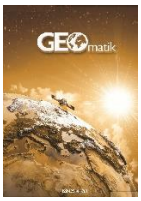
Geomatik

https://dergipark.org.tr/tr/pub/geomatik

e-ISSN 2564-6761



Python kullanarak GPS iz verilerinin kümelenmesi ve optimizasyonu: İstanbul Göztepe Kavşağı Mevkisi örneği

Emrullah Demiral*[1], İsmail Rakıp Karaş[2]

[1]*Karabük Üniversitesi, Safranbolu ŞYD MYO, Bilgisayar Teknolojileri Bölüm, Karabük, Türkiye, emrullahdemiral@gmail.com*
[2]*Karabük Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölüm, Karabük, Türkiye, ismail.karas@karabuk.edu.tr*

Kaynak Göster: Demiral, E., & Karaş, İ. R. (2025). Python kullanarak GPS iz verilerinin kümelenmesi ve optimizasyonu: İstanbul Göztepe Kavşağı Mevkisi örneği. Geomatik, 10 (1), 15-28.

DOI: 10.29128/geomatik.1502643

Anahtar Kelimeler
GPS İzleri
Harita Üretimi
Python
Kümeleme ve Optimizasyon
Büyük Veri

Araştırma Makalesi
Geliş: 19.06.2024
Revize: 05.08.2024
Kabul: 08.08.2024
Çevrim İçi Yayınlanma: 01.11.2024

Öz

Bu çalışma, noktalar ve rotalardan oluşan GPS iz verilerini içeren veri setlerinin kümeleme ve optimize edilmesine yönelik bir metodoloji sunmaktadır. Mesafe ve doğrultu eşik değerleri kullanılarak, coğrafi olarak yakın ve yön olarak hizalanmış noktalar gruplandırılmaktadır. Bu işlem, Python'da NumPy ve Geopy kütüphaneleri kullanılarak yapılmıştır. Bu yaklaşım, çeşitli coğrafi bilgi sistemleri (CBS) ve GPS izleri kullanılarak yol orta hattının çıkarılması problemleri için veri seti boyutunun azaltılması ve optimize edilmesi açısından faydalıdır. Farklı eşik değerleri için veri seti optimize edilmiş ve deneysel sonuçlar, veri seti boyutunu önemli ölçüde azaltırken, kritik mekânsal ilişkileri koruma açısından yöntemin etkinliğini göstermektedir. İstanbul, Göztepe Kavşağı Mevkiinde taksilerden kaydedilmiş GPS izleri üzerinde yapılan çalışma, metodolojinin uygulanabilirliğini ve veri seti boyutunu azaltma potansiyelini ortaya koymaktadır. Geliştirilen kodlar, GitHub'ta açık erişime sunularak paylaşılmıştır.

Clustering and optimization of GPS trajectories data using Python: a case study of Göztepe Junction in İstanbul

Keywords
GPS Trajectories
Map Construction
Python
Clustering and Optimizing
Big Data

Research Article
Received: 19.06.2024
Revised: 05.08.2024
Accepted: 08.08.2024
Online Published: 01.11.2024

Abstract

This study presents a methodology for clustering and optimizing datasets containing GPS track data consisting of points and routes. Using distance and direction angle threshold values, geographically close and directionally aligned points are grouped. This process is carried out in Python using the NumPy and Geopy libraries. This approach is beneficial for reducing and optimizing dataset sizes for various Geographic Information Systems (GIS) and map generation problems using GPS tracks. The dataset was optimized for different threshold values, and experimental results demonstrate the method's effectiveness in significantly reducing the dataset size while preserving critical spatial relationships. The study conducted on GPS tracks recorded from taxis at the Göztepe Junction in Istanbul reveals the methodology's applicability and potential for dataset size reduction. The developed code has been made publicly accessible and shared on GitHub.

1. Giriş

GPS iz verileri, günümüzde çeşitli uygulamalarda yaygın olarak kullanılmaktadır. Bu veriler, yol orta hattının çıkarılması, taşıt takip sistemleri, coğrafi bilgi sistemleri (CBS), yolculuk planlaması ve trafik yönetimi gibi alanlarda önemli bilgiler sağlamaktadır (Fasy ve ark., 2013; Guo ve ark., 2007). GPS iz verilerinin doğru bir şekilde analiz edilmesi ve işlenmesi, bu uygulamaların etkinliği ve doğruluğu için kritik öneme sahiptir(Dal Poz ve ark., 2022; Krumm ve ark., 2008).

Bu çalışmada, İstanbul'un yoğun trafiğe sahip bölgelerinden biri olan Göztepe Kavşağı Mevkii'nde çalışan taksilerden elde edilen GPS iz verileri kullanılmıştır. Çalışmanın temel amacı, bu GPS iz verilerini kullanarak coğrafi olarak yakın ve yön olarak hizalanmış noktaları gruplayıp her bir grup için temsil noktaları üreterek veri setini optimize etmektir. Bu hedef doğrultusunda uygun mesafe ve doğrultu eşik değerlerinin belirlenebilmesi için bir yazılım geliştirmiştir. Bu yazılım ile farklı mesafe ve eşik değerleri için veri seti optimize edilebilir ve sonuçlar karşılaştırılarak en uygun mesafe eşik değerlerine karar verilebilir. Bu optimizasyon işlemi, CBS uygulamalarında yol orta hattının çıkarılması için kullanılan veri seti boyutunu azaltarak işlem maliyetlerini düşürebilir ve sistem performansını artırabilir. Gerek duyulması halinde, veri setine yapılacak manuel müdahaleleri kolaylaştırabilir.

Geliştirilen metodoloji, Python programlama dili ile NumPy, Geopy ve Matplotlib gibi kütüphaneler kullanılarak uygulanmıştır. Mesafe ve doğrultu eşik değerleri uygulanarak, coğrafi olarak yakın ve yön olarak hizalanmış noktalar gruplanmaktadır. Farklı mesafe ve doğrultu eşik değerleri ile yapılan deneyler, veri setinin boyutunu önemli ölçüde azaltmıştır. Aynı güzergâh üzerinde kaydedilmiş farklı veri kayıtlarına ait GPS izlerinden coğrafi olarak birbirine yakın ve aynı yönde hareket özelliği gösteren izler gruplanır ve birleştirilir. Böylece her bir grubu temsilen yeni bir temsil noktası elde edilir. Temsil noktaları sadece GPS izlerinin birbirine yakınlıklarına göre sınıflandırılarak oluşturulmamıştır. DBSCAN, OPTICS, kdTree gibi makine öğrenmesi yöntemleri kullanılarak birbirine yakın noktaların sınıflandırılması ile ilgili literatürde birçok çalışma mevcuttur (Badran ve ark., 2023; Ezzat ve ark., 2018; Stanojevic ve ark., 2018). Ancak sadece mesafeye bakılarak noktaların sınıflandırılması ve her bir sınıf için yeni temsil noktalarının üretilmesi yaklaşımlarını temel alarak üretilen yol ağlarında topolojik problemler gözlemlenmektedir. Genellikle, benzer çalışmalarda kullanılan veri setlerinde yükseklik (elevation) bilgisi yer almamaktadır. Alt geçit ve üst geçit gibi gerçekte kesişmeyen yolların ayrımı için yükseklik bilgisi önemli bir parametredir. Yükseklik bilgisinin veri setlerinde yer almaması bu ayrımı zorlaştırmaktadır. Alt geçit ve üst geçit gibi bölgelerin üst üste geldiği alanlarda kaydedilmiş GPS izleri coğrafi yakınlık gösterir. Yükseklik parametresinin olmaması ve sadece coğrafi yakınlık parametresi kullanılarak işlemlerin yapılması durumunda bu gibi alanlarda kaydedilmiş GPS izleri hangi güzergahta kaydedildiği gözetilmeksizin aynı grup içerisinde bulunabilmekte ve birleştirilerek temsil noktası oluşturulmaktadır. Oysaki ilgili bölge için alt geçit güzergahında kaydedilmiş ve birbirine coğrafi olarak yakın olan noktalar bir grup, üst geçit güzergahında kaydedilmiş ve birbirine coğrafi olarak yakın olan noktalar ise başka bir grup olarak sınıflandırılmalıdır. Böylece topolojik olarak kesişmeyen alt ve üst geçit noktalarını temsil için iki ayrı temsil noktası elde edilebilir ve yol ağındaki topolojik yapı korunabilir. Bu bağlamda, bu çalışmada önerilen yöntem GPS izlerinin kaydı esnasındaki sırasını esas alarak elde edilen ilişkiselliği kullanır ve hareket doğrultusunu hesaplar. Coğrafi yakınlığın yanında doğrultu parametresi de işin içine katılarak alt ve üst geçit gibi alanlarda kaydedilen noktalar için istenmeyen kesişme durumlarının engellenmesi hedeflenmiştir. Çalışma, bu yönü ile ele alındığında, uygun mesafe ve doğrultu eşik değerlerinin belirlenmesi ile üretilen temsil noktalarının yol ağı üzerinde temsil ettiği noktalar için topolojik yapıyı ortaya koyacak kritik mekânsal ilişkileri koruma yeteneğini göstermiştir.

Bu çalışmada, farklı açı ve mesafe eşik değerleri için çıktılar üretilerek veri setinin en uygun optimizasyon seviyesine ulaşması için gerekli eşik değerlerinin belirlenmesinde yardımcı olacak bir yazılım geliştirilmiştir. Mevcut literatürde, GPS iz verilerinin kümeleme ve optimizasyonu genellikle GPS izlerine dayalı yol orta hattının çıkarılması bağlamında ele alınmaktadır (Chen ve ark., 2020; Gao ve ark., 2021; Leichter ve Werner, 2019; Li ve ark., 2016; Ni ve ark., 2018; Qiu ve Wang, 2016; Tang ve ark., 2017; Yang ve ark., 2019). Bu bağlamda bu çalışma, GPS izleri kullanılarak yol orta hattının çıkarılması konusu için bir ön aşama oluşturmaktadır.

Günümüzde konum, hız ve zaman belirleme amaçlı kullanılan başlıca Uydu Konum Belirleme sistemleri; GPS (ABD), GLONASS (Rusya), BEIDOU/COMPASS (Çin), QZSS (Japonya), IRNSS/GAGAN (Hindistan), GALLILEO (AB) olarak sınıflandırılabilir (Koca ve Ceylan, 2018). GNSS (Global Navigation Satellite Systems) iz kayıtlarının doğruluğu cihaz hassasiyeti, arazi şartları ve bina yoğunluğu gibi birçok farklı parametreye bağlıdır. Belirli şartlar altında, hassas ölçüm cihazları kullanılarak elde edilen koordinat doğruluğunu cm hata düzeylerinde elde etmek mümkündür (Atiz ve ark., 2022; Pırtı ve Yazıcı, 2022). Taksilerdeki navigasyon cihazlarından elde edilen veriler için hata düzeyleri değişkenlik göstermektedir. Genellikle aracın durakladığı veya yüksek hızlara çıktığı anlarda kaydedilen verilerde hatalı kayıt riski daha yüksektir. Hata oranı ile ilgili metre cinsinden bir değer vermek mümkün olmamakla birlikte taksi iz verileri düşük doğruluklu veriler olarak nitelendirilmektedir (Zheng ve ark., 2014). Taksilerden elde edilen GPS iz verileri kullanılarak yol orta hattı çıkarımı ile ilgili yapılan çalışmalarda kullanılan veri setleri çoğunlukla csv dosya formatındadır. Zhao ve arkadaşları (2024), konu ile ilgili çalışmalarda sıklıkla kullanılan veri setlerine erişimi sağlayan internet bağlantı adreslerini bir araya getirerek liste halinde, projeleri için hazırladıkları web sayfasında paylaşmışlardır (URL-1). Veri setleri kaynak web sayfalarından indirilerek incelendiğinde bir bölgeye ait cvs uzantılı noktalar (Points) ve seyahatler (Trips) şeklinde iki dosya veya bu iki dosyadaki bilgileri içerecek şekilde düzenlenmiş tek

bir dosya olduğu görülmüştür. Noktalar dosyası nokta IDsi ve nokta koordinatlarını içerirken seyahatler dosyası ise bir taksinin bir yerden başka bir yere seyahati esnasında kaydedilmiş sıralı noktaların ID'lerini zaman damgası ile birlikte içermektedir (URL-2). Tek bir dosya olarak paylaşılan veri setleri ise her bir satırda, her bir nokta için seyahat ID, Nokta ID ve koordinatları içermektedir (URL-1). Bu veri setlerinin tamamında nokta koordinatları olarak enlem (latitude) ve boylam (longitude) değerleri bulunurken yükseklik (elevation) değeri ise yer almamaktadır. NMEA, GPX, KML, PLT ve Excel gibi veri formatlarında kaydedilmiş iz verileri de bir ön işlemden geçirilerek csv formatına dönüştürülebilir. Nitekim bu çalışmada kullanılan veri seti de Excel formatında olup veri seti içerisinde yer alan her bir nokta için koordinatların yanı sıra adres, hız limitlerinin aşılması durumu, hedefe olan uzaklık, tüm zamanlar toplam katedilen mesafe, araç plakası, günlük katedilen mesafe, yol hız sınırı, araç hızı gibi birçok bilgide yer almaktadır. Çalışmada kullanılan veri seti Bölüm 2.1'de açıklanan ön işlemlerden geçirilerek sadeleştirilmiş ve csv dosya formatına dönüştürülmüştür.

Bu çalışmada, özellikle büyük veri setlerinin işlenmesinde karşılaşılan zorluklar ve GPS iz verilerinin öznitelik değerlerinin korunması dikkate alınarak temel bir yaklaşım ile GPS izlerine dayalı yol orta hattının çıkarılması ile ilişkili bir ön işlem olarak veri boyutunun optimizasyonunu amaçlamaktadır. Her bir GPS noktasının zaman bilgisi ve kayıt cihazı ID'siyle sıralı olarak kaydedilmesi, aracın rotası ve hareket yönü hakkında önemli bilgiler sağlamaktadır.

2. Yöntem

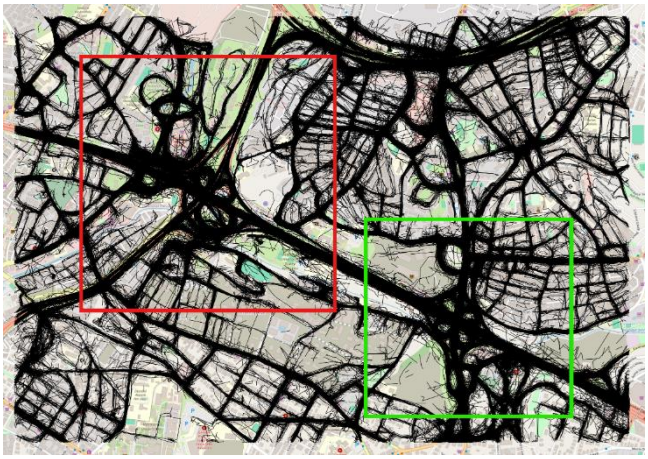
Bu çalışmada kullanılan yöntemler ve adımlar şu şekildedir:

2.1. Veri toplama ve ön işleme

İstanbul'un Göztepe Kavşağı Mevkii'nde çalışan taksilerden elde edilen GPS iz verileri, veri setinin ana kaynağını oluşturmaktadır. Her taksiye ait GPS cihazları, belirli zaman aralıklarında konum bilgisi ve zaman damgası ile veri toplamıştır. Veriler 24-27 Haziran 2019 tarihleri arasında toplanmış yaklaşık 4 günlük bir veridir. Veri seti içerisinde adres, hız limitlerinin aşılması durumu, hedefe olan uzaklık, tüm zamanlar toplam katedilen mesafe, araç plakası, günlük katedilen mesafe, yol hız sınırı, araç hızı gibi birçok bilgide yer almaktadır ancak çalışma kapsamında bu veriler dikkate alınmamıştır. Veri setinde yer alan kayıt zamanı, araç ID (kayıt cihazı ID), enlem (latitude) ve boylam (longitude) değerleri filtrelenerek yeni bir veri seti oluşturulmuştur. Oluşturulan bu ham veri seti toplamda 1224016 adet GPS iz noktası içermektedir. Ancak bunların bazıları birden fazla defa kaydedilmiş nokta tekrarlarını da içermektedir. 353266 nokta yinelendiği için silinmiş ve 770723 adet tekrarsız kayıt elde edilmiştir. Taksi navigasyon cihazları için yazılım üreten özel bir firmadan temin edilen ve Excel formatında olan veri seti, düzenlemeleri kolaylaştırmak için csv formatına dönüştürülmüştür. Kodlama esnasında veriler csv dosyasından okunmaktadır.

Veri seti, zaman damgası ve GPS cihazı ID'si ile sıralanmış şekilde öncelikle düzenlenmiştir. Her bir GPS kaydının zaman sırası, aracın izlediği rotayı belirlemek için kritik öneme sahiptir. Her bir noktaya ait zaman damgası dikkate alınarak, sıralı noktaların kayıt zamanlarının zamansal farkı 1 dk (60sn) dan küçük veya eşit zaman farkına sahipse ve bu ardışık noktaların aralarındaki mesafe 120 metreden küçük veya eşitse, bu noktaların aynı rotada olduğu kabul edilmiş ve bu rotalara aynı rota numarası atanarak veri seti düzenlenmiştir.

Şekil 1'de ham veri seti OpenStreetmap haritası üzerinde görselleştirilerek sunulmuştur (URL-4). Tüm görüntü ve grafiklerin daha yüksek çözünürlükteki versiyonları Github'ta Görseller klasörü içerisinde ayrıca paylaşılmıştır (URL-3). Kırmızı renk ile kutu içerisine alınan bölüm Çalışma Alanı 1 ve yeşil kutu içerisine alınan bölüm ise Çalışma Alanı 2 olarak isimlendirilmiştir. Bu çalışma alanlarının her biri büyük ve karmaşık kavşak yapılarını içermektedir. Çalışma Alanı 1 ve Çalışma Alanı 2 için ayrı ayrı her bir alan içerisinde kalan noktalar ham veriden filtrelenerek Çalışma Alanı 1 ve Çalışma Alanı 2'ye ait veri setleri oluşturulmuştur. Bu çalışma, Çalışma Alanı 2'ye ait veri setine odaklanmaktadır.

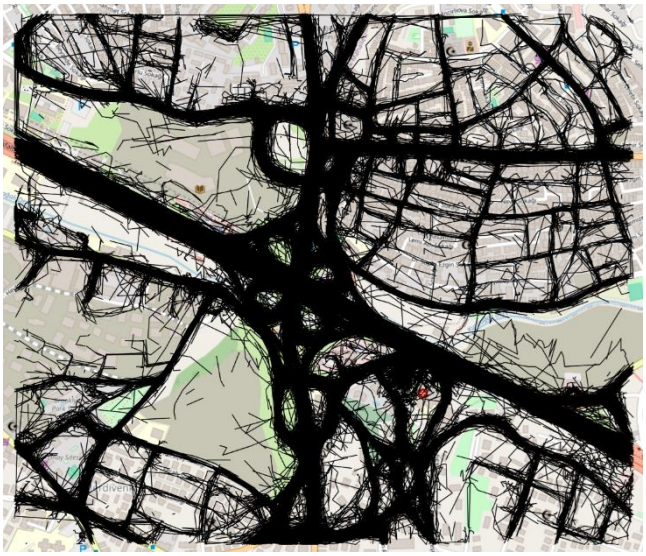


Şekil 1. Ham veri seti OpenStreetMap üzerinde görselleştirilmiş hali (URL-3).

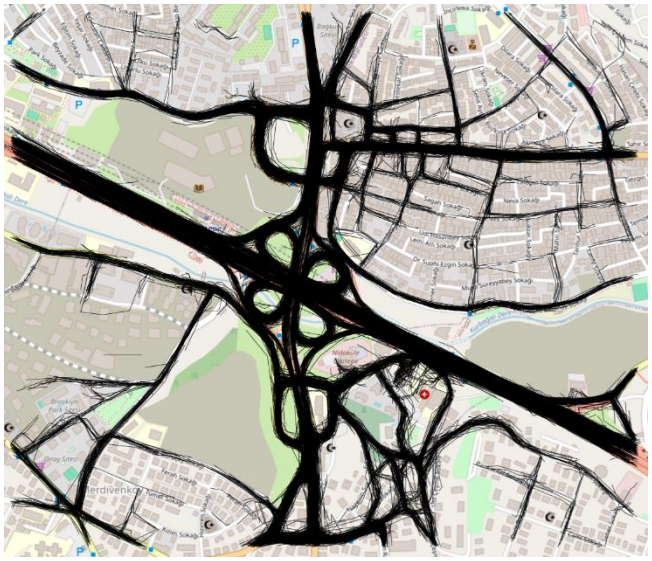
2.1.1. Çalışma Alanı 2 üzerinde ön işlemler

Çalışma Alanı 2 veri seti oluşturulurken çalışma alanına ait köşe koordinatlar belirlenerek bölge tanımlanmış ve bu bölge içerisinde kalan noktalar ham veri setinden filtrelenerek Çalışma Alanı 2 ye ait ham bir veri seti elde edilmiştir. Çalışma Alanı 2 ye ait ham veri seti içerisinde 251160 adet tekrarsız nokta kaydı bulunmaktadır. Çalışma Alanı 2'ye ait ham veri seti Şekil 2a'da gösterilmiştir.

Veri birtakım ön işlemden geçirilmiş ve gürültüden büyük ölçüde arındırılmıştır. Şekil 2a'da sunulan ham veri gürültü ve hatalı kayıtlar içermektedir. Şekil 2b'de sunulan görüntü ise ön işlemler sonrası elde edilen veri setinin harita üzerindeki görselidir. Yapılan ön işlemler aşağıda sıralanmış ve Şekil 3'te iş akış şeması gösterilmiştir.



Şekil 2a. Çalışma Alanı 2'ye ait ham verinin harita üzerindeki görünümü (URL-3).



Şekil 2b. Çalışma Alanı 2 ön işlem sonrası gürültüden temizlenmiş verinin harita üzerindeki görünümü (URL-3).

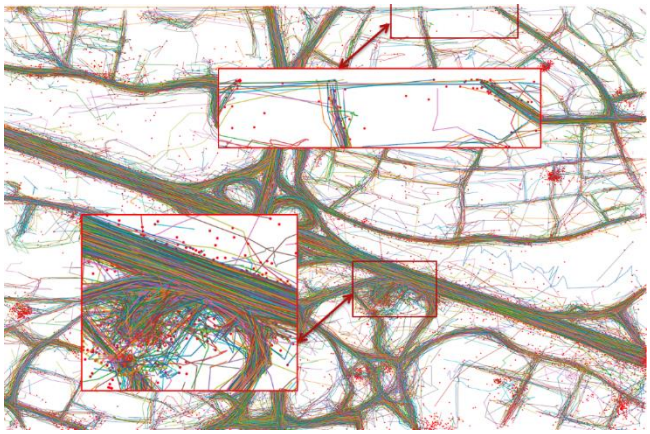
Şekil 3. Ön işlemler iş akış şeması.

1-Rotaların yeniden oluşturulması: Rotalar tüm veri setine ait ham veri setinde daha önceden oluşturulmuş olduğu için çalışma alanı dışına çıkıp daha sonra alana geri dönen gezintiler için rotalarda hatalı kayıtlar oluşmaktadır. Bu tip hatalı kayıtlar tespit edilerek rotalar yeniden düzenlenmiştir. Hatanın oluştuğu hat koparılarak öncesi ve sonrası yeni rota numaraları ile ayrı rotalar olarak kaydedilmiştir.

2-Tekil noktaların silinmesi: Tek bir noktadan oluşan rotalar herhangi bir bağlantı içermeyeceği için bu noktalar silinmiştir.

3-Bekleme anında oluşan zigzaglar azaltılması: Veriler taksilere ait olup yolcu indirip bindirme esnasında, taksi duraklarında veya benzinliklerde yapılan beklemeler esnasında kaydedilen GPS izleri GPS sisteminin hata payına bağlı olarak hatalı ardışık kayıtların oluşmasına sebebiyet vermektedir. Bu verilerde veri setinden temizlenmesi gerekmektedir. Ancak burada etkin bir temizleme yapılamadığı gözlemlenmiştir. Aynı rotadaki doğruların bir bölgedeki yoğunluğuna ve birbirleri ile kesişme durumlarına bakılarak, kesişen doğrular rotadan çıkarılmıştır. Deneysel çalışmalar ve görseller karşılaştırıldığında bekleme anında oluşan hatalı kayıtlarda bir miktar azalma olduğu söylenebilir.

Yukarıda açıklanan durumlarla ilgili olarak Şekil 4'te hatalı kayıt örnekleri sunulmuştur. Alanın sınır bölgelerinde yer alan birleşimler alan dışına çıkıp daha sonra alana geri dönen rota kayıtlarından kaynaklanmaktadır. Rotalar yeniden düzenlenerek bu sorun aşılmıştır. Kırmızı işaretli tekil noktalar herhangi başka bir nokta ile bağlantısı olmayan noktalardır. Veri setinden tekil noktalar silinmiştir. Görselde alt bölümde yer alan hata örneği ise genellikle taksinin bekleme anında oluşan zigzaglı veri kayıtları için bir örnektir. Zigzaglı veri kayıtlarının azaltılması için her bir rotadaki ardışık sıralı noktaların aralarındaki mesafeler kontrol edilerek ardışık iki nokta arası mesafe 10 metreden küçükse bu iki nokta birleştirilmiş ve tek bir nokta ile temsil edilmiştir.



Şekil 4. Çalışma Alanı 2'ye ait hatalı kayıt örnekleri.

4-Ardışık noktaların birleşimi: Aynı rotada aralarındaki mesafe 10 metreden daha az olan ardışık noktalar birleştirilmiştir. Bu şekilde tüm rotalar için bu işlem yapılmaktadır. Tüm rotalar dolaşıldıktan sonra ilk rotaya dönülerek aynı işlem tüm rotalar için tekrar edilmiştir ve şekilde 20 tekrardan sonra işlem

sonlandırılmıştır. Bu işlem döngüsü daha fazla tekrarda daha iyi sonuçlar verebileceği gibi belirli bir aşamadan sonra rotalarda önemli veri kayıplarına da sebep olabilmektedir. Örneğin, bir rotada konum doğruluğu ve veri kayıt hassasiyeti yüksek bir cihazla ortalaması 10 metreden kısa aralıklarla sıralı noktaların kaydedilmiş olduğunu varsayalım. Her seferin aralarında 10 metreden daha kısa mesafe olan ardışık noktalar birleştirileceği için belirli bir iterasyondan sonra bu örnek rotadaki nokta sayısının tek bir noktaya düşmesi durumu deneysel sonuçlarda gözlemlenmiştir. Bu durum ilgili rota bilgisinin tamamen yok olması anlamını taşımaktadır. Deneysel gözlemler neticesinde işlemler 20 tekrardan sonra sonlandırılmıştır.

Tüm noktaları içeren ham veri setinde rotalar oluşturulurken ardışık iki nokta arası mesafe en fazla 120 metre olarak alınmıştı. Böylece bir rotadaki noktalar arası mesafe en az 10 m en fazla ise 120 m olmuştur. İki nokta arası mesafe hesaplanırken Python Geopy.distance kütüphanesi kullanılmıştır (URL-5). Bu kütüphane Jeodezik mesafe hesaplaması yöntemini kullanır (Karney, 2013). Jeodezik mesafe hesaplaması dünya yüzeyinde iki nokta arasındaki en kısa yolu belirlemek için elipsoid (WGS-84) modelini kullanmaktadır. Dünya tam bir küre değil, biraz basık bir elipsoid (yaklaşık olarak "spheroid") olduğu için, bu model daha doğru sonuçlar sağlar. Sharmila ve Sabarish taksilerden toplanan GPS izleri ile yapılan çalışmalarda mesafe ölçümleri için çoğunlukla kullanılan Öklid (Euclidean), Hausdorff ve Haversine mesafeleri gibi farklı mesafe ölçüm yöntemlerini hem TN291 hem de Microsoft T-Drive veri setleri üzerinde K-Ortalamalar ve DBSCAN algoritmaları ile uygulamış ve doğrulamışlardır. Sharmila ve Sabarish'in çalışmalarına göre Haversine mesafe ölçümü hem Öklid hem de Hausdorff mesafelerinden daha iyi performans göstermiş ve hareket eden nesnelerin yörünge verileri açısından daha iyi geçerlilik ve doğruluk sonuçları elde edilmesini sağlamıştır (Sharmila ve Sabarish, 2021). Haversine formülünü kullanarak mesafe ölçümü yapan bir Python modülünün yayınlandığı web sayfasında, Dünya neredeyse küresel olduğundan, Haversine formülü, ortalama %1'den daha az bir hatayla, Dünya yüzeyinin iki noktası arasındaki mesafenin iyi bir tahminini sağladığı belirtilmektedir ve formül aşağıdaki şekilde verilmiştir. (URL-6).

$$a = sin^2\left(\frac{x_{lat} - y_{lat}}{2}\right) \quad (1)$$

$$\mathrm{b} = \cos(x_{lat})\cos(y_{lat})\, sin^2\left(\frac{x_{lon} - y_{lon}}{2}\right) \quad (2)$$

Denklem 1 ve Denklem 2 ile hesaplanan değerler Denklem 3'te yerine yazılarak $D(x, y)$ hesaplanır.

$$D(x, y) = 2\arcsin\left(\sqrt{a + b}\right) \quad (3)$$

Denklem 3'te verilen formül ile hesaplanan Haversine (veya büyük daire) mesafesi, bir kürenin yüzeyindeki iki nokta arasındaki açısal mesafedir. Her noktanın radyan cinsinden verilen koordinatları için ilk koordinatının enlem, ikincisinin ise boylam olduğu varsayılır. İki nokta, x ve y şeklinde gösterilmiştir.

Benzer şekilde dünya yüzeyindeki herhangi iki nokta arasındaki mesafe ölçümü için kullanılan bir başka Python modülü olan Geopy kütüphanesi incelendiğinde; Jeodezik mesafe, dünyanın elipsoidal modelinin yüzeyindeki en kısa mesafe olarak tanımlanmakta ve varsayılan algoritma olarak Karney (jeodezik) tarafından verilen yöntemi kullanmaktadır. Ek olarak modülün yayınlandığı web sayfasında Yerkürenin WGS-84 modeli için yaklaşık %0,5'e kadar bir hata hatayla sonuçların elde edildiği belirtilmektedir (URL-5). Kayney'in çalışması incelendiğinde iç içe birçok karmaşık denklemden yararlanılarak hesaplamaların yapıldığı görülmüştür. Karney iki nokta arasındaki mesafeyi s12 olarak tanımlar ve Denklem 4'te verilen formül ile hesaplar (Karney, 2013).

$$s_{12} = a\bar{w}\sigma_{12} \quad (4)$$

Denklem 4'te yer alan a, $\bar{w}$ ve σ_{12} ifadeleri aşağıda açıklanmıştır.

a: Elipsoidin büyük yarıçapı (eşdeğer yarıçapı) (Karney, 2013). Bu değer Geopy Python kütüphanesinde WGS-84 modeli için yaklaşık olarak 6371009 metre olarak alınmıştır (URL-5).

$\bar{w}$: Bu değer, dünya üzerindeki iki nokta arasındaki mesafeyi hesaplamak için kullanılan bir yardımcı değişkendir ve elipsoid şeklindeki dünya modeline dayanır. $\bar{w}$ değeri, enlem farkları ve dünya'nın eksantrikliği (e) kullanılarak hesaplanır (Karney, 2013).

σ_{12}: iki nokta arasındaki jeodezik (en kısa yol) uzunluğunun hesaplanmasında kullanılan bir parametredir. Bu parametre, iki nokta arasındaki coğrafi koordinatlara (enlem ve boylam) dayalı olarak hesaplanır ve büyük çember üzerinde bu iki nokta arasındaki açısal mesafeyi ifade eder (Karney, 2013).

Birbirini takip eden GPS izleri arasındaki mesafelerin birbirine yakınlığı dikkate alındığında Öklid, Hausdorff, Haversine ve Jeodezik mesafe ölçüm yöntemleri ile hesaplanan mesafelerin birbirine çok yakın olması beklenir. Bu bağlamda, çok büyük veri setleri üzerinde işlem yaparken hesaplama karmaşıklığını azaltmak için daha az işlem karmaşasına sahip Denklem 5'te verilen Öklid mesafesi de kullanılabilir.

$$D(x, y) = \sqrt{(x_{lat} - y_{lat})^2 + (x_{lon} - y_{lon})^2} \quad (5)$$

Yukarıda belirtilen hususlar dikkate alınarak bu çalışmada, hata oranı Haversine yöntemine göre daha düşük olduğu belirtilen Jeodezik mesafe hesaplaması yöntemi kullanılmıştır. Jeodezik mesafe hesaplaması yöntemini kullanan algoritmanın zamansal karmaşıklığı bu çalışmada kullanılan veri setinin aşırı büyük olmamasından dolayı göz ardı edilmiştir.

Yukarıda bahsi geçen düzeltmelere ek bir rotadaki en az nokta sayıları da düzenlenmiştir. Bir rota en az sıralı iki nokta içermelidir ancak hataları minimize etmek adına en az 3 nokta içeren rotalar filtrelenerek yeni veri seti oluşturulmuş ve Çalışma Alanı 2 için Şekil 2b'deki görsel elde edilmiştir. Veri seti, Points_DataSet (Nokta Koordinatlarını içerir) ve Routes_DataSet (Rota Numaralarını içerir) şeklinde 2 ayrı veri seti olarak

kaydedilmiştir. Tüm bu işlemler neticesinde Çalışma alanı 2'ye ait ham veri seti içerisinde bulunan 251160 adet farklı nokta sayısı, 71645 adete düşmüştür.

2.2. GPS izlerinin birleştirilmesi ve optimizasyonu: algoritma

Çalışma Alanı 2 için ön işlemden geçirilen veri seti Points_DataSet (Nokta Koordinatlarını içerir) ve Routes_DataSet (Rota Numaralarını içerir) şeklinde iki ayrı tablo olarak kaydedilmiştir. Veri yapısı GPX veya NMEA gibi standart bir veri kayıt formatında değildir. İşlem kolaylığı açısından, her bir nokta için nokta kimliği ve koordinatlarını içeren bir Noktalar (Points_DataSet) veri seti ve bir de aracın hareketi esnasında bir rota (güzergâh) üzerinde kaydedilmiş sıralı noktaların tutulduğu Rotalar (Routes_DataSet) veri seti oluşturulmuş ve csv formatında kaydedilmiştir.

Points_DataSet: Her bir nokta hakkında PointID, enlem, boylam, ağırlık ve birleşim sayısı gibi bilgileri içerir. Bu veri seti İstanbul Göztepe kavşağı mevkisinde kaydedilmiş 71645 nokta adet nokta içermektedir.

Routes_DataSet: Bir noktanın varsa bir sonraki ve önceki noktalara olan bağlantılarını ve doğrultu açılarını içerir. Bu veri setinde, bir rotadaki herhangi bir nokta için RotaID, PointID, DirectionAngleToPrev, DirectionAngleToNext şeklinde bilgiler saklanmaktadır. Bu veri seti için bir rotanın en az 3, en fazla ise 133 adet nokta içerdiği görülmüştür.

RotaID: Noktanın ait olduğu Rota Numarası

PointID: Noktanın ID'si, Point_DataSet'den noktanın coğrafi koordinat değerlerine ulaşmak için gerekli

DirectionAngleToPrev: Aynı rotada varsa kendinden önceki noktaya doğru doğrultu açısı

DirectionAngleToNext: Aynı rotada varsa kendinden sonraki noktaya doğru doğrultu açısı

Doğrultu açısı hesaplanırken Python GeographicLib kütüphanesi kullanılmıştır (URL-7). Bu kütüphane Jeodezik yöntemi kullanarak iki nokta arasındaki doğrultu açısını hesaplar. Mesafe hesaplamasında olduğu gibi Karney'in çalışmasını referans alan bu kütüphane Denklem 9'da verilen formül ile iki nokta arasındaki doğrultu açısını hesaplar (Karney, 2013).

$$\Delta\lambda = y_{lon} - x_{lon} \tag{6}$$

$$A = cos(y_{lat})\, sin(\Delta\lambda) \tag{7}$$

$$B = cos(x_{lat})sin(y_{lat}) - sin(x_{lat})cos(y_{lat})cos(\Delta\lambda) \tag{8}$$

$$\alpha_1 = atan2(A, B) \tag{9}$$

Burada, $atan2(A, B)$ fonksiyonu, B'nin A'ya oranının ters tanjantını (arctangent) verir ve doğru çeyreği belirler.

Karney'nin çalışmaları, Vincenty'nin formüllerine göre daha kararlı ve yüksek doğrulukta sonuçlar üretir (Karney, 2013). Karney'nin yöntemlerini uygulamak için, onun geliştirdiği ve çeşitli programlama dillerinde mevcut olan jeodezik hesaplama kütüphanelerini kullanmak genellikle en etkili yoldur. Bu kütüphaneler, iteratif çözümleri ve elipsoid üzerindeki düzeltmeleri doğru şekilde uygulayarak enlem ve boylamdan azimut ve mesafe hesaplamalarını kolaylaştırır. Jeodezik hesaplama yönteminin hazır kütüphane sayesinde kullanım kolaylığı olması açısından bu çalışmada GeographicLib kütüphanesi içerisinde yer alan açı hesaplama fonksiyonu kullanılmıştır. Fonksiyonun kolay kullanımı için aşağıdaki şekilde bir metot tanımlanmıştır.

```
def get_DirectionAngle(lat1,long1,lat2,long2):
    return Geodesic.WGS84.Inverse(lat1, long1,
    lat2, long2)['azi1']
```

Doğrultu hesaplaması yapılacak noktalar sıralı olarak fonksiyon içerisinde tanımlanır ve hesaplama yapılır. Burada doğrultu, birinci tanımlanan noktadan ikinci tanımlanan noktaya doğru olacak şekilde kabul edilmektedir. Fonksiyon Azimut açısını hesaplar. Azimut açısı, bir noktadan bir nesneye olan yönü belirlemek için kullanılan yatay açı ölçüsüdür. Referans yönü genellikle kuzeydir ve saat yönünde ölçülür. Yukarıda tanımlanan metot -180° ile 180° arasında Azimut açısı değeri döndürür (URL-7). Bu çalışmada, yukarıdaki metot kullanılarak hesaplanan açı değeri DirectionAngle (doğrultu açısı) olarak isimlendirilmiştir. -180° ile 180° aralığında dönen değer eğer negatifse 360 ilave edilerek pozitif yapılır. Böylece DirectionAngle değeri 0° ile 360° derece aralığında elde edilmiş olur.

GPS iz verileri, mesafe ve yön eşik değerleri kullanılarak gruplandırılmıştır. Her bir nokta için ait olduğu rotada (varsa) öncesindeki ve (varsa) sonrasındaki noktaya doğru doğrultu açıları hesaplanmış ve Routes_DataSet tablosuna DirectionAngleToPrev ve DirectionAngleToNext şeklinde eklenmiştir. İlerleyen bölümlerde detayları açıklanan belirli bir kurala göre veri setinden sıra ile seçilen her bir nokta için, seçilen noktaya belirlenen mesafeden daha yakın noktalar gruplanır ve daha sonra bu grup içerindeki noktaların doğrultularına bakılarak seçilen nokta ile ileri veya geri yönlü doğrultuları coğrafi olarak yakın olan noktalar grup içerisinde kalırken doğrultu eşik değerini aşan durumdaki noktalar ise gruptan çıkartılır. Böylece seçilen nokta ile coğrafi olarak belirli bir mesafe yakınlıktaki ve belirli bir eşik değer altında açısal farka sahip doğrultudaki noktalar tespit edilmiş olur. Her bir grup, coğrafi olarak yakın mesafedeki ve hareket yönü olarak da yakın doğrultudaki noktalardan oluşmaktadır. Buradaki amaç birbirine yakın ve aynı doğrultudaki noktaları birleştirerek, birleşen grup için bir temsil noktası oluşturmak ve veri setindeki nokta sayısını azaltmaktır.

İlgili tablolar üzerinde yapılan işlemler için Python programlama dili ve NumPy kütüphanesi kullanılmıştır. (URL-8). NumPy kütüphanesi veri setleri üzerinde çalışırken oldukça hızlı işlemler yapılabilmesine olanak sağlar. Verinin okunması, sıralanması, tekrarların silinmesi gibi birçok işlemi pratik olarak yapma imkânı sunar.

Bu çalışmada bahsi geçen verinin işlenmesi, noktaların gruplanması ve birleştirilmesi, verinin optimizasyonu ile ilgili tüm kodlar, giriş ve çıkış veri setleri, harita çıktıları ve grafikler Github'ta açık erişim olarak yayınlanmıştır (Demiral, 2024).

Verinin Optimizasyonu için yapılan işlemler aşağıda sırası ile açıklanmıştır.

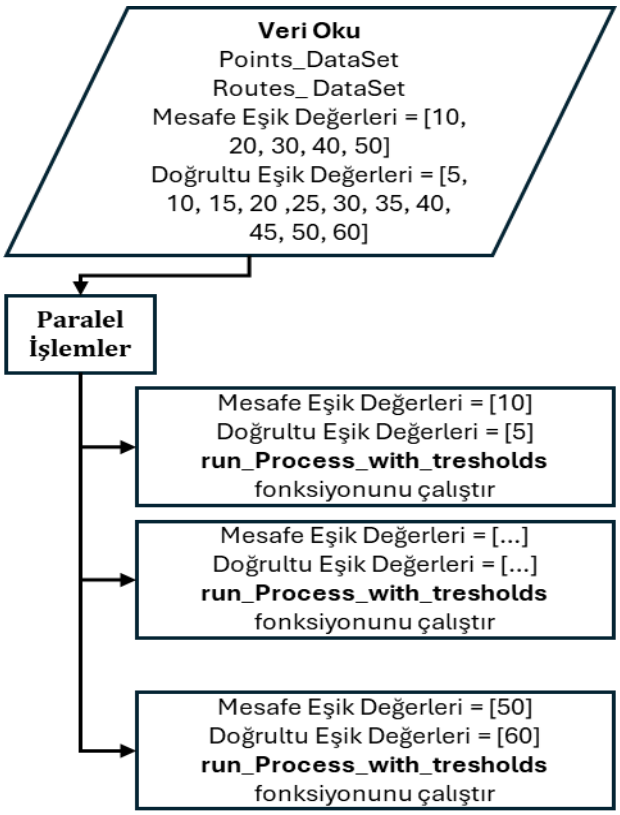
Verilen algoritma, GPS verilerini kullanarak belirli mesafe ve açı farkı eşik değerlerine göre noktaları gruplandırmayı ve birleştirmeyi amaçlamaktadır. Süreç, büyük veri setlerini daha yönetilebilir hale getirmek ve navigasyon sistemlerinde daha verimli rotalar oluşturmak için kullanılır. Kod, noktaların merkezden uzaklıklarına göre sıralanmasını, yakın komşuların belirlenmesini ve gruplandırılmasını, ardından gruplandırılmış noktaların birleştirilmesini içerir.

Bu algoritma paralel işleme (parallel processing) kullanılarak hızlandırılabilir. Python'da *ProcessPoolExecutor* kullanılarak farklı mesafe ve açı eşik değerleri için işlem süreçleri paralel olarak yürütülür. Bu, işlemlerin daha hızlı tamamlanmasını sağlar ve büyük veri setlerinin işlenmesini daha verimli hale getirir.

Bu bölümde uygulanan işlemler, kullanılan algoritmalar ve çıktılara ait detaylar şu şekildedir:

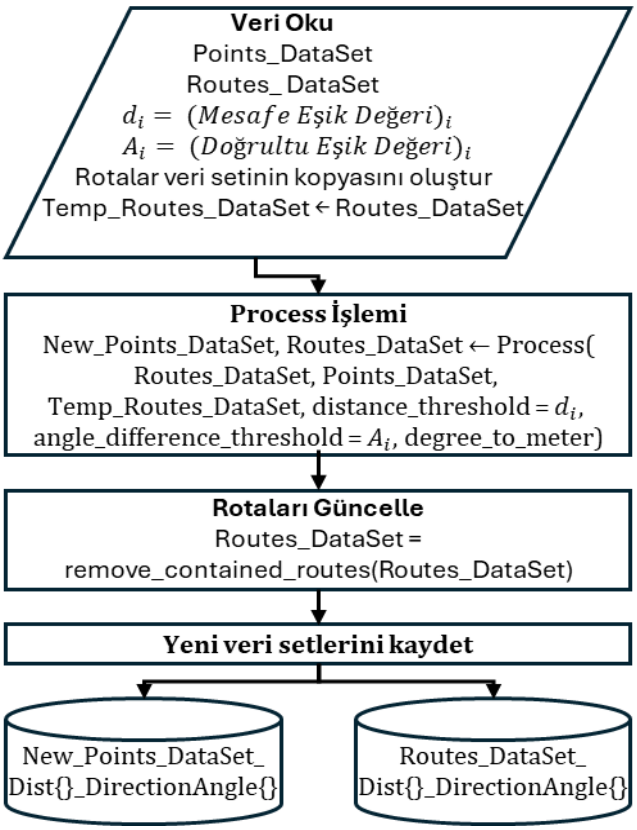
Ön işlemlerden sonra oluşturulan Noktalar (Points_DataSet) ve Rotalar (Routes_DataSet) veri setleri okunarak işlemlere başlanır. Daha sonra test edilmek istenen Mesafe ve Doğrultu Açısı eşik değerleri listesi girdi olarak okunur. Şekil 5'te algoritmanın paralel işlemleri gösteren genel akış şeması yer almaktadır.

run_Process_with_thresholds adında bir işlev tanımlanmıştır. Bu işlev farklı mesafe ve açı eşik değer girdilerine göre paralel işlem yapılabilmesine olanak tanır. Farklı mesafe ve açı eşik değer girdileri ile yapılan işlemler birbirinden bağımsızdır.



Şekil 5. Algoritma genel akış şeması.

run_Process_with_thresholds fonksiyonu ile ilgili olarak algoritma ve Şekil 6'da algoritma akış şeması verilmiştir.



Şekil 6. run_Process_with_thresholds fonksiyonu algoritma akış şeması.

Algoritma: run_Process_with_thresholds Fonksiyonu

```
_____________________________________________
Giriş: (distance_threshold: Mesafe eşiği,
angle_difference_threshold: Açı farkı eşiği,
degree_to_meter: Dereceyi metre cinsine dönüştürme
katsayısı)

_____________________________________________
Prosedür:
1.Globalleri Tanımla:
• global Points_DataSet, Routes_DataSet,
  Temp_Routes_DataSet
2.Başlangıç Veri Kümesini Yükle:
• Points_DataSet, Routes_DataSet ←
  ReadDataSets(FileName1='Points_DataSet',
  FileName2='Routes_DataSet_Dist20_DirectionAngle
  12')
• Temp_Routes_DataSet ← Routes_DataSet'in bir
  kopyası
3.İşleme Sürecini Başlat:
• New_Points_DataSet, Routes_DataSet ←
  Process(Routes_DataSet, Points_DataSet,
  Temp_Routes_DataSet, distance_threshold,
  angle_difference_threshold, degree_to_meter)
4.Rotaları Güncelle:
• Routes_DataSet =
  remove_contained_routes(Routes_DataSet)
5.Yeni Veri Setlerini Kaydet:
• {}/New_Points_DataSet_Dist{}_DirectionAngle{}.cs
  v dosyasına New_Points_DataSet kaydet
• {}/Routes_DataSet_Dist{}_DirectionAngle{}.csv
  dosyasına Routes_DataSet kaydet

_____________________________________________
```

Açıklama:

1. ReadDataSets Fonksiyonu:
   İlk veri kümelerini diskten yükler.
2. Process Fonksiyonu:
   Belirtilen eşiklerle birlikte Routes_DataSet ve Points_DataSet üzerinde birleştirme işlemini yürütür. İşlem sonucunda New_Points_DataSet ve Routes_DataSet tabloları oluşturulur

3. remove_contained_routes Fonksiyonu:

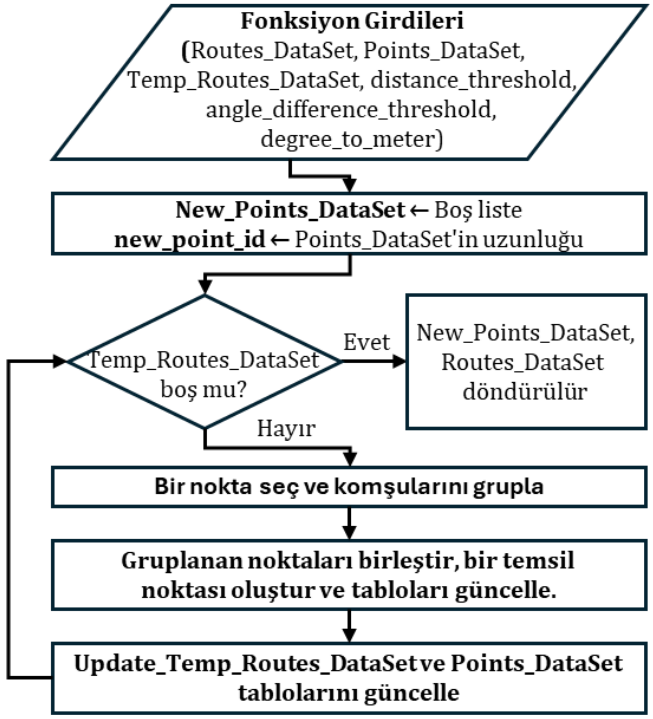
İçerilmiş rotaları kaldırır. Birleşim sonrası bir rotadaki tüm noktalar bir başka rotada da olabilir. Bu durum, uzun bir rotanın kısa bir rotadaki tüm noktaları tamamen içermesi durumudur. Bu durumda kısa rota, uzun rota tarafından zaten temsil edileceği için veri tekrarının önüne geçmek adına veri seti yeniden düzenlenir ve tekrarlı rotalar temizlenir.

4. Dosya Kaydetme ve Ekrana Yazdırma:

Oluşturulan yeni veri kümelerini belirtilen dosya adlarına kaydeder

Dosya okuma ve kaydetme işlemleri genel işlemler olduğu için bu aşamalar üzerinde durulmamıştır. *run_Process_with_thresholds* fonksiyonu içerisinde *Process, remove_contained_routes* şeklinde iki alt fonksiyon çalıştırmaktadır. Alt işlemleri gerçekleştiren fonksiyonların her birine ait algoritmalar ve akış şemaları hiyerarşik sırasına uygun şekilde aşağıda yer almaktadır.

Process fonksiyonu ile ilgili olarak Şekil 7'de algoritma akış şeması verilmiştir.



Şekil 7. Process fonksiyonu algoritma akış şeması.

Fonksiyon 1: Process Fonksiyonu

```
_____________________________________________
Giriş: (Routes_DataSet: Rota veri kümesi,
Points_DataSet: Nokta veri kümesi,
Temp_Routes_DataSet: Geçici rota veri kümesi,
distance_threshold: Noktaların birleştirilmesi
için mesafe eşiği, angle_difference_threshold:
Noktaların birleştirilmesi için açı farkı eşiği,
degree_to_meter: Dereceyi metre cinsine dönüştürme
katsayısı)

_____________________________________________
Prosedür:
1.New_Points_DataSet ← Boş liste
2.new_point_id ← Points_DataSet'in uzunluğu
3.Temp_Routes_DataSet boş olana kadar:
• current_group, current_group_indices ←
  SelectPointAndGetNeigbours(Points_DataSet,
  Temp_Routes_DataSet, distance_threshold,
  angle_difference_threshold, degree_to_meter)
• New_Points_DataSet, Routes_DataSet, new_point_id
  ←
  MergeGrupedPointsAndUpdateTables(current_group,
  current_group_indices, Points_DataSet,
  New_Points_DataSet, Routes_DataSet,
  new_point_id)
• Temp_Routes_DataSet, Points_DataSet ←
  Update_Temp_Routes_DataSet_and_Points_DataSet(c
  urrent_group, current_group_indices,
  Temp_Routes_DataSet, Points_DataSet)
4.Çıkış:
• New_Points_DataSet, Routes_DataSet döndürülür
_____________________________________________
```

Açıklama:

1. SelectPointAndGetNeigbours_Optimized Fonksiyonu:

Points_DataSet ve Temp_Routes_DataSet üzerindeki nokta gruplarını belirler. Bu gruplar, belirli mesafe ve açı eşiklerini kullanarak seçilir.

2. MergeGrupedPointsAndUpdateTables Fonksiyonu:

Seçilen nokta gruplarını birleştirir ve yeni bir nokta oluşturur. Bu işlemi yaparken New_Points_DataSet ve Routes_DataSet tablolarını günceller.

3. Update_Temp_Routes_DataSet_and_Points_DataSet Fonksiyonu:

İşlenen noktaları ve rotaları Temp_Routes_DataSet ve Points_DataSet üzerinde günceller.

Process fonksiyonu içerisinde işlemler rotalar üzerinden yürütülmektedir. Temp_Routes_DataSet içerisindeki rotalar işlenerek işlem yapılan rota Temp_Routes_DataSet den çıkarılır ve böylece tekrar işleme alınmaz. Yukarıda listelenen üç alt fonksiyon *Process* fonksiyonu içerisinde çalıştırmaktadır. Alt işlemleri gerçekleştiren fonksiyonların her birine ait algoritmalar ve akış şemaları hiyerarşik sırasına uygun şekilde aşağıda yer almaktadır.

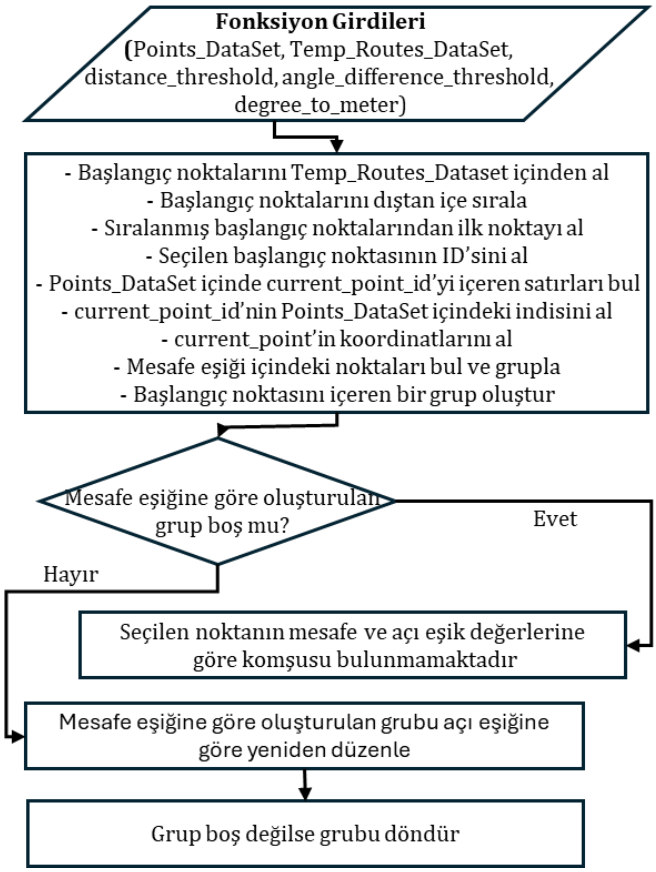
Fonksiyon 1.1: SelectPointAndGetNeigbours

```
_____________________________________________
Giriş: (Points_DataSet: Nokta veri kümesi,
Temp_Routes_DataSet: Geçici rota veri kümesi,
distance_threshold: Noktaların birleştirilmesi
için maksimum mesafe eşiği,
angle_difference_threshold: Noktaların
birleştirilmesi için maksimum açı farkı eşiği,
degree_to_meter: Dereceyi metre cinsine dönüştürme
katsayısı)

_____________________________________________
Prosedür:
1.start_points ←
get_start_points(Temp_Routes_DataSet): Başlangıç
noktalarını Temp_Routes_DataSet içinden al.
2.sorted_start_points ←
Sort_innermost_points(start_points): Başlangıç
noktalarını dıştan içe sırala.
3.current_point ← sorted_start_points'tan bir nokta
çıkar (pop(0)): Sıralanmış başlangıç noktalarından
ilk noktayı al.
4.current_point_id ← current_point'ın 1. indeksi:
Başlangıç noktasının ID'sini al.
5.mask ← Points_DataSet içinde current_point_id'yi
içeren satırları bul.
6.current_point_indices ← mask'ten 0. indeksi:
current_point_id'nin Points_DataSet içindeki
indisini al.
7.current_point_coords ←
Points_DataSet[current_point_indices][1:3]:
current_point'ın koordinatlarını al (Lat ve Lon
değerleri).
8.nearby_points_IDs_InPointDataSet ←
find_points_within_distance(current_point_indices,
Points_DataSet, distance_threshold,
degree_to_meter): Mesafe eşiği içindeki noktaları
bul.
```

9.current_group ← [current_point_id]: Başlangıç noktasını içeren bir grup oluştur.
10. Eğer nearby_points_IDs_InPointDataSet'in uzunluğu > 0 ise:
- grouped_nearby_points_withDirectionAngle_InPoint DataSet ← group_nearby_points(current_point_id, nearby_points_IDs_InPointDataSet, Points_DataSet, Temp_Routes_DataSet, angle_difference_threshold): Açı eşiği ile birleştirilmiş yakın noktaları grupla.
- current_group'a grouped_nearby_points_withDirectionAngle_InPoin tDataSet'i ekle.

11.mask ← Points_DataSet içinde current_group'u içeren satırları bul.
12.current_group_indices ← mask'ten indisleri al
13.Çıkış:
- current_group ve current_group_indices döndürülür.



Şekil 8. SelectPointAndGetNeigbours fonksiyonu algoritma akış şeması.

Açıklama:

1. get_start_points Fonksiyonu:

Temp_Routes_DataSet içinden başlangıç noktalarını belirler. Her bir rotanın ilk noktası başlangıç noktasıdır. Başlangıç noktaları grubu oluşturulur.

2. Sort_innermost_points Fonksiyonu:

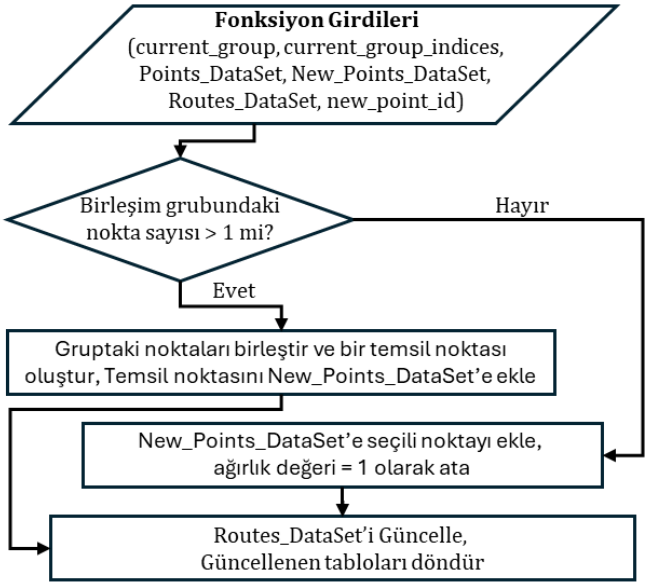
Başlangıç noktalarını içten dışa sıralar. Tüm noktaların ortalamasını alarak orta nokta bulunur. Orta noktaya en yakın başlangıç noktası tespit edilir. En içteki başlangıç noktası birleştirme işlemi için seçilir. Birleştirme işlemi her zaman en içteki başlangıç noktaları üzerinden yürütülür.

3. find_points_within_distance Fonksiyonu:

Verilen mesafe eşiği içinde belirli bir noktanın yakınındaki noktaları belirler ve gruplar.

4. group_nearby_points Fonksiyonu:

Belirli bir noktanın çevresindeki noktaları açı eşik değerine göre gruplar.



Şekil 9. MergeGrupedPointsAndUpdateTables fonksiyonu algoritma akış şeması.

Fonksiyon1.2: MergeGrupedPointsAndUpdateTables

Giriş: (current_group: Birleştirilecek nokta grubu, current_group_indices: Points_DataSet içindeki current_group'un indisleri, Points_DataSet: Nokta veri kümesi, New_Points_DataSet: Yeni nokta veri kümesi, Routes_DataSet: Rota veri kümesi, new_point_id: Yeni oluşturulacak noktanın ID'si)

Prosedür:
1.Eğer current_group içinde birden fazla nokta varsa:
- total_lat ← current_group_indices içindeki noktaların Points_DataSet'teki latitude değerlerinin ağırlıklı toplamı
- total_lon ← current_group_indices içindeki noktaların Points_DataSet'teki longitude değerlerinin ağırlıklı toplamı
- total_weight ← current_group_indices içindeki noktaların ağırlıklarının toplamı
- avg_lat ← total_lat / total_weight
- avg_lon ← total_lon / total_weight
- new_point_id ← new_point_id + 1
- new_point ← [new_point_id, avg_lat, avg_lon, total_weight, 1]
- New_Points_DataSet'e new_point ekle

2.Değilse (yani current_group sadece bir nokta içeriyorsa):
- point_id ← int(current_group_indices[0])
- avg_lat, avg_lon, total_weight ← Points_DataSet[point_id, 1:4]
- new_point_id ← new_point_id + 1
- new_point ← [new_point_id, avg_lat, avg_lon, total_weight, 0]
- New_Points_DataSet'e new_point ekle

3.Routes_DataSet'i güncelle:
- current_group'u içeren rotaların ikinci sütununu (Routes_DataSet[:, 1]) new_point_id ile güncelle.

4.Çıkış:
- Güncellenmiş New_Points_DataSet, Routes_DataSet ve new_point_id döndürülür.

Açıklama:

1. total_lat, total_lon:
   Ağırlıklı ortalama hesaplaması yapar, birden fazla noktanın birleşimini temsil eder.
2. new_point:
   Yeni oluşturulan noktanın bilgilerini ve birleşim durumunu (1 veya 0) içerir. Nokta birleşmişse 1, birleşmemişse 0 değeri atanır. Daha sonra gerek görülürse birleşime uğramamış noktalar istisnai noktalar(gürültü) kabul edilip verisetinden silinebilir.
3. Routes_DataSet[np.isin(Routes_DataSet[:, 1], current_group), 1] = new_point_id: current_group'u içeren rotaların ikinci sütununu new_point_id ile günceller.

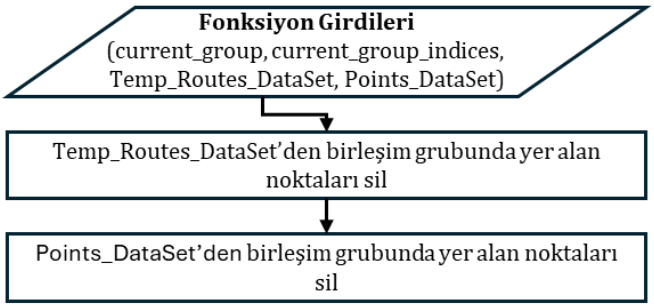
Fonksiyon1.3: Update_Temp_Routes_DataSet_and_Points_DataSet

```
Giriş: (current_group: Güncellenecek nokta grubu, current_group_indices: Points_DataSet içindeki current_group'un indisleri, Temp_Routes_DataSet: Geçici rota veri kümesi, Points_DataSet: Nokta veri kümesi)

Prosedür:
1.Temp_Routes_DataSet içinde current_group'a ait olmayan rotaları filtrele:
• mask ← Temp_Routes_DataSet'in ikinci sütununda ([:, 1]) current_group içinde olmayanları seç (invert=True).
• Temp_Routes_DataSet ← mask'e göre filtrelenmiş Temp_Routes_DataSet.
2.Points_DataSet içinde current_group'a ait olan noktaları kaldır:
• mask_points ← Points_DataSet'in current_group_indices indislerini False, geri kalanlar True yaparak oluştur.
• Points_DataSet ← mask_points'e göre filtrelenmiş Points_DataSet.
3.Çıkış:
• Güncellenmiş Temp_Routes_DataSet ve Points_DataSet döndürülür.
```

Açıklama:

1. mask: Temp_Routes_DataSet içinde current_group'a ait olmayan rotaları belirlemek için kullanılır.
2. mask_points: Points_DataSet içinde current_group_indices indislerine sahip noktaları kaldırmak için kullanılır.
3. Fonksiyon, Temp_Routes_DataSet ve Points_DataSet üzerinde yapılan değişiklikleri geri döndürür.

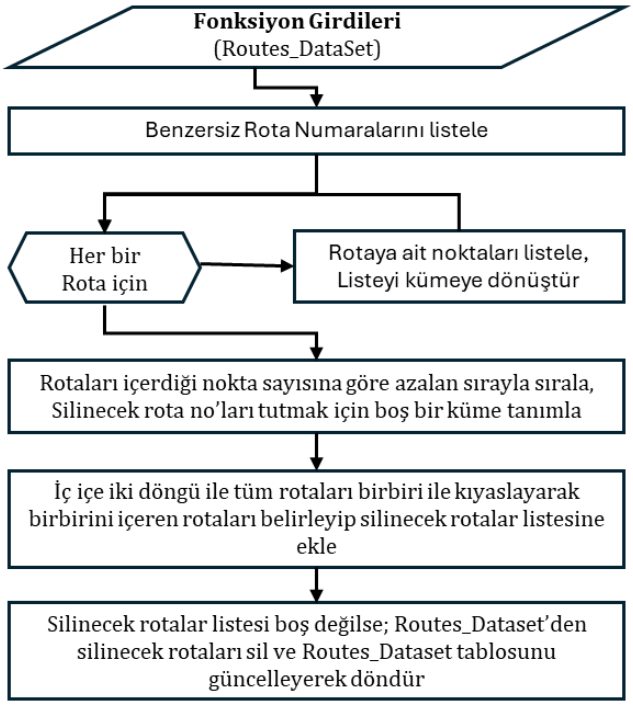


Şekil 10. Update_Temp_Routes_DataSet_and_Points_DataSet fonksiyonu algoritma akış şeması.

Fonksiyon 2: remove_contained_routes Fonksiyonu

```
Giriş: Routes_DataSet: Rota veri kümesi

Prosedür:
1.unique_routes ← Routes_DataSet[:, 0]'dan benzersiz rotaları al.
2.route_points_dict ← Boş bir sözlük: Her rota için nokta kümesini tutacak.
3.Her bir route_no için:
    • route_points ← Routes_DataSet içinde route_no'ya ait olan rotaların nokta listesi oluştur.
    • route_points_dict[route_no] ← route_points: route_points_dict içine route_no ve ona ait nokta kümesini ekle.
4.sorted_routes ← route_points_dict'i nokta kümesine göre azalan sırayla sırala.
5.to_remove ← Boş bir küme: Silinecek rotaları tutacak.
6.Her bir route1 ve route1_points için (sorted_routes içinde):
• Eğer route1 zaten to_remove kümesinde ise devam et.
• Her bir route2 ve route2_points için (route1'den sonraki rotalar için):
    o Eğer route2 zaten to_remove kümesinde ise devam et.
    o Eğer route1_points, route2_points'i içeriyorsa:
        • to_remove kümesine route1'i ekle ve döngüyü kır.
    o Eğer route2_points, route1_points'i içeriyorsa:
        • to_remove kümesine route2'yi ekle.
7.mask ← Routes_DataSet içinde to_remove kümesinde olmayan rotaları işaretleyen boolean maske.
8.Routes_DataSet ← mask'i kullanarak Routes_DataSet'i güncelle (sadece to_remove kümesinde olmayan rotaları tut).
9.Çıkış:
• Güncellenmiş Routes_DataSet döndürülür.
```



Şekil 11. remove_contained_routes fonksiyonu algoritma akış şeması.

Açıklama:

1. np.unique(Routes_DataSet[:, 0]): Rotaların benzersiz numaralarını alır.

2. Routes_DataSet[Routes_DataSet[:, 0] == route_no][:, 1]: Belirli bir rota numarasına sahip rotaların nokta kümesini alır.
3. sorted_routes: route_points_dict'i nokta sayısına göre sıralar.
4. to_remove: İç içe geçmiş rotaları tespit etmek için kullanılır.
5. np.isin(Routes_DataSet[:, 0], list(to_remove), invert=True): to_remove kümesinde olmayan rotaları seçer.

Yukarıda yer alan algoritma işlemlerini özetlemek gerekirse;

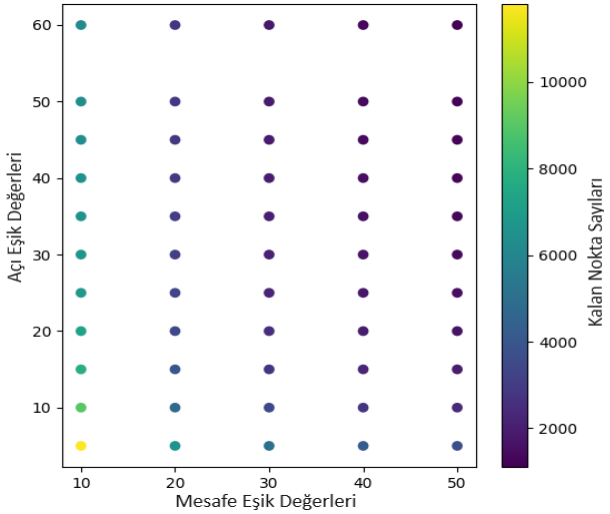
Veri setindeki noktalar ve rotaları içeren Points_DataSet (Nokta Koordinatlarını içerir) ve Routes_DataSet (Rota Numaralarını içerir) tabloları üzerinde yürütülen algoritmalar öncelikle paralel programlamaya uygun şekilde tasarlanmıştır. Kullanıcı tarafından veri setine bağlı olarak belirlenecek açı ve mesafe eşik değerleri sıralı bir liste halinde tanımlanır ve aynı anda farklı eşik değerleri için işlemler paralel olarak yürütülür. Belirlen açı ve mesafe eşik değerlerinin her biri için aynı işlemler yapılmaktadır. Tüm rotalardaki başlangıç noktaları belirlenir ve en içteki noktadan işleme başlanır. Seçilen bu nokta için mesafe eşik değerine göre komşuları gruplanır. Komşuları içinden açı eşik değerine göre yeni bir alt grup oluşturulur ve böylece hem mesafe hem de açı eşik değerlerine uygun komşular elde edilmiş olur. Seçilen en içteki başlangıç noktası ve komşuları birleştirilerek yeni bir temsil noktası ile temsil edilir. Birleşimde yer alan tüm noktalar temsil noktası ile güncellenir. İşlem bu şekilde işlenmemiş nokta kalmayıncaya kadar devam ettirilir ve tüm noktalar için bir temsil noktası elde edilir. Nokta yoğunluğu az olan bölgelerde yer alan bazı noktalar birleşime uğramayabilir. Bu durumda dikkate alınmış ve bu gibi noktalar korunarak aynen eklenmiştir ancak birleşim durum değişkenine değer olarak 0 değeri atanmıştır. Çıktılar dikkate alınarak kullanıcı gerek görürse manuel veya otomatik olarak bu noktaları gürültü kabul ederek veri setinden silebilir.

3. Bulgular

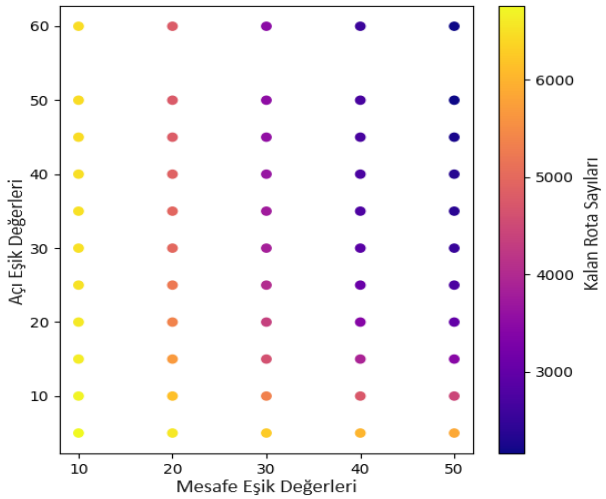
Gruplandırılmış GPS noktaları üzerinde yapılan optimizasyon işlemi, farklı açı ve mesafe eşik değerleri için tekrarlanarak en uygun eşik değerinin belirlenmesi hedeflenmiştir. Matplotlib kütüphanesi kullanılarak elde edilen çıktılar görselleştirilmiş ve analiz edilmiştir (URL-9).

Çalışma Alanı 2 için ön işlemden sonra 71645 adet nokta ve bu noktaları içeren 6797 adet rota Points_DataSet ve Routes_DataSet veri setleri halinde kaydedilmişti. Mesafe ve doğrultu açısı eşik değerlerine bağlı olarak birleşimler neticesinde elde edilen temsil noktaları ile Points_DataSet ve Routes_DataSet veri setleri güncellenmiştir. Mesafe eşik değerleri olarak [10, 20, 30, 40, 50] metre Doğrultu Açısı eşik değerleri olarak ise [5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 60] derece açıları seçilmiştir. Farklı veri setleri için farklı eşik değerleri ile program kullanılabilir haliyle Github'ta(link) erişime açık olarak paylaşılmıştır.

Şekil 12a ve Şekil 12b'de farklı mesafe ve açı eşik değerleri için Points_DataSet ve Routes_DataSet'te Optimizasyon işlemlerinden sonra kalan nokta ve rota sayılarına ait grafik yer almaktadır. Bir noktaya yakın komşu noktalar belirlenirken eşik mesafe değeri ne kadar büyük belirlenirse komşuların arandığı çember yarıçapı büyüyeceği için koşulu sağlayan komşu sayınında artması beklenir. Benzer şekilde açı eşik değeri de büyüdükçe koşulu sağlayan komşu sayısının artması beklenir. Bu durumda her iki koşulu aynı anda sağlayan nokta sayısı ne kadar çok ise bu noktaların birleşimi neticesinde elde edilmiş olan bir temsil noktası o kadar fazla noktayı temsil edecektir. Sonuç olarak, mesafe ve açık eşik değerleri artarsa, daha çok nokta, daha az nokta ile temsil edilir ve birleşimler neticesinde elde edilen temsil noktalarının sayısının giderek azalması beklenir.



Şekil 12a. Farklı açı ve mesafe eşik değerlerine bağlı olarak kalan nokta sayısı grafiği.



Şekil 12b. Farklı açı ve mesafe eşik değerlerine bağlı olarak kalan rota sayısı grafiği.

Belirlenen eşik değerlerine göre Şekil 12a ve Şekil 12b'de ki grafik incelendiğinde, yüksek açı ve mesafe eşik değerleri için nokta ve rota sayısının beklenildiği gibi diğer durumlardan daha az olduğu görülmektedir. En düşük mesafe 10 metre ve en düşük açı 5 derece için temsili nokta sayısının 11799 olduğu görülmüştür. Tüm eşik değerleri için elde edilen birleşim noktası sayısı ve rota sayısı aşağıdaki tablolarda (Tablo 3, 4)detaylı olarak sunulmuştur.

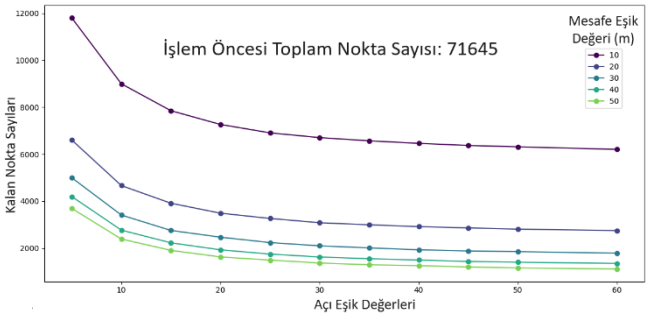
Tablo 3. Açı ve mesafe eşik değerlerine bağlı olarak kalan nokta sayısı (işlem öncesi nokta sayısı: 71645).

Angle\Distance	10 m	20 m	30 m	40 m	50 m
5⁰	11799	6600	4993	4192	3688
10⁰	8994	4660	3406	2763	2384
15⁰	7848	3909	2750	2224	1903
20⁰	7263	3485	2463	1925	1622
25⁰	6905	3261	2233	1744	1488
30⁰	6703	3078	2096	1620	1363
35⁰	6567	2991	2007	1546	1289
40⁰	6458	2913	1926	1491	1252
45⁰	6369	2858	1878	1430	1196
50⁰	6312	2806	1848	1398	1155
60⁰	6201	2746	1783	1345	1109

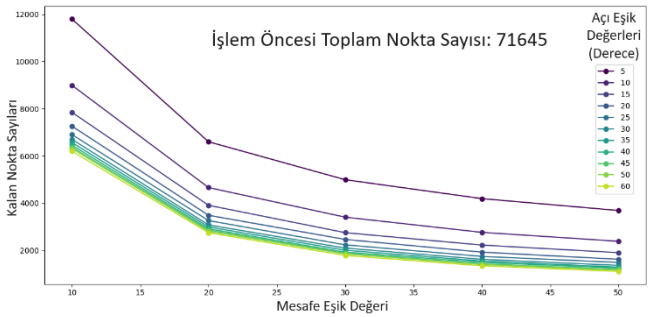
Tablo 4. Açı ve mesafe eşik değerlerine bağlı olarak kalan rota sayısı (işlem öncesi rota sayısı: 6797).

Angle\Distance	10 m	20 m	30 m	40 m	50 m
5⁰	6759	6566	6274	6017	5860
10⁰	6700	6141	5341	4744	4444
15⁰	6616	5665	4615	3896	3447
20⁰	6584	5374	4339	3418	2998
25⁰	6515	5218	4023	3081	2747
30⁰	6493	4978	3821	2892	2536
35⁰	6496	4960	3751	2767	2391
40⁰	6485	4891	3627	2732	2344
45⁰	6444	4803	3560	2691	2248
50⁰	6449	4781	3515	2695	2176
60⁰	6467	4791	3490	2565	2161

Şekil 13a ve Şekil 13b'de Nokta sayılarındaki değişimi gösteren her bir mesafe eşik değeri için açı eşik değerleri değişim eğrileri ve her bir açı eşik değeri için mesafe eşik değerleri değişim eğrileri gösterilmiştir.



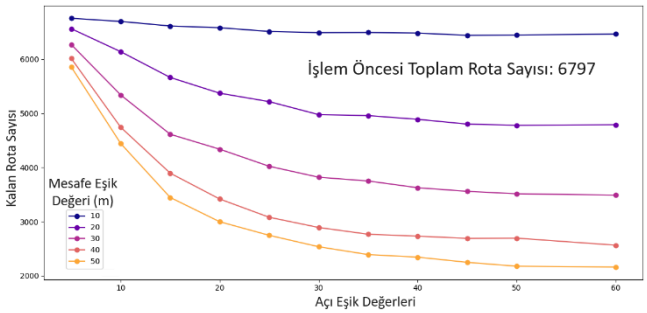
Şekil 13a. Mesafe eşik değerlerine göre nokta sayısındaki değişimi gösteren açı eşik değerleri değişim eğrileri.



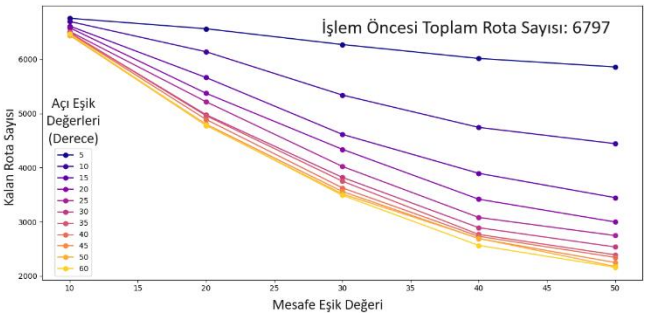
Şekil 13b. Açı eşik değerlerine göre nokta sayısındaki değişimi gösteren mesafe eşik değerleri değişim eğrileri.

Şekil 14a ve Şekil 14b'de Rota sayılarındaki değişimi gösteren her bir mesafe eşik değeri için açı eşik değerleri değişim eğrileri ve her bir açı eşik değeri için mesafe eşik değerleri değişim eğrileri gösterilmiştir.

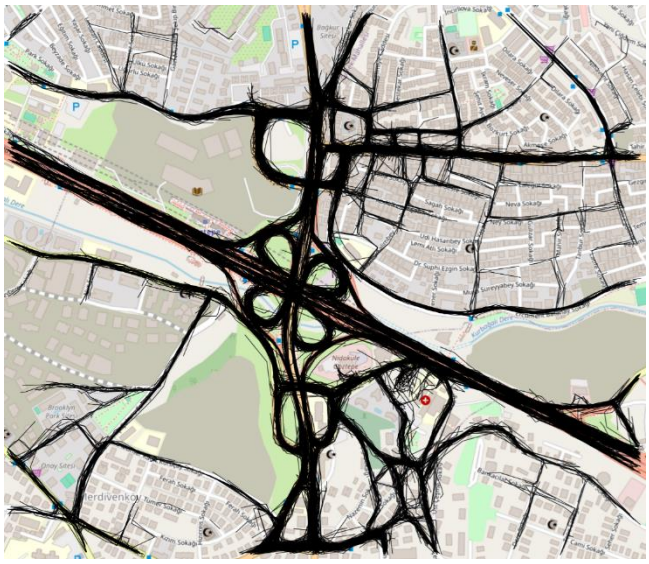
Sonuç grafikleri ve tabloların beklentilerle uyumlu olduğu görülmüştür. Ayrıca harita üzerinde görselleştirmede yapılmıştır. Şekil 15'te mesafe eşik değeri 10 ve açı eşik değeri 5 için optimize edilmiş veri setlerine ait görsel örnek olarak sunulmuştur. Diğer eşik değerleri için de verilerin harita üzerinde ayrı ayrı gösterilebilmesi geliştirilen ve Github'ta paylaşılmış olan yazılım ile mümkündür.



Şekil 14a. Rota sayılarındaki değişimi gösteren mesafe eşik değerleri için açı eşik değerleri değişim eğrileri ayrı ayrı.



Şekil 14b. Rota sayılarındaki değişimi gösteren açı eşik değerleri için mesafe eşik değerleri değişim eğrileri ayrı ayrı.



Şekil 15. Mesafe eşik değeri 10 ve açı eşik değeri 5 için optimize edilmiş veri setlerine ait görsel örneği.

4. Sonuçlar

Geliştirilen metodoloji, veri setinin boyutunu önemli ölçüde azaltırken kritik mekânsal ilişkileri koruma yeteneğini değerlendirmek için deneysel olarak test edilmiştir.

Sonuçlar, yöntemin etkinliğini ve pratik uygulanabilirliğini göstermektedir. Özellikle düşük mesafe ve açı eşik değerlerinin seçilmesi durumunda, verilerin mekânsal doğruluğunu korurken veri seti boyutunu önemli ölçüde azalttığı görülmüştür. Bu bulgu, literatürdeki diğer çalışmalarla da tutarlıdır (Ezzat ve ark., 2018; Stanojevic ve ark., 2018). Elde edilen sonuçlar ve harita çıktılarına ait görseller incelendiğinde görülmüştür ki, açı ve mesafe eşik değerleri ne kadar küçük belirlenirse temsili noktalar da temsil ettikleri noktalara dair, doğrultu ve konum bilgisini yakın derecede temsil etmektedir. Yine tersi bir durum olarak açı ve mesafe eşik değerleri büyük seçilirse bu durumda gerçeğe uymayan, birleşimde yer alan noktaları tam olarak temsil etmeyen birleşim noktaları elde edilebilir. Birbirine yakın ve aynı doğrultuda ancak birbiri ile kesişmeyen yollar büyük eşik değerleri için kesişebilir ki bu istenmeyen bir durumdur. Alt ve üst geçit şeklinde kesişmeyen yollar doğru eşik değerleri belirlenmediği durumda kesişebilir ve bu da yine istenmeyen durumlara başka bir örnektir. Farklı eşik değerleri ile harita çıktılarının da üretiliyor olması kıyaslama imkânı sunmaktadır. Bu sayede gözlem yolu ile en uygun eşik değeri belirlenebilir ve veri optimize edilebilir.

Ham veri setinde sadece latitude ve longitude değerleri mevcuttur. Elevation değeri ham veri setinde bulunmamaktadır. Benzer şekilde taksilerden toplanan GPS iz verilerinde genellikle yükseklik bilgisini içeren elevation değeri bulunmamaktadır. Bundan dolayı GPS iz verileri kullanılarak yol orta hattı çıkarımı ile ilgili yapılan çalışmalarda alt ve üst geçitlerin hatalı kesişimleri, yan yolların hatalı birleşimleri gibi durumlar gözlemlenmektir. Üretilen yol ağında haliyle birçok topolojik problem ortaya çıkabilmektedir.

Badran ve arkadaşlarının (2023) yol orta hattı çıkarımı konusu ile ilgili literatür incelemesine yönelik yaptıkları çalışmada, yapılan çalışmaların yol orta hattının yönlü olarak çıkarımı ve keşişim bölgelerinin tespiti ile sınırlı olduğu, çalışmalarda elde edilen yol ağı ile gerçek yol ağı arasındaki benzerliği değerlendirmek için doğruluk göstergesi olarak Denklem 12'de verilen, Biagioni ve Eriksson (2012) tarafından tanıtılan ve en yaygın değerlendirme yöntemi olan F Skoru formülünün kullanıldığı belirtilmiştir. Ek olarak, incelenen yöntemler arasında yüksek F skoru değerine sahip çalışmaların olduğu ve gelecekte yapılacak çalışmalar için model kalitesini iyileştirebileceği, model oluşturmak için ihtiyaç duyulan kaynakları azaltırken modellerin güncellenme sıklığını arttırabileceği için muazzam bir değere sahip olduğuna değinilmiştir (Badran ve ark., 2023; Biagioni ve Eriksson, 2012).

$$\text{Hassasiyet} = \frac{Doğru\ Çıkarılanlar}{Gerçek\ Noktaların\ Tamamı} \quad (10)$$

$$Geri\ Çağırma = \frac{Doğru\ Çıkarılanlar}{Hesaplanan\ Noktaların\ Tamamı} \quad (11)$$

$$F\ skoru = 2 * \frac{(\text{Hassasiyet} * \text{Geri Çağırma})}{(\text{Hassasiyet} + \text{Geri Çağırma})} \quad (12)$$

Gelecekte yapılması planlanan çalışmalar arasında GPS izlerine dayalı otomatik veya yarı otomatik yöntemlerle yol orta hattının çıkarımı konusu da yer almaktadır. Bu bağlamda, bu çalışmada geliştirilen yazılım, birçok eşik değerini birlikte mukayese imkânı vermesinden dolayı veri seti için ideal eşik değerlerinin belirlenmesi için önemlidir. Ayrıca temsil noktalarının coğrafi konumlarının ve topolojik ilişkilerinin; temsil ettikleri noktaların özelliklerini en doğru şekilde yansıtacak uygun eşik değerlerinin belirlenmesi sonrasında nokta ve rota sayısının anlamlı düzeyde azalması sonucunda gürültüye sebep olan hatalı kayıtların görünürlüğünü artmıştır. Bu durum otomatik ya da yarı otomatik yöntemlerle veri setine müdahale edilebilmesini ve gürültülü rotaların silinebilmesini kolaylaştırmaktadır.

Bilgilendirme/Teşekkür

Bu çalışma, Karabük Üniversitesi Bilimsel Araştırma Projeleri (BAP) birimi tarafından desteklenmiştir. Proje Numarası: KBÜBAP-17-DR-437. Desteklerinden dolayı minnettarız.

Araştırmacıların katkı oranı

Emrullah Demiral: Literatür taraması, Yazılım geliştirme, Makale yazımı; **İsmail Rakıp Karaş:** Danışman, Düzenleme

Çatışma Beyanı

Herhangi bir çıkar çatışması bulunmamaktadır.

Kaynakça

Ahmed, M., Fasy, B. T., Hickmann, K. S., & Wenk, C. (2013). Path-Based Distance For Street Map Comparison. https://doi.org/10.1145/2729977

Atiz, Ö. F., Konukseven, C., Öğütcü, S., & Alçay, S. (2022). Comparative Analysis of The Performance of Multi-GNSS RTK: A Case Study in Turkey. International Journal of Engineering And Geosciences, 7(1), 67–80. https://doi.org/10.26833/ijeg.878236

Badran, A., El-Geneidy, A., Miranda-Moreno, L. (2023). A Review of Techniques to Extract Road Network Features from Global Positioning System Data for Transport Modelling. Transport Reviews, 44(1), 69–84. doi.org/10.1080/01441647.2023.2229521

Biagioni, J., & Eriksson, J. (2012). Inferring road maps from global positioning system traces survey and comparative evaluation. Transportation Research Record, 2291, 61–71.
https://doi.org/10. 3141/2291-08

Chen, B., Ding, C., Ren, W., & Xu, G. (2020). Extended Classification Course Improves Road Intersection Detection from Low-Frequency GPS Trajectory

Data. ISPRS International Journal of Geo-Information, 9(3), 181. Doi:10.3390/İjgi9030181

Dal Poz, A. P., Martins, E. F. O., & Zanin, R. B. (2022). Road Network Extraction Using Gps Trajectories Based on Morphological and Skeletonization Algorithms. The International Archives of The Photogrammetry, Remote Sensing and Spatial Information Sciences, XLIII-B4-2022, 239–245. Doi:10.5194/Isprs-Archives-XLIII-B4-2022-239-2022

E., Demiral. (2024). Project files and codes at Github repository. Retrieved 18 June 2024, From https://Github.Com/Edemiral78/3_Mergepointsfromoutmost.Git

Gao, S., Li, M., Rao, J., Mai, G., Prestby, T., Marks, J., & Hu, Y. (2021). Automatic Urban Road Network Extraction from Massive GPS Trajectories of Taxis. In Handbook of Big Geospatial Data (Pp. 261–283). Cham: Springer International Publishing. https://doi.org/10.1145/347090.347119

Guo, T., Iwamura, K., & Koga, M. (2007). Towards High Accuracy Road Maps Generation from Massive GPS Traces Data. In 2007 IEEE International Geoscience And Remote Sensing Symposium (Pp. 667–670). IEEE. Doi:10.1109/IGARSS.2007.4422884

Karney, C.F.F. Algorithms for geodesics. J Geod 87, 43–55 (2013). https://doi.org/10.1007/s00190-012-0578-z

Koca, B., & Ceylan, A. (2018). Uydu Konum Belirleme Sistemlerindeki (GNSS) Güncel Durum ve Son Gelişmeler. Geomatik, 63–73. https://doi.org/10.29128/geomatik.348331

Krumm, J., Davies, N., & Narayanaswami, C. (2008). User-Generated Content. IEEE Pervasive Computing, 7(4), 10–11. Doi:10.1109/MPRV.2008.85

Leichter, A., & Werner, M. (2019). Estimating Road Segments Using Natural Point Correspondences of GPS Trajectories. Applied Sciences, 9(20), 4255. https://doi.org/10.3390/app9204255

Li, H., Kulik, L., & Ramamohanarao, K. (2016). Automatic Generation and Validation of Road Maps from GPS Trajectory Data Sets. In Proceedings of the 25th ACM International on Conference on Information and Knowledge Management (pp. 1523–1532). New York, NY, USA: ACM. https://doi.org/10.1145/2983323.2983797

M. Ezzat, M. Sakr, R. Elgohary, and M. E. Khalifa, "Building Road Segments and Detecting Turns from GPS Tracks," J Comput Sci, vol. 29, pp. 81–93, Nov. 2018, https://doi.org/10.1016/j.jocs.2018.09.011

Ni, Z., Xie, L., Xie, T., Shi, B., & Zheng, Y. (2018). Incremental Road Network Generation Based on Vehicle Trajectories. ISPRS International Journal of Geo-Information, 7(10), 382. https://doi.org/10.3390/ijgi7100382

Pırtı, A., & Yazıcı, D. (2022). İnternet Tabanlı GNSS Yazılımlarının Doğruluk Açısından Değerlendirilmesi. Geomatik, 7(2), 88–105. https://doi.org/10.29128/geomatik.882843

R. Stanojevic, S. Abbar, S. Thirumuruganathan, S. Chawla, F. Filali, and A. Aleimat, "Robust Road Map Inference through Network Alignment of Trajectories," in Proceedings of the 2018 SIAM International Conference on Data Mining, Philadelphia, PA: Society for Industrial and Applied Mathematics, 2018, pp. 135–143. https://doi.org/10.1137/1.9781611975321.15

Qiu, J., & Wang, R. (2016). Automatic Extraction of Road Networks from GPS Traces. Photogrammetric Engineering & Remote Sensing, 82(8), 593–604. https://doi.org/10.14358/PERS.82.8.593

Tang, L., Ren, C., Liu, Z., & Li, Q. (2017). A Road Map Refinement Method Using Delaunay Triangulation for Big Trace Data. ISPRS International Journal Of Geo-Information, 6(2), 45. https://doi.org/10.3390/ijgi6020045

URL-1. (2024). A list of public GPS Trajectories datasets. Erişim: 29 Temmuz 2024, https://vis.cs.kent.edu/TrajAnalytics/casestudy.php

URL-2. (2024). Athens, Berlin and Chicago GPS trajectories datasets. Erişim: 29 Temmuz 2024, http://mapconstruction.org/

URL-3. (2024). Grafik ve şekillerin yüksek çözünürlüklü versiyonları. Erişim: 29 Temmuz 2024, https://github.com/edemiral78/3_MergePointsFromOutmost/tree/main/G%C3%B6rseller

URL-4. (2024). OpenStreetMap. Erişim: 29 Temmuz 2024, https://www.openstreetmap.org/copyright

URL-5. (2024). Geopy Dokümantasyonu. Erişim: 29 Temmuz 2024, https://geopy.readthedocs.io/en/stable/

URL-6. (2024). Haversine Mesafesi Python Modülü. Erişim: 29 Temmuz 2024, https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.haversine_distances.html

URL-7. (2024). GeographicLib Dokümantasyonu. Erişim: 29 Temmuz 2024, https://geographiclib.sourceforge.io/

URL-8. (2024). NumPy Dokümantasyonu. Erişim: 29 Temmuz 2024, https://numpy.org/doc/stable/

URL-9. (2024). Matplotlib Dokümantasyonu. Erişim: 29 Temmuz 2024, https://matplotlib.org/stable/contents.html

Yang, J., Mariescu-Istodor, R., & Fränti, P. (2019). Three Rapid Methods for Averaging GPS Segments. Applied Sciences, 9(22), 4899. https://doi.org/10.3390/app9224899

Zheng, Z., Rasouli, S., & Timmermans, H. (2014). Evaluating the Accuracy of GPS-based Taxi Trajectory Records. Procedia Environmental Sciences, 22, 186–198. https://doi.org/10.1016/J.Proenv.2014.11.019