**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 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.

Kötü tanımlanmamış problemler için uygun olan yöntemler şu özelliklere karakterize edilebilir:

* 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 n\*n 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:
  + pi​: 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.

**📅 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).

**🔍 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.

**🎯 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:**

* **V**erilen: 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)

**(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:** vector of real values containing triplets of atoms coordinates

**🎯 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.

**metin, ekran görüntüsü içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.**

**🧠 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. (yörüngee tabanlı)
* **Population-based:** Aynı anda birçok çözümle çalışır. . (nüfus tabanlı)

**B. Arama Alanı:**

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

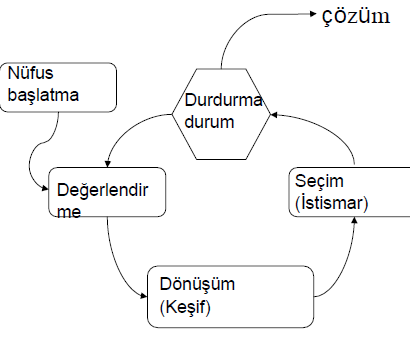
**C. Metafor Kaynağına Göre:**

* **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.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

**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**

metin, ekran görüntüsü, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

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

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.**

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

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.**

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

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.**

**LECTURE 3**

**Trajectory Based Search Algorithms (II) -Yörünge Tabanlı Arama Algoritmaları (II)**

* Simulated Annealing (SA) – Benzetilmiş Tavlama
* Tabu Search (TS) – Tabu Arama
* Variable Neighborhood Search (VNS) – Değişken Komşuluk Arama
* Guided Local Search – Yönlendirilmiş Yerel Arama
* Greedy Randomized Adaptive Search Procedure (GRASP) – Açgözlü Rastgele Uyarlanabilir Arama Yöntemi

**Simulated Annealing (Benzetilmiş Tavlama)**

**Temel Fikir:**Simulated Annealing, minimizasyon problemlerinde, çözümü kötüleştiren (amaç fonksiyonu değerini artıran) çözümleri bile belirli bir olasılıkla kabul ederek yerel minimumlara takılmadan daha iyi çözümler bulmayı hedefler**.**

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

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.**

**Fiziksel Tavlama Süreci ile Benzetim:**

| **Fiziksel Tavlama** | **Algoritmadaki Karşılığı** |
| --- | --- |
| **Madde yüksek sıcaklığa kadar ısıtılır** | **Başta kötü çözümler yüksek olasılıkla kabul edilir** |
| **Madde yavaşça soğutulur** | **Sıcaklık azalır, algoritma kararlı hale gelir** |
| **Parçacıklar düzenli hale gelir** | **Optimum ya da yakın çözüme ulaşılır** |

**Temel Bileşenler:**

* **Başlangıç Sıcaklığı: Çözüm uzayının daha geniş taranmasını sağlar.**
* **Soğuma Çizelgesi (Temperature Schedule): Sıcaklığın zamanla nasıl azalacağını belirler.**
* **Kabul Olasılığı: Kötü bir çözümün kabul edilme ihtimali:  
  P=e−ΔE/TP=e−ΔE/T  
  (ΔE: amaç fonksiyonundaki artış, T: sıcaklık)**

**Tarihçe ve Katkıda Bulunanlar:**

* **Metropolis (1953): Kabul olasılığı mekanizmasını geliştirdi.**
* **Kirkpatrick, Gelatt, Vecchi (1983): Simulated Annealing’i optimizasyona uyguladı.**
* **Cerny (1985): Yöntemi kombinatoryal problemlere taşıdı.**

**Analogy (Benzetim)**

Simulated Annealing algoritmasının çalışma mantığı, fiziksel tavlama süreci ile paralellik gösterir. Aşağıdaki tablo bu benzetimi özetlemektedir:

| **Fiziksel Süreç (Physical Process)** | **Minimizasyon Problemi (Minimization Problem)** |
| --- | --- |
| Sistem enerjisi (System energy) | Amaç fonksiyonu (Objective function) |
| Sistem durumu (System state) | Konfigürasyon / Aday çözüm (Configuration / Candidate solution) |
| Sistem durumundaki değişiklik | Mevcut çözümde yapılan küçük değişiklik (Perturbation of the current configuration) |
| Sıcaklık (Temperature) | Optimizasyon sürecini kontrol eden parametre (Parameter controlling the optimization process) |

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

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.**

**Bu benzetim, algoritmanın fiziksel tavlama sürecinden nasıl esinlendiğini ve optimizasyon problemlerindeki karşılıklarını açık şekilde gösterir.**

**Some Physics (Fiziksel Temeller)**

**Simulated Annealing algoritmasının dayandığı fiziksel prensipler:**

* Her sistem durumu (state) için bir olasılık vardır.
* Bu olasılık, durumun enerjisine (E(s)) ve sistemin sıcaklığına (T) bağlıdır.
* Olasılık, Boltzmann dağılımı ile hesaplanır:

Burada;

* E(s) : Durum ss'nin enerjisi
* T : Sistem sıcaklığı
* kB​ : Boltzmann sabiti
* Z(T) : Normalizasyon faktörü (partition function) [ tüm durumların olasılıklarının toplamını 1 yapar]

**yazı tipi, diyagram, el yazısı, beyaz içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**Not:**Bu dağılım, yüksek sıcaklıklarda tüm durumların daha eşit olasılıkla seçilmesine, sıcaklık azaldıkça ise düşük enerjili durumların daha yüksek olasılıkla seçilmesine olanak tanır.

Some Physics – Temperature Effects (Fizikte Sıcaklığın Etkisi)

Simulated Annealing’de sıcaklık (T) değerinin olasılık dağılımı üzerindeki etkisi:

| **Sıcaklık Durumu** | **Olasılık Dağılımı Özelliği** |
| --- | --- |
| **T → ∞ (çok büyük sıcaklık)** | e−E(s)kBTe−kB​TE(s)​ ifadesinin argümanı yaklaşık 0 olur, yani tüm durumlar eşit olasılıkla seçilir. |
| **T → 0 (çok küçük sıcaklık)** | Sadece enerji değeri sıfır olan durumların olasılığı sıfırdan farklı olur, yani yalnızca en düşük enerji durumları seçilir. |

**Özet:**

* Yüksek sıcaklıkta sistem çok fazla çeşitlilikle rastgele davranır.
* Düşük sıcaklıkta ise sistem kararlı hale gelir ve yalnızca en iyi çözümleri tercih eder**.**

**Using Physics in Optimization (Fiziğin Optimizasyona Uygulanması)**

**Fiziksel ilkelerden yola çıkarak optimizasyon problemi nasıl çözülür?**

**Temel Fikir:**

Amaç, sıcaklık değeri giderek azaltılan bir sistemde, Boltzmann dağılımına uygun şekilde çözüm konfigürasyonları üretmek.

**Zorluk:**

* Boltzmann dağılımı için gerekli olan partition function Z(T)Z(T)’yi hesaplamak çok zordur.
* Z(T)Z(T) tüm olası konfigürasyonlar üzerinden toplam almayı gerektirir.
* Gerçek dünya problemlerinde bu, tüm çözümleri tek tek denemek anlamına gelir (yani exhaustive search).

**Çözüm:**

* Boltzmann dağılımı doğrudan hesaplanmaz.
* Onun yerine, bu dağılıma sahip bir stokastik sürecin (Markov zinciri) simülasyonu yapılır.
* Bu yöntem, Metropolis algoritması olarak bilinir.
* **Kısaca:  
  Fizikteki olasılık dağılımı doğrudan uygulanamaz, ama simüle edilebilir. Böylece SA algoritması, teorideki fiziksel modelin pratik bir karşılığı haline gelir.**

**Metropolis Algorithm (1953)**

**Amaç:**

Daha kötü çözümleri belirli bir olasılıkla kabul ederek, arama sürecinin yerel minimumlara sıkışmasını önlemek.

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

Açıklama otomatik olarak oluşturuldu**

**Türkçe:**

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

Açıklama otomatik olarak oluşturuldu**

**Properties of the Metropolis Algorithm**

**Metropolis Algoritmasının Özellikleri**

* **Alternatif kabul olasılığı (lojistik fonksiyon):**

**metin, yazı tipi, beyaz, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu**

* **Uygulama detayı:**Olasılığa göre değer atamak için (0,1) aralığında rastgele sayı üretilir.

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

Açıklama otomatik olarak oluşturuldu**

* Sıcaklık parametresinin etkisi:
* Yüksek T değerleri: Her konfigürasyon için yüksek kabul olasılığı → saf rastgele arama
* Düşük T değerleri: Sadece düşük enerji (iyi çözüm) durumları yüksek kabul olasılığı → açgözlü arama (gradyan inişine benzer)

**Properties of the Metropolis Algorithm**

Metropolis Algoritmasının Özellikleri

* Yeni konfigürasyonları oluşturmak için kullanılan pertürbasyon (bozma) kuralları, çözülecek probleme bağlıdır.

| Problem Türü | Pertürbasyon Kuralı ve Örnekler |
| --- | --- |
| Sürekli optimizasyon | S′=S+zS′=S+z, burada z=(z1,...,zn)z=(z1​,...,zn​) ve zizi​ dağılıma göre üretilir:  - Normal dağılım N(0,T)N(0,T) - Cauchy dağılımı Cauchy(T)Cauchy(T) (Hızlı SA için)  - Diğer dağılımlar |
| Kombinatoryal optimizasyon | Yeni konfigürasyon, mevcut konfigürasyonun komşuluğundan  - Deterministik  - Rastgele seçilir |

* Örnekler:
  + TSP (Gezgin Satıcı Problemi): 2-opt dönüşümü
  + Knapsack (Sırt Çantası Problemi): Bir nesneyi ekleme/çıkarma (0→1, 1→0)

**Simulated Annealing – Genel Yapı**

Simulated Annealing (SA) = Azalan sıcaklık değerleriyle Metropolis algoritmasının tekrar tekrar uygulanmasıdır.

**Genel Yapı (Pseudocode):metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu**

Amaç: Sıcaklığı yavaş yavaş düşürerek optimum çözüme ulaşmak.  
 Sorun: Soğuma şeması (cooling schedule) nasıl seçilmeli?

**Cooling Schemes (Soğutma Yaklaşımları)**

Simulated Annealing algoritmasında sıcaklığın (T) zamanla düşürülmesi gerekir. Bu işlem "cooling scheme" (soğutma şeması) ile yapılır.

Bazı yaygın soğutma fonksiyonları:

| Fonksiyon Türü | Açıklama |
| --- | --- |
| T(k) = T(0) / (k + 1) | Lineer azalış |
| **T(k) = T(0) / ln(k + c)** | **Logaritmik azalış (c küçük bir sabit)** |
| **T(k) = a · T(k–1)** | **Geometrik azalış (örnek: a = 0.995, a < 1)** |

**📌**Not: Başlangıç sıcaklığı T(0), algoritmanın ilk iterasyonlarında neredeyse tüm yeni çözümlerin kabul edileceğişekilde büyük seçilmelidir.  
➡️ Bu, arama uzayının etkili keşfini sağlar.

**Convergence Properties (Yakınsama Özellikleri)**

Simulated Annealing algoritmasının global minimuma yakınsayabilmesi için bazı şartların sağlanması gerekir:

✅ Yakınsama için gereken koşullar:

1. Her iki konfigürasyon arasında geçiş olasılığı sıfırdan büyük olmalı:  
   Pg(S(k+1)=x’ | S(k)=x) > 0  
   ➤ Arama uzayında herhangi bir durumdan diğerine ulaşmak mümkün olmalı.
2. Metropolis kabul olasılığı kullanılmalı:  
   Pa(S(k+1)=x’ | S(k)=x) = min{1, exp(-(f(x’)-f(x))/T)}  
   ➤ Daha kötü çözümler de belirli bir olasılıkla kabul edilebilmeli.
3. Logaritmik soğutma şeması kullanılmalı:  
   T(k) = C / ln(k + c)  
   ➤ Sıcaklık yavaşça düşürülmeli.

**Sonuç:**

**Bu koşullar sağlandığında:**

P(|f(S(k)) - f(S\*)| < ε) → 1 olur, k → ∞ iken.  
Yani algoritma küresel minimuma olasılıksal olarak yakınsar.

📌 Not: Logaritmik soğutma, teorik olarak garantili yakınsama sağlasa da, pratikte çok yavaş bir algoritmayla sonuçlanır.

**Variant: Tsallis Kabul Olasılığı**

Simulated Annealing algoritmasında, Metropolis yerine alternatif bir kabul olasılığı fonksiyonu kullanılabilir. Bu varyantlardan biri de Tsallis istatistiklerine dayanan Tsallis kabul olasılığıdır.

**📌 Tsallis kabul olasılığı formülü:**

Aşağıdaki gibi tanımlanır:

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

Açıklama otomatik olarak oluşturuldu**

**Değişkenler:**

* Δf=f(x′)−f(x)Δf=f(x′)−f(x) → Yeni çözümün mevcut çözüme göre farkı
* TT → Sıcaklık
* q∈(0,1)q∈(0,1) → Tsallis parametresi

**Açıklama:**

* Δf≤0Δf≤0: Daha iyi bir çözüm bulunduysa her zaman kabul edilir.
* Δf>0Δf>0: Daha kötü çözüm belirli bir olasılıkla kabul edilir.
* Tsallis olasılığı, Metropolis'e göre daha esnek bir kontrol sağlar.

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

**Veri Kümesi:**  
Test örneği: eil51 (51 şehirden oluşur)

**Parametreler:**

* Toplam 5000 iterasyon
* Sıcaklık her 100 iterasyonda bir değiştirilir
* Soğuma fonksiyonu**:**

**yazı tipi, beyaz, metin, tipografi içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**Şehirlerin Konumu:**

Bu örnekteki şehirler, eil51 test dosyasına göre belirli bir düzende (örneğin düzlemde x-y koordinatları ile) yerleştirilmiştir. Bu veri kümesi, gerçek bir harita yerine test amaçlı oluşturulmuş koordinatlar içerir**.**

**ekran görüntüsü, dikdörtgen, kalıp, desen, düzen, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**metin, diyagram, kalıp, desen, düzen, origami içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**Tabu Search (Tabu Arama)**

Yaratıcı: Fred Glover (1986)  
Kullanım Alanı: Özellikle kombi̇natoryel opti̇mi̇zasyon problemleri̇

Özellikleri:

* Yinelenmeli yerel arama tekniğidir.
* Mevcut çözümün komşuluk alanı araştırılır.

Komşuluk nedir?

* Komşuluk, mevcut yapılandırmadan bir kez arama operatörü uygulanarak ulaşılabilen tüm konfigürasyonların (çözümlerin) kümesidir.
* Bu operatörler, çözülmek istenen probleme özeldir.

Bir Yapılandırmanın Komşuluk Örnekleri (Neighbourhoods)

* Gezgin Satıcı Problemi (TSP):  
  Mevcut çözüm S'ye 2-opt dönüşümü uygulanarak elde edilebilecek konfigürasyonlar.
* Sırt Çantası Problemi (Knapsack):  
  S'den Hamming mesafesi 1 olan ikili vektörler (yalnızca 1 biti farklı olanlar).
* Atama Problemi (Bin Packing):
  + Bir nesneyi bir kutudan başka bir kutuya taşıma
  + Farklı kutularda bulunan iki nesnenin yerini değiştirme

**Tabu Listesi**

* Tabu listesi, belirli sayıda sonraki yinelemede kabul edilmeyecek olan yapılandırmaları içerir.
* Genellikle sabit boyutlu dairesel bir liste olarak uygulanır.
* Amaç: Aramanın önceki çözümlere geri dönmesini engellemek ve daha geniş bir arama alanı keşfetmektir.

**Genel Yapı (Tabu Search - Genel Akış)**

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

Açıklama otomatik olarak oluşturuldu**

**Notlar:**

1. Eğer komşuluk kümesi çok büyükse, sadece bir örneklem değerlendirilerek işlem yapılabilir.
2. Tabu Search, arama geçmişini açıkça kullanır:
   * Tabu listesi, kısa vadeli bir hafıza gibi işleyerek  
     yerel minimumlardan kaçmayı ve  
     arama alanının daha geniş keşfini sağlar.

**Not:**

* Aday çözümlerin tamamını tabu listesinde saklamak verimsiz olabilir.
* Bunun yerine aday çözümlerin bazı özellikleri saklanabilir.

**Özelliklere örnekler:**

* Permütasyon tipi kodlamalar (örn. Gezgin Satıcı Problemi - TSP): İki elemanın yer değiştirmesi bir özellik olabilir. (i,j) özelliği, si ile sj'nin yer değiştirdiğini belirtir.
* İkili (binary) kodlamalar (örn. Sırt çantası problemi): Bir bileşenin değişmesi bir özelliktir. (i,0) özelliği si'nin 0 olduğunu, (i,1) ise si'nin 1 olduğunu belirtir.

**Ek Notlar:**

* Tam çözümler yerine özelliklerin kullanılması bazı aşırı kısıtlamalara neden olabilir (bazı iyi çözümler dışlanabilir).
* Bu tür durumları önlemek için istek (aspiration) kuralı kullanılır: Şu ana kadar bulunan en iyi çözümden daha iyi olan konfigürasyonlar, yasaklı özelliklere sahip olsalar bile kabul edilir.

**Yoğunlaştırma (Intensification):**Amacı: Umut vadeden bölgelerin daha fazla araştırılmasıdır (yani, mevcut çözümden uzak olan konfigürasyonları cezalandırmak).

**Uygulama:**

* Bir bileşenin veya özelliğin değişmeden kaldığı iterasyon sayısını say.  
  → İlgili sayacı yüksek olan bileşenler "iyi" bileşenler olarak kabul edilir.  
  → Çünkü bu bileşenler, mevcut ve yeni konfigürasyonlar arasındaki rekabette korunmuştur.
* Arama süreci, en iyi konfigürasyondan yeniden başlatılır.  
  → Ancak, iyi bileşenler/sabit özellikler korunur.  
  → Komşuluk sadece diğer bileşenlerin değiştirilmesiyle oluşturulur.

**Çeşitlendirme (Diversification):**Amacı: Daha önce ziyaret edilmemiş bölgelerin keşfedilmesini sağlamak (yani, mevcut çözüme yakın konfigürasyonları cezalandırmak).

**Uygulama:**

* Her bileşen için kullanılan değerlerin frekansını hesapla.  
  → Düşük frekansa sahip değerler "az araştırılmış" olarak kabul edilir.
* Arama sürecini, az araştırılmış değerleri içeren konfigürasyonlardan başlat veya
* Uygunluk fonksiyonunu değiştirerek, sık kullanılan bileşen değerlerini cezalandır.  
  → Böylece arama, yeni veya daha az keşfedilmiş bölgelere yönlendirilmiş olur.

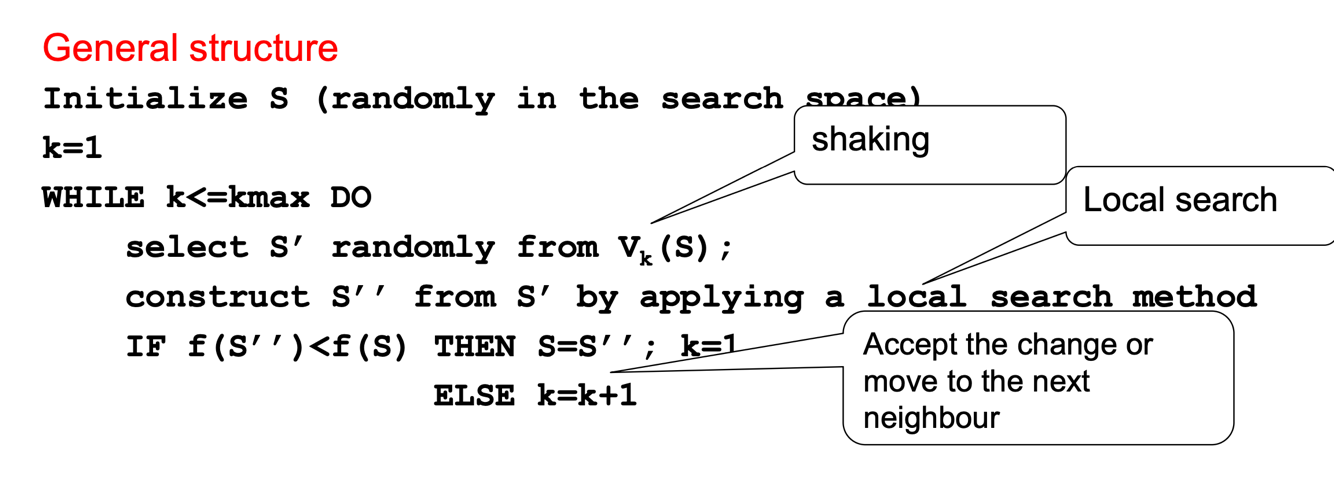
**Variable Neighborhood Search (VNS) – Değişken Komşuluk Araması  
*Mladenović, P. Hansen, 1997***

**Fikir:**VNS, artan şekilde keşfedilen V1, V2, ..., Vkmax olmak üzere bir komşuluk kümesi kullanır. Her bir komşulukta arama, yerel arama yöntemiyle gerçekleştirilir.

**Not:**Bir konfigürasyon x için komşuluk kümesi, çözülecek probleme bağlı olarak belirlenir; ancak şu koşulu sağlar:

* Eğer k1 < k2 ise, Vk1(x) elemanları x’den daha az işlemle elde edilebilir,
* Vk1(x), Vk2(x) kümesine dahil olabilir ama zorunlu değildir.

**Örnek:**Gezgin Satıcı Problemi (TSP) için Vk(x), x’den rastgele seçilen k düğümün yer değiştirilmesiyle (k swap) elde edilen rotasyonları içerebilir.

****

**Remark:**Belirli bir k için uygulanan yerel arama, sadece Vk(S) ile sınırlı değildir.

**Guided Local Search (Yönlendirilmiş Yerel Arama)**

**Creators: Voudouris & Tsang, 1999**

**Idea:**

* Amaç fonksiyonu, lokal optimum bölgelerden kaçmak için dinamik olarak değiştirilir (cezalandırılan değerler daha az tercih edilir hale gelir).
* Bu değişiklik, kaçınılması gereken aday çözümlerin özelliklerine karşılık gelen bazı ceza terimlerinin eklenmesine dayanır (örneğin, TSP’de böyle bir özellik iki konumu bağlayan bir kenar olabilir).

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

Açıklama otomatik olarak oluşturuldu**

* **p: ceza parametresi sayısı**
* **λi​: ceza parametresi (kontrol faktörü veya düzenleme etkisi)**

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

Açıklama otomatik olarak oluşturuldu**

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

Açıklama otomatik olarak oluşturuldu**

**Genel Yapı ve Açıklaması**

Bu yapı Guided Local Search (Yönlendirilmiş Yerel Arama) algoritmasının temel yapısını ve mantığını ifade ediyor.

**Terimler ve Fonksiyonlar:**

* S: Başlangıç konfigürasyonu (yani algoritmanın başladığı çözüm).
* LocalSearch(S, f): Mevcut çözüm S üzerinde, amaç fonksiyonu f’ye göre bir yerel arama uygular ve iyileştirilmiş çözüm si’yi döner.
* Ii(S): Eğer i’inci özellik (feature) S çözümünde varsa 1, yoksa 0 döner.  
  Yani, çözümün hangi özellikleri içerdiğini gösterir.
* pi: i’inci özellik için ceza katsayısı (penalty). Bu değer o özelliğin ceza miktarını tutar ve algoritmanın o özelliği kullanmasını azaltmak için kullanılır.
* U(s, i): Özelliğin cezalandırılmadan önceki faydası veya maliyeti (Utility) olarak düşünülebilir.  
  Formül:

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

Açıklama otomatik olarak oluşturuldu**

Burada cici​ o özelliğin maliyeti, pipi​ ise ceza katsayısıdır.  
Ceza arttıkça UU azalır, yani algoritma o özelliği kullanmaktan kaçınır.

* f\_mod(S): Ceza terimleri eklenmiş modifiye edilmiş amaç fonksiyonu, yani:

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

Açıklama otomatik olarak oluşturuldu**

**Burada λ ceza terimlerinin toplam amaç fonksiyonuna etkisini kontrol eden katsayıdır.**

**Algoritmanın Adımları:**

1. Başlangıç:  
   S başlangıç konfigürasyonu olarak seçilir.
2. Tekrarla (Repeat):
   * Yerel arama yapılır: s=LocalSearch(S,f) Yani, mevcut çözüm S için amaç fonksiyonu f’yi iyileştirecek şekilde yerel arama yapılır ve yeni çözüm s elde edilir.
   * Maksimal faydaya (gain) sahip olan özellikler tespit edilir. Bu özellikler, cezalandırıldığında algoritmanın en çok etkilenmesini sağlar (yani U(s,i)U(s,i) en yüksek olanlar).
   * Bu özelliklerin ceza katsayıları 1 artırılır: pi=pi+1pi​=pi​+1. Böylece, algoritma bu özellikleri tekrar seçmekten kaçınmaya başlar.
   * Amaç fonksiyonu güncellenir:

**f=Update(f,p)f=Update(f,p)**

**Bu işlem cezalar eklenmiş amaç fonksiyonunu yeniler.**

1. **Durdurma koşulu sağlanana kadar devam edilir.**

Özet:

* Guided Local Search algoritması, klasik yerel arama algoritmalarına ceza terimleri ekleyerek çalışır.
* Algoritma, belirli özelliklere (features) ceza uygulayarak, tekrar eden yerel minimumlara takılmayı önlemeye çalışır.
* Ceza katsayıları arttıkça algoritma bu özellikleri kullanan çözümlerden uzaklaşır ve daha geniş arama alanlarını keşfeder.
* λλ katsayısı, ceza terimlerinin ağırlığını belirler; çok büyük olursa ceza aşırı etkili olur, çok küçük olursa ceza etkisi azalır.

**GRASP**GRASP = Aç Gözlü Rastgele Uyarlanabilir Arama Yöntemi [Feo & Resende, 1995]

Fikir:  
GRASP, keşif (exploration) ve sömürü (exploitation) arasında bir denge sağlayarak iyi başlangıç çözümleri oluşturmayı hedefler.

Özellikleri:

* Başlangıç çözümünü bileşen bileşen oluşturur (parça parça yapılandırır).
  + Örneğin, TSP’de (Gezgin Satıcı Problemi) her adımda bir konum ya da kenar eklenir.
  + Sırt çantası probleminde ise her adımda bir obje seçilip eklenir.
* Önemli nokta, aday çözüme hangi bileşenin dahil edileceğinin seçilmesidir:  
  GRASP, sıralı bir listedeki bileşenleri rastgele ancak eşit olmayan (non-uniform) bir şekilde seçer; yani iyi bileşenlerin seçilme olasılığı daha yüksektir.

Özellikler  
• Liste boyutu zamanla değişebilir (uyarlanabilir özellik) ve iki uç durum arasında değişir:  
– Boyut = 1 -> açgözlü seçim (her adımda en iyi bileşen seçilir)  
– Boyut = mümkün olan maksimum değer sayısı (eğer seçim eşit olasılıklıysa bu tamamen rastgele arama olur)

**metin, yazı tipi, çizgi, cebir içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**Overall structure**

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

Açıklama otomatik olarak oluşturuldu**

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

Açıklama otomatik olarak oluşturuldu**

Özet

Yörünge tabanlı arama yöntemlerinin temel adımları:

* Başlatma:  
  Aday çözüm parça parça oluşturulur. Bunu rastgele yapabiliriz veya açgözlü (en iyi seçeneği seçen) bir yöntem kullanabiliriz.
* Yeni aday oluşturma:
  + Yapılandırma: Başlangıç aşamasında çözümü doğrudan oluşturmak için kullanılır.
  + Bozma: Mevcut çözüme küçük veya büyük değişiklikler yaparak yeni adaylar üretmek. Bu seçim rastgele ya da seçkincilik (elitist) olabilir.
* Komşuluk yapısı:  
  Komşuluk (yani yeni adayların nereden seçileceği) sabit ya da değişken olabilir. (Örneğin, VNS’de değişkendir.)
* Bozma büyüklüğü:  
  Küçük değişiklikler (yerel arama) veya büyük değişiklikler (ILS gibi) yapılabilir.
* Yeni adayı kabul etme:
  + Sadece daha iyi ise kabul et (Hill Climbing, iniş yöntemi gibi).
  + Daha kötü adayları da kabul et (durgunluğu ve takılı kalmayı önlemek için), bu kabul olasılığı kalite kaybına ve kontrol parametrelerine bağlıdır (Simulated Annealing gibi).
  + Daha iyi adaylar zaten kullanılmış ve yasaklanmışsa, onları da göz önünde bulundurmaz (Tabu Search gibi).

Yörünge Tabanlı Arama Yöntemlerinin Temel Bileşenleri:

Durdurma koşulu belirlenir:

* Kaliteye dayalı: Kabul edilebilir kalitede bir çözüm bulunduğunda durur.
* Davranışa dayalı: Son iterasyonlarda iyileşme olmazsa durur.
* Kaynaklara dayalı: Belirli sayıda iterasyon ya da amaç fonksiyonu değerlendirmesi yapıldığında durur.

**LECTURE 4**

**Population based metaheuristics – Popülasyon Tabanlı Sezgisel Yöntemler**

* Particularities – Özellikler / Ayırt Edici Noktalar
* Classes of population based metaheuristics – Popülasyon tabanlı sezgisel yöntemlerin sınıfları
* General structure – Genel yapı
* Main components – Ana bileşenler

**Evolutionary algorithms – Evrimsel Algoritmalar**

* Encoding – Kodlama / Birey temsil yöntemi
* Selection – Seçim (doğal seçilim mekanizması)
* Reproduction: crossover and mutation – Üreme: çaprazlama (crossover) ve mutasyon (mutation)

**Population based metaheuristics – Popülasyon Tabanlı Sezgisel Yöntemler**

**Problem çözümü =**  
• Çözüm, bir **aday çözümler popülasyonu (population of candidate solutions)** üzerinden aranır.  
• Arama, çözümün ne kadar yaklaşıldığını "ölçen" bir **değerlendirme fonksiyonu** (**fitness function**) ile yönlendirilir.

**There are two main search mechanisms: – İki temel arama mekanizması vardır:**

• **Exploration** = **Arama alanının keşfi**  
Popülasyon öğeleri, problem hakkında bilgi toplamak için arama alanını keşfeder.  
(Bu süreçte popülasyon öğeleri arasında **iş birliği (cooperation)** bulunur.)

• **Exploitation** = **Keşfedilen bilginin kullanımı**  
Keşif sırasında toplanan bilgiler kullanılarak potansiyel çözüm(ler) iyileştirilir.  
(Bu süreçte popülasyon öğeleri arasında **rekabet (competition)** bulunur.)

**Not:** Bu paradigma üzerine kurulu birçok sezgisel yöntem (**metaheuristic**) bulunmaktadır.

diyagram, çizgi, öykü gelişim çizgisi; kumpas; grafiğini çıkarma, metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Evolutionary algorithms – Evrimsel algoritmalar**

Doğal evrim süreçlerinden ilham alan ve bireylerin (çözümlerin) nesiller boyunca evrimleşerek daha iyi çözümler üretmesini amaçlayan algoritmalar:

* Genetic Algorithms – Genetik Algoritmalar
* Evolution Strategies – Evrim Stratejileri
* Evolutionary Programming – Evrimsel Programlama
* Genetic Programming – Genetik Programlama

**Swarm intelligence algorithms – Sürü zekâsı algoritmaları**

Doğadaki sosyal canlıların (karıncalar, arılar, kuşlar vb.) kolektif davranışlarından esinlenerek geliştirilen algoritmalar:

* Particle Swarm Optimization – Parçacık Sürü Optimizasyonu
* Ant Colony Optimization – Karınca Kolonisi Optimizasyonu
* Artificial Bee Colony – Yapay Arı Kolonisi Algoritması

… ve çok sayıda biyolojik ilhamlı meta-sezgisel yöntem (bio-inspired metaheuristics)

**Other population based algorithms – Diğer popülasyon tabanlı algoritmalar**

* Differential Evolution – Diferansiyel Evrim Algoritması
* Estimation of Distribution Algorithms – Dağılım Tahmini Algoritmaları

**General Structure – Genel Yapı**

1. Population initialization  
   (rastgele ya da belirli sezgisel yöntemlere dayalı olarak popülasyonun başlatılması)
2. Population evaluation  
   (bireylerin/değerlerin değerlendirilmesi, yani amaç fonksiyonu değerlerinin hesaplanması)
3. REPEAT – TEKRARLA
   * Yeni aday çözümlerden oluşan bir popülasyon oluştur  
     (Construct a new population of candidate solutions)
   * Aday çözümleri değerlendir  
     (Evaluate the candidate solutions)
   * Yeni popülasyonun bireylerini seç  
     (Select the elements of a new population)
4. UNTIL <stopping condition> – DURANA KADAR <durdurma koşulu**>**

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

Açıklama otomatik olarak oluşturuldu**

Remark – Not:

* Yeni aday çözümlerin oluşturulma şekli, kullanılan algoritma türüne bağlıdır.  
  (The generation of new candidate solutions depends on the algorithm type.)
* Her döngüde (nesilde) tamamen yeni bir popülasyon oluşturulur ve bu popülasyon önceki popülasyonla "hayatta kalma" için rekabet eder.  
  (At each cycle (generation), an entire new population is constructed and it competes with the previous population for survival.)  
  Bu stratejiye generational ya da synchronous updating stratejisi denir.  
  (This is a so-called generational or synchronous updating strategy.)

**Variant – Varyant:**

Aday çözümler oluşturuldukları anda analiz edilir ve gerekirse mevcut popülasyona dahil edilir.  
(The candidate solutions are analyzed and potentially assimilated into population just after their generation)  
Bu stratejiye steady state (kararlı durum) ya da asynchronous updating strategy (eş zamansız güncelleme stratejisi) denir.

**Algoritmanın Genel Yapısı**

1. Population initialization:  
   Popülasyonun başlatılması: (s₁, s₂, ..., sₘ)  
   (rastgele veya belirli sezgisel yöntemlerle başlangıç çözümleri oluşturulur)
2. Population evaluation:  
   Popülasyonun değerlendirilmesi  
   (her çözüm için amaç (objective) fonksiyonu değeri hesaplanır)
3. REPEAT – TEKRARLA  
   FOR i = 1 to m:  
   (her birey için aşağıdaki adımlar uygulanır)
   * Yeni bir aday çözüm oluştur  
     (construct a new candidate solution (sᵢ'))
   * Yeni aday çözümü değerlendir  
     (evaluate the new candidate solution)
   * Yeni çözümün mevcut popülasyona kabul edilip edilmeyeceğine karar ver  
     (decide if the new candidate solution is accepted into the population)
4. **UN**TIL <stopping condition> – DURANA KADAR <durdurma koşulu>

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

Açıklama otomatik olarak oluşturuldu**

**Remark – Not:**

Yeni aday çözüm genellikle, mevcut popülasyondaki en kötü elemandan daha iyi ise kabul edilir.  
(A new candidate solution is usually accepted if it is better than the worst element of the current population.)

Bu varyantta, popülasyon adım adım güncellenir ve her zaman bir nesil (generation) beklenmez, bu da algoritmanın daha hızlı tepki verebilmesini sağlar.

Evolutionary computing (Evrimsel hesaplama)

Evrimsel hesaplama, doğal evrimden ilham alınarak tasarlanan ve uygulanan tekniklerin bütünüdür.  
İlham kaynağı: türlerin evrimi

* Türler, üreme sırasında gerçekleşen çaprazlama (crossover) ve rastgele mutasyonlar sonucu yeni özellikler kazanarak evrimleşirler.
* Evrim sürecinde en uygun bireyler hayatta kalır; yani çevreye en iyi uyum sağlayanlar seçilir.

**Analogy: Evolution - Optimization**

**(Evrim ile Optimizasyon Arasındaki Benzerlik)**

| Evolutionary process (Evrimsel süreç) | Problem solving (Problem çözme) |
| --- | --- |
| Natural environment | Information about the problem |
| Individual (chromosome) | Configuration (candidate solution) |
| Population of individuals | Population of candidates |
| Fitness (degree of adaptation to the environment) | Measure of the solution quality |
| Selection | Exploitation mechanism |
| Reproduction (crossover and mutation) | Exploration mechanism |

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

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.**

Bu tablo, doğal evrim sürecindeki temel kavramların optimizasyondaki karşılıklarını gösterir.  
Evrimde “uygunluk” bireyin çevreye uyumunu ölçerken, optimizasyonda “fitness” çözüm kalitesini ifade eder.  
Seçim ve üreme mekanizmaları, problem çözmede çözüm kalitesinin iyileştirilmesi ve çeşitliliğin sağlanması için kullanılır.

**Evrimsel Hesaplama: Temel Kavramlar (düzeltilmiş)**

* Kromozom (Chromosome):  
  Bir bireye (problemin potansiyel çözümüne) karşılık gelen gen dizisi.  
  Örnek: (1,0,0,1)
* Popülasyon (Population):  
  Sonlu sayıda bireyden (kromozomlardan, yani aday çözümlerden) oluşan küme.  
  Örnek: {(0,0,0,0), (0,0,1,1), (1,0,0,1), (1,0,1,0)}
* Genotip (Genotype):  
  Bir bireyin tüm gen dizisi (yani kromozomunun tamamı).
* Fenotip (Phenotype):  
  Genotip tarafından ortaya çıkarılan dışa vurulan özellikler ya da çözümler.  
  Örnek: {0, 3, 9, 10}

**Fitness (Uygunluk)**

Bir bireyin (aday çözümün) kalite ölçüsü (çözülmek istenen probleme göre değerlendirilir).  
Örnek: İkili dizideki 1'lerin sayısını maksimize etmek isteniyor.  
(1,0,0,1) dizisinin fitness değeri = 2 (çünkü iki tane 1 var).

**Nesil (Generation)**

Popülasyonun evrimsel sürecindeki aşama (arama sürecindeki iterasyon).

**Üreme (Reproduction)**

Mevcut popülasyondan (ebeveynlerden) yeni bireylerin (çocukların) oluşturulması.  
Bunu iki temel operatör yapar:

* Crossover (Çaprazlama):  
  İki kromozomun parçalarını değiş tokuş ederek yeni bireyler oluşturma.  
  Örnek:  
  (1,0,0,1) → (1,0,1,1)  
  (0,0,1,1) → (0,0,0,1)
* Mutation (Mutasyon):  
  Kromozomdaki genlerden bir veya birkaçının rastgele değiştirilmesi.  
  Örnek:  
  (1,0,1,1) → (1,1,1,1)

**Evrimsel Hesaplama: Uygulamalar**

* Planlama (Scheduling): Araç rotalama problemleri, ders programı oluşturma, telekomünikasyon ağlarında yönlendirme
* Tasarım (Design): Dijital devreler, filtreler, sinir ağları
* Modelleme (Modelling): Ekonomi, finans, tıp vb. alanlarda öngörücü modeller
* Veri Madenciliği (Data mining): Mühendislik, biyoloji, tıp vb. alanlarda sınıflandırma sistemlerinin tasarımı

**Designing an EvolutionaryAlgorithm (Evrimsel Algoritma Tasarımı)**

**Components:**

**metin, ekran görüntüsü, diyagram, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**Evrimsel Algoritmanın (EA) Genel Yapısı**

1. Popülasyonun Başlatılması  
   Başlangıçta, rastgele veya belirli bir yöntemle bir popülasyon oluşturulur.
2. Popülasyonun Değerlendirilmesi  
   Popülasyondaki bireylerin kalitesi (fitness) ölçülür.
3. Tekrarla (Döngü Başlatılır):
   * Ebeveynlerin Seçilmesi  
     Yeni nesil oluşturmak için uygun bireyler seçilir.
   * Çaprazlama (Crossover)  
     Seçilen ebeveynlerden yeni bireyler (çocuklar) oluşturulur.
   * Mutasyon (Mutation)  
     Yeni bireylerde küçük rastgele değişiklikler yapılır.
   * Yeni Bireylerin Değerlendirilmesi  
     Oluşan bireylerin kalitesi ölçülür.
   * Hayatta Kalanların Seçilmesi  
     Popülasyon için en uygun bireyler seçilir ve sonraki döngüye aktarılır.
4. Durdurma Koşulu Sağlanana Kadar Devam Edilir  
   Örneğin, belirli bir nesil sayısına ulaşılana ya da yeterli kalite elde edilene kadar devam edilir.

**Structure of an Evolutionary Algorithm (EA)**

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

Açıklama otomatik olarak oluşturuldu**

**Directions in Evolutionary Computing (Evrimsel Hesaplamada Yönelimler)**

1. Genetic Algorithms (Holland, 1962-1967)

* Encoding (Kodlama): Binary (İkili)
* Crossover (Çaprazlama): Main operator (Ana işlem)
* Mutation (Mutasyon): Secondary operator (İkincil işlem)
* Applications (Uygulamalar): Combinatorial optimization (Kombinatoryal optimizasyon)

**2. Genetic Programming (Koza, 1990)**

* Encoding (Kodlama): Tree-like structures (Ağaç yapısı)
* Crossover (Çaprazlama): Main operator (Ana işlem)
* Mutation (Mutasyon): Secondary operator (İkincil işlem)
* Applications (Uygulamalar): Program evrimi (Programların evrilmesi)

**3. Evolution Strategies (Rechenberg, Schwefel 1965)**

* Encoding (Kodlama): Real (Gerçek sayılar)
* Mutation (Mutasyon): Main operator (Ana işlem)
* Recombination (Yeniden Kombinasyon): Secondary operator (İkincil işlem)
* Applications (Uygulamalar): Continuous optimization (Sürekli optimizasyon)

**4. Evolutionary Programming (L. Fogel, D. Fogel, 1960-1970)**

* Encoding (Kodlama): Real / state diagrams (Gerçek sayılar / durum diyagramları)
* Mutation (Mutasyon): The only operator (Tek işlem)
* Applications (Uygulamalar): Continuous optimization (Sürekli optimizasyon)

| **Yöntem (Method)** | **Kodlama (Encoding)** | **Ana İşlem (Main Operator)** | **İkincil İşlem (Secondary Operator)** | **Uygulama Alanları (Applications)** |
| --- | --- | --- | --- | --- |
| **Genetic Algorithms** | **Binary (İkili)** | **Crossover (Çaprazlama)** | **Mutation (Mutasyon)** | **Combinatorial optimization (Kombinatoryal optimizasyon)** |
| **Genetic Programming** | **Tree-like structures (Ağaç yapısı)** | **Crossover (Çaprazlama)** | **Mutation (Mutasyon)** | **Program evolution (Program evrimi)** |
| **Evolution Strategies** | **Real numbers (Gerçek sayılar)** | **Mutation (Mutasyon)** | **Recombination (Yeniden kombinasyon)** | **Continuous optimization (Sürekli optimizasyon)** |
| **Evolutionary Programming** | **Real / state diagrams (Gerçek / Durum diyagramları)** | **Mutation (Mutasyon)** | **None (Yok)** | **Continuous optimization (Sürekli optimizasyon)** |

**Encoding (Kodlama)**

Kodlama, bir evrimsel algoritmanın tasarımında kilit unsurlardan biridir. Kullanılacak kodlama yöntemi, doğrudan çözülmek istenen probleme bağlıdır.

Başlıca Kodlama Türleri:

* Binary Encoding (İkili Kodlama):
  + Klasik Genetik Algoritmalar (GA) için kullanılır.
  + Her birey (çözüm), 0 ve 1'lerden oluşan bir bit dizisiyle temsil edilir.
* Real Encoding (Gerçek Sayılarla Kodlama):
  + Sürekli (continuous) optimizasyon problemleri için uygundur.
  + Genellikle Evrim Stratejileri (Evolution Strategies - ES) tarafından kullanılır.
* Problem-Related Encoding (Probleme Özgü Kodlama):
  + Problemin doğasına göre uyarlanır.
  + Örneğin:
    - Permütasyon (örn. Gezgin Satıcı Problemi - TSP)
    - Ağaç yapısı (tree) (örn. Genetik Programlama - GP)
    - Graf yapısı (graph)
  + Evrimsel Programlama (Evolutionary Programming) ve Genetik Programlama (Genetic Programming) içinde yaygın olarak kullanılır.

**Binary Encoding (İkili Kodlama)**

Chromosome (Kromozom):  
Bir birey (çözüm), 0 ve 1’lerden oluşan bir ikili dizi (binary sequence) ile temsil edilir.

Arama Uzayı (Search Space):  
0,1n  
Burada n, problemin boyutunu belirtir.

**Örnek 1: ONEMAX Problemi**

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

Açıklama otomatik olarak oluşturuldu**

**Açıklama:**Her bir bit 1 olursa toplam değer artar. Bu nedenle en iyi çözüm tüm bitlerin 1 olmasıdır:  
(1,1,...,1)

**Örnek 2: Knapsack (Sırt Çantası) Problemi**

Verilenler:

* n adet nesne, her biri:
  + Ağırlık: w1,w2,...,wn
  + Değer: v1,v2,...,vn
* Kapasitesi C olan bir sırt çantası

Amaç:  
Toplam ağırlık kapasiteyi aşmadan, toplam değeri maksimum olacak şekilde nesne alt kümesi seçmek.

Kodlama Şekli:

(s1,s2,...,sn)

* si=1 : i. nesne seçilir
* si=0 : i. nesne seçilmez

**Örnek 3: Sürekli Tanımlı Bir Fonksiyonun Optimizasyonu**

(Optimization of a Function Defined on a Continuous Domain)

Amaç:  
Sürekli tanımlı bir fonksiyonun maksimum ya da minimum değerini bulmak.

**metin, yazı tipi, makbuz, beyaz içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**Adım 1: Gerçek Sayıların Normalizasyonu**

**Her xideğeri, kendi tanım aralığında [0,1] aralığına normalize edilir:**

**metin, yazı tipi, beyaz, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**Adım 2: Tamsayıya Dönüştürme**

**Normalize edilmiş videğeri r bitlik çözünürlükle sayısallaştırılır:**

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

Açıklama otomatik olarak oluşturuldu**

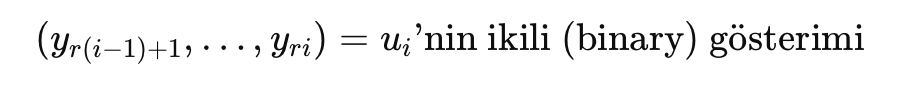
**Adım 3: İkilik Temsil**

**Her ui​, r bitlik ikili sayı biçimine çevrilir ve tümü birleştirilerek bir kromozom oluşturulur:**

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

Açıklama otomatik olarak oluşturuldu**

**Burada,**

****

Bu dönüşüm süreci sayesinde sürekli tanımlı bir fonksiyon, ikili (binary) kromozomlar üzerinde çalışan evrimsel algoritmalara uygun hale getirilmiş olur.

**Not: İkili (Binary) Kodlamanın Dezavantajı ve Gray Kod Çözümü**

❗ Sorun: Hamming Mesafesi

İkili kodlamada birbirine sayısal olarak çok yakın iki değer, Hamming mesafesi açısından çok uzak olabilir**.**

**🔍 Örnek:**

* **7 = (0111)<sub>2</sub>**
* **8 = (1000)<sub>2</sub>**

Bu iki sayı yalnızca 1 farkla ayrılırken, ikili karşılıkları 4 bitlik fark içerir.  
Bu durum, genotipik uzayda yakın olan bireylerin, fenotipik olarak uzak olmasına ve evrimsel aramayı zorlaştırmasına neden olur.

**✅ Çözüm: Gray Kod Kullanımı**

Gray kodu, ardışık değerler için sadece bir bitin değiştiği bir kodlama yöntemidir.  
Bu sayede genotipik uzayda yapılan küçük değişiklikler, fenotipik uzayda da küçük değişiklikler olarak yansıtılır**.**

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

Açıklama otomatik olarak oluşturuldu**

**Avantaj:**

* Daha düzgün fitness yüzeyleri elde edilir.
* Arama süreci daha kararlı hale gelir.
* Yakın çözümler birbirine benzer genetik temsillere sahip olur.

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

Açıklama otomatik olarak oluşturuldu**

Problem-related encoding (Probleme özel kodlama):  
Bu tür kodlama, çözülecek probleme özgüdür. Yani her problem için kodlama yöntemi farklı olabilir ve problemin yapısına göre özel olarak tasarlanır**.**

Örnek: Permütasyon benzeri kodlama  
Burada, çözüm bir permütasyon (sıralama) şeklindedir.  
Örneğin**:**

(s1,s2,...,sn)

burada her sisi​, 1’den n’e kadar olan sayılardan biridir ve bütün si​’ler birbirinden farklıdır. Yani, aynı sayı (örneğin aynı şehir) birden fazla kez yer almaz.

**Problem: Gezgin Satıcı Problemi (TSP - Traveling Salesman Problem)**

* si​ burada ziyaret edilen şehirlerin sırasını gösterir.
* s1= ilk ziyaret edilen şehir,
* s2 = ikinci ziyaret edilen şehir,
* …
* sn​ = son ziyaret edilen şehir.

**Önemli nokta:**Bu kodlama yöntemi, TSP’nin kısıtlarını otomatik olarak sağlar. Çünkü her şehir sadece bir kez ziyaret edilir, yani her sayı yalnızca bir kez dizide bulunur. Böylece turdaki tekrarlar engellenmiş olur ve geçerli bir çözüm oluşturulur**.**

**Özetle:**Problem-related encoding, problemin özel yapısına göre çözümün temsil edilmesini sağlar. TSP gibi sıralama/kombinasyon problemi için permütasyon şeklinde kodlama yapılır ve böylece çözümler geçerli (kısıtları sağlayan) olur**.**

**Evaluation of the Population Elements (Popülasyon Elemanlarının Değerlendirilmesi)**

Fitness (Uyumluluk Değeri):

* Bireyin (çözüm adayının) kalitesini ölçer.
* Fitness değeri ne kadar yüksekse, o bireyin popülasyonda hayatta kalma (seçilme) olasılığı o kadar yüksek olur.

Problem Türüne Göre Fitness Fonksiyonunun Değişimi:

1. Kısıtsız Optimizasyon Problemi (Unconstrained Optimization):

* Maksimizasyon problemi için fitness fonksiyonu, hedef fonksiyonun kendisi ile doğru orantılıdır. Yani hedef fonksiyon değeri arttıkça fitness da artar.
* Minimizasyon problemi için fitness fonksiyonu, hedef fonksiyonun tersi ile orantılıdır. Yani hedef fonksiyon küçüldükçe fitness değeri büyür.

1. Kısıtlı Optimizasyon Problemi (Constrained Optimization):

* Fitness fonksiyonu hem hedef fonksiyona hem de kısıtlamalara bağlıdır.
* Yani, çözümün kalitesi sadece hedef fonksiyonun değeri ile değil, aynı zamanda kısıtların ne kadar sağlandığıyla da ölçülür.
* Kısıtları sağlamayan çözümlere düşük fitness verilir veya cezalandırılır.

**Özetle:**Fitness fonksiyonu, algoritmanın hangi çözümlere öncelik vereceğini belirler ve problem tipine göre değişkenlik gösterir. Kısıtlı problemler için fitness hesaplamasında kısıtların dikkate alınması çok önemlidir.

Constraints: Penalty Method ile Kısıtların Amaç Fonksiyonuna Dahil Edilmesi

Amaç, kısıtlı optimizasyon problemlerinde kısıtları cezalandırma yöntemiyle (penalty method) amaç fonksiyonuna dahil etmektir.

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

Açıklama otomatik olarak oluşturuldu**

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

Açıklama otomatik olarak oluşturuldu**

**Açıklama:**

* Eğer kısıtlar sağlanıyorsa (gi(x)≤0gi​(x)≤0 ve hi(x)=0hi​(x)=0 ise), ϕϕ fonksiyonu sıfır olur, yani ceza uygulanmaz.
* Eğer kısıt sağlanmıyorsa, ϕϕ fonksiyonu negatif bir değer alır ve amaç fonksiyonunun değerinden bu ceza çıkarılır, yani o çözüm daha düşük değer alır.
* Bu sayede, optimizasyon algoritması kısıtları ihlal eden çözümlerden uzak durur.

Bu yöntem, kısıtlı optimizasyon problemlerini, kısıtsız optimizasyona benzer şekilde çözmeyi sağlar ancak kısıt ihlallerine göre ceza vererek aramayı yönlendirir.

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

Açıklama otomatik olarak oluşturuldu**

**Not (Remark):**Ceza katsayıları a ve b, amaç fonksiyonu ile kısıtlar arasındaki göreli önemi kontrol eder.

Yani, a ve b değerleri ne kadar büyük olursa, kısıtların sağlanması o kadar çok öncelik kazanır ve kısıt ihlalleri daha ağır şekilde cezalandırılır. Küçük değerlerde ise, amaç fonksiyonunun iyileştirilmesi ön planda olur.

**Selection (Seçim)**

**Aim (Amaç):**

* Mevcut popülasyondaki hangi bireylerin yeni nesil bireylerin oluşturulmasında kullanılacağına karar vermek (ebeveyn seçimi).
* Yeni oluşan yavru popülasyondan hangi bireylerin bir sonraki nesle geçeceğine karar vermek (hayatta kalanların seçimi).

**Basic idea (Temel fikir):**

* Yüksek fitness değeri olan bireylerin seçilme şansı daha yüksektir.

**Selection mechanisms (Seçim mekanizmaları):**

* Proportional selection (Oransal seçim)
* Rank based selection (Sıralamaya dayalı seçim)
* Tournament selection (Turnuva seçimi)
* Truncation selection (Kesme seçimi)

**Selection Procedure (Seçim Süreci)**

**Current population:**  
 P=(x1,x2,...,xm)   
 (Mevcut popülasyon)

**Steps (Adımlar):**

1. Fitness values (Fitness değerleri):  
    (F1,F2,...,Fm)
2. Selection probabilities (Seçilme olasılıkları):

yazı tipi, metin, sayı, numara, beyaz içeren bir resim

Açıklama otomatik olarak oluşturuldu

(Toplam fitness değerine göre bireylerin seçilme olasılıkları)

1. Remark (Not):  
   Eğer Fi​ değerleri 0 veya negatif olabilir ise, bu değerler aşağıdaki şekilde değiştirilebilir:

yazı tipi, beyaz, metin, tipografi içeren bir resim

Açıklama otomatik olarak oluşturuldu

(ϵ küçük pozitif bir sayı, örn. 0.0001)

1. Compute the selection probabilities (Seçilme olasılıklarını hesapla).
2. Generate random values according to the distribution (Dağılıma göre rastgele değerler üret).

**metin, çizgi, yazı tipi, beyaz içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**Implementation (Uygulama):**

* (i) Roulette Wheel (Rulet Çarkı)
* (ii) Stochastic Universal Sampling (SUS) (Stokastik Evrensel Örnekleme)

**Roulette Wheel Selection (Rulet Çarkı Seçimi)**

* Popülasyondaki mm eleman için, seçim olasılıklarına orantılı olarak alanları farklı olan mm sektöre bölünmüş bir rulet çarkı düşünelim.
* Ruleti çevir (spin). Gösterge önünde duran sektörün indeksi, popülasyondan seçilecek elemanı belirler.

**metin, diyagram, daire, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**Proportional selection**

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

Açıklama otomatik olarak oluşturuldu**

**Remark (Notlar)**

1. Bu algoritma, olasılık dağılımından rastgele bir değişkenin simülasyonuna karşılık gelir.
2. Bir çalıştırmada (bir turda) sadece bir elemanın indeksi elde edilir.
3. Aynı anda birden fazla elemanın indeksini seçmek için **SUS (Stochastic Universal Sampling)** yöntemi kullanılabilir.

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

Açıklama otomatik olarak oluşturuldu**

**Remark (Not):**

* **k**, seçilmesi gereken eleman sayısını temsil eder.
* **c[1..m]** dizisi, her elemanın kaç tane kopyasının seçildiğini tutar.

Yani, seçim işlemi sonunda elimizde, her bir elemanın kaç kere seçildiğini gösteren bir sayı dizisi olacak. Bu, özellikle çoklu seçimler veya nüfus oluşturma süreçlerinde kullanılır.

**Dezavantajlar (Disadvantages):**

1. Pozitif olmayan değerler:  
   Amaç fonksiyonu pozitif değerler içermiyorsa, fitness değerlerinin dönüştürülmesi gerekir. Çünkü seçim algoritmaları genellikle pozitif değerlere ihtiyaç duyar.
2. Aşırı uyumlu bireylerin çoğalması:  
   En iyi bireyin fitness değeri ile diğer bireylerin arasındaki fark çok büyükse, popülasyon çoğunlukla en iyi bireyin kopyalarıyla dolabilir. Bu durum evrim sürecini durdurur veya çeşitliliği azaltır, dolayısıyla algoritmanın etkinliği düşer.

**Rank Based Selection (Sıralamaya Dayalı Seçim)**

Özellikler:

* Seçim olasılıkları, bireylerin fitness değerlerine göre artan şekilde sıralanması sonrası aldıkları ranka bağlı olarak hesaplanır. (Maksimizasyon problemi varsayılmıştır.)

Adımlar:

1. Popülasyondaki bireylerin fitness değerlerini artarak sıralayın.
2. Aynı fitness değerine sahip bireyler için bir rank (sıra) belirleyin. En küçük fitness değeri rank 1 olarak atanır.
3. Popülasyonu aynı ranga sahip bireylerden oluşan sınıflara (k tane) ayırın.
4. Her sınıfın seçim olasılığını hesaplayın:

**yazı tipi, çizgi, diyagram, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu**

Burada i sınıfın rank’ıdır.

1. Sınıfları roulette wheel veya Stochastic Universal Sampling (SUS) yöntemleriyle seçin. Ardından, seçilen sınıflardan rastgele bireyler seçin.

**Tournament Selection (Turnuva Seçimi)**

Temel Fikir:  
Bir eleman, popülasyondaki diğer elemanlarla yapılan karşılaştırma sonucuna göre seçilir.

Adımlar:

1. Popülasyondan rastgele k adet eleman seçilir.
2. Seçilen bu k eleman arasından en iyi olanı seçilir.

**Notlar:**

* Genellikle k=2k=2 kullanılır (ikili turnuva).
* Turnuva seçiminde kısıtlar fitness fonksiyonuna dahil edilmek zorunda değildir. Örneğin, kısıtları daha az ihlal eden eleman daha iyi sayılabilir.

**Truncated Selection (Kırpılmış Seçim)**

Temel Fikir:

* Ebeveynler ve yeni nesil bireylerin birleşik popülasyonundan en iyi kk adet eleman seçilir.

Özellikler:

1. Bu yöntem tamamen deterministiktir (rastgelelik içermez).
2. Özellikle Evolution Strategies (Evrim Stratejileri) alanında yaygın olarak kullanılır**.**

**Properties of Selection — Elitism (Seçkinlik)**

* Bir seçim yöntemi elitizm özelliğine sahipse, bu demektir ki popülasyondaki en iyi birey her zaman seçilir ve sonraki nesillerde mutlaka kalır.
* Truncation selection (kırpılmış seçim), bu elitizm özelliğini her zaman sağlayan tek yöntemdir.
* Elitizmi sağlamak için, mevcut popülasyondaki en iyi birey açıkça yeni popülasyona eklenebilir (örneğin, en kötü bireyin yerine koyarak**).**

**Selection Pressure (Seçim Baskısı)**

* Seçim baskısı, popülasyondaki en iyi bireyin tüm popülasyonu ele geçirme olasılığını ölçer.
* Yüksek seçim baskısı, arama alanında çok fazla sömürü (exploitation) yapılmasına yol açar ve keşif (exploration) azalır; bu durum, algoritmanın lokal optimumda takılıp kalmasına ya da duraklamaya sebep olabilir.
* Olasılıksal seçim yöntemlerinde (örneğin, orantılı seçim, turnuva seçimi), seçim baskısı “takeover time” ile ölçülebilir:
  + Bu, yalnızca seçim uygulanıyor olsaydı, tüm popülasyonun en iyi bireyin kopyaları ile dolması için gereken jenerasyon sayısıdır.

**Crossover (Çaprazlama)**

**Amaç:**Popülasyondaki iki veya daha fazla bireyi birleştirerek bir veya birkaç yeni birey (offspring) oluşturmak**.**

**Not:**Genetik algoritmalarda genellikle iki ebeveynden iki çocuk oluşturulur.

Çeşitleri:

* One cut-point crossover: Tek bir kesim noktası belirlenir ve genler bu noktadan sonra değiş tokuş edilir.
* Uniform crossover: Genler rastgele ve bağımsız olarak ebeveynler arasında karıştırılır.
* Convex crossover: Genler ebeveynlerin genotiplerinden ağırlıklı ortalama alınarak oluşturulur (genellikle sürekli değerler için).
* Problem özelinde tasarlanmış crossover: Belirli probleme özgü kurallarla crossover gerçekleştirilir.

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

Açıklama otomatik olarak oluşturuldu**

**Cut-points Crossover**

Özellikler:

1. Seçilen her ebeveyn çifti üzerinde, belirli bir olasılıkla (genellikle 0.2 ile 0.9 arasında, Pc) crossover uygulanır.
2. Kesim noktaları rastgele seçilir.
3. Sayısal deneyler, iki kesim noktası kullanılarak yapılan crossover'ın tek kesim noktası kullanılan crossover’dan daha iyi sonuçlar verdiğini göstermektedir**.**

**Uniform Crossover**

Özelliği:  
Ebeveynlerin genleri, çocukların genleri için rastgele seçilir.

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

Açıklama otomatik olarak oluşturuldu**

**Yani, çocukların genleri ebeveynlerden bağımsız ve rastgele seçilir. Gen pozisyonundaki genler ebeveynlerden biri tarafından alınır, diğer ebeveyne ait gen diğer çocukta olur.**

**Convex Crossover**

**Amaç:  
Gerçek değerli (real-valued) vektörler için kullanılır.**

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

Açıklama otomatik olarak oluşturuldu**

**Açıklamalar:**

* Birden fazla ebeveyn varsa (daha genel durumda), sadece bir çocuk üretilir.
* aa katsayısı bileşen başına farklı seçilebilir; yani her ii için farklı aiai​ kullanılabilir.
* Daha genel varyasyonlarda aa katsayısı (−p,p+1)(−p,p+1) aralığından seçilebilir.
* Eğer bileşen başına farklı katsayılar kullanılıyorsa formül şu şekilde olur:

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

Açıklama otomatik olarak oluşturuldu**

**Crossover for permutation-like elements**

Amaç:  
Permütasyon (sıralama) şeklindeki bireylerin genetik algoritmada çaprazlanmasıdır. Burada genetik yapıların (bireylerin) elemanları birbirinden farklı ve sıralı olur (örneğin şehirlerin ziyaret sırası).

Özellik:

* Geleneksel crossover yöntemleri (örneğin tek noktalı crossover) permütasyonlar için doğrudan uygulanamaz çünkü offspring’da aynı eleman birden fazla olabilir veya bazı elemanlar eksik kalabilir.
* Bu nedenle özel, problem bazlı crossover yöntemleri gerekir.

**Örnek: TSP Problemi için Crossover**

Problem:  
Bir tur, şehirlerin ziyaret sırasını belirten bir permütasyondur. Örneğin,  
Parent 1: A B C D E F G  
Parent 2: A B E G D C F

Kesim noktası (Cutpoint): 3 olarak seçilmiş.

Amaç:  
Ebeveynlerden alınan kısımlarla yeni tur (offspring) oluşturmak.

**Örnek Crossover İşlemi**

* Parent 1’in ilk 3 şehri çocuk 1’e kopyalanır: A B C
* Parent 2’nin kalan şehirleri sırasıyla alınır ama çocukta olmayanlar eklenir: E G D F
* Çocuk 1: A B C E G D F

**Benzer şekilde,**

* Parent 2’nin ilk 3 şehri çocuk 2’ye kopyalanır: A B E
* Parent 1’in kalan şehirleri sırasıyla alınır ama çocukta olmayanlar eklenir: C D F G
* Çocuk 2: A B E C D F G

**Özet**

* Amaç: Hem ebeveynlerin bilgilerini kullanmak, hem de çocukların geçerli bir permütasyon olması için (aynı elemandan birden fazla olmaması ve hiç eksik olmaması) özel yöntem kullanmak.
* Bu tür çaprazlama permutation-based crossover olarak adlandırılır.
* Bu yöntemler, tur yapısı veya sıralama problemine özgüdür**.**

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

Açıklama otomatik olarak oluşturuldu**

**Mutation (Mutasyon)**

**Amaç:**

* Gen havuzuna (gene pool) yeni genler eklemek.
* Mevcut genotipte olmayan varyasyonları ortaya çıkararak çeşitliliği artırmak.
* Böylece algoritmanın yerel minimumlara takılmasını önlemek ve arama alanını daha iyi keşfetmek.

**Notlar:**

* Mutasyonun nasıl yapılacağı seçilen kodlama (encoding) tipine bağlıdır.
* Kodlama tipine göre mutasyon farklı şekillerde uygulanır.

**Mutasyon türleri ve uygulamaları:**

1. Binary encoding (İkili kodlama):
   * Bazı genler rastgele seçilir ve 0 ise 1, 1 ise 0 yapılır (bitlerin tamamlanması/complementing).
   * Örnek: 10101 → mutasyon sonrası 10001 (ikinci bit değişmiş).
2. Real encoding (Gerçek sayılarla kodlama):
   * Gen değerlerine küçük rastgele sapmalar (perturbation) eklenir.
   * Örnek: 3.14 → 3.18 gibi hafif değişiklikler.
3. Specific encoding (Problem/özgü kodlama):
   * Probleme özel mutasyonlar yapılır, genellikle özel heuristiklere dayanır.
   * Örnek: Permutasyonlarda iki şehir yer değiştirilir.

**Mutasyon seviyeleri:**

1. Local mutation:
   * Birey bazında uygulanır (tek kromozom üzerinde).
   * Küçük, yerel değişiklikler.
2. Global mutation:
   * Popülasyon genelinde daha geniş bir gen havuzunda uygulanır.
   * Daha büyük çapta çeşitlilik yaratabilir**.**

**Özet:**

* Mutasyon, genetik algoritmalarda çeşitlilik yaratmak ve keşif yeteneğini artırmak için kritik bir adımdır.
* Kodlama tipine göre farklı mutasyon stratejileri vardır.
* Doğru mutasyon oranı (genellikle düşük, örneğin %1-%5) algoritmanın başarısını önemli ölçüde etkiler.

**Chromosome Level Mutation (Kromozom Seviyesinde Mutasyon) Adımları:**

1. Mutasyona uğrayacak kromozomları seç:
   * Tüm popülasyondan küçük bir mutasyon olasılığı (probability) ile rastgele kromozomlar seçilir.
   * Örnek: Eğer popülasyon büyüklüğü mm ise, mutasyon olasılığı

yazı tipi, tipografi, sayı, numara, beyaz içeren bir resim

Açıklama otomatik olarak oluşturuldu olarak alınabilir.

1. Her seçilen kromozom için rastgele bir gen seç:
   * Seçilen kromozom içindeki genler arasından rastgele bir tanesi belirlenir.
2. Seçilen geni mutasyona uğrat:
   * Bu gen, kodlama tipine göre mutasyona tabi tutulur (örneğin ikili kodlama ise 0 ise 1 yapılır, gerçek değer ise küçük bir değişiklik yapılır).

Notlar:

* Mutasyon olasılığı çok yüksek olmamalıdır, aksi halde genetik çeşitlilik zarar görür ve algoritmanın performansı düşer.
* yazı tipi, tipografi, sayı, numara, beyaz içeren bir resim

  Açıklama otomatik olarak oluşturuldu​ formülü genellikle önerilen bir kuraldır; burada mm popülasyon büyüklüğüdür.
* Bu yöntem sayesinde, her nesilde sadece az sayıda genetik varyasyon yaratılır.

**Pool Gene Level Mutation (Gen Havuzu Seviyesinde Mutasyon):**

Varsayımlar:

* Popülasyondaki tüm kromozomlar yan yana (ardışık olarak) birleştirilir.
* Böylece tek uzun bir gen dizisi (örneğin uzun bir ikili dizi) oluşur.

Mutasyon İşleyişi:

* Bu uzun gen dizisindeki her bir gen teker teker incelenir.
* Her gen için, belirlenmiş bir mutasyon olasılığı (Pm) üzerinden rastgele karar verilir:
  + Eğer karar "mutasyona uğrayacak" ise gen değiştirilir.
  + Değilse, aynı kalır.

**Önemli Notlar:**

1. Bir kromozomda birden fazla gen değişebilir çünkü bütün gen havuzu tek tek kontrol edilir.
2. Bu yöntem, genetik çeşitliliği artırmak için daha agresif mutasyon sağlar.
3. Mutasyon olasılığı genellikle küçük tutulur, çünkü çok fazla mutasyon algoritmanın kararlılığını bozabilir.

**Control Parameters in Evolutionary Algorithms (Evrimsel Algoritmalarda Kontrol Parametreleri):**

Evrimsel algoritmaların davranışı, popülasyon yapısını ve evrim mekanizmalarını kontrol eden parametrelere bağlıdır. Bu parametreler algoritmanın performansını ve verimliliğini doğrudan etkiler.

Temel Kontrol Parametreleri:

1. Population Size (Popülasyon büyüklüğü):  
   Popülasyondaki birey sayısı. Çok küçükse çeşitlilik azalır, çok büyükse hesaplama maliyeti artar.
2. Stopping Condition Parameters (Durdurma koşulu parametreleri):
   * Maksimum nesil sayısı
   * Maksimum amaç fonksiyonu değerlendirme sayısı
   * Maksimum durağanlık (stagnasyon) adımı sayısı (iyileşme olmadan geçen nesil sayısı)
3. Selection Parameters (Seçim parametreleri):
   * Örneğin, turnuva seçimi için turnuva büyüklüğü (kaç birey yarışacak)
4. Crossover Parameters (Çaprazlama parametreleri):
   * Çaprazlama olasılığı (Pc): İki ebeveynin çaprazlama ile karışma olasılığı
5. Mutation Parameters (Mutasyon parametreleri):
   * Mutasyon olasılığı (Pm): Bir genin mutasyona uğrama olasılığı
   * Rastgele değişikliklerde kullanılan olasılık dağılımlarına ilişkin parametreler (örneğin, dağılımın varyansı veya standart sapması

**Özet:**

* Bu parametreler algoritmanın keşif (exploration) ve sömürü (exploitation) yeteneklerini dengeler.
* İyi ayarlanmış parametreler, daha hızlı ve daha kaliteli çözümler bulunmasını sağlar.
* Parametre ayarları genellikle problem türüne ve büyüklüğüne göre değişir.

**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