KARINCA KOLONİSİ OPTİMİZASYONU TABANLI META-SEZGİSEL METOT VE UYGULAMASI

Özgür BAŞKAN [1] Soner HALDENBİLEN[2] Hüseyin CEYLAN[3] Halim CEYLAN[4]

[1,2,3,4]İnşaat Mühendisliği Bölümü
Mühendislik Fakültesi
Pamukkale Üniversitesi, Kınıklı, DENİZLİ
Email: obaskan@pau.edu.tr shaldenbilen@pau.edu.tr hceylan@pau.edu.tr halimc@pau.edu.tr

Özet

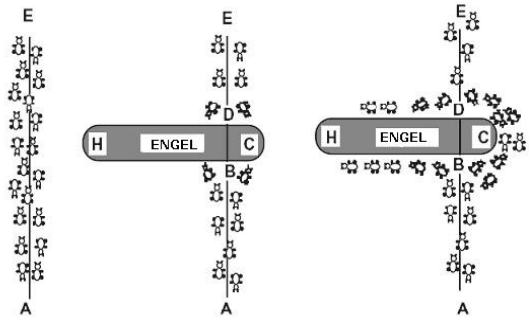
Karınca Kolonisi Optimizasyonu (KKO) 1990'lı yılların başında uygulama alanı bulan meta-sezgisel bir metottur. Bu çalışmada KKO tabanlı sezgisel bir metot geliştirilmiş ve algoritma Microsoft Excel programı altında Visual Basic (VBA) diliyle kodlanmıştır. Geliştirilen algoritma, global minimum ve optimumun bulunması için literatürdeki birçok test fonksiyonu üzerinde denenmiş ve literatürdeki diğer çalışmalarla karşılaştırılmıştır. Sonuç olarak geliştirilen KKO tabanlı sezgisel optimizasyon metodu ile oldukça başarılı sonuçlar alınmıştır.

1. Giriş

Karınca Kolonisi Optimizasyonu (KKO) algoritmaları, optimizasyon problemlerinin çözümü için gerçek karınca kolonilerinin hareket davranışlarının gözlemlenmesi ile ortaya atılmıştır. Karınca kolonileri yuva ve yiyecek kaynağı arasında her zaman en kısa yolu bulabilmektedirler. Karınca kolonilerinin yiyecek ve yuva arasında sergilemiş olduğu bu çözüm sürecinin optimizasyon problemlerinin çözümünde kullanılabileceği ilk defa 1992 yılında ortaya atılmıştır [1]. KKO yöntemi son zamanlarda literatürde birçok optimizasyon probleminin çözümü için kullanılmaktadır [2-3]. $F(x)$ fonksiyonu göz önüne alındığında bütün x değerleri için eğer $F(x_{min}) \leq F(x)$ ise bu durumda x_{min} noktası fonksiyonun minimum noktası olur. Tanımlanan fonksiyon birçok yerel minimuma sahip olabileceği gibi fonksiyonun global minimum noktası bunlardan bir tanesidir. Global minimumun bulunmasında stokastik metotların birçok avantajı vardır. Konveks ya da sürekli olmayan yapıdaki fonksiyonların global minimum noktalarının bulunabilmesi için şimdiye kadar birçok sezgisel metot geliştirilmiştir [4-5-6]. Bu çalışmada önerilen KKO tabanlı sezgisel metot her bir iterasyonda çözüm uzayının en iyi çözümü veren parametreler etrafında belli kısıtlar içerisinde yeniden oluşturulması ve sonraki iterasyonlara buna bağlı olarak devam edilmesi prensibine dayanmaktadır. Bildirinin 2. bölümünde KKO ve önerilen algoritma hakkında kısa bir bilgi, 3. bölümde algoritmanın test fonksiyonları üzerindeki performansı ardından sonuçlar ve öneriler verilmiştir.

2. Karınca Kolonisi Optimizasyonu

KKO son zamanlarda çözümü zor optimizasyon problemlerinin çözümünde kullanılan metasezgisel bir yaklaşımdır [7]. İlk çalışmalarda KKO algoritması karınca sistemi olarak önerilmiş ve gezgin satıcı problemi üzerine uygulanmıştır. KKO algoritmaları, optimizasyon problemlerinin çözümü için gerçek karıncaların yiyecek bulma davranışlarının gözlemlenmesi ile ortaya çıkmıştır [8]. Şekil 1'de görüldüğü gibi gerçek karıncalar yiyeceğe giden yolları üzerine bir engel koyulduğu zaman iki yoldan bir tanesini tercih edeceklerdir. Şekil 1(a)'da gösterildiği AE yolu üzerindeki karınca kolonisi yolu üzerinde Şekil 1(b)'deki gibi bir engel olduğu zaman karıncalar engel etrafından dönebilmek için HB ve BC yollarından bir tanesini tercih edeceklerdir.



(a) (b) (c)

Şekil 1: Gerçek karınca davranışları

Tekniğin en temel unsurlarından biri haberleşme aracı olarak kullanılan ve problemlerde çözümün kalitesini gösteren gerçek karıncaların geçtikleri yollara bıraktıkları feromen kimyasalıdır. Feromen kimyasalı karıncalar tarafından güncellenmekte ve

bir bilgiyi temsil etmektedirler. Bir yolda feromen izinin yoğun olması o yolun tercih edilme olasılığını artırır. Karınca kolonisi ilk olarak deterministik düşünceye göre eşit olasılıkta seçim yapacak -stokastik düşünceye göre mutlaka bir yol diğerinden daha tercih edilebilir durumdadır- ve kısa olan yolu tercih eden karıncalar yiyeceğe ulaşıp daha kısa zamanda yuvalarına geri döneceklerdir. Bu süreç sırasında karıncalar geçtikleri yerlere feromen denen kimyasal maddeyi bırakacaklar ve kısa olan yolda az bir zaman sonra daha fazla feromen birikmeye başlayacaktır (Şekil 1.c). Karıncalar bir sonraki turlarında artık feromenin fazla olduğu kısa olan yolu tercih etmeye başlayacaklar ve bir süre sonra karınca kolonisinin tamamı yiyeceğe ulaşmak için kısa olan yolu tercih edecektir. Karıncaların bu davranış kalıplarının incelenmesi ile bu sistemin özellikle en kısa yol problemleri olmak üzere pek çok optimizasyon problemlerinde kullanılabileceği ortaya atılmıştır [1],[9].

2.1. KKO tabanlı Meta-Sezgisel Algoritma

Önerilen algoritmada *k* adet karınca çözüm uzayı içinden rastgele olarak seçilen *k* adet vektör olarak nitelendirilmiştir. Algoritmada her bir iterasyon sonunda elde edilen en iyi fonksiyon değerini veren parametreler etrafında feromen güncellenmesi Denklem (1) yardımıyla yapılır.

$$\tau_t = \tau_{t-1} + (0.01 * f(x_{t-1}^{eniyi})) \quad (1)$$

Burada τ_t feromen miktarı olup gerçek karınca davranışlarındaki iyi çözümü temsil edebilmek için bir önceki iterasyondaki en iyi parametreler etrafında yoğunlaştırılır. Ayrıca karıncaların her bir iterasyonda hareket yönünü tayin edebilmesi için Denklem (2) kullanılır.

$$x_t^{\,k} = x_{t-1}^{eniyi} \pm \alpha \quad (2)$$

Burada $x_t^{\,k}$ *t*. iterasyondaki yeni üretilen karınca vektörü, $x_{t-1}^{\,eniyi}$ bir önceki iterasyondaki en iyi karınca vektörü, α ise başlangıç olarak rastgele seçilen sıçrama uzunluğudur. Rastgele seçilen sıçrama uzunluğunu, iterasyonlar boyunca global minimum noktasını geçmemesi için her bir iterasyonda belli bir miktarda azaltılır (Şekil 2). Denklem (2)'de (±) işaretinden hangisinin kullanılacağı yani hareket yönü ara *f*(*x*) değerini veren yeni karınca vektörünün Denklem (3) yardımıyla hesaplanmasıyla belirlenir.

$$\bar{x}_{baş}^{\,eniyi} = x_{baş}^{\,eniyi} + (x_{baş}^{\,eniyi} * 0.01) \quad (3)$$

Eğer $f(\bar{x}_{baş}^{\,eniyi}) \le f(x_{baş}^{\,eniyi})$ ise Denklem (2)'de (+) işareti tersi durumda ise (–) işareti kullanılır. Önerilen algoritmada ayrıca gerçek karınca davranışlarındaki feromen buharlaşmasını temsil eden yaklaşım kullanılmasına gerek duyulmadan oldukça iyi sonuçlar alınmıştır. Algoritmada ilk olarak rastgele seçilen karınca kolonisini temsil eden vektörler, sonraki iterasyonda en iyi çözüm değerini veren parametrelere ve β ile belirtilen katsayı ile bağlantılı olarak sınırlanır. β değeri algoritmanın performansında oldukça etkili olup, fonksiyon tipine ve çözüm uzayının büyüklüğüne göre farklı değerleri alabilmektedir. İlk olarak çözüm uzayının büyüklüğüyle orantılı olarak seçilen β değeri sonraki iterasyonlarda belli bir değerde azaltılmakta ve iterasyon süreci boyunca çözüm uzayının sınırlandırılması ve yakınsama yeteneğinin böylece artırılması sağlanmaktadır. Şekil 2'de önerilen algoritmanın adımları görülmektedir.

Başlangıç

FOR i=1 to I (I=iterasyon sayısı)

Rastgele başlangıç değerlerini üret

IF I=2 THEN başlangıç değerlerini sınırla ($x_{t-1}^{\,eniyi} + \beta; x_{t-1}^{\,eniyi} - \beta$)

En iyi $f(x)$ *değerini hesapla*

En iyi $f(x)$ *değerini veren x değerini saklı tut*

Feromen Güncellenmesi

Denklem(1) ile feromen güncellenmesi

Çözüm Evresi

Denklem(2) ile hareket yönünün belirlenmesi

FOR i=1 to k

α *vektörünün üretilmesi*

Her bir karınca için yeni x değerlerini hesapla

Yeni $f(x)$ *değerini hesapla*

IF $f(x_t^{\,eniyi}) \le f(x_{t-1}^{\,eniyi})$ *THEN*

$x^{global\min} = x_t^{\,eniyi}$ *ELSE*

$x^{global\min} = x_{t-1}^{\,eniyi}$

END

$\beta_t = \beta_{t-1} * 0.99$

$\alpha_t = \alpha_{t-1} * 0.99$

END

Şekil 2: KKO tabanlı sezgisel algoritma

Bu bölümde KKO tabanlı sezgisel algoritma hakkında bilgi verilmiş ve algoritma adımları açıklanmıştır. Üçüncü bölümde önerilen algoritmanın test fonksiyonları üzerindeki performansı ve karşılaştırmalar verilecektir.

3. Test Fonksiyonları

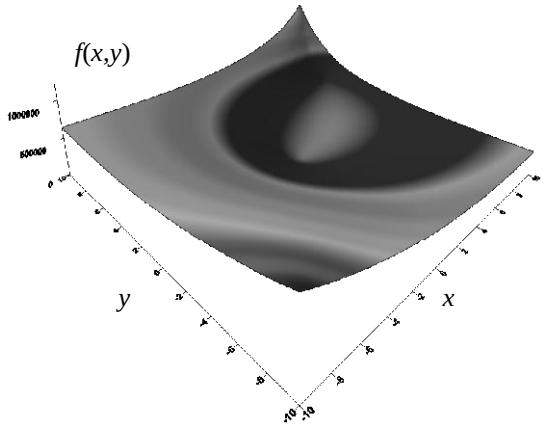
KKO tabanlı sezgisel algoritma 5 farklı test fonksiyonu üzerinde uygulanmış ve algoritmanın performansı karşılaştırmalı olarak verilmiştir. Koloni büyüklüğü bütün problemler için k=20 seçilmiştir.

3.1. Test Problem 1

İki değişkenli test fonksiyonu Denklem 4'de verilmiştir.

$$f(x,y) = (100*(x-y^2)^2) + (1-x)^2 \quad (4)$$

Fonksiyonun minimum noktası x=1 ve y=1 için $f(x,y)$=0 olmaktadır. Başlangıç değerleri β=2, α=1/rastgele(10) şeklindedir. Şekil 3'de fonksiyonun amaç fonksiyonunun değişimi görülmektedir.



Şekil 3: Test problem 1 amaç fonksiyonu değişimi

Tablo 1'de KKO tabanlı sezgisel algoritma ile alınan sonuçlar ve karşılaştırmalar verilmiştir.

Tablo 1: Test problem 1 için önerilen algoritma ile diğer yöntemlerin karşılaştırılması

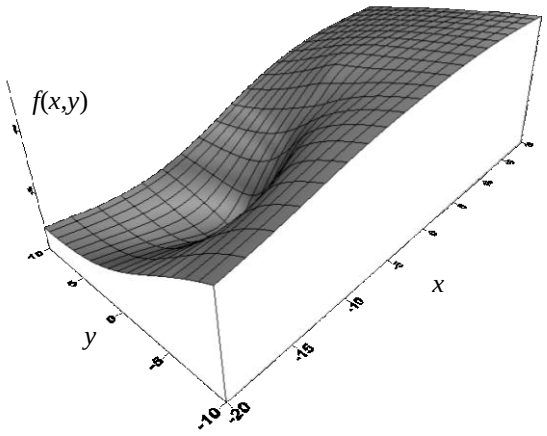
Algoritma	***x***	***y***	***f*(*x*,*y*)**	**İterasyon sayısı**
[4]	1	1	4.02E-016	50000
[5]	1	1	0	50000
[6]	1	1	0	3600
KKO tabanlı algoritma	1	1	0	3418

3.2. Test Problem 2

Problem 2 olarak seçilen fonksiyon 2 değişkenli olup fonksiyonun minimum noktası [-10,10] aralığında x= -10 ve y=0 ve $f(x,y)$= -10 olarak tanımlanmıştır. Tanımlanan fonksiyon Denklem (5)'de verilmiştir. Amaç fonksiyonunun değişimi Şekil 4'de görülmektedir.

$$f(x,y) = \frac{x}{1+|y|} \quad (5)$$

Algoritma için başlangıç değerleri β=5, α=1/rastgele(10) şeklindedir. Tablo 2'de çözüm sonuçları ve karşılaştırmalar verilmiştir.



Şekil 4: Test problem 2 amaç fonksiyonu değişimi

Tablo 2: Test problem 2 için önerilen algoritma ile diğer yöntemlerin karşılaştırılması

Algoritma	***x***	***y***	***f*(*x*,*y*)**	**İterasyon sayısı**
[4]	-10	6.67E-008	-10	50000
[5]	-10	8.07E-011	-10	50000
[6]	-10	0	-10	3750
KKO tabanlı algoritma	-10	0	-10	2787

3.3. Test Problem 3

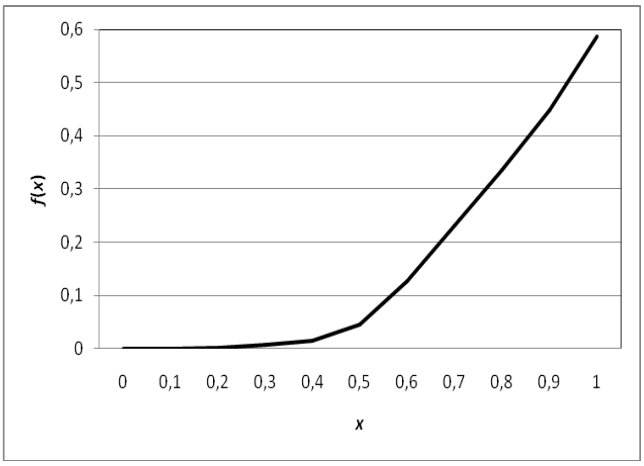
Karşılaştırma problemi olarak seçilen bir diğer fonksiyon Denklem(6)'da verilmiştir. Fonksiyon tek değişkenli olup x=0 noktasında $f(x)$=0 değerini almaktadır. Algoritma için başlangıç değerleri β=2, α=1/rastgele(6) şeklindedir. Tablo 3'de çözüm sonuçları ve karşılaştırmalar verilmiştir.

$$f(x)=\left[x*\sin(\frac{1}{x})\right]^4+\left[x*\cos(\frac{1}{x})\right]^4 \qquad (6)$$

Tablo 3: Test problem 3 için önerilen algoritma ile diğer yöntemlerin karşılaştırılması

Algoritma	*x*	*f*(*x*)	İterasyon sayısı
[4]	-2.53E-011	2.21E-043	50000
[5]	7.79E-012	1.40E-045	50000
[6]	9.92E-012	5.60E-045	5000
KKO tabanlı algoritma	-6.16E-020	9.02E-078	4198

Şekil 5'de test problem 3 olarak verilen denklemin amaç fonksiyonunun değişimi görülmektedir.



Şekil 5: Test problem 3 amaç fonksiyonu değişimi

3.4. Test Problem 4

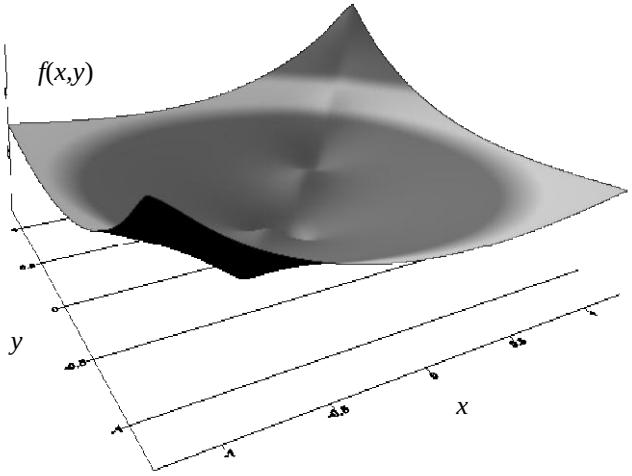
Önerilen KKO tabanlı algoritma için seçilen 4. test problemi Denklem (7)'de verilmiştir. Fonksiyon 2 değişkenli olup minimum noktası x=0 ve y=0 noktasında $f(x,y)$=0 olmaktadır. Algoritma için başlangıç değerleri β=4, α=1/rastgele(6) şeklindedir.

$$x^2+2y^2-0.3\cos(3\pi x)-0.4\cos(4\pi y)+0.7 \qquad (7)$$

Tablo 4'de çözüm sonuçları ve karşılaştırmalar, Şekil 6'da ise amaç fonksiyonunun değişimi gösterilmiştir.

Tablo 4: Test problem 4 için önerilen algoritma ile diğer yöntemlerin karşılaştırılması

Algoritma	*x*	*y*	*f*(*x*,*y*)	İterasyon sayısı
[6]	0	0	0	3750
[10]			2.98E-08	4000
KKO tabanlı algoritma	0	0	0	1832



Şekil 6: Test problem 4 amaç fonksiyonu değişimi

3.5. Test Problem 5

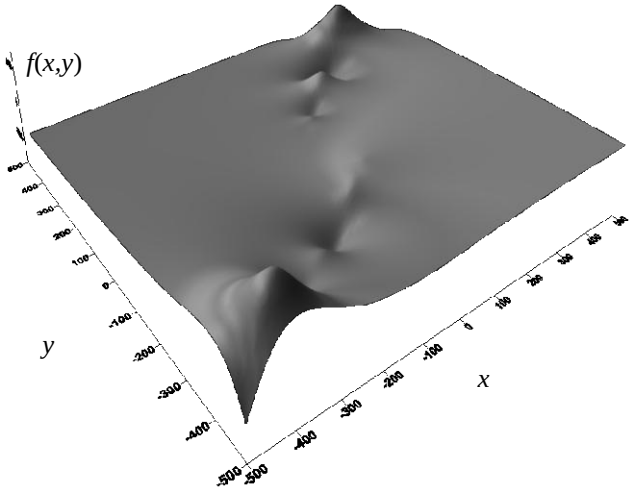
Karşılaştırma problemi olarak son test fonksiyonu Denklem (8)'de verilmiştir. Schwefel test fonksiyonu olarak literatürde bilinen fonksiyon birçok yerel minimuma sahip olması açısından global minimum noktasının bulunması oldukça zor olan bir fonksiyondur. Önerilen algoritma ile bu fonksiyon içinde kesin sonuç alınmıştır. Fonksiyonun -500 ≤ x_i ≤ 500 aralığında minimum noktası $\min f(x)=-n*418.9829$ olup bu değeri veren değer $x_i=420.9687$ olarak literatürde verilmiştir [7].

$$f(x)=\sum_{i=1}^{n}-x_i*\sin\sqrt{|x_i|} \qquad (8)$$

Tablo 5'de test problem 5 için çözüm sonuçları ve Şekil 7'de ise amaç fonksiyonunun değişimi verilmiştir. Algoritma için başlangıç değerleri β=250, α=1/rastgele(10) şeklindedir. n=2 değeri için çözüm yapılmıştır.

Tablo 5: Test problem 5 KKO tabanlı algoritma sonuçları

Algoritma	x_i	$f(x)$	**İterasyon sayısı**
KKO tabanlı algoritma	420.9687	-837.9658	1176



Şekil 7: Test problem 5 amaç fonksiyonu değişimi

4. Sonuçlar ve Öneriler

Bu çalışmada global minimum ve optimumun bulunabilmesi için KKO tabanlı meta sezgisel bir metot önerilmiştir. Algoritma Microsoft Excel programı altında VBA kodu ile yazılmıştır. Geliştirilen algoritma 5 farklı test fonksiyonu üzerinde test edilmiş ve başarılı sonuçlar alınmıştır. Daha sonraki çalışmalarda KKO tabanlı algoritmanın diğer sezgisel metotlarla birleştirilerek daha hızlı algoritmaların üretilmesine çalışılacak ve ayrıca koloni büyüklüğünün ve algoritma parametrelerinin önerilen algoritma üzerindeki performansının test edilmesine yönelik çalışmalar yapılacaktır.

5. Teşekkür

Bu çalışma Pamukkale Üniversitesi Bilimsel Araştırma Projeleri Biriminin desteklemiş olduğu BAP-FBE-002 nolu proje kapsamında gerçekleştirilmiştir.

6. Kaynaklar

[1] M. Dorigo, *Optimisation, learning and natural algorithms*. PhD Thesis, Politecnico di Milano, Italy, 1992.

[2] M. Guntsch, M. Middendorf, Applying Population Based ACO to Dynamic Optimization Problems, *ANTS 2002*, LNCS 2463, 111-122, 2002.

[3] M. Dorigo, L. M. Gambardella, Ant colony system: A cooperative learning approach to the traveling salesman problem, *IEEE Transactions on Evoluationary Computation*, 1, 53-66, 1997.

[4] C. Hamzacebi, F. Kutay, A heuristic approach for finding the global minimum: adaptive random search technique, *Applied Mathematics and Computation*, 173(2), 1323-1333, 2006.

[5] M. D. Toksarı, Ant colony optimization for finding the global minimum, *Applied Mathematics and Computation*, 176, 308-316, 2006.

[6] M. D. Toksarı, A heuristic approach to find the global optimum of function, Journal of Computational and Applied Mathematics, 209(2), 160-166, 2007.

[7] M. Dorigo, G. Di Caro, Ant colony optimisation: A new meta-heuristic. *In: Proceedings of the 1999 Congress on Evolutionary Computation*, 2, 1470-1477, 1999.

[8] J. L. Denebourg, J. M. Pasteels, J. C. Verhaeghe, Probabilistic Behaviour in Ants: a Strategy of errors?, *Journal of Theoretical Biology*, 10, 259-271, 1983.

[9] M. Dorigo, T. Stützle, *Ant Colony Optimization*, The MIT Press, Massachusetts Institute of Technology, Cambridge, 2004.

[10] Y. D. Kwon, S. B. Kwon, J. Kim, Convergence enhanced genetic algorithm with successive zooming method for solving continuous optimization problems, *Computers and structures*, 81, 1715-1725, 2003.