Bölüm 5

GEZGİN SATICI PROBLEMİNİN ÇÖZÜMÜ İÇİN GENETİK ALGORİTMA TABANLI BİR YAZILIM ÖNERİSİ[1]

Kadir KIRDA[2]

1. GİRİŞ

Tarih boyunca insanlar, hep daha iyi bir hayat sürebilmek için nesilden nesile gelişmeye devam etmiş, yaşam standartlarını yükseltecek sistemler geliştirmiş ve buluşlar yapmışlardır. Doğal yaşam, insanlık tarihine yön veren icatlar da dâhil olmak üzere pek çok buluş için zengin bir esin kaynağı olmuştur. Bunun örneklerinden biri; kuşların uçuş özelliklerinden ilham alınarak tasarlanan uçaklardır. Günler süren yolculukları birkaç saate indiren uçuşlar, dünyanın en uzak noktalarının bile çok daha kolay ulaşılabilir olmasını sağlamıştır. Bunu takiben, küresel ticaretin ivme kazanması, küreselleşmenin hızlanması gibi çeşitli sosyo-ekonomik etkiler ortaya çıkmıştır. Başka bir örnek, beynin çalışma sisteminden yola çıkılarak geliştirilen yapay sinir ağlarıdır. Yapay sinir ağlarındaki teorik ilerlemeler, özellikle bilişim teknolojilerinde son dönemdeki hızlı gelişimle birleşerek, yapay zekâ uygulamalarının günümüzde önemli başarılara imza atmasını mümkün kılmıştır. Genetik algoritma da böyle bir bakış açısının ürünüdür. Temeli Darvin'in evrim teorisine dayanan genetik algoritma, günümüzde biyoenformatik, mühendislik, ekonomi gibi pek çok alanda kullanılmaktadır. Genetik algoritma, bireyler arası gen aktarımının nesilden nesile evrilen bir süreçte nasıl işlediğine dair sezgisel bir algoritma türüdür.

Genetik algoritmanın uygulama potansiyelinin başarıyla sergilendiği disiplinlerden biri, optimizasyon problemlerinden gezgin satıcı problemidir. Gezgin satıcı problemi, belirli şehirler ya da konumlar arasında, her birine sadece bir kere uğrayarak en düşük maliyetli ya da en kısa mesafeli rotayı belirlemeye

[1] Bu çalışma Doç. Dr. Özlem DOĞAN danışmanlığında 25/06/2013 tarihinde tamamladığımız "Evsel İlaç Atıklarının Toplanması Projesindeki Tersine Lojistik Sürecinin Modellenmesi İçin Genetik Algoritmaların Kullanılması" başlıklı doktora tezi esas alınarak hazırlanmıştır.

[2] Dr. Öğr. Üyesi, Artvin Çoruh Üniversitesi, Hopa İktisadi ve İdari Bilimler Fakültesi, İşletme Bölümü Sayısal Yöntemler AD, kadirkirda@artvin.edu.tr, ORCID iD: 0000-0003-0779-0175

odaklanmaktadır. Bu çalışmada, ilk olarak genetik algoritmanın temel özellikleri, bileşenleri ve işleyiş mekanizması detaylı bir şekilde incelenmektedir. İkinci bölümde, gezgin satıcı problemi çerçevesinde, gerçek dünya senaryosuna dayanan bir örnek üzerinde durulmaktadır. Problemin çözümünde kullanılmak üzere Python diliyle bir yazılım geliştirilmiştir. Söz konusu yazılım, araştırmacılar için açık kaynak kodlu olarak sunulmuştur. Bu çalışma, sadece gezgin satıcı probleminin genetik algoritma ile nasıl ele alınabileceğini göstermekle kalmayıp, aynı zamanda geliştirilen yazılım aracılığıyla bilimsel topluluğa katkıda bulunmayı hedeflemektedir.

2. GENETİK ALGORİTMALAR

Darwin tarafından ortaya konan doğal seçilim ilkesi, biyolojik organizmaların belirli bir görevi gerçekleştirebilmek için en uygun olanın hayatta kalma olasılığının daha yüksek olduğu bir evrim sürecini kavramsallaştırmaktadır. Bu teorinin, albatros kuşlarının aerodinamik kanat yapıları veya köpek balıkları ile yunuslar arasındaki morfolojik benzerlikler gibi somut örnekleri mevcuttur. Doğal ortamında çok etkili sonuçları olan bu sürecin ilerleyen zamanda optimizasyon problemlerinde de uygulanması fikri ortaya çıkmıştır (Sivanandam ve Deepa, 2008, s. 15).

Doğal seçilim ve genetik sürecinin benzetimi olan genetik algoritma, öncelikle kromozomlardan oluşan ve popülasyon adı verilen bir çözüm kümesi ile başlar. Her bir kromozom bir çözüm alternatifini temsil eder. Popülasyon, ardışık iterasyonlar aracılığıyla evrilmektedir (Huang vd., 2005, s. 277).

Geleneksel optimizasyon yöntemleriyle karşılaştırıldığında, genetik algoritmaların dört belirgin özelliği bulunmaktadır (Kahvecioğlu, 2004, s. 348):

1. Genetik algoritmalar, parametre kodlamalarıyla işlem yapar.
2. Optimizasyonun odaklandığı araştırma, bir noktalar kümesi üzerinden gerçekleştirilir.
3. Genetik algoritmada, sadece bir amaç fonksiyonu bilgisiyle araştırma yapılır. Fazladan bilgiye ihtiyaç duyulmaz.
4. Genetik algoritmalarda, olasılık tabanlı geçiş kuralları uygulanarak araştırma uzayı içerisinde ilerlenir.

Genetik algoritmaların uygulanmasında izlenen ana prosedür şu adımları içermektedir (Mitchell ve Taylor, 1999, s. 594):

1. Rastgele jenerasyon yöntemiyle **n** adet kromozomdan oluşan bir başlangıç popülasyonu oluşturulur.

2. Belirlenmiş bir uygunluk fonksiyonu ile popülasyondaki her bir kromozomun uygunluk değeri belirlenir.
3. Aşağıdaki adımlar aracılığıyla yeni bir nesil popülasyon hazırlanır:
   a) Mevcut kromozomlar içinden tercih edilen seçilim yöntemleri ile kromozom çiftleri seçilir.
   b) Seçilen çaprazlama yöntemiyle kromozom çiftlerinden yeni kromozomlar, başka bir ifadeyle yeni çözümler oluşturulur.
   c) Bir olasılık değerine göre rastgele seçilen kromozomlarda mutasyon işlemi gerçekleştirilir. Bu kromozomlar, yeni nesillerde yer alırlar.
4. Yeni oluşturulmuş olan popülasyon, bir önceki popülasyonun yerini alır.
5. Sürecin ikinci adımına geri dönülür.

Bu prosedür, bazı farklılıklar olsa da genetik algoritma uygulamalarının temelini oluşturmaktadır. Bu süreçte belirlenmesi gereken bazı ayrıntılar bulunmaktadır. Algoritmanın etkinliği, kromozomların kodlanma yöntemi, popülasyon büyüklüğü, seçilim, çaprazlama ve mutasyon oranları gibi parametrelerin belirlenmesiyle yakından ilişkilidir (Mitchell ve Taylor, 1999, s. 595).

2.1. Genetik Algoritmaların Bileşenleri

Genetik algoritmaların etkili bir şekilde uygulanabilmesi adına, çözüm kümesinin temel karakteristiklerinin tanımlanması esastır. İlgili bileşenler aşağıda kavramsal bir çerçevede sunulmaktadır.

2.1.1. Gen ve Kromozom

Gen, genetik algoritmaların çözüm mimarisini oluşturan temel bileşendir. Bir gen dizisinden bir kromozom meydana gelir (Sivanandam ve Deepa, 2008, s. 41). Başka bir deyişle, genlerin belli bir düzende sıralanmasıyla oluşan gen setine "kromozom" adı verilmektedir. Kromozom, çözülmek istenen problemin karar değişkenlerinin bir arada bulunduğu bir dizidir (Şen, 2004, s. 25).

Her bir kromozom, ilgili problem için çözüm seçeneklerinden birini temsil eder. Sürece başlarken, rastgele oluşturulan bir dizi kromozom ile bir başlangıç çözüm havuzu oluşturulur. Bu havuz içerisinde çaprazlama ve mutasyon operatörleriyle yeni kromozomlar (çözümler) geliştirilir. Yeni üretilen kromozomların başarısı, ebeveyn kromozomlarına göre değişkenlik gösterebilir (Chan vd., 2005, s. 347). Daha uygun çözümler popülasyon içerisinde muhafaza edilirken, hedeflenen performans ölçütüne göre, popülasyonun sürekli olarak iyileşmesi amaçlanmaktadır. Optimizasyon süreci, sonlandırma kriterlerinden

biri karşılandığında sona erer.

Kromozomların genlerden oluşumu, belirli bir sistematiğe göre gerçekleştirilir. Bu sistematiğe kodlama adı verilir. Kodlama türü, araştırma konusu, çözümün değerlendirilmesi ve genetik operatörlerin uygulanabilirliği göz önünde bulundurularak belirlenir. Kodlama teknikleri çok çeşitlilik gösterse de ikili kodlama (0 ve 1 kullanılır) ve permütasyon kodlama (sıralamaya dayalı), kullanım kolaylıkları nedeniyle yaygın olarak başvurulan yöntemler arasındadır.

2.1.2. Topluluk Büyüklüğü

Genetik algoritma problemlerinde, her bir kromozomun belirli bir çözümü temsil ettiği bir popülasyon bulunmaktadır. Bu popülasyonda yer alan kromozomların toplam sayısı "topluluk büyüklüğü" olarak tanımlanır. İlk aşamada, kromozomlar başarı dereceleri gözetilmeksizin rastgele bir biçimde meydana getirilir. Genetik algoritma süreci boyunca topluluk büyüklüğünde herhangi bir değişim gerçekleştirilmez ve araştırmacının belirlemiş olduğu bu büyüklük süreç tamamlanana dek sabit kalır.

Topluluk büyüklüğünün uygun düzeyde olmaması, problemin çözümünü iki yönde etkiler. Birincisi, topluluk büyüklüğü için gereğinden düşük bir değer seçildiğinde başarılı bireylerin popülasyondan çıkarılması riski ortaya çıkar. İkincisi, topluluk büyüklüğü değeri gereğinden büyük olduğunda ise hesaplamaların süresi uzayabilir ve başarılı çözüme ulaşmak zorlaşabilir. Bu sebeple, topluluk büyüklüğünün optimal bir seviyede tutulması esastır.

2.1.3. Uygunluk Değeri

Genetik algoritma metodolojisinde kritik öneme sahip temel bileşenlerden biri, tasarlanan çözümlerin performansını gösteren uygunluk değeridir. Çözüm havuzunda bulunan bireylerin yani kromozomların her birinin bir uygunluk değeri bulunur. Uygunluk değerlerini hesaplayabilmek için bir uygunluk fonksiyonu tanımlanır.

Uygunluk değeri, bireylerin çaprazlama için seçilmesi, çözümden çıkacak bireylerin belirlenmesi ve sonlandırma koşulunun kontrol edilmesi gibi işlemlerde kullanılır. Söz gelimi, her bir evrim adımında çaprazlama sonucunda popülasyonun hacmi artmaktadır. Bu durumda topluluk büyüklüğünün korunması için yeni eklenen bireylerin sayısı kadar birey popülasyondan çıkarılmalıdır. Uygunluk değeri, bireylerin performanslarının birbiriyle karşılaştırılmasını ve sıralanmasını mümkün kılar. Genetik algoritmanın temel prensiplerine göre mevcut iyi çözümlerden daha üstün çözümler elde etmek amacıyla, performansı düşük olan

çözümler sistemden elimine edilmelidir. Bu yaklaşım ile uygunluk değeri düşük olan bireyler popülasyondan çıkarılarak, popülasyon büyüklüğü istikrarlı bir seviyede tutulur.

2.2. Genetik Operatörler

Genetik algoritma bileşenleri belirlendikten sonra, sürecin geliştirilmesinde hangi yöntem ve parametrelerin kullanılacağı belirlenmelidir. Bu tasarımda genetik operatörler kritik bir rol oynar. Böylelikle genetik algoritma süreci tasarlanır ve çalıştırılır. Genetik operatörlerin temel özellikleri aşağıda detaylandırılmıştır.

2.2.1. Seçilim

Genetik algoritmada çözümlerin nesilden nesile gelişmesi amaçlanır. Bu kapsamda popülasyondaki bireylerden ebeveynler seçilir ve bu ebeveynlerden yeni bireyler oluşturulur. Seçilim, yeni bireylerin oluşturulabilmesi için eşleşecek ebeveynlerin tespit edilmesi sürecidir. Seçilimin amacı, daha üstün uygunluk değerlerine sahip bireylerin popülasyonda dominant hale gelmesini sağlamaktır (Sivanandam ve Deepa, 2008, s. 46).

Teoriye göre, daha iyi bireylerin yaşamlarını sürdürmesi gerekir ve bu bireylerden yeni bireyler türetilmelidir. Bu sebeple seçilim yöntemlerinin tümünde daha iyi uygunluk değerine sahip bireylerin seçilme ihtimali daha yüksektir. En yaygın kullanılan seçilim yöntemleri; rulet seçilimi, turnuva seçilimi ve sıralı seçilim yöntemleridir (Cevre vd., 2007).

Gezgin satıcı problemine özgü bir seçilim yöntemi bulunmamakla birlikte, genetik algoritmalarda genel olarak kullanılan seçilim yöntemlerinden biri tercih edilebilir. Bu çalışmadaki uygulamada sıralı seçilim yöntemi kullanılmıştır.

2.2.2. Çaprazlama

Çaprazlama, eşleştirme için oluşturulan havuzda yer alan bireyler arasındaki gen değişimi işlemidir. Sürecin amacı, bir kromozomdaki zayıf genlerin diğer kromozomdan alınan güçlü genlerle değiştirilmesidir. Bu süreç, daha iyi kromozomların ortaya çıkmasını sağlayarak çözümün gelişmesini sağlar (Chan vd., 2005, s. 49).

Eşleştirme havuzundaki kromozomlar, çiftler halinde eşleştirilir. Çaprazlama sürecinde genler, belirlenen bir prosedüre göre kromozom çiftleri arasında transfer edilir. Bu prosedür, problemin yapısına uygun olarak seçilir. Çaprazlama, genel olarak tek noktalı ve çok noktalı olmak üzere iki şekilde uygulanır (Chan

vd., 2005, s. 49).

Algoritma uygulanmadan önce araştırmacı tarafından belirlenen çaprazlama oranı, çaprazlama yapılacak bireylerin popülasyona oranını ifade eder. Bu oran, uygulamanın başlangıcından sonuna kadar sabit olabileceği gibi değişken oran da tercih edilebilir. Yüksek çaprazlama oranı, yeni bireylerin daha hızlı oluşmasını sağlarken, iyi çözümlerin popülasyondan çıkarılması riskini taşır. Çaprazlama oranı düşük olduğunda ise her iterasyonda az sayıda birey türetileceği için araştırma sürecinin daha uzun sürmesine yol açar (Kahvecioğlu, 2004, s. 349).

Bir çaprazlama işlemi, temel olarak iki aşamadan oluşur. Birinci aşamada, çaprazlama yapılacak bireyler rastgele olarak seçilir. İkinci aşamada ise rastgele türetilen değerler kullanılarak çaprazlama işlemi uygulanır. Başarım düzeylerinin yüksek olması için uygun bir çaprazlama oranının seçilmesi gerekir (Kahvecioğlu, 2004, s. 349).

2.2.3. Mutasyon

Genetik algoritma sürecinin rastgelelik özelliğinden ötürü çözümler kimi zaman global optimum yerine yerel optimuma takılırlar. Bundan kurtulabilmek için mutasyon işlemi uygulanır. Mutasyon, başka bir yolla bulunamayacak çözümlerin elde edilmesini sağlayan, tamamıyla rastgele işleyen bir yöntemdir (Karova vd., 2005, s. 3).

Mutasyon işlemi için topluluktaki bazı bireyler rastgele olarak seçilirler ve mutasyon noktası yine aynı şekilde rastgele olarak belirlenir. Daha sonra dizinin ilgili konumundaki değer değiştirilir (Karova vd., 2005, s. 1). Mutasyonda, belirlenmiş olan mutasyon olasılığına dayalı olarak dizinin değerinde değişiklik yapılır (Chen ve Chen, 1997, s. 1323). Aşağıda ikili kodlanmış bir kromozomun mutasyon örneği verilmiştir. İlk satırda mevcut kromozomun gen dizilimi görülmektedir. Koyu biçimlendirilmiş 1 rakamı mutasyon işlemi için seçilen değerdir. Sonraki satırda ise 1 değeri 0 yapılarak mutasyon işlemi tamamlanmaktadır.

- 1 0 0 1 1 0 1 **1** 0 1 0 1 (Mutasyon öncesi)
- 1 0 0 1 1 0 1 **0** 0 1 0 1 (Mutasyon sonrası)

Mutasyon işleminin yapılması için belirlenen mutasyon olasılığı genellikle 0,01 gibi çok düşük bir değerdir. Mutasyon olasılığının yüksek olması, kromozomların haddinden fazla değiştirilmesine neden olabilir ve bu durum optimuma yaklaşmayı zorlaştırabilir.

2.3. Sonlandırma Koşulu

Genetik algoritma, bir sezgisel arama algoritmasıdır ve global optimumu

her zaman garanti etmez. Ancak optimuma yakın bir sonucu garanti eder. Bu belirsizlik durumundan ötürü genetik algoritmada sürecin sonlandırılması için en iyi ve tek bir kural bulunmamaktadır. Genetik algoritma, yapısı gereği evrimsel bir süreçte durmaksızın işletilmektedir. Dolayısıyla bir sonlandırma stratejisi belirlenmelidir.

Genetik algoritma sürecinin durması için kesin bir bilgi olmamakla birlikte, genel kabul görmüş çeşitli sonlandırma stratejileri vardır. Bu stratejiler maddeler halinde şöyle sıralanabilir (Haupt ve Haupt, 2004, s. 109):

1. Doğru cevaba ulaşılması durumunda,
2. Belirlenen sayıdaki iterasyon boyunca herhangi bir gelişme görülmediğinde,
3. Popülasyonun ortalaması ya da standart sapması istenen düzeye ulaştığında,
4. Belirlenmiş nesil sayısına ulaşıldığında genetik algoritma süreci sonlandırılabilir.

3. GENETİK ALGORİTMA TABANLI YAZILIM

Bu çalışmada, genetik algoritma tabanlı geliştirilen bir yazılım üzerinde bir problem çözümü ele alınmaktadır. Odaklanılan problemde İzmir ilinde, hane halkı tarafından eczanelere getirilen süresi geçmiş atık statüsündeki ilaçların uygun şekilde bertaraf edilmesi amacıyla toplanması söz konusudur. Görevlendirilen araç, listedeki tüm eczaneleri dolaşarak atık ilaçları toplamalıdır. Amaç, en az mesafeyi kat ederek tüm konumların ziyaret edilmesidir (Kırda, 2013). Projeyle ilgili daha kapsamlı bilgi ve uygulamada kullanılan veriler için YÖK'ün tez arşivinden ilgili yayına ulaşılabilir.

Optimizasyonun başarısı, uygunluk değeriyle ölçülmektedir. Buradaki uygunluk değeri toplam kat edilen mesafedir. Tatmin edici bir uygunluk değerine ulaşılması, çözümün başarısını gösterir.

3.1. Gezgin Satıcı Problemi

Uygulamanın problemi Gezgin Satıcı Problemi (GSP) olarak sınıflandırılmaktadır. GSP, her bir coğrafi noktanın sadece bir kere ziyaret edilmesi şartıyla, tüm konumların en düşük maliyetle veya en kısa yoldan dolaşılması amacını taşıyan bir optimizasyon problemidir. Polinom zaman içinde tam olarak çözülemeyen zorlukta olan bu tip problemleri çözebilmek için özellikle büyük ölçekli problemlerde sezgisel yaklaşımlar yaygın olarak kullanılmaktadır (Potvin, 1996, s. 339; Sansarcı vd., 2009, s. 22).

Genetik algoritmanın GSP'de uygulanmasında üzerinde durulması gereken birkaç konu vardır. Bunlardan birincisi kromozomların kodlanmasıdır. GSP'nin

temel yapısı sıralamaya dayalı olduğundan, literatürde genellikle permütasyon kodlamanın tercih edildiği görülmektedir. Permütasyon kodlama yönteminde her bir kromozom farklı bir sıralamayı temsil eder. Sıralamadaki her bir rakam, bir konumun sıra numarasıdır. Daha başarılı sıralama daha kısa mesafe anlamına gelir. Kromozomlardan oluşan küme ise popülasyon olarak isimlendirilir.

Dikkat edilmesi gereken bir diğer konu, uygunluk fonksiyonudur. Uygunluk, her bir çözümün değerini gösterir. Bu değer toplam kat edilen mesafe olabileceği gibi tur esnasında ortaya çıkan maliyetlere dayalı olarak da belirlenebilir.

Seçilim için GSP'ye özel bir algoritma bulunmamaktadır. Genetik algoritma için kullanılan yöntemlerden biri tercih edilebilir. Bu çalışmada sıra numarasının doğrusal olarak önem kazandığı sıralı seçilim yöntemi kullanılmıştır. Seçilme olasılığı formülü kullanılarak belirlenir. Sıra numarası ile gösterilir ve uygunluk değeri sırasına göre sondan başa doğru numaralandırılır. Popülasyon büyüklüğü ile gösterilir. Popülasyon büyüklüğü 20 olan bir problemde sondan 5'inci kromozom için seçilme şansı şeklinde hesaplanır. Bu hesaba göre, en iyi kromozomun seçilme şansı, en kötü kromozoma göre **n** kat fazladır. Bu yöntemle tüm yeni kromozomlar için seçilim işlemi tamamlanır.

3.2. İşleyiş ve Altyapı

Bu başlıkta, uygulamaya konu olan problem çözümü için planlanan işleyişin ana hatları ve bu plana uygun olarak geliştirilen yazılımın sahip olduğu özelliklere yer verilmektedir.

3.2.1. Uygulamanın Genel İşleyişi

Uygulama süreci ve genel işleyiş aşağıdaki gibi maddeler halinde özetlenebilir:

1. Veri derleme: Tüm coğrafi noktaların enlem ve boylam bilgileri harita üzerinden tespit edilerek Microsoft Excel dosyasına eklenir.
2. Veri girişi: Excel ile oluşturulan xlsx uzantılı veri dosyası, geliştirilen yazılıma yüklenir.
3. Parametrelerin girilmesi: Genetik algoritma sürecinde gerekli parametreler sürecin en başında arayüz üzerinde ilgili kutucuklara girilir.
4. Sürecin başlatılması: Arayüzdeki başlatma butonu tıklanarak evrim süreci başlatılır.
5. Görselleştirme: Süreç boyunca, uygunluk değerinde bir gelişme olduğunda toplam mesafe bilgisi ve harita üzerindeki rota otomatik olarak güncellenir.
6. Sürecin sonlanması: Maksimum nesile ulaşıldığında süreç kendiliğinden durur. Alternatif olarak kullanıcı müdahalesi ile süreç duraklatılabilir. Eğer maksimum nesile ulaşılmadıysa sürecin kaldığı yerden devam ettirilmesi

mümkündür.

7. Kaydetme: En iyi uygunluk değerine ait rota, Excel dosya formatında dışarıya aktarılabilir.

3.2.2. Yazılımın Teknik Özellikleri

Bahsi geçen uygulama sürecinin gerçekleştirilmesi amacıyla bir masaüstü yazılım geliştirilmiştir. Yazılımın kullanıcı arayüzü, herhangi bir kodlama bilgisi olmayan kişilerin de genetik algoritma sürecini yürütebilmesine olanak sağlamaktadır. Python programlama dili kullanılarak geliştirilen yazılımda arayüz tasarımı için PyQt, veri seti işlemleri için Pandas, asenkron harita güncellemesi için Folium, hesaplamalar için Numpy ve harita mesafesi için Haversine isimli Python kütüphaneleri kullanılmıştır. Yazılımın temel özellikleri şöyle sıralanabilir:

- Excel dosya formatında veri girişi
- Harita üzerinde dinamik veri görselleştirme
- Parametre giriş paneli
- Elle sonlandırma/sürdürme özelliği
- En başarılı rotanın Excel formatında kaydedilmesi

Geliştirilen yazılım https://github.com/kadirkirda/route-finder proje sayfasında açık kaynaklı olarak kullanıma sunulmuştur.

3.3. Uygulamanın Bileşenleri

Genetik algoritma sürecinde temel nokta uygulamanın konusudur. Buna göre veri kaynağı, uygulama mimarisi ve fonksiyonlar değişkenlik gösterir. Bu çalışmada konumların sıralamasına ilişkin bir optimizasyon amacı bulunduğundan, veri kaynağı olarak coğrafi konumlar kullanılmakta, diğer bütün bileşenler ve fonksiyonlar ise buna bağlı olarak belirlenmektedir.

3.3.1. Veri Kaynağı: Coğrafi Koordinatlar

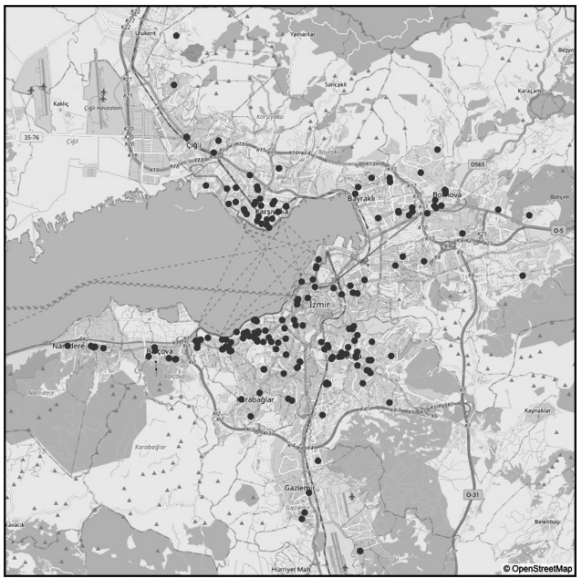
Uygulamadaki çözümlerin başarı kriteri, katedilen toplam mesafenin optimizasyonudur. Dolayısıyla mesafelerin hesaplanabilmesi için öncelikle tüm konumların koordinatının bilinmesi gerekmektedir. Coğrafi koordinatlar "Google Maps" web uygulaması kullanılarak belirlenmiştir. Her bir konum için aşağıdaki süreç takip edilerek veri kaynağı oluşturulur:

1. Adres, arama kutusuna yazılarak ve/veya tarama yöntemiyle harita üzerinde bulunur.
2. İlgili coğrafi noktaya farenin sağ tuşuyla tıklanır. Açılan menüde enlem ve boylam değerlerinin yazılı olduğu bir satır görülmektedir. Bu satıra

tıklandığında adres panoya kopyalanır.

3. Bu adres, veri setinin bulunduğu dosyaya panodan yapıştırma yöntemiyle eklenir.

Belirlenen konumların veri setinde toplanmasının ardından tüm coğrafi konumların işaretlendiği bir harita oluşturulmuştur (Şekil 1). Görselleştirme işlemi, tüm konumların toplu olarak görülmesini ve olası hataların kontrol edilerek önceden engellenmesini sağlar.



Şekil 1. İlgili Konumların Harita Üzerinde Görünümü

3.3.2. Konumlar Arası Mesafeler

Koordinatlar, sadece ilgili noktaların coğrafi konumlarını göstermektedir. Çözümlerin değerini bulabilmek için bundan daha fazlasına ihtiyaç vardır. Oluşturulan bir çözüm (kromozom), konumların tümünün dolaşılmasını temsil eden bir sıralamadır. Cevaplanması gereken soru, söz konusu çözümdeki sıraya göre yolculuk yapıldığında kat edilecek toplam mesafenin ne kadar olacağıdır. Bunun için öncelikle her bir konumun diğer tüm konumlara olan uzaklıkları belirlenmelidir.

İki konumun birbirine olan uzaklığı doğrusal olarak ölçülebilir. Ancak dünyanın küresel yapısından ötürü, gerçeğe daha yakın olması sebebiyle Haversine metodu kullanılmıştır. Her iterasyonda ihtiyaç duyulan mesafelerin yeni baştan

hesaplanması, fazladan işlem zamanı gerektirmektedir. Bunu aşabilmek için konumlar arasındaki mesafeler, sürecin başında hesaplanarak iki boyutlu bir dizide saklanmaktadır.

Konumlar arası mesafeler, pratik olması açısından sadece matematiksel hesaplamalar ile bulunmuştur. Gerçek uygulamalarda coğrafi konumlar arası gerçek mesafelerin kullanılması daha uygun olacaktır.

3.3.3. Uygunluk Değeri: Toplam Mesafe

Kromozomlar oluşturulduklarında başarı değerlerinin belirlenmesi gerekir. Uygunluk değeri, sözü geçen başarı değeridir. Uygunluk değerleri, sıralama kümesi olarak belirlenen bir rotanın izlenmesi sonucunda elde edilecek toplam mesafe ölçüsüdür. Bunu belirlemek amacıyla sıralamanın en başından itibaren sırasıyla ardışık tüm nokta çiftleri arasındaki mesafeler toplam mesafeye eklenir. Son olarak başlangıç ve bitiş noktaları arasındaki mesafe toplam mesafeye dâhil edilir. Söz gelimi bir çözüm kümesi {3, 2, 1, 4} şeklinde sıralandıysa, bu kümedeki ardışık olan konumlar arasındaki mesafeler ilgili veri setinden çağırılır. Başka bir ifadeyle, 3 ile 2, 2 ile 1, 1 ile 4 ve 4 ile 3 arasındaki mesafe değerleri toplanarak çözümün değeri bulunur. Bu değer, uygunluk değeri olarak kullanılır. Bu işlemin sözde kodu aşağıda yer almaktadır:

```
BAŞLA

toplam_mesafe = 0
i = 1 // eleman indisi
İÇİN i < dizi_elemanlarının_sayısı:
        toplam_mesafe =+ i. eleman ile (i+1). eleman arası mesafe
        i'yi 1 artır
        İÇİN SON
toplam_mesafe =+ ilk ve son eleman arasındaki mesafe
BİTİR
```

3.3.4. Çözüm Kümesi: Popülasyon ve Kromozomlar

Canlıların yıllar içinde varlıklarını nesilden nesile sürdürmelerine benzer biçimde, genetik algoritmada da çözümlerin nesilden nesile gelişmesi için bir popülasyon oluşturulur. Daha sonra genetik algoritma operatörleri kullanılarak iteratif olarak yeni çözümler geliştirilir.

Sürecin başında oluşturulan başlangıç popülasyonu, rastgele türetilmiş belli sayıda kromozomdan meydana gelir. Her bir kromozom, 1'den başlayarak ardışık

sıralı değerlerin tümünü içeren, bir karışık sıralı sayı dizisidir. Dizinin her bir elemanı bir konumun sıra numarasına karşılık gelir. Bu sebeple ziyaret edilmesi gereken bir adresi gösteren sayı, sıralamada sadece bir kere yer alır. Her bir kromozom, iyi veya kötü olduğuna bakılmaksızın, aynı sayıları içeren farklı bir sıralamadan oluşur.

En küçük bileşenden başlayarak sistematik olarak sıralamak gerekirse, genler coğrafi noktaların sıra numarasını gösterir. Genlerin bir araya gelmesiyle kromozom yani çözüm oluşur. Kromozomlar ise popülasyonu meydana getirir. Zaman içinde kromozomların eklenip çıkarılmasıyla değişime uğrayan popülasyonlar ise nesilden nesile geçişi temsil eder.

Uygulamada, başlangıç popülasyonunu oluşturmak için, öncelikle coğrafi koordinatı verilen tüm noktalar, birden başlayarak ardışık sıralı olarak numaralandırılır. Daha sonra bu sayılardan karışık sıralı bir sayı dizisi oluşturulur. Her bir sıralama ayrı bir kromozomdur, başka bir deyişle farklı bir çözümdür. Rastgele sıralama türetme yöntemi sadece başlangıç çözümü için geçerli olup ilerleyen süreçte hiçbir şekilde bu yöntem kullanılmamaktadır. Sonraki nesillerde kromozomlar genetik algoritma operatörleriyle oluşturulur.

3.3.5. Başlangıç Parametreleri

Genetik Algoritma sürecine başlamadan önce bazı parametrelerin belirlenmesi gerekir. Bunun için yazılımın arayüzünde aşağıdaki parametrelerin girilebileceği bir panel bulunmaktadır:

Maksimum Nesil (Max Generation): Süreç, kullanıcı tarafından durdurulmadıkça, oluşturulacak en son nesilin sayısını gösterir. Bu sayıya ulaşıncaya kadar hesaplama devam eder ve uygunluk değerinde gelişmeler görülebilir. Kullanıcı süreci duraklatsa bile, maksimum nesil değerine ulaşıncaya kadar süreç kaldığı yerden devam ettirilebilir.

Popülasyon Büyüklüğü (Population Size): Genetik algoritma süreci başladıktan sonra sabit bir değer olarak kalacak topluluk büyüklüğü değeridir. Yeni çözümler (kromozomlar) eklendikçe aynı sayıda çözüm popülasyondan çıkarılır.

Çocuk Sayısı (Number of Children): Her yeni nesilde türetilecek yeni çözüm sayısıdır.

Mutasyon Olasılığı (Mutation Probability): Mutasyon, çözümlerde rastgele olarak küçük değişiklikler yapılmasını sağlar. Genellikle 0,01 gibi düşük olasılık değeri seçilir. Rastgele türetilen sayı, bu değerin altında olduğunda mutasyon

işlemi yapılır.

3.4. Temel Fonksiyonlar

Yazılımın işleyişini sağlayabilmek için arka planda çalışan çok sayıda fonksiyon bulunmaktadır. Aşağıda, sürecin işleyişinde etkili olan en önemli fonksiyonlara değinilmektedir. Konuyla teknik açıdan ilgilenen okuyucular, projenin sayfasında paylaşılan kodları inceleyerek tüm ayrıntılara ulaşabilirler.

3.4.1. Çaprazlama Fonksiyonu

Genetik algoritmanın en kritik aşaması, çaprazlama işlemidir. Bu işlem, yeni bireylerin oluşmasını sağlar. Çaprazlama işlemi, problemin doğası gereği yerel arama metodunun kullanılmasını zorunlu kılar. Burada amaç, çaprazlama işlemi sonucunda ortaya çıkan yeni çözümlerin yerel minimumlara takılma riskini en aza indirecek ve potansiyel olarak başarılı çözümler türetebilecek çaprazlama ve yerel arama fonksiyonlarını bir arada kullanmaktır. Sengoku ve Yoshihara (1998) tarafından yapılan çalışmada 2-Opt yerel arama algoritmasıyla yapılan iyileştirmelerde çoğunlukla yerel minimuma takılma sorununun ortaya çıktığı belirtilmiş ve buna yönelik olarak Açgözlü Alttur Çaprazlama (Greedy Subtour Crossover) yöntemi önerilmiştir. Bu doğrultuda, uygulamadaki çaprazlama fonksiyonu için Açgözlü Alttur Çaprazlama ve yerel arama fonksiyonu için ise 2-Opt Yerel Arama yöntemi kullanılmıştır.

Açgözlü Alttur Çaprazlama yönteminde (Şekil 2) kromozomdaki bir nokta tesadüfi olarak seçilir. Verilen örnekte C noktası seçilmiştir. Daha sonra her iki ebeveyn kromozomlardaki genler biri sağa doğru, diğeri sola doğru olmak üzere sırayla yeni kromozoma eklenir. Bu işlem, daha önce eklenmemiş tüm genler için devam eder. Her iki ebeveyn kromozomda daha önceden eklenmiş olan genlere sıra geldiğinde, bu noktadan sonraki eklenmemiş olan genler yeni kromozoma rastgele olarak eklenir. Açgözlü Alttur Çaprazlama'nın sözde kodu aşağıda yer almaktadır:

BAŞLA

FONKSİYON çaprazla

AL birinci_rota, ikinci_rota, mesafeler_matrisi
rota_eleman_sayısı = eleman_sayısını_al(birinci_rota)
seçilen_değer = 1 ile rota_eleman_sayısı arasında rastgele bir tam sayı seç
birinci_indeks = birinci_rota'nın seçilen_değer sıra numarasını bul
ikinci_indeks = ikinci_rota'nın seçilen_değer sıra numarasını bul

OLDUĞUNDA her iki indeks de dizi sonuna gelinceye kadar devam et

birinci_rotada birinci_indeksten önceki sıradaki değeri yeni rotaya ekle

birinci_indeksi bir eksilt

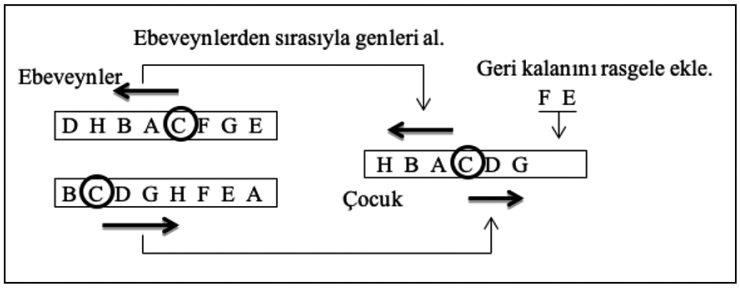
ikinci_rotada ikinci_indeksten sonraki sıradaki değeri yeni rotaya ekle

ikinci_indeksi bir artır

BİTİŞ OLDUĞUNDA

kalan değerleri yeni rotaya rastgele ekle

BİTİR

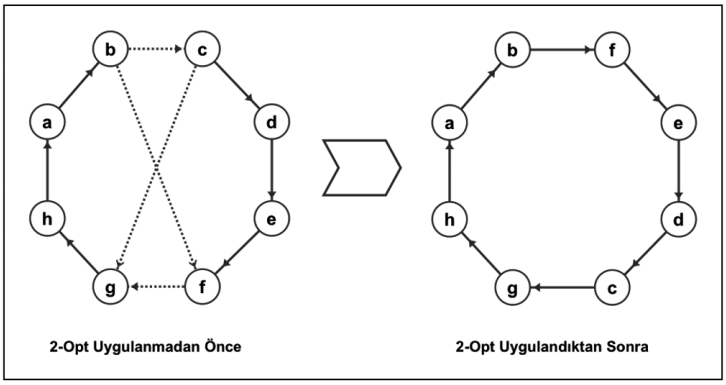


Şekil 2. Açgözlü Alttur Çaprazlama (Sengoku ve Yoshihara, 1998, s. 284)

3.4.2. 2-Opt Yerel Arama Fonksiyonu

Gezgin satıcı probleminin çözümünde, genetik algoritma ile birlikte Croes (1958) tarafından önerilen 2-Opt yerel arama sezgiseli kullanıldığında "Melez Genetik Algoritma" olarak isimlendirilen algoritma daha yüksek bir isabet gücüne sahiptir (Lin ve Hu, 2008, s. 465).

Gezgin satıcı probleminin çözüm algoritmaları arasında en iyi bilinen yerel arama algoritması olan 2-Opt prosedüründe, bir turun alt turları tersine döndürülerek iyileşme sağlanır (Sengoku ve Yoshihara, 1998, s. 284). Algoritmanın anlaşılması için oluşturulmuş örnekte {a→b→c→d→e→f→g→h→a} rotasını izleyen sekiz konumdan oluşan bir tur yer almaktadır (Şekil 3). [bc] ve [fg] kenarlarına 2-Opt uygulanması için [bf]+[cg] < [bc]+[fg] koşulu kontrol edilir. Bu koşul doğru ise alt tur tersine çevrilir. Yeni tur {a→b→f→e→d→c→g→h→a} olur. 2-Opt yerel minimumların bulunmasında güçlü bir metot olmasıyla birlikte, genetik algoritma ile bir arada kullanıldığında hem yerel hem de global minimumların bulunmasında güçlü bir yapıya sahip olmaktadır.



Şekil 3. 2-Opt Yerel Arama Metodunun Şematik Gösterimi
Kaynak: Sengoku ve Yoshihara (1998) çalışmasından uyarlanmıştır.

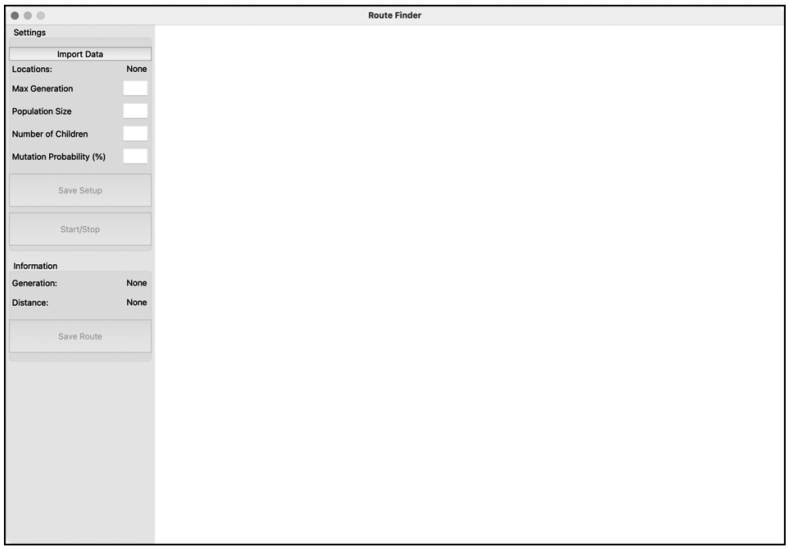
3.4.3. Popülasyonun Revize Edilmesi

Yeni çözümler popülasyona eklendikten sonra, tüm bireyler uygunluk değerlerine göre küçükten büyüğe doğru sıralanır. En başarılı çözüm en düşük uygunluk değerine sahip olan birinci sıradaki çözümdür. Son sıralardaki çözümlerin uygunluk değerleri daha kötüdür. Sürecin başında belirlenmiş olan popülasyon büyüklüğü parametresini aşan çözümler son sıradan başlayarak popülasyondan çıkartılır. Böylece çözümler sürekli gelişirken, popülasyon büyüklüğü nesilden nesile değişmeden devam eder.

3.5. Geliştirilen Yazılım ile Problemin Çözümü

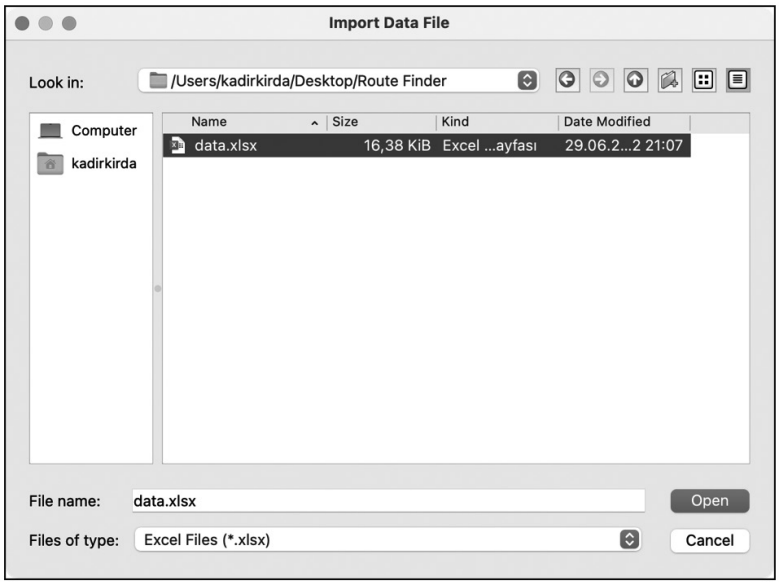
Bu başlıkta, geliştirilen yazılım ve bu yazılım kullanılarak ilgili problemin çözümü anlatılacaktır. Yazılımın kaynak kodlarına https://github.com/kadirkirda/route-finder adresinden erişilebilir. Proje, en yaygın özgür yazılım lisanslarından biri olan MIT lisansına sahiptir.

Yazılımın arayüzü ilk açıldığında, sol tarafta ayarlamaların yapıldığı bölüm ve sağ tarafta haritanın görüntüleneceği beyaz bir ekran bulunur (Şekil 4).



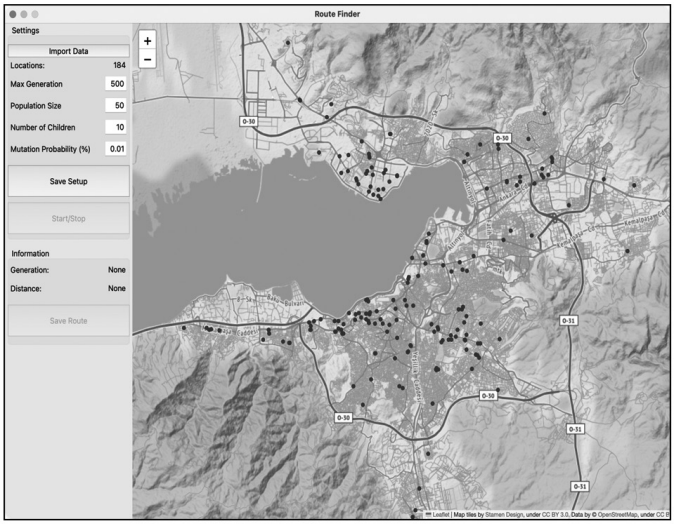
Şekil 4. Geliştirilen Uygulamanın Açılış Ekranı

Sol taraftaki panelde en üstte "Import Data" butonu tıklandığında oluşturulan veri setinin yükleneceği pencere açılır (Şekil 5).



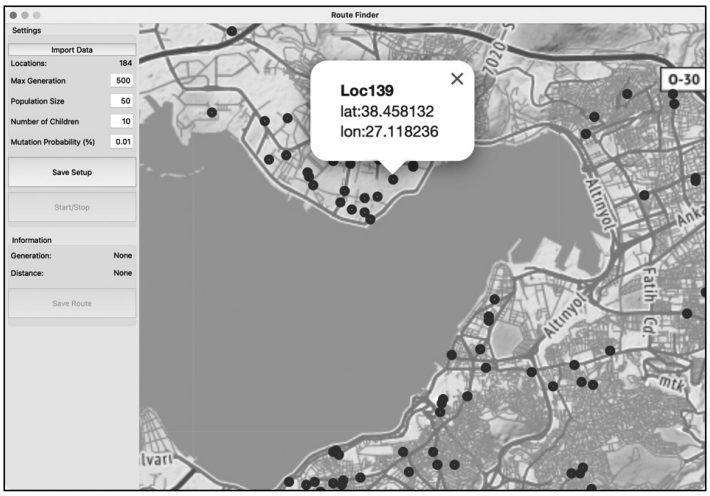
Şekil 5. Verilerin Yüklenmesi

Örnek veri dosyasına projenin sayfasından ulaşılabilir. Veriler yüklendiğinde, sağ taraftaki beyaz ekranda veri setindeki konumların işaretlendiği bir harita görüntülenir (Şekil 6).



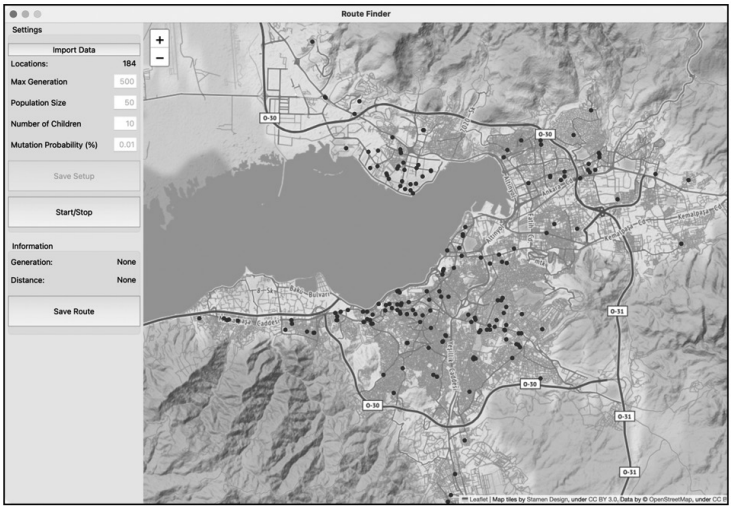
Şekil 6. Yüklenen Konumların Harita Üzerinde İşaretlenmesi

Ayrıca "Locations" kısmında toplam konum sayısını görmek mümkündür. Harita üzerinde konumları gösteren işaretler tıklanarak ilgili konumun adı, enlem ve boylam bilgileri kontrol edilebilir (Şekil 7).



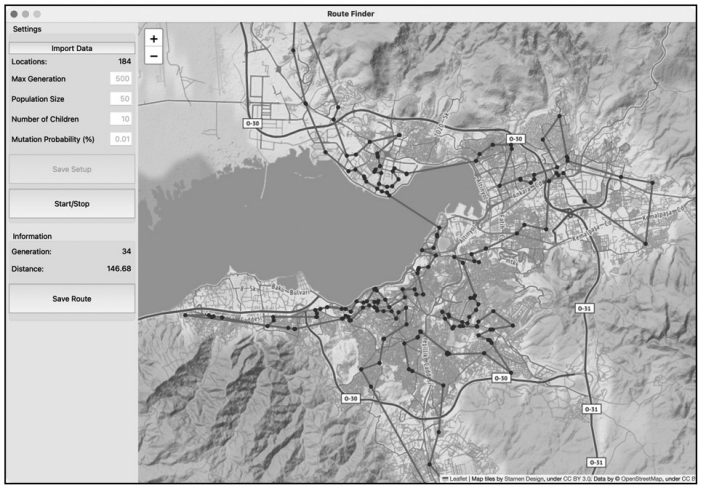
Şekil 7. Konum Ayrıntılarının Görüntülenmesi

Veriler yüklendikten sonraki aşama, genetik algoritma sürecinin parametrelerinin belirlenmesidir. İlgili kutucuklara maksimum nesil (Max Generation), popülasyon büyüklüğü (Population Size), her yeni nesilde eklenecek çözüm sayısı (Number of Children) ve mutasyon olasılığı (Mutation Probability) bilgileri için istenen değerler girilerek ayarları kaydetme (Save Setup) butonuna tıklanır ve parametreler kaydedilir. Parametreler kaydedildikten sonra herhangi bir hataya yer vermemek için bir sonraki veri yüklemesine kadar ilgili değerler sabitlenir (Şekil 8).



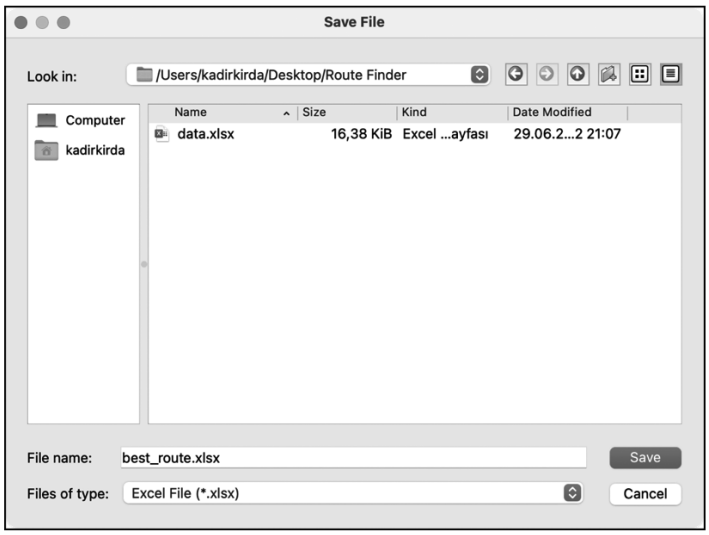
Şekil 8. Parametrelerin Kaydedilmesi

Verilerin yüklenmesi ve parametrelerin kaydedilmesinden sonra sürecin başlatılması için "Start/Stop" butonuna basmak yeterlidir. Bu durumda süreç iterasyonlar şeklinde nesilden nesile devam eder ve haritada en başarılı rota çizilir (Şekil 9). Bu rota, popülasyondaki en iyi uygunluk değerine sahip kromozoma aittir. Son nesile ulaşıncaya kadar süreç devam eder. Bu aşamada süreci sonlandırmak mümkündür. Bunun için "Start/Stop" butonuna bir kez basmak gerekir. Süreci kaldığı yerden devam ettirmek için de aynı buton kullanılır. Tek butona birden fazla işlev yüklemedeki amaç, karmaşıklıktan uzaklaşmak ve kullanıcı dostu bir arayüz tasarlamaktır. Son nesile varıncaya kadar bu işlev kullanılabilir.



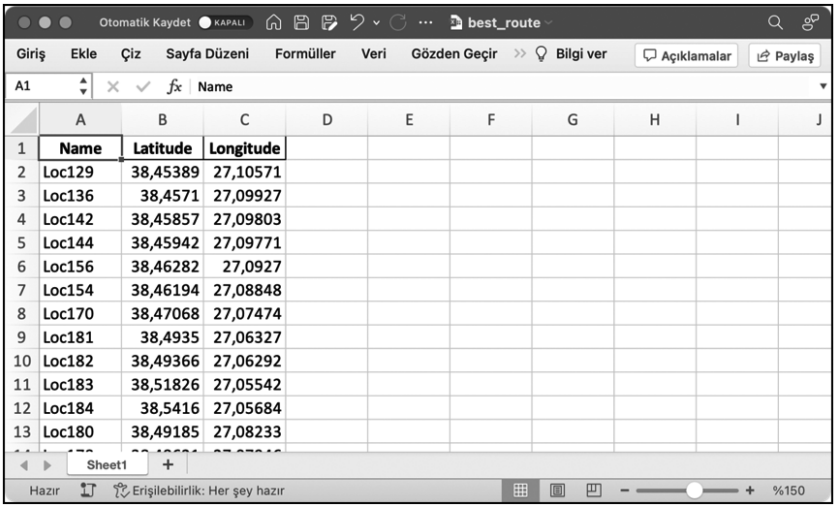
Şekil 9. Geliştirilen Çözümün Harita Üzerinde Çizilmesi

Uygunluk değeri "Distance" kısmında gösterilir. İstenilen sonuca ulaşıldığında, çözümün arayüz üzerinden dışarıya aktarılması mümkündür. Bunun için, "Save Route" butonuna tıklanır. En iyi sıralamaya ait veri dosyasının kaydedileceği yeri belirlemek için bir pencere açılır (Şekil 10).



Şekil 10. En İyi Çözümün Dışa Aktarılması

Kaydedilen dosyada ilgili konumların isim, enlem ve boylam bilgileri, konumların ziyaret edilme sırasına göre listelenir (Şekil 11).



	A	B	C
1	**Name**	**Latitude**	**Longitude**
2	Loc129	38,45389	27,10571
3	Loc136	38,4571	27,09927
4	Loc142	38,45857	27,09803
5	Loc144	38,45942	27,09771
6	Loc156	38,46282	27,0927
7	Loc154	38,46194	27,08848
8	Loc170	38,47068	27,07474
9	Loc181	38,4935	27,06327
10	Loc182	38,49366	27,06292
11	Loc183	38,51826	27,05542
12	Loc184	38,5416	27,05684
13	Loc180	38,49185	27,08233

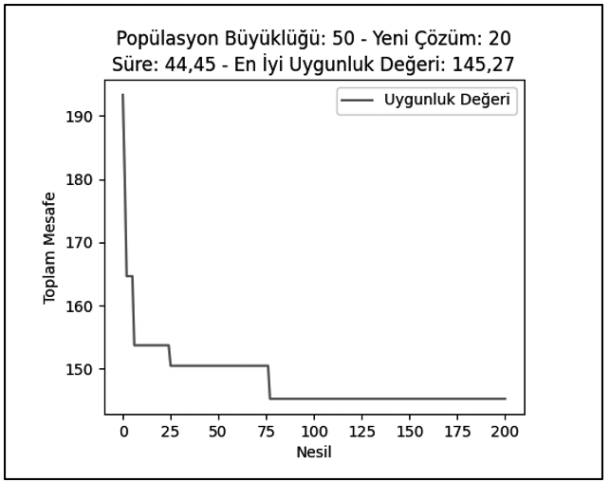
Şekil 11. En İyi Çözüme Ait Sıralama Bilgisi

3.6. Deneysel Sonuçlar

Uygulama için 184 konuma ait enlem ve boylam bilgileri bir Excel dosyasında derlenmiş ve yazılımın daha önce bahsedilen işleyişindeki gibi yüklenmiştir. Test amaçlı olarak iki parametrede değişiklik yapılmış ve toplam üç farklı konfigürasyon çalıştırılmıştır. Bu seçenekler için popülasyon büyüklüğü (Population Size) ve yeni çözüm sayısı (Number of Children) parametreleri sırasıyla {50-20}, {100-20}, {100-40} olarak ayarlanmıştır. Bunlar dışındaki parametreler için aşağıdaki değerler kullanılmıştır:

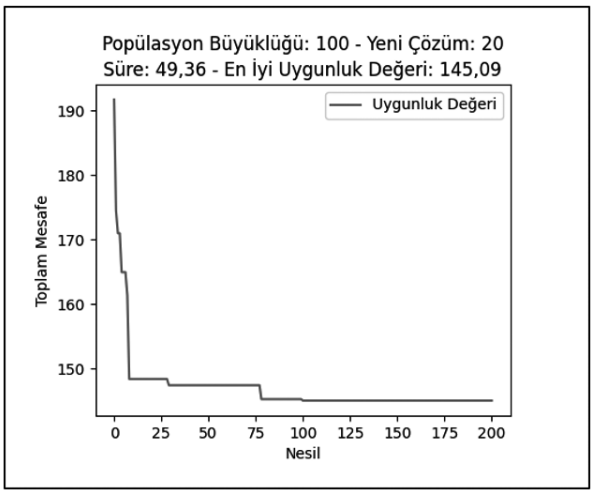
- Maksimum nesil (Max Generation): 200
- Mutasyon olasılığı (Mutation Probability): 0,01

Eğitim sürecinin grafiksel gösterimleri incelendiğinde (Şekil 12-13-14) genetik algoritma sürecinde birbirine yakın fakat farklı uygunluk değerlerine ulaşıldığı görülmektedir. Yakın değerlerin elde edilmesi, yöntemin tutarlılığını göstermektedir. Farklı çözümlere ulaşılması ise genetik algoritmanın sezgisel algoritma olması dolayısıyla optimum olmasa da optimuma yakın çözümler elde etmeyi sağladığının bir göstergesi olduğu söylenebilir.

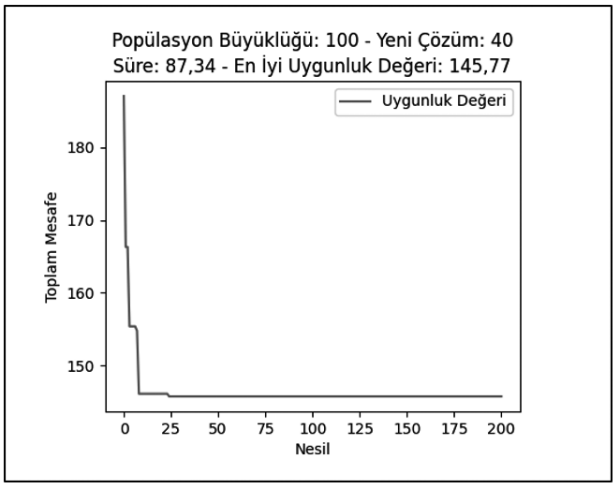


Şekil 12. Popülasyon Büyüklüğü: 50 – Yeni Çözüm: 20

İşlem süreleri incelendiğinde, ilk iki deneme arasında büyük bir fark gözlenmemekte (44,45sn, 49,36sn) fakat üçüncü denemenin işlem süresinde (87,34sn) belirgin derecede farklılığın olduğu görülmektedir. Bu durumda, popülasyon büyüklüğünün süre açısından büyük bir etkisinin olmadığı, yeni çözüm sayısının ise işlem süresini doğrudan etkilediği sonucu çıkarılabilir.



Şekil 13. Popülasyon Büyüklüğü: 100 – Yeni Çözüm: 20



Şekil 14. Popülasyon Büyüklüğü: 100 – Yeni Çözüm: 40

Üç denemenin grafikleri incelendiğinde, göze çarpan başka bir nokta ise minimuma ulaşma hızıdır. Yeni çözüm sayısı parametresi yüksek olan son denemede (Şekil 14) çok daha hızlı şekilde minimuma ulaşıldığı görülebilir. Bahsi geçen denemenin nesilden nesile ilerleyişi esnasında toplam mesafe yani uygunluk değerinin ilerleyişi ve her bir nesilde ne kadar süre harcandığına ilişkin kayıtların başlangıç bölümü aşağıdaki tabloda görülmektedir (Tablo 1).

Tablo 1. Genetik Algoritma Sürecinin Ayrıntılı Kayıtları

Nesil Numarası	Toplam Mesafe (km)	İşlem Süresi (sn)
Başlangıç Popülasyonu	187,06	2,22
1. Nesil	166,31	0,63
2. Nesil	166,31	0,58
3. Nesil	155,40	0,76
4. Nesil	155,40	0,62
5. Nesil	155,40	0,69
6. Nesil	155,40	0,45
7. Nesil	154,79	0,52
8. Nesil	146,13	0,46
9. Nesil	146,13	0,61
10. Nesil	146,13	0,44
11. Nesil	146,13	0,43

Tablo 1. Genetik Algoritma Sürecinin Ayrıntılı Kayıtları		
Nesil Numarası	Toplam Mesafe (km)	İşlem Süresi (sn)
12. Nesil	146,13	0,41
13. Nesil	146,13	0,45
14. Nesil	146,13	0,43
15. Nesil	146,13	0,45
16. Nesil	146,13	0,45
17. Nesil	146,13	0,43
18. Nesil	146,13	0,45
19. Nesil	146,13	0,45
20. Nesil	146,13	0,44
21. Nesil	146,13	0,41
22. Nesil	146,13	0,41
23. Nesil	146,13	0,40
24. Nesil	145,77	0,41
25. Nesil	145,77	0,43

Yapılan çıkarımlar için birkaç farklı deneme arasındaki farklılıklar esas alınmıştır. Daha derin bir değerlendirme yapabilmek için çok sayıda deneme yapılmalı ve istatistiksel analiz yöntemleriyle sonuçlar incelenmelidir. Bu çalışmadaki amaç, genetik algoritmanın işleyişini ve farklı parametrelerle hangi değişimlerin gözlenebileceğini okuyuculara sunmaktır. Paylaşılan yazılım ile okuyucuların daha kapsamlı analizler yapabilmesi mümkündür.

4. SONUÇ

Genetik algoritma, optimum çözümü çok zor olan problemler için bir sezgisel çözüm yöntemidir. Optimum çözümü garanti etmese de optimuma yakın çözüme ulaşmayı garanti eder. Evrim teorisinden yola çıkılarak geliştirilen bu yöntem ile çözümü elde edilebilecek problemlerden biri gezgin satıcı problemidir. Bu çalışmada bu tip bir problemin çözümü için genetik algoritma tabanlı bir yazılım geliştirilmiştir. Geliştirilen yazılım, araştırmacıların ilgili alandaki araştırma ve geliştirme olanaklarına katkı sağlamak amacıyla açık kaynaklı olarak paylaşılmıştır. Genetik algoritma, teoride anlaşılması ve uygulanması kolaydır. Bununla birlikte çok sayıda hesaplama gerektirdiği için pratikte yazılım desteğine ihtiyaç duyulur. Geliştirilen yazılım, uygulama arayüzü sayesinde kodlama bilmeyen

kullanıcılar da dâhil olmak üzere, gezgin satıcı probleminde genetik algoritmanın uygulanması konusunda araştırmalar yapmak isteyen kişilerin kullanabilecekleri bir araçtır. Bu çalışma geliştirilen yazılım ile birlikte ele alındığında, öne çıkan noktalar aşağıdaki gibi özetlenebilir:

1. Örnek uygulama sonuçlarından da anlaşılabildiği gibi genetik algoritmanın GSP'de kullanılmasıyla optimum sonuca ulaşılma garantisi olmasa da optimuma yakın sonuçların elde edilebileceği görülmektedir. Çoğu durumda sezgisel hesaplamaların kazandırdığı zaman, küçük maliyet farklarından daha değerli olarak görülmektedir.
2. Yazılım sayesinde GSP kategorisindeki problemlerin çözümü için veri setini hazırlamak dışında bir zorlukla karşılaşmadan, hızlıca çözüme ulaşma olanağı bulunmaktadır.
3. Entegre harita özelliği, coğrafi konumların harita üzerindeki kontrolünü ve hesaplamalardan sonra çözümün başka bir araca gerek duyulmadan interaktif olarak görselleştirilmesini sağlamaktadır.
4. Parametrelerin girilebildiği panel sayesinde farklı parametre seçeneklerini deneme ve sonuçları görme imkânı bulunmaktadır.
5. Kodlamada Python dili kullanılmıştır. Python akademik araştırmalarda en yaygın kullanılan programlama dillerinden biridir. Python dilini bilen ve öğrenen herkesin kodlardan yararlanma fırsatı bulunmaktadır.
6. Ayrıca, bahsedilen tüm bu özelliklere sahip bir projeye rastlanmamıştır. Dolayısıyla bu çalışmanın özgün bir çalışma olduğu söylenebilir.

Teknolojik gelişmelerin ışığında, bilimsel araştırmalar da sürekli yeni fikirlerin ortaya çıkmasına katkı sağlamaktadır. Şöyle ki; sonraki çalışmalarda mevcut yazılıma yeni özellikler katılarak, algoritmanın performansını geliştirmek mümkün olabilir. Ya da çaprazlama algoritmaları eklenerek birden fazla çaprazlama algoritması arasında seçim yapılması sağlanabilir. Bundan başka; yazılımın web uygulamasına dönüştürülmesi seçeneği değerlendirilebilir. Bu ve benzeri birçok olanak bulunmaktadır. Günümüzde teknolojideki gelişmeler, geçmişte olduğundan çok daha hızlıdır.

Göz ardı edilmemesi gereken önemli bir konu da açık kaynaklı yazılım alanındaki gelişmelerdir. Hem özgür yazılımı destekleyen birey ve topluluklar, hem de yazılım şirketlerinin desteği sayesinde araştırmacılar, yazılım geliştirme konusunda ihtiyaç duyulabilecek her türlü kaynağa zaman kaybetmeden ulaşabilmektedir. Bu olanakların doğru şekilde değerlendirilmesi, bilimsel alanda olduğu kadar kalkınma ve sürdürülebilirlik açısından da fayda sağlayacaktır.

KAYNAKÇA

Cevre, U., Özkan, B. ve Uğur, A. (2007). *Gezgin Satıcı Probleminin Genetik Algoritmalarla Eniyilemesi ve Etkileşimli Olarak İnternet Üzerinde Görselleştirilmesi*. Proceedings from XI I."Türkiye'de İnternet "Konferansı,Ankara.

Chan, F. T. S., Chung, S. H. ve Wadhwa, S. (2005). A hybrid genetic algorithm for production and distribution. *Omega*, *33*(4), 345-355. doi:10.1016/j.omega.2004.05.004

Chen, T. Y. ve Chen, C. J. (1997). Improvements of simple genetic algorithm in structural design. *International Journal for Numerical Methods in Engineering*, *40*(7), 1323-1334. doi:10.1002/(SICI)1097-0207(19970415)40:7<1323::AID-NME117>3.0.CO;2-T

Croes, G. A. (1958). A Method for Solving Traveling Salesman Problems. *Operation Research*, *6*(6), 791-812.

Haupt, R. L. ve Haupt, S. E. (2004). *Practical Genetic Algorithms*. Canada: A John Wiley & Sons, Inc., Publication.

Huang, G. Q., Zhang, X. Y. ve Liang, L. (2005). Towards integrated optimal configuration of platform products, manufacturing processes, and supply chains. *Journal of Operations Management*, *23*(3-4), 267-290. doi:10.1016/j.jom.2004.10.014

Kahvecioğlu, A. (2004). *Onarılabilir elemanlara önleyici bakımın etkisi ve optimizasyonu*. Proceedings from Bakım Teknolojileri Kongresi ve Sergisi, Denizli.

Karova, M., Smarkov, V. ve Stoyan, P. (2005). Genetic operators crossover and mutation in solving the TSP problem. *International Conference on Computer Systems and Technologies,* 2-7. Retrieved from https://ecet.ecs.ru.acad.bg/cst05/Docs/cp/SIII/IIIA.20.pdf

Kırda, K. (2013). *Evsel İlaç Atıklarının Toplanması Projesindeki Tersine Lojistik Sürecinin Modellenmesi İçin Genetik Algoritmaların Kullanılması.* Doktora Tezi. Dokuz Eylül Üniversitesi, İzmir.

Lin, C.-h. ve Hu, J.-w. (2008). A Genetic Algorithm with Priority Selection for the Traveling Salesman Problem. *World Academy of Science, Engineering and Technology*, *42*, 465-475.

Mitchell, M. ve Taylor, C. E. (1999). Evolutionary Computation: An Overview. *Annual Review of Ecological Systems*, *30*, 593-616.

Potvin, J. Y. (1996). Genetic Algorithms for the Traveling Salesman Problem. *Annals of Operations Research*, *63*, 339-370.

Sansarcı, E., Bayraktar, D., Aktel, A. ve Çelebi, D. (2009). *Gezgin Satıcı Problemi İçin Bir Memetik Algoritma Önerisi*. Proceedings from Yöneylem Araştırması ve Endüstri Mühendisliği 29. Ulusal Kongresi, Ankara.

Şen, Z. (2004). *Genetik Algoritmalar ve Eniyileme Yöntemleri*. İstanbul: Su Vakfı Yayınları.

Sengoku, H. ve Yoshihara, I. (1998). A fast TSP solver using GA on JAVA. *in Proc. 3rd Int. Symp. Artif. Life and Robot*, 283-288.

Sivanandam, S. N. ve Deepa, S. N. (2008). *Introduction to Genetic Algorithms*. New York: Springer.