	Önerilen metod
B1	7160	0.1461	0.1484	0.5271	0.264807
B2	14323	0.2439	0.2530	0.5782	0.244246
B3	28350	0.3644	0.3697	0.6272	0.260741
B4	68090	0.4907	0.4945	0.6807	0.287186
B5	127672	0.5475	0.5485	0.6788	0.291400

Entropi değerinin büyüklüğü doğrudan haritanın kaç farklı alana bölündüğü ile ilgilidir. Şekil 5.35 içerisinde hücre sayılarına göre entropi değerleri tüm metodlar için kıyaslanmıştır. Bu kıyaslamada B5 veri seti kullanılmıştır. Daha anlamlı karşılaştırma yapabilmek adına Çakmak'ın tezinde yer alan seviye 1, 2, 3, 4 ve 5 yani 20°, 10°, 5°, 2.5° ve 1.25° kare büyüklükleri bizim tezimizdeki derinlik değeriyle kıyaslanmıştır. Bizim tezimiz içerisinde yer alan 2, 3, 4, 5 ve 6 değerleri kullanılmıştır. Bu şekil dikkate alındığında metodumuz çok daha az hücelere bölmesine karşın karmaşıklığı iyi yönetmiştir. Entropi değerleri bizim metodumuz için, tüm diğer metodlara göre daha başarılı olmuştur. Çok az hücre bölünmesiyle çok daha iyi entropi değerleri elde edilmiştir.



Şekil 5.35: Tüm metodların hücre sayısına göre entropi karşılaştırması

5.2 Test ve Çıktı Yorumları

Tüm bu testler ve çıktıları göz önünde bulundurduğunda önerdiğimiz metod çalışma zamanı açısından her ne kadar yüksek performans sunmasa da görsellik ve entropi değerleri açısından önemli başarılar elde etmiştir. Çalışma zamanı optimize edilmiş fonksiyonlar ve daha iyi donanımlarla birlikte düşürülebilir. Entropi değerlerinin kullandığımız bu yöntemle düşürülmesi karmaşıklık yönetimi açısından önemlidir.

BÖLÜM 6

SONUÇ

Biz bu çalışmada konumsal verinin kullanıcıya hızlı, temiz ve anlaşılır bir şekilde gösterilmesi konusunda çalıştık. Bu konuda daha önce yapılmış çalışmalarını inceledik. Çalışmamız içerisinde hem doğal görünümü hem de bilgisayar grafiğinde kullanılması sebebiyle Voronoi diyagramlarıyla gösterimler yapmayı tercih ettik. Çalışma içerisinde ihtiyaç duyulan en kısa mesafedeki noktayı bulma amacıyla Ball-tree algoritmasını kullandık. Bir kısmı bizim tarafımızdan oluşturulan bir kısmı da daha önce kullanılan veri setleri üzerinde çalışmalar gerçekleştirdik.

Çalışmamızda ilk kare boyutu, derinlik, yoğunluk eşiği, baskınlık eşiği gibi parametreleri kullandık. Bu parametrelere göre bölütlemeler gerçekleştirdik.

Kullanılan veri setleri ve parametrelere göre önerdiğimiz metodu farklı testlere tabi tuttuk. Bu testlerin sonucunda parametrelere göre farklı entropi değerlerinde farklı haritalar elde ettik. Aynı araçlar kullanılarak oluşturulan ve farklı noktalar içeren veri setleri değiştiğinde ise benzer haritalar ve entropi değerlerinin oluştuğunu gösterdik. Bu durum önerilen metodun tutarlılığı açısından önemlidir.

Daha önce aynı konu üzerinde benzer çalışmalarını farklı metodolojilerle deneyen araştırmacıların algoritmaları ile kendi oluşturduğumuz algoritmayı aynı ortamda aynı veri setleri üzerinde benzer parametrelerle testlere tabi tuttuk.

Yaptığımız testler sonucunda her ne kadar çalışma zamanı diğer metodlara göre daha uzun olsa da daha iyi görseller elde ettik. Ayrıca karmaşıklık hesabında önemli yeri olan entropi değerini kullandık. Yaptığımız entropi hesaplamaları sonucunda karmaşıklığın önerilen metod tarafından iyi yönetildiği gözlemlenmektedir.

Yaptığımız çalışmada yer alan rastgele nokta seçimi bazı görsel problemlere yol açmaktadır. Bu problemlerin çözümü amacıyla rastgele nokta seçimi sınırlandırılabilir ve bu sayede daha iyi görseller elde edilebilir.

Ayrıca bu çalışmanın iki tip (renk) yerine daha çok renk kullanılarak yapılması da farklı bir çalışma olarak ele alınabilir. Voronoi diyagramları kareler üzerinden değil de doğrudan oluşturulabilir. Yine bu yöntem farklı bir araştırmanın konusu olabilir.

KAYNAKLAR

- [1] P. A. Burrough, R. McDonnell, R. A. McDonnell, and C. D. Lloyd, *Principles of geographical information systems*. Oxford university press, 2015.
- [2] F. Aurenhammer, “Voronoi diagrams—a survey of a fundamental geometric data structure”, *ACM Computing Surveys (CSUR)*, vol. 23, no. 3, pp. 345–405, 1991.
- [3] J. M. Rumbold and B. K. Pierscioneck, “What are data? A categorization of the data sensitivity spectrum”, *Big data research*, vol. 12, pp. 49–59, 2018.
- [4] V. Estivill-Castro, “Why so many clustering algorithms: a position paper”, *ACM SIGKDD explorations newsletter*, vol. 4, no. 1, pp. 65–75, 2002.
- [5] S. L. Horowitz, “Picture segmentation by a directed split-and-merge procedure”, in *IJCPR*, 1974, pp. 424–433.
- [6] I. D. Cooper, “What is a mapping study?”, *Journal of the Medical Library Association: JMLA*, vol. 104, no. 1, p. 76, 2016.
- [7] G. Voronoi, “Nouvelles applications des paramètres continus à la théorie des formes quadratiques. Deuxième mémoire. Recherches sur les paralléloèdres primitifs.”, *Journal für die reine und angewandte Mathematik (Crelles Journal)*, vol. 1908, no. 134, pp. 198–287, 1908.
- [8] Markó, “Real-time Cave Destruction Using 3 D Voronoi”, 2018.
- [9] H Minkowski, “Theorie der Konvexen Körper, Gesammelte Abhandlungen II”, *BG Teubner, Leipzig, Berlin*, 1911.
- [10] S. Fortune, “A sweepline algorithm for Voronoi diagrams”, *Algorithmica*, vol. 2, no. 1-4, p. 153, 1987.
- [11] N. S. Altman, “An introduction to kernel and nearest-neighbor nonparametric regression”, *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.
- [12] K. Chapron, “Méthode hybride pour la reconnaissance d’activités de la vie quotidienne et d’exercices physiques en temps réel”, Ph.D. dissertation, Université du Québec à Chicoutimi, 2017.

- [13] S. M. Pirayonesi and T. E. El-Diraby, “Role of Data Analytics in Infrastructure Asset Management: Overcoming Data Size and Quality Problems”, *Journal of Transportation Engineering, Part B: Pavements*, vol. 146, no. 2, p. 04 020 022, 2020.
- [14] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.
- [15] D. Coomans and D. L. Massart, “Alternative k-nearest neighbour rules in supervised pattern recognition: Part 1. k-Nearest neighbour classification by using alternative voting rules”, *Analytica Chimica Acta*, vol. 136, pp. 15–27, 1982.
- [16] S. M. Omohundro, *Five balltree construction algorithms*. International Computer Science Institute Berkeley, 1989.
- [17] M. Dolatshah, A. Hadian, and B. Minaei-Bidgoli, “Ball*-tree: Efficient spatial indexing for constrained nearest-neighbor search in metric spaces”, *arXiv preprint arXiv:1511.00628*, 2015.
- [18] T. Liu, A. W. Moore, and A. Gray, “New algorithms for efficient high-dimensional nonparametric classification”, *Journal of Machine Learning Research*, vol. 7, no. Jun, pp. 1135–1158, 2006.
- [19] N. Kumar, L. Zhang, and S. Nayar, “What is a good nearest neighbors algorithm for finding similar patches in images?”, in *European Conference on Computer Vision*, Springer, 2008, pp. 364–378.
- [20] A. M. Kibriya and E. Frank, “An empirical comparison of exact nearest neighbour algorithms”, in *European Conference on Principles of Data Mining and Knowledge Discovery*, Springer, 2007, pp. 140–151.
- [21] J. L. Bentley, “Multidimensional binary search trees used for associative searching”, *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.
- [22] M. Aly, P. Welinder, M. Munich, and P. Perona, “Scaling object recognition: Benchmark of current state of the art techniques”, in *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, IEEE, 2009, pp. 2117–2124.
- [23] M. De Berg, O. Cheong, M. Van Kreveld, and M. Overmars, “Orthogonal range searching: Querying a database”, *Computational Geometry: Algorithms and Applications*, pp. 95–120, 2008.

- [24] J. MacQueen, “Some methods for classification and analysis of multivariate observations”, in *Proceedings of The Fifth Berkeley Symposium on Mathematical Statistics and Probability*, Oakland, CA, USA, vol. 1, 1967, pp. 281–297.
- [25] S. Lloyd, “Least squares quantization in PCM”, *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.
- [26] E. Mirkes, “K-means and K-medoids applet”, *University of Leicester*, 2011.
- [27] J. M. Pena, J. A. Lozano, and P. Larranaga, “An empirical comparison of four initialization methods for the k-means algorithm”, *Pattern Recognition Letters*, vol. 20, no. 10, pp. 1027–1040, 1999.
- [28] M. Worboys, *GIS: A Computer Science Perspective*, 1994.
- [29] K.-T. Chang, *Programming ArcObjects with VBA: a task-oriented approach*. CRC Press, 2007.
- [30] M. Segal, K. Akeley, and C. Frasier, *The OpenGL graphics system: A specification*, 1992.
- [31] B. Cakmak, *Complexity Management in Visualization of Very Large Spatial Data*, 2017.
- [32] D. Guo, “Flow mapping and multivariate visualization of large spatial interaction data”, *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 6, pp. 1041–1048, 2009.
- [33] K.-H. Anders, “A hierarchical graph-clustering approach to find groups of objects”, in *Proceedings 5th Workshop on Progress in Automated Map Generalization*, Citeseer, 2003, pp. 1–8.
- [34] J. Komzak and M. Eisenstadt, “Visualisation of entity distribution in very large scale spatial and geographic information systems”, in *KMI-TR-116*, KMi, The Open University, 2001.
- [35] X. Zhu and D. Guo, “Mapping large spatial flow data with hierarchical clustering”, *Transactions in GIS*, vol. 18, no. 3, pp. 421–435, 2014.
- [36] C. M. Gold, “Problems with handling spatial data—the Voronoi approach”, *CISM journal*, vol. 45, no. 1, pp. 65–80, 1991.
- [37] P. Secchi, S. Vantini, and V. Vitelli, “Bagging Voronoi classifiers for clustering spatial functional data”, *International Journal of Applied Earth Observation and Geoinformation*, vol. 22, pp. 53–64, 2013.

- [38] J. Chen, R. Zhao, and Z. Li, “Voronoi-based k-order neighbour relations for spatial analysis”, *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 59, no. 1-2, pp. 60–72, 2004.
- [39] S. Smorodinsky, “Geometric permutations and common transversals”, Ph.D. dissertation, Citeseer, 1998.

EK A

Algorithm 1 METHOD 1

```
* a structure for data points
PointInfo(color, posX, posY, index)
    init(color, posX, posY, index)
* a class that generates grid tree and checks level of the tree
GridTree(self, pointSet, cellSize, parent)
    init(pointSet, cellSize, level, parent, children, entropy, color, xEdges, yEdges)
    distributePointsToCells(self):
        * Calculate the row and column info of each point P in pointSet using
        tree level information
        for P in pointSet
            column = (P.posX - xEdges.min) \ self.cellSize
            row = (P.posY - yEdges.min) \ self.cellSize
            update entropy, color of grid
    decideColor(self):
        1 - Find the dominant point in color
        2 - update self.color
    calculateEntropy(self):
        1 - Calculate entropy using given information
        2 - update self.entropy
    splitCell(self):
        1 - Split parent cell and generate new child nodes
        2 - update self.children
    splitUntilLimitSize(maximumLevel):
        self.calculateEntropy()
        self.splitCell()
        if self.level >= maximumLevel return
        for child in self.children
            if child == None return
            child.splitUntilLimitSize(minCellSize)
```

Çakmak tarafından uygulanan Metod 1 adlı algoritma

EK B

Algorithm 2 METHOD II

```
* a class that generates grid and checks maximum allowed tree level
AdaptiveSquareTree(self, pointSet, cellSize)
    init(pointSet, size, parent, level, entropy, xMin, yMin
        children, color)
    distributePointsToChildren(self):
        * distribute points to rows and columns
        for P in pointSet
            column = (P.posX - xMin) \ (self.size)
            row = (P.posY - yMin) \ (self.size)
            update children.pointSet
        * update empty children
        * Total children = [topLeft, topRight, bottomRight, bottomLeft ]
        for i in range 4
            children[i] = AdaptiveSquareTree(children[i].pointSet, size*0.5)
            update children[i].entropy, children[i].color, children[i].level
    decideColor(self):
        count dominant color(feature) in number then
        update self.features, self.color
    calculateEntropy(self):
        call self.decideColor() & Calculate entropy using given information
        update self.entropy
    splitCell(self):
        if children == None
            self.distributePointsToChildren()
    splitUntilLimits(maxLevel, entropyThreshold):
        self.calculateEntropy()
        self.splitCell()
        if self.entropy <= entropyThreshold return
        if self.level >= maxLevel return
        for child in self.children
            if child == None return
            child.splitUntilLimits(maxLevel, entropyThreshold)
```

Çakmak tarafından uygulanan Metod 2 adlı algoritma

EK C

Algorithm 3 METHOD III

```
* a class that generates grid checks minimum cell size
AdaptiveRectangleTree(self, pointSet, numberOfSlices)
    init(pointSet, sizeX, sizeY, parent, level, entropy, xMin, yMin
        verticalSlices, horizontalSlices, decidedSlices, color)
    distributePointsToPossibleSlices(self):
        * distribute points to possible horizontal and vertical rows and columns
        for P in pointSet
            columnV = (P.posX - xMin) \ (self.sizeX)
            rowV = (P.posY - yMin) \ ((self.sizeX)*(numberOfSlices))
            columnH = (P.posX - xMin) \ ((self.sizeY)*(numberOfSlices))
            rowH = (P.posY - yMin) \ (self.sizeY)
            update verticalSlices.pointSet & verticalSlices.pointSet
        * generate horizontal and vertical slices (children)
        for i in range numberOfSlices
            update verticalSlices[i] =
                AdaptiveRectangleTree(verticalSlices[i].pointSet, numberOfSlices)
            update horizontalSlices[i] =
                AdaptiveRectangleTree(horizontalSlices[i].pointSet, numberOfSlices)
            update children.entropy, children.color , children.level
        * pick best slice (child) option and point it as decided slice
        update self.decidedSlices
    decideColor(self):
        count dominant color(feature) in number then
        update self.features, self.color
    calculateEntropy(self):
        call self.dediceColor() & Calculate entropy using given information
        update self.entropy
    decideSlices(self):
        if decidedSlices == None
            self.distributePointsToPossibleSlices()
    splitUntilLimits(maxLevelSize, entropyThreshold):
        self.calculateEntropy()
        self.decideSlices()
        if self.entropy <= entropyThreshold return
        if self.sizeX or self.sizeY <= maxLevelSize return
        for child in self.children
            if child == None return
            child.splitUntilLimits(maxLevelSize, entropyThreshold)
```

Çakmak tarafından uygulanan Metod 3 adlı algoritma