Natural Computing: An Overview of Particle Swarm Optimization and Differential Evolution and their Applications

Hajar AlRawi - MinnatAllah Hassan

211410810 – 213410352

Research project prepared for the following course:

Introduction to Artificial Intelligence (CS370)

Course instructor:

Dr. Sarab AlMuhaideb

March 30, 2016

Table of Contents

1. INTRODUCTION

1.1 Overview

Natural computing (NC) is the field of study that investigates both human-designed computing techniques inspired by nature and computing taking place in nature in terms of information processing [1]. It is a highly interdisciplinary field that encompasses three types of approaches: 1) *computing inspired by nature*; 2) *the simulation and emulation of natural phenomena in computing*; and 3) c*omputing with natural materials* [2]. The first class takes inspiration from nature for the development of novel problem-solving techniques to find solutions to complex problems. Next, the second approach is based on the use of computers to synthesize natural phenomena with the aim of creating patterns, forms, behaviours, and organisms that more or less resemble 'life-as-we-know-it'. Finally, the third employs natural materials such as molecules to compute, consequently may come to substitute or supplement the current silicon-based computers. Many models have been developed in each branch of study, however in this research mainly nature-inspired models of computation are inspected.

1.2 Main Concepts

In NC many thought-provoking concepts that exist in nature are emulated in computing paradigms. One such concept is *self-reproduction* which refers to the process by which organisms produce other cells and organisms of the same kind. This is particularly relevant for the creation of a machine that could replicate itself, which is what Ulam and von Neumann attempted to do [3]. Another prominent concept in natural computing is the *natural immune system* which protects the body from pathogens and toxins. Numerous functionalities allow the immune system to perform this job quickly and effectively. They are: *distinguishing self from nonself* by monitoring and classifying the antigens (protein markers) on the surface of cells; *learning* as infection by one invader may influence a change in the immune-response to a completely different invader; *memory* via an immunological memory for instance, subsequent attacks by previously encountered antigens cause existing antibodies to be dispatched promptly to neutralise the pathogen; *self-regulation* as the immune system has a self-check to prevent excessive reactions; *feature extraction* exhibited in the production of antibodies for each particular antigen; and *fault tolerance* since the immune system's protective mechanisms permit human cognitive process to continue in spite of attack to human body. An additional concept is *self-organization*. Self-organizing systems can respond to changes in the requirements and the environment without direct human intervention by changing their structure and function. This concept is particularly apparent in a *swarm* - *"*an aggregate of organisms especially when in motion" [2]. The collective behaviour of a swarm allows simple units to achieve complex processes. *Adaptability* is a feature wherein a system or process may be changed in order to work better in some situation. This concept is invoked in the study of species evolution. A further model is the compartmentalised organisation of cells [1]. The components of living cells demonstrate a hierarchy due to the presence of membranes. Rules govern the transport of materials among these constituents, and therefore display a selectivity to what molecules may

pass through them. There are countless phenomena in nature that provide insights into new possible domains in computing, the above are only a selected few.

1.3 Main Models

Several of the main concepts in NC are modelled and implemented in computing applications.

Some of the main computational models in NC are listed below under their corresponding category [1]:

a. Nature-inspired models
    1. Cellular Automata [3]
    2. Neural Computation [4]
    3. Evolutionary Computation [5]
    4. Swarm Intelligence [6]
    5. Artificial Immune Systems [7] [8]
    6. Artificial Life [9]
    7. Membrane Computing [10] [11]
    8. Amorphous Computing [12]

b. Synthetic implementation of nature
    1. Molecular Computing [13]
    2. Quantum Computing [14] [15]

c. Computing with natural materials
    1. Computational Systems Biology [16]
    2. Gene Regularity Network [17]

1.4 Report Organization

The paper presents Natural Computing as the main area of discussion. The emergence of NC is summarized in section 2.1, while the NC models which are the scope of this report are analysed in sections 2.2.1 for PSO and 2.2.2 for DE. For each model, a brief overview of the basic concepts will be presented, in addition to the main variants, implementation techniques, and applications. Section 3 of the paper includes a conclusive summery of the report in 3.1, a brief comparison of two models in 3.2, and personal views on each model's advantages and disadvantages in 3.3. Lastly, section 4 of the document includes a complete list of the references used throughout the report.

2. LITERATURE REVIEW

2.1 Emergence of NC

Biologically-inspired computing models have grown in popularity because they provide a simple yet efficient way of dealing with the complexity of natural, real-world problems. Many computer scientists seek to understand, capture, and then bring the elegance of natural systems into the artificial computing world. The following are examples of classical areas of nature-inspired computing models [1].

Since the early 1940s researchers have been interested in finding parallels and designing abstractions of natural phenomena [2]. One of the earliest examples of nature-inspired models is cellular automata. This system consist of a grid dynamic cells that update their state according to some transition rules [18].

McCulloch and Pitts proposed the first model of how a neuron processes information in 1943 [4]. The emergence of this model lead to two things: understanding living organisms' nerves system, and using that knowledge to develop intelligent machines with great computational capabilities. Fundamentally, biologists and neuroscientists have been pursuing the first goal, while the second has evolved to a computer science discipline known as artificial neural networks [19].

Inspired by the Darwinian theory of evolution, evolutionary computation has emerged as a model that suggest imitating the biological process of not a single organ, like the brain in neural networks, but rather the entire species of organisms to develop evolutionary algorithms and systems. The artificial evolutionary system evaluates an initially random population based on a fitness function, and then generates the next population base on the best parents from the current one [1] [18] [19].

Other well-known, bio-inspired models have been developed over the years to fulfil and solve certain problems such as: particle swarm optimization, that imitate the collective behaviour of swarms, artificial immune systems, molecular computing, quantum computing, and many more [1] [19].

2.2 NC Models

2.2.1 Particle Swarm Optimization

A. Biological Concept

Particle swarm optimization (PSO) is a population-based computational model inspired by the social behaviour of swarms in nature. It was developed in 1995 by social psychologist James Kennedy and electrical engineer Russel Eberhart to imitate the collective intelligence of individual animals in a swarm [20]. PSO is an example of swarm intelligence, which models the local and global interaction of decentralized, self-organized systems. Swarm intelligence simulates the collective behaviour of individual agent interacting with one another and within the populace to learn from the environment in order to find a solution. Moreover, PSO combines self-experiences of a particle with social experiences of the swarm. The algorithm operates on the principle of having a population (swarm) of candidate solutions (particles) move around the search space according to some formulas, while updating a particle's best known position and the entire swarm's best known position over time. An analogy to this algorithm might be a swarm of birds searching for food. Each bird is a particle that makes use of its own gained knowledge of the environment, as well as the knowledge gained by the swarm as a whole to gradually move the swarm around the search space towards the best food source. A bird's movement at each iteration is influenced by its current location, how close any other bird is to the food source, and the swarm's location as a whole. Even with the absence of a centralized control system to guide the particles in the cluster and govern their behaviour, particles interact locally with one another and collectively generate a complex global behaviour leading to a solution [20] [21] [22].

B. Main variants

Since particle swarm optimization is relatively new, many researchers took an interest in exploring it, and suggested several versions to improve it further. In reference [23], the variants of PSO are categorized into two groups: basic variants and modification variants. The basic variants aim to improve the speed of convergence and quality of the solution found. In addition, they are influenced by changes made to the parameters such as swarm size, dimension of the problem, acceleration coefficients, and random vectors that determine the stochastic influence, number of iterations, and the random initialization values given to each particle in the initial iteration [22]. On the other hand, modified versions of PSO are influenced by the following: changes made to the search space, adjustments made to the parameters, and hybrid with other algorithms [23].

- Basic variants:

***Velocity clamping*:** In this version, the global exploration of a particle is controlled by setting a maximum allowed speed limit for a particle (v_{max}). Without setting a limit, the velocity of a particle can grow infinitely and gradually the swarm would drift away from its optimum.

Velocity clamping does not affect the position of the particle, rather it reduces the size of the step velocity to ensure the particle remains in the search space [23] [24].

Inertia weight: This mechanism was developed with a desire to balance the exploration-exploitation abilities of a swarm. The main purpose behind introducing inertia weight was to control the initial velocity and eliminate the need of velocity clamping altogether. The inertia weight w controls the momentum of a particle by determining how much the previous velocity contributes to the current velocity. A large value of w result exploration, while exploitation is determined by a smaller value of w [25] [26] [27].

***Constriction coefficient*:** an adaptive, dynamic model that uses a new parameter X to guarantee quick convergence of the particles over time, without the need for inertia weight and maximum velocity parameters. The search scope is determined by either one of the following cases: If the previous positions of particle best and neighbour best are close, then the particle will perform a local search. However, if the two positions are far from each other, the particle will perform a global search [28] [29] [30].

- Modification variants:

Hybrid PSO: An approach that aims to combine PSO with other optimization techniques. This approach is implemented in order to get high convergence speed and accuracy. An example of a hybrid model is the combination of PSO with Back-Propagation algorithm [31] [32].

***Single solution PSO*:** static, unconstrained, single-objective problems require finding a single solution in the search space. For that, several versions of PSO were developed for the purpose of locating a single solution for this kind of problems [33].

Niching with PSO: niching refers to algorithms that locate multiple solutions to a problem in a multidimensional space. In this model, a large count of individuals compete for the use of limited resources, like in scheduling problems [34] [35] [36]

Discrete PSO: the difference between a standard PSO and a discrete PSO is the search space. The movement of a particle in a standard PSO means a change in the value of the position in one or more dimensions. On the other hand, movement in a discrete environment means a change in the probability of the position coordinate to be a zero or a one [33].

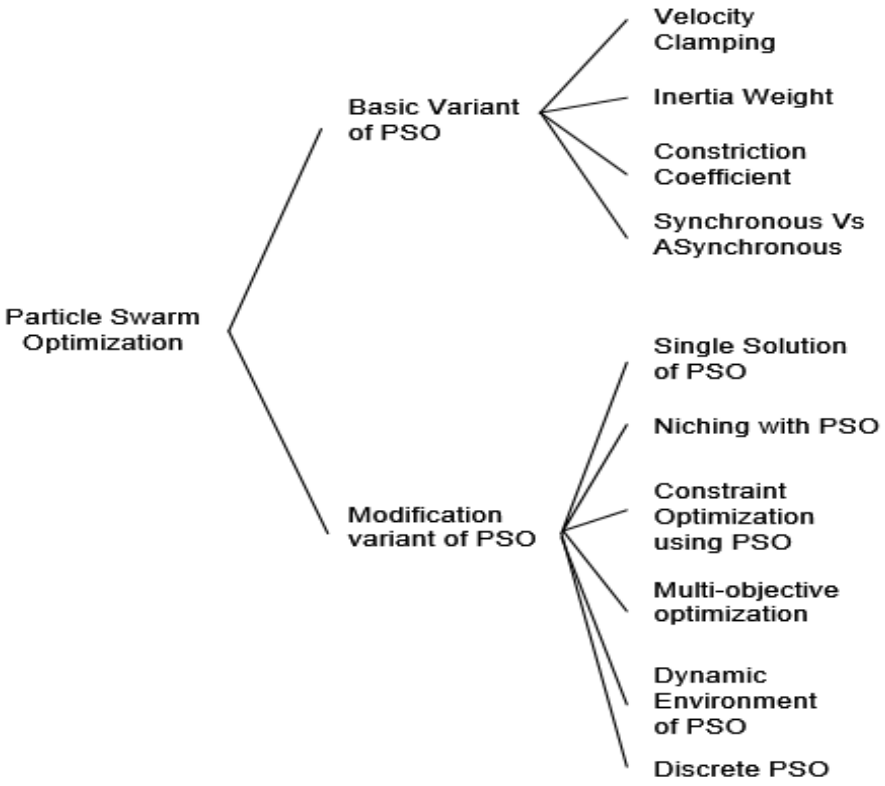

Figure 1: Variants of PSO *[23]*

C. Implementation of one variant

The standard PSO algorithm begins by creating a random set of particles, and assigning each particle an initial velocity. All particles must collectively explore the search space to locate the global optimum. Each particle relies on two sources of knowledge: a particle's memory of its own personal best position (personal best –$pBest$), and knowledge of the swarm's best position that makes it closer to its target (global best –$gBest$). A particle's position is influenced by its velocity.

We will use $x_i(t)$ to denote particle i position in the search space at time$t$. The position and velocity of a particle are updated in each iteration using the following equations [23]:

For the velocity updating formula:

$$v_i(t) = v_i(t-1) + c_1 r_1\big(pBest(t) - x_i(t-1)\big) + c_2 r_2(gBest(t) - x_i(t-1)) \quad (1)$$

For the position updating formula:

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (2)$$

Where: i – particle index, v_i – velocity of the ith particle, t – discrete time index, c_1 and c_2 – accelerating coefficients, r_1 and r_2 – random vectors, $pBest$ – best position found by the ith particle (personal best), $gBest$ – best position found by the swarm (global best, best of personal bests), and x_i – position of the ith particle.

The pseudo code of the algorithm is as follows [37]:

```
Randomly generate an initial population
FOR each particle i
   Initialized position and velocity using equations (1) and (2) respectively
END
REPEAT
       FOR EACH particle evaluate current fitness value
          If current value at time t is better than pBest value in history
                 Set current value to be the new pBest
       END
       Get the best pBest of all particles and set it as gBest
       FOR EACH particle
          Calculate velocity according to equation (1)
          Calculate new position according to equation (2)
       END
WHILE termination condition not met
```

D. Applications of PSO

#	**Application**
1	Optimization of oil and gas field development planning [38].
2	Analyse human tremor [39].
3	Register 3D to-3D biomedical image [40].
4	Playing games [41].
5	Control reactive power and voltage [42] [43].
6	Optimization of photovoltaic systems [44].
7	Project scheduling problems [45].
8	Design of Combinational Logic Circuit [46].
9	Antenna design [47].
10	Neural network training [48].

PSO wasn't designed for a particular problem and therefore can be optimized for a wide range of applications to solve a variety of problems. The paper [49] presents a taxonomy of those applications.

2.2.2 Differential Evolution

Differential Evolution (DE) is a global optimization method developed by Rainer Storn and Kenneth Price [50] in 1996. It falls under the paradigm of Evolutionary Computing. In the following, the biological analogy for the algorithm and its adaptation in computing will be examined.

A. Biological Concept

Evolutionary computation drew its inspiration from the theory of Darwinian Evolution. In his theory of evolution, Charles Darwin [51] stated that through natural selection offspring are produced that are more adapted to survive and hence more likely to produce descendants. Evolution occurs as a result of *genetic variation* and *selection.* Genetic variation refers to the diversity of gene frequencies in chromosomes located within the nuclei of cells. An organism's genotype (genetic makeup) will provide it with a selective advantage if the corresponding phenotype (physical and chemical characteristics) increases its adaptability to the environment it inhabits. The main sources of genetic variation are *reproduction* -both asexual and sexual- and *mutation*, where the latter is a deviation in the genotype causing the appearance of new traits in the offspring. Sexual reproduction results in *recombination*- crossing-over of parental genetic material causing offspring to exhibit features common to both parents, while mutation introduces new and possibly favourable characteristics in offspring. With every new generation, the more advantageous traits will become more common in the population, therefore leading to an improvement in the function of an organism. In an artificial evolutionary environment each individual is represented as a point in a search space of potential solutions to a problem. Organisms - potential solutions - are denoted as vectors and are evolved over time to explore the search space to reach an optimal solution. DE differs from other evolutionary algorithms in terms of mutation and recombination methods. As its name suggests, a *difference vector* is used to perturb the population. One or more difference vectors are scaled and added to a member in the population that further undergoes a crossover operation; epitomizing mutation and recombination respectively. This measure of dispersion characterizes DE as a fairly fast and robust optimization method.

B. Main Variants

In this section first the classical DE algorithm Storn and Price [50] developed will be considered, then the various DE schemes will be introduced.

We consider a population of $\boldsymbol{NP}$ $\boldsymbol{D}$-dimensional parameter vectors $\boldsymbol{x_i}$ for each generation $\mathbf{G}$. Where

$\boldsymbol{x_i}$ is the vector representation of individuals,

$\boldsymbol{G}$ is the generation number,

$\boldsymbol{i}$ = 1, 2 … NP are the vector indices and

$\boldsymbol{j}$ = 1, 2…D are parameter indices.

NP remains constant and must have a value of at least four. The initial vector population is chosen randomly and should cover the entire parameter space. As a rule use uniform probability distribution for all random decisions by defining the upper and lower bounds as x_j^U and x_j^L, respectively for each parameter and selecting the initial parameter values uniformly on the interval $[x_j^L, x_j^U]$.

Each vector undergoes DE's basic strategy, which is as follows:

1. **Mutation**

For each target vector a mutant vector (also known as donor vector) $v_{i,\,G+1}$ is generated using three individuals from the population through

$$v_{i,G+1} = x_{r_1,G} + F \cdot \left(x_{r_2,G} - x_{r_3,G}\right),$$

with random indexes r_1, r_2 , $r_3 \in \{1, 2 \ldots NP\}$, such that i, r_1, r_2 and r_3 are distinct - NP $\geq$ 4 to allow for this condition. F is a real and constant factor $\in$ [0, 2], which controls the amplification of the differential variation $(x_{r_2,G} - x_{r_3,G})$.

2. **Crossover / Recombination**

In order to increase diversity a trial vector $u_{ji,G+1}$ is formed which is developed from elements in the target vector $x_{i,G}$, and the elements of the mutant vector, $v_{i,G+1}$, where:

$$u_{ji,G+1} = \begin{cases} v_{ji,G+1} & if\ (randb(j) \leq CR)\ or\ j = rnbr(i) \\ x_{ji,G} & if\ (randb(j) > CR)\ and\ j \neq rnbr(i) \end{cases},$$

randb(j) is a random number generator with outcome $\in$ [0, 1].
CR is the crossover constant $\in$ [0, 1] which is the probability that elements of the mutant vector enter the trial vector and is determined by the user.
rnbr(i) is a randomly chosen index $\in$ [1, 2,..., D] which ensures that $u_{i,G+1}$ gets at least one parameter from $v_{i,G+1}$ and moreover $u_{i,G+1}$ is not equal to $x_{i,G}$.

3. **Selection**

This mimics survival of the fittest and keeps the population size constant. To determine whether or not a vector should become a member of generation G+1 the trial vector $u_{i,G+1}$ is compared to the target vector $x_{i,G}$. If vector $u_{i,G+1}$ yields a smaller cost function value than $x_{i,G}$, then $x_{i,G+1}$ is set to $u_{i,G+1}$; otherwise, the old value $x_{i,G}$ is retained.

Mutation, recombination and selection continue until some stopping criterion is reached.

In order to classify the different variants, the following notation is used:

$DE/x/y/z$

Where:

X specifies the vector to be mutated which can be "rand" (a randomly chosen population vector) or "best" (the vector of lowest cost from the current population).
Y is the number of difference vectors used for the mutation of x.
Z denotes the crossover scheme. This may be "exp" for exponential or "bin" for binomial
Using this notation, the previous DE-strategy can be written as: DE/rand/1/bin.

It is the method of creating the mutant vector as well as the type of recombination operator used, which differentiates between the various DE schemes. Some strategies Storn and Price suggested are summarized below [50]:
1. DE/best/1/exp
2. DE/rand/1/exp
3. DE/rand-to-best/1/exp
4. DE/best/2/exp
5. DE/rand/2/exp
6. DE/best/1/bin
7. DE/rand/1/bin
8. DE/rand-to-best/1/bin
9. DE/best/2/bin
10. DE/rand/2/bin

Omitting the crossover scheme (exp/bin) there are 5 basic variants that are elaborated:

1. **DE/best/1**

Very similar to 2 except that the vector with lowest cost in the population is perturbed as opposed to any random vector.

2. **DE/rand/1**

Classical variant previously explained in section 2.2.2.B.

3. **DE/rand-to-best/1**

Similar to 2, the only difference being that in the mutation stage two random vectors from the population as well as the best vector are used to generate the mutant (donor) vectors.
$v_{i,G+1} = x_{i,G} + \lambda \cdot (x_{best,G} - x_{i,G}) + F \cdot (x_{r_2,G} - x_{r_3,G})$
λ is another control parameter of DE in [0, 2]. It's usual that λ = F.

4. **DE/best/2**

Under this scheme the mutant vector is formed using two difference vectors with the best vector in the current generation.
$v_{i,G+1} = x_{best,G} + F \cdot (x_{r_1,G} + x_{r_2,G} - x_{r_3,G} - x_{r_4,G})$

5. **DE/rand/2**

Any random vector is chosen to be mutated using two difference vectors.

$v_{i,G+1} = x_{r_1,G} + F_1 \cdot (x_{r_2,G} - x_{r_3,G}) + F_2 \cdot (x_{r_4,G} - x_{r_5,G})$

F_1 and F_2 are in the range [0, 1], however it's best that $F_1 = F_2 = F$.

C. Implementation of one variant

The algorithm for the most common variant DE/rand/1/bin is as shown below:

```
Algorithm DE
Input: Randomly initialized position and velocity of the particles: xi(0)
Output: Position of the approximate global optima X

Begin Initialize population;
      Evaluate fitness;
      For i = 0 to max-iteration do
            Begin Create Difference-Offspring;
            Evaluate fitness;
            If an offspring is better than its parent
                  Then replace the parent by offspring in the next generation;
            End If;
      End For;
End.
```

D. Applications of DE

#	**Application**
1	Optimizing Si–H clusters (hydrogenated amorphous silicon) [52].
2	Non-Imaging optical design [53].
3	In industrial compressor supply systems [54].
4	To solve multi-sensor fusion problems [55].
5	Determination of the earthquake hypocenter [56].
6	3-D medical image registration [57].
7	For the design of practical and highly powerful erasure codes [58] .
8	To design digital filters [59].
9	For the optimization of radial active magnetic bearings [60].
10	To automatically and reliably analyze X-ray reflectivity (XRR) data. [61].

3. CONCLUSION

3.1 Summary

Natural computing, though a relatively new discipline has achieved breakthroughs in the field of technology. The physical world is the most complex system and attempting to model nature or create abstractions of its most influential processes unsurprisingly opens new doors for development capabilities. In this research the merits of natural computing overall were elaborated and two of its various models were investigated. Ultimately, PSO and DE are iotas in the domain of natural computing, nonetheless their merit is well deserved.

3.2 Comparison of the Models

<table>
<tr><th colspan="2" rowspan="2"></th><th>PSO</th><th>DE</th></tr>
<tr><td colspan="2">Similarities [62]</td><td colspan="2">• Both derive inspiration from the dynamics of an entire species of organisms.
• Are conceptually simple, global optimization methods, metaheuristic algorithms.
• Use primitive mathematical operations and can be implemented easily in many languages.
• Performance is not severely affected by growth of search space.</td></tr>
<tr><td rowspan="4">Differences [63] [64]</td><td>Biological Analogy</td><td>Individuals are cooperating.</td><td>Individuals are competing.</td></tr>
<tr><td>Solution Space Characteristics</td><td>The new swarm of particles is much different than old ones.</td><td>Crossover is carried out between a solution from the population and a newly generated one.</td></tr>
<tr><td>Parameter Tuning</td><td>Sensitive to parameter changes.</td><td>Same settings can be used for various problems.</td></tr>
<tr><td>Optimality of solution</td><td>Obtains good results quickly.</td><td>Obtains better results in the long term.</td></tr>
<tr><td>Performance Consistency</td><td>Dependent on the randomized initialization of individual particles.</td><td>Algorithm performs consistently, produces similar results over many trials.</td></tr>
</table>

3.3 Personal Views

PSO and DE are both very popular in the domain of machine intelligence. It is the current trend that PSO is combined with evolutionary computing techniques. The operators in DE provide more optimization opportunities if applied to particles in the PSO search space; many different approaches to doing so have been reported.

4. REFERENCES

[1] L. Kari and G. Rozenberg, "The Many Facets of Natural Computing," *Communications of the ACM,* vol. 51, no. 10, pp. 72-83, 2008.

[2] L. N. d. Castro, "Fundamentals of natural computing: an overview," *Physics of Life Review,* vol. 4, pp. 1-36, 2007.

[3] J. von Neumann and E. a. c. b. A.W.Burks, Theory of Self-Reproducing Automata, U. Illinois Press, 1966.

[4] W. McCulloch and W. Pitts, "A Logical Calculus of the Ideas ImmInent in Nervous Activity," *Bulletin of Mathematical Biophysics,* vol. 5, pp. 115-133, 1943.

[5] T. Bǎck, D. Fogel and Z. Michalewicz, Handbook of Evolutionary Computation, UK: IOP Publishing, 1997.

[6] A. Engelbrecht, Fundamentals of Computational Swarm Intelligence, Wiley and Sons, 2005.

[7] L. de Castro and J. Timmis, "Artificial Immune Systems: A New Computational Intelligence Approach," *Springer,* 2002.

[8] J. Farmer, N. Packard and A. Perelson, "The immune system adaptation and machine learning," *Physica D,* vol. 22, pp. 1887-204, 1986.

[9] C. Langton and editor, Artificial Life, Addison-Wesley Longman, 1990.

[10] G. Paun, Membrane Computing: An Introduction, Springer, 2002.

[11] G. Paun and G. Rozenberg, "A guide to membrane computing," *Theoretical Computer Science,* vol. 287, no. 1, pp. 73-100, 2002.

[12] H. Abelson, D. Allen, D. Coore, C. Hanson, H. G, T. Knight Jr., R. Nagpal, E. Rauch, G. Sussman and R. Weiss, "Amorphous Computing," *CACM,* vol. 43, no. 5, pp. 74-82, 2000.

[13] J. Adleman, "Molecular computation of solutions to combinatorial problems," *Science,* vol. 266, pp. 1021-1024, 1994.

[14] P. Kaye, R. Laflamme and M. Mosca, An Introduction to Quantum Computing, Oxford University Press , 2007.

[15] M. Hirvensalo, Quantum Computing, Springer, 2004.

[16] B. Di Ventura, C. Lemerle, K. Michalodimitrakis and L. Serrano, " From in vivo to in silico biology and back," *Nature,* vol. 443, pp. 527-533, 2006.

[17] L. Cardelli, "Machines of systems biology," *Bulletin of EATCS,* vol. 93, pp. 176-204, 2007.

[18] L. N. de Castro, Natural Computing for Simulation and Knowledge Discovery, USA: Medical Information Science Reference (IGI Global), 2013.

[19] L. N. de Castro, Fundamentals of Natural Computing: Basic Concepts, Algorithms, and Applications, Boca Raton: Chapman & Hall/CRC, 2006.

[20] R. Poli, J. Kennedy and T. Blackwell, "Particle swarm optimization: An overview," *Swarm Intell,* 2007.

[21] V. Kalivarapu, Improving solution characteristics of particle swarm optimization through the use of digital pheromones, parallelization, and graphical processing units (GPUs), 2008.

[22] S. Chavan and N. Adgokar, "An Overview on Particle Swarm Optimization: Basic Concepts and Modified Variants," *International Journal of Science and Research (IJSR),* vol. 4, no. 5, pp. 255-261, 2015.

[23] D. Rini, S. M. Shamsuddin and S. S. Yuhaniz, "Particle Swarm Optimization: Technique, System and Challenges," *International Journal of Computer Applications,* vol. 14, pp. 19-27, 2011.

[24] F. Shahzad, A. R. Baig, S. Masood, M. Kamran and N. Naveed, "Opposition-Based Particle Swarm Optimization with Velocity Clamping (OVCPSO)," *Advances in Computational Intelligence, AISC,* vol. 116, pp. 339-348, 2009.

[25] A. Chatterjee and P. Siarry, "Nonlinear inertia weight variation for dynamic adaptation in particle swarm optimization," *Computers & Operations Research,* vol. 33, no. 3, pp. 859-871, 2006.

[26] Y. Li, "Dynamic Particle Swarm Optimization Algorithm for Resolution of Overlapping Chromatograms," *Natural Computation,* vol. 3, pp. 246 - 250, 2009.

[27] P. K. Tripathi, S. Bandyopadhyay and S. K. Pal, "Multi-Objective Particle Swarm Optimization with time variant inertia and acceleration coefficients," *Information Sciences,* vol. 177, no. 22, p. 5033–5049, 2007.

[28] M. Clerc and J. Kennedy, "The particle swarm - explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation,* vol. 6, no. 1, pp. 58 - 73, 2002.

[29] R. C. Eberhar and Y. Shi, "Comparing Inertia Weights and Constriction factors in particle swarm optimization," *Evolutionary Computation,* vol. 1, pp. 84 - 88, 2000.

[30] A. Ratnaweera, S. K. Halgamuge and H. C. Watson, "Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients," *IEEE Transactions on Evolutionary Computation,* vol. 8, no. 3, pp. 240 - 255, 2004.

[31] Y. Marinakis and M. Marinaki, "A Hybrid Particle Swarm Optimization Algorithm for the Permutation Flowshop Scheduling Problem," *Optimization Theory, Decision Making, and Operations Research Applications,* vol. 31, pp. 91-101, 2012.

[32] S. Mirjalili and S. Z. M. Hashim, "A new hybrid PSOGSA algorithm for function optimization," *Computer and Information Application (ICCIA),* pp. 374 - 377, 2010.

[33] M. R. AlRashidi and M. E. El-Hawary, "Hybrid Particle Swarm Optimization Approach for Solving the Discrete OPF Problem Considering the Valve Loading Effects," *IEEE Transactions on Power Systems,* vol. 22, no. 4, pp. 2030 - 2038, 2007.

[34] A. Nickabadi, M. M. Ebadzadeh and R. Safabakhsh, "DNPSO: A Dynamic Niching Particle Swarm Optimizer for multi-modal optimization," *Evolutionary Computation,* pp. 26 - 32, 2008.

[35] C. Sun, H. Liang, L. Li and D. Liu, "Clustering with a Weighted Sum Validity Function Using a Niching PSO Algorithm," *Networking, Sensing and Control,* pp. 368 - 373, 2007.

[36] X. Li, "Niching Without Niching Parameters: Particle Swarm Optimization Using a Ring Topology," *IEEE Transactions on Evolutionary Computation,* vol. 14, no. 1, pp. 150 - 169, 2009.

[37] M. Imran, R. Hashim and N. Khalid, "An Overview of Particle Swarm Optimization Variants," *Procedia Engineering* , pp. 491-496, 2013.

[38] M. Shirangi and L. Durlofsky, "Closed-Loop Field Development Under Uncertainty by Use of Optimization With Sample Validation," *SPE Journal,* vol. 20, no. 5, pp. 908-922, 2015.

[39] R. Eberhart and X. Hu, "Human Tremor Analysis Using Particle Swarm Optimization," *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406).*

[40] M. Wachowiak, R. Smolikova, Y. Zheng, J. Zurada and A. Elmaghraby, "An Approach to Multimodal Biomedical Image Registration Utilizing Particle Swarm Optimization," *IEEE Transactions on Evolutionary Computation IEEE Trans. Evol. Computat.,* vol. 8, no. 3, pp. 289-301, 2004.

[41] L. Messerschmidt and A. Messerschmidt, "Learning to Play Games Using a PSO-Based Competitive Learning Approach," *IEEE Transactions on Evolutionary Computation IEEE Trans. Evol. Computat.,* vol. 8, no. 3, pp. 280-288, 2004.

[42] H. Yoshida, K. Kawata, Y. Fukuyama, S. Takayama and Y. Takayama, "A particle swarm optimization for reactive power and voltage control considering voltage security assessment," *IEEE Trans. Power Syst. IEEE Transactions on Power Systems,* vol. 15, no. 4, pp. 1232-1239, 2000.

[43] M. Abido, "Optimal Design of Power System Stabilizers Using Particle Swarm Optimization," *IEEE Transactions on Energy Conversion IEEE Trans. On Energy Conversion,* vol. 17, no. 3, pp. 406-413, 2002.

[44] M. Seyedmahmoudian, S. Mekhilef, R. Rahmani, R. Yusof and A. Shojaei, "Maximum power point tracking of partial shaded photovoltaic array using an evolutionary algorithm: A particle swarm optimization technique," *J. Renewable Sustainable Energy Journal of Renewable and Sustainable Energy*, vol. 6, no. 2, 2014.

[45] R. Chen, "Particle swarm optimization with justification and designed mechanisms for resource-constrained project scheduling problem," *Expert Systems with Applications,* vol. 38, no. 6, pp. 7102-7111, 2011.

[46] C. Coello and C. A., "Use of Particle Swarm Optimization to Design Combinational Logic Circuits.," in *Evolvable Systems: From Biology to Hardware*, Berlin, Springer, 2003, pp. 398-409.

[47] N. Jin and Y. Rahmat-Samii, "Advances in Particle Swarm Optimization for Antenna Designs: Real-Number, Binary, Single-Objective and Multiobjective Implementations," *IEEE Trans. Antennas Propagat. IEEE Transactions on Antennas and Propagation,* vol. 55, no. 3, pp. 556-567, 2007.

[48] M. Meissner, M. Schmuker and G. Schneider, "Optimized Particle Swarm Optimization (OPSO) and its application to artificial neural network training," *BMC Bioinformatics,* 2006.

[49] D. Sedighizadeh and E. Masehian, "Particle Swarm Optimization Methods, Taxanomy and Aplications," *Journal of Computer Theory and Engineering,* vol. 1, no. 5, pp. 1793-8201, 2009.

[50] R. Storn and K. Price, "Differential Evolution-A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces," *Journal of Global Optimization,* vol. 11, pp. 341-359, 1997.

[51] C. Darwin, The Origin of Species by Means of Natural Selection, Adamant Media Corp., 2001. Original 1859.

[52] N. Chakraborti, "Genetic Algorithms and Related Techniques for Optimizing Si-H Clusters: A Merit Analysis for Differential Evolution," in *Diferential Evolution - Natural Computing Series* , G. Rozenberg, T. Bäck, A. Eiben, K. J. N and H. Spaink, Eds., Springer Berlin Heidelberg, 2005, pp. 313-326.

[53] D. Corcoran and S. Doyle, "Non-Imaging Optical Design Using Differential Evolution," in *Diferential Evolution - Natural Computing Series*, Springer Berlin Heidelberg, 2005, pp. 327-337.

[54] "Optimization of an Industrial Compressor Supply Systems," in *Differential Evolution - Natural Computing Series*, Springer Berlin Heidelberg, 2005, pp. 339-351.

[55] R. Joshi and A. Sanderson, "Minimal Representation Multi-Sensor Fusion Using Differential Evolution," in *Differential Evolution - Natural Computing*, Springer Berlin Heidelberg, 2005, pp. 353-378.

[56] B. RĤžek and M. Kvasniþka, " Determination of the Earthquake Hypocenter: A Challenge for the Differential Evolution Algorithm," in *Differential Evolution - Natural Computing Series*, Springer Berlin Heidelberg, 2005, pp. 379-391.

[57] M. Salomon, G. Perrin, F. Heitz and J. Armspach, "Parallel Differential Evolution: Application to 3-D Medical Image Registration," in *Differential Evolution - Natural Computing*, Springer Berlin Heidelberg, 2005, pp. 393-411.

[58] A. Shokrollahi and R. Storn, "Design of Efficient Erasure Codes with Differential Evolution," in *Differential Evolution - Natural Computing Series*, Springer Berlin Heildelberg, 2005, pp. 413-427.

[59] R. Storn, "8 FIWIZ – A Versatile Program for the Design of Digital Filters Using Differential Evolution," in *Differential Evolution - Natural Computing Series*, Springer Berlin Heidelberg, 2005, pp. 429-445.

[60] G. Štumberger, D. Dolinar and K. Hameyer, "Optimization of Radial Active Magnetic Bearings by Using Differential Evolution and the Finite Element Method," in *Differential Evolution - Natural Computing Series*, Springer Berlin Heildelberg, 2005, pp. 449-462.

[61] M. Wormington, K. M. Matney and D. .. Bowen, "Application of Differential Evolution to the Analysis of X-Ray Reflectivity Data," in *Evolutionary Computing - Natural Computing Series*, Springer Berlin Heidelberg, 2005, pp. 463-478.

[62] S. Das, A. Abraham and A. Konar, "Particle Swarm Optimization and Diff erential Evolution Algorithms: Technical Analysis, Applications and Hybridization Perspectives," *Studies in Computational Intelligence (SCI)*, vol. 116, pp. 1-38, 2008.

[63] R. Ugolotti, Y. S. G. Nashed, P. Masejo, S. M. L. Ivekovic and S. Cagnoni, "Particle Swarm Optimization and Differential Evolution for model-based object detection," *Applied Soft Computing*, vol. 13, no. 6, pp. 3092-3105, 2013.

[64] J. Vesterstrom and R. Thomsen, "A Comparative Study of Differential Evolution, Particle Swarm Optimization, and Evolutionary Algorithms on Numerical Benchmark Problems," *Evolutionar Computation*, vol. 2, pp. 1980-1987, 2004.