Optimizasyon Problemleri için Metasezgisel Yöntemler

Dr.Öğr.Üyesi Rafet Durgut

Final Projesi

Optimizasyon Problemi Çözümü

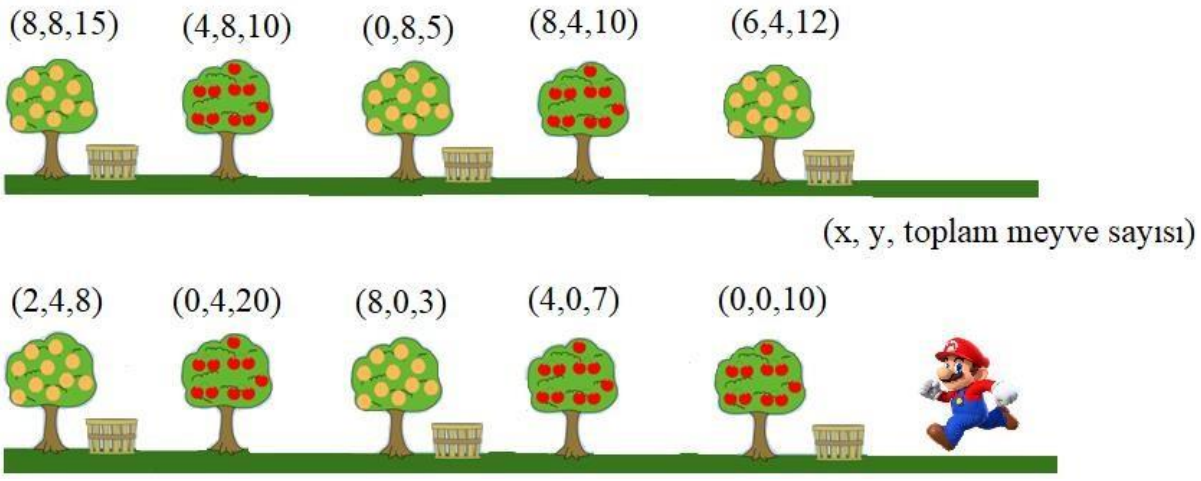
İbrahim Mete Çiçek – 2028130021

Problemin Açıklanması

Bu final projesi için meyve toplama problemini (Facility location problem) düşündüm. Problemde bir fabrikanın büyük meyve bahçesi var ve bu bahçede x koordinatı, y koordinatı ve toplam meyve sayısı kullanılarak temsil edilen 2 boyutlu uzaya yayılan n ağaç (n = 10) vardır. Yapılan iş miktarının minimum olması için 6 sepet (12 gerçek değerli karar değişkeni) kullanmamız gerekir. Amacımız ağaçtan ağaca giderken yapılan işi yani hareketin cost'unu minimum olmasını sağlayan yolu seçmek diğer bir deyişle minimum cost'u elde etmektir.

Bu projede Yapay Arı Kolonisi (Artificial Bee Colony), Tepe Tırmanma (Hill Climbing), Benzetilmiş Tavlama (Simulated Annealing),Geç Kabul Edilen Tepe Tırmanma (Late Accepted Hill Climbing), Genetik Algoritması (Genetic Algorithm) gibi farklı optimizasyon algoritmaları sıfırdan uygulanmaktadır.
Kovaryans Matris adaptasyonu ve Parçacık Sürüsü Algoritması (Particle Swarm Algorithm) gibi algoritmalar için harici Python kütüphaneleri kullanılmıştır. Son olarak, her algoritmanın rastgele ağaç üzerinde beş kez çalıştığı ve amaç fonksiyonunun ortalama ve standart sapmasının sunulduğu bir algoritma karşılaştırması tablo halinde sunulmuştur.

Burada ele alınan sorun, her biri (x, y) koordinatları ve her ağaçtaki toplam meyve sayısı kullanılarak temsil edilen n sayıda ağaca sahip olduğumuz tesis konumu problemidir. Bu bilgi"meyve_toplama.txt" dosyasında x_koordinatı, y_koordinatı ve meyve_sayısı dosyası olarak bulunur. Hareket miktarının minimum olması için m kutu yerleştirmemiz gerekiyor. Burada m= 6 kutu, yani toplam 12 gerçek değer karar değişkeni yerleştirmemiz gerekiyor.



meyve_toplama.txt dosyasını okuyacaksanız: 10 satır vardır. Her satır (x-koordinatı, y-koordinatı, o ağaçtaki toplam meyve sayısı) gibi değerlere sahip bir ağacı temsil eder. Yani ilk satırı (0,0,10) okursanız 0,0'da 10 meyve olan bir ağaç var demektir.

Bu boşlukta tutmamız gereken 6 sepetimiz var, bu 12 koordinat / gerçek değer bulmamız gerektiği anlamına geliyor (her sepetin x-koordinatı ve y-koordinatı vardır) bu nedenle, amacımız minimum hedef olacak şekilde 12 gerçek değer bulmalıyız yani ağaçlarda toplanan en yakın meyve sepeti olan, ağırlıklı mesafeyi en aza indirir.

Optimizasyon işlevi: tüm ağacın toplamı (her ağaçtaki meyve * ağacın en yakın sepetine uzaklığı)

Amaç fonksiyonu

$$\textbf{Objective function = Amount of work done = Minimize } f(x_1,x_2,....,x_{2m}) = \Sigma_i\, w_i\, \min_j(d(t_i,b_j))$$

d: Öklid uzaklığı

wi: Ağaçtaki toplam meyve sayısı

x: Karar değişkenleri

ti: Ağaçların bulunduğu konumlar

bj: Sepetlerin bulunduğu konumlar

Amaç fonksiyonumuz aslında en aza indirmemiz gereken maliyet fonksiyonudur. Yani bu bir minimization problemidir. Bu tür optimizasyon problemlerinde mesafe yerine koordinatlar verilir ve kullanılan algoritmalar ona göre uzaklıkları hesaplayarak cost değeri bulur. Birkaç şehre hizmet vermek için bir süpermarketin nereye yerleştirileceğini seçerken de tamamen aynı sorun ortaya çıkıyor.

Algoritma seçimi:

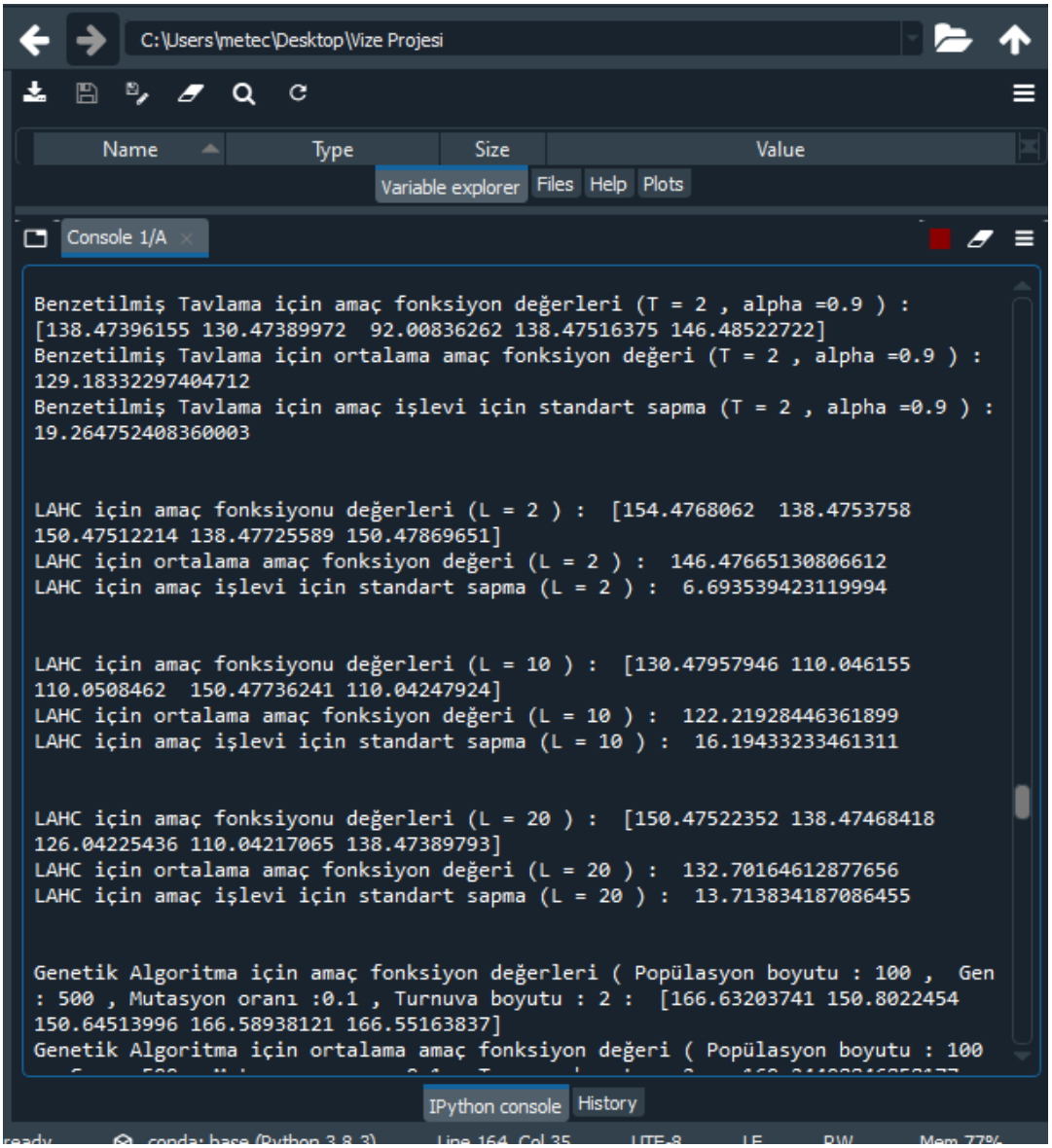
Artificial Bee Colony ABC, Hill Climbing HC, Simulated Annaling SA (Başlangıç sıcaklığı T, Temp. Decay rate alpha), Late Acceptance Hill Climbing LAHC (History length L), Genetic algoritması GA (Population Size P, Kuşak sayısı G, Mutasyon Oranı M, Turnuva boyutu T), Kovaryans Matris Uyarlama CMA (Popülasyon boyutu P) ve son olarak Parçacık sürüsü Optimizasyonu PSO (Parçacık hızı ölçeklendirme faktörü W, P en iyi ölçekleme faktörü CP, Gbest ölçeklemefaktörü CG).

Ayrıca, fitness değerlendirme bütçemizi her algoritma ve popülasyon tabanlı algoritmalar için 50.000 ile sınırlandırdık. Bütçe: budget = popsize * ngens.

Çözümün Uygulanması

Python'da problemin optimizasyonu için genetik algoritmasını uygulayacak olursak, projemizin çıktısı aşağıda bulunan şekildeki gibi oluyor. İsterseniz şimdi süreci başından anlatalım. İlk olarak program daha anlaşılır olsun diye her koda comment olarakaçıklama ekledim. Programdaki iterasyon sayısı isteğe bağlı olarak değiştirilebilinir.

Projede amacımız olan benzetilmiş tavlama kullanarak özellik seçimi yapılabilmesi için genetik algoritma ve karar ağacı sınıflandırmasından yararlanılarak bir kod bloğu oluştururdu ve sonuç kısmında bu programın olasılık hesaplarının yanı sıra grafiksel olarak da çıktıları elde edildi.

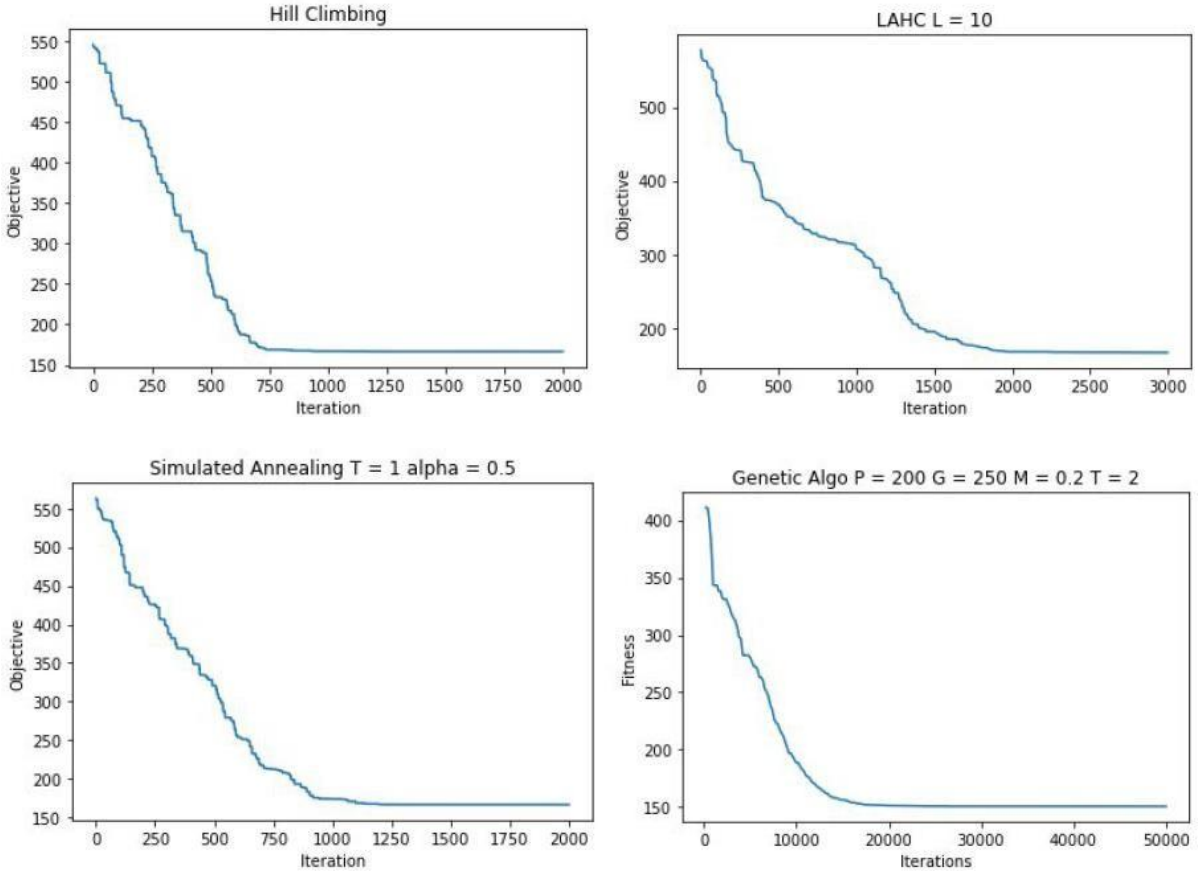


Deneysel tasarım:

Bu deney için, her bir optimizasyon algoritması için hiperparametre sayısını değiştirenstandart faktöriyel tasarımı dikkate aldık. Aşağıda gözlem tablosu:

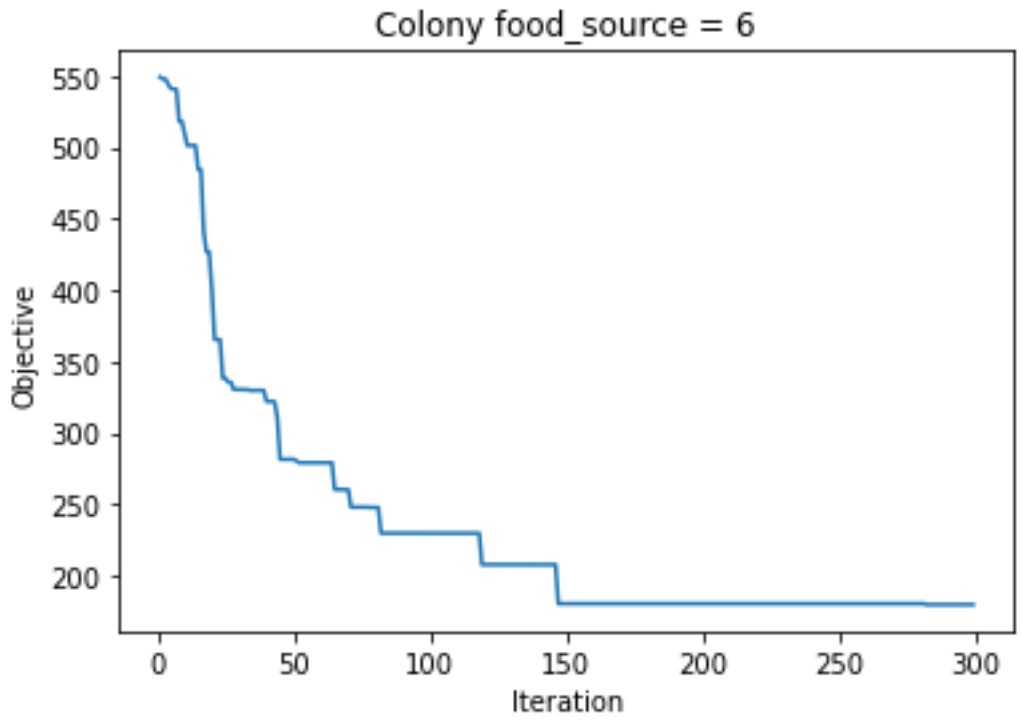
Number	Algorithm	Param 1	Param 2	Param 3	Param 4	Obj. Mean	Obj. Std
1	HC	None	None	None	None	166.472992089	0.00144282126
2	SA	T=1	alpha=0.5	None	None	160.873341994	6.97426134480
3	SA	T=1	alpha=0.9	None	None	149.675299433	15.2620742312
4	SA	T=2	alpha=0.5	None	None	163.274173077	6.40058763596
5	SA	T=2	alpha=0.9	None	None	139.273875228	3.91926213101
6	LAHC	L=2	None	None	None	142.474381190	5.05821926105
7	LAHC	L=10	None	None	None	128.387331119	31.3675077547
8	LAHC	L=20	None	None	None	132.165396314	15.1839485588
9	GA	P=100	G=500	M=0.1	T=2	157.100584968	7.82689336872
10	GA	P=100	G=500	M=0.2	T=2	150.541201248	0.02633051417
11	GA	P=200	G=250	M=0.1	T=2	150.720782980	0.09180058409
12	GA	P=200	G=250	M=0.2	T=2	149.410270353	2.28201608506
13	CMA	P=10	None	None	None	96.8498447256	13.3346914426
14	CMA	P=100	None	None	None	70.6832815730	5.36656314596
15	CMA	P=200	None	None	None	68.0000000000	7.0485839e-5
16	PSO	W=0.5	CP=0.5	CG=0.5	None	82.4618125553	11.1715928550
17	PSO	W=0.5	CP=0.5	CG=0.9	None	75.1610568684	7.06410655617
18	PSO	W=0.5	CP=0.9	CG=0.5	None	74.1301686804	6.62400698337
19	PSO	W=0.5	CP=0.9	CG=0.9	None	77.9993539430	4.25961091501
20	PSO	W=1	CP=0.5	CG=0.5	None	76.1540917550	5.87013540415
21	PSO	W=1	CP=0.5	CG=0.9	None	71.4149603427	3.46251961388

Plots (Objective Function vs Iteration):



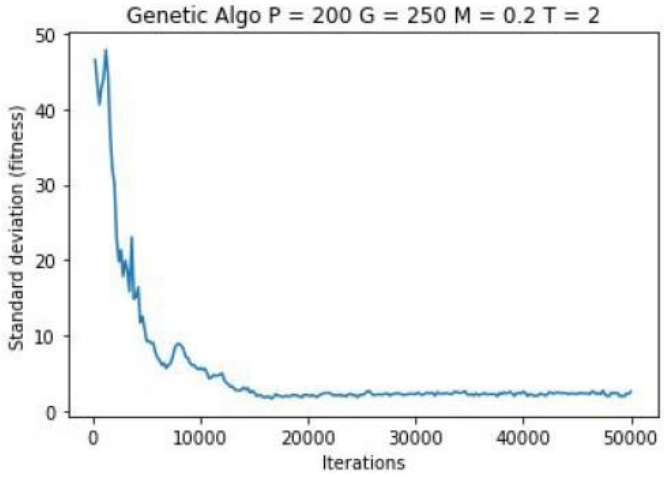
Artificial Bee Colony ABC (Yapay Arı Kolonisi):

```
Console 1/A

 ABC için amaç fonksiyonu değerleri (#bees = 6) :  [187.58510243 189.69311343
185.60341659 140.80062929 178.07755422]
ABC için ortalama amaç fonksiyonu değeri (#bees  = 6) :  176.35196319254186
ABC için amaç fonksiyonu için standart sapma (#bees  = 6) :  18.20238502588163


 ABC için amaç fonksiyonu değerleri (#bees = 7) :  [182.28890692 182.69050743
199.39070602 169.74048883 201.08556729]
ABC için ortalama amaç fonksiyonu değeri (#bees  = 7) :  187.03923529754655
ABC için amaç fonksiyonu için standart sapma (#bees  = 7) :  11.752293006788886


 ABC için amaç fonksiyonu değerleri (#bees = 8) :  [175.69585855 187.09389255
177.20192875 178.77192776 157.12074272]
ABC için ortalama amaç fonksiyonu değeri (#bees  = 8) :  175.17687006553373
ABC için amaç fonksiyonu için standart sapma (#bees  = 8) :  9.852246556329929


 ABC için amaç fonksiyonu değerleri (#bees = 9) :  [187.9932962  132.50622984
182.06547393 181.5888634  179.80099013]
ABC için ortalama amaç fonksiyonu değeri (#bees  = 9) :  172.79097069964251
ABC için amaç fonksiyonu için standart sapma (#bees  = 9) :  20.329932341876198

In [2]:
```



Plot (Objective Function standard deviation vs Iteration):

Aşağıdaki Genetik Algoritmasının grafiksel çıktısını yorumlayacak olursak. Grafikte fitness'ın iterasyon sayısına göre değişimi gösteriliyor. Fitness'ın treshold'u 4 olarak kabul edersek, bu değerden sonraki değerlerde model sabit yani en optimal olan sonuç çıkıyor.



Sonuçlar / Gözlemler:

Bu deneyden sonra, CMA'nın 68 ortalama objektif değeri ve 7.0485839e-5 minimum standartsapması ile diğer tüm algoritmalar arasında en iyi performansı gösterdiğini gözlemleyebiliriz ki bu onu çok kararlı bir algoritma yapar.

Ayrıca PSO'nun, W = 1, CP = 0.5 ve CG = 0.9 yapılandırması için 71.41496 ortalama hedef değeriyle CMA ile eşit ve diğerlerinden daha iyi performans gösterdiğini gözlemledik.

Arama ortamının doğası, biraz gürültüyle çok modlu görünmektedir. Yukarıdaki amaç fonksiyonunun birden fazla optimal çözümü vardır.

Nüfus tabanlı algoritmanın bütçeyi sabit tutması için algoritma, nüfus artırıldığında iyi performans gösterir. Bunu G = 100 objektif değerlerinin 157 ve 150 çıkması durumundagözlemleyebiliriz, oysa G = 200 için 150 ve 149'dur (GA Algoritması). Daha yüksek popülasyon değerleri kullanarak daha fazla test yapabiliriz.

Yukarıdaki nesnel fonksiyona karşı yineleme grafiklerinden, algoritmaların yaklaşık 2000 yinelemeye yakınlaştığı sonucuna varabiliriz. Bu nedenle, algoritma son derece sömürücü olacağı ve araştırıcı olmayacağı için daha fazla yinelemenin faydası olmayacaktır.