# Metasezgisel Yaklaşım Algoritmalarının Python Programlama Dili Üzerinde İncelenmesi

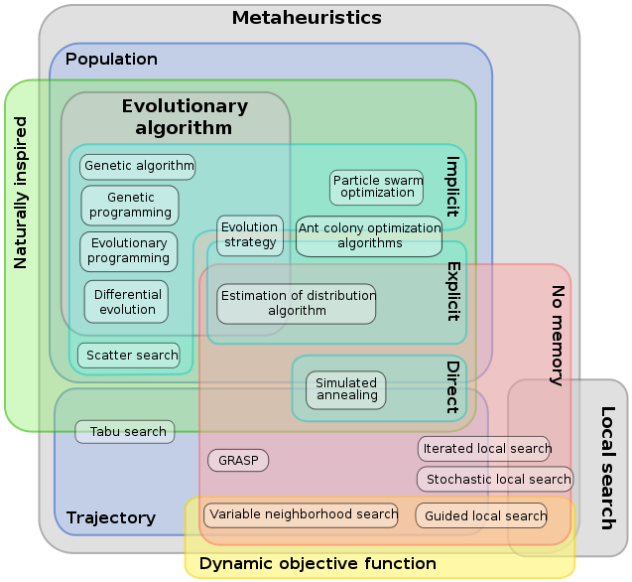
Fatih ATEŞ
Bursa Teknik Üniversitesi, 19360859074@btu.edu.tr

***Özetçe-*** **Optimizasyon, bir problem için eldeki amaç(lar) doğrultusunda çeşitli kısıtlamalar sağlanarak en optimum çözümün elde edilme sürecidir. Gelişen dünya beraberinde optimizasyon sorunlarını da beraberinde getirmiştir. Sezgisel yaklaşım algoritmaları, ayrık bir değer aralığında değişen bir amacın, optimum noktasını sezgisel yaklaşımlar aracılığıyla elde etmeyi amaçlayan algoritmalardır. Bir optimizasyon algoritmasının, her problem üzerinde veya her test fonksiyonu üzerinde başarılı olması beklenemez. Bu nedenle kullanılacak sezgisel algoritma amaç bazlı olarak tercih edilmelidir. Bu çalışmada Tavlama Benzetimi(TB) ve Parçacık Sürüsü Optimizasyonu(PSO) algoritmaları ele alınmıştır. Ackley, Beale, Goldstein-Price ve Levi test fonksiyonları ile Python programlama dili üzerinde incelemeleri gerçekleştirilmiştir.**

*Anahtar Kelimeler*- Optimizasyon, Algoritma, Metasezgisel, Tavlama, Tavlama Benzetimi, Sürü, Parçacık Sürüsü Optimizasyonu, Test fonksiyonu

## 1. GIRIŞ

Optimizasyon, bir problemin belirlenen kısıtlar altında çözüm uzayındaki optimum çözüm noktasının bulunmasıdır. Günümüzde optimizasyon problemlerinin çözümünde matematiksel teknikler ve sezgisel teknikler kullanılmaktadır. Matematiksel teknikler, problem çözerken tüm çözüm uzayını taradığı için çözüm uzayının geniş olduğu durumlarda daha maliyetli sonuçlar ortaya çıkarmaktadır. Bu tip problemlerde, çözüm uzayını sezgisel tarayarak daha kısa sürede çözüme ulaşan sezgisel algoritmaların kullanılması daha avantajlıdır (Çelik, 2013). Metasezgisel algoritmalar, temel sezgisel tekniklerin problemlere özgü uyarlanması ile elde edilen çözüm yöntemleridir. Metasezgisel terimi iki Yunanca kelime olan; "meta" (üst seviye boyunca) ve "heuristic" bulan kelimelerinin birleşimiden türetilmiştir . Bu algoritmalar, yüksek seviyeli çalışma ortamında, verimli arama işlemleri kullanarak çözüm uzayındaki optimum çözüme daha hızlı şekilde ulaşmaktadır (Çelik, 2013). Metasezgisel optimizasyon algoritmalarının en önemli avantajlarından biri de, yerel optimum noktalara takılmadan global sonuca ulaşabilme yeteneğidir (Laporte, 2006). Metasezgisel algoritmalar, belirli bir problem değil, bütün tümleşik problemlere uygulanabilmekte olup yapıları esnektir (Çelik 2013). Başka bir deyişle metasezgisel algoritmalar, üzerinde birkaç değişiklik yapılarak belirli problemlere adapte edilebilen aynı zamanda farklı optimizasyon problemleri üzerinde uygulanabilen, genel algoritmik yapı olarak görülebilir. Metasezgisel algoritmaların sınıflandırılması Şekil 1.1'de gösterilmiştir.




**Şekil 1.1**. Metasezgisel Algoritmaların Sınıflandırılması (Wikipedia, 2012)

## 2. METASEZGISEL ALGORITMALAR

### 2.1. TAVLAMA BENZETIMI

Tavlama benzetiminin temeli Metropolis ve diğerlerinin (1953) çalışmasına dayanmaktadır. Bu çalışmada belirli bir ısı seviyesinde, bir katının ısıl dengeye ulaşmasını taklit eden bir algoritma geliştirilmiştir. Daha sonar Kirkpatrick ve diğerleri (1983) tarafından Metropolis ve diğerlerinin (1953) geliştirmiş olduğu algoritmayı temel alarak kombinatoryel eniyileme problemlerinde kullanılabilecek bir algoritma önerilmiştir (Akbulut, 2020).

TB, genel minimum iç enerji konfigürasyonuna ulaşmak amacıyla tavlama işlemini kullanarak büyüyen kristallerin istatistiksel sürecinin simülasyonuna dayanan stokastik bir yaklaşımdır. Daha açık bir ifadeyle katı bir malzemenin önce belirli bir sıcaklığa kadar ısıtılıp sonra soğutulması yoluyla malzemenin kalitesinin artırılmasını sağlayan fiziksel tavlama sürecinin taklidine dayanmaktadır. Bu yaklaşımda, metal gibi bir katı madde yüksek bir sıcaklığa kadar ısıtlır ve erimiş bir hale getirilir. Atomlar, bu durumda özgürce hareket edebilirler. Bununla birlikte, atomların hareketleri sıcaklığın düşürülmesi yoluyla kısıtlanır. Sıcaklık

azaldıkça, atomlar daha az hareket etmeye eğilimlidir ve Kristal formları mümkün olan en az dahili enerjiye sahiptir. Kristallerin oluşumu temelde soğutma oranı ile ilgilidir. Erimiş metallerin soğutma hızı çok hızlı olduğunda, bu sıcaklığın çok hızlı bir şekilde azaldığı anlamına geleceğinden Kristal halini elde edemeyebilir. Bunu yerine, Kristal duruma kıyasla daha yüksek bir enerji durumuna sahip olan bir polikristalin durumuna ulaşabilir. Mühendislik uygulamalarında hızlı soğutmanın sonunda malzemenin içinde kusurlar meydana gelir. Bu nedenle en düşük enerji durumana (iç enerji) ulaşmak için ısıtılmış katı (erimiş metal) sıcaklığının yavaş ve kontrollü bir oranda azaltılması gerekir. Yavaş bir oranda bu soğutma, tavlama olarak adlandırılır (Koç, 2019).

Tavlama benzetiminde soğutma işleminin kontrolünü, bir diğer deyişle hangi hızda yapılacağını ayarlamayı sağlayan fonksiyon, optimum çözümün aranması esnasında daha iyi olmasalar da genel en iyi çözümün bulunmasına yardımcı olacak çözümlerin kabul edilebilirliğine ilişkin olasılığı ifade eder. Aramaya ilk başlandığında bu olasılığın daha iyi olmayan çözümlerin kabul edildiği ve bu doğrultuda yerel en iyi çözümlerden kaçınarak genel en iyi çözümü bulmayı sağlayan yeterince yüksek bir değerde olacak şekilde ayarlanması gerekir. En iyi çözümün aranması süresince olasılığın gittikçe azaldığı dolayısıyla yerel en iyi çözümden kaçmanın zorlaştığı görülür. Burada amaçlanan, eldeki yerel en iyi çözümün komşuları aranarak yeni bir yerel en iyi çözüme geçmektense genel en iyi çözüme ulaşılmaya çalışılmasıdır (Koç, 2019).

Tavlama benzetiminde kötü çözümü seçme olasılığı sistemli bri şekilde sıcaklıkla azaltılır. TB Algoritmasının temel amacı, çözüm uzayında aranmadık bölge bırakmamaktır. Tavlama Benzetimi'nin kombinatoryal optimizasyon problemleri için optimum yakın çözümler veren kullanışlı bir yöntem olarak kullanıldığı söylenebilir. TB, gezgin satıcı probelmi, çizelgeleme, karesel atama problemi, şebeke tasarımı gibi birçok kombinatoryal eniyileme probleminin çözümünde kullanılmıştır (Sarıkaya, 2014).

Herhangi bir problemin çözümünde TB algoritmasının kullanılması için bazı parametrelerin belirlenmesi gerekir. Bu parameterler; başlangıç sıcaklığı ($T_0$), her sıcaklıktaki iterasyon sayısı, soğutma fonksiyonu, algoritmayı durdurma kriteri. Başlangıç sıcaklığı bir girdi parametresidir. Sıcaklık, kötü çözümlerin kabul edilme olasılığını control etmek için kullanılır. İterasyon sayısı, her sıcaklıkta üretilen çözümlerin sayısıdır. Soğutma fonksiyonu, bir önceki iterasyon sıcaklığına bağlı olarak mevcut iterasyondaki sıcaklığı belirler. Başlangıç sıcaklığı ile birlikte iterasyon sayısı ve soğutma fonksiyonu, soğutma çizelgesi olarak adlandırılır. Bu çiczelge, çözüm kalitesinde veya yakınsama oranında büyük etkiye sahiptir. Her sıcaklık değişiminde elde edilen çözüm, çok sayıda ardışık sıcaklık değişimlerinde değişmiyor ise TB Algoritması durdurulur (Sarıkaya, 2014).

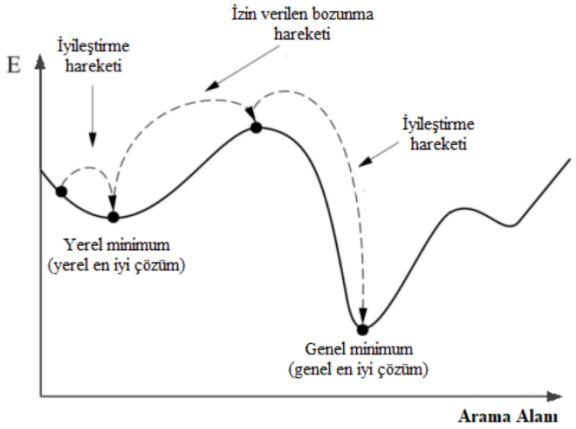
Güner ve Altıparmak (2003), fiziksel tavlama ile kombinatoryal eniyileme arasındaki ilişkiyi Çizelge 2.1'deki gibi ifade etmiştir (Akbulut, 2020).

| Termodinamik Simülasyon | Kombinatoryel En İyileme Problemi |
|---|---|
| Sistem Durumları | Uygun Çözümler |
| Enerji | Amaç Fonksiyonu |
| Durumun Değişimi | Komşu Çözüm |
| Sıcaklık | Kontrol Parametresi |
| Donma Durumu | Sezgisel Çözüm |

**Çizelge 2.1.** Fiziksel tavlama ile kombinatoryal eniyileme arasındaki ilişki

### 2.1.1 METODOLOJI

TB algoritması rastgele başlangıç çözümüyle başlar ve her yinelemede yerel komşulukla sonraki çözümü üretir. Sistemin enerji düzeyini temsil eden 'E' amaç fonksiyonunu geliştiren (maddenin/sistemin enerjisini azaltan) yani eniyileyen yeni çözüm her zaman kabul edilir. Öte yandan, sistemin sıcaklığını arttırmayam üsade eden ya da sistemdeki amaç fonksiyonundan uzaklaşmaya/bozunmaya belirli bir düzeyde izin veren geçici çözüm önerileri de kabul edilmektedir (Akbulut 2020). Algoritmanın temel işleyişi Şekil 2.1 ile ifade edilmiştir.




**Şekil 2.1**. Tavlama Benzetimi Algoritmasının Temel İşleyişi (Koç, 2020)

Mevcut ve yeni üretilen çözümün arasındaki enerji farkı (ΔE), amaç fonksiyonunun mevcut çözümün rassal olarak üretilen komşuluğu aracılığıyla oluşturulan yeni çözümü ile amaç fonksiyonunun mevcut çözümü arasındaki farkı ifade eder. Yeni çözümün kabul edilme durumu Denklem 2.1'de gösterildiği gibi Boltzmann dağılımına dayanmaktadır.

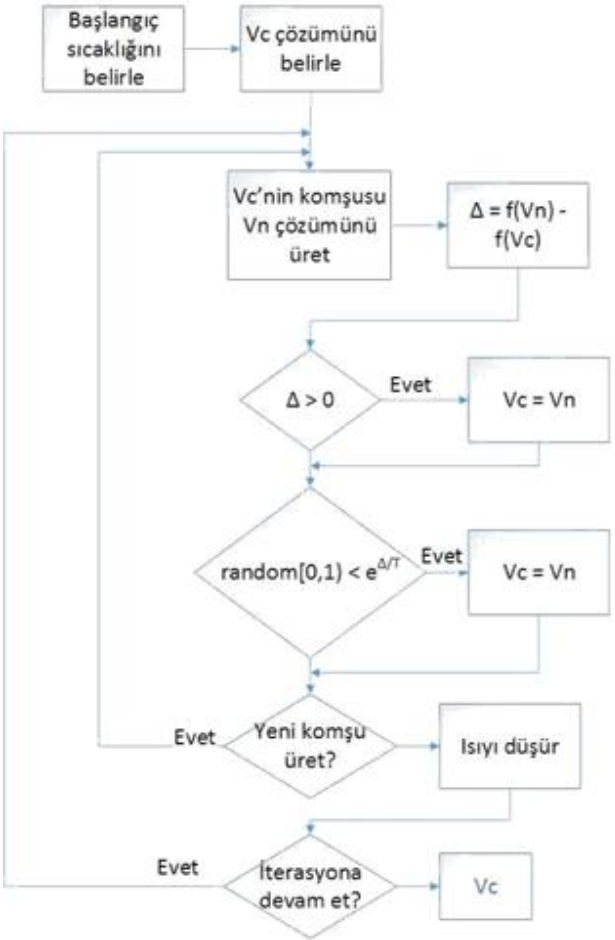
$$P(E) = e^{(-\Delta E/kB*T)}$$

**Denklem 2.1** Boltzmann dağılımı

Eğer ki rastgele üretilen sayı Bolztmann Dağılımından küçük ise yeni çözüm Kabul edilir, eğer rastgele üretilen sayi Boltzmann Dağılımından küçük değilse yeni çözüm kabul edilmez, eski çözüm ile iterasyona devam edilir. TB Algoritmasının sözde kodu Şekil 2.2 ile verilmiştir Akış şeması ise Şekil 2.3 ile verilmiştir.

```
Başlangıç çözümü seç
Başlangıç sıcaklığı seç (T0 > 0 )
Sıcaklık değişim sayacını ayarla
Tekrarla
Tekrarlama sayısını belirle
    Tekrarla
    Yeni bir komşu çözümü üret (Komşu arama algoritması)
    Amaç fonksiyonu değişimi hesapla (Δ= mutlak(f(Komşu Çözüm) −f(Çözüm)))
    Eğer(f(Komşu Çözüm) < f(Çözüm) ) ise
        yeni çözümü seç
    Değil ise
        Eğer ras[0,1) < e^(-ΔE/kB*T) ise
            yeni çözümü seç
    Tekrarlama sayısını bir arttır
    Tekrarlama Sayısı=N(Sıcaklık Değişimi) şartını sağlayana kadar devam et
Sıcaklık değişimini bir arttır
Sıcaklığı güncelle
İterasyon sonlanıncaya kadar devam et
```

**Şekil 2.2.** Tavlama Benzetimi Sözde Kodu




**Şekil 2.3** Tavlama Benzetimi Akış Şeması
(Şefik Temel, Mustafa Ö. Cingiz, Oya Kalıpsız)

### 2.1.2 SONUÇ

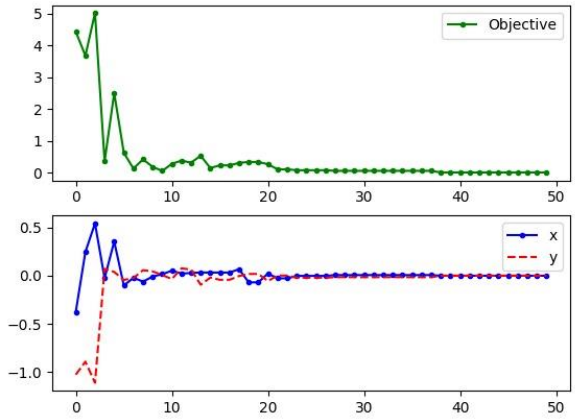
Şekil 2.3 üzerinde bulunan akış şemasında ifade edilmiş olan TB algoritması Python dili üzerinde gerçekleştirilmiştir, üzerinde Ackley, Beale, Goldstein-Price ve Levi Test fonksiyonları gerçekleştirilmiştir. Gerçekleştirilen testlerden elde edilen sonuçlar sırasıyla aşağıda sunulmaktadır.

Tüm testler için kullanılan parametreler ve değerleri Çizelge 2.2 ile verilmiştir.

| Parametre | Değer |
|---|---|
| startedLocation | [0.5, -0.5] |
| numberOfIterations | [50, 500] |
| temperatureValues | 700 |
| fraction | 0.88 |
| functionToOptimize | // Test fonksiyonu |
| functionRange | // Test fonksiyonunun aralığı |

**Çizelge 2.2.** TB Test fonksiyonlarında kullanılan parametreler

TB üzerinde Ackley test fonksiyonu ile üretilen değerlerin grafiği Şekil 2.4 ile verilmiştir.



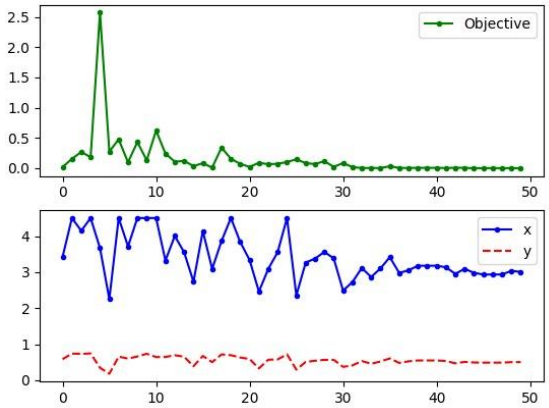

**Şekil 2.4.** Ackley test fonksiyonu grafiği

Ackley test fonksiyonu için değer aralığı ve beklenen en iyi değerler Çizelge 2.3 ile verilmiştir.

| Parametre | Değer |
|---|---|
| Değer aralığı | X => [-5, 5], Y => [-5, 5] |
| Elde edilebilecek en iyi çözüm değerleri | X = 0, Y = 0 |
| Elde edilebilecek en optimum değer | 0 |
| Elde edilen çözüm değerleri | X = -0.00119562669897455<br>Y= -0.0008832548326439538 |
| Elde edilen optimum değer | 0.004263276791252935 |

**Çizelge 2.3.** Ackley test fonksiyonundan elde edilen değerler

TB üzerinde Beale test fonksiyonu ile üretilen değerlerin grafiği Şekil 2.5 ile verilmiştir.



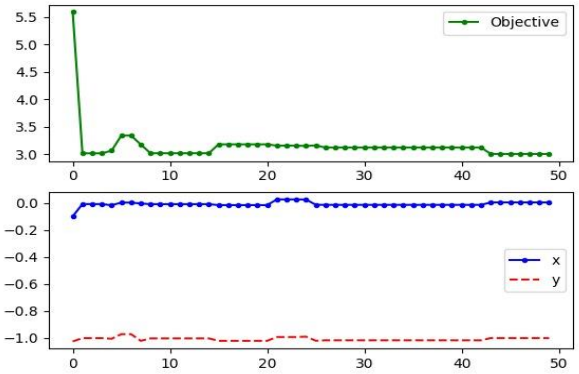

**Şekil 2.5.** Beale test fonksiyonu grafiği

Beale test fonksiyonu için değer aralığı ve beklenen en iyi değerler Çizelge 2.4 ile verilmiştir.

| Parametre | Değer |
|---|---|
| Değer aralığı | X => [-4.5, 4.5], Y => [-4.5, 4.5] |
| Elde edilebilecek en iyi çözüm değerleri | X = 3, Y = 0.5 |
| Elde edilebilecek en optimum değer | 0 |
| Elde edilen çözüm değerleri | X = 3.00448000488404<br>Y= 0.5025383356664512 |
| Elde edilen optimum değer | 5.07253128713777724e-05 |

**Çizelge 2.4.** Beale test fonksiyonundan elde edilen değerler

TB üzerinde Goldstein-Price test fonksiyonu ile üretilen değerlerin grafiği Şekil 2.6 ile verilmiştir.



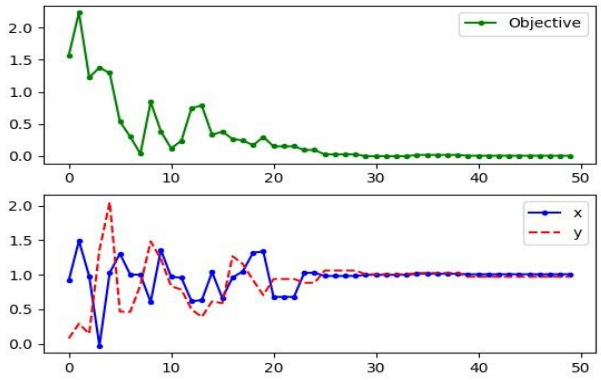

**Şekil 2.6.** Goldstein-Price test fonksiyonu grafiği

Goldstein-Price test fonksiyonu için değer aralığı ve beklenen en iyi değerler Çizelge 2.5 ile verilmiştir.

| Parametre | Değer |
|---|---|
| Değer aralığı | X => [-2, 2], Y => [-2, 2] |
| Elde edilebilecek en iyi çözüm değerleri | X = 0, Y = -1 |
| Elde edilebilecek en optimum değer | 3 |
| Elde edilen çözüm değerleri | X = 0.004099428038386865<br>Y = -1.0001308106302074 |
| Elde edilen optimum değer | 3.00436862126945 |

**Çizelge 2.5.** Goldstein-Price test fonksiyonundan elde edilen değerler

TB üzerinde Levi test fonksiyonu ile üretilen değerlerin grafiği Şekil 2.7 ile verilmiştir.




**Şekil 2.7.** Levi test fonksiyonu grafiği

Levi test fonksiyonu için değer aralığı ve beklenen en iyi değerler Çizelge 2.6 ile verilmiştir.

| Parametre | Değer |
|---|---|
| Değer aralığı | X => [-10, 10], Y => [-10, 10] |
| Elde edilebilecek en iyi çözüm değerleri | X = 1, Y = 1 |
| Elde edilebilecek en optimum değer | 0 |
| Elde edilen çözüm değerleri | X = 1.008089079742875<br>Y= 0.9719072291301988 |
| Elde edilen optimum değer | 0.006684399680206016 |

**Çizelge 2.6.** Levi test fonksiyonundan elde edilen değerler

Sonuç olarak üretilen algoritmanın beklenen değerlere oldukça yaklaştığı gözlemlenmiştir.

### 2.2 PARÇACIK SÜRÜSÜ OPTIMIZASYONU

Parçacık sürüsü Optimizasyonu (PSO), kuş ve balık sürülerinin sosyal davranışları gözlemlenerek geliştirilmiş bir metasezgisel algoritmadır. 1995 yılında Eberhart ve Kennedy tarafından ortaya atılmıştır. Parçacık Sürüsü Optimizasyonuna aynı zamanda kuş sürüsü optimizasyonu da denilmektedir (Çetin, 2013).

Bu algoritmada; balık ve kuş sürüleri yiyecek ya da barınak bulmak amacıyla belirli bir alan taramaktadırlar. PSO, bu sürülerin sosyal davranışlarından oluşur. Bu davranışlardan ilki sürü içerisindeki her bir parçacığın geçmiş hatıraları içerisinden en iyi konuma gitme davranışıdır. İkinci davranış ise sürü içerisinde bulunan yiyeceğe en yakın parçacığı takip etme hareketidir. Son davranış ise parçacığın geniş alan taramasını sağlayan geçmiş hız değerleridir. Bu davranışlar PSO'nun temelini oluşturmaktadır (Çetin, 2013).

#### 2.2.1 METODOLOJI

PSO algoritması popülasyon temelli bir metasezgisel algoritmadır. PSO algoritması rastgele çözümler içeren bir popülasyonla başlar ve her iterasyonda güncelleme yaparak küresel optimum yanıtı vermeye çalışır. Sürü içerisindeki her bir kuş bir cevabı temsil etmektedir. Aynı zamanda her bir kuş hareket ettiği boyutta bir yanıt üretmektedir. Verilen cevaplardan her biri kuşun o andaki pozisyonudur. Bu

pozisyona $p_{best}$ adı verilir. Algoritmada $p_{best}$ değerlerinin en iyisine ise $g_{best}$ adı verilir ve bu değerde aranan optimum değerdir.

Örneğin D boyutlu bir arama uzayında hareket eden S adet parçacığın hız ve konumları aşağıdaki gibi ifade edilebilir. Burada, X konum matrisi Denklem 2.2 ile verilmiştir, V hız matrisi ise Denklem 2.3 ile verilmiştir.

$$X = \begin{bmatrix} X_{11} & X_{12} & \dots & X_{1D} \\ X_{21} & X_{22} & \dots & X_{2D} \\ \dots & \dots & \dots & \dots \\ X_{S1} & X_{S2} & \dots & X_{SD} \end{bmatrix}$$

**Denklem 2.2**

$$V = \begin{bmatrix} V_{11} & V_{12} & \dots & V_{1D} \\ V_{21} & V_{22} & \dots & V_{2D} \\ \dots & \dots & \dots & \dots \\ V_{S1} & V_{S2} & \dots & V_{SD} \end{bmatrix}$$

**Denklem 2.3**

Denklem 2.2 ve Denklem 2.3 matrislerinde bir i'nci parçacığın konumu $X_i = [X_{i1} \quad X_{i2} \quad \dots \quad X_{iD}]$ şeklinde, i'nci parçacığın hızı ise $V_i = [V_{i1} \quad V_{i2} \quad \dots \quad V_{iD}]$ matrisi ile ifade edilir. Parçacığın en iyi konumu (yerel en iyi, $p_{best}$) ise Denklem 2.4 ile verilen matris $p_{best}$ matrisidir.

$$p_{best} = \begin{bmatrix} pbest_{11} & pbest_{12} & \dots & pbest_{1D} \\ pbest_{21} & pbest_{22} & \dots & pbest_{2D} \\ \dots & \dots & \dots & \dots \\ pbest_{S1} & pbest_{S2} & \dots & pbest_{SD} \end{bmatrix}$$

**Denklem 2.4**

pbest matrisi S parçacığın D boyutundaki hareketlerinde geçerli zamana kadar geçen sürede bulunduğu en iyi konumu tutar. i'nci parçacığın bulunduğu en iyi konum $p_{best,i} = [pbest_{i1} \quad pbest_{i2} \quad \dots \quad pbest_{iD}]$ matrisi şeklinde ifade edilir.

Sürünün küresel olarak bulunduğu en iyi konum $g_{best}$, $p_{best}$ matrisi içerisindeki en iyi konumu ifade eder. Denklem 2.5 ile gbest matrisi verilmiştir.

$$g_{best} = [gbest_{11} \quad gbest_{12} \quad \dots \quad gbest_{1D}]$$

**Denklem 2.5**

PSO kavramsal olarak, parçacıkların hızlarının her bir nesilde kendi yerel en iyi konumlarına ve sürünün global en iyi konumuna göre belirlenmesine dayanır. Evrimsel gelişim süresinde her bir parçacığın hızı ve konumu Denklem 2.6 ve Denklem 2.7 ile güncellenir (Gözde ve diğerleri, 2008).

$$v_{i,d}^{(t+1)} = w * v_{i,d}^{t} + c_1 * r_1 * \left(pbest_{i,d} - x_{i,d}^{t}\right) + c_2 * r_2 * (gbest_d - x_{i,d}^{t})$$

**Denklem 2.6**

$$x_{i,d}^{(t+1)} = x_{i,d}^{t} + v_{i,d}^{(t+1)}$$

**Denklem 2.7**

**i = 1, 2, …, S**

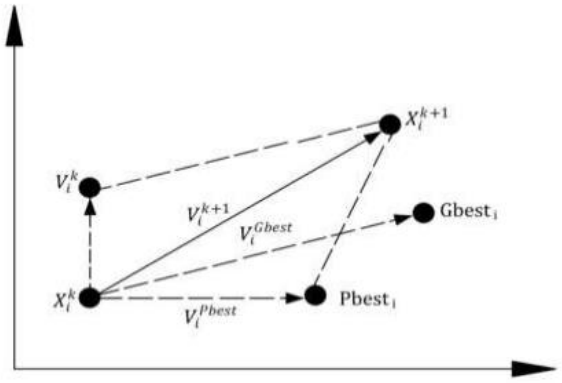
Denklem 2.6'da $c_1$ ve $c_2$ sabitleri pozitif değerli olup genelde [0.2, 2] aralığında probleme bağlı olarak değişkenlik gösterebilir. Probleme göre sabit bir katsayı verilebilir. Bazı problemlerde parçacıkların anılarının hesabı katılması sebebiyle $p_{best}$ gibi değerlere bağlı olarak dinamik atama yapılabilir. $r_1$ ve $r_2$ sayıları rassal olarak üretilen katsayılardır. Her iterasyonda probleme verilen yanıta bir rastlantısallık katmaktadır. $r_1$ ve $r_2$ sayıları genelde [0,1) aralığında üretilmektedir. W ise atalet momenti olup genellikle 0.1 ile 1 aralığında değişmektedir (Çetin, 2013). PSO'da eylemsizlik ağırlığı global ve yerel arama yeteneğini dengelemek için kullanılmaktadır. Büyük eylemsizlik momenti global arama, küçük eylemsizlik momenti ise yerel arama yapmayı kolaylaştırır. Böylece eylemsizlik momenti yerel ve global araştırma arasındaki dengeyi sağlar ve en az sayıdaki iterasyonla sonuca götürmeyi amaçlar. Buradaki her bir parçacık; sürüdeki sadece en iyi parçacığın değil sürüdeki diğer tüm parçacıkların tecrübelerinden de yararlanmış olur (Tamer ve Karakuzu, 2006).

W'nın doğrusal azalması Denklem 2.8 ile verilmiştir (Kennedy ve Eberhart, 1995).

$$w = w_{maks} - iterasyon * \frac{w_{maks} - w_{min}}{iterasyon_{maks}}$$

**Denklem 2.8**

PSO'da parçacıklar çoklu arama uzayında iterasyon süresince kadar pozisyonlarını sürekli değiştirirler. Arama uzayındaki değişimler Şekil 2.8 ile verilmiştir.




**Şekil 2.8** PSO parametrelerinin vektör olarak gösterimi (Hamed Hosseini, Mehdi Shahbazian, Mohammad Ali Takassi)

Şekilde 2.4 üzerindeki parametreler şunlardır:

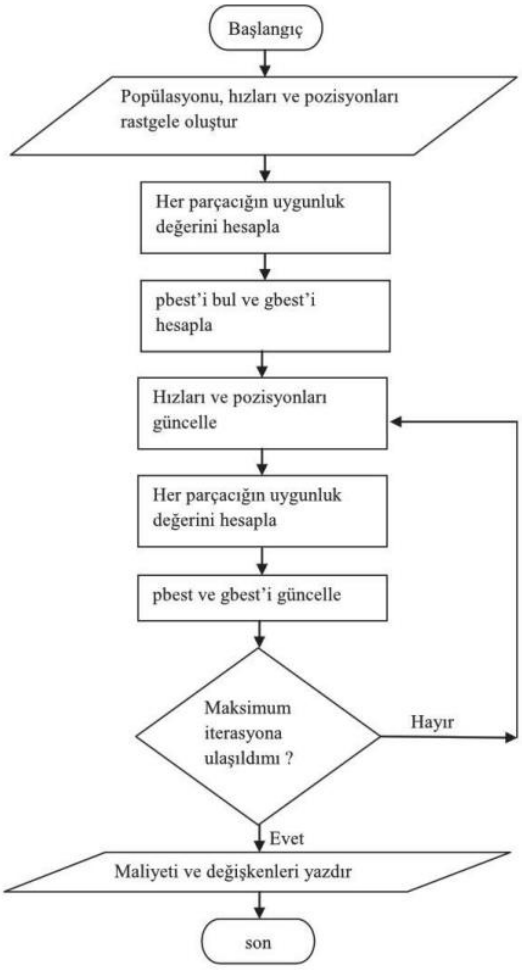
- $x^k$: anlık lokasyon
- $x^{k+1}$: yeni lokasyon
- $v^k$: anlık hızı
- $v^{k+1}$: yeni hızı
- $v^{pbest}$: pbest ile hesaplanan hız
- $v^{gbest}$: gbest ile hesaplanan hız
- $p_{best}$: yerel en iyi lokasyon
- $g_{best}$: küresel en iyi lokasyon

PSO algoritması sözde kodu Şekil 2.9 ile verilmiştir.

```
Rassal olarak başlangıç sürüsü tanımla
   for t=1: maksimum iterasyon
      w(omega)'yı hesapla
      for i=1: parçacık sayısı
         for d=1: boyut
            v_i,d(t+1) = w*v_i,d(t)+c1*r1*(p_i-x_i,d(t))+c2*r2*(p_g-x_i,d(t))
            x_i,d(t+1) = x_i,d(t)+v_i,d(t+1)
         yeni objective değeri hesapla
      if f(x_i,d(t)) < f(p_i(t)) then
         p_i(t) = x_i,d(t)
         f(p_g(t)) =f(p_i(t))
         if f(x_i,d(t)) < f(g_i(t)) then
            g_i(t) = x_i,d(t)
            f(g_g(t)) =f(p_i(t))
         end
      end
```

**Şekil 2.9** Parçacık Sürüsü Optimizasyonu Sözde kodu

PSO algoritması için akış şeması Şekil 2.10 ile verilmiştir.




**Şekil 2.10** Parçacık Sürüsü Optimizasyonu Akış Şeması

### 2.2.2 SONUÇ

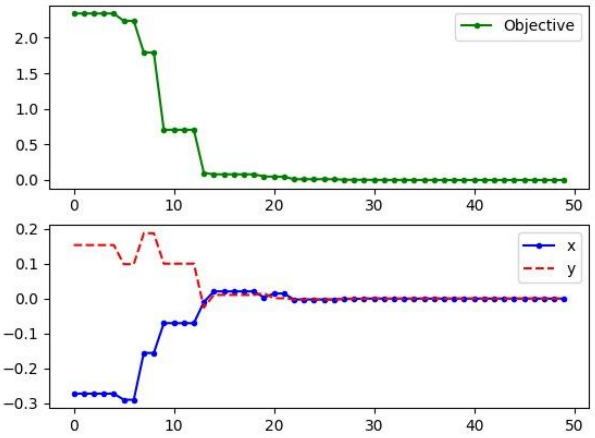
Şekil 2.10 üzerinde bulunan akış şemasında ifade edilmiş olan PSO algoritması Python dili üzerinde gerçekleştirilmiştir, üzerinde Ackley, Beale, Goldstein-Price ve Levi Test fonksiyonları gerçekleştirilmiştir. Gerçekleştirilen testlerden elde edilen sonuçlar sırasıyla aşağıda sunulmaktadır.

Tüm testler için kullanılan parametreler ve değerleri Çizelge 2.7 ile verilmiştir.

| Parametre | Değer |
|---|---|
| functionToOptimize | // Optimize edilecek fonksiyon |
| lowerBoundary | // Test fonskyionu alt aralıkları |
| upperBoundary | // Test fonksiyonu üst aralıkları |
| particleSize | 100 |
| c1 | 0.5 |
| c2 | 0.5 |
| numberOfIteratıons | 50 |

**Çizelge 2.7.** PSO Test fonksiyonlarında kullanılan parametreler

PSO algoritması üzerinde Ackley test fonksiyonu ile üretilen değerlerin grafiği Şekil 2.11 ile verilmiştir.



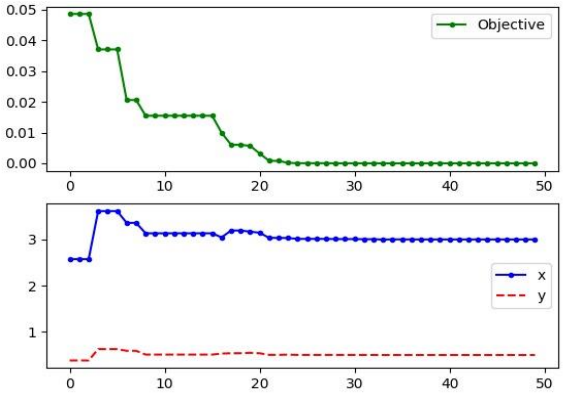

**Şekil 2.11.** Ackley test fonksiyonu grafiği

Ackley test fonksiyonu için değer aralığı ve beklenen en iyi değerler Çizelge 2.8 ile verilmiştir.

| Parametre | Değer |
|---|---|
| Değer aralığı | X => [-5, 5], Y => [-5, 5] |
| Elde edilebilecek en iyi çözüm değerleri | X = 0, Y = 0 |
| Elde edilebilecek en optimum değer | 0 |
| Elde edilen çözüm değerleri | X = -3.1871733347611965e-08<br>Y = 1.802381914336243e-07 |
| Elde edilen optimum değer | 5.177005206746799e-07 |

**Çizelge 2.8.** Ackley test fonksiyonundan elde edilen değerler

PSO algoritması üzerinde Beale test fonksiyonu ile üretilen değerlerin grafiği Şekil 2.12 ile verilmiştir.



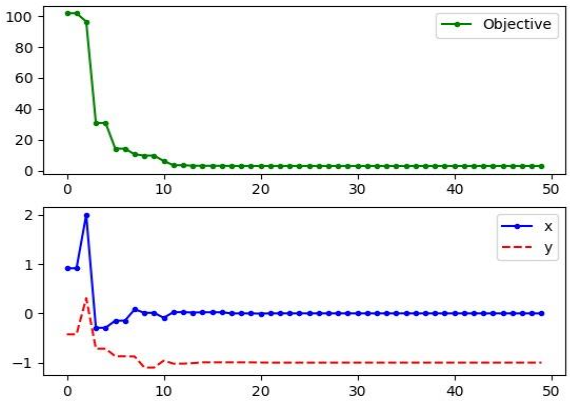

**Şekil 2.12.** Beale test fonksiyonu grafiği

Beale test fonksiyonu için değer aralığı ve beklenen en iyi değerler Çizelge 2.9 ile verilmiştir.

| Parametre | Değer |
|---|---|
| Değer aralığı | X => [-4.5, 4.5], Y => [-4.5, 4.5] |
| Elde edilebilecek en iyi çözüm değerleri | X = 3, Y = 0.5 |
| Elde edilebilecek en optimum değer | 0 |
| Elde edilen çözüm değerleri | X = 3.000004103428678<br>Y = 0.5000009276859919 |
| Elde edilen optimum değer | 2.881213431705946e-12 |

**Çizelge 2.9.** Beale test fonksiyonundan elde edilen değerler

PSO algoritması üzerinde Goldstein-Price test fonksiyonu ile üretilen değerlerin grafiği Şekil 2.13 ile verilmiştir.

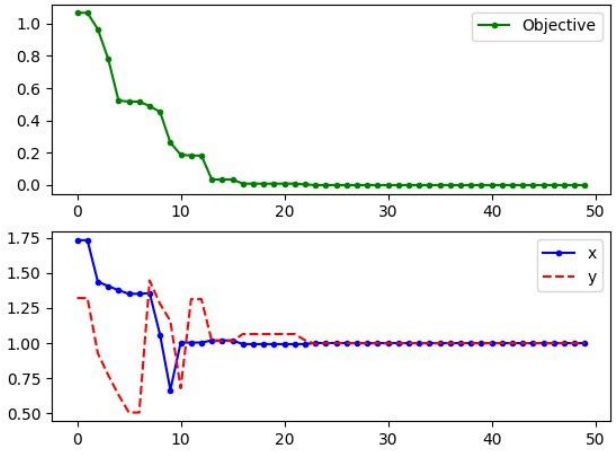



**Şekil 2.13.** Goldstein-Price test fonksiyonu grafiği

Goldstein-Price test fonksiyonu için değer aralığı ve beklenen en iyi değerler Çizelge 2.10 ile verilmiştir.

| Parametre | Değer |
|---|---|
| Değer aralığı | X => [-2, 2], Y => [-2, 2] |
| Elde edilebilecek en iyi çözüm değerleri | X = 0, Y = -1 |
| Elde edilebilecek en optimum değer | 3 |
| Elde edilen çözüm değerleri | X = 1.443907680869234e-08<br>Y = -0.999999988551917 |
| Elde edilen optimum değer | 3.00000000000008 |

**Çizelge 2.10.** Goldstein-Price test fonksiyonundan elde edilen değerler




**Şekil 2.14.** Levi test fonksiyonu grafiği

Levi test fonksiyonu için değer aralığı ve beklenen en iyi değerler Çizelge 2.11 ile verilmiştir.

| Parametre | Değer |
|---|---|
| Değer aralığı | X => [-10, 10], Y => [-10, 10] |
| Elde edilebilecek en iyi çözüm değerleri | X = 1, Y = 1 |
| Elde edilebilecek en optimum değer | 0 |
| Elde edilen çözüm değerleri | X = 0.9999999871832931<br>Y = 1.0000003902407675 |
| Elde edilen optimum değer | 1.6704346415824158e-13 |

**Çizelge 2.11.** Levi test fonksiyonundan elde edilen değerler

Sonuç olarak üretilen algoritmanın beklenen değerlere oldukça yaklaştığı gözlemlenmiştir. Dahası Python üzerinde veri tiplerinin bir sınırı bulunduğundan dolayı parçacık sayısı yükseltildiğide tam anlamıyla global optimuma ulaşılabilmektedir.

## 3. KAYNAKLAR


Çelik, Y. (2013). Optimizasyon Problemlerinde Bal Arılarının Evlilik Optimizasyonu Algoritmasının Performansının Geliştirilmesi. Doktora Tezi. Konya, Türkiye: Selçuk Üniversitesi Fen Bilimleri Enstitüsü.

Laporte, G. (2006). Classical And Modern Heuristics For The Vehicle Routing Problem. International transactions in operation research , 285-300.

Metropolis N [ve diğerleri] Equa-tion of state calculations by fast computing machines [Dergi] // J. Chem. Phys. - 1953. - Cilt 21. - s. 1087–1092 .

Hatice Erdoğan AKBULUT – Müfredat temelli üniversite ders çizelgeleme problem için bir tavlama benzetimi algoritması 2020 sy. 43-45

Bilgen Ayann KOÇ – Hemşire nöber çizelgeleme probleminin tavlama benzetimi algoritması ile çözümü 2019 sy. 11-18

Hüseyin Ali SARIKAYA– Bütünleşik tedarik zinciri ağında tesis yeri seçimi problem için bulanık çok amaçlı programlama modeline sezgisel bir yaklaşım: Tavlama benzetimi algoritması 2014 sy. 92-94

Güner, Ertan ve Fulya ALTIPARMAK, "İki Ölçütlü Tek Makinali Çizelgeleme Problemi için Sezgisel Bir Yaklaşım" 3003 sy. 27-42

Erhan ÇETİN – Parçacık sürüsü optimizasyonu tabanlı pıd controller ile AA servomotor denetimi 2013 sy. 18-21

Gözde, H., Kocaarslan, İ., Taplamacıoğlu, M.C., Çam, E., 2008. İki bölgeli güç sisteminde parçacık sürüsü algoritması ile yük-frekans kontrolü 55 optimizasyonu. Elektrik-Elektronik ve Bilgisayar Mühendisliği Sempozyumu ELECO'08, 26-30Kasım, Bursa, 212-216.

Yüksel Çelik, İlker Yıldız ve Alper Talha Karadeniz (2019).Avrupa Bilim ve Teknoloji Dergisi Özel Sayı, S. 463-477, Ekim 2019