Research Article

Physics-Inspired Optimization Algorithms: A Survey

Anupam Biswas, K. K. Mishra, Shailesh Tiwari, and A. K. Misra
Department of Computer Science & Engineering, Motilal Nehru National Institute of Technology Allahabad, Allahabad 211004, India

Correspondence should be addressed to K. K. Mishra; mishrakrishn@gmail.com

Received 7 February 2013; Revised 22 May 2013; Accepted 24 May 2013

Academic Editor: Qingsong Xu

Copyright © 2013 Anupam Biswas et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Natural phenomenon can be used to solve complex optimization problems with its excellent facts, functions, and phenomenon. In this paper, a survey on physics-based algorithm is done to show how these inspirations led to the solution of well-known optimization problem. The survey is focused on inspirations that are originated from physics, their formulation into solutions, and their evolution with time. Comparative studies of these noble algorithms along with their variety of applications have been done throughout this paper.

1. Introduction

Leonid Kantorovich introduced linear programming for optimizing production in plywood industry in 1939 and probably it was the first time the term optimization of a process was used, though Fermat and Lagrange used calculus for finding optima and Newton and Gauss proposed methods for moving towards an optimum. Every technological process has to achieve optimality in terms of time and complexity and this led the researchers to design and obtain best possible or better solutions. In previous studies, several mathematical solutions were provided by various researchers such as LP [1], NLP [2] to solve optimization problems. The complexity of the proposed mathematical solutions is very high which requires enormous amount of computational work. Therefore, alternative solutions with lower complexity are appreciated. With this quest, nature-inspired solutions are developed such as GA [3], PSO [4], SA [5], and HS [6]. These nature-inspired metaheuristic solutions became very popular as the algorithms provided are much better in terms of efficiency and complexity than mathematical solutions. Generally, these solutions are based on biological, physical, and chemical phenomenon of nature.

In this paper, the algorithms inspired by the phenomenon of physics are reviewed, surveyed, and documented. This paper mainly focuses on the following issues:

(i) most inspirational facts and phenomena,
(ii) their formulation into a solution,
(iii) parameters considered for this formulation,
(iv) effectiveness of these parameters,
(v) variation with time in inspiration,
(vi) other biological influences,
(vii) convergence, exploration, and exploitation,
(viii) Various applications.

The rest of the paper is organized as follows. Section 2 overviews the history of physics-inspired algorithms and also the description of few major algorithms. In Section 3 a correlative study of these major algorithms is done on the basis of their inspirational theory and formulation method. Various parameters used in these algorithms along with their variants and respective applications are also discussed in this section. In Section 4, finally the overall study is concluded.

2. Historical Study

by quantum mechanics in 1982, paved way for physics-inspired optimization algorithms. With this, the concept of quantum computing was developed and in 1995 Narayanan and Moore [9] proposed Quantum-Inspired Genetic Algorithm (QGA). This is the beginning of physics-inspired optimization algorithms. After half a decade later, in 2002, Han and Kim proposed Quantum-Inspired Evolutionary Algorithm (QEA). In 2004 Quantum-Inspired Particle Swarm Optimization (QPSO) was proposed by Sun et al. [10] and in 2007 another swarm-based Quantum Swarm Evolutionary Algorithm (QSE) was proposed by Wang et al. [11]. Apart from the quantum mechanics, other principles and theorems of physics also begun to draw the attention of researchers. In 2003, Birbil and Fang [12] proposed Electromagnetism-like (EM) mechanism based on the superposition principle of electromagnetism. Big Bang-Big Crunch (BB-BC) [13] based on hypothetical theorem of creation and destruction of the universe was proposed in 2005. Based on Newton’s gravitational law and laws of motion algorithms emerged such as CFO [14] by Formato in 2007, GSA by Rashedi et al. [15], APO by Xie et al. [16] in 2009, and GIO by Flores et al. [17] in 2011. Hysteretic Optimization (HO) [18] based on demagnetization process was proposed in 2008. In 2010, Kaveh and Talatahari proposed CSS [19] based on electrostatic theorems such as Coulomb’s law, Gauss’s law, and superposition principle from electrostatics and Newton’s laws of motion. In 2011, Shah-Hosseini proposed Spiral Galaxy-Based Search Algorithm (GbSA) [20], Jiao et al. [21] proposed QICA in 2008 based on quantum theory and immune system. Li et al. [22] proposed CQACO based on quantum theory and ant colony in 2010. Most recently in 2012, Zhang et al. [23] proposed IGGA based on gravitational law and immune system, and Jinlong and Gao [24] proposed QBSO based on quantum theory and bacterial forging. These major algorithms along with their modified, improved, and hybrid versions along with the year of proposal are shown in Figure 1.

We have categorized these algorithms with their variants and their notion of inspiration as follows:

(A) Newton’s gravitational law
   (i) Pure physics
   (1) CFO
      (a) Variant (pure physics)
      (1) ECFO
   (2) APO
      (a) Variant (pure physics)
      (1) EAPO
      (2) VM-APO
   (3) GSA
      (a) Variant (pure physics)
      (1) BGSA
      (2) MOGSA
      (b) Variant (semiphysics)
      (1) PSOGSA
   (4) GIO
   (ii) Semiphysics
      (1) IGOA

(B) Quantum mechanics
   (i) Pure physics
      (1) QGA
         (a) Variant (pure physics)
         (1) RQGA
         (2) QGO
      (b) Variant (semiphysics)
         (1) HQGA
      (2) QEA
         (a) Variant (pure physics)
         (1) BQEA
         (2) vQEA
         (3) IQEA
   (ii) Semiphysics
      (1) QPSO
      (2) QSE
      (3) QICA
      (4) CQACO
      (5) QBSO

(C) Universe theory
   (i) Pure physics
      (1) BB-BC
         (a) Variant (pure physics)
         (1) UBB-CBC
         (2) GbSA
3. Algorithms

3.1. Newton’s-Gravitation-Law-Based Algorithms

3.1.1. CFO. CFO [14] is inspired by the theory of particle kinematics in gravitational field. Newton’s universal law of gravitation implies that larger particles will have more attraction power as compared to smaller particles. Hence, smaller ones will be attracted towards the larger ones. As a result, all smaller particles will be attracted towards the largest particle. This largest particle can be resembled as global optimum solution in case of optimization. To mimic this concept in CFO, a set of solutions is considered as probes on the solution space. Each probe will experience gravitational attraction due to the other. Vector acceleration experienced by probe \( p \) with respect to other probes at iteration \( t \) is given by the equation below:

\[
a_p^t = G \sum_{k \neq p}^N \left( F_i^k - F_p^k \right) \cdot \left( F_i^k - F_p^k \right)^\alpha \times \frac{ \left( R_i^k - R_p^k \right)}{ \left\| R_i^k - R_p^k \right\|^\beta}.
\]  

(1)

Here, \( G \) is CFO’s gravitational constant, \( R_i^k \) and \( F_i^k \) are the position of a probe \( p \) and objective function value at that position, respectively, at iteration \( t \), \( R_p^k \) and \( F_p^k \) are the position of all other probe, and objective function value at that position, respectively, at iteration \( t \), \( U(\cdot) \) is the unit step function. The CFO exponents \( \alpha \) and \( \beta \), by contrast, have no analogues in nature but these exponents provide flexibility to the algorithm. These parameters have drastic effect on overall exploration and convergence of the algorithm. The algorithm does not have any apparent mechanism for exploitation.

In this equation, \( U(F_i^k - F_p^k) \cdot (F_i^k - F_p^k)^\alpha \) defines CFO’s mass which is analogous to real objects mass in space.

The \( a_p^t \) causes the probe \( p \) to move from position \( R_p^t \) to \( R_{p+1}^t \) and the new position is obtained by the following equation:

\[
R_{p+1}^t = R_p^t + \frac{1}{2} a_p^t \Delta t^2,
\]

(2)

Here, \( \Delta t \) is the time interval between iterations. Recently, Ding et al. proposes an extended version of CFO, namely, ECFO [25]. Applications of this algorithm are neural network [26] and antenna applications [27, 28].

3.1.2. APO. APO [16] is based on the concept of artificial physics or physicomimetics [29], which was applied to robots. Analogous to Newton’s gravitation law a new kind of force law is defined as follows:

\[
F = G \frac{m_i m_j}{r^p},
\]

(3)

where \( F \) is the force exerted between particles \( m_i \) and \( m_j \) in a hypothetical universe, \( G \) is gravitational constant, and \( r \) is the distance between particles \( m_i \) and \( m_j \). Unlike real universe, the value of \( p \) is not always equal to 2; instead it varies from −5 to +5.

Mass is defined in APO as follows:

\[
m_i = e^{f(x_{best})} f(x_i) / f(x_{best}),
\]

(4)

Considering value of \( p = -1 \) in (3), the force in APO is defined as follows:

\[
F_{ij}^k = \begin{cases} 
G m_i m_j (x_i^k - x_j^k) & \text{if } f(x_i) < f(x_j), \\
-G m_i m_j (x_j^k - x_i^k) & \text{if } f(x_i) \geq f(x_j), 
\end{cases}
\]

(5)

where \( F_{ij}^k \) is the \( k \)th component of force exerted on particle \( i \) by particle \( j \), \( x_i^k \) and \( x_j^k \) are the \( k \)th dimension of particles \( i \) and \( j \), respectively. The \( k \)th component of the total force \( F_i^k \) exerted on particle \( i \) by all other particles is given by the following:

\[
F_i^k = \sum_{j=1}^N F_{ij}^k \forall i \neq \text{best}.
\]

(6)

Velocity and positions of particles are updated with following equation:

\[
\begin{align*}
\dot{x}_i^k(t+1) &= \omega v_i^k(t) + \frac{\lambda F_i^k}{m_i}, \\
v_i^k(t+1) &= \chi_i^k(t) + v_i^k(t+1),
\end{align*}
\]

(7)

where \( \lambda \) is uniformly distributed random variable in [0, 1], \( w \) is user-defined weight \( 0 < w < 1 \).

Main exploitation and convergence component of APO algorithm is the computation of force exerted on each particle by others. Overall exploration of algorithm is controlled by the weight parameter \( w \). The parameter \( \lambda \) is actually for putting limitation to convergence. But, due to randomness, it also serves for exploration. To overcome the lack of convergence component, an extended version of APO is proposed in [30], where individual particle’s best position is tracked in iteration and utilized in velocity updating. A vector model of APO is defined in [31].

3.1.3. GSA. GSA [15] is inspired by Newton’s law of universal gravitation and law of motion. In addition to this, another fact of physics is also considered, according to which the actual value of gravitational constant \( G \) depends on the actual age of the universe. So \( G \) at time \( t \) can be expressed as follows:

\[
G(t) = G(t_0) \times \left( \frac{t_0}{t} \right)^\beta, \quad \beta < 1,
\]

(8)
where \( G(t_0) \) is the value of the gravitational constant at the first cosmic quantum-interval of time \( t_0 \), \( \beta \) is a time-dependent exponent.

In GSA, the solution space is considered as an imaginary universe. Every point in solution space is considered as an agent having mass. To compute mass of any agent \( i \), a parameter \( m_i(t) \) is computed. The parameter \( m_i(t) \) and mass \( M_i(t) \) of agent \( i \) are computed as follows:

\[
m_i(t) = \frac{\text{fit}_i(t) - \text{worst}(t)}{\text{best}(t) - \text{worst}(t)},
\]

\[
M_i(t) = \frac{m_i(t)}{\sum_{j=1}^{N} m_j(t)},
\]

where \( \text{fit}(t) \) is the fitness value of the agent \( i \) at time \( t \).

Force exerted on each considered agent is computed as follows:

\[
F_{ij}^d = G(t) \frac{M_{ia}(t) \times M_{aj}(t)}{R_{ij}(t) + \varepsilon} \left( x_{id}^d(t) - x_{jd}^d(t) \right),
\]

\[
R_{ij} = \|X_i(t) - X_j(t)\|_2,
\]

\[
F_i^d(t) = \sum_{j=1,j \neq i}^{N} \text{rand}_j F_{ij}^d(t),
\]

where \( F_{ij}^d \) is the force acting on mass \( M_i \) and mass \( M_j \) at time \( t \), \( M_{ia} \) is the active gravitational mass related to agent \( j \), \( M_{aj} \) is the passive gravitational mass related to agent \( i \), \( G(t) \) is gravitational constant at time \( t \), \( \varepsilon \) is a small constant, \( R_{ij}(t) \) is the Euclidian distance between two agents \( i \) and \( j \), \( F_i^d(t) \) is the total force that acts on agent \( i \) in a dimension \( d \) at time \( t \), and \( \text{rand}_j \) is a random number in the interval \([0, 1]\).

Acceleration of any agent \( i \) at time \( t \) in direction \( d \) is computed with equation given below:

\[
d_i^d(t) = \frac{F_{ij}^d}{M_i(t)}.
\]

The next position of each agent and at which velocity they will move is calculated as follows:

\[
v_i^d(t+1) = \text{rand}_i \times v_i^d(t) + d_i^d(t),
\]

\[
x_i^d(t+1) = x_i^d(t) + v_i^d(t + 1).
\]

The concept of variable gravitational constant \( G \) provides a good mechanism for convergence to the algorithm. As in subsequent iterations the value of \( G \) gradually increases, attraction force experienced by each agent also increases. Thus, agents converge towards the better agents with incremental attraction. However, the effect of attraction force is controlled by a random parameter. This random control of force ensures exploitation as well as exploration. Another random parameter used in velocity updating also implies exploration of search space.

In [32] binary version of GSA is proposed, multiobjective GSA [33] is proposed by Mirjalili and Hashim, and a hybrid of PSO and GSA is proposed in [34].

Applications of GSA algorithm are in power system [35–42], economic dispatch problem [43, 44], Wessinger’s equation [45], fuzzy system [46, 47], forecasting of oil demand [48], slope stability analysis [49], clustering [50–52], prototype classification [53], feature selection [54], web services [55], PID controller [56], antenna application [47], and so forth.

3.1.4. IGOA. IGOA [23] algorithm is an improved version of GSA [15]. The gravitation-law-based algorithm GSA can easily fall into local optimum solution and convergence rate is also comparatively slow [57]. To overcome these problems, IGOA introduces new operators which are inspired from biological immune system (BIS) [58]. In BIS mainly two kinds of activities take place, activities of antibodies and activities of antibody. An antigen can only be negotiated with corresponding right antibody which comes from mother during birth. But for an unknown antigen, BIS also can act accordingly by learning. That means BIS has immune memory and antibody diversity. IGOA mimics this mechanism to avoid falling into local optimum. In this case, local optimum is similar to the unknown antigen in BIS. In IGOA vaccination and memory antibody replacement is used to improve the convergence speed and antibody diversity mechanism to catch the diversity of solution space along with GSA. IGOA is a newly proposed algorithm and not yet applied in any real-life application.

3.1.5. GIO. GIO [17] algorithm is similar to GSA [15] and CSS [19] where each point in the search space is assigned mass and charges, respectively. Although perspective of assigning masses or charges to each point is similar, the way of assignment and notion is different. CSS is inspired from electrostatic dynamics law, whereas GSA is inspired from Newton’s gravitational laws and laws of motion. GIO is also inspired from Newton’s law but unlike GSA this algorithm keeps hypothetical gravitational constant \( G \) as constant. Force exerted between two bodies is computed as follows:

\[
F_{ij} = \frac{M(f(B_i)) \cdot M(f(B_j))}{|B_i - B_j|^2} \vec{B}_{ij},
\]

where \( B_i \) is the position of the \( i \)th body and \( B_j \) is that of \( j \)th body, \( F_{ij} \) is exerting force on the mass \( B_i; |B_i - B_j| \) is the Euclidean distance and \( \vec{B}_{ij} \) is the unit vector between bodies \( B_i \) and \( B_j \); \( f(B_i) \) is the fitness of body \( B_i \), \( M(f(B_i)) \) is corresponding mass of body \( B_i \) and is computed as follows:

\[
M(f(B_i)) = \left( \frac{f(B_i) - \min f(B)}{\max f(B) - \min f(B)} \right) \times (1 - \text{mapMin}) + \text{mapMin},
\]

where \( \min f(B) \) is the minimum fitness value of the positions of the bodies so far, \( \max f(B) \) is the maximum fitness value of the positions of the bodies so far. mapMin is a constant used to limit the fitness value \( f(B_i) \) to a mass in the interval
3.2. Quantum-Mechanics-Based Algorithms

3.2.1. QGA

According to quantum mechanics, electrons are moving around the nucleus in an arc path, known as orbits. Depending on the angular momentum and energy level, electrons are located in different orbits. An electron in lower level orbit can jump to higher level orbit by absorbing certain amount of energy; similarly higher level electron can jump to lower energy level by releasing certain amount of energy. This kind of jumping is considered as discrete. There is no intermediate state in between two energy levels. The position where an electron lies on the orbit is unpredictable; it may lie at any position in orbit at a particular time. Unpredictability of electron’s position is also referred as superposition of electron.

In classical computing, a bit is represented either by 0 or 1, but in quantum computing this is termed as qubit. State of a qubit can be 0 or 1 or both at the same time in superposition state. This superposition of qubit mimics the superposition of electrons or particles. State of qubit at any particular time is defined in terms of probabilistic amplitudes. The position of an electron is described in terms of qubits by a vector called quantum state vector. A quantum state vector can be described with the equation given below:

$$|\Psi\rangle = \alpha |0\rangle + \beta |1\rangle,$$

where $\alpha$ and $\beta$ are complex numbers that specify the probability amplitudes of obtaining the qubit in “0” state and in “1” state, respectively. In this case, the value of $\alpha$ and $\beta$ always satisfies the equation $|\alpha|^2 + |\beta|^2 = 1$. For $n$ positions of electrons, states can be described by $n$ state vectors. These $n$ positions of an electron can be known simultaneously.

QGA [9] utilized the concept of parallel universe in GA [3] to mimic quantum computing. According to this parallel universe interpretation, each universe contains its own version of population. All populations follow the same rules, but one universe can interfere in population of other universe. This interference occurs as in the form of a different kind of crossover called interference crossover, which provides good exploration capability to the algorithm. In QGA, all the solutions are encoded using superposition and all of these solutions may not be valid, which creates problems during implementation of crossover. Udrescu et al. propose RQGA [62], which provides a mechanism to overcome this problem. Hybrid versions [63] merge QGA with permutation-based GA and [64] merge QGA with real-valued GA. Malossini and Calarco propose QGOA [65] very similar to QGA with special quantum-based selection and fitness evaluation methods.

Many applications have been developed in recent years on the basis of this algorithm such as structural aligning [66], clustering [67, 68], TSP [69], combinatorial optimization problem [70], web information retrieval [71], computational grid [72], software testing [73], dynamic economic dispatch [74], area optimization [75], operation prediction [76], computer networking [77, 78], PID controller [79], multivariate problem [80], course timetabling [81], minimal readact [82], image applications [83–86], smart antenna [87], hardware [88], fuzzy system [89, 90], neural network [91], and robot application [92].

3.2.2. QEA

Quantum bit and superposition of states are the main basis of this algorithm. QEA [93] is originally
inspired by quantum computing, which itself is inspired by the quantum mechanics. In QEA, the state of a qubit or Q-bit is represented as pair of numbers \((\alpha, \beta)\) in a column matrix \[
\begin{bmatrix}
\alpha_1 \\
\beta_1 \\
\vdots \\
\alpha_n \\
\beta_n
\end{bmatrix},
\]
where \(|\alpha|^2 + |\beta|^2 = 1\) and \(|\alpha|^2\) gives the probability that the Q-bit will be found in the “0” state and \(|\beta|^2\) gives the probability that the Q-bit will be found in the “1” state.

A Q-bit individual which is a string of Q-bits is defined as follows:
\[
\begin{bmatrix}
\alpha_1 \\
\beta_1 \\
\vdots \\
\alpha_m \\
\beta_m
\end{bmatrix},
\tag{22}
\]
where \(|\alpha_i|^2 + |\beta_i|^2 = 1\), \(i = 1, 2, \ldots, m\). With this Q-bit representation, a population set is formulated and operations are performed on that population. Zhang and Gao further improved this algorithm as IQEA [94], by introducing probability amplitude ratio \(\gamma_i = |\beta_i|/|\alpha_i|\) if \(\alpha_i \neq 0\) and \(\gamma_i = |\alpha_i|/|\beta_i|\) if \(\beta_i \neq 0\) to define relative relationship between \(\alpha\) and \(\beta\). As quantum rotation gate is unable to cover the entire search space since it outputs discrete values, a mechanism for calculating rotation angle of quantum rotation gate is defined. Platel et al. propose versatile QEA [95], with introducing new concept of hitchhiking phenomenon inspired by quantum computing, which itself is inspired by quantum systems [93].

Applications of QEA-related algorithms are combinatorial optimization [97, 98], image segmentation [99], Knapsack Problems [100–102], resource optimization [103, 104], numerical optimization [105, 106], extrusion [107], unit commitment problem [108, 109], power system [110, 111], signaling [112], face identification [113, 114], financial data analysis [115], Option pricing model calibration [116, 117], stock market prediction [118], and so forth.

3.2.3. QSE. QSE [11] takes the concept from both QEA [93] and PSO [4]. Similar to PSO’s swarm intelligent concept, quantum swarms are represented using Q-bits. Unlike QEA, representation of Q-bit in QSE changes probabilistic parameters. \(\alpha\) and \(\beta\) are replaced with angular parameters \(\sin\theta\) and \(\cos\theta\), where \(\theta\) is quantum angle. Q-bit \(|\theta\rangle\) is represented as \[
\begin{bmatrix}
\sin\theta \\
\cos\theta
\end{bmatrix},
\]
where \(|\sin\theta|^2 + |\cos\theta|^2 = 1\). For \(m\) Q-bits, this can be represented as \[
\begin{bmatrix}
\sin\theta_1 \\
\cos\theta_1 \\
\vdots \\
\sin\theta_m \\
\cos\theta_m
\end{bmatrix}.
\]
Each bit position \(x^t_{ij}\) of each individual, at time \(t\), is determined by the following
\[
x^t_{ij} = \begin{cases} 
1 & \text{if random } [0, 1] > |\cos\theta_{ij}|^2 \\
0 & \text{otherwise} 
\end{cases} \tag{23}
\]
Velocity is updated as in PSO. Another quantum-swarm-based PSO called QPSO was proposed by Sun et al. [10]. Unlike QSE state of particle is not determined by the probabilistic angular parameters. Here, state of particle is determined by a wave function \(\Psi(x)\) as follows:
\[
\Psi(x) = \frac{1}{\sqrt{L}} e^{-\langle k - x | L \rangle}. \tag{24}
\]
Here, \(c\) and \(x\) are the center or current best and current location vector, \(L\) is called creativity or imagination parameter of particle. Location vector is defined as:
\[
x(t) = c \pm \frac{L}{2} \ln \left( \frac{1}{R} \right). \tag{25}
\]
Here, \(R\) is a random number in range \([0, 1]\). The creativity parameter \(L\) is updated as follows:
\[
L(t + 1) = 2 \times \alpha \times |c - x(t)|. \tag{26}
\]
Here, \(\alpha\) is the creative coefficient and acts as main ingredient for convergence towards the optima. Huang et al. [119] have improved this later on by considering global best instead of current best.

Applications of these algorithms are flow shop scheduling [120], unit commitment problem [121, 122], neural network [123], power system [124–126], vehicle routing problem [127–129], engineering design [130, 131], mining association rules [132], and so forth.

3.2.4. QICA. Basic concept of QICA [21] is Artificial Immune System’s clonal selection, which is hybridized with the framework of quantum computing. Basic quantum representation aspect is similar to QEA [93]. QICA introduces some new operators to deal with premature convergence and diverse exploration. The clonal operator \(\Theta\) is defined as follows:
\[
\Theta(Q(t)) = \left[ \Theta(q_1) \Theta(q_2) \cdots \Theta(q_n) \right]^T, \tag{27}
\]
where \(Q(t)\) is quantum population and \(\Theta(q_i) = I_q q_i\), \(I_q\) is the identity matrix of dimensionality \(D_q\), which is given by the following:
\[
D_q = I_q = \left[ N_c \times \frac{A(q_i)}{\sum_{i=1}^n A(q_i)} \right]. \tag{28}
\]
Here, \(A()\) is function for adaptive self-adjustment and \(N_c\) is a given value relating to the clone scale. After cloning these are added to population.

The immune genetic operator consists of two main parts, that is, quantum mutation and recombination. Before performing quantum mutation, population is guided towards the best one by using following equation:
\[
\begin{bmatrix}
\alpha^p_{i1} \\
\beta^p_{i1}
\end{bmatrix} = U(\theta_i) \times \begin{bmatrix}
\alpha^o_{i1} \\
\beta^o_{i1}
\end{bmatrix}, \tag{29}
\]
where \(\alpha^o_{i1}\) and \(\beta^o_{i1}\) are updated values, \(\alpha^p_{i1}\) and \(\beta^p_{i1}\) are previous values of probabilistic coefficients, \(U(\theta_i) = \begin{bmatrix} n \sin \theta_i \sin \delta_i \\
\cos \theta_i \cos \delta_i \end{bmatrix}\) is quantum rotation gate and \(\theta_i\) is defined as follows:
\[
\theta_i = k \times f(\alpha_i, \beta_i), \tag{30}
\]
where \(k\) is a coefficient which determines the speed of convergence and the function \(f(\alpha_i, \beta_i)\) determines the search
direction. This updated population is mutated using quantum NOT gate as:

\[ Q^x(t) = \sqrt{1 - |Q^x(t)|^2}. \]  

(31)

Quantum recombination is similar to interference crossover in QGA [9]. Finally, the clonal selection operator selects the best one from the population observing the mutated one and original population. Clonal operator of QICA increases exploitative power drastically in contrast to QEA.

3.2.5. CQACO. CQACO [22] merges quantum computing and ACO [133]. Ant’s positions are represented by quantum bits. This algorithm also represents qubits similar to QEA [93] and QICA [21] and uses the concept of quantum rotation gate as in QICA. Similar to CQACO, Wang et al. [134] proposed quantum ant colony optimization. Another variant is proposed by You et al. [135] in 2010. Quantum concept with ACO provides good exploitation and exploration capability to these algorithms. Applications of these algorithms are fault diagnosis [136], robot application [137], and so forth.

3.2.6. QBSO. QBSO [24] is the newest among all the quantum-based algorithms. This algorithm is semiphysics-inspired, as it incorporates concepts of both bacterial forging and quantum theory. In other words, QBSO is an improved version of BFO [138]. As BFO is unable to solve discrete problems, QBSO deals with this problem by using quantum theory to adapt the process of the BFO to accelerate the convergence rate. BFO consists of chemotaxis, swarming, reproduction, elimination, and dispersal processes whereas QBSO consists mainly of three of them, chemotaxis, reproduction, and elimination dispersal. In QBSO also the qubit is defined as in (21). Quantum bacterium of S bacteria is represented in terms of the three processes, that is, bit position, chemotactic step, and reproduction loop.

The ith quantum bacterium’s quantum jth bit at the nth chemotactic step of the nth reproduction loop in the pth elimination dispersal event is updated as follows:

\[ \Psi_{ij}(m+1, n, p) = \begin{cases} \sqrt{1 - (\Psi_{ij}(m, n, p))^2}, & \text{if } \theta_{ij}^{p+1} = 0 \text{ and } \rho(m+1, n, p) < c_1 \\ \text{abs} \left( -\sqrt{1 - (\Psi_{ij}(m, n, p))^2} \times \sin \theta_{ij}^{p+1} \right), & \text{otherwise,} \end{cases} \]  

(32)

where \( \theta_{ij}^{p+1} \) is the quantum rotation angle, which is calculated through (33), \( t \) is the iteration number of the algorithm, \( \rho_{ij} \) is uniform random number in range [0, 1], and \( c_1 \) is mutation probability which is a constant in the range [0, 1].

After updating quantum bacterium, the corresponding bit position in the population \( x_{ij} \) is updated with (34), where \( y_{ij} \) is uniform random number between 0 and 1:

\[ y_{ij} = e_1 \left( b_j (m, n, p) - x_{ij} (m, n, p) \right). \]  

(33)

Here, \( e_1 \) is attracting effect factor and \( b_j \) is the jth bit position of global optimal bit:

\[ x_{ij} (m+1, n, p) = \begin{cases} 1, & \text{if } y_{ij} (m+1, n, p) > (\Psi_{ij}(m+1, n, p))^2 \\ 0, & \text{if } y_{ij} (m+1, n, p) \leq (\Psi_{ij}(m+1, n, p))^2. \end{cases} \]  

(34)

Fitness value of each point solution in population is represented as the health of that particular bacterium.

3.3. Universe-Theory-Based Algorithms

3.3.1. BB-BC. BB-BC [13] algorithm is inspired mainly from the expansion phenomenon of Big Bang and shrinking phenomenon of Big Crunch. The Big Bang is usually considered to be a theory of the birth of the universe. According to this theory, all space, time, matter, and energy in the universe were once squeezed into an infinitesimally small volume and a huge explosion was carried out resulting in the creation of our universe. From then onwards, the universe is expanding. It is believed that this expansion of the universe is due to Big Bang. However, many scientists believe that this expansion will not continue forever and all matters would collapse into the biggest black hole pulling everything within it, which is referred as Big Crunch.

BB-BC algorithm has two phases, namely, Big Bang phase and Big Crunch phase. During Big Bang phase, new population is generated with respect to center of mass. During Big Crunch phase, the center of mass is computed which resembles black hole (gravitational attraction). Big Bang phase ensures exploration of solution space. Big Crunch phase fulfills necessary exploitation as well as convergence.

BB-BC algorithm suffers botching all candidates into a local optimum. If a candidate with best fitness value converges to an optima at the very beginning of the algorithm, then all remaining candidates follow that best solution and trapped into local optima. This happens because the initial population is not uniformly distributed in the solution space. So, this algorithm provides a methodology to obtain uniform initial population in BB-BC. Initially, that is, at level 1, two candidates \( C_1 \) and \( C_2 \) are considered; at level 2, \( C_1 \) and \( C_2 \) are subdivided into \( C_3 \) and \( C_4 \); at level 3, \( C_1 \) and \( C_2 \) are again divided into \( C_5 \), \( C_6 \), \( C_7 \), \( C_8 \), \( C_9 \), and \( C_{10} \) and so on. This kind of division continues until we get the required numbers of candidates for initial population. In this way at nth level, \( C_1 \) and \( C_2 \) are subdivided into \( 2^n - 2 \) candidates and include in population. In addition to this, in Big Crunch phase chaotic map is introduced, which improves convergence speed of algorithm. In this Chaotic Big Crunch phase, next position of each candidate is updated as follows:

\[ x_i^{\text{new}} = x_i \pm \frac{\alpha(t)(x_{\text{max}} - x_{\text{min}})}{t}, \]  

(35)
where \( \alpha_{t+1} = cf(\alpha(t)), 0 < \alpha(t) < 1 \), here \( cf(x) \) is a chaotic map or function. BB-BC with uniform population is called UBB-BC and with chaotic map is called BB-CBC. If both are used, then it is called UBB-CBC.

Applications of this algorithm are fuzzy system [139–141], target tracking [142, 143], smart home [144], course timetabling [145], and so forth.

3.3.2. GbSA. GbSA [20] is inspired by spiral arm of spiral galaxies to search its surrounding. This spiral movement recovers from botching into local optima. Solutions are adjusted with this spiral movement during local search as well. This algorithm has two components:

1. SpiralChaoticMove,
2. LocalSearch.

SpiralChaoticMove actually mimics the spiral arm nature of galaxies. It searches around the current solution by spiral movement. This kind of movement uses some chaotic variables around the current best solution. Chaotic variables are generated with formula \( x_{mr1} = \lambda x_r (1 - x_r) \). Here, \( \lambda = 4 \) and \( x_r = 0.19 \). In this way, if it obtains a better solution than the current solution, it immediately updates and goes for LocalSearch to obtain more suitable solution around the newly obtained solution. GbSA is applied to Principle Component Analysis (PCA). LocalSearch ensures exploitation of search space and SpiralChaoticMove provides good exploration mechanism of search space ensuring reachability of algorithm towards the global optimum solution.

3.4. Electromagnetism-Based Algorithms

3.4.1. Electromagnetism-Like: EM. EM [12] algorithm is based on the superposition principle of electromagnetism, which states that the force exerted on a point via other points is inversely proportional to the distance between the points and directly proportional to the product of their charges. Points in solution space are considered as particles. The charge of each point is computed in accordance with their objective function value. In classical physics, charge of a particle generally remains constant, but in this heuristic the charge of each point is not constant and changes from iteration to iteration. The charge \( q^i \) of each point \( i \) determines its power of attraction or repulsion. This charge of a particle is evaluated as follows:

\[
d^i = e^{-n((f(x^i) - f(x^{best}))/\sum_{j=1}^{m}(f(x^j) - f(x^{best})))}, \quad \forall i, \quad (36)
\]

where \( m \) is the total number of points and \( n \) is the number of dimensions. This formula shows that points having better objective values will possess higher charges. This heuristic does not use signs to indicate positive or negative charge as in case of electric charge. So, direction of force (whether attractive or repulsive force) is determined by the objective function values (fitness) of two particular points. If point \( x^j \) has better value than \( x^i \), then corresponding force is considered as attractive otherwise repulsive. That means that \( x^{best} \) attracts all other points towards it. The total force \( F^i \) (attractive or repulsive) exerted on point \( i \) is computed by the following equation:

\[
F^i = \sum_{j \neq i}^{m} \begin{cases} 
\left( x^j - x^i \right) \frac{q^j q^i}{\|x^j - x^i\|^2}, & \text{if } f(x^j) < f(x^i) \ , \forall i. \\
\left( x^i - x^j \right) \frac{q^i q^j}{\|x^j - x^i\|^2}, & \text{if } f(x^i) \geq f(x^j). 
\end{cases}
\]

After evaluating the total force vector \( F^i \), the point \( i \) is moved in the direction of the force with a random step length as given in (38). Here, RNG is a vector whose components denote the allowed feasible movement towards the upper bound or the lower bound

\[
x^j = x^j + \lambda \frac{F^i}{\|F^i\|} \text{(RNG)} \quad (38)
\]

EM algorithm provides good exploration and exploitation mechanism with computation of charge and force. Exploration and convergence of EM are controlled by the random parameter \( \lambda \). Exploration is also controlled with RNG, by limiting movements of particles.

Debels et al. [146] propose a hybrid version of EM combining the concept of GA with EM.

Numerous applications are developed on the basis of this algorithm such as scheduling problems [147–150], course timetabling [151], PID controller [152], fuzzy system [153–155], vehicle routing problem [156], networking [157], inventory control [158], neural network [159, 160], TSP [161, 162], feature selection [163], antenna application [164], robotics application [165], flow path designing [166], and vehicle routing [167].

3.5. Glass-Demagnetization-Based Algorithms

3.5.1. HO. HO [18] is inspired by the demagnetization process of a magnetic sample. A magnetic sample comes to a very stable low-energy state called ground state, when it is demagnetized by an oscillating magnetic field of slowly decreasing amplitude. After demagnetization, the system are shakeup repeatedly to obtain improved result. HO simulates these two processes of magnetic sample to get low-energy state by repeating demagnetization followed by a number of shakeups.

The process of demagnetization is mainly for exploration and convergence. After exploring, better solutions are searched by performing a number of shake-up operations. The algorithm possesses two kinds of stopping conditions, firstly fixed number of shakeups \( n_s \) for each instance of a given size \( N \) and secondly required number of shakeups \( M_{req} \) to obtain the current low-energy state or global optimum. Besides this \( M_{req} \) repetition of shakeup in current low-energy state, minimum number of shakeups \( n_{s_{min}} \) is set to ensure that
algorithm does not accept suboptimum too early. Similarly, the maximum number of shakeups \( n_{\text{max}} \) is also set to avoid wasting time on hard situation. HO algorithm is applied to TSP [168], spin glasses [169], vehicle routing problem [170], protein folding [171], and so forth.

### 3.6. Electrostatics-Based Algorithms

#### 3.6.1. CSS

This algorithm inherits Coulomb’s law, Gauss’s law and superposition principle from electrostatics, and the Newtonian laws of mechanics. CSS [19] deploys each solution as a Charged Particle (CP). If two charged particles having charges \( q_i \) and \( q_j \) reside at distance \( r_{ij} \), then according to Coulomb’s law electric force \( F_{ij} \) exerted between them is as follows:

\[
F_{ij} = k_e \frac{q_i q_j}{r_{ij}^2}.
\]

Here, \( k_e \) is a constant called the Coulomb constant. Now if \( q_i \) amount of charges is uniformly distributed within a sphere of radius “a” then electric field \( E_{ij} \) at a point outside the sphere is as follows:

\[
E_{ij} = k_e \frac{q_i}{a^2 r_{ij}}.
\]

The electric field at a point inside the sphere can be obtained using Gauss’s law as follows:

\[
E_{ij} = k_e \frac{q_i}{a^2 r_{ij}}.
\]

The resultant force on a charge \( q_j \) at position \( r_j \) due to the electric field of a charge \( q_i \) at position \( r_i \) can be expressed in vector form as:

\[
F_{ij} = \frac{r_i - r_j}{r_i - r_j}.
\]

For multiple charged particles, this equation can be expressed as follows:

\[
F_j = k_e \sum_{i,j \neq j} \left( \frac{q_i q_j}{a^2 r_{ij}} \cdot i_1 + \frac{q_i q_j}{r_{ij}^2} \cdot i_2 \right) \frac{r_i - r_j}{r_i - r_j} + \frac{r_i - r_j}{r_i - r_j} \left( \frac{1}{r_i} = 1, i_2 = 0 \iff r_{ij} < a \right)
\]

\[
\frac{r_i - r_j}{r_i - r_j} \left( \frac{1}{r_i} = 0, i_2 = 1 \iff r_{ij} \geq a. \right.
\]

To formulate the concept of charges into CSS algorithm, a set of charged particles are considered. Each point in the solution space is considered as possible positions of any charged particle. Charges in each charged particle is computed as follows:

\[
q_i = \frac{\text{fit} (i) - \text{fitworst}}{\text{fitbest} - \text{fitworst}}.
\]

Distance among particles is computed with the following:

\[
r_{ij} = \frac{|X_i - X_j|}{\| (X_i + X_j) / 2 - X_{\text{best}} \|^2 + \varepsilon}.
\]

Radius of particle is computed with

\[
a = 0.10 \times \max \{ x_{i, \text{max}} - x_{i, \text{min}} \mid i = 1, 2, \ldots, n \}.
\]

The value of the resultant electrical force acting on a charged particle is determined as follows:

\[
F_j = q_j \sum_{i,j \neq j} \left( \frac{q_i}{a^2 r_{ij}^2} \cdot i_1 + \frac{q_i}{r_{ij}^2} \cdot i_2 \right) P_j (X_i - X_j).
\]

Here, \( P_j \) defines the attractiveness or repulsiveness of the force exerted. A good particle may attract a bad one and similarly bad one can also attract good one. So, if bad one attracts good one, then it is not suitable for an optimization problem. The parameter \( P_j \) limits these kinds of attractions as follows:

\[
P_j = \begin{cases} 
1, & \text{if } \left( \frac{\text{fit} (i) - \text{fitworst}}{\text{fitbest} - \text{fitworst}} \right) > \text{rand} \\
0, & \text{otherwise.} 
\end{cases}
\]

Again in Newtonian mechanics or classical mechanics the velocity \( v \) of a particle is defined as follows:

\[
v = \frac{r_{\text{new}} - r_{\text{old}}}{t_{\text{new}} - t_{\text{old}}} = \frac{r_{\text{new}} - r_{\text{old}}}{\Delta t}.
\]

Displacement from \( r_{\text{old}} \) to \( r_{\text{new}} \) position along with acceleration \( a \) can be expressed as follows:

\[
r_{\text{new}} = \frac{1}{2} a \cdot \Delta t^2 + v_{\text{old}} \cdot \Delta t + r_{\text{old}}.
\]

Newton’s second law states that “the acceleration of an object is directly proportional to the net force acting on it and inversely proportional to its mass;” that is, \( F = ma \), so \( r_{\text{new}} \) can be expressed as follows:

\[
r_{\text{new}} = \frac{1}{2} F \cdot \Delta t^2 + v_{\text{old}} \cdot \Delta t + r_{\text{old}}.
\]

In CSS movements due to the electric force exerted among those particles are measured and accordingly new positions of particles are updated. New position \( (X_{j, \text{new}}) \) of CP and with which velocity \( (V_{j, \text{new}}) \) will reach the position \( (X_{j, \text{old}}) \) is computed as follows:

\[
X_{j, \text{new}} = \text{rand}_{j} \cdot k_a \cdot \frac{F_j}{m_j} \cdot \Delta t^2 + \text{rand}_{\mu} \cdot k_v \cdot \Delta t + X_{j, \text{old}},
\]

\[
V_{j, \text{new}} = \frac{X_{j, \text{new}} - X_{j, \text{old}}}{\Delta t}.
\]

Here, \( k_a \) is the parameter related to the attracting forces and \( k_v \) is velocity coefficient. The effect of the pervious velocity and the resultant force acting on a charged particle can be
decreased with parameter $k_v$ or increased with parameter $k_a$. These parameters can be computed as follows:

\[ k_v = 0.5 \left( 1 - \frac{\text{iter}}{\text{iter}_{\text{max}}} \right), \]
\[ k_a = 0.5 \left( 1 + \frac{\text{iter}}{\text{iter}_{\text{max}}} \right), \]

where \( \text{iter} \) is the current iteration number and \( \text{iter}_{\text{max}} \) is the maximum number of iterations.

The CSS algorithm possesses good exploring as well as exploiting capability of solution domain. Exploitation of CP is mainly ensured by the resulting electric force $F_j$ of any particle $j$. Handling of attractiveness and repulsiveness of resulting force of any CP with the noble concept of parameter $P_{ij}$ is very effective for exploitation. However, whether CP is going to explore or exploit the search space depends on the parameters $k_v$ and $k_a$. Higher value of $k_v$ implies higher impact on resulting electric force, which results exploitation of search space. Whereas higher value of $k_a$ implies high exploration. Initially, values of $k_v$ and $k_a$ are almost same, but gradually $k_a$ increases and $k_v$ decreases. Hence, at the beginning, the algorithm explores the search space. As in successive iterations $k_a$ increases, gradually the effect of attraction of good solutions also increases. Thus, the algorithm ensures convergence towards better solutions. The algorithm does not suffer from premature convergence due high exploration at the beginning of the algorithm. However, since good solution attracts others, if initial set of CPs not uniformly distributed over solution space, then the algorithm may be trapped into any local optima.

Applications of this algorithm are mainly related to structural engineering designs [172–175] and geometry optimization [176].

4. Conclusion

In this paper, we have categorically discussed various optimization algorithms that are mainly inspired by physics. Major areas covered by these algorithms are quantum theory, electrostatics, electromagnetic, Newton’s gravitational law, and laws of motion. This study shows that most of these algorithms are inspired by quantum computing and significant numbers of applications are developed on the basis of them. Parallel nature of quantum computing perhaps attracts researchers towards quantum-based algorithms. Another most attractive area of physics for inspiration is Newton’s gravitational laws and laws of motion. We have realized that hybridization of quantum computing and biological phenomenon draws most attention these days. As biological phenomenon suggests best strategies and quantum computing provide simultaneity to those strategies; so merging of both into one implies better result. In this paper, we have studied formational aspects of all the major algorithms inspired by physics. We hope, this study will definitely be beneficial for new researchers and motivate them to formulate great solutions from those inspirational theorems of physics to optimization problems.

**Abbreviations**

- ACO: Ant colony optimization
- APO: Artificial physics optimization
- BB-BC: Big bang-big crunch
- BFO: Bacterial forging optimization
- BGS: Binary gravitational search algorithm
- BIS: Biological immune system
- BQEA: Binary Quantum-inspired evolutionary algorithm
- CFO: Central force optimization
- CQACO: Continuous quantum ant colony optimization
- CSS: Charged system search
- EM: Electromagnetism-like heuristic
- GA: Genetic Algorithm
- GbSA: Galaxy-based search algorithm
- GIO: Gravitational interaction optimization
- GSA: Gravitational search algorithm
- HO: Hysteretic optimization
- HQGA: Hybrid quantum-inspired genetic algorithm
- HS: Harmony search
- IGGA: Immune grativation inspired optimization algorithm
- IQEA: Improved quantum evolutionary algorithm
- LP: Linear programming
- MOGSA: Multiobjective gravitational search algorithm
- NLP: Nonlinear programming
- PSO: Particle swarm optimization
- PSOGSA: PSO gravitational search algorithm
- QBSO: Quantum-inspired bacterial swarm optimization
- QEA: Quantum-inspired evolutionary algorithm
- QGA: Quantum-inspired genetic algorithm
- QGO: Quantum genetic optimization
- QICA: Quantum-inspired immune clonal algorithm
- QPSO: Quantum-behaved particle swarm optimization
- QSE: Quantum swarm evolutionary algorithm
- QPGA: Reduced quantum genetic algorithm
- SA: Simulated annealing
- TSP: Travelling salesman problem
- UBB-CBC: Unified big bang-chaotic big crunch
- VM-APO: Vector model of artificial physics optimization
- vQEA: Versatile quantum-inspired evolutionary algorithm.

**References**


[38] P. Purwoharjono, M. Abdillah, O. Penangsang, and A. Soepriyanto, “Voltage control on 500 kV Java-Bali electrical power system for power losses minimization using gravitational search


