Süleyman Demirel Üniversitesi
Fen Bilimleri Enstitüsü Dergisi
Cilt 20, Sayı 2, 319-323, 2016

Süleyman Demirel University
Journal of Natural and Applied Sciences
Volume 20, Issue 2, 319-323, 2016

DOI: 10.19113/sdufbed.55498

Gezgin Satıcı Problemi İçin Merkezden Kenarlara Hipersezgisel Yöntem

Fidan NURİYEVA[1,2], Gözde KIZILATEŞ[*3]

[1]Dokuz Eylül Üniversitesi, Fen Fakültesi, Bilgisayar Bilimleri Bölümü, 35160, İzmir
[2]Azerbaycan Ulusal Bilimler Akademisi, Yönetim Sistemleri Enstitüsü, Bakü
[3]Ege Üniversitesi, Fen Fakültesi, Matematik Bölümü, 35100, İzmir

(Alınış / Received: 03.03.2016, Kabul / Accepted: 13.05.2016, Online Yayınlanma / Published Online: 10.06.2016)

Anahtar Kelimeler
Simetrik Gezgin Satıcı Problemi,
Hipersezgisel Algoritma,
Ekleme Sezgiseli,
En Kısa Yol Algoritması,
En Yakın Komşu Algoritması

Özet: Bu makalede Gezgin Satıcı Problemi için yeni bir hipersezgisel algoritma önerilmiştir. Bu yöntemde önce N adet şehir içerisinden merkez şehir ve 4 uç şehir seçilip, sonra ise merkez ile ikişer-ikişer uç şehirlerin orta noktaları belirlenerek merkez şehirden başlanarak bu 9 şehirden geçen bir devre oluşturulmuştur. Daha sonra "en kısa yol" ve "ekleme sezgiseli" algoritmaları kullanılarak bulunan devre tüm şehirlerden geçecek şekilde genişletilmiştir. Önerilen algoritmalar ile kütüphane problemleri üzerinde hesaplama denemeleri yapılmış, elde edilen sonuçlar "en yakın komşu" algoritmasından elde edilen sonuçlar ile karşılaştırılmıştır. Hesaplama denemeleri önerilen algoritmanın verimli olduğunu göstermektedir.

A New Hyper-Heuristic Method for Traveling Salesman Problem from Center to Margins

Keywords
Symmetric Travelling Salesman Problem,
Hyper-Heuristic Algorithm,
Insertion Heuristic,
Shortest Path Algorithm,
Nearest Neighbor Algorithm

Abstract: In this study a new hyper-heuristic algorithm is proposed for Traveling Salesman Problem. The central point of the tour is identified and four point vertices are selected from N cities first in the proposed method. Secondly, the mid-points of the center and each of the two point vertices are calculated one by one in order to establish a tour that includes a total of nine cities. This tour is extended afterwards in order to contain all vertices by using "nearest neighbor" and "insertion heuristic" algorithms. Computational experiments were conducted with a library of sample instances for the TSP and the results were compared with the results obtained from "Nearest Neighbor" algorithm. The proposed algorithm in this study is shown to be efficient regarding the experimental results.

1. Giriş

Gezgin Satıcı Problemi (GSP), aralarındaki uzaklıkları bilinen n adet şehrin (tepe, nokta, düğüm, yerleşim yeri, müşteri, şube vb.) her birine yalnız bir kere uğrayıp başlangıç noktasına geri dönen en kısa uzunluğa sahip şehir sıralamasının bulunmasını amaçlayan bir optimizasyon problemdir. Problem ilk kez 1932 yılında tanımlanmıştır [1]. Dağıtım, planlama, lojistik gibi alanlar başta olmak üzere birçok sektörde geniş uygulama alanına sahip olan ve pratikte karşılaşılan çoğu problemle ilişkili olan GSP, optimizasyon alanında araştırmacılar tarafından uzun yıllardır yoğun olarak çalışılan ve çözümler üretilen çözümü zor (NP-hard) bir problemdir [2].

Bu çalışmada, Gezgin Satıcı Problemi için "En Kısa Yol" ve "Ekleme Sezgiseli" algoritmalarına dayanan yeni bir hipersezgisel algoritma önerilmiştir. Makalenin diğer bölümleri şu şekilde hazırlanmıştır: İkinci bölümde GSP için literatürde bulunan çözüm yöntemlerinden bahsedilmiştir. Üçüncü bölümde algoritmada kullanılan "En Kısa Yol", "Ekleme Sezgiseli" ve "En Yakın Komşu" algoritmalarına değinilmiştir. Dördüncü bölümde önerilen yeni hipersezgisel algoritma detaylı bir şekilde anlatılmıştır. Beşinci bölümde algoritmanın farklı problemler için bulduğu sonuçlar tablo üzerinde gösterilmiştir. Makalenin son bölümü olan sonuç kısmında ise, yapılan çalışmalar özetlenmiş, önerilen algoritmaya ilişkin elde edilen genel sonuçların, karşılaştırılan algoritma ile kıyaslaması verilmiş ve ileriye dönük öneriler yer almıştır.

* İlgili yazar: gozde.kizilates@gmail.com

2. GSP İçin Bazı Çözüm Yöntemleri

GSP'yi çözmek için önerilen algoritmalar temel olarak üç çeşittir. Bunlar; Kesin Algoritmalar, Yaklaşık Algoritmalar ve Sezgisel Algoritmalardır [1-5].

2.1. Kesin algoritmalar

Kesin algoritmalar, genellikle, GSP'nin tamsayılı lineer programlama formülünden elde edilen veya tüm olasılıkları deneyen yöntemlerdir. Fakat bu algoritmaların hesaplanabilirlikleri pahalıdır. Bu yaklaşımlara örnek olarak "Dal ve Sınır" algoritması verilebilir. Diğer yöntemler ise lineer programlamaya dayanan yöntemler ve dinamik programlama yöntemleridir [2].

2.2. Yaklaşık algoritmalar

Yaklaşık yöntemler optimum çözüme ulaşmayı garanti etmezler, ancak optimum değerden en fazla ne kadar uzak çözüm bulunabileceğini belirlerler. GSP için bilinen iyi yaklaşım algoritmalarına Minimum Kapsayan Ağaca Dayalı Algoritma ve Christofides Algoritması örnek gösterilebilir [1,3,6].

2.3. Sezgisel algoritmalar

Kesin algoritmalar genelde olası tüm durumların denendiği algoritmalardır. Bu yüzden bu algoritmaların çalışma zamanı verimli değildir. Bu durumda, optimum çözüme yakın bir çözüm, daha kısa zamanda, kesin algoritmalar kullanılmadan da bulunabilir. Pratikte sezgisel algoritmalar, kesin algoritmalar ile karşılaştırıldığı zaman daha çok tercih edilmektedir. GSP'yi çözen sezgisel algoritmalar üçe ayrılır: Tur oluşturan sezgiseller, Turu geliştiren sezgiseller ve bu iki yöntemin ortak olarak kullanıldığı melez yöntemlerdir [2,4].

2.3.1. Tur oluşturan sezgisel algoritmalar

Tur oluşturan algoritmalar bir sonuç buldukları zaman bu sonucu geliştirmezler. Bir sonuç elde edildiği zaman algoritmanın çalışması sonlanır. En çok bilinen tur oluşturan sezgisel algoritmalar En yakın komşu, Açgözlü, Ekleme Sezgiseli algoritmalarıdır [4].

2.3.2. Turu geliştiren sezgisel algoritmalar

Bu algoritmalar oluşturulmuş bir turu geliştirmeyi amaçlarlar. Bu algoritmalara örnek olarak 2-opt, 3-opt ve Lin-Kernighan gibi yerel eniyileme algoritmaları örnek verilebileceği gibi, Tabu araması, Genetik algoritmalar, Benzetim Tavlama ve Karınca Kolonisi Algoritması gibi Yapay Zeka yöntemleri de örnek olarak verilebilir [4].

2.3.3. Melez yöntemler

Melez yöntemler ilk önce tur oluşturma daha sonra da turu geliştirmeye dayanan, bu iki yöntemin ortak kullanıldığı algoritmalardır. Melez yöntemlere örnek olarak Yinelemeli Lin-Kernighan algoritması verilebilir. En başarılı sonuçlar melez yöntemlerden elde edilmektedir [4].

3. Önerilen Yöntemde Kullanılan Algoritmalar

Çalışmanın bu bölümünde GSP için önerilen yeni yöntemde kullanacağımız "En Kısa Yol", "Ekleme Sezgiseli" ve "En Yakın Komşu" algoritmalarından bahsedilecektir.

3.1. En kısa yol algoritması

En Kısa Yol Algoritması bir çizgede arasında en az bir yol bulunan tepeler arasında gidilebilen en kısa yolu bulmayı amaçlar. En kısa yolu bulmak için birçok algoritma mevcuttur. Bunlardan bazıları Djikstra, Bellman-Ford, Floyd-Warshall algoritmalarıdır. Djikstra algoritması bir tepeden diğer tüm tepelere olan en kısa yolları bulur. Bellman-Ford algoritması bir tepeden diğer tüm tepelere olan en kısa yolları ayrıt ağırlıkları negatif olan çizgelerde bulur. Floyd-Warshall algoritması ise tüm tepe ikilileri arasındaki en kısa yolları bulur [2].

3.2. Ekleme sezgiseli

Bu algoritma alt tur ile başlayıp her defasında alt turdaki şehirlerin arasına onlara en yakın şehrin eklenmesine dayanan bir yöntemdir [4].

Algoritmanın adımları aşağıdaki gibidir:

Adım 1. En kısa uzunluklu ayrıtı seç ve alt tur oluştur.
Adım 2. Alt turda bulunmayan ve alt turdaki şehirlerden herhangi iki şehre en yakın olan bir şehir seç.
Adım 3. Bu şehri alt turda yakın olduğu şehirlerin arasına ekle.
Adım 4. Eklenecek şehir kalmayıncaya kadar Adım 2'ye git.

3.3. En yakın komşu algoritması

Bu algoritma en sade ve en açık GSP sezgiselidir. Bu algoritmada amaç daima en yakın şehri ziyaret etmektir [4].

Algoritmanın adımları aşağıdaki gibidir:

Adım 1. Rastgele bir şehir seç.
Adım 2. En yakın ziyaret edilmemiş şehri bul ve o şehre git.
Adım 3. Ziyaret edilmemiş şehir kaldı mı? Eğer cevap EVET ise Adım 2'ye dön.
Adım 4. İlk şehre geri dön.

4. Gezgin Satıcı Problemi İçin Yeni Hipersezgisel Algoritma

Makalenin bu bölümünde Simetrik Gezgin Satıcı Problemi için önerilmiş olan yeni hipersezgisel algoritma ile ilgili bilgi verilecektir. Bu yöntemde önce merkez (O) tepe bulunur. Daha sonra merkez tepeye uzak olan 4 uç tepe (A1, A2, A3, A4), sonra ise merkez tepe ile (O) ikişer - ikişer uç tepelerin (O, A1, A3), (O, A2, A3), (O, A2, A4), (O, A4, A1) orta noktaları (A5, A6, A7, A8) belirlenir. Bu 9 tepe ile (O → A5 → A6 → A7 → A2 → A3 → A1 → A4 → A8 → O) sırasında kapalı bir tur elde edilir. Daha sonra "En kısa yol" algoritması ile bu turdaki iki tepe arasındaki en uzun yol bulunarak adım adım yeni tepeler tura eklenir. Bu yöntem ile tura eklenecek tepe kalmadığı zaman kalan tepeler "Ekleme sezgiseli" algoritması kullanılır ve tüm tepelerden geçecek şekilde tur genişletilir (MKHA1 Algoritması).

Ayrıca makalede önerilen bu algoritmanın daha basit bir şekli olan ve başlangıç turunun 5 tepe ile oluşturulduğu algoritma da önerilmiştir (O → A1 → A3 → A2 → A4 → O) (MKHA2 Algoritması).

MKHA 1 Algoritmasının adımları aşağıdaki gibidir:

MKHA 1 algoritması

Adım 1. Merkez tepeyi bul (A0).

Algoritmada belirtilen merkez tepenin koordinatları aşağıdaki gibi hesaplanıyor.

$$x_0 = \frac{1}{n}\sum_{i=1}^{n} x_i , \qquad y_0 = \frac{1}{n}\sum_{i=1}^{n} y_i \qquad (1)$$

Adım 2. Merkeze en uzak tepeyi bul (A1).
Adım 3. A1 tepesine en uzak tepeyi bul (A2).
Adım 4. A1 ve A2 tepelerine en uzak tepeyi bul (A3).
Adım 5. A1, A2 ve A3 tepelerine en uzak tepeyi bul (A4).
Adım 6. A0, A1 ve A3'ün orta noktasının koordinatları aşağıda gösterildiği gibi bulunur (A5).

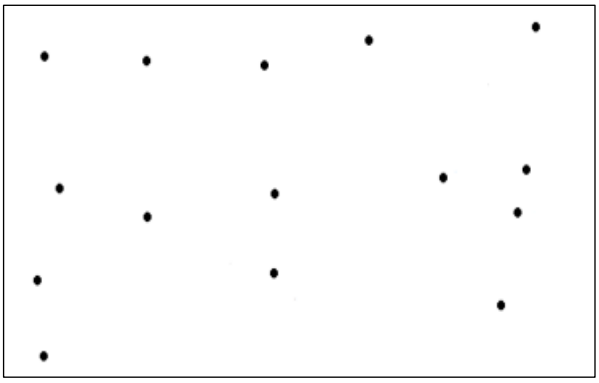
$$x_5 = \frac{1}{3}(x_0 + x_1 + x_3), \quad y_5 = \frac{1}{3}(y_0 + y_1 + y_3) \qquad (2)$$

Koordinatları en yakın tepe A0, A1 ve A3 tepelerinin orta noktası olacaktır.

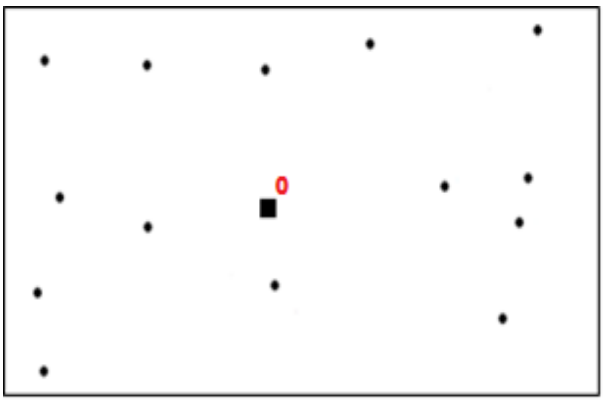
Benzer şekilde, (A0, A3, A2) tepelerinin orta noktası olarak A6, (A0, A2, A4) tepelerinin orta noktası olarak A7, (A0, A4, A1) tepelerinin orta noktası olarak A8 bulunur.

Adım 7. Başlangıç tur aşağıdaki şekilde oluşturulur.
A0→A5→A6→A7→A2→A3→A1→A4→A8→A0
Adım 8. Oluşturulan turdaki en uzun ayrıtı bul.
Adım 9. Bu ayrıtın iki uç tepesini birleştiren bu ayrıt dışındaki en kısa yolu bul.
Adım 10. Eğer yeni bulunmuş ayrıtların uzunlukları toplamı bu iki uç tepe arasındaki ayrıtın uzunluğundan küçük ise bu ayrıtı sil ve yeni bulunmuş tepe ve ayrıtları seçilmişler listesine ekle ve turu genişlet. Adım 8'e git.
Adım 11. Aksi halde oluşturulan turdaki bir sonraki en uzun ayrıtı bul ve Adım 9'a git.
Adım 12. Boşta kalan tepeler var ise bu tepeleri "Ekleme Sezgiseli" yöntemi ile turlara ekle.
Adım 13. Dur.

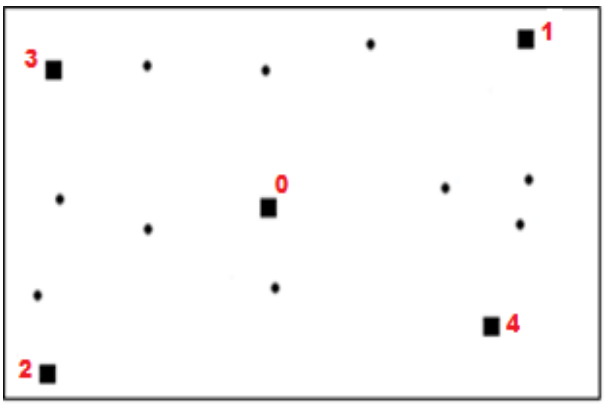
MKHA 1 algoritmasının nasıl çalıştığını bir örnek üzerinde gösterelim. Tepelerimiz Şekil 1'de gösterildiği gibi olsun.



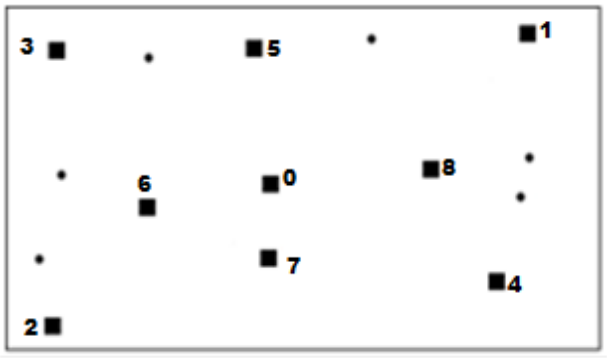
Şekil 1. Örnekteki tepelerin yerleri



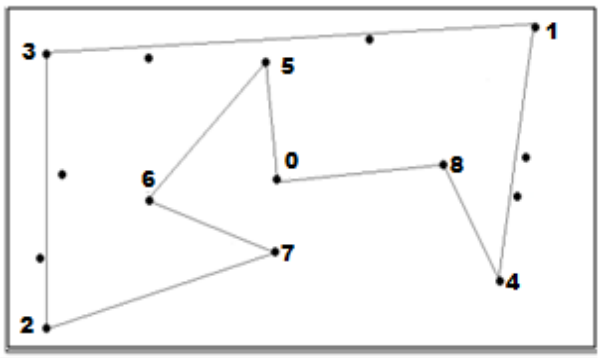
Şekil 2. Merkez tepenin bulunması



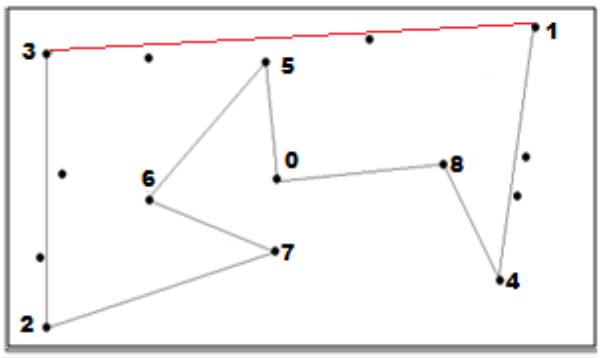
Şekil 3. Uç tepelerin bulunması



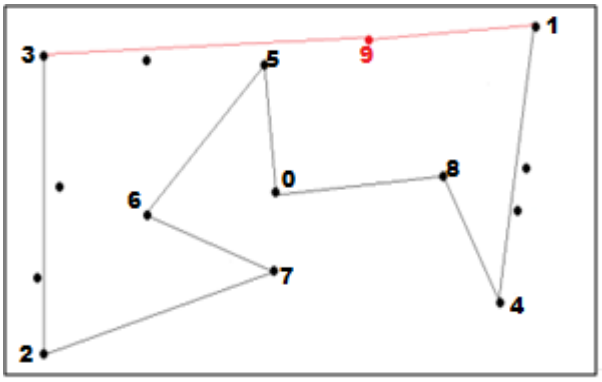
Şekil 4. Orta tepelerin bulunması



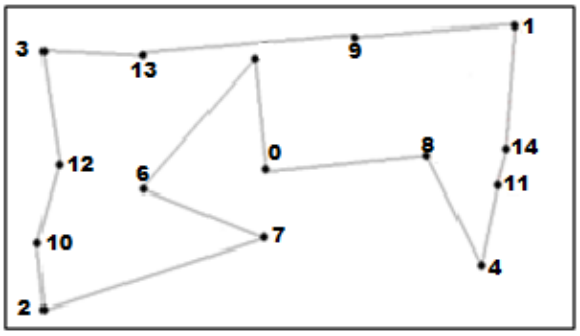
Şekil 5. A0→A5→A6→A7→A2→A3→A1→A4→A8→A0 turunun bulunması



Şekil 6. Oluşturulan turdaki en uzun ayrıtın bulunması ve bu ayrıtın iki ucunu birleştiren bu ayrıt dışındaki en kısa yolun bulunması



Şekil 7. Yeni bulunmuş ayrıtlar seçilmiş ayrıttan küçük ise seçilmiş ayrıtın silinmesi ve yeni bulunmuş tepe ve ayrıtların seçilmişler listesine eklenip, turun genişletilmesi



Şekil 8. Turun tamamlanması

Önerilen algoritmanın daha basit bir versiyonu olan MKHA 2 algoritmasının adımları aşağıdaki gibidir:

MKHA 2 algoritması

Adım 1. Merkez tepeyi bul (A0).
Adım 2. Merkez tepeye en uzak tepeyi bul (A1).
Adım 3. A1 tepesine en uzak tepeyi bul (A2).
Adım 4. A1 ve A2 tepelerine en uzak tepeyi bul (A3).
Adım 5. A1, A2 ve A3 tepelerine en uzak tepeyi bul (A4).
Adım 6. Başlangıç tur aşağıdaki şekilde oluşturulur. A0→A1→A3→A2→A4→A0
Adım 7. Oluşturulan turdaki en uzun ayrıtı bul.
Adım 8. Bu ayrıtın iki uç tepesini birleştiren bu ayrıt dışındaki en kısa yolu bul.
Adım 9. Eğer yeni bulunmuş ayrıtların uzunlukları toplamı bu iki uç tepe arasındaki ayrıtın uzunluğundan küçük ise bu ayrıtı sil ve yeni bulunmuş tepe ve ayrıtları seçilmişler listesine ekle ve turu genişlet. Adım 7'ye git.
Adım 10. Aksi halde oluşturulan turdaki bir sonraki en uzun ayrıtı bul ve Adım 8'e git.
Adım 11. Boşta kalan tepeler var ise bu tepeleri "Ekleme Sezgiseli" yöntemi ile turlara ekle.
Adım 12. Dur.

5. Hesaplama Denemeleri

Önerilen algoritmaların C++ programlama dili ile programları yazılmış ve algoritmalar TSPLIB kütüphane problemleri (http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp/)[7] üzerinde test edilmiştir. Elde edilen sonuçlar "En Yakın Komşu" algoritması ile kıyaslanmış ve tablo üzerinde gösterilmiştir. Tabloda yer alan ilk satır kütüphane problemlerinin isimlerini ve boyutlarını (şehir sayısı) yansıtmaktadır. Diğer satırlarda ise sırasıyla adı geçen problemler için bilinen optimum sonuçlar, En Yakın Komşu algoritması ile elde edilen sonuçlar, diğer iki satırda ise önerilen algoritmalar ile elde edilen sonuçlar yer almaktadır. Tabloda seçili olan hücreler optimuma en yakın olan değerleri temsil etmektedir. Hesaplama denemeleri sonuçları algoritmanın verimli olduğu göstermektedir.

Tablo 1. Hesaplama denemeleri sonuçları

	Eil 51	Eil 76	Rat 99	Rd 100	Eil 101	Ch 130	Ch 150	Rat 195	D 198	Rd 400	D 493	Rat 575	D 657	Rat 783	D 1291
Optimum	426	538	1211	7910	629	6110	6528	2323	15780	15281	35002	6773	48912	8806	50801
NN	514	712	1565	9941	825	7575	8195	2762	18655	19168	43646	8449	61874	11255	60996
MKHA1	480	581	1361	8588	694	6670	7479	2652	170601	17561	39854	8097	58602	10784	62737
MKHA2	464	599	1323	8940	715	6780	7709	2641	18350	18213	40816	8269	59098	10706	65575

6. Sonuç

Sonuç olarak, bu makalede Gezgin Satıcı Problemi için "Ekleme Sezgiseli" ve "En Kısa Yol" algoritmalarına dayanan hipersezgisel bir algoritma önerilmiştir. Önerilen algoritmalar C++ programlama dilinde kodlanmış ve kütüphane problemleri üzerinde hesaplama denemeleri yapılmıştır. Hesaplama denemeleri önerilen algoritmanın verimli olduğunu göstermektedir.

Bu makalede 2015 yılı, 9-11 Eylül tarihlerinde Orta Doğu Teknik Üniversitesinde (ODTÜ) düzenlenen 35. Ulusal Yöneylem Araştırması ve Endüstri Mühendisliği (YAEM-2015) Kongresinde sunulan algoritma iyileştirilmiş ve örnekler üzerinde test edilmiştir [8].

Kaynakça

[1] Gutin, G., Punnen, A.P. 2002. The Travelling Salesman Problem and Its Variations, Kluwer Academic Publishers, 830p.

[2] Lawler, E.L., Lenstra, J.K., Rinnoy Kan, A.H.G., Shmoys, D.B. 1986. The Travelling Salesman Problem: A Guided Tour of Combinatorial Optimization, John Wiley&Sons.

[3] Garey, M.R., Johnson D. S. 1979. Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, San Francisco, 338p.

[4] Johnson, D.S., McGeoch, L.A. 1995. The Traveling Salesman Problem: A Case Study. Pp. 215-310. E. H. L. Aarts and J. K. Lenstra (Editors). Local Search in Combinatorial Optimization. Publisher: Wiley and Sons, New York.

[5] Diaby, M., Karwan M. H. 2016. Advanced in Combinatorial Optimization: Linear Programming Formulations of the Traveling Salesman and Other Hard Combinatorial Optimization Problems, World Scientific Publishing Company, 220p.

[6] Karagul, K., Aydemir, E., Tokat, S. 2016. Using 2-Opt Based Evolution Strategy for Traveling Salesman Problem. An International Journal of Optimization and Control: Theories and Applications (IJOCTA), 6(2).

[7] Library of Traveling Salesman Problems, http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp/ (Erişim Tarihi: 21.02.2016).

[8] Nuriyeva, F., Kızılateş, G. 2015. Gezgin Satıcı Problemi için Merkezden Kenarlara Hipersezgisel Algoritması. Yöneylem Araştırması ve Endüstri Mühendisliği (YAEM-2015) 35. Ulusal Kongresi, 9-11 Eylül, Ankara, s. 203.