Gezgin Satıcı Probleminin Çözümüne Yönelik Yeni Bir Sezgisel Çözüm Algoritması

Özgür BAŞKAN[1] Cenk OZAN[2]

[1]İnşaat Mühendisliği Bölümü,
Mühendislik Fakültesi,
Pamukkale Üniversitesi, Kınıklı, DENİZLİ
[2]İnşaat Mühendisliği Bölümü,
Mühendislik Fakültesi,
Adnan Menderes Üniversitesi, Efeler, AYDIN

E-mail: obaskan@pau.edu.tr cenk.ozan@adu.edu.tr

Özet

Bu çalışmada Gezgin Satıcı Problemi'nin (GSP) çözümüne yönelik yeni bir sezgisel çözüm algoritması geliştirilmiştir. GSP bilindiği gibi aralarındaki mesafeler bilinen n adet şehrin her birine yalnız bir kez uğranarak başlangıç noktasına geri dönülmesi esnasında kat edilen toplam mesafenin en kısa olduğu turun bulunması olarak tanımlanır. Literatürde çözümü oldukça zor olan problemlerin başında gelen GSP, optimizasyon alanındaki araştırmacılar tarafından uzun yıllardır çalışılmaktadır. Çalışmada geliştirilen sezgisel algoritma toplum tabanlı olup uygulaması oldukça basittir. Sayısal uygulamalar geliştirilen algoritmanın GSP'nin çözümünde kullanılabileceğini göstermiştir.

1. Giriş

Gezgin Satıcı Problemi (GSP) *n* adet şehrin her birinden yalnız bir kez geçen en az maliyetli turun bulunması problemi olarak tanımlanmaktadır. Problemin zorluğu *n*'in büyük değerleri için çözüm havuzunu oluşturan olası turların sayısının oldukça fazla olmasındandır. Bu nedenle nokta sayısının az olduğu durumlarda kesin çözüme ulaşmak mümkün olabilirken, nokta sayısı arttıkça alternatif tur sayısı hızla artmakta ve optimum çözümün bulunabilmesi ya mümkün olamamakta ya da çok fazla süre gerektirmektedir. GSP günlük hayatta karşımıza çıkan ulaşım ve lojistik uygulamaları, araç rotalama problemleri, stok alanındaki malzeme toplama problemleri, uçaklar için havaalanı rotalaması, vb. birçok problemin temel mantığını oluşturmaktadır. GSP matematik, yöneylem araştırması, mühendislik, yapay zeka ve fizik gibi farklı alanlardaki çok sayıda araştırmacının ilgisini çekmekte olup literatürde en çok çalışılan optimizasyon problemlerinden birisidir. GSP'nin araştırmacıların ilgisini çekmesinin en önemli nedeni kolayca formüle edilmesine rağmen çok zor çözülebilen NP-zor sınıfı problemlerden birisi olmasıdır.

GSP'nin çözüm yöntemleri genel olarak ikiye ayrılmaktadır. (1) Dal-Sınır, Dinamik Programlama, Dal-Kesme vb. kesin çözümün elde edilebildiği yöntemler bu gruba girmektedir [1]. Bu tür yöntemler belli bir boyuta kadar olan problemler için başarılı sonuçlar veriyor olsa da GSP'de nokta sayısı arttıkça çözüm için gerekli olan işlem süresinin çok fazla olmasından dolayı optimum sonuçlara ulaşmak imkansız olabilmektedir. (2) Optimum çözümün bulunması kesin olmayan ancak daha az sayıda işlem gerektiren yöntemlerdir. Bu yöntemler genel olarak sezgisel yöntemler olarak tanımlanır ve optimum sonucu garanti etmemekle birlikte, kısa çözüm süresi ile tatmin edici sonuçlar alınabilmektedir.

Literatürde GSP'nin hızlı ve etkin çözümü amacıyla birçok yöntem geliştirilmiştir. Tamsayı ve dinamik programlama [2-4] yöntemleri kullanıldığı gibi son yıllarda sezgisel metotların GSP çözümündeki performanslarının test edilmesi amacıyla birçok çalışma yapılmıştır. Örnek olarak GSP'nin çözümünde Yapay Sinir Ağları ve Genetik Algoritma (GA) metotları kullanılmıştır [5-6]. Bunun yanında modifiye Karınca Kolonisi Optimizasyonu (KKO) algoritması GSP'ye uygulanmış ve literatürdeki diğer algoritmalarla karşılaştırılmıştır [7]. GSP'nin çözümünde değişken stratejili çoklu KKO algoritması geliştirilmiş ve başarılı sonuçlar alınmıştır [8]. Benzer şekilde iyileştirilmiş KKO ile GSP çözümü gerçekleştirilmiştir [9]. Yerel arama stratejili GA GSP'ye uygulanmış ve başarılı sonuçlar elde

edilmiştir [10]. Pekiştirmeli öğrenme ile GA metodu birleştirilmiş ve farklı GSP'ler üzerinde test edilmiştir [11]. GA'dan faydalanarak rota planlamasına yönelik bir uygulama yazılımı geliştirilmiştir [1].

Literatürden görülebileceği gibi son yıllarda sezgisel metotlar GSP'nin çözümünde geniş bir kullanım alanı bulmaktadır. Bu çalışmada da bu amaçla yeni bir sezgisel çözüm algoritması geliştirilmiştir. Çalışmanın ikinci bölümünde GSP'nin matematiksel ifadesi, üçüncü bölümde geliştirilen sezgisel algoritma, sonraki bölümde sayısal uygulamalar ve son bölümde sonuçlar ve öneriler yer alacaktır.

2. Problem formülasyonu

GSP; *n* adet şehri dolaşan, her birinden yalnız bir kez geçen ve başladığı noktaya dönen en az maliyetli turun belirlenmesi problemidir ve amaç fonksiyonu Denklem (1)'de verildiği gibi ifade edilebilir [12]. *n* adet şehirden oluşan bir GSP'de olası tur sayısı $(n-1)!/2$ olarak belirlenebilir. Örnek olarak 20 şehirden oluşan bir GSP'de olası tur sayısı $6.08*10^{16}$'dır. Denklem (1)'de verilen amaç fonksiyonuna ait kısıtlar Denklem (2-5)'de verilmiştir.

$$\min z = \sum_{i=1}^{n} \sum_{j=1, i\neq j}^{n} c_{ij} x_{ij} \tag{1}$$

$$\sum_{i=1,\ i\neq j}^{n} x_{ij} = 1, \quad j = 1,2,...,n \tag{2}$$

$$\sum_{j=1,\ j\neq i}^{n} x_{ij} = 1, \quad i = 1,2,...,n \tag{3}$$

$$\sum_{i,j\in S,\ i\neq j}^{n} x_{ij} \leq |S| - 1, \quad \forall S \subset \{1,2,....,n\} \tag{4}$$

$$x_{ij} = \begin{cases} 1, \ i \text{ noktasından } j \text{ noktasına gidiliyor ise} \\ 0, \text{ aksi durumda} \end{cases} \tag{5}$$

Burada; c_{ij}, *i* ve *j* noktaları arasındaki mesafeyi; x_{ij}, *i* noktasından *j* noktasına gidilip gidilmediğini gösteren ifade olup gerçekleşmesi halinde 1, aksi durumda ise 0 değerini almaktadır. Denklem (2-3)'de verilen kısıtlar her noktaya yalnız bir kez uğranılmasını sağlamak amacıyla kullanılmaktadır. Olası alt turlardan kurtulmak amacıyla Denklem (4)'de verilen kısıt kullanılmaktadır.

3. Sezgisel çözüm algoritması

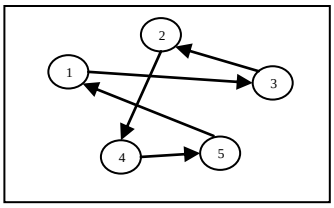
Geliştirilen sezgisel algoritmada iki parametre kullanılmaktadır. Bunlardan biri tüm toplum tabanlı sezgisel yaklaşımlarda kullanılan toplum büyüklüğü (*m*) ve diğeri ise jenerasyon sayısıdır. Bu değişkenlerin dışında algoritmanın başka tür parametre içermemesi yöntemin parametre duyarlılığını azaltmaktadır. Geliştirilen algoritma 3 adımdan oluşmakta olup MATLAB kodu Ekler bölümünde verilmiştir.

ADIM 1: İlk olarak toplum büyüklüğü (*m*) ve probleme özgü şehir sayısı *n* olarak verilirse *m**(*n*+1) boyutunda çözüm matrisi oluşturulur.

$$\mathbf{A} = \begin{pmatrix} a_{11} & \cdots & a_{1(n+1)} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{m(n+1)} \end{pmatrix}_{m*(n+1)} \begin{bmatrix} z_1 \\ \cdots \\ z_m \end{bmatrix} \tag{6}$$

Burada; a_{ij} değerleri (i=1,2,…,m ; j=1,2,…,(n+1)) GSP'de şehir numaralarını temsil etmektedir. **A** matrisindeki ilk ve son sütundaki elemanlar aynı değere sahip olup probleme özgü olası turun başlayıp bittiği şehir numarası olarak temsil edilmektedir. Bu değerlerin dışındaki matris elemanları Adım 1'de rastgele oluşturulur. Ayrıca, *z* değeri (z=1,2,….,m) **A** matrisindeki her bir satırda belirlenen olası tur konfigürasyonuna bağlı olarak Denklem (1) ile elde edilen amaç fonksiyonu değeridir.

ADIM 2: Bu adımda Adım 1'de oluşturulan başlangıç toplumundaki her bir satırdaki olası turların şehir numaraları (başlangıç ve bitiş numaraları hariç) kendi içlerinde rastgele olarak değiştirilir. Örnek olarak Şekil 1'de verilen 5 adet şehirden oluşan GSP'de başlangıç toplumunun ilk satırı Tablo 1'de görüldüğü gibi rastgele değiştirilerek yeni bir vektör oluşturulur.



Şekil 1. Gezgin Satıcı Problemi.

Tablo 1. Rastgele Olası Tur Oluşturma

Adım 1 **(1. satır vektörü)**	**1**	**3**	**2**	**4**	**5**	**1**
Adım 2 **(1. satır vektörü)**	**1**	**5**	**4**	**2**	**3**	**1**

Elde edilen yeni olası tur vektörü için amaç fonksiyonu Denklem (1) ile hesaplanır ve elde edilen değer eski değerden daha iyi ise Adım 2'de elde edilen yeni vektör eski vektörün yerini alır. Aksi durumda Adım 3'e geçilir. Bu süreç **A** matrisinin tüm satırlarına uygulanır.

ADIM 3: Bu adım, Adım 2'de elde edilen yeni vektörün daha iyi sonuç vermemesi durumunda uygulanır. Tablo 2'de görüldüğü gibi bir önceki jenerasyonda en iyi amaç fonksiyonu değerini veren olası tur vektörü içinde rastgele seçilen 2 adet şehir numarasının değiştirilmesi (mutasyon) neticesinde yeni bir olası tur vektörü oluşturulur.

Tablo 2. En iyi tur vektörünün mutasyonu

En iyi tur vektörü	**1**	**4**	**2**	**3**	**5**	**1**
Mutasyona uğramış en iyi tur vektörü	**1**	**3**	**2**	**4**	**5**	**1**

Bu aşamadan sonra elde edilen yeni tur vektörünün Denklem (1) yardımı ile amaç fonksiyonu değeri hesaplanır. Elde edilen değer toplumdaki tur vektörünün amaç fonksiyonu değeri ile karşılaştırılır. Bu değer öncekinden daha iyi ise vektör değiştirilir aksi durumda bir sonraki jenerasyona gidilir. Bu adımda amaç önceki jenerasyonlarda elde edilen en iyi tur vektörlerinin etkisinin sonraki jenerasyonlara aktarılmasını sağlamak ve bu sayede belli sayıdaki jenerasyondan sonra optimum ya da optimuma yakın sonuçlara ulaşılabilmesini sağlamaktır.

4. Sayısal uygulamalar

Çalışmada geliştirilen sezgisel algoritma 7,15 ve 16 adet şehirden oluşan GSP'ler üzerinde test edilmiştir. 7 ve 15 şehirli GSP'lere ait uzaklık matrisleri Tablo 3 ve 4'de verilmiştir. 16 şehirli GSP uzaklık matrisi ise literatürden elde edilebilir [13]. 7 şehirden oluşan GSP'de olası tur sayısı 360 iken 15 şehirli GSP'de olası tur sayısı $4.36*10^{10}$ olarak belirlenmektedir. 16 şehirli problemde (ulysses16) ise 15 şehirli probleme göre değişken sayısı 1 adet artmasına rağmen olası tur sayısı $6.54*10^{11}$ olmaktadır. Diğer bir deyişle 1 adet şehrin probleme eklenmesi, olası tur sayısını 15 kat artırmaktadır.

Tablo 3. GSP (7 şehir)

	1	**2**	**3**	**4**	**5**	**6**	**7**
1	-	75	99	9	35	63	8
2	51	-	86	46	88	29	20
3	100	5	-	16	28	35	28
4	20	45	11	-	59	53	49
5	86	63	33	65	-	76	72
6	36	53	89	31	21	-	52
7	58	31	43	67	52	60	-

Tablo 4. GSP (15 şehir)

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
1	-	1	2	4	9	8	3	2	1	5	7	1	2	9	3
2	1	-	5	3	7	2	5	1	3	4	6	6	6	1	9
3	2	5	-	6	1	4	7	7	1	6	5	9	1	3	4
4	4	3	6	-	5	2	1	6	5	4	2	1	2	1	3
5	9	7	1	5	-	9	1	1	2	1	3	6	8	2	5
6	8	2	4	2	9	-	3	5	4	7	8	3	1	2	5
7	3	5	7	1	1	3	-	2	6	1	7	9	5	1	4
8	2	1	7	6	1	5	2	-	9	4	2	1	1	7	8
9	1	3	1	5	2	4	6	9	-	3	3	5	1	6	4
10	5	4	3	4	1	7	1	4	3	-	9	1	8	5	2
11	7	6	5	2	3	8	7	2	3	9	-	2	1	8	1
12	1	6	9	1	6	3	9	1	5	1	2	-	5	4	3
13	2	6	1	2	8	1	5	1	1	8	1	5	-	9	6
14	9	1	3	1	2	2	1	7	6	5	8	4	9	-	7
15	3	9	4	3	5	5	4	8	4	2	1	3	6	7	-

7 şehirli GSP'nin çözümünde toplum büyüklüğü m=10 ve jenerasyon sayısı 500 olarak alınmıştır. Elde

edilen en iyi tur 74 jenerasyon sonunda elde edilmiş ve Tablo 5'de verilmiştir. Bulunan sonuçlar literatür ile uyumludur.

Tablo 5. En iyi tur (7 şehir)

Şehir Numaraları								Mesafe
1	7	2	6	5	3	4	1	158

15 şehirli GSP'nin çözümünde *m*=50 ve jenerasyon sayısı 10000 alınmış ve literatürde verilen en iyi tur ve mesafe değerleri 5531 jenerasyon sonunda elde edilmiştir. Söz konusu problemin en iyi mesafe değeri 17 olup bu değeri sağlıyan 2 farklı tur konfigürasyonu Tablo 6'da verilmiştir.

Tablo 6. En iyi tur (15 şehir)

En iyi tur (1. seçenek)								Mesafe
1	9	3	5	8	12	10	15	17
11	13	6	4	7	14	2	1	
En iyi tur (2. seçenek)								
1	2	8	12	4	7	14	6	17
13	11	15	10	5	3	9	1	

Diğer bir örnek olan 16 şehirli GSP'nin (ulysses16) çözümü için *m*=50 ve maksimum jenerasyon sayısı 10000 alınmış ve 6144 jenerasyon sonunda literatürde verilen optimum mesafe değeri elde edilmiştir. Çözüm sonunda elde edilen optimum tur Tablo 7'de verilmiştir.

Tablo 7. En iyi tur (16 şehir)

Şehir Numaraları									Mesafe
1	14	13	12	7	6	15	5	11	6859
9	10	16	3	2	4	8	1		

5. Sonuçlar

Çalışmada GSP'nin çözümüne yönelik yeni bir sezgisel çözüm algoritması geliştirilmiştir. Algoritmanın toplum büyüklüğü ve jenerasyon sayısı dışında parametre içermemesi yöntemin parametrelere karşı duyarlılığının en az seviyeye indirilmesini sağlamıştır. Geliştirilen algoritma 7,15 ve 16 adet şehirden oluşan GSP'ler üzerinde test edilmiş ve literatürde verilen optimum çözümler her problem için elde edilmiştir. Bilindiği gibi sezgisel algoritmaların sürekli olarak iyileştirilmeye açık olmasından dolayı gelecek çalışmalarda geliştirilen algoritmaya farklı yerel arama operatörlerinin eklenmesi ve bu sayede metodun daha etkin hala getirilebilmesine yönelik çalışmalar yapılacak ve daha büyük ölçekli GSP'ler üzerinde test edilecektir.

6. Kaynaklar

[1] S.Çolak, "Genetik Algoritmalar Yardımı ile Gezgin Satıcı Probleminin Çözümü Üzerine Bir Uygulama", Çukurova Üniversitesi, Sosyal Bilimler Enstitüsü Dergisi, 19(3), 2010, 423-438.

[2] B.W. Douglas, Introduction to graph theory (Section ed.). Beijing: Pearson Education Asia Limited and China Machine Press, 2006, 74‑78.

[3] S. Climer, W.X. Zhang, "Cut-and-solve: An iterative search strategy for combinatorial optimization problems" Artificial Intelligence, 170(8‑9), 2006, 714‑738.

[4] D.S. Johnson, L.A. McGeoch, The traveling salesman problem and its variations, combinatorial optimization. London: Springer Press, 2004, 445‑487.

[5] K.S. Leung, H.D. Jin, Z.B. Xu, "An expanding self-organizing neural network for the traveling salesman problem", Neurocomputing, 6, 2004, 267‑292.

[6] H. D. Nguyen, I. Yoshihara, K. Yamamori, M. Yasunaga, "Implementation of an effective hybrid GA for large-scale traveling salesman problem", IEEE Transactions on System, Man, and Cybernetics – Part B, 37(1), 2007, 92‑99.

[7] G. Shang, Z. Lei, Z. Fengting, Z. Chunxian, "Solving Traveling Salesman Problem by Ant Colony Optimization Algorithm with Association Rule", Third International Conference on Natural Computation, 2007.

[8] I. Ellabib, P. Calamai, O. Basir, "Exchange strategies for multiple ant colony system", Information Sciences, 177(5), 2007, 1248‑1264.

[9] L. Li, S. Ju, Y. Zhang, "Improved ant colony optimization for the traveling salesman problem" In Proceedings of the 2008 international

conference on intelligent computation technology and automation, 1, 2008, 76‑80.

[10] M.F. Taşgetiren, P.N. Suganthan, Q.K. Pan, Y.C. Liang, "A genetic algorithm for the generalized traveling salesman problem", In Proceedings of the 2007 IEEE congress on evolutionary computation, Singapore, 2007, 2382‑2389.

[11] F. Liu, G.Z. Zeng, "Study of genetic algorithm with reinforcement learning to solve the TSP", Expert System with Applications, 36(3), 2009, 6995‑7001.

[12] G. Dantzig, R. Fulkerson, S. Johnson, "Solution of a large-scale traveling salesman problem", Operations Research, 2, 1954, 393-410.

[13] TSPLIB.http://www.iwr.uniheidelberg.de/groups /comopt/software/TSPLIB95/tsp/, 2014.

7. Ekler

```
NP=10;    %toplum büyüklüğü
D=7;      %şehir sayısı
maxgen=500; %maksimum jenerasyon sayısı
load maliyet.txt %maliyet matrisi
for i=1:NP
    y=randperm(D);
    y(y==1)=[];
    x(i,1:D-1)=y;
end
k=ones(NP,1);
l=ones(NP,1);
n=zeros(NP,1);
x=[k x(:,1:D-1) l n];
for i=1:NP
    for j=1:D
        x(i,D+2)=x(i,D+2)+maliyet(x(i,j),x(i,j+1));
    end
end
for gen=1:maxgen
    for i=1:NP
        x1=x(i,2:D);
        r=x1(randperm(length(x1)));
        m(i,1:D+1)=x(i,1:D+1);
        m(i,2:D)=r;
        m(i,D+2)=0;
        for j=1:D
            m(i,D+2)=m(i,D+2)+maliyet(m(i,j),m(i,j+1));
        end
        if m(i,D+2)<x(i,D+2)
            x(i,:)=m(i,:);
        else
            if gen>1
                w1=round(rand*(D-2)+2);
                w2=round(rand*(D-2)+2);
                best_c=best;
                best(best_c==w1)=w2;
                best(best_c==w2)=w1;
                m(i,2:D)=best(1,2:D);
                m(i,D+2)=0;
                for j=1:D
                    m(i,D+2)=m(i,D+2)+maliyet(m(i,j),m(i,j+1));
                end
                if m(i,D+2)<x(i,D+2)
                    x(i,:)=m(i,:);
                end
            end
        end
    end
      [fmin,K]=min(x(:,D+2));
       best=x(K,:);
end
best %en iyi tur
fmin % amaç fonksiyonu en iyi değeri
```