Araştırma Makalesi
Zeki Sistemler Teori ve Uygulamaları Dergisi 6(1) (2023) 24-33
DOI: 10.38016/jista.986793

Yerel Arama Bölümü Güncellenmiş Arı Algoritması ile Gezgin Satıcı Problemi Optimizasyonu

Murat Şahin[1*]

[1] Kontrol Sistemleri Geliştirme Müdürlüğü, Roketsan A. Ş., Ankara, Türkiye

msahin@roketsan.com.tr

Öz

Klasik optimizasyon yöntemleri ile çok sayıda bağlantıya sahip gezgin satıcı problemlerinin çözülebilmesi zordur. Bu kapsamda, aramalarını optimum bir çözüme yönlendiren meta-sezgisel algoritmalar tercih edilmektedir. Bu çalışmada, bu meta-sezgisel algoritmalardan biri olan ve bal arılarının yiyecek arama yöntemlerinden esinlenerek geliştirilen Arı Algoritması incelenmiştir. Çalışmanın amacı, Arı Algoritmasının gezgin satıcı problemlerinin çözümüne yönelik etkinliğinin artırılmasıdır. Klasik Arı Algoritması içerisine Değişken Çoklu Ekleme operatörü eklenmiş ve yakın komşuluk bölgeleri içerisinde arama yapılarak, farklı gezgin satıcı problemleri için testler yapılmıştır. Yapılan testler sonucunda bu algoritma ile literatürdeki diğer Arı Algoritmalarına göre çok daha iyi sonuçlar elde edildiği görülmüştür. Geliştirilen algoritma ile 100 şehirlik problemlerde sapma değerleri %1,40-2,80 aralığından %0,11-0,50 aralığına ve 200 şehirlik problemlerde de %8,10-9,67 aralığından %2.00-2,79 aralığına indirildiği gözlemlenmiştir.

Anahtar kelimeler: Arı Algoritması, değişken çoklu ekleme, gezgin satıcı problemi, komşuluk, optimizasyon

Traveling Salesman Problem Optimization with Local Search Section Updated Bees Algorithm

Abstract

Solving traveling salesman problems with many connections with classical optimization methods is challenging. In this context, meta-heuristic algorithms that direct their searches to an optimum solution are preferred. In this study, the Bee Algorithm, one of these meta-heuristic algorithms inspired by the foraging methods of honey bees, was examined. The study aims to increase the Bees Algorithm's effectiveness in solving traveling salesman problems. The Variable Multiple Insertion operator has been added to the Classical Bee Algorithm, and optimizations have been made for different traveling salesman problems by searching within close neighborhood regions. As a result of the tests, it was seen that much better results were obtained with this algorithm compared to other Bee Algorithms in the literature. The algorithm has reduced from 1.40-2.80% to 0.11-0.50% in problems of 100 cities and from 8.10-9.67% to 2.00-2.79% in issues of 200 cities.

Keywords: The Bees Algorithm, optimization, the traveling salesman problem, variable multiple insertion, neighborhood

1. Giriş (Introduction)

Optimizasyon, belirli şartlar altında en iyi çözümü bulma işlemidir ve mühendislik çalışmalarının önemli bir parçasıdır. Mühendisliğin tasarım, geliştirme ve üretim gibi alanlarında yapılan çalışmalarda, harcanılan çabayı, maliyeti minimuma indirmek veya kazancı maksimuma çıkarmak hedeflenir. Bu hedeflerin amaç fonksiyonları belirli kısıtlar altında formüle edilir ve optimizasyon çalışmaları ile fonksiyonların maksimum veya minimum değerini veren değişkenler bulunur. (Rao, 2019).

Tüm optimizasyon problemlerini verimli bir şekilde çözmek için kullanılabilecek tek bir yöntem yoktur. Bu nedenle, farklı tipte optimizasyon problemlerinin çözümü için bir dizi optimizasyon yöntemi geliştirilmiştir. Bu yöntemlerden matematiksel optimizasyon teknikleri; öngörülen bir kısıtlamalar seti altında çeşitli değişkenlerden oluşan bir fonksiyonunun

* Sorumlu yazar.
E-posta adresi: msahin@roketsan.com.tr

Alındı : 24 Ağustos 2021
Revizyon : 17 Ocak 2022
Kabul : 12 Ocak 2023

minimumunu bulmakta yararlıdır. Stokastik süreç teknikleri; bir dizi rasgele değişken tarafından tanımlanan problemleri analiz etmek için kullanılabilir. İstatistiksel yöntemler; kişinin deneysel verileri analiz etmesini ve fiziksel durumun en doğru halini elde etmek için ampirik modeller oluşturmasını sağlar (Manea vd., 2019). Geleneksel olmayan optimizasyon yöntemleri olarak da adlandırılan modern optimizasyon yöntemleri ise, son yıllarda, karmaşık mühendislik problemlerini çözmek amacıyla ortaya çıkmıştır. Yöntemlerin temelinde sezgisel fikri bulunmaktadır (Khan vd., 2015). Bu yöntemlerin çoğu biyolojik, moleküler, böcek sürüsü ve nörobiyolojik sistemlerin belirli özelliklerine ve davranışlarına dayanmaktadır. Aramalarını optimal bir çözüme doğru yönlendirmek için, doğanın ilkelerini taklit ederler. Tepe tırmanışı (hill climbing) ve rastgele yürüme (random walk) gibi standart arama algoritmalarından en önemli farklılıklardan biri, tek bir çözüm yerine her yineleme için bir çözüm popülasyonu kullanmasıdır. Bir optimizasyon probleminin tek bir optimum değeri varsa, tüm popülasyon üyelerinin bu optimum çözüme yaklaşması beklenir (Hussain vd., 2019). Meta-sezgisel algoritmalar günümüzde büyük veri analizinden, 3 boyutlu paketleme ve depolama problemlerine kadar birçok farklı alanda kullanılmaktadır (Bayraktar vd., 2021; Erdem vd., 2021).

Meta-Sezgisel algoritmaların tercih edildiği alanlardan biri de Gezgin Satıcı Probleminin de (GSP) dahil olduğu kombinatoryal optimizasyon problemleridir. GSP, çok sayıda şehirde satış yapacak bir satış elemanının, bu şehirleri ziyaret ederken minimum yol kat etme prensibine dayanır (Daoqing ve Mingyan, 2020). Bilgisayar ağlarının oluşturulması, lojistik planlama, otonom araçların hareket planı vb. birçok alanda kullanılmaktadır (Xie vd., 2019). Literatürde inceleme yapıldığında GSP optimizasyonu için, Aslan Sürüsü Optimizasyonu (Daoqing ve Mingyan, 2020), Karınca Kolonisi Optimizasyonu (Mavrovouniotis vd., 2017), Yapay Arı Kolonisi Algoritması (Dong vd., 2019), Benzetilmiş Tava Algoritması (Wang vd., 2019) ve Genetik Algoritma (Al-Furhud ve Ahmed, 2020) gibi sezgisel algoritmaların kullanıldığı görülmektedir.

Popüler meta-sezgisel algoritmalardan birisi de Arı Algoritması'dır (AA). Pham ve arkadaşları tarafından 2005 yılında önerilen AA, popülasyon temelli arama algoritmasıdır. Algoritma, bal arılarının nektar kaynağı arama davranışınlarını taklit eder. Temelde, rastgele arama ile birlikte bir tür komşu bölge araması yapar ve hem tümleşik hem de fonksiyonel optimizasyon için kullanılabilir (Pham vd., 2006). Temel algoritmanın bünyesinde herhangi bir denklem bulunmaz, bu nedenle diğer algoritmalara göre daha basit bir yapısı vardır. Basit yapısına rağmen, birçok farklı uygulama alanında kullanılabilecek esnek ve güçlü bir algoritmadır (Zarchi ve Attaran, 2017). AA'nın; mekanik parçaların tasarımlarının optimize edilmesi (Nafchi vd., 2012), kontrol sistemleri optimizasyonu (Coban ve Ercin, 2012), robotik parçaların tasarımı (Acar vd., 2018), devre tasarımı optimizasyonu (Mollabakhshi ve Eshghi, 2013), araç yönlendirme planlaması (Alzaqebah vd., 2018), elektrik motoru tasarımı optimizasyonu (Braiwish vd., 2014) gibi birçok farklı alanda başarı ile kullanıldığı bilinmektedir. Bununla birlikte algoritmanın, GSP kapsamında iyileştirme çalışmaları devam etmektedir. 51 şehirlik problemler (Koç, 2010) ile başlayan bu çalışmalar, sonraki yıllarda 100 şehirlik (Otri, 2011) ve 200 şehirlik (Zeybek vd., 2021; Ismail vd., 2020) problemler ile devam etmiştir. İlgili çalışmalar incelendiğinde, 100 şehirin üzerindeki problemler için en iyi değerlere yaklaşılamadığı görülmektedir. Bu durum, bu çalışmanın hazırlanmasındaki ana motivasyon kaynağı olmuştur. Makalenin ikinci bölümünde AA'nın genel yapısı detaylı bir şekilde incelenmiştir. Üçüncü bölümde GSP'nin yapısı ve AA'nın uygulanışı ele alınmıştır. Bu bölümde lokal arama kapsamında daha önce kullanılan standart operatörlere ek olarak değişken çoklu ekleme operatörü eklenmiştir. Komşuluk araması ise yakın bölgeleri kapsayacak şekilde daraltılmıştır. Dördüncü bölümde ise test sonuçları ve analizler mevcuttur.

2. Arı Algoritması (The Bees Algorithm (BA))

AA, popülasyon tabanlı bir arama algoritmasıdır. Algoritma, lokal ve global arama bölümü olmak üzere, temel olarak 2 bölümden oluşur. Önemli parametreler; keşif (scout) arıların sayısı (n), yerel arama için iyi bölgelerin (best site) sayısı (m), yerel arama için elit bölgelerin (elite site) sayısı (e), elit bölgelere yollanan arıların (recruited bees) sayısı (nep), iyi bölgelere yollanan arıların sayısı (nsp), komşuluk boyutu (ngh) ve yineleme sayısıdır (I) (Baronti vd., 2020).

Algoritmanın sözde-kodu Şekil 1'de verilmiştir. Algoritma, parametrelerin belirlenmesi ile başlar (1. adım). Daha sonra keşif arılar *n* adet arama alanına rastgele yerleştirilir ve ilk popülasyon oluşturulur (2. adım). İlk popülasyon, minimizasyon problemleri için küçükten büyüğe, maksimizasyon problemleri için ise büyükten küçüğe doğru sıralanır (3. adım). Sonraki adımda lokal arama bölümü başlar ve ilk olarak elit bölge araması yapılır (5-11 adımlar). Her elit bölge için *nep* sayıda, *ngh* komşuluk sınırları içerisinde rastgele denemeler yapılır. Bu denemeler sonucunda bulunan en iyi değer, önceki popülasyondaki değerden daha iyi ise ilgili bölge için kaydedilir. 12-18 adımlar arasında ise iyi bölge araması yapılır. Her iyi bölge için *nsp* sayıda, *ngh* komşuluk sınırları içerisinde rastgele denemeler yapılır ve elit bölgede olduğu gibi bu denemeler sonucunda bulunan en iyi değer, önceki popülasyondaki değerden daha iyi ise ilgili bölge için kaydedilir.

Sonraki adımlarda global arama işlemi gerçekleştirilir (19-21). n bölge içinde, lokal aramanın yapılmadığı bölgeler için rastgele yeni değerler atanır. Bu bölüm özellikle lokal minimumlara takılmamak için önemlidir. Lokal aramada ulaşılamayan bölgelerde daha iyi değerler var ise onlara ulaşılmasını sağlayabilir.

Sonraki adımda popülasyon yeniden sıralanır ve elit-iyi bölgeler güncellenir. Böylece her yinelemede, elit bölgelerde daha çok arama yapılacaktır. Bu adımlar, belirli sonlandırma kriteri karşılanana kadar tekrarlanır (Baronti vd., 2020; Yüce vd., 2014).

```
1: Algoritma parametrelerini belirle
2: İlk popülasyonu rastgele üret
3: Popülasyonu sırala
4: while (durdurma kriteri sağlanmadıysa) do
// Elit bölge
5:    for i=1:e
6:       for j=1:nep
7:          n(i) için ngh komşuluk araması yap
8:          n(i)' nin en iyi komşusunu belirle
9:       end
10:      En iyi komşu, n(i)'den iyi ise kayıt et
11:   end
// İyi bölge
12:   for i=e+1:m
13:      for j=1:nsp
14:         n(i) için ngh komşuluk araması yap
15:         n(i)' nin en iyi komşusunu belirle
16:      end
17:      En iyi komşu, n(i)'den iyi ise kayıt et
18:   end
// Global arama
19:   for i=m+1:n
20:      Rastgele yeni bölgeler üret
21:   end
22:   Popülasyonu sırala
23: end while
```

Şekil 1. AA Sözde Kodu
(Pseudo Code of the BA)

3. AA ile GSP Optimizasyonu (Optimisation of TSP with the BA)

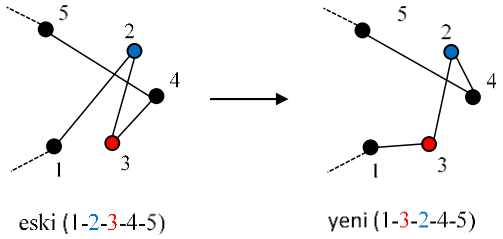
GSP'de turun maliyeti doğrudan turun uzunluğuna bağlıdır. Birbirini izleyen iki şehir arasındaki mesafe aşağıdaki şekilde tanımlanır (Karaboğa ve Görkemli, 2019):

$$d\big(T(i),T(i+1)\big)=\sqrt{(x_i-x_{i+1})^2+(y_i-y_{i+1})^2} \quad (1)$$

x ve y, şehirlerin koordinatlarını gösterir. Denklem (2)'de toplam tur mesafesinin hesaplanması gösterilmiştir (Karaboğa ve Görkemli, 2019):

$$\sum_{i=1}^{n-1} d\big(T(i),T(i+1)\big)+d\big(T(n),T(1)\big) \quad (2)$$

Denklemde, n toplam şehir sayısını temsil eder. GSP'ye yönelik hazırlanan optimizasyon algoritmalarında, arama bölümleri ve komşuluk tanımları sürekli problemlerden farklıdır. Genel olarak güzergah üzerindeki şehirlerin konumu belirli kurallara göre değiştirilerek daha az maliyetle yeni güzergahlar bulunmaya çalışılır. AA için, Ismail ve ark. lokal arama bölümünde (Ismail vd., 2020) Takas, Ekleme ve Ters Çevirme olmak üzere üç farklı operatör önermişlerdir. Takas operatöründe rastgele seçilen iki şehrin sırası birbiriyle değiştirilir (Ismail vd., 2020). Bu çalışmada ise birinci şehir tüm şehirler içinden rastgele seçilirken, ikinci şehir ilkinin yakın komşuluğunda rastgele seçilir. Şekil 2 ve 3'te takas operatörünün uygulanışı ve sözde kodu verilmiştir.

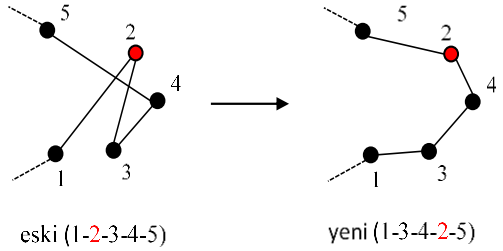


Şekil 2. Takas İşlemi (Swap)

```
1: ss:sehir sayısı
2: n_i:seçili rota
3: Function Takas (n_i,ss)
4:   Rastgele 1. sayıyı üret s1
//   (1 – ss arasında)
5:   Komşulukta s2 seç
//   (s1 hariç)
6:   n_i(s2)=n_i(s1)
7:   n_i(s1)=n_i(s2)
8: End Function
```

Şekil 3. Takas İşlemi Sözde Kodu (Psedue Code of Swap)

Ekleme operatöründe, rastgele seçilen bir şehrin sıralaması, yakın komşulukta rastgele seçilen farklı bir sıraya eklenir (Ismail vd., 2020). Şekil 4'te örnek üzerinden gösterilmiştir. Görüldüğü üzere ilk güzergahdaki 2 numaralı şehrin yeri, 4 numaralı şehirden sonraki bölüme eklenmiş ve çok daha iyi bir güzergah elde edilmiştir.

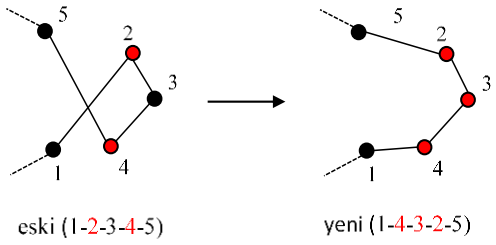


Şekil 4. Ekleme İşlemi (Insertion)

Tersine çevirme operatöründe ise, rastgele belirlenen iki şehir arasındaki sıralama tersine çevrilir (Ismail vd., 2020). Şekil 6'da örnek üzerinden gösterilmiştir. İlk güzergahdaki 2 ve 4 numaralı şehirler arasındaki sıralama tersine çevirilmiştir.

```
1: ss:sehir sayısı
2: n_i:seçili rota
3: Function Ekleme (n_i,ss)
4:   Rastgele 1. sayıyı üret s1
5:   Yakın Komşulukta s2 seç
//   (s1 ve s1-1 hariç)
6:   if s1<s2
7:     n_i=n_i([1:s1-1  s1+1:s2  s1  s2+1:end])
8:   else
9:     n_i=n_i([1:s2  s1  s2+1:s1-1  s1+1:end])
10:  end
11: End Function
```

Şekil 5. Ekleme İşlemi Sözde Kodu (Psedue Code of Insertion)



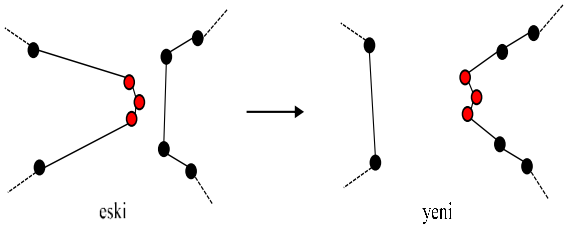
Şekil 6. Tersine Çevirme İşlemi (Reversion)

```
1: ss:sehir sayısı
2: n_i:seçili rota
3: Function TersCevirç (n_i,ss)
4:   Rastgele 1. sayıyı üret s1
5:   Komşulukta 2. sayıyı üret s2
6:   if s1<s2
7:     n_i(s1:s2)= n_i(s2:-1:s1)
8:   else
9:     n_i(s1:-1:s2)= n_i(s2:s1)
10:  end
11: End Function
```

Şekil 7. Tersine Çevirme İşlemi Sözde Kodu (Psedue Code of Reversion)

Bu çalışmada 3 operatöre ek olarak değişken çoklu ekleme operatörü eklenmiştir. Şekil 3'ten görüldüğü üzere, ekleme operatörünün güzergah üzerindeki etkisi çok güçlüdür. Özellikle şehir sayısının 100 ve üzerinde olduğu problemlerde, birden fazla şehrin yakın bölgelerde bulunduğu ve topluca yer değişimine uygun olduğu görülmektedir. Tek şehirli ekleme operatörünün aksine, rastgele belirlenen bir bölgeden birden çok ve değişken sayıda şehirler alınır. Sayının değişken olması ise, kümelenen şehir sayısının da değişken olmasından kaynaklanmaktadır. Bu şehirler komşulukta belirlenen ikinci alana yerleştirilir. Çoklu seçim yapılırken, seçilecek şehir sayısı ise deneysel çalışmalar ile belirlenmiştir.

Değişken çoklu ekleme operatörünün güzergah üzerindeki etkisi Şekil 7'de ve sözde kodu Şekil 8'de gösterilmiştir.



Şekil 7. Değişken Çoklu Ekleme İşlemi (Variable Multiple Insertion)

```
1: ss:sehir sayısı
2: n_i:seçili rota
3: Function CokluEkleme(n_i,ss)
4:   Rastgele sayı üret c (2-7 arasında)
5:   Rastgele 1. bölgeyi seç s1
6:   for i=1:c
7:     dizi(i)=(s1+i-1)
8:   end
9:   Komşulukta rastgele 2. bölgeyi seç s2
10: if s1<s2
11:   n_i=n_i([1:s1-1  s1+c:s2  s1:s1+c-1  s2+1:end])
12: else
13:  n_i= n_i([1:s2  s1:s1+c-1  s2+1:s1-1  s1+c:end])
14: end
15: End Function
```

Şekil 8. Değişken Çoklu Ekleme İşlemi Sözde Kodu (Psedue Code of Variable Multiple Insertion)

Çalışmada geliştirilen algoritmanın sözde kodu Şekil 9'da gösterilmiştir. Görüldüğü üzere genel AA yapısı içerisinde 4 farklı operatör seçimi bulunmaktadır. Her bir yinelemede, bu operatörlerden biri rastgele seçilerek uygulanmaktadır. Elit bölgelerde daha fazla sayıda arama yapılmaktadır. (21-36 arası bölümler, 5-20 arası bölüm ile aynı olup sadece *nep* yerine *nsp* gelmektedir.)

Algoritma parametreleri belirlenirken diğer kombinatoryal problemlere yönelik hazırlanan çalışmalarda kullanılan değerler dikkate alınmıştır (Alzaqebah vd., 2018; Zeybek vd., 2021; Ismail vd., 2020; Yuce vd., 2014). Bu çalışmalarda popülasyon sayıları 20 ile 100 aralığında değişmektedir. İşlem gücünü düşürmek amacıyla popülasyon sayısı 20 olarak belirlenmiştir. GSP'lerdeki olası çözümlerin sayısı doğrudan şehir sayısına bağlıdır ve (n-1)!/2 ile ifade edilir (Ahmed, 2010). Bu durumda rastgele oluşturulmuş bir rotanın optimum rotaya yakın olması pek olası değildir. Bu nedenle çalışmada lokal arama bölgesi de 20 olarak belirlenmiştir ve global arama bölümü kullanılmamıştır. (Analiz bölümünde global aramanın da olduğu denemeler gerçekleştirilmiş ve sonuçlar gösterilmiştir.) Elit bölge sayısı popülasyonun %20'si yani 4 olarak belirlenmiştir. Elit bölgede arama için 300 arı, diğer bölgelerde de 100 arı kullanılmıştır. Tüm parametreler Tablo 1'de verilmiştir. Algoritmanın tekrarlanabilirliğini görmek amacıyla, tüm problemler için ard arda 10ar test yapılmıştır.

```
1: Algoritma parametrelerini belirle
2: İlk popülasyonu rastgele üret
3: Popülasyonu sırala
4: while (durdurma kriteri sağlanmadıysa) do
5:    for i=1:e
6:       for j=1:nep
7:          Rastgele sayı üret s (1-4)
8:          if s==1
9:             Takas(n(i))
10:         elseif s==2
11:            Ekleme(n(i))
12:         elseif s==3
13:            TersCevir(n(i))
14:         else
15:            CokluEkleme(n(i))
16:         end
17:         n(i)' nin en iyi komşusunu belirle
18:      end
19:      En iyi komşu, n(i)'den iyi ise kayıt et
20:   end
21:   for i=e+1:m
22:      for j=1:nsp
23:         Rastgele sayı üret s (1-4)
24:         if s==1
25:            Takas(n(i))
26:         elseif s==2
27:            Ekleme(n(i))
28:         elseif s==3
29:            TersCevir(n(i))
30:         else
31:            CokluEkleme(n(i))
32:         end
33:         n(i)' nin en iyi komşusunu belirle
34:      end
35:      En iyi komşu, n(i)'den iyi ise kayıt et
36:   end
37:   Popülasyonu sırala
38: end while
```

Şekil 9. GSP için AA Sözde Kodu
(BA Pseudo Code for TSP)

Algoritmayı test etmek amacıyla TSPLIB (Library of Traveling Salesman Problem) kütüphanesi içerisinden, farklı sayılara ve yerleşim planlarına sahip problemlerden bazıları seçilmiştir (TSPLIB, 2020). Sonuçları optimum değerlerle karşılaştırmak için, hem 10 tekrarın ortalamalarının hem de bu tekrarlar içerisinde bulunan en iyi değerlerin yüzde olarak sapmaları da hesaplanmıştır. Bu kapsamda (3) ve (4)'te verilen eşitlikler kullanılmıştır (Ismail vd., 2020).

Tablo 1. AA Parametreleri
(BA Parameters)

Parametre Tipi	*Parametre Değeri*
Toplam test sayısı	*10*
İterasyon sayısı	*1000*
Keşif arısı sayısı (n)	*20*
Elit bölge sayısı (e)	*4*
İyi bölge sayısı (m)	*20*
Elit bölgedeki arı sayısı (nep)	*300*
İyi bölgedeki arı sayısı (nsp)	*100*
Komşuluk (ngh)	*10*

$$S\_ort = \frac{(Ort.-Opt.)}{(Opt.)} * 100\% \quad (3)$$

$$S\_min = \frac{(Min.-Opt.)}{(Opt.)} * 100\% \quad (4)$$

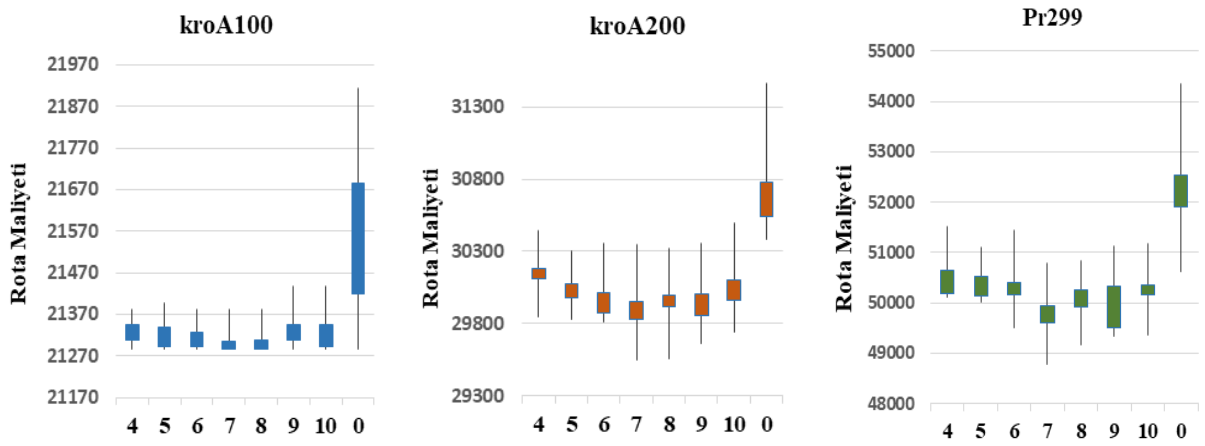
Eşitliklerde "Ort." ile testlerin ortalama değeri, "Opt." ile problemin bilinen optimum değeri, "Min." ile testlerde elde edilen en iyi değer gösterilmektedir.

4. Deneysel Çalışmalar ve Sonuçlar
(Experimental Studies and Results)

Algoritma Matlab R2018a programında geliştirilmiştir. Çalışmaların yapıldığı bilgisayar, Lenovo Thinkpad, Intel Core i5-8265U CPU 1.60 GHz, 8 GB RAM ve Windows 10 Pro özelliklerine sahiptir

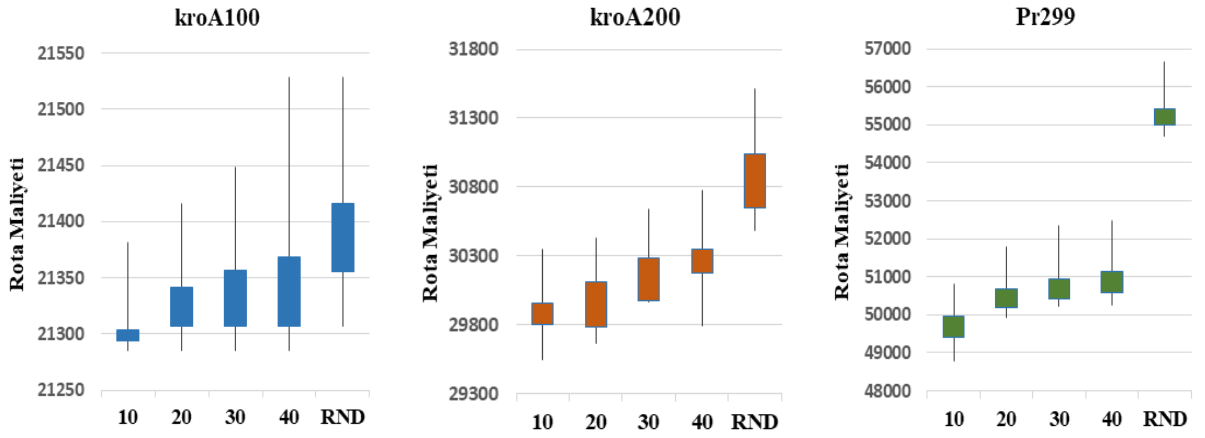
4. 1. Algoritmanın Analizi **(Analysis of Algorithm)**

Algoritmayı analiz etmek için 100, 200 ve 299 şehirlik problemler için deneyler gerçekleştirilmiştir. Her deney için 10 farklı tekrar yapılmış ve sonuçlar kutu grafiklerinde gösterilmiştir. İlk analiz çalışması, çoklu ekleme operatörü için seçilecek şehir sayısı limitini belirlemek için yapılmıştır. Bu kapsamda şehir sayılarının üst limiti 4 ile 10 arasında değiştirilmiştir. Analiz sonuçları Şekil 10'da gösterilmiştir.



Şekil 10. Çoklu Ekleme Operatörü Analizi (Analysis of Variable Multiple Insertion)

Çoklu ekleme operatöründe, üst limit olarak 6, 7 ve 8 değerleri için en iyi sonuçların elde edildiği görülmüştür. Bunlar arasından en iyisi 7 olduğu için, operatörde şehir sayıları 2 ile 7 arasında rastgele değişecek şekilde ayarlanmıştır. Operatörün kullanılmadığı durum ise Şekil 10'da "0" ile gösterilmiştir. İkinci analiz çalışması komşuluk değeri için gerçekleştirilmiştir. Bu kapsamda en yakın 10, 20, 30, 40 komşu için ve ayrıca yakın komşuluğun olmadığı yani rastgele aramanın yapıldığı deneyler gerçekleştirilmiştir. Analiz sonuçları Şekil 11'de gösterilmiştir.



Şekil 11. Komşuluk Analizi (Analysis of Neighborhood)

Şekil 11'den görüleceği üzere en iyi sonuçlara, yakın komşuluk değerlerinde ulaşılmaktadır. Harita üzerinde şehirler arası bağlantıları kendilerine yakın komşular üzerinden gerçekleştirmek rota maliyetlerini düşürmektedir. Algoritmada komşuluk limit değeri 10 olarak belirlenmiştir. Operatörlerde seçim yapılırken ikinci şehir, ilk belirlenen şehire en yakın 10 komşu şehir arasından rastgele belirlenmektedir. Grafiklerde RND ile komşuluk sınırlamasının olmadığı durum gösterilmiştir. Üçüncü analiz çalışması global aramanın etkinliği üzerine gerçekleştirilmiştir. Global arama için popülasyon sayısı 100'e çıkarılmıştır, bunlardan 20'si lokal arama bölgesi, 80'i global arama bölgesidir. Analiz sonuçları Tablo 2'de gösterilmiştir. Tablodan görüldüğü üzere global aramanın dahil olduğu durum ile olmadığı durum arasındaki değerler çok yakındır. Bu nedenle algoritmada global arama bölümü kullanılmamıştır.

Tablo 2. Global Arama Sonuçları (Results of Global Search)

GSP (Opt.)	Global Olmadan		Global Dahil	
	S_min	S_ort	S_min	S_ort
KroA100 (21282)	0.02	0.11	0.02	0.11
KroA200 (29368)	0.62	2.00	0.60	2.04
Pr299 (48191)	1.20	3.40	1.27	3.37

4. 2. Deneysel Çalışmalar (Experimental Studies)

Deneylerde kullanılan problemlerin isimleri, optimum rota uzunlukları ve testlerde elde edilen sonuçlar Tablo 3'te verilmiştir. Tabloda Ort. Süre ile en iyi değerlerin bulunma süreleri gösterilmiştir. Tablodan görüleceği üzere 52 ile 124 arasında şehire sahip problemler için bilinen en iyi değerlere veya %0.03'lere

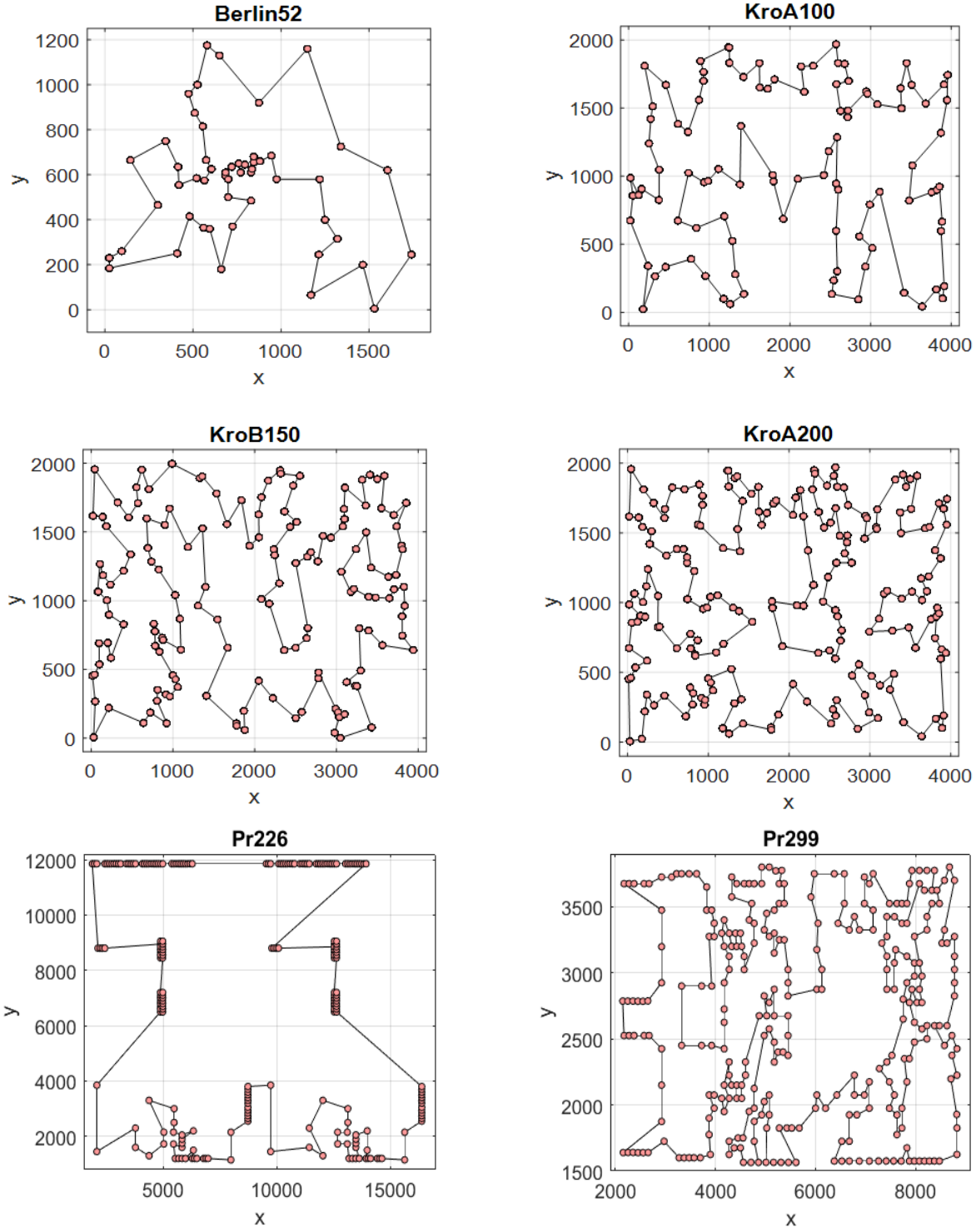
varan sapma değerlerine ulaşılmıştır. Bu problemlerde 10 testin ortalamasında da %0.50'den küçük sapma değerleri elde edilmiştir. 200 şehirlik problemlerin çözümlerinde ise ortalama değerler %2.00-2.79 aralığında elde edilmiştir. 1000 iterasyonluk çalışma süreleri; 100 şehirlik problemler için 20 saniye, 200 şehirlik problemler için ise 80 saniye civarlarında olmuştur. (Çalışma süreleri Matlab'daki tic-toc fonksiyonları ile hesaplanmıştır.)

Algoritmayı kıyaslamak için, farklı yapıda AA ile gerçekleştirilmiş son güncel çalışmalar seçilmiştir. Karşılaştırma sonuçları Tablo 4'te verilmiştir. Bu çalışmada elde edilen sonuçlar İyileştirilmiş Arı Algoritması (I-AA) altında verilmiştir. Karşılaştırılan çalışmaların ilkinde Ismail ve ark. (Ismail vd., 2020) çalışmalarında lokal arama için Takas, Ekleme ve Ters Çevirme operatörlerini kullanmışlardır. Komşuluk aramalarında tüm bölgeyi kullanmışlar ve yakın komşuluklara öncelik verilmemiştir. (Bu algoritmanın sonuçları Tablo 4.'te Klasik Arı Algoritması (K-AA) altında verilmiştir.) İkinci çalışmada ise Zeybek ve ark. (Zeybek vd., 2021), lokal arama bölümünde Bakış Noktası (Vantage Point) yöntemini kullanmışlardır. (Bu algoritmanın sonuçları Tablo 4.'te Bakış Noktası Arı Algoritması (BN-AA) altında verilmiştir.)

Tablo 3. GSP Verileri ve Test Sonuçları (TSP Data and Test Results)

GSP (Opt.)	Min.	Ort.	S_min	S_ort	Ort. Süre (s)
Berlin52 (7542)	7544	7544	0.03	0.03	7.98
Pr76 (108159)	108159	108277	0	0.11	17.17
KroA100 (21282)	21285	21304	0.02	0.11	27.36
KroB100 (22141)	22139	22251	0	0.50	24.86
KroC100 (20749)	20750	20814	0.01	0.31	22.41
KroD100 (21389)	21294	21449	0	0.28	25.40
KroE100 (22068)	22073	22141	0.02	0.33	25.22
Pr124 (59030)	59030	59095	0	0.11	26.88
Pr144 (58537)	58535	58731	0	0.33	41.11
Ch150 (6528)	6564	6660	0.56	2.03	36.18
KroA150 (26524)	26671	26954	0.55	1.62	44.87
KroB150 (26130)	26275	26425	0.56	1.13	39.18
KroA200 (29368)	29549	29955	0.62	2.00	79.98
KroB200 (29437)	30056	30258	2.10	2.79	80.73
Pr226 (80369)	80860	81397	0.61	1.28	86.11
Gil262 (2378)	2454	2471	3.22	3.91	95.43
Pr299 (48191)	48771	49828	1.20	3.40	110.83
Lin318 (42029)	43589	44090	3.71	4.90	111.05
Pcb442 (50778)	53715	55011	5.78	8.33	136.3

Tablo 4'te bu çalışmada geliştirilen algoritmanın, AA ile gerçekleştirilen diğer güncel çalışmalar (Zeybek vd., 2021; Ismail vd., 2020) ile kıyaslandığında çok daha iyi sonuçlar elde ettiği görülmektedir. 100 şehirlik problemlerde ortalmada %0.25-0.50 aralığında sapmalarla optimum değerlere ulaşılabilmektedir. Diğer algoritmaların ulaşabildiği en iyi değer KroE100 için %1.39 olabilmiştir. KroE100 için I-AA ile %0.33 sapma değeri ile çözüm üretilmiştir. Şehir sayısının artması ile beraber diğer algoritmalar ile olan sapma değerleri farkları da artmaktadır. 200 şehirlik problemlerde I-AA ile %2.00-2.79 aralığında sapma değerleri elde edilirken, BN-AA ile %8.10-9.67 sapma değerlerine ulaşılabilmiştir. Diğer algoritmalara göre en belirgin fark değişken çoklu ekleme operatörünün bulunmasıdır. Bu operatör sayesinde, bir adımda çok sayıda şehrin yeri değiştirilerek, daha az sürede daha iyi rotalar elde edilebilmektedir. Diğer önemli faktör ise yakın komşuluk seçimleridir. Testlerde elde edilen en iyi sonuçlardan bazıları Şekil 12'de gösterilmiştir.



Şekil 12. GSP Testleri Sonuçları (Results of TSP Tests)

5. Sonuçlar (Conclusions)

Bu çalışmada, lokal arama bölgesi güncellenmiş AA ile Gezgin Satıcı Problemlerinin çözümü üzerinde durulmuştur. Lokal arama bölümüne, genellikle kullanılan Takas, Ekleme ve Ters Çevirme operatörlerine ek olarak, değişken çoklu ekleme operatörü eklenmiştir ve arama işlemi yakın komşuluğu kapsayacak şekilde daraltılmıştır. Bu iyileştirmelerin etkileri, AA ile yapılan diğer çalışmalar ile kıyaslandığında doğrudan görülmektedir. 100 şehirlik problemlerde 10 deneme içerisinde, en iyi çözümlere veya %0.02lik sapma değerleri ile çözümlere ulaşılabilmiştir. 150 şehirlik problemlerde %0.5 civarında, 200 şehirlik problemlerde %0.6-2.1 aralığında ve 318 şehirlik problemde ise %3.71

civarında sapma değerleri ile çözümlere ulaşılabilmiştir. Bu değerlerin diğer çalışmaların sonuçlarından çok daha iyi olduğu karşılaştırma tablosunda görülmektedir. Çözüm süreleri ise 100 şehirli problemler için 20 saniye, 200 şehirlik problemler için 80 saniye civarında olmuştur. Benzer çalışmalarda sürelerle ilgili bilgi olmadığı için karşılaştırma yapılamamıştır. Bu çalışmada elde edilen sonuçlar, diğer araştırmacıların çalışmalarına yönelik karşılaştırma fırsatı da vermektedir. Gelecek çalışmalar kapsamında özellikle 200 üzeri şehire sahip problemler için, lokal arama bölümüne farklı operatörler eklenerek iyileştirme çalışmaları yapılabilir.

Tablo 4. Algoritmaların Karşılaştırılması (Comparison of Algorithms)

GSP (Opt.)	**I-AA**		K-AA		BN-AA	
	S_min	S_ort	S_min	S_ort	S_min	S_ort
Berlin52 (7542)	0.03	**0.03**	-	-	**0**	0.63
KroA100 (21282)	**0.02**	**0.11**	0.88	2.02	0.42	1.70
KroB100 (22141)	**0**	**0.50**	2.15	3.33	0.26	1.74
KroC100 (20749)	**0.01**	**0.31**	1.06	2.85	0.09	1.81
KroD100 (21389)	**0**	**0.28**	0.01	2.37	0.86	2.84
KroE100 (22068)	**0.02**	**0.33**	0.9	1.94	0.23	1.39
KroA150 (26524)	**0.55**	**1.62**	3.78	5.08	3.06	4.70
KroB150 (26130)	**0.56**	**1.13**	3.69	5.08	2.73	3.79
KroA200 (29368)	**0.62**	**2.00**	7.43	9.69	7.16	8.10
KroB200 (29437)	**2.10**	**2.79**	8.79	10.47	8.05	9.67
Lin318 (42029)	**3.71**	**4.90**	-	-	7.02	7.57

Kaynaklar (References)

Acar, O., Kalyoncu, M., Hassan, A. (2018). The Bees' Algorithm for Design Optimization of a Gripper Mechanism, Journal of Selcuk-Technic (ICENTE'18) Special Issue, 69-86.

Ahmed, Z. H. (2010). Genetic Algorithm for the Traveling Salesman Problem Using Sequential Constructive Crossover Operator. International Journal of Biometrics ve Bioinformatics (IJBB), 3(6), 96–105. doi: 10.14569/IJACSA.2020.0110275.

Al-Furhud, M. A., Ahmed, Z. H. (2020). Genetic Algorithms for the Multiple Travelling Salesman Problem. (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 11, No. 7, pp. 553-560.

Alzaqebah, M., Jawarneh, S., Sarim, H. M., Abdullah, S. (2018). Bees Algorithm for Vehicle Routing Problems with Time Windows, International Journal of Machine Learning and Computing, Vol. 8, No. 3, 236-240.

Baronti, L., Castellani, M., Pham, D. T. (2020). An analysis of the search mechanisms of the bees algorithm, Swarm and Evolutionary Computation, 59, 100746.

Bayraktar, T., Ersöz, F., & Kubat, C. (2021). Effects of Memory and Genetic Operators on Artificial Bee Colony Algorithm for Three-Dimensional Bin Packing Problem (No. 5754). EasyChair.

Braiwish, N. Y., Anayi, F. J., Fahmy, A. A., Eldukhri, E. E. (2014). Design optimisation of Permanent Magnet Synchronous Motor for electric vehicles traction using the Bees Algorithm, 49th International Universities Power Engineering Conference (UPEC), Cluj-Napoca, Romania.

Daoqing, Z., Mingyan, J. (2020). Parallel Discrete Lion Swarm Optimization Algorithm for Solving Traveling Salesman Problem. Journal of Systems Engineering and Electronics, Vol. 31, No. 4, pp.751 – 760.

Coban, R., Ercin, O. (2012). Multi-objective Bees Algorithm to Optimal Tuning of PID Controller, Çukurova University Journal of the Faculty of Engineering and Architecture, 27(2), 13-26.

Dong, X., Lin, Q., Xu, M., Cai, Y. (2019). Artificial bee colony algorithm with generating neighbourhood solution for large scale coloured traveling salesman problem. IET Intelligent Transport Systems, Vol. 13 Iss. 10, pp. 1483-1491.

Erdem, E., Aydın, T., & Erkayman, B. (2021). Flight scheduling incorporating bad weather conditions through big data analytics: A comparison of metaheuristics. Expert Systems, 38(8), e12752.

Hussain, K., Salleh, M. N. M., Cheng, S., Shi, Y. (2019). Metaheuristic research: a comprehensive survey, Artificial Intelligence Review, volume 52, 2191–2233.

Internet: The TSPLIB Symmetric Traveling Salesman Problem Instances, http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html, 22.12.2020.

Ismail, A. H., Hartono, N., Zeybek, S., Pham, D. T. (2020). Using the Bees Algorithm to solve combinatorial optimisation problems for TSPLIB, IOP Conf. Series: Materials Science and Engineering 847, 1-9.

Karaboga, D., Gorkemli, B. (2019). Solving Traveling Salesman Problem by Using Combinatorial Artificial Bee Colony Algorithms, International Journal on Artificial Intelligence Tools Vol. 28, No. 1, 1950004, 1-28.

Khan, S., Asjad, M., Ahmad, A., (2015). Review of Modern Optimization Techniques. International Journal of Engineering and Technical Research V4(04), 984-988.

Koc E. (2010). Bees algorithm: theory, improvements and applications Thesis (Cardiff University, UK).

Manea,D., Titan, E., Serban, R. R., Mihai, M. (2019). Statistical applications of optimization methods and mathematical programming. Proceedings of the International Conference on Applied Statistics 1(1), 312-328.

Mavrovouniotis, M., Müller, F. M., Yang, S. (2017). Ant Colony Optimization With Local Search for Dynamic Traveling Salesman Problems. IEEE Transactions on Cybernetics, Vol. 47, No. 7, pp. 1743-1756.

Mollabakhshi, N., Eshghi, M. (2013). Combinational circuit design using bees algorithm, IEEE Conference Anthology, China.

Nafchi, A. M., Moradi, A., Ghanbarzadeh, A., Yaghoubi, S., Moradi, M. (2012). An Improved Bees Algorithm For Solving Optimization Mechanical Problems, 20th Annual International Conference on Mechanical Engineering-ISME, School of Mechanical Eng., Shiraz University, Shiraz, Iran.

Otri S. (2011). Improving the bees algorithm for complex optimisation problems Thesis (Cardiff University, UK).

Pham, D.T., Ghanbarzadeh, A., Koc, E., Otri, S., Rahim, S., Zaidi, M. (2006). The Bees Algorithm-A Novel Tool for Complex Optimisation Problems, Intelligent production machines and systems, 454-459.

Rao, S., S. (2019). Engineering Optimization Theory and Practice. John Wiley & Sons, New Jersey, A.B.D.

Xie, J., Carrillo, L. R. G., Jin, L. (2019). An Integrated Traveling Salesman and Coverage Path Planning Problem for Unmanned Aircraft Systems. IEEE Control Systems Letters, Vol. 3, No. 1, pp. 67-72.

Wang, L., Cai, R., Lin, M., Zhong, Y. (2019). Enhanced List-Based Simulated Annealing Algorithm for Large-Scale Traveling Salesman Problem. IEEE Access Year: 2019 | Volume: 7, pp. 144366-144380.

Yuce, B., Mastrocinque, E., Lambiase, A., Packianather, M. S., Pham, D. T. (2014). A multi-objective supply chain optimisation using enhanced Bees Algorithm with adaptive neighbourhood search and site abandonment strategy, Swarm and Evolutionary Computation Volume 18, October, 71-82.

Zarchi, M., Attaran, B. (2017). Performance improvement of an active vibration absorber subsystem for an aircraft model using a bees algorithm based on multi-objective intelligent optimization, Engineering Optimization Volume 49, Issue 11, 1905-1921.

Zeybek S., A. H. Ismail, N. Hartono, M. Caterino, K. Jiang. (2021). An Improved Vantage Point Bees Algorithm to Solve Combinatorial Optimization Problems from TSPLIB, Macromolecular Symposia, 396, 2000299, pp. 1–4.