# CENG 415 Evrimsel Hesaplama

## Bölüm 4: Temsil, Mutasyon ve Rekombinasyon

Şevket Umut Çakır

Pamukkale Üniversitesi

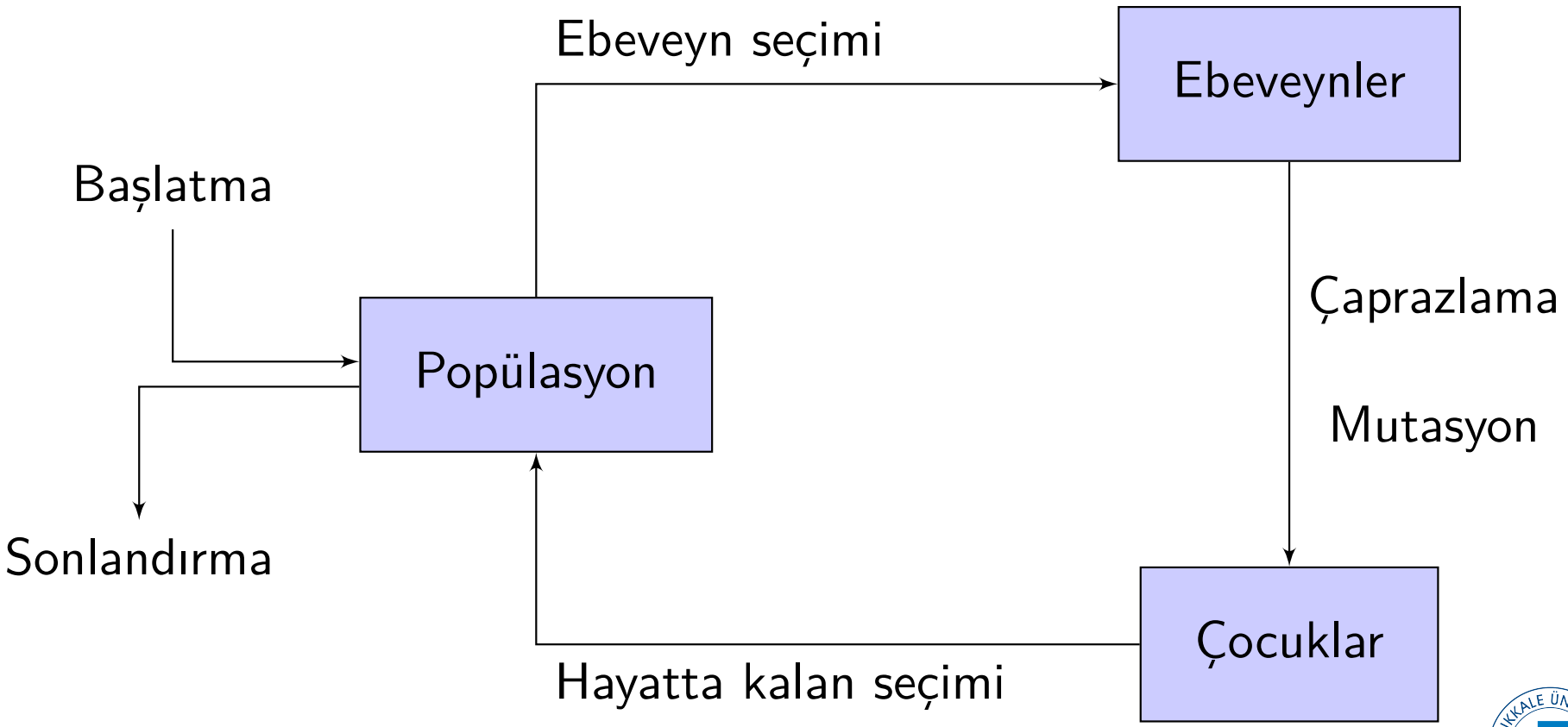
8 Kasım 2020

# Temsil, Mutasyon ve Rekombinasyon

- Temsil ve çeşitlilik(varyasyon) operatörlerinin rolleri
- Genomların en yaygın temsilleri:
  - İkili(binary)
  - Tamsayı
  - Gerçek-değerli ya da kayan noktalı
  - Permütasyon
  - Ağaç

# Evrimsel Hesaplama Şeması

## Genel Şema

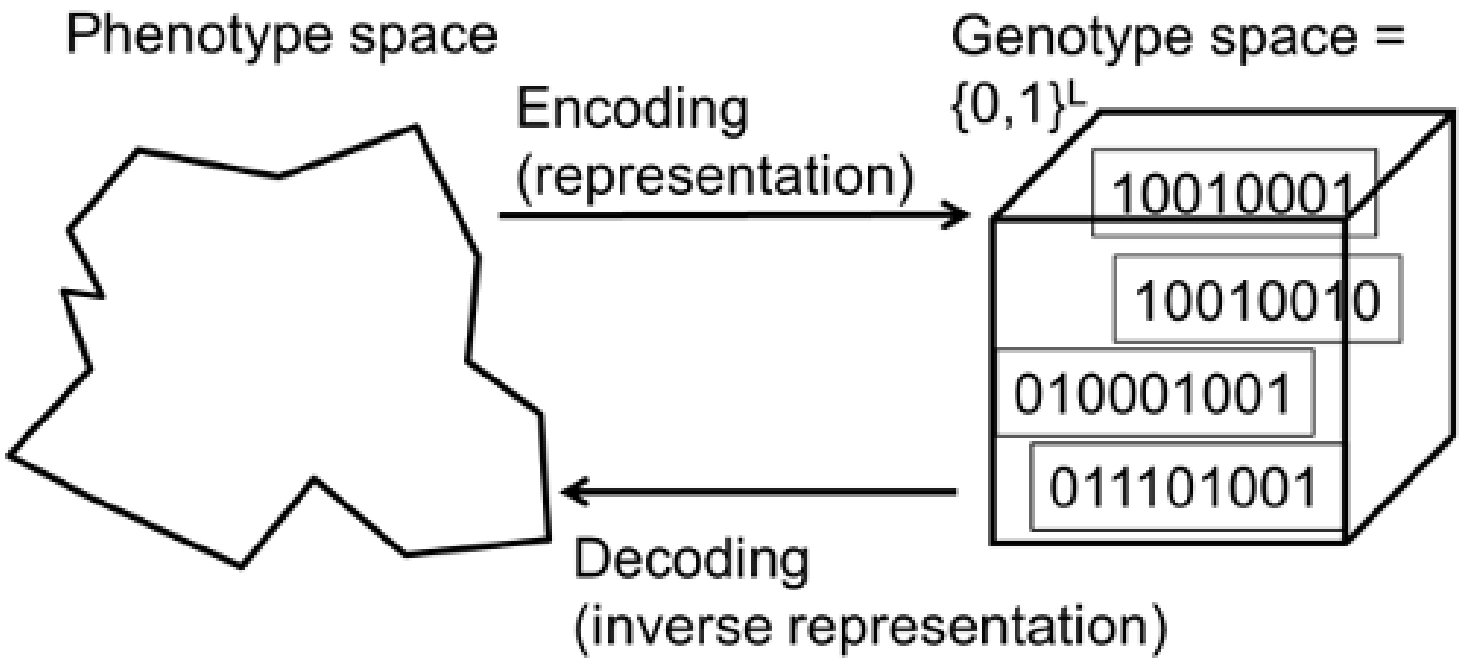


# Temsil ve Varyasyon Operatörlerinin Rolü

- Bir EA geliştirmenin ilk ve en zor aşaması: Problem için *doğru* temsili seçmek
- Varyasyon operatörleri: Mutasyon ve çaprazlama
- İhtiyaç duyulan varyasyon operatörleri seçilen temsile bağlıdır
- TSP(GSP)
  - Mümkün temsil biçimleri nelerdir?

# İkili Temsil

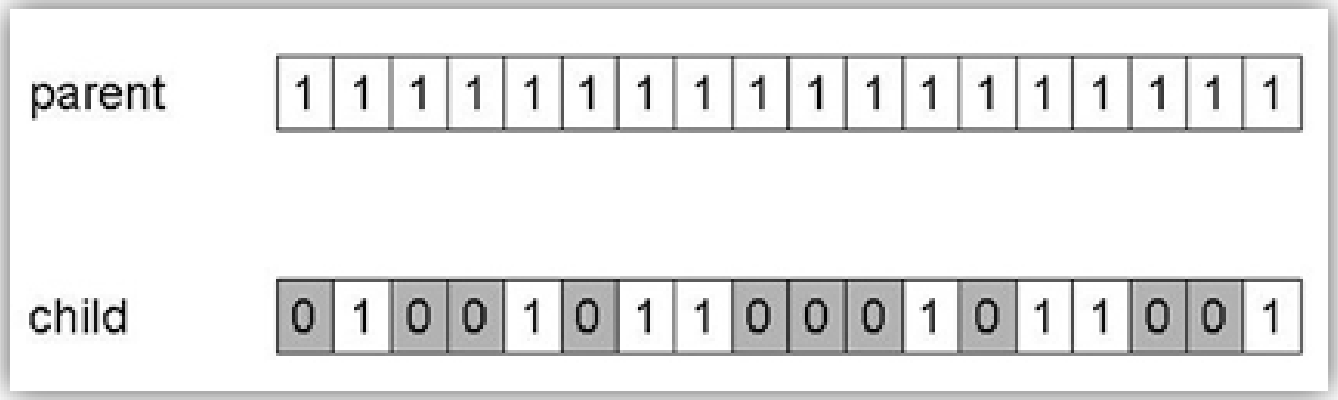
- İlk temsil biçimlerinden biridir
- Genotip, bir dizi ikili rakamdan oluşur



# İkili Temsil

## Mutasyon

- Her bir geni bağımsız olarak $p_m$ olasılığı ile değiştir
- $p_m$ mutasyon oranı olarak adlandırılır
  - Genellikle $\frac{1}{popülasyon\ \ boyutu}$ ile $\frac{1}{kromozom\ \ uzunluğu}$ arasında değişir

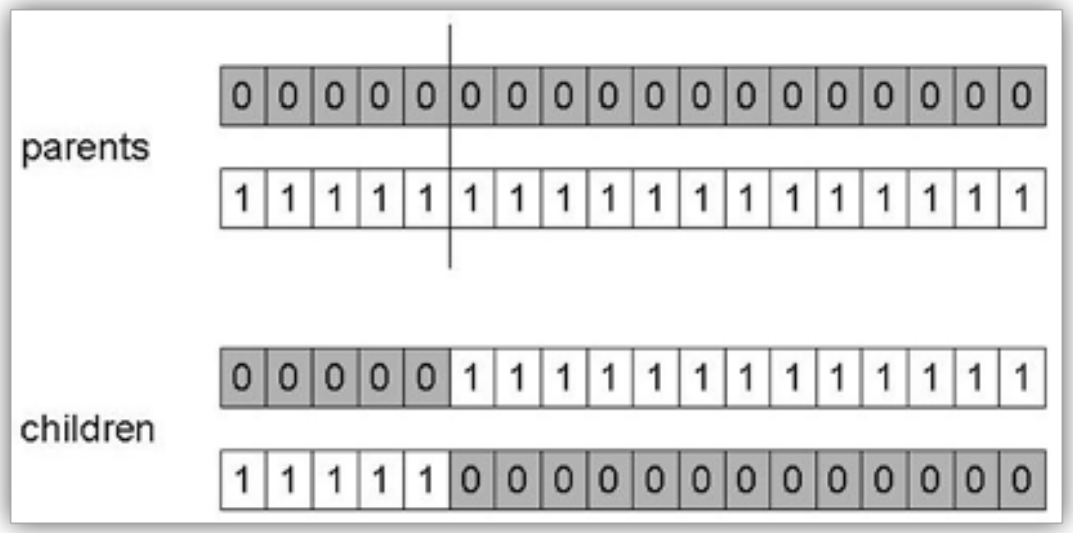



- Mutasyon değişken etkiye sahip olabilir(grey kod kullanın)
  - **00000000** $\rightarrow$ **10000000**: değişiklik çok büyük

# İkili Temsil

## Tek noktalı çaprazlama(1-point crossover)

- İki ebeveyn için rastgele bir nokta seçin $[1, l-1]$
- Ebeveynleri bu geçiş noktasında ayırın
- Kuyrukları değiş tokuş ederek çocukları oluşturun
- $p_c$ tipik olarak (0.6, 0.9) aralığında



# İkili Temsil

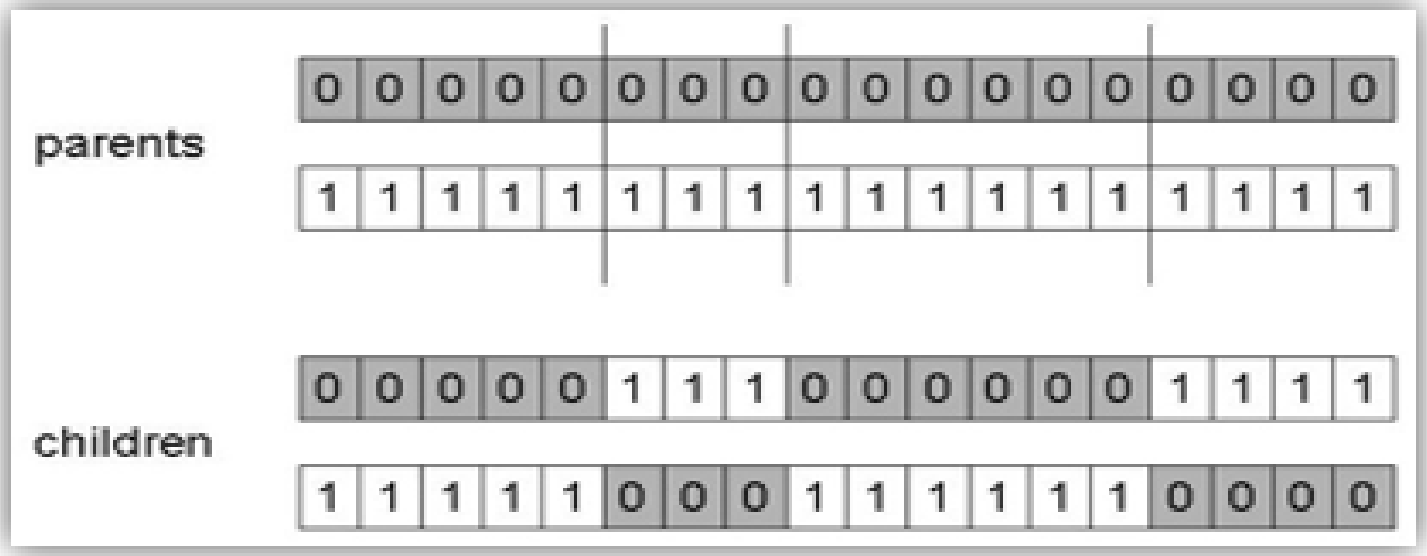
## Alternatif Çaprazlama Operatörleri

- Neden başka çaprazlama operatörlerine ihtiyacımız var?
- Tek noktalı çaprazlamanın performansı değişkenlerin temsilde belirme sırasına bağlıdır
  - Birbirine yakın genleri bir arada tutma olasılığı daha yüksektir
  - Ebeveynin zıt uçlarındaki genleri bir arada tutamaz
  - Bu durum konumsal ön yargı(positional bias) olarak bilinir
  - Problemimizin yapısı bilindiğinde faydalanılabilir, fakat genellikle böyle değildir

# İkili Temsil

## n-noktalı Çaprazlama

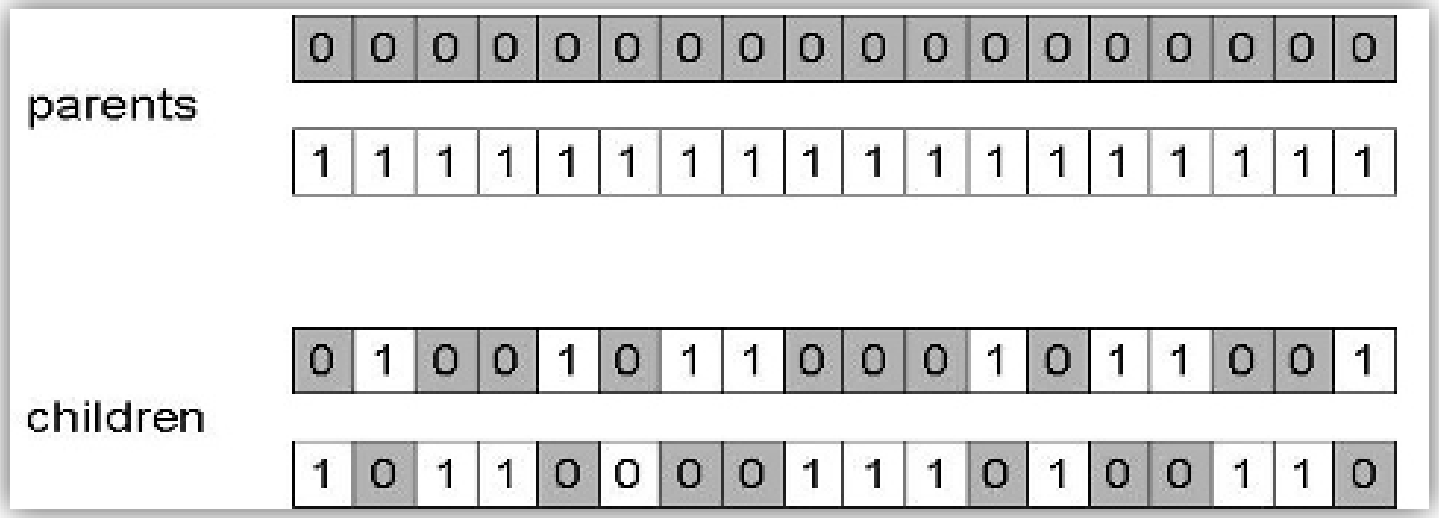
- Rastgele n çaprazlama noktası seçilir
- Bu noktalardan parçalanır
- Ebeveynler arasında değişen parçalar yapıştırılır
- 1-noktalı çaprazlamanın genel halidir(yine de biraz konumsal önyargı içerir)



# İkili Temsil

## Birörnek Çaprazlama(Uniform Crossover)

- Ebeveynlerden birisine **yazı**, diğerine **tura** verelim
- İlk çocuğun her bir geni için bir para(yazı-tura) atalım
- İkinci çocuk için genin ters kopyasını oluşturalım
- Kalıtım konumdan bağımsızdır



# İkili Temsil

## Çaprazlama VEYA Mutasyon

- On yıllık uzun tartışma: Hangisi daha iyi / gerekli
- Cevap:(en azından oldukça geniş bir uzlaşma)
  - Probleme bağlıdır, fakat
  - Genellikle her ikisine de sahip olmak iyidir
  - Her ikisinin de başka rolü vardır
  - Yalnızca mutasyon bulunan EA mümkündür, yalnızca çaprazlama bulunan EA çalışmaz

# İkili Temsil

## Çaprazlama VEYA Mutasyon

- **Keşif(Exploration)**: Arama alanında gelecek vaat eden alanları keşfetmek, yani problem hakkında bilgi edinmek
- **Sömürü(Exploitation)**: Gelecek vaat eden bir alanda optimizasyon yapmak, yani bilgiyi kullanmak
- Aralarında işbirliği VE rekabet var
- Çaprazlama keşif amaçlıdır, iki (ebeveyn) alan "arasında" bir yerde bir alana büyük bir sıçrama yapar
- Mutasyon sömürücüdür, rastgele küçük sapmalar yaratır, böylece ebeveynin yakınında (alanında) kalır

# İkili Temsil

## Çaprazlama VEYA Mutasyon

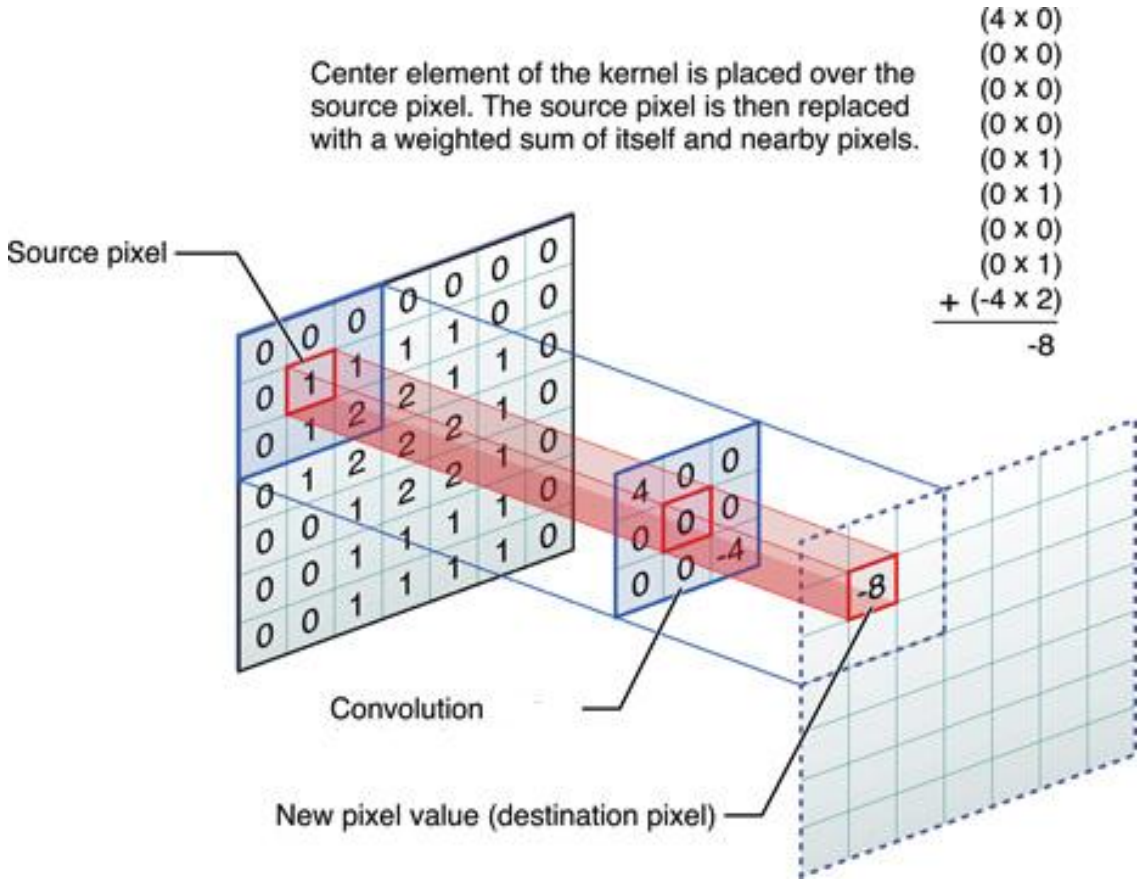
- Yalnızca çaprazlama, iki ebeveynden gelen bilgileri birleştirebilir
- Yalnızca mutasyon yeni bilgiler sunabilir (aleller)
- Çaprazlama popülasyonun alel frekanslarını değiştirmez
- Optimuma ulaşmak için genellikle "şanslı" bir mutasyona ihtiyacınız vardır

# Tamsayı Temsili

- Günümüzde genel olarak sayısal değişkenleri doğrudan kodlamanın daha iyi olduğu kabul edilmektedir (tam sayılar, kayan nokta değişkenleri)
- Bazı problemlerin doğal olarak tamsayı değişkenleri vardır, ör. görüntü işleme parametreleri (sonraki sayfa)
- Diğerleri, sabit bir kümeden kategorik değerler alır, ör. mavi, yeşil, sarı, pembe
- n-noktalı / birörnek çaprazlama çalışır
- Bit-çevirme(bit-fliping) mutasyonu genişletilebilir
  - "Sürünme"(creep), yani benzer değere geçme olasılığı daha yüksektir
    - Her gene p olasılığı ile küçük bir(pozitif veya negatif) değer ekleme
  - Rastgele sıfırlama(random resetting), özellikle kategori verilerinde
    - $p_m$ olasılığı ile yeni bir değer rastgele olarak seçilir
- Çaprazlama, ikili temsildeki ile aynı

# Tamsayı Temsili

## Görüntü İşleme Katlama Örneği

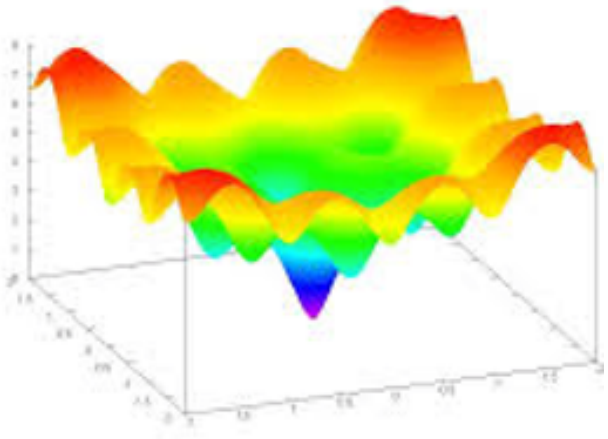



Şekil: Görüntü işleme katlama(convolution) örneği

# Gerçek-değerli ya da Kayan-noktalı Temsil

- Bir çok problem gerçek-değerli bir doğaya sahiptir $f : \mathbb{R}^n \rightarrow \mathbb{R}$
- Örnek: Ackley'in fonksiyonu(evrimsel hesaplamada sıklıkla kullanılır)

$$f(x) = -20 \cdot e^{-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}} - e^{\frac{1}{n}\sum_{i=1}^{n} cos(2\pi x_i)} + 20 + e$$



# Gerçek-değerli ya da Kayan-noktalı Temsil

## Gerçek değerlerin bit dizileri ile temsili

$\{a_1, \cdots, a_n\} \in \{0,1\}^L$ şeklinde temsil edilen $z \in [x,y] \subseteq \mathbb{R}$

- $[x,y] \rightarrow \{0,1\}^L$ ters çevrilebilir olmalıdır(genotip başına bir fenotip)
- $\Gamma : \{0,1\}^L \rightarrow [x,y]$ temsili tanımlar
  $\Gamma(a_1, \cdots, a_L) = x + \frac{y-x}{2^L-1} \cdot (\sum_{j=0}^{L-1} a_{L-j} \cdot 2^j)$
- Sonsuz değer aralığında sadece $2^L$ değer mevcut
- L, olası maksimum çözüm hassasiyetini belirler
- Yüksek hassasiyet $\rightarrow$ uzun kromozomlar (yavaş eğitim)

# Gerçek-değerli ya da Kayan-noktalı Temsil

## Birörnek Mutasyon

- Kayan noktalı sayıların genel şeması
  $\overline{x} =< x_1, \cdots, x_l > \rightarrow \overline{x'} =< x'_1, \cdots, x'_l >$
  $x_i, x'_i \in [LB_i, UB_i]$
- Birörnek mutasyon

  $\overline{x'}$, $[LB_i, UB_i]$ arasından rastgele olarak seçilir
- İkili temsildeki bit çevirmeye(bit-fliping) ve tamsayı temsilindeki rastgele sıfırlamaya(random reseting) benzer

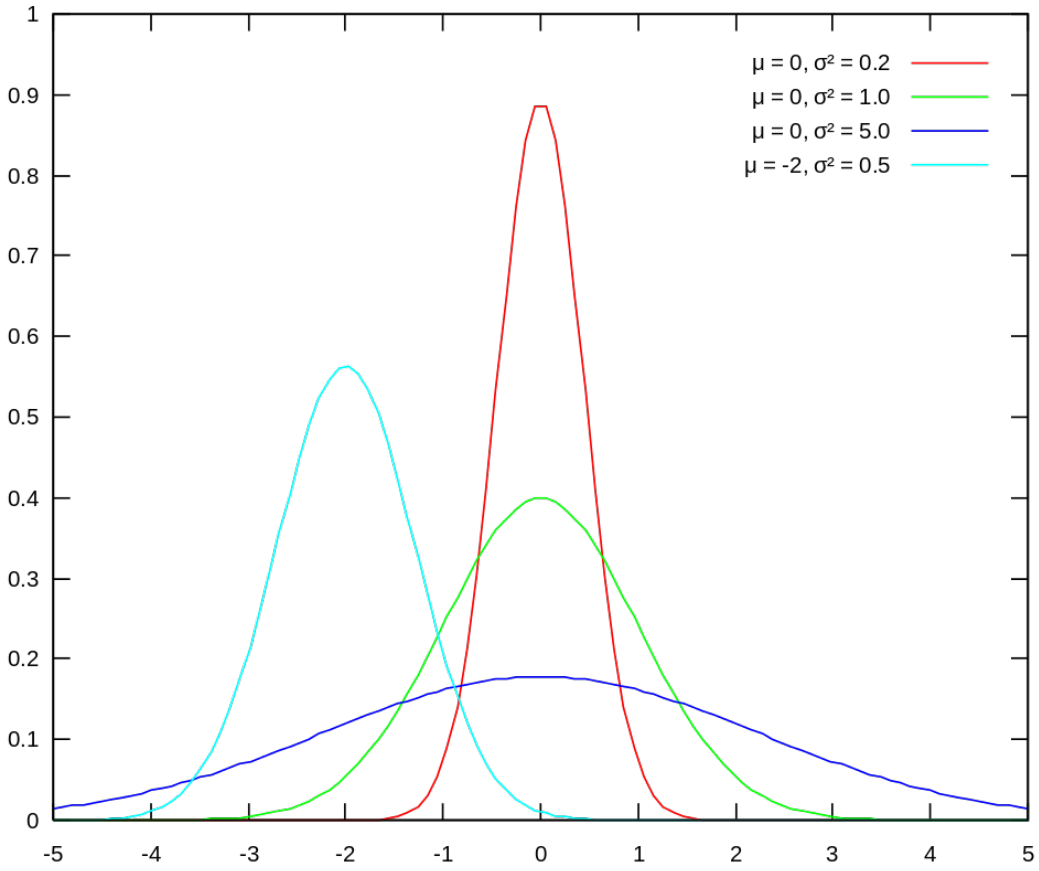
# Gerçek-değerli ya da Kayan-noktalı Temsil

## Birörnek Olmayan Mutasyon

- Birörnek olmayan mutasyonlar
  - Çoğu şema olasılıklıdır, ancak genellikle değerde yalnızca küçük bir değişiklik yapar
  - En yaygın yöntem, her değişkene $N(0, \sigma)$ Gauss dağılımından alınan rastgele sapma eklemek ve uygun aralığa kısıtlamaktır($[LB_i, UB_i]$ varsa) $x_i' = x_i + N(0, \sigma)$
  - Standart sapma $\sigma$, mutasyon adım boyutu, değişim miktarını belirler(dağılımdan seçilen sayıların $\frac{2}{3}$'ü $[-\sigma, +\sigma]$ aralığındadır)

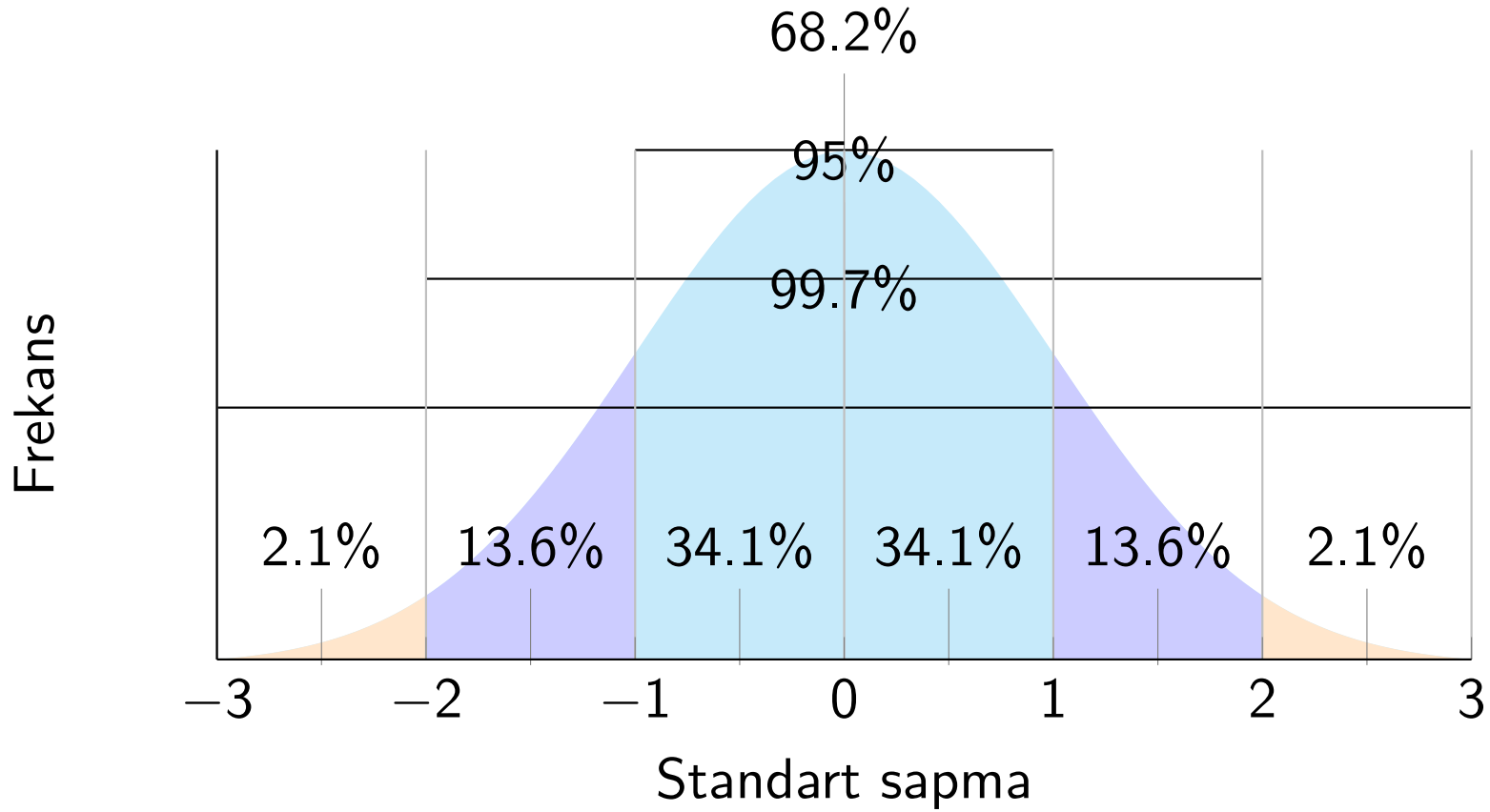
# Gerçek-değerli ya da Kayan-noktalı Temsil

## Normal(Gauss) Dağılım



# Gerçek-değerli ya da Kayan-noktalı Temsil

## Normal(Gauss) Dağılım




Şekil: Normal(Gauss) Dağılım

# Gerçek-değerli ya da Kayan-noktalı Temsil

## Kendinden Uyarlamalı(Self-Adaptive) Mutasyon

- Adım boyutları genoma dahil edilir, varyasyon ve seçime dahil edilir $< x_1, \cdots, x_n, \sigma >$
- Mutasyon adım boyutu kullanıcı tarafından ayarlanmaz, ancak çözümle birlikte gelişir
- Evrimsel araştırma sürecinin farklı aşamalarında farklı mutasyon stratejileri uygun olabilir

# Gerçek-değerli ya da Kayan-noktalı Temsil

## Kendinden Uyarlamalı(Self-Adaptive) Mutasyon

- Önce $\sigma$'yı mutasyona uğrat
- Net mutasyon etkisi $< x, \sigma > \rightarrow < x', \sigma' >$
- Sıralama önemli:
  - önce $\sigma \rightarrow \sigma'$
  - sonra $x \rightarrow x' = x + N(0, \sigma)$
- Gerekçe: yeni $< x', \sigma' >$ iki kez değerlendirilir
  - Eğer $f(x')$ iyiyse $x'$ iyidir
  - $\sigma'$, oluşturduğu $x'$ iyiyse iyidir
- Mutasyon sırasını tersine çevirmek işe yaramaz

# Gerçek-değerli ya da Kayan-noktalı Temsil

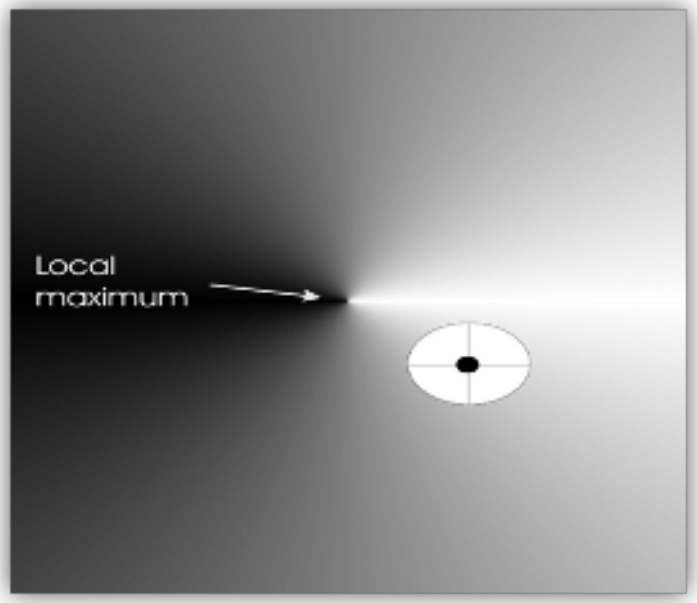
## Bir $\sigma$ ile ilintisiz(uncorrelated) mutasyon

- Kromozomlar: $< x_1, \cdots, x_n, \sigma >$
  - $\sigma' = \sigma \cdot e^{\tau \cdot N(0,1)}$
  - $x_i' = x_i + \sigma' \cdot N(0,1)$
- Tipik olarak "öğrenme oranı(learning rate)" $\tau \propto \frac{1}{\sqrt{n}}$
- Sınır kuralı: $\sigma' < \varepsilon_0 \Rightarrow \sigma' = \varepsilon_0$

# Gerçek-değerli ya da Kayan-noktalı Temsil

## Bir $\sigma$ ile ilintisiz(uncorrelated) mutasyon

Eşit olasılığa sahip mutantlar




Daire: Oluşturulma şansı aynı olan mutantlar

# Gerçek-değerli ya da Kayan-noktalı Temsil

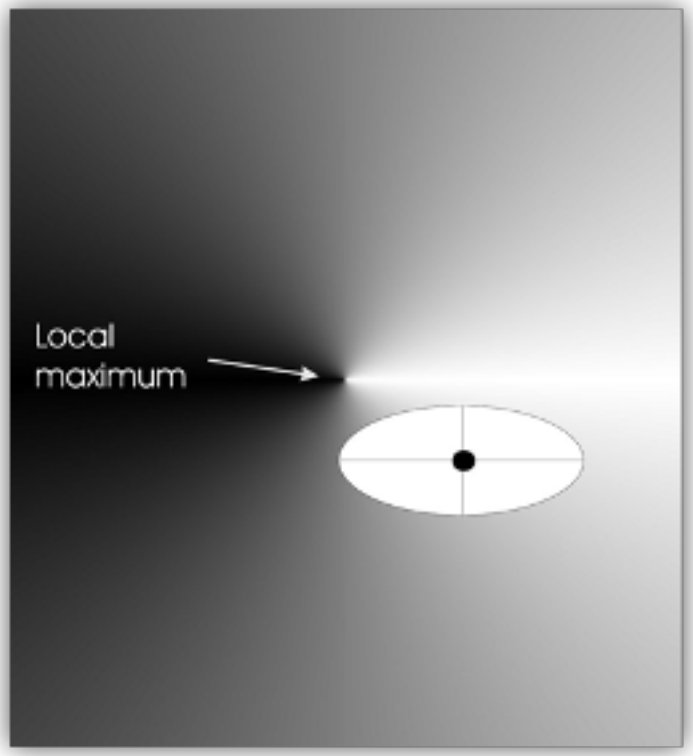
## n adet $\sigma$ ile ilintisiz(uncorrelated) mutasyon

- Kromozomlar: $< x_1, \cdots, x_n, \sigma_1, \cdots, \sigma_n >$
  - $\sigma_i' = \sigma_i \cdot e^{\tau' \cdot N(0,1) + \tau \cdot N_i(0,1)}$
  - $x_i' = x_i + \sigma_i' \cdot N_i(0,1)$
- İki öğrenme oranı:
  - $\tau'$ genel öğrenme oranı
  - $\tau$ koordinat tabanlı öğrenme oranı
- $\tau' \propto \frac{1}{\sqrt{2n}}$ ve $\tau \propto \frac{1}{\sqrt{2\sqrt{n}}}$
- Sınır kuralı: $\sigma_i' < \varepsilon_0 \Rightarrow \sigma_i' = \varepsilon_0$

# Gerçek-değerli ya da Kayan-noktalı Temsil

## n adet $\sigma$ ile ilintisiz(uncorrelated) mutasyon

Eşit olasılığa sahip mutantlar




Elips: Oluşturulma şansı aynı olan mutantlar

# Gerçek-değerli ya da Kayan-noktalı Temsil

## İlintili mutasyonlar

- Kromozomlar: $< x_1, \cdots, x_n, \sigma_1, \cdots, \sigma_n, \alpha_1, \cdots, \alpha_k >$, $k = \frac{n \cdot (n-1)}{2}$
- Kovaryans matrisi(covariance matrix) C şu şekilde tanımlanır:
  - $c_{ii} = \sigma_i^2$
  - $c_{ij} = 0$ eğer i ve j ilintili değilse
  - $c_{ij} = \frac{1}{2}(\sigma_i^2 - \sigma_j^2) tan(2\alpha_{ij})$ eğer i ve j ilintili ise

# Gerçek-değerli ya da Kayan-noktalı Temsil

## İlintili mutasyonlar

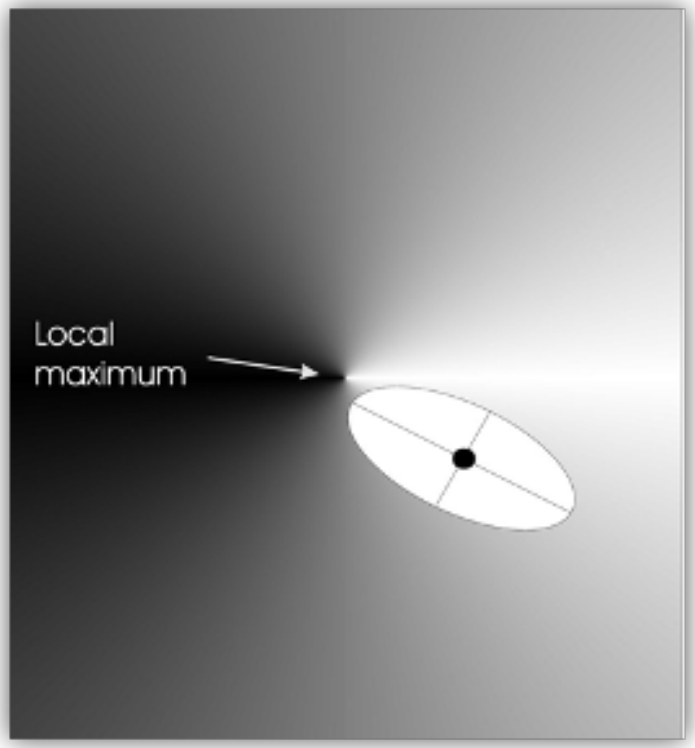
Mutasyon mekanizması şöyle olur:

- $\sigma_i' = \sigma_i \cdot e^{\tau' \cdot N(0,1) + \tau \cdot N_i(0,1)}$
- $\alpha_j' = \alpha_j + \beta \cdot N(0,1)$
- $x' = x + N(0, C')$
  - $x, < x_1, \cdots, x_n >$ vektörünü temsil eder
  - $C'$, $\alpha$ değerlerinin mutasyonunun ardından kovaryans matrisi $C$'yi temsil eder
- $\tau \propto \frac{1}{\sqrt{2\sqrt{n}}}$, $\tau' \propto \frac{1}{\sqrt{2n}}$ ve $\beta \approx 5°$
- $\sigma_i' < \varepsilon_0 \Rightarrow \sigma_i' = \varepsilon_0$
- $|\alpha_j'| > \pi \Rightarrow \alpha_j' = \alpha_j' - 2\pi sign(\alpha_j')$
- Kovaryans Matris Adaptasyon Evrim Stratejisi (CMA-ES) muhtemelen sayısal optimizasyon için en iyi EA'dır, bkz. CEC-2005 yarışması

# Gerçek-değerli ya da Kayan-noktalı Temsil

## İlintili mutasyonlar

Eşit olasılığa sahip mutantlar




Elips: Oluşturulma şansı aynı olan mutantlar

# Gerçek-değerli ya da Kayan-noktalı Temsil

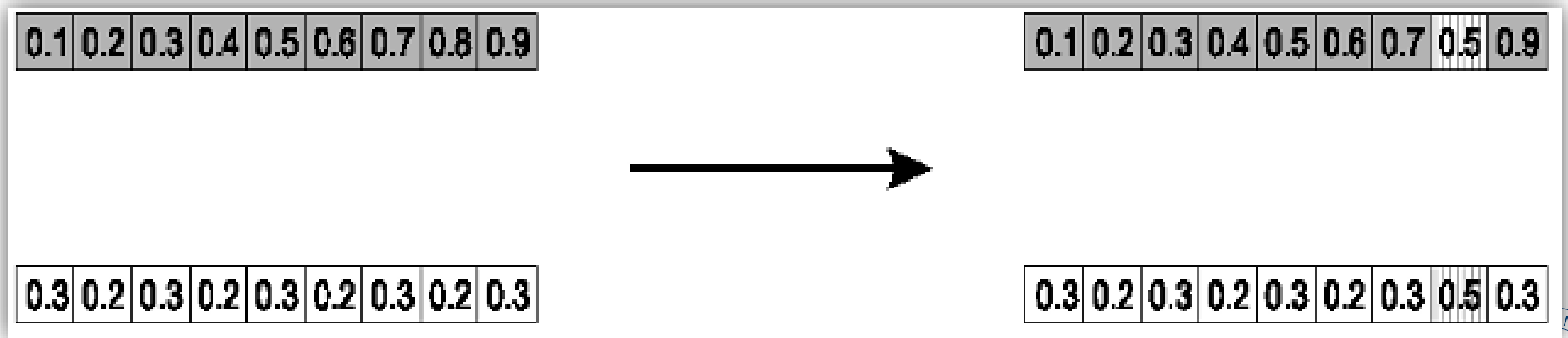
## Çaprazlama Operatörleri

- Ayrık:
  - z çocuğundaki her bir alel değeri, ebeveynlerin birinden(x,y) eşit olasılıkla gelir $z_i = x_i$ *veya* $y_i$
  - n-noktalı veya birörnek kullanılabilir
- Ortada:
  - Ebeveynler arasında çocuk oluşturma fikrinden yararlanır(*aritmetik rekombinasyon* olarak da bilinir)
  - $z_i = \alpha \cdot x_i + (1 - \alpha)y_i$, $\alpha$ değeri: $0 \leq \alpha \leq 1$
  - $\alpha$ parametresinin değeri:
    - sabit: birörnek aritmetik çaprazlama
    - değişken (Örn: popülasyonun yaşına bağlı)
    - her seferinde rastgele seçilir

# Gerçek-değerli ya da Kayan-noktalı Temsil

## Tek Aritmetik Çaprazlama

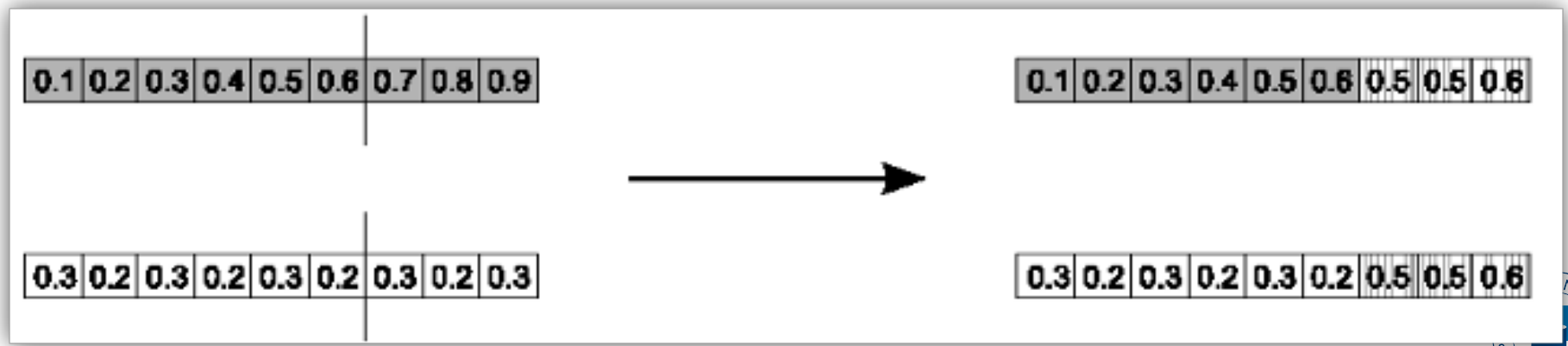
- Ebeveynler: $< x_1, \cdots, x_n >$ ve $< y_1, \cdots, y_n >$
- Rastgele bir tek gen(k) seçin
- Çocuk 1: $< x_1, \cdots, x_{k-1}, \alpha \cdot y_k + (1 - \alpha) \cdot x_k, \cdots, x_n >$
- Diğer çocuk için tersini oluştur, ör: $\alpha = 0.5$



# Gerçek-değerli ya da Kayan-noktalı Temsil

## Basit Aritmetik Çaprazlama

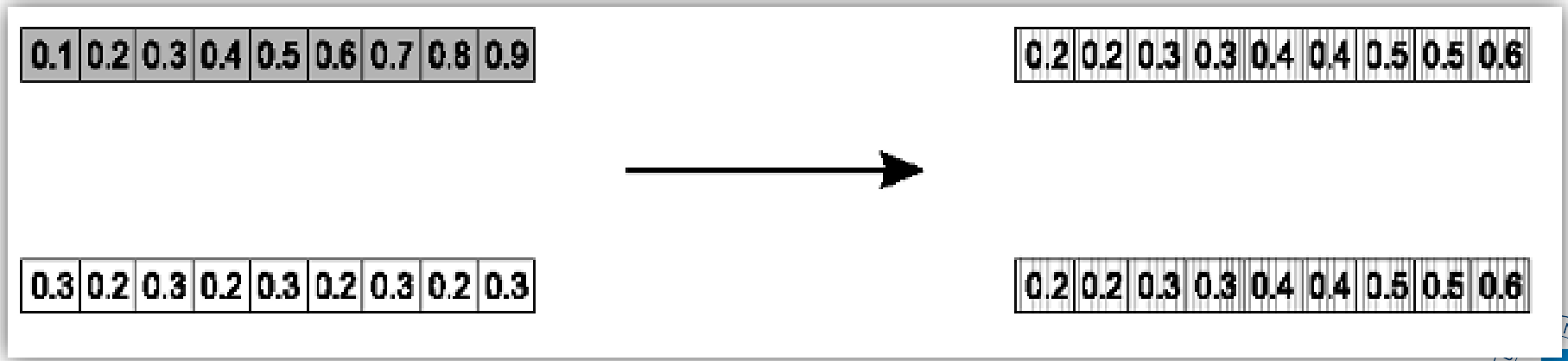
- Ebeveynler: $< x_1, \cdots, x_n >$ ve $< y_1, \cdots, y_n >$
- Rastgele bir tek gen(k) seçin ve bu noktadan sonra değerleri karıştırın
- Çocuk 1:
  $< x_1, \cdots, x_k, \alpha \cdot y_{k+1} + (1-\alpha) \cdot x_{k+1}, \cdots, \alpha \cdot y_n + (1-\alpha) \cdot x_n >$
- Diğer çocuk için tersini oluştur, ör: $\alpha = 0.5$



# Gerçek-değerli ya da Kayan-noktalı Temsil

## Tam Aritmetik Çaprazlama

- Ebeveynler: $< x_1, \cdots, x_n >$ ve $< y_1, \cdots, y_n >$
- Rastgele bir tek gen(k) seçin ve bu noktadan sonra değerleri karıştırın
- Çocuk 1: $\alpha \cdot \overline{x} + (1 - \alpha) \cdot \overline{y}$
- Diğer çocuk için tersini oluştur, ör: $\alpha = 0.5$



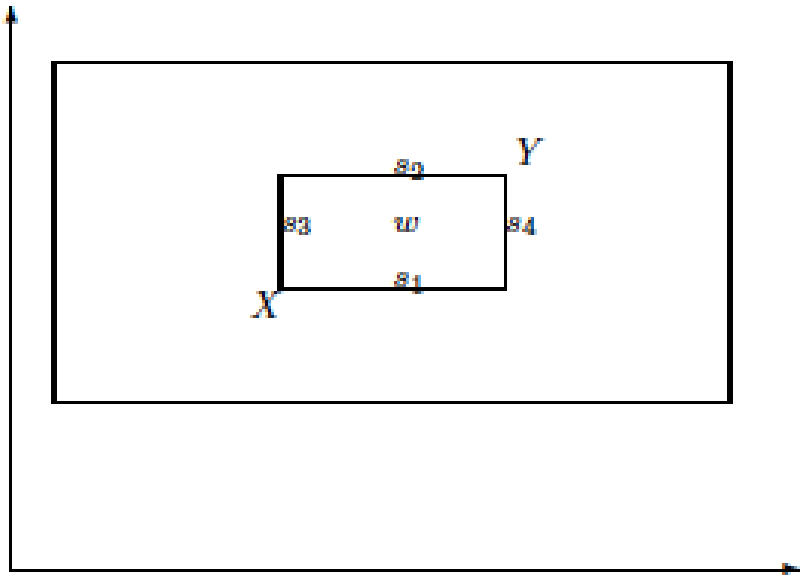
# Gerçek-değerli ya da Kayan-noktalı Temsil

## Karışım(Blend) Çaprazlama

- Ebeveynler: $< x_1, \cdots, x_n >$ ve $< y_1, \cdots, y_n >$
- $x_i < y_i$ olduğunu varsayalım
- $d_i = y_i - x_i$
- Rastgele örnek: $z_i = [x_i - \alpha d_i, x_i + \alpha d_i]$ aralığında

# Gerçek-değerli ya da Kayan-noktalı Temsil

## Farklı olası çocuklara genel bakış




- Tek aritmetik çaprazlama: $\{s_1, s_2, s_3, s_4\}$
- Basit / tam aritmetik çaprazlama: iç kutu, ($\alpha = 0.5$ için w)
- Karışım çaprazlama: dış kutu

# Gerçek-değerli ya da Kayan-noktalı Temsil

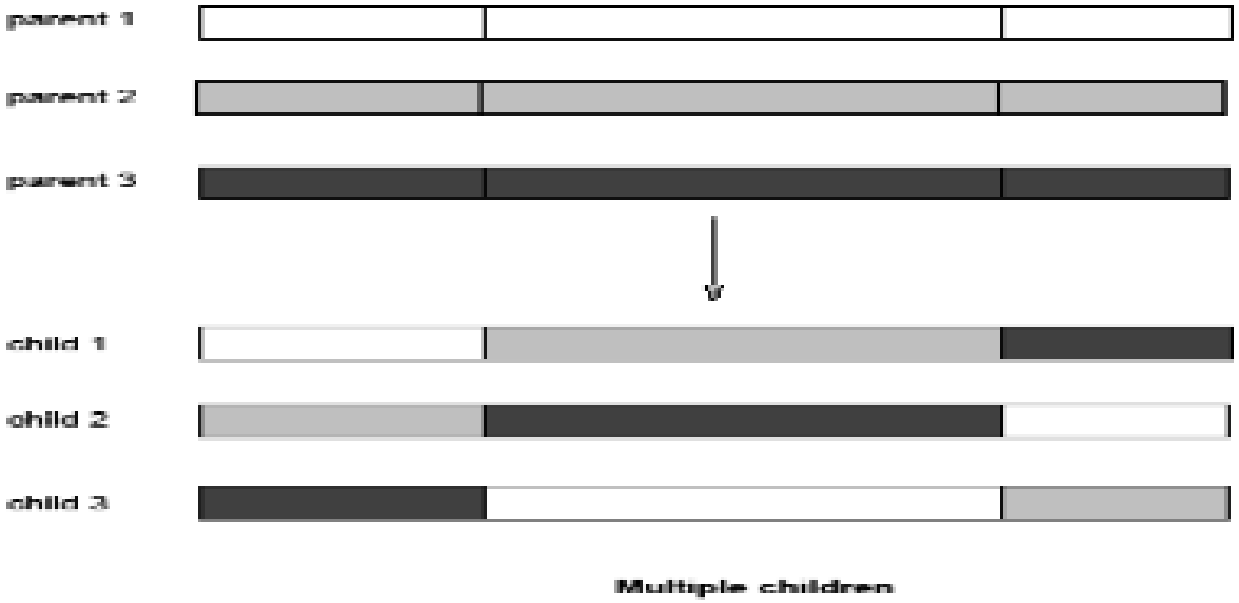
## Çoklu Ebeveyn Çaprazlaması

- Evrimsel hesaplamayı doğanın pratikleri tarafından kısıtlamayabiliriz
- Mutasyonun n=1 ebeveyni ve "geleneksel" çaprazlamanın n=2 ebeveyni kullandığını düşünürsek, n>2 araştırılmaya değerdir
- 1960'lardan beri ortalıkta, hala nadir ama araştırmalar işe yaradığını gösteriyor

# Gerçek-değerli ya da Kayan-noktalı Temsil

## Çoklu Ebeveyn Çaprazlaması, tip 1

- Fikir: ebevenleri parçalara ayırın ve yeniden birleştirin
- Örnek: n ebeveyn için çapraz geçiş(çaprazlama)
  - $n-1$ çaprazlama noktası seçin(her ebeveynde aynı noktalar)
  - Ebeveynlerin parçalarından n tane çocuğu, köşegen boyunca etrafını sararak oluşturun




- Bu operatör 1 noktalı çaprazlamayı genelleştirir

# Gerçek-değerli ya da Kayan-noktalı Temsil

## Çoklu Ebeveyn Çaprazlaması, tip 2

- Fikir: (gerçek değerli) alellerin aritmetik kombinasyonu
- n ebeveyn için aritmetik çaprazlama
  - Çocuktaki i. alel ebeveynlerdeki i. alellerin ortalamasıdır
- Çocuk ağırlık merkezi olur
- Genetik algoritmada garip olabilir, evrim stratejisinde uzun süredir bilinen ve kullanılan

# Permütasyon Temsili

- Sıralama problemleri özel bir tür oluşturur
- Görev(veya çözüm), bazı nesnleri belirli bir sıraya göre düzenlemektir
- Örnek: Üretim planlama: önemli olan hangi öğelerin diğerlerinden önce planlandığıdır(sıra)
- Örnek: Gezgin satıcı problemi(TSP): önemli olan hangi öğelerin yan yana oluştuğudur(komşuluk)
- Bu problemler genellikle bir permütasyon olarak ifade edilir
  - n değişken varsa, temsil, her biri bir kez beliren n tamsayının listesi gibidir

# Permütasyon Temsili

## TSP Örneği

- Problem:
  - Verilen n şehir için
  - En kısa uzunluğa sahip tam turu bul
- Kodlama:
  - Şehirleri $1, 2, \cdots, n$ şeklinde numaralandır
  - Bir tam tur bir permütasyona eşittir(örn: n=4 için [1,2,3,4,], [3,4,2,1] mümkün)
- Arama uzayı BÜYÜKTÜR: 30 şehir için $30! \approx 10^{32}$ mümkün tur sayısı vardır

# Permütasyon Temsili

## Mutasyon

- Normal mutasyon operatörleri kabul edilemez çözümlere yol açar
  - Örneğin bit-tabanlı mutasyon için i. genin değeri $j$ olsun
  - i. gen için $k$ değerine geçmek, $k$'nin iki defa yer almasına ve $j$'nin kaybolmasına neden olacaktır
- Bu nedenle en az iki değer değiştirilmelidir
- Mutasyon parametresi artık bazı operatörlerin her konumda ayrı ayrı uygulamak yerine tüm dizeye bir kez uygulanması olasılığını yansıtır

# Permütasyon Temsili

## Yer değiştirme(swap) mutasyonu

- Rastgele iki alel seç ve yerlerini değiştir

# Permütasyon Temsili

## Araya ekleme(insert) mutasyonu

- Rastgele iki alel seç
- İkinciyi birincinin arkasına gelecek şekilde taşı ve diğerlerini kaydır
- Bu işlem, sıralama ve komşuluk bilgisinin çoğunu koruyacaktır

# Permütasyon Temsili

## Karıştırma(scramble) mutasyonu

- Genlerin rastgele bir alt kümesini seçin
- Bu konumlardaki alelleri rastgele yeniden düzenleyin

# Permütasyon Temsili

## Ters çevirme(inverse) mutasyonu

- Rastgele iki alel seçin ve aralarındaki alt dizeyi ters çevirin
- Çoğu komşuluk bilgisini korur(sadece ikisi bozulur) fakat sıralama bilgisi bozulur

# Permütasyon Temsili

## Çaprazlama operatörleri

- "Normal" çaprazlama operatörleri genellikle kabul edilemez çözümlere yol açacaktır


- İki ebeveynden gelen sıralama ve komşuluk bilgilerini birleştirmeye odaklanan bir çok özelleşmiş operatör tasarlanmıştır

# Permütasyon Temsili

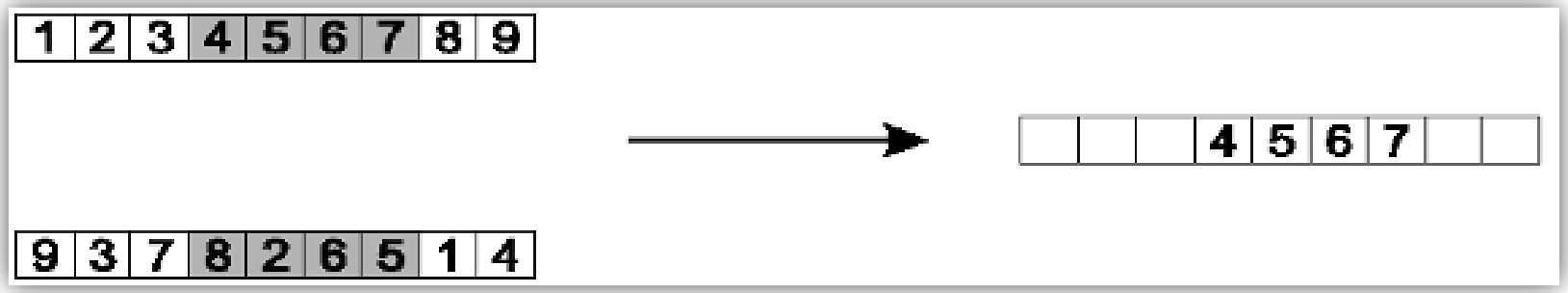
## Order çaprazlama

- Fikir, öğelerin meydana geldiği göreceli düzeni korumaktır
- Sözde kod:
  1. İlk ebeveynden rastgele bir bölüm seçin
  2. Bu bölümü ilk çocuğa kopyalayın
  3. Birinci kısımda olmayan sayıları ilk çocuğa kopyalayın:
     - kopyalanan parçanın kesim noktasından başlayarak,
     - ikinci ebeveyndeki sırayı kullanarak
     - sona ulaşınca başa dönerek
  4. Ebeveyn rollerinin tersine çevrildiği ikinci çocuk için benzer

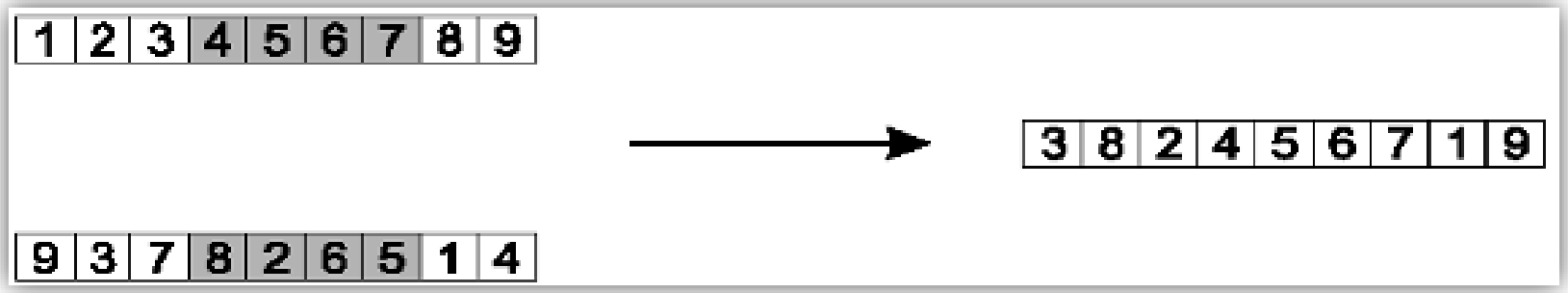
# Permütasyon Temsili

## Order çaprazlama

- Rastgele seçilen alt kümeyi birinci ebeveynden kopyala




- Kalan kısmı ikinci ebeveynden 1,9,3,8,2 sırasıyla kopyala



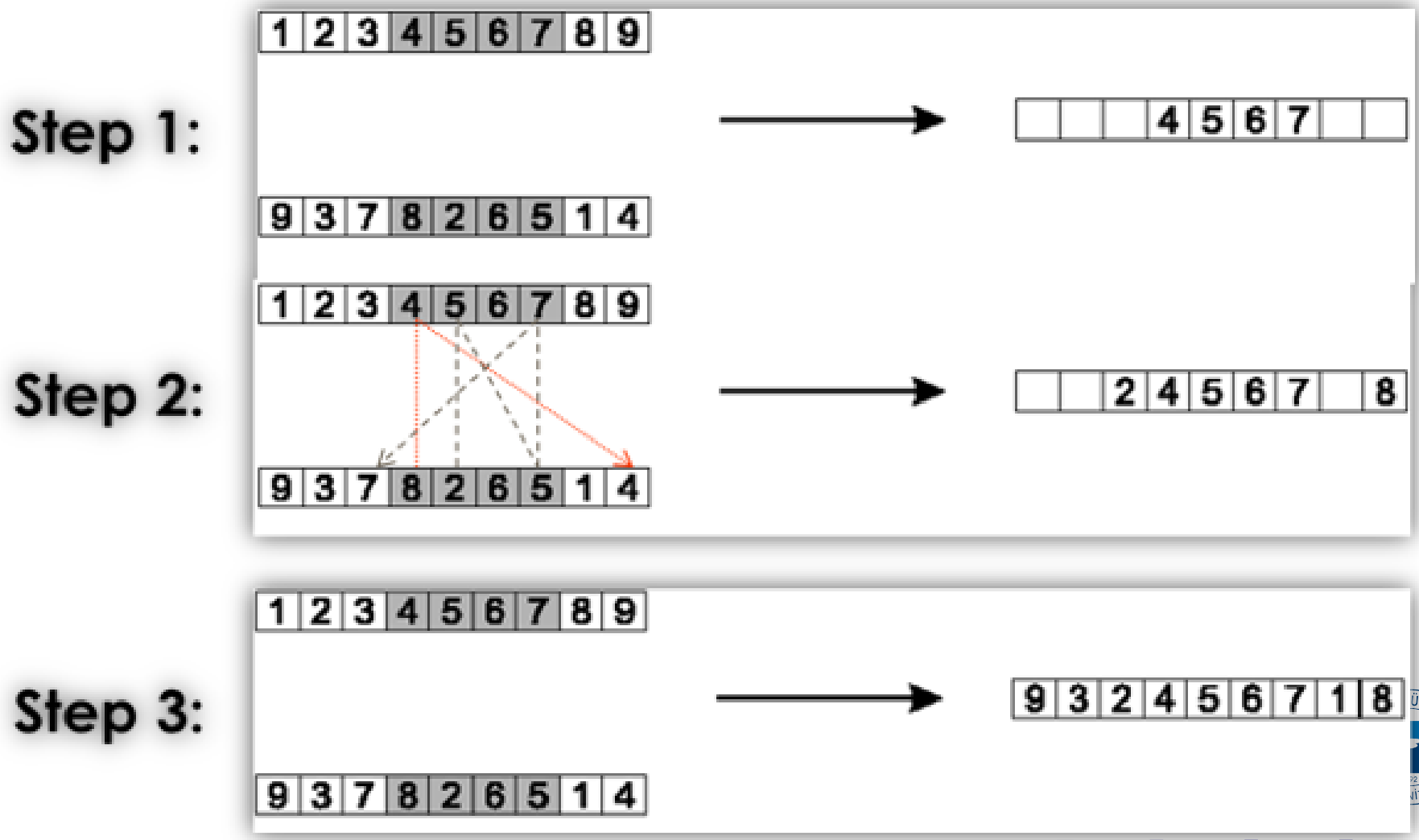
# Permütasyon Temsili

## Kısmen eşleşmiş çaprazlama(PMX)

- P1 ve P2 için sözde kod:
  1. Rastgele iki geçiş noktası seçin ve aralarındaki segmenti ilk ebeveynden (P1) ilk çocuğa kopyalayın.
  2. İlk geçiş noktasından başlayarak, ikinci ebeveynin (P2) bu segmentinde kopyalanmamış olan öğeleri arayın.
  3. Bunların her biri için (i diyelim), P1'den yerine hangi öğenin (j diyelim) kopyalandığını görmek için yavruya bakın.
  4. i'yi P2'de j'nin işgal ettiği konuma yerleştirin, çünkü j'yi oraya koymayacağımızı biliyoruz (zaten dizimizde olduğu gibi).
  5. P2'de j'nin işgal ettiği yer yavruda zaten bir k öğesi tarafından doldurulmuşsa, P2'de k'nin işgal ettiği konuma i'yi koyun.
  6. Çaprazlama bölümdeki unsurları ele aldıktan sonra, bu yavruda kalan pozisyonlar P2'den doldurulabilir ve ikinci çocuk, ebeveyn rolleri tersine çevrilerek benzer şekilde yaratılır.

# Permütasyon Temsili

## Kısmen eşleşmiş çaprazlama(PMX)



# Permütasyon Temsili

## PMX Örnek

# Permütasyon Temsili

## Döngü(cycle) çaprazlama

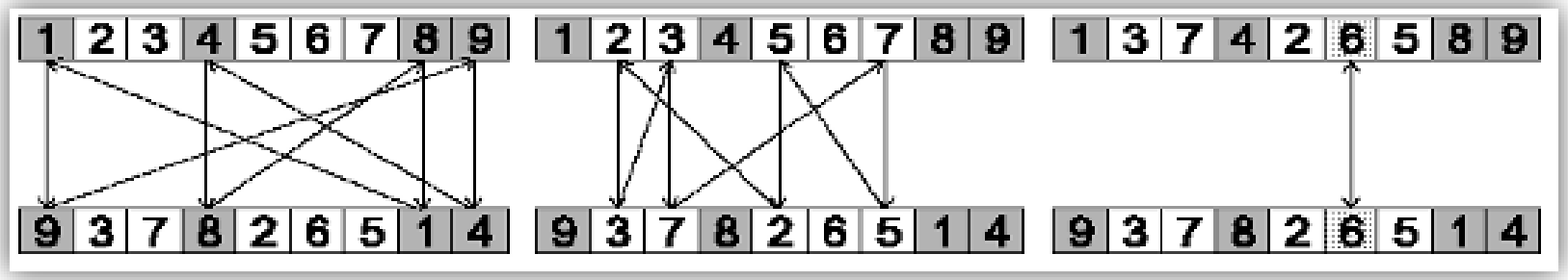
**Temel fikir:** Her alel, konumu ile birlikte bir ebeveynden gelir.
Sözde kod:

1. Aşağıdaki şekilde P1'den alel döngüsü yapın:
   a. P1'in ilk aleli ile başlayın
   b. P2'de *aynı konumdaki* alele bakın
   c. P1'de aynı alelin pozisyonuna gidin
   d. Bu aleli döngüye ekleyin
   e. P1'deki ilk alele rastlayana kadar adım **b**'den **d**'ye kadar olan kısmı tekrarlayın
2. İlk çocuktaki döngünün alellerini birinci ebeveynde sahip oldukları pozisyonlara yerleştirin
3. Sonraki döngüyü ikinci ebeveynden alın

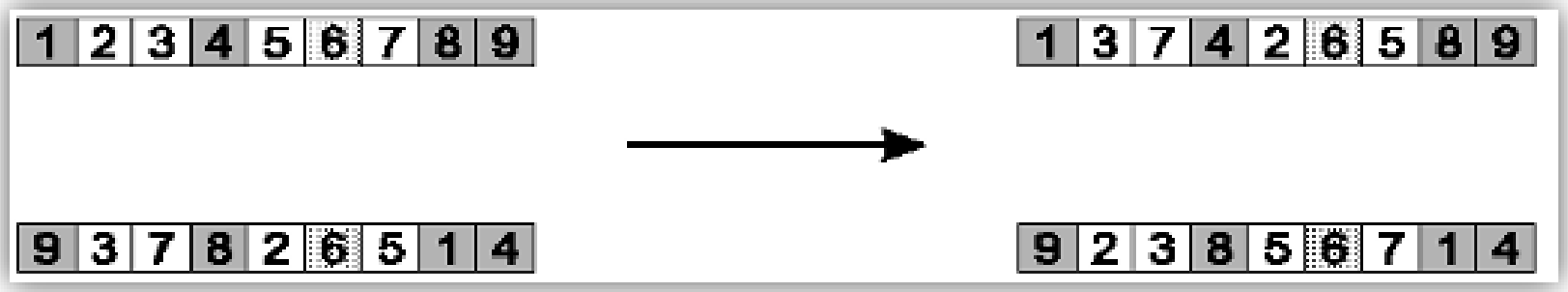
# Permütasyon Temsili

## Döngü(cycle) çaprazlama

- Adım 1: döngüleri belirle




- Adım 2: alternatif döngüleri çocuklara kopyala



# Permütasyon Temsili

## Kenar(edge) çaprazlama

- elemanİki ebeveynde hangi kenarların bulunduğunu listeleyen bir tablo oluşturarak başlar, eğer her ikisi için de ortak bir kenar varsa + ile işaretlenir
- Örn: **[1 2 3 4 5 6 7 8 9]** ve **[9 3 7 8 2 6 5 1 4]**

| Element | Edges | Element | Edges |
|---|---|---|---|
| 1 | 2,5,4,9 | 6 | 2,5+,7 |
| 2 | 1,3,6,8 | 7 | 3,6,8+ |
| 3 | 2,4,7,9 | 8 | 2,7+, 9 |
| 4 | 1,3,5,9 | 9 | 1,3,4,8 |
| 5 | 1,4,6+ | | |

# Permütasyon Temsili

## Kenar(edge) çaprazlama

1. Rastgele bir başlangıç elemanı seçin ve çocuğa ekleyin
2. `mevcut_eleman=kayit` atamasını yapın
3. Tabloda mevcut elemana olan tüm referansları silin
4. Listeyi mevcut eleman için kontrol edin:
   - Eğer ortak kenar varsa, sıradaki eleman olarak bunu seçin
   - Aksi halde listedeki en kısa listeye sahip olan elemanı seçin
   - Eşitlik durumunda rastgele(standart olması için küçük olan)
5. Boş bir liste gelmişse
   - Rastgele bir eleman seçilir

# Permütasyon temsili

## Kenar(edge) çaprazlama

| Element | Edges | Element | Edges |
|---|---|---|---|
| 1 | 2,5,4,9 | 6 | 2,5+,7 |
| 2 | 1,3,6,8 | 7 | 3,6,8+ |
| 3 | 2,4,7,9 | 8 | 2,7+, 9 |
| 4 | 1,3,5,9 | 9 | 1,3,4,8 |
| 5 | 1,4,6+ | | |

| Choices | Element selected | Reason | Partial result |
|---|---|---|---|
| All | 1 | Random | [1] |
| 2,5,4,9 | 5 | Shortest list | [1 5] |
| 4,6 | 6 | Common edge | [1 5 6] |
| 2,7 | 2 | Random choice (both have two items in list) | [1 5 6 2] |
| 3,8 | 8 | Shortest list | [1 5 6 2 8] |
| 7,9 | 7 | Common edge | [1 5 6 2 8 7] |
| 3 | 3 | Only item in list | [1 5 6 2 8 7 3] |
| 4,9 | 9 | Random choice | [1 5 6 2 8 7 3 9] |
| 4 | 4 | Last element | [1 5 6 2 8 7 3 9 4] |

# Permütasyon Temsili

## Kenar çaprazlama örnek

| Eleman | Komşuluk |
|---|---|
| 1 | |
| 2 | |
| 3 | |
| 4 | |
| 5 | |
| 6 | |
| 7 | |
| 8 | |

| Seçenekler | Seçilen | Sebep | Kısmi Çözü |
|---|---|---|---|
| hepsi | 5 | rastgele seçim | 5 |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

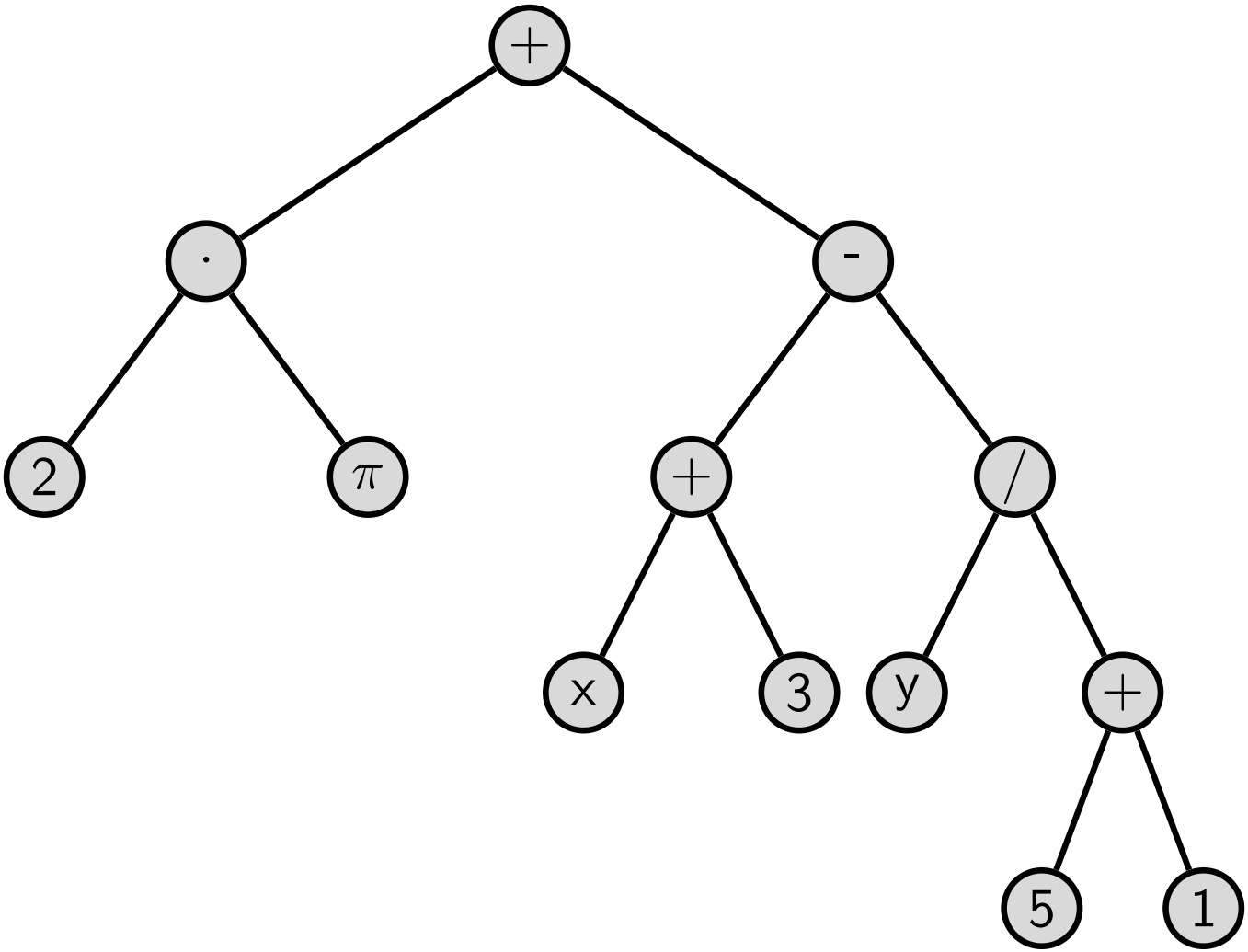
# Ağaç Temsili

- Ağaçlar evrensel bir biçimdir
- Aritmetik ifade: $2 \cdot \pi + ((x + 3) - \frac{y}{5+1})$
- Mantıksal ifade: $(x \wedge \mathit{True}) \implies ((x \vee y) \vee (z \iff (x \wedge y)))$
- Kaynak kod:

```
i = 1;
while(i < 20)
{
        i = i + 1;
}
```

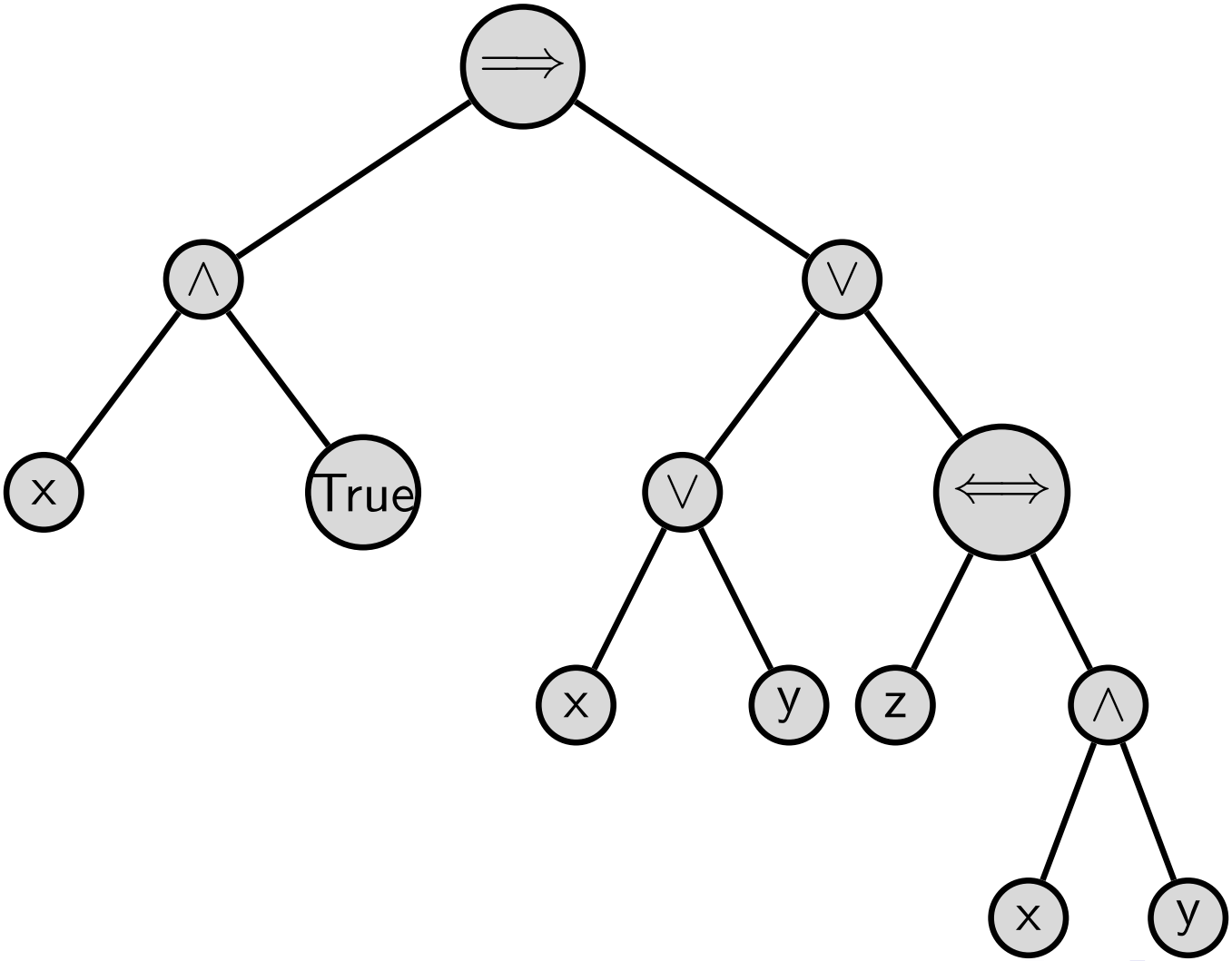
# Ağaç Temsili

$2 \cdot \pi + ((x + 3) - \frac{y}{5+1})$



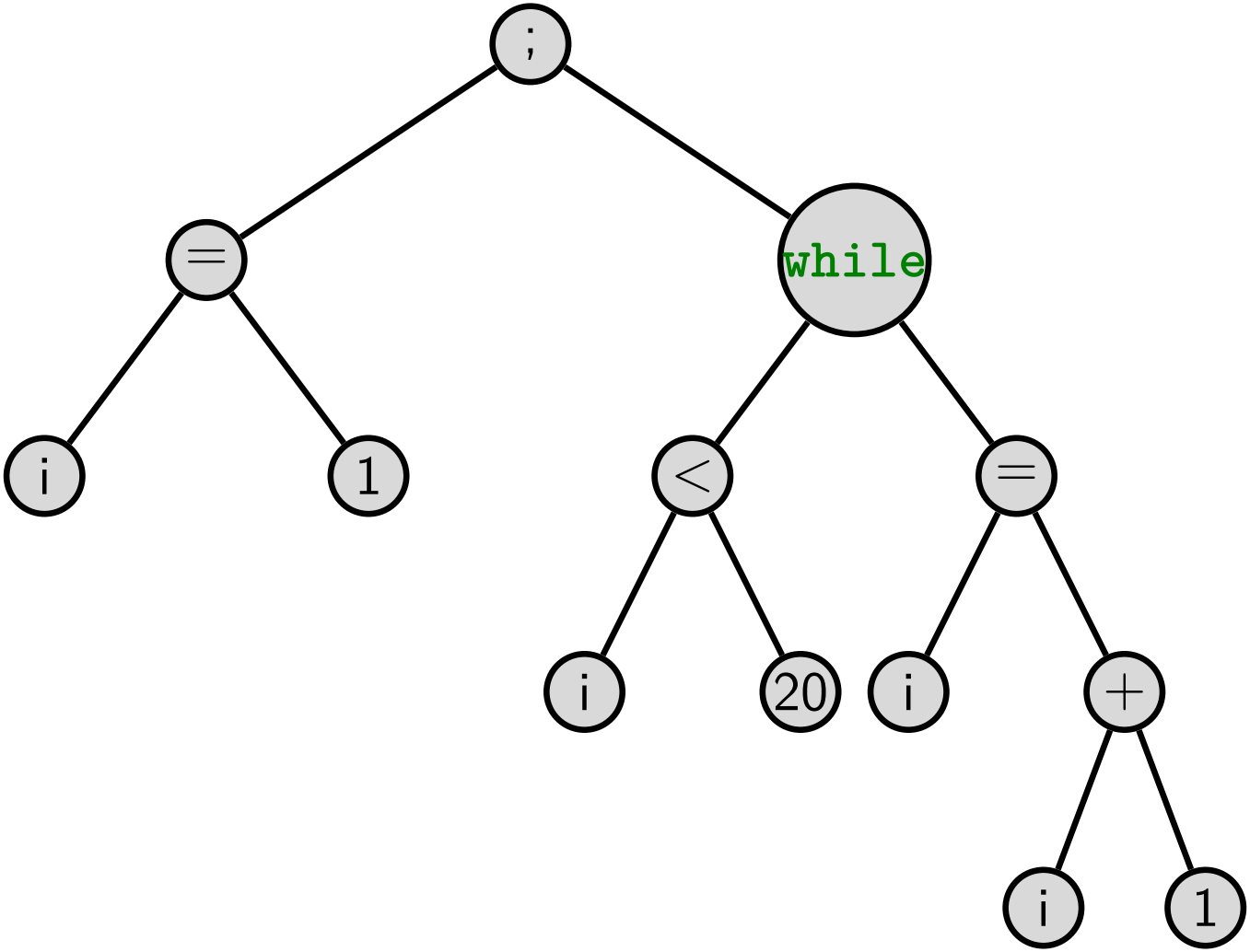
# Ağaç Temsili

$(x \wedge True) \implies ((x \vee y) \vee (z \iff (x \wedge y)))$



# Ağaç Temsili

```
i=1; while(i<20){i=i+1;}
```



# Ağaç Temsili

- *Genetik algoritma, evrim stratejisi, evrimsel programlama* yöntemlerinde kromozomlar doğrusal yapılardır(bit dizesi, tamsayı dizesi, gerçek-sayı vektörü, permütasyon gibi)
- Ağaç yapılı kromozomlar doğrusal olmayan yapılardır
- *Genetik algoritma, evrim stratejisi, evrimsel programlama* yöntemlerinde kromozom boyutu sabittir
- Genetik programlamadaki ağaçlar derinlik ve genişlik olarak değişiklik gösterebilir

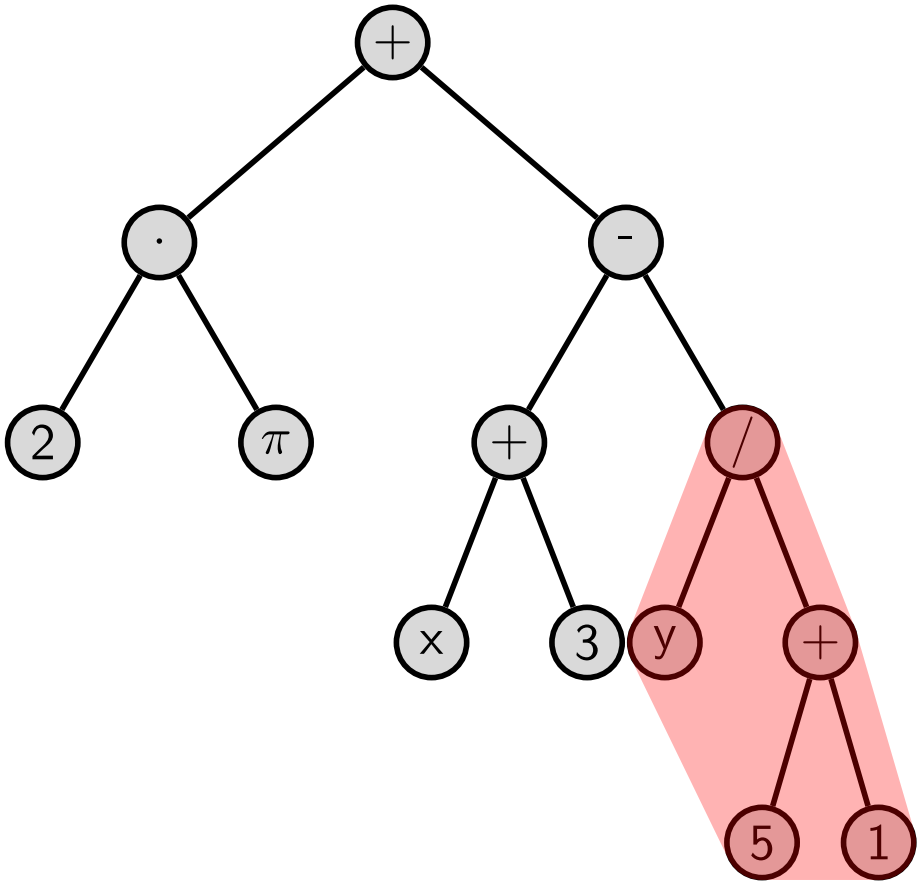
# Ağaç Temsili

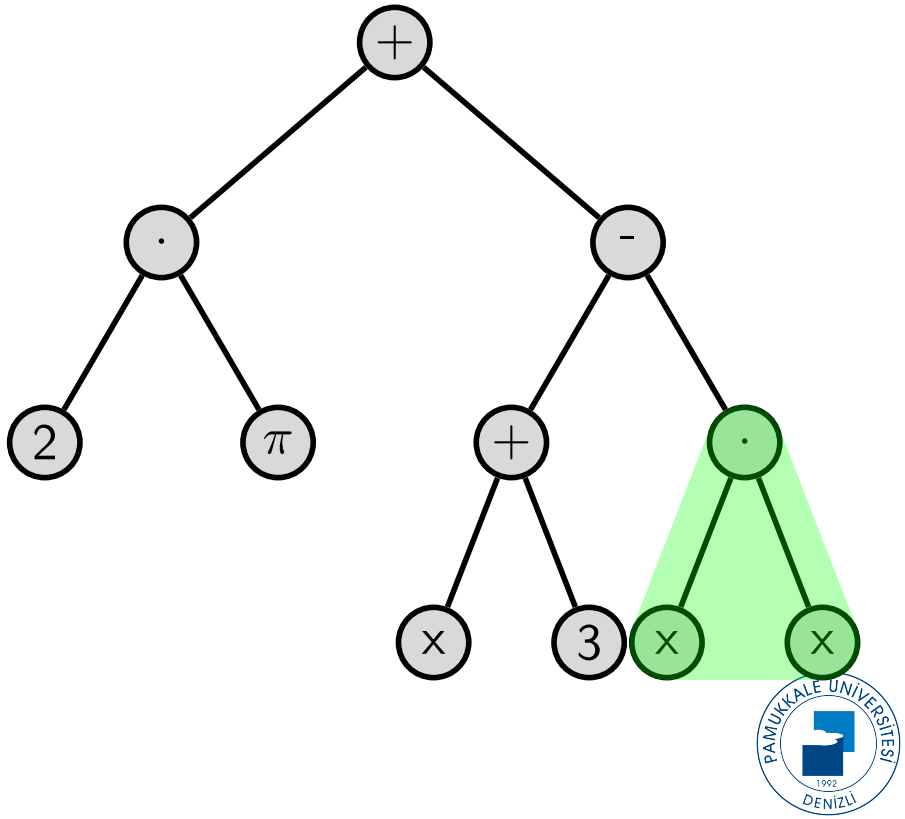
- Sembolik ifadeler aşağıdakilerle tanımlanabilir
  - Terminal kümesi T
  - Fonksiyon kümesi F(fonksiyon sembollerinin arite bilgisi/parametre sayısı ile)
- Aşağıdaki genel özyinelemeli tanımlama benimsenir
  - Her $t \in T$ doğrudur
  - Eğer $f \in F$, $arity(f) = n$ ve $e_1, \cdots, e_n$ doğru ifadelerse $f(e_1, \cdots, e_n)$ doğrudur
  - Doğru ifadelerin başka biçimi yoktur
- Genel olarak genetik programlamada ifadelerin türü yoktur(aynı türden)(kapanış(closure) özelliği: herhangi bir $f \in F$, $g \in F$ fonksiyonunu argüman olarak alabilir)

# Ağaç Temsili

## Mutasyon

En yaygın mutasyon: rastgele seçilen alt ağacı rastgele oluşturulmuş ağaçla değiştirin



# Ağaç Temsili

## Mutasyon

- Mutasyonun iki parametresi vardır:
  - Mutasyonu seçme olasılığı $p_m$
  - Değiştirilecek alt ağacın kökü olarak bir iç nokta seçme olasılığı
- Dikkat çekici bir şekilde $p_m$'nin 0 [2] veya 0.05 gibi çok küçük olması önerilir[1]
- Çocuğun boyutu ebeveynin boyutunu aşabilir

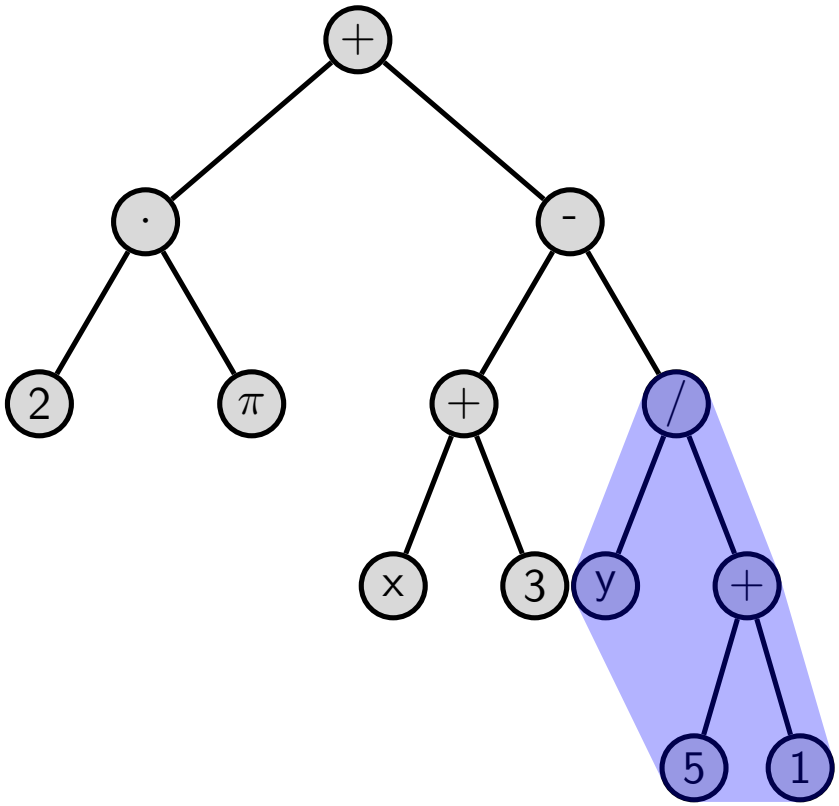
# Ağaç Temsili

## Rekombinasyon

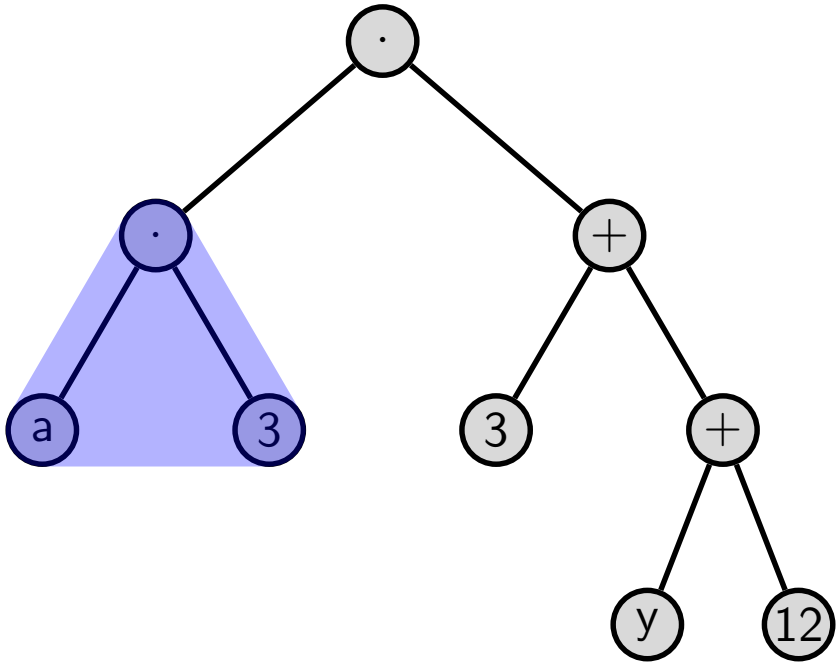
- En yaygın rekombinasyon: ebeveynler arasında rastgele seçilen iki alt ağaç değişimi
- Rekombinasyonun iki parametresi vardır:
  - Rekombinasyonu seçme olasılığı $p_c$
  - Her ebeveynin içinde çaprazlama noktası olarak bir iç nokta seçme olasılığı
- Yavruların boyutu ebeveynlerinkini aşabilir

# Ağaç Temsili

## Rekombinasyon



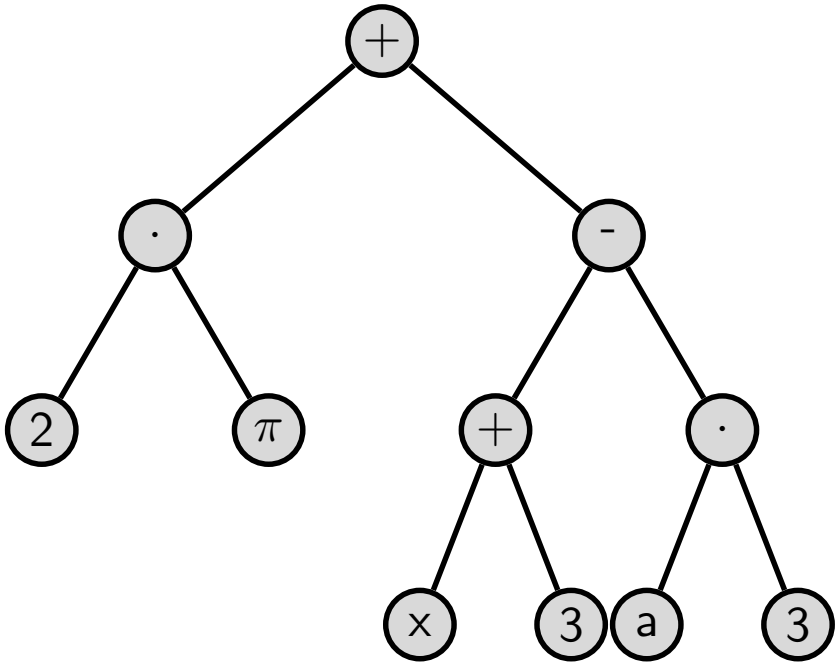

Şekil: Ebeveyn 1



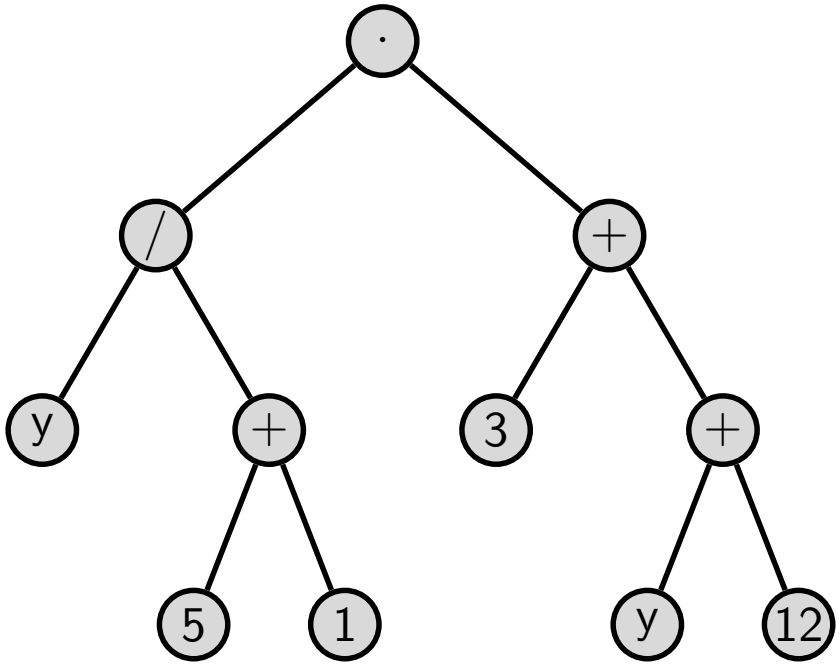

Şekil: Ebeveyn 2

# Ağaç Temsili

## Rekombinasyon




Şekil: Çocuk 1




Şekil: Çocuk 2

# Kaynaklar


Wolfgang Banzhaf, Peter Nordin, Robert E Keller, and Frank D Francone.
*Genetic programming: An Introduction*.
Springer, 1998.

John R Koza and John R Koza.
*Genetic programming: on the programming of computers by means of natural selection*, volume 1.
MIT press, 1992.