**LECTURE 1**

**Metaheuristic Algorithms**

Bu dersin konuları:

* Bu ders ne hakkında?
* Zor optimizasyon problemleri
* Meta-sezgisel (metaheuristic) sınıflandırması
* Ana konular, organizasyon ve değerlendirme kuralları

 Bilgisayar bilimi derslerinin çoğunda olduğu gibi bu ders de problem çözmeyle ilgilidir.

 Amacı, zor problemleri çözmek için teknikler sunmaktır.

 Bazı problemler:

* Hem insanlar hem de bilgisayarlar için zordur (örneğin, büyük kombinatoryal problemler, çoklu hedefli optimizasyonlar vb.)
* Bilgisayarlar için zor ama insanlar için kolaydır (örneğin karakter, yüz veya ses tanıma) – bu tür problemler "iyi tanımlanmamış" problemlerdir.

**Hesaplama Olarak Zor Problemler**

* Bu problemler çok büyük çözüm alanlarına sahiptir ve bu nedenle polinomsal zamanlı kesin algoritmalarla çözülemezler (NP-zor problemler).
* Örnekler:
  + SAT (Boolean Doyurulabilirlik Problemi): n değişken için 2ⁿ olası kombinasyon vardır.
  + Gezgin Satıcı Problemi (TSP): n şehir için (n−1)! olası tur vardır.

**İyi Tanımlanmamış Problemler (Ill-posed problems)**

* İnsanlar için kolay ama bilgisayarlar için zor olan problemlerin özelliği, onların iyi tanımlanmamış olmalarıdır; yani, problemin tüm özelliklerini yansıtan soyut bir model oluşturmak zordur.
* → Aşağıdaki iki problemi ele alalım:

1. Bir şirketin çalışanlarını iki kategoriye ayırın: birinci kategori, şirketteki ortalama maaştan daha yüksek gelire sahip olan tüm çalışanları içerecek; ikinci kategori ise diğer çalışanları içerecek.
2. Bir şirketin çalışanlarını iki kategoriye ayırın: birinci kategori, banka kredisi için iyi aday olan tüm çalışanları içerecek; ikinci kategori ise diğer çalışanları içerecek.

**İyi Tanımlanmamış Problemler**

* İlk problem durumunda, kural tabanlı bir sınıflandırıcı oluşturmak kolaydır:

EĞER gelir > ortalama ise → Sınıf 1  
AKSİ TAKDİRDE → Sınıf 2

* İkinci problem durumunda ise bir sınıflandırıcı oluşturmak o kadar kolay değildir çünkü belirli bir çalışanın banka kredisi için güvenilir olup olmadığına karar vermek için dikkate alınması gereken birçok birbirine bağlı unsur vardır (sağlık durumu, aile, kariyer gelişimi vb.).
* Bir banka uzmanı, karar verirken kendi deneyimine (önceki başarılı ve başarısız vakalar) dayanır.

**İyi Tanımlanmış ve Kötü Tanımlanmış Problemler**

* İyi tanımlanmış problemler ile kötü tanımlanmış problemler arasındaki farklar:

**İyi Tanımlanmış Problemler:**

* Problemi tanımlayan soyut bir model vardır.
* Sonuç olarak, bir çözüm yöntemi yani bir algoritma vardır.

**Kötü Tanımlanmış Problemler:**

* Kolayca biçimlendirilemezler.
* Sadece sonucu bilinen bazı örnekler vardır.
* Problemin verileri eksik veya tutarsız olabilir.
* Bu nedenle, geleneksel yöntemler uygulanamaz.

**İyi Tanımlanmamış Problemler**

İyi tanımlanmamış problemler için uygun olan yöntemler şu özelliklere sahip olmalıdır:

* Örneklerden modeller çıkarabilme yeteneği (öğrenme)
* Dinamik ortamlarla başa çıkabilme yeteneği (uyarlanabilirlik)
* Gürültülü, eksik veya tutarsız verilerle başa çıkabilme yeteneği (sağlamlık)
* Makul bir sürede cevap verebilme yeteneği (verimlilik)

Tüm bu özelliklere sahip bir sistem tasarlamak genellikle bir optimizasyon problemini çözmeye yol açar (= bir hatayı en aza indiren, bir maliyeti en aza indiren ya da bir kalite ölçütünü maksimize eden tasarım değişkenlerini bulmak)

**Optimizasyon Problemleri**

* Kabaca tanım: Arama uzayında, bir veya birden fazla optimizasyon ölçütünü (amaç fonksiyonları olarak adlandırılır) optimize eden (minimize eden veya maksimize eden) ve bir veya birden fazla kısıtı sağlayan bir veya birkaç öğeyi bulmaktır.

**Matematiksel Tanım (Tek Amaçlı Optimizasyon):**

**metin, yazı tipi, ekran görüntüsü içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.**

**Optimizasyon Zorlukları**

Arama uzayı  
= Tasarım değişkenlerinin değer aldığı alan

* Büyük boyut  
  (çok sayıda tasarım değişkeni)
* Karmaşık kısıtlarla tanımlanır  
  (uygun çözüm bölgesi kolayca erişilebilir değildir)

Amaç fonksiyonu

* “Kara kutu” fonksiyon  
  (sadece fonksiyonun değerleri hesaplanabilir; özellikleri hakkında bilgi yoktur)
* Çok sayıda optimum vardır  
  (çok modlu optimizasyon)
* Gürültülü fonksiyon:  
  Aynı argüman için yapılan farklı değerlendirmeler farklı sonuçlar verebilir  
  (gürültülü veya dinamik optimizasyon)
* Birden fazla (birbiriyle çelişen) amaç ölçütü vardır  
  (çok amaçlı optimizasyon)

Not:

* Geleneksel teknikler (örneğin matematiksel programlama) ya uygulanamaz ya da verimsizdir.

**Optimizasyon Problemi Sınıfları**

* **Ayrık (kesikli) arama uzayı** → **Kombinatoryal optimizasyon**  
  **Örnekler:**
  + Rota belirleme (örneğin: araç rotalama)
  + Planlama (örneğin: zaman çizelgesi hazırlama, görev zamanlama)
  + Dağıtım (örneğin: kaynak tahsisi)
  + Seçim (örneğin: öznitelik/özellik seçimi)
* **Sürekli arama uzayı** → **Sürekli optimizasyon**  
  **Örnekler:**
  + Parametre tahmini
  + Minimum enerji yapılandırmalarını bulma
  + Uyarlanabilir sistemlerin eğitimi

**🔄 Routing (Yönlendirme)**

🛻 Vehicle Routing Problem (VRP) – Araç Rotalama Problemi

* Tanım: Bir dizi lokasyonu ziyaret edecek ve bazı kısıtları sağlayacak şekilde en düşük maliyetli rotayı bulma problemidir.

🧭 Özel Durum: Travelling Salesman Problem (TSP) – Gezgin Satıcı Problemi

* Tanım: Tam bağlantılı (complete) bir graf içinde minimum maliyetli bir Hamiltonian devre (her düğümü bir kez ziyaret edip başlangıç noktasına dönülen yol) bulmaktır.

🧮 Çözüm Temsili 1: Binary Allocation Matrix (ikili atama matrisi)

* Açıklama:  
  n x n boyutlu bir matris düşün.
  + Aij=1A\_{ij} = 1Aij​=1: i. adımda j. düğüm (lokasyon) ziyaret edilmiş demektir.
  + Aij=0A\_{ij} = 0Aij​=0: i. adımda j. düğüm ziyaret edilmemiştir.
* Kısıtlar (Constraints):
  + Her satırda yalnızca bir tane 1 olmalı (her adımda yalnızca bir düğüm ziyaret edilir).
  + Her sütunda yalnızca bir tane 1 olmalı (her düğüm yalnızca bir kez ziyaret edilir).
* Arama Uzayı (Search Space):
  + Bu matrislerin oluşturduğu küme.
  + Boyut: 
* Amaç Fonksiyonu (Objective Function):  
  Verilen maliyet matrisine göre toplam rotanın maliyetini hesaplar.

metin, yazı tipi, beyaz, çizgi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.​

Burada:

* + AijA\_{ij}Aij​: i. adımda j. düğüm ziyaret edildiyse 1.
  + cjkc\_{jk}cjk​: j düğümünden k düğümüne gitmenin maliyeti.

🔁 Çözüm Temsili 2: Permütasyon (Sıralama Dizisi)

* Tanım:
  + pip\_ipi​: i. adımda ziyaret edilen düğümün indeksidir.
  + Yani, permütasyon dizisi rotayı belirtir: örn. [1, 3, 2, 4] gibi.
* Kısıtlar:
  + Permütasyonun tüm elemanları farklı olmalı (her düğüm bir kez ziyaret edilmeli).
* Arama Uzayı (Search Space):
  + n düğüm varsa, olası permütasyonların sayısı:
    - Asimetrik problemde: n!
    - Simetrik problemde: (n−1)!/2
* Amaç Fonksiyonu:
  + Her iki ardışık düğüm arasındaki maliyetleri toplayarak toplam yol maliyetini bulur.

📌 Özet:

Bu slaytta, Gezgin Satıcı Problemi (TSP) için iki farklı çözüm gösterimi sunulmuş:

1. Binary matrix (ikili matris): Hangi adımda hangi düğümün ziyaret edildiğini belirtir.
2. Permutation (permütasyon): Ziyaret sırasını doğrudan belirtir.

Her iki gösterim için de:

* Kısıtlar belirtilmiş.
* Arama uzayının büyüklüğü verilmiş.
* Hedef fonksiyonu (yani toplam maliyet) matematiksel olarak ifade edilmiş.

**📅 Planlama Problemi (Planning Problem)**

**🔹 Problem Tanımı:**

**Aşağıdaki ögeler verilir:**

* **Etkinlikler (events): Örneğin dersler, sınavlar.**
* **Odalar (rooms): Etkinliklerin yapılabileceği sınıflar, salonlar.**
* **Zaman dilimleri (time slots): Hangi zamanlarda etkinlikler yapılacak.**

**🎯 Amaç:**

**Her etkinliği bir oda ve bir zaman dilimi ile eşleştiren bir program (takvim/schedule) oluşturmak. Bu program çeşitli kısıtları sağlamalıdır.**

**🧱 Kısıtlar (Constraints)**

**📌 1. Zorunlu (Hard) Kısıtlar – Sağlanmak zorunda**

Bunlar sağlanmazsa program geçersizdir (infeasible):

* Her etkinlik sadece bir kez yerleştirilmeli.
* Aynı anda ve aynı odada yalnızca bir etkinlik olabilir.
* Oda, etkinliğin gereksinimlerini karşılamalı (örneğin kapasite, projektör vb.).
* Aynı kişilerin katılması gereken iki etkinlik aynı anda olamaz.

→ Bu ilişkiler çatışma grafiği (conflict graph) ile gösterilir:  
Grafın düğümleri etkinliklerdir. İki düğüm arasında bağlantı varsa, o iki etkinlik aynı anda yapılamaz**.**

**🔄 2. Yumuşak (Soft) Kısıtlar – İhlal edilebilir ama cezalandırılır**

Bu kısıtlar ihlâl edilebilir, ancak çözüm kalitesi düşer. Örnek:

* Bir katılımcı için art arda çok fazla etkinlik olmamalı (örneğin 3'ten fazla ders arka arkaya olmasın).
* Bazı katılımcılar için, günde yalnızca bir etkinlik olmamalı (boş geçen günleri azaltma hedefi).

🧠 Fikir: Bu yumuşak kısıtlar birer optimizasyon kriterine dönüştürülebilir.  
Amaç: İhlal sayısını (veya etkisini) en aza indirmek.

**🔍 Arama Uzayı (Search Space)**

* Her bir (zaman dilimi, oda) çifti bir etkinlik ya da "boş" olabilir.
* Etkinlik sayısı: k
* Zaman dilimi sayısı: m
* Oda sayısı: n

→ Arama uzayı büyüklüğü (kısıtsız durumda): metin, ekran görüntüsü, yazı tipi, çizgi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

**📈 Optimizasyon Yaklaşımı**

* **Geçerli (feasible) çözümleri bulmak için önce hard kısıtlar sağlanmalı.**
* **Daha sonra soft kısıtları minimum düzeyde ihlal eden çözümler tercih edilir.**

**🎯 1. Kaynak Tahsisi Problemi (Resource Allocation - Cloud Resource Allocation)**

**🧩 Problem:**

* **Verilen:**
  + n tane görev (task) → Her biri belirli bir kaynak ihtiyacına sahip.
  + m tane sanal makine (VM) → Her birinin belirli kapasitesi var.
* **Amaç:**Görevleri sanal makinelere öyle ata ki:
  + Her görevin ihtiyaçları karşılanmış olsun.
  + Kullanılan sanal makine sayısı veya toplam maliyet/enerji tüketimi minimum olsun.

**🔧 Çözüm Temsili:**

* V = (v₁, v₂, ..., vₙ)  
  vᵢ: Görev i'nin atandığı sanal makinenin indeksi.

**🔍 Arama Uzayı:**

* Bütün görevler için m adet sanal makineden birini seçebiliriz → mⁿ

**Selection**

**🧠 2. Öznitelik Seçimi (Feature/Attribute Selection)**

**🧩 Problem:**

* **Verilen: Çok sayıda öznitelik (attribute) içeren bir veri kümesi.**
* **Amaç:**
  + **En yüksek sınıflandırma doğruluğunu elde etmek.**
  + **Aynı zamanda işlem maliyetini ve model karmaşıklığını azaltmak.**

**🔧 Çözüm Temsili:**

* **İkili vektör (binary vector):  
  S = (s₁, s₂, ..., sₙ)  
  sᵢ = 1 → öznitelik i seçildi, 0 → seçilmedi**

**🔍 Arama Uzayı:**

* **Toplam öznitelik sayısı n → 2ⁿ farklı seçim kombinasyonu**

**🎯 Amaç Fonksiyonu:**

* **Sınıflandırma doğruluğu (maximize edilir)**

**🖼️ 3. Görüntü Eşleme (Image Registration - Parametre Tahmini)**

**🧩 Problem:**

* Verilen: Aynı nesneye ait iki görüntü.
* Amaç: İkinci görüntüye bir dönüşüm (rotation, translation vs.) uygulayarak birinci görüntü ile olabildiğince benzer hale getirmek.

**🔧 Çözüm Temsili:**

* Gerçek sayı vektörü: Her bir parametre (dönme açısı, kaydırma mesafesi vb.) bir elemandır.

**⚛️ 4. Moleküler Yapı Optimizasyonu (Parameter Estimation - Atom Konumlandırma)**

**🧩 Problem:**

* Verilen: Belirli sayıda atom.
* Amaç: Atomların uzaydaki koordinatlarını öyle seç ki sistem iç enerjisi minimum olsun.

**🔧 Çözüm Temsili:**

**🎯 Amaç Fonksiyonu:**

* **Potansiyel enerji fonksiyonu (örnek: Lennard-Jones), minimize edilir.**

**⚠️ Zorluk:**

* **Küresel optimumu bulmak çok zordur. Özellikle atom sayısı arttıkça yerel minimumlara takılma olasılığı artar.**

**🎯 Adaptif Sistemlerin Eğitimi (Training Adaptive Systems)**

Eğitim, modelin adaptif (uyarlanabilir) parametrelerini öyle bir şekilde tahmin etmek (bulmak) demektir ki, bir hata fonksiyonu (loss function) minimize edilsin veya bir kalite kriteri maksimize edilsin.

**metin, ekran görüntüsü, yazı tipi, çizgi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.**

**🔗 Yapay Sinir Ağı (ANN)**

**Yapay sinir ağı, birbirine bağlı yapay nöronlardan oluşur. Bu nöronlar, ağırlıklar (weights) olarak adlandırılan uyarlanabilir parametrelerle karakterize edilir.**

**Bir yapay sinir ağının yapısı:**

* **Mimari (kaç katman, her katmanda kaç nöron vb.)**
* **İşleyiş (veri nasıl akar, aktivasyon fonksiyonu nasıl çalışır)**
* **Denetimli eğitim (Supervised training)**
  + **Amaç: Ağ tarafından verilen yanıt ile beklenen yanıt arasındaki farkı minimize eden ağırlıkları bulmak.**

**🎨 Sanatsal Yapıların Tasarımı (Görseller, Sesler vb.)**

* **Amaç: Kullanıcının verdiği öznel (subjektif) bir kritere göre optimizasyon yapmak.**
* **Örnek: Bir görselin "sanatsal kalitesi" kullanıcı tarafından puanlanır.**
* **Amaç fonksiyonu, kullanıcı değerlendirmesine dayanır (otomatik değil!)**

**🧠 Metaheuristik Algoritmalar**

**🧾 Terminoloji:**

* Heuristic (Sezgisel): Yunanca "bulmak, keşfetmek" anlamına gelir.
* Meta: Yunanca "öte, daha yüksek seviye" anlamındadır.

**💡 Metaheuristiklerin Temel Fikirleri:**

1. **Aday çözümler üzerinde yinelemeli iyileştirmeler yaparlar.**
2. Sezgisel yöntemleri "akıllı" şekilde kullanarak:
   * Arama uzayını **keşfederler** (exploration)
   * Önceki aramalardan öğrenilen bilgileri **kullanırlar** (exploitation)
3. **Genel algoritmalardır**  
   Aynı yöntem, farklı problemlere küçük değişikliklerle uygulanabilir.
4. Genellikle **rastlantısallık (stochastic)** içerir.  
   Rastgelelik, problem hakkında tam bilgi olmamasını telafi etmek içindir.

**✅ Avantajlar:**

* Sadece **amaç fonksiyonu** değerine ihtiyaç duyarlar (fonksiyon **sürekli veya türevlenebilir olmak zorunda değildir** → **kara kutu optimizasyon**).
* Uygun şekilde keşif yapılırsa **küresel optimum** bulunabilir.

**❌ Dezavantajlar:**

* **Kuramsal garantiler çok azdır.**  
  (Yaklaşım kalitesi ya da yakınsama garantisi matematiksel olarak zayıftır.)
* Tasarımları genellikle doğadan veya sosyal etkileşimlerden **esinlenerek sezgisel** şekilde yapılır.

**🔍 Arama (Search) Şemalarına Göre Sınıflandırma:**

**A. Arama Şeması:**

* **Trajectory-based:** Tek bir aday çözüm üzerinden ilerler.
* **Population-based:** Aynı anda birçok çözümle çalışır.

**B. Arama Alanı:**

* **Lokal:** Çevresel alanı araştırır.
* **Global:** Tüm uzayı taramaya çalışır.

**C. Metafor Kaynağına Göre:**

* **Biyolojik Evrim:** Genetik Algoritmalar
* **Sosyal Etkileşim:** Sürü Zekâsı (Swarm Intelligence)
* **Fizik:** Tavlama Benzeşimi (Simulated Annealing)
* **Diğer:** Kimyasal süreçler, kültürel modeller vb.

**🌐 Amaç:**

* Rastgele başlatılmış bir popülasyon üzerinden,
* Sürekli **keşif (exploration)** ve **sömürü (exploitation)** işlemleri yaparak,
* Kaliteli çözümlere ulaşmak.

Lecture 2

**Yörünge Tabanlı Arama Algoritmaları (I)**

* **Motivasyon:** Yerel ve küresel optimizasyon
* **Yerel arama algoritmalarının genel yapısı**
* **Yerel Arama Deterministik Yöntemleri:**  
  – Pattern Search  
  – Nelder Mead
* **Yerel Arama Rastgele Yöntemleri:**  
  – Matyas  
  – Solis-Wets
* **Küresel arama için Meta-sezgisel Yöntemler:**  
  – Rastgele başlatmalarla yerel arama  
  – Tekrarlanan yerel arama

metin, ekran görüntüsü, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

**Ayrık Arama Uzayı:**

* Bir elemanın komşuluğu sonlu bir kümedir ve tamamen keşfedilebilir.
* **Özel durum (permütasyon benzeri çözümler):**  
    
  metin, yazı tipi, ekran görüntüsü, beyaz içeren bir resim

  Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

**Sürekli Arama Uzayı:**

a) Amaç fonksiyonu türevlenebilir ise – arama yönü, amaç fonksiyonundaki değişikliklere göre belirlenir  
→ Artış (minimizasyon için) veya azalış (maksimizasyon için) yönü  
– Gradyan yöntemi (birinci türevler → birinci derece yöntemler)  
– Newton benzeri yöntemler (ikinci türevler → ikinci derece yöntemler)

b) Amaç fonksiyonu türevlenemez (veya süreksiz) ise  
– Doğrudan arama yöntemleri (örnek: Nelder Mead)  
– Küçük rastgele bozmalara dayalı yöntemler  
(türev kullanılmaz → sıfırıncı derece yöntemler)

**Yerel Arama: Genel Yapı**

**Gösterimler:**  
S – arama alanı  
f – amaç fonksiyonu  
S\* – yerel/küresel optimum kümesi  
s = (s₁, s₂, ..., sₙ) : S'nin bir elemanı / konfigürasyon / aday çözüm  
s\* = şu ana kadar keşfedilen en iyi eleman  
s\* = optimal çözüm

**metin, ekran görüntüsü, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.**

**LECTURE5**

**Evrim Stratejileri**

**Özellikleri (Particularities)**

Evrim stratejileri, sürekli (continuous) optimizasyon problemlerini çözmek için kullanılan evrimsel tekniklerdir.

**Tarihçe:** İlk evrim stratejisi 1964 yılında Berlin Teknik Üniversitesi'nde öğrenciler olan Bienert, Rechenberg ve Schwefel tarafından esnek bir boru tasarımı amacıyla geliştirilmiştir.

**Ana fikirler** [Beyer & Schwefel – *ES: A Comprehensive Introduction*, 2002]:

* Bir aday (birden çok değişken içeren) kullanılır ve bu aday iteratif (yinelemeli) olarak evrimleştirilir.
* Tüm değişkenler aynı anda, çoğunlukla hafifçe ve rastgele şekilde değiştirilir.
* Yeni değişken kümesi, cihazın kalitesini azaltmıyorsa kabul edilir, aksi takdirde eski duruma dönülür.

**Veri kodlaması:** Gerçel sayılarla (float) yapılır (bireyler, amaç fonksiyonunun tanım kümesine ait gerçek değerli vektörlerdir).

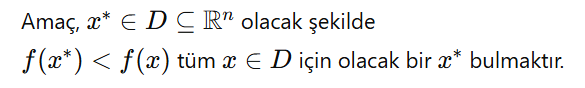
**Ana işlemci:** Mutasyon (parametreli rastgele bozulmaya dayanır).  
**İkincil işlemci:** Rekombinasyon (yeniden kombinasyon).  
**Özellik:** Mutasyon kontrol parametrelerinin kendini uyarlayabilmesi (self-adaptation).

**Genel Yapı (General Structure)**

**Algoritmanın yapısı:**

1. Popülasyonun başlatılması
2. Popülasyonun değerlendirilmesi
3. **TEKRAR:**
   * Rekombinasyon yoluyla yavru (offspring) üret
   * Mutasyon ile yavruyu değiştir
   * Yavruyu değerlendir
   * Hayatta kalanları seç
4. **<durma koşulu sağlanana kadar>**

**Problem (Minimizasyon Problemi):**

****

Popülasyon, tanım kümesinden alınan elemanlardan oluşur (gerçek bileşenli vektörler).

**Not:** Bir yapılandırma (konfigürasyon), fff değeri daha küçükse daha iyidir.

**Kaynakla ilgili kriterler:**

* Nesil sayısı (generations number)
* Fonksiyon değerlendirme sayısı (nfe: number of function evaluations)

**Yakınsama ile ilgili kriterler:**

* F fonksiyonunun değeri

Bu bölümde **Evrim Stratejileri** (Evolution Strategies) içinde **Rekombinasyon** (Recombination) işlemi anlatılmaktadır. Rekombinasyon, ebeveynlerden (parent'lar) yeni bireyler (offspring'ler) üretme işlemidir. Detaylara birlikte bakalım:

**🔁 Rekombinasyon Nedir?**

**Amaç:**  
Birden fazla ebeveynden (ρ tane) yola çıkarak yeni bir birey (offspring) oluşturmak.

**🔸 Rekombinasyon Türleri**

**1. Ara (intermediate) / Konveks Rekombinasyon**

Yeni birey, ebeveynlerin doğrusal (linear) veya konveks (toplam ağırlık 1) kombinasyonu şeklinde oluşturulur.  
Yani:

yazı tipi, beyaz, diyagram, çizgi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Bu, çocuğun her geninin (bileşeninin), ebeveynlerin aynı gendeki değerlerinin ağırlıklı ortalaması olduğu anlamına gelir.

**2. Ayrık (Discrete) Rekombinasyon**

Her bileşen (gen) rastgele bir ebeveynden seçilir.  
Yani, çocuğun genleri, sırayla ve rastgele bir şekilde ebeveynlerin genlerinden alınır.

metin, yazı tipi, el yazısı, beyaz içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

**3. Geometrik Rekombinasyon**

Benzer şekilde konveks/ara birleştirmeye benzer, ancak bileşen bazında uygulanabilir: yazı tipi, el yazısı, metin, çizgi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

**4. Heuristik Rekombinasyon**

Bir ebeveyn daha iyi olduğu sürece, diğerinden "ileriye doğru" bir tahmin yapılır:

metin, ekran görüntüsü, yazı tipi, cebir içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

**5. SBX – Simulated Binary Crossover (Simüle Edilmiş İkili Çaprazlama)**

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.Buradaki β\betaβ, özel bir dağılıma göre üretilen rastgele bir değerdir.

k: bir doğal sayı, **k ne kadar büyükse**, çocuklar ebeveynlerine **daha yakın** olur.

**🔍 Özetle:**

* Rekombinasyon, genetik bilgiyi birleştirerek yeni bireyler üretir.
* Ara/ayrık/geometrik/heuristik gibi farklı yaklaşımlar vardır.
* SBX, gerçek sayılarla çalışan ama binary çaprazlamayı taklit eden güçlü bir tekniktir.

**🔹 Mutasyon: Temel Fikir**

**Türkçesi:**  
Her bireyin bileşenlerine rastgele bir vektör eklenerek popülasyondaki her eleman bozulur (perturb edilir).

Formül:

x′=x+z

(z: rastgele vekto¨r, ortalaması 0, kovaryans matrisi C)x' = x + z \quad \text{(z: rastgele vektör, ortalaması 0, kovaryans matrisi C)}x′=x+z(z: rastgele vekto¨r, ortalaması 0, kovaryans matrisi C)

**Ne anlatıyor:**  
Mutasyon, her çözümün üzerine küçük bir rastgele değişiklik ekleyerek yeni çözümler üretir. Genetik algoritmalardaki gibi büyük değişimlerle değil, küçük değişimlere öncelik verir.

**🔹 Mutasyon Varyantları (1): Bağımsız ve Aynı Dağılımlı Bileşenler**

**Türkçesi:**

* Her bileşen bağımsız ve aynı dağılıma sahiptir.
* Örnekler:
  + a) Her bileşen, [-s, s] aralığında rastgele değer (uniform dağılım)
  + b) Her bileşen, N(0, s) normal dağılımından gelir

**Kovaryans matrisi:**

yazı tipi, el yazısı, metin, hat sanatı, kaligrafi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Burada s tek kontrol parametresidir.

**🔹 Mutasyon Varyantları (2): Bağımsız ve Farklı Dağılımlı Bileşenler**

**Türkçesi:**

* Her bileşen yine bağımsızdır ama farklı dağılımlara sahiptir.
* Örnekler:
  + a) Bileşen zi, [-si, si] aralığında uniform dağılımlı
  + b) Bileşen zi, N(0, si) dağılımlı

**Kovaryans matrisi:**

yazı tipi, metin, beyaz, hat sanatı, kaligrafi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

**Ne anlatıyor:**  
Her boyutun farklı büyüklükte bozulmasına izin verir. Her bileşen ayrı bir s\_i ile kontrol edilir.

**🔹 Mutasyon Varyantları (3): Bağımlı Bileşenler**

**Türkçesi:**

* Bileşenler birbirine bağımlıdır.
* Örnek:

z∼N(0,C)

(C genel kovaryans matrisi)

**Kontrol parametreleri:**

* Diagonal elemanlar: s1, s2, ..., sn (mutasyon adımları)
* Diagonal olmayanlar: a\_ij (rotasyon açısı)

**c\_ij hesaplama:**



**Ne anlatıyor:**  
Bu yöntemde boyutlar arası etkileşim de hesaba katılır. Daha karmaşık ama daha gerçekçi bozulmalar yapılabilir.

**🔹 Mutasyonda Kontrol Parametresi Seçimi**

**Türkçesi:**

* s küçükse → küçük bozulmalar
* s büyükse → büyük bozulmalar  
  Bu parametrenin nasıl seçileceği önemlidir.

**Çözümler:**

* Heuristik yöntemler (örneğin 1/5 kuralı)
* Kendini uyarlayan (self-adaptive) yaklaşımlar

**🔹 1/5 Kuralı (Rechenberg)**

**Türkçesi:**

* Mutasyon başarılıysa (yeni çözüm daha iyiyse), başarı oranı ps hesaplanır:

metin, yazı tipi, ekran görüntüsü, çizgi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Kural:

* + Eğer ps > 1/5: s artırılır
  + Eğer ps < 1/5: s azaltılır
* Bu kural, yalnızca bir aday içeren Evrim Stratejileri (ES - Evolution Strategies) için önerilmiştir; birden fazla adaydan oluşan popülasyonlar için doğrudan uygulanamaz.

**Ne anlatıyor:**  
Bu sezgisel kuralla, s otomatik olarak ayarlanır. Optimum değer yaklaşık c = 0.817 önerilir.

**🔹 Kendini Uyarlayan (Self-Adaptive) Mutasyon**

**Türkçesi:**

* Bireyler artık sadece çözüm değil, aynı zamanda mutasyon parametrelerini de içerir:

metin, yazı tipi, el yazısı, hat sanatı, kaligrafi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

* Bu parametreler de mutasyona ve rekombinasyona uğrar.

**Örnek formül:**

metin, yazı tipi, makbuz, ekran görüntüsü içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

**Ne anlatıyor:**  
Mutasyon parametreleri de evrimsel süreçte değişerek optimize edilir. Böylece daha başarılı bireylerin parametreleri gelecek nesillere aktarılır.

* **Kontrol parametrelerine karşılık gelen bileşenleri değiştir**  
  (örneğin: mutasyon şiddetini belirleyen s gibi parametreler)
* **Karar değişkenlerine karşılık gelen bileşenleri değiştir**  
  (yani, problemin çözümünü temsil eden x değerlerini)

**Örnek: Bağımsız bozulmalar (independent perturbations) durumu:**

Her bir çözüm bireyinin şu şekilde güncellendiği varsayılır:

* x: Karar değişkenleri vektörü
* s: Mutasyon şiddetini belirleyen parametreler (her bileşen için bir si)

**🔹 Michalewicz'in Önerdiği Mutasyon (1996)**

**Türkçesi:**  
Zamanla azalan bozulma miktarı önerir.

Formül:

metin, yazı tipi, el yazısı, hat sanatı, kaligrafi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

* ai,bia\_i, b\_iai​,bi​: ilgili değişkenin alt ve üst sınırları
* u∈(0,1)u \in (0, 1)u∈(0,1): rastgele sayı
* ttt: güncel iterasyon
* TTT: toplam iterasyon sayısı

**Ne anlatıyor:**  
Zaman geçtikçe (t arttıkça), yapılan bozulmalar küçülür. Yani algoritma daha dikkatli ince ayar yapmaya başlar (küçük adımlarla yerel optimumu bulmaya çalışır).

**Hayatta Kalanların Seçimi (Survivors Selection)**

**Varyantlar:**

* **(μ, λ) ve (μ + λ)**
  + μ ebeveynden λ > μ sayıda yavru (offspring) üretilir ve bunlar arasından en iyi μ tanesi hayatta kalır.
  + μ ebeveynden λ yavru üretilir ve **ebeveynler ile yavruların birleşiminden** en iyi μ tanesi seçilir (**truncation selection** denir).  
    Bu, elitist bir seçimdir (popülasyondaki en iyi birey korunur).

**Not:** Eğer ebeveyn sayısı ρ ise, bu durumda kullanılan notasyonlar:

* (μ/ρ + λ) ve (μ/ρ, λ)

**Özel Durumlar:**

* **(1+1):** Bir ebeveynden bir yavru üretilir, en iyi olan seçilir.
* **(1+λ):** Bir ebeveynden birkaç yavru üretilir, en iyi birey seçilir.
* **(μ+1):** μ bireyden bir yavru üretilir, bu yavru popülasyondaki en kötü bireyden daha iyiyse yer değiştirir.

**Stratejiler:**

**Generasyonel (Senkron) Strateji:**

* Her nesilde **tamamen yeni bir yavru popülasyonu** oluşturulur.
* Seçim, sadece yavrulara veya ebeveyn+yavru birleşimine uygulanabilir.
* Bu senkron bir süreçtir.

**Sürekli (Asenkron) Strateji (Steady State):**

* Her adımda **yalnızca bir yavru birey** üretilir.
* Bu yavru yeterince iyiyse popülasyona dahil edilir.
* Bu, asenkron bir süreçtir.

**ES Varyantları (Evolution Strategies Çeşitleri)**

**Stratejiler:**

* Notasyon:  
  **(μ, λ, ρ, k)** stratejisi
  + Her bireyin sınırlı bir ömrü vardır (**k nesil** boyunca yaşar).
  + **Yeniden birleşim (recombination)**, **ρ ebeveyne** dayanır.
  + **Hızlı evrim stratejileri (Fast Evolution Strategies):**
    - Bozulmalar (perturbation), **Cauchy dağılımı**na göre yapılır.

**ES Davranışının Analizi**

**Değerlendirme Kriterleri:**

1. **Etkililik (Effectiveness):**
   * Belirli sayıda değerlendirmeden (**nfe – number of function evaluations**) sonra elde edilen **amaç fonksiyonu değeri**.
2. **Başarı Oranı (Success Ratio):**
   * Algoritmanın hedefe ulaştığı çalıştırma sayısının, toplam çalıştırma sayısına oranı.
3. **Verimlilik (Efficiency):**
   * Amaç fonksiyonu **istenilen doğrulukla** hedefe ulaşana kadar gereken **değerlendirme sayısı**.

**Özet (Summary)**

| **Kavram** | **Açıklama (İngilizce)** | **Türkçesi** |
| --- | --- | --- |
| **Encoding** | Real vectors | Gerçek sayı vektörleri ile kodlama |
| **Recombination** | Discrete or intermediate | Ayrık (discrete) veya ara (intermediate) birleşim |
| **Mutation** | Random additive perturbation (uniform, Gaussian, Cauchy) | Rastgele ekleme bozulması (uniform, Gauss, Cauchy dağılımlı) |
| **Parents selection** | Uniformly random | Ebeveynlerin rastgele ve eşit olasılıkla seçilmesi |
| **Survivors selection** | (μ,λ) or (μ+λ) | (μ,λ) ya da (μ+λ) seçme stratejisi |
| **Particularity** | Self-adaptive mutation parameters | Kendi kendine uyarlanabilen mutasyon parametreleri |

**1. Encoding (Kodlama)**

* **Ne anlatılıyor?**  
  Bireyler (çözümler), **gerçek sayılarla (float)** temsil edilir.  
  Örneğin: bir çözüm x=[1.2,3.5,−0.8]x = [1.2, 3.5, -0.8]x=[1.2,3.5,−0.8] gibi bir vektördür.
* **Neden önemli?**  
  ES sürekli (gerçek değerli) problemleri çözmek için tasarlanmıştır.

**2. Recombination (Yeniden birleşim)**

* **Ne anlatılıyor?**  
  Yeni birey (çocuk) oluşturmak için **ebeveynlerin** özellikleri karıştırılır.
  + **Discrete (ayrık):** Her bileşen rastgele bir ebeveynden alınır.
  + **Intermediate (ara):** Bileşenler aritmetik ortalama gibi birleştirilir.
* **Neden önemli?**  
  Çeşitliliği artırır ve daha iyi çözümlere ulaşma şansı verir.

**3. Mutation (Mutasyon)**

* **Ne anlatılıyor?**  
  Bireyler küçük **rastgele bozulmalarla** değişir.
  + **Uniform:** Sabit aralıkta rastgele sayı eklenir.
  + **Gaussian:** Ortalama 0, sapması s olan normal dağılımdan gelen sayı eklenir.
  + **Cauchy:** Daha uç değerlere izin veren dağılımdan bozulma.
* **Neden önemli?**  
  Arama alanında keşif yapılmasını sağlar.

**4. Parents selection (Ebeveyn seçimi)**

* **Ne anlatılıyor?**  
  Ebeveynler **rastgele** seçilir.
* **Neden önemli?**  
  Basit ama etkili bir seçme yöntemidir; daha karmaşık seçim stratejileri yoktur.

**5. Survivors selection (Hayatta kalanların seçimi)**

* **Ne anlatılıyor?**  
  Yeni nesilde kimlerin kalacağı belirlenir:
  + **(μ, λ):** Sadece çocuklar arasından en iyiler seçilir (elitizm yok).
  + **(μ + λ):** Ebeveynler ve çocuklar birlikte değerlendirilir (elitizm vardır).
* **Neden önemli?**  
  Seçim stratejisi, algoritmanın ne kadar hızlı ve dengeli evrileceğini etkiler.

**6. Particularity (Özgünlük)**

* **Ne anlatılıyor?**  
  Mutasyonun kontrol parametreleri (**örneğin s**) bireylerle birlikte evrilir (self-adaptation).
* **Neden önemli?**  
  Algoritmanın otomatik olarak **kendi ayarlarını optimize etmesini** sağlar.

**Özetle:**

Bu yapı, evrim stratejilerinin nasıl çalıştığını (bireylerin nasıl üretildiği, değiştirildiği ve seçildiği) anlatır. Özellikle sürekli (gerçek sayılarla ifade edilen) optimizasyon problemlerinde etkilidir. Kendini uyarlayan mutasyon parametreleri, algoritmanın çevresine göre evrilmesine olanak tanır.

LECTURE5

**Evrimsel Programlama ve Genetik Programlama**

**Slogan:**

“Bilgisayarlar, açıkça programlanmadan sorunları nasıl çözmeyi öğrenebilir? Başka bir deyişle, bilgisayarlara ne yapılması gerektiği nasıl öğretilebilir ama tam olarak nasıl yapılacağı söylenmeden?”

– **Arthur Samuel’a atfedilir, 1959**

**Kökenleri:**

* **L. Fogel (1960):**  
  Doğal evrimden esinlenerek otomatik olarak zeki davranışlar sergileyen sistemler üreten yöntemlerin geliştirilmesi.
* **D. Fogel (1990):**  
  Son 20 yılda evrimsel programlama, genel zeki davranışları taklit etmekten çok, **problemleri çözmeye (optimizasyon ve tasarım)** odaklanmaya başlamıştır.

**Özellikler:**

* Çeşitli kodlama biçimleri vardır (örneğin: gerçek sayı vektörleri, durum diyagramları, yapay sinir ağı yapıları).
* **Sadece mutasyon (değişim) kullanılır**, çaprazlama (recombination) kullanılmaz.
* Güncel versiyonlar: **Kendini uyarlayan (self-adaptive)**

**Geleneksel (ilk) yönelim:**

* **Tahmin yeteneğine sahip sistemler geliştirme** (örneğin sonlu durum makineleri).
* Bu sistemlerin **başarımı**, davranışları analiz edilerek (yani tahmin yeteneğine göre) ölçülür.
* "Fitness" (uygunluk), sistemin davranış kalitesini temsil eder.

**Sonlu Durum Makinesi (FSM - Finite State Machine):**

Bir FSM şu şekilde tanımlanır:

**FSM = (S, I, O, T, s0)**

* **S**: Durum kümesi (örneğin: {even, odd})
* **I**: Girdi alfabesi (örneğin: {0, 1})
* **O**: Çıktı alfabesi (örneğin: {0, 1})
* **T**: Geçiş kuralları (T: S x I → S x O)
* **s0**: Başlangıç durumu

**Basit Bir Test Problemi:**

**Amaç:** Bir FSM tasarlamak. Bu FSM, bir ikili (binary) dizide kaç tane ‘1’ olduğu bilgisini kontrol edecek (tek mi çift mi olduğunu).

**Parite Problemi:**

* Durumlar: S = {even (çift), odd (tek)}
* Girdi: I = {0, 1}
* Çıktı: O = {0, 1}

**FSM çıktısı:**

* **Son durum 0** ise: Dizide **çift sayıda** ‘1’ vardır.
* **Son durum 1** ise: Dizide **tek sayıda** ‘1’ vardır.

**State Diagram = Labeled Directed Graph**

**Durum Diyagramı = Etiketli ve Yönlü Grafik**

metin, diyagram, daire, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

**EP Tasarımı (EP Design):**

**1. Seçim Yap:**

* **S** = Durum kümesi (örneğin: {even, odd})
* **I** = Girdi alfabesi (örneğin: {0, 1})
* **O** = Çıktı alfabesi (örneğin: {0, 1})

**2. Popülasyon Başlatma (Population initialization):**

* Rastgele FSM’ler üret. Yani başlangıçta çeşitli yapılar oluşturulur.

**3. Düğümler için Etiket Üret (Generate labels for nodes):**

* Her duruma bir isim verilir (örnek: S1, S2 ya da even, odd gibi).

**4. Okları (arc) oluştur (Generate arcs):**

* Girdilere göre durumlar arasında geçiş okları tanımlanır.

**5. Etiketler oluştur (Generate labels):**

* Her ok için girdi/çıktı etiketleri eklenir.

**Mutasyon Türleri (Mutation):**

Evrimsel programlamada FSM’ler mutasyonlarla geliştirilir. Burada çeşitli mutasyon türleri listelenmiş:

1. **Çıktı Sembolünü Değiştir (Mutation of the output symbol):**
   * Bir geçişin (okun) çıktısı değiştirilir (örnek: 1/1 → 1/0).
2. **Bir Oku Yeniden Yönlendir (Redirect an arc / mutate the target node):**
   * Bir ok, başka bir duruma yönlendirilir. Yani hedef düğüm değiştirilir.
3. **Düğüm Ekle / Sil (Add/eliminate nodes):**
   * FSM’ye yeni durum eklenebilir veya var olan bir durum çıkarılabilir.
4. **Başlangıç Durumunu Değiştir (Change the initial state):**
   * FSM’nin başladığı durum değiştirilebilir.

**🔍 Genel Anlamda Ne Anlatılıyor?**

Bu kısımda, **FSM tasarımı için evrimsel programlama adımları** açıklanıyor. Yani:

* Rastgele FSM’ler üretilir.
* Sonra bu FSM’ler, çeşitli **mutasyonlar** uygulanarak nesiller boyunca geliştirilir.
* Amaç: Belirli bir görevi (örneğin parite kontrolü) doğru ve verimli şekilde yapan FSM’ler üretmektir.

metin, diyagram, ekran görüntüsü, daire içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.z

Bu görselde, **Evrimsel Programlama'da mutasyon işlemi** ile bir FSM'nin (Sonlu Durum Makinesi) nasıl değiştirilebileceği örneklenmiş. Özellikle **"bir okun hedef düğümünü değiştirme (change the target node of an arc)"** adlı bir mutasyon tipi gösteriliyor.

**Sol Taraftaki Şekil – Orijinal FSM:**

* **Durumlar:** even ve odd
* **Geçişler (oklar):**
  + even durumunda:
    - 0 girilirse, yine even’de kalır → Çıkış: 0/0
    - 1 girilirse, odd'a geçer → Çıkış: 1/1
  + odd durumunda:
    - 0 girilirse, yine odd’te kalır → Çıkış: 0/1
    - 1 girilirse, even'e geçer → Çıkış: 1/0

Bu FSM, **1'lerin sayısının tek mi çift mi olduğunu kontrol eden** (parite problemi) düzgün bir örnektir.

**Sağ Taraftaki Şekil – Mutasyon Uygulanmış FSM:**

Bu FSM, **sol FSM'nin bir mutasyona uğramış hali**dir. Spesifik olarak:

* 0/0 etiketli ok **önceden even'dan even'a gidiyordu**.
* Şimdi **odd'dan even'a gidiyor.** (Hedef düğüm değişti)

Bu değişiklik, FSM'nin davranışını etkiler. Artık:

* odd durumunda 0 girilirse, even durumuna geçilir.
* Bu, 1'lerin sayısını doğru sayma mantığını bozabilir, çünkü parite takibi yanlış yapılmış olur.

**Kodlama Özellikleri (Encoding Particularities):**

* **Popülasyon**, içinde **etiketli grafikler (labeled graphs)** bulundurur.  
  Yani her birey (çözüm) bir grafik (düğüm + kenar) şeklinde temsil edilir.

**Temsil Türleri (Types of Representation):**

* **Açık (Explicit)** ve **Kapalı (Implicit)** temsil biçimleri vardır.

**🔷 Açık Temsil (Explicit Representation) Varyantları:**

**✅ 1. Sabit sayıda düğüm + sabit sayıda kenar:**

* Bu durumda **sadece kenar ağırlıkları (weight)** değiştirilebilir.
* Tüm kenar ağırlıkları bir **dizi (array)** içinde tutulur → Bu, ağırlık matrisinin düzleştirilmiş (linearize) hali gibi düşünülebilir.

**✅ 2. Sabit sayıda düğüm + değişken sayıda kenar:**

* Kenarlar yine bir **ağırlık dizisi** ile temsil edilir.
* Silinen kenarlar için özel bir değer (örneğin -1 gibi) kullanılır.

**✅ 3. Değişken sayıda düğüm:**

* Hem **hangi düğümlerin var olduğu**, hem de **kenar ağırlıkları** değişebilir.
* Burada iki yapı kullanılır:
  + **İkili dizi (binary array)** → Her düğümün var olup olmadığını gösterir.
  + **Kenar ağırlıkları listesi**

**🔧 Mutasyon Türleri (Mutation):**

1. **Kenar ağırlıklarını değiştir**
   * Ayrık ağırlıklarda: rastgele yeni değerlerle değiştir.
   * Gerçek sayı ağırlıklarda: küçük rastgele eklemeler yap (örnek: 0.1 artır).
2. **Düğüm varlık bilgilerini değiştir**
   * Rastgele seçilen bazı değerler ters çevrilir (0 → 1, 1 → 0)
   * Böylece düğüm **eklenmiş veya çıkarılmış** olur.

**📌 Not:**

Her mutasyon türü belirli bir olasılıkla uygulanır:

* **p1:** kenar ağırlığı değişimi
* **p2:** kenar ekleme
* **p3:** kenar silme
* **p4:** düğüm ekleme
* **p5:** düğüm silme

Ve bu olasılıkların toplamı 1 olur:  
p1 + p2 + p3 + p4 + p5 = 1

**🔁 Çaprazlama (Crossover):**

**▶️ Sabit yapıya sahip grafiklerde (örneğin sabit uzunlukta dizi):**

* Normal genetik algoritmalardaki gibi **standart çaprazlama** yapılabilir.

**▶️ Rastgele yapılı grafiklerde:**

* **İki alt grafiğin yerleri değiştirilir.**
  + Belirli bir düğüm kümesine ait kenarlar karşılıklı değiştirilir.
  + Örneğin bir ağın bir parçası diğer ağla yer değiştirir.

**🔍 Not – Kapalı (İmplicit) Temsil:**

* Kapalı temsilde, doğrudan grafik değil, **grafiği tanımlayan özet bilgiler** (soyut tanımlar) evrimleştirilir.

**Örnek:**

* Temsil şu şekilde olur:
  + Düğüm sayısı
  + Bağlantı derecesi (bir düğüm ortalama kaç düğüme bağlı)
  + Ağırlıklar için olasılık dağılımı
* Değerlendirme sırasında, bu tanıma göre **gerçek bir grafik oluşturulur** ve bu örnek test edilir.

**✅ Bir Yapılandırmanın Değerlendirilmesi (Evaluation of a configuration):**

* **Simülasyon yapılır:** Verilen bir **test veri kümesi** üzerinde yapılandırma (örneğin FSM ya da sinir ağı) çalıştırılır.
* **Başarı oranına göre fitness değeri verilir:**  
  Yani yapılandırma testlerde ne kadar başarılıysa, **fitness (uygunluk) değeri o kadar yüksek olur.**

**📌 Alanın Güncel Durumu (Current Status in the Field):**

* Günümüzde EP, **hesaplamalı yapıların evrimsel tasarımına** (örneğin **yapay sinir ağları**) yönlendirilmiştir.  
  Yani amaç artık sadece "zeki davranışlar" üretmek değil, **doğrudan işe yarayan yapılar tasarlamak**.

**▶️ İkinci ve Güncel Yönelim (Second / Current Direction):**

Bu yaklaşım, **optimizasyon yöntemlerine** (özellikle **Evrim Stratejilerine (Evolution Strategies)**) benzer.

**Özellikleri:**

1. **Sadece mutasyon operatörü kullanılır**
   * **Çaprazlama (crossover)** **yoktur**.
2. **Mutasyon işlemi rastgele bozulma (perturbation)** şeklindedir:
   * Yeni çözüm şu şekilde elde edilir:

x′=x+N(0,s)x' = x + N(0, s)x′=x+N(0,s)

* + - Burada N(0, s) = ortalaması 0 olan normal (gauss) dağılımdan alınan rastgele bir değer
    - s = standart sapma (ne kadar bozulma olacağı)

1. **s değeri fitness'a bağlıdır:**
   * **Yüksek fitness** → küçük s → küçük değişiklik (çözüm iyiyse fazla oynamaya gerek yok)
   * **Düşük fitness** → büyük s → daha büyük değişiklik (kötüyse büyük değişiklik gerek)
2. **Popülasyon güncellemesi:**
   * Başlangıçta m birey var.
   * Her bireyden mutasyonla m tane **çocuk (child)** üretilir → toplam 2m birey olur.
   * Sonra en iyiler **turnuva seçimi** veya **truncation** (en iyileri al) ile seçilir → yeni nesil.

**🔁 MetaEP – Kendini Uyarlayan Varyantlar (Self-adaptive Variants):**

* **MetaEP** adı verilen sürümler vardır.
* Bunlar, **Evrim Stratejileri**'nde olduğu gibi, **mutasyon şiddetini (s)** otomatik olarak evrimleştirir.
* Yani algoritma sadece çözümleri değil, **çözüm üretme şeklini de** optimize eder.

**📘 MetaEP'de Mutasyon Mekanizması:**

MetaEP, hem çözüm vektörünü (x) hem de **kontrol parametrelerini** (yani mutasyon şiddeti s) **birlikte evrimleştirir.**

Matematiksel olarak şu şekilde gösteriliyor:

metin, yazı tipi, el yazısı, hat sanatı, kaligrafi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

**🧬 Genetik Programlama (GP) Nedir?**

* **Öncüsü:** J. Koza
* **Tanım:** Genetik Programlama, bir problemi çözen **çalışan bir bilgisayar programını otomatik olarak oluşturma** yöntemidir.
* GP, doğrudan "nasıl yapılacağını" yazmadan, sadece "**ne yapılmak istendiği**" bilgisine dayanarak uygun bir program üretir.

**🔍 En Basit Uygulama: Sembolik Regresyon (Symbolic Regression)**

**Genetik Programlama'nın tipik örneği, sembolik regresyon problemidir.**

**🟦 Sayısal (Numerical) Regresyon – Geleneksel Yöntem**

* **Girdi:**  
  (1,3), (2,5), (3,7), (4,9) gibi sayısal veri çiftleri (x, y)
* **Model:**  
  f(x)=ax+bf(x) = ax + bf(x)=ax+b gibi bir fonksiyon
* **Amaç:**  
  a ve b parametrelerini bulmak  
  → Örnek sonuç: a=2,b=1a = 2, b = 1a=2,b=1

🧠 Burada **arama (search)** işlemi, **parametreler uzayında** yapılır.

**🟦 Sembolik (Symbolic) Regresyon – GP Kullanımı**

* **Girdi:**  
  Aynı sayısal veri çiftleri: (1,3), (2,5), (3,7), (4,9)
* **Alfabe:**
  + **Terminal semboller:** sabitler (1, 2, …), değişkenler (x)
  + **Non-terminal (işlemci) semboller:** +, −, \*, / gibi işleçler
* **Amaç:**  
  Girdi ile çıktı arasındaki ilişkiyi tanımlayan **matematiksel ifadeyi otomatik oluşturmak**
* **Çıktı:**  
  2⋅x+12 \cdot x + 12⋅x+1 ya da  
  x+x+1x + x + 1x+x+1 gibi eşdeğer ifadeler

🧠 Burada **arama**, artık parametrelerde değil, **ifadeler uzayında (expression space)** yapılır.

**🎯 Özetle:**

| **Özellik** | **Sayısal Regresyon** | **Sembolik Regresyon (GP)** |
| --- | --- | --- |
| Girdi | (x, y) çiftleri | (x, y) çiftleri |
| Aranan şey | a, b gibi katsayılar | Matematiksel ifade |
| Arama uzayı | Parametre uzayı | İfade uzayı (expression tree) |
| Çıktı | Sayılar (örneğin a=2, b=1) | Fonksiyon (örneğin 2\*x+1) |

**📦 Encoding (Kodlama) – GP’de Birey Temsili**

**🧩 Genetik Programlama’da:**

* Bireyler (çözümler) genellikle **ağaç yapısında** temsil edilir.
* Bu ağaçlar, bir **ifade (expression)** veya **program** olabilir.

**📘 Örnek 1: Aritmetik İfade**

**daire, taslak, diyagram, çizgi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.**

Verilen ifade: a \* b + sin(c)

**🧱 Bileşenler:**

* **Terminal (uç noktalar)**:
  + Değişkenler: a, b, c
  + Sabitler: 1, 2, 3 gibi sayılar
  + Rastgele sabitler (ephemeral random constants)
* **Non-terminal (düğüm)**:
  + İşlem ve fonksiyonlar: +, \*, sin

**📌 Alternatif gösterimler:**

* **Prefix (ön ek / pre-order):** + \* a b sin c
* **Postfix (son ek / post-order):** a b \* c sin +

📘 **Örnek 2: C kodu (program yapısı olarak ifade)**

diyagram, taslak, çizgi, beyaz içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

📌 **Problem:**  
Basit görünen bu kod bile, ağaç olarak temsil edildiğinde **karmaşık** hale gelebilir. Bu, Genetik Programlama’nın zorluklarından biridir.

Özet: terminaller ve nonterminal kümeleri çözülecek probleme bağlı olarak seçilir

**🔄 GP’nin Genel Yapısı (Koza, 2003)**

1. **Terminal ve Non-terminal kümeleri belirlenir:**
   * Problemin yapısına göre seçilir.
2. **Popülasyon oluşturulur.**
3. **Evrimsel işlemler uygulanır:**
   * **Seçim (Selection)**
   * **Çaprazlama (Crossover)** ❗ (veya)
   * **Mutasyon (Mutation)**  
     ⚠️ Not: Genellikle **çaprazlama veya mutasyon** dönüşümlü uygulanır, ikisi birden değil.

**🛠️ Uygulama (Implementation)**

* **Klasik programlama dili:** LISP
  + Çünkü ifadeleri doğal olarak liste yapısıyla temsil eder. (örneğin: (+ (\* a b) (sin c)))
* **Zorluk:**  
  Üretilen her yapı, **sözdizimsel olarak geçerli (syntactically correct)** olmalıdır. Yanlış biçimli yapılar hata verir.

**🌱 Ağaç Oluşturma Fonksiyonu: Generate(T, N, A)**

| **Sembol** | **Açıklama** |
| --- | --- |
| T | Terminal kümesi |
| N | Non-terminal (fonksiyon) kümesi |
| A | Maksimum derinlik (depth) |

**Fonksiyonun Mantığı:**

metin, yazı tipi, ekran görüntüsü, cebir içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Bu işlem, rastgele ama mantıklı bir **ifade ağacı** üretir.

**🎯 Özetle:**

* GP bireyleri, **ağaç yapılarla** temsil edilir.
* Düğümler operatörleri, yapraklar değişken ve sabitleri temsil eder.
* Ağaçlar geçerli matematiksel/programatik yapılar olmalıdır.
* Bu yapılar, rastgele ama tanımlı kurallara göre oluşturulur ve evrimsel olarak geliştirilir.

**🧬 Popülasyon Elemanlarının Diğer Türleri**

Genetik programlamada yalnızca matematiksel ifadeler veya programlar değil, başka yapı türleri de kullanılabilir. İşte bazı örnekler:

**📌 Kullanılabilecek birey türleri:**

* **Karar ağaçları (Decision Trees):**  
  Sınıflandırma ve regresyon problemleri için kullanılır.
* **If-Then kuralları:**  
  Örnek: if temperature > 30 then class = hot
* **Yapay sinir ağları (Neural Networks):**  
  Evrimsel algoritmalarla ağ yapısı ve ağırlıkları öğrenilebilir.
* **Mantıksal ifadeler (Logical Expressions):**  
  Boolean mantık tabanlı ifadeler (örnek: A AND NOT B)
* **Binary Decision Diagrams (BDD):**  
  Boolean fonksiyonlarını verimli temsil eden grafik yapıları.
* **Gramerler (Grammars):**  
  ➤ **Gramatik Evrim (Grammatical Evolution):**  
  Evrimsel algoritmalarla dilbilgisel kurallar (gramer) üzerinden programlar oluşturulur.

**🧪 Fitness (Uygunluk) Hesaplaması**

Her bireyin (örneğin bir ifade ağacı veya programın) **başarısı**, belirli bir test veri seti üzerinden değerlendirilir:

**🧭 Adımlar:**

1. **Genotip (chromosome)** → **Fenotip (çözüm)**:
   * Genetik yapıya karşılık gelen çözüm (örneğin bir matematiksel formül) oluşturulur.
2. **Test veri kümesi** kullanılarak bu çözümün çıktısı hesaplanır.
3. **Hedef değerle karşılaştırılır:**
   * Eğer hesaplanan sonuç, istenen sonuca **yakınsa**, o bireyin **fitness değeri yüksektir**.
   * Amaç: “Tahmin edilen” değer ile “gerçek” değer arasındaki farkı **minimuma indirmek**.



**Crossover:**

İki ebeveyn (ağaç yapıları) bazı alt ağaçları değiş tokuş ederek iki yeni yavru (yine ağaç yapıları) oluştururlar.

diyagram, çizim, taslak, çizgi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.diyagram, taslak, çizim içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Ebeveynlerin ve çocukların önek (prefiks) biçimleri:

metin, yazı tipi, ekran görüntüsü, çizgi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

**Not:**  
Bu, Genetik Algoritmalarda (GAs) kullanılan çaprazlamaya benzer, ancak değiş tokuş edilen kısımların büyüklükleri genellikle farklıdır.

**Mutasyon:**  
Bazı elemanların rastgele değiştirilmesinden oluşur.

* Bir yaprak düğümün (leaf node) sembolünü başka bir terminal sembolle değiştir (sabitler için bu mutasyon, evrim stratejilerindeki mutasyona benzer olabilir).
* Bir yaprak düğümü bir ağaç ile değiştir (büyüme mutasyonu - growing mutation).
* İç düğüme (internal node) karşılık gelen sembolü aynı ariteye (aynı sayıda argümana sahip) sahip başka bir non-terminal ile değiştir.
* Bir alt ağacı bir terminal düğümle değiştir (budama mutasyonu - pruning mutation).

**Not:**  
Mutasyon, rastgele oluşturulmuş bir elemanla çaprazlama (crossover) yapılarak da uygulanabilir.

**Mutasyon:**  
Bazı elemanların rastgele değiştirilmesinden oluşur.

diyagram, çizim, taslak içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

**Bloat Problemi:**

Popülasyonda karmaşık yapılar baskın hale gelir.

**Çözümler:**

* Yapı karmaşıklığı için bir eşik değeri kullan (örneğin, ağaç derinliği) ve bu eşikten daha büyük (daha derin) yapıları reddet.
* Uygunluk (fitness) hesaplamasında yapı karmaşıklığına bağlı bir ceza terimi kullan; bu terim karmaşık yapıları cezalandırır.

**Açıklama:**

Bloat problemi, genetik programlamada çözümlerin gereğinden fazla karmaşık ve büyük hale gelmesi durumudur. Bu durum hem hesaplama maliyetini artırır hem de gereksiz karmaşıklığa yol açar. Yukarıdaki çözümlerle, çok büyük veya karmaşık yapıların popülasyonda hakim olması önlenir. Böylece daha yalın ve etkili çözümler bulunabilir.

**Linear Genetic Programming [Brameier, Banzhaf, 2003]**

****

**Özellikleri:**

* Programları satır dizisi olarak (örneğin, assembler dili gibi) oluşturmak için kullanılır.
* İşlemler kayıt (register) üzerinde gerçekleştirilir.
* Komutlar: if ve goto içerir.
* Yorum satırları, sonuca etki etmeyen işlem adımlarını temsil eder (DNA’daki kodlamayan bölgeler yani intronlara benzer).
* Crossover: farklı uzunluktaki kromozomlara uyarlanmış tek noktalı çaprazlama çeşidini kullanır (program bir kromozomdur, her satır bir gendir).

**Açıklama:**

Linear Genetic Programming (Doğrusal Genetik Programlama), programları klasik ağaç yapısı yerine satır satır (linear) kod olarak temsil eder. Bu yöntem genellikle işlemci kayıtları üzerinde çalışır ve koşullu dallanma (if, goto) gibi kontrol yapıları içerir. Yorum satırları ise genetik programlama sürecinde işe yaramayan ama yapıyı koruyan kod parçalarıdır.  
Çaprazlama işlemi, genlerin (satırların) farklı uzunluktaki programlarda bile etkili şekilde karışmasını sağlar.

**GEP (Gene Expression Programming - Gen İfade Programlama) (C. Ferreira, 2001)**

**Kromozom:**

* Sabit uzunlukta birkaç gen içerir.
* Her gen bir baş (head) ve bir kuyruktan (tail) oluşur.
* Baş kısmı hhh sembol içerir (hem terminal hem de non-terminal semboller).
* Kuyruk kısmı yalnızca terminal semboller içerir.
* Kuyruktaki eleman sayısı h×(n−1)+1h \times (n - 1) + 1h×(n−1)+1’dir; burada nnn, baş kısmında görünen fonksiyonların veya operatörlerin maksimum aritesi (yani argüman sayısı)dır.

**Örnek:**  
Bir gen uzunluğu 13 olan bir gen:

13=6+(6×(2−1)+1)=h+(h×(n−1)+1)

Gen örneği:

+ \* sin a b c

b a c c b a a

* İlk 6 eleman ifade ile ilişkilidir (ağaç yapısının genişlik öncelikli (breadth first) taraması).
* Diğer tüm elemanlar terminaldir ve fenotip dönüşümünde kullanılmazlar (genotipte var ama fenotipte etkisizdirler).

**Açıklama:**

Gene Expression Programming (GEP), genetik programlama türlerinden biridir. Burada kromozomlar sabit uzunlukta genlerden oluşur ve her genin bir "baş" ve bir "kuyruk" kısmı vardır. Baş kısmı fonksiyonları ve değişkenleri içerebilirken, kuyruk sadece değişkenler (terminal semboller) içerir. Bu yapı sayesinde, genotip ile fenotip (programın gerçek ifadesi) arasında esnek bir dönüşüm yapılabilir.

Örneğin, yukarıdaki genin başındaki semboller gerçek ifadeyi oluştururken, kuyruk kısmı genetik operatörlerin (örneğin mutasyon) işlemesini sağlar ama fenotipte görünmeyebilir

GEP: Baş (head) kısmını kuyruktaki (tail) sembollerle genişleterek sözdizimsel olarak doğru ifadeler üretmeye olanak tanır.

taslak, diyagram içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

**GEP:**

İki gen içeren bir kromozom örneği:

+ \* sin a b c

b a c c b a a

\* \* / a b c

b a c c b a a

**Açıklama:**  
Kromozoma karşılık gelen fenotip (ifade), iki gene karşılık gelen bileşenlerin birleştirilmesiyle elde edilir

taslak, beyaz, diyagram, daire içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

**Uygulamalar:**

* Veriden modeller çıkarma (örneğin, tahmin modelleri)
* Veriden kurallar çıkarma
* Elektrik devreleri tasarımı
* Dayanıklı sistemlerin sentezi
* Evrimleşebilir donanım (Evolvable hardware)
* Paralel uygulamalar tasarımı
* Hücresel otomata tasarımı
* Sinyal/görüntü işleme filtreleri tasarımı
* Çok ajanlı stratejilerin oluşturulması
* Oyun stratejilerinin oluşturulması
* Kuantum algoritmalarının oluşturulması

**LECTURE 7**

**Biyo-ilhamlı Meta-sezgisel Yöntemler (Bio-inspired Metaheuristics)**

-Sürü Zekası (Swarm Intelligence):

* Karınca Kolonisi Optimizasyonu (Ant Colony Optimization - ACO)
* Parçacık Sürü Optimizasyonu (Particle Swarm Optimization - PSO)
* Yapay Arı Kolonisi (Artificial Bees Colony - ABC)

-Diğer Biyo-ilhamlı Meta-sezgisel Yöntemler (Other Bio-inspired Metaheuristics)

**Sürü Zekâsı (Swarm Intelligence)**

* Sürü zekâsı, **bazı kendini organize eden sistemlerin kolektif davranışlarından ilham alan zeki teknikler bütünüdür**.
* Bu terim, **1989 yılında Gerardo Beni ve Jing Wang** tarafından **robotlar için kontrol sistemleri** bağlamında ortaya atılmıştır.
* Sürü zekâsı teknikleri, şu özelliklere sahip **ajan kümelerini (etmenleri)** kullanır:
  + **Basit işleyiş kuralları**
  + **Yerel etkileşimler**
  + **Merkezi bir kontrolün olmaması**
* Bu özelliklere sahip doğal sistemler şunlardır:
  + **Karınca kolonileri**
  + **Arı kolonileri**
  + **Kuş sürüleri**
  + **Balık sürüleri**
* Bu tür doğal sistemler, **optimizasyon ve veri analizi problemlerinin çözümünde kullanılan tekniklere model oluşturmaktadır**.

**Karınca Sistemleri (Ant Systems)**

**İlham Kaynağı:**

Karınca kolonilerinin aşağıdaki davranışları:

* **Yiyecek ararken gösterdikleri davranış** → **Optimizasyon problemi çözme**: Yani, yiyecek kaynağı ile yuva arasındaki **en kısa yolu bulma** problemi.
* **Yuvalarını organize etme davranışları** → **Veri kümeleme problemi çözme**: Yani, **benzerliklerine göre veri öğelerini düzenleme** problemi.

**Temel Unsurlar:**

* Karıncalar, **feromon** adı verilen bazı kimyasal maddeleri kullanarak **dolaylı yoldan iletişim kurarlar**; bu iletişim sürecine **stigmerji** denir (özellikle optimizasyon problemlerinin çözümünde faydalıdır).
* Aynı yuvaya ait karıncalar birbirlerini **koku yoluyla tanırlar** (bu, veri kümeleme için kullanışlıdır).

**Stigmerji Örneği: Çift Köprü Deneyi (Deneubourg, 1990)**

* Karınca türü: **Arjantin karıncası**
* Yiyecek ile yuva arasında iki farklı geçiş yolu vardır.
* **Başlangıçta**, karıncalar **rastgele** bir yol seçer.
* Karınca, yiyecekten yuvaya dönerken **yol boyunca feromon bırakır**.
* **Daha kısa yol**, daha kısa sürede **daha yüksek feromon yoğunluğuna** sahip olur.



**Stigmerji Mekanizmasının Devamı:**

* İki yol arasında **feromon yoğunluğu farklı** olursa, karıncalar **yoğunluğu yüksek olan yolu** tercih eder.
* Zamanla, **daha çok karınca** bu kısa ve feromon açısından zengin yolu seçer.
* Bu da daha fazla feromon birikmesine yol açar.  
  ➤ Bu durum, **pozitif geri besleme** (positive feedback) örneğidir.

**Feromonların Buharlaşması:**

* Feromon yoğunluğu zamanla **buharlaşma nedeniyle azalabilir**.
* Buharlaşma, özellikle **dinamik (değişken) ortamlarda** faydalıdır; çünkü ortamda değişiklikler olduğunda sistem eski çözüme takılı kalmaz.

**Bir Optimizasyon Problemini Çözmek – Karınca Kolonisi Optimizasyonu (Ant Colony Optimization - ACO)**

**Temel Fikir:**

Problem çözümünün inşasında, **stigmerji mekanizması ile bilgi alışverişi yapan yapay karınca (ajan) grupları** kullanılır. Bu karıncalar, çözümün kalitesiyle ilgili bilgileri paylaşarak çözüm üretir.

**Örnek Problem: Gezgin Satıcı Problemi (Travelling Salesman Problem - TSP)**

* **Girdi:**  
  Etiketlenmiş bir grafik (şehirlere karşılık gelen düğümler ve aralarındaki bağlantıların maliyetleri belirtilmiş).
* **Çıktı:**  
  Tüm şehirlerin **en düşük toplam maliyetle ziyaret edildiği bir sıralama**.

**Karınca Kolonisi Optimizasyonu (Ant Colony Optimization - ACO)**

**Gezgin Satıcı Problemi için ACO [Dorigo, 1992]:**

* Belirli sayıda karınca, **tekrarlamalı (iteratif) bir süreçte** yer alır.
* Her iterasyonda, **her bir karınca** grafikteki tüm düğümleri ziyaret ederek bir **rota oluşturur**.
* Rota oluştururken her karınca aşağıdaki kuralları izler:
  + **Aynı düğümü iki kez ziyaret etmez.**
  + **Bir sonraki düğüme gitme kararı**, iki faktöre dayalı olarak **olasılıksal (probabilistik)** biçimde verilir:
    - O kenarın (yolun) **maliyeti**
    - O kenar üzerindeki **feromon yoğunluğu**
* Tüm karıncalar rotalarını tamamladıktan sonra:
  + Feromonlar, **buharlaşma süreci** ile güncellenir (bir kısmı azalır),
  + **Toplam maliyeti düşük olan rotalar** üzerinde bulunan kenarlara **ödül olarak daha fazla feromon eklenir**.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

**Gösterimler:**

* tmax: Maksimum iterasyon sayısı
* m: Karınca (ajan) sayısı
* i(k): Karınca k tarafından oluşturulan tur
* p: Düğüm sırası
* P: Geçiş olasılığı
* tau: Feromon yoğunluğu

**ACO'nun Varyantları (İlk Varyantlar):**

Not: Varyantlar, **geçiş olasılıklarının hesaplanması** ve **feromon güncelleme kuralları** bakımından birbirinden farklılık gösterir

**Gezgin Satıcı Problemi için Orijinal ACO Varyantı (AS – Ant System)**

* **Çözüm Temsili:**  
  (i₁, i₂, …, iₙ) → Düğüm indekslerinin bir **permütasyonu** (her şehir bir kez ziyaret edilir).

**Geçiş Olasılıkları:**

* Karınca k, iterasyon t'de i düğümünden j düğümüne geçerken:

N(i, k): Karınca k tarafından henüz ziyaret edilmemiş ve i düğümüne bağlı olan düğümlerin listesi

yazı tipi, çizgi, metin, el yazısı içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

Burada:

* τij(t)\tau\_{ij}(t)τij​(t): i ile j arasındaki kenar üzerindeki **feromon miktarı**
* ηij\eta\_{ij}ηij​: i ile j arasındaki kenarın **sezgisel değeri** (genellikle 1/mesafe)
* α\alphaα: Feromonun etkisini belirleyen katsayı
* β\betaβ: Sezgisel bilginin etkisini belirleyen katsay

**Orijinal Varyant (AS - Ant Systems)**

**Feromon Güncelleme (Her İterasyonun Sonunda):**

Feromon miktarı aşağıdaki formülle güncellenir:

metin, yazı tipi, el yazısı, beyaz içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

metin, yazı tipi, ekran görüntüsü, beyaz içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.metin, yazı tipi, el yazısı, hat sanatı, kaligrafi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

**🔍 Not (Remark):**

Başka bir varyant, feromon seviyelerini sadece **son iterasyonda oluşturulan en iyi turun** (T\*) **uzunluğu (L\*)** kullanılarak ayarlamaya dayanabilir:

\***Özetle:**

* **Klasik yaklaşım**: Tüm karıncaların çözümlerine göre feromon ekler.
* **Alternatif yaklaşım**: Sadece **en iyi çözümü** üreten karıncanın turunu dikkate alır.

**1. Max-Min Ant System (MMAS)**

* **Feromon konsantrasyonu** (yoğunluğu), belirli bir aralıkta sınırlandırılmıştır.  
  👉 Bu, feromonun **aşırı artmasını** ya da **tamamen yok olmasını** engeller.
* Feromon miktarı **sadece** önceki iterasyonda bulunan **en iyi tura ait kenarlar** için artırılır.  
  👉 (Bir önceki slaytta bahsedilen "best tour" temelli güncelleme yöntemi uygulanır.)

**2. Ant Colony System (ACS)**

* MMAS'taki gibi **global feromon güncellemesine** ek olarak:  
  ➕ Her bir karınca, **bir kenarı ziyaret ettiğinde anlık (lokal) bir feromon güncellemesi** de yapar.

Bu sayede, algoritma **daha fazla çeşitlilik sağlar** ve **erken yoğunlaşmayı** önler.

**Feromon Konsantrasyonunun Başlangıç Değeri:**

* Tüm kenarlar için **başlangıçta** feromon değeri genellikle **sabit bir küçük sayı** olarak atanır

**🐜 Karınca Sistemleri (Ant Systems)**

**🔧 Gerçek Dünya Problemlerinde Uygulamaları:**

* **Yönlendirme problemleri**:
  + Telekomünikasyon ağlarında veya araçların rota planlamasında kullanılır.
* **Dinamik optimizasyon problemleri**:
  + Zamanla değişen problemlerde çözüm bulmak için uygundur.
* **Görev zamanlama (task scheduling)**:
  + Görevlerin en verimli sırayla planlanması

**🧠 AntClust – Karınca Kümeleme Algoritması [Labroche, 2002]**

**İlham kaynağı:**  
Karıncaların “koloni kapanması (colonial closure)” davranışı:  
Kendi yuvasına ait olmayan karıncaları kimyasal kokularla ayırt etme yeteneği.

**🧪 Karınca Etkileşimi:**

* Karıncalar “buluşmalar” yoluyla birbirlerini karşılaştırır.
* Bu karşılaştırmalarda kokular (odors) karşılaştırılır.

**🧬 AntClust Terimleri:**

| **Terim** | **Anlamı** |
| --- | --- |
| **Ant** | Yapay karınca (bir veriyle temsil edilir) |
| **Nest** | Benzer kokulara sahip karıncalar (aynı kümeye ait) |
| **Odor Template** | Karıncanın kokusunu temsil eden yapı |
| **Meeting** | İki karıncanın karşılaştırılması |
| **Nest Creation** | Yeni küme (yeni yuva) oluşturma |
| **Migration** | Bir karıncanın başka kümeye geçmesi |
| **Elimination** | Bir karıncanın kümeden çıkarılması |

**📂 Kümeleme Süreci (Clustering Process)**

| **Süreç Adımı** | **Açıklama** |
| --- | --- |
| **Veri** | m adet veri (her biri bir karınca) |
| **Cluster** | Benzer verilerin oluşturduğu sınıflar |
| **Similarity threshold** | Benzerlik eşiği |
| **Comparison** | İki verinin karşılaştırılması |
| **Cluster initiation** | Yeni bir kümenin başlatılması |
| **Data transfer** | Bir verinin kümeden başka kümeye taşınması |
| **Data elimination** | Verinin kümeden çıkarılması (uyumsuzsa) |

**🧱 AntClust Karakteristikleri**

Her karınca aşağıdaki özelliklere sahiptir:

| **Parametre** | **Açıklama** |
| --- | --- |
| **x** | Karıncaya ait veri |
| **L** | Karıncanın bulunduğu kümenin etiketi |
| **T** | Benzerlik eşiği (threshold) |
| **A** | Karıncanın yaptığı buluşma sayısı |
| **M** | Karıncanın yuvasının büyüklüğüne dair algısı |
| **M+** | Karıncanın, yuvanın diğer üyeleri tarafından kabul edilme derecesi algısı |

**🔁 AntClust Aşamaları**

1. **Threshold öğrenme aşaması**  
   → Karıncalar, benzerlik eşiği olan T'yi öğrenir.
2. **Buluşma (meetings) aşaması**  
   → Karıncalar birbirleriyle etkileşime girerek küme yapısını şekillendirir.
3. **Kümeleri iyileştirme (refinement) aşaması**  
   → Küme yapısı optimize edilir; yanlış yerleştirmeler düzeltilir.

**🧠 Eşik Öğrenme Aşaması (Threshold’s learning phase):**

Her karınca için eşik değeri (**T**) şu şekilde tahmin edilir:  
Karıncanın temsil ettiği veri ile diğer veriler arasındaki **maksimum** ve **ortalama benzerlik** değerlerine göre.

metin, yazı tipi, el yazısı, diyagram içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir. çizgi film içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

**🔄 Rastgele Karşılaşmalar Aşaması (Random Meetings Phase)**

* **Her karşılaşmada şu işlemler yapılır:**
  + **kM kez** olmak üzere **rastgele karınca çiftleri** seçilir.
  + Karınca **i**, karınca **j** ile karşılaştığında **benzerlik değeri** S(i,j) hesaplanır.
  + Sonra şu karar verilir:

Eğer S(i,j) > Ti **ve** S(i,j) > Tj  
→ Karıncalar **birbirini kabul eder**  
Aksi takdirde → **Birbirlerini reddederler metin, ekran görüntüsü, çizgi film içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.**

**✅ Kabul Kuralları (Acceptance Rules)**

**Kural 1:**

Eğer iki etiketsiz (yani kümesi olmayan) karınca karşılaşırsa  
→ Yeni bir küme (yuva) oluştururlar.

diyagram, ekran görüntüsü içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

**Kural 2:**

Eğer etiketsiz bir karınca, etiketli (yani bir kümeye ait) bir karınca ile karşılaşırsa  
→ Etiketsiz karınca, diğer karıncanın kümesine katılır.

diyagram, çizgi film, ekran görüntüsü, kırpıntı çizim içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

**Kural 3:**

Aynı kümeye ait iki karınca karşılaşırsa  
→ Her iki karıncanın da algı parametreleri (**M** ve **M⁺**) artırılır.

**Kural 5:**

Farklı kümelere ait iki karınca karşılaşırsa  
→ **Daha düşük M** değerine sahip karınca, diğer karıncanın kümesine katılır.

**❌ Reddetme Kuralı (Rejection Rule)**

**Kural 4:**

Eğer aynı kümeye ait iki karınca **birbirini reddederse**:

* **Daha düşük M⁺** değerine sahip karınca, kümeden **çıkarılır** ve parametreleri sıfırlanır.
* Diğer karıncanın **M** değeri artırılır. ekran görüntüsü içeren bir resim

  Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

**🧠 Kavramlar Hakkında Kısa Bilgi:**

* **T**: Karıncanın benzerlik eşiği.
* **S(i,j)**: Karınca i ve j arasındaki benzerlik.
* **M**: Karıncanın kendi yuvasının büyüklüğüne dair algısı.
* **M⁺**: Karıncanın yuvada kabul görme seviyesine dair algısı.
* **Etiket (Label)**: Hangi kümeye ait olduğunu belirten işaret.

:

**🌀 Parçacık Sürü Optimizasyonu (Particle Swarm Optimization - PSO)**

**📌 Geliştiriciler:**

* **James Kennedy** ve **Russell Eberhart** tarafından 1995 yılında **doğrusal olmayan fonksiyonların optimizasyonu** için geliştirilmiştir.

**🧠 İlham Kaynağı:**

* **Kuş sürülerinin** ya da **balık sürülerinin** kolektif davranışı.
* Bu davranışlara dayanarak, **kuşlar bir arama uzayında “uçan parçacıklar” gibi düşünülür**, ve optimum (en iyi) çözüme ulaşmaya çalışırlar.

**💡 Temel Fikir:**

* **Arama uzayına yerleştirilmiş bir grup “parçacık”** (agent) kullanılır.
* Her bir parçacık aşağıdaki özelliklerle tanımlanır:
  + 📍 **Konumu (position)** – Çözüm uzayındaki mevcut yeri.
  + 🔄 **Hızı (velocity)** – Hareket yönü ve büyüklüğü.
  + ⭐ **Şimdiye kadar bulduğu en iyi konum (personal best)** – Parçacığın kendi bulduğu en iyi çözüm.
  + 🌍 **Sürünün bulduğu en iyi konum (global best)** – Tüm parçacıklar arasında bulunan en iyi çözüm.

**🔁 Algoritmanın Genel Yapısı:**

1. **Parçacıkların konumlarını başlat**
2. **Konumları değerlendir (amaç fonksiyonuna göre)**
3. **Yerel (kişisel) ve küresel (sürü) hafızaları başlat**
4. **Tekrarla (durdurma koşulu sağlanana kadar):**
   * Yeni **hızları hesapla**
   * **Konumları güncelle**
   * Yeni **konumları değerlendir**
   * **Yerel ve küresel hafızayı güncelle**
5. **Durma koşulu** (örneğin maksimum iterasyon sayısı veya yeterince iyi bir çözüm bulunması) sağlanınca dur.

**🎯 Kullanım Alanları:**

* Fonksiyon optimizasyonu
* Yapay zeka ve makine öğrenmesinde parametre ayarlamaları
* Sürekli ya da kesikli problemler için çözüm arama

**Parçacık Sürü Optimizasyonu (Particle Swarm Optimization - PSO)** algoritmasında **parçacıkların hız ve konum güncellemelerinin nasıl yapıldığı** matematiksel olarak açıklanmıştır. Aşağıda formüller ve ilgili terimler detaylı şekilde açıklanmıştır:

**🧮 Formüller:**

**1. Hız Güncelleme Formülü:**

yazı tipi, çizgi, metin, diyagram içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.Bu formül, parçacığın jjj. boyutundaki hızının nasıl güncellendiğini gösterir.

**Anlamı:**

* vij(t)v\_i^j(t)vij​(t): Parçacık i’nin j. boyuttaki mevcut hızı
* xij(t)x\_i^j(t)xij​(t): Parçacık i’nin j. boyuttaki mevcut konumu
* pij(t)p\_i^j(t)pij​(t): Parçacık i’nin şimdiye kadar bulduğu en iyi konum (**personal best**)
* pbj(t)p\_b^j(t)pbj​(t): Sürünün bulduğu en iyi konum (**global best**)
* r1(t),r2(t)r\_1(t), r\_2(t)r1​(t),r2​(t): [0,1] aralığında rastgele değerler (olasılık etkisi)
* c1,c2c\_1, c\_2c1​,c2​: Öğrenme katsayıları (genelde 1-2 civarındadır)

Bu formül şu 3 etkiyi toplar:

1. **Atalet etkisi** (önceki hız): vij(t)v\_i^j(t)vij​(t)
2. **Bireysel öğrenme** (kendi en iyisine yönelme): (pij(t)−xij(t))(p\_i^j(t) - x\_i^j(t))(pij​(t)−xij​(t))
3. **Sosyal öğrenme** (sürü en iyisine yönelme): (pbj(t)−xij(t))(p\_b^j(t) - x\_i^j(t))(pbj​(t)−xij​(t))

**2. Konum Güncelleme Formülü:**

xij(t+1)=xij(t)+vij(t+1)x\_i^j(t+1) = x\_i^j(t) + v\_i^j(t+1)xij​(t+1)=xij​(t)+vij​(t+1)

**Anlamı:**

* Parçacığın yeni konumu, mevcut konumuna güncellenmiş hızı eklenerek elde edilir.

**📌 Alt Notlar:**

* i=1,...,mi = 1, ..., mi=1,...,m: m tane parçacık var
* j=1,...,nj = 1, ..., nj=1,...,n: n boyutlu bir çözüm uzayındasın
* Bu işlem her iterasyonda tekrarlanır

**🔁 Süreç Özeti:**

1. Her parçacığın hızı ve konumu hesaplanır.
2. Kendi bulduğu en iyi ve sürüdeki en iyi pozisyona doğru hareket eder.
3. Zamanla tüm parçacıklar **en iyi çözüme yakınsar**.

**🔧 PSO Varyantları:**

**1. Atalet Faktörü (Inertia Weight - w) Kullanımı**

Hız güncelleme denklemine **inertia weight (w)** eklenerek parçacıkların önceki hızlarına ne kadar sadık kalacakları kontrol edilir.

**2. Kısıtlama Faktörü (Constriction Factor - γ)**

Kennedy & Clerc (2002) tarafından önerilen bu yöntem, hızın kontrolsüz artmasını önlemek için formüle **bir sabit çarpan (γ)** eklenmesini önerir:

| **Varyant** | **Amaç** | **Faydası** |
| --- | --- | --- |
| **Atalet Faktörü (w)** | Önceki hız etkisini kontrol etme | Arama ve sömürü arasında denge |
| **Kısıtlama Faktörü (γ)** | Hızın büyümesini sınırlama | Daha stabil yakınsama |
| **Yerel En İyi (Neighborhood Best)** | Çeşitliliği artırma | Daha az erken yakınsama |

**🐝 Artificial Bee Colony (ABC) Algoritması Nedir?**

ABC, meta-sezgisel bir optimizasyon algoritmasıdır ve **bal arılarının bal kaynağı arama davranışına** dayanır.

**🧠 İlham Kaynağı:**

* Arıların bal kaynaklarını (yani iyi çözüm noktalarını) keşfetme ve paylaşma stratejileri.

**🧩 Arı Türleri ve Roller:**

ABC algoritmasında 3 tür arı vardır:

| **Arı Türü** | **Görevi** | **Anlamı** |
| --- | --- | --- |
| **Employed (Çalışan) arılar** | Belli bir çözüm noktasında çalışırlar. Oranın kalitesini değerlendirip bilgi paylaşırlar. | Sömürü (exploration of known solutions) |
| **Onlooker (Gözlemci) arılar** | Kovan içinde bekler, employed arıların verdiği bilgiye göre iyi kaynak seçer. | Bilgiye dayalı sömürü (exploitation) |
| **Scout (Keşifçi) arılar** | Yeni rastgele çözüm noktaları ararlar. | Keşif (exploration of new areas) |

**🔄 Algoritmanın Adımları:**

Adım 1: Çalışan arıların pozisyonlarının rastgele başlatılması  
Adım 2: Durdurma koşulu sağlanana kadar:

* Çalışan arılar, bulundukları konumun kalitesi hakkında bilgiyi gözlemcilere (onlookers) gönderir; her gözlemci birkaç çalışan arıdan bilgi alır; seçim, analiz edilen konumların uygunluk (fitness) değerleri kullanılarak hesaplanan olasılık dağılımına göre yapılır.
* Çalışan arılar, bulundukları konumun çevresini keşfeder ve eğer çevrede daha iyi bir konum bulursa oraya geçer; arı, belirli bir adım sayısı içinde daha iyi bir konum bulamazsa rastgele başka bir konuma yerleştirilir (örneğin, bir keşifçi arı tarafından sağlanan pozisyona).
* Keşifçi (scout) arılar pozisyonlarını rastgele değiştirir.

Detaylar:

* Gösterimler: NB = çalışan arı sayısı, NO = gözlemci sayısı,  
  f = uygunluk fonksiyonu (fitness function),  
  n = problem boyutu
* Gözlemciler tarafından seçilen yeni konumun olasılık dağılımı:

yazı tipi, el yazısı, çizgi, beyaz içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

* Yeni pozisyonun bir gözlemci (onlooker) tarafından seçimi, rulet tekniği kullanılarak uygulanabilir.
* Çalışan arılar ise aşağıdaki kurala göre yeniden konumlandırılır (k, rastgele seçilen bir çalışan arının indeksidir, ise [-1,1] aralığında rastgele bir değerdir).

yazı tipi, el yazısı, metin, hat sanatı, kaligrafi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

**Bakteriyel Beslenme Optimizasyonu (BFO)**

**Yaratıcısı:**

K. Passino (2002) – Başlangıçta dağıtık sistemlerde kontrol ve optimizasyon amaçlı kullanılmıştır.

**İlham Kaynağı:**

Bakterilerin (örneğin *Escherichia Coli*) beslenme mekanizmaları.  
Amaç: Bakteri, birim zamanda topladığı enerjiyi (besini) **maksimuma çıkarmaya** çalışır.

**Genel Yapı:**

1. **Başlangıç:**  
   Rastgele pozisyonlarla m tane bakteri popülasyonu başlatılır.
2. **Döngüler (Algoritmanın aşamaları):**
   * **l = 1, ..., Ne:** Eleme ve dağılma döngüsü
   * **k = 1, ..., Nr:** Üreme döngüsü
   * **j = 1, ..., Nc:** Kemotaksi (hareket) döngüsü
     + **Kemotaksi:** Bakterilerin pozisyonunu ayarlaması
     + **Sürüklenme (Swarming):** Besin açısından zengin bölgelere doğru çekilme
     + **Popülasyonun değerlendirilmesi**
   * **Üreme:** Kaliteli bakteriler çoğaltılır, düşük kaliteli olanlar yok edilir
   * **Eleme ve Dağıtım:** Bazı bakteriler rastgele yok edilir ve yenileri rastgele üretilir

**Temel Mekanizmalar:**

**1. Kemotaksi (Chemotaxis):**

Bakteriler, flagella (kamçı) yardımıyla iki şekilde hareket eder:

* **Tumble (Sarsılma):** Rastgele yön değiştirir
* **Swim (Yüzme):** Seçilen yönde ilerleme

Her bakteri aşağıdaki şekilde tanımlanır:

* **Pozisyon:** Arama uzayındaki konum (çözümün yaklaşık hali)
* **Adım büyüklüğü (C(i))**: Her yön için ayrı olabilir
* **Hareket formülü:**

yazı tipi, el yazısı, metin, hat sanatı, kaligrafi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

* + pi: i. bakterinin pozisyonu
  + C(i)C(i)C(i): adım boyutu
  + d(i)d(i)d(i): rastgele yön vektörü (normalize edilmiş)

**2. Sürüklenme (Swarming):**

* Bakteriler, besin açısından zengin bölgelerde bir araya gelip bir halka (ring) oluştururlar.
* Bu etki, **amaç fonksiyonuna** eklenen bir terimle modellenir.
* Bu ek terim, diğer bakterilerin pozisyonlarını dikkate alarak bir bakterinin nasıl hareket edeceğini etkiler.

metin, yazı tipi, ekran görüntüsü, çizgi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

**3. Üreme (Reproduction):**

* Kalitesiz (hasta) bakteriler **ölür** → Popülasyonun en kötü yarısı elenir.
* Kaliteli (sağlıklı) bakteriler **çoğaltılır** → En iyi yarısı kopyalanarak yeni bakteriler oluşturulur.

**4. Eleme ve Dağıtım (Elimination and Dispersal):**

* Ortam koşullarının değişmesi (örneğin sıcaklık artışı) bakterilerin ölmesine neden olabilir.
* Bazı bakteriler rastgele seçilerek **yok edilir** ve yerine **yeni rastgele bakteriler** yerleştirilir.

**Uygulama Alanları:**

* Dinamik kaynak tahsisi
* Kontrol sistemlerinin optimizasyonu
* Yapay sinir ağlarının eğitilmesi