Bacterial Foraging Optimization Algorithm

(Bakterilerin yiyecek arama optimizasyonu algoritması)

Pamukkale Üniversitesi, Çağatay Çalı

Optimizasyon Algoritmaları

Optimizasyon, bir problemde belirli koşullar altında mümkün olan alternatifler içinden en iyisini seçme işlemidir. Optimizasyon problemleri için birçok algoritma önerilmiştir. Sezgisel algoritmalar, büyük boyutlu optimizasyon problemleri için, kabul edilebilir sürede optimuma yakın çözümler verebilen algoritmalardır. Genel amaçlı sezgisel optimizasyon algoritmaları, biyoloji tabanlı, fizik tabanlı, sürü tabanlı, sosyal tabanlı, müzik tabanlı ve kimya tabanlı olmak üzere altı farklı grupta değerlendirilmektedir. Sürü zekâsı tabanlı optimizasyon algoritmaları kuş, balık, kedi ve arı gibi canlı sürülerinin hareketlerinin incelenmesi ile geliştirilmiştir. Bu çalışmada, sürü zekâsı optimizasyon algoritmaları (Ateşböceği Algoritması, Ateşböceği Sürü Optimizasyonu, Karınca Koloni Optimizasyonu, Parçacık Sürü Optimizasyonu, Yapay Balık Sürüsü Algoritması, Bakteriyel Besin Arama Optimizasyon Algoritması, Kurt Koloni Algoritması) tanıtılmış ve bu optimizasyonlardan kedi sürüsü optimizasyonu ile yapay arı koloni algoritması ayrıntılı olarak incelenmiştir.

Sürü Zekâsı Optimizasyon Algoritmaları

Sürü, birbirleriyle etkileşen dağınık yapılı bireyler yığını anlamında kullanılır. Bireyler insan veya karınca olarak ifade edilebilir. Sürülerde N adet temsilci bir amaca yönelik davranışı gerçekleştirmek ve hedefe ulaşmak için birlikte çalışmaktadır. Kolaylıkla gözlenebilen bu "kollektif zekâ" temsilciler arasında sık tekrarlanan davranışlardan doğmaktadır. Temsilciler faaliyetlerini idare etmek için basit bireysel kurallar kullanmakta ve grubun kalan kısmıyla etkileşim yolu ile sürü amaçlarına ulaşmaktadır. Grup faaliyetlerinin toplamından bir çeşit kendini örgütleme doğmaktadır. Aşağıda şimdiye kadar farklı araştırmacıların önerdiği sürü zekâsı optimizasyon algoritmaları alt başlıklar halinde açıklanmıştır.

Bakteriyel Besin Arama Optimizasyon Algoritması

Bakteriyel Besin Arama Algoritması, E. coli bakterisinin beslenme davranışından esinlenerek karmaşık mühendislik problemlerini çözmek için geliştirilmiş bir hesaplama tekniğidir. Bakteriler, karmaşık yaşam formlarındaki diğer canlılara göre çok daha basit yapıdadırlar. Sınırlı algı ve hareket kabiliyetlerini kullanarak optimum düzeyde enerji harcayıp beslenme faaliyetlerini gerçekleştirmeleri gerekmektedir. Diğer yaşam formlarına nazaran modellenebilmeleri daha kolaydır. Bu türden canlılardan biri olan E. coli bakterisi, yapısı ve çalışma şekli en iyi anlaşılan mikroorganizmalardan birisidir. E. coli bakterisi besin maddesine ulaştığında diğer bakterileri uyarıcı etkiye sahip kimyasal bir madde salgılamaktadır. Bu madde, diğer E. coli bakterilerinin besini bulan bakterinin bulunduğu yere doğru hareket etmesini sağlamaktadır. Eğer gıda yoğunluğu çok fazla ise bakteriler kenetlenerek grup halinde hareket edebilmektedirler (Başbuğ, 2008).

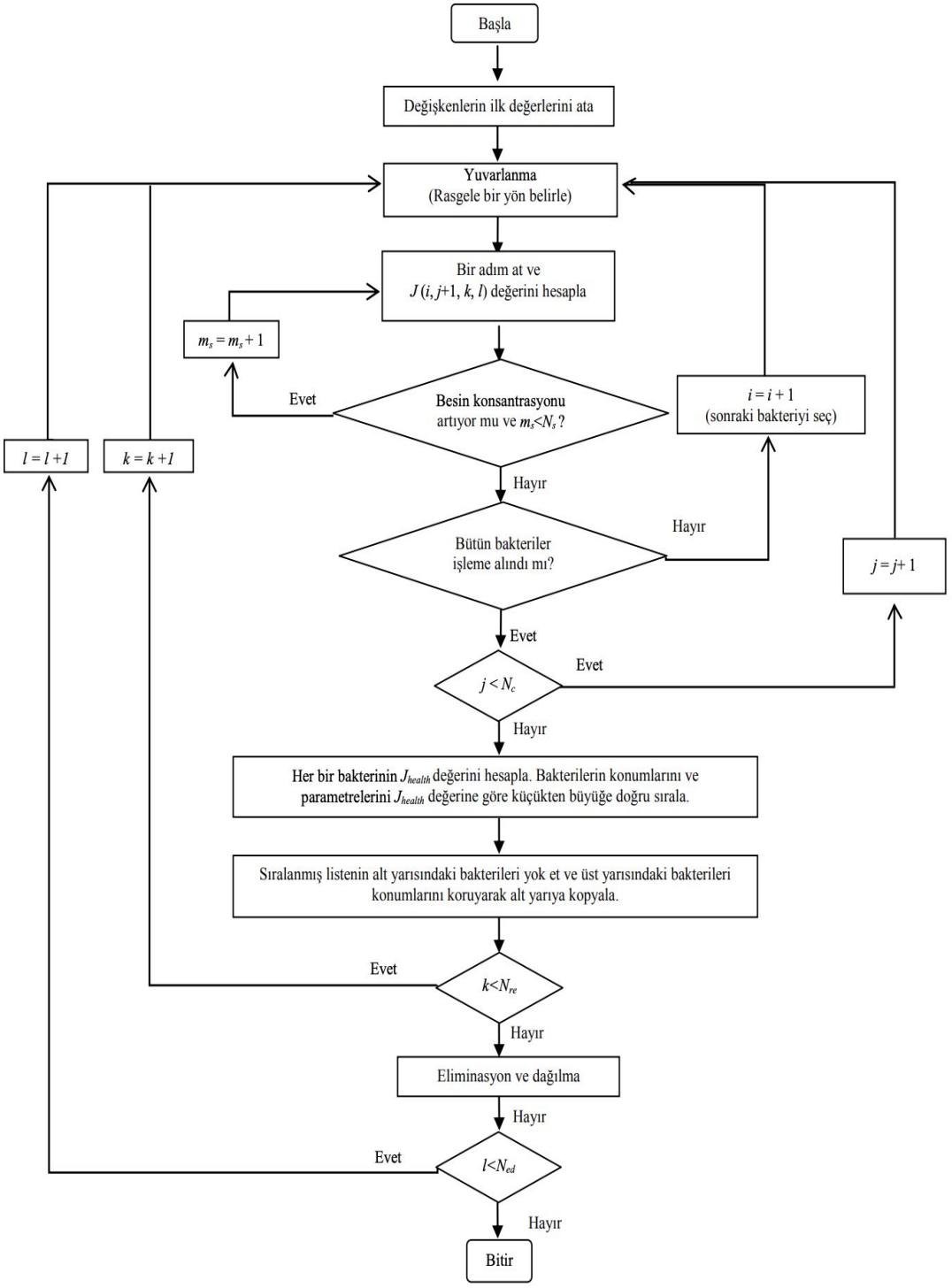
Algoritmanın Biyolojik Temelleri

Escherichia Coli (E. Coli) bakterisi, dünyadaki birçok canlının bağırsaklarında yaşayan bir bakteridir. E.Coli bakterilerinin besin arama sürecindeki davranışlarına dayanmaktadır. Algoritmada ilk adım, problemi algoritmaya uygun şekilde kodlamaktır. Sonrasında sürü bilgilerine göre bakterileri ayarlamak ve son olarak optimal çözümü aramaktır.

Algoritmanın optimizasyon döngüsü üç olaydan meydana gelir.

1. *Kemotaksis olayı (Chemotaxis event):* E. Coli bakterileri kamçılarıyla hareket etmektedirler. Bakterinin besin arama sürecinde, kamçıların dönmesi bakterinin şu anki ortamının değerlendirilmesine göre olur ve sonrasında şu anki pozisyonun değiştirilip değiştirilmeyeceğine ve bazı parametreler ışığında (bir sonraki hareketin yönü ve adım uzunluğu) nasıl değiştirileceğine karar verilir. BBAOA'da yön değiştirmenin formülü şu şekildedir: $\theta\ i\ (j+1,k,l) = \theta i\ (j,k,l)+C(i)\varphi(j)\ (1)$ Burada θi (j,k,l) i. bakterinin, şu anki pozisyonunu göstermektedir. j, k ve l, kemotaksis, üreme, eliminasyon ve dağılma olaylarının indislerini göstermektedir. φ(j) kamçı hareketine bağlı olan hareket yönünü ifade ederken, C(i) adım uzunluğuna karşılık gelmektedir.
2. *Üreme olayı (Reproduction event)*: Üreme olayı şu şekilde uygulanır: Popülasyondaki bakteri sayısı S ise, S/2 sayıda bakteri popülasyondan çıkarılır. Öncelikle her bir bakteri, konumlarındaki değerlendirme kriterine göre sıralanır. Sonrasında, sıralamanın son yarısındaki bakteriler popülasyondan çıkarılarak ilk yarısındakilerin her birinin aynı konumda yer alacak şekilde bir kopyası alınır.
3. *Eliminasyon ve dağılma olayı (Elimination and dispersal event):* Eliminasyon ve dağılma olasılığına (Ped) bağlı olarak gerçekleştirilir. Belirli bir bakteri dağılma olayına maruz kalırsa, o bakteri yok edilir ve yeni bir bakteri üretilir. BBAOA'nın temel adımları şu şekildedir [16]:
    a. Adım 1: Popülasyonu oluşturma
    b. Adım 2: Bakterileri değerlendirme fonksiyonuna göre değerlendir
    c. Adım 3: Optimizasyon için üç döngü:
        i. İç döngü: Kemotaksis olayı,
        ii. Orta döngü: Üreme olayı,
        iii. Dış döngü: Eliminasyon ve dağılma olayı
    d. Adım 4: Son çözümü belirlemek için optimal bakterinin kodunu çöz.

BBAOA ile ilgili daha detaylı bilgi konusunda Passino [16] çalışması incelenebilir. BBAOA'nın akış diyagramı Şekil 1'de verilmiştir.



Şekil 1: BBAOA'nın akış diyagramı

Algoritmanın Sözde Kodu: (Kaba kod)

```
Input: Problem_size, Cells_num, N_ed, N_re, N_c, N_s, Step_size, d_attract, w_attract, h_repellant, w_repellant, P_ed
Output: Cell_best
Population ← InitializePopulation(Cells_num, Problem_size)
For (l = 0 To N_ed)
  For (k = 0 To N_re)
    For (j = 0 To N_c)
      ChemotaxisAndSwim(Population, Problem_size, Cells_num, N_s, Step_size, d_attract, w_attract, h_repellant,
w_repellant)
      For (Cell ∈ Population)
        If (Cost(Cell) ≤ Cost(Cell_best))
          Cell_best ← Cell
        End
      End
    End
    SortByCellHealth(Population)
    Selected ← SelectByCellHealth(Population, Cells_num / 2)
    Population ← Selected
    Population ← Selected
  End
  For (Cell ∈ Population)
    If (Rand() ≤ P_ed)
      Cell ← CreateCellAtRandomLocation()
    End
  End
End
Return (Cell_best)
```

Pseudocode for the BFOA.

```
Input: Population, Problem_size, Cells_num, N_s, Step_size, d_attract, w_attract, h_repellant, w_repellant
For (Cell ∈ Population)
  Cell_fitness ← Cost(Cell) + Interaction(Cell, Population, d_attract, w_attract, h_repellant, w_repellant)
  Cell_health ← Cell_fitness
  Cell' ← ∅
  For (i = 0 To N_s)
    RandomStepDirection ← CreateStep(Problem_size)
    Cell' ← TakeStep(RandomStepDirection, Step_size)
    Cell'_fitness ← Cost(Cell') + Interaction(Cell', Population, d_attract, w_attract, h_repellant, w_repellant)
    If (Cell'_fitness > Cell_fitness)
      i ← N_s
    Else
      Cell ← Cell'
      Cell_health ← Cell_health + Cell'_fitness
    End
  End
End
```

Pseudocode for the `ChemotaxisAndSwim` function.

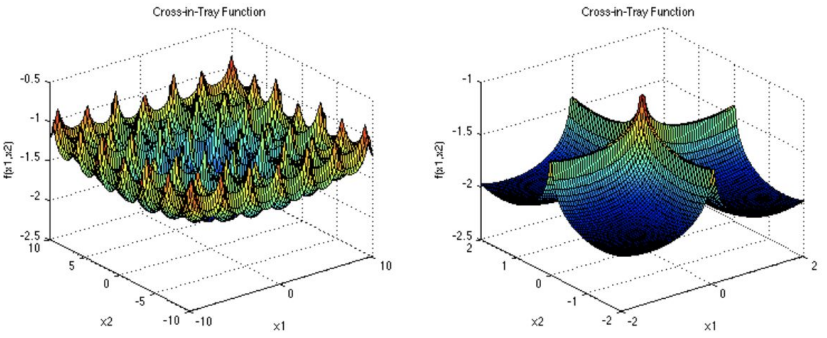
Clever Algorithms

Algoritmanın akış şeması, sözde kodu ve örnek uyarlanmış matlab kodu dikkate alınarak (Bacterial Foraging Optimization Rolando Gonzales 2015), Julia programlama dili kullanılarak daha öncesinden belirlenmiş problemlerin optimizasyon algoritması ile çözümlenmesi için gerekli kod yazılmıştır. Jupyter notebook Julia çekirdeği kullanılarak algoritmaların çözümlenmesi sağlanmıştır.

İlgili Optimizasyon Test Problemleri

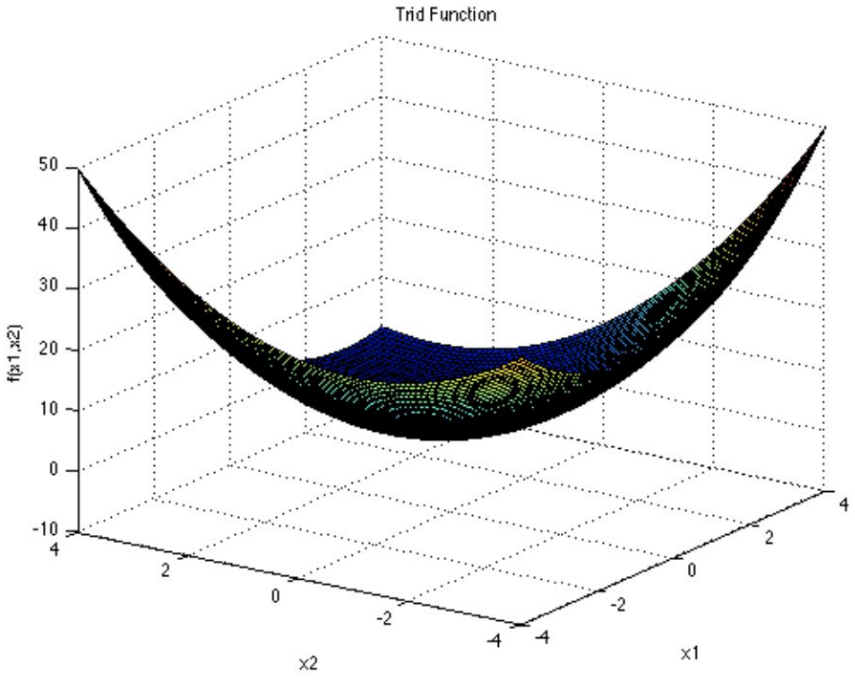
Problemler BFO.ipynb dosyasında ilişkilendirilmiştir. Açıklamalar ve yorum satırları ile Julia diline uyarlanmıştır.

1. Cross-in-Tray



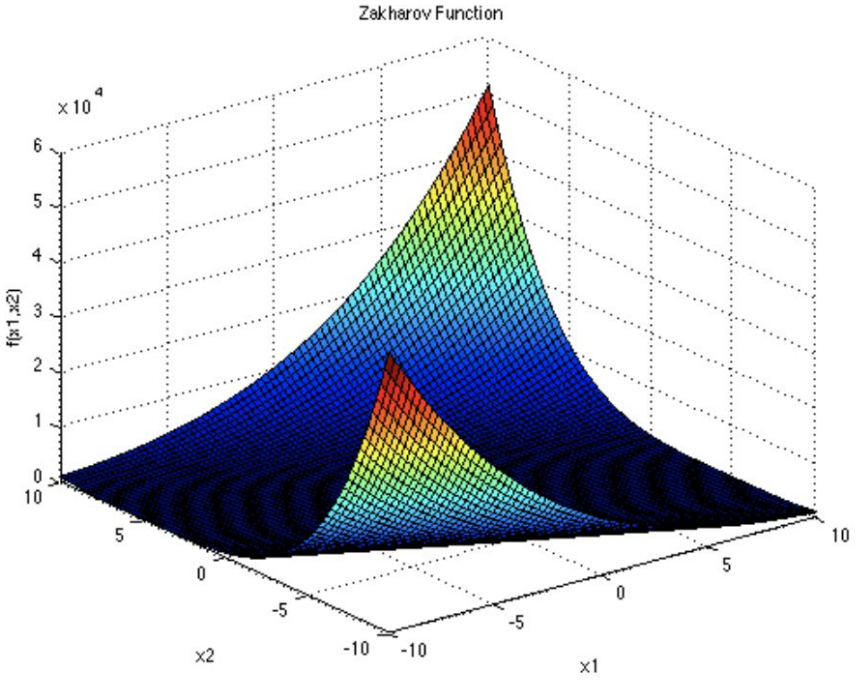
$$f(\mathbf{x}) = -0.0001 \left(\left| \sin(x_1)\sin(x_2)\exp\left(\left| 100 - \frac{\sqrt{x_1^2 + x_2^2}}{\pi} \right| \right) \right| + 1 \right)^{0.1}$$

2. Trid



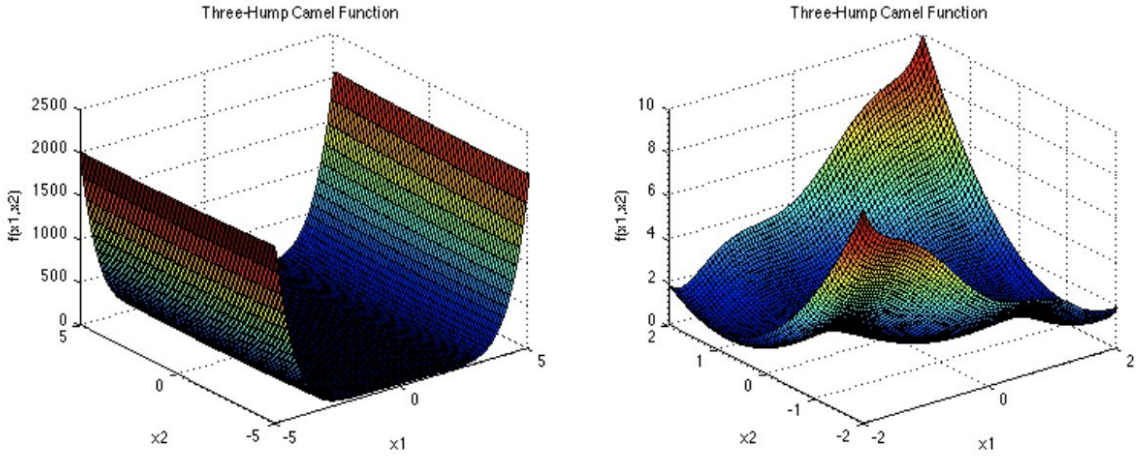
$$f(\mathbf{x}) = \sum_{i=1}^{d} (x_i - 1)^2 - \sum_{i=2}^{d} x_i x_{i-1}$$

3. Zakharov



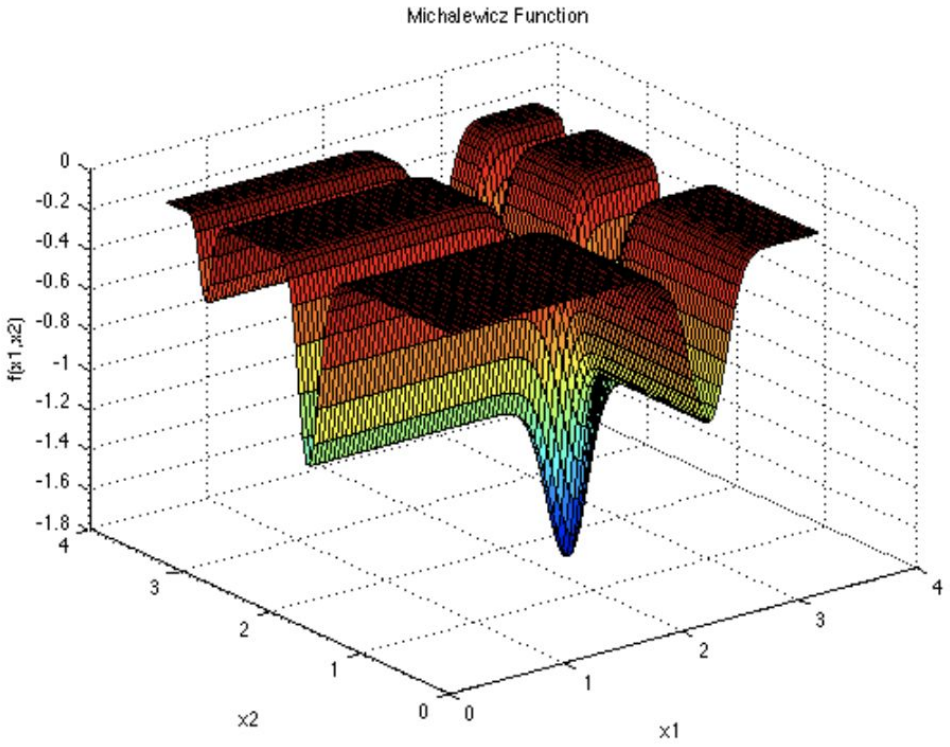
$$f(\mathbf{x}) = \sum_{i=1}^{d} x_i^2 + \left(\sum_{i=1}^{d} 0.5ix_i\right)^2 + \left(\sum_{i=1}^{d} 0.5ix_i\right)^4$$

4. Three-Hump Camel



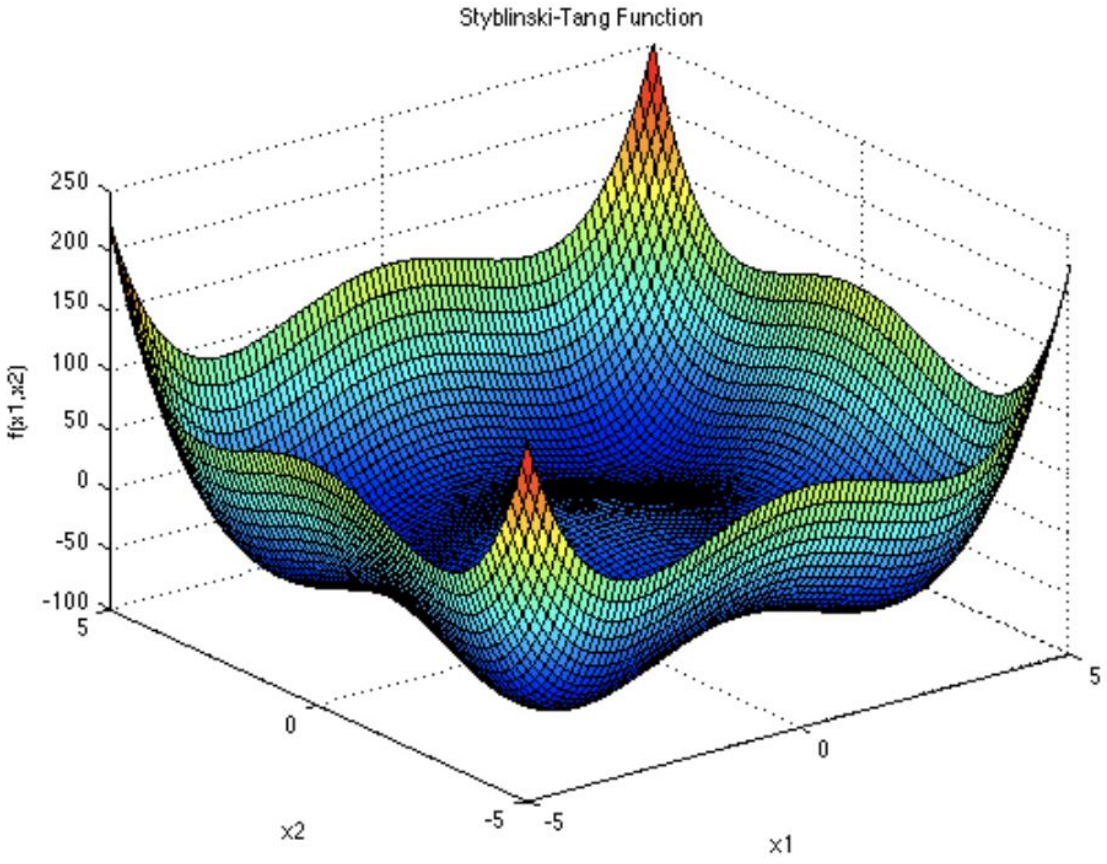
$$f(\mathbf{x}) = 2x_1^2 - 1.05x_1^4 + \frac{x_1^6}{6} + x_1x_2 + x_2^2$$

5. Michalewicz



$$f(\mathbf{x}) = -\sum_{i=1}^{d} \sin(x_i) \sin^{2m}\left(\frac{i x_i^2}{\pi}\right)$$

6. Styblinski-Tang



$$f(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{d}(x_i^4 - 16x_i^2 + 5x_i)$$

Algoritmalar Matlab programlama dilinden Julia programlama diline uyarlanmış olup, bakterilerin yiyecek arama algoritmasını test etmek amaçlı Jupyter Notebook üzerinden grafiksel olarak (plot) nihayetlendirilmiştir. İlgili kaynak kod ektedir, çevrimiçi erişmek için GitHub'dan görüntülenebilir.

Kaynakça

- Başbuğ S. 2008. Bakteriyel besin arama algoritması ile lineer anten dizilerinin diyagram sıfırlaması. Yüksek Lisans Tezi, Fen Bilimleri Enstitüsü, Konya.
- Biswas, A., Dasgupta, S., Das, S., Abraham A. 2007a. Synergy of pso and bacterial foraging optimization: a comparative study on numerical benchmarks. Second International Symposium on Hybrid Artificial Intelligent Systems
- Biswas, A., Dasgupta, S., Das, S., Abraham, A., 2007b A synergy of differential evolution and bacterial foraging algorithm for global optimization. International Journal on Neural and Mass-Parallel Computing and Information Systems Neural Network World, 17: 6, 607-626.
- Mishra, S., 2005 A hybrid least square-fuzzy bacteria foraging strategy for harmonic estimation. IEEE Trans. Evol. Comput. 9: 1, 61–73.