CENG 415 Evrimsel Hesaplama

Bölüm 6: Popüler Evrimsel Algoritma Çeşitleri

Şevket Umut Çakır

Pamukkale Üniversitesi

3 Aralık 2020

Anahat

1. Geçmiş Evrimsel Algoritma Çeşitleri
   - Genetik Algoritmalar
   - Evrim Stratejileri
   - Evrimsel Programlama
   - Genetik Programlama
2. Daha Yeni Sürümler
   - Diferansiyel Evrim
   - Parçacık Sürüsü Optimizasyonu
   - Dağıtım Algoritmalarının Tahmini
   - Sınıflandırıcı Sistemlerin Öğrenilmesi

Genetik Algoritmalar

Hızlı Genel Bakış

- 1960'larda ABD'de geliştirildi
- İlk isimler: J. Holland, K. DeJong, D. Goldberg
- Genellikle şunlara uygulanır:
  - Ayrık fonksiyon optimizasyonu
  - Kıyaslama(benchmark)
  - Basit problemler ikili gösterim
- Özellikler:
  - Çok hızlı değil
  - Yeni çeşitler eksik(elitizm, stokastik evrensel örnekleme)
  - Genellikle teorisyenler tarafından modellenir

Genetik Algoritmalar

Hızlı Genel Bakış

- Holland'ın orijinal GA'sı artık basit genetik algoritma (SGA) olarak biliniyor
- Diğer GA'lar aşağıdakileri faklı kullanır:
  - Temsiller
  - Mutasyonlar
  - Çaprazlamalar
  - Seçme mekanizmaları

Genetik Algoritmalar

Teknik özet tablosu

Temsil	Bit dizileri
Rekombinasyon	Tek noktalı çaprazlama
Mutasyon	Bit-çevirme
Ebeveyn seçimi	Uygunlun orantılı - Rulet çarkı ile uygulanmış
Hayatta kalan seçimi	Kuşak modeli

Genetik Algoritmalar

SGA Üreme Döngüsü

- Çiftleşme havuzu için **ebeveynleri seçin**

  (çiftleşme havuzunun boyutu = popülasyon boyutu)
- Çiftleşme havuzunu karıştırın(shuffle)
- Her ardışık çifte p_c olasılığı ile **çaprazlama uygulayın**, aksi halde($r > p_c$) ebeveynleri kopyalayın
- Her yavruya **mutasyon uygulayın**(her bit için bağımsız olarak p_m olasılığı ile bit çevirme)
- **Tüm popülasyonu** ortaya çıkan yavrularla **değiştirin**

Genetik Algoritmalar

Örnek: Goldberg kitabından[4]

- Basit problem $\{0, 1, \cdots, 31\}$ üzerinde x^2'yi maksimize et
- GA yaklaşımı
  - Temsil: ikili kod, örn: 01101 $\leftrightarrow$ 13
  - Popülasyon boyutu: 4
  - Tek noktalı çaprazlama, bit tabanlı mutasyon
  - Rulet çarkı seçim
  - Rastgele başlatma(random initialisation)
- Elle yapılan bir nesil döngüsü gösteriliyor

Genetik Algoritmalar

x^2 örneği: Seçim

String no.	Initial population	x Value	Fitness $f(x) = x^2$	$Prob_i$	Expected count	Actual count
1	0 1 1 0 1	13	169	0.14	0.58	1
2	1 1 0 0 0	24	576	0.49	1.97	2
3	0 1 0 0 0	8	64	0.06	0.22	0
4	1 0 0 1 1	19	361	0.31	1.23	1
Sum			1170	1.00	4.00	4
Average			293	0.25	1.00	1
Max			576	0.49	1.97	2

Genetik Algoritmalar

x^2 örneği: Çaprazlama

String no.	Mating pool	Crossover point	Offspring after xover	x Value	Fitness $f(x) = x^2$
1	0 1 1 0 | 1	4	0 1 1 0 0	12	144
2	1 1 0 0 | 0	4	1 1 0 0 1	25	625
2	1 1 | 0 0 0	2	1 1 0 1 1	27	729
4	1 0 | 0 1 1	2	1 0 0 0 0	16	256
Sum					1754
Average					439
Max					729

Genetik Algoritmalar

x^2 örneği: Mutasyon

String no.	Offspring after xover	Offspring after mutation	x Value	Fitness $f(x) = x^2$
1	0 1 1 0 0	1 1 1 0 0	26	676
2	1 1 0 0 1	1 1 0 0 1	25	625
2	1 1 0 1 1	1 1 0 1 1	27	729
4	1 0 0 0 0	1 0 1 0 0	18	324
Sum				2354
Average				588.5
Max				729

Genetik Algoritmalar

Basit GA

- Birçok (erken) araştırmaya konu olmuştur
  - Hala sıklıkla yeni GA'lar için referans olarak kullanılmaktadır
- Birçok eksiklik gösterir, ör.
  - Temsil çok kısıtlayıcı
  - Mutasyon ve çaprazlama operatörleri yalnızca bit dizesi ve tamsayı gösterimleri için geçerlidir
  - Yakın uygunluk değerlerine sahip popülasyonları birleştirmek için seçim mekanizması duyarlı
  - Kuşak(Nesil) popülasyon modeli (SGA üreme döngüsünde adım 5), açık hayatta kalanlar seçimi ile geliştirilebilir

Evrim Stratejileri

Hızlı Genel Bakış

- 1960'larda Almanya'da geliştirildi
- İlk isimler: I. Rechenberg, H.-P. Schwefel
- Genellikle şunlara uygulanır:
  - Sayısal optimizasyon
- İlişkilendirilen özellikler:
  - Hızlı
  - Gerçek değerli optimizasyon için iyi optimize edici
  - Nispeten fazla teori
- Özel:
  - (mutasyon) parametrelerinin kendi kendine adaptasyonu standardı

Evrim Stratejileri

ES Teknik özet tablosu

Temsil	Gerçek değerli vektörler
Rekombinasyon	Ayrık veya ara değer
Mutasyon	Gauss sarsımı(perturbation)
Ebeveyn seçimi	Tekörnek rastgele
Hayatta kalan seçimi	(μ, λ) veya $(\mu + \lambda)$

Evrim Stratejileri

Örnek: (1+1) ES

- Görev: $f : \mathbb{R}^n \rightarrow \mathbb{R}$'i minimize et
- Algoritma: "İki üyeli ES" kullanarak
  - $\mathbb{R}^2$'den doğrudan kromozom olarak vektörler
  - Popülasyon boyutu: 1
  - Sadece bir çocuk oluşturan mutasyon
  - Açgözlü(greedy) seçim

Evrim Stratejileri

Giriş örneği: mutasyon mekanizması

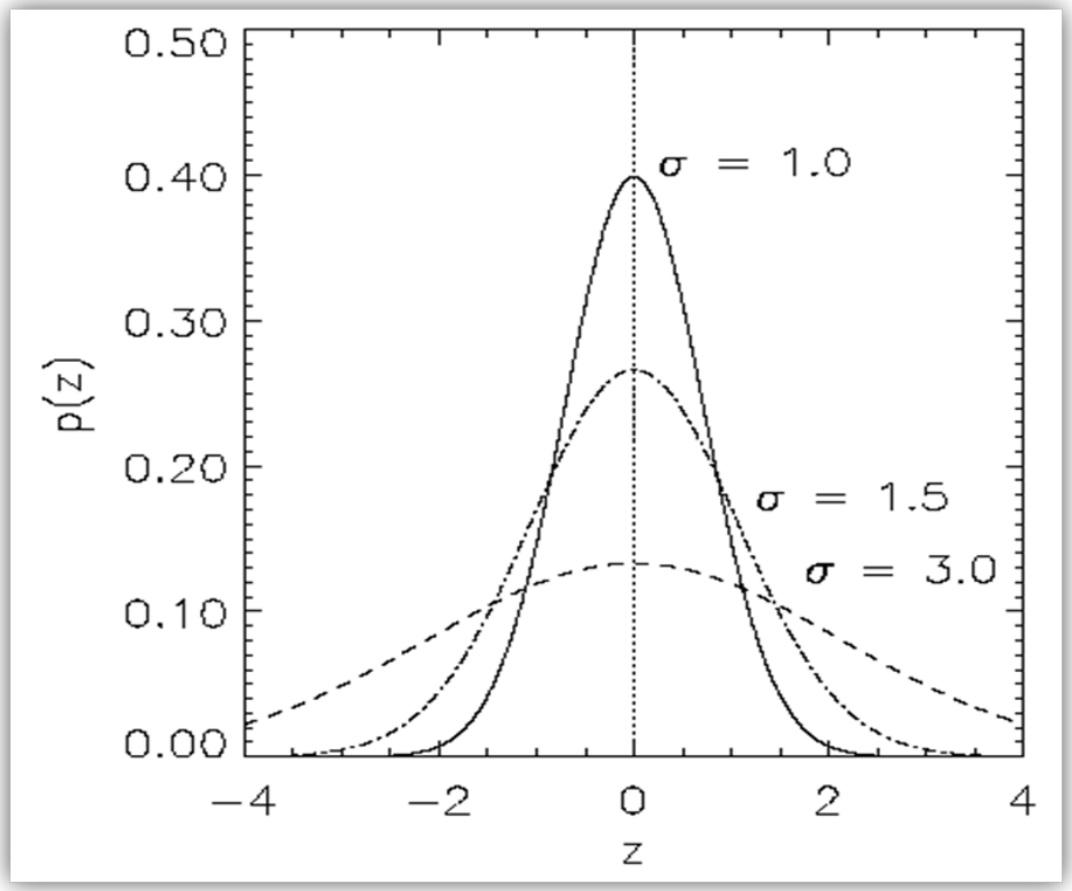
- Normal dağılımdan alınan z değerleri $N(\xi, \sigma)$
  - Ortalama ξ 0'a ayarlanır
  - Varyasyon σ, mutasyon adım boyutu olarak adlandırılır
- σ "1/5 başarı kuralı" ile anında çeşitlenir:
- Bu kural, her k yinelemeden sonra σ değerini sıfırlar
  -

$$\sigma = \begin{cases} \frac{\sigma}{c}, & p_s > \frac{1}{5} \\ \sigma \cdot c, & p_s < \frac{1}{5} \\ \sigma, & p_s = \frac{1}{5} \end{cases}$$

- p_s: başarılı mutasyon yüzdesi, $0.8 \leq c \leq 1$

Evrim Stratejileri

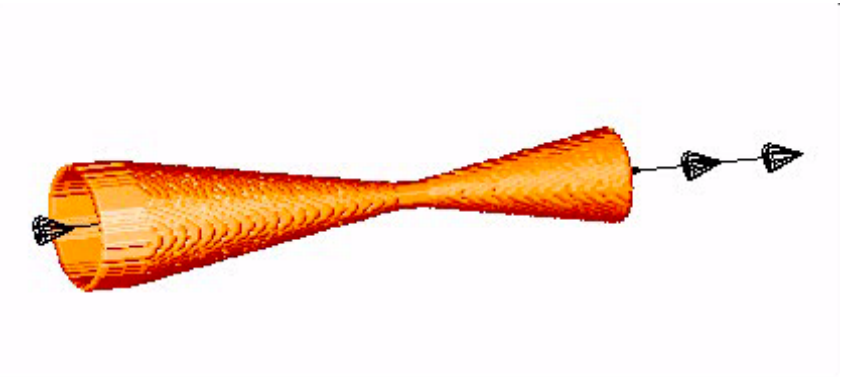
Normal dağılımın gösterimi



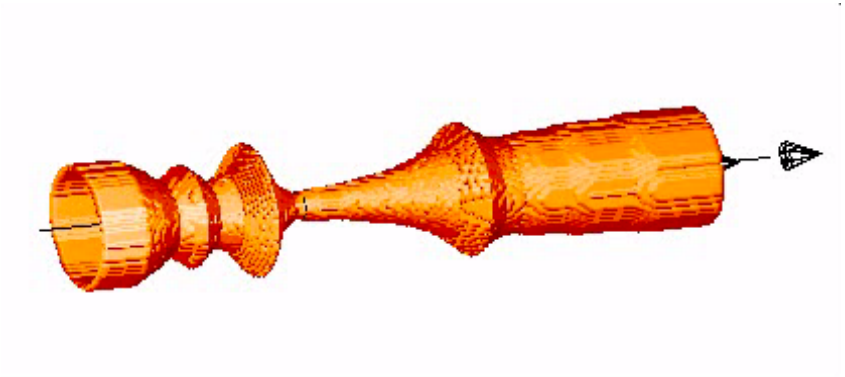
Evrim Stratejileri

Başka bir tarihsel örnek: jet nozul deneyi

- Görev: bir jet nozulun şeklini optimize edin
- Yaklaşım: şekillendirmek için rastgele mutasyonlar + seçim



Şekil: Başlangıç şekli



Şekil: Son şekil

Evrim Stratejileri

Ünlü jet nozul deneyi[7] (film)

Jet nozul deneyi videosu

Evrim Stratejisi

Temsil

- Kromozomlar üç bölümden oluşur:
  - Nesne değişkenleri: $x_1, \cdots, x_n$
  - Strateji parametreleri:
    - Mutasyon adım boyutları $\sigma_1, \cdots, \sigma_n$
    - Dönme açıları $\alpha_1, \cdots, \alpha_n$
- Her bileşen her zaman mevcut değildir
- Tam boyut: $\langle x_1, \cdots, x_n, \sigma_1, \cdots, \sigma_n, \alpha_1, \cdots, \alpha_k \rangle$

  $k = \frac{n(n-1)}{2}$ (i,j çiftlerinin sayısı)

Evrim Stratejileri

Rekombinasyon

- Tek çocuk oluşturur
- Pozisyon/değişken başına hareket
  - Ebeveyn değerlerinin ortalaması veya
  - Ebeveyn değerlerinden birini seçme
- İki veya daha fazla ebeveynden
  - Çocuk yapmak için seçilmiş iki ebeveyni kullanmak
  - Her pozisyon için iki ebeveyn seçmek

Evrim Stratejileri

	İki sabit ebeveyn	Her i için seçilen iki ebeveyn
$z_i = \frac{x_i + y_i}{2}$	Yerel aracı (local intermediary)	Global aracı (global intermediary)
z_i, x_i veya y_i olur, rastgele seçilir	Yerel ayrık (local discrete)	Global ayrık (global discrete)

Evrim Stratejileri

Ebeveyn seçimi

- Bir operatörün ihtiyaç duyduğu durumlarda ebeveynler tek tip rastgele dağılımla seçilir
- Böylece: ES ebeveyn seçimi tarafsızdır - her bireyin seçilme olasılığı aynıdır

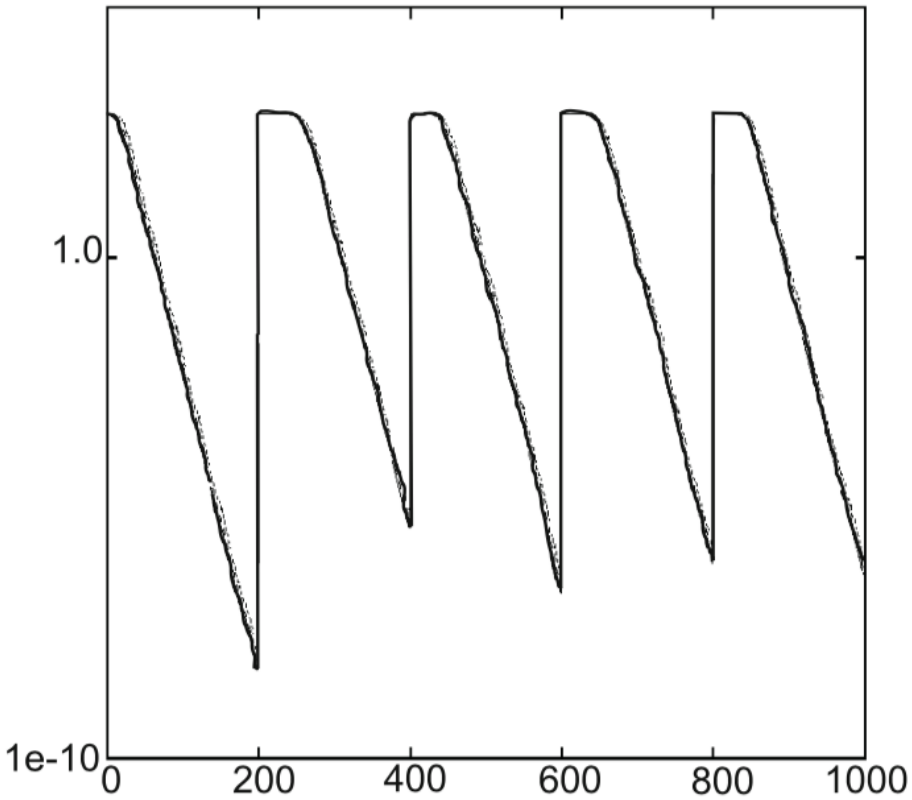
Evrim Stratejileri

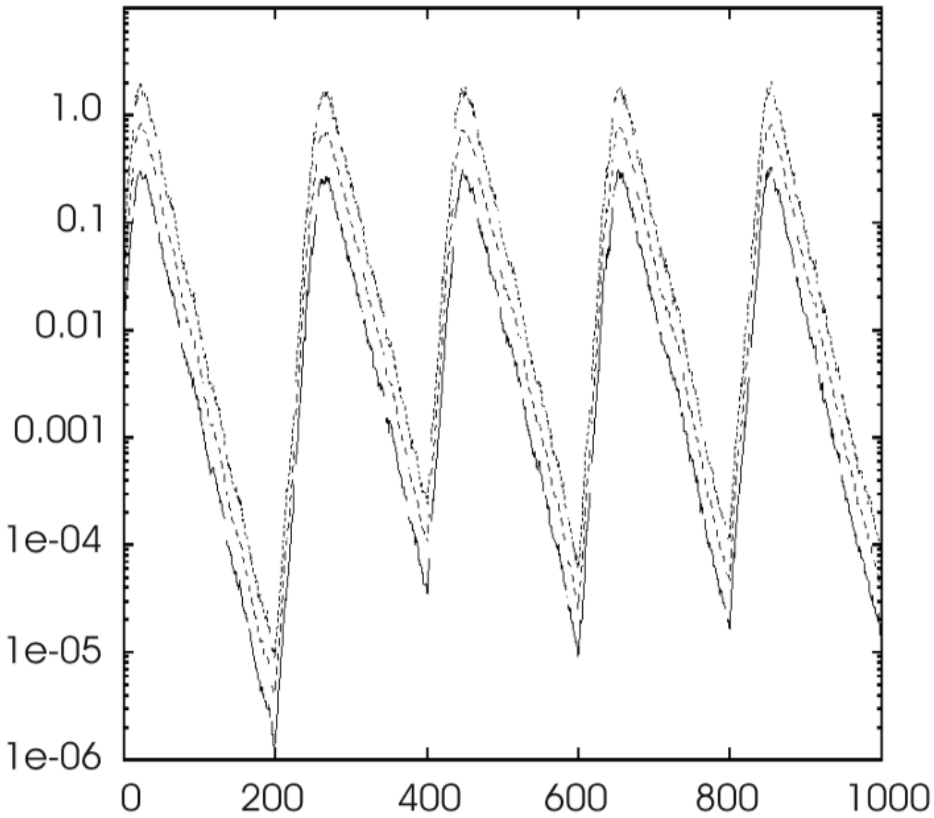
Kendi kendine adaptasyon

- Dinamik olarak değişen bir fitness ortamı göz önüne alındığında (optimum konum her 200 kuşakta bir değişir)
- Kendinden uyarlamalı ES şunları yapabilir:
  - optimum olanı takip etmek ve
  - her değişimden sonra mutasyon adım boyutunu ayarlamak

Evrim Stratejileri

Kendi kendine adaptasyon





Şekil: Uygunluk değerindeki(sol) ve mutasyon adım boyutundaki(sağ) değişim

Evrim Stratejileri

Kendi kendine adaptasyon için ön koşullar

- Farklı stratejiler barındırmak için $\mu > 1$
- Yavru fazlası üretmek için $\lambda > \mu$
- Yanlış uyarlanmış σ'lardan kurtulmak için (μ, λ)-seçimi
- Strateji parametrelerini (aracı)rekombinasyon yoluyla karıştırmak

Evrim Stratejileri

Seçim Basıncı

- **Seçim basıncı**: Daha iyi(uygun) bireylerin ebeveyn olma veya hayatta kalma olasılığının fazla olması
- Devralma(takeover) süresi $\tau*$, seçim basıncını ölçmek için bir ölçüdür
- Popülasyonu en iyi bireyin kopyalarıyla doldurana kadar geçen süre
- Goldberg ve Deb[5] formülleştirmiştir:

$$\tau* = \frac{ln\lambda}{ln(\mu/\lambda)}$$

- Bir genetik algoritmada uygunluk orantılı seçim(FPS) için devralma süresi $\tau* = \lambda ln\lambda$

Evrim Stratejileri

Örnek: Vişne brendi deneyi[6]

- Görev: bir hedef rengi (iyi bilinen bir vişne brendi rengini) veren bir renk karışımı oluşturmak
- Malzemeler: su + kırmızı, sarı, mavi boya
- Temsil: $\langle w, r, y, b \rangle$ kendi kendine adaptasyon yok
- Önceden tanımlanmış bir toplam hacim (30 ml) verecek şekilde ölçeklenen değerler
- Mutasyon düşük/orta/yüksek σ değerleri eşit şansla kullanılır
- Seçim: (1,8) strateji

Evrim Stratejileri

Örnek: Vişne brendi deneyi[6]

- Uygunluk: öğrenciler karışımı etkili bir şekilde yapıyor ve hedef renkle karşılaştırıyor
- Sonlandırma kriteri: karışık renkten memnun öğrenci
- Çözüm çoğunlukla 20 nesilde bulunur
- Doğruluk çok iyi

Evrim Stratejileri

Örnek uygulama: Ackley fonksiyonu[1]

- Ackley fonksiyonu (n=30 olarak kullanılmış)

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

- Temsil:
  - $-30 < x_i < 30$
  - 30 adet adım boyutu
- (30, 200)-seçim
- Sonlandırma: 200000 uygunluk değerlendirmesinden sonra
- Sonuçlar: ortalama en iyi çözüm $7.48 \cdot 10^{-8}$(çok iyi)

Evrimsel Programlama

Hızlı Genel Bakış

- 1960'larda ABD'de geliştirildi
- İlk isimler: D. Fogel
- Genellikle şunlara uygulanır:
  - Geleneksel EP: sonlu durum makineleri ile tahmin
  - Çağdaş EP: (sayısal) optimizasyon
- İlişkilendirilen özellikler:
  - Çok açık çerçeve: herhangi bir temsil veya mutasyon işlemi kullanılabilir
  - ES ile melezlenmiş (çağdaş EP)
  - Sonuç olarak: "standart" EP'nin ne olduğunu söylemek zor
- Özel:
  - Rekombinasyon yok
  - Parametrelerin kendi kendine uyarlanması standardı (çağdaş EP)

Evrimsel Programlama

EP Teknik özet tablosu

Temsil	Gerçek değerli vektörler
Rekombinasyon	Yok
Mutasyon	Gauss sarsımı(perturbation)
Ebeveyn seçimi	Deterministik(her ebeveyn bir yavru)
Hayatta kalan seçimi	Olasılıksal $(\mu + \mu)$

Evrimsel Programlama

Tarihsel EP Perspektifi

- EP zeka elde etmeyi hedefler
- Zeka, uyarlanabilir davranış olarak görüldü
- Çevrenin tahmini, uyarlanabilir davranış için bir ön koşul olarak kabul edildi
- Dolayısıyla, tahmin etme yeteneği zekanın anahtarıdır

Evrimsel Programlama

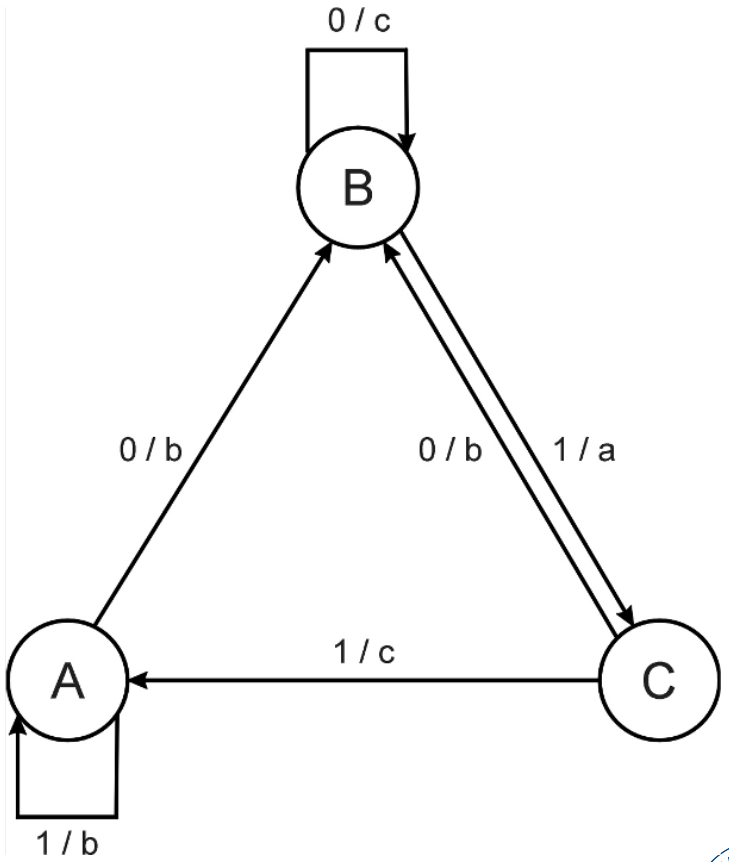
Sonlu durum makineleri ile tahmin

- Sonlu durum makineleri:
  - Durumlar S
  - Girişler I
  - Çıkışlar O
  - Geçiş fonksiyonu δ: $S \times I \rightarrow S \times O$
  - Giriş akışını çıkış akışına dönüştürür
- Tahminler için kullanılabilir, ör. sıradaki bir sonraki giriş sembolünü tahmin etmek için

Evrimsel Programlama

Sonlu Durum Makinesi(FSM) Örneği

- FSM'yi şu şekilde düşünelim:
  - S={A,B,C}
  - I={0,1}
  - O={a,b,c}
  - δ diyagramda verilmiş

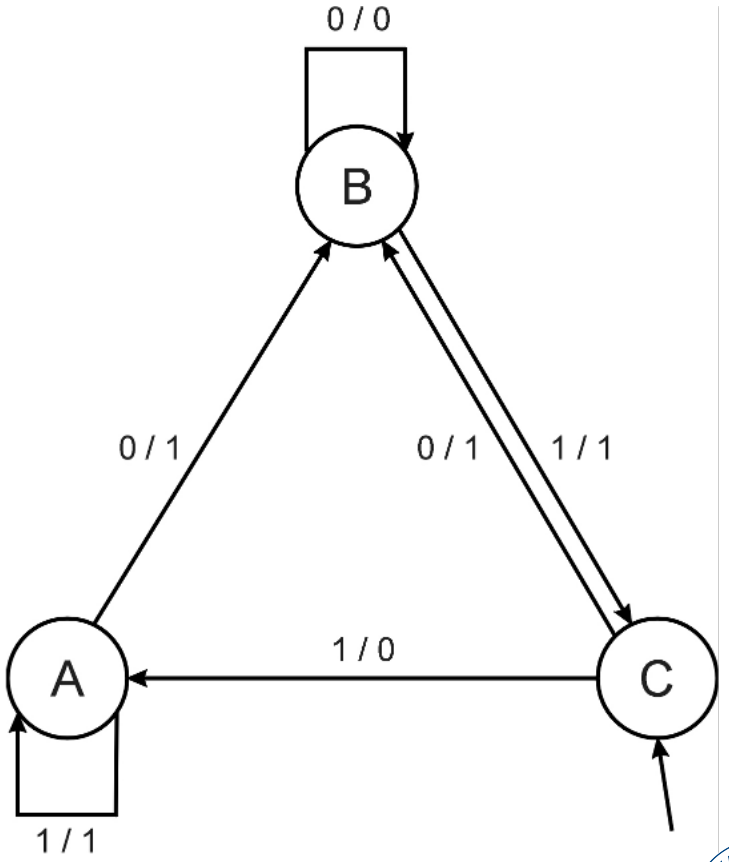


Şekil: Örnek FSM

Evrimsel Programlama

Öngörücü/Tahminci olarak FSM

- Aşağıdaki FSM'yi düşünün
- Görev: sonraki girişi tahmin et
- Kalite: $giriş_{i+1} = çıkış_i$ yüzdesi
- Giriş değeri **011101**
- **110111** çıktısına yol açar
- Kalite: $\frac{3}{5}$



Şekil: Sonlu durum makinesi

Evrimsel Programlama

Asal sayıları tahmin etmek için FSM'leri geliştirmek

- n asalsa $P(n) = 1$, aksi halde 0
- $I = N = \{1, 2, 3, \cdots, n, \cdots\}$
- $O = \{0, 1\}$
- Doğru tahmin $çikiş_i = P(giriş_{i+1})$
- Uygunluk(fitness) fonksiyonu:
  - Bir sonraki girişin doğru tahmini için 1 puan
  - Yanlış tahmin için 0 puan
  - "Çok fazla" durum için ceza

Evrimsel Programlama

Asal sayıları tahmin etmek için FSM'leri geliştirmek

- Ebeveyn seçimi: Her FSM bir kez mutasyona uğratılır
- Mutasyon operatörleri (biri rastgele seçilir):
  - Çıkış sembolünü değiştirme
  - Durum geçişini değiştirin (yani kenarı yeniden yönlendirme)
  - Bir durum ekleme
  - Bir durum silme
  - Başlangıç durumunun değiştirilmesi
- Hayatta kalan seçimi: (μ, μ)
- Sonuçlar: aşırı uydurma, 202 girdiden sonra en iyi FSM bir duruma sahipti ve her iki çıkış da 0 idi, yani her zaman "asal değil" öngördü
- Ana nokta: mükemmel doğruluk değil, simülasyonlu evrimsel sürecin akıllı görevler için iyi çözümler yaratabileceğinin kanıtı

Evrimsel Programlama

Modern EP

- Genel olarak önceden tanımlanmış temsil yok
- Böylece: önceden tanımlanmış mutasyon yok (temsil ile eşleşmelidir)
- Genellikle mutasyon parametrelerinin kendi kendine adaptasyonunu uygular

Evrimsel Programlama

Temsil

- Sürekli parametre optimizasyonu için
- Kromozomlar iki kısımdan oluşur:
  - Nesne değişkenleri: $x_1, \cdots, x_n$
  - Mutasyon adım boyutları: $\sigma_1, \cdots, \sigma_n$
- Tam boyut: $\langle x_1, \cdots, x_n, \sigma_1, \cdots, \sigma_n \rangle$

Evrimsel Programlama

Mutasyon

- Kromozomlar: $\langle x_1, \cdots, x_n, \sigma_1, \cdots, \sigma_n \rangle$
- $\sigma'_i = \sigma_i \cdot (1 + \alpha \cdot N(0,1))$
- $x'_i = x_i + \sigma'_i \cdot N_i(0,1)$
- $\alpha \approx 0.2$
- Sınır kuralı: $\sigma' < \varepsilon_0 \Rightarrow \sigma' = \varepsilon_0$
- Önerilen ve denenen diğer değişiklikler:
  - Standart sapma yerine varyans kullanmak
  - σ değerini sonradan mutasyona uğratmak
  - Diğer dağılımlar, örneğin Gaussian yerine Cauchy

Evrimsel Programlama

Rekombinasyon

- Yok
- Gerekçe: Arama alanındaki bir nokta, bir birey için değil, bir tür anlamına gelir ve türler arasında çaprazlama olamaz
- Çok sayıda tarihsel tartışma "mutasyon vs çaprazlama"

Evrimsel Programlama

Ebeveyn seçimi

- Her birey mutasyonla bir çocuk yaratır
- Böylece:
  - Deterministik
  - Fitness tarafından önyargılı değil

Evrimsel Programlama

Dama oyuncuları geliştirmek[3, 2]

- Hareketlerin gelecekteki değerlerini değerlendirmek için yapay sinir ağları geliştirildi
- Sinir ağları 5046 ağırlıklı sabit bir yapıya sahiptir, bu değerler geliştirildi ("krallar" için $+1$ ağırlık)
- Temsil:
  - nesne değişkenleri için 5046 gerçek sayı vektörü (ağırlıklar)
  - σ değerleri için 5046 gerçek sayının vektörü
- Mutasyon:
  - Gauss, lognormal şema, önce σ
  - Ayrıca kralların ağırlığı için özel mekanizma
- Popülasyon büyüklüğü: 15

Evrimsel Programlama

Dama oyuncuları geliştirmek[3, 2]

- Turnuva boyutu q = 5 (round-robin)
- Programlar (sinir ağları içeride) diğer programlara karşı oynar, insan eğitmeni veya fiziksel bağlantılı zeka yoktur
- 840 nesilden sonra (6 ay!) En iyi strateji İnternet aracılığıyla insanlara karşı test edildi
- Program, derecelendirilen tüm oyuncuların % 99,61'inden daha iyi performans gösteren "uzman sınıfı" sıralaması kazandı

Genetik Programlama

Hızlı Genel Bakış

- 1990'larda ABD'de geliştirildi
- İlk isimler: J. Koza
- Genellikle şunlara uygulanır:
  - Makine öğrenmesi görevleri(tahmin, sınıflandırma)
- İlişkilendirilen özellikler:
  - Yapay sinir ağları ve benzerleriyle rekabet eder
  - Büyük popülasyonlara ihtiyaç duyarlar(binlerce)
  - Yavaş
- Özel:
  - Doğrusal olmayan kromozomlar
  - Mutasyon mümkün ama gerekli değil

Genetik Programlama

Teknik özet tablosu

Temsil	Ağaç yapıları
Rekombinasyon	Alt ağaçların değişimi
Mutasyon	Ağaçta rastgele değişimler
Ebeveyn seçimi	Uygunlun orantılı
Hayatta kalan seçimi	Kuşak modeli yer değişim

Genetik Programlama

Örnek: Kredi puanlaması

- Banka, yapılan iyi ve kötü kredi başvurularını ayırt etmek istiyor
- Geçmiş verilerle eşleşen model gerekli

ID	Çocuk sayısı	Maaş	Medeni durum	Uygun?
ID-1	2	45000	Evli	0
ID-2	0	30000	Bekar	1
ID-3	1	40000	Boşanmış	1
...				

Genetik Programlama

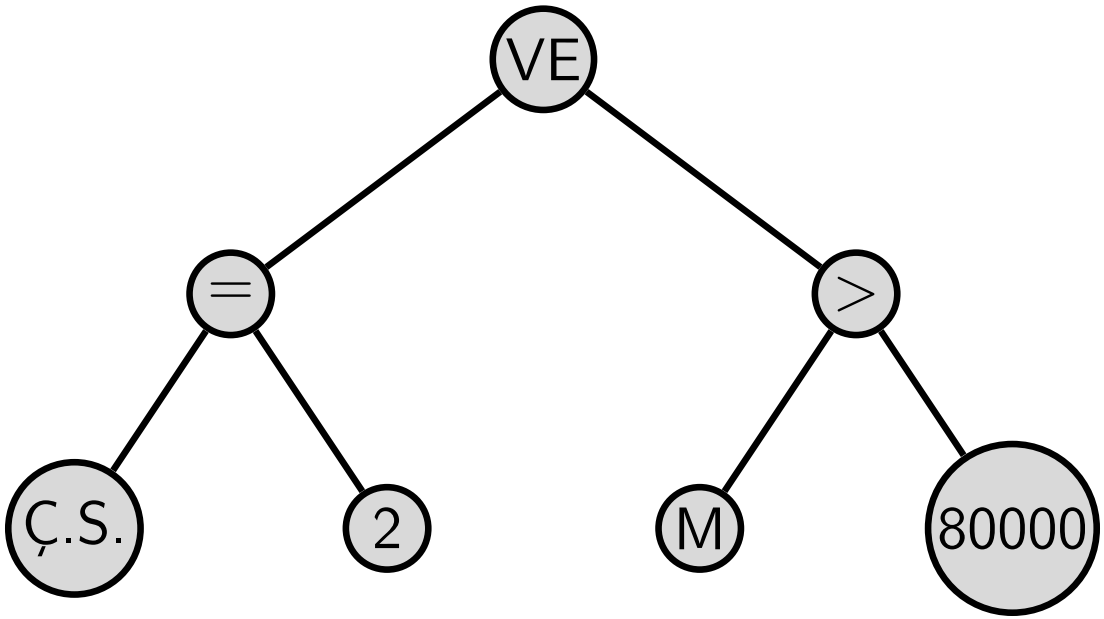
Örnek: Kredi Puanlaması

- Olası bir model:
- EĞER **(Ç.S.=2) VE (M>80000)** İSE **iyi** AKSİ HALDE **kötü**
- Genel olarak:
- EĞER **formül** İSE **iyi** AKSİ HALDE **kötü**
- Tek bilinmeyen doğru formüldür, dolayısıyla
- Arama uzayı(fenotipler) formüllerin kümesidir
- Bir formülün uygunluğu: temsil ettiği modelin iyi sınıflandırılmış vakalarının yüzdesi

Genetik Programlama

Örnek: Kredi Puanlaması

- EĞER **(Ç.S.=2) VE (M>80000)** İSE **iyi** AKSİ HALDE **kötü** ağacı aşağıdaki gibi temsil edilebilir



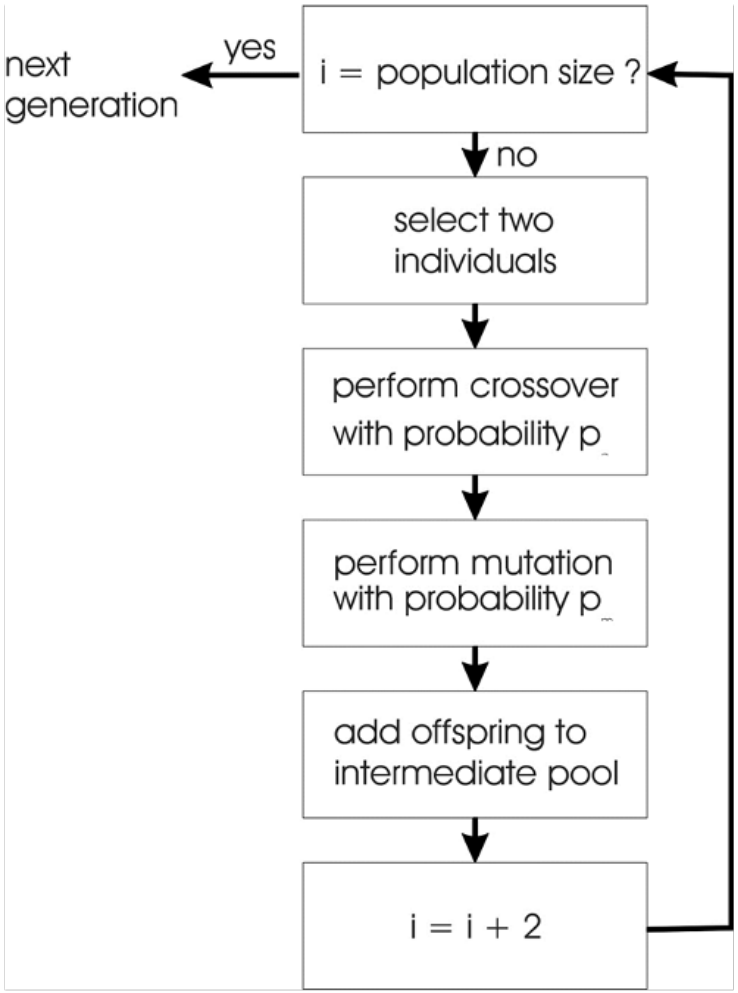
Genetik Programlama

Çocuk Oluşturma Şeması

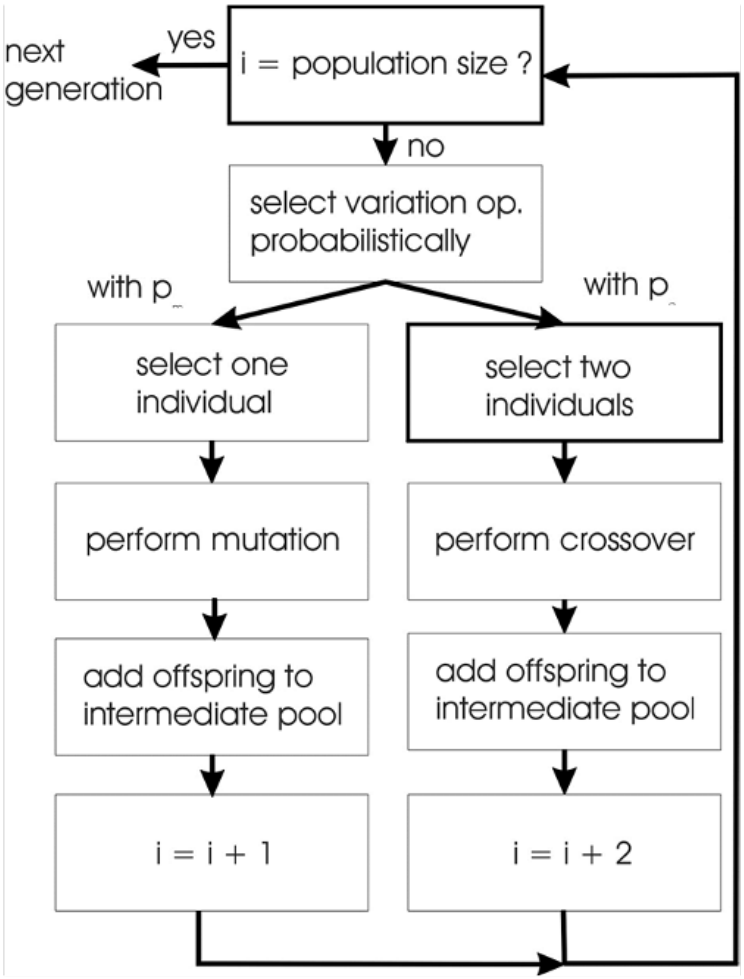
- Karşılaştırma
  - Çaprazlama **ve** mutasyonu ardışık olarak kullanan GA şeması(olasılıksal olarak)
  - Çaprazlama **veya** mutasyonu kullanan GP şeması(olasılıksal olarak seçilir)

Genetik Programlama

GA vs GP



GA flowchart



GP flowchart

Genetik Programlama

Seçim

- Ebeveyn seçimi genellikle uygunluk orantılı
- Büyük popülasyonlarda aşırı seçim(over-selection)
  - Popülasyonu uygunluğa göre sıralayın ve iki gruba ayırın
  - grup 1: popülasyonun en iyi %x'i, grup 2: kalan %(100-x)'i
  - Seçim işleminin %80'i grup 1'den, %20'si grup 2'den yapılır
  - 1000, 2000, 4000, 8000 popülasyon boyutları için $x = \%32, \%16, \%8, \%4$
  - motivasyon: verimliliği artırmak için %'ler pratik kuraldan gelir
- Hayatta kalan seçimi
  - Tipik: kuşak(generational) şeması (dolayısıyla hiçbiri)
  - Son zamanlarda kararlı durum(steady state) elitizmi ile popüler hale gelmiştir

Genetik Programlama

Başlatma(Initialisation)

- Ağaçların maksimum derinliği D_{max} ayarlanır
- Tam yöntem(full method): Her dalın derinliği $= D_{max}$
  - $d < D_{max}$ derinliğindeki düğümler fonksiyon kümesi F'den rastgele seçilir
  - $d = D_{max}$ derinliğindeki düğümler terminal kümesi T'den rastgele seçilir
- Büyüme yöntemi(grow method): Her dalın derinliği $\leq D_{max}$
  - $d < D_{max}$ derinliğindeki düğümler $F \cup T$'den rastgele seçilir
  - $d = D_{max}$ derinliğindeki düğümler T'den rastgele seçilir
- Genel GP başlatma: *ramped half-and-half*: popülasyonun her bir yarısı büyüme ve tam yöntemleri ile oluşturulur

Genetik Programlama

Şişirme(bloat)

- **Bloat**: "en şişmanın hayatta kalması(surival of the fattest)": popülasyondaki ağaç boyutları zamanla büyür
- Sebepleri ile ilgili araştırma ve tartışma devam etmektedir
- Önlem alınması gerekir
  - "Çok büyük" çocuklar doğuracak varyasyon operatörlerinin yasaklanması
  - Karamsarlık baskısı: aşırı büyük olmanın cezası

Genetik Programlama

Örnek: Sembolik regresyon

- R^2'de verilen bazı noktalar $(x_1, y_1), \cdots, (x_n, y_n)$
- $f(x)$ fonksiyonunu bulun, öyle ki $\forall i = 1, \cdots, n : f(x_i) = y_i$
- Olası GP çözümü
  - Temsil: $F = \{+, -, /, sin, cos\}$, $T = R \cup \{x\}$
  - Uygunluk hata değeridir $err(f) = \sum_{i=1}^{n} (f(x_i) - y_i)^2$
  - Tüm operatörler standart
  - Popülsayon boyutu=1000, *ramped half-and-half* başlatma
  - Sonlandırma: n "isabet" ya da 50000 uygunluk değerlendirmesi(isabet: $|f(x_i) - y_i| < 0.0001$)

Diferansiyel Evrim

Hızlı Genel Bakış

- 1995'de ABD'de geliştirildi
- İlk isimler: Storn, Price
- Genellikle şunlara uygulanır:
  - Doğrusal olmayan ve türevlenemeyen sürekli uzay fonksiyonları
- İlişkilendirilen özellikler:
  - Popülasyonlar listedir
  - Yeni bir birey oluşturmak için 4 ebeveyn gerekir
  - Temel vektörün değiştirilmesi şekline göre farklı çeşitleri mevcuttur
- Özel
  - Diferansiyel mutasyon

Diferansiyel Evrim

Teknik özet tablosu

Temsil	Gerçek değerli vektörler
Rekombinasyon	Tekörnek çaprazlama
Mutasyon	Diferansiyel mutasyon
Ebeveyn seçimi	3 gerekli vektörün tekörnek rastgele seçimi
Hayatta kalan seçimi	Deterministik elitist değişim(ebeveyn vs çocuk)

Diferansiyel Evrim

Diferansiyel Mutasyon

- $\mathbb{R}^n$'de aday çözüm vektörlerinin bir popülasyonu verildiğinde
- Bir sarsım(perturbation) vektörü eklenerek yeni bir mutant oluşturulur

  $\overline{v}' = \overline{v} + \overline{p}$

  $\overline{p} = F \cdot (\overline{y} - \overline{x})$

  y ve x rastgele seçilen popülasyon üyeleri, ölçekleme faktörü $F > 0$ popülasyonun geliştiği hızı kontrol eden bir gerçel sayıdır

Diferansiyel Evrim

Tekörnek Çaprazlama

- DE ufak bir değişiklikle tekörnek çaprazlama kullanır
  - Rastgele seçilen bir pozisyonda, çocuk alel, rastgele bir karar vermeden birinci ebeveynden alınır (ikinci ebeveynin kopyasının oluşturulması imkansız hale gelir)
- Devralınan mutant alellerin sayısı bir binom dağılımını izler

Diferansiyel Evrim

Evrim Döngüsü

- Popülasyon bir listedir, uygunluk değeri ile ilişkili değildir $P = \langle \overline{x}_1, \cdots, \overline{x}_\mu \rangle$
- Bir mutant vektör popülasyonu oluşturma $M = \langle \overline{v}_1, \cdots, \overline{v}_\mu \rangle$
  - Her yeni mutant için, P popülasyonundan rastgele 3 vektör seçilir(temel vektör u ve diğer iki adet y ve z)
- Deneme vektör popülasyonu oluşturulur $T = \langle \overline{u}_1, \cdots, \overline{u}_\mu \rangle$
  - $\overline{x}_i$ ile $\overline{v}_i$ arasında tekörnek çaprazlama
- Her bir x_i ve u_i çiftine deterministik seçim uygulanır
  - Bir sonraki nesildeki i. eleman, en yüksek uygunluk değerine sahip olandır

Diferansiyel Evrim

Farklı Çeşitler

- temel vektörü değiştirme şekline göre farklı çeşitleri vardır
- Çeşitler **DE/a/b/c** şeklinde tanımlanır
  - a temel vektör(rastgele(rand) veya en iyi(best))
  - b sarsım vektörünü tanımlamak için gerekli vektör sayısı
    örneğin 4 rastgele seçilen
    $\overline{p} = F \cdot (\overline{y} - \overline{z} + \overline{y}' - \overline{z}')$
  - c çaprazlama şemasını belirtir("bin" tekörnek çaprazlamayı temsil eder)

Diferansiyel Evrim

Darth Vader Resmini Geliştirmek

Darth Vader Resmini Geliştirmek

Diferansiyel Evrim

Darth Vader Resmini Geliştirmek

- Bireyler [0,1] aralığında 18000 değerle temsil edilir
- Popülasyon boyutu: 400, $F = 0.1$, Çaprazlama oranı 0.1
- Uygunluk = beklenen değerle mevcut değer arasındaki karesel fark toplamı

Parçacık Sürüsü Optimizasyonu

Hızlı Genel Bakış

- 1995'te geliştirildi
- İlk isimler: Kennedy, Eberhart
- Genellikle şunlara uygulanır:
  - Doğrusal olmayan fonksiyonların optimizasyonu
- İlişkilendirilen özellikler:
  - Çaprazlama yok
  - Her aday çözüm kendi sarsım(perturbation) vektörünü taşır
- Özel
  - kuş sürüsü / balık yetiştiriciliğinin sosyal davranışından esinlenmiştir
  - Konum ve hıza sahip parçacıklar, genotip ve mutasyona sahip bireylerdir

Parçacık Sürüsü Optimizasyonu

Teknik özet tablosu

Temsil	Gerçek değerli vektörler
Rekombinasyon	Yok
Mutasyon	Hız vektörü ekleme
Ebeveyn seçimi	Deterministik(her ebeveyn mutasyon yoluyla bir yavru üretir)
Hayatta kalan seçimi	Kuşak(yavrular ebeveynlerin yerine geçer)

Parçacık Sürüsü Optimizasyonu

Temsil

- Her popülasyon üyesi, birinci vektörün aday çözüm olduğu ve ikincisinin $\mathbb{R}^n$'de bir pertürbasyon vektörü olduğu bir çift olarak kabul edilebilir $\langle \overline{x}, \overline{p} \rangle$
- Pertürbasyon vektörü, çözüm vektörünün yeni bir tane oluşturmak için nasıl değiştirileceğini belirler:

  $\overline{x}' = \overline{x} + \overline{p}'$

  $\overline{p}'$, $\overline{p}$ ve birtakım ek bilgilerle oluşturulur

Parçacık Sürüsü Optimizasyonu

Temsil

- Bir üye, uzayda konumu ve hızı olan bir nokta olarak kabul edilir
- Pertürbasyon vektörü bir hız vektörüdür ve yeni bir hız vektörü üç bileşenin ağırlıklı toplamı olarak tanımlanır:
  - Mevcut hız vektörü
  - Üyenin mevcut konumu ile şimdiye kadarki en iyi konumu arasındaki vektör farkı
  - Mevcut konum ile şimdiye kadarki popülasyondaki en iyi konum arasındaki vektör farkı

  $\overline{v}' = w \cdot \overline{v} + \phi_1 U_1 \cdot (\overline{y} - \overline{x}) + \phi_2 U_2 \cdot (\overline{z} - \overline{x})$

  w ve ϕ_i ağırlıklar, U_1 ve U_2 rasgeleleştirme matrisleri
- Kişisel en iyi ve küresel en iyi hatırlanması gerektiğinden, popülasyonlar listedir

Parçacık Sürüsü Optimizasyonu

Daha İyi Temsil

- Bu nedenle $\langle \overline{x}, \overline{p} \rangle$ daha iyi bir şekilde yeniden yazılabilir
- Her üçlü mutant üçlü ile değiştirilir

  $\left\langle \overline{x}_i, \overline{v}_i, \overline{b}_i \right\rangle \rightarrow \left\langle \overline{x}'_i, \overline{v}'_i, \overline{b}'_i \right\rangle$

  değiştirme aşağıdaki formüllerle yapılır

  $\overline{x}'_i = \overline{x} + \overline{v}'_i$

  $\overline{v}'_i = w \cdot \overline{v}_i + \phi_1 U_1 \cdot (\overline{b}_i - \overline{x}_i) + \phi_2 U_2 \cdot (\overline{c} - \overline{x}_i)$

  $\overline{c}$ popülasyondaki global en iyiyi temsil eder

$$\overline{b}'_i = \begin{cases} \overline{x}'_i & eğer f(\overline{x}'_i) < f(\overline{b}_i) \\ \overline{b}_i & aksi\ halde \end{cases}$$

Parçacık Sürüsü Optimizasyonu

Örnek: Hareket eden hedef

Örnek: Hareket eden hedef

Parçacık Sürüsü Optimizasyonu

Örnek: Hareket eden hedef

- Optimum tasarım alanı boyunca rastgele hareket eder
- Parçacıklar optimumun konumunu bilmezler, ancak hangi parçacığın en yakın olduğunu ve ona hangi parçacığın çekildiğini bilirler
- Ön koşul: düşük w değeri, kişisel en iyi ($\overline{b}_i$) sıfırdır

Dağıtım Algoritmalarının Tahmini

- Sunumdaki video ile ilgili bilgi yok

Sınıflandırıcı Sistemlerin Öğrenilmesi

Hızlı Genel Bakış

- İlk 1976'da tanımlandı
- İlk isimler: Holland
- Genellikle şunlara uygulanır:
  - Kural kümeleriyle çalışan makine öğrenimi görevleri
  - Mevcut durum ortamına en iyi yanıtı veren
- İlişkilendirilen özellikler:
  - Öğrenme algoritması ve sınıflandırıcı sistemin kombinasyonudur
  - Genetik algoritmaları kullanır
  - Michigan tarzı (kural bireydir) vs Pittsburgh tarzı (kural kümesi bireydir)

Sınıflandırıcı Sistemlerin Öğrenilmesi

Giriş Örneği: Çoklayıcı

- k-bit çoklayıcı, k uzunluğunda bit dizisidir
  - $k = l + 2^l$
  - l adres bölümüdür
  - 2^l veri bölümüdür
  - Örneğin $l = 2$, $k = 6$, 101011 doğru dize
  - Adres bölümü tarafından belirtilen veri bitinin değerini döndür (veri bölümü 1011'in 10 konumunu al, bu da 0'dır)
  - Ödüller doğru cevaba atanabilir

Sınıflandırıcı Sistemlerin Öğrenilmesi

Örnek İterasyon: Minimal Classifier System(MCS)[8]

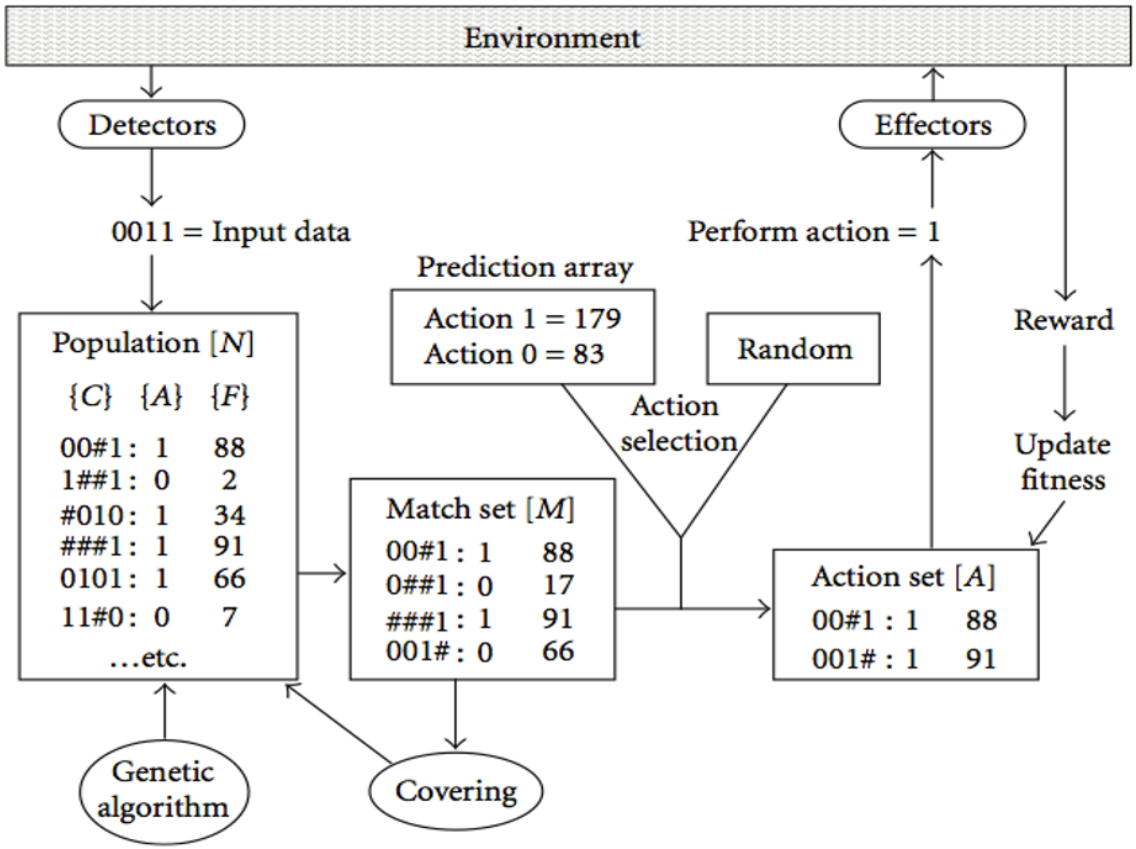


FIGURE 3: MCS algorithm—an example iteration.

Sınıflandırıcı Sistemlerin Öğrenilmesi

Teknik özet tablosu, Michigan tarzı

Temsil	{Koşul: eylem: kazanç, doğruluk} koşulu dizisi {0,1, #} alfabesini kullanır
Rekombinasyon	Koşul/eylemlerde tek noktalı çaprazlama
Mutasyon	Eylem/koşullarda uygun şekilde ikili sıfırlama
Ebeveyn seçimi	Çevresel nişler içinde uygunluk orantılı
Hayatta kalan seçimi	Stokastik, aynı çevresel nişi kapsayan kural sayısıyla ters orantılıdır
Fitness	Alınan her ödül, tahmin edilen kazanç ve doğruluk değerlerini takviyeli öğrenmeye(reinforcement learning) göre günceller

Sınıflandırıcı Sistemlerin Öğrenilmesi

Temsil

- Kural tabanının her kuralı bir demettir(tuple) {koşul: eylem: kazanç}
- Eşleşme kümesi(Match set): koşulu ortamdaki mevcut girdilerle eşleşen kuralların alt kümesi
- Eylem kümesi(Action set): seçilen eylemi savunan eşleşme setinin alt kümesi
- Daha sonra, kural grubu, sistemin tahmin edilen getirinin alınan ödülle ne kadar iyi eşleştiğine ilişkin deneyimini yansıtan bir doğruluk değeri içerir

Sınıflandırıcı Sistemlerin Öğrenilmesi

Michigan ve Pittsburgh Tarzları[8]

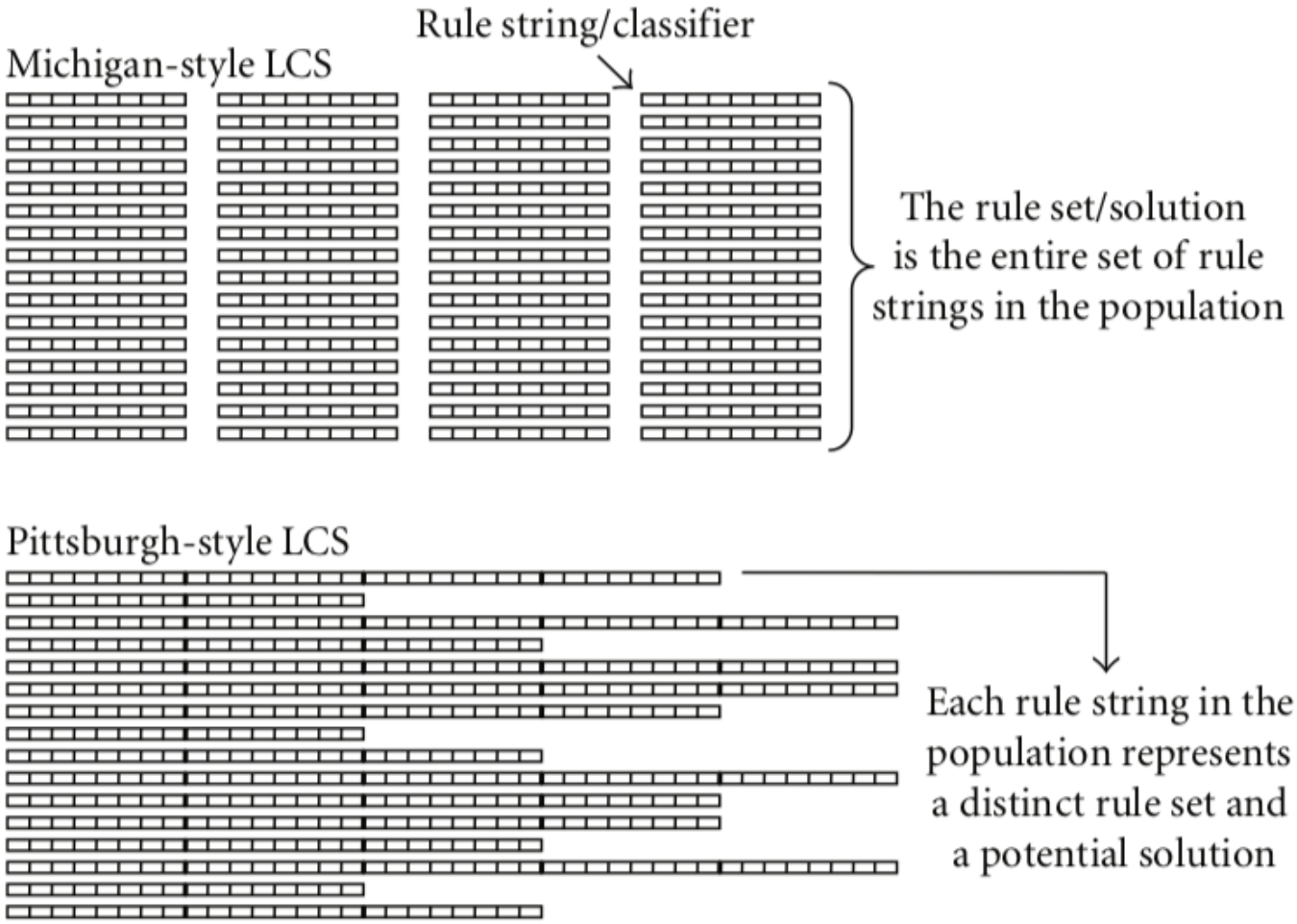


FIGURE 4: Michigan versus Pitt-style systems.

Kaynaklar I


Thomas Bäck and Hans-Paul Schwefel.
An overview of evolutionary algorithms for parameter optimization.
Evolutionary computation, 1(1):1–23, 1993.


Kumar Chellapilla and David B. Fogel.
Evolving an expert checkers playing program without using human expertise.
IEEE Transactions on Evolutionary Computation, 5(4):422–428, 2001.


David B Fogel.
Evolving a checkers player without relying on human experience.
intelligence, 11(2):20–27, 2000.


David E Goldberg.
Genetic algorithms in search.
Optimization, and Machine Learning, 1989.

Kaynaklar II

David E. Goldberg and Kalyanmoy Deb.
A comparative analysis of selection schemes used in genetic algorithms.
In *Foundations of Genetic Algorithms*, pages 69–93. Morgan Kaufmann, 1991.

Michael Herdy.
Evolution strategies with subjective selection.
In *International Conference on Parallel Problem Solving from Nature*, pages 22–31. Springer, 1996.

Jürgen Klockgether and Hans-Paul Schwefel.
Two–phase nozzle and hollow core jet experiments.
In D. G. Elliott, editor, *Proc. Eleventh Symp. Engineering Aspects of Magnetohydrodynamics*, pages 141–148, Pasadena CA, 1970. California Institute of Technology.

Kaynaklar III


Ryan J Urbanowicz and Jason H Moore.
Learning classifier systems: a complete introduction, review, and roadmap.
Journal of Artificial Evolution and Applications, 2009, 2009.