

T.C.
ERCIYES ÜNİVERSİTESİ
BİLİMSEL ARAŞTIRMA PROJELERİ
KOORDİNASYON BİRİMİ

**KOMBİNASYONEL YAPAY ARI KOLONİSİ ALGORİTMALARININ
TEST PROBLEMLERİ ÜZERİNDE GERÇEKLEŞTİRİLMESİ**

Proje No: FBA-11-3536

Proje Türü: Normal Araştırma Projesi

SONUÇ RAPORU

Proje Yürütücüsü:

Prof. Dr. Derviş Karaboğa
Mühendislik Fakültesi/Bilgisayar Mühendisliği Bölümü

Araştırmacı:

Beyza Görkemli
Fen Bilimleri Enstitüsü/ Bilgisayar Mühendisliği Bölümü

Haziran 2013

KAYSERİ

İÇİNDEKİLER

	Sayfa No
TEŞEKKÜR	4
ÖZET	5
ABSTRACT	5
1. GİRİŞ	6
2. TEMEL KOMBİNASYONEL OPTİMİZASYON PROBLEMLERİ	7
3. YAPAY ARI KOLONİSİ ALGORİTMASI	9
4. KOMBİNASYONEL ABC ALGORİTMASI	11
5. QUICK ABC ALGORİTMASI	13
6. QUICK CABC ALGORİTMASI	14
7. DENEY ÇALIŞMALARI VE SİMÜLASYON SONUÇLARI	15
8. SONUÇ	16
9. KAYNAKLAR	16

TEŞEKKÜR

Bu çalışma Erciyes Üniversitesi BAP birimi tarafından desteklenmiştir.

Proje numarası: FBA-11-3536

ÖZET

KOMBİNASYONEL YAPAY ARI KOLONİSİ ALGORİTMALARININ TEST PROBLEMLERİ ÜZERİNDE GERÇEKLEŞTİRİLMESİ

Yapay Arı Kolonisi algoritması 2005 yılında literatüre tanıtılmış, günümüze kadar farklı alanlarda, çeşitli sürekli optimizasyon problemlerinde popüler bir şekilde kullanılmış ve bu çalışmalarda oldukça başarılı sonuçlar ortaya konmuştur. Bu proje kapsamında gerçek hayat problemlerinin büyük bir kısmını oluşturan ve çözümü de büyük önem arzeden çeşitli kombinasyonel optimizasyon problemlerinde etkin bir optimizasyon işleminin kısa bir zamanda gerçekleştirilebilmesi için Yapay Arı Kolonisi optimizasyonu tabanlı yeni algoritmalar geliştirilmiştir. Geliştirilen kombinasyonel algoritmaların bazı TSP literatür problemlerinde etkinliği araştırılmış elde edilen sonuçlar literatürdeki farklı yöntemlerle kıyaslanmıştır. Sonuçlar geliştirilen algoritmaların kombinasyonel optimizasyon problemlerinde etkin bir şekilde kullanılabilir nitelikte olduğunu göstermiştir.

ABSTRACT

IMPLAMENTATION OF COMBINATORIAL ARTIFICIAL BEE COLONY ALGORITHMS ON TEST PROBLEMS

Artificial Bee Colony algorithm is a popular swarm intelligence based algorithm that was introduced to literature in 2005. This algorithm has been applied to various continuous optimization problems successfully. In this project, new combinatorial Artificial Bee algorithms were improved to solve different combinatorial optimization problems effectively and fast. These algorithms were tested on some TSP benchmarks and compared with some other methods in the literature. Simulation results showed that, proposed algorithms can be used to solve combinatorial optimization problems effectively.

1. GİRİŞ

Gezgin Satıcı Problemi (Travelling Salesman Problem -TSP), kombinasyonel optimizasyonun en saf haline yönelik bir problemidir. Bu nedenle bu tür optimizasyon için geliştirilen genel yöntemlerin çoğu öncelikle TSP üzerinde test edilerek etkinliği araştırılmaktadır. Bu proje kapsamında da geliştirilen yöntemler TSP için test edilecektir.

TSP NP zor yapıda bir kombinasyonel optimizasyon problemidir [1]. Rego vd. TSP için geliştirilmiş sezgisel yöntemlerle ilgili bir inceleme çalışması yapmışlardır [34]. Bu inceleme çalışmasında, önde gelen TSP sezgiselleri ve bu yöntemlerin başarısını sağlayan özel bileşenler ele alınmıştır ve ayrıca çalışmada çeşitli simetrik ve asimetrik TSP test problemleri için deneysel analiz verilmiştir. Literatürde kabul edilebilir hesaplama zamanları dahilinde daha iyi sonuçlar elde edebilmek için birçok meta sezgisel algoritma bu probleme uygulanmıştır.

Kombinasyonel optimizasyonda TSP üzerinde yapılan çalışmaların bir kısmı karınca kolonisi optimizasyonu algoritması ve bu algoritmanın türevlerini içermektedir: Dorigo ve Gambadella karınca koloni sistemini TSP'ye uygulamışlardır [2], Zhang ve Feng Maks-Min karınca sistemi ile TSP için bir bilimsel çalışma yapmışlardır [3], Ilie ve Badica dağıtık bir mimari üzerinde karınca koloni optimizasyon algoritmalarını için dağıtık bir yaklaşım önermişlerdir [4], Gan, Guo, Chang ve Yi yine TSP için bir karınca koloni optimizasyon algoritması geliştirmişler ve bu çalışmalarında temel karınca koloni optimizasyon algoritmasının durgunluk davranışına ve erken yakınsama problemine çözüm olması için kaşif karakterisitği kullanmışlardır [5], Puris, Bello ve Herrera yaptıkları çalışmada, karınca koloni optimizasyonunda elde edilen çözümlerin kalitesini artırmak için iki aşamalı bir yaklaşım geliştirmişlerdir [6]. Zhou ve Wang yoğunlaştırma ve çeşitlendirme arasında bir denge oluşturabilmek adına karınca koloni optimizasyonuna kendi kendini değerlendirme stratejisini adapte etmişler ve tanımladıkları yeni yaklaşımı TSP'ye uygulamışlardır [35]. Wang, Zhao ve Zhou, yol belirlerken karıncaların sadece feromon bilgisine ve sezgisel bilgiye bakmadıkları, son iterasyondaki çözümü de dikkate aldıkları, hafızalı bir karınca koloni optimizasyonu önermiş ve TSP'de uygulama çalışması yapmışlardır [36]. Wei vd. sürekli araştırma uzayında çalışan bir optimizasyon yöntemi olan kaotik karınca sürüsü (chaotic ant swarm - CAS) algoritmasını bir ayrık optimizasyon problemi olan TSP'ti çözebilmek için modifiye etmişler ve CAS-TSP metodunu üretmişlerdir [37].

Wong, Low, ve Chong TSP için arı kolonisi optimizasyonu (bee colony optimization-BCO) tanımlamışlardır [7]. Daha sonra bu algoritma tarafından üretilen çözümlerin geliştirilebilmesi için BCO algoritmasını 2-opt sezgiseli ile bütünleştirmişlerdir [8].

Bunların dışında da bir çok araştırmacı doğadan esinlenilerek oluşturulmuş farklı yöntemlerle TSP problemine çözüm aramıştır. Bu yöntemlerin içinde ısı işlem [10], tabu araştırması [11, 12], parçacık sürü optimizasyonu [13, 14], self organising map [15], genetik algoritmalar [16-18], MBO (Marriage honey bee optimization) [38] mevcuttur. Bu yöntemlere ek olarak daha iyi sonuçlar elde etmek için bu metotların lokal arama sezgiselleri ile kombinasyonlarından oluşan hibrit algoritmalar oldukça yoğun bir şekilde kullanılmıştır [17, 19, 20]. Lokal arama ile hibrit bir kültürel algoritma Kim ve Cho tarafından TSP'ye uygulanmıştır [9]. Geng vd. TSP'de etkin bir yerel araştırma sağlamak için ısı işlem araştırması ve açgözlü araştırma (greedy search) tekniklerine dayanan bir yöntem önermişlerdir [39].

Chen ve Chien TSP'yi çözmek için genetik algoritma, ısı işlem, karınca koloni sistemi ve parçacık sürü optimizasyonu tekniklerini birlikte kullanarak yeni bir metot sunmuşlardır [40].

Dong, Guo ve Tickle karınca koloni optimizasyonunun yerel minimumlara takılma problemini ortadan kaldırmak ve TSP'ye daha etkin çözümler sunabilmek için Genetik algoritma ve karınca koloni optimizasyonunu birlikte kullanarak hibrit bir yaklaşım önermişlerdir [41]. Marinakis, Marinaki ve Dounias öklid mesafelerinin söz konusu olduğu TSP için bal arıları eşleşme optimizasyonuna (honey bees mating optimization-HBMO) dayalı olan bir hibrit yaklaşım geliştirmişlerdir [42]. Bu çalışmada hibrit yapı oluşturulırken HBMO algoritması, Multiple Phase Neighborhood Search-Greedy Randomized Adaptive Search Procedure (MPNS-GRASP) algoritması ve Expanding Neighborhood Search Strategy kullanılmıştır. Deng vd. TSP'nin çözümü için genetik algoritma, parçacık sürü optimizasyonu ve karınca koloni optimizasyonunu birleştirerek iki aşamalı bir hibrit optimizasyon algoritması oluşturmuşlardır [43]. Liao, Yau ve Chen de TSP'nin çözümüne yönelik olarak iki aşamalı bir evrimsel algoritma geliştirmişlerdir [44]. Bu algoritmada birinci aşamada Fuzzy C-Means kümeleme, kural tabanlı bir rota permütasyonu, rassal bir yer değiştirme stratejisi ve bir küme birleştirme prosedürü kullanılırken ikinci aşamada çalışmada önerilen bir genetik tabanlı parçacık sürü optimizasyonu uygulanmıştır. Robati vd. dengelenmiş bulanık küme teorisinden faydalanarak kombinasyonel optimizasyon problemlerinin çözümünde kullanılabilmesi için parçacık sürü optimizasyonunun yeni bir versiyonunu literatüre tanıtmışlardır [45]. Kombinasyonel optimizasyon problemleri için geliştirilen hibrit metasezgisel yöntemlere dair bir inceleme çalışması Blum vd. tarafından gerçekleştirilmiştir [46].

Ivanko TSP'de verilen bir işin daha düşük boyutlarda alt görevlere ardışık rasyonel ayrışması için bir algoritma önermiştir [47]. Shen ve Zhang lineer programlamadaki gölge fiyat kavramını kullanarak yeni bir evrimsel algoritma ortaya koymuş ve TSP üzerinde test etmişlerdir [48]. Zhang vd. yeni bir ORC-SOM (overall-regional competitive SOM) algoritması geliştirmiş ve öklid mesafesini kullanan simetrik TSP çözümünü gerçekleştirmişlerdir [49]. TSP'yi çözmek için evrimsel hesaplamaya dayalı algoritmalar paralelleştirilerek de kullanılmıştır [50,51].

2011 yılında Karaboga ve Gorkemli (proje ekibi) TSP için kombinasyonel bir yapay arı kolonisi algoritması geliştirmiş ve farklı genetik algoritmalarla, önerdikleri algoritmanın performansını kıyaslamışlardır [54]. Yapılan bu konferans çalışması, elde edilen sonuçların başarısı ve motive edici niteliği nedeniyle projelendirilmiş ve daha geniş veri kümeleri üzerinde test edilmesi, yeni yaklaşımlarla yöntemin daha da geliştirilmesi planlanmıştır.

2. TEMEL KOMBİNASYONEL OPTİMİZASYON PROBLEMLERİ

Kombinasyonel olarak eniyilemesi gerçekleştirilen ve gerçek hayatta farklı birçok uygulaması bulunan temel problemlerin bazıları aşağıda belirtilmiştir:

Gezgin Satıcı Problemi

Gezgin satıcı probleminde temel prensip şu şekildedir; ziyaret edilmesi gereken şehirler bellidir. Satıcı, belirlenen bir başlangıç şehriden itibaren her şehri sadece bir kez ziyaret etmek koşulu ile tüm şehirleri dolaşarak başlangıç şehrine geri döner ve böylece kapalı bir tur oluşturur.

Bu problemde amaç maliyeti minimize etmektir, bu da direkt olarak tur mesafesini, yani satıcının katettiği yolu minimize etmek ile ilişkilendirilmiştir. Örneğin öklid mesafesi dikkate alınarak çözülmeye çalışılan bir gezgin satıcı probleminde i . ve $i+1$. şehirler arasındaki mesafe aşağıdaki denklemle hesaplanabilir:

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

Minimize edilmeye çalışılan toplam tur uzunluğu ise aşağıdaki şekilde hesaplanır (n toplam şehir sayısını göstermektedir.)

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

Montaj Hattı Dengelme Problemi

Montaj hatları oldukça önemli akış tipi üretim sistemlerindendir. Montajı yapılacak parçalar, öncelikli işler gibi üretim kısıtları dikkate alınarak sıralı istasyonlardan geçirilir. İmalatta verimi artırmak adına, üretimde yoğun olarak kullanılan montaj hatlarının dengelenmesi, işletmeler için büyük önem arz etmektedir.

Montaj hatlarının dengelenmesi probleminde montajı tamamlanmış bir ürünün elde edilebilmesi için gerçekleştirilmesi gereken işler, bu işlerin süreleri ve öncelik diyagramları belirlidir. Hattan bir ürünün çıkma süresi yani çevrim süresi belirli ise amaç, istasyon sayısının minimizasyonu olabilir. Ya da istasyon sayısı belli ise bir ürünün elde edilme zamanı (çevrim süresi) minimize edilerek optimizasyon gerçekleştirilmeye çalışılabilir.

Çizelgeleme Problemi

Kısıtlı miktardaki kaynakların görev yahut faaliyetlere dağıtılması/atanmasıdır. Atama işlemi gerçekleştirilirken zaman kavramı da dikkate alınır. Böylece örneğin bir üretim sisteminde, ürünü oluşturan iş parçalarının mevcut makinelerde, ne zaman ve hangi sıra ile gerçekleştirileceği belirlenmiş olur.

En Kısa Yol Problemi

Çizge teorisinde, bir ağda iki düğüm arasındaki en kısa mesafenin bulunması temeline dayanan bir problemidir. Yani belirlenen yoldaki tüm kenarların toplam ağırlığı minimize edilmeye çalışılır.

Araç Rotalama Problemi

Bir ya da birkaç depodan, belirli noktalara ürün dağıtımı yahut toplanması işlemi araç rotalama problemi olarak adlandırılmaktadır. Burada rotalama gerçekleştirilirken, özellikle servis süresi ve araç kapasite kısıtları sağlanmaya çalışılır. Araç rotalama problemlerinin gerçek hayat örnekleri olarak gazete dağıtımı, okul servisleri, çöp toplanması işlemleri de verilebilir.

Araç rotalama probleminde kaynak depodan başlanarak, dağıtım noktalarına belirli bir sıra dahilinde uğranır ve tekrar kaynak depoya geri dönülür. Bu şekilde belirli sayıda araç için rota oluşturulur. Tüm dağıtım noktaları araç rotalarının birinde yer almalıdır ve elbette her araç için toplam dağıtım miktarı aracın kapasitesini aşmamalıdır.

Bu problemde amaç, kısıtlar sağlanırken maliyetleri minimize etmektir. Bu maliyetler, toplam harcama zaman, katedilen mesafe ve kullanılan araç sayısıdır. Bunların yanında yahut bunlardan farklı olarak müşteri memnuyeti ve kar maksimize edilmeye de çalışılabilir.

Ulaştırma Problemi

Ürünlerin, birden fazla sayıda üretim (yükleme) merkezinden, yine birden fazla tüketim (boşaltım) merkezine iletilmeleri ile ilgilenen bir problemidir. Ulaştırma probleminde, hangi üretim noktasından hangi boşaltım merkezine, ne miktarda ürün taşınacağı belirlenmeye çalışılır.

Optimizasyon yapılırken maliyetler minimize edilir. Aynı zamanda üretim ve tüketim noktaları için arz-talep dengeleri de sağlanmaya çalışılır.

Atama Problemi

İşlerin işçilere ya da işlerin makinelere atanması gibi problemleri içerir. Yani belirli sayıda işlem görecektir unsur, yine belirli sayıdaki işlem merkezine dağıtılır.

Maliyetler minimize edilmeye, toplam etkinlik ise maksimize edilmeye çalışılabilir.

Sırt Çantası Problemi

Sırt çantası problemi mevcut ürünlerin, belirli bir kapasiteye sahip bir çantaya yerleştirilmesini konu edinir. Burada çantanın kapasitesinin aşılmaması önemlidir. Yerleştirme yapılırken, kapasiteyi aşmadan, en yüksek derecede karı getirecek parça miktarları seçilmeye çalışılır.

3. YAPAY ARI KOLONİSİ ALGORİTMASI

Yapay Arı Kolonisi (Artificial Bee Colony -ABC) algoritması, çok modlu ve çok boyutlu nümerik optimizasyon problemleri için Karaboğa tarafından 2005 yılında önerilen sürü zekası tabanlı bir algoritmadır [21, 22]. Sürekli optimizasyon problemleri için oldukça iyi sonuçlar vermektedir [23, 24]. ABC optimizasyonu, oldukça yaygın bir şekilde birçok probleme uygulanmış ve araştırmacılar tarafından ABC algoritması ile ilgili olarak, çok değişik alanlarda farklı bilimsel çalışmalar gerçekleştirilmiştir [25-31,53]. Literatürde bazı kesikli problemler için geliştirilmiş olan kesikli ABC yaklaşımları da yer almaktadır [32, 33, 52].

ABC algoritmasında arılar 3 temel görevle özelleştirilmişlerdir. Bunlardan birincisi görevli arılardır. Yuvalardan çıktıklarında akıllarında belirli bir yiyecek kaynağı olan bu arılar, yuvaya döndüklerinde faydalandıkları yiyecek kaynağı ile ilgili dans ederler. Bu dansları izleyerek tüketmek için gidecekleri yiyecek kaynağına karar veren arı grubu ise gözücü arılardır.

Algoritmada gözcü arılar yiyecek kaynaklarının kalite değerlerine göre belirlenen bir olasılık dahilinde yiyecek kaynakları arasından seçim yaparlar. Son farklı arı grubu ise kaşif arılardır. Bu arılar diğer arıların bilgi ve tecrübelerine bakmaksızın yeni yiyecek kaynakları bulur ve tüketmeye başlarlar, çalışmaya görevli arı olarak devam ederler. Bu sayede bir yandan bilinen kaynaklar tüketilirken bir yandan da yeni kaynaklardan koloninin haberinin olması sağlanır. Araştırmanın başında kolonideki tüm arılar kaşif durumundadırlar ve rastgele çözümlerle yani yiyecek kaynaklarıyla araştırmaya başlanır. İleriki çevrimlerde ise yiyecek kaynakları tükendikçe, ilgili yiyecek kaynağını tüketen görevli arı kaşif konumuna geçer. Algoritmada yiyecek kaynaklarının tükenme durumunu kontrol edebilmek için "limit" adında bir parametre kullanılmaktadır. Herbir çözüm için çözümü iyileştirme denemeleri sayısı tutulur ve her çevrimde, maksimum deneme sayısına sahip yiyecek kaynağının bu deneme değerinin, önceden belirlenmiş *limit* değerini aşıp aşmadığına bakılır. Aştıysa, o çözüm yani yiyecek kaynağı terk edilir ve araştırma işlemine rastgele belirlenmiş yeni bir çözümle devam edilir. ABC algoritması için temel algoritmik yapı aşağıdaki gibidir:

Başlangıç aşaması

REPEAT

 Görevli arı fazı

 İzci arı fazı

 Kaşif arı fazı

 Şu ana kadar bulunan en iyi çözümü hafızaya al.

UNTIL (durdurma kriteri sağlanana kadar)

Başlangıç aşamasında l_i ve u_i ile belirlenen sınırlar dahilinde Denklem 3 ile bütün yiyecek kaynakları rassal olarak belirlenir. $x_{m,i}$, denklemde m . çözümün i . boyutuna ait değeri göstermektedir.

$$x_{m,i} = l_i + rand(0,1) * (u_i - l_i) \quad (3)$$

Daha sonra yiyecek kaynakları, yani rassal olarak üretilen çözümler değerlendirilir ve amaç fonksiyonu değerleri hesaplanır.

Görevli arı aşmasında ise, yuvadan akıllarındaki yiyecek kaynağının pozisyon bilgisi ile ayrılan arılar, yiyecek kaynağına ulaştıklarında bu kaynağın komşuluğunda bir araştırma yaparlar. Komşu çözüm üretirken kullandıkları bağıntı Denklem 4 ile belirtilmiştir:

$$v_{m,i} = x_{m,i} + \phi_{m,i}(x_{m,i} - x_{k,i}) \quad (4)$$

x_k , rastgele seçilmiş bir yiyecek kaynağını temsil ederken i de rastgele seçilmiş bir boyutu göstermektedir. $\phi_{m,i}$ ise, $[-1,1]$ aralığında üretilmiş rassal bir sayıdır. Aday çözüm v_m üretildikten sonra eğer ürettikleri komşu çözüm eskisinden daha iyi ise kaynak olarak onu benimser ve tüketmeye başlarlar.

Denklem 5 ile yiyecek kaynaklarının $f(x_m)$ amaç fonksiyon değerlerinden $fit(x_m)$ uygunluk değerleri hesaplanmaktadır.

$$fit(x_m) = \begin{cases} \frac{1}{1 + f(x_m)} & \text{if } f(x_m) \geq 0 \\ 1 + abs(f(x_m)) & \text{if } f(x_m) < 0 \end{cases} \quad (5)$$

Görevli arılar yuvaya döndüklerinde herbiri kedi yiyecek kaynakları hakkında bilgi veren danslar yapmakta, bu dansları izleyen gözcü arılar da gidecekleri kaynakları kalite değerlerine göre bir olasılık dahilinde seçmektedir. Bunun için aşağıdaki olasılık formülü kullanılabilir:

$$p_m = \frac{fit(x_m)}{\sum_{m=1}^{SN} fit(x_m)} \quad (6)$$

Gözcü arılar da seçtikleri yiyecek kaynaklarına ulaştıklarında, görevli arılarla aynı yöntemle komşu kaynaklar üretmekte ve yaptıkları aç gözlü seleksiyonla yollarına hangi kaynakla devam edeceklerini belirlemektedirler.

Kaşif fazında ise, her iterasyon sonunda her bir yiyecek kaynağının geliştirilememe sayısına bakılmakta, en yüksek değere sahip kaynağın bu geliştirilememe sayısı *limit* paramtresiyle karşılaştırılmakta ve eğer *limit*'ten daha büyükse, yiyecek kaynağı terkedilmekte, ilgili arı ise araştırma işlemine Denklem 3 ile üretilen rassal bir çözümle kaşif olarak devam etmektedir.

4. KOMBİNASYONEL ABC ALGORİTMASI

Kombinasyonel ABC (Combinatorial ABC -CABC) algoritması 2011 yılında proje ekibi tarafından geliştirilmiş ve bu çalışma dahilinde projelendirilmiş temel yöntemdir [54]. Bu algortmada, standart ABC algoritması ile aynı temel adımlar kullanılmaktadır. Burada TSP problemi üzerinden gidildiği için amaç toplam kapalı tur uzunluğunu minimize etmektir. Bu nedenle her bir çözümün kalitesini gösteren *fit_i* değeri aşağıdaki denklemle hesaplanmıştır:

$$fit_i = \frac{1}{1 + f(x_i)} \quad (7)$$

Burada $f(x_i)$, x_i çözümünün tur mesafesini göstermektedir.

CABC algoritmasının adımları aşağıda belirtildiği gibidir:

1. Parametrelere başlangıç değerlerini ata (koloni büyüklüğü *cs*, maksimum çevrim sayısı *MaxNumber*).
2. Yiyecek kaynaklarına (x_i) başlangıç değerlerini ata. $i = 1, 2, \dots, cs$.
3. Çözümleri değerlendir.
4. En iyi çözümü hafızaya al.
5. $t=0$
6. Repeat
 - *Görevli arı aşaması:* herbir görevli arı için;
 - x_i komşuluğunda yeni bir aday çözüm v_i üret ve üretilen çözümü değerlendir.
 - x_i ve v_i arasında aç gözlü seleksiyon uygula.
 - x_i çözümlerinin kalite değerlerine göre aşağıdaki bağıntıyı kullanarak P_i hesapla.

$$P_i = \frac{0.9 \times \tilde{fit}_i}{\tilde{fit}_{best}} + 0.1 \quad (8)$$

- Gözcü arı aşaması: Herbir gözcü arı için;
 - x_i komşuluğunda yeni bir aday çözüm v_i üret ve üretilen çözümü değerlendir.
 - x_i ve v_i arasında aç gözlü seleksiyon uygula.
- Elde edilen en iyi çözümü hafızaya al.
- Tüketilen kaynak olup olmadığına bak, varsa, kaşif için yeni bir çözüm üret ve tükenen çözümle yer değiştir.
- $t = t + 1$

7. Until ($t = MaxNumber$)

Albayrak ve Allahverdi yaptıkları çalışmada genetik algoritmanın TSP'deki başarımını artırmaya yönelik olarak, bir mutasyon operatörü geliştirmişlerdir [18]. Geliştirdikleri GSTM operatörünü, TSP literatür problemleri üzerinde farklı 7 mutasyon operatörü ile kıyaslamışlardır. Elde ettikleri hata değerleri GSTM'nin diğer operatörlere nazaran daha iyi sonuçlar ürettiğini göstermektedir. Bu nedenle, görevli ve gözcü arıların komşuluk araştırma mekanizmaları için CABC algoritmasına GSTM operatörü adapte edilmiştir. Bu mekanizmanın detayları aşağıda verildiği gibidir:

1. Populasyondan rassal olarak bir çözüm x_k seç. ($k \neq i$)
2. Yine rassal olarak bir şehir $x_i[j]$ seç.
3. Araştırma yönünü belirleyen parametre Φ için rastgele bir değer seç. ($\Phi = \{-1, 1\}$)
4. If ($\Phi = -1$) then
 - $x_i[j]$ nin önceki şehri olarak $x_k[j]$ nin önceki şehrini ata.
 - Else
 - $x_i[j]$ nin sonraki şehri olarak $x_k[j]$ nin sonraki şehrini ata.
- End if // Bu işlem sonucunda yeni bir kapalı tur $T^{\#}$ ortaya çıkar.
5. Yapılan yeni bağlantı sonucunda da açık bir alt tur T^* oluşacaktır. Bu alt turun ilk şehri R_1 , son şehri ise R_2 olarak belirlenir.
6. If ($random \leq P_{RC}$) then
 - T^* , en az genişlemeyi sağlayacak şekilde $T^{\#}$ 'e eklenir.
 - Else
 - If ($random \leq P_{CP}$)
 - R_1 pozisyonundan başlanarak, T^* 'in her bir şehri T 'ye P_L ihtimaline göre rolling ya da mixing yapılarak yerleştirilir.
 - Else
 - R_1 ve R_2 noktaları için komşu listelerinden rastgele birer komşu seç. (NL_{R1} and NL_{R2})
 - Maksimum kazanç sağlayacak şekilde, NL_{R1} veya NL_{R2} noktalarını ters çevir, öyle ki, bu noktalar R_1 veya R_2 noktalarına komşu alsun.
 - Ters çevirme işlemi gerçekleşmediyse tekrarla.
- End if
- End if

Burada, $x_i[j]$, i çözümünde j şehrinin pozisyonunu göstermektedir. $random$ ise (0,1) aralığında üretilen rastgele bir sayıdır. T , orjinal turu göstermektedir.

GSTM parametreleri: P_{RC} yeniden bağlantı ihtimali, P_{CP} düzeltme ve bozma ihtimali, P_L doğrusallık ihtimalidir. Minimum alt tur uzunluğu L_{MIN} , maksimum tur uzunluğu ise L_{MAX} , komşuluk listesi boyutu ise NL_{MAX} tır.

R noktasının kazancı G_R aşağıdaki denklem ile hesaplanmaktadır:

$$G_R = d[T[R], T[R+1]] + d[T[NL_R], T[NL_R+1]] - (d[T[R], T[NL_R]] + d[T[R+1], T[NL_R+1]]) \quad (9)$$

5. QUICK ABC ALGORİTMASI

Proje dahilinde, görevli arıların bazı davranışları daha detaylı olarak modellenerek, kombinasyonel problemlerde de etkin bir yakınsama gerçekleştireceği düşünülen yeni bir algoritma geliştirilmiş ve yerel araştırma kabiliyetinin iyi olması ve hızlı yakınsaması nedeniyle bu algoritmaya quick ABC (qABC) algoritması adı verilmiştir [55].

Gerçek arı kolonilerinde, görevli bir arı daha önce ziyaret ettiği yiyecek kaynağını tüketirken, gözcü arılar görevli arıların danslarına bakarak kendilerine *bir yiyecek kaynağı bölgesi* seçerler. Bu alana ulaştıkları zaman ise bölgede yer alan yiyecek kaynaklarını inceleyerek en uygun olanı tercih eder ve onu tüketmeye başlarlar. Yani, gözcü arılar yiyecek kaynaklarını seçerken görevli arılardan farklı bir yol izlerler. Ancak standart ABC algoritmasında bu detay dikkate alınmamış ve görevli arılar ve gözcü arılar için aday çözümün belirlenmesi aynı formülle gerçekleştirilmiştir. qABC algoritmasında, gözcü arıların farklı davranışlarını modellemek için aşağıdaki formül kullanılmıştır:

$$v_{N_m,j}^{best} = x_{N_m,j}^{best} + \phi_{m,j} (x_{N_m,j}^{best} - x_{k,j}) \quad (10)$$

Bu formülde, $x_{N_m}^{best}$, x_m 'in kendisi ve diğer komşu çözümleri (N_m) arasındaki en iyi çözümü göstermektedir. Burada, x_m 'in komşuluğuna karar verebilmek için bir benzerlik ölçütünün kullanılması gerekmektedir. Bu anlamda uygun bir komşuluk tanımlayabilmek için farklı yaklaşımlar kullanılabilir. **Bu yeni tanımlanan formülle belirlenen yaklaşım kullanılarak, kombinasyonel ve ikili problemler de qABC algoritması ile optimize edilebilir.** Örnek olarak nümerik optimizasyon problemleri için komşuluklar belirlenirken x_m ve diğer çözümler arasındaki ortalama öklid mesafesi dikkate alınarak bir benzerlik ölçütü elde edilebilir. $d(m,i)$, x_m ve x_i arasındaki öklid mesafesini göstermek üzere, x_m için ortalama öklid mesafesi md_m aşağıdaki gibi hesaplanır.

$$md_m = \frac{\sum_{i=1}^{SN} d(m,i)}{SN - 1} \quad (11)$$

Eğer bir çözümün x_m 'e olan öklid mesafesi md_m 'den küçük ise, o çözüm x_m 'in bir komşusu olarak değerlendirilir. Yani bir gözcü arı, görevli arıların danslarını izledikten sonra onlardan etkilenecek x_m yiyecek kaynağının merkezlik ettiği bir bölge seçer. İlgili bölgeye (N_m) ulaştıktan sonra ise bölge içindeki tüm yiyecek kaynaklarına bakar ve geliştirmek için en

iyisini ($x_{N_m}^{best}$) seçer. x_m de dahil olmak üzere N_m içerisinde S adet çözüm mevcutsa, eniyi çözüm aşağıdaki şekilde belirlenmektedir:

$$fit(x_{N_m}^{best}) = \max(fit(x_{N_m}^1), fit(x_{N_m}^2), \dots, fit(x_{N_m}^S)) \quad (12)$$

Bir çözümün x_m 'in bir komşusu olup olmadığını belirlemek için daha genel ve esnek bir tanımlama aşağıdaki şekilde gerçekleştirilebilir:

$$\begin{aligned} & \text{if } d(m,i) \leq r * md_m \text{ then } x_i \text{ is the neighbour of } x_m, \\ & \text{else not} \end{aligned} \quad (13)$$

Bu yeni tanımlama ile standart ABC parametrelerine yeni bir parametre olan r eklenmiştir. Yapılan tanımda r parametresi, komşuluk yarıçapı olarak yer almaktadır ve $r \geq 0$ olarak kullanılmalıdır.

$r = 0$ olduğu durumda, $x_{N_m}^{best}$ çözümü x_m olacağı için Denklem 10 da standart ABC algoritmasında kullanılan aday çözüm üretme denklemi ile aynı şekilde çalışacaktır. r değeri arttıkça x_m 'in komşuluğu genişleyecek, azaldıkça ise x_m 'in komşuluğu daralacaktır.

6. QUICK CABC ALGORİTMASI

CABC ve qABC birlikte kullanılarak, quick CABC (qCABC) algoritması geliştirilmiştir. Burada temel kritik nokta, CABC'nin qABC ile yeniden yapılandırılabilmesi için TSP'ye uygun bir benzerlik ölçütünün belirlenmesidir. Nümerik problemler için önerilen öklid mesafesi yerine kombinasyonel optimizasyon için kullanılmak üzere, bu benzerlik ölçütü şu şekilde oluşturulmuştur:

x_m ve x_i arasında bir mesafe tanımı yapılmıştır. Bu tanım yapılırken, kapalı turlarda yer alan, iki şehir arasındaki her bir bağlantının farklı olup olmadığına ilişkin bir yaklaşım önerilmiştir. Buna göre iki çözüm için kapalı tur içerisinde yer alan bağlantılarda benzerliği ortaya koyan bir mesafe vektörü dv_{mi} tutulur:

$$\begin{aligned} dv_{mi}(j) &= 0, \quad \text{if } x_i[j]_{afterCity} = x_m[j]_{afterCity} \\ dv_{mi}(j) &= 1, \quad \text{else not} \end{aligned} \quad (14)$$

n şehir sayısını göstermek üzere, dv_{mi} vektörü kullanılarak $d(m,i)$ mesafesi hesaplanır:

$$d(m,i) = \sum_{j=1}^{n-1} dv_{mi}(j) \quad (15)$$

qCABC algoritması elde edilirken, iki çözüm arasındaki mesafe, öklid bağıntısı yerine, (14) ve (15) bağıntıları aracılığı ile belirlenmiştir. Geri kalan kısımlar için temel qABC bakış açısı aynen CABC'ye uygulanmıştır.

7. DENEY ÇALIŞMALARI VE SİMÜLASYON SONUÇLARI

CABC ve qCABC yöntemleri TSP üzerinde test edilmiştir. Bunun için literatürden 14 farklı problem belirlenmiş ve Tablo 1'de verilen parametre seti ile 10 bağımsız koşma alınmıştır. Her bir deneyin sonucunun istatistiksel analizi için ortalama, standart sapma, en iyi, en kötü, ortalama hata, en iyinin hatası, en kötünün hatası tablolarda belirtilmiştir. Bu kapsamda Tablo 2, $Ldiv=1$ olarak belirlenmiş CABC algoritması; Tablo 3, $Ldiv=2$ olarak belirlenmiş CABC algoritması; Tablo 4, $Ldiv=3$ olarak belirlenmiş CABC algoritması; Tablo 5, $Ldiv=4$ olarak belirlenmiş CABC algoritması; Tablo 6, $Ldiv=1$ olarak belirlenmiş qCABC algoritması; Tablo 7, $Ldiv=2$ olarak belirlenmiş qCABC algoritması; Tablo 8, $Ldiv=3$ olarak belirlenmiş qCABC algoritması; Tablo 9, $Ldiv=4$ olarak belirlenmiş qCABC algoritması ile elde edilen sonuçları göstermektedir.

$$Limit = \frac{CS * D}{Ldiv} \quad (16)$$

Burada CS koloni büyüklüğünü, D ise problemin boyutunu göstermektedir.

Daha önce de belirtildiği gibi, Albayrak ve Allahverdi'nin bir çalışmalarında genetik algoritmanın TSP'deki başarımını artırmaya yönelik bir mutasyon operatörü geliştirilmiştir [18]. Aynı çalışmalarında geliştirdikleri GSTM operatörü, TSP literatür problemleri üzerinde farklı 7 mutasyon operatörü ile kıyaslanmıştır. CABC ve qCABC algoritmalarının performansları da bu yöntemlerle mukayese edilmek istendiğinden değerlendirme sayısı ve diğer ortak parametreler [18] ile aynı alınmıştır. Başlangıç çözümleri üretilirken tüm algoritmalarda en yakın komşu tur oluşturma sezgiseli kullanılmıştır.

[18] çalışmasında GSTM operatörü parametreleri için önerilen aşağıdaki değerler aynen alınmıştır.

$P_{RC}=0.5$, $P_{CP}=0.8$, $P_L=0.2$, $L_{MIN}=2$, $NL_{MAX}=5$.

CABC ve qCABC algoritmalarının kullandıkları araştırma mekanizmasında $n/2$ değerinin $Int(\sqrt{n})$ değerinden daha uygun olduğu düşünüldüğü için $L_{MAX}=n/2$ olarak alınmıştır. Burada n toplam şehir sayısıdır.

Tablo 1. Parametre değerleri

Parametre	CABC	qCABC
Koloni büyüklüğü	40	40
Çevrim Sayısı	20000	20000
$Ldiv$	1, 2, 3, 4	1, 2, 3, 4
r	-	1

Yapılan deneylerde en iyinin hatası üzerinden elde edilen sonuçlara göre $Ldiv$ parametresinin rolü CABC için Tablo 10'da, qCABC içinse Tablo 11'de verilmiştir.

52, 200, 442 ve 1577 şehirli 4 problem için (Berlin52, KroA200, Pcb442, Fl1577) sırasıyla Şekil 1-4'te CABC ve qABC algoritmalarının yakınsama grafikleri çizilmiştir. Bu grafikler çizilirken her bir çevrimdeki optimum değer 10 koşma için ortalama değeri baz alınmıştır.

Grafikler incelendiğinde qCABC'nin CABC'ye göre oldukça hızlı yakınsama sağladığı görülmektedir.

CABC ve qCABC sonuçlarında yukarıda sonuçları verilen *Ldiv* parametresine yönelik deneyler dikkate alınarak ortalama hatanın en küçük olduğu *Ldiv* değeri problem için uygun parametre olarak kabul edilmiş ve ona dair sonuçlar Tablo 12'de [18] çalışmasındaki farklı genetik algoritma türlerinin sonuçları ile kıyaslanmıştır.

Tabloda en başarılı sonuçlar koyu renkle vurgulanmıştır. Bu durumda tablo incelendiğinde CABC ve özellikle qCABC'nin farklı genetik algoritma çeşitlerine göre oldukça başarılı sonuçlar verdiği görülmektedir.

8. SONUÇ

Kesikli optimizasyon problemleri için ABC optimizasyonuna dayalı iki farklı yöntem geliştirilmiştir. Bu yöntemler CABC ve qCABC olarak belirtilmiştir. Bu yöntemlerin test problemleri üzerindeki performanslarının analizi gerçekleştirilmiş ve elde edilen sonuçlar literatürdeki başka yöntemlerle kıyaslanmıştır. Yapılan deneyler geliştirilen ABC tabanlı yöntemlerin kombinasyonel optimizasyon problemlerine oldukça etkin sonuçlar üretebilecek kabiliyette olduğunu göstermiştir.

KAYNAKLAR

- [1] G. Laporte, "The traveling salesman problem: An overview of exact and approximate algorithms," *European Journal of Operational Research*, vol. 59, no. 2, pp. 231-247, 1992.
- [2] M. Dorigo, and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 53-66, April 1997.
- [3] Z. Zhang, and Z. Feng, "A novel max-min ant system algorithm for traveling salesman problem," in *Proceedings of Intelligent Computing and Intelligent Systems (ICIS 2009)*, 2009, pp. 508-511.
- [4] S. Ilie, and C. Badica, "Effectiveness of solving traveling salesman problem using ant colony optimization on distributed multi-agent middleware," in *Proceedings of the International Multiconference on Computer Science and Information Technology*, 2010, pp.197-203.
- [5] R. Gan, Q. Guo, H. Chang, and Y. Yi, "Improved ant colony optimization algorithm for the traveling salesman problems," *Journal of Systems Engineering and Electronics*, vol. 21, no. 2, pp. 329-333, April 2010.
- [6] A. Puris, R. Bello, and F. Herrera, "Analysis of the efficacy of a two-stage methodology for ant colony optimization: case of study with TSP and QAP," *Expert Systems with Applications*, vol. 37, no. 7, pp. 5443-5453, July 2010.

- [7] L. P. Wong, M. Y. H. Low, and C. S. Chong, "A bee colony optimization algorithm for traveling salesman problem," in Proceedings of Second Asia International Conference on Modelling & Simulation (AMS 2008), 2008, pp. 818-823.
- [8] L. P. Wong, M. Y. H. Low, and C. S. Chong, "Bee colony optimization with local search for traveling salesman problem," in Proceedings of Industrial Informatics (INDIN 2008), 2008, pp. 1019-1025.
- [9] Y. Kim, and S. B. Cho, "A hybrid cultural algorithm with local search for traveling salesman problem," in Proceedings of Computational Intelligence in Robotics and Automation (CIRA 2009), 2009, pp. 188-192.
- [10] J.W. Pepper, B.L. Golden and E.A. Wasil, "Solving the traveling salesman problem with annealing-based heuristics: a computational study," IEEE Transactions on Systems, Man and Cybernetics-Part A, vol.32, pp.72-77, 2002.
- [11] Y. He, Y. Qiu, G. Liu and K. Lei, "A parallel adaptive tabu search approach for traveling salesman problems," in Proceedings of IEEE International Conference on Natural Language Processing and Knowledge Engineering, 2005, pp.796-801.
- [12] N. Yang, P. Li and B. Mei, "An angle-based crossover tabu search for the traveling salesman problem," in Proceedings of International Conference on Natural Computation, 2007, pp.512-516.
- [13] W.L. Zhong, J. Zhang and W.N. Chen, "A novel discrete particle swarm optimization to solve traveling salesman problem," in Proceedings of IEEE Congress on Evolutionary Computation, 2007, pp.3283-3287.
- [14] Z. Yuan, L. Yang, Y. Wu, L. Liao and G. Li, "Chaotic particle swarm optimization algorithm for traveling salesman problem," in Proceedings of IEEE International Conference on Automation and Logistics, 2007, pp.1121-1124.
- [15] A. Zhu and S.X. Yang, "An improved self-organizing map approach to traveling salesman problem," in Proceedings of IEEE International Conference on Robotics, Intelligent Systems and Signal Processing, 2003, pp.674-679.
- [16] J.D. Wei and D.T. Lee, "A new approach to the traveling salesman problem using genetic algorithms with priority encoding," in Proceedings of Congress on Evolutionary Computation, 2004, pp.1457-1464.
- [17] W. Xuan and Y. Li, "Solving traveling salesman problem by using a local evolutionary algorithm," in Proceedings of the IEEE International Conference on Granular Computing, 2005, pp.318-321.
- [18] M. Albayrak, and N. Allahverdi, "Development a new mutation operator to solve the traveling salesman problem by aid of genetic algorithms," Expert Systems with Applications, vol. 38, no. 3, pp. 1313-1320, March 2011.
- [19] C.M. White and G.G. Yen, "A hybrid evolutionary algorithm for traveling salesman problem," in Proceedings of the IEEE Congress on Evolutionary Computation, 2004, pp.1473-1478.
- [20] B. Freisleben and P. Merz, "A genetic local search algorithm for solving symmetric and asymmetric traveling salesman problems," in Proceedings of International Conference on Evolutionary Computation, 1996, pp. 616-621.
- [21] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Technical Report-TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.

- [22] D. Karaboga, "Artificial bee colony algorithm," www.scholarpedia.org/article/Artificial_bee_colony_algorithm, Scholarpedia. 5(3):6915, 2010.
- [23] D. Karaboga and B. Basturk, "On the performance of artificial bee colony (ABC) algorithm," *Applied Soft Computing*, vol. 8, no. 1, pp. 687-697, January 2008.
- [24] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) Algorithm," *Journal of Global Optimization*, vol. 39, no. 3, pp. 459-471, November 2007.
- [25] D. Karaboga, C. Ozturk, "A novel clustering approach: Artificial Bee Colony (ABC) algorithm," *Applied Soft Computing*, vol. 11, no.1, pp. 652-657, 2011.
- [26] S. Hemamalini, S.P. Simon, "Artificial bee colony algorithm for economic load dispatch problem with non-smooth cost functions," *Electric Power Components and Systems*, vol. 38, no. 7, pp. 786-803, 2010.
- [27] R. Srinivasa Rao, S.V.L. Narasimham and M. Ramalingaraju, "Optimization of distribution network configuration for loss reduction using artificial bee colony algorithm," *International Journal of Electrical Power and Energy Systems Engineering (IJEPESE)*, vol. 1, no. 2, Spring 2008.
- [28] E. Hetmaniok, D. Slota and A. Zielonka, "Solution of the inverse heat conduction problem by using the ABC algorithm," *7th International Conference on Rough Sets and Current Trends in Computing*, JUN 28-30, 2010 Univ Warsaw, Warsaw, Poland, *Rough Sets and Current Trends in Computing, Lecture Notes in Artificial Intelligence*, vol. 6086, pp. 659-668, 2010.
- [29] D. Jeya Mala, V. Mohan, and M. Kamalapriya, "Automated software test optimisation framework - An artificial bee colony optimisation-based approach," *IET Software*, vol. 4, no. 5, pp. 334-348, 2010.
- [30] D. Karaboga, B. Akay, "PID controller design by using artificial bee colony, harmony search and the bees algorithms," *Proceedings of the Institution of Mechanical Engineers, Part I, Journal of Systems and Control Engineering*, vol. 224, no. 17, pp. 869-883, 2010.
- [31] D. Karaboga, C. Ozturk, "Neural networks training by artificial bee colony algorithm on pattern classification," *Neural Network World*, vol. 19, no. 3, pp. 279-292, 2009.
- [32] Q. K. Pan, M. F. Tasgetiren, P. N. Suganthan and T. J. Chua, "A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem," *Information Sciences*, in press.
- [33] A. Singh, "An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem," *Applied Soft Computing*, vol. 9, no. 2, pp. 625-631, March 2009.
- [34] C. Rego, D. Gamboa, F. Glover and C. Osterman, "Traveling salesman problem heuristics: Leading methods, implementations and latest advances", *European Journal of Operational Research*, vol. 211, no. 3, pp. 427-441, 2011.
- [35] X. F. Zhou and R. L. Wang, "Self-evolving ant colony optimization and its application to traveling salesman problem", *International Journal of Innovative Computing Information and Control*, vol. 8, no. 12, pp. 8311-8321, 2012.
- [36] R. L. Wang, L. Q. Zhao and X. F. Zhou, "Ant colony optimization with memory and its application to traveling salesman problem", *IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences*, vol. E95A, no. 3, pp. 639-645, 2012.
- [37] Z. Wei, F. Ge, Y. Lu, L. Li and Y. Yang, "Chaotic ant swarm for the traveling salesman problem", *Nonlinear Dynamics*, vol. 65, no. 3, pp. 271-281, 2011.

- [38] Y. Celik and E. Ulker, " Marriage honey bee optimization approach to the symmetric traveling salesman problem", *Energy Education Science and Technology Part B-Social and Educational Studies*, vol. 4, no. 2, pp. 595-602, 2012.
- [39] X. Geng, Z. Chen, W. Yang, D. Shi and K. Zhao, " Solving the traveling salesman problem based on an adaptive simulated annealing algorithm with greedy search", *Applied Soft Computing*, vol. 11, no. 4, pp. 3680-3689, 2011.
- [40] S. M. Chen and C. Y. Chien, "Solving the traveling salesman problem based on the genetic simulated annealing ant colony system with particle swarm optimization techniques", *Expert Systems with Applications* , vol. 38, no. 12, pp. 14439-14450, 2011.
- [41] G. Dong, W. W. Guo and K. Tickle, " Solving the traveling salesman problem using cooperative genetic ant systems ", *Expert Systems with Applications* , vol. 39, no. 5, pp. 5006-5011, 2012.
- [42] Y. Marinakis, M. Marinaki and G. Dounias, " Honey bees mating optimization algorithm for the Euclidean traveling salesman problem", *Information Sciences* , vol. 181, no. 20, pp. 4684-4698, 2011.
- [43] W. Deng, R. Chen, B. He, Y. Liu, L. Yin and J. Guo, " A novel two-stage hybrid swarm intelligence optimization algorithm and application", *Soft Computing* , vol. 16, no. 10, pp. 1707-1722, 2012.
- [44] Y. F. Liao, D. H Yau and C. L. Chen, " Evolutionary algorithm to traveling salesman problems", *Computers & Mathematics with Applications*, vol. 64, no. 5-SI, pp. 788-797, 2012.
- [45] A. Robati, G. A. Barani, H. N. A. Pour, M. J. Fadaee and J. R. P. Anaraki, " Balanced fuzzy particle swarm optimization", *Applied Mathematical Modelling* , vol. 36, no. 5, pp. 2169-2177, 2012.
- [46] C. Blum, J. Puchinger, G. R. Raidl and A. Roli, " Hybrid metaheuristics in combinatorial optimization: A survey", *Applied Soft Computing*, vol. 11, no. 6, pp. 4135-4151, 2011.
- [47] E. E. Ivanko, " Method of scaling in approximate solution of the traveling salesman problem", *Automation and Remote Control* , vol. 72, no. 12, pp. 2527-2540, 2011.
- [48] G. Shen and Y. Q. Zhang, " A new evolutionary algorithm using shadow price guided operators", *Applied Soft Computing*, vol. 11, no. 2, pp. 1983-1992, 2011.
- [49] J. Zhang, x. Feng, B. Zhou and D. Ren, " An overall-regional competitive self-organizing map neural network for the Euclidean traveling salesman problem", *Neurocomputing* , vol. 89, pp. 1-11, 2012.
- [50] J. Zhao, Q. Liu, W. Wang, Z. Wei and P. Shi, " A parallel immune algorithm for traveling salesman problem and its application on cold rolling scheduling", *Information Sciences*, vol. 181, no. 7, pp. 1212-1223, 2011.
- [51] S. M. Chen and C. Y. Chien, " Parallelized genetic ant colony systems for solving the traveling salesman problem", *Expert Systems with Applications* , vol. 38, no. 4, pp. 3873-3883, 2011.
- [52] B. Niu, Y. Chen, L. Tan, H. Wang and L. Li, " Discrete Artificial Bee Colony Algorithm for Low-Carbon Traveling Salesman Problem", *Journal of Computational and Theoretical Nanoscience*, vol. 9, no. 10, pp. 1766-1771, 2012.
- [53] D. Karaboga, B. Gorkemli, C. Ozturk, and N. Karaboga, "A comprehensive survey: artificial bee colony (ABC) algorithm and applications," *Artificial Intelligence Review*, DOI: 10.1007/s10462-012-9328-0, 2012.
- [54] D. Karaboga, B. Gorkemli, " A combinatorial artificial bee colony algorithm for traveling salesman problem ", *Proceedings in 2011 international symposium on innovations in intelligent systems and applications (INISTA2011)*, pp 50–53, 2011.
- [55] D. Karaboga, B. Gorkemli, " A quick artificial bee colony -qABC- algorithm for optimization problems", *Proceedings in 2012 International Symposium on Innovations in*

Intelligent Systems and Applications (INISTA), pp.1-5, 2-4 July 2012, doi: 10.1109/INISTA.2012.6247010.

Tablo 2: $Ldiv=1$ olarak belirlenmiş CABC algoritmasının sonuçları

L1		Berlin5 2	KroA10 0	Pr144	Ch150	KroB150	Pr152	Rat195	D198	KroA20 0	Pr226	Pr299	Lin318	Pcb442	Fl1577
CABC	Mean	7542	21297.4	58631	6565	26350.1	73855.7	2358.2	15866.5	29551.5	81152.3	48775	43155	51607.3	22852
	Std	0	29.4183 6	35.3072 2	14.7037 4	52.2904 3	142.6464 5	3.3704 5	28.3275 4	35.3673 5	144.8199 2	53.9870 3	123.1072 7	156.9159 3	105.5035 5
	Best	7542	21282	58590	6549	26265	73682	2353	15825	29487	80929	48693	42947	51285	22737
	Worst	7542	21373	58699	6589	26420	74055	2363	15919	29611	81382	48876	43314	51831	23041
	Mean Error(%)	0	0.07236	0.16058	0.56678	0.84232	0.23574	1.5152 8	0.54816	0.62482	0.97462	1.21184	2.67910	1.63318	2.71023
	Best Error(%)	0	0	0.09054	0.32169	0.51664	0	1.2914 3	0.28517	0.40520	0.69678	1.04168	2.18420	0.99846	2.19335
	Worst Error(%)	0	0.42759	0.27674	0.93443	1.10983	0.50622	1.7219 1	0.88086	0.82743	1.26043	1.42142	3.05741	2.07373	3.55971

Tablo 3: $Ldiv=2$ olarak belirlenmiş CABC algoritmasının sonuçları

L2		Berlin5 2	KroA10 0	Pr144	Ch150	KroB150	Pr152	Rat195	D198	KroA20 0	Pr226	Pr299	Lin318	Pcb442	Fl1577
CABC	Mean	7542	21291	58641.6	6561.2	26338	73792	2355.4	15854.4	29517.5	81133	48743.1	43037.2	51539.1	22813.7
	Std	0	8.57904	41.0638 5	7.7175 1	59.6405 9	142.2385 3	6.0033 3	25.8503 3	46.5623 2	123.4479 6	98.0616 6	186.2089 1	108.3138 4	92.8030 7
	Best	7542	21282	58590	6549	26186	73682	2343	15825	29467	80872	48640	42756	51309	22665
	Worst	7542	21305	58701	6575	26411	74153	2362	15908	29599	81326	48980	43437	51686	23027
	Mean Error(%)	0	0.04228	0.17869	0.5085 7	0.79601	0.14929	1.3947 4	0.47148	0.50905	0.95061	1.14564	2.39881	1.49887	2.53809
	Best Error(%)	0	0	0.09054	0.3216 9	0.21431	0	0.8609 5	0.28517	0.33710	0.62586	0.93170	1.72975	1.04572	1.86974
	Worst Error(%)	0	0.10807	0.28016	0.7199 7	1.07539	0.63923	1.6788 6	0.81115	0.78657	1.19075	1.63723	3.35006	1.78817	3.49678

Tablo 4: $Ldiv=3$ olarak belirlenmiş CABC algoritmasının sonuçları

L3		Berlin5 2	KroA10 0	Pr144	Ch150	KroB150	Pr152	Rat195	D198	KroA20 0	Pr226	Pr299	Lin318	Pcb442	Fl1577
CABC	Mean	7542	21291.4	58649.4	6556.6	26311.6	73797.1	2350	15862.4	29545.2	81005.9	48657.2	43018.1	51555.3	22814.3
	Std	0	11.4122 7	66.6126 1	8.8566 3	49.6954 7	109.6653 5	7.2249 5	18.6021 5	34.6086 6	89.3705 2	138.3523 0	127.4523 0	76.0368 9	80.5121 7
	Best	7542	21282	58537	6542	26211	73682	2337	15838	29503	80908	48470	42810	51424	22702
	Worst	7542	21317	58746	6571	26390	74017	2361	15901	29599	81215	48838	43201	51683	22946

	Mean Error(%)	0	0.04416	0.19201	0.43811	0.69498	0.15621	1.16229	0.52217	0.60337	0.79246	0.96740	2.35337	1.53078	2.54078
	Best Error(%)	0	0	0	0.21446	0.30998	0	0.60266	0.36755	0.45968	0.67065	0.57894	1.85824	1.27220	2.03604
	Worst Error(%)	0	0.16445	0.35703	0.65870	0.99502	0.45465	1.63581	0.76679	0.78657	1.05264	1.34257	2.78855	1.78226	3.13272

Tablo 5: $Ldiv=4$ olarak belirlenmiş CABC algoritmasının sonuçları

L4		Berlin52	KroA100	Pr144	Ch150	KroB150	Pr152	Rat195	D198	KroA200	Pr226	Pr299	Lin318	Pcb442	Fl1577
CABC	Mean	7542	21292.6	58632.9	6566.4	26334.6	73842.2	2352.3	15863.9	29532.6	80999.6	48649.2	43126.7	51556.3	22791.1
	Std	0	13.62497	38.25820	8.93532	44.23844	93.52518	4.75499	24.37396	37.76029	98.17453	86.04510	118.04325	89.36112	45.79181
	Best	7542	21282	58590	6549	26217	73682	2345	15826	29503	80860	48512	42898	51389	22703
	Worst	7542	21330	58701	6583	26387	74017	2359	15905	29628	81178	48808	43292	51693	22848
	Mean Error(%)	0	0.04980	0.16382	0.58823	0.78300	0.21742	1.26130	0.53168	0.56047	0.78463	0.95079	2.61176	1.53275	2.43651
	Best Error(%)	0	0	0.09054	0.32169	0.33295	0	0.94705	0.29150	0.45968	0.61093	0.66609	2.06761	1.20327	2.04054
	Worst Error(%)	0	0.22554	0.28016	0.84252	0.98354	0.45465	1.54972	0.79214	0.88531	1.00660	1.28032	3.00506	1.80196	2.69225

Tablo 6: $Ldiv=1$ olarak belirlenmiş qCABC algoritmasının sonuçları

L1		Berlin52	KroA100	Pr144	Ch150	KroB150	Pr152	Rat195	D198	KroA200	Pr226	Pr299	Lin318	Pcb442	Fl1577
qCABC	Mean	7542	21286.5	58652.7	6565.8	26358.6	73860.1	2352.9	15877.4	29525.4	81100.5	48677.9	43086.6	51550.3	22786.4
	Std	0	7.55314	32.37915	9.07524	46.43533	138.63942	5.66480	16.42071	32.97938	212.07038	159.86272	133.83811	85.81031	92.03933
	Best	7542	21282	58590	6549	26311	73682	2338	15856	29477	80765	48379	42846	51378	22670
	Worst	7542	21305	58701	6581	26452	74064	2357	15914	29597	81432	48957	43288	51695	23016
	Mean Error(%)	0	0.02114	0.19765	0.57904	0.87485	0.24171	1.28712	0.61723	0.53595	0.91017	1.01035	2.51635	1.52093	2.41538
	Best Error(%)	0	0	0.09054	0.32169	0.69269	0	0.64571	0.48162	0.37115	0.49272	0.39011	1.94389	1.18161	1.89221
	Worst Error(%)	0	0.10807	0.28016	0.81188	1.23230	0.51844	1.46362	0.84917	0.77976	1.32264	1.58950	2.99555	1.80590	3.44734

Tablo 7: $Ldiv=2$ olarak belirlenmiş qCABC algoritmasının sonuçları

L2		Berlin52	KroA100	Pr144	Ch150	KroB150	Pr152	Rat195	D198	KroA200	Pr226	Pr299	Lin318	Pcb442	Fl1577
----	--	----------	---------	-------	-------	---------	-------	--------	------	---------	-------	-------	--------	--------	--------

	Worst Error(%)	0	0.05638	0.31603	0.65870	1.07156	0.53473	1.63581	0.76679	0.92958	1.04891	1.47330	2.96224	1.73894	3.49229
--	----------------	---	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------

Tablo 10. CABC'de problemler için uygun *Ldiv* değerleri

		Berlin52	KroA100	Pr144	Ch150	KroB150	Pr152	Rat195	D198	KroA200	Pr226	Pr299	Lin318	Pcb442	Fl1577
<i>Ldiv</i>	En iyinin hatası üzerinden	1, 2, 3, 4	1, 2, 3, 4	3	3	2	1, 2, 3, 4	3	1, 2	2	4	3	2	1	2
	Ortalama hata üzerinden	1, 2, 3, 4	2	1	3	3	2	3	2	2	4	4	3	2	4

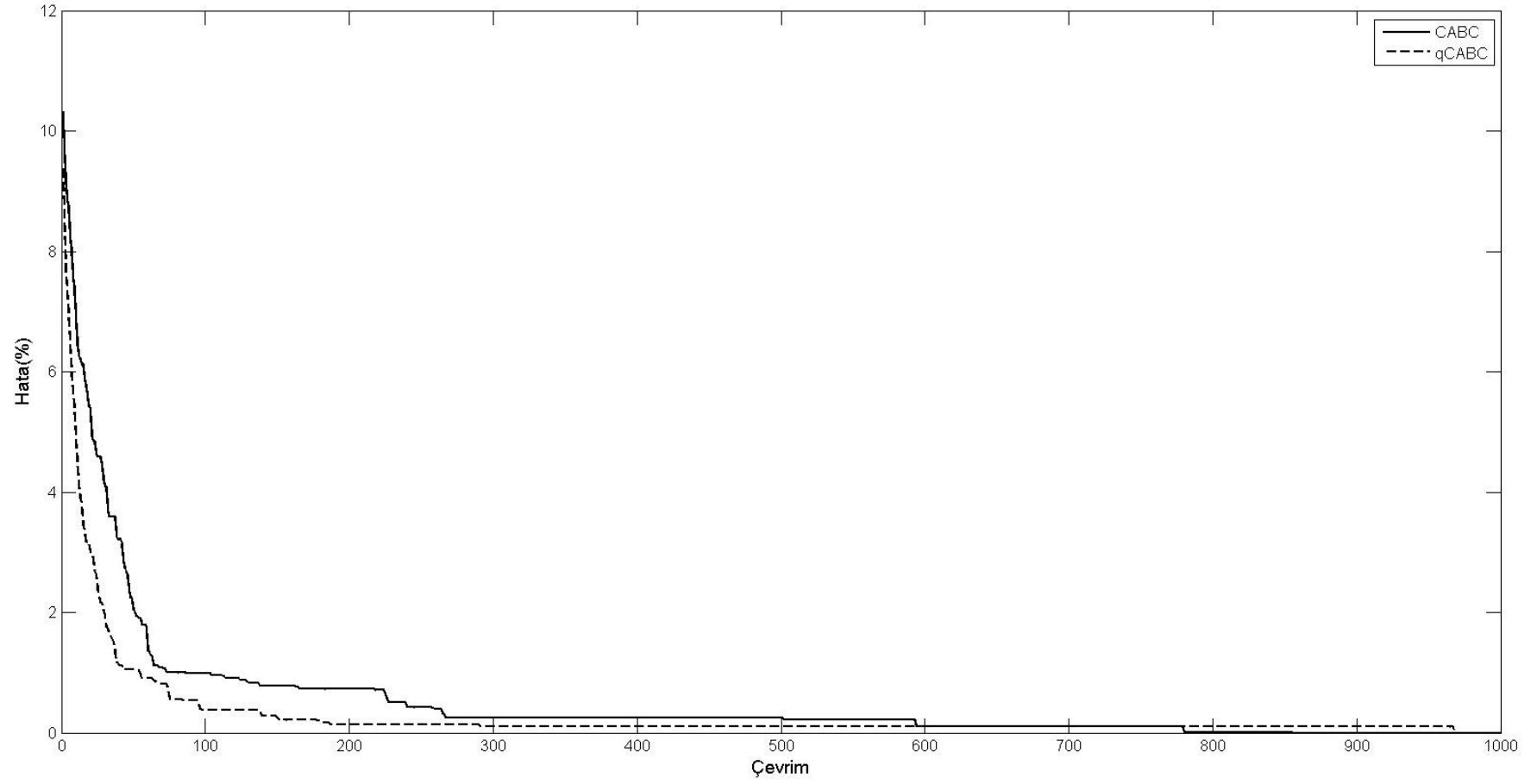
Tablo 11. qCABC'de problemler için uygun *Ldiv* değerleri

		Berlin52	KroA100	Pr144	Ch150	KroB150	Pr152	Rat195	D198	KroA200	Pr226	Pr299	Lin318	Pcb442	Fl1577
<i>Ldiv</i>	En iyinin hatası üzerinden	1, 2, 3, 4	1, 2, 3, 4	4	2	3	1, 2, 3, 4	4	3	4	1	2	2	3	1
	Ortalama hata üzerinden	1, 2, 3, 4	4	4	2	3	4	4	3	3	4	4	3	3	1

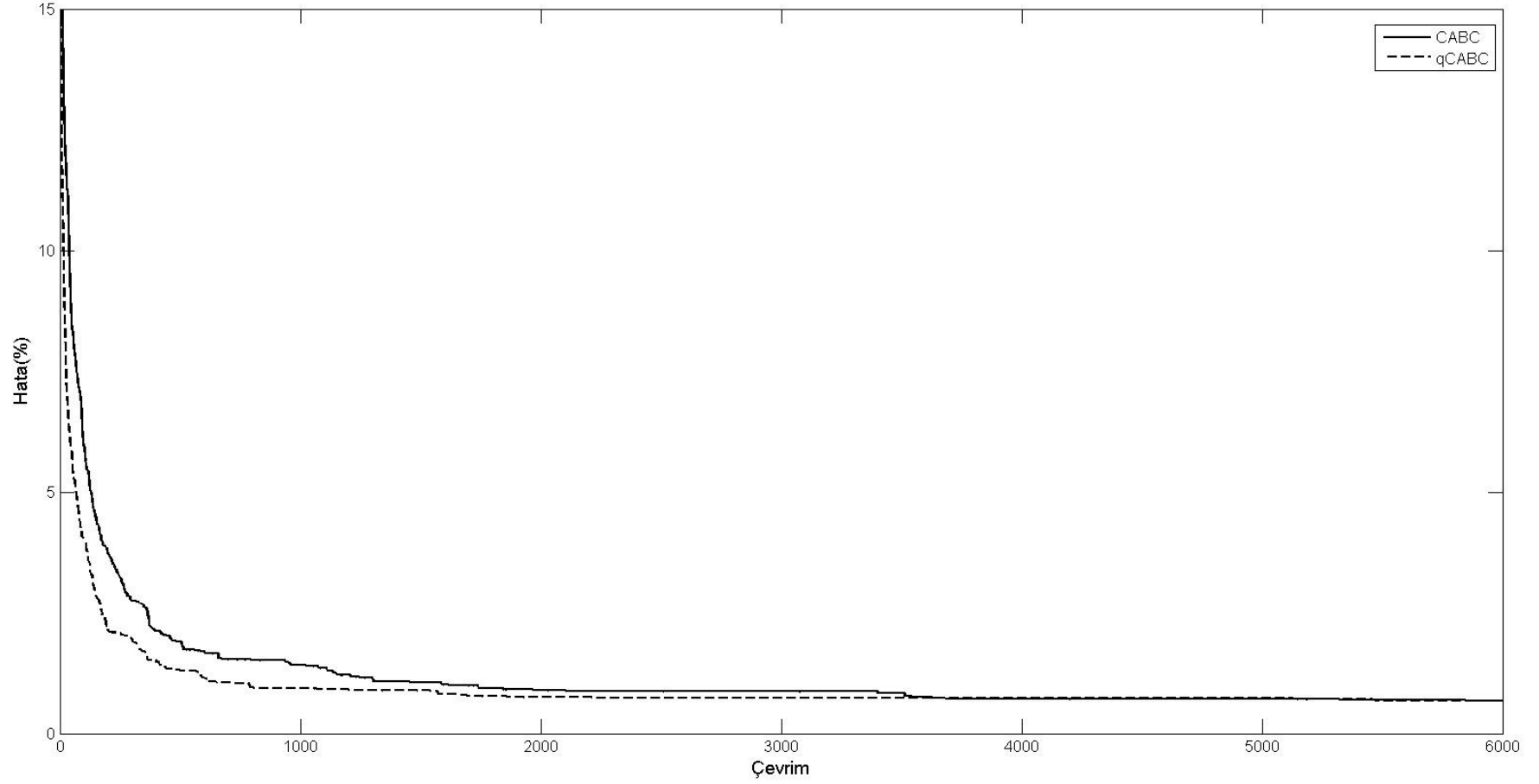
Tablo 12. CABC, qCABC ve farklı genetik algoritma çeşitlerinin kıyaslanması

	Problem	BERLIN52	KROA100	PR144	CH150	KROB150	PR152	RAT195	D198	KROA200	PR226	PR299	LIN318	PCB442	FL1577
Method															
EXC	Best Err. (%)	0	0.1692	0.2426	1.9761	2.9277	2.0358	4.477	1.9709	2.5061	4.9186	10.2198	10.1026	7.3437	-
	Ave. Err. (%)	2.0353	2.8714	0.533	2.8232	5.0919	3.99	5.9836	4.9728	4.9877	7.2295	15.1775	11.7195	10.4774	-
DISP	Best Err. (%)	0.0663	4.2947	1.4845	3.4161	7.8263	4.7216	5.0366	3.891	6.7284	4.1322	12.2616	10.9996	11.3435	-
	Ave. Err. (%)	1.424	8.0552	1.7628	3.9859	10.4635	5.5932	6.9393	5.9835	8.8276	7.993	16.849	12.9013	12.3053	-
INV	Best Err. (%)	0	2.6125	1.0284	2.9412	7.1183	3.1202	3.7021	2.6743	6.2245	8.1512	10.9543	9.541	10.5124	-
	Ave. Err. (%)	1.1854	4.7965	1.9755	3.6596	9.0521	5.0318	6.3108	4.9823	8.3594	8.8249	15.706	13.0386	11.6322	-
INS	Best Err. (%)	0	0.4652	0.1879	0.6434	1.8178	1.3938	4.2617	2.0279	2.1554	3.8423	3.8804	6.3099	8.9685	-
	Ave. Err. (%)	0.1525	3.0726	1.3802	2.3943	4.5488	2.8468	5.7426	3.41	4.2819	6.484	11.4187	10.3935	11.0585	-

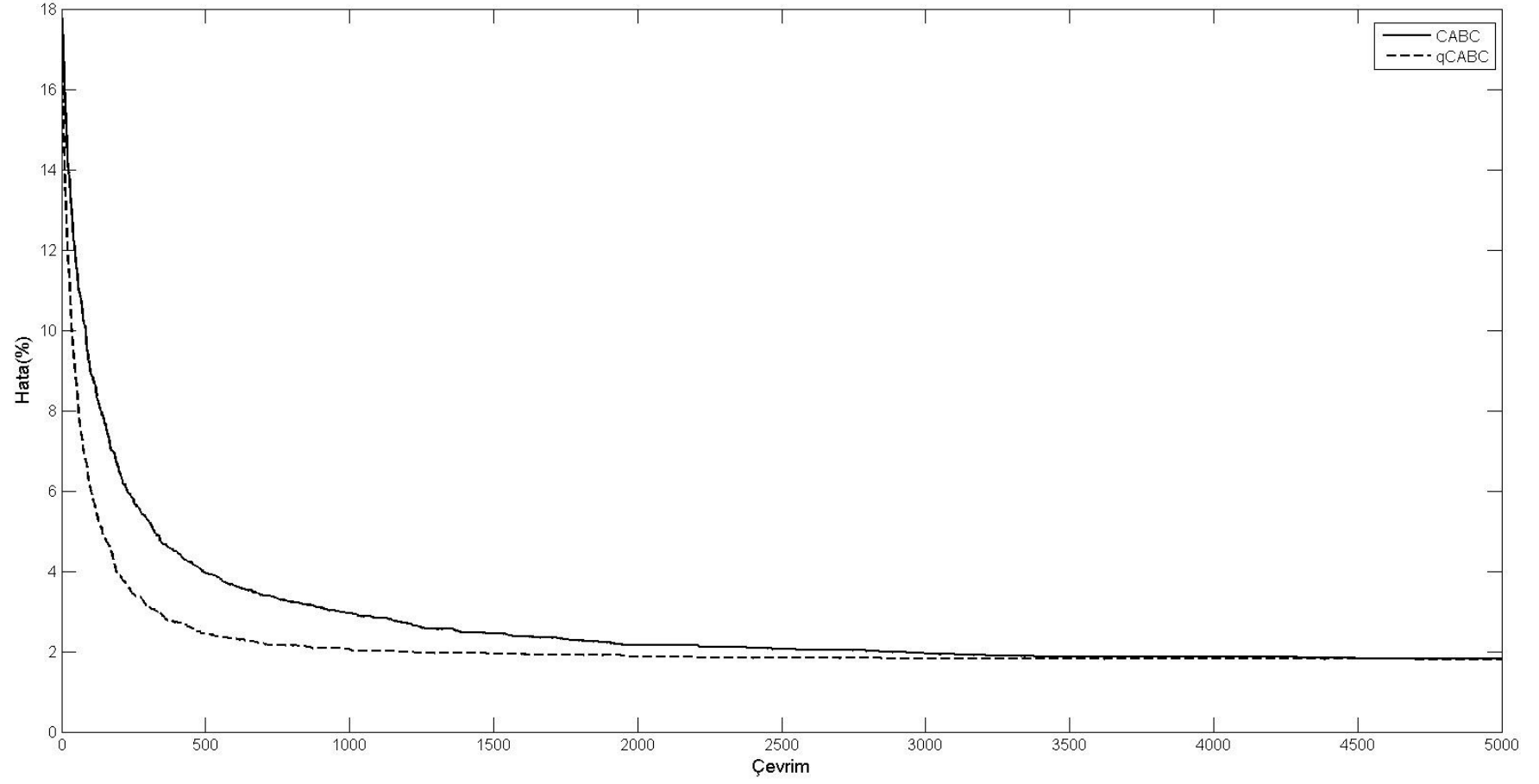
SIM	Best Err. (%)	0	0.1692	0.2255	0.8425	2.8397	1.5282	2.3676	1.6984	1.5766	2.5769	6.439	6.3932	6.3728	-
	Ave. Err. (%)	0.6099	1.3114	1.1717	1.5809	3.8596	2.7976	4.1886	2.5241	3.2457	4.1795	9.1264	8.714	8.124	-
SCM	Best Err. (%)	0	0.8176	0.1589	0.9344	3.2836	1.5146	4.6492	3.1179	3.9873	3.7875	10.8776	9.1651	7.8597	-
	Ave. Err. (%)	0	3.6966	1.3016	2.1078	7.0609	2.694	5.9923	4.5818	5.4502	5.0694	13.8686	10.4456	10.6251	-
GSM	Best Err. (%)	0	0.3007	0.2426	1.6544	3.452	2.2977	4.3048	3.0482	4.4675	5.0517	8.8191	9.4126	5.7643	-
	Ave. Err. (%)	0.9931	3.698	0.6891	2.4908	4.9541	2.9112	6.5476	4.2421	5.7018	6.4485	13.6295	11.0036	8.0389	-
GSTM	Best Err. (%)	0	0	0	0.4596	0.9644	0.7695	0.6027	0.3866	0.8683	0.7242	1.2326	0.9827	2.0501	-
	Ave. Err. (%)	0	1.1836	1.0809	0.6357	1.7616	1.6202	1.8425	1.2193	1.5432	1.5287	2.9169	3.3099	2.7758	-
CABC	Best Err. (%)	0	0	0.0905	0.2145	0.3100	0	0.6027	0.2852	0.3371	0.6109	0.6661	1.8582	1.0457	2.0405
	Ave. Err. (%)	0	0.0423	0.1606	0.4381	0.6950	0.1493	1.1623	0.4715	0.5090	0.7846	0.9508	2.3534	1.4989	2.4365
qCABC	Best Err. (%)	0	0	0	0.1838	0.2411	0	0.2152	0.2091	0.2962	0.5873	0.4710	1.8987	1.1166	1.8922
	Ave. Err. (%)	0	0.0113	0.1483	0.4197	0.8182	0.1357	1.1494	0.4943	0.4668	0.8108	0.9985	2.3914	1.4227	2.4154



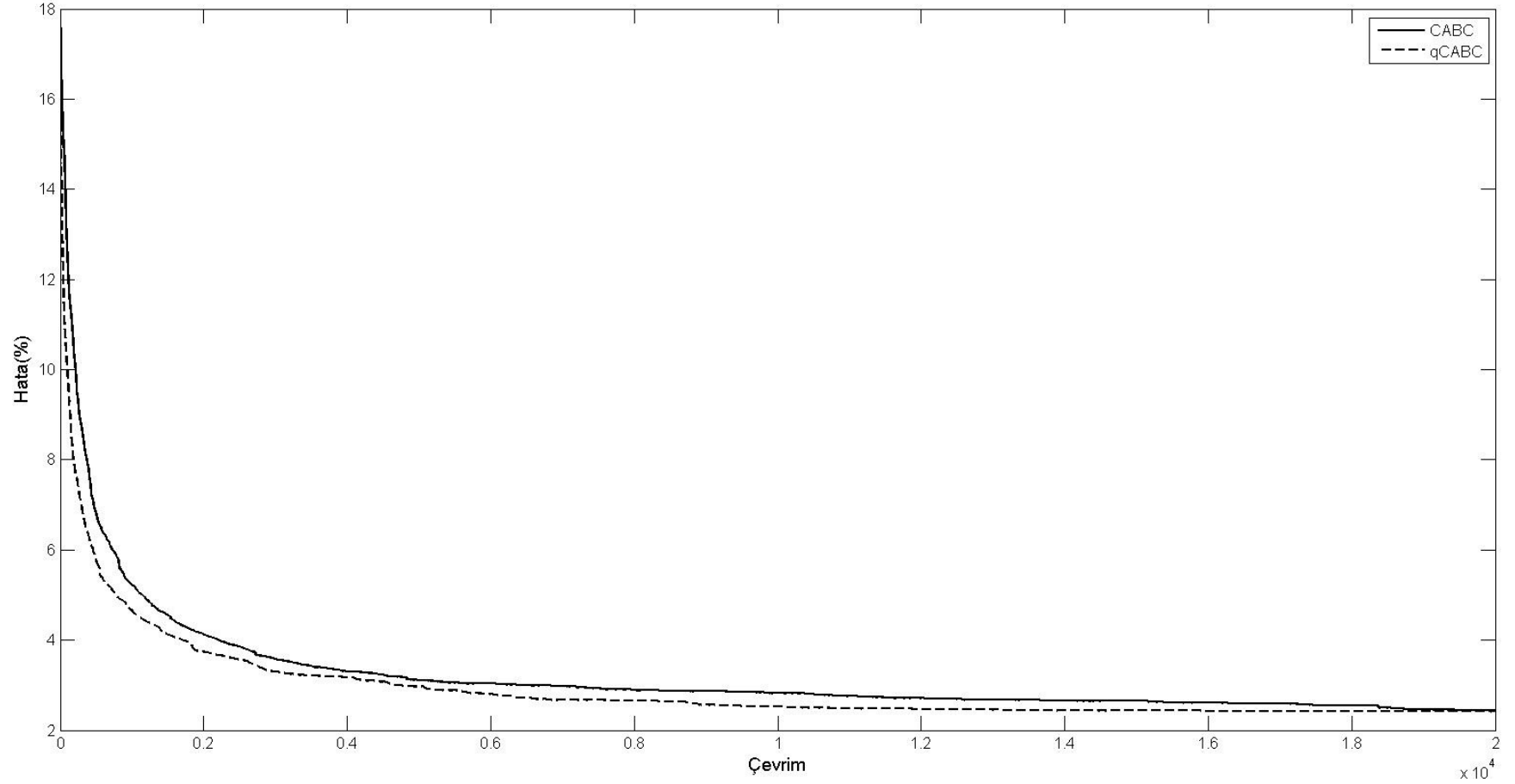
Şekil 1. Berlin52 problemi için yakınsama grafiği



Şekil 2. KroA200 problemi için yakınsama grafiği



Şekil 3. Pcb442 problemi için yakınsama grafiği



Şekil 4. F11577 problemi için yakınsama grafiği

PROJE SONUÇLARI VE VERİLERİ KULLANILARAK ŞU ANA KADAR ÜRETİLEN YAYINLAR:

* D. Karaboga, B. Gorkemli, "A quick artificial bee colony -qABC- algorithm for optimization problems", Proceedings in 2012 International Symposium on Innovations in Intelligent Systems and Applications (INISTA), pp.1-5, 2-4 July 2012, doi: 10.1109/INISTA.2012.6247010.