2.1. Yapay Arı Kolonisi

Yapay Arı Kolonisi (ABC), bal arılarının bilgi paylaşımı amacıyla yaptıkları dans ve yiyecek aramak için sürü halinde gösterdikleri zeki davranışlardan esinlenerek geliştirilen sezgisel bir optimizasyon algoritmasıdır. Esinlenilen arı kolonisi modeli üç tür arı grubundan oluşur. Bu arılar: işçi arılar, gözcü arılar ve kâşif arılar olarak adlandırılır. Koloninin ilk yarısını işçi arılar, ikinci yarısını gözcü arılar oluşturur. Arama uzayında her bir besin kaynağı için sadece bir işçi arı görevlendirilir. Dolaysıyla işçi arıların sayısı kovanın etrafında taranan besin kaynağı sayısına eşittir. Kolonideki arılar tarafından besin kaynağı tüketilen işçi arı, kâşif arıya dönüşür ve yeni bir kaynak aramaya başlar. Algoritmanın temel adımları Şekil 2.1'de görülmektedir (Karaboğa, 2005).

```
Kâşif arılar ile ilk besin kaynaklarını oluştur
REPEAT
        İşçi arıları kaynaklarına gönder ve kaynaklardaki yiyecek miktarını belirle
        Kaynakların gözcü arılar tarafından tercih edilme ihtimallerini hesapla
        Gözcü arıları seçtikleri kaynaklara gönder ve yiyecek miktarlarını belirle
        Arıların gitmekten vazgeçtiği kaynağa gidişleri durdur
        Kâşif arıları rasgele yeni kaynak aramak için arama uzayına gönder
        Şimdiye kadar bulunan en iyi çözümü hafızaya al
UNTIL (durdurma ölçütleri sağlanıncaya kadar)
```

Şekil 2.1. ABC algoritmasının temel adımları

ABC algoritmasında arama sürecinin her çevrimi üç aşamadan oluşur. Bu aşamalar sırasıyla önce işçi ve daha sonra gözcü arıların besin kaynaklarına giderek kaynaklardaki yiyecek miktarının belirlenmesi ve tükenen kaynakların belirlenerek bu kaynaklardaki işçi arıların kâşif arıya dönüştürülmeyle olası yeni kaynakların belirlenmesi aşamalarıdır. Kâşif arı sayısı, limit parametresi ile kontrol edilir. Bir kaynağı temsil eden çözüm belli bir sayıda denemeden sonra hala iyileştirilemiyorsa; bu kaynakta besin kalmadığı ve tükendiği varsayılarak terkedilir. Böylece bu kaynağın işçi arısı kâşif arıya dönüşür. Bu amaçla her besin kaynağı için bir geliştirilememe sayacı tutulur.

İlk olarak besin kaynaklarının sayısı, durdurma ölçütü ve limit parametresinin değeri tanımlanır. Her biri arama uzayında olası bir çözümü temsil eden besin

kaynaklarının başlangıç değerleri rasgele bir şekilde denklem (2.1) kullanılarak tanımlanır ve her bir besin kaynağı için bir işçi arı atanır. Kovandaki gözcü arıların sayısı da işçi arıların sayısına eşittir.

$$x_i^j = x_{min}^j + rand(0,1)\left(x_{max}^j - x_{min}^j\right) \text{ ve } i = \{1, 2, \dots, SN\} \tag{2.1}$$

Burada x_i^j kolonideki i.çözümün j.parametresini göstermektedir. SN besin kaynaklarının sayısıdır. $j = \{1, 2, \dots D\}$ olmak üzere D olası bir çözümü oluşturan karar değişkenlerinin sayısıdır. x_{min}^j ve x_{max}^j ise j.karar değişkenin alt ve üst sınır değerleridir. Kolonin başlangıç çözümleri üretildikten sonra işçi arıların uygunluk değerleri denklem (2.2) ile hesaplanır (Karaboğa, 2005; Öztürk, 2011).

$$fit_i = \begin{cases} {}^{1}/_{(1+f_i)} & f_i \geq 0 \\ 1 + abs(f_i) & f_i < 0 \end{cases} \tag{2.2}$$

Burada fit_i kolonideki i.çözümün uygunluk değeri gösterilmektedir. f_i ise i.çözümün amaç fonksiyon değeridir. Çözülecek optimizasyon problemi için özel olarak tanımlanmalıdır.

Bu noktadan sonra algoritmanın çevrimsel süreci başlar. İlk aşama yukarıda bahsedildiği gibi işçi arı fazıdır. Kolonideki tüm işçi arılar sırasıyla hafızalarındaki besin kaynağının komşuluğunda başka bir besin kaynağı seçer ve kendi çözümünü iyileştirmeye çalışır. Komşu çözüm kendisinden farklı olarak koloniden rasgele seçilir. Denklem (2.3) kullanılarak yeni çözüm hesaplanır. Her çözüm için $[1, D]$ aralığında rasgele birer parametre seçilir ve belirlenen komşu çözüm ile delta değeri bu parametre için hesaplanır.

$$delta = \emptyset \times \left(x_i^j - x_k^j\right) \quad \text{ve} \quad v_i^j = x_i^j + delta \tag{2.3}$$
$$i = \{1, 2, \dots, SN\}, i \neq j \quad \text{ve} \quad j \in \{1,2, \dots, D\}$$

Burada $delta$ yeni v_i çözümünün seçilen j. parametresi için hesaplanan değişim miktarıdır. $\emptyset$ parametresi $[-1, 1]$ aralığında rasgele üretilen bir reel sayıdır. x_k ile çözümün komşu çözümü gösterilir ve kolonideki k. çözümdür. İşçi arıların komşularıyla yeni çözümler üretmesi sonucunda karar değişkenleri önceden belirlenen aralıkların dışına çıkabilir. Böyle bir durumda denklem (2.4) kullanılarak sınır değerleri yeni çözüm için ayarlanır.

$$v_i^j = \begin{cases} x_{min}^j & v_i^j < x_{min}^j \\ v_i^j & x_{min}^j \leq v_i^j \leq x_{max}^j \\ x_{max}^j & v_i^j > x_{max}^j \end{cases} \quad (2.4)$$

Yeni üretilen çözümlerin uygunluk değerleri denklem (2.2) ile hesaplanır. Burada eski ve üretilen çözüm arasında aç-gözlü bir seçim işlemi uygulanır. Eğer yeni üretilen çözüm eskisinden daha iyi ise yeni çözüm hafızaya alınır ve iyileştirilememe sayacı sıfırlanır. Aksi halde besin kaynağının iyileştirilememe sayacı bir arttırılır.

İşçi arılar arama işini tamamlayıp kovana döndüklerinde yaptıkları danslarla besin kaynaklarının kalitesiyle ilgili bilgiyi paylaşırlar. Dansı izleyip kaynak seçimini buna göre yapan arılara gözcü arılar denir. Gözcü arılar kaynakları besin miktarlarına bağlı olarak denklem (2.5) ile verilen olasılık değerine göre seçerler. Burada uygunluğa bağlı seçim işlemi için rulet tekerleği, oransal seçim, turnuva seçimi vb. herhangi bir metot kullanılabilir. Temel ABC algoritmasında, uygunluk değerleri ile orantılı dilim boyutlarına göre seçim işleminin yapıldığı bir rulet tekerliği kullanılır (Karaboğa, 2005; Öztürk, 2011).

$$p_i = \frac{fit_i}{\sum_{n=1}^{SN} fit_i} \quad (2.5)$$

Bu olasılıklı seçim planı sayesinde uygunluk değeri yüksek olan çözümlerin ziyaret edilme sıklığı arttırılmış olur. Böylece iyi çözümlerle arama işlemine pozitif bir geri besleme sağlanmış olur.

İkinci aşama olan Gözcü arı fazına geçilir. Burada tüm kaynaklar için $[0,1]$ aralığında rasgele bir değer üretilir. Eğer bu kaynağın hesaplanmış olasılık değeri rasgele üretilen bu sayıdan büyükse, işçi arı aşamasında olduğu gibi, gözcü arı denklem (2.3) ile besin kaynağının pozisyonunda değişikliğe gider. Kaynağın yeni uygunluk değeri hesaplanır. Aç-gözlü seçim işlemine göre yeni kaynak daha iyi ise gözcü arı eskisini unutur ve yenisini hafızaya alır ya da eskisini saklamaya devam eder. Bu durumda iyileştirilememe sayacı bir attırılır. Bu işlemler tüm gözcü arılar için tekrarlanır.

İşçi ve gözcü arı fazları tamamlandıktan sonra, algoritmanın bir çevrimindeki son aşama olan kâşif arı fazına geçilir. Bu aşamada terkedilecek herhangi bir kaynak olup olmadığı kontrol edilir. Buna karar vermek için kaynakların iyileştirilememe

sayaçları limit parametresi ile karşılaştırılır. Eğer sayaç değeri limitten büyükse bu kaynak tükenmiş anlamına gelir. Bu durumda kaynakta görevli olan işçi arı kaynağı terk eder ve kâşif arıya dönüşerek yeni kaynak arar. Tükenen kaynaklar yerine denklem (2.1) kullanılarak yeni çözüm üretilir. Böylece arama sürecine negatif geri besleme yapan ve dalgalanmalara neden olan kötü çözümlerin atılması sağlanır.

Algoritma sonlanma koşulu sağlanıncaya kadar bu üç fazı tekrarlamaya devam eder. Sezgisel algoritmalarda bu amaçla kullanılan iki yaklaşım vardır. Birincisi maksimum çevrim sayısı, ikincisi ise maksimum uygunluk fonksiyonu değerlendirme sayısıdır (*FEs*). ABC algoritmasının önerilen temel sürümünde sonlanma koşulu olarak maksimum çevrim sayısı (MNC) kullanılır. Farklı algoritmaların karşılaştırılmasında bu farklığın üstesinden gelmek için $MNC = FEs/NP$ işlemi yapılır. Ancak bu işlem her çevrimde bir fonksiyon değerlendirmesi yapan algoritmalar için geçerlidir. ABC algoritması tek çevrim içinde işçi ve gözcü arı fazlarında ayrı ayrı fonksiyon değerlendirmeleri yapar. ABC algoritmasının yalnız çevrim sayısına göre çalıştırılması bir eksiklik olarak görülebilir. Bu eksikliği kapatmak amacıyla Mernik ve ark. (2015) ABC'ye maksimum fonksiyon değerlendirme sayısına göre sonlanma biçimini eklediler. ABC algoritmasının çalışma adımlarına bakıldığında kullanılan iki adet kontrol parametresi daha vardır. Bunlar işçi ve gözcü arıların sayısına eşit olarak ayarlanan koloni boyutu SN, diğer ise limit parametresidir.

Temel ABC algoritması üzerinde şimdiye kadar farklı araştırmacılar tarafından çeşitli değişiklikler önerilmiştir. Burada temel algoritmanın ana hatlarının dışına çıkmadan sadece geliştiricilerinin önerdiği bazı parametreler üzerinde durulacaktır. Algoritmanın mevcut kontrol parametrelerine ek olarak iki kontrol parametresi daha önerdiler (Akay ve Karaboğa, 2012). Kısıtlı-optimizasyon problemlerinde algoritmanın kimi zaman çözüme yavaş yakınsamasından dolayı modifikasyon oranı (MR) ve ölçekleme faktörü (SF) parametreleri eklendi.

Modifikasyon Oranı (MR): Temel ABC algoritmasında yeni bir çözüm üretilirken çözümün sadece bir parametresi değiştirilir. Bu da yakınsamanın yavaş olmasına neden olabilir. Bu frekans düzensizliğini düzeltmek amacıyla MR parametresi eklenmiştir. Bu parametre ile her bir x_i^j parametresi için $[0,1]$ aralığında rasgele üretilen bir R_i^j reel sayısı üretilir ve denklem (2.6) çözümün tüm parametreleri için kullanılarak yeni çözüm üretilir.

$$v_i^j = \begin{cases} x_i^j + \emptyset.\left(x_i^j - x_k^j\right) & R_i^j < MR \\ x_i^j & R_i^j \geq MR \end{cases} \quad (2.6)$$

Küçük bir *MR* değeri çözümün daha yavaş iyileşmesine neden olurken, büyük bir *MR* değeri ise çözümde çok fazla çeşitliliğe neden olabilir.

Ölçekleme Faktörü (SF): Bu parametre delta değişkeniyle ilgili değişiklik öngörür. Temel ABC algoritmasında delta değişkeni yerel minimumlara takılmayı önlemek amacıyla mevcut çözüme önceki çözümün $[-1,1]$ aralığında rasgele bir oranda ($\emptyset$) eklenmesini sağlar. Bu düzensizliğin büyüklüğünü kontrol edebilmek amacıyla $[-SF, SF]$ aralığında seçilen bir değişken önerilmiştir. Bu değer algoritma çalışmaya başlamadan önce ayarlanır. Küçük bir *SF* değeri hassas bir ayarlama sağlamasının yanında yavaş yakınsamaya neden olur. Büyük bir SF değerinin seçilmesi ise yakınsama hızını arttırabilirken, yeni çözümler bulma yeteneğini kısıtlayabilir. Bu sıkıntılardan dolayı uygun *SF* değerinin belirlenmesini otomatikleştiren bir yaklaşım önerilmiş ve *Adaptif SF* (*ASF*) olarak adlandırılmıştır. *ASF* Rechennberg'in 1/5 mutasyon kuralına göre denklem (2.7) kullanılarak her m adet çevrimde bir hesaplanarak güncellenir (Akay ve Karaboğa, 2012).

$$SF(t+1) = \begin{cases} SF(t) * 0.85 & \phi(m) < 1/5 \\ SF(t)/0.85 & \phi(m) > 1/5 \\ SF(t) & \phi(m) = 1/5 \end{cases} \quad (2.7)$$

2.2. Parçacık Sürü Optimizasyonu

Parçacık Sürü Optimizasyonu (PSO) popülasyon temelli sezgisel bir optimizasyon tekniğidir. Kuş ve balık sürülerinin sosyal davranışlarından esinlenilerek geliştirilmiştir. PSO algoritması evrimsel hesaplama teknikleri ile benzerlikler gösterir. Sistem rasgele üretilen aday çözümlerden oluşan bir popülasyonla başlatılır ve en iyi çözüm için iteratif olarak güncellemeler yapar. PSO algoritmasında çaprazlama ve mutasyon gibi evrimsel operatörler yoktur. Parçacık olarak adlandırılan aday çözümler, mevcut global ve yerel en iyi çözümleri takip ederek arama uzayında dolaşırlar. Esinlendiği analoji ile ifade edilirse parçacıklar arama uzayında uçarak besin kaynaklarını arar. PSO; fonksiyon optimizasyonu, yapay sinir ağları eğitimi, bulanık