Academic Platform Journal of Engineering and Science

journal homepage: http://apjes.com/

0-1 Çok Boyutlu Sırt Çantası Probleminin Feromonal Yapay Arı Koloni (fYAK) Algoritması ile Çözümü

*Dursun Ekmekci

Karabük Üniversitesi TOBB MYO Bilgisayar Programcılığı Bölümü, dekmekci@karabuk.edu.tr,

Araştırma Makalesi Geliş Tarihi: 30.10.2019 Kabul Tarihi: 09.04.2020

Öz

Optimizasyon algoritmaları, geliştirilme tarzları itibariyle bazı problemlere daha çok odaklanarak, daha başarılı çözümler üretebilmektedirler. Örneğin sayısal çözüm yaklaşımıyla üretilen yapay arı koloni (YAK) algoritması, nümerik optimizasyon problemlerinde daha başarılı sonuçlara ulaşabilirken, karınca koloni optimizasyonu (KKO), gezgin satıcı problemi (GSP) benzeri ayrık yapılı optimizasyon problemlerinde daha başarılı çözümler üretebilir. 0-1 optimizasyon problemleri, ayrık yapılı problemlerdir. Ancak çözüm elemanları itibariyle optimizasyon problemlerinin üçüncü grubu olarak değerlendirilebilir. Bu çalışmada 0-1 çok boyutlu sırt çantası problemleri için YAK ve KKO algoritmalarının melez versiyonu olarak geliştirilen fYAK algoritması önerilmiştir. Algoritma performansı, popüler test problemleri üzerinde denenmiş ve elde edilen sonuçlar YAK ve KKO sonuçlarıyla karşılaştırılmıştır.

Anahtar Kelimeler: Yapay arı koloni algoritması, Karınca Koloni Optimizasyonu, Feromonal Yapay arı koloni algoritması

The Solution of 0-1 Multidimensional Knapsack Problem with Pheromonal Artificial Bee Colony (pABC) Algorithm

*[1]Dursun Ekmekci

[1]Karabuk University, dekmekci@karabuk.edu.tr

Abstract

Optimization algorithms can produce more successful solutions by focusing more on some problems in terms of their development style. For example, the artificial bee colony (ABC) algorithm produced by the numerical solution approach can achieve more successful results in numerical optimization problems, whereas ant colony optimization (ACO) can produce more successful solutions in discrete structure problems such as traveling salesman problem (TSP). 0-1 optimization problems are discrete structured problems. However, it can be considered as the third group of optimization problems in terms of solution items. In this study, the pABC algorithm developed as a hybrid version of ABC and ACO algorithms for 0-1 multidimensional knapsack problems was proposed. The performance of pABC was tested on popular benchmark problems, and the results obtained by the algorithm were compared with the results of ABC and ACO.

Keywords: Artificial Bee Coloni Algorithm, Ant Colony Optimization, Pheromonal Artificial Bee Coloni Algorithm

1. GİRİŞ

Günlük yaşamda farklı birçok alanda kullanılan optimizasyon kavramı, matematik alanında, bir fonksiyonun minimum ya da maksimum noktalarını bulma işlemi olarak tanımlanabilir. Literatür kaynakları, matematikteki ilk optimizasyon kullanımını, Newton'un çalışmalarına dayandırmaktadır [1]. Newton, bir fonksiyonu 0 (sıfır) yapan kök değerlerini araştırırken ayrıca fonksiyonu minimum ve maksimum yapan sonuçları da aramıştır. Yapay zekâ açısından bakıldığında ise optimizasyon, bir problemin istenen kısıtları sağlayan optimal değerini (minimum ya da maksimum) arama işlemi olarak yorumlanır. Optimizasyon probleminde, değişkenler gerçek sayılardan oluşuyorsa, bu tür problemler sürekli optimizasyon problemleri olarak adlandırılır. Problem değişkenleri, sonlu bir kümedeki

*[1]Sorumlu Yazar: Karabük Üniversitesi Demir Çelik Kampüsü TOBB Teknik Bilimler Meslek Yüksekokulu M127 78050 / KARABÜK, dekmekci@karabuk.edu.tr 0-370 418 9417

Doi: 10.21541/apjes.640252

nesnelerden oluşuyorsa bu problemlere, ayrık yapılı (kombinatoryal) optimizasyon problemleri denir. İkinci kategorideki problemlerde, problemin çözüm uzayı, değişken sayısıyla bağlantılı olarak, üstel biçimde genişlediğinden, bu tür problem çözümlerinde klasik yöntemler tercih edilmez. Dolayısıyla kombinatoryal optimizasyon problemlerini çözmek için, problemin arama uzayının tamamını taramak yerine, geçerli çözüm bölgelerine yönelmeyi hedefleyen zeki sezgisel yöntemler geliştirilmiştir. Sezgisel yöntemler, optimum çözümü bulmayı garanti etmeseler de düşük hesaplama maliyetiyle, geçerli çözümler üretebilen yöntemlerdir. Bu yöntemler en genel anlamda iki grupta toplanabilir: evrimsel algoritmalar, sürü zekâsı tabanlı algoritmalar. Sürü zekâsı tabanlı algoritmalar, kuş sürüleri, göçmen kuşlar, ateş böcekleri, balık sürüleri gibi genellikle sürü halinde yaşayan canlılar ile, arı, karınca, bakteri gibi koloni halinde yaşayan mikroorganizmaların, aralarında herhangi bir hiyerarşi olmadan oluşturdukları iş bölümünü taklit eder. Bu yöntemler farklı türlerdeki optimizasyon problemleri için, standart biçimleriyle kullanılabildikleri gibi, bir algoritmanın vurgulayıcı bileşeni farklı bir algoritmaya entegre edilerek geliştirilen karma biçimlerde de kullanılabilmektedirler. Gezgin satıcı problemi (GSP) gibi tek kısıtlı, tek boyutlu ve tek amaç fonksiyonuna sahip optimizasyon probleminde başarılı çözümler üretebilen bu yöntemler, çok kısıtlı, çok boyutlu ya da çok amaçlı optimizasyon problemlerinde de başarılı çözümler üretebilmektedirler.

Bu çalışmada, 0-1 çok boyutlu sırt çantası problemi ele alınmış, çözüm yöntemi olarak feromonal yapay arı koloni (fYAK) algoritması tercih edilmiştir. Yapay arı koloni (YAK) algoritması, kâşif arıların rassal besin arayışıyla arama bölgesinin farklı alanlarına yayılabilir. Ancak sömürü faaliyetine hem işçi ve hem de gözcü arılar katılsa da algoritma, bu alanda yetersiz kalmaktadır. Karınca koloni optimizasyonu (KKO), tüm farklı algoritmik modellerinde vurgulayıcı bileşeni olarak "feromon" yaklaşımıyla güçlü bir sömürü yeteneğine sahiptir. Özellikle kombinatoryal optimizasyon problemlerindeki başarılarıyla öne çıkan bu algoritmalar, uygulandıkları optimizasyon problemi için birkaç çevrimde bile başarılı çözümlere ulaşabilir. Ancak araştırmacıların birleştikleri eleştiri, KKO algoritmalarının keşif yeteneğinin yetersizliği yönündedir [2] [3]. fYAK, KKO'nun öne çıkan "feromon" bileşenini YAK algoritmasına entegre ederek standart YAK'ın sömürü yeteneğini güçlendiren ve diğer taraftan, kaşif arıların bağımsız besin arayışını değiştirmeden, algoritmanın keşif yeteneğini koruyan bir melez yöntemdir.

1.1. 0-1 Çok Boyutlu Sırt Çantası Problemi

Sırt çantası problemi, kombinatoryal optimizasyon problem türlerinden biri olarak, yöneylem araştırması literatüründe yaygın bir inceleme alanına sahiptir. Farklı mühendislik disiplinlerinde, karar verme konusunda karşılaşılan birçok problem, sırt çantası problemi yaklaşımıyla yorumlanarak çözümler geliştirilebilir. Klasik sırt çantası probleminde amaç, sınırlı boyuttaki sırt çantasını, değerleri (p) ve hacimleri verilmiş n adet nesneden seçimler yaparak doldurmak ve yüklenen nesnelerin toplam değerini en çok seviyeye (p_{max}) getirmektir. Bu bağlamda 0-1 sırt çantası problemi NP-tam (Non-deterministic Polynomial - Complete) karakterli bir problemdir [4]. Yüklenecek nesnelerin hacimlerinin yanı sıra ağırlıkları da hesaba katılırsa, sırt çantasının taşıma kapasitesi de düşünülerek problem iki boyutlu hale gelecektir. Probleme benzer kısıtlar eklenerek boyutu geliştirilebilmektedir. Bu durum göz önünde bulundurularak, problemin matematiksel modeli (1), (2) ve (3) ile özetlenebilir.

$$p_{max} \sum_{j=1}^{n} p_j x_j \tag{1}$$

$$Kısıtlar \sum_{j=1}^{n} w_{ij} x_j < B_i \qquad i = 1,2,3,..m \tag{2}$$

$$x_j \in \{0,1\} \tag{3}$$

İfadelerde m, problemdeki kısıt sayısını, diğer bir iadeyle problemin boyutunu, p_j, j. nesnenin değerini, w_{ij} ise j. nesnenin i. boyuttaki maliyetini temsil eder. x_j değişkeni j. nesnesi sırt çantasına yüklendiğinde 1, yüklenmediğinde 0 değerini alır.

1.2. Literatür Taraması

Literatürde, 0-1 çok boyutlu sırt çantası problemi için, farklı teknikler içeren çok sayıda çözüm önerisi sunulmuştur. Bu çalışmaların birinde problem çözümü için genetik algoritma önerilmektedir [5]. Problemin "çok seçimli" modelini ele aldıkları çalışmalarında Rafael Parra-Hernandez ve Nikitas J. Dimopoulos, yeni bir sezgisel metot geliştirmişlerdir [6]. Aynı problem daha sonra dal-sınır tekniğiyle de çözülmüştür [7]. Çok boyutlu sırt çantası problemi çözümünde, dal-sınır yönteminden başka etiketleme algoritması [8] ve epsilon-kısıt yöntemi [9] gibi kesin metotlar da kullanılmıştır. Karınca kolonisinin tercih edildiği bir yöntemde, yerel optimal çözüme takılmayı önlemek için farklı teknikler geliştirilirken [10], diğer yöntemde yeni bir feromon güncelleme yaklaşımı denenmiştir [11]. Hanafi ve Wilbaut problem çözümünde dağıtık arama algoritmasını tercih etmişlerdir [12]. Sürü zekası tabanlı sezgisel yöntemlerin kullanıldığı çalışmaların birinde ikili meyve sineği [13], diğerinde ise göçmen kuşlar optimizasyon algoritmaları ile çözümler aranmıştır [14]. Klamroth ve Wiecek uygulanabilecek dinamik programlama yöntemlerini, teorik açıdan detaylı olarak incelemişlerdir [15]. Diğer bir çalışmada, problem farklı bir türeviyle incelenmiştir [16].

2. ÖNERİLEN METOT

YAK algoritması, nümerik optimizasyon problemleri için geliştirilmiş, az sayıda kontrol parametresi ve kolay uygulanabilirlik avantajıyla geniş kullanım alanına erişmiş bir metasezgisel yöntemdir. Algoritma, ayrıca temel yaklaşımı değiştirilmeden, farklı yapılardaki optimizasyon problemlerine de başarıyla uygulanabilmektedir. YAK algoritması, oluşturulduğu günden bugüne, değişik

uzantıları, modifikasyonları ve farklı tekniklerle melezleştirilerek geliştirilmiş modelleri olan bir metasezgiseldir. Bu bağlamda, algoritmanın varyantlarını ve uygulandığı çalışmaları inceleyen çok sayıda literatür incelemesi hazırlanmıştır [17] [18] [19].

Genel bir tanımlamayla, YAK, başlangıçta rassal çözümler oluşturur ve iteratif olarak her bir çevrimde daha başarılı çözümler türeterek optimal çözüme ulaşmayı hedefler. Algoritmanın yerel optimuma takılmaması için kâşif arılar devreye girer ve rastgele ürettikleri çözümlerle aramayı farklı alanlara kaydırır. Dolayısıyla YAK, arama alanının geniş bir alanını tarayabilmektedir. KKO algoritması ise, özellikle ayrık yapılı optimizasyon problemlerinde, birkaç çevrimde bile başarılı çözümlere ulaşabilen bir metasezgisel yöntemdir. KKO, iteratif adımlarla, çözüm bileşenleri arasındaki korelasyonu analiz eder ve bu parçaları en doğru sırada yerleştirerek optimal çözümü oluşturmaya çalışır. Bu özelliğiyle KKO, güçlü bir sömürü yeteneğine sahiptir. Bu kapsamda, YAK ve KKO algoritmalarının bahsedilen avantajlarını birlikte kullanan ya da birleştiren farklı literatür çözümleri bulunmaktadır. Bu çalışmaların birinde, YAK ile sınıflandırma, KKO ile rotalama/yönlendirme yapılmıştır [20]. Diğer bir çalışmada optimizasyon problemi iki bölüme ayrılmış, optimal yönlendirme için KKO, boyutlandırma için YAK kullanılmıştır [21]. Farklı bir çalışmada, KKO ile ilk çözümler oluşturulur ve sonrasında işçi arılar, bu çözümlerden yeni çözümler üreterek birinci seviye aramayı tamamlar [22]. Her döngüde, genel feromon güncellemesi gözcü arıların en başarılı sonucuna göre yapılır ve kötü çözümler filtrelenir. Feromon bileşeninin YAK'a entegre edildiği bir yöntemde [23], bal arıları arasındaki iletişim için feromon salgısı kullanılmaktadır. Kâşif arılar çözüm üretirken, rassal çözümler oluşturmak yerine işçi ve gözcülerin yaydıkları feromon salgısını referans alırlar. Yazarlar, sonraki çalışmalarında, yaklaşımı, birkaç modifikasyonla geliştirmişlerdir [24].

YAK modelinde, her bir çevrimde işçi ve gözcü arı fazlarındaki aramalar, algoritmanın sömürü yeteneğini artırmaya yöneliktir. İşçi arı fazında mevcut çözümler sırasıyla değerlendirilirken, gözcü arı fazında rulet tekerleği kullanılarak başarılı çözümlerin değerlendirilme olasılığı artırılır. Kâşif arılar ise, bağımsız aramalarla, aramayı yerel en iyi çözüme takılmaktan kurtarır. [23] ve [24] deki çalışmalarda, feromon bileşeni işçi ve gözcülerle elde edilen çözümlerden sonra, uygulanmış, kâşif arıların daha başarılı çözümler üretmesi kolaylaştırılmıştır. fYAK algoritmasında ise, feromon, işçi ve gözcü arıların haberleşmesinde kullanılmakta, bu sayede gözcüler, işçilerin tecrübesinden daha fazla yararlandırılmaktadır. Kâşif arı fazında ise, YAK standardında olduğu gibi rassal çözümler üretilerek, algoritmanın keşif yeteneği korunmaktadır. fYAK algoritması, nümerik [25] ve ayrık yapılı [26] optimizasyon problemlerine uygulandığında başarılı çözümler üretebilmiştir. Bu çalışmada ise, fYAK, çözüm elemanları 0 ve 1'lerden oluşan optimizasyon problemine uygulanmış, performansı araştırılmıştır. Algoritmanın çözüm üretme ve feromonu uygulama biçimi Bölüm 3'te detaylı olarak paylaşılmaktadır. Önerilen yöntemin daha iyi aktarılabilmesi için, öncelikle standart YAK analiz edilecek ve KKO'nun feromon bileşeni açıklanacaktır.

2.1. Yapay Arı Koloni (YAK) Algoritması

Bal arılarının gerçek yaşamdaki yiyecek arama davranışını taklit eden YAK algoritması, Karaboğa tarafından 2005 yılında geliştirilmiştir [27]. Algoritma, aralarında herhangi bir hiyerarşi olmadan, yiyecek arama faaliyetine katılan üç bal arısı türünün iş bölümünü modeller. Bu arılar: kovan çevresini bağımsız biçimde rastgele tarayan kâşif arılar, kâşif arılarca belirlenen kaynaklardan besin toplamakla görevli işçi arılar ve işçi arılardan edindikleri bilgiye göre yöneleceği besin kaynağını kendisi tercih eden gözcü arılardır. Algoritma yaklaşımında besin kaynağı, problemin her bir olası çözümünü temsil eder. Dolayısıyla bal arıları yiyecek arama sürecinde, en uygun besin kaynağına ulaşmayı hedefler. Besin kaynağı sayısı, işçi arı ve gözcü arı sayısına eşittir. Bu bağlamda kolonideki arı sayısı, besin kaynağı sayısının iki katı kadardır. İşçi arılar, görevli oldukları besin kaynağı komşuluğundaki diğer kaynakları da ziyaret ederek taramayı güçlendirir. Daha kaliteli kaynak bulduklarında, bir sonraki uçuşlarında aç gözlü yaklaşım gereği, bu yeni kaynağa yöneleceklerdir. Aksi takdirde yine mevcut kaynak komşuluğunda, yeni kaynaklar ararlar. İşçi arıların diğer bir görevi ise, görevli oldukları besin kaynağıyla ilgili bilgiyi gözcü arılarla paylaşmaktır. Gözcüler, işçilerden farklı olarak, yönelecekleri besin kaynağını kendileri tercih eder. Böylece yeni çözüm türetmede, başarılı çözümlerin seçilme olasılığı artırılmış olur. İşçi ve gözcü arılarca devam eden besin toplama sürecinde, kaynaklardaki besinler azalacak ve zamanla yetersiz hale gelecektir. Bu durumda ilgili arılar, kâşif arı yaklaşımıyla kovan çevresinde yeni besin kaynakları arayacaklardır. Kâşif arı rolünün algoritmadaki karşılığı, yerel optimum tuzağını aşabilmek için, arama alanının farklı bölgelerine sıçramak olarak yorumlanabilir. Anlatılanlar ışığında YAK algoritması şu şekilde özetlenebilir:

Algoritma 1- YAK Algoritması

```
Başlangıç besin kaynaklarının bulunması
Kaynak kalitelerinin belirlenmesi
Tekrarla
    İşçi arı safhası
    Gözcü arı safhası
    Kâşif arı safhası
    O ana kadar bulunan en iyi çözümün kaydedilmesi
Şartlar sağlanıncaya kadar
```

Kâşif arıların doğada rastgele yiyecek aramasıyla başlayan algoritmada, (4) denklemiyle ifade edildiği şekilde yiyecek kaynakları rastgele yerleştirilir.

$$s_{m,i} = l_i + rand(0,1) * (l_i - u_i) \quad (4)$$

(4) denkleminde, S çözümler kümesi, $s_{m,i}$ m. çözümün, i. elemanının değeridir. l_i, çözümün alt sınırını, u_i ise üst sınırını temsil eder. Başlangıç çözümleri oluşturulduktan

sonra, her bir çözümün, problemin amaç fonksiyonuna göre, *f(s)* değeri hesaplanır.

Belirlenen besin kaynakları komşuluğunda her bir işçi arı tarafından bulunan diğer besin kaynağı, (5) denklemiyle ifade edilmiştir.

$$t_{m,i} = s_{m,i} + \phi_{m,i}(s_{m,i} - s_{r,i}) \quad (5)$$

(5) denkleminde s_r rastgele seçilen bir çözüm, *i* çözümdeki rastgele seçilen eleman ve $\phi_{m,i}$ [-1, 1] aralığında rastgele seçilen katsayıdır.

Türetilen yeni çözümün uygunluk değeri hesaplanır ve eski çözümün uygunluk değeriyle karşılaştırılarak daha başarılı çözüm tercih edilir. Mevcut çözüm seçilirse, çözümün başarısızlık sayacı 1 artırılırken, yeni çözüm seçildiğinde ilgili çözümün başarısızlık sayacı sıfırlanacaktır. s_m çözümünün uygunluk değeri $fit(s_m)$, problemin amaç fonksiyonuna göre hesaplanan $f(s_m)$ değeri kullanılarak, (6) denklemiyle hesaplanmaktadır.

$$fit(s_m) = \begin{cases} 1/(1+f(s_m)) & eğer\ (f(s_m)) \geq 0 \\ 1+abs(f(s_m)) & eğer\ (f(s_m)) < 0 \end{cases} \quad (6)$$

YAK algoritmasında her bir çözümün, gözcü arılarca seçilme olasılığı, çözümlerin uygunluk değerlerine bağlıdır. Buna göre s_m çözümünün seçilme olasılığı r_m, (7) denklemiyle hesaplanmaktadır.

$$r_m = \frac{fit(s_m)}{\sum_{i=1}^{SN} fit(s_i)} \quad (7)$$

Gözcü arı safhasında da mevcut çözüm kullanılarak yeni bir çözüm türetilirken (5) denklemi uygulanır. Türetilen yeni çözümün uygunluk değeri (6) denklemiyle hesaplanır ve işçi arılardaki aç gözlü yaklaşım uygulanır. Kâşif arı safhasında ise, "başarısızlık sayacı", "limit" seviyesine kadar artan çözümler terk edilir. Bu çözümlerin yerine, (4) denklemiyle rastgele yeni çözümler oluşturulur.

2.2. Karınca Koloni Optimizasyonu (KKO)'nun Feromon Bileşeni

Sürü zekası temelli metasezgisel yöntemlerden biri olarak KKO, 1991 yılında Dorigo tarafından, gerçek karıncaların yiyecek arama davranışının modellenmesiyle geliştirilmiştir [28]. Karıncalar, yiyecek kaynağı bulduklarında, bu kaynaktan yuvalarına en kolay yoldan besin taşıyabilmek için en düşük maliyetli güzergahı belirlemeye çalışırlar. Seyahatleri süresince yaydıkları feromon salgısıyla, kendilerini takip eden karıncaların daha kısa yolu bulmalarına yardımcı olurlar. Kolonideki karıncaların yuva-kaynak arası birkaç dolaşımından sonra, en kısa yol belirlenmiş olur.

Karıncaların, en kısa yolu belirlemek için aralarındaki iletişimde kullandıkları feromon salgısı, KKO yaklaşımının temel bileşenidir. Yapay karıncalar, feromon salgısını, çözüm uzayını örneklemek için kullanan, rassal arama elemanlarıdır [29]. Araştırmacılar, KKO'nun farklı birçok uzantısını ve türevini geliştirmişlerdir. Bunlardan en yaygın kullanım alanı olan, karınca sistemi (KS), karınca koloni sistemi (KKS) ve maksimum-minimum karınca sistemi (MMKS) algoritmaları, feromon izini farklı tekniklerle güncelleyen ve değerlendiren yöntemlerdir.

fYAK yönteminde düğüm seçiminde feromondan yararlanma ve her bir çevrimdeki feromon güncellemesi KKS'ye benzer. KKS algoritmasında *i* düğümündeki bir karınca için sonraki düğümün seçiminde iki alternatif vardır. Hangi alternatifin uygulanacağı rassal seçimle belirlenir. [0,1] aralığında rastgele seçilen *q* değeri için $q \leq q0$ durumunda (8)'de gösterilen ilk alternatif uygulanır. *q0* parametresi için genellikle 1'e yakın bir değer belirlendiğinden, ilk alternatifin seçilme olasılığı yüksektir.

$$\max([\tau(i,u)]^{\alpha}.[\eta(i,u)]^{\beta}) \quad (8)$$

(8) de *u*, *i* düğümünden sonra ziyaret edilebilecek alternatif düğümleri temsil eder. $\tau(i,u)$, *i-u* düğümleri arasındaki feromon izidir. $\eta(i,u)$, *i-u* düğümleri arası uzaklığın ($\delta(i,u)$) tersidir ($\eta(i,u)=1/\delta(i,u)$). α ve β parametreleri ise sırasıyla feromon izinin ve uzaklığın önem seviyesini belirleyen sezgisel parametrelerdir.

İkinci alternatifte ise ($q>q0$) bir sonraki düğüm, hesaplanan seçilme olasılıklarına bağlı olarak, rastgele seçilir. Feromon izinin daha yoğun olduğu düğümlerin seçilme olasılıklarının da daha yüksek olduğu bu alternatifte, ziyaret edilebilecek düğümlerin seçilme olasılıkları (9) ile belirlenir.

$$p_{i,j} = \begin{cases} \frac{[\tau(i,u)]^{\alpha}.[\eta(i,u)]^{\beta}}{\sum_{u \in V}[\tau(i,u)]^{\alpha}.[\eta(i,u)]^{\beta}} & eğer(j \in F) \\ 0 & aksi\ halde \end{cases} \quad (9)$$

Alternatif seçimlere dayalı olarak tüm düğümleri dolaşan karıncalar yuvaya döndüklerinde, her bir düğüm arasındaki feromon değeri güncellenir. KKS'de feromon değerleri iki yolla güncellenir: yerel feromon güncellemesi, genel feromon güncellemesi. Yerel feromon güncellemesi (10) eşitliğiyle yapılır.

$$\tau(i,j) = (1-\rho) * \tau(i,j) + \sum_{k=1}^{S} \Delta\tau(i,j)^k \quad (10)$$

(10) eşitliğinde ρ, (0,1] aralığında belirlenen feromon buharlaşma katsayısı, $\Delta\tau(i,j)^k$ ise karınca *k*'nın *i-j* düğümleri arasına eklediği feromon değeridir. Bu değer (11) ile hesaplanır.

$$\Delta\tau(i,j)^k = \begin{cases} \frac{1}{L_k} & karınca\ k, i-j'den\ geçmişse \\ 0 & aksi\ halde \end{cases} \quad (11)$$

L_k, karınca *k*'nın toplam rota maliyetidir.

Genel feromon güncellemesinde ise (10) ve (11)'deki işlemler, her bir çevrimde yalnızca en başarılı karınca rotasına uygulanır.

2.3. Feromonal Yapay Arı Koloni (fYAK) Algoritması

fYAK algoritmasının standart YAK'tan ayırıldığı temel nokta, işçi ve gözcü arıların aralarındaki haberleşme tekniği ve buna bağlı olarak arama uzayındaki muhtemel çözümün temsilidir. Standart YAK algoritmasında her bir muhtemel çözüm besin kaynağı pozisyonuyla temsil edilirken, fYAK'ta toplanan polenlerden oluşan besin ile temsil edilir. Buna bağlı olarak, işçi ve gözcü arılar arasındaki haberleşmede kuyruk dansı yerine feromon salgısı kullanılır. İşçi arılar, besin elde etmek için çiçeklerden polen toplarken, çiçekler arasında feromon yayarlar ve gözcü arılar yönelecekleri çiçekleri seçerken, ortamdaki feromon yoğunluğunu da göz önünde bulundururlar. Gözcü arı safhasında aç gözlü seçim, yeni çözüm ile çözüm dizisi itibariyle yeni çözüme en çok benzeyen mevcut çözüm arasında uygulanır. fYAK algoritmasının temel adımları şu şekildedir:

Algoritma 2- fYAK Algoritması

```
Algoritma parametreleri için değerler ata
Başlangıç besinlerini oluştur
Başlangıç besinlerinin kalitesini değerlendir
Döngü sayacını 1 olarak ayarla
Tekrarla
    Tüm işçi arılar için {
        (4) ile yeni çözüm üret
        Çözüm maliyetini fit(sm) hesapla
        Açgözlü seçim uygula}
    (10) ve (11) ile yerel feromon güncelle
    Genel feromon güncelle
    Tüm gözcü arılar için {
        Bir sonraki düğüm seçiminde {
            [0,1] arası rastgele q üret
            Eğer (q<=q0) ise
                (8) ile sonraki düğümü seç
            Aksi takdirde
                (9) ile sonraki düğümü seç}
        (6) ile çözümün uygunluk değerini hesapla
        Yeni çözüme en çok benzeyen çözümü bul
        Açgözlü seçim uygula}
    Terkedilmiş çözümler için (4)'le yeni çözüm üret
    En iyi çözümü hafızaya al
    Döngü sayacı += 1
Döngü sayacı = iterasyon sayısı olana kadar
```

3. 0-1 fYAK MODELİ VE SEÇİLEN PROBLEME UYGULANIŞI

Önceki bölümde detaylıca açıklanan fYAK algoritması, çalışma kapsamında dizi elemanları 0 ve 1'lerden oluşan ve problemin birden fazla kıstasına cevap verebilen 0-1 çok boyutlu sırt çantası problemi için tasarlanmıştır.

Başlangıç çözümleri için, (4) denkleminde her bir çözüm elemanının alt ve üst sınırları 0 ve 1 olarak belirlenmiş ve rastgele çözümler oluşturulmuştur. İşçi arı safhasında türetilen her bir çözüm elemanı (12) ile belirlenmiştir.

$$t_{m,i} = \begin{cases} 1 & eğer\ (s_{m,i} = s_{r,i} = 1) \\ 0 & eğer\ (s_{m,i} = s_{r,i} = 0) \\ rastgele\ (0,1) & eğer\ (s_{m,i}! = s_{r,i}) \end{cases} \quad (12)$$

İşçi arı safhasında oluşturulan her bir dizi, problemin her bir boyutundaki kapasite kısıtına göre kontrol edilmiş, herhangi bir boyuttaki kapasite aşıldığında çözüm geçersiz sayılmıştır. Türetilen geçersiz çözümlerin $f(s_m)$ değeri 0 olarak belirlenmiş, geçerli çözümlerin $f(s_m)$ değeri ise, (13) eşitliğinde gösterildiği gibi, çözümün elde ettiği toplam faydaya eşitlenmiştir.

$$f(s_m) = \sum_{j=1}^{n} p_j x_j \quad (13)$$

Türetilen her bir çözümün uygunluk değeri, ilgili çözümün $f(s_m)$ değerinin, tüm nesnelerin toplam faydasına (14) oranlanmasıyla (15) bulunmuştur.

$$Toplam\ Fayda = \sum_{j=1}^{n} p_j \quad (14)$$

$$fit(s_m) = \frac{f(s_m)}{Toplam\ Fayda} \quad (15)$$

Ardından türetilen çözüm ile mevcut çözüm arasında, uygunluk değerlerine göre aç gözlü seçim uygulanmıştır.

Yerel feromon güncellemesi yapılırken (10) denklemi uygulanmış, ancak denklemdeki $\Delta\tau(i,j)^k$ değeri (16) ile belirlenmiştir.

$$\Delta\tau(i,j)^k = \begin{cases} \frac{1}{L_k} & eğer\ (i = j = 1\ ve\ i,j \in s_k) \\ 0 & aksi\ halde \end{cases} \quad (16)$$

Genel feromon güncellemesinde (10) ve (16)'teki işlemler, yalnızca çevrimdeki en başarılı işçi arı için uygulanmıştır.

Gözcü arı safhasındaki ilk alternatif ($q<=q0$) seçimde (17), ikinci alternatifte ise (18) uygulanmıştır.

$$t_{m,j} = \begin{cases} 1 & eğer\ (\Delta\tau(i,j) \geq 0.5) \\ 0 & aksi\ halde \end{cases} \quad (17)$$

$$t_{m,j} = \begin{cases} 1 & eğer\ (\Delta\tau(i,j) < 0.5) \\ 0 & aksi\ halde \end{cases} \quad (18)$$

Gözcü arı safhasında oluşturulan her bir çözümün uygunluk değeri (6) ile belirlenmiştir. Yeni çözümdeki dizilim, mevcut çözüm dizileriyle karşılaştırılmış ve yeni çözüme en çok benzeyen mevcut çözümle uygunluk değerlerine göre aç gözlü tercih uygulanmıştır.

4. DENEYSEL ÇALIŞMALAR

fYAK algoritmasının 0-1 çok boyutlu sırt çantası problemindeki çözüm yaklaşımını analiz edebilmek ve performansını değerlendirebilmek için algoritma, .NET platformunda C# programlama dilinde kodlanmıştır. Uygulama, i7-4710MQ 2.50 işlemcili, 8 GB RAM ve Windows 7 işletim sistemli makinede, 4.7 .NET framework kurulu ortamda çalıştırılmıştır. Çalışma kapsamında her bir test problemi için, standart YAK, standart KKO ve fYAK algoritmaları ile birbirinden bağımsız olarak 30'ar deneme yapılmıştır.

4.1. Test Problemleri

Geliştirilen uygulama, OR kütüphanesinden alınmış, literatürde yaygın olarak kullanılan test problemleri üzerinde denenmiştir. Problemlerdeki boyut sayıları (m), nesne sayıları (n) ve optimum çözüm değerleri Tablo 1'de gösterilmektedir.

Tablo 1. Test problemleri

Problem		Boyut Sayısı (m)	Nesne Sayısı (n)	Optimum Sonuç
No	Adı			
1	mknap1-1	10	6	3800
2	mknap1-2	10	10	8706,1
3	mknap1-3	10	15	4015
4	mknap1-4	10	20	6120
5	mknap1-5	10	28	12400
6	mknap1-6	5	39	10618
7	mknap1-7	5	50	16537
8	mknapcb1-5.100-00	5	100	24381
9	mknapcb1-5.100-01	5	100	24274
10	mknapcb1-5.100-02	5	100	23551
11	mknapcb1-5.100-03	5	100	23534
12	mknapcb1-5.100-04	5	100	23991
13	mknapcb4-10.100-00	10	100	23064
14	mknapcb4-10.100-01	10	100	22801
15	mknapcb4-10.100-02	10	100	22131
16	mknapcb4-10.100-03	10	100	22772
17	mknapcb4-10.100-04	10	100	22751

4.2. Parametre Seçimi

Denemelerde, algoritma parametreleri için değerler belirlenirken, literatür çalışmalarında YAK ve KKO algoritmalarının ayrık yapılı optimizasyon problemlerindeki sonuçlarından yararlanılmıştır. Bu bağlamda algoritma parametreleri için seçilen değerler, Tablo 2'de gösterilmektedir.

Tablo 2. Parametre seti

Algoritma	Koloni Boyutu	Limit	α	β	ρ	$q0$
YAK	100	250	-	-	-	-
KKO	100	-	1	5	0.1	0.8
fYAK	100	250	1	5	0.1	0.8

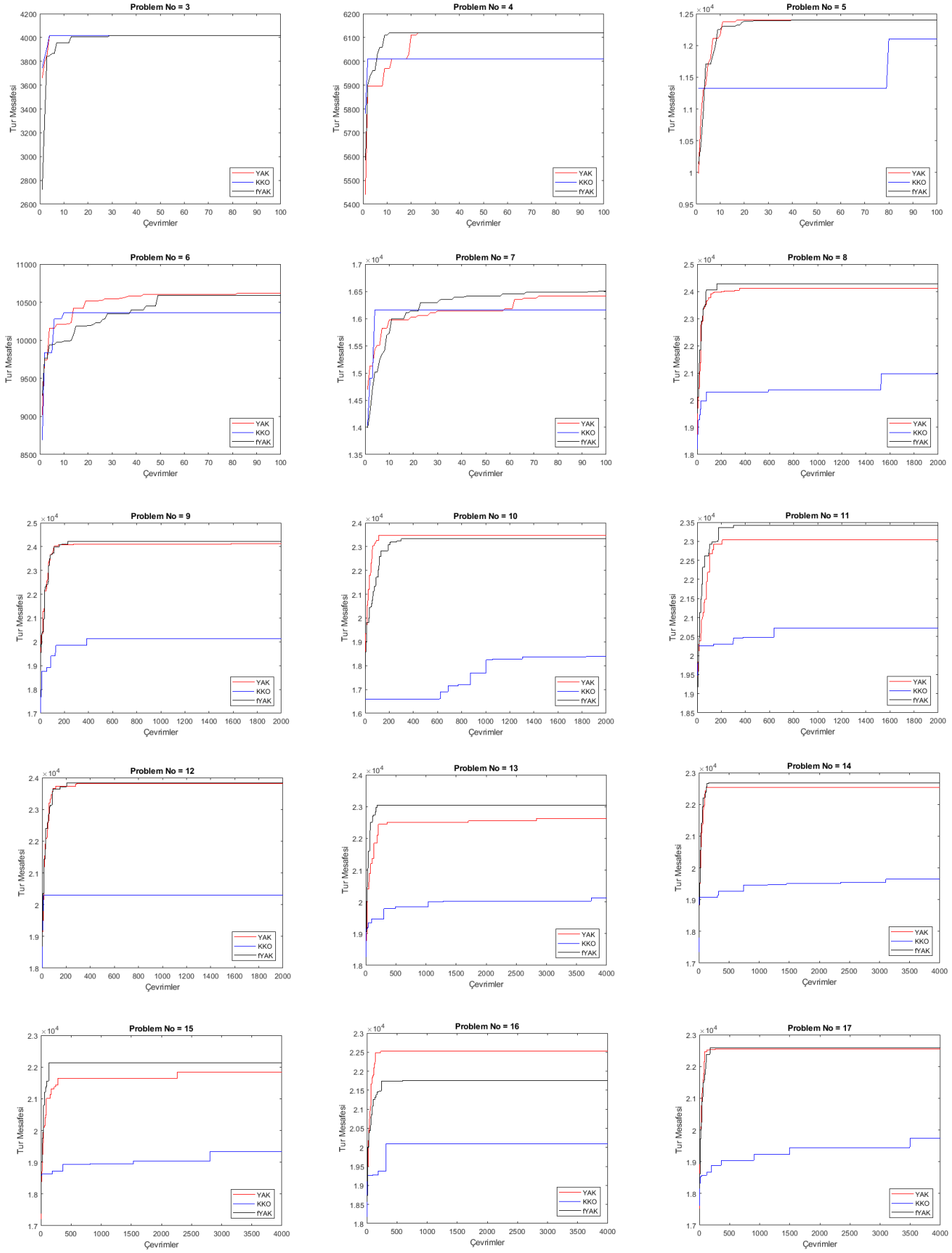
Algoritmaların her bir çevrimde, işlem sayılarındaki farklılık ve YAK ile fYAK algoritmalarında kâşif arı sayılarındaki belirsizlikten dolayı, adil bir karşılaştırma için algoritmalar eşit sürelerde çalıştırılmıştır. Bu kapsamda her bir denemede 1-7 numaralı problemler 3 saniye, 8-12 numaralı problemler 10 saniye ve 13-17 numaralı problemler 20 saniye sürelerde çalıştırılmıştır.

5. BULGULAR VE KARŞILAŞTIRMA

Bu bölümde; YAK, KKO ve fYAK algoritmalarının Tablo 1'de listelenen test problemlerindeki çözüm yaklaşımları incelenmiş ve elde ettikleri sonuçlar birbirleriyle karşılaştırılmıştır. Her üç algoritma da 1 ve 2 numaralı problemler için ilk çevrimde optimum sonuca ulaşmışlardır. Algoritmaların, 3-7 problem çözümlerinde ilk 100 çevrimde, 8-12 numaralı problem çözümlerinde ilk 2000 çevrimde ve 13-17 numaralı problemler için ilk 4000 çevrimde elde ettikleri yakınsaklık performansları Şekil 1'deki grafiklerde sunulmaktadır.

Şekil 1'deki grafiklere bakarak, KKO algoritmasının grafiklerde gösterilen çevrim aralığında, problemlerin genelinde YAK ve fYAK algoritmalarının gerisinde kaldığı söylenebilir. Çözüm için önerilen fYAK algoritmasının, çözüm sürecinde yerel en iyi çözüme takılmadan, daha iyi çözümler üretebildiği görülebilmektedir.

Algoritmaların, çalışma süreleri sonunda her bir problem çözümünde elde ettikleri en iyi sonuçlar, sonuç ortalamaları ve sonuçların, hesaplanan standart sapma değerleri (SS), Tablo 3'te gösterilmektedir. Tabloda ayrıca, algoritmaların buldukları en iyi sonucun, optimal sonuca yakınlık seviyesi (Oran) de ilgili sütunlarda yüzdelik seviyelerde verilmektedir. En başarılı sonuçlar kalın puntolarla ifade edilmiştir.



Şekil 1. YAK, KKO ve fYAK algoritmalarının 0-1 çok boyutlu sırt çantası problem çözümündeki yakınsaklık performansları

Tablo 3. YAK, KKO ve fYAK ile elde edilen sonuçlar

No	Optimum Sonuç	YAK				KKO				fYAK			
		En İyi	Oran	Ortalama	SS	En İyi	Oran	Ortalama	SS	En İyi	Oran	Ortalama	SS
1	3800	**3800**	0,00	**3800,00**	0,00	**3800**	0,00	**3800,00**	0,00	**3800**	0,00	**3800,00**	0,00
2	8706,1	**8706,1**	0,00	**8706,10**	0,00	**8706,1**	0,00	**8706,10**	0,00	**8706,1**	0,00	**8706,10**	0,00
3	4015	**4015**	0,00	3981,66	11,36	**4015**	0,00	**4001,17**	5,16	**4015**	0,00	3996,47	5,41
4	6120	**6120**	0,00	6003,19	60,03	6005	0,02	5947,83	39,55	**6120**	0,00	**6114,00**	3,58
5	12400	**12400**	0,00	12258,90	71,89	11758	0,05	10947,00	93,27	**12400**	0,00	**12390,00**	25,67
6	10618	**10618**	0,00	**10599,04**	7,93	10295	0,03	9461,84	59,43	**10618**	0,00	10541,39	15,62
7	16537	**16537**	0,00	14950,29	51,83	15037	0,09	13503,67	65,47	**16537**	0,00	**16389,87**	37,55
8	24381	24279	0,00	23951,91	71,04	22157	0,09	21780,31	57,98	**24381**	0,00	**24219,41**	29,33
9	24274	24151	0,01	23168,00	35,52	22253	0,08	20225,00	62,53	**24274**	0,00	**24094,72**	22,76
10	23551	**23551**	0,00	23437,11	9,32	20957	0,11	18947,67	44,78	23534	0,00	**23468,95**	7,64
11	23534	23056	0,02	22703,41	18,94	21045	0,11	20118,00	32,94	**23534**	0,00	**23474,49**	8,11
12	23991	**23991**	0,00	23078,55	7,44	20783	0,13	18549,33	99,75	**23991**	0,00	**23489,26**	4,66
13	23064	22879	0,01	**22659,47**	8,32	21375	0,07	19545,63	16,81	**23064**	0,00	22601,13	11,47
14	22801	22750	0,00	**22044,67**	8,80	20005	0,12	19739,55	25,59	**22801**	0,00	21780,57	15,45
15	22131	**22131**	0,00	**22016,00**	7,06	19884	0,10	19544,00	10,95	22124	0,00	22063,88	8,02
16	22772	22582	0,01	21970,13	6,89	21119	0,07	20076,74	28,86	**22772**	0,00	**22260,19**	5,09
17	22751	22148	0,03	21916,59	8,91	20073	0,12	19257,41	40,56	**22751**	0,00	**22018,30**	6,13

Tablo 3'teki sonuçlar incelendiğinde, 17 test problemi için KKO algoritmasıyla toplamda 3 problemde, YAK algoritmasıyla 10 problemde ve fYAK algoritmasıyla 15 problemde optimal sonuçlara ulaşabildiği görülmektedir. Tablo, sonuç ortalamaları açısından değerlendirildiğinde ise, KKO 3, YAK 6 ve fYAK 12 problemde en başarılı ortalama sonuca ulaşmıştır. fYAK algoritmasıyla elde edilen sonuçların standart sapma değerlerinin düşük seviyelerde kalması, algoritmanın kararlı yapıda başarılı çözüm üretme davranışında olduğu gözlemlenebilmektedir.

Tablo 4'te ise fYAK algoritmasıyla 8-17 numaralı test problemlerinden elde edilen en iyi sonuçlar, [30] ve [31]'deki çalışmalarda paylaşılan, bu örnekler için elde edilmiş en iyi sonuçlarla karşılaştırılmaktadır. Çalışmalarda, en iyi sonucu paylaşılan algoritmalar şunlardır: Genetik algoritma (GA), filtre ve fan sezgiseli (F & F), iki kendinden uyarlamalı kontrol ve onarım operatörü tabanlı parçacık sürü optimizasyonu (SACRO-BPSO), melez kuantum parçacık sürü optimizasyonu (QPSO), iki aşamalı tabu-evrimsel algoritma (TPTEA), modifiyeli çok katlı optimizasyon (MMVO), yerel topolojik hızlandırılmış ikili parçacık sürü algoritması (BAPSAL), kuantum tabanlı guguk kuşu algoritması (QICSA), ceza kullanan parçacık sürü optimizasyonu (PSO-P), yeni melez ikili parçacık sürü optimizasyonu (NHBPSO), hızlandırılmış ikili parçacık sürü algoritması (BAPSA).

Tablo 4. En iyi fYAK sonuçlarının literatürdeki en iyi sonuçlarla karşılaştırılması

No	MMVO	BAPSAL	QICSA	PSO-P	NHBPSO	BAPSA	GA	F & F	SACRO-BPSO(1)	SACRO-BPSO(2)	QPSO	TPTEA	fYAK
8	24192	24253	23416	22525	23936	23987	**24381**	**24381**	24343	24343	**24381**	**24381**	**24381**
9	**24274**	24106	22880	22244	23827	23752	**24274**	**24274**	**24274**	**24274**	**24274**	**24274**	**24274**
10	23538	23468	22525	21822	23234	2342	**23551**	**23551**	23538	23538	**23551**	**23551**	23534
11	23288	23153	22727	22057	23032	23189	**23534**	**23534**	23527	23527	**23534**	**23534**	**23534**
12	23947	23855	22854	22167	23652	23504	**23991**	**23991**	**23991**	23966	**23991**	**23991**	**23991**
13	22805	22816	21796	20895	22687	22712	**23064**	**23064**	**23064**	**23064**	**23064**	**23064**	**23064**
14	22630	22309	21348	20663	22256	22305	**22801**	**22801**	22739	22750	**22801**	**22801**	**22801**
15	**22131**	21785	20961	20058	21744	21725	**22131**	**22131**	**22131**	**22131**	**22131**	**22131**	22124
16	22347	22419	21377	20908	22341	22191	**22772**	**22772**	**22772**	22717	**22772**	**22772**	**22772**
17	22417	2242	21251	20488	22204	22292	**22751**	**22751**	**22751**	**22751**	**22751**	**22751**	**22751**

Tablo 4'teki sonuçlar incelendiğinde, GA, F & F, QPSO ve TPTEA algoritmalarının karşılaştırılan 10 test probleminin tamamında en iyi sonuca ulaştığı gözlemlenmektedir. fYAK algoritması ise, 8 problemde en iyi sonuca ulaşarak, diğer 8 algoritmadan daha başarılı sonuçlar üretmiştir. fYAK, 10 numaralı problemde optimum sonucun yaklaşık % 0.0007, 15 numaralı problemde ise optimum sonucun yaklaşık % 0.0003 gerisinde kalmıştır.

6. SONUÇ

Sezgisel algoritmalar, nümerik optimizasyon problemleri için, belirlenen sınır aralıklarında rassal sayılar üreterek çözüm ararken, ayrık yapılı problemlerde rastgele nesneler seçerek çözüm oluştururlar. 0-1 yapılı problemler ise sezgisel algoritmalara farklı yöntemler eklenerek çözülebilen problemlerdir. Ayrıca bu problemlere eklenen her bir kısıt, probleme yeni bir boyut kazandırmakta ve çözümü güçleştirmektedir. Çalışma kapsamında, nümerik problemler için geliştirilmiş ve arama uzayının farklı

alanlarına dağılabilen YAK algoritması ile ayrık problemlerde başarılı bir sömürü yeteneğine sahip KKO yöntemi birleştirilerek geliştirilen fYAK yöntemi ile 0-1 çok boyutlu optimizasyon problemleri için çözümler aranmıştır. Farklı hacimlerdeki benchmarking problemleri üzerinde test edilen algoritmanın, karşılaştırma sonuçları itibariyle, bu tür problemlere uygulanabildiği gözlemlenmiş ve YAK algoritmasının sömürü yeteneğini güçlendirdiği kanıtlanmıştır.

KAYNAKÇA

[1] S. I. Gass and A. A. Assad, An Annotated Timeline of Operations Research. Boston: Kluwer Academic Publishers, 2004.

[2] J.-W. Lee, D.-H. Lee, and J.-J. Lee, "Global path planning using improved ant colony optimization algorithm through bilateral cooperative exploration," in 5th IEEE International Conference on Digital Ecosystems and Technologies (IEEE DEST 2011), 2011, vol. 5, no. June, pp. 109–113.

[3] A. Aljanaby, K. R. Ku Mahamud, and N. Norwawi, "Interacted Multiple Ant Colonies Optimization Framework: an Experimental Study of the Evaluation and the Exploration Techniques to Control the Search Stagnation," Int. J. Adv. Comput. Technol., vol. 2, no. 1, pp. 78–85, Mar. 2010.

[4] M. Ehrgott and X. Gandibleux, "A survey and annotated bibliography of multiobjective combinatorial optimization," OR-Spektrum, vol. 22, no. 4, pp. 425–460, 2000.

[5] P. C. Chu and J. E. Beasley, "A Genetic Algorithm for the Multidimensional Knapsack Problem," J. Heuristics, vol. 4, pp. 63–86, 1998.

[6] R. Parra-Hernandez and N. J. Dimopoulos, "A New Heuristic for Solving the Multichoice Multidimensional Knapsack Problem," IEEE Trans. Syst. Man, Cybern. - Part A Syst. Humans, vol. 35, no. 5, pp. 708–717, Sep. 2005.

[7] A. Sbihi, "A best first search exact algorithm for the Multiple-choice Multidimensional Knapsack Problem," J. Comb. Optim., vol. 13, no. 4, pp. 337–351, Apr. 2007.

[8] M. E. Captivo, J. Climaco, J. Figueira, E. Martins, and J. L. Santos, "Solving bicriteria 0 – 1 knapsack problems using a labeling algorithm," Comput. Oper. Res., vol. 30, pp. 1865–1886, 2003.

[9] M. Laumanns, L. Thiele, and E. Zitzler, "An efficient , adaptive parameter variation scheme for metaheuristics based on the epsilon-constraint method," Eur. J. Oper. Res., vol. 169, pp. 932–942, 2006.

[10] L. Ke, Z. Feng, Z. Ren, and X. Wei, "An ant colony optimization approach for the multidimensional knapsack problem," J. Heuristics, vol. 16, no. 1, pp. 65–83, Feb. 2010.

[11] M. Kong, P. Tian, and Y. Kao, "A new ant colony optimization algorithm for the multidimensional Knapsack problem," Comput. Oper. Res., vol. 35, no. 8, pp. 2672–2683, Aug. 2008.

[12] S. Hanafi and C. Wilbaut, "Scatter Search for the 0–1 Multidimensional Knapsack Problem," J. Math. Model. Algorithms, vol. 7, no. 2, pp. 143–159, Jun. 2008.

[13] L. Wang, X. Zheng, and S. Wang, "Knowledge-Based Systems A novel binary fruit fly optimization algorithm for solving the multidimensional knapsack problem," Knowledge-Based Syst., vol. 48, pp. 17–23, 2013.

[14] V. Tongur and E. Ülker, "Migrating Birds Optimization (MBO) algorithm to solve 0-1 multidimensional knapsack problem," in 2017 International Conference on Computer Science and Engineering (UBMK), 2017, pp. 786–789.

[15] K. Klamroth and M. M. Wiecek, "Dynamic Programming Approaches to the Multiple Criteria Knapsack Problem," Nav. Res. Logist., vol. 47, pp. 57–76, 2000.

[16] Ö. Karsu, "Eşitlikçi Çok Amaçlı Sırt Çantası Problemi," Gazi ÜniversitesiFen Bilim. Derg., vol. 6, no. 2, pp. 358–373, 2018.

[17] D. Karaboga, B. Gorkemli, and C. Ozturk, "A comprehensive survey: artificial bee colony (ABC) algorithm and applications," Artif. Intell. Rev., vol. 42, no. 1, pp. 21–57, 2014.

[18] S. Neelima, N. Satyanarayana, and P. K. Murthy, "A Comprehensive Survey on Variants in Artificial Bee Colony (ABC)," Int. J. Comput. Sci. Inf. Technol., vol. 7, no. 4, pp. 1684–1689, 2016.

[19] B. Akay and D. Karaboga, "A survey on the applications of artificial bee colony in signal, image, and video processing," Signal, Image Video Process., vol. 9, no. 4, pp. 967–990, 2015.

[20] W. Gong, "ABC-ACO for Perishable Food Vehicle Routing Problem with Time Windows," 2010 Int. Conf. Comput. Inf. Sci., pp. 1261–1264, 2010.

[21] M. Kefayat, A. Lashkar Ara, and S. A. Nabavi Niaki, "A hybrid of ant colony optimization and artificial bee colony algorithm for probabilistic optimal placement and sizing of distributed energy resources," Energy Convers. Manag., vol. 92, pp. 149–161, Mar. 2015.

[22] P. Shunmugapriya and S. Kanmani, "A hybrid algorithm using ant and bee colony optimization for feature selection and classification (AC-ABC Hybrid)," Swarm Evol. Comput., vol. 36, pp. 27–36, Oct. 2017.

[23] H. Wei, J. Ji, Y. Qin, Y. Wang, and C. Liu, "A Novel Artificial Bee Colony Algorithm Based on Attraction Pheromone for the Multidimensional Knapsack Problems," in Artificial Intelligence and Computational Intelligence, no. 1, H. Deng, D. Miao, J. Lei, and F. L. Wang, Eds. Springer Berlin Heidelberg, 2011, pp. 1–10.

[24] J. Ji, H. Wei, C. Liu, and B. Yin, "Artificial Bee Colony Algorithm Merged with Pheromone Communication Mechanism for the 0-1 Multidimensional Knapsack Problem," Math. Probl. Eng., vol. 2013, pp. 1–13, 2013.

[25] D. Ekmekci, "A Pheromonal Artificial Bee Colony -pABC-Algorithm for Optimization Problems," in 2019 16th International Multi-Conference on Systems, Signals & Devices (SSD), 2019, no. M, pp. 452–456.

[26] D. Ekmekci, "A Pheromonal Artificial Bee Colony (pABC) Algorithm for Discrete Optimization Problems," Appl. Artif. Intell., vol. 33, no. 11, pp. 935–950, Sep. 2019.
[27] D. Karaboga, "An Idea Based on Honey Bee Swarm for Numerical Optimization," Kayseri, Turkey, 2005.
[28] M. Dorigo, V. Maniezzo, and A. Colorni, "Positive feedback as a search strategy," Milano, Italy, 1991.
[29] M. Dorigo and C. Blum, "Ant colony optimization theory: A survey," Theor. Comput. Sci., vol. 344, no. 2–3, pp. 243–278, Nov. 2005.
[30] M. Abdel-Basset, D. El-Shahat, H. Faris, and S. Mirjalili, "A binary multi-verse optimizer for 0-1 multidimensional knapsack problems with application in interactive multimedia systems," Comput. Ind. Eng., vol. 132, no. September 2018, pp. 187–206, Jun. 2019.
[31] X. Lai, J.-K. Hao, F. Glover, and Z. Lü, "A two-phase tabu-evolutionary algorithm for the 0–1 multidimensional knapsack problem," Inf. Sci. (Ny)., vol. 436–437, pp. 282–301, Apr. 2018.