İstanbul Üniversitesi İşletme Fakültesi Dergisi
Istanbul University Journal of the School of Business
Cilt/Vol:43, Sayı/No:1, 2014, 1–27
ISSN: 1303–1732 – www.ifdergisi.org © 2014

Gezgin satıcı problemlerinin metasezgiseller ile çözümü

Sultan Kuzu[1]
Sayısal Yöntemler, İşletme Fakültesi, İstanbul Üniversitesi, İstanbul, Türkiye

Onur Önay[2]
Sayısal Yöntemler, İşletme Fakültesi, İstanbul Üniversitesi, İstanbul, Türkiye

Uğur Şen[3]
Sosyal Bilimler Enstitüsü, İstanbul Üniversitesi, İstanbul, Türkiye

Mustafa Tunçer[4]
Sosyal Bilimler Enstitüsü, İstanbul Üniversitesi, İstanbul, Türkiye

Bahadır Fatih Yıldırım[5]
Sayısal Yöntemler, İşletme Fakültesi, İstanbul Üniversitesi, İstanbul, Türkiye

Timur Keskintürk[6]
Sayısal Yöntemler, İşletme Fakültesi, İstanbul Üniversitesi, İstanbul, Türkiye

Özet

Bu çalışmada, NP-zor problem sınıfından olan gezgin satıcı probleminin (GSP), stokastik optimizasyon tekniklerinin en genel sınıfı olan metasezgisel yöntemlerle çözümü ele alınmıştır. Klasik matematiksel yöntemlerle çözümü zor ve belli bir boyuttan sonra imkânsız olan problemler için metasezgisel yöntemler etkin bir çözüm alternatifidir. Uluslararası literatürde sıklıkla kullanılan metasezgisel yöntemlerin GSP problemlerine uygulanması konusunda genel bakış içeren çalışmaya, ulusal literatürde rastlanmamıştır. Bu amaçla yaygın kullanıma sahip 8 metasezgisel yöntem tanıtılarak bu yöntemler literatürden alınan farklı boyutlarda problemlere uygulanmıştır. Sonuçlar raporlanmış ve farklı açılardan yorumlanmıştır.

Anahtar Kelimeler: *Gezgin Satıcı Problemi, Metasezgisel Yöntemler, Benchmark Problemleri, Optimizasyon*

Solving travelling salesman problems with metaheuristics

Abstract

This study deals with the travelling salesman problem (TSP) with metaheuristics which is the most general class of the stochastic optimization techniques. The TSP is a NP-hard problem in optimization studied in both operations research and computer science. Metaheuristics are efficient alternative techniques for NP-hard and greater dimensional problems and are impossible to solve by classic mathematical techniques. Although widespread uses of metaheuristics exist in international literature, in a study of Turkish literature there were none encountered that contain a general view of them and their applications to the TSP. For this reason, 8 metaheuristic techniques are introduced and applied to different dimensional benchmark problems which are taken from the literature. Results are reported and commented in different ways.

Keywords: *Travelling Salesman Problem, Metaheuristics, Benchmark Problems, Optimization*

[1] *sultan.kuzu@istanbul.edu.tr (S. Kuzu)*

[2] *onur.onay@istanbul.edu.tr (O. Önay)*

[3] *ugursen1991@gmail.com (U. Şen)*

[4] *mtunncer@gmail.com (M. Tunçer)*

[5] *bahadirf.yildirim@istanbul.edu.tr (B. F. Yıldırım)*

[6] *tkturk@istanbul.edu.tr (T. Keskintürk)*

1. Giriş

GSP ilk olarak 1930'lu yıllarda matematiksel olarak tanımlanmıştır. Problem tanımı basit olmasına rağmen çözümü zordur. Problemde kullanılan şehir sayısının artışına paralel olarak çözüm uzayı genişlemekte, problemin çözüm zamanı ve zorluğu artmaktadır. Bu nedenle problemin çözümünde analitik çözüm yöntemleri yetersiz kalmaktadır. Tüm çözüm uzayını taramak yerine mantıksal çıkarımlar ile çözüm uzayında kısmi taramalar yapan metasezgisel yöntemler problemin çözümünü garanti etmemekle beraber, çözüm maliyetini azaltmaktadır.

GSP, tek bir amaç fonksiyonu ve tanımlanması kolay kısıtlardan oluşan bir problem olduğu için yeni geliştirilen metasezgisellerin sınanmasına, mevcut metasezgisellerin karşılaştırılmasına olanak sağladığından araştırmacı ve akademisyenler tarafından çalışmalarda sıklıkla kullanılmaktadır. GSP'nin literatürde sıklıkla kullanılmasının bir başka nedeni ise gerçek hayat problemlerine kolaylıkla uyarlanabilir olmasıdır.

Bu çalışmada, iyi bilinen ve geniş bir kullanım alanına sahip metasezgiseller GSP'ye uygulanmış ve sonuçlar ayrıntılı olarak karşılaştırılmıştır. Birinci bölümde GSP'nin matematik modeli ve kullanım alanlarına değinilmektedir. Bir sonraki bölümde problemle ilgili literatür taramasına yer verilmiştir. Üçüncü bölüm de çalışmada kullanılan metasezgisellerin teorisi anlatılmıştır. Dördüncü ve beşinci bölümler ise uygulama sonuçlarına ve bu sonuçların farklı açılardan karşılaştırmalarına ayrılmıştır.

2. Gezgin Satıcı Problemi

GSP, *n* adet şehir arasındaki mesafelerin bilindiği durumda, şehirlerin her birine yalnız bir kez uğramak şartıyla, başlangıç noktasına geri dönülmesi esasına dayalı, tur boyunca kat edilen toplam yolun en kısa olduğu şehir sıralamasının (optimal rota) bulunmasının amaçlandığı bir problemdir. Dağıtım, rotalama, kuruluş yeri belirleme, planlama, lojistik gibi problemlerde geniş bir uygulama alanına sahip olan gezgin satıcı problemi, aynı zamanda optimizasyon alanında, araştırmacılar tarafından üzerinde uzun yıllardır çalışmalar yapılan NP-hard (çözümü zor) sınıfında yer alan bir problemdir.

Eğer şehirler düğümlerle, yollar ise hatlar ile gösterilirse problem çizge üzerinde minimum maliyetli kapalı yolun bulunmasına karşılık gelmektedir [1].

Problemde düğüm sayısı arttıkça problemin çözümünde harcanan zaman üstel olarak artmaktadır. GSP'de artan düğüm sayısına paralel olarak çözüm zamanının üstel olarak artışı Tablo 1'de gösterilmektedir.

Tablo 1 Hamilton Döngülerinin Değerlendirilmesi [1]

Düğüm Sayısı	Döngü Sayısı (n-1)!	Gerekli Zaman
12	39.916.800	0.004 saniye
13	479.001.600	0.05 saniye
14	6.227.020.800	1 saniye
15	87.178.291.200	9 saniye
16	1.307.647.368.000	2 dakika
17	2.1 * 1013	35 dakika
18	3.6 * 1014	10 saat
19	6.4 * 1015	7.5 gün
20	1.2 * 1017	140 gün
21	2.4 * 1018	7.5 yıl
22	5.1 * 1019	160 yıl
23	1.1 * 1021	3.500 yıl
24	2.6 * 1022	82.000 yıl
25	6.2 * 1023	2 milyon yıl

Problemde başlangıç şehri verilmişse, mümkün olan Hamilton yolları sayısı geriye kalan (n-1) adet şehrin yer değişmesine, yani (n-1)!'e eşit olmaktadır. Bu durumda problem

basit olmasına rağmen, problemin çözümünü çözüm uzayının tamamını taramakla bulmak çok iyi bir yaklaşım değildir. Problemin en azından bir basit çözümünün olacağı kesindir. Bu sebeple GSP problemlerinin çözümünde sezgisel ve metasezgisel tekniklerin kullanılması etkin bir yoldur [2].

2.1. Matematiksel Model

Gezgin satıcı problemine ait tamsayılı doğrusal programlama modeli aşağıda gösterilmiştir [4]:

$$\text{Min} \quad \sum_{i=0}^{n} \sum_{j \neq i,\, j=0}^{n} c_{ij} x_{ij}$$

$$s.t. \quad 0 \leq x_{ij} \leq 1 \qquad i, j = 0,1,\ldots,n$$

$$x_{ij} \text{ tam sayı} \qquad i, j = 0,1,\ldots,n$$

$$\sum_{i=0, i \neq j}^{n} x_{ij} = 1 \qquad j = 0,1,\ldots,n$$

$$\sum_{j=0, j \neq i}^{n} x_{ij} = 1 \qquad i = 0,1,\ldots,n$$

$$u_i - u_j + n x_{ij} \leq n-1 \quad 1 \leq i \neq j \leq n$$

GSP'nin matematiksel modelinde kullanılan c_{ij}, *i.* şehirden *j.* şehire olan uzaklığı (mesafe) ifade etmektedir. x_{ij} ise *i.* şehirden *j.* şehire gidilmesi durumunda 1, diğer durumlarda 0 değerini alan bir parametredir. GSP modelinin amacı c_{ij}, x_{ij} çarpım toplamlarının minimize edilmesidir. Modelde *n,* şehir sayısını ifade ederken, *u* parametresi ise eksiksiz tam bir tur yapılması için koşulları sağlamak üzere modele eklenen yapay bir değişkendir.

2.2. GSP'nin Kullanım Alanları

GSP, optimizasyon teknikleri üzerine çalışma yapan araştırmacılar için ideal bir test ortamı oluşturmaktadır. Araştırmacılar geliştirdikleri yöntem(ler)in üstünlüklerini ispat etmek üzere literatürde sıklıkla kullanılan GSP'den faydalanmaktadır. GSP birçok gerçek hayat problemine uygulanmakla birlikte en çok yol ve rota planlama gibi konularda kullanılmaktadır [1].

GSP'nin kullanım alanları şu şekilde sıralanabilir [5, 6]:

- GSM operatörlerinin baz istasyonlarının yerleşim yerlerinin belirlenmesi,
- Birçok ulaşım ve lojistik uygulamaları,
- Malzeme akış sistem tasarımı,
- Araç rotalama problemleri,
- Depolardaki vinç güzergâhlarının programlaması,
- Stok alanındaki malzeme toplama problemleri,
- Uçaklar için havaalanı rotalaması,
- Elektronik devre tasarımı
- Röportaj zamanlama,
- Matbaa zamanlama,
- Bilgisayar kablolama,
- Ekip planlama,
- Misyon planlama,
- Navigasyon uydu sisteminin ölçme ağlarının tasarımı,
- Sipariş toplama.

3. Literatür

Literatürde klasik GSP'nin amaç fonksiyonunun değiştirilmesi ve/veya kısıtlarının farklılaştırılması ile elde edilen farklı çeşitleri bulunmaktadır.

Simetrik GSP, noktalar arası mesafelerin gidiş-dönüş için aynı olduğu durumlarda, *Asimetrik GSP* ise farklı olduğu durumlarda söz konusudur [7].

Kârlı GSP, bütün düğüm noktalarının ziyaret edilme zorunluluğunun bulunmadığı GSP'nin genelleştirilmiş halidir. Her noktaya ait bir kâr değeri vardır. Amaç, toplanan kâr ile yapılan yol masraflarının eşzamanlı eniyilenmesidir. Bu iki eniyileme kriteri ya amaç fonksiyonunda bulunmakta ya da kısıt olarak yazılmaktadır [8].

Zaman Pencereli GSP (ZPGSP), her şehrin önceden belirlenen zaman pencereleri içinde ziyaret edilmesi kısıtını ekleyerek oluşturulur [9].

*Belirsiz GSP'*de gerçek hayattaki pek çok belirsizliğin hesaplamalara dâhil edilmesi söz konusudur. Bu tip GSP problemlerine örnek olarak, trafik, hava durumu gibi etkenlerin, tur boyunca yollarda geçecek süre üzerinde belirsizlik oluşturması verilebilir. Araştırmacılar ağırlıklandırmalar yaparak ve belirsizlik teoremini kullanarak Belirsiz GSP üzerinde çözümler aramaktadırlar [10].

İki Depolu Heterojen GSP'de (2-HTSP), bir hedef seti üzerinde, iki farklı merkezden hareket eden ve birbirinden farklı olan araçlar tüm hedeflere uğrayarak araç başına minimum mesafeyle turlarını tamamlarlar [11][11].

Çoklu GSP'de (*ÇGSP)* , *n* adet şehir, her biri ayrı bir satıcıya atanmak üzere *m* adet tura bölünmektedir. ÇGSP, GSP'den daha zor bir problemdir, zira çözümü için öncelikle her bir satıcıya hangi şehirlerin atanacağının ve satıcıların turları üzerindeki şehirlerin optimal sıralamasının belirlenmesi gerekmektedir [12]

Dinamik GSP, zaman içerisindeki değişimleri dikkate alan ve sürekli yeni bir optimum bulmayı amaçlayan GSP çeşididir. Bu değişkenlik problemin çözümünü karmaşıklaştırmaktadır [12].

Genelleştirilmiş GSP'de, bir gezgin satıcı *s* salkımlı, *n* düğümlü bir ağda bir başlangıç noktasından başlayıp her salkımdan bir düğüme sadece bir defa uğrayıp başladığı yere dönmek durumundadır. Uğrayacağı yerlerin sıralarını belirlerken de kat edeceği toplam mesafenin veya yapacağı harcamanın en küçük olmasını amaçlamaktadır. Uçaklar için havaalanı rotalaması, elektronik devre tasarımı, posta kutusuna dağıtım problemleri, malzeme akış sistemleri tasarımı Genelleştirilmiş GSP'ye örnek olarak verilebilir [13].

Açık Döngülü GSP'de, gezgin satıcı her noktaya uğramakta ancak başladığı noktaya geri dönme zorunluluğu bulunmamaktadır. Her şehir gezgin satıcı tarafından ziyaret edilmiş olmaktadır [14].

GSP çözümünde ilk olarak klasik yöntemler uygulanmıştır. Bu yöntemler kesin ve sezgisel yöntemlerden oluşmaktadır. Lineer Programlama [15], dinamik programlama [16], dal-kesim yöntemi [17, 18] gibi kesin yöntemler küçük problemlerin çözümünde kullanılırken; 2opt [19], 3opt [20], Markov zinciri [21], tavlama benzetimi [22-24], tabu arama [25, 26] gibi sezgisel yöntemler büyük problemlerin çözümü için kullanılmaktadır. Bunun yanında daha etkin çözümler için agresif prensiplere dayalı en yakın komşu [27], minimum kapsayan ağaç [28] gibi yöntemler de uygulanabilmektedir. Çözüm uzayının çok büyük olduğu durumlarda klasik metotlar GSP çözümünde yetersiz kalmaktadır. Bu yetersizliklerin üstesinden gelmek için yeni yöntemlere ihtiyaç duyulmuştur.

Populasyon temelli optimizasyon algoritmaları, genellikle tabiattan ilham alınarak geliştirilmiş tekniklerdir [29]. Tabiatta yaşayan varlıklar, işleyen ve gelişen doğal sistemler; bilim, teknoloji vb. farklı alanlarda yapılan tasarımlar ve icatlar için ilginç ve

değerli bir ilham kaynağıdır. Genetik Algoritma [30-42], Karınca Kolonisi Algoritması [43-50], Arı Kolonisi Algoritması [51-54], Yapay Sinir Ağları [55-58], Yapay Bağışıklık Sistemi [59-62], Parçacık Sürü Optimizasyonu [63], Akıllı Su Damlaları [64], Elektromagnetizm Benzetimli Mekanik Algoritması [65] bu alanda yer alan GSP çözüm tekniklerindendir.

4. Çalışmada Kullanılan Metasezgisel Yöntemler

Bu çalışmada GSP'nin çözümünde kullanılan, tek noktadan arama yapan Tepe Tırmanma, İteratif Yerel Arama, Tavlama Benzetimi, Tabu Arama ve Kanguru Algoritmaları ile popülasyon temelli yöntemlerden Yapay Arı Kolonisi, Genetik Algoritma ve Karınca Kolonisi Algoritması bölüm alt başlıklarında açıklanmıştır.

4.1. Tepe Tırmanma Algoritması (TT)

TT algoritması, yerel arama sınıfında yer alan iteratif bir optimizasyon yöntemidir. Tanımlanan kurallar doğrultusunda bir çözümden diğer komşu çözüme ulaşma temeline dayanmaktadır [66].

TT algoritmasında iyi bir komşuluk yapısı seçilmesinin, metodun etkinliğinde önemi büyüktür. Algoritmanın zayıf yanı yerel ve global arasında ayrım yapamamasından kaynaklı yerel optimuma takılmasıdır [67].

TT algoritmasında kopya işlemi ile başlangıç rotası hafızaya alınmakta ve tweak işlemi ile kopyalanan rota üzerinde değişiklikler yapılarak yeni bir rota elde edilmektedir. Algoritma boyunca mevcut rota ile oluşturulan rotanın kaliteleri (uygunluk) karşılaştırıldığı için rotaları bozmamak adına kopyalama işlemi yapılmaktadır [67] .

4.2. İteratif Yerel Arama Algoritması (İYA)

İYA, matematik ve bilgisayar bilimlerinde uygulanan yerel arama, tepe tırmanma gibi metotların geliştirilmiş halidir.1980'lerde ortaya çıkmıştır. Yerel arama metotları bazen yerel minimum veya maksimuma takılabilir [69].

İYA'nın genel fikri; tüm çözüm uzayını aramaya odaklanmaktansa daha yerel optimumlar tarafından çevrili küçük bir alt uzaya odaklanmaktır. İYA ile bir optimizasyon problemiyle bir uygulamayı ele almak 4 temel birleşenle gerçekleşir.

Bunlar şu şekildedir;

1.Rassal bir başlangıç çözümü oluşturmak
2.Yerel arama
3.Perturb yapmak
4.Kabul kriterini uygulamak

Probleme rassal bir başlangıç çözümüyle başlanmaktadır. Yerel arama safhasında hafızaya kopyalanarak alınan başlangıç rotası üzerinde küçük bir değişiklik (tweak) yapılarak yeni bir çözüm elde edilmekte ve kalite olarak başlangıç çözüm ile karşılaştırılmaktadır. İzleyen aşamada mevcut rota üzerinde büyük değişiklikler yapan perturb işlemi yapılarak çözüm kalitesi artırılmaya çalışılmaktadır. Büyük değişiklikler ile ifade edilen işlem, lokal çözüm alanından sıçrama yaptırabilecek değişikliklerdir. Örneğin, GSP probleminde uzak kaydırma işleminin düğüm sayısına bağlı olarak bir defadan çok daha fazla yapılmasıdır. Son olarak kabul kriteri uygulanarak döngü tamamlanmaktadır. Bu bileşenler arası etkileşimler hesaplanmalı ve metodun kalitesi arttırılmaya çalışılmadır [70]

İYA; grafik çizimlerinde, Steiner minimum yayılan ağaç problemi çözümünde, GSP'de ve maksimum gerçeklenebilirlik problemi gibi problemlerin çözümünde kullanılmıştır.

4.3. Tavlama Benzetimi Algoritması (TB)

TB algoritması, ilk olarak 1983 yılında Kirkpatrick, Gelatt ve Vecchi [22] tarafından sunulmuş olup, optimizasyon problemlerinin çözümü için geliştirilmiş bir yerel arama algoritmasıdır [71, 72].

TB algoritması adını erimiş metalin soğutulması işlemi olan, tavlama işleminden almaktadır [68]. Bu işlemde metalik yapıdaki kusurları azaltmak için bir materyal ısıtılır, daha büyük kristal boyuta ve minimum enerji ile katı kristal duruma yavaşça soğutulur. Tavlama işlemi, sıcaklığın ve soğuma katsayısının dikkatlice kontrolünü gerektirir [71]. Tavlama işlemi sonucunda oluşan kristalleşme, metalin mekanik özelliklerini iyileştiren moleküler yapısındaki değişikliklerle oluşmaktadır [72]. Tavlama işlemindeki ısının davranışı, optimizasyondaki kontrol parametresiyle aynı gibi görülür. Isının, daha iyi sonuçlara doğru algoritmaya rehberlik eden bir rolü vardır. Bu durum ancak kontrollü bir tutum içinde, ısının kademeli olarak düşürülmesiyle yapılabilir. Eğer ısı aniden düşürülürse, algoritma lokal minimum ile durur [73]. TB algoritması; birçok değişkene sahip fonksiyonların maksimum veya minimum değerlerinin bulunması için, özellikle de birçok yerel minimuma sahip doğrusal olmayan fonksiyonların minimum değerlerinin bulunması için tasarlanmıştır [72].

Tavlama benzetimi algoritmasında; sıcaklık, sıcaklığın düşürülmesi, tekrar işlemi (döngü) gibi parametreler vardır. Algoritmada bir başlangıç aday çözümü belirlenip, bu çözüm başlangıçta en iyi çözüm olarak kabul edilir. Başlangıç aday çözümünün kopyasına "tweak" işlemi (başlangıç aday çözümüne ufak bir değişiklik yaparak yeni bir çözüm üretme işlemi) uygulanır. Daha sonra ki adımlar da ise en iyi çözüm kabul edilen başlangıç aday çözümü ile yeni üretilen çözümün hangisinin daha iyi olduğu (kaliteleri) veya 0-1 arasında üretilen bir rassal sayının $< e^{((\text{Kalite(Y)} - \text{Kalite(B)})/\text{sıcaklık})}$ durumu araştırılır [68]. Eğer yeni sonuç daha iyi ise en iyi çözüm olarak yeni çözüm atanır. Sıcaklık parametresi, tavlama yönteminde başlangıçta belirlenen bir değerdir. Bu değer, oluşturulan döngünün sonunda, belirli bir oranda (sıcaklığın düşürülme parametresi kadar) azaltılır. Bu işlemler en iyi çözüm bulunana kadar veya algoritmanın çalıştırılması için belirlenen bir süre bitene kadar veya sıcaklık parametresi sıfır veya sıfırdan küçük olana kadar sürdürülebilir.

4.4. Tabu Arama Algoritması (TA)

Glover [74] tarafından ortaya atılan ve Hansen [75] tarafından türetilmiş versiyonları bulunan TA algoritması, temelde son çözüme götüren adımın dairesel hareketler oluşturmasını engellemek için cezalandırılarak bir sonraki döngüde tekrar edilmesinin yasaklanması üzerine kurgulanmıştır [76].

TA algoritması döngü ya da çalışma süresi boyunca üretilen çözümler ile ilgili bilgileri saklamak üzere tasarlanmış dinamik bir hafızaya sahiptir. Tabu listesi olarak da adlandırılan bu hafızada saklanan bilgiler, araştırma uzayında yeni çözüm kümelerinin oluşturulması için kullanılır [77]. TA algoritması, mevcut çözümlerden küçük bir değişim ile (tweak) elde ettiği denenmemiş bir çözüm kümesi üreterek optimizasyona başlamaktadır. TA algoritmasında, çözüm uzayında bir yerel minimum noktada takılmayı engelleyebilmek için oluşturulan yeni çözüme, o anki çözümden daha kötü olsa bile müsaade edilmektedir. Ancak kötü çözüme izin verilmesi algoritmayı bir kısır döngü içerisine sokabilecektir. Algoritmanın kısır döngüye girmesine engel olmak için, bir tabu listesi oluşturulur ve o anki çözüme uygulanmasına izin verilmeyen tüm yasaklı hareketler tabu listesinde saklanır.

Bir hareketin tabu listesine alınıp alınmayacağını belirlemek için, tabu kısıtlamaları adı verilen bazı kriterler kullanılmaktadır. Tabu listesinin kullanılması, belirli bir iterasyon

sayısınca daha önce denenmiş çözümlerin tekrar edilmesini engellediği için arama esnasında bir bölgede takılma ihtimalini azaltmaktadır [78].

TA, mümkün bir çözüm ile başlar. Bu çözüm, problemin matematiksel ifadesinde geçen kısıtları tatmin eden bir çözümdür. Tabu aramanın performansı başlangıç çözümüne bağlıdır. Bu nedenle mümkün olduğunca iyi olan bir çözüm ile başlamak gerekir.

Tabu listesi ilk giren ilk çıkar (first in first out = FIFO) mantığında çalışan bir listedir. Algoritmada belirlenen karakteristik özelliklere göre tabu listesi sürekli olarak yukarıdan doldurulmaya başlar. Bulunan elemanların sayısı liste uzunluğunu (*l*) aştığında yeni gelen elemanın listeye eklenmesi için listenin en sonundaki eleman listeden çıkarılır [68].

4.5. Kanguru Algoritması (KA)

KA, ilk olarak Pollard [79] tarafından kesikli logaritmik optimizasyon problemlerinin çözümü için geliştirilmiş bir algoritmadır. Van Oorschot ve Wiener [80] yılında yapmış oldukları çalışmada Pollard tarafından ileri sürülen kanguru metodunun paralel halini geliştirmişlerdir.

Bir iteratif iyileştirme metodu olan KA; TA, TB gibi sezgisellerden ilham alınarak geliştirilmiştir. Komşu arama algoritması olarak da sınıflandırılan algoritmada, kesikli optimizasyon problemlerinin çözümünde belli bir noktadan çözüme başlamakta, yakın komşuluklar belli bir komşuluk fonksiyonuna göre aranmaktadır. Bu aşamaya $descent$ denmektedir.

Başlangıç çözümü u, komşu çözümler u' dür. Mevcut en iyi çözüm u^* dür ve aranan komşu çözümlerde u^* bulunursa bu çözüm u yerine geçmektedir. Belirli sayıda (A) iterasyonun tamamlanması halinde amaç fonksiyonu değerinde iyileşme gözlemlenmiyorsa algoritmanın ikinci adımına geçilir. İkinci adım $jump$ olarak adlandırılır. $jump$ prosedürü çözümün yerel optimuma takılmasına engel olmak için geliştirilmiştir. Bu aşamada çözüm uzayının farklı noktalarını taramak için çözüm seti tesadüfi olarak değiştirilmektedir. Belirlenen adım sayısı kadar gelişme olmazsa bulunan en iyi $jump$ noktasından yakın komşu arama prosedürüne, $descent$, geçilir. Her iki prosedürde de mevcut en iyi çözümden daha iyisi bulunduğunda sayaç (t) sıfırlanır ve t, A'dan büyük olana kadar arama adımlarına devam edilir [81].

A parametresi, mevcut çözümde herhangi bir gelişme olmaksızın gerçekleştirilebilecek maksimum iterasyon sayısıdır. descent prosedüründe, A kadar iterasyonda gelişme olmazsa $jump$ prosedürüne geçiş yapılır. Aynı şekilde $jump$ prosedüründe A kadar iterasyonda iyileşme olmazsa $descent$ prosedürüne dönülür. Bu süreç KA' ya ait bir döngü olup durdurma kriteri sağlanan kadar devam eder [82].

4.6. Yapay Arı Kolonisi Algoritması (YAK)

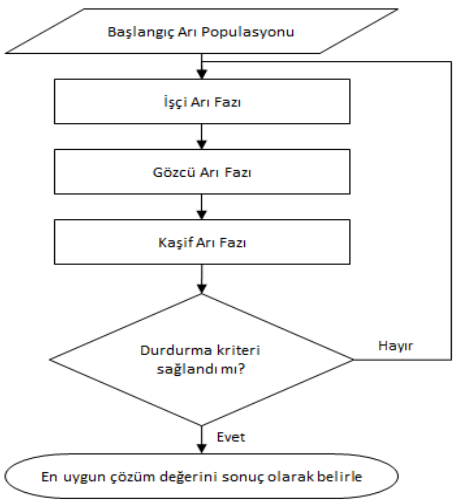
YAK algoritması, arılardaki yiyecek arama davranışları temel alınarak geliştirilen bir algoritmadır [83]. Doğada arılar yiyecek kaynaklarından nektar toplama, bulunan nektarı zamanı ve yolu minimize ederek en verimli şekilde kovana getirme işini içgüdüsel olarak yaparlar. Kaliteli bir yiyecek kaynağı bulan arılar, buldukları kaynaklardan toplayabildiği miktardaki nektarı kovana getirdikten sonra tekrar kaynağa dönmeden önce "dans alanında" (dancing area) "salınım dansı" (waggle dance) ile kendi kaynağı hakkındaki yön, uzaklık ve nektar miktarı bilgilerini diğer arılarla paylaşırlar. Bu başarılı mekanizma sayesinde koloni, kaliteli yiyecek kaynaklarının olduğu bölgelere yönlendirilebilmektedir. YAK sistemi, üç temel bileşenden oluşmaktadır. Bunlar, yiyecek kaynakları, arılar ve kovan.

Yiyecek kaynakları, arıların yiyecek bulmak için gittikleri kovan etrafındaki nektar kaynaklarıdır. Bir yiyecek kaynağının değeri, kaynağın çeşidi, yuvaya olan uzaklığı, nektar miktarı veya nektarın çıkarılmasının kolaylığı gibi birçok faktöre bağlıdır. Yiyecek kaynakları optimize edilmeye çalışılan problemin olası çözümlerine karşılık gelmektedir [54]. Bir kaynağa ait nektar miktarı o kaynakla ifade edilen çözümün kalite değerini ifade eder.

YAK Algoritmasında bir kolonide üç grup arı bulunmaktadır: işçi arılar, gözcü arılar ve kaşif arılar [54]. Kâşif Arılar, arı sistemi içerisinde tamamen bağımsız davranarak herhangi bir ön bilgi kullanmadan yeni yiyecek kaynaklarını arayan arı grubudur. Bir kovandaki kâşif arıların kovandaki tüm arılara oranı %5-10 civarındadır [83]. Rasgele yiyecek kaynağı ararlar. Buldukları yeni yiyecek kaynağı bir önceki yiyecek kaynağından daha iyi ise, yeni yiyecek kaynağını hafızalarında güncellerler. Kovana dönerler, dans alanında salınım dansı yaparak gözcü arılarla yeni yiyecek kaynağının bilgilerini paylaşırlar. İşçi Arılar, belirli bir yiyecek kaynağından nektar getiren arılardır. Mevcut durumda nektar kaynağından faydalanmaya ve çalışmaya devam ederler. Aynı zamanda nektar topladıkları yiyecek kaynaklarına giderken komşu yiyecek kaynaklarını (komşu çözümler) da araştırırlar. Belirlenen hata olasılığına [84] göre karşılaştıkları yeni daha iyi ya da daha kötü yiyecek kaynağını kabul eder veya reddederler. Eğer yeni çözümü kabul ederlerse ziyaret sayaçları sıfırlanır ve yeni yiyecek kaynağı hafızalarında güncellenir. Kovana döndüklerinde dans alanında salınım dansı yaparak gözcü arılarla yeni yiyecek kaynağının bilgilerini paylaşırlar. Aynı yiyecek kaynağına her gittiklerinde ziyaret sayaçları bir artırılır. Arının ziyaret sayısı, verilen maksimum ziyaret sayısını aştığında arı kaynağı terk eder ve gözcü arıya dönüşür. Gözcü Arılar, kâşif ve işçi arıların salınım dansını izleyerek yiyecek kaynakları hakkında bilgi edinen ve daha sonra bu bilgiye göre hareket ederek yiyecek kaynaklarına ulaşan arılardır. Daha iyi yiyecek kaynağı bulan kâşif veya işçi arılar dans alanında salınım dansı yaptığında gözcü arılar yeni yiyecek kaynağını belirlenen ikna olasılığına [84] göre kabul eder veya reddederler. Kabul ettikleri takdirde yeni çözüm hafızalarında güncellenir.

Kovan, arıların barındıkları, yiyeceklerini depoladıkları ve bilgi paylaştıkları yapılardır. Arılar arasındaki bilgi değişimi ortak bilginin sağlanmasındaki en önemli olaydır. Bu bilgi değişiminin gerçekleştiği dans alanı kovanın en önemli bölümüdür. Dans alanında yapılan salınım dansı ile yiyecek kaynaklarının yer ve kalite bilgileri paylaşılır[84].

YAK Algoritmasına ait akış diyagramı Şekil 1'de gösterilmiştir.



Şekil 1 Yapay Arı Kolonisi Algoritmasına Ait Adımlar

4.7. Genetik Algoritma (GA)

GA doğrusal olmayan, çok değişkenli, zor problemlerin çözümü için geliştirilmiş, popülasyon temelli sezgisel bir yöntemdir [85-87]. Önbilgi ve varsayımlar olmadan, amaç fonksiyonu ile çalışabilmektedir. Problem değişkenleri, kromozom denen dizilerde, genlerle temsil edilmektedir. Her bir değişken kodlama biçimine bağlı olarak tek ya da bir grup genle tanımlanmaktadır. Seçim, çaprazlama ve mutasyon olarak adlandırılan genetik operatörlerle iterasyonlar boyunca kromozomlarda birtakım değişiklikler yapılmakta ve en iyi sonucu verecek çözüm seti aranmaktadır.

GA'da ilk olarak kodlama biçimine karar verilir. Genellikle ikili kodlama, permutasyon kodlama ve gerçek değerli kodlama kullanılmaktadır. İkili kodlama 1 ve 0 değerlerinden oluşup değişkenler değer aralığına göre belirlenen sayıda genden oluşan ikili düzende temsil edilmektedir. Permutasyon kodlama, sıralamanın önemli olduğu ve tekrarın mümkün olmadığı, en kısa yol, gezgin satıcı, araç rotalama vb. problemlerin çözümünde kullanılır. Gerçek değerli kodlama ise değişkenlerin doğrudan kendi değerleriyle temsil edildikleri kodlama biçimidir.

Popülasyon temelli olan GA'da çözüm uzayındaki arama tek bir noktadan değil, noktalar kümesinden yapılmaktadır. Uygulayıcı tarafından belirlenen miktardaki kromozom, popülasyonu oluşturmaktadır. Çözüme tesadüfi olarak veya basit tekniklerle oluşturulabilen başlangıç populasyonu ile başlanır. Genlerdeki değişken değerleri, fonksiyonda yerine konularak kromozomun uygunluk değeri elde edilir. Genetik operatörlerden ilk olarak seçim operatörü uygulanır. Amaç, popülasyonda daha iyi bireylerin çoğaltılması, uygunluğu düşük olan bireylerin elenmesidir. Birçok seçim yöntemi vardır. Bunlara, rulet tekerleği seçimi, turnuva seçimi, genel stokastik örnekleme ve sıralı seçim örnek olarak verilebilir [88]. Seçim sonrası çaprazlama operatörü uygulanmaktadır. Çaprazlamada amaç, iki bireyin farklı birtakım özelliklerini taşıyan ve daha iyi bireyler elde etmektir. Böylelikle, uygunluğu daha yüksek çözüm alternatifleri üretilmeye çalışılır. İkili kodlamada genellikle tek nokta [89], iki nokta ve çok noktalı çaprazlama, permutasyon kodlamada pozisyona dayalı, sıralı [85] ve dairesel çaprazlama [90] kullanılmaktadır. Gerçek değerli kodlamaya ise aritmetik çaprazlama, kesikli üretim, çizgi üretim örnek olarak verilebilir. Mutasyon operatörü, bir daha ulaşılması mümkün olmayan çözümlerin kaybına karşı koruma sağlamaktadır [85]. Düşük bir olasılıkla herhangi bir gen üzerinde yapılan tesadüfi değişikliklerdir. İkili düzende, genin değeri 1 ise 0'a, 0 ise 1'e dönüştürülmesi şeklinde gerçekleştirilmektedir. Permutasyon kodlamada yakın kaydırma, uzak kaydırma, toplu kaydırma, tesadüfi değişim, sıralı değişim gibi birçok mutasyon çeşidi vardır. Gerçek değerli kodlamada ise mevcut değişken değerinin belirlenen mutasyon adımı miktarınca azaltılması veya eşit olasılıkla arttırılması şeklinde mutasyon uygulanmaktadır.

Genetik operatörler başlangıç popülasyonuna uygulanır ve yeni bir jenerasyon elde edilir. Tamamlanma kriteri sağlanana kadar genetik operatörlerle yeni popülasyonlar üretilir. Döngü tamamlandığında algoritma durdurulmakta ve mevcut en iyi çözüm sonuç olarak belirlenmektedir [91].

4.8. Karınca Kolonisi Algoritması (KKA)

KKA, karıncaların yön seçme duyularından ve besin kaynaklarına ulaşma mantıklarından esinlenerek geliştirilmiş bir metasezgisel yöntemdir. Gerçek karıncaların yuvaları ile yiyecek topladıkları noktalar arasındaki mesafeyi enküçükleyen rotayı, salgıladıkları feromon kimyasalı sayesinde belirlemeleri üzerine kurgulanmıştır [92].

Gerçek karıncalar yiyecek aramak üzere yuvalarından ayrıldıklarında izleyecekleri rotayı yoldan daha önce geçen karıncaların salgıladıkları feromon sayesinde belirlemektedir.

Feromon karıncaların bacaklarından salgılanan bir kimyasal olup, belirli bir süre boyunca yol üzerinde kalmakta ve zamanla buharlaşarak yok olmaktadır.

Başlangıçta rassal olarak yiyecek aramaya çıkan karıncalardan en kısa mesafeli yolu kullanan karıncalar daha kısa sürede yuvaya dönüşe geçerek, yol üzerine daha fazla feromon bırakmış olmaktadır. Yiyecek arama işlemi devam ettikçe, kısa mesafeler üzerinde feromon miktarı yoğunlaşmakta, sürenin de kısalığına paralel olarak feromon buharlaşma oranı da azalmaktadır. Aynı şekilde daha uzun mesafeli rotalar üzerinde başlangıçta daha az olan feromon nedeniyle karıncalar tarafından tercih edilme oranı azalmakta ve bir süre sonra yol üzerinde bulunan feromon tamamen buharlaşarak hiçbir karıncanın kullanmayacağı bir yol olmaktadır [93].

Gerçek karıncaların yiyecek arama davranışlarından esinlenerek geliştirilen KKA ilk olarak 1992 yılında yayımlanan bir doktora tezinde [94] GSP üzerinde uygulanarak literatüre geçmiştir. Yapay karıncalardan oluşan KKA'da başlangıç çözümü olarak karıncalar rassal yollar üzerinden turlarına başlamakta ve bir turu tamamlayarak yuvaya dönmektedirler. Tur boyunca karıncaların takip ettikleri yollar üzerine feromon eklemesi yapılmaktadır.

İlk iterasyonun tamamlanmasının ardından ikinci iterasyonda daha kısa mesafelerde nispeten daha fazla feromon bulunacağı için yapay karıncaların bir kısmı izledikleri rota yerine daha çok feromon bulunan yeni rotayı takip etmeye başlayacaktır. Her iterasyon sonunda yollar üzerinde bulunan feromon miktarları kullanıcının belirlediği bir oranda buharlaştırılarak local güncelleme yapılmaktadır. Bu sayede çok kötü çözümlere izin verilmediği gibi, feromon miktarının nispeten fazla olduğu rotalar da en iyi çözüm olarak kabul edilmemektedir. Bu durum algoritmanın yerel optimumlara takılmasına mani olmaktadır.

Global feromon güncellemesi aşamasında ise en iyi çözümü üreten rota ek feromon eklemesi yapılmak suretiyle yapay karıncalar için daha cazip kılınmaktadır [95].

5. Uygulama

Çalışmada, kullanılan metasezgisel yöntemler; GSP literatüründe yer alan ve kıyaslama işlemlerinde sıklıkla kullanılan *burma14, ulysses16, gr17, gr21, ulysses22, gr24, bays19, dantzig42, swiss42, gr48, hk48, eil51, berlin52, brazil58, st70, eil76, pr76* problemlerine uygulanmıştır. Problemlere ait veriler TSPLIB [96] sitesinden alınmıştır.

Kullanılan metasezgisellere ait algoritmalar C#, Java ve MATLAB programlama dillerinde kodlanmıştır. Belirlenen algoritmalar her bir problem için 20 defa çalıştırılmış ve sonuçları; bulunan en iyi değer, minimum (Min), bulunan en kötü değer, maksimum (Maks), 20 değerin ortalaması olarak Ortalama ve optimumu bulma sayısı (OBS) başlıkları altında değerlendirilmiştir. Kullanılan metasezgisellerin GSP problemine uygulanması ve parametre değerleri aşağıda başlıklar halinde verilmiştir.

5.1. Tepe Tırmanma Algoritması

TT algoritmasında oluşturulan rassal başlangıç rotası tweak işlemi ile edilen yeni rassal rota ile karşılaştırılmakta, yeni rota ile daha kısa tur elde edilmesi durumunda başlangıç rotası güncellenmektedir. Yeni rota oluşturulması işlemi ayrı bir döngüde belirli sayıda rassal rotalar oluşturularak en iyi rotanın yeni rota kabul edilmesi prensibi ile gerçekleştirilmiştir. Tweak işlemi için güncel rota üzerinde belirlenen rassal iki nokta arası noktaların yer değiştirilmesini esas alan ters çevirim (inversion) hareket operatörü kullanılmıştır. Ters çevirim operatörünün GSP problemlerinde tek bir adım ile optimale ulaşma olasılığı yüksektir [97]. Çalışmada kullanılan 18 GSP problemine ait çözüm değerleri, 3000 iterasyondan oluşan 20 çalıştırma ile elde edilmiştir.

5.2. İteratif Yerel Arama Algoritması

Problemlerin çözümünde kullanılan perturb işlemi ile algoritmanın yerel optimum değerlere yüksek oranda takılması engellenmiştir. Bu durum, yöntem optimum sonucu bulamasa dahi düşük bir standart sapma gösterecek şekilde belirli bir ortalama değer etrafında toplanmasına neden olmaktadır. Yerel optimumlar üzerinden yapılan perturb hareketi ile algoritmanın kötü değerlere takılması engellenmektedir. İYA algoritmasında T (time) parametresi kullanılmış, parametre değeri 50 saniye, iterasyon sayısı ise 2000 olarak belirlenmiştir.

5.3. Tavlama Benzetimi Algoritması

Problemler çözülürken sıcaklık, sıcaklığın düşürülme oranı, döngü sayısı parametreleri için problemin çeşidine göre farklı değerler kullanılmıştır. Probleme özgü farklı parametrelerin kullanılmasıyla algoritmanın optimal veya optimale yakın çözüm bulması sağlanmıştır. Optimal veya optimale yakın sonuçların bulunmasında kullanılan parametreler, denemeler yapılarak belirlenmiştir. TB kullanılarak çözülen problemler ve parametreler Tablo 2'de gösterilmiştir.

Tablo 2 TB kullanılarak çözülen problemler ve parametreleri

Problem Adı	Sıcaklık (t)	Döngü Sayısı	Sıcaklığı Azaltma Oranı	Problem Adı	Sıcaklık (t)	Döngü Sayısı	Sıcaklığı Azaltma Oranı
burma14	26,0	10.000	0,9999	swiss42	20,0	50.000	0,9999
ulysses16	26,0	10.000	0,9999	gr48	31,0	50.000	0,9999
gr17	23,0	8.000	0,9999	hk48	20,0	50.000	0,999
gr21	20,0	10.000	0,9999	eil51	19,0	50.000	0,999
ulysses22	26,0	20.000	0,9999	berlin52	38,0	70.000	0,9999
gr24	20,0	20.000	0,9999	brazil58	24,0	80.000	0,9999
fri26	11,0	30.000	0,9999	st70	26,0	90.000	0,999
bays29	29,3	50.000	0,9999	eil76	26,0	100.000	0,999
dantzig42	9,0	50.000	0,9999	pr76	47,0	100.000	0,999

Tablo 2'de görüldüğü üzere, problemlerin çözümünde kullanılan parametrelerin değerleri problemlere göre değişiklik gösterebilmektedir. Sıcaklık ve sıcaklığın azaltılma oranı parametrelerinin değerleri denemeler sonucu elde edilen en iyi sonuca göre belirlenmiştir. Döngü sayıları ise problemlerin boyutları arttıkça, genelde arttırılmıştır.

5.4. Tabu Arama Algoritması

Çalışmada hareket kısıtlamalı TA algoritması kullanılmıştır. Tabu listesi hareket kısıtlama üzerine kurgulanmıştır. Bu bağlamda düğüm sayısı *n* olmak üzere, hareket kısıtlamayı kontrol edecek *nxn* boyutlu kare matris (tabu matrisi) oluşturulmuştur. Tweak işlemi için güncel rota üzerinde belirlenen rassal iki nokta arası noktaların yer değiştirilmesini esas alan ters çevirim (inversion) hareket operatörü kullanılmıştır. Ters çevirim hareket operatörü çalışma koşulu tabu listesinin kontrolüne bağlıdır. Oluşturulan yeni rota üzerindeki düğümler tabu matrisinin satır indisleri, sıralamaları ise sütun indisleri kabul edilerek tabu matrisinin ilgili hücrelerine bir ceza (maliyet) ile eklenmiştir. Her iterasyonun tweak işlemi esnasında tabu matrisi kontrol edilerek, belirlenen rassal noktaların tabu matrisinde cezalandırılmış bir hücreye denk gelip gelmediği kontrol edilmekte, tweak işleminin yapılması için belirlenen noktaların cezalandırılmamış hücrelere denk gelmiş olması koşulu aranmaktadır. Her iterasyon sonunda tabu matrisinde cezalandırılmış hücrelerdeki ceza bir azaltılmakta, aynı şekilde yeni rota tabu

matrisine işlenmektedir. Bu işlem ile kötü sonuçlardan ve denenmiş rotalardan belirli bir döngü boyunca kaçınılmaktadır. Bununla beraber algoritmanın kötü sonuçları ve denenmiş rotaları tweak ile değiştirmek suretiyle optimuma ulaşma olasılığı bulunduğu için, algoritmanın tabu matrisinin dinamik yapısı gereği, kötü sonuç ve denenmiş rotalar tüm döngüler boyunca engellenmemektedir.

Çalışmada kullanılan hareket kısıtlamalı TA algoritmasında problemlere göre değişkenlik arz eden tek parametre, tabu matrisinde hücrelere atanan ceza parametresi olmuştur. Ceza parametresi düğüm sayısına bağlı bir parametre olarak tanımlanmış olup, düğüm sayısı *n* ile gösterilmek üzere, *n*/1,00 - *n*/2,00 aralığında 2 basamak duyarlı ondalık sayılar ile problem boyutuna göre sezgisel olarak belirlenmiştir. İterasyon sayısı ise 3000'dir.

5.5. Kanguru Algoritması

KA parametreleri, yöntemin gezgin satıcı problemine ilk defa uygulandığı Erdem ve Keskintürk'ün [82] çalışmasından alınmıştır. Çalışmada iniş (descent) fonksiyonu için uzak kaydırma; zıplama (jump) için ise çoklu uzak kaydırma operatörü kullanılmıştır. İterasyon sayısı 3000 olarak belirlenmiştir.

5.6. Yapay Arı Kolonisi Algoritması

YAK algoritmasında, iterasyon sayısı, arı sayısı (toplam), kâşif arı oranı (kâşif arıların toplam arılara oranı), işçi arı oranı (işçi arıların toplam arılara oranı), gözcü arı oranı (gözcü arıların toplam arılara oranı), maksimum ziyaret sayısı (işçi arının yiyecek kaynağına geliştirme olmaksızın yapacağı en fazla ziyaret sayısı), ikna olasılığı (gözcü arıların, işçi arı veya kâşif arının salınım dansından etkilenme olasılığı) ve hata olasılığı (işçi arının daha iyi bir çözümü kabul etmeme ya da daha kötü bir çözümü kabul etme olasılığı) parametreleri kullanılmıştır.

Parametreler deneme yoluyla belirlenmiş ve problemlere uygulanmıştır. Uygulanan parametreler Tablo 3'te gösterilmiştir.

Tablo 3 YAK algoritması GSP Problemlerine Göre Parametre Değerleri

Problemler	İterasyon sayısı	Arı sayısı	Kâşif arı oranı	İşçi arı oranı	Gözcü arı oranı	Maks ziyaret sayısı	İkna olasılığı	Hata olasılığı
burma14 ulysses16	30.000	45	0.15	0.70	0.15	15	0.90	0.01
gr17	30.000	50	0.15	0.70	0.15	15	0.90	0.01
gr21 ulysses22 gr24	40.000	65	0.15	0.70	0.15	20	0.90	0.01
fri26	50.000	75	0.15	0.70	0.15	25	0.90	0.01
bays29	60.000	90	0.15	0.70	0.15	30	0.90	0.01
dantzig42 swiss42	85.000	120	0.10	0.70	0.20	40	0.90	0.01
gr48 eil51 hk48 berlin52	100.000	150	0.10	0.70	0.20	50	0.90	0.01
brazil58	120.000	180	0.10	0.70	0.20	60	0.90	0.01

Problemler	İterasyon sayısı	Arı sayısı	Kâşif arı oranı	İşçi arı oranı	Gözcü arı oranı	Maks ziyaret sayısı	İkna olasılığı	Hata olasılığı
st70	140.000	210	0.10	0.70	0.20	70	0.90	0.01
eil76 pr76	150.000	225	0.10	0.70	0.20	75	0.90	0.01

Tablo 3'de de görüldüğü üzere problemin düğüm sayısı arttığında çözüm uzayı büyüyeceği için iterasyon sayısı ve arı sayısı parametre değerleri de artırılmıştır. Maksimum ziyaret sayısı düğüm sayısına yakın tutulmuştur. Kaşif arıların ortalama çözüm kaliteleri düşük ve arama süreleri uzun olduğundan düğüm sayısı 30'dan büyük olan problemlerde kaşif arı oranı düşürülmüştür.

5.7. Genetik Algoritma

Çalışmada permütasyon kodlamalı GA kullanılmıştır. Satır vektörlerinden oluşan kromozomların her biri problemin düğüm sayısına göre değişen uzunlukta belirlenmiştir. Popülasyon büyüklüğü 20 olarak belirlenmiş olup her bir popülasyon 20 satırdan ve düğüm sayısı kadar sütundan oluşan bir matristen oluşmaktadır.

Rulet tekerleği, seçim operatöründe kullanılan seçim yöntemidir. Çaprazlama için permütasyon kodlamaya uygun olarak iki nokta, pozisyona dayalı ve kısmi planlı çaprazlama eşit olasılıklarla kullanılmıştır [98]. Bir çaprazlama oranı belirlenmemiştir. Geliştirilen çaprazlama operatöründe mevcut birey sayısı kadar yeni birey üretilmekte ve bu iki grup birey arasından uygunluğu en iyi popülasyon büyüklüğü kadar birey yeni jenerasyona aktarılmaktadır. Mutasyon operatörü olarak ters çevirim kullanılmıştır [97]. Mutasyon oranı %0.5 olarak belirlenmiştir. İterasyon sayısı ise optimumu bulduğunda durdurulma kriterini içermekle birlikte 1-50 düğüm problemleri için 1000, daha büyük problemler için 2000 iterasyondur. Her iterasyonda algoritmanın performansını arttırmaya yönelik ters çevirim yerel arama yöntemi kullanılmıştır.

5.8. Karınca Kolonisi Algoritması

Test problemlerinin çözümünde Karınca Sistemi Algoritmasının operatörleri kullanılmıştır. Feromon güncellemesinde, tüm turlar ve mevcut n iterasyonun en iyisine ilaveten mevcut en iyi çözümden daha iyi bir çözüm bulunduğunda uygulanan bir güncelleme eklenmiştir. α ve β değerleri tüm problemler için 1 olarak kabul edilmiştir. Karınca sayısı 10 ve iterasyon sayısı 1000 olarak belirlenmiştir. Feromon güncelleme oranı %35'tir. Problem boyutuna bağlı olarak değişen bir parametre söz konusu değildir. Ayrıca algoritmanın performansını arttırmaya yönelik her iterasyonda en iyi değere sahip karınca turuna ters çevirim lokal araması 20 defa uygulanmıştır.

6. Bulgular

Çalışmanın bu bölümünde belirlenen 14-76 düğümlü benchmark problemlerinin 8 algoritma ile çözümünden elde edilen bulgular yorumlanmıştır. Her bir teknik, her bir problem için 20 kez çalıştırılmış ve sonuçları kayıt edilmiştir. 20 çalıştırmadan elde edilen sonuçların; ortalama değerleri (Ortalama) ve optimumu bulma sayıları (OBS) Tablo 4'te, en küçük değerler (Min) ve en büyük değerler (Maks) ise Tablo 5'te verilmiştir. Ayrıca tablolarda ele alınan problemlerin bilinen optimum sonuçları ile bulunan sonuçları karşılaştırılmıştır.

Tablo 4 Metasezgisel Tekniklerle Elde Edilen Ortalama Sonuçlar ve Optimumu Bulma Sayıları

GSP Problemleri		Tepe Tırmanma		İteratif Yerel Arama		Tavlama Benzetimi		Tabu Arama		Kanguru		Yapay Arı Kolonisi		Genetik Algoritma		Karınca Kolonisi	
Ad	Optimum	Ortalama	OBS	Ortalama	OBS	Ortalama	OBS	Ortalama	OBS	Ortalama	OBS	Ortalama	OBS	Ortalama	OBS	Ortalama	OBS
burma14	**3323**	3519,0	3	3338,0	0	3335,5	18	3384,0	16	3323,0	20	3323,0	20	3323,0	20	3323,0	20
ulysses16	**6859**	7310,0	0	7137,0	0	6924,3	4	6948,0	8	6883,4	5	6859,0	20	6860,9	17	6859,0	20
gr17	**2085**	2354,0	0	2185,0	0	2114,7	7	2120,0	2	2094,8	7	2085,0	20	2085,0	20	2085,0	20
gr21	**2707**	3767,0	0	3227,2	0	3027,9	4	2831,0	15	2849,6	12	2707,0	20	2726,3	18	2707,0	20
ulysses22	**7013**	8497,0	0	7414,0	0	7580,0	2	7195,0	12	7093,7	8	7054,0	10	7036,1	15	7013,0	20
gr24	**1272**	1786,0	0	1344,1	0	1335,5	1	1341,0	7	1342,7	1	1281,0	17	1291,2	6	1272,0	20
fri26	**937**	1395,0	0	1150,4	0	969,7	3	988,0	10	1004,9	2	948,0	8	955,0	7	937,0	20
bays29	**2020**	2187,0	0	2207,3	0	2062,6	1	2100,0	2	2182,6	0	2034,0	5	2034,5	3	2020,0	20
dantzig42	**699**	1459,0	0	720,1	2	796,0	0	748,0	1	811,2	0	744,0	1	727,6	2	699,0	20
swiss42	**1273**	2444,0	0	1393,5	0	1483,9	0	1378,0	1	1456,6	0	1370,0	0	1342,2	2	1297,8	11
gr48	**5046**	10388,0	0	5454,5	0	5895,6	0	5364,0	0	5699,2	0	5595,0	1	5183,3	0	5172,8	0
hk48	**11461**	23823,0	0	13159,9	0	15399,9	0	12287,0	1	12621,5	0	12465,0	0	12011,5	0	11712,0	1
eil51	**426**	596,0	0	473,6	0	480,6	0	457,0	0	470,4	0	459,0	0	443,7	0	443,7	0
berlin52	**7542**	10978,0	0	8404,4	0	9100,6	0	8338,0	1	8841,5	0	8594,0	0	8146,6	0	7819,1	4
brazil58	**25395**	58144,0	0	29360,9	0	32426,6	0	26666,0	1	29970,6	0	28951,0	0	26405,2	1	26263,7	0
st70	**675**	1880,0	0	843,2	0	863,2	0	849,0	0	829,4	0	830,0	0	731,3	0	715,7	0
eil76	**538**	1367,0	0	668,7	0	649,6	0	604,0	0	611,6	0	622,0	0	573,5	0	573,5	0
pr76	**108159**	308382,0	0	136655,5	0	165690,0	0	119510,0	0	130888,0	0	130729,0	0	116416,1	0	114392,2	0

Tablo 4 incelendiğinde GA, KKA, YAK algoritması gibi popülasyon temelli algoritmalar, problemlerin optimum sonuçlarına en çok ulaşabilen yöntemler olurken, TT ve IYA başta olmak üzere tek noktadan arama yapan yöntemlerinin optimum çözüme nadir olarak ulaşabildiği görülmektedir. Bunun yanında; st70, eil76, pr76 gibi bazı problemlerde hiçbir yöntem belirlenen iterasyon sayısında optimum sonucu bulamamıştır. burma14 probleminde IYA hariç tüm yöntemler optimum sonuca en az 3 kere ulaşmıştır. TT yöntemi 3 defa optimumu bulurken diğer yöntemlerden TB, 18; TA, 16; KA, YAK, GA ve KKA 20 kez olmak üzere yöntemlerin çoğu 20 çalıştırmanın her birinde veya çoğunda optimum değere ulaşmışlardır.

Her bir yöntemin her bir problem için ortalama sonuçlar ile problemin optimum sonucu karşılaştırılarak algoritma tarafından üretilen sonuçların optimum değere ne kadar yakın veya uzak olduğu görülebilir. Örneğin, Ullyses16 probleminde optimum değer 6859 olup, 20 sonucun ortalamaları; TT 7310,0; IYA: 7137,0; TB: 6924,3; TA: 6948,0; KA: 6883,4; YAK: 6859,0; GA, 6860,9; KKA, 6859,0 olduğu görülmektedir. Ullyses16 problemi için optimum sonuç ile algoritmaların ortalama sonuçları karşılaştırıldığında optimumu bulma sayısı fazla olan yöntemlerin optimuma daha yakın ortalamalar bulduğu görülmektedir.

Genel olarak popülasyon temelli metasezgisellerin, tek noktadan arama yapan algoritmalara göre daha iyi sonuçlar verdiği söylenebilir. Tek nokta arama yapan metasezgisellerden TB, TA ve KA ise diğerlerine göre popülasyon temelli algoritmalara daha yakın sonuçlar vermiştir. Bu, hem ortalama değerler hem de optimumu bulma sayıları açısından söylenebilir.

TT ve İYA basit mantıkla çalışan metasezgiseller olup, daha çok melez algoritmalarda lokal arama için kullanılmaktadır. Populasyon temelli algoritmalara baktığımızda ise belli bir problem büyüklüğüne kadar oldukça iyi sonuçlar verdiği görülmüştür.

Popülasyon temelli algoritmalar birden çok noktadan arama yapmaktadırlar. YAK'ta arılar, GA'da kromozomlar ve KKA'da karıncalar çözüm uzayını paralel olarak taramakta olup aynı zamanda birbirleriyle bilgi alışverişi gerçekleştirmektedirler. Optimum bulma sayılarına bakıldığında sırasıyla KKA, YAK ve GA'nın başarılı olduğu söylenebilir. Ortalamalarda da aynı şekilde belli bir problem büyüklüğüne göre optimuma oldukça yakın ya da eşit, diğer problemlerde de kabul edilebilir sınırlarda sonuçlar vermişlerdir.

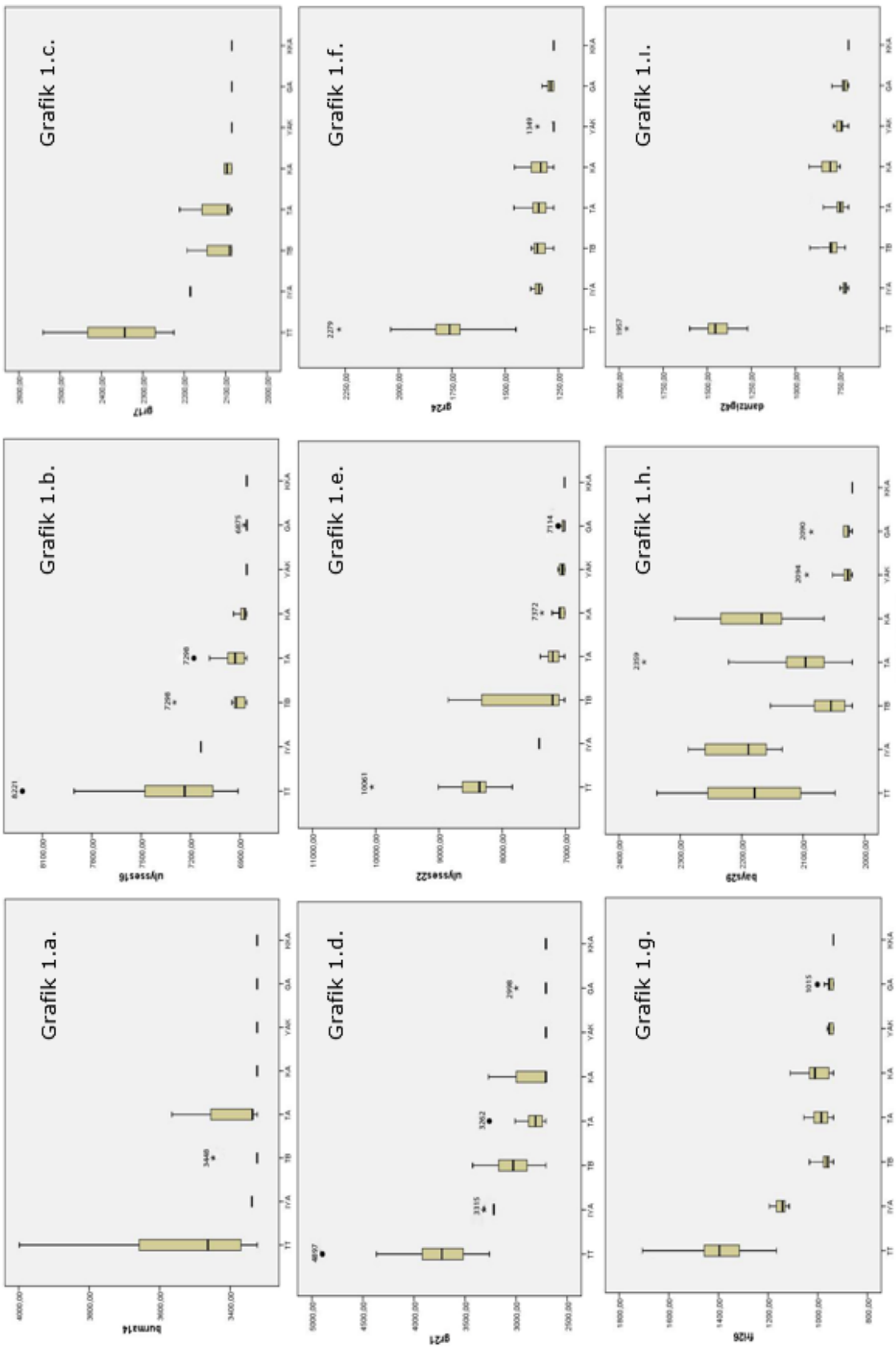
Amaç, belli bir probleme en iyi sonucu bulmak değil, belli iterasyon sayılarında metasezgisellerin GSP'deki performanslarını karşılaştırmak olduğundan, optimum sonuca ulaşılamayan problemler için farklı iterasyon sayıları ve performansı arttırıcı eklemeler yapılmamıştır.

Tablo 5'te problemlerin çözümünden elde edilen minimum ve maksimum değerler gösterilmiştir. Tablo ile hedeflenen farklı metasezgiseller farklı boyutlardaki GSP'lere uygulandığında bulunan minimum değerlerin ve optimum değerden en uzak değerlerin gösterilmesidir. Aşağıda Grafik 1 başlığı altında toplanan kutu grafikler ise her bir problem için yöntemlerin 20 kez çalıştırılmasıyla elde edilen sonuçların dağılımını göstermektedir. Grafiklerde sonuçların ortalama etrafında yığılmaları kutucuklarla, ortalamalardan hareketle hesaplanan Maks ve Min değerleri ise alt ve üst sınırlarla gösterilmiştir. Ayrıca bu sınırların dışında kalan ve ortalamalardan büyük sapmalar gösteren uç değerler (outliers) ise işaretçilerle gösterilmiştir.

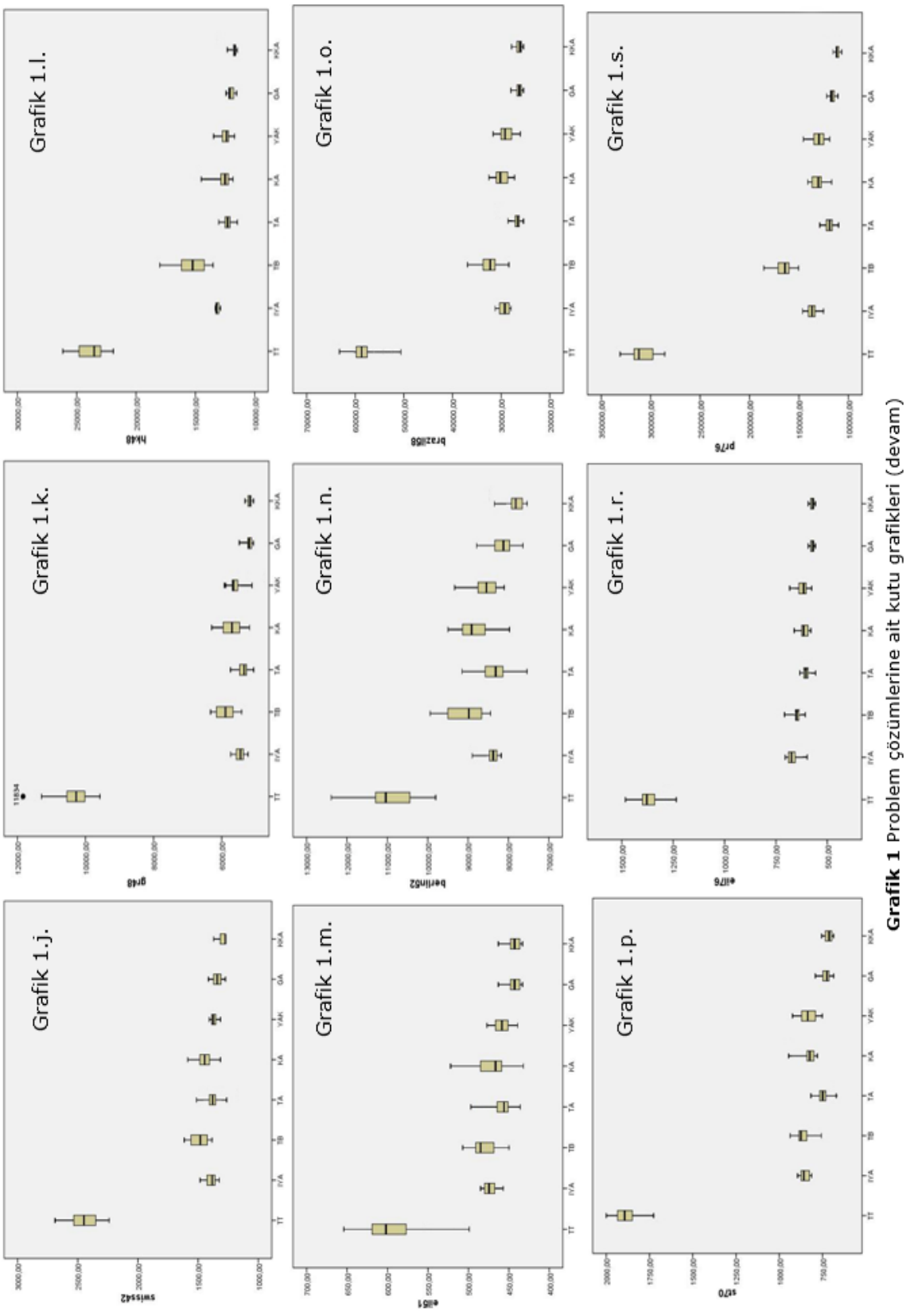
Grafik 1 grubuna göre; TT algoritması tüm problemlerde çok geniş aralıkta sonuçlar üretmiştir. KK, YAK, GA gibi populasyon temelli algoritmalarda çözüm aralıkları daha az sapma gösterecek şekilde dar bir seyir izlemektedir. Daha dar aralıkta çözüm üreten bu algoritmaların optimum sonuçlara daha çok ulaştığı görülmektedir.

Tablo 5 Metasezgisel Tekniklerle Elde Edilen Minumum ve Maksimum Değerler

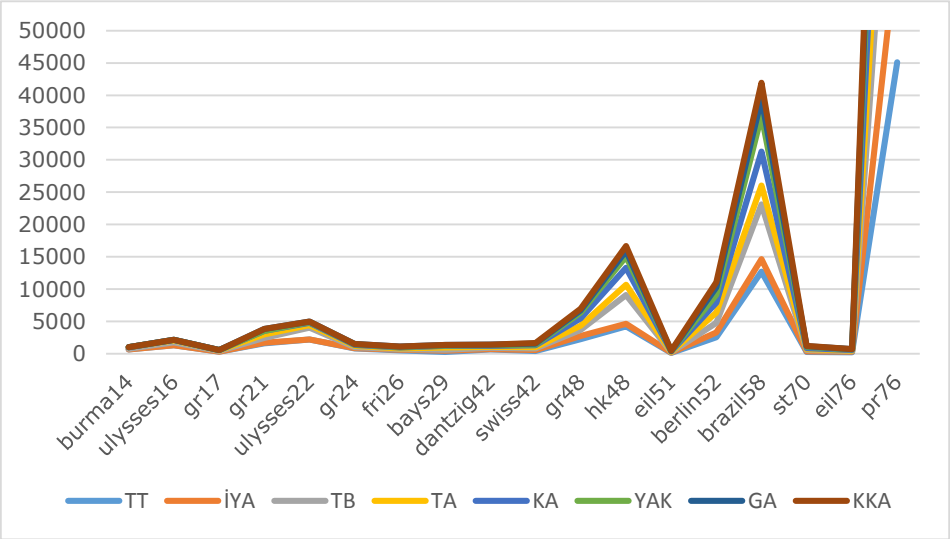
GSP Problemleri		Tepe Tırmanma		İteratif Yerel Arama		Tavlama Benzetimi		Tabu Arama		Kanguru		Yapay Arı Kolonisi		Genetik Algoritma		Karınca Kolonisi	
Ad	Optimum	Min	Maks	Min	Maks	Min	Maks	Min	Maks	Min	Maks	Min	Maks	Min	Maks	Min	Maks
burma14	**3323**	3323	3998	3338	3338	3323	3448	3323	3565	3323	3323	3323	3323	3323	3323	3323	3323
ulysses16	**6859**	6913	8221	7137	7137	6859	7298	6859	7180	6859	6939	6859	6859	6859	6875	6859	6859
gr17	**2085**	2225	2541	2185	2185	2085	2193	2085	2211	2085	2103	2085	2085	2085	2085	2085	2085
gr21	**2707**	3263	4897	3217	3315	2707	3425	2707	3262	2707	3270	2707	2707	2707	2998	2707	2707
ulysses22	**7013**	7842	10061	7414	7414	7013	8853	7013	7397	7013	7372	7013	7114	7013	7114	7013	7013
gr24	**1272**	1439	2279	1325	1380	1272	1377	1272	1474	1272	1456	1272	1349	1272	1328	1272	1272
fri26	**937**	1166	1703	1115	1195	937	1038	937	1056	937	1111	937	962	937	1015	937	937
bays29	**2020**	2048	2338	2134	2287	2020	2153	2020	2359	2066	2309	2020	2094	2020	2090	2020	2020
dantzig42	**699**	1271	1957	699	747	719	914	699	834	747	922	699	785	699	790	699	699
swiss42	**1273**	2240	2689	1324	1484	1385	1615	1273	1513	1315	1586	1315	1408	1273	1414	1273	1371
gr48	**5046**	9577	11834	5218	5729	5414	6322	5055	5734	5177	6289	5046	5946	5061	5401	5059	5313
hk48	**11461**	21915	26174	12880	13332	13506	17995	11461	13029	11824	14418	11706	13462	11532	12418	11461	12186
eil51	**426**	500	654	457	485	450	507	436	497	432	522	439	477	433	463	439	463
berlin52	**7542**	9805	12391	8182	8885	8444	9938	7542	9150	7975	9498	8107	9335	7641	8787	7542	8335
brazil58	**25395**	50901	63633	27945	31223	28420	36898	25395	28342	27264	32513	26080	31657	25395	27983	25400	27853
st70	**675**	1712	2000	812	894	755	937	677	817	780	944	752	924	686	792	686	755
eil76	**538**	1243	1482	606	705	614	698	563	634	580	661	576	694	558	593	558	593
pr76	**108159**	285844	330947	125250	146283	150306	185553	110023	129050	116897	141029	118891	145627	110641	122012	110212	120366



Grafik 1 Problem çözümlerine ait kutu grafikleri



Grafik 1 Problem çözümlerine ait kutu grafikleri (devam)



Grafik 2 Problemlere Göre Algoritmaların Çözüm Aralıkları

Tablo 5'te yer alan veriler kullanılarak elde edilen Grafik 2'de görüldüğü üzere hk48, brazil58 ve pr76 problemlerinde tüm algoritmalar geniş aralıklarda sonuçlar üretmiştir. Bu problemlerde algoritmaların optimumu bulma sayılarının sıfır ya da sıfıra yakın olduğu Tablo 4'te görülmektedir. Bu durum çözüm aralığının genişliği ile optimumu bulma sayıları arasında ters bir ilişki olduğu şeklinde yorumlanmıştır.

hk48, brazil58 ve pr76 problemlerine ait uzaklık matrisleri incelendiğinde şehirler (düğümler) arasındaki mesafelerin diğer problemlere oranla daha büyük değerler olduğu görülmüştür. Bu durumda algoritma optimum rotaya çok yakın bir alternatif rota bulsa dahi iki sonuç arası fark daha büyük olmaktadır. Bu durumun algoritmaların çözüm aralığını genişlettiği söylenebilir. Ayrıca hiçbir metasezgisel yöntemin optimum sonucu bulamadığı pr76 problemlinde çözüm aralığı diğer problemlere göre daha büyük olmuştur.

Tablo 4 ve Tablo 5'ten hareketle metasezgisel yöntemlerin her bir problem için bulduğu minimum ve ortalama değerlerin optimumdan sapmaları aşağıdaki şekilde hesaplanmıştır:

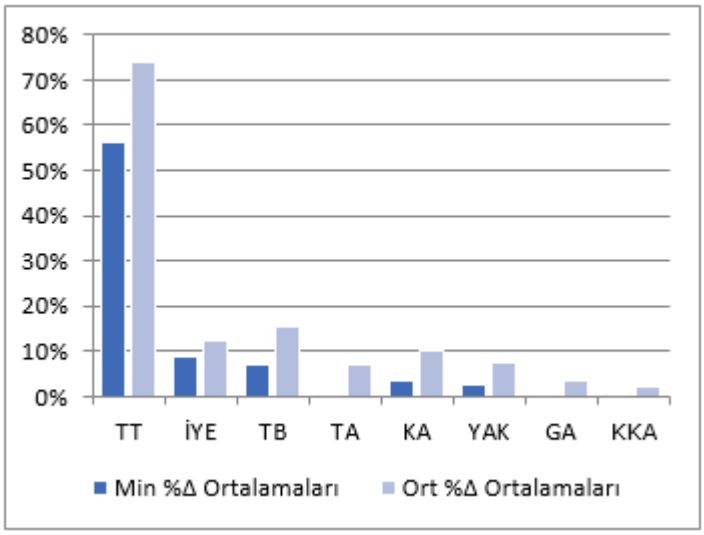
$$Min\ \%\Delta = \frac{Minimum - Optimum}{Optimum}\ x\ 100, \qquad Ort\ \%\Delta = \frac{Ortalama - Optimum}{Optimum}\ x\ 100$$

Her bir problem için hesaplanan Min %Δ ve Ort %Δ' ların her bir metasezgisel yöntem için ortalamaları Tablo 6 de verilmiştir.

Tablo 6 Minimumların ve Ortalamaların Optimumdan Sapmalarının Ortalamaları

Metasezgisel Yöntem	Min %Δ Ortalamaları	Ort %Δ Ortalamaları
Tepe Tırmanma (TT)	56,4%	73,9%
İteratif Yerel Arama (İYE)	8,7%	12,4%
Tavlama Benzetimi (TB)	7,3%	15,4%
Tabu Arama (TA)	0,5%	7,1%
Kanguru Algoritması (KA)	3,6%	10,3%

Yapay Arı Kolonisi (YAK)	2,6%	7,3%
Genetik Algoritma (GA)	0,6%	3,4%
Karınca Kolonisi Algoritması (KKA)	0,5%	2,0%



Grafik 3 Minimum ve Ortalamaların Optimumdan Sapmalarının Ortalamaları

Tablo 6 ve Grafik 3'te görüldüğü üzere popülasyon temelli algoritmaların (YAK, GA, KKA) genel olarak tek noktadan arama yapan algoritmalara (KA, TT, İYE, TB, TA) göre optimuma daha yakın çözümler bulmuştur. TA algoritması ise popülasyon temelli algoritmalar kadar iyi çözümler elde etmiştir.

Minimum %Δ ve Ortalama %Δ değerlerine bakıldığında, GA ve KKA algoritmalarının her bir problem için bulduğu en iyi çözümlerin (Min) ve her bir problem için bulduğu tüm çözüm ortalamalarının (ortalama) diğer algoritmalara göre, optimuma daha yakın sonuçlar elde ettiğini görülmektedir. TA algoritmasının ise bulduğu en iyi çözümler (Min) GA ve KKA kadar optimuma yakın olmasına rağmen her bir problem için bulduğu tüm çözüm ortalamaları GA ve KKA algoritmalarına göre optimumdan daha uzaktır. Buna göre, GA ve KKA algoritmaları çözüm kalitesinin sürekliliği açısından ön plana çıkmaktadır.

7. Sonuç

Uluslararası literatürde sıklıkla rastlanan metasezgisel yöntemlerin GSP'ye uygulanması çalışmaları, ulusal literatürde yeterince işlenmemiştir. Özellikle bilinen belli başlı metasezgisellerin birlikte ele alındığı bir uygulama çalışması bulunmamaktadır. Bu eksikliğin de giderilmesi amacıyla, çalışmada 8 metasezgisel yöntem hakkında bilgi verilmiş daha sonra da literatürden alınan 18 gezgin satıcı problemi ile uygulamaları yapılmıştır. Bulunan sonuçlar tablolarda verilip sonuçlar yorumlanmıştır.

Genel olarak birden fazla noktada arama yapan metasezgisellerin, tek nokta araması yapan algoritmalara göre daha iyi sonuç verdiği görülmüştür. Bunların içerisinden KKA, GSP problemlerine oldukça tatmin edici sonuçlar vermiştir. Tek nokta arama yapan metasezgisellerden de TA diğerlerine göre öne çıkmıştır. Mevcut iterasyon sayılarında,

belli bir düğüm sayısından sonra, iyi sonuçlar bulunsa da optimumu bulmak zorlaşmış ve bazı problemlerde hiç bulunamamıştır.

Stokastik çözüm yöntemlerinden olan metasezgiseller optimizasyon problemlerinde sıklıkla kullanılmaktadır. Çalışmada GSP problemine uygulanan yöntemler, modellenme, hızlı sonuç üretme ve probleme göre esnetilme açılarından klasik yöntemlere göre daha kolay kullanılabilecek yöntemlerdir. Bu çalışma genişletilerek, farklı GSP problemleri üzerinde denemeler yapılabileceği gibi, son dönemlerde yaygın olarak literatürde karşılaşılan melez metasezgiseller geliştirilerek çözüm performansı ve hızı geliştirilebilir.

Bu çalışmada temel olarak metasezgisellerin makul zaman diliminde ürettiği sonuçlar karşılaştırıldığından CPU zamanları eklenmemiştir. Ayrı bir çalışma olarak algoritmaların iterasyon sayıları belli bir mantık çerçevesinde eşitlenerek, optimumu ya da en iyi çözümü bulma zamanları da karşılaştırılabilir.

Kaynakça

[1] E. Ateş, *Karınca Kolonisi Optimizasyonu Algoritmaları İle Gezgin Satıcı Probleminin Çözümü Ve 3 Boyutlu Benzetimi*, Basılmamış Lisans Tezi, Ege Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, İzmir, 2012.

[2] V. V. Nabiyev, *Yapay Zeka - İnsan Bilgisayar Etkileşimi,* Seçkin Yayıncılık, Ankara, 2007.

[3] E. Önder, M. Özdemir, B.F. Yıldırım, Combinatorial Optimization Using Artificial Bee Colony Algorithm And Particle Swarm Optimization. *Kafkas Üniversitesi İktisadi ve İdari Bilimler Fakültesi (KAUİİBF) Dergisi*, 4, 6, 59-70 (2013).

[4] C.H. Papadimitriou, K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity,* Mineola, NY: Dover, 308-309, 1998.

[5] İ. Kara, T. Derya, E. Demir, T. Bektaş, "Genelleştirilmiş Gezgin Satıcı Probleminin Tamsayılı Doğrusal Karar Modeli", *Yöneylem Araştırması / Endüstri Mühendisliği 25. Ulusal Kongresi*, Koç Üniversitesi, 4-6 Temmuz, İstanbul, 2005.

[6] R. Matai, S.P. Singh, M.L. Mittal, "Traveling Salesman Problem: An Overview of Applications, Formulations, and Solution Approaches" in *Traveling Salesman Problem, Theory and Applications* Donald Davendra (Ed.), InTech, Croatia, 2010, 1-24.

[7] P. Mattsson, *The Asymmetric Traveling Salesman Problem*, Uppsala Universitet, 2010

[8] N. Aras, B. Boyacı, D. Koşucuoğlu, D. Aksen, Karlı Gezgin Satıcı Problemi için Sezgisel Yöntemler, *Yöneylem Araştırması / Endüstri Mühendisliği 27. Ulusal Kongresi,* İzmir, 2007.

[9] Ö.N. Koç, *Zaman Pencereli Gezgin Satıcı Problemi İçin Yeni Karar Modelleri,* Başkent Üniversitesi, Fen Bilimleri Enstitüsü, Basılmamış Yüksek Lisans Tezi, 2012.

[10] B. Zhang, J. Peng, "Uncertain Traveling Salesman Problem". Erişim linki: http://orsc.edu.cn/online/110731.pdf, 24 Ocak 2014 .

[11] S. Yadlapalli, S.Rathinam, S. Darbha, 3-Approximation Algorithm for a Two Depot, Heterogeneous Traveling Salesman Problem. *Optimization Letters*, 6, 1, 141-152 (2012).

[12] B.W. Haskell, A. Toriello, M. Poremba, D.J. Epstein, A Dynamic Traveling Salesman Problem with Stochastic Arc Costs Department of Industrial and Systems Engineering University of Southern California Los Angeles, California (2013).

[13] İ. Kara, E. Demir, Genelleştirilmiş Gezgin Satıcı Poblemi İçin Yeni Tamsayılı Karar Modelleri, *Yöneylem Araştırması / Endüstri Mühendisliği 26. Ulusal Kongresi*, Kocaeli Üniversitesi, 3-5 Temmuz, Kocaeli, 2006.

[14] C.S. Helvig, G. Robins, A. Zelikovsky, The Moving-Target Traveling Salesman Problem Volition Inc. *Journal of Algorithms*, 153-174 (1998).

[15] M. Diaby, The Traveling Salesman Problem: A Linear Programming Formulation. *WSEAS Transactions on Mathematics*, 6, 6, 745–754 (2007).

[16] C. Malandraki, RB. Dial, A Restricted Dynamic Programming Heuristic Algorithm for Time Dependent Traveling Salesman Problem. *European Journal of Operations Research*, 90, 1, 45–55 (1996).

[17] M.Padherg, R. Rinaldi, Optimization of a 532-City Symmetric Travelling Salesman Problem by Branch and Cut. *Operations Research Letters*, 6, 1, 1-7 (1987).

[18] M. Padberg, G. Rinaldi, Branch-and-Cut Approach to a Variant of the Traveling Salesman Problem. *Journal of Guidance, Control, and Dynamics*, 11, 5, 436–440 (1988).

[19] S.Lin, B. Kernighan, An Effective Heuristic Algorithm for the Traveling-Salesman Problem. *Operations Research*, 21, 2, 498-516 (1973).

[20] A. Punnen, F. Margot, S.Kabadi, TSP Heuristics: Domination Analysis and Complexity. *Algorithmica*, 35, 111–127 (2003).

[21] O. Martin, S. Otto, E. Felten, Large-step markov chains for the traveling salesman problem. *Complex Systems*, 5, 3, 299-326 (1991).

[22] S. Kirkpatrick, C. D. Gelatt, M. P. Vechi, Optimization by Simulated Annealing. *Science*, 220, 4598, 671-680 (1983).

[23] S. Kirkpatrik, Optimization by Simulated Annealing: Quantitative Studies. *Journal of Statistical Physics*, 34, 1984, 975-986 (1984).

[24] JB. Wu, SW. Xiong, N. Xu, Simulated Annealing Algorithm Based on Controllable Temperature for Solving TSP. *Application Research of Computers*, 24, 5, 66–89 (2007).

[25] M.Dam, M. Zachariasen, *Tabu Search on the geometric travelling salesman problem. Meta-heuristics: Theory and Applications*, Kluwer Academic Publishers, Boston, 571-587, 1996.

[26] Y. Wu, X. Zhou, Meliorative Tabu Search Algorithm for TSP Problem. *Journal of Computer Engineering and Applications*, 44, 1, 57–59 (2008).

[27] C.A. Hurkens, G.J. Woeginger, On the Nearest Neighbor Rule for the Traveling Salesman Problem. *Operations Research Letters*, 32, 1, 1-4 (2004).

[28] M. Dry, K. Preiss, J. Wagemans, Clustering, Randomness, and Regularity: Spatial Distributions and Human Performance on the Traveling Salesperson Problem and Minimum Spanning Tree Problem. *The Journal of Problem Solving*, 4, 1, 2 (2012).

[29] S.Akyol, B. Alataş, Güncel Sürü Zekası Optamizasyon Algoritmaları. *Nevşehir Üniversitesi Fen Bilimleri Enstitüsü Dergisi*, 1, 1, 36-50 (2012).

[30] M.B. Fogel, The Genetic Algorithm for TSP. *IEEE Transaction on systems*, 16, 1-13 (1986).

[31] S. Chatterjee, C. Carrera, L. A. Lynch, Genetic Algorithms and Traveling Salesman Problems. *European Journal of Operational Research*, 93, 3, 490-510 (1996).

[32] P. Larranaga, C.M.H. Kujipers, R.H. Murga, I.Innza, S. Dizdarevic, Genetic Algorithms for the Travelling Salesman Problem: A Review of Representations and Operators. Artificial Intelligence Review, 13, 2, 129-170 (1999).

[33] H.K. Tsai, J.M. Yang, Y.F. Tsai, C.Y. Kao, Heterogeneous Selection Genetic Algorithms for Traveling Salesman Problems. *Engineering Optimization*, 35, 3, 297–311 (2003).

[34] S.S. Ray, S. Bandyopadhyay, S.K. Pal, New Operators of Genetic Algorithms for Travelling Salesman Problem. Proceedings of the 17th International Conference on Pattern Recognition, 23 – 26 August 2004, Cambridge, 497-500, (2004).

[35] J.H. Yang, C.G. Wu, H.P. Lee, Y.C. Liang, Solving Traveling Salesman Problems Using Generalized Chromosome Genetic Algorithm. *Progress in Natural Science*, 18, 887–892 (2008).

[36] Z. Tao, TSP Problem Solution Based On Improved Genetic Algorithm. *In proceedings of the 2008 Fourth International Conference on Natural Computation*, ICNC, IEEE Computer Society, Washington, DC, vol. 01, 686-690, 2008.

[37] B.F. Al-Dulaimi, A.A. Hamza, Enhanced Travelling Salesman Problem Solving by Genetic Algorithm Technique (TSPGA*). Proceedıngs Of World Academy Of Scıence, Engıneerıng And Technology*, 28, 296-302 (2008).

[38] M. Bhattacharyya, A.K. Bandyopadhyay, Comparative Study of Some Solution Methods for Traveling Salesman Problem Using Genetic Algorithms. *Cybernetics and Systems*, 40, 1–24 (2009).

[39] O.M. Sallabi, Y. El-Haddad, An Improved Genetic Algorithm to Solve the Travelling Salesman Problem. *World Academy of Science, Engineering and Technology*, 3, 403-406 (2009).

[40] L. Shi, Z.Li, An Improved Pareto Genetic Algorithm for Multi-Objective TSP. *Fifth International Conference on Natural Computation*, Tianjian, China, 585–588, 2009.

[41] S. Elaoud, J. Teghem, T. Loukil, Multiple Crossover Genetic Algorithm for the Multi-Objective Traveling Salesman Problem. *Electron Notes Discrete Math*, 36, 939–946 (2010).

[42] M.Albayrak, N. Allahverdi, Development a New Mutation Operator to Solve the Traveling Salesman Problem by Aid of Genetic Algorithms. *Expert Systems with Applications,* 38, 1313–1320 (2011).

[43] M. Dorigo, L. M. Gambardella, Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. *IEEE Transactions on Evolutionary Computation*, 1, 53–66 (1997).

[44] T. Stutzle, H. Hoos, "The MAX–MIN Ant System and Local Search for the Traveling Salesman Problem". *In Proceedings of the IEEE International Conference on Evolutionary Computation*, Piscataway, USA, 309–314, 1997.

[45] J. Montgomery, M. Randall, The Accumulated Experience Ant Colony for the Traveling Salesman Problem. *International Journal of Computational Intelligence and Applications,* 3, 189–198 (2003).

[46] C. Garcı´a-Martı´nez, O. Cordo´n, F. Herrera, An Empirical Analysis of Multiple Objective Ant Colony Optimization Algorithms for the Bi-Criteria TSP. Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics), 61–72, (2004).

[47] J. L. Liu, Rank-Based Ant Colony Optimization Applied to Dynamic Traveling Salesman Problems. *Engineering Optimization*, 37, 8, 831–847 (2005).

[48] Q. Zhu, S. Chen, "A new Ant Evolution Algorithm to Resolve TSP Problem". *IEEE Sixth Conference On Machine Learning and Applications* (ICMLA), Cicinnati, Ohio, USA, 62-66, 2007.

[49] F. Herrera, C. Garcia-Martinez, O. Cordon, A Taxonomy and an Empirical Analysis of Multiple Objective Ant Colony Optimization Algorithms for the Bi-Criteria TSP. European Journal of Operations Research, 80, 1, 116–148 (2007).

[50] C. Xu, J. Xu, H. Chang, "Ant Colony Optimization Based on Estimation of Distribution For The Traveling Salesman Problem", *Fifth international conference on natural computation*, Tianjin, China, 19–23, 2009.

[51] L. Wong, Y.H.P. Low, C.S. Chong, "A Bee Colony Optimization Algorithm for Traveling Salesman Problem. Modeling ve Simulation", *AICMS 08. Second Asia International Conference on Modelling and Simulation*, Kuala Lumpur, Malaysia, 818-823, 2008.

[52] L. Wong, Y.H.P. Low, C.S. Chong, A Bee Colony Optimization with Local Search for Travelling Salesman. *International Journal on Artificial Intelligence Tools,* 19, 3, 305–334 (2010).

[53] X. Zhang, Q. Bai, X. Yun, "A New Hybrid Artificial Bee Colony Algorithm for the Traveling Salesman Problem", *IEEE 3rd International Conference on Communication Software and Networks*, China, 155-159, 2011.

[54] D. Karaboğa, B. Görkemli, *"A combinatorial Artificial Bee Colony Algorithm for Traveling Salesman Problem", INISTA 2011 - 2011 International Symposium on Innovations in Intelligent Systems and Applications*, İstanbul, Turkey, 50-53, 2011.

[55] T. Nakaguchi, K. Jin'no, M. Tanaka, "Hysteresis Neural Networks for Solving Travelling Salesman Problems", *2000 IEEE International Symposium on Circuits and Systems*, Switzerland, 03, 153-156, 2000.

[56] K.S. Leung, H.D. Jin, Z.B. Xu, An Expanding Self-Organizing Neural Network for the Traveling Salesman Problem. *Neurocomputing*, 6, 267–292, (2004).

[57] T.A.S. Masutti, L.N. de Castro, A Self-Organizing Neural Network Using İdeas From The İmmune System To Solve The Traveling Salesman Problem. *Information Sciences*, 179, 1454–1468 (2009).

[58] M. Li, Z. Yi, M. Zhu, Solving TSP by using Lotka-Volterra Neural Networks. *Neurocomputing,* 72, 16–18, 3873–3880 (2009).

[59] L.N. De Castro, F.J. Von Zuben, The Clonal Selection Algorithm With Engineering Applications. *In Proceedings of GECCO*, 2000, 36-39, (2000).

[60] S. Gao, H. Dai, G. Yang, Z. Tang, A Novel Clonal Selection Algorithm and İts Application to Traveling Salesman Problem. *IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences Series A*, 90, 10, 2318 (2007).

[61] H. Dai, Y. Yang, C. Li, Distance Maintaining Compact Quantum Crossover Based Clonal Selection Algorithm. *Journal of Convergence Information Technology*, 5, 10, 56-65 (2010).

[62] A. Baykasoglu, A. Saltabas, A. S. Tasan, K. Subulan, Yapay Bagışıklık Sisteminin Çoklu Etmen Benzetim Ortaminda Realize Edilmesi ve GSP Uygulanması, *Journal of the Faculty of Engineering ve Architecture of Gazi University*. 27, 4, 901-909 (2012).

[63] K.P. Wang, L. Huang, C.G. Zhou, W. Pang, "Particle swarm optimization for traveling salesman problem. In Machine Learning and Cybernetics", *2003 International Conference on Machine Learning and Cybernetics*, China 3, 1583-1585 2003.

[64] H. Shah-Hosseini, "Problem Solving by Intelligent Water Drops", *2007 IEEE Congress on Evolutionary Computation (CEC 2007)*, Siingapore, 3226-3231, 2007.

[65] N. Javadian, M.G. Alikhani, R. Tavakkoli-Moghaddam, A Discrete Binary Version of the Electromagnetism-Like Heuristic for Solving Traveling Salesman Problem. *In Advanced Intelligent Computing Theories and Applications. With Aspects of Artificial Intelligence*, 123-130 (2008).

[66] M. Gerşil, T. Palamutçuoğlu, Ders Çizelgeleme Probleminin Melez Genetik Algoritmalar İle Performans Analizi. *Niğde Üniversitesi İİBF Dergisi,* 6, 1, 242-262 (2013).

[67] V. Yiğit, O. Türkbey, Tesis Yerleşim Problemlerine Sezgisel Metotlarla Yaklaşım, Gazi Üniversitesi Mühendislik. *Mimarlık Fakültesi Dergisi*, 18, 4, 45-56 (2003).

[68] S. Luke, *Essentials of Metaheuristics*, Second Edition, Lulu, USA, 2013.

[69] H.R. Lourenço, O.C. Martin, T. Stutzle, "A Beginner's Introduction to Iterated Local Search", *4th Metaheuristics International Conference (MIC'01),* Porto, Portugal, 16-20, 2001.

[70] A. Juan, H.R. Lourenço, M. Mateo, R. Luo, Q. Castella, Using Iterated Local Search For Solving the Flow-Shop Problem:Paralleization. *Parametrization and Randomization Issues*, 103-126 (2013).

[71] N.S. Kumbharana, G.M. Pandey, A Comparative Study of ACO, GA and SA for Solving Travelling Salesman Problem. *International Journal of Societal Applications of Computer Science*, Vol 2, Issue 2, (February 2013).

[72] A. Söke, Z. Bingül, İki Boyutlu Giyotinsiz Kesme Problemlerinin Benzetilmiş Tavlama Algoritması ile Çözümlerinin İncelenmesi. *Politeknik Dergisi*, 8, 1, 25-35 (2005).

[73] A. Tokgöz, S. Bulkan, Weapon Target Assignment with Combinatorial Optimization Techniques. (*IJARAI) International Journal of Advanced Research in Artificial Intelligence*, 2, 7 (2013).

[74] F. Glover, Future Paths For İnteger Programming and Links to Artificial İntelligence, *Computer and Operation Research*, 13, 533-549 (1986).

[75] P. Hansen, "The Steepest Ascent Mildest Descent Heuristic for Combinatorial Programming", *Proceedings of the Congress on Numerical Methods in Combinatorial Optimization*, Capri, Italy, 1986.

[76] T. Cura, *Modern Sezgisel Teknikler ve Uygulamaları*, Papatya Yayıncılık, İstanbul, 2008.

[77] D. Karaboğa, *Yapay Zekâ Optimizasyon Algoritmaları*, Genişletilmiş II. Basım, Nobel Yayın Dağıtım, İstanbul, 2011.

[78] K. Güney, A. Akdağlı, "Tabu Araştırma Algoritması İle Lineer Anten Dizisinin Genlik Uyarım Katsayılarının Belirlenmesi", *8. Ulusal Elektrik Elektronik Bilgisayar Mühendisliği Kongresi, Gaziantep Üniversitesi*, 456-459, 1999.

[79] J.M. Pollard, Monte Carlo Methods for Index Computation (mod p). *Mathematics of Computation*, 32, 143, 918–924 (1978).

[80] P. van Oorschot, M. Wiener, On Diffie-Hellman Key Agreement with Short Exponents. *Advances in Cryptology*, 332-343 (1996).

[81] L. Duta, J.M. Henrioud, I. Caciula, "A Real Time Solution to Control Disassembly Processes", *Proceedings of the 4th IFAC Conference on Management and Control of Production and Logistics*, Sibiu, 2007.

[82] Y.E. Demirtaş, T. Keskintürk, Kanguru Algoritması ve Gezgin Satıcı Problemine Uygulanması. *İstanbul Ticaret Üniversitesi Fen Bilimleri Dergisi*, 10, 19, 51-63 (2011).

[83] D. Karaboğa, An İdea Based On Honey Bee Swarm For Numerical Optimization. Technical Report TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.

[84] J. McCaffrey, "Natural Algorithms: Use Bee Colony Algorithms to Solve Impossible Problems", MSDN Magazines, erişim linki: http://msdn.microsoft.com/en-us/magazine/gg983491.aspx, 24 Ocak 2014.

[85] D.E. Goldberg, *Genetic Algorithms in Search Optimization and Machine Learning*, Addison Wesley Publishing Company, USA, 1989.

[86] Z. Michalewicz, *Genetic Algorithms + Data Structure = Evolution Programs*, Springer-Verlag, Berlin, 1992.

[87] C.R. Reeves, *Modern heuristic techniques for combinatorial problems*, McGraw-Hill Book Company Inc., Europe. 1995.

[88] M. Obitko, Genetic Algorithms, (Online), erişim linki: http://cs.felk.cvut.cz/~xobitko/ga/ Hochschule für Technik und Wirtschaft Dresden (FD), (1998).

[89] R. Sarker, C. Newton, A Genetic Algorithm for Solving Economic Lot Size Scheduling Problem. *Computer & Industrial Engineering*, 12, 5, 195-196 (2002).

[90] R. Cheng, M. Gen, T. Yasuhiro, A Tutorial Survey of Job-Shop Scheduling Problems Using Genetic Algorithms, Part II: Hybrid Genetic Search Strategies. *Computers and Industrial Engineering*, 36, 343-364 (1999).

[91] T. Keskintürk, S. Şahin, Doğrusal Olmayan Regresyon Analizinde Gerçek Değer Kodlamalı Genetik Algoritma. *İTÜ Sosyal Bilimler Dergisi*, 8, 167-178 (2009).

[92] T. Keskintürk, H. Söyler, Global Karınca Kolonisi Optimizasyonu. *Gazi Üniversitesi Mimarlık ve Mühendislik Dergisi*, 21,689-698 (2006).

[93] H. Söyler, T. Keskintürk, "Karınca Kolonisi Algoritması İle Gezen Satıcı Probleminin Çözümü", *8. Türkiye Ekonometri ve İstatistik Kongresi*, Malatya, 1-11, 2007.

[94] M. Dorigo, *Ottimizzazione, apprendimento automatico, ed algoritmi basati su metafora naturale (Optimization, Learning and Natural Algorithms),* Ph.D.Thesis, Politecnico di Milano, Italy, in Italian. DT.01-POLIMI92, 1992.

[95] J. Brownlee, *Clever Algorithms, Nature-Inspired Programming Recipes*, Open Source Book, erişim linki: http://www.CleverAlgorithms.com, 2012.

[96] TSPLIB. Erişim linki: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/ Online, 12.12.2013.

[97] K. Deep, H. Mebrahtu, Variant of partially mapped crossover for the Travelling Salesman Problems. *International Journal of Combinatorial Optimization Problems and Informatics*, 3, 1, 47-69, (2012).

[98] T. Keskintürk, "Permutasyon Kodlamalı Genetik Algoritmada Operatörlerin Etkinliklerinin Araştırılması", *VI. Ulusal Üretim Araştırmaları Sempozyumu*, İstanbul, 225-231, 2006.