Yapay Arı Koloni (Artificial Bee Colony, ABC) Algoritması ile Yapay Sinir Ağlarının Eğitilmesi

Artificial Bee Colony (ABC) Algorithm on Training Artificial Neural Networks

Derviş Karaboga, Bahriye Akay

Bilgisayar Mühendisliği Bölümü
Erciyes Üniversitesi, Kayseri
karaboga@erciyes.edu.tr, bahriye@erciyes.edu.tr

Özetçe

Bu çalışmada sinyal işleme uygulamalarında oldukça sık kullanılan Yapay Sinir Ağlarının eğitiminde, son zamanlarda geliştirilen Yapay Arı Kolonisi algoritmasının performansı test edilmiş ve sonuçları yine popülasyon tabanlı olan Farksal Gelişim Algoritması ve Parçacık Sürüsü Optimizasyon Algoritmaları ile karşılaştırılmıştır. Yapay Sinir Ağları eğitiminde ele alınan problemler için ABC algoritmasının üstün performans gösterdiği görülmüştür.

Abstract

In this work, performance of the Artificial Bee Colony Algorithm, a recently proposed algorithm, has been tested on training on Artificial Neural Networks which are widely used in signal processing applications and the performance of the algorithm has been compared to Differential Evolution and Particle Swarm Optimization Algorithms which are also population-based algorithms. Results show that ABC algorithm outperforms the other algorithms.

1. Giriş

Sürü zekası; termitler, arılar, karıncalar, kuşlar, balık sürüleri gibi aralarında etkileşim olan böceklerin veya diğer sosyal hayvanların topluluk halindeki davranışlarını örnek alarak, problemlere çözüm getirmeyi amaçlayan bir yapay zeka tekniğidir. Arı kolonilerinin kovan etrafında dolaşarak birbirlerine bilgi aktarımları, karıncaların geçtikleri yollara kimyasal madde bırakarak diğer karıncalara bilgi aktarımları, kuş sürülerinin ve balık sürülerinin konum ve hızlarını ayarlayarak ilerlemeleri sürü zekasına temel teşkil eden zeki davranışlardır.

Bir sürüde iki önemli işlev vardır: a) kendi başına organize olabilme (self-organization) b) iş bölümü. Kendi başına organize olabilme; bir sistemdeki temel birimlerin, diğer birimlerle etkileşimden aldıkları bilgileri kullanarak kendi başlarına işlev görerek sistemin bütününü etkilemeleridir. Sistemin diğer birimleri ile etkileşiminde temel komşuluk bilgilerinden faydalanılır. Yani sistemin bir bütün olarak genel başarımı ile ilgili bilgiler söz konusu değildir. Bonabeau ve ark. kendi başına organize olabilmeyi pozitif geri besleme (positive feedback), negatif geri besleme (negative feedback), salınımlar (fluctuation) ve çoklu etkileşim olmak üzere dört özellik ile karakterize etmişlerdir [1]. Pozitif geri besleme daha uygun yapıların bulunmasına sağlayan davranışladır. Karıncalardaki kimyasal maddenin bırakılması ve diğer karıncaların bu maddeyi takibi ile daha uygun yolların bulunması, arıların dans etmeleri ile zengin nektar kaynakları hakkındaki bilgileri diğer arılara iletmeleri pozitif geri beslemeye örnek verilebilir. Negatif geri besleme ise toplanan bilgilerin kararlı hale gelebilmesi için çalışır. Tüm bireylerinin aynılaşarak topluluğun doyuma ulaşmasını engeller. Salınımlar da yeni kaynak keşiflerinin yapılabilmesi için rasgele dolaşımlar gibi düşünülebilir. Çoklu etkileşim ile de bir bireyin diğer bireye ait bilgiyi kullanabilmesi ifade edilmektedir.

İş bölümü, topluluktaki bireylerin eş zamanlı olarak farklı işleri gerçekleştiriyor olmasıdır. Özelleşmiş bireylerin bir arada çalışarak gösterdikleri performans, bu şekilde bir iş bölümüne tabi olmayan bireylerin gösterdikleri performanstan daha etkili olmaktadır ve bu özellik araştırma uzayındaki değişimlere cevap verebilmeyi sağlamaktadır.

Literatürde yapay arı kolonisi algoritmasının dışında, sürü zekasındaki bu özellikleri temel alan ve arıların davranışını modelleyen çeşitli metotlar bulunmaktadır [2]-[6]. Ancak bunlardan sadece Yang tarafından geliştirilen Virtual Bee Algoritması [6] nümerik problemler için çözüm üretmektedir ve bu yaklaşımda sadece problemlerin iki boyutlu olması durumunda kullanılmaktadır. Bu çalışmada Karaboga tarafından geliştirilen [7] ve literatürde mevcut algoritmalarla kıyaslandığında oldukça iyi sonuçlar üreten Yapay Arı Kolonisi Algoritmasının [8]-[10] sinyal işlemede çokça kullanılan yapay sinir ağlarının eğitiminde gösterdiği performans analiz edilerek, bu performansı Farksal Gelişim (Differantial Evolution, DE) [11] ve Parçacık Sürüsü Optimizasyon (Particle Swarm Optimizasyon, PSO) [12] algoritmaları ile kıyaslanmıştır.

İkinci bölümde Yapay Arı Kolonisi Algoritması, üçüncü bölümde yapay sinir ağlarının eğitimi, dördüncü bölümde çalışmada kullanılan problemler ile algoritmalara has kontrol parametrelerinin değerleri ve son olarak beşinci bölümde sonuçlar sunulacaktır.

2. Yapay Arı Kolonisi Algoritması

Yapay arı kolonisi algoritmasında, bir koloni de üç grup arı bulunmaktadır: işçi arılar, gözcü arılar ve kaşif (scout) arılar. Modelimizde, kolonin yarısı işçi, yarısı gözcü arı olarak seçilmiştir. Her bir nektar kaynağı için sadece bir işçi arı bulunmaktadır. Yani işçi arıların sayısı nektar kaynağı sayısına eşittir. Algoritmanın temel adımları ise şu şekildedir:

Initialization
REPEAT

- İşçi arıları kaynaklara gönder ve nektar miktarlarını hesapla
- Gözcü arıları kaynaklara gönder ve nektar miktarlarını hesapla
- Rasgele yeni kaynaklar bulmaları için kaşif arıları gönder
- O ana kadarki en iyi kaynağı hafızada tut

UNTIL (durma kriteri sağlanana kadar)

Her bir çevrim üç adımdan oluşmaktadır: işçi ve gözcü arıların kaynaklara gönderilmesi, gidilen kaynakların nektar miktarlarının hesaplanması, kaşif arının belirlenerek yeni bir kaynağa rasgele konumlanması. Yiyecek kaynakları optimize edilmeye çalışılan problemin olası çözümlerine karşılık gelmektedir. Bir kaynağa ait nektar miktarı, o kaynakla ifade edilen çözümün kalite değerini ifade etmektedir. Gözcü arılar rulet tekerleği prensibine [13] göre gidecekleri kaynakları belirlemektedirler. Her kolonide rasgele araştırma yapan kaşif arılar bulunmaktadır. Bu arılar yiyecek ararken herhangi bir ön bilgi kullanmamakta, tamamen rasgele araştırma yapmaktadırlar. Dolayısıyla arama maliyetleri düşüktür ve de buldukları kaynağın ortalama kalite değeri düşüktür. Zengin nektar kaynağına sahip keşfedilmemiş kaynakları bulmaları da olasıdır. ABC algoritmasında işçi arılardan biri seçilerek kaşif arı haline gelmektedir. Bu seçme işlemi "limit" parametresine göre yapılmaktadır. Bir kaynağı ifade eden çözüm belli sayıdaki deneme ile geliştirilememişse bu kaynak terk edilir ve bu kaynağa gidip gelen işçi arı kaşif arı haline gelir. Kaynağın terk edilmesi için belirlenmiş deneme sayısı "limit" parametresi ile belirlenmektedir.

Gürbüz bir arama sürecinde keşif (exploration) ve keşfedilenden faydalanma (exploitation) aynı anda gerçekleşmelidir. ABC algoritmasında gözcü ve işçi arılar keşfedilen kaynaklardan faydalanma işleminde, kaşif arılar ise keşif sürecinde görev alırlar. Gerçek arılarda taşıma hızı kolonin bir kaynağı bulması ve onu kovana getirmesi ile belirlenirken, yapay arılar durumunda bulunan çözümün kalite değeri yani uygunluğu ile belirlenir.

Diğer sosyal yiyecek arayıcıları gibi, arılar E/T değerini yani birim zamanda yuvaya getirilen yiyecek miktarını belirten enerji fonksiyonunu maksimize etmek için çalışırlar. Bir maksimizasyon probleminde de amaç fonksiyonunun $F(\theta_i)$, $\theta_i \in R^p$, maksimize edilmesi işlemi gerçeklenir. θ_i , i. kaynağın pozisyonu olmak üzere $F(\theta_i)$ bu nektar miktarına karşılık gelir ve $E(\theta_i)$ ile orantılıdır. c çevrim sayısı (cycle), S: kovan etrafındaki nektar kaynağı sayısı olmak üzere $P(c) = \{\theta_i(c) | i = 1,2,\ldots,S\}$ tüm kaynakların pozisyon bilgilerini içeren nektar kaynağı popülasyonudur. Daha önce belirtildiği gibi gözcü arıların bir kaynağı seçmeleri $F(\theta)$ değerine bağlı idi. Kaynağın nektar miktarı ne kadar fazla olursa, bu kaynağı bir gözcü arı tarafından seçilme olasılığı o kadar fazla olmaktadır. Yani θ_i pozisyonundaki bir kaynağın seçilme olasılığı şu şekildedir:

$$P_i = \frac{F(\theta_i)}{\sum_{k=1}^{S} F(\theta_k)} \quad (1)$$

Gözcü arı, işçi arıların dansını izledikten ve (1) eşitliğindeki olasılık değeri ile θ_i konumundaki kaynağı seçtikten sonra, bu kaynağın komşuluğunda bir kaynak belirler ve kaynağın nektarını almaya başlar. Yani θ_i civarındaki kaynaklar arasında bir kıyaslama yapar. Seçilen komşuya ait pozisyon bilgisi şu şekilde hesaplanmaktadır:

$$\theta_i(c+1) = \theta_i(c) \pm \phi_i(c) \quad (2)$$

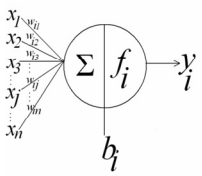
$\phi_i(c)$, θ_i civarında daha fazla nektara sahip bir kaynak bulabilmek için kullanılan, rasgele üretilen adım büyüklüğüdür. $\phi_i(c)$, k i'den farklı rasgele üretilen popülasyondaki bir çözüme ait indis olmak üzere $\theta_i(c)$ ve $\theta_k(c)$ çözümlerinin bazı bölümlerinin farkının alınması ile hesaplanır. $\theta_i(c+1)$ e ait nektar miktarı $F(\theta_i(c+1))$, $\theta_i(c)$ konumundaki kaynağa ait nektar miktarından daha fazla ise arı kovana giderek bu bilgisini diğerleri ile paylaşır ve yeni pozisyon olarak $\theta_i(c+1)$ aklında tutar, aksi durumda $\theta_i(c)$ yi hafızasında saklamaya devam eder. θ_i konumundaki nektar kaynağı "limit" parametresi sayısınca gelişememiş ise θ_i deki kaynak terk edilir ve o kaynağın arısı kaşif arı haline gelerek rasgele araştırma yapar, yeni bulduğu kaynak θ_i ye atanır.

3. Yapay Sinir Ağlarının Eğitilmesi

Yapay sinir ağları nöron denilen aralarında bağların bulunduğu işlemci elemanlardan oluşmaktadır (Şekil 1). i. nörona ait çıkış (3) denklemi ile verilebilir:

$$y_i = f_i\left(\sum_{j=1}^{n} w_{ij} x_j + b_i\right) \quad (3)$$

y_i nöron çıkışı, x_j nörona ait j. giriş w_{ij} nöron ve x_j giriş arasındaki bağlantı ağırlığı, b_i eşik (yada bias) değeri ve f_i de transfer fonksiyonudur.



Şekil 1 Yapay Sinir Ağına ait İşlemci Birimi

Bu fonksiyon genelde heaviside, sigmoid yada Gaussian gibi doğrusal olmayan tipte seçilmektedir. Genellikle adaptasyon network çıkışındaki E hatasının minimize edilmesiyle sağlanır. Hata fonksiyonu (4) ile verilir:

$$E(w(t)) = \frac{1}{n}\sum_{j=1}^{n}\sum_{k=1}^{K}(o_k - d_k)^2 \quad (4)$$

$E(w(t))$ t. iterasyondaki hata değeri, $w(t)$ t. iterasyondaki ağırlıklar, d_k istenen çıkış, o_k $w(t)$ ağırlıkları ile hesaplanan network çıkışıdır. Optimizasyon işlemi optimum $w(t)$ ağırlık değerlerinin bulunarak $E(w(t))$ hata fonksiyonunun minimize edilmesidir.

Klasik tekniklerin sahip olduğu erken yakınsama, lokal minimalara takılma, hesaplama karmaşıklığının yüksek olması gibi dezavantajlar, yapay sinir ağları eğitiminde sezgisel yaklaşımların kullanımını ortaya çıkarmıştır. Gelişim algoritmaları ile yapay sinir ağlarına ait ağırlık değerleri bireyler olarak düşünülür, çaprazlama, mutasyon gibi operatörlerle ağırlık matrisi değiştirilerek yapay sinir ağının ürettiği hata değeri minimize edilmeye çalışılır. Bu işlemler şu döngü ile verilebilir:

1. çaprazlama ve mutasyon gibi operatörlerle bir sonraki ağırlıkları içeren popülasyonu üret
2. üretilen ağırlık değerleri ile tasarlanan yapay sinir ağının uygunluk değerini hesapla ve uygunluk temelli seçme işlemi uygula
3. istenen sonuç elde edilmişse dur, yoksa 1. adıma dön.

4. Çalışma

Bu çalışmada yapay sinir ağlarında eğitilmek üzere 3 problem ele alınmıştır. Bunlar XOR problemi, 3-Bit Parity problemi, 4-bit Encoder-Decoder problemleridir.

4.1. XOR Problemi

XOR problemi ikili formdaki iki girişi alarak (0 0;0 1;1 0;1 1)→(0;1;1;0) şeklinde eşleme yapan zor bir sınıflandırma problemidir. Çalışmada 2-2-1 yapısında biasa sahip olmayan 6 ağırlık değerine sahip XOR6, yine 2-2-1 yapısında, 6 ağırlık ve 3 bias parametresine sahip XOR9 problemi, 2-3-1 yapısında 9 ağırlık, 4 bias değerine sahip 13 parametreli XOR13 problemleri ile çalışılmıştır. Bu problemlerin parametre aralıkları XOR6, XOR9 ve XOR13 için sırasıyla [-100,100], [-10,10] ve [-10,10]'dur.

4.2. 3-Bit Parity Problemi

3-Bit parity probleminde ikili formdaki 3 adet giriş toplanarak 2 sayısına göre modu alınmaktadır. Yani girişlerdeki 1 sayısı tek ise 1, çift sayıda 1 var ise 0 değeri üretmektedir. (0 0 0;0 0 1;0 1 0;0 1 1; 1 0 0; 1 0 1; 1 1 0; 1 1 1) → (0; 1; 1; 0; 1; 0; 0; 1) . Bu problem için 3-3-1 yapısında, 12 ağırlık ve 4 bias değerine toplam 16 parametreye sahip bir yapay sinir ağı kurularak optimize edilmeye çalışılmıştır. Parametre değeri aralığı [-10,10]'dur.

4.3. 4-Bit Encoder-Decoder Problemi

Bu problemde 4 giriş, 4 çıkış bulunmaktadır. İkili formdaki 4 bitlik girişin sadece bir biti 1 olabilmektedir. Problemin çıkışı ise giriş ne ise odur. Yani (0 0 0 1;0 0 1 0;0 1 0 0;1 0 0 0)→(0 0 0 1;0 0 1 0;0 1 0 0;1 0 0 0) şeklindedir. Girişteki küçük bir değişim çıkışta küçük değişimlere yol açtığından gerçek dünya problemlerine benzer bir sınıflandırma problemidir [14]. 4-2-4 lük 16 ağırlık ve 6 biasa sahip toplam 22 parametrelik bir ağ yapısı parametreleri optimize edilmeye çalışılmaktadır. Parametre aralığı [-10,10]'dur.

Tablo 1: Çalışmada Kullanılan Problemler

Problem	Aralık	Yapı	D
XOR6	[-100,100]	2-2-1	6
XOR9	[-10,10]	2-2-1	9
XOR13	[-10,10]	2-3-1	13
3-Bit Parity	[-10,10]	3-3-1	16
4-Bit Enc.-Dec.	[-10,10]	4-2-4	22

4.4. Parametre Değerleri

Her bir algoritma 30 kez koşulmuştur ve her bir koşmada rasgele başka bir popülasyon ile başlanmıştır. Tüm yapılarda, tüm katmanlarda logaritmik sigmoid transfer fonksiyonu kullanılmıştır. Eğitim işlemi ortalama karesel hata 0.01 yada daha küçük olduğunda veya maksimum jenerasyona/çevrime erişince durdurulmuştur. Her problemin zorluğu farklı olduğu için farklı parametre setleri kullanılmıştır. Tüm problemler arasında yapıda bias kullanılmadığı için XOR6 en zor olanıdır. Bu nedenle bu problemde algoritmalar daha uzun koşulmuştur.

4.4.1. ABC Parametre Değerleri

ABC algoritmasında, limit değeri, S kaynak sayısı; D problemin boyutu olmak üzere S*D olarak belirlenmiştir Koloni boyutu (2*S) tüm problemler için 50 olarak belirlenmiştir.

4.4.2. DE Parametre Değerleri

DE algoritmasında iki çözüm arasındaki farkı ölçekleyen F değeri 0.5, popülasyondaki farklılığı sağlayan çaprazlama operatörünün değeri 0.9 olarak seçilmiştir. Bu değerler literatürde önerilen değerlerdir. Popülasyon büyüklüğü ABC de olduğu gibi 50 olarak seçilmiştir.

4.4.3. PSO Parametre Değerleri

Bilişsel (cognitive) ve sosyal bileşenler (social), sırasıyla, bireyin kendisine ve popülasyona ait bilgilerin ağırlıklandırılması için kullanılırlar. Çalışmamızda, bilişsel ve sosyal parametrelerin değerleri 1.8, parametrenin daha önceki değerinin o anki değeri etkilemesinin bir oranı olan ivme (inertia) parametresinin değeri de [16] çalışmasında önerildiği üzere 0.6 olarak seçilmiştir. Sürü 50 bireyden oluşmaktadır.

Maksimum amaç fonksiyonu değerlendirme sayısı tüm algoritmalarda XOR6, XOR9, XOR13, 3-Bit Parity ve 4-Bit Encoder-Decoder problemleri için sırasıyla 375 000, 5000, 3750, 50000 ve 50000 olarak seçilmiştir.

5. Sonuçlar

Kontrol parametrelerine bir önceki bölümde verilen değerler ile algoritmalar 30 kez koşulduğunda elde edilen istatistiksel sonuçlar Tablo 2-4' de verilmektedir. E-6 dan daha küçük değerler 0 olarak kaydedilmiştir. Tablo 2'deki ortalama MSE değerlerine göre ABC, problemlerin geneli üzerinde daha iyi sonuç vermektedir. XOR6 probleminde DE ve PSO algoritmalarının başarısız, ABC algoritmasının ise %100 başarılı olduğu görülmektedir. Tablo 3'deki ortalama jenerasyon sayılarından da DE ve PSO algoritmalarının XOR6 probleminde tüm jenerasyonları tamamladıkları halde hatayı 0.01 değerinin altına düşüremedikleri görülmektedir. Başarı oranlarına bakıldığında XOR13 ve 4-bit encoder-decoder probleminde DE algoritmasının %100, ABC algoritmasının ise tüm problemlerde %100 başarılı olduğu görülmektedir. Yani ABC algoritması her bir koşmada istenen başarı kriterini sağlamıştır. PSO algoritmasının performansının DE ve ABC algoritmalarına göre daha düşük olduğu görülmektedir. Sonuç olarak ABC algoritması incelenen problem türlerinde, yapay sinir ağlarının eğitimi konusunda DE ve PSO algoritmalarına göre daha üstün bir performans sergilemiştir.

Tablo 2: 30 koşmaya ait ortalama MSE değerleri (MMSE) ve Standard sapmaları (SDMSE)

		PSO	DE	ABC
XOR6	**MMSE**	0.097255	0.069908	0.007051
	SDMSE	0.027367	0.000000	0.002305
XOR9	**MMSE**	0.057676	0.007789	0.006956
	SDMSE	0.061530	0.009581	0.002402
XOR13	**MMSE**	0.014549	0.005293	0.006079
	SDMSE	0.024627	0.002651	0.003182
3-Bit P.	**MMSE**	0.040365	0.024716	0.006679
	SDMSE	0.049018	0.031960	0.002820
4-Bit EC	**MMSE**	0.022389	0.007677	0.008191
	SDMSE	0.031639	0.001708	0.001864

Tablo 3: 30 koşmaya ait ortalama jenerasyon/çevrim değerleri (MGC) ve Standard sapmaları (SDGC)

		PSO	DE	ABC
XOR6	**MGC**	7500	7500	2717.4
	SDGC	0	0	3.4
XOR9	**MGC**	53	31.9	32
	SDGC	39.6	23.2	0.2
XOR13	**MGC**	23.4	22.5	28.2
	SDGC	15.3	12.7	1.2
3-Bit P.	**MGC**	449.9	327.1	179
	SDGC	456.2	391.4	12.8
4-Bit EC	**MGC**	263.7	106.4	185
	SDGC	370.2	64.56	5.9

Tablo 4: 30 koşmaya ait başarı yüzdeleri

	PSO	DE	ABC
XOR6	0	0	100
XOR9	60	93	100
XOR13	93	100	100
3-Bit P.	60	77	100
4-Bit EC	80	100	100

6. Kaynakça

[1] Bonabeau, E., Dorigo, M. and Theraulaz, G., Swarm Intelligence: From Natural to Artificial Systems. New York, NY: Oxford University Press, 1999.

[2] Tereshko, V., Reaction-diffusion model of a honeybee colony's foraging behaviour, M. Schoenauer, et al, Eds., Parallel Problem Solving from Nature VI (Lecture Notes in Computer Science, Vol. 1917) Springer-Verlag: Berlin, p. 807-816, 2000.

[3] Tereshko, V. and Lee, T., How information mapping patterns determine foraging behaviour of a honey bee colony, Open Systems and Information Dynamics, v.9, 181-193, 2002.

[4] Tereshko, V. and Loengarov, A., Collective Decision-Making in Honey Bee Foraging Dynamics, Computing and Information Systems Journal, ISSN 1352-9404,Volume 9, No 3, October 2005.

[5] Abbass, H. A., MBO: marriage in honey bees optimization-a Haplometrosis polygynous swarming approach, Evolutionary Computation, Proceedings of the 2001 Congress on Volume 1, Page(s):207 – 214, 27-30 May 2001.

[6] Yang, X., Engineering Optimizations via Nature-Inspired Virtual Bee Algorithms, Lecture Notes in Computer Science, Springer-Verlag GmbH,Volume 3562, pp. 317, 2005.

[7] Karaboga, D., An Idea Based On Honey Bee Swarm For Numerical Optimization, Technical Report-TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.

[8] Basturk, B., Karaboga, D., An Artificial Bee Colony (ABC) Algorithm for Numeric function Optimization, IEEE Swarm Intelligence Symposium 2006, May 12-14, 2006, Indianapolis, Indiana, USA.

[9] Karaboga D, Basturk B, Artificial Bee Colony (ABC) Optimization Algorithm for Solving Constrained Optimization Problems, IFSA 2007, Cancun, Mexico, June, 18-21, 2007, Accepted

[10] Karaboga D, Basturk B., A Powerful And Efficient Algorithm For Numerical Function Optimization: Artificial Bee Colony (ABC) Algorithm, Journal of Global Optimization, In Press.

[11] Storn, R. and Price, K., Differential evolution -- a simple and efficient heuristic for global optimization over continuous spaces, Journal of Global Optimization, 11, 341-359, 1997.

[12] Kennedy, J. and Eberhart, R. C., Particle swarm optimization, 1995 IEEE International Conference on Neural Networks, 4, 1942-1948, 1995.

[13] Goldberg, D. E., Genetic Algorithms in Search, Optimization and Machine Learning (Addison-Wesley Pub. Co. 1989) ISBN: 0201157675

[14] Fahlman, S. , An empirical study of learning speed in back-propagation networks, Technical Report CMU-CS-88-162, Carnegie Mellon University, Pittsburgh, PA 15213, September 1988.

[15] Corne, D. and Dorigo, M. and Glover, F., New Ideas in Optimization, McGraw-Hill, 1999.

[16] Vesterstrom, J. and Thomsen, R., A Comparative Study of Differential Evolution Particle Swarm Optimization and Evolutionary Algorithms on Numerical Benchmark Problems, IEEE Congress on Evolutionary Computation (CEC'2004), 3, 1980-1987, Piscataway, New Jersey, 2004,