İleri Teknoloji Bilimleri Dergisi
Cilt 4, Sayı 1, 14-22, 2015

Journal of Advanced Technology Sciences
Vol. 4, No 1, 14-22, 2015

PARÇACIK SÜRÜ OPTİMİZASYONU İLE KISITSIZ OPTİMİZASYON TEST PROBLEMLERİNİN ÇÖZÜMÜ

[*]Pakize ERDOĞMUŞ, [*]E. YALÇIN

[*]Düzce Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü,
Düzce, TÜRKİYE

Özet- Kuş ve balık sürülerinin davranışlarından esinlenilerek geliştirilmiş bir yöntem olan Parçacık Sürü Optimizasyonu (PSO) hızlı yakınsayan bir algoritma olması sebebi ile son yıllarda Genetik Algoritma (GA) ve Benzetim Tavlama (BT) algoritmalarının ardından en çok çalışılan sezgisel optimizasyon algoritması olmuştur. Bu çalışmada, literatürde yer alan test problemleri önce standart PSO ile ve daha sonra PSO'da yaptığımız bir iyileştirme ile geliştirilen yeni algoritma ile çeşitli değişken sayıları için çözülmüş ve bu iki algoritmanın performansı mukayese edilmiştir. Problemlerin optimum çözümlerinin standart sapması, en iyi çözüm, ortalama çözüm süreleri tablo halinde sunulmuştur. Sonuçlardan görüleceği üzere geliştirilen PSO algoritmasının standart PSO'ya göre daha iyi sonuçlar verdiği görülmüştür.

Anahtar Kelimeler- Parçacık Sürü Optimizasyonu, Sezgisel Araştırma, Kısıtsız Optimizasyon

THE SOLUTIONS OF UNCONSTRAINED OPTIMIZATION BENCHMARK PROBLEMS WITH PARTICLE SWARM OPTIMIZATION

Abstract- Inspired from birds and flocks, Particle Swarm Optimization (PSO) is the most studied optimization methods after Genetic Algorithm and Simulated Annealing, because of the fact that PSO converges the optimum rapidly. In this study, some benchmark problems for various variable numbers given in the literature have been solved firstly with PSO, later a novel algorithm developed with an improvement from PSO and the performances of these two algorithms have been compared. The standard deviation of the optimum solutions, best optimum value, mean of the solution time have been presented with tables. Comparing with the PSO, a novel PSO outperformed PSO.

Key Words- Particle Swarm Optimization, Heuristic Search, Unconstrained Optimization

[*] pakizeerdogmus@duzce.edu.tr

1. GİRİŞ (INTRODUCTION)

Optimizasyon son yıllarda hemen her alanda kullanılan bir tekniktir. Matematiksel olarak çeşitli değişkenlere bağlı olarak değişen bir amaç fonksiyonunu verilen kısıtlacı şartlar altında en optimum(en küçük veya en büyük) yapan değerlerin bulunması işlemi olarak tanımlanan optimizasyon, tasarım mühendisliğinde, elektrik mühendisliğinde, ekonomide, bilgisayar ağlarında, robotikte, görüntü işlemede ve daha birçok alanda uygulama alanı bulmuştur. Tasarlanan bir makinanın optimum boyutlarının bulunması, bir işletmenin karının maksimize edilmesi veya üretilen bir ürünün minimum maliyetle üretimi, bir kişinin maksimum kar edebilmesi için elindeki parayı hangi yatırım araçlarına ne miktarda yatırması gerektiği, maksimum çekim alanı oluşturulacak şekilde sensör yerleşimi, elektrik yük akışının optimize edilmesi, bir robotun engellere çarpmadan minimum yolu takip etmesi problemleri birer optimizasyon problemidir.

Optimizasyon problemlerinin çözümünde klasik yöntemler yetersiz kaldığı için 1970'li yıllarda yeni çözüm yöntemleri arayışına gidilmiş ve sezgisel yöntemler çalışılmaya başlanmıştır. Özellikle son yıllarda kompleks ve zor optimizasyon problemlerinin çözümü için bir çok sezgisel algoritma geliştirilmiştir. Gerçek hayat problemlerinin geleneksel yöntemlerle çözülmesinin zorluğu ve etkin olarak çözülememeleri ilgiyi bu alana kaydırmıştır[1]. Bu algoritmalar tabiattan topluma, kültürden politika ve insana kadar çevremizde gördüğümüz hemen herşeyin modellenmesi ile geliştirilmiştir [2]. Günümüzde biyolojik sistemlerden esinlenilerek geliştirilmiş bir çok sezgisel yöntem optimizasyon problemlerinin çözümünde kullanılmaktadır. Bu alanda GA bir ilk olup, evrim teorisinden esinlenilerek ortaya çıkarılmıştır. 1976 yılında Holland[3] tarafından geliştirildiğinden beri en çok çalışılan sezgisel algoritmadır. GA, uzay mühendisliği, astronomi ve astrofizik, jeofizik, finans, malzeme mühendisliği ve daha çok sayıda alanda uygulanmış bir algoritmadır[4].

Biyolojik sistemlerde, bireyin çevresiyle ve diğer bireylerle olan etkileşimi ve ortak davranışlarının incelenerek modellenmesi ile sürü zekasına dayalı bir çok algoritma geliştirilmiştir. 1990'lı yılların başında M. Dorigo ve arkadaşları tarafından geliştirilen Karınca Kolonisi Optimizasyonu(Ant Colony Optimization(ACO)) [5] karıncaların sürü zekasını, özellikle ayrık optimizasyon problemlerinin çözümüne uyarlamış bir algoritmadır. PSO (Particle Swarm Optimization) kuş ve balık sürülerinden esinlenilerek oluşturulan bir optimizasyon tekniğidir. 1995 yılında J.Kennedy ve R.C.Eberhart tarafından geliştirilmiştir [6]. PSO, fonksiyon optimizasyonu, yapay sinir ağları eğitimi, otomotik kontrolde parametrelerin ayarlanması gibi birçok optimizasyon problemini başarı ile çözmüştür[7].

Sürü zekasına dayalı olarak geliştirilen bir başka algoritma Yapay Arı Kolonisi(Artificial Bee Colony(ABC)) Optimizasyon algoritması 2005 yılında Karaboğa tarafından tanıtılmıştır[8]. ABC algoritması da bal arılarının yiyecek arama davranışını temel alarak rotalama ve çizelgeleme alanında bir çok optimizasyon problemini başarı ile çözmüştür.

Mercan Resifleri Optimizasyon algoritması ise 2014 yılında Sanz ve arkadaşları tarafından tanıtılmıştır. Mobil ağların optimizasyonunda genellikle çekim alanı ve maliyete bağlı yerleştirmeler yapılır. Bu çalışmada ise elektromanyetik kirliliği minimize edecek yerleşim problemi Mercan Resiflerinden esinlenilerek geliştirilen algoritma ile çözümlüştür [9].

Guguk kuşu algoritması(Cuckoo Optimization Algorithm(COA)) guguk kuşlarının kuluçka paraziti türlerini temel alarak 2011 yılında geliştirilen global optimizasyon algoritmasıdır[10]. Ateş böceği optimizasyonu (Firefly Optimization(FO) ve Yarasa algoritması (Bat Algorithm(BA)) da sırası ile 2009 ve 2010 yıllarında geliştirilen sezgisel optimizasyon algoritmalarıdır[11,12].

Bu çalışmanın ikinci bölümünde PSO algoritması ve geliştirilen algoritma tanıtılmıştır. Üçüncü bölümde mukayese için kullanılan test problemleri ve bu problemlerin çözümüne ait tablolar sunulmuştur. Çalışmanın son bölümünde ise sonuçlar karşılaştırılarak öneriler sunulmuştur.

2. PARÇACIK SÜRÜ OPTİMİZASYONU

PSO balıklar, kuş sürüleri ve bazı hayvanların davranışlarından esinlenilerek, sürekli optimizasyon problemleri için geliştirilmiş bir metoddur. Balık sürüleri, kuş sürüleri ve diğer sosyal hayvanlar incelendiğinde bu hayvanların yiyecek ararken etkileşim içerisinde oldukları ve birinin yiyecek bulması ile diğerlerinin de sürüden kopmadan konumlarını yiyeceğin olduğu yöne çevirdikleri ve hızlarını da buna göre güncelledikleri görülmüştür. Bu sosyal etkileşim PSO ile modellenmiştir.
PSO, GA gibi evrimsel bir algoritmadır. GA'da o anki tüm çözümlere populasyon adı verilirken, PSO'da sürü(swarm) adı verilir. PSO, GA gibi evrimsel bir algoritma olmasına rağmen çaprazlama ve mutasyon gibi operatörleri olmadığından daha basittir ve optimum çözüme yakınsaması daha hızlıdır. Sürüyü oluşturan her bir kuş bir parçacık(particle) olarak adlandırılır. PSO'da her bir parçacık bir aday çözümdür. Örneğin D(dimension) değişkenli bir problemin uygunluk fonksiyonu hesabı için *n* adet parçacık ile bir başlangıç çözümü denklem (1)'de verilmiştir.

$$\begin{bmatrix} x_{11} & x_{12} & \dots & x_{1D} \\ x_{21} & x_{22} & \dots & x_{2D} \\ \dots & \dots & \dots & \dots \\ x_{n1} & x_{n2} & \dots & x_{nD} \end{bmatrix} \quad (1)$$

i. parçacık için uygunluk fonksiyonu aşağıda denklem (2)'de verilmiştir.

$$f_{fitness}=f(x_{i1},x_{i2},x_{i3},\dots,x_{iD}) \quad (2)$$

Yukarıdaki matriste parçacık olarak adlandırılan her bir satır bir çözümü ifade etmektedir. n parçacık için n adet çözüm elde edilir. Parçacıklar araştırma uzayında iki önemli parametreye göre hareket ederler. Her iterasyonda önce her bireyin kendi en iyisi ***pbest*** ve tüm bireylerin en iyisi ***gbest*** elde edilir. n adet parçacık ile çalışan bir PSO'da her iterasyonda n adet ***pbest*** var iken sadece bir ***gbest*** mevcuttur. Algoritmanın başında her parçacık için başlangıç değer aynı zamanda ***pbest*** değeridir. Sonraki iterasyonlarda diğer parçacıkların durumuna göre konumu güncellenir ve o anki konum önceki ***pbest*** ile karşılaştırılır. Eğer ***pbest***'ten daha iyi bir çözüm ise artık yeni çözüm ***pbest*** olarak atanır.

PSO'da sürü bir başlangıç çözümü ile başlar. Başlangıçta her bir parçacığın hız ve konum değeri vardır. Parçacık konumları tanımlı aralık içindeki değişken değerleridir. Parçacıklar hızlarını hem kendilerine hemde sürüye göre ayarlayarak hareket ederler. Parçacıklar her bir iterasyonda optimum çözüme biraz daha yaklaşırlar. ***i.*** Parçacığa ait hız ve konum formülleri denklem (3) ve (4)'te verildiği gibidir[13,14].

$$v_i^{k+1} = K(v_i^k + \varphi_1 rand()(p_{best\,i}{}^k - x_i^k) + \varphi_2 rand()(g_{best} - x_i^k)) \quad (3)$$

Bu denklemde ***k*** iterasyon sayısıdır. v_i^k ***i.*** parçacığın ***k.*** iterasyondaki hızıdır. v_i^{k+1} ise bir sonraki iterasyon için hızıdır. K, φ_1 ve φ_2 birbirine bağlı katsayılardır. K kısıtlama faktörü parçacıkların hareketlerindeki osilasyonu sönümleme etkisine sahiptir. Bu sebeple parçacıklar zamanla sonuca yakınsarlar. φ_1 and φ_2 sosyal(social) ve bilişsel(cognitive) parametrelerdir. rand düzgün rastgele dağılımdan rastgele çekilen (0-1) arası bir sayıdır. Hız hesaplandıktan sonra konum aşağıdaki formülle hesaplanır.

$$x_i^{k+1} = x_i^k + v_i^{k+1} \quad (4)$$

K φ_1 and φ_2 'ye bağlı bir sabittir. Denklem (5)'te K sabitinin formülü verilmiştir.

$$K = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|} \quad \varphi = \varphi_1 + \varphi_2 \quad \varphi > 4 \tag{5}$$

φ_1, parçacığın kendi tecrübelerine göre hareket etmesini(bilişsel), φ_2 ise sürüdeki diğer parçacıkların tecrübelerine göre (sosyal) hareket etmesini sağlar. Şekil 1'de PSO'nun akış şeması verilmiştir.

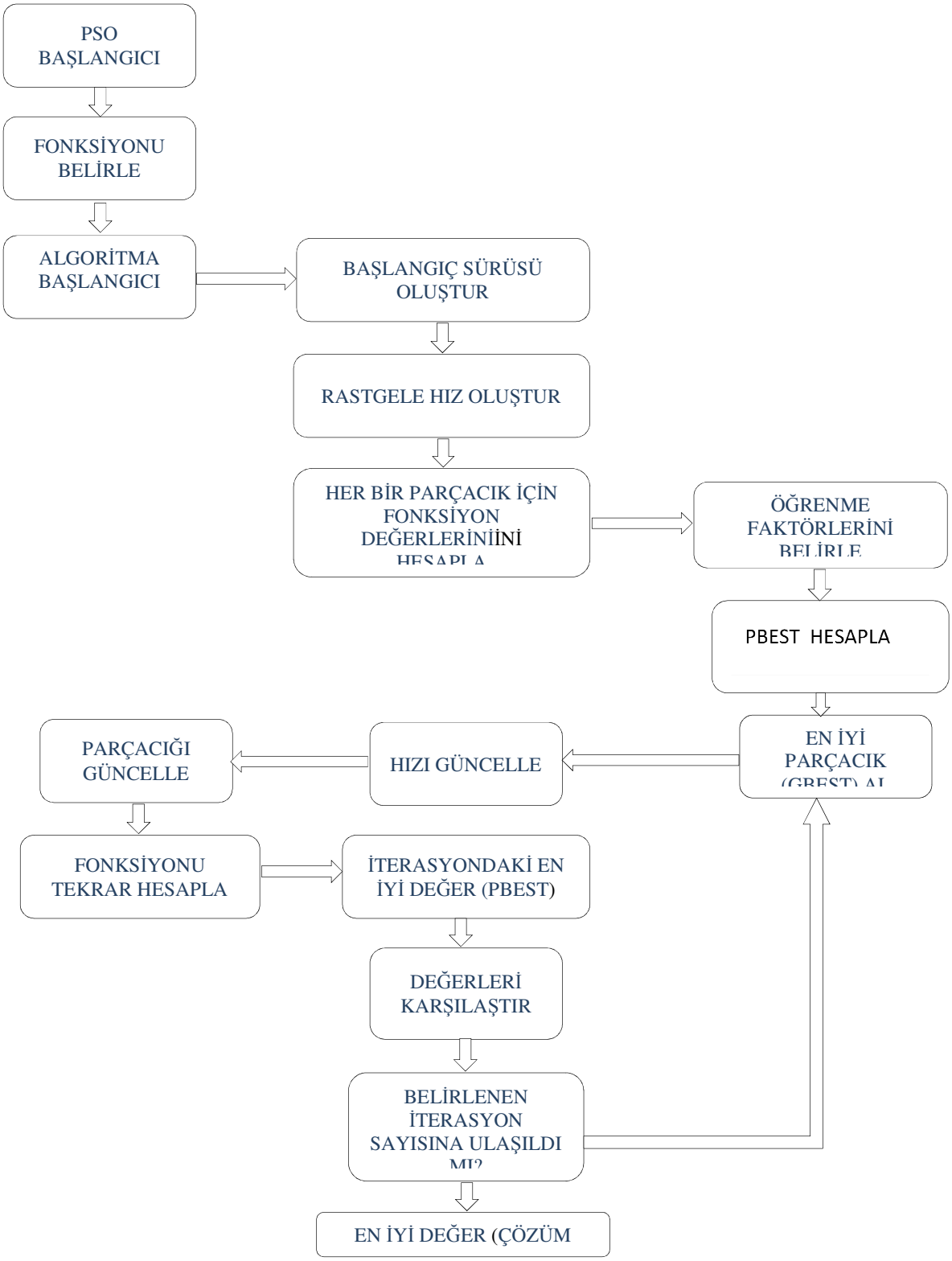
PSO 'da Parçacık Sayısı n 20 ile 40 arasında olabilir. Daha hassas ölçüm yapılması gereken durumlarda parçacık sayısı arttırılabilir. Problem değişken sayısı ve değişkenlerin değer aralığı probleme göre değişir. φ_1 ve φ_2 genellikle 2'ye yakın değerler seçilir. Sırası ile 2 ve 2.1 değerleri alınarsa, K sabiti 0.7298 olarak kullanılır[14].

3. GELİŞTİRİLEN PSO ALGORİTMASI

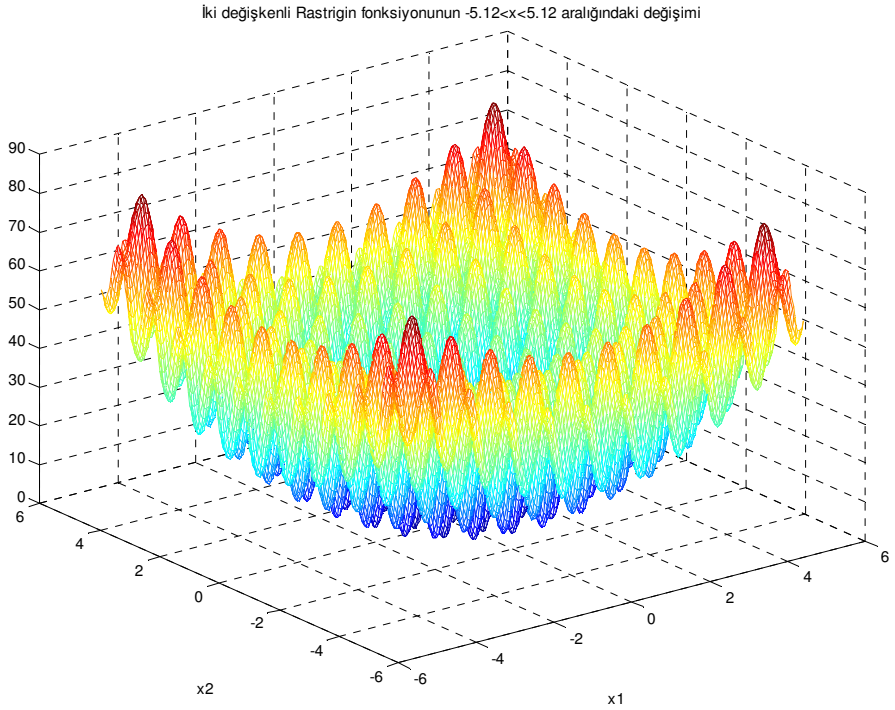
Yukarıda denklemleri verilen PSO algoritmasında, K kısıtlama faktörü parçacıkların hareketlerindeki osilasyonu sönümleme etkisine sahip olup algoritma boyunca sabittir. Oysa sezgisel araştırma yöntemlerinin bilinen iki önemli özelliği araştırma ve yoğunlaşmadır. Araştırma(exploration) ile sayılmayacak kadar çok çözüme sahip olan optimizasyon problemi, çözüm uzayının iyice taranmasını sağlar. Bu şekilde algoritmanın başında yerel çözümlere takılmadan küresel en iyiye ulaşılması hedeflenmektedir. Algoritmanın sonlarında ise araştırma yolu ile çözüm uzayı iyice taranmış ve artık en iyi çözümün yeri kabaca tesbit edilmiştir. Yoğunlaşma ile kabaca elde edilen çözümün hassas değerlerinin elde edilmesi gerekir.

Bu amaçla standart PSO algoritmasında kullanılan iterasyon sayısının yarısı kadar iterasyonda K değeri daha yüksek tutulmuştur. Bu ilk çözümlerin pbest ve gbest değerleri ve daha düşük K değeri ile algoritma bir kez daha çalıştırılarak en iyi değerler elde edilmiştir. Bu çalışmada Standart PSO için φ_1 and φ_2 sırası ile 2 ve 2.1 alınarak K sabiti 0.7298 olarak kullanılmıştır. Geliştirilen algoritmada ise ilk durumda K 0.8298 olarak ele alınmış, daha sonra K 0.7298'e düşürülmüştür.

Çalışmada standart algoritmadaki değişikliğin algoritmanın performansına etkisini test etmek için kısıtsız optimizasyon test problemleri seçilmiştir. Bu problemler literatürde geliştirilen algoritmaların performansını test etmede kullanılan problemler olup, bu test problemlerinin tamamına Global Optimization Test Problems [15] web sayfasından ulaşılabilir. Bu problemlerden kısıtsız optimizasyon problemleri basit ve kompleks olmak üzere ikiye ayrılır. Ayrıca kısıtlı optimizasyon test problemleri de mevcuttur. Bu problemlerin en önemli özelliği bir küresel optimum ve çok sayıda yerel optimuma sahip olmalarıdır. Bu problemlerden Rastrigin fonksiyonunun n=2 değişken için grafiği aşağıdaki Şekil 2'de verilmiştir.



Şekil 1. Parçacık sürü optimizasyonunun akış diyagramı



Şekil 2. Rastrigin Fonksiyonu Grafiği

Çalışmada kullanılan test problemleri aşağıdaki Tablo 1' ve Tablo 2'de verilmiştir.

Tablo 1. Basit kısıtsız optimizasyon test problemleri

Fonksiyon adı	**Değişken kısıtları**	**Fonksiyon(f(x)), (x* , f(x*)) (Optimum değerler)**
Branin	$-5 \le x_1 < 10$ $0 \le x_2 < 15$	$f(x) = (x_2 - \frac{5}{4\pi^2} x_1^2 + \frac{5}{\pi} x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})\cos(x_1) + 10$ $x^* = (-\pi, 12.275), (\pi, 2.275), (9.42478, 2.475)$ $f(x^*) = 0.397887$
Bohachevsk	$-100 \le x_i \le 100$ $i = 1,2$	$f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$ $x^* = (0,0)$ $f(x^*) = 0$
Beale	$-4.5 \le x_i \le 4.5$ $i = 1,2$	$f(x) = (1.5 - x_1 + x_1 x_2)^2 + (2.25 - x_1 + x_1 x_2^2)^2 + (2.625 - x_1 + x_1 x_2^3)^2$ $x^* = (3,0.5)$ $f(x^*) = 0$

Tablo 2. Kompleks kısıtsız optimizasyon test problemleri

Fonksiyon adı	**Değişken kısıtları**	**Fonksiyon(f(x)), (f(x*)) ve x*(Optimum değerler)**
Rastrigin(20)	$-5.12 \le x_i \le 5.12$ $i = 1,2,...,n$	$f(x) = 10n + \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i))$ $x_i^* = 0, i = 1,2,...,n$ $f(x^*) = 0$
Grievank (20)	$-600 \le x_i \le 600$ $i = 1,2,...,n$	$f(x) = \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) + 1$ $x_i^* = 0, i = 1,2,...,n$ $f(x^*) = 0$

Test aşamasında her bir fonksiyon için değişken sayısı her değiştirildiğinde program iki algoritma için de 2 kez çalıştırılmıştır. Tüm programlar Matlab'da© yazılmıştır. Program her çalıştırıldığında, PSO algoritmasını 30 kez çalıştırıp sonuçları Excell© dosyasına kaydetmektedir. Aşağıdaki tablolarda test fonksiyonlarının bulunan en iyi değerleri, değerlerin standart sapması, 30 çalışmanın ortalama çözüm süresi değerleri verilmiştir. Test1 geleneksel PSO algoritmasını, Test2 ise çalışmada geliştirilen PSO algoritmasını ifade etmektedir. Yine her bir problemin en iyi değeri ve bu değeri sağlayan değişken değerleri de sonuçların mukayesesini kolaylaştırmak için tabloda verilmiştir. Kolay test problemleri 25 parçacık ve 100 iterasyon ile, zor test problemleri 40 parçacık ve 1000 iterasyon ile çözülmüştür.
Tablo 3'ten Tablo 7'ya kadar ki tablolar simülasyon sonuçlarını göstermektedir. Tablo 3, Tablo 4 ve Tablo 5 kolay test problemleri, Tablo 6 ve Tablo 7 zor test problemleri için simülasyon sonuçlarını göstermektedir.

Tablo 3. Branin fonksiyonu test sonuçları

Test no	Süre(sn)	Standart Sapma	x_1	x_2	*f(x*)*
Literatürde verilen optimum değerler					
			-π π 9.42478	12.275 2.275 2.475	0.397887
Test1					
1	1.8532	0,162143	3,1333536	2,304815	0,400541
2	1.8438	0,19194	3,145805	2,247061	0,397973
Test2					
1	1.9179	1,82289E-07	3,141593	2.25	0,397887
2	1.9916	0	-3,1416	12,25	0,397887

Tablo 4. Bohachevsky fonksiyonu test sonuçları

Test no	Süre(sn)	Standart Sapma	x_1	x_2	*f(x*)*
Literatürde verilen optimum değerler					
			0	0	0
Test1					
1	1.8157	1,636228	0,008148	0,024988	0,021758
2	1.8073	1,662756	0,009904	-0,01202	0,0062498
Test2					
1	1.8681	1,16783E-06	-5E-6	2,59E-6	5,81E-10
2	1.8651	2,41882E-06	5,83E-06	-3,4E-6	8,69E-10

Tablo 5. Bealee Fonksiyonu Test Sonuçları

Test no	Süre(sn)	Standart Sapma	x_1	x_2	*f(x*)*
Literatürde verilen optimum değerler					
			3	0,5	0
Test1					
1	1.8528	0,90748	2,977786	0,501949	0,001358
2	1.8642	1,320036	3,205034	0,556912	0,008603
Test2					
1	1.9208	0,121723	2,999996	0,500002	2,51E-10
2	1.9003	0,098105	3,000003	0,5	9,11E-1

Tablo 6. Rastrigin Fonksiyonu (n=20) Test Sonuçları

Test no	Süre(sn)	Standart Sapma	x_1	x_{20}	*f(x*)*
Literatürde verilen optimum değerler					
			0	0	0
Test1					
1	3.2894	22,33585	3,140633	-0,97166	166,1561
2	3.3871	24,59577	-0,32081	2,060939	176,2903
Test2					
1	4.2539	15,14371	0,989263	0,022971	28,06025
2	4.3252	15,07511	1,031468	-0,96714	25,5118

Tablo 7. Griewank Fonksiyonunun (n=20) Test Sonuçları

Test no	Süre(sn)	Standart Sapma	x_1	x_{20}	*f(x*)*
Literatürde verilen optimum değerler					
			0	0	0
Test1					
1	3.3132	0,035716	-3,38157	2,994367	0,89709
2	3.5112	0,029734	-0,23549	0,445136	0,913113
Test2					
1	4.3498	0,047744	-0,00992	-0,22559	0,005409
2	4.4216	0,079669	-0,03221	-0,38033	0,014953

4. SONUÇ VE TARTIŞMA (CONCULUSION AND DISCUSSION)

Bu çalışmada, son yıllarda çok çalışılan bir algoritma olan PSO algoritmasının performansının arttırılması amaçlanmıştır. Sezgisel optimizasyon algoritmalarının başarısını etkileyen en önemli faktör, parametrelerinin optimizasyonudur. Her optimizasyon algoritmasının

parametreleri vardır. İterasyon sayısı bunlardan en çok kullanılan ve bilinen parametredir. İterasyon sayısının az olması algoritmanın yakınsayamamasına, fazla olması ise çözüm süresinin ve bilgisayar kaynaklarının gereksiz kullanımına yol açar.
Problemlerin test aşamasında kısıtsız optimizasyon problemlerinin kolay olan problemleri Branin, Bohachevsky, Beale ve zor problemlerden olan Rastrigin ve Grievank çözdürülmüştür. Değişken sayısı arttıkça problem uzayı oldukça karmaşık hale geldiği için algoritmalar kolay problemlerde optimum çözümü oldukça kolay bulmakta, zor problemlerde ise optimum çözümü bulmakta zorlanmaktadırlar. Tüm simülasyonlar Intel i7-3612 QM 2.1GHz işlemci ve 4GB RAM'e sahip bir bilgisayarda simüle edilmiştir.
Algoritmanın performansını arttırmak amacıyla, algoritma iki aşamalı olarak çalıştırılmıştır. İlk aşamada K faktörü 0.8298 olarak, ikinci aşamada 0.7298 olarak çalıştırılmıştır. Elde edilen sonuçlar tablolarda sunulmuştur. Elde edilen sonuçlardan görüleceği üzere K parametresinin iki aşamalı olarak çok basit bir şekilde adaptif yapılması PSO algoritmasının başarısını arttırmaktadır. Algoritmanın başarısı ile birlikte çözüm süresi de artmıştır. Tablolardan görüldüğü üzere çözüm süresindeki artış bağıl olarak en fazla %5'lerde iken optimum çözümde çok daha büyük iyileşme görülmüştür. Gerçek zamanlı uygulamalarda geliştirilen PSO algoritmasının kullanılması tavsiye edilir.

5. KAYNAKLAR (REFERENCES)

[1]. Haklı, H., Uğuz, H, (2014). A novel particle swarm optimization algorithm with Levy flight, *Applied Soft Computing*, 23, 333-345.

[2]. Kashan, A.H, (2014). League Championship Algorithm (LCA): An algorithm for global optimization inspired by sport championships, *Applied Soft Computing*, 16,171-200.

[3]. Goldberg, D. E., (1989). Genetic Algorithms in Search Optimization and Machine Learning, *Addison Wesley*, ISBN 0201157675.

[4]. Lim, T. Y., (2014). Structured population genetic algorithms: a literature survey, *Artif Intell Rev*, 41, 385-399.

[5]. Dorigo, M., Blum, C., (2005). Ant colony optimization theory: A survey, *Theoretical Computer Science*, 344(2-3), 243-278.

[6]. Kennedy, J., Eberhart, R., (1995). Particle swarm optimization, *Neural Networks, Proceedings. IEEE International Conference on*, 4, 1942-1948.

[7]. Erdogmus, P., (2010). Particle Swarm Optimization Performance On Special Linear Programming Problems, *Scientific Research and Essays*, 5(12), 1506-1518.

[8]. Karaboga, D., Gorkemli, B., (2014). A quick artificial bee colony (qABC) algorithm and its performance on optimization problems, *Applied Soft Computing*, 23, 227-238.

[9]. Mehrabian, R., Lucas, C., (2006). A novel numerical optimization algorithm inspired from weed colonization. *Ecological Informatics*, 1(4), 355-366.

[10]. Rajabioun, R., (2011). Cuckoo Optimization Algorithm, *Applied Soft Computing*, 11(8), 5508-5518.

[11]. Mishra, S., Shaw, K., Mishra, D., (2012). A New Meta-heuristic Bat Inspired Classification Approach for Microarray Data, *Procedia Technology*, 4, 802-806.

[12]. Yang, X. S, Hosseini,S. S. S, Gandomi, A. H., (2012) Firefly Algorithm for solving non-convex economic dispatch problems with valve loading effect, *Applied Soft Computing*, 12(3), 1180-1186.

[13]. Clerc, M., (1999). The swarm and the queen: towards a deterministic and adaptive particle swarm optimization, *Evolutionary Computation, CEC 99. Proceedings of the 1999 Congress on*, 3, 1951-1957.

[14]. Parrott, D., Xiaodong L., (2006). Locating and tracking multiple dynamic optima by a particle swarm model using speciation, *Evolutionary Computation, IEEE Transactions on*, 10(4), 440,458.

[15]. http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO.htm (Erişim tarihi: 3th of September, 2014).