Asimetrik Gezgin Satıcı Problemi İçin Bir Evrimsel Strateji Algoritması

Korhan Karabulut

Yaşar Üniversitesi, Mühendislik Fakültesi, Yazılım Mühendisliği Bölümü, İzmir, +90 232 570 70 70, korhan.karabulut@yasar.edu.tr

Geliş / Received: 4 Ağustos (August) 2016
Kabul / Accepted: 14 Aralık (December) 2016
DOI: 10.18466/cbayarfbe.280728

Özet

Gezgin satıcı problemi NP-zor sınıfındaki en bilinen problemlerden birisidir. En temel hali bile birçok pratik problemin modellenmesi için kullanılabildiği için üzerinde birçok akademik çalışma yapılmaktadır. Asimetrik gezgin satıcı problemi özellikle büyük şehirlerde ortaya çıkan tek yönlü yollar, trafik sıkışıklığı gibi durumları da problem tanımına eklenebilmesine izin verir. Bu nedenle, simetrinin her zaman geçerli olmadığı gerçek hayat problemlerinin modellenmesinde yaygın olarak kullanılmaktadır. Bu çalışmada asimetrik gezgin satıcı probleminin çözümü için bir evrimsel strateji algoritması önerilmektedir. Geliştirilen evrimsel strateji algoritması yeni çözümlerin üretilmesi için boz ve yeniden yap algoritmasını kullanmaktadır. Bu algoritmada belirlenen sayıda çözüm bileşeni rasgele seçilerek çözümden çıkartılır. Sonraki adımda, çıkartılan bileşenler uygunluk değerini en küçük yapacak biçimde çözüme tekrar eklenir. Boz ve yeniden yap algoritması, dolayısı ile evrimsel strateji algoritması için önemli bir parametre olan bozma boyutu, evrimsel stratejisi algoritmasının özuyarlama özelliği kullanılarak güncellenmektedir. Özuyarlama özelliği algoritmanın iyi sonuç üreten parametre değerini öğrenmesini ve arama süresince aramanın o anki durumunun gerektirdiği biçimde değiştirilmesini sağlamaktadır. Elde edilen yeni çözümlerin daha da iyileştirilmesi için yerel arama aşamasında 3-opt algoritması kullanılmaktadır. Geliştirilen evrimsel strateji algoritmasının başarımının test edilmesi için literatürde en çok kullanılan problem kümesi olan TSPLIB kütüphanesi problemleri kullanılmış ve elde edilen sonuçlar sunulmuştur. Geliştirilen algoritma problemlerin optimum değerlerini çoğu zaman elde etmiş, elde edemediği durumlarda da optimum değerden sapma en çok %1 olarak gerçekleşmiştir. Geliştirilen algoritmanın asimetrik gezgin satıcı probleminin kısa sürede etkin biçimde çözülmesi için kullanılabilecek bir yöntem olduğu gösterilmiştir.

Anahtar Kelimeler — 3-opt algoritması, asimetrik gezgin satıcı problemi, boz ve yeniden yap algoritması, evrimsel strateji algoritması, özuyarlama

An Evolutionary Strategy Algorithm for the Asymmetric Travelling Salesman Problem

Abstract

Travelling salesman problem is one of the best known NP-hard problems. It attracts many researchers, since it can be used to model many practical problems even in its most basic form. Asymmetric travelling salesman problem allows adding situations like one way roads, traffic congestion, that are especially common in large cities to the problem definition. For this reason, it is used to model real life problems where symmetry does not always hold. An evolutionary strategy algorithm for solving the asymmetric travelling salesman problem is proposed in this paper. The evolutionary strategy algorithm uses ruin and recreate algorithm to generate new solutions. In this algorithm, a given number of solution components are selected randomly and removed from the solution. In the next stage, removed solution components are reinserted into the solution in such a way that minimizes the fitness value. The ruin size, which is an

important parameter for the ruin and recreate algorithm, hence also for the evolutionary strategy algorithm, is updated using the self-adapting feature of the evolutionary strategy algorithm. Self-adaption allows the algorithm to learn the parameter values that produce good results and change these values according the requirements in the current state of the search. 3-opt algorithm is used in the local search phase in order to further improve the newly generated solutions. TSPLIB problem instances which are the most widely used problem set in the literature has been used to test the performance of the developed algorithm and the results are presented. The developed algorithm is able to find the optimum solutions most of the time, and in cases it cannot find the optimum values, the percent deviation from the optimum values is at most 1%. It is shown that, the proposed algorithm is an effective method to solve the asymmetric travelling salesman problem.

Keywords — 3-opt algorithm, asymmetric travelling salesman problem, evolutionary strategy algorithm, ruin and recreate algorithm, self-adaption

1 Giriş

Gezgin satıcı problemi (GSP) iyi bilinen ve üzerinde en çok çalışılan eniyileme problemlerindendir. GSPde amaç, verilen bir yönsüz çizgedeki n tane düğümün her birisini sadece bir kez ziyaret edecek biçimde en az maliyete sahip olan turu bulmaktır. Tur başladığı düğümde sonlanmalıdır. GSP NP-zor bir problemdir ve rotalama, çizelgeleme, planlama, lojistik, baskılı devre kartlarında delik açılması, mikroçip üretimi, DNA dizileme gibi birçok pratik problemin modellenmesinde kullanılmıştır [1].

Klasik GSPye ek olarak, pratik uygulamalarda ortaya çıkabilen farklı senaryoları da dikkate almak üzere GSPnin birçok türü de tanımlanmıştır. Bu türlerden en çok bilinenlerden birisi iki düğüm arasındaki maliyetin ya da uzaklığın farklı olabildiği, hatta bazen düğümler arasında sadece tek yönlü bağlantının olduğu asimetrik GSP (AGSP)dir. AGSPyi temsil etmek için kullanılan çizge GSPden farklı olarak yönlüdür.

AGSPnin araç rotalama, kurulum zamanı içeren çizelgeleme, dağıtım gibi önemli uygulama alanları vardır. AGSP pratik problemlerde ortaya çıkan belirli bir yöndeki trafik sıkışıklığı, tek yönlü yollar, hava ve yol koşulları gibi durumların dikkate alınmasına izin verir. AGSP, GSPye benzer görünse de, Rodríguez ve Ruiz [2] asimetrinin çözüm yöntemleri üzerindeki etkisini araştırmış ve çözümü daha zor hale getirdiğini göstermiştir.

AGSP de GSP gibi NP-zor sınıfındadır, dolayısı ile kesin yöntemler sadece küçük problem boyutlarında kullanılabilmektedir. Bu nedenle literatürde AGSPnin çözümü için önerilen birçok sezgisel yöntem bulunmaktadır. Freisleben ve Merz [3] DPX (Distance Preserving Crossover) adını verdikleri yeni bir çaprazlama yöntemi ve yerel eniyileme için Lin-Kerninghan algoritması kullanan melez bir genetik algoritma önermiştir. Daha sonra bu yöntemi daha iyi hale getirmek için yerel arama adımında 3-opt ve 4-opt kullandıkları yeni bir melez genetik algoritma geliştirmişlerdir [4]. Nagata ve Soler [5] daha önce simetrik GSP için geliştirilmiş olan EAX (Edge Assembly Crossover) yöntemini [6] AGSP'ye adapte etmiş ve bu operatörü yerel arama adımında 3-opt kullandıkları bir melez genetik algoritma içerisinde kullanmışlardır. Xing ve ark. [7], geliştirilmiş (improved) genetik algoritma ve eniyileme stratejilerini birleştiren melez bir yöntem önermişlerdir. Geliştirilen diğer sezgisel yöntemlere uç (extremal) eniyileme algoritması [8], karınca kolonisi eniyilemesi [9] ve yarasa algoritması [10] örnek olarak verilebilir.

1.1 AGSPnin formülasyonu

AGSPnin formal olarak tanımı aşağıda verilmiştir. AGSP, $G = (V, A)$ biçiminde yönlü bir çizge olarak tanımlanabilir. Burada $V = \{v_1, v_2, \dots, v_n\}$ her biri bir şehire karşılık gelecek biçimde bir köşeler kümesi, $A = \{(v_i, v_j) : v_i, v_j \in V, i \neq j\}$ ise her bir köşe çifti arasındaki bağlantıyı temsil eden yönlü kenarlar kümesidir. Her bir kenara karşılık gelen c_{ij} biçiminde gösterilen bir maliyet vardır. Simetrik GSPde $c_{ij} = c_{ji}$ iken, AGSPde bu kural şart değildir, dolayısıyla genel olarak $c_{ij} \neq c_{ji}$ kabul edilir. Bazı durumlarda bir düğümden diğerine geçiş olmasına rağmen tersi yönde bir bağlantı olmayabilir, bu durum $c_{ij} = \infty$ biçimde gösterilir. Tüm turlar depo olarak belirlenen düğümden başlamak ve depoda bitmek zorundadır. Bir tur permütasyonu π olarak gösterilirse, AGSPde

amaç $f(\pi) = \sum_{k=0}^{n} c(\pi_k, \pi_{k+1}) + c(\pi_n, 0)$ biçiminde tanımlanan tur maliyetini en küçük yapmaktır.

1.2 Evrimsel Stratejiler

Evrimsel stratejiler (ES) 1960 ve 1970lerde Almanya'da ortaya çıkan, Rechenberg [11] ve Schwefel [12] tarafından geliştirilen bir meta sezgisel eniyileme yöntemidir. Diğer evrimsel algoritmalar gibi, popülasyondaki bireylere bir döngü içerisinde genetik işleçler uygulayarak üretilen bireylerin seçilimden geçirilmesi temeline dayanır. ES, hem gerçek parametre, hem de ayrık eniyileme problemlerinde kullanılabilir. Tarihsel olarak ES genetik farklılık oluşturmak, yani yeni bireyler üretmek için, sadece değişinim (mutasyon) işlecini kullanır. Gerçek parametreli problemlerde değişinim normal dağılımdan oluşturulan rasgele bir değerin her bir problem bileşenine eklenmesi ile gerçekleştirilir.

Çok bireyli bir ESde toplam μ adet birey bulunur. Her bir nesilde λ adet yeni birey oluşturulur. ESde çocukları oluşturacak ebeveynler genetik algoritmalardaki gibi uygunluk değerlerine orantılı olarak değil, rasgele olarak seçilir. Bir sonraki nesilde yer alacak μ adet bireyin seçilimi için iki temel yöntem vardır: artı (plus) ve virgül (comma) stratejileri [13]. Artı stratejisinde her bir nesilde ebeveynler ve onlardan oluşturulan çocuklar birlikte değerlendirilerek en kötü bireyler (yeni ya da eski) silinir. Yani, $\mu + \lambda$ tane bireyden en kötü λ birey popülasyondan silinir. Virgül stratejisinde ise seçilim sadece çocuklar arasında gerçekleşir; λ tane çocuk arasından μ tanesi seçilir. Diğer çocuklar ve tüm ebeveynler popülasyondan silinir.

ESnin en önemli özelliklerden birisi özuyarlama (self-adaption)dır [13]. Özuyarlama, aramanın durumuna göre değişinim boyutunun otomatik olarak değiştirilmesi özelliğidir. ESde özuyarlama, değişinim boyutu gibi strateji parametrelerinin bireyleri temsil etmek için kullanılan genotipin bir parçası olması ve nesiller içerisinde bireyle birlikte taşınması yöntemine dayanır. Burada varsayım, iyi bir bireyi oluşturan parametrenin de iyi olduğu ve bir sonraki nesile aktarılması gerektiğidir.

1.2 Boz ve Yeniden Yap (Ruin and Recreate) Algoritması

Boz ve Yeniden Yap algoritması Schrimpf ve ark. [14] tarafından geliştirilen basit ancak etkili bir yöntemdir. Var olan bir çözümün bir bölümünün bozulması yani çözümden çıkartılması ve çıkartılan bileşenlerin çözüme yeniden eklenerek yeni bir çözüm oluşturulması esasına dayanan genel amaçlı bir yöntemdir. Schrimpf ve ark. [14] tarafından GSP, zaman pencereli araç rotalama problemi ve ağ eniyilemesi problemlerinin çözümüne uygulanmıştır. Bu yöntem daha sonra karesel atama problemi (quadratic assignment problem) [15], akış tipi çizelgeleme ve kutu paketleme problemi [16] gibi başka problemlerin çözümü için de kullanılmıştır.

Boz ve Yeniden Yap algoritmasında BB (Bozma Boyutu - Ruin Size) tane çözüm bileşeni rasgele olarak seçilerek çözümden çıkartılır. Daha sonra çıkartılan her bir bileşen sıra ile kısmı çözüme kısmi uygunluk değerini eniyileyecek biçimde eklenir. Bu işlem tekrar tam bir sonuç elde edilene kadar devam eder.

2 Asimetrik Gezgin Satıcı Problemi İçin Önerilen Evrimsel Strateji Algoritması

AGSPnin çözümü için önerilen ESnin detayları aşağıda belirtilmiştir.

2.1 Çözüm Gösterimi

Geliştirilen ES çözümlerin temsil edilmesi için permütasyon gösterimi kullanmaktadır. n tane şehir için n uzunluğunda bir tamsayı vektörü kullanılmaktadır. Her bir şehir için tekrar etmeyecek biçimde 1 ile n arasında bir tamsayı değer atanır. Depo permütasyonda kullanılmamaktadır. Uygunluk değeri hesaplanırken depodan permütasyondaki ilk şehire gidiş ve permütasyondaki son şehirden depoya dönüş eklenmektedir.

Özuyarlama için yeni bireylerin oluşturulmasında kullanılan boz ve yeniden yap algoritmasının bozma boyutu (BB) parametresi yani bozma aşamasında kaç adet şehirin permütasyondan çıkartılacağı bilgisi de ilgili çözüm içerisinde tutulmaktadır. Bu parametre de çözüm içerisinde aşağıda detayı verildiği biçimde permütasyonla birlikte değişinime uğramaktadır. Burada temel varsayım, iyi bir bireyin oluşmasını sağlayan bozma boyutunun da iyi olması, dolayısı ile iyi parametrelerin ait oldukları iyi bireylerle birlikte uzun süre popülasyonda kalabilmesi ve yeni çözümler üretmek için kullanılmasıdır. İyi çözüm üretmeyen parametre değerleri de kötü bireylerle birlikte popülasyondan silinir.

2.2 İlk Popülasyon

İlk popülasyonun oluşturulması aşamasında ilk olarak 3 tane birey çizelgeleme problemlerinde toplam tamamlanma zamanı eniyileme kriteri için bilinen en

iyi sezgisel algoritmalardan olan NEH [17] algoritmasından esinlenerek oluşturulmuştur. NEH algoritmasında çizelgelenecek tüm işlerin toplam işlem süreleri toplanarak, bu süreler büyükten küçüğe doğru sıralanır. Daha sonra oluşan sıradaki toplam işlem süresi en uzun olan ilk iki iş toplam tamamlanma süreleri en küçük olacak biçimde kısmi permütasyona atanır. Sonraki adımda, her bir iş oluşan sıralamaya göre ele alınır ve kısmi permütasyondaki olası tüm pozisyonlara eklenir. İlgili iş permütasyonda kısmi uygunluk değerini minimize eden pozisyona atanır. Tüm işler bu şekilde permütasyona yerleştirilir ve bu algoritmanın sonunda tam bir çözüm elde edilir.

Önerilen sezgisel algoritma NEH algoritmasında işlerin toplam tamamlanma zamanına göre büyükten küçüğe sıralanması yönteminden esinlenerek, şehirden depoya ve depodan şehire olan uzaklıklarına göre en uzaktan en yakına göre sıralanması ve daha sonra NEH algoritmasındaki gibi şehirlerin sıra ile ele alınarak adım adım bir permütasyon oluşturulmasına dayanır.

Geliştirilen bu sezgisel yöntemde üç farklı tür sıralama yapılmaktadır. Bunlardan ilki depodan ilgili şehire olan uzaklık, ikincisi şehirden depoya ve depodan şehire olan uzaklığın ortalaması ve üçüncüsü ise şehirden depoya ve depodan şehire olan uzaklığın toplamıdır.

İlk popülasyondaki diğer bireyler ise oluşturulan bu üç çözümden bir tanesi rasgele seçilerek, bu çözüm içerisinden rasgele olarak seçilen şehirlerin karşılıklı olarak permütasyondaki konumlarının değiştirilmesi biçiminde küçük değişimler uygulanması ile oluşturmaktadır. Dolayısı ile ilk popülasyondaki diğer bireyler bu üç çözümden birisi temel alınarak üretilmektedir.

2.3 Yerel Arama

İlk popülasyonda üretilen bireylere ve ESde Boz ve Yeniden Yap algoritması ile üretilen yeni bireylere, uygunluk fonksiyonu değerini daha da iyileştirebilmek için 3-opt algoritması uygulanmaktadır. 2-opt gibi turda yön değişimi yapan algoritmalar AGSPde iyi çalışmamaktadır. Bu nedenle bu çalışmada 3-opt algoritması kullanılmaktadır. 3-opt algoritmasında üç tane köşe kırılarak yeniden bağlanmaktadır. Yeniden bağlanma birden fazla biçimde olabilir [18]. Bu bağlama yöntemlerinden bazıları yön değişimine neden olmaktadır. Bu çalışmada kullanılan yeniden bağlama yöntemi Nagata ve Soler [5] tarafından da kullanılan, yön değişikliğe neden olmayacak yeniden bağlama biçimidir. Bu yöntemde $(v_1, v_2), (v_3, v_4), (v_5, v_6)$ kırılacak üç köşeyi temsil ederse, yeniden bağlanma $(v_1, v_4), (v_3, v_6), (v_5, v_2)$ biçiminde yapılmaktadır.

Çalışmada kullanılan 3-opt algoritması işlemci zamanı bakımından maliyetli bir algoritmadır. Bu nedenle tüm olası değişikliklerin incelenerek içlerinden en iyisinin seçilmesini gerektiren en iyi geliştirme (best improvement) kuralı yerine, daha az komşuluğun incelenmesini gerektiren ilk geliştirme (first improvement) kuralı tercih edilmiştir. Bu yöntemde herhangi bir iyileştirme bulunduğunda diğer olası değişimler incelenmez.

2.4 Seçilim Yöntemi

ESnin kombinatorel eniyileme gibi ayrık problemlere uygulanmasında artı stratejisinin kullanılması tavsiye edilmektedir [19,20]. Bu çalışmada geliştirilen ES algoritmasında da artı stratejisi kullanılmıştır.

2.5 ES Algoritmasının Detayı

Geliştirilen algoritmanın sözde kodu Şekil 1'de verilmiştir.

Yordam *ES*
  İlk popülasyonu 2.2 *de belirtildiği gibi oluştur*
  Popülasyonu değerlendir ve S_{eniyi}*'yi bul*
  Tüm bireylere 3*Opt uygula*
  Bitiş Koşulu Sağlanmadığı Sürece ***tekrarla***
    i ***için*** 1 *den* 7
      $*$ *μye kadar* ***tekrarla***
        λ_i
        $\lambda_{i_{bb}} = \lambda_{i_{bb}}.e^{\tau \cdot N(0,1)}$
        eğer $\lambda_{i_{bb}} < min_{bb}$ ***ise*** $\lambda_{i_{bb}} = min_{bb}$
        eğer $\lambda_{i_{bb}} > maks_{bb}$ ***ise*** $\lambda_{i_{bb}} = maks_{bb}$
        $\lambda_i = BozYenidenYap(\lambda_i, \lambda_{i_{bb}})$
        $\lambda_i = 3Opt(\lambda_i)$
        eğer $\lambda_i < S_{eniyi}$ ***ise*** $S_{eniyi} = \lambda_i$
        λ_i*yi popülasyona ekle*
    tekrar sonu
  Popülasyondan en iyi μ bireyi seç ve diğerlerini sil
  Çeşitlilik kontrolü yap
  tekrar sonu

Yordam Sonu

Şekil 1. Önerilen ES algoritması

Algoritma ilk olarak bölüm 2.2de anlatıldığı biçimde ilk popülasyonun oluşturulması ile başlar. Daha sonra ilk popülasyondaki tüm bireylere bölüm 2.3de anlatılan 3-opt algoritması ile yerel arama uygulanır.

Popülasyondaki tüm bireylerin uygunluk değerleri hesaplanır ve popülasyondaki en iyi birey kaydedilir. Daha sonra algoritmanın bitiş kriteri sağlanmadığı sürece döngü içerisinde ilk olarak ebeveynler arasından bir birey rasgele olarak seçilir. İlk olarak bozma boyutu parametresi $bb' = bb \cdot e^{\tau \cdot N(0,1)}$ denklemi ile güncellenir. Burada τ öğrenme oranı, $N(0,1)$ ortalaması 0, standart sapması 1 olan normal dağılımdan rasgele seçilen bir sayıdır. Bir sonraki adımda yeni bozma boyutunun önceden belirlenmiş olan en küçük ve en büyük bozma boyutu değerlerinin arasında kalması sağlanır. Daha sonra yeni birey bu yeni oluşturulan bozma boyutu parametresini kullanan boz ve yeniden yap algoritması ile elde edilir. Sonraki adımda yeni oluşturulan bireye 3-opt algoritması ile yerel arama uygulanır. Bu işlemden sonra elde edilen birey popülasyona eklenir. Eğer yeni birey o ana kadar bilinen en iyi bireyden daha iyi ise en iyi birey olarak kaydedilir.

Yeni birey oluşturma işlemi toplam $7 * \mu$ adet yeni birey oluşturulana kadar devam eder. Daha sonra artı stratejisi kullanılarak popülasyondaki birey sayısı tekrar μ olacak biçimde seçilim aşaması işletilir. Bu aşamada ebeveynler ve yeni bireyler dahil olmak üzere toplam $8 * \mu$ birey uygunluk değerlerine göre sıralanarak içlerinden en iyi μ tane birey seçilir ve diğer bireyler popülasyondan silinir.

Popülasyon tabanlı eniyileme yöntemlerinde popülasyon içerisinde çeşitliliğin korunması iyi sonuçlar bulmak için önemlidir. Eğer tüm bireyler aynı bireyin kopyası ise ya da birbirlerine çok benzer ise algoritmanın yerel optimuma takılma olasılığı yüksektir. Bu nedenle, önerilen ES algoritmasında yeni popülasyon oluşturulduktan sonra çeşitlilik kontrolü yapılmakta ve gerekirse, çeşitliliği artırmak amacıyla bireyler üzerinde küçük değişiklikler yapılmaktadır.

Popülasyondaki çeşitliliğin korunabilmesi için yeni seçilen popülasyondaki bireylerin uygunluk değerlerinin standart sapması hesaplanır. Eğer standart sapma değeri 1den küçükse bireyleri farklılaştırmak için popülasyon içerisinden popülasyon boyutunun yarısı kadar birey rasgele seçilir. Seçilen bireylerde yine rasgele olarak seçilen iki şehrin karşılıklı yer değiştirilmesi işlemi uygulanır. Geliştirilen çeşitlilik kontrol algoritması Şekil 2'de gösterilmektedir.

Yordam *Çeşitlilik Kontrol*ü
  Popülasyondaki bireylerin uygunluk değerlerinin standart sapmasını hesapla
  eğer *standart sapma* $<$ 1 ***ise***
    i ***için*** 1 *den* μ
    /2ye kadar ***tekrarla***
      μ_i
      $\mu_i' = YerDeğiştir(\mu_i)$
    tekrar sonu
  ise sonu
Yordam Sonu

Şekil 2. Çeşitlilik Kontrol Algoritması

3 Deneysel Çalışma

Önerilen ES algoritmasının başarımını ölçmek için algoritma C++ programlama dili ile kodlanmış, 2.66 GHz Intel Core2 Q9400 işlemci ve 3 GB belleğe sahip bir bilgisayarda çalıştırılmıştır. Algoritmanın çalıştırılmasında kullanılan parametreler Çizelge 1'de verilmiştir.

Çizelge 1. ES algoritmasında kullanılan parametreler

Parametre	Açıklama	Kullanılan Değer
μ	popülasyon boyutu	50
τ	öğrenme oranı	1
min_{bb}	en küçük bozma boyutu	2
$maks_{bb}$	en büyük bozma boyutu	n (Problemdeki şehir sayısı)

Deneylerde TSPLIB (http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95)'den alınan 27 adet AGSP problemi kullanılmıştır. Bu problemlerdeki toplam şehir sayıları 17 ile 443 arasında değişmektedir. Problemlerin toplam şehir sayısı ve bu problem için hesaplanan optimum değerler Çizelge 2'de gösterilmiştir.

Çizelge 2. Deneylerde kullanılan TSPLIB AGSP problemleri

Problem Adı	Şehir Sayısı	Optimum Değer
br17	17	39
ft53	53	6905
ft70	70	38673

ftv33	34	1286
ftv35	36	1473
ftv38	39	1530
ftv44	45	1613
ftv47	48	1776
ftv55	56	1608
ftv64	65	1839
ftv70	71	1950
ftv90	91	1579
ftv100	101	1788
ftv110	111	1958
ftv120	121	2166
ftv130	131	2307
ftv140	141	2420
ftv150	151	2611
ftv160	161	2683
ftv170	171	2755
kro124p	100	36230
p43	43	5620
rbg323	323	1326
rbg358	358	1163
rbg403	403	2465
rbg443	443	2720
ry48p	48	14422

Algoritmanın sonlanma kriteri olarak optimum değerin bulunması ya da optimum bulanamadığı durumda maksimum çalışma süresi olan 60 saniyenin tamamlanması kullanılmıştır. Problemlerin optimum değeri bilindiği için başarı kriteri olarak algoritmanın optimum değere ulaşma sayısı kullanılmıştır. Ek olarak optimum değerden olan ortalama yüzde uzaklık ve ortalama çalışma süresi de çizelgede gösterilmektedir. Ortalama göreceli yüzde sapma (hata) $\sum_{i=1}^{R} \frac{R_i - Optimum}{Optimum} / R * 100$ formülü ile hesaplanmıştır. Burada R_i algoritmanın ilgili problemdeki i'inci çalışmasında bulunan en küçük uygunluk değeri, $Optimum$ problemin optimum uygunluk değeri, R ise toplam çalıştırılma sayısıdır. Geliştirilen ES algoritması her bir problem için 50 kez çalıştırılmıştır. Çizelgedeki ortalama süre optimum sonucun bulunma ya da en iyi sonucun bulunması sürelerinin ortalamasıdır. Elde edilen sonuçlar Çizelge 3'de gösterilmiştir.

Çizelge 3. ES algoritmasının TSPLIB AGSP problemleri için elde ettiği sonuçlar

Problem Adı	Başarı sayısı	Ortalama yüzde sapma	Ortalama süre (saniye)
br17	50	0.00	0.00
ft53	45	0.01	8.32
ft70	38	0.02	10.88
ftv33	50	0.00	0.02
ftv35	50	0.00	0.20
ftv38	50	0.00	0.40
ftv44	50	0.00	0.24
ftv47	50	0.00	0.32
ftv55	50	0.00	1.32
ftv64	48	0.03	12.94
ftv70	27	0.13	7.90
ftv90	41	0.07	3.16
ftv100	36	0.09	4.76
ftv110	36	0.09	9.12
ftv120	10	0.67	19.58
ftv130	11	0.24	16.88
ftv140	4	0.47	22.30
ftv150	13	0.99	35.42
ftv160	7	1.01	35.86
ftv170	7	1.03	45.48
kro124p	26	0.01	14.16
p43	50	0.00	0.10
rbg323	50	0.00	5.70
rbg358	50	0.00	16.46
rbg403	50	0.00	8.30
rbg443	50	0.00	11.38
ry48p	28	0.26	4.90
Ortalama	**36.19**	**0.19**	**10.97**

Geliştirilen ES algoritması 27 problemden 12sinde 50 tekrarın tümünde optimum sonuca ulaşabilmiştir. Bulunan optimum sonuç sayılarının ortalaması 36.19dur. Dolayısı ile algoritma toplam 1350 tekrarın %72sinde optimum sonuca ulaşabilmiştir. Optimum sonuca ulaşamadığı durumlarda ise ortalama sapma en çok %1.03, ortalamada ise %0.19 olarak gerçekleşmiştir. Algoritma optimum sonuca ulaşamasa bile optimum değere çok yakın bir değer elde etmiştir. Ayrıca bu sonuçları elde etmek için gereken ortalama süre 10.97 saniye olmuştur.

Çizelge 4 ve Çizelge 5de ES algoritmasının literatürde güncel bir çalışmada sunulan gelişmiş yarasa algoritması (improved bat algorithm – IBA) [10] ile karşılaştırılması verilmektedir. Osaba ve ark. [10] ilgili çalışmada 15 tane TSPLIB probleminin sonuçlarını

raporlamışlardır, dolayısı ile çizelgelerde sadece bu problemlere ait sonuçlar karşılaştırılmaktadır. Çizelgelerde algoritmaların elde ettiği uygunluk değerlerinin ortalamaları, en iyi değerleri, standart sapmaları ve ortalama süreleri saniye olarak verilmiştir. Elde edilen sonuçların karşılaştırılması sonucu ESnin başarımının IBA[10]'dan daha iyi olduğu görülmektedir.

Çizelge 4. IBA [10] algoritmasının TSPLIB AGSP problemleri için elde ettiği sonuçlar

Problem Adı	Ortalama	En iyi	Standart sapma	Ortalama süre
br17	39.0	39	0.0	0.2
ft53	7294.1	7001	196.9	6.5
ft70	40309.7	39901	237.2	8.2
ftv33	1318.1	1286	25.7	2.2
ftv35	1493.7	1473	8.9	2.5
ftv38	1562.0	1530	13.7	3.1
ftv44	1683.7	1613	27.2	5.0
ftv47	1863.6	1796	39.3	4.7
ftv55	1737.5	1608	50.5	6.9
ftv64	1999.2	1879	68.2	7.2
ftv70	2233.2	2111	48.8	8.1
kro124p	39213.7	37538	947.5	15.4
p43	5620.0	5620	0.0	3.0
rbg323	1640.9	1615	30.4	243.6
ry48p	14544.8	14422	79.7	4.2

Çizelge 5. ES algoritmasının TSPLIB AGSP problemleri için elde ettiği sonuçlar

Problem Adı	Ortalama	En iyi	Standart sapma	Ort. süre
br17	39.0	39	0.0	0.00
ft53	6905.5	6905	1.7	8.32
ft70	38681.2	38673	14.5	10.88
ftv33	1286.0	1286	0.0	0.02
ftv35	1473.0	1473	0.0	0.20
ftv38	1530.0	1530	0.0	0.40
ftv44	1613.0	1613	0.0	0.24
ftv47	1776.0	1776	0.0	0.32
ftv55	1608.0	1608	0.0	1.32
ftv64	1839.6	1839	2.9	12.94
ftv70	1952.4	1950	3.0	7.90
kro124p	39213.7	36235.3	5.5	14.16
p43	5620.0	5620	0.0	0.0
rbg323	1326.0	1326	0.0	5.70
ry48p	14458.9	14422	41.7	4.90

Elde edilen sonuçlar önerilen algoritmanın optimum ya da optimuma çok yakın sonuçları güvenilir ve hızlı bir biçimde bulabildiğini göstermektedir.

4 Sonuç

Bu çalışmada AGSP probleminin çözümü için bir ES algoritması sunulmuştur. Sunulan ES algoritmasının başarımının test edilmesi için TSPLIB kütüphanesinden alınan optimum değerleri bilinen problemler kullanılmıştır. Elde edilen sonuçlar önerilen algoritmanın ilgili problemlerin çözümü için çok etkin bir yöntem olduğunu göstermektedir. AGSP doğrudan ya da dolaylı olarak birçok gerçek hayat probleminin çözümünde kullanılabilir. Dolayısı ile bu çalışmada sunulan algoritmanın bu tür problemlerin çözümünde hızlı ve etkin bir yöntem olarak kullanılabilme potansiyeli yüksektir.

Ayrıca önerilen ES algoritması diğer GSP türevleri için de herhangi bir değişiklik yapmadan ya da küçük değişiklikler yapılarak kullanılabilir.

5 Kaynaklar

[1] Abraham, A.P. The Traveling Salesman Problem: Applications, Formulations and Variations. In Combinatorial Optimization, Volume 12, The traveling salesman problem and its variations; Gutin, G; Punnen, A, Eds.; Springer US: Boston, 2002; 1-28.

[2] Rodríguez, A.; Ruiz, R. The effect of the asymmetry of road transportation networks on the traveling salesman problem. Computers & Operations Research 2012; 39, Issue 7, 1566-1576.

[3] Freisleben, B.; Merz, P. A genetic local search algorithm for solving symmetric and asymmetric traveling salesman problems, Proceedings of the IEEE Conference on Evolutionary Computation, Nagoya, 1996; IEEE Press, 1996.

[4] Merz, P.; Freisleben, B. Genetic local search for the TSP: new results, Proceedings of the IEEE International Conference on Evolutionary Computation, Indianapolis, 1997; IEEE Press, 1997.

[5] Nagata, Y.; Soler, D. A new genetic algorithm for the asymmetric traveling salesman problem. Expert Systems with Applications 2012; 39, Issue 10, 8947-8953.

[6] Nagata, Y.; Kobayashi, S. Edge assembly crossover: A high-power genetic algorithm for the traveling salesman problem, Proceedings of the 7th international conference on genetic algorithms, Michigan State University, East Lansing, MI, July 19-23, 1997; Morgan Kaufmann Publishers Inc. San

Francisco, 1997.
[7] Xing, L.N.; Chen, Y.W.; Yang, K.W.; Hou, F; Shen, X.S.; Cai, H.P.. A hybrid approach combining an improved genetic algorithm and optimization strategies for the asymmetric traveling salesman problem. Engineering Applications of Artificial Intelligence 2008; 21, 1370-1380.
[8] Chen, Y.W.; Zhu, Y.J.; Yang, G.K.; Lu, Y.Z. Improved extremal optimization for the asymmetric traveling salesman problem, Physica A: Statistical Mechanics and its Applications 2011; 390, Issues 23–24, 4459-4465.
[9] Bai, J.; Yang, G.K.; Chen, Y.W.; Hu, L.S.; Pan, C.C. A model induced max-min ant colony optimization for asymmetric traveling salesman problem. Applied Soft Computing 2013; 13, Issue 3, 1365-137.
[10] Osaba, E.; Yang, X.S.; Diaz, F.; Lopez-Garcia, P.; Carballedo, R. An improved discrete bat algorithm for symmetric and asymmetric Traveling Salesman Problems. Engineering Applications of Artificial Intelligence 2016; 48, 59-71.
[11] Rechenberg, I. Evolutionsstrategie – Optimierung technischer Systeme nach Prinzipien der biologischen Evolution (PhD thesis), 1971.
[12] Schwefel, H.S. Numerische Optimierung von Computer-Modellen (PhD thesis), 1974.
[13] Beyer, H.G.; Schwefel, H. P. Evolution Strategies: A Comprehensive Introduction. Journal Natural Computing 2002; 1(1), 3–52.
[14] Schrimpf, G.; Schneider, J.; Stamm-Wilbrandt, H.; Dueck, G. Record Breaking Optimization Results Using the Ruin and Recreate Principle. Journal of Computational Physics 2000; Volume 159, Issue 2, 139-171.
[15] Misevicius, A. Genetic algorithm hybridized with ruin and recreate procedure: application to the quadratic assignment problem. Knowledge-Based Systems 2003; 16(5), 261-268.
[16] Burke, E.; Curtois, T.; Hyde, M.; Kendall, G.; Ochoa, G.; Petrovic, S., Vazquez-Rodriguez, J.A.; Gendreau, M. Iterated local search vs. hyper-heuristics: Towards general-purpose search algorithms, Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2010, Barcelona, Spain, July 18-23, 2010; IEEE Press, 2010.
[17] Nawaz, M.; Enscore, E. E.; Ham, I. A heuristic algorithm for the m-machine n-job flow-shop sequencing problem. Omega 1983; 11, 91–95.
[18] Lin, S. Computer Solutions of the Traveling Salesman Problem. Bell System Technical Journal 1965; 44, 2245–2269.
[19] Herdy, M. Application of the 'evolutionsstrategie' to discrete optimization problems, PPSN I Proceedings of the 1st Workshop on Parallel Problem Solving from Nature, October 01 - 03, 1990; Schwefel, H.P., Männer, R. Eds.; Springer-Verlag London, UK, 1991.
[20] Beyer, H.G. Some aspects of the 'evolution strategy' for solving TSP-like optimization problems, Proceedings of the Parallel Problem Solving from Nature 2, 1992; Männer, R., Manderick, B. Eds.; Elsevier, Amsterdam, 1992.