İstanbul Üniversitesi İşletme Fakültesi Dergisi
Istanbul University Journal of the School of Business Administration
Cilt/Vol:37, Sayı/No:1, 2008, 22-38
ISSN: 1303-1732 - www.ifdergisi.org © 2008

Doğrusal olmayan küresel optimizasyon problemleri için tabu arama algoritmasının kullanılması

Tunçhan Cura[1]
Sayısal Yöntemler Anabilim Dalı, İşletme Fakültesi
İstanbul Üniversitesi, İstanbul, Türkiye

Özet

Tabu arama algoritmaları atama, programlama (scheduling), en kısa yol ve gezen satıcı gibi tümleşik optimizasyon problemlerinde sıklıkla kullanılmasına karşılık, sürekli küresel optimizasyon problemlerinde çok nadir kullanılmaktadır. Bu çalışmada doğrusal olmayan fonksiyon optimizasyonu problemi tabu arama algoritması kullanılarak çözülmüştür. Geliştirilen algoritma yedi değişkenli bir minimizasyon probleminde test edilmiş ve anlamlı sonuçlar elde edilerek bu tür problemler için tabu arama algoritmasının nasıl kullanılabileceğine dair örnek teşkil etmiştir. Örnek problemin çözümü açıklanırken ileride anlatılacak olan tabu yapıları çok fazla olduğundan yalnız ilk ve beşinci tabu yapıları gösterilmiştir.

Anahtar Sözcükler: *Tabu arama, optimizasyon, doğrusal olmayan*

Using tabu search algorithm for nonlinear global optimization problems

Abstract

Although tabu search algorithms have been used for combinatorial problems such as assignment, scheduling, shortest path, and travelling salesman problem often, they have been used for continous global optimization problems rarely. In this study, the optimization of a nonlinear function problem is solved by using tabu search algorithm. The algorithm, which is developed in this study, is tested in a minimization problem having seven variables and it forms an example that shows how to use tabu search algorithm for such problems with geting rational results. Because of a large number of tabu structures which are explained ahead, just the first and the fifth tabu structures are shown while explaining how the sample problem is solved.

Keywords: Tabu search, optimization, nonlinear.

1. Giriş

Tabu arama(TA), başlangıçta tümleşik optimizasyon problemleri için geliştirilmiş son sezgisel yaklaşımlardan birisidir. TA bu tür problemlere uygulandığında başarılı bir performans göstermiştir. Ancak TA' nın sürekli optimizasyon problemlerinin çözümüne katkıları, tavlama benzetimi (simulated annealing) algortiması ve genetik algoritmalar gibi diğer sezgisel yaklaşımlarla kıyaslandığında hala oldukça sınırlıdır [1]. Bu çalışmada sürekli küresel optimizasyon problemlerinin çözümüne yönelik bir TA yaklaşımı

[1] tunchan@istanbul.edu.tr (T. Cura)

geliştirilmiş, özellikle doğrusal olmayan minimizasyon problemleri çalışma kapsamı içinde tutulmuştur.

$$\min_{x \in R^n} f(x)$$

Görüldüğü gibi f, R^n içinde tanımlı gerçel değerli bir fonksiyondur. Ancak bu çalışmada kullanılan teknik hiç değiştirilmeksizin maksimizasyon problemleri için de kullanılabilir. Bilindiği gibi;

$$\max_{x \in R^n} f(x) = \min_{x \in R^n} \; -f(x)$$

olduğundan, maksimizasyon fonksiyonu -1 ile çarpıldığı taktirde minimize edilerek optimum nokta bulunacaktır. TA' nın doğrusal olmayan sürekli bir optimizasyon probleminde küresel minimum noktaya yaklaşması çok fazla hesaplama gerektirdiğinden, Hedar ve Fukushima' nın [1] yapmış oldukları çalışma örnek alınarak lokal arama metotlarından da yararlanılmıştır.

TA çözümde bir değişiklik yapmayı kabul etmeden önce mevcut çözümlerin komşularının, başka bir ifadeyle yakın çözümlerin bulunduğu kümede arama yapar. Çözüm uzayında yapılabilecek mümkün hareketler kümesinin üretilmesi ve bunlardan birisinin kabul edilmesi iterasyonlar boyunca devam eder [2]. TA' nın temelleri, uygunluk sınırlarını veya doğal olarak bariyer vazifesi yapan lokal optimalliği aşmak ve sistematik olarak kısıtları zorlayarak yasak alanlarda araştırma yapmaya izin vermek için dizayn edilmiş metotlara dayanır. Bu tür prosedürlerin ilk örnekleri, sistematik olarak uygunluk koşullarını zorlayan vekil kısıt metotlarını (surrogate constraint methods) ve düzlem kesme yaklaşımlarını (cutting plane approaches) temel alan sezgisel yaklaşımları kapsamaktaydı. TA' nın modern biçimi Glover ile şekillenmiştir. Metodun bazı türetilmiş fikirleri ise Hansen tarafından geliştirilmiştir [3].

Glover' in ilk sunumundan bu yana TA alanında çok sayıda çalışma ortaya çıkmıştır. Bu çalışmaların büyük çoğunluğu türetilmiş optimizasyon problemleri ile ilgiliyken, sürekli optimizasyon problemlerine yönelik birkaç çalışma olmuştur [1]. Battiti ve Tecchiolli [4] "sürekli tepkili Tabu Arama" (continuous reactive Tabu search) başlıklı ilgi çekici bir sürekli TA metodu sunmuştur. Metotları, en umut verici, başka bir ifadeyle içerisinde optimuma en yakın çözümleri barındıran kutucukların yerini belirlemeye ve daha sonra lokal arama için bu kutucuklar içerisinden başlangıç noktası seçmeye çalışmaktadır. Bu çalışmada buna benzer bir yöntem kullanılmıştır.

2. TA HAFIZA ELEMANLARI

Hafıza kavramı, özellikle geniş alanlı hafızaya sahip yüksek düzeyli TA' da olduğu gibi bir tür öğrenme süreci içinde kullanıldığında TA' da temel role sahiptir. Kuvvetlendirme ve çeşitlendirme şemalarında etkili hafıza kullanımı, TA' yı akıllı bir arama tekniği yapar [1].

TA lokal optimalliğin ötesindeki çözüm uzayını araştırmak için lokal sezgisel arama prosedürü sunan bir tekniktir. TA' nın ana parçalarından birisi uyarlanabilir hafızasıdır. Böylece daha esnek bir arama davranışı ortaya çıkar. Bu sebeple hafıza temelli stratejiler TA yaklaşımlarının ayırt edici özellikleridir [5].

Tabu yapısı (tabu structure) olarak da bilinen hafıza listesinden bu çalışmada iki adet kullanılmıştır. Daha önce de ifade edildiği gibi, bu çalışmada TA algoritması öncelikli olarak en umut verici kutucukların yerini belirlemeye çalışmaktadır. Bu durumda minimize edilmeye çalışılan amaç fonksiyonundaki her bir değişken için 10 kutucuk

oluşturulmuştur. Kutucuklar farklı aralıklardaki gerçek sayılarla doludur. Örnek olarak aşağıdaki fonksiyon alınmıştır:

$min\ f(x, y) = x^2-4x+y^2-y-xy$ (1)

Amaç fonksiyonunda görüldüğü gibi iki değişken vardır. İki adet tabu yapısı olduğundan bahsedilmişti. Söz konusu iki yapı her bir değişken için geçerli olduğundan, toplam dört tabu yapısı olmaktadır. Tabu yapıları da 10 kutucuktan oluşmaktadır. Kutucuk sayısı azaltılıp çoğaltılabilir. Bu, problemin yapısına ve beklenen çözüm süresinin uzunluğuna bağlıdır. Üçüncü kısımda görüleceği gibi kutucuk sayısının arttırılması güvenilir sonuç elde edilebilmesi için gerekli olan iterasyon sayısının artmasına neden olacaktır. Böylece çözüm süresi uzayacaktır. Buna göre ilk tabu yapısı tablo 1 ve tablo 2' deki gibi olur.

Tablo 1: Birinci değişkenin tabu süresi için tabu yapısı

Değişken	Alt:-100,000,000 Üst:-80,000,000	Alt:-80,000,000 Üst:-60,000,000	Alt:-60,000,000 Üst:-40,000,000	Alt:-40,000,000 Üst:-20,000,000	Alt:-20,000,000 Üst:0	Alt:0 Üst:20,000,000	Alt:20,000,000 Üst:40,000,000	Alt:40,000,000 Üst:60,000,000	Alt:60,000,000 Üst:80,000,000	Alt:80,000,000 Üst:100,000,000
	1	2	3	4	5	6	7	8	9	10
x	0	0	0	0	0	0	0	0	0	0

Tablo 2: İkinci değişkenin tabu süresi için tabu yapısı

Değişken	Alt:-100,000,000 Üst:-80,000,000	Alt:-80,000,000 Üst:-60,000,000	Alt:-60,000,000 Üst:-40,000,000	Alt:-40,000,000 Üst:-20,000,000	Alt:-20,000,000 Üst:0	Alt:0 Üst:20,000,000	Alt:20,000,000 Üst:40,000,000	Alt:40,000,000 Üst:60,000,000	Alt:60,000,000 Üst:80,000,000	Alt:80,000,000 Üst:100,000,000
	1	2	3	4	5	6	7	8	9	10
y	0	0	0	0	0	0	0	0	0	0

Bu tabu yapıları, tabu süresini (tabu tenure) hafızada tutmaktadır ve başlangıçta tüm hücreler sıfır değerine sahiptir. Her bir farklı değişken için farklı tablo kullanılmasının sebebi her bir değişkenin arama aralığının farklı olabileceğinden kaynaklanmaktadır. Örnekte her iki değişken için söz konusu aralıklar eşittir ve -100,000,000 ile 100,000,000 aralığındadır. Ancak çoğunlukla farklı olmaları gerekecektir.

$$\left.\begin{array}{l} x = 65312455 \\ y = -12861631 \end{array}\right\} \Rightarrow \begin{array}{l} x\ için\ tabusüreyapisi_9 = tabu \quad süresi \\ y\ için\ tabusüreyapisi_5 = tabu \quad süresi \end{array}$$

olur. Bunun anlamı *x için tabusüreyapısı*$_9$ ve *y için tabusüreyapısı*$_5$ 0' a eşit olana kadar bu kutucuklardan *x* ve *y* değişkenleri için değer seçilemeyeceğidir. Başka bir ifadeyle bu çalışmada kullanıldığı gibi *tabusüresi = 3* olarak alınırsa üç iterasyon boyunca *x* değişkenine 60,000,000 ile 80,000,000 arasından, *y* değişkenine ise -20,000,000 ile 0 arasından bir değer atanamayacaktır.

Dikkat edilmesi gereken bir husus ise söz konusu tabloda bir önceki kolonun üst değeri ile bir sonraki kolonun alt değeri birbirine eşittir. Örneğin;

x için tabusüreyapısı$_9$*'un üst değeri = x için tabusüreyapısı*$_{10}$*' un alt değeri = 80,000,000* dur.

x değişkenine 80,000,000 değeri atandığında hangi kutucukta yeralacağı sorusunun cevabı ise indisi küçük olan dokuzuncu kutucuk olacaktır. Böyle bir durum tesadüfi rakamlar bilgisayarla seçildiğinden ihmal edilebilecek kadar küçük bir olasılıktır. Çünkü 80,000,000.000000001 veya 79,999,999,999999999' un yeri net olarak bellidir. Ayrıca değişkenlerin şu an için alabileceği değerler -100,000,000 ile 100,000,000 arasında gibi görünse de iterasyonlar ilerledikçe bu durumun değişecektir.

Başka bir tablo ise seçilmiş olan değişkenlerin en küçük amaç fonksiyonu değerlerinin saklanması için tutulur. Böylece değişkenlerin hangi kutucuklarda daha küçük amaç fonksiyonu değeri elde ettikleri, başka bir ifadeyle hangi kutucukların daha umut verici olduğu hafızada tutulmaktadır. Tablolardaki hücreler başlangıçta çok yüksek değerlerlerle doldurulur. Bu çok yüksek değer *M* ile gösterilecektir. Buna göre *x* değişkeninin amaç fonksiyon değerleri için tabu yapısı tablo 3' teki gibi olur.

Tablo 3: Birinci değişkenin amaç fonksiyon değerleri için tabu yapısı

Değişken	Alt:-100,000,000 Üst:-80,000,000	Alt:-80,000,000 Üst:-60,000,000	Alt:-60,000,000 Üst:-40,000,000	Alt:-40,000,000 Üst:-20,000,000	Alt:-20,000,000 Üst:0	Alt:0 Üst:20,000,000	Alt:20,000,000 Üst:40,000,000	Alt:40,000,000 Üst:60,000,000	Alt:60,000,000 Üst:80,000,000	Alt:80,000,000 Üst:100,000,000
	1	2	3	4	5	6	7	8	9	10
X	M	M	M	M	M	M	M	M	M	M

$$\left.\begin{array}{l} x = 65312455 \\ y = -12861631 \end{array}\right\} \Rightarrow f(x,y) \cong 5.27 \times 10^{15}$$

olur ve

Eğer *x* için *tabufonksiyonyapısı*$_9$ > 5.27×10^{15} $\Rightarrow$

x için *tabufonksiyonyapısı*$_9$ = 5.27×10^{15}

Eğer *x* için *tabufonksiyonyapısı*$_5$ > 5.27×10^{15} $\Rightarrow$

x için *tabufonksiyonyapısı*$_5$ = 5.27×10^{15}

Başlangıçta tüm kutucuklar *M* gibi çok yüksek bir değere sahip olduğundan her iki kutucuk ta fonksiyon değerinden düşük olacak ve söz konusu kutucukların yeni değerleri 5.27×10^{15} olacaktır.

3. NELDER-MEAD LOKAL KOMŞU ARAMA STRATEJİSİ

Geniş bir tanım aralığı içerisinde tesadüfi olarak atanan değişken değerleriyle hatalı sonuçlar elde etme ihtimali artacağından, bu çalışmada tesadüfi atanan değişken değerleri Nelder-Mead algoritması ile lokal en iyi komşuya doğru yaklaşacak şekilde değişmektedir.

Birden fazla değişkenli fonksiyonların lokal minimumunu bulmak için oldukça basit bir yöntem Nelder ve Mead tarafından geliştirilmiştir. İki değişken için yapı bir üçgen biçimini alır ve metot üçgenin köşelerindeki fonksiyon değerlerini karşılaştırarak arama yapar. Fonksiyon değerinin en büyük olduğu en kötü köşe reddedilir ve başka bir köşeyle yer değiştirilir. Yeni üçgen biçimlendirilir ve tekrar arama devam eder. Köşelerdeki fonksiyon değerleri ve üçgenin alanı gittikçe küçülür [6]. Notasyon aşağıdaki gibidir:

W: En kötü köşe.

G: İyi köşe.

B: En iyi köşe.

$$M = \frac{B+G}{2}$$

$$R = 2M - W$$

$$E = 2R - M$$

$$S = \frac{B+W}{2}$$

Böylece algoritma aşağıdaki gibidir [6]:

EĞER f (R) < f (G) İSE yordam$_1$' i çalıştır *DEĞİLSE yordam*$_2$' yi çalıştır

yordam$_1$
BAŞLA
EĞER f (*B*) < *f*(*R*) *İSE*
W ile R çözümleri yer değişir
DEĞİLSE
E çözümü ve f(*E*) *hesaplanır*
EĞER f (*E*) < *f*(*B*) *İSE*
W ile E çözümleri yer değişir
DEĞİLSE
W ile R çözümleri yer değişir

yordam$_2$
BAŞLA
EĞER f (*R*) < *f*(*W*) *İSE*
W ile R çözümleri yer değişir
EĞERSONU
{*C = (W + M)/2 veya*
C = (M + R)/2 çözümü hesaplanır}
f(*C*) *hesaplanır*
EĞER f(*C*) < *f*(*W*) *İSE*
W ile C çözümleri yer

EĞERSONU *değişir*

EĞERSONU *DEĞİLSE*

BİTİR *S çözümü hesaplanır*

W ile S çözümleri yer

değişir

G ile M çözümleri yer

değişir

EĞERSONU

BİTİR

Bu çalışmada kullanılan algoritma ise çok ufak bir hızlandırıcı farkla aşağıdaki hali almıştır:

BAŞLA

R çözümü ve f(R) hesaplanır

E çözümü ve f(E) hesaplanır

$C_1 = \frac{W+M}{2}$ *çözümü ve f*(C_1) *hesaplanır*

$C_2 = \frac{M+R}{2}$ *çözümü ve f*(C_2) *hesaplanır*

V matrisine sırasıyla R, E, C_1 *ve* C_2 *çözüm vektörlerini herbiri bir satır olacak şekilde ata*

F vektörüne sırasıyla f(R), f(E), $f(C_1)$ *ve* $f(C_2)$ *yi ata*

F vektörü ve V matrisininin satırlarını, F vektöründe karşılık gelen değeri küçükten büyüğe olacak şekilde sırala

EĞER $F_1 < f(W)$ *İSE*

V$_1$ *çözümü (V' nin 1. satırı) ile W çözümü yer değiştirilir*

DEĞİLSE

S çözümü hesaplanır

W ile S çözümleri yer değişir

G ile M çözümleri yer değişir

EĞERSONU

BİTİR

Görüldüğü gibi örnek olarak eğer *f(E) < f(R)* ve *f(R) < f(B)* ise birinci algoritmada *R* ile *W* yer değişirken, ikinci algoritmada *E* ile *W* yer değişecektir. Böylece daha küçük fonksiyon değeri sonuca daha hızlı götürecektir.

Her değişken için tanım aralığı içerisinde yeralacak şekilde üç farklı tesadüfi nokta seçilir. Değişken çiftleri Nelder-Mead metodu için başlangıç üçgenini oluşturmuş olacaktır. Köşelerdeki, başka bir ifadeyle *B, G* ve *W* arasındaki mutlak değerce en büyük fark 0,05' e eşit veya küçükse algoritma durdurulur. Bunu sağlamadığı halde denemeler 5000

iterasyona ulaştığı takdirde yine durdurularak *B* köşesindeki değişken değerleri tabu yapılarında anlatıldığı gibi işlenir. Nelder-Mead metodu yalnız sürekli fonksiyonlarda kullanılabilmektedir. Oysa bu çalışmanın kapsamına süreksiz fonksiyonlar da girebilmektedir. Ancak Nelder-Mead metodu yardımcı ve hızlandırıcı bir yöntem olarak kullanılmıştır. Aşağıda buna bir örnek verilmiştir:

İterasyon sayısı = 5000

B(x=3.0, y=1.8) ve f(B)=-6.96 (1 numarlı fonksiyona göre)

G(2.8125, 2.0375) f(G)=-6.9564

W(3.15, 2.25) f(W)=-6.9525

olur.

x değişkeni için |3.0–2.8125|=0.1875; |3.0–3.15|=0.15; |2.8125–3.15|=0.3375

y değişkeni için |1.8–2.0375|=0.2375; |1.8–2.25|=**0.45**; |2.0375–2.25|=0.2125

En büyük fark 0.45 > 0.05 ancak 5000. iterasyon olduğu için algoritma sonlanarak *B* vektörü her iki tabu yapısında gerekli değişiklerin yapılması için kullanılacaktır:

$$x\ için\ tabusüreyapisi_5 = tabu\quad süresi = 3$$
$$y\ için\ tabusüreyapisi_5 = tabu\quad süresi = 3$$

$$Eger\ x\ için\ tabufonksiyonyapisi_5 > -6.96 \Rightarrow x\ için\ tabufonksiyonyapisi_5 = -6.96$$
$$Eger\ y\ için\ tabufonksiyonyapisi_5 > -6.96 \Rightarrow y\ için\ tabufonksiyonyapisi_5 = -6.96$$

4. TA İLE DOĞRUSAL OLMAYAN OPTİMİZASYON ALGORİTMASI

Algoritma esas olarak iç içe geçmiş iki iterasyondan oluşmaktadır. Birinci iterasyon tabu yapıları için mevcut kutucukların tanım aralıklarının belirlenmesi, ikinci iterasyon ise söz konusu aralıklar içinde yukarıda anlatıldığı biçimde arama yapılması görevine sahiptir. Ana yapı aşağıda verilmiştir.

İterasyon sayısı ve iterasyon azaltma katsayısı (8 numaralı satır) deneye dayalı olarak belirlenmiş olup farklı çalışmalarda farklı değerler kullanılabilir. Dikkat edilecek olursa 2 ile işaretlenmiş satırda *en iyi fonksiyon değeri* değişkenine çok yüksek bir değer atanmıştır. Mevcut fonksiyon değeri ile kıyaslanarak iyileştikçe bu değişkende tutulacaktır.

Üç numaralı satırda her iki tabu yapısı başlangıç durumuna getirilmektedir. Bunun anlamı mevcut tabu süre yapılarındaki tüm kutucuklara sıfır, tabu fonksiyon değerindeki tüm kutucuklara da *M* değerinin konulmasıdır.

İtererasyon sayısı = 15000

En iyi fonksiyon değeri = M (2)

Her iki tabu yapısını başlangıç durumuna getir (3)

BAŞLA

    En iyi fonksiyon değeri değişti = hayır (4)

    ÇALIŞTIR (5)

EĞER En iyi fonksiyon değeri değişti = hayır ise

En iyi kutucukları belirle (6)

Yeni kutucukları belirle (7)

Her iki tabu yapısını başlangıç durumuna getir

İterasyon sayısı = yuvarla(iterasyon sayısı * 0.914) (8)

iterasyon sayısı >= 2716 OLDUĞU SÜRECE TEKRAR ET

Dört numaralı satırda mevcut iterasyon sayısı kadar yapılacak denemede en baştan beri bulunmuş olan en iyi fonksiyon değerinin değişip değişmediğini kontrol etmek üzere bir işaret değişkeni kullanılmaktadır.

Beş numaralı satırda mevcut kutucuklara göre *iterasyon sayısı* değişkenine bağlı olarak daha önce anlatıldığı gibi üçer köşe seçilerek Nelder-Mead algoritmasına gönderilir. Başlangıçta 15000 kez üçer nokta seçilecek, Nelder-Mead algoritmasına gönderilecek ve algoritmanın sonuç değeri tabu yapılarına işlenecektir. Unutulmaması gereken bir husus ise seçilen üçer noktaya *tabu süre yapısının* etkisi olduğudur. Söz konusu alt algoritma aşağıda açıklanmıştır.

BAŞLA

i=1

Tabu süre yapısına dikkat ederek tesadüfî olarak üç başlangıç köşesi

seç

Üç köşenin fonksiyon değerlerine göre B, G ve W çözümlerini

belirle

Nelder-Mead algoritmasını çalıştır bulduğu en iyi fonksiyon değerini

fonksiyondeğeri değişkenine ata

Nelder-Mead algoritmasından geri dönen B çözümü için

tabusüreyapısı ve *tabufonksiyonyapısı* işlenir

EĞER *fonksiyondeğeri* < *Eniyifonksiyondeğeri* İSE

BAŞLA

Eniyifonksiyondeğeri = fonksiyon değeri

Mevcut B çözümünü *en iyi çözüm olarak* sakla

En iyi fonksiyon değeri değişti = evet

BİTİR

i = i + 1

i < iterasyon sayısı OLDUĞU SÜRECE TEKRAR ET

Daha önce verilmiş olan fonksiyon (1) minimize edilmeye çalışıldığında ilk ÇALIŞTIR algoritması sonucunda oluşan tabu süre yapısı tablo 4 ve tablo 5' teki gibi olmuştur.

Tablo 4: İlk ÇALIŞTIR algoritması sonucunda elde edilen x değişkeninin tabu fonksiyon yapısı

Değişken	Alt:-100,000,000 Üst:-80,000,000	Alt:-80,000,000 Üst:-60,000,000	Alt:-60,000,000 Üst:-40,000,000	Alt:-40,000,000 Üst:-20,000,000	Alt:-20,000,000 Üst:0	Alt:0 Üst:20,000,000	Alt:20,000,000 Üst:40,000,000	Alt:40,000,000 Üst:60,000,000	Alt:60,000,000 Üst:80,000,000	Alt:80,000,000 Üst:100,000,000
	1	2	3	4	5	6	7	8	9	10
x	M	M	M	M	-0.245	-6.9999	M	M	M	M

Tablo 5: İlk ÇALIŞTIR algoritması sonucunda elde edilen y değişkeninin tabu fonksiyon yapısı

Değişken	Alt:-100,000,000 Üst:-80,000,000	Alt:-80,000,000 Üst:-60,000,000	Alt:-60,000,000 Üst:-40,000,000	Alt:-40,000,000 Üst:-20,000,000	Alt:-20,000,000 Üst:0	Alt:0 Üst:20,000,000	Alt:20,000,000 Üst:40,000,000	Alt:40,000,000 Üst:60,000,000	Alt:60,000,000 Üst:80,000,000	Alt:80,000,000 Üst:100,000,000
	1	2	3	4	5	6	7	8	9	10
y	M	M	M	M	-3.9999	-6.9999	M	M	M	M

Altı ve yedi numaralı satırlar bir arada incelenmelidir. Öncelikli olarak yeni kutucukların belirlenmesinin ne anlama geldiği açıklanacaktır. Tablo 4' e bakılacak olursa başlangıçta *x* değişkeni için 5 ve 6 numaralı kutucuklarda 15000 iterasyon sonucu bulunan en küçük fonksiyon değerleri sırasıyla -0.245 ve -6.9999' dur. Diğerleri değişmemiş ve *M* olarak kalmıştır. Bunun sebebi ise diğer kutulardan da tesadüfi olarak seçim yapılmış ancak amaç fonksiyonun basit olmasından dolayı, Nelder-Mead metodu sayesinde bunların 5 ve 6. kutucuklara yaklaşmış olmasıdır. Burdan da anlaşılabileceği gibi *x* değişkeni için artık daha hassas bir arama 5 ve 6. kutucuklara yoğunlaşarak olmalıdır. Yeni tanım aralığı belirlenerek ÇALIŞTIR algoritması tekrar koşturulmalıdır. Bunun için en küçük fonksiyon değerine sahip üç kutucuk seçilerek bunların alt ve ust değerlerinin, fonksiyon değerlerine göre ağırlıklı ortalamaları alınır. Ancak fonksiyon değeri daha düşük olana daha yüksek ağırlık verilmelidir. Bunun için herhangi bir değişkene yönelik aşağıdaki formülasyon kullanılmıştır:

$$\partial = min(0, tabufonksiyonyapısı_1, \ldots, tabufonksiyonyapısı_{10}) \qquad (9)$$

*enküçükalt$_i$:*Fonksiyon değeri en küçük i. kutucuğun alabileceği alt değeri.

*enküçüküst$_i$:*Fonksiyon değeri en küçük i. kutucuğun alabileceği üst değeri.

*enküçükkutucukfonksiyon$_i$:*Fonksiyon değeri en küçük i. kutucuktaki mevcut fonksiyon değeri

$$alt^* = \frac{\sum_{i=1}^{3} enkucukkutucukalt_i \times \frac{1}{enkucukkutucukfonksiyon_i - \partial + 1}}{\sum_{i=1}^{3} \frac{1}{enkucukkutucukfonksiyon_i - \partial + 1}}$$

$$ust^* = \frac{\sum_{i=1}^{3} enkucukkutucukust_i \times \frac{1}{enkucukkutucukfonksiyon_i - \partial + 1}}{\sum_{i=1}^{3} \frac{1}{enkucukkutucukfonksiyon_i - \partial + 1}}$$

$\theta = (ust^* - alt^*) \times 0.75$ (10)

$alt = alt^* - \theta$

$ust = ust^* + \theta$

Söz konusu formülasyonu tablo 4' te yeralan *x* değişkeni için kullanılacak olursa öncelikli olarak en küçük fonksiyon değerine sahip üç kutucuk seçilecektir. 4 ve 5. kutucuklar dışındakiler birbirine eşit olduğu için ilk olan kutucuk seçilir. Böylece;

$\partial = min\ (0, M, -0.245, -6.9999) = -6.9999$

$\frac{1}{M + 6.9999 + 1} \cong 0$ (M çok büyük bir sayı olduğu için)

$$alt^* = \frac{(0 \times 1) + (-20{,}000{,}000 \times 0.129) + (-100{,}000{,}000 \times 0)}{1 + 0.129 + 0} = -2{,}285{,}208.1488$$

$$ust^* = \frac{(20{,}000{,}000 \times 1) + (0 \times 0.129) + (-80{,}000{,}000 \times 0)}{1 + 0.129 + 0} = 17{,}714{,}791.8512$$

$\theta = (17{,}714{,}791.8512 + 2{,}285{,}208.1488) \times 0.75 = 15{,}000{,}000$

$alt = -2{,}285{,}208.1488 - 15{,}000{,}000 = -17{,}285{,}208.1488$

ust = 17,714,791.8512 + 15,000,000 = 32,714,791.8512

olur. Görüldüğü gibi başlangıçta -100,000,000 ile 100,000,000 arasında olan *x* değişkeninin tanım aralığı artık -17,285,208.1488 ile 32,714,791.8512 arasına inmiştir. Aynı işlemler *y* değişkeni için de yapıldığında *y* değişkeninin yeni tanım aralığı

belirlenecektir. Buna göre *x* değişkeni için yeni amaç fonksiyonu tabu yapısı tablo 6' daki gibi olur. Artık aralık daraldığı için deneme sayısı azaltılmış ve 15000 × 0.914 = 13710 olmuştur.

Tablo 6: İkinci ÇALIŞTIR algoritması başlarken x değişkeninin tabu fonksiyon yapısı

Değişken	Alt:-17,285,208.1488 Üst:-12,285,208.1488	Alt:-12,285,208.1488 Üst:-7,285,208.1488	Alt: -7,285,208.1488 Üst:-2,285,208.1488	Alt: -2,285,208.1488 Üst:2,714,791.8512	Alt: 2,714,791.8512 Üst:7,714,791.8512	Alt: 7,714,791.8512 Üst:12,714,791.8512	Alt:12,714,791.8512 Üst:17,714,791.8512	Alt: 17,714,791.8512 Üst:22,714,791.8512	Alt:22,714,791.8512 Üst:27,714,791.8512	Alt:27,714,791.8512 Üst:32,714,791.8512
	1	2	3	4	5	6	7	8	9	10
x	M	M	M	M	M	M	M	M	M	M

Ağırlıklı ortalamalar fonksiyon değerine göre alınmaktadır. Ancak en küçük değere en büyük ağırlığın verilebilmesi için *1 / fonksiyon değeri* biçiminde formüle edilmiştir. Bununla beraber 0' dan küçük fonksiyon değerleri olduğunda ağırlıklandırma yanlış neticelere neden olacaktır. Böylelikle 9 numaralı eşitlikte, en küçük fonksiyon değeri 0' dan küçükse ∂ değişkenine aktarılmaktadır. Her bir fonksiyon değerinden çıkartıldığında fonksiyon değerlerinin hepsi pozitif olmuş olur. Sıfıra bölme hatasına engel olmak için bölene bir eklenmiştir.

Denemelere dayalı 0.75 olarak belirlenmiş bir genişleme katsayısıyla tanım aralığı 10 numaralı eşitlik sayesinde her iki yöne doğru bir miktar genişlemektedir. Böylece başlangıçta tanımlanmış olan aralığın dışına yönelten bir aramaya izin verilmiş olacaktır. Örneğin yukarıdaki örnekte optimum değer *x < -100,000,000* veya *x > 100,000,000* noktalarında yer alıyorsa, birinci ÇALIŞTIR algoritması neticesinde 1. veya 10. kutucuklar daha düşük fonksiyon değerine sahip olacak böylelikle 100,000,000' u aşan veya -100,000,000' un altında kalan bir tanım aralığı ikinci ÇALIŞTIR algoritmasının başlangıcında belirlenmiş olacaktır. Ayrıca daha fazla komşuya yer vermek daha sağlıklı sonuçlar elde etmeyi sağlamaktadır.

Daha önceki ÇALIŞTIR algoritmalarının herhangi birinde elde edilmiş en iyi değerden daha iyisinin mevcut ÇALIŞTIR algoritmasında elde edilemediği durumlarda, yukarıda bahsi geçen üç en iyi kutucuk seçimi ikiye indirilmiştir. O ana kadar elde edilmiş olan en iyi değer hangi kutucuğa denk geliyorsa o kutucuk ta söz konusu ağırlıklandırmaya dahil edilmiştir. 6 numaralı satırda bu işlem gerçekleştirilmiştir.

4.1. Örnek

Algoritmanın test edilmesi adına örnek bir doğrusal olmayan model seçilmiş ve optimize edilmeye çalışılmıştır. Model aşağıdaki gibidir:

$$\min \quad f(x_1, x_2, x_3, x_4, x_5, x_6, x_7) = x_1^2 + 3x_2^2 + x_3^2 + x_4^2 + 2x_5^2 + 2x_6^2 + x_7^2 - 2x_1x_2 - x_3 - x_4x_5 - 2x_2x_3 - x_4x_6 - x_7$$

$$x_1 \times x_2 \times x_3 \times x_4 \times x_5 \times x_6 \times x_7 \geq 78125$$

Kısıt amaç fonksiyonu içine aşağıdaki biçimde dahil edilmiştir:

$$\min \quad x_1^2 + 3x_2^2 + x_3^2 + x_4^2 + 2x_5^2 + 2x_6^2 + x_7^2 - 2x_1x_2 - x_3 - x_4x_5 - 2x_2x_3 - x_4x_6 - x_7 + \max(78125 - x_1 \times x_2 \times x_3 \times x_4 \times x_5 \times x_6 \times x_7, 0) \times 9 \times 10^5$$

Tablo 7: İlk ÇALIŞTIR algoritması sonucu tüm değişkenlerin tabu fonksiyon yapıları

Değişken	Alt:-100,000,000 Üst:-80,000,000	Alt:-80,000,000 Üst:-60,000,000	Alt:-60,000,000 Üst:-40,000,000	Alt:-40,000,000 Üst:-20,000,000	Alt:-20,000,000 Üst:0	Alt:0 Üst:20,000,000	Alt:20,000,000 Üst:40,000,000	Alt:40,000,000 Üst:60,000,000	Alt:60,000,000 Üst:80,000,000	Alt:80,000,000 Üst:100,000,000
	1	2	3	4	5	6	7	8	9	10
x_1	4.7×10^{15}	2.4×10^{15}	1.1×10^{15}	3.9×10^{14}	9.6×10^{11}	2.4×10^{12}	2.1×10^{14}	1.0×10^{15}	2.1×10^{15}	3.4×10^{15}
x_2	1.1×10^{16}	4.9×10^{15}	2.7×10^{15}	6.9×10^{14}	9.6×10^{11}	2.4×10^{12}	5.7×10^{14}	2.1×10^{15}	4.4×10^{15}	8.9×10^{15}
x_3	6.4×10^{15}	2.5×10^{15}	1.3×10^{15}	4.2×10^{14}	9.6×10^{11}	2.8×10^{12}	2.4×10^{14}	9.4×10^{14}	2.4×10^{15}	4.2×10^{15}
x_4	5.6×10^{15}	4.0×10^{15}	1.7×10^{15}	4.4×10^{14}	9.6×10^{11}	2.8×10^{12}	4.3×10^{14}	1.5×10^{15}	3.1×10^{15}	5.4×10^{15}
x_5	1.8×10^{16}	7.5×10^{15}	3.8×10^{15}	1.2×10^{15}	9.6×10^{11}	1.1×10^{13}	9.1×10^{14}	3.9×10^{15}	7.6×10^{15}	1.2×10^{16}
x_6	1.5×10^{16}	6.6×10^{15}	3.4×10^{15}	1.1×10^{15}	2.8×10^{12}	9.6×10^{11}	1.1×10^{15}	3.4×10^{15}	7.0×10^{15}	1.3×10^{16}
x_7	1.0×10^{16}	4.7×10^{15}	2.0×10^{15}	5.9×10^{14}	9.6×10^{11}	2.8×10^{12}	5.7×10^{14}	2.0×10^{15}	4.2×10^{15}	7.6×10^{15}

Tablo 8: Beşinci ÇALIŞTIR algoritması sonucu x_1 değişkeninin tabu fonksiyon yapısı

Değişken	Alt:-398,245.5502	Alt:-320,120.5502	Alt:-241,995.5502	Alt:-163,870.5502	Alt:-85,745.5502	Alt:-7,620.5502	Alt:70,504.4497	Alt:148,629.4497	Alt:226,754.4497	Alt:304,879.4497
	1	2	3	4	5	6	7	8	9	10
x_1	8.3×10^{10}	3.9×10^{10}	2.8×10^{10}	7.1×10^{9}	3.5×10^{9}	1.7×10^{9}	7.3×10^{9}	2.2×10^{10}	4.3×10^{10}	5.9×10^{10}

Tablo 9: Beşinci ÇALIŞTIR algoritması sonucu x_2 değişkeninin tabu fonksiyon yapısı

Değişken	Alt:-475,627.6451	Alt:-397,502.6451	Alt:-319,377.6451	Alt:-241,252.6451	Alt:-163,127.6451	Alt:-85,002.6451	Alt:-6,877.6451	Alt:71,247.3548	Alt:149,372.3548	Alt:227,497.3548
	1	2	3	4	5	6	7	8	9	10
x_2	2.7×10^{11}	1.6×10^{11}	9.0×10^{10}	4.5×10^{10}	1.2×10^{10}	3.5×10^{9}	1.7×10^{9}	1.3×10^{10}	3.3×10^{10}	7.0×10^{10}

Tablo 10: Beşinci ÇALIŞTIR algoritması sonucu x_3 değişkeninin tabu fonksiyon yapısı

Değişken	Alt:-465,265.1753	Alt:-387,140.1753	Alt:-309,015.1753	Alt:-230,890.1753	Alt:-152,765.1753	Alt:-74,640.1753	Alt:3,484.8246	Alt:81,609.8246	Alt:159,734.8246	Alt:237,859.8246
	1	2	3	4	5	6	7	8	9	10
x_3	1.5×10^{11}	6.6×10^{10}	3.2×10^{10}	2.0×10^{10}	5.5×10^{9}	3.5×10^{9}	1.7×10^{9}	6.3×10^{9}	2.3×10^{10}	5.0×10^{10}

Tablo 11: Beşinci ÇALIŞTIR algoritması sonucu x_4 değişkeninin tabu fonksiyon yapısı

Değişken	Alt:-486,623.7520	Alt:-408,498.7520	Alt:-330,373.7520	Alt:-252,248.7520	Alt:-174,123.7520	Alt:-95,998.7520	Alt:-17,873.7520	Alt:60,251.2479	Alt:138,376.2479	Alt: 216,501.2479
	1	2	3	4	5	6	7	8	9	10
x_4	1.7×10^{11}	9.5×10^{10}	5.2×10^{10}	3.3×10^{10}	1.4×10^{10}	6.6×10^{9}	1.7×10^{9}	7.0×10^{9}	2.3×10^{10}	5.7×10^{10}

Tablo 12: Beşinci ÇALIŞTIR algoritması sonucu x_5 değişkeninin tabu fonksiyon yapısı

Değişken	Alt:-459,492.7119	Alt:-381,367.7119	Alt: -303,242.7119	Alt:-225,117.7119	Alt:-146,992.7119	Alt:-68,867.7119	Alt:9,257.2880	Alt:87,382.2880	Alt:165,507.2880	Alt:243,632.2880
	1	2	3	4	5	6	7	8	9	10
x_5	5.4×10^{11}	2.0×10^{11}	1.0×10^{11}	5.8×10^{10}	1.4×10^{10}	1.7×10^{9}	7.3×10^{9}	2.8×10^{10}	6.3×10^{10}	1.2×10^{11}

Tablo 13: Beşinci ÇALIŞTIR algoritması sonucu x_6 değişkeninin tabu fonksiyon yapısı

Değişken	Alt:-303,818.8697	Alt:-225,693.8697	Alt:-147,568.8697	Alt:-69,443.8697	Alt:8,681.1302	Alt: 86,806.1302	Alt:164,931.1302	Alt: 243,056.1302	Alt:321,181.1302	Alt:399,306.1302
	1	2	3	4	5	6	7	8	9	10
x_6	1.3×10^{11}	5.4×10^{10}	1.7×10^{10}	1.7×10^{9}	4.9×10^{9}	2.5×10^{10}	6.1×10^{10}	1.2×10^{11}	2.0×10^{11}	3.0×10^{11}

Tablo 14: Beşinci ÇALIŞTIR algoritması sonucu x_7 değişkeninin tabu fonksiyon yapısı

Değişken	Alt:-276,166.4455	Alt:-198,041.4455	Alt:-119,916.4455	Alt:-41,791.4455	Alt:36,333.5544	Alt:114,458.5544	Alt:192,583.5544	Alt:270,708.5544	Alt:348,833.5544	Alt:426,958.5544
	1	2	3	4	5	6	7	8	9	10
x_7	6.2×10^{10}	2.4×10^{10}	3.5×10^{9}	1.7×10^{9}	5.8×10^{9}	2.3×10^{10}	5.2×10^{10}	8.7×10^{10}	1.4×10^{11}	2.1×10^{11}

ÇALIŞTIR algoritması 19 defa koşturulduğunda tüm arama sona erecektir ($15000\times 0.914^{20} < 2716$). ÇALIŞTIR algoritmasının her sonlanışında mevcut tanım aralığının $\frac{2.5}{10}$'u bir sonrakine gönderilmektedir. Buna göre en son ÇALIŞTIR algoritmasının değişkenleri aradığı aralığın yukarıdaki örnek için genişliği 200,000,000 × $\left(\frac{2.5}{10}\right)^{18}$ =0.00291 olacaktır. Sonuç değerler aşağıdaki gibidir:

$f(x_1, x_2, x_3, x_4, x_5, x_6, x_7)$ = 81.1492037056704

x_1 = 7.505881883026328 x_2 = 5.849254549323604

x_3 = 7.918629682859468 x_4 = 5.525423303436403

x_5 = 3.277608595558137 x_6 = 3.2781105029890725

x_7 = 3.785223627768666

SONUÇ

Yukarda verilen örnek problem MS Office 2003 Solver' da çözülmeye çalışılmıştır. Başlangıç noktaları bu örnekteki gibi -100,000,000 ile 100,000,000 arasında tesadüfi olarak seçildiğinde aşağıdaki sonucu bulmuştur:

$f(x_1, x_2, x_3, x_4, x_5, x_6, x_7)$ = 96,0973397646863

x_1 = -7,44416533014627 x_2 = -5,62991507109314

x_3 = -7,04674526752675 x_4 = -5,76103961351767

x_5 = -3,41650158141594 x_6 = -3,41650992088881

x_7 = 3,93385574924436

Yukarıdaki değerler başlangıç noktaları olarak kabul edilerek tekrar çözülmeye çalışıldığında, solver daha iyiye götürememiş burada kalmıştır. Bu çalışmada kullanılan algoritmayla bulunan x_i değişkeni değerlerine yakın başlangıç noktaları seçildiğinde ise algoritmanın bulduğu sonuçla aynı sonucu bulmuştur.

Literatürde bir çok araştırmacı tarafından test amaçlı kullanılan ve Rosenbrock Muz Fonksiyonu (Rosenbrock's Banana Function) olarak bilinen aşağıdaki fonksiyonun minimum değerli noktası $x = 1$, $y = 1$ ve $f(x, y) = 0$' dır.

$$min\ f(x, y) = 100(x - y^2)^2 + (1 - x)^2$$

$$0 \le x, y \le 6.$$

Fonksiyonun tanım aralığı [0, 6] olarak verildiği için başlangıç noktaları bu aralıktan tesadüfi olarak seçildiğinde, TA temelli algoritma ile aşağıdaki sonuç bulunmuştur:

$f(x, y) = 1.7064047456062012 \times 10^{-25}$

$x = 1.0000000000000955$

$y = 1.0000000000001508$

Bir başka fonksiyon ise minimum değerli noktası $x = 3$, $y = 3$ ve $f(x, y) = 0$ olan aşağıdaki fonksiyondur:

$$min\ f(x,y) = \frac{(x-3)^8}{1+(x-3)^8} + \frac{(y-3)^4}{1+(y-3)^4}$$

TA temelli algoritma kullanılarak başlangıç noktaları -100000000 ile 100000000 arasında tesadüfi olarak seçilmiş ve aşağıdaki sonuç bulunmuştur:

$f(x, y) = 1.3669227486301094 \times 10^{-36}$

$x = 3.0000049654961534$

$y = 3.00000000090924$

Görüldüğü gibi bu çalışmada kullanılan TA algoritmasının anlamlı sonuçlar verdiği görülmektedir.

Kaynaklar

[1] Hedar, A., R., Fukushima, M., Tabu Search directed by direct search methods for nonlinear global optimization, *European Journal of Operational Research*, 170, s: 329-349, (2006).

[2] Pukkala, T., Heinonen, T., Optimizing heuristic search in forest planning, *Nonlinear Analysis: Real World Applications*, 7, s: 1284-1297, (2006).

[3] Reeves, C., *Modern Heuristic Techniques for Combinatorial Problems*, McGraw-Hill Book C., London, 1995

[4] Battiti, R., Tecchiolli, G., The continuous reactive tabu search: Blending combinatorial optimization and stochastic search for global optimization, *Annals of Operations Research*, 63, s: 153-188, (1996)

[5] Duarte, A., Marti, R., Tabu search and GRASP for the maximum diversity problem, *European Journal of Operational Research*, 178, s:71-84, (2007)

[6] Mathews, J., H., Fink, K., K., , *Numerical Methods Using Matlab*, Prentice-Hall Inc., New Jersey, 2004.