

Melez Bir Eniyileme Yöntemi ile Rota Planlama

Barış ÖZKAN, Utku CEVRE, Yrd. Doç. Dr. Aybars UĞUR

Ege Üniversitesi, Bilgisayar Mühendisliği Bölümü, İzmir
barisozkan@mail.ege.edu.tr, utkucevre@mail.ege.edu.tr, aybars.ugur@ege.edu.tr

Özet: Rota planlama, aralarındaki uzaklıklar belirli olan şehirlerden geçen en uygun maliyetli güzergahı bulmayı amaçlayan bir problem türüdür. Yoğun olarak üzerinde çalışılan bir rota planlama problemi de Gezgin Satıcı Problemi'dir. Gezgin Satıcı Problemi (GSP), verilen belirli sayıda şehrin tümünü en az maliyetle dolaşmayı sağlayan turu belirlemeyi hedefleyen bir kombinasyonel eniyileme problemidir. Çözümü için birçok alanda kullanılmakta olan çok sayıda sezgisel algoritma geliştirilmiştir. Çalışmamızda, rota planlama için GSP üzerinde, Genetik Algoritmaları yerel arama sezgileri ile birleştiren melez bir çözüm geliştirilmiş ve Java ortamında gerçekleştirilmiştir. Uygulama örneği olarak, Türkiye'nin 81 ilini karayolu bağlantısı ile dolaşan en kısa güzergah bulunmuştur. Yöntemimiz ayrıca TSPLIB içerisindeki değişik veri setleriyle test edilmiş ve sonuçlar sunulmuştur. Son olarak geliştirdiğimiz esnek ve etkileşimli programın araştırmacılar ve diğer kullanıcılar açısından yararları kısaca belirtilmiştir.

Anahtar Sözcükler: Rota Planlama, Gezgin Satıcı Problemi, Genetik Algoritmalar, Yerel Arama Sezgileri, Eniyileme

Route Planning With a Hybrid Optimization Technique

Abstract: Route planning is a type of problem which aims to find the optimal tour for cities between which the distances are known. One of the extensively studied route planning problems is Traveling Salesman Problem. Traveling Salesman Problem (TSP) is a combinatorial optimization problem which aims to find the optimal total routing cost for a given collection of cities. A variety of local heuristics are available for solving TSP. In this study a solution method for route planning on TSP is developed using genetic algorithms with local search heuristics and implemented in Java platform. The shortest tour that traverses the 81 cities of Turkey which are connected through highways was specified as a case study. Our method is also tested with numerous benchmark problems from TSPLIB and results presented. As a result the advantages of this flexible and interactive program for researchers and other users are briefly discussed.

Keywords: Route Planning, Traveling Salesman Problem, Genetic Algorithms, Local Search Heuristics, Optimization.

1. Giriş

Rota planlama (Route planning), belirli bir harita üzerindeki herhangi bir A noktasından B noktasına giden en uygun yolu bulmayı temel alan bir problem türüdür. [10] Uygunluk kriterleri maliyete (yol uzunluğu, zaman gibi)

göre belirlenir. Problemin çözümünde, başlangıç noktasından bitiş noktasına kadar olan şehirlerin (noktaların) dolaşılma sırası rotayı belirler. Rota planlama problemlerinin en belirgin uygulaması planlanan rotadaki başlangıç ve bitiş şehirlerinin aynı olduğu Gezgin Satıcı Problemidir. Kimi durumlarda ise A şehirden

B şehrine gitmenin maliyeti, B şehrinden A şehrine gitmenin maliyetinden farklı olabilir. Bazı şehirler arasında tek yön yolların olması, gidiş ve geliş yönlerindeki yolların trafik sıklığının getireceği süre farklılıkları gibi durumlar dikkate alındığında Simetrik Olmayan GSP devreye girer.

Gezgin Satıcı Problemi (Traveling Salesman Problem) veya kısaca GSP (TSP), aralarındaki uzaklıklar bilinen N adet noktanın (şehir, parça veya düğüm gibi) her birisinden yalnız bir kez geçen en kısa veya en az maliyetli turun bulunmasını hedefleyen bir problemdir [6]. Ayrık ve Kombinasyonel Eniyileme (Combinatorial Optimization) problemlerinin kapsamına girer. Problemdaki maliyet, uzaklık, zaman veya para gibi unsurlardan biri olabilir. Çizge teorisi içinde ise, GSP “Verilen bir ağırlıklı çizgede (düğümner yani köşeler şehirleri; kenarlar ise şehirlerin arasındaki yolları göstermek; ağırlıklar da yolun maliyeti veya uzunluğu olmak üzere), en düşük maliyetli Hamilton Çevriminin bulunması” şeklinde tanımlanabilir. Öklit bazlı (Euclidian) GSP için düğümler, R_2 (herhangi bir d için, R_d olacak şekilde) uzayındadır [3].

En kısa turu bulmanın en temel yolu, verilen N adet şehir için tüm şehir permütasyonlarını listeleterek her bir olası güzergahın toplam yol uzunluğunu hesaplamaktır. En küçük değere sahip olan güzergah veya güzergahlar kesin çözümdür. Bu yöntemin zaman karmaşıklığı $O(n!)$ 'dir. Örneğin 30 şehir olduğu durumda bile, permütasyonların sayısı, bilgisayarın kısa sürede çözemeyeceği kadar büyük (30!) olmaktadır. Bu nedenle, kesin yöntemlerin yanında, kısa sürede iyi çözümlerin bulunmasını sağlayan yaklaşım yöntemleri (sezgi ve yaklaşım algoritmaları gibi) de günümüzde birçok alanda kullanılmaktadır [13][34]. Problemin çözümü günlük hayatta, yol ve rota planlama (uçak, otobüs, dağıtım kamyonları, bilgisayar ağları, posta taşıyıcılar, vb.), iş planlama, baskı

devre kartlarındaki delgi işlemi sırasının belirlenmesi gibi birçok alanda kullanılmaktadır.

Bu çalışmada, GA ve 2-opt yöntemleri ile, uzaklıkları verilen yerleşim noktalarını dolaşarak en uygun rotayı bulan melez bir yöntem ve yazılım geliştirilmiştir. Rota planlama örneği olarak GSP seçilmiş; uygulama örneği olarak ise Türkiye'nin 81 ilinin karayolu ağı üzerindeki en uygun dolaşım sırası belirlenmiştir. Çözüm Java üzerinde görselleştirilmiş ve Internet üzerinden de ulaşılabilecek şekilde, web-tabanlı gerçekleştirilmiştir. Makalenin ikinci bölümünde kullanılan eniyileme yöntemleri olan genetik algoritmalar ve yerel arama sezgileri (2-opt) incelenerek temel kavramlarla ilgili bilgi verilmiş, birlikte kullanımının getireceği avantajlar belirtilmiştir. Makalenin üçüncü bölümünde, geliştirilen Java tabanlı GSP çözücü araçtan bahsedilmiştir ve kullanım bilgileri verilmiştir. Dördüncü bölümde uygulama örneğimize değinilmiş ve planlanan rotaya yer verilerek, yöntemimizin deneysel test sonuçları sunulmuştur. Beşinci bölümde ise problemin çözüm ve aracın geliştirme sürecinden elde edilen sonuç ve değerlendirmelere yer verilmiştir.

2. Eniyileme Yöntemleri

GSP, en iyiye yakın çözümleri bulmaya çalışan kombinasyonlara dayalı eniyileme yöntemleri için standart bir test ortamı haline gelmiştir [14]. Literatürde, yerel arama algoritmaları, genetik algoritmalar [15], benzetimli tavlama (simulated annealing) [23], yasak arama (tabu search) [9], karınca kolonisi optimizasyonu [25][1] ve yapay sinir ağları [20] gibi eniyileme tabanlı birçok yaklaşım yöntemi önerilmiştir. [21] ve [22], GSP için kullanılan sezgiler ve yerel arama algoritmaları hakkında mükemmel literatür taraması niteliğindedir. Sengoku ve Yoshihara'nın 1998'te yaptıkları çalışma [31] GSP'nin GA ile çözümü alanındaki en önemli işlerden biridir, çünkü geliş-

tirdikleri algoritma, mutasyon evresinde çok etkili olan 2opt [7] yöntemini kullanmaktadır. Literatürde, Dallandır ve Kes (Branch and Cut) yöntemi tabanlı büyük GSP örneklerinde bile kesin sonuç verebilen algoritmalar da tanımlanmıştır [2][30][5]. Concorde bilgisayar kodu, simetrik GSP çözücülere en meşhur örnektir ve Mart 2005'te bir devre kartı içerisindeki 33,810 noktanın tümünün en etkin hangi sırada dolaşılması gerektiğini belirlemiştir.

Problemin çözüm sürecinde melez bir yöntem geliştirirken bizim yararlandığımız eniyileme teknikleri ise genetik algoritmalar ve 2-opt yerel arama sezgisidir.

2.1 Genetik Algoritmalar

Temel ilkeleri ilk kez 1975 yılında John Holland tarafından ortaya atılmış olan Genetik algoritmalar, yapay zekanın hızlı gelişen alanlarından. Özellikle kombinasyonel eniyileme problemlerine yaklaşık iyi sonuçlar bulmayı hedefleyen arama yöntemleridir. Çözümde kullanılacak rastgele seçilmiş bir çözüm kümesi oluşturabilmek için evrimsel mekanizmaların kullanıldığı bu yöntemlerin temel mantığı topluluğun nesilden nesle geçmesi sırasında kötü çözümlerin yok olmasına ve iyi çözümlerden daha iyi çözümlere ulaşılmasına dayanır. [6]

Genetik algoritmalar, arama uzayının büyük ve karmaşık olduğu, mevcut bilgiyle sınırlı arama uzayında çözümün zor olduğu, problemin belirli bir matematiksel modelle ifade edilemediği ve geleneksel eniyileme yöntemlerinden istenen sonucun alınmadığı GSP gibi problemlerde etkili ve kullanışlıdır. Grefenstette et al. [18], Goldberg and Lingle [16], Oliver et al. [29] GSP'ni genetik algoritmalar (GA) ile çözmeye çalışan ilk araştırmacılarıdır [24]. O tarihten bu yana GA'lar bu yönde çok umut verici sonuçlar üretmiş [33] ve GSP için GA tabanlı birçok yaklaşım geliştirilmiştir.

Genetik algoritmalarındaki temel kavramlar genetik biliminde karşılık bulmaktadır. Kromozomlar, problem için olası çözümleri temsil etmektedir. Topluluk (populasyon) kromozomlardan oluşan kümedir. Uygunluk değeri, çözümün kalitesini belirler ve uygunluk fonksiyonu kullanılarak hesaplanır. Çaprazlama ve mutasyon olası çözümleri temsil eden kromozomlar üzerinde uygulanan operasyonlardır. Yeni nesiller, seçilen bireylerin çaprazlama ve mutasyon gibi genetik operatörlerden geçirilmesi ile elde edilir.

Bir problemin çözümünde genetik algoritmalar kullanılırken aşağıdaki adımlar izlenir:

1. **[Başlat]** N adet kromozom (probleme uygun çözümler) içeren topluluğu oluştur.
2. **[Uygunluk]** Her x kromozomu için $f(x)$ uygunluk değerini hesapla.
3. **[Yeni Topluluk]** Aşağıdaki adımları tekrarlayarak yeni populasyonu oluştur.
 - a. **[Seçilim]** Topluluktan uygunluk değerlerini dikkate alarak (uygunluk değeri daha iyi olanların seçilme olasılığı yüksek olacak şekilde) iki kromozom seç.
 - b. **[Çaprazlama]** Belirli bir çaprazlama olasılığıyla ebeveynlerden gelen kromozomları çaprazlayarak yeni birey oluştur. Çaprazlama yapılmazsa ebeveynlerden gelen kromozomları aynen bir sonraki nesle kopyala.
 - c. **[Mutasyon]** Yeni bireyi belirli bir olasılığa göre mutasyona uğrat.
 - d. **[Ekleme]** Oluşturulan bireyi yeni topluluğa ekle.
4. **[Değiştir]** Önceki topluluğu, yeni toplulukla değiştir.

5. [Test] Sonlandırma koşulu sağlandıysa mevcut topluluktaki en iyi çözümü döndür, sağlanmadıysa 2. adıma dön.

2.1.1 Kromozomların Kodlanması

Kromozomların kodlanmasındaki yaklaşım, problemin türüne ve özelliklerine göre farklılık gösterir. En sık kullanılan yöntemler ikili kodlama ve permutasyon kodlamadır. İkili kodlamada her kromozom 1 ve 0'lerden oluşan bir karakter dizisi biçiminde ifade edilir. İkili kodlamanın uygun olduğu bir problem örneği olarak Knapsack (sırt çantası) problemi verilebilir. Permutasyon kodlamada ise her kromozom, ilgili karakterin sıralamadaki pozisyonunu belirten sayılardan oluşan bir dizi ile ifade edilir. Permutasyon kodlama, genelde Gezgin Satıcı Problemi gibi sıralama problemlerinde kullanılır.

2.1.2 Seçim Yöntemleri

Bir sonraki nesle geçerken, yeni topluluğu oluşturmak için mevcut topluluktan çaprazlama ve mutasyon işlemlerine tabi tutulacak bireylerin seçilmesi gerekir. Teoriye göre iyi olan, bir başka deyişle uygunluk değeri yüksek olan bireyler yaşamını sürdürmeli ve bu bireylerden yeni bireyler oluşturulmalıdır. Bu nedenle tüm seçim yöntemlerinde uygunluk değeri fazla olan bireylerin seçilme olasılığı daha yüksek tutulur. Yaygın olarak bilinen seçim yöntemleri Rulet Seçilimi, Turnuva Seçilimi ve Sıralı Seçilimdir. Tablo 1'de rulet seçilimi ve sıralı seçim yöntemleri karşılaştırılmıştır.

Rulet Seçilimi: Bu yöntemde bir bireyin seçilme olasılığı, o bireyin uygunluk değerinin tüm bireylerin uygunluk değerleri toplamına oranı kadardır. Uygunluk değerleri arasındaki farkların fazla olması durumunda rulet seçilimi hep aynı çözümler etrafında dolaşma dolayısıyla da yerel minimumlara takılma sorununa yol açabilir.

Sıralı Seçilim: Bu yöntemde en kötü uygunlukta olan bireye 1 değeri verilir, ondan daha iyi olana 2, daha iyisine 3 değeri verilerek devam edilir. Bir bireyin seçilme olasılığı, o bireye verilen değer tüm bireylere verilen değerler toplamına oranı kadardır. Amaç düşük uygunlukta bireylere de seçilme şansı tanmaktır, ancak bu durum çözümün daha geç yakınsamasına neden olabilir.

Turnuva Seçilimi: Topluluk içerisinde rastgele k adet (3,5,7) birey belirlenir. Bu bireylerin içerisinde uygunluk değeri en iyi olan birey seçilir. Buradaki k değeri topluluk büyüklüğü göz önünde bulundurularak belirlenmelidir. Amaç rastgelelikten yararlanarak çeşitliliği arttırmaktır.

Uygunluk	Rulet Seçilimi	Sıralı Seçilim
16	16/20	3/6
3	3/20	2/6
1	1/20	1/6

Tablo 1: Rulet Seçilimi ve Sıralı Seçilimin Karşılaştırılması

2.1.3 Çaprazlama

Çaprazlama işlemi, iyi çözümlerin farklı bölümlerini birleştirip daha iyi çözümler oluşturabilmek için kullanılır. Çaprazlamanın en basit yolu rastgele bir çaprazlama noktası belirleyip, bu noktadan önceki bölümü ilk ebeveyninden, sonraki bölümü ise diğer ebeveyninden alarak yeni bir birey oluşturmaktır. Permutasyon kodlama içeren genetik algoritmalarda sıra mantığını koruyan çaprazlama yöntemlerinin kullanılması gerekir. Başlıca çaprazlama yöntemleri tek noktalı, iki noktalı ve aritmetik çaprazlamadır. Sıra mantığını koruyan bazı çaprazlama operatörleri ise Order Crossover [32], Modified Crossover [8], Partially Mapped Crossover [16], Cycle Crossover [29], 2-quick / 2-repair [17] şeklinde belirtilebilir.

2.1.4 Mutasyon

Birey bir sonraki nesle geçirilirken kromozomu oluşturan karakter dizisinde yapılan rastgele değişikliğe mutasyon denir. Mutasyon, oluşan yeni çözümlerin önceki çözümü kopyalamasını önleyerek çeşitliliği sağlamak ve sonuca daha hızlı ulaşmak amacıyla gerçekleştirilir. Mutasyon ihtimali çok düşük (%0.01 gibi) tutulmalıdır. Aksi halde uygun çözümler de bozulacak ve Genetik Algoritmanın, çalışması sırasında problemlerle karşılaşılmasına yol açacaktır. Displacement Mutation [27], Exchange Mutation [4], Insertion Mutation [10] [27], Simple Inversion Mutation [18] [19] ve Inversion Mutation [11] [12] [24] sıra tabanlı mutasyon operatörlerinden bazılarıdır.

2.1.5 Seçkinlik (Elitizm)

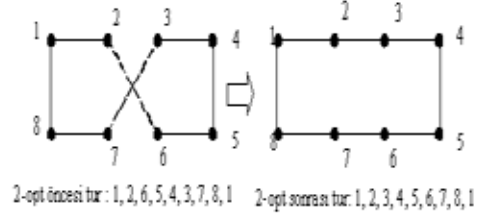
Seçim, çaprazlama ve mutasyon işlemleri sonrasında mevcut topluluğun en iyi uygunluk değerine sahip bireyi bir sonraki nesle aktarılabilir. Bunu önlemek için bu işlemlerden sonra, bir önceki topluluğun en iyi (elit) bir veya daha çok bireyi, yeni oluşturulan topluluğa doğrudan aktarılır. Seçkinlik adı verilen bu yaklaşım genetik algoritmalarda yaygın olarak kullanılmakta ve daha kaliteli çözümler için olanak sağlamaktadır.

2.2 Yerel Arama Sezgileri

Geleneksel arama yöntemlerinde yeni düğümlerin açılmasında kullanılan yöntem, başlangıç düğümüne olan uzaklığa dayanır. Oysa hedef düğümüne olan uzaklığın bilinmesi halinde yalnızca gerekli düğümler açılarak iyileştirme sağlanabilir. Hedefe olan uzaklık tahminlenebilir. Bu tahmine Sezgi (Heuristic) ya da $h(n)$ adı verilir. İyi bir sezgi, arama süresini üstelden doğrusala indirir. Örnek olarak bir karayolu ağı üzerinde rota planlama problemlerinde, bir noktadan diğer noktaya olan düz hat uzaklığı uygun bir sezgi ölçüsü sayılabilir. En iyi

öncelikli arama (Best-First Search), A*, yinelemeli derinleşen A* (Iterative-Deepening A* - IDA*), 2-opt ve 3-opt, belli başlı yerel arama sezgileri arasında sayılabilir.

GA'lar yerel arama sezgileri ile beraber kullanıldıklarında gerçekten kaliteli sonuçlar üretmektedirler [28]. En bilinenleri 2-opt ve 3-opt yöntemleri olup algoritmanın mutasyon aşamasında kullanılabilirler [26]. 2-opt sezgisi bir turdaki iki kenarı siler, böylece turu iki parçaya ayırır ve daha sonra bu parçaları ters çevirerek birleştirir (Şekil 1). Böylece genetik algoritmanın yerel minimumlara takılma olasılığı büyük ölçüde azalır.



Şekil 1: 2-opt yerel arama sezgisi

3. Java İle Geliştirilen Araç

Makale kapsamında yol planlama için geliştirdiğimiz ve gezgin satıcı problemini çözen Java tabanlı simülasyon aracı, Türkiye'nin 81 ilinin karayolları bağlantısı ve 31 ilinin hava yolları bağlantısı için mesafeye bağlı rota çözümleri vermektedir. Ayrıca kullanıcı aynı araç üzerinde etkileşimli olarak düzenleyebildiği nokta kümeleri için, kendi belirlediği problemler ile, TSPLIB kütüphanesinden alınmış bazı önemli GSP'leri de çözdürebilmektedir. Sonuçta bulunan güzergahlar ise hesaplanan uzunluklarla birlikte bilgisayar grafikleri kullanarak ekranda gösterilmektedir. İşbu araç, "http://yzgrafik.ege.edu.tr/~aybars/RouteOpt" adresinde, kullanıcıların erişimine de açılmıştır.

3.1 Aracın Kullanımı

Kullanıcılar istedikleri noktalara fare ile tıklayarak ya da noktanın koordinat değerlerini girip ŞEHİR EKLE düğmesine basarak haritaya şehir ekleyebilirler. Herhangi bir yanlış ekleme durumunda ise eklenen şehrin üzerine fare ile tıklayarak o şehri haritadan silebilirler. Bunun yanında kullanıcılar istedikleri sayıda şehirden oluşan rastgele haritalar yaratabilirler. Bunun için şehir sayısını girerek RASTGELE HARİTA OLUŞTUR düğmesine basmaları yeterli olacaktır. Araçta hazır bulunan çok sayıda TSPLIB dosyasından herhangi biri seçilip HARİTA YÜKLE düğmesine basılarak, örnek GSP'ler haritaya aktarılabilir. Yazılımın rota planlama için uygulama örneği olan turkiye81 ve thy31 haritaları da buradan yüklenmektedir. Çok büyük veya çok küçük haritalar ekrana belirli bir oranda ölçeklendirilerek aktarılır. Bu harita ölçeği ekranda gösterilmektedir. Haritaya eklenen şehirlere verilen numaralar (veya varsa şehir isimleri) Şehir Bilgisi Göster kutucuğu işaretlendiğinde görünür hale gelir.

Gezgin satıcı problemi için, kullanıcı, topluluk büyüklüğü, çaprazlama oranı, mutasyon oranı, maksimum nesil sayısı ve hedeflenen uzaklık bilgilerini girebilir. Topluluk büyüklüğü her yeni nesilde oluşturulacak olan çocuk sayısıdır. Bu parametrenin varsayılan değeri 50'dir. Toplam şehir sayısı arttığında, çözüme erken nesillerde ulaşmak için topluluk büyüklüğünün yüksek tutulması gerekir. Çaprazlama oranı kromozom çiftlerinin çaprazlamaya girme ihtimalini belirtir. Çaprazlama oranının varsayılan değeri 0.75'tir. Buna göre her kromozom çifti $\frac{3}{4}$ ihtimalle çaprazlamaya girer. Mutasyon oranı kromozomların mutasyona uğraması ihtimalidir. Bu oranın varsayılan değeri 0.00001'dir. Buna göre her kromozom $\frac{1}{100000}$ olasılıkla mutasyona uğrar. Mutasyon oranı, iyi çözümleri kaybetmeyi önlemek için düşük tutulmalıdır. Maksimum nesil sayısı, hedeflenen uzaklığa ulaşılmaması halinde genetik algoritmanın çalışmasının duracağı

neslin numarasıdır. Bu parametrenin varsayılan değeri 25'tir. Hedeflenen uzaklık, tüm şehirlerin dolaşıldığı bir tur için, eşit veya altında olması arzu edilen toplam uzunluktur. Bu değere ulaşıldığında genetik algoritma yeni nesiller üretmeyi durdurur. Hedeflenen uzaklık için varsayılan değer 100'dür. Bu parametreler isteğe bağlıdır. Eğer kullanıcılar tarafından bilgi girilmezse, uygulama varsayılan değerlerle işleyecektir. Genetik algoritmayı çalıştırmak için BAŞLAT düğmesine basılmalıdır. Sonuçta bulunan tur haritada çizdirilir, toplam uzunluk ise ekrana yazdırılır (Şekil 8). Nokta yerleşimi tamamen değiştirilmek istendiğinde veya baştan yeni bir şehir kümesi eklenmek istendiğinde ise TEMİZLE düğmesine basılır.

Uygulama Türkçe ve İngilizce dil seçimine izin vermektedir. İngilizce tercih edilmesi halinde: topluluk büyüklüğü, *population size*; çaprazlama oranı, *crossover rate*; mutasyon oranı, *mutation rate*; maksimum nesil, *maximum generation*; hedeflenen uzaklık, *aimed distance* adını alacaktır. Bu durumda koordinat vererek şehir eklemek için *ADD CITY*, rastgele harita oluşturmak için *RANDOM MAP*, hazır TSPLIB haritalarını yüklemek için *LOAD MAP*, ekranı temizlemek için *CLEAR* ve genetik algoritmayı çalıştırmak için *START* düğmesine basılmalıdır.

4. Deneysel Sonuçlar

Uyguladığımız melez yöntemin başarısını ölçmek için, bir dizi standart veri üzerinde deneyler gerçekleştirilmiştir. Tablo 2'de simetrik GSP örnek kümelerinden Berlin52, KroA100, KroA150 ve KroA200 için elde ettiğimiz sonuçlar gösterilmektedir. Deneyler 50'şer kere tekrarlanmış ve her deney için 50'şer nesil üretilmiştir. Bu deneylerdeki en iyi değer ve deneyler ortalaması kaydedilmiştir. Melez yöntemimizin, oldukça az iterasyondan sonra iyi sonuçlar ürettiği görülmektedir.

GSP Örneği	Bilinen En İyi Sonuç	En İyi
Berlin52	7544.37	7544.37
KroA100	21285.44	21285.44
KroA150	26524.86	26524.86
KroA200	29368	29375.91
Türkiye81	9954	9947
Thy31	-	4419.84

Tablo 2: TSPLIB95 örnekleri ve Uygulama örneği için elde edilen sonuçlar

Test sonuçları daha detaylı incelendiğinde, sabit topluluk büyüklüğü için, maksimum nesil parametresi belirli bir değerin üstüne çıktıktan sonra, bu parametrenin öneminin azaldığı görülmüştür. Bu nedenle maksimum nesil parametresinin belli bir noktadan sonra artırılması gerek kalmamaktadır.



Şekil 2: Türkiye81 için bulunan en iyi rota

Rota planlamanın uygulama örneği olarak belirlediğimiz Türkiye'nin 81 ilini birbirine bağlayan en kısa rota (türkiye81) Şekil 2'de görülmektedir. Şekildeki rotanın şehir-şehir doluşılma sırası ise Tablo 3'te verilmiştir.

Rota planlama için geliştirdiğimiz diğer uygulama örneği olan Türk Hava Yolları'nın uçuş yaptığı ve sivil havaalanı bulunan 31 ili bir-

birine bağlayan en kısa rotayı (thy31) bulma probleminin çözümü Şekil 3'te, illerin listesi ise Tablo 4'te gösterilmektedir.

Şehir 1 - Şehir 2	Şehir 1 - Şehir 2
İzmir - Manisa	Hakkari - Şırnak
Manisa - Balıkesir	Şırnak - Siirt
Balıkesir - Çanakkale	Siirt - Batman
Çanakkale - Edirne	Batman - Diyarbakır
Edirne - Kırklareli	Diyarbakır - Mardin
Kırklareli - Tekirdağ	Mardin - Urfa
Tekirdağ - İstanbul	Urfa - Adıyaman
İstanbul - Kocaeli	Adıyaman - Maraş
Kocaeli - Yalova	Maraş - Gaziantep
Yalova - Bursa	Gaziantep - Kilis
Bursa - Bilecik	Kilis - Hatay
Bilecik - Sakarya	Hatay - Osmaniye
Sakarya - Düzce	Osmaniye - Adana
Düzce - Bolu	Adana - İçel
Bolu - Zonguldak	İçel - Karaman
Zonguldak - Bartın	Karaman - Konya
Bartın - Karabük	Konya - Aksaray
Karabük - Kastamonu	Aksaray - Nevşehir
Kastamonu - Sinop	Nevşehir - Niğde
Sinop - Samsun	Niğde - Kayseri
Samsun - Ordu	Kayseri - Sivas
Ordu - Giresun	Sivas - Tokat
Giresun - Trabzon	Tokat - Amasya
Trabzon - Rize	Amasya - Çorum
Rize - Artvin	Çorum - Yozgat
Artvin - Ardahan	Yozgat - Kırşehir
Ardahan - Kars	Kırşehir - Kırıkkale
Kars - Iğdır	Kırıkkale - Çankırı
Iğdır - Ağrı	Çankırı - Ankara
Ağrı - Erzurum	Ankara - Eskişehir
Erzurum - Bayburt	Eskişehir - Kütahya
Bayburt - Gümüşhane	Kütahya - Afyon
Gümüşhane - Erzincan	Afyon - Uşak
Erzincan - Tunceli	Uşak - Isparta
Tunceli - Elazığ	Isparta - Burdur
Elazığ - Malatya	Burdur - Antalya
Malatya - Bingöl	Antalya - Denizli
Bingöl - Muş	Denizli - Muğla
Muş - Bitlis	Muğla - Aydın
Bitlis - Van	Aydın - İzmir
Van - Hakkari	

Tablo 3: Türkiye81 rotasındaki şehirlerin doluşım sırası

Şehir 1 - Şehir 2	Şehir 1 - Şehir 2
İzmir - Bodrum	Mardin - Batman
Bodrum - Denizli	Batman - Muş
Denizli - Antalya	Muş - Van
Antalya - Konya	Van - Ağrı
Konya - Nevşehir	Ağrı - Kars
Nevşehir - Kayseri	Kars - Erzurum
Kayseri - Adana	Erzurum - Trabzon
Adana - Hatay	Trabzon - Erzincan
Hatay - Gaziantep	Erzincan - Sivas
Gaziantep - Urfa	Sivas - Samsun
Urfa - Maraş	Samsun - Ankara
Maraş - Adıyaman	Ankara - Eskişehir
Adıyaman - Malatya	Eskişehir - İstanbul
Malatya - Elazığ	İstanbul - Bursa
Elazığ - Diyarbakır	Bursa - İzmir
Diyarbakır - Mardin	

Tablo 4: Thy31 rotasındaki şehirlerin dolaşım sırası



Şekil 3: Thy31 için bulunan en iyi rota

5. Sonuçlar

Geliştirdiğimiz web-tabanlı araç, kullanıcıların dolaşılacak tüm konumları etkileşimli olarak veya toplu halde bilgisayara girerek en uygun dolaşma sırasını ve toplam yol uzunluğunu (maliyeti) bulabilmesini sağlamaktadır. Değişik ölçeklerdeki problemler için, şehirleri dolaşma sırası bulunarak yol gösterilmekte, en uygun yol uzunluğu hesaplanmakta ve dolaşım görüntülenmektedir. Şehir konumları etkileşimli olarak girilebildiği gibi dosyalardan da okunabilmektedir.

Araç, eniyileme konusunda çalışmaya başlayanlar veya yeni eniyileme yöntemi geliştirmeye çalışan araştırmacılar için yararlı olacaktır. Araştırmacılar, kendi sonuçlarını araçtan elde edecekleri sonuçlar ile karşılaştırabileceklerdir. Değişik meslek gruplarından kişiler, günlük hayatta veya işlerinde karşılına çıkan parça toplama, parça yerleştirme ve dolaşmaya dayalı birçok problemi farklı nokta sayıları ve konumları için deneyerek en uygun sonucu alabilmektedirler. Aracımız, bir fabrikadaki parçaların toplanmasından, bir şehirdeki posta dağıtımına, baskı devrelerdeki bileşenlerin yerleştirilmesinden, tur organizasyonlarında belirtilen tüm turistik merkezlere uğramaya kadar birçok uygulamada kullanılabilir. Araç, eniyileme konusunda çalışmaya başlayanlar veya yeni eniyileme yöntemi geliştirmeye çalışan araştırmacılar için yararlı olacaktır. Araştırmacılar, kendi sonuçlarını araçtan elde edecekleri sonuçlar ile karşılaştırabileceklerdir. Değişik meslek gruplarından kişiler, günlük hayatta veya işlerinde karşılına çıkan parça toplama, parça yerleştirme ve dolaşmaya dayalı birçok problemi farklı nokta sayıları ve konumları için deneyerek en uygun sonucu alabilmektedirler. Aracımız, bir fabrikadaki parçaların toplanmasından, bir şehirdeki posta dağıtımına, baskı devrelerdeki bileşenlerin yerleştirilmesinden, tur organizasyonlarında belirtilen tüm turistik merkezlere uğramaya kadar birçok uygulamada kullanılabilir.

Uygulamayı geliştirirken Java ortamını tercih etmemizin nedeni, Java'nın nesneye dayalı programlamayı desteklemesi ve platform bağımsızlığı sunarak, aracın çok sayıda insana ulaşmasını sağlamasıdır. Araç, değişik eniyileme yöntemi geliştiren araştırmacılar için, kendi sonuçlarını genetik algoritma sonuçları ile karşılaştırma olanağı da sunmaktadır.

Programın oluşturulması sırasında, altyapıda genetik algoritmalarla birlikte 2-opt yerel arama sezgisini de kullandık. Genetik algoritmalar, birçok güçlü yönlerine rağmen tek başlarına kullanıldıklarında yeterince etkin sonuçlar vermezler. Bunun nedeni, algoritmanın yerel minimumlara takılmasıdır. 2-opt gibi yerel arama sezgileriyle genetik algoritmaların birlikte kullanıldığı melez yöntemler ise hem yerel minimumlardan kurtulmayı hem de çözümün daha kısa sürede yakınsamasını sağlarlar. Melez yöntemlerde, genetik algoritmalarla birlikte yerel arama sezgileri dışında, Karınca Kolonisi Optimizasyonu (ACO - Ant Colony Optimization) gibi yeni eniyileme tekniklerinden de yararlanılabilir [35].

Rota planlama konusunda bu tür web-tabanlı eniyileme araçlarının ve simülasyon ortamlarının geliştirilmesi, araştırmacılara test olanağı sunması ve değişik meslek gruplarındaki kişilerin ellerindeki problemlere çözüm getirmesinin yanında, genetik algoritmalar ve yerel arama sezgileri gibi yapay zeka teknikleri ile daha çok kişinin de tanışmasını sağlamaktadır. Aynı zamanda bu tekniklerin öğrenilme sürecine yardım etmekte ve önemlerinin anlaşılmasına katkıda bulunmaktadır. Çalışmamız, ayrıca endüstride rota planlama ve GSP'ye dayalı değişik problemlerin çözümünde kullanılabilen eğlenceli, kullanımı kolay, etkileşimli ve esnek yazılım araçlarının nesne yönelimli yaklaşımla geliştirilmesi için de bir örnek teşkil etmektedir.

6. Kaynaklar

- [1] Angus D, Hendtlass T, Dynamic Ant Colony Optimisation. *Applied Intelligence*. 23(1): 33-38, 2005.
- [2] Applegate D, Bixby R, Chvátal V, Cook W, On the solution of traveling salesman problems, in *Documenta Mathematica: Proc. Int. Congr. Mathematicians*, vol. 3, 1998, pp. 645–656.
- [3] Arora S., Polynomial time approximation schemes for Euclidean TSP and other geometric problems *FOCS '96: Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, IEEE Computer Society, 1996, 2
- [4] Banzhaf, W., 'The Molecular Travelling Salesman', *Biological Cybernetics*, 64, 1990, pp 7-14.
- [5] Baraglia R, Hidalgo JI, Perego R, A hybrid Heuristic for the Traveling Salesman Problem", *IEEE Trans. on Evolutionary Computation*, Vol. 5, pp.613-622, 2001.
- [6] Cevre U., Özkan B., Uğur A., Gezgini Satıcı Probleminin Genetik Algoritmalarla Eniyilmesi ve Etkileşimli Olarak İnternet Üzerinde Görselleştirilmesi, *INET-TR 2007*, 2007
- [7] Croes, G.A., 'A Method for Solving Traveling Salesman Problems', *Operations Res.* 6, 1958.
- [8] Davis, L., 'Applying Adaptive Algorithms to Epistatic Domains', *Proceedings of the International Joint Conference on Artificial Intelligence*, 1985, pp 162-164.
- [9] Fiechter L, A Parallel Tabu Search Algorithm for Large Traveling Salesman Problems, *Discrete Applied Mathematics and Combinatorial Operations Research and Computer Science*, 51, pp. 243-267, 1994.
- [10] Fogel, D.B., 'An Evolutionary Approach to the Travelling Salesman Problems', *Biological Cybernetics*, 60, 1988, pp 139-144.
- [11] Fogel, D.B., 'Empirical Estimation of the Computation Required to Discover Approximate Solutions to the Travelling Salesman Problem Using Evolutionary Programming', *Proceedings of 2nd Annual Conference on Evolutionary Programming*, 1993, pp 56-61.
- [12] Fogel, D.B., 'Applying Evolutionary Programming to Selected Travelling Salesman Problems', *Cybernetics and Systems: An International Journal*, 24, 1993, pp 27-36.
- [13] Gambardella L. M., Dorigo M., Solving symmetric and asymmetric TSPs by ant colonies *Evolutionary Computation*, 1996., *Proceedings of IEEE International Conference on*, 1996, 622-627
- [14] Garey MR, Johnson DS, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, 1979.

- [15] Goldberg DE, Genetic Algorithms in Search, Optimization and Machine Learning. Reading, MA: Addison-Wesley, 1989.
- [16] Goldberg DE, Lingle Jr. R, Alleles, Loci and the TSP, Proceedings of the First International Conference on Genetic Algorithms and Their Applications, Lawrence Erlbaum, Hillsdale, New Jersey, pp. 154-159, 1985.
- [17] Gorges-Schleuter, M., ‘ASPARAGOS An Asynchronous Parallel Genetic Optimization Strategy’, Proceedings of the Third International Conference on Genetic Algorithms, 1989, pp 422-427.
- [18] Grefenstette J, Gopal R, Rosmaita B, Gucht DV, Genetic Algorithms for the TSP, in Grefenstette, J.J. (Ed.), Proceedings of the First International Conference on Genetic Algorithms and Their Applications, Lawrence Erlbaum, Hillsdale, New Jersey, pp. 160-165, 1985.
- [19] Holland, J., ‘Adaptation in Natural and Artificial Systems’, Ann Arbor USA, University of Michigan, 1975.
- [20] Jin HD, Leung KS, Wong ML, Xu ZB, An efficient self-organizing map designed by genetic algorithms for the traveling salesman problem, IEEE Trans. Syst., Man, Cyber.—Part B: Cyber. 33 (6) (2003) 877–887.
- [21] Johnson DS, McGeoch LA, The traveling salesman problem: A case study in local optimization, in Local Search in Combinatorial Optimization. Hoboken, NJ: Wiley, 1997, pp. 215–310.
- [22] Johnson DS, McGeoch LA, Experimental Analysis of Heuristics for the STSP. The Traveling Salesman Problem and its Variations, pages 369-443. Kluwer Academic Publishers, 2002.
- [23] Laarhoven PV, Aarts EHL, Simulated Annealing: Theory and Applications. Kluwer Academic Publishers, 1987.
- [24] Larranaga P, Kuijpers CMH, Murga RH, Inza I, Dizdarevic S, Genetic algorithms for the travelling salesman problem: A review of representations and operators, Artificial Intelligence Review, 13, pp. 129—170, 1999.
- [25] Li Y, Gong S, Dynamic Ant Colony Optimisation for TSP. International Journal of Advanced Manufacturing Technology, 22(7-8):528-533, 2003.
- [26] Marinakis Y, Migdalas A, Pardalos PM, A Hybrid Genetic-GRASP Algorithm Using Lagrangean Relaxation for the Traveling Salesman Problem, J. Comb. Optim. 10(4): 311-326, 2005.
- [27] Michalewicz, Z., ‘Genetic Algorithms + Data Structures = Evolution Programs’, Berlin Germany, Springer Verlag, 1992.
- [28] Nguyen HD, Yoshihara I, Yamamori K, Yasunaga M, Implementation of an Effective Hybrid GA for Large-Scale Traveling Salesman Problems, IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics, vol. 37, pp. 92-99, 2007.
- [29] Oliver IM, Smith DJ, Holland JRC, A Study of Permutation Crossover Operators on the TSP, Genetic Algorithms and Their Applications: Proceedings of the Second International Conference, Lawrence Erlbaum, Hillsdale, New Jersey, pp. 224-230, 1987.
- [30] Padberg M, Rinaldi G, Optimization of a 532-city symmetric genetic traveling salesman problem by branch and cut, Oper. Res. Lett., vol. 6, no. 1, pp. 1–7, 1987.

- [31] Sengoku, H., Yoshihara, I., 'A Fast TSP Solution using Genetic Algorithm', 1998.
- [32] Syswerda, G., 'Schedule Optimization Using Genetic Algorithms', Handbook of Genetic Algorithms, New York NY, Van Nostrand Reinhold, 1991, pp 350-372.
- [33] Tsai HK, Yang JM, Tsai YF, Kao CY, Some issues of designing genetic algorithms for traveling salesman problems, Soft Computing, volume 8, pages 689-697, 2004.
- [34] Tsujimura M., Entropy-based genetic algorithm for solving TSP Knowledge-Based Intelligent Electronic Systems, 1998. Proceedings KES '98. 1998 Second International Conference on, 1998, 2, 285-290
- [35] Uğur A., Aydın D., Ant System Algoritmasının Java ile Görselleştirilmesi, Akademik Bilişim 2006, 2006