Zeki Optimizasyon Teknikleri
Intelligent Optimization Techniques

Hazırlayan: M. Ali Akcayol
Gazi Üniversitesi
Bilgisayar Mühendisliği Bölümü

Bu dersin sunumları, “Singiresu S. Rao, Engineering Optimization: Theory and Practice, Wiley, 2009.” kitabı kullanılarak hazırlanmıştır.

İçerik

- Evrimsel algoritmalar
- Diferansiyel gelişim algoritması
- Evrimsel programlama

Evrimsel algoritmalar

- Evrimsel algoritmalar, **genellikle popülasyon tabanlıdır.**
- Popülasyondaki **her birey** çözümün tamamını temsil eder ve **çözüm olmaya adaydır.**
- Evrimsel algoritmalar, **deneme** ve **yanılma** ile **faydalı çözümlerin güçlendirilmesi** ve **faydasız olanların elimine edilmesini** iteratif olarak **gerçekleştirir.**
- Aday çözümler amaç fonksiyonuna **uygunluk düzeyi arttıkça sonraki jenerasyonlara aktarılma olasılığını artırır.**
- Evrimsel algoritmalarda, popülasyondaki bireyler arasında **özellik aktarımı yapılabilmektedir.**
- Bireylerde **farklı çözüm adayları** oluşturmak **için rastgele değişiklikler yapılabilmektedir.**
- **Popülasyondaki en iyi birey problemin çözümü olarak alınır.**

İçerik

Diferansiyel gelişim algoritması

- Diferansiyel gelişim algoritması, **aday çözümlerden oluşan bir popülasyon oluşturur.**
- **İteratif olarak** popülasyon üzerinde **birleştirme**, **değerlendirme** ve **seçme işlemlerini yapar.**
- **Birleştirme** yaklaşımı ile **popülasyondan rastgele seçilen iki birey arasındaki farkın** ağırlığına göre **üçüncü bir bireyle birleştirme yapar.**
- Bu yaklaşım ile popülasyondaki **bireylerin çeşitliliğinin artırılması amaçlanmıştır.**
- Diferansiyel gelişim algoritması, **nonlineer** ve **türevlenemez sürekli fonksiyonların optimizasyonu için tasarlanmıştır.**

Diferansiyel gelişim algoritması

- Ağırlık faktörü $F \in [0, 2]$ **arasındadır** (fark değişimi katsayısı).
- **Ağırlık faktörü** genellikle **0,8 seçilir.**
- **Crossover** ağırlığı $CR \in [0, 1]$ **arasında rastgele seçilir.**
- **Crossover** ağırlığı genellikle **0,9 seçilir.**
- **Başlangıç popülasyonu** tamamen **rastgele oluşturulur.**
- Algoritma kabul edilen konfigürasyonu tanımlayan özel bir **terminolojiye sahiptir (DE/x/y/z).**
- **x çeşitlendirilecek çözümü** (rastgele veya en iyi alınır), **y ise x üzerinde kullanılacak fark vektörlerini**, **z birleştirme operatörünü** (binomial veya exponential) gösterir.
- Popüler konfigürasyonlar:
  - **DE/rand/1/***
  - **DE/best/2/***

Diferansiyel gelişim algoritması

- DE/rand/1/bin için pseudocode.

Pseudocode for Differential Evolution

Input: $Population_{size}$, $Problem_{size}$, $Weighting_{factor}$, $Crossover_{rate}$
Output: S_{best}
1 Population $\leftarrow$ `InitializePopulation`($Population_{size}$, $Problem_{size}$);
2 `EvaluatePopulation`(Population);
3 $S_{best} \leftarrow$ `GetBestSolution`(Population);

Diferansiyel gelişim algoritması

- DE/rand/1/bin için pseudocode - devam.

4 **while** $\neg$ `StopCondition()` **do**
5 | NewPopulation $\leftarrow \emptyset$;
6 | **foreach** $P_i \in$ Population **do**
7 | | $S_i \leftarrow$ `NewSample`(P_i, Population, $Problem_{size}$, $Weighting_{factor}$, $Crossover_{rate}$);
8 | | **if** `Cost`(S_i) $\leq$ `Cost`(P_i) **then**
9 | | | NewPopulation $\leftarrow S_i$;
10 | | **else**
11 | | | NewPopulation $\leftarrow P_i$;
12 | | **end**
13 | **end**
14 | Population $\leftarrow$ NewPopulation;
15 | `EvaluatePopulation`(Population);
16 | $S_{best} \leftarrow$ `GetBestSolution`(Population);
17 **end**
18 **return** S_{best};

Diferansiyel gelişim algoritması

- NewSample fonksiyonu.

Pseudocode for the `NewSample` function

```
   Input: P_0, Population, NP, F, CR
   Output: S
 1 repeat
 2 |   P_1 ← RandomMember(Population);
 3 until P_1 ≠ P_0 ;
 4 repeat
 5 |   P_2 ← RandomMember(Population);
 6 until P_2 ≠ P_0 ∨ P_2 ≠ P_1 ;
 7 repeat
 8 |   P_3 ← RandomMember(Population);
 9 until P_3 ≠ P_0 ∨ P_3 ≠ P_1 ∨ P_3 ≠ P_2 ;
```


Diferansiyel gelişim algoritması

- NewSample fonksiyonu - devam.

```
10 CutPoint ← RandomPosition(NP);
11 S ← 0;
12 for i to NP do
13 |   if i ≡ CutPoint ∧ Rand() < CR then
14 |   |   S_i ← P_3i + F × (P_1i - P_2i);
15 |   else
16 |   |   S_i ← P_0i;
17 |   end
18 end
19 return S;
```

Diferansiyel gelişim algoritması

Sezgiseller

- Diferansiyel gelişim algoritması, **nonlineer** ve **türevlenemez sürekli fonksiyonların optimizasyonu için tasarlanmıştır.**
- Ağırlık faktörü $F \in [0, 2]$ **arasındadır** (fark değişimi katsayısı).
- **Ağırlık faktörü** genellikle **0,8 seçilir.**
- **Crossover** ağırlığı $CR \in [0, 1]$ **arasında rastgele seçilir.**
- **Crossover** ağırlığı genellikle **0,9 seçilir.**
- **Başlangıç popülasyonu** tamamen **rastgele oluşturulur.**

Diferansiyel gelişim algoritması

Sezgiseller

- Algoritma kabul edilen konfigürasyonu tanımlayan özel bir **terminolojiye sahiptir (DE/x/y/z).**
- **x çeşitlendirilecek çözümü** (rastgele veya en iyi alınır),
- **y ise x üzerinde kullanılacak fark vektörlerini**,
- **z birleştirme operatörünü** (binomial veya exponential) gösterir.
- Popüler konfigürasyonlar:
  - **DE/rand/1/***
  - **DE/best/2/***

İçerik

- Evrimsel algoritmalar
- Diferansiyel gelişim algoritması
- Evrimsel programlama

Evrimsel programlama

Strateji

- Evrimsel programlama, **doğal seleksiyon teorisinden esinlenerek geliştirilmiştir.**
- **Bir türün popülasyonu tekrar üretilir** ve **küçük fenotipik değişkenliği olan nesiller oluşturur.**
- **Yeni nesil** ve **ebeveynleri probleme daha uygun olabilmek için yarışır.**
- Problemin **çözümüne daha uygun bireyler sonraki jenerasyona aktarılır.**

Evrimsel programlama

Strateji

- Yeni jenerasyona **aktarılan bireyler** kendi kendilerini **yeniden üretir.**
- **Yeni jenerasyon oluşturma**, problemin çözümü elde edilinceye kadar tekrar eder.
- Evrimsel programlamanın amacı, **uygun aday topluluğundaki birey sayısını maksimize etmektir.**
- Aday çözümlerin gösterimi, **fitness function tarafından doğrudan değerlendirilebilir.**

Evrimsel programlama

- Doğal seleksiyon teorisinden esinlenerek geliştirilmiştir.

Algorithm 3.6.1: Pseudocode for Evolutionary Programming.

```
Input: Population_size, ProblemSize, BoutSize
Output: S_best
Population ← InitializePopulation(Population_size, ProblemSize);
EvaluatePopulation(Population);
S_best ← GetBestSolution(Population);
while ¬StopCondition() do
    Children ← ∅;
    foreach Parent_i ∈ Population do
        Child_i ← Mutate(Parent_i);
        Children ← Child_i;
    end
```

Evrimsel programlama

- Doğal seleksiyon teorisinden esinlenerek geliştirilmiştir.

```
10    EvaluatePopulation(Children);
11    S_best ← GetBestSolution(Children, S_best);
12    Union ← Population + Children;
13    foreach S_i ∈ Union do
14        for 1 to BoutSize do
15            S_j ← RandomSelection(Union);
16            if Cost(S_i) < Cost(S_j) then
17                Si_wins ← Si_wins + 1;
18            end
19        end
20    end
21    Population ← SelectBestByWins(Union, Population_size);
22 end
23 return S_best;
```


Evrimsel programlama

- **Aday çözümlerin gösterimi probleme özgüdür** (Örneğin: sürekli bir fonksiyonun optimizasyonu için reel sayı kullanılabilir.).
- **Turnuva seçimi için örnek boyutu** genellikle **%5-%10 arasındadır.**
- Evrimsel programlama, mevcut aday çözümlerden **yeni aday çözümlerin oluşturulmasında, sadece mutasyon operatörünü kullanır.**
- **Crossover operatörü evrimsel programlamada kullanılmaz.**
- Evrimsel programlama, **ebeveyn** ve **çocuk aday çözümler arasındaki bağlantı ile ilgilenir.**
- Sürekli fonksiyon optimizasyonunda **Gaussian tabanlı mutasyon operatörü** kullanılır.

Ödev

- Evrimsel algoritmaların bilişim alanında uygulamasını içeren bir makale araştırma ödevi hazırlayınız.