QDDS – A Novel Quantum-inspired Swarm Optimizer: Theoretical Foundations, Convergence Analyses and Application Perspectives

By

Saptarshi Sengupta

Dissertation

Submitted to the Faculty of the
Graduate School of Vanderbilt University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY
in

Electrical Engineering

December 14, 2019

Nashville, Tennessee

Approved:

Alan Peters, Ph.D.

Douglas Hardin, Ph.D.

Nilanjan Sarkar, Ph.D.

Don Mitchell Wilkes, Ph.D.

Kazuhiko Kawamura, Ph.D.

Dedicated to my parents Malabika and Saibal Sengupta

ACKNOWLEDGEMENTS

I am thankful to all the wise and visionary people who made this solo adventure of being thousands of miles away from home for four and a half years worthwhile. I am grateful for this incredible opportunity and thoroughly enjoyed pushing myself and discovering a completely new side of me, one that loves to travel – one which would not exist had I not chosen to take up the Ph.D. program at Vanderbilt.

I am indebted to my research advisor Dr. Alan Peters for believing in me and motivating me to pursue my interests and broadly carve out my research direction within swarm intelligence. His relentless support in the form of insightful inputs on dynamical systems, constructive criticism and seeing me through difficult days when results were hard to come by, is the reason I am here today. I am also thankful to Dr. Kazuhiko Kawamura for being an active member of my committee and for being a great source of support on top of providing reality checks and useful inputs on the feedback control aspect of swarms. His knowledge and vision helped shape my dissertation greatly. I am thankful to Dr. Douglas Hardin and Dr. Mitch Wilkes for agreeing to be on my committee and providing their interpretations of and possible extensions of the dissertation work. I would also like to thank Dr. Nilanjan Sarkar for the stimulating discussions on multiagent systems and challenging me with some thought-provoking questions.

I am grateful to my alma mater South Point High School for instilling in me the *courage to know* and to the wonderful teachers who helped shape my understanding of the world as a pupil of science. I am grateful for having gifted mentors at Netaji Subhash Engineering College – Prof. Supriya Dhabal, Prof. Niladri Shekhar Mishra and Prof. Chira Ranjan Datta who helped make my transition from college to graduate school smooth and for teaching me that humility is a key correlator of consistent and impactful work in the long run. I want to convey a special note of thanks to Dr. Shyamali Mukherjee and Dr. Salil K. Das for making my stay in Nashville multidimensional. I am thankful to my friends – Sanchita, Krishnendu, Cole, Peng, Semiu, Nazirah, Daniel and Charreau for being there for me. My partners in mischief – Dipan, Aniket, Debopam, Swagatam, Sahana, Soham and Shouvik: thanks for staying by my side through thick and thin.

Lastly, to my family in Kolkata – none of this would have been possible without your unwavering belief in me and your love and support. The more I begin to make sense of the innate complexity and scale of natural processes and the randomness at play therein, the more I am convinced how precious and optimal having a loving family is. This one therefore, is for you.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

NATURE-INSPIRED OPTIMIZATION

1.1 Ill-structured Problems and the No-Free-Lunch Theorem

Several optimization problems across science and engineering are ill-structured, in that their error surfaces manifest themselves as a hyper-surface in a multidimensional parameter space, containing local minima and saddle points in addition to the global minima that represents the desired solution. These points of interest are often difficult to explore too, given the non-quadratic nature of the error surface which is frequently laden with ridges and valleys. This introduces an issue known as *solution stagnation* in some classical approaches such as gradient descent, making further learning infeasible.

Stochastic search techniques, when used in a structured manner tend to intelligently traverse the parameter space while skipping an exhaustive approach. The central idea in such approaches is to generate random 'guesses' or estimates and use information from the past to intelligently alter the estimates. The expectation from such alterations is that there is a non-zero probability of generating an altered estimate that lowers the error or the offset from the true global optima. There has been a lot of work that looks at the mechanisms for such alteration or 'evolution' and some of the more robust mechanisms that replicate solution evolutions are outlined in [1][2][3][4][103][119][129]. Evolutionary algorithms typically initialize with a random set of solutions and leverage some model/s of optimization inspired off natural processes to evolve the solutions towards promising regions in the parameter space. The No Free Lunch (NFL) Theorem [203] by Wolpert and Macready establishes that

no single optimization algorithm can produce superior results when averaged over all objective functions. Thus, the thrust has been on creating application-specific optimization algorithms that do largely do well on a broad range of problems but perform exceptionally well when tuned to account for the specifics of a problem. Essentially, an evolutionary algorithm tries to meet the following mathematical objective/s:

$$minimize_{\,\boldsymbol{x} \in \mathbf{R^n}}\, \boldsymbol{f_i}(\boldsymbol{x}) \qquad i = (1,2,\dots,M)$$

subject to:

$$\boldsymbol{h_j}(\boldsymbol{x}) = \mathbf{0} \qquad j = (1,2,\dots,J) \tag{1.1}$$

$$\boldsymbol{g_k}(\boldsymbol{x}) \leq \mathbf{0} \qquad k = (1,2,\dots,K) \tag{1.2}$$

where $\boldsymbol{f_i(x)}$, $\boldsymbol{h_j(x)}$ and $\boldsymbol{g_k(x)}$ are functions of the decision vector $\boldsymbol{x} = (x_1, x_2, \dots, x_n)^T$

$\boldsymbol{x}$ – Design or decision variables (real continuous, discrete or a mixture of both)

$\boldsymbol{f_i(x)}$ – Objective/Cost Functions. When $i = 1$, the problem is single objective

$\mathbf{R^n}$ – Design/Search space spanned by the decision vector $\boldsymbol{x}$

$\tilde{\mathbf{R}}^{\mathbf{n}}$ – Solution space spanned by objective/cost function values

$\boldsymbol{h_j}$ and $\boldsymbol{g_k}$ are constraints (minimization problem)

1.2 Design of Particle Swarm Intelligence: Classical and Quantum-inspired

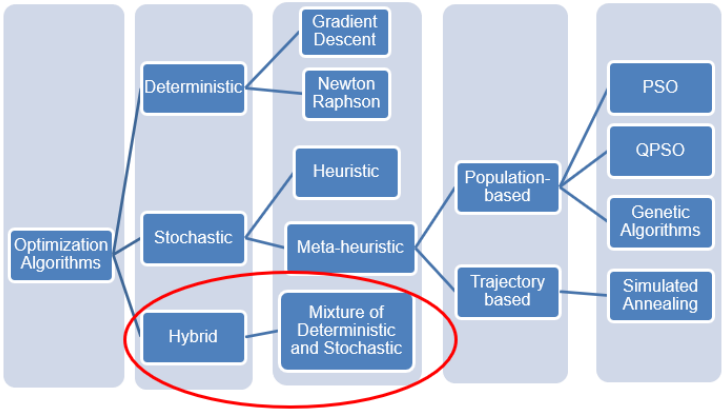


Figure 1.1 Generalized Classification of Optimization Algorithm Types

Swarm-based algorithms are metaheuristics, i.e. higher level heuristics used in generating partial search heuristics which may provide a sufficiently good solution to an optimization problem which is hard due to computational or informational limitations. The word '*meta*' is synonymous to the high-level implementations of these algorithms and typically metaheuristic algorithms rely on a tradeoff between randomization and local search by using trial and error as a means to find an approximate solution to the problem at hand within a reasonable bound of time. The expectation is that among the pool of candidate solutions generated, some are acceptable but there is no guarantee of optimality. Reliance is primarily on two major components – exploration (diversification) and intensification (exploitation) with a judicious balance between the two being a primary necessity. The essence of randomization in most metaheuristics in a random walk.

$$G_N = \sum_{i=1}^{N} \mathrm{X}_i = \mathrm{X}_1 + \mathrm{X}_2 + \mathrm{X}_3 + \cdots \mathrm{X}_N \tag{1.3}$$

$$= \sum_{i=1}^{N-1} X_i + X_N$$

$$= G_{N-1} + X_N$$

The current state G_N depends only on previous state G_{N-1}. X_i is a random step drawn from a distribution of choice and the step size may be adaptive and variable, depending on the specifics of the problem. A couple of key issues that elevate the solution quality in random search algorithms are: (a) **Initial seed quality** – the solution initialization choices can be random, but they should be of good quality and (b) **Effective guiding strategies –** Simple random walks are expensive, thus mimicry of natural models of optimization are a feasible choice. Fig. 1.1 shows a generalized classification of deterministic, stochastic and hybrid optimization algorithms.

1.2.1 Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a metaheuristic global optimization paradigm that has gained prominence in the last two decades due to its ease of application in unsupervised, complex multidimensional problems which cannot be solved using traditional deterministic algorithms. The canonical particle swarm optimizer is based on the flocking behavior and social co-operation of birds and fish schools and draws heavily from the evolutionary behavior of these organisms. A detailed walkthrough of the PSO algorithm with special emphasis on the development, deployment and improvements of its most basic as well as some of the very recent state–of-the-art implementations is of the essence in going forward with more sophisticated swarm-based algorithms discussed later. Concepts and directions on choosing the inertia weight, constriction factor, cognition and social weights and perspectives on convergence, parallelization, elitism, niching and discrete optimization

as well as neighborhood topologies need to be outlined, too. Hybridization attempts with other evolutionary and swarm paradigms in selected applications will be covered and an up-to-date primer put forward for the interested reader.

1.2.2 The Quantum Double Delta Swarm (QDDS) Algorithm

A novel nature-inspired algorithm (the Quantum Double Delta Swarm or QDDS) modeled after the mechanism of convergence to the center of attractive potential field generated within a single well in a double Dirac delta well setup will be put forward and the preliminaries discussed. Theoretical foundations and experimental illustrations will be incorporated to provide a first basis for further development, specifically in refinement of solutions and applicability to problems in high dimensional spaces. Simulations carried out over varying dimensionality on four benchmark functions, viz. Rosenbrock, Rastrigrin, Griewank and Sphere as well as multidimensional Finite Impulse Response (FIR) Filter design problem with different population sizes will illustrate the algorithm yields superior results in comparison to some related reports in the literature while reinforcing the need of substantial future work to deliver near-optimal results consistently, especially if dimensionality scales up.

1.2.3 The Chaotic QDDS (C-QDDS) Algorithm

The QDDS approach introduces a new networked, fully-connected metaheuristic optimization scheme inspired by the convergence mechanism to the center of potential generated within a single well of a spatially co-located double-delta well setup. It mimics the wave nature of candidate positions in solution spaces and draws upon quantum mechanical interpretations much like other quantum-inspired computational intelligence

paradigms. A Chebyshev map driven chaotic perturbation in the optimization phase of the QDDS algorithm will be introduced to diversify weights placed on contemporary and historical, socially-optimal agents' solutions [234]. This will be followed by a characterization of solution quality on a suite of 23 single-objective functions and a comparative analysis with eight other related nature-inspired approaches will be presented. By comparing solution quality and successful runs over dynamic solution ranges, insights about the nature of convergence will be obtained. A two-tailed t-test will establish the statistical significance of the solution data whereas Cohen's d and Hedge's g values will provide a measure of effect sizes. The trajectory of the fittest pseudo-agent will be traced over all iterations to comment on the dynamics of the system and prove that the proposed algorithm is theoretically globally convergent under the assumptions adopted for proofs of other closely-related random search algorithms. The organization of the corresponding chapters is as follows:

Chapter 2 walks through the motivations and developments in particle swarm systems including notes on the foundations and historical overview of stochastic multi-agent systems inspired off natural processes. An introduction to the QDDS mechanism is laid out and is followed by the motivations of using a chaotic map within the optimization phase of the QDDS.

Chapter 3 elaborates on the historical and recent developments surrounding particle swarm systems and puts forward varied perspectives on hybridization attempts with other deterministic and stochastic algorithm within the intellectual neighborhood of evolutionary computation and swarm intelligence.

In Chapter 4, the mechanisms that underpin the QDDS algorithm are detailed and the conditions of convergence are derived followed by experimental validation.

Chapter 5 derives its way through the classical and quantum interpretations in these multiagent systems followed by swarm propagation under the influence of a double Dirac delta well and sets up its quantum mechanical model while taking into consideration a correction in the form of a Chebyshev map driven dynamical weight and provides an involved algorithmic procedure for purposes of reproducibility. Following this, it details some benchmarking problems and graphically illustrates their three-dimensional visualizations. This is followed by comparative analyses of iterations on the benchmarks and statistical significance tests, taking into account the contribution of effect sizes. The trajectory of the best performing agent in each iteration is tracked along the function contours and the limitations and successes of the approach are identified. Critical analyses are presented in light of the findings and a global convergence proof is given for the algorithm, and finally, future directions are charted out.

Chapter 6 presents visualizations of the swarming behavior on the Rastigrin, Griewank and Shekel's Family Function 3 and provides a two-dimensional view of swarm convergence to promising regions near the global optima for high dimensional cases.

Chapter 7 reports performance metrics of C-QDDS on: a) High Dimensional Finite Impulse Response Filter Design of orders 10 and 20 as well as outlines the design and analysis of an automated, bi-layered deep neural network architecture recommendation scheme wherein random architectures are evolved using C-QDDS to maximize classification accuracy on test data. The neuro-evolution testbed is bi-layered in that it uses an outer C-QDDS driven architecture proposal generation layer which serves to render a population of

neural network architectures which form a networked, fully connected swarm. The inner layer evaluates these architecture proposals on the training data by trying overfitting it using a multi-agent optimization in the training phase instead of backpropagation – the lower the classification error, the better the learning algorithm. On top of that, by establishing a gradual increase in classification accuracy by evolving the architectures, the efficacy of the outer layer as a neural architecture search is validated.

CHAPTER 2

PARTICLE SWARM SYSTEMS: MOTIVATIONS AND DEVELOPMENTS

2.1 Foundations of Particle Swarm Systems

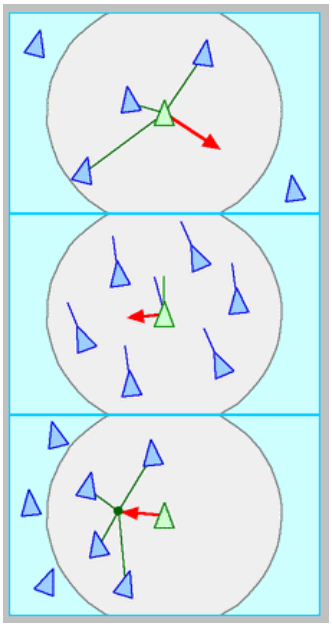
The last two decades has seen unprecedented development in the field of Computational Intelligence with the advent of the GPU and the introduction of several powerful optimization algorithms that make little or no assumption about the nature of the problem. Particle Swarm Optimization (PSO) is one among many such techniques and has been widely used in treating ill-structured continuous/discrete, constrained as well as unconstrained function optimization problems [1]. Much like popular Evolutionary Computing paradigms such as Genetic Algorithms [2] and Differential Evolution [3], the inner workings of the PSO make sufficient use of probabilistic transition rules to make parallel searches of the solution hyperspace without explicit assumption of derivative information. The underlying physical model upon which the transition rules are based is one of emergent collective behavior arising out of social interaction of flocks of birds and schools of fish. Since its inception in 1995, PSO has found use in an ever-increasing array of complex, real-world optimization problems where conventional approaches either fail or render limited usefulness. Its intuitively simple representation and relatively low number of adjustable parameters make it a popular choice for many problems which require approximate solutions up to a certain degree. There are however, several major shortcomings of the basic PSO that introduce failure modes such as stagnation and convergence to local optima which has led to extensive studies (such as [4-5]) aimed at mitigation and resolution of the same. In this review, the

foundations and frontiers of advances in PSO have been reported with a thrust on significant developments over the last decade.

2.2 Historical Overview of Particle Swarm Intelligence

Agents in a natural computing paradigm are decentralized entities with generally no perception of the high-level goal in pursuit yet can model complex real-world systems. This is made possible through several low-level goals which when met facilitate meaningful collective behavior arising from these seemingly unintelligent and non-influential singular agents. An early motivation can be traced from Reeves' introduction of *particle systems* in the context of modeling natural objects such as fire, clouds and water in computer-based animations while at Lucasfilm Ltd (1983) [6]. In the course of development, agents or 'particles' are generated, undergo transformations in form and move around in the modeling environment and eventually are rejected or 'die'. Reeves concluded that such a model is able to represent the dynamics and form of natural environments which were rendered infeasible using classical surface-based representations [6]. Subsequent work by Reynolds in the *Boid Model* (1986) established simple rules that increased autonomy of particle behavior and laid down simple low-level rules that *boids* (bird-oid objects) or particles could obey to give rise to emergent behavior [7]. The complexity of the Boids Model is thus a direct derivative of the simple interactions between the individual particles. Reynolds formulated three distinct rules of flocking for a particle to follow: *separation, alignment* and *cohesion* (shown in Fig. 2.1). While the separation principle allows particles to move away from each other to avoid crowding, the alignment and cohesion principles necessitate directional updates to move towards the average heading and position of nearby flock members respectively. The inherent nonlinearity of the boids render *chaotic behavior* in the emergent group dynamics

whereas the negative feedback introduced by the simple, low level rules effect in *ordered behavior*. The case where each boid knows the whereabouts of every other boid has $O(n^2)$ complexity making it computationally infeasible. However, Reynolds propositioned a neighborhood model with information exchange among boids in a general vicinity, thereby reducing the complexity to O(n) and speeding up the algorithmic implementation.



1. **Separation**: Steer to avoid crowding the local flockmates
2. **Alignment**: Steer towards average heading of local flockmates
3. **Cohesion**: Steer to move toward average position of local flockmates

Figure 2.1. **Steering Behaviors of Boids in Reynolds' Flocking Rules [244]**

The Particle Swarm Optimization algorithm was formally introduced in 1995 by Eberhart and Kennedy through an extension of Reynold's work. By incorporating local information exchange through nearest neighbor velocity matching, the flock or *swarm* prematurely converged in a unanimous fashion. Hence, a random perturbation or *craziness* was introduced in the velocities of the particles leading to sufficient variation and subsequent lifelike dynamics of the swarm. Both these parameters were later eliminated as the flock seemed to converge onto *attractors* equally well without them. The paradigm thus ended up

with a population of agents which were more in conformity with the dynamics of a swarm than a flock.

2.3 Introduction to the Quantum Double Delta Swarm Approach

The traditional Quantum-behaved Particle Swarm Optimization (QPSO) algorithms [217][218][219] (both Types I and II) extend the classical Newtonian dynamics of agent propagation in the canonical Particle Swarm Optimization (PSO) [1][220][236] to a quantum framework. The convergence of particles to promising regions in the solution space in QPSO is driven by an attractive potential field directed towards the center of a singular Dirac Delta well. This is followed by the collapse of the wavefunction indicative of the particles' states using recursive Monte Carlo. The traditional types of QPSO are quite efficient and inexpensive candidates suitable for highly non-linear and non-convex optimization leveraging the quantum nature of the particles and the corresponding wavefunctions that sample a larger region of the solution space as compared to their classical counterpart employing a binary strategy: a particle is either present at a unique location or not. However, the intuitively simple rendering of the binding force exerted by a single delta well on a particular particle has been the subject of scrutiny and further research as demonstrated by [221]. One line of thought contends that the attractive coupling offered by a multi-well attractor is stronger than that offered by a singular one therefore facilitating a stronger stable equilibrium criterion [222]. Xie et al. [221] have recently proposed a quantum-behaved PSO based on a double delta model which assimilates the following three components: a) the global best (gbest) position, b) an agent's location with respect to the gbest position and c) an agent's location with respect to the mean of individual agents' best positions. The authors chose to model the personal and global best positions as centers of two singular delta

potential wells, thereby arriving at a multi-scale representation of a double delta potential well. Simulations on widely used test functions such as Rosenbrock, Rastrigrin, Griewank and Sphere using varying population sizes for problem dimensionality 10 through 30 at a step size of 10 indicate the effective outcomes obtained using the algorithm. Further, the authors note that using two attractors in place of one increased the global search capability and convergence accuracy. In our work however, we are concerned more about finding the state equations of particles based on two spatially co-located delta potential wells. To this effect, an isolated system of double Dirac-delta potential wells and convergence to its centers are considered. The resulting iterative state updates mimic the trajectory of a bound particle ($E<0$) as it moves towards the lowest energy configuration viz. the point where the attractive potential is the least i.e. the center of the dominant well. By identifying constraints on the motion of the particles guided by their probabilistic nature of existence, an iterative scheme of convergence is put forward. The resulting algorithm (QDDS) is tested on a suite of benchmark functions which have many local minima, are bowl-shaped or valley-shaped. Trial evaluations indicate the efficiency of the algorithm in finding solutions of acceptable quality with room for improvement both in terms of computational expense and finetuning of solutions.

2.4 Motivations for the Chaotic QDDS Approach

With sensor fusion and big data taking center stage in ubiquitous computing niches, the importance of customized, application-specific optimization paradigms is gaining recognition. The computational intelligence community is poised for exponential growth as nature-inspired modeling becomes ever more practicable in the face of abundant computational power. Thus, it is in the interest of exploratory analysis to mimic different

natural systems in order to gain adequate understanding of when and on which kinds of problems certain types of biomimicry work particularly well. In this work, a subclass of the modeling paradigm of quantum-mechanical systems involving two Dirac delta potential functions is studied. The technique chosen for the study, viz. the Quantum-Double Delta Swarm (QDDS) algorithm [229], extends the well-known Quantum-behaved Particle Swarm Optimization (QPSO) [217][218][219] using an additional Dirac delta well and imposing motional constraints on particles to effect in convergence to a single well under the influence of both. The particles in QDDS are centrally pulled by an attractive potential field and a recursive Monte Carlo relation is established by collapse of the wave functions around the center of the wells. The methodology has been put forward and tested on select unimodal and multimodal benchmarks in Sengupta et al. [229] and generates promising solution quality when compared to Xi et al. [232]. In this work, we primarily report performance improvements of the QDDS algorithm when its solution update process is influenced by a random perturbation drawn from a Chebyshev chaotic map. The perturbation seeks to diversify the weight array corresponding to the current and socially-optimal agents' solutions. A detailed performance characterization over twenty-three single-objective, unimodal and multimodal functions of fixed and varying dimensions is carried out. The characterization is repeated for eight other nature-inspired approaches to provide a basis for comparison. The collective potential (cost) quality and precision data from the experimentation provide information on the operating conditions and tradeoffs while the conclusion drawn from a subsequent two-tailed *t*-test points to the statistical significance of the results at the $\Theta = 0.05$ level. We follow the path of the best performing agent in any iteration across all iterations and critically analyze the dynamical limitations of the algorithm

(we assume that one iteration is equivalent to an atomic level function evaluation). Consequently, we also look at the global convergence proof of Random Search algorithms [233] and contend that the proposed algorithm theoretically converges to the global infimum under certain weak assumptions adopted for convergence proofs of similar random search techniques.

CHAPTER 3

PARTICLE SWARMS: HISTORICAL AND RECENT DEVELOPMENTS

3.1 Working Mechanism of the Canonical PSO

The PSO algorithm employs a swarm of particles which traverse a multidimensional search space to seek out optima. Each particle is a potential solution and is influenced by experiences of its neighbors as well as itself. Let $\mathbf{x}_i(t)$ be the position in the search space of the *i*-th particle at time step *t*. The initial velocity of a particle is regulated by incrementing it in the positive or negative direction contingent on the current position being less than the best position and vice-versa (Shi and Eberhart, 1998) [8].

$$vx[\,][\,] = vx[\,][\,] + 2 * rand(\,) * (pBest[\,][\,] - presentx[\,][\,]) + 2 * rand(\,) * (pBest[\,][gbest\,] - presentx[\,][\,]) \quad (3.1)$$

The random number generator was originally multiplied by 2 in [1] so that particles could have an overshoot across the target in the search space half of the time. These values of the constants, known as the cognition and social acceleration co-efficient were found to effect superior performance than previous versions. Since its introduction in 1995, the PSO algorithm has undergone numerous improvements and extensions aimed at guaranteeing convergence, preserving and improving diversity as well as offsetting the inherent shortcomings by hybridizing with parallel paradigms from evolutionary computation.

3.2 Perspectives on Development

3.2.1 Inertia Weight

The initialization of particles is critical in visiting optima when the initial velocity is zero. This is because the *pBest* and *gBest* attractors help intelligently search the

neighborhood of the initial kernel but do not facilitate exploration of new regions in the search space. The velocity of the swarm helps attain this purpose, however suitable clamps on the velocity are needed to ensure the swarm does not diverge. Proper selection of the maximum velocity v_{max} is important to maintain control: a large v_{max} introduces the possibility of global exploration whereas a small value implies a local, intensive search. Shi and Eberhart suggested an *'inertia weight'* ω which is used as a control parameter for the swarm velocity [8], thereby making possible the modulation of the swarm's momentum using constant, linear time-varying or even non-linear temporal dependencies [9]. However, the inertia weight could not fully do away with the necessity for velocity clamping [10]. To guarantee convergent behavior and to come to a balance between exploitation and exploration the value of the inertia weight must be chosen with care. An inertia weight equal or greater than one implies the swarm velocity increases over time towards the maximum velocity v_{max}. Two things happen when the swarm velocity accelerates rapidly towards v_{max}: particles cannot change their heading to fall back towards promising regions and eventually the swarm diverges. On the other hand, an inertia weight less than one reduces the acceleration of the swarm until it eventually becomes a function of only the acceleration factors. The exploratory ability of the swarm suffers as the inertia goes down, making sudden changes in heading possible as social and cognitive factors increasingly control the position updates. Early works on the inertia weight used a constant value throughout the course of the iterations but subsequent contributions accommodated the use the dynamically changing values. The de facto approach seemed to be to use a large initial value of ω to help in global exploration followed by a gradual decrease to hone in on promising areas towards the latter part of the search process.

Efforts to dynamically change inertia weight is organized in following categorizations:

3.2.1.1 Random Selection (RS)

In each iteration, a different inertia weight is selected, possibly drawn from an underlying distribution with a mean and standard deviation of choice. However, care should be taken to ensure convergent behavior of the swarm.

3.2.1.2 Linear Time Varying (LTV)

Usually, the implementation of this kind decreases the value of ω from a preset high value of ω_{max} to a low of $\omega_{min.}$ Standard convention is to take ω_{max} and ω_{min} as 0.9 and 0.4. The LTV inertia weight can be expressed as [11-12]:

$$\omega_t = (\omega_{max} - \omega_{min}) \frac{(t_{max} - t)}{t_{max}} + \omega_{max} \tag{3.2}$$

where t_{max} is the number of iterations, t is the current iteration and ω_t is the value of the inertia weight in the t-th iteration.

There are some implementations that look at the effects of increasing the inertia weight from an initial low value to a high value, the interested reader should refer to [13][14].

3.2.1.3 Non-Linear Time Varying (NLTV)

As with LTV inertia weights, NLTV inertia weights too, tend to fall off from an initial high value at the start of the optimization process. Nonlinear decrements allow more time to fall off towards lower end of the dynamic range, thereby enhancing local search or exploitation. Naka et al. [15] proposed the following nonlinear time varying inertia weight:

$$\omega_{t+1} = (\omega_t - 0.4) \frac{(t_{max} - t)}{t_{max} + 0.4} \tag{3.3}$$

where $\omega_{t=0}$ = 0.9 is the initial choice of ω. Clerc introduced the concept of relative improvement of the swarm in developing an adaptive inertia weight [16]. The change in the inertia of the swarm is in proportion to the relative improvement of the swarm. The relative improvement κ^i_t is estimated by:

$$\kappa_t^i = \frac{f(lbest_t^t) - f(x_t^t)}{f(lbest_t^t) + f(x_t^t)} \tag{3.4}$$

Clerc's updated inertia weight can be expressed as:

$$\omega_{t+1} = \omega_0 + (\omega_{t_{max}} - \omega_0)\frac{e^{m_i(t)} - 1}{e^{m_i(t)} + 1} \tag{3.5}$$

where ω_{tmax}=0.5 and ω_0<1. Each particle has a unique inertia depending on its distance from the local best position.

3.2.1.4 Fuzzy Adaptive (FA)

Using fuzzy sets and membership rules, ω can be dynamically updated as in [17]. The change in inertia is computed using fitness of gBest particle as well as that of the current value of ω. The change is implemented through use of a set of fuzzy rules [17-18]. The choice of ω is dependent on the problem in hand, specifically on the nature of the search space.

3.2.2 Constriction Factor

Clerc demonstrated that to ensure optimal trade-off between exploration and exploitation, the use of a constriction coefficient χ may be necessary [19-20]. The constriction co-efficient was developed from eigenvalue analyses of computational swarm dynamics in [19]. The velocity update equation changes to:

$$vx[\,][\,] = \chi(vx[\,][\,] + \Omega_1 * rand() * (pBest[\,][\,] - presentx[\,][\,])$$
$$+\Omega_2 * rand() * (pBest[\,][gbest\,] - presentx[\,][\,])) \tag{3.6}$$

where χ was shown to be:

$$\chi = \frac{2\nu}{\left|2-\Omega-\sqrt{\Omega(\Omega-4)}\right|} \tag{3.7}$$

$$\Omega = \Omega_1 + \Omega_2 \tag{3.8}$$

Ω_1 and Ω_2 can be split into products of social and cognitive acceleration coefficients c_1 and c_2 times random noise r_1 and r_2. Under the operating constraint that $\Omega \geq 4$ and $\nu \in [0,1]$, swarm convergence is guaranteed with particles decelerating as iteration count increases. The parameter ν controls the local or global search scope of the swarm. For example, when ν is set close to 1, particles traverse the search space with a predominant emphasis on exploration. This leads to slow convergence and a high degree of accuracy in finding the optimum solution, as opposed to when ν is close to zero in which case the convergence is fast but the solution quality may vary vastly. This approach of constricting the velocities is equivalent in significance to the inertia weight variation, given its impact on determining solution quality across neighborhoods in the search space. Empirical studies in [21] demonstrated that faster convergence rates are achieved when velocity constriction is used in conjunction with clamping.

3.2.3 Cognition and Social Velocity Models of the Swarm

One of the earliest studies on the effect of different attractors on the swarm trajectory update was undertaken by Kennedy in 1997 [22]. The cognition model considers only the cognitive component of the canonical PSO in Equation (3.1).

$$v_{t+1}[\,][\,] = (v_t[\,][\,] + C_1 * rand(\,) * (pBest[\,][\,] - presentx[\,][\,]) \tag{3.9}$$

The cognition model performs a local search in the region where the swarm members are initialized and tends to report suboptimal solutions if the acceleration component and upper bounds on velocity are small. Due to its weak exploratory ability, it is also slow in

convergence. This was reported by Kennedy [22] and subsequently the subpar performance of the model was confirmed by the works of Carlisle and Dozier [23]. The social model, on the other hand, considers only the social component.

$$v_{t+1}[\,][\,] = (v_t[\,][\,] + C_1 * rand(\,) * (pBest[\,][\,gBest] - presentx[\,][\,]) \qquad (3.10)$$

In this model, the particles are attracted towards the global best in the feasible neighborhood and converge faster with predominantly exploratory behavior. This was reported by Kennedy [22] and confirmed by Carlisle and Dozier [23].

3.2.4 Cognitive and Social Acceleration Coefficients

The acceleration coefficients C_1 and C_2 when multiplied with random vectors r_1 and r_2 render controllable stochastic influences on the velocity of the swarm. C_1 and C_2, simply put, are weights that capture how much a particle should weigh moving towards its cognitive attractor (pBest) or its social attractor (gBest). Exchange of information between particles mean they are inherently co-operative, thus implying that an unbiased choice of the acceleration coefficients would make them equal. For case specific implementations, one may set C_1=0 or C_2=0 with the consequence being that individual particles will rely solely on their own knowledge or that individual particles will only rely on the knowledge of the best particle in the entire swarm. It is obvious that multimodal problems containing multiple promising regions will benefit from a balance between social and cognitive components of acceleration.

Stacey et al. used mutation functions in formulating acceleration coefficients and by keeping the step size of mutation equal to v_{max}, improvements were noticed over the general implementation (MPSO-TVAC) [24]. Jie et al. introduced new Metropolis coefficients in PSO, leading to better efficiency and stability [25]. It hybridizes Particle Swarm

Optimization with Simulated Annealing [26] and reduces runtime as well as the number of iterations.

3.2.4.1 Choice of Values

In general, the values of C_1 and C_2 are kept constant. An empirically found optimum pair seems to be 2.05 for each of C_1 and C_2 and significant departures or incorrect initializations lead to divergent behavior. Ratnaweera et al. suggested that C_1 should be decreased linearly over time, whereas C_2 should be increased linearly [27]. Clerc's fuzzy acceleration reports improvements using swarm diversity and the ongoing iteration by adaptively refining coefficient values [16].

3.2.5 Topologies

The topology of the swarm of particles establishes a measure of the degree of connectivity of its members to the others. It essentially describes a subset of particles with whom a particle can initiate information exchange [28]. The original PSO outlined two topologies that led to two variants of the algorithm: *lBest PSO* and *gBest PSO*. The *lBest* variant associates a fraction of the total number of particles in the neighborhood of any particular particle. This structure leads to multiple best particles, one in each neighborhood and consequently the velocity update equation of the PSO has multiple social attractors. Under such circumstances, the swarm is not attracted towards any single global best rather a combination of sub-swarm bests. This brings down the convergence speed but significantly increases the chance of finding global optima. In the *gBest* variant, all particles simultaneously influence the social component of the velocity update in the swarm, thereby leading to an increased convergence speed and a potential stagnation to local optima if the true global optima is not where the best particle of the neighborhood is.

There have been some fundamental contributions to the development of PSO topologies over the last two decades [29-31]. A host of PSO topologies have risen out of these efforts, most notably the Random Topology PSO, The Von-Neumann Topology PSO, The Star Topology PSO and the Toroidal Topology PSO. In [31], Mendes et al. studied several different sociometry with a population size of 20 where they quantified the effect of including the past experiences of an individual by implementing with and without, the particle of interest. Interested readers can also refer to the recent work by Liu et al to gain an understanding about topology selection in PSO driven optimization environments [32].

3.2.6 Analysis of Convergence

In this section, the underlying constraints for convergence of the swarm to an equilibrium point are reviewed. Van den Bergh and Engelbrecht as well as Trelea noted that the trajectory of an individual particle would converge contingent upon meeting the following condition [33-35]:

$$1 > \omega > \frac{\Omega_1+\Omega_2}{2} - 1 \geq 0 \tag{3.11}$$

The above relation can be simplified by replacing the stochastic factors with the acceleration coefficients C_1 and C_2 such that when C_1 and C_2 are chosen to satisfy the condition in Eq. (3.12), the swarm converges.

$$1 > \omega > \frac{C_1+C_2}{2} - 1 \geq 0 \tag{3.12}$$

Studies in [19, 34-35] also lead to the implication that a particle may converge to a single point X' which is a stochastic attractor with pBest and gBest being ends of two diagonals. This point may not be an optimum and particles may prematurely converge to it.

$$X' = \frac{\Omega_1 pBest+\Omega_2 gBest}{\Omega_1+\Omega_2} \tag{3.13}$$

3.2.7 Velocity and Position Update Equations of the Standard PSO

The following equations describe the velocity and position update mechanisms in a standard PSO algorithm:

$$v_{ij}(t+1) = \omega v_{ij}(t) + r_1(t)C_1\left(pbest_{ij}(t) - x_{ij}(t)\right) + r_2(t)C_2\left(gbest(t) - x_{ij}(t)\right) \quad (3.14)$$

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1) \quad (3.15)$$

r_1 and r_2 are independent and identically distributed random numbers whereas C_1 and C_2 are the cognition and social acceleration coefficients. x_{ij} , v_{ij} are position coordinates and velocity of the i_{th} agent in the j_{th} dimension. $pbest_{ij}(t)$ and $gbest(t)$ represent the personal and global best locations in the *t-th* iteration. The first term in the right-hand side of eq. (3.14) makes use of ω, which is the inertia weight and the next two terms are excitations towards promising regions in the search space as reported by the personal and global best locations. The personal best replacement procedure assuming a function minimization objective is discussed in eq. (3.16). The global best *gbest(t)* is the minimum cost bearing element of the temporal set of personal bests $pbest_i(t)$ of all particles over all iterations.

$$\textbf{if: } cost\left(x_i(t+1)\right) < cost\left(pbest_i(t)\right) \Rightarrow pbest_i(t+1) = x_i(t+1) \quad (3.16)$$

$$\textbf{else: } pbest_i(t+1) = pbest_i(t)$$

3.2.8 Survey of Hybridization Approaches

A hybridized PSO implementation integrates the inherent social, co-operative character of the algorithm with tested optimization strategies arising out of distinctly different traditional or evolutionary paradigms towards achieving the central goal of intelligent exploration-exploitation. This is particularly helpful in offsetting weaknesses in the underlying algorithms and distributing the randomness in a guided way. The literature

on hybrid PSO algorithms is quite rich and growing by the day. In this section, some of the most notable works as well as a few recent approaches have been outlined.

3.2.8.1 Hybridization of PSO using Genetic Algorithms (GA)

Popular approaches in hybridizing GA and PSO involve using the two approaches sequentially or in parallel or by using GA operators such as selection, mutation and reproduction within the PSO framework. Authors in [36] used one algorithm until stopping criterion is reached to use the final solution in the other algorithm for fine tuning. How the stopping criterion is chosen varies. They used a switching method between the algorithms when one algorithm fails to improve upon past results over a chosen number of iterations. In [37] the first algorithm is terminated once a specified number of iterations has been exceeded. The best particles from the first algorithm populate the particle pool in the second algorithm and the empty positions are filled using random generations. This preserves the diversity of the otherwise similar performing population at the end of the first phase. Authors in [37] put forth the idea of exchanging fittest particles between GA and PSO, running in parallel for a fixed number of iterations.

In Yang et al.'s work on PSO-GA based hybrid evolutionary algorithm (HEA) [38], the evolution strategy of particles employs a two-phase mechanism where the evolution process is accelerated by using PSO and diversity is maintained by using GA. The authors used this method to optimize three unconstrained and three constrained problems. Li et al. used mechanisms such as nonlinear ranking selection to generate offspring from parents in a two-stage hybrid GA-PSO where each stage is separately accomplished using GA and PSO [39]. Valdez et al. proposed a fuzzy approach in testing PSO-GA hybridization [40]. Simple fuzzy rules were used to determine whether to consider GA or PSO particles and change their

parameters or to take a decision. Ghamisi and Benediktsson [55] introduced a feature selection methodology by hybridizing GA and PSO. This method was tested on the Indian Pines hyperspectral dataset as well as for road detection purposes. The accuracy of a Support Vector Machine (SVM) classifier on validation samples was set as the fitness score. The method could select the most informative features within an acceptable processing time automatically and did not require the users to set the number of desired features beforehand. Benvidi et al. [56] used a hybrid GA-PSO algorithm to simultaneously quantify four commonly used food colorants containing tartrazine, sunset yellow, ponceau 4R and methyl orange, without prior chemical separation. Results indicated the designed model accurately determined concentrations in real as well as synthetic samples. From observations the introduced method emerged as a powerful tool to estimate the concentration of food colorants with a high degree of overlap using nonlinear artificial neural network. Yu et al. used a hybrid PSO-GA to estimate energy demand of China in [57] whereas Moussa and Azar introduced a hybrid algorithm to classify software modules as fault-prone or not using object-oriented metrics in [58]. Nik et al. used GA-PSO, PSO-GA and a collection of other hybridization approaches to optimize surveyed asphalt pavement inspection units in massive networks [59]. Premlatha and Natarajan [52] proposed a discrete version of PSO with embedded GA operators for clustering purposes. The GA operator initiates reproduction when particles stagnate. This version of the hybrid algorithm was named DPSO with mutation-crossover.

In [53] Abdel-Kader proposed a GAI-PSO hybrid algorithm for k-means clustering. The exploration ability of the algorithm was used first to find an initial kernel of solutions containing cluster centroids which was subsequently used by the k-means in a local search.

For treating constrained optimization problems, Garg used a PSO to operate in the direction of improving the vector while using GA to update decision vectors [60]. In [61], Zhang et al. carried out experimental investigations to optimize the performance of a four-cylinder, turbocharged, direct-injection diesel engine. A hybrid PSO and GA method with a small population was tested to optimize five operating parameters, including EGR rate, pilot timing, pilot ratio, main injection timing, and injection pressure. Results demonstrated significant speed-up and superior optimization as compared to GA. Li et al. developed a mathematical model of the heliostat field and optimized it using PSO-GA to determine the highest potential daily energy collection (DEC) in [62]. Results indicated that DEC during the spring equinox, summer solstice, autumnal equinox and winter solstice increased approximately by $1.1x10^5$ MJ, $1.8x10^5$ MJ, $1.2x10^5$ MJ and $0.9x10^5$ MJ, respectively.

A brief listing of some of the important hybrid algorithms using GA and PSO are given below in Table 3.1.

Table 3.1 A collection of hybridized GA-PSO algorithms

Author/s:	Year	Algorithm	Area of Application
Robinson et al. [36]	2002	GA-PSO, PSO-GA	Engineering design optimization
Krink and Løvbjerg [41]	2002	Life Cycle Model	Unconstrained global optimization
Conradie et al. [42]	2002	SMNE	Neural Networks
Grimaldi et al. [43]	2004	GSO	Electromagnetic Application
Juang [44]	2004	GA-PSO	Network Design
Settles and Soule [45]	2005	Breeding Swarm	Unconstrained Global Optimization
Jian and Chen [46]	2006	PSO-RDL	Unconstrained Global Optimization
Esmin et al. [47]	2006	HPSOM	Unconstrained Global Optimization
Kim [48]	2006	GA-PSO	Unconstrained Global Optimization
Mohammadi and Jazaeri [49]	2007	PSO-GA	Power Systems
Gandelli et al. [50]	2007	GSO	Unconstrained Global Optimization
Yang et al. [38]	2007	HEA	Constrained and Unconstrained Global Optimization
Kao and Zahara [51]	2008	GA-PSO	Unconstrained Global Optimization
Premlatha and Natrajan [52]	2009	DPSO-mutation-crossover	Document Clustering
Abdel Kader [53]	2010	GAI-PSO	Data Clustering
Kuo and Hong [54]	2013	HGP1, HGP2	Investment Portfolio Optimization
Ghamisi and Benedictsson [55]	2015	GA-PSO	Feature Selection
Benvidi et al [56]	2016	GA-PSO	Spectrophotometric determination of synthetic colorants
Yu et al. [57]	2011	GA-PSO	Estimation of Energy Demand
Moussa and Azar [58]	2017	PSO-GA	Classification
Nik, Nejad and Zakeri [59]	2016	GA-PSO, PSO-GA	Optimization of Surveyed Asphalt Pavement Inspection Unit
Garg [60]	2015	GA-PSO	Constrained Optimization
Zhang et al. [61]	2015	PSO-GA	Biodiesel Engine Performance Optimization
Li et al. [62]	2018	PSO-GA	Optimization of a heliostat field layout

3.2.8.2 Hybridization of PSO using Differential Evolution (DE)

Differential Evolution (DE) by Price and Storn [63] is a very popular and effective metaheuristic for solving global optimization problems. Several approaches of hybridizing DE with PSO exist in literature, some of which are elaborated in what follows. Hendtlass [64] introduced a combination of particle swarm and differential evolution algorithm (SDEA) and tested it on a graduated set of trial problems. The SDEA algorithm works the same way as a particle swarm one, except that DE is run intermittently to move particles from worse performing areas to better ones. Experiments on a set of four benchmark problems, viz. The Goldstein-Price Function, the six hump camel back function, the Timbo2

Function and the n-dimensional 3 Potholes Function showed improvements in performance. It was noted that the new algorithm required more fitness evaluations and that it would be feasible to use the component swarm based algorithm for problems with computationally heavy fitness functions. Zhang and Xie [65] introduced another variant of a hybrid DE-PSO. Their strategy employed different operations at random, rather than a combination of both at the same time. Results on benchmark problems indicated better performance than PSO or DE alone. Talbi and Batouche [66] used DEPSO to approach the multimodal rigid-body image registration problem by finding the optimal transformation which superimposed two images by maximization of mutual information. Hao et al. [67] used selective updates for the particles' positions by using partly a DE approach, partly a PSO approach and tested it on a suite of benchmark problems.

Das et al. scrapped the cognitive component of the velocity update equation in PSO and replaced it with a weighted difference vector of positions of any two different particles chosen randomly from the population [68]. The modified algorithm was used to optimize well-known benchmarks as well as constrained optimization problems. The authors demonstrated the superiority of the proposed method, achieved through a synergism between the underlying popular multi-agent search processes: the PSO and DE. Luitel and Venayagamoorthy [69] used a DEPSO optimizer to design linear phase Finite Impulse Response (FIR) filters. Two different fitness functions: one based on passband and stopband ripple, the other on MSE between desired and practical results were considered. While promising results were obtained with respect to performance and convergence time, it was noted that the DEPSO algorithm could also be applied to the personal best position, instead of the global best. Vaisakh et al. [70] came up with a DEPSO algorithm to achieve optimal

reactive power dispatch with reduced power and enhanced voltage stability. The IEEE-30 bus test system is used to illustrate its effectiveness and results confirm the superiority of the algorithm proposed. Huang et al. [71] studied the back analysis of mechanics parameters using DEPSO-ParallelFEM - a hybrid method using the advantages of DE fused with PSO and Finite Element Method (FEM). The DEPSO algorithm enhances the ability to escape local minima and the FEM increases computational efficiency and precision through involvement of Cluster of Workstation (COW), MPI (Message Passing Interface), Domain Decomposition Method (DDM) [72-73] and Object-oriented Programming (OOP) [74-75]. A computational example supports the claim that it is an efficient method to estimate and back analyze the mechanics parameters of systems.

Xu et al. [76] applied their proposed variant of DEPSO on data clustering problems. Empirical results obtained on synthetic and real datasets showed that DEPSO achieved faster performance than when either of PSO or DE is used alone. Xiao and Zuo [77] used a multi-population strategy to diversify the population and employ every sub-population to a different peak, subsequently using a hybrid DEPSO operator to find the optima in each. Tests on the Moving Peaks Benchmark (MPB) problem resulted in significantly better average offline error than competitor techniques. Junfei et al. [78] used DEPSO for mobile robot localization purposes whereas Sahu et al. [79] proposed a new fuzzy Proportional–Integral Derivative (PID) controller for automatic generation control of interconnected power systems. Seyedmahmoudian et al. [80] used DEPSO to detect maximum power point under partial shading conditions. The proposed technique worked well in achieving the Global Maximum Power Point (GMPP): simulation and experimental results verified this under different partial shading conditions and as such its reliability in tracking the global optima

was established. Gomes and Saraiva [81] described a hybrid evolutionary tool to solve the Transmission Expansion Planning problem. The procedure is phased out in two parts: first equipment candidates are selected using a Constructive Heuristic Algorithm and second, a DEPSO optimizer is used for final planning. A case study based on the IEEE 24-Bus Reliability Test System using the DEPSO approach yielded solutions of acceptable quality with low computational effort.

Boonserm and Sitjongsataporn [82] put together DE, PSO and Artificial Bee Colony (ABC) [129] coupled with self-adjustment weights determined using a sigmoidal membership function. DE helped eliminate the chance of premature convergence and PSO sped up the optimization process. The inherent ABC operators helped avoid suboptimal solutions by looking for new regions when fitness did not improve. A comparative analysis of DE, PSO, ABC and the proposed DEPSO-Scout over benchmark functions such as Rosenbrock, Rastrigin and Ackley was performed to support the claim that the new metaheuristic performed better than the component paradigms viz. PSO, DE and ABC.

A brief listing of some of the important hybrid algorithms using SA and PSO are given below in Table 3.2.

Table 3.2 A collection of hybridized DE-PSO algorithms.

Author/s:	Year	Algorithm	Area of Application
Hendtlass [64]	2001	SDEA	Unconstrained Global Optimization
Zhang and Xie [65]	2003	DEPSO	Unconstrained Global Optimization
Talbi and Batouche [66]	2004	DEPSO	Rigid-body Multimodal Image Registration
Hao et al. [67]	2007	DEPSO	Unconstrained Global Optimization
Das et al. [68]	2008	PSO-DV	Design Optimization
Luitel and Venayagamoorthy [69]	2008	DEPSO	Linear Phase FIR Filter Design
Vaisakh et al. [70]	2009	DEPSO	Power Dispatch
Huang et al. [71]	2009	DEPSO-ParallelFEM	Back Analysis of Mechanics Parameters
Xu et al. [76]	2010	DEPSO	Clustering
Xiao and Zuo [77]	2012	Multi-DEPSO	Dynamic Optimization
Junfei et al. [78]	2013	DEPSO	Mobile Robot Localization
Sahu et al. [79]	2014	DEPSO	PID Controller
Seyedmahmoudian et al. [80]	2015	DEPSO	Photovoltaic Power Generation
Gomes and Saraiva [81]	2016	DEPSO	Transmission Expansion Planning
Boonserm and Sitjongsataporn [82]	2017	DEPSO-Scout	Numerical Optimization

3.2.8.3 Hybridization of PSO using Simulated Annealing (SA)

Zhao et al. [83] put forward an activity network based multi-objective partner selection model and applied a new heuristic based on PSO and SA to solve the multi-objective problem. Yang et al. [84] produced one of the early works on PSO-SA hybrids in 2006 which detailed the embedding of SA in the PSO operation. They noted the efficient performance of the method on a suite of benchmark functions commonly used in the Evolutionary Computing (EC) literature. Gao et al. [85] trained a Radial Basis Function Neural Network (RBF-NN) using a hybrid PSO with chaotic search and simulated annealing. The component algorithms can learn from each other and mutually offset weak performances. Benchmark function optimization and classification results for datasets from the UCI Machine Learning Repository [86] demonstrated the efficiency of the proposed method. Chu et al. [87] developed an Adaptive Simulated Annealing-Parallel Particle Swarm Optimization (ASA-PPSO). ASA-PPSO uses standard initialization and evolution characteristics of the PSO and uses a greedy approach to replace the memory of best

solutions. However, it also infuses an ‘infix’ condition which checks the latest two global best solutions and when triggered, applies an SA operator on some recommended particles or on all. Experimental analyses on benchmark functions established the usefulness of the proposed method.

Sadati et al. [88] formulated the Under-Voltage Load Shedding (UVLS) problem using the idea of static voltage stability margin and its sensitivity at the maximum loading point or the collapse point. The PSO-B-SA proposed by the authors was implemented in the UVLS scheme on the IEEE 14 and 18 bus test systems and considers both technical and economic aspects of each load. The proposed algorithm can reach optimum solutions in minimum runs as compared to the other competitive techniques in [88], thereby making it suitable for application in power systems which require an approximate solution within a finite time bound. Ma et al. [97] approached the NP-hard Job-shop Scheduling Problem using a hybrid PSO with SA operator as did Ge et al. in [89], Zhang et al. in [93] and Song et al. in [90]. A hybrid discrete PSO-SA algorithm was proposed by Dong et al. [91] to find optimal elimination orderings for Bayesian networks. Shieh et al. [92] devised a hybrid SA-PSO approach to solve combinatorial and nonlinear optimization problems. Idoumghar et al. [94] hybridized Simulated Annealing with Particle Swarm Optimization (HPSO-SA) and proposed two versions: a sequential and a distributed implementation. Using the strong local search ability of SA and the global search capacity of PSO, the authors tested out HPSO-SA on a set of 10 multimodal benchmark functions noting significant improvements. The sequential and distributed approaches are used to minimize energy consumption in embedded systems memories. Savings in terms of energy as well as execution time were

noted. Tajbaksh et al. [95] proposed the application of a hybrid PSO-SA to solve the Traveling Tournament Problem.

Niknam et al. [96] made use of a proposed PSO-SA to solve the Dynamic Optical Power Flow Problem (DOPF) with prohibited zones, valve-point effects and ramp-rate constraints taken into consideration. The IEEE 30-bus test system was used to show the effectiveness of the PSO-SA in searching the possible solutions to the highly nonlinear and nonconvex DOPF problem. Sudibyo et al. [98] used SA-PSO for controlling temperatures of the trays in Methyl tert-Butyl Ether (MTBE) reactive distillation in a Non-Linear Model Predictive Control (NMPC) problem and noted the efficiency of the algorithm in finding the optima as a result of hybridization. Wang and Sun [99] applied a hybrid SA-PSO to the K-Means clustering problem.

Javidrad and Nazari [100] recently contributed a hybrid PSO-SA wherein SA contributes in updating the global best particle just when PSO does not show improvements in the performance of the global best particle, which may occur several times during the iteration cycles. The algorithm uses PSO in its initial phase to determine the global best and when there is no change in the global best in any particular cycle, passes the information on to the SA phase which iterates until a rejection takes place using the Metropolis criterion [101]. The new information about the best solution is then passed back to the PSO phase which again initiates search with the obtained information as the new global best. This process of sharing is sustained until convergence criteria are satisfied. Li et al. [102] introduced an efficient energy management scheme in order to increase the fuel efficiency of a Plug-In Hybrid Electric Vehicle (PHEV).

A brief listing of some of the important hybrid algorithms using SA and PSO are given below in Table 3.3.

Table 3.3 A collection of hybridized SA-PSO algorithms.

Author/s:	Year	Algorithm	Area of Application
Zhao et al. [83]	2005	HPSO	Partner Selection for Virtual Enterprise
Yang et al. [84]	2006	PSOSA	Global Optimization
Gao et al. [85]	2006	HPSO	Optimizing Radial Basis Function
Chu et al. [87]	2006	ASA-PPSO	Global Optimization
Sadati et al. [88]	2007	PSO-B-SA	Under-voltage Load Shedding Problem
Ge et al. [89]	2007	Hybrid PSO with SA operator	Job-shop Scheduling
Song et al. [90]	2008	Hybrid PSO with SA operator	Job-shop Scheduling
Dong et al. [91]	2010	PSO-SA	Bayesian Networks
Shieh et al. [92]	2011	SA-PSO	Global Optimization
Zhang et al. [93]	2011	Hybrid PSO with SA operator	Job-shop Scheduling
Idoumghar et al. [94]	2011	HPSO-SA	Embedded Systems
Tajbaksh et al. [95]	2012	PSO-SA	Traveling Tournament Problem
Niknam et al. [96]	2013	SA-PSO	Dynamic Optical Power Flow
Ma et al. [97]	2014	Hybrid PSO with SA operator	Job-shop Scheduling
Sudibyo et al. [98]	2015	SA-PSO	Non-Linear Model Predictive Control
Wang and Sun [99]	2016	SA-PSO	K-Means Clustering
Javidrad and Nazari [100]	2017	PSO-SA	Global Optimization
Li et al. [102]	2017	SA-PSO	Parallel Plug-In Hybrid Electric Vehicle

3.2.8.4 Hybridization of PSO using Ant Colony Optimization (ACO)

Ant Colony Optimization (ACO) proposed by Marco Dorigo [103] captured the organized communication triggered by an autocatalytic process practiced in ant colonies. In later years, Shelokar et al. [104] proposed PSACO (Particle Swarm Ant Colony Optimization) which implemented rapid global exploration of the search domain, while the local search was pheromone-guided. The first part of the algorithm works on PSO to generate initial solutions, while the positions of the particles are updated by ACO in the next part. This strategy proved to reach almost optimal solutions for highly non-convex problems. On

the other hand, Kaveh et al. [105] introduced Discrete Heuristic Particle Swarm Ant Colony Optimization (DHPSACO) incorporating a fly-back mechanism [106]. It was concluded to be a fast algorithm with high convergence speed. Niknam and Amiri [107] combined a fuzzy adaptive Particle Swarm Optimization, Ant Colony Optimization and the K-Means algorithm for clustering analysis over a number of benchmark datasets and obtained improved performance in terms of good clustering partitions. They applied Q-learning, a reinforcement learning technique to ACO to come up with Hybrid FAPSO-ACO-K Algorithm. They compared the results with respect to PSO-ACO, PSO, SA, TS, GA, ACO, HBMO, PSO-SA, ACO-SA, K-Means and obtained better convergence of FAPSO-ACO-K in most cases provided the number of clusters known beforehand.

Chen et al. [108] proposed a Genetic Simulated Annealing Ant Colony system infused with Particle Swarm Optimization. The initial population of Genetic Algorithms was given by ACO, where the interaction among different groups about the pheromone information was controlled by PSO. Next, GA controlled by SA mutation techniques were used to produce superior results. Xiong and Wang [109] used a two-stage hybrid algorithm (TAPC) combining adaptive ACO and enhanced PSO to overcome the local optima convergence problem in a K-means clustering application environment. Kıran et al. [110] came up with a novel hybrid approach (HAP) combining ACO and PSO. While initially the individual behavior of randomly allocated swarm of particles and colony of ants gets predominance, they start getting influenced by each other through the global best solution, which is determined by comparing the best solutions of PSO and ACO at each iteration. Huang et al [111] introduced continuous Ant Colony Optimization (ACOR) in PSO to develop hybridization strategies based on four approaches out of which a sequence based approach

using an enlarged pheromone-particle table proved to be most effective. The opportunity for ACOR in exploring the search space is more as solutions generated by PSO is associated with a pheromone-particle table. Mahi et al [112] came up with a hybrid approach combining PSO, ACO and 3-opt algorithm where the parameters concerning the optimization of ACO are determined by PSO and the 3-opt algorithm helps ACO avoid stagnation in local optima. Kefi et al. [113] proposed Ant-Supervised PSO (ASPSO) and applied it to the Travelling Salesman Problem (TSP) where the optimum values of the ACO parameters α and β, which used to determine the effect of pheromone information over the heuristic one, are updated by PSO instead of being constant as in traditional ACO. The pheromone amount and the rate of evaporation are detected by PSO: thus with the set of supervised and adjusted parameters given by PSO, ACO plays the key optimization methodology. Lazzus et al. [114] demonstrated vapor-liquid phase equilibrium by combining similar attributes of PSO and ACO (PSO+ACO), where the positions discovered by the particles of PSO were fine-tuned by the ants in the second stage through pheromone-guided techniques.

Mandloi et al. [115] presented a hybrid algorithm with a novel probabilistic search method by integrating the distance oriented search approach practiced by ants in ACO and velocity oriented search mechanism adopted by particles in PSO, thereby substituting the pheromone update of ACO with velocity update of PSO. The probability metric used in this algorithm consists of weighted heuristic values obtained from transformed distance and velocity through a sigmoid function, ensuring fast convergence, less complexity and avoidance of stagnation in local optima. Indadul et al. [116] solved the Travelling Salesman Problem (TSP) coordinating PSO, ACO and K-Opt Algorithm where the preliminary set of particle swarm is produced by ACO. In the latter iterations of PSO if the position of a particle is not

changed for a given interval, then K-Opt Algorithm is applied to it for upgrading the position. Liu et al. [117] relied on the local search capacity of ACO and global search potential of PSO and conglomerated them for application in optimizing container truck routes. Junliang et al. [118] proposed a Hybrid Optimization Algorithm (HOA) that exploits the merit of global search and fast convergence in PSO and in the event of premature convergence lets ACO take over. With its initial parameters set by PSO, the algorithm then converges to the optimal solution.

A brief listing of some of the important hybrid algorithms using PSO and ACO are given below in Table 3.4.

Table 3.4 A collection of hybridized PSO-ACO algorithms.

Author/s:	Year	Algorithm	Area of Application
Shelokar et al. [104]	2007	PSACO	Improved continuous optimization
Kaveh and Talatahari [105]	2009	DHPSACO	Truss structures with discrete variables
Kaveh and Talatahari [106]	2009	HPSACO	Truss structures
Niknam and Amiri [107]	2010	FAPSO-ACO-K	Cluster analysis
Chen et al. [108]	2011	ACO and PSO	Traveling salesman problem
Xiong and Wang [109]	2011	TAPC	Hybrid Clustering
Kıran et al. [110]	2012	HAP	Energy demand of Turkey
Huang et al. [111]	2013	ACOR	Data clustering
Mahi et al. [112]	2015	PSO, ACO and 3-opt algorithm	Traveling salesman problem
Kefi et al. [113]	2015	ASPSO	Traveling salesman problem
Lazzus et al. [114]	2016	PSO+ACO	Interaction parameters on phase equilibria
Mandloi and Bhatia [115]	2016	PSO, ACO	Large-MIMO detection
Indadul et al. [116]	2017	PSO, ACO and K-Opt Algorithm	Traveling salesman problem
Liu et al. [117]	2017	PSO, ACO	Container Truck Route optimization
Junliang et al. [118]	2017	HOA	Traveling salesman problem

3.2.8.5 Hybridization of PSO using Cuckoo Search (CS)

Cuckoo Search (CS) proposed by Xin-She Yang and Suash Deb [119] was developed on the basis of breeding behavior of cuckoos associated with the levy flight nature of birds and flies. Subsequently, Ghodrati and Lotfi [120] introduced Hybrid CS/PSO Algorithm capturing the ability of the cuckoos to communicate with each other in order to decrease the chances of their eggs being identified and abandoned by the host birds, by using Particle Swarm Optimization (PSO). In the course of migration each cuckoo records its personal best, thus generating the global best and governing their movements accordingly. Nawi et al. [121] came up with Hybrid Accelerated Cuckoo Particle Swarm Optimization (HACPSO) where the initial population of the nest are given by CS whereas Accelerated PSO (APSO) guides the agents towards the solution of the best nest. HACPSO was shown to perform classification problems with fast convergence and improved accuracy over constituent algorithms.

Enireddy and Kumar [122] proposed a hybrid PSO-CS for optimizing neural network learning rates. The meta parameter optimization of CS, i.e. the optimal values of the parameters of CS governing the rate of convergence of the algorithm were obtained through PSO which was shown to guarantee faster learning rate of neural networks with enhanced classification accuracy. Ye et al. [123] incorporated CS with PSO in the optimization of Support Vector Machine (SVM) parameters used for classification and identification of peer-to-peer traffic. At the beginning of each iteration, the optimal positions generated by PSO serve as the initial positions for CS and the position vectors of CS-PSO are considered as the pair of candidate parameters of SVM. The algorithm aims at finding the optimal tuning parameters of SVM through calculating the best position vectors.

Li and Yin [124] proposed a PSO inspired Cuckoo Search (PSCS) to model the update strategy by incorporating neighborhood as well as best individuals, balancing the exploitation and exploration capability of the algorithm. Chen et al. [125] combined the social communication of PSO and searching ability of CS and proposed PSOCS where cuckoos close to good solutions communicate with each other and move slowly near the optimal solutions guided by the global bests in PSO. This algorithm was used in training feedforward neural networks. Guo et al. [126] mitigated the shortcoming of PSO of getting trapped into local optima in high dimensional intricate problems through exploiting the random Levy step size update feature of CS, thus strengthening the global search ability. Also to overcome the slower convergence and lower accuracy of CS, they proposed hybrid PSOCS which initially uses Levy flight mechanism to search and then directs the particles towards optimal configuration through updating the positions given by PSO ensuring avoidance of local optima with randomness involved in Levy flights, thus providing improved performance.

Chi et al. [127] came up with a hybrid algorithm CSPSO where the initial population was based on the principles of orthogonal Latin squares with dynamic step size update using Levy flight process. The global search capability of PSO has been exploited ensuring information exchange among the cuckoos in the search process. Dash et al. [128] introduced improved cuckoo search particle swarm optimization (ICSPSO) where the optimization strategy of Differential Evolution (DE) Algorithm is incorporated for searching effectively around the group best solution vectors at each iteration, ensuring the global search capability of hybrid CSPSO and implemented this in designing linear phase multi-band stop filters.

A brief listing of some of important hybrid algorithms using PSO and CS are given below in Table 3.5.

Table 3.5 A collection of hybridized PSO-CS algorithms.

Author/s:	Year	Algorithm	Area of Application
Ghodrati and Lotfi [120]	2012	Hybrid CS/PSO	Global optimization
Nawi et al. [121]	2014	HACPSO	Classification
Enireddy and Kumar [122]	2015	Hybrid PSO CS	Compressed image classification
Ye et al. [123]	2015	Hybrid CSA with PSO	Optimization of parameters of SVM
Li and Yin [124]	2015	PSCS	Global optimization
Chen et al. [125]	2015	PSOCS	Artificial Neural Networks
Guo et al. [126]	2016	PSOCS	Preventive maintenance period optimization model
Chi et al. [127]	2017	CSPSO	Optimization problems
Dash et al. [128]	2017	ICSPSO	Linear phase multi-band stop filters

3.2.8.6 Hybridization of PSO using Artificial Bee Colony (ABC)

Artificial Bee Colony (ABC) was proposed by Karaboga and Basturk [129] and models the organized and distributed actions adapted by colonies of bees. Shi et al. [130] proposed an integrated algorithm based on ABC and PSO (IABAP) by parallelly executing ABC and PSO and exchanging information between the swarm of particles and colony of bees. El-Abd [131] combined ABC and Standard PSO (SPSO) to update the personal best in SPSO using ABC at each iteration and applied it in continuous function optimization. Kıran and Gündüz [132] came up with a hybrid approach based on Particle Swarm Optimization and Artificial Bee Colony algorithm (HPA) where at the end of each iteration, recombination of the best solutions obtained by PSO and ABC takes place and the result serves as global best for PSO and neighbor for the onlooker bees in ABC, thus enhancing the exploration-exploitation capability of the algorithm.

Xiang et al. [133] introduced a Particle Swarm inspired Multi-Elitist ABC algorithm (PS-MEABC) in order to enhance the exploitation strategy of ABC algorithm by modifying the food source parameters in on-looker or employed bees phase through the global best as well

as an elitist selection from the elitist archive. Vitorino et al. [134] came up with a way to deal with the issue that PSO is not always able to employ its exploration and exploitation mechanism in a well-adjusted manner, by trying a mitigation approach using the diversifying capacity of ABC when the agents stagnate in the search region. As the particles stagnate, the ABC component in Adaptive PSO (APSO) introduces diversity and enables the swarm to balance exploration and exploitation quotients based on fuzzy rules contingent upon swarm diversity.

Lin and Hsieh [135] used Endocrine-based PSO (EPSO) compensating a particle's adaptability by supplying regulatory hormones controlling the diversity of the particle's displacement and scope of search and combined it with ABC. The preliminary food locations of employed bees phase in ABC is supplied by EPSO's individual bests controlled by global bests. Then onlooker bees and scout bees play an important role in improving the ultimate solution quality. Zhou and Yang [136] proposed PSO-DE-PABC and PSO-DE-GABC based on PSO, DE and ABC to cope with the lack of exploitation plaguing ABC. Divergence is enhanced by creating new positions surrounding random particles through PSO-DE-PABC, whereas PSO-DE-GABC creates the positions around global best with the divergence being taken care by differential vectors and Dimension Factor (DF) optimizing the rate of search. The novelty in the scouting technique enhances search at the local level.

Li et al. [137] introduced PS-ABC comprising a local search phase of PSO and two global search phases of ABC. Depending on the extent of aging of personal best in PSO each entity at each iteration adopts either of PSO, onlooker bee or scout bee phase. This algorithm was shown to be efficient for high dimensional datasets with faster convergence. Sedighizadeh and Mazaheripour [138] came up with a PSO-ABC algorithm with initially assigned personal

bests for each entity and further refining it through a PSO phase and ABC phase and the best of all personal bests is returned as global best at the end. This algorithm was shown to find the optimal route faster in vehicle routing problems compared to some competing algorithms. A brief listing of some of the important hybrid algorithms using PSO and ABC are given below in Table 3.6.

Table 3.6 A collection of hybridized PSO-ABC algorithms.

Author/s:	Year	Algorithm	Area of Application
Shi, et al. [130]	2010	IABAP	Global optimization
El-Abd [131]	2011	ABC-SPSO	Continuous function optimization
Kıran and Gündüz [132]	2013	HPA	Continuous optimization problems
Xiang, et al. [133]	2014	PS-MEABC	Real parameter optimization
Vitorino, et al. [134]	2015	ABeePSO	Optimization problems
Lin and Hsieh [135]	2015	EPSO_ABC	Classification of Medical Datasets Using SVMs
Zhou and Yang [136]	2015	PSO-DE-PABC and PSO-DE-GABC	Optimization problems
Li, et al. [137]	2015	PS-ABC	High-dimensional optimization problems
Sedighizadeh and Mazaheripour [138]	2017	PSO-ABC	Multi objective vehicle routing problem

3.2.8.7 Hybridization of PSO using Other Social Metaheuristic Approaches

The following section discusses some instances where the Particle Swarm Optimization algorithm has been hybridized with other commonly used social metaheuristic optimization algorithms for use in an array of engineering applications. Common techniques include Artificial Immune Systems [139-142], Bat Algorithm [143], Firefly Algorithm [144] and Glow Worm Swarm Optimization Algorithm [145].

3.2.8.7.1 Artificial Immune Systems (AIS)

Zhao et al. [146] introduced a human-computer cooperative PSO based Immune Algorithm (HCPSO-IA) for solving complex layout design problems. The initial population is supplied by the user and the initial algorithmic solutions are generated by a chaotic strategy. By introducing new artificial individuals to replace poor performing

individuals of the population, HCPSO-IA can be refined to incorporate a man-machine synergy by using knowledge about which key performance indices such as envelope area and static non-equilibrium value can be significantly increased.

El-Shirbiny and Alhamali [147] used a Hybrid Particle Swarm with Artificial Immune Learning (HPSIL) to solve Fixed Charge Transportation Problems (FCTPs). In the proposed algorithm a flexible particle structure, decoding as well as allocation scheme are used instead of a Prufer number and a spanning tree is used in Genetic Algorithms. The authors noted that the allocation scheme guaranteed finding an optimal solution for each of the particles and that the HPSIL algorithm can be implemented on both balanced and unbalanced FCTPs while not introducing any dummy supplier or demand.

3.2.8.7.2 Bat Algorithm (BA)

A communication strategy of hybrid PSO with Bat Algorithm (BA) was proposed in [148] by Pan et al. wherein several worst performing particles of PSO are replaced by the best performing ones in BA and vice-versa, after executing a fixed number of iterations. This communication strategy facilitates information flow between PSO and BA and can reinforce the strengths of each algorithm between function evaluations. Six benchmark functions were tested producing improvements in convergence speed and accuracy over either of PSO or BA. An application of hybrid PSO-BA in medical image registration was demonstrated by Manoj et al [149] where the authors noted the hybrid algorithm was more successful in finding optimal parameters for the problem as compared to relevant methods in use.

3.2.8.7.3 Firefly Algorithm (FA)

To utilize different advantages of PSO and Firefly Algorithm (FA), Xia et al. [150] proposed three novel operators in a hybrid algorithm (FAPSO) based on the two. During the optimization, the population is divided into two groups and each choses PSO or FA as their basic search technique for parallel executions. Apart from this, the information exchange about optimal solutions in case of stagnation, a knowledge based detection operator and a local search for tradeoff between exploration and exploitation as well as the employment of a BFGS Quasi-Newton method to enhance exploitation led to several important observations. For instance, the exchange of optimal solutions led to an enriched and diverse population and the inclusion of the detection operator and local search was justifiable for multimodal optimization problems where refinement of solution quality was necessary. Arunachalam et al. [151] presented a new approach to solve the Combined Economic and Emission Dispatch (CEED) problem having conflicting economic and emission objective using a hybrid of PSO and FA.

3.2.8.7.4 Glow Worm Swarm Optimization (GSO)

Shi et al. [152] introduced a hybrid PSO and Glow Worm Swarm (GSO) algorithm (HEPGO) based on selective ensemble learning and merits of PSO and GSO. HEPGO leverages the ability of the GSO to capture multiple peaks of multimodal functions due to its dynamic sub-groups and the global exploration power of the PSO and provides promising results on five benchmark minimization functions. Liu and Zhou [153] introduced GSO in the working of PSO to determine a perception range, within the scope of perception of all particles to find an extreme value point sequence. A roulette wheel selection

scheme for picking a particle as the global extreme value is followed and the authors note that this can overcome the convergence issues faced by PSO.

A brief listing of some of the important hybrid algorithms using PSO and other approaches such as AIS, BA, FA and GSO are given below in Table 3.7.

Table 3.7 A collection of PSO algorithms hybridized with approaches as AIS, BA, FA and GSO

Author/s:	Year	Algorithm	Area of Application
Shi et al. [152]	2012	HEPGO	Global Optimization
El-Shirbiny and Alhamali [147]	2013	HPSIL	Fixed Charge Transportation Problems
Liu and Zhou [153]	2013	New (GSO-PSO)	Constrained Optimization
Zhao et al [146]	2014	HCPSO-IA	Complex layout design problems
Arunachalam et al. [151]	2014	HPSOFF	Combined Economic and Emission Dispatch Problem
Pan et al. [148]	2015	Hybrid PSO-BA	Global Optimization
Manoj et al. [149]	2016	PSO-BA	Medical Image Registration
Xia et al. [150]	2017	FAPSO	Global Optimization

3.2.9 Parallelized Implementations of PSO

The literature points to several instances of PSO implementations on parallel computing platforms. The use of multiple processing units onboard a single computer renders feasibility to the speedup of independent computations in the inherently parallel structure of PSO. Establishing sub-swarm based parallelism leads to different processors being assigned to sub-swarms with some mechanism of information exchange among them. On the other hand, master-slave configurations attempt to designate a master processor which assigns slave processors to work on fitness evaluation of many particles simultaneously. Early work by Gies and Rahmat-Samii reported a performance gain of 8 times using a system with 10 nodes for a parallel implementation over s serial one [154]. Schutte et al. [155] evaluated a parallel implementation of the algorithm on two types of test problems: a) large scale analytical problems with inexpensive function evaluations and b) medium scale problems on biomechanical system identification with computationally heavy

function evaluations. The results of experimental analysis under load-balanced and load-imbalanced conditions highlighted several promising aspects of parallelization. The authors used a synchronous scheme based on a master-slave approach. The use of data pools in [156], independent evaluation of fitness leading to establishing the dependency of efficiency on the social information exchange strategy in [157] and exploration of enhanced topologies for information exchange in multiprocessor architectures in [158] may be of relevance to an interested reader.

Rymut's work on parallel PSO based Particle Filtering showed how CUDA capable Graphics Processing Units (GPUs) can accelerate object tracking algorithm performances using adaptive appearance models [159]. A speedup factor of 40 was achieved by using GPUs over CPUs. Zhang et al. fused GA and PSO to address the sample impoverishment problem and sample size dependency in particle filters [160-161]. Chen et al. [162] proposed an efficient parallel PSO algorithm to find optimal design criteria for Central Composite Discrepancy (CCD) criterion whereas Awwad used a CUDA based approach to solve the topology control issue in hybrid radio frequency and wireless networks in optics [163]. Qu et al. used a serial and parallel implementation of PSO in the Graph Drawing problem [164] and reported that both methods are as effective as the force-directed method in the work, with the parallel method being superior to the serial one when large graphs were considered. Zhou et al. found that using a CUDA implementation of the Standard PSO (SPSO) with a local topology [165] on four benchmark problems, the runtime of GPU-SPSO indicates clear superiority over CPU-SPSO. They also noted that runtime and swarm size assumed a linear relationship in case of GPU-SPSO. Mussi et al. reported in [166] an in-depth performance evaluation of two variants of parallel algorithms with the sequential implementation of PSO over standard

benchmark functions. The study included assessing the computational efficiency of the parallel methods by considering speedup and scaleup against the sequential version.

3.3 Niche Formation and Multi-objective Optimization

A function is multimodal if it has more than one optimum. Multimodal functions may have one global optimum with several local optima or more than one global optimum. In the latter case, optimization algorithms are refined appropriately to be effective on multimodal fitness landscapes as two specific circumstances may arise otherwise. First, standard algorithms used may be unable to distinguish among the promising regions and settle on a single optimum. Second, the algorithm may not converge to any optima at all. In both cases, multi-objective optimization criteria are not satisfied and further modifications are needed.

3.3.1 Formation of Niches in PSO

The formation of niches in swarms is inspired by the natural phenomenon of co-existence of species who are competing and co-evolving for shared resources in a social setting. The work of Parsopoulous et al. [167-168] was among the first ones to appropriately modify PSO to make it suitable for handling multimodal functions with multiple local optima through the introduction of function "stretching" whereby fitness neighborhoods are adaptively modified to remove local optima. A sequential niching technique proposed by Parsopoulous and Vrahatis [169] identified possible solutions when their fitness dropped below a certain value and raised them, at the same time removing all local optima violating the threshold constraint. However, the effectiveness of the stretching method is not uniform on all objective functions and introduced false minima in some cases [33]. This approach was improved by the introduction of the *Deflection* and *Repulsion* techniques in [170] by Parsopoulous and Vrahatis. The nbest PSO by Brits et al. [171] used local neighborhoods

based on spatial proximity and achieved a parallel niching effect in a swarm whereas the NichePSO by the same authors in [172] achieved multiple solutions to multimodal problems using sub-swarms generated from the main swarm when a possible niche was detected. A speciation-based PSO [173] was developed keeping in mind the classification of particles within a threshold radius from the neighborhood best (also known as the seed), as those belonging to a particular species. In an extension which sought to eliminate the need for a user specified radius of niching, the Adaptive Niching PSO (ANPSO) was proposed in [174-175]. It adaptively determines the radius by computing the average distance between each particle and its closest neighbor. A niche is said to have formed if particles are found to be within the niching radius for an extended number of iterations in which case particles are classified into two groups: niched and un-niched. A global PSO is used for information exchange within the niches whereas an lbest PSO with a Von-Neumann topology is used for the same in case of un-niched particles. Although ANPSO eliminates the requirement of specifying a niche radius beforehand, the solution quality may become sensitive due to the addition of new parameters. The Fitness Euclidean Distance Ratio PSO (FER-PSO) proposed by Li [176] uses a *memory swarm* alongside an *explorer swarm* to guide particles towards the promising regions in the search space. The memory swarm is constructed out of personal bests found so far whereas the explorer swarm is constructed out of the current positions of the particle. Each particle is attracted towards a fittest and closest point in the neighborhood obtained by computing its Fitness Euclidean Ratio (FER). FER-PSO introduces a scaling parameter in the computation of FER, however it can reliably locate all global optima when population sizes are large. Clustering techniques such as k-means have been incorporated into the PSO framework by Kennedy [177] as well as by Passaro and

Starita [178] who used the Bayesian Information Criterion (BIC) [179] to estimate the parameter k and found the approach comparable to SPSO and ANPSO.

3.3.2 Niching in Dynamic Environments and Challenges

Several hard challenges are posed by environments which are dynamic as well as multimodal, however sub-population based algorithms searching in parallel are an efficient way to locate multiple optima which may undergo any of shape, height or depth changes as well as spatial displacement. Multi-Swarm PSO proposed by Blackwell et al. [180], rPSO by Bird and Li [181], Dynamic SPSO by Parrot and Li [182] and the *lbest* PSO with Ring Topology by Li [183] are some of the well-known approaches used in such environments. Since most niching algorithms utilize global information exchange at some point in their execution, their best case computational complexity is O (N^2). This issue coupled with performance degradation in high dimensional problems and the parameter sensitivity of solutions make niching techniques an involved process for any sufficiently complex multimodal optimization problem.

3.4 Discrete Hyperspace Optimization

3.4.1 Variable Round-Off

Discrete variables are rounded off to their nearest values by clamping at the end of every iteration or at the end of the optimization process and can offer significant speedup. However, unintelligent round-offs may throw the particle towards a comparably infeasible region and result worse fitness values. With that said, some studies have shown interesting results and garnered attention for a discrete version of PSO (DPSO).

3.4.2 Binarization

A widely used binarization approach maps the updated velocity at the end of an iteration into the closed interval [0,1] using a sigmoid function. The updated velocity represents the probability that the updated position takes the value of 1 since a high enough velocity implies the sigmoid function outputs 1. The value of the maximum velocity is often clamped to a low value to make sure there is a chance of a reversal of sigmoid output value. Afshinmanesh et al. [184] used modified flight equations based on XOR and OR operations in Boolean algebra. Negative selection mechanisms in immune systems inspire a velocity bounding constraint on such approaches. Deligkaris et al [185] used a mutation operator on particle velocities to render better exploration capabilities to the Binary PSO.

3.4.3 Set Theoretic Approaches

Chen et al. [186] used a set representation approach to characterize the discrete search spaces of combinatorial optimization problems. The solution is represented as a crisp set and the velocity as a set of possibilities. The conventional operators in position and velocity update equations of PSO are replaced by operators defined on crisp sets and sets of possibilities, thus enabling a structure similar to PSO but with applicability to a discrete search space. Experiments on two well-known discrete optimization problems, viz. the Traveling Salesman Problem (TSP) and the Multidimensional Knapsack Problem (MKP) demonstrated the promising nature of the discrete version. Gong et al. [187] proposed a set-based PSO to solve the Vehicle Routing Problem (VRP) with Time Windows (S-PSO-VRPTW) with the general method of selecting an optimal subset from a universal set using the PSO framework. S-PSO-VRPTW considers the discrete search space as an arc set of the complete graph represented by the nodes in VRPTW and regards a potential solution as a

subset of arcs. The designed algorithm when tested on Solomon's datasets [188] yielded superior results in comparison to existing state-of-the-art methodologies.

3.4.4 Penalty Approaches

KitaYama et al. [189] setup an augmented objective function with a penalty approach such that there is a higher incentive around discrete values. Points away from discrete values are penalized and the swarm effectively explores a discrete search space, although at a heavy computational burden for complex optimization problems. By incorporating the penalty term the augmented objective function turns non-convex and continuous, thereby making it suitable for a PSO based optimization.

3.4.5 Hybrid Approaches

Nema et al. [190] used the deterministic Branch and Bound algorithm to hybridize PSO in order to solve the Mixed Discrete Non-Linear Programming (MDNLP) problem. The global search capability of PSO coupled with the fast convergence rate of Branch and Bound reduces the computational effort required in Non-Linear Programming problems. Sun et al. [191] introduced a constraint preserving mechanism in PSO (CPMPSO) to solve mixed-variable optimization problems and reported competitive results when tested on two real-world mixed-variable optimization problems. Chowdhury et al. [192] considered the issue of premature stagnation of candidate solutions especially in single objective, constrained problems when using PSO and noted its pronounced effect in objective functions that make use of a mixture of continuous and discrete design variables. In order to address this issue, the authors proposed a modification in PSO which made use of continuous optimization as its primary strategy and subsequently a nearest vertex approximation criterion for updating of discrete variables. Further incorporation of a diversity preserving mechanism introduced

a dynamic repulsion directed towards the global best in case of continuous variables and a stochastic update in case of discrete ones. Performance validation tests were successfully carried out over a set of nine unconstrained problems and a set of 98 Mixed Integer Non-Linear Programming (MINLP) problems.

3.4.6 Some Application Instances

Laskari et al [193] tested three variants of PSO against the popular Branch and Bound method on seven different integer programming problems. Experimental results indicated that the behavior of PSO was stable in high dimensional problems and in cases where Branch and Bound failed. The variant of PSO using constriction factor and inertia weight was the fastest whereas the other variants possessed better global exploration capabilities. Further observation also supported the claim that the variant of PSO with only constriction factor was significantly faster than the one with only inertia weight. These results affirmed that the performance of the variants was not affected by truncation of the real parameter values of the particles. Yare and Venayagamoorthy [194] used a discrete PSO for optimal scheduling of generator maintenance, Eajal and El-Hawary [195] approached the problem of optimal placement and sizing of capacitors in unbalanced distribution systems with the consideration of including harmonics. More recently, Phung et al. [196] used a discretized version of PSO path planning for UAV vision-based surface inspection and Gong et al. [197] attempted influence maximization in social networks. Aminbakhsh and Sonmez [198] presented a Discrete Particle Swarm Optimization (DPSO) for an effective solution to large-scale Discrete Time-Cost Trade-off Problem (DTCTP). The authors noted that the experiments provided high quality solutions for time-cost optimization of large size projects within seconds and enabled optimal planning of real life-size projects. Li et al. [199]

modeled complex network clustering as a multi-objective optimization problem and applied a quantum inspired discrete particle swarm optimization algorithm with non-dominated sorting for individual replacement to solve it. Experimental results illustrated its competitiveness against some state-of-the-art approaches on the extensions of Girvan and Newman benchmarks [200] as well as many real-world networks. Ates et al. [201] presented a discrete Infinite Impulse Response (IIR) filter design method for approximate realization of fractional order continuous filters using a Fractional Order Darwinian Particle Swarm Optimization (FODPSO).

3.5 Ensemble Particle Swarm Optimization

The No Free Lunch (NFL) Theorem [203] by Wolpert and Macready establishes that no single optimization algorithm can produce superior results when averaged over all objective functions. Instead, different algorithms perform with different degrees of effectiveness given an optimization problem. To this end, researchers have tried to put together ensembles of optimizers to obtain a set of candidate solutions given an objective function and choose from the promising ones. Existing ensemble approaches such as the Multi-Strategy Ensemble PSO (MEPSO) [202] uses a two-stage approach: Gaussian local search strategy to improve convergence capability and Differential Mutation (DM) to increase the diversity of the particles. The Heterogenous PSO in [204] uses a pool of different search behaviors of PSO and empirically outperforms the homogenous version of PSO. The Ensemble Particle Swarm Optimizer by Lynn and Suganthan [205] uses a pool of PSO strategies and gradually chooses a suitable one through a merit-based scheme to guide the particles' movement in a particular iteration.Shirazi et al. proposed a Particle Swarm Optimizer with an ensemble of inertia weights in [206] and tested its effectiveness by incorporating it into a heterogenous

comprehensive learning PSO (HCLPSO) [207]. Different strategies such as linear, logarithmic, exponential decreasing, Gompertz, chaotic and oscillating inertia weights were considered and compared against other strategies on a large set of benchmark problems with varying dimensions to demonstrate the suitability of the proposed algorithm.

3.6 Notes on Benchmark Solution Quality and Performance Comparison Practices

3.6.1 Performance on Simple Benchmarks

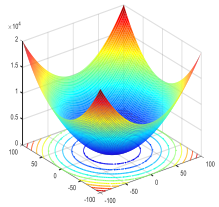
To provide an intuitive understanding of the performances of some of the many PSO-based algorithmic variants, let us consider a few commonly used unimodal and multimodal benchmark functions. These functions (F1-F8) are either unimodal, simple multimodal or unrotated multimodal ones. The purpose of the following table is to provide a first-course reference for introductory purposes, however for a full-scale performance analysis one should consider rotated multimodal and compositional functions as well – there should be a good mix of separable and non-separable benchmarks before any inference on accuracy and/or efficiency is drawn.

Table 3.8 Benchmark Functions F1-F8

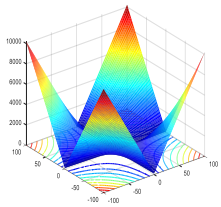
Function	Name	Expression	Range	Min
F1	Sphere	$f(x) = \sum_{i=1}^{n} x_i^2$	[-100,100]	f(x*) = 0
F2	Schwefel's Problem 2.22	$f(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	[-10,10]	f(x*) = 0
F3	Schwefel's Problem 1.2	$f(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2$	[-100,100]	f(x*) = 0
F4	Generalized Rosenbrock's Function	$f(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	[-n,n]	f(x*) = 0
F5	Generalized Schwefel's Problem 2.26	$f(x) = -\sum_{i=1}^{n} (x_i \sin(\sqrt{\lvert x_i \rvert}))$	[-500,500]	f(x*) = -12569.5
F6	Generalized Rastrigrin's Function	$f(x) = An + \sum_{i=1}^{n} [x_i{}^2 - A\cos(2\pi x_i)]$, A=10	[-5.12, 5.12]	f(x*) = 0
F7	Ackley's Function	$f(x) = -20 \exp\left(-0.2 \sqrt{\frac{1}{d} \sum_{i=1}^{d} x_i^2} \right) - \exp\left(\sqrt{\frac{1}{d} \sum_{i=1}^{d} \cos(2\pi x_i)} \right) + 20 + \exp(1)$	[-32.768, 32.768]	f(x*) = 0
F8	Generalized Griewank Function	$f(x) = 1 + \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}})$	[-600,600]	f(x*) = 0

Table 3.9 3D Plots of the Benchmark Functions

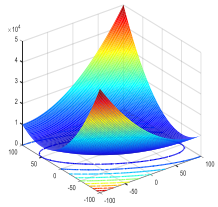
F1



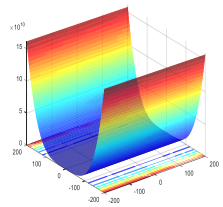
F2



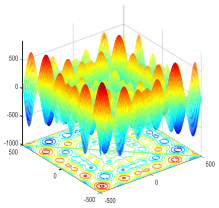
F3



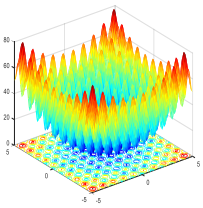
F4



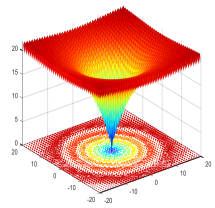
F5



F6



F7



F8

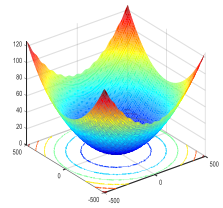


Table 3.10 Performances of Some Variants of PSO on F1-F8

Function	Performance	PSO [208]	PSO [209]	PSO [210]	PSOGSA [210]	DEPSO [211]
F1	Mean	1.36e-04	1.8e-03	2.83e-04	6.66e-19	1.60e−26
	St. Dev	2.02e-04	NR	NR	NR	6.56e−26
F2	Mean	4.21e-02	2.0e+0	5.50e-03	3.79e-09	2.89e−13
	St. Dev	4.54e-02	NR	NR	NR	1.54e−12
F3	Mean	7.01e+01	4.1e+03	5.19e+3	4.09e+02	3.71e−01
	St. Dev	2.21e+01	NR	NR	NR	2.39e−01
F4	Mean	9.67e+01	3.6e+04	2.01e+02	5.62e+01	4.20e+01
	St. Dev	6.01e+01	NR	NR	NR	3.28e+01
F5	Mean	-4.84e+03	-9.8e+03	-5.92e+03	-1.22e+04	4.68e+03
	St. Dev	1.15e+03	NR	NR	NR	9.42e+02
F6	Mean	4.67e+01	5.51e+01	7.23e+01	2.27e+01	4.07e+01
	St. Dev	1.16e+01	NR	NR	NR	1.19e+01
F7	Mean	2.76e-01	9.0e-03	4.85e-10	6.68e-12	2.98e−13
	St. Dev	5.09e-01	NR	NR	NR	1.51e−12
F8	Mean	9.21e-03	1.0e-02	5.43e-03	1.48e-03	1.69e−02
	St. Dev	7.72e-03	NR	NR	NR	1.82e−02

*NR – Not Reported

The results in Table 3.10 provide a high-level, intuitive understanding of benchmarking experiments using PSO and a few of its variants. In order to comment on the performance of the algorithms, statistical significance tests with an appropriate confidence level (generally alpha=0.01 or 0.05) are commonly carried out.

3.6.2 Studies on Performance Comparison Practices

Sergeyev et al. [212] proposed a visual technique for comparison of different approaches towards global optimization problems from both stochastic point of view with metaheuristics inspired by nature as well as deterministic approach based on mathematical programming. They have presented proposed and aggregated operational zones for effective comparison of deterministic and stochastic algorithms for various computational budgets. They commented on the competitive nature of the two algorithms and the fact that they surpass each other based on the available cost function evaluations. However, one shortcoming of this approach is the apparent under-performance of an algorithm with respect

to its potential, given that it stagnates in a local minimum before the users' computational budget is exhausted. In order to work around this the authors put forward two budgetary upper limits viz. n_{max} and N_{max} (typically, $N_{max} >> n_{max}$). The underlying strategy is to let any algorithm operate on any objective function within an upper limit of $n_{max..}$ It is only necessary to check if the function is approximated within the global budget N_{max} or if a successful trial is reported conditioned upon success in either an individual trial with a maximum local budget of n_{max} as well as in a batch of trials each with the same local budget of n_{max}. Post-optimization data is used to construct *aggregate operational zones*. This approach relies on different reinitializations across k different local budgets with the aim of maximizing an exploration-exploitation gain, while keeping the global budget constant. It is important to note that different trials may require different runtimes to converge and that the condition to check if an objective function is *successfully* approximated is rather flexible.

Kvasov and Mukhametzhanov [213-214] considered the problem of finding the global optimum $\boldsymbol{f^*}$ and the corresponding argument $\boldsymbol{x^*}$ of continuous and finite-dimensional objective functions (specifically, unidimensional ones) of multimodal, non-differentiable nature from a constrained optimization standpoint. Testing was carried out on 134 multimodal, constrained functions of univariate nature with respect to various performance comparison indices, totaling over 125,000 trials using 13 test methods. The experimental results provide critical insight into the comparative efficiencies of Lipschitz-based deterministic approaches versus nature-inspired metaheuristic ones, with future directions pointed at an extension of similar analyses for the multidimensional case. Readers are directed to [213] for an involved understanding of the test methodology and to [215-216] for comparison criteria for application and validation in some practical test problems.

CHAPTER 4

THE QUANTUM DOUBLE DELTA SWARM ALGORITHM

4.1 Swarm Propagation Using a Double Delta Potential Well

We start from the time-independent Schrodinger's wave equation which is stated as:

$$[-\frac{\hbar^2}{2m}\nabla^2 + V(r)]\psi(r) = E\,\psi(r) \tag{4.1}$$

ψ(r), *V(r)*, *m*, *E* and ħ represent the wave function, the potential function, the reduced mass, the energy of the particle and Planck's constant respectively. Let us consider a particle in a double delta well, whose potential can be expressed as:

$$V(r) = -\alpha\{\delta(r+a) + \delta(r-a)\} \tag{4.2}$$

α expresses the strength of the well and {-*a*, *a*} are the centers of the two wells. Considering the even solutions of the time-independent Schrodinger's equation (Eq. 4.1) and assuming *V*= 0 at regions away from the centers of the two wells we get:

$$-\frac{\hbar^2}{2m}\frac{d^2}{dr^2}\psi(r) = E\psi(r) \tag{4.3}$$

The intention is to find solutions of the wave function ψ in a double delta well setup for E<0 (bound states) in regions $\mathbb{R}1:\ r \in (-\infty, a),\ \mathbb{R}2:\ r \in (-a, a)\ and\ \mathbb{R}3:\ r \in (a, \infty)$. Assuming k to be equal to $(\sqrt{2mE}/\hbar)$, the even solutions can be expressed as [223]:

$$\psi_e(x) = \begin{cases} Ae^{-kr} & r > a \\ Be^{-kr} + Ce^{kr} & 0 < r < a \\ Be^{kr} + Ce^{-kr} & -a < r < 0 \\ Ae^{kr} & r < -a \end{cases} \tag{4.4}$$

To solve for the constants described in the above equation, we solve for the continuity of the wave function ψ_e at $r = a$ and $r = -a$ and for the continuity of the derivative of the wave function at $r = 0$. Hence, we arrive at the equations for ψ_e stated below [223]:

$$\psi_e(r) = \begin{cases} B(1+e^{2ka})e^{-kr} & r > a \\ B(e^{-kr}+e^{kr}) & -a < r < a \\ B(1+e^{2ka})e^{kr} & r < -a \end{cases} \tag{4.5}$$

We are not considering the solution of the odd wave function ψ_o here as the existence of a solution is not guaranteed [223]. It is also interesting to note that the bound state energy in a double delta potential well is lower than that compared to a single delta potential well by approximately by a factor of $(1.11)^2 \approx 1.2321$ [222]. Next, if we consider the behavioral dynamics of a particle to be compliant with the Schrodinger wave equation i.e. Eq. (4.1), then we need to find the particle's probability density function for its behavioral characterization, which is given by the square of the magnitude of the wave function described in Eq. (4.5). Thus, we have to find $|\psi(r)^2|$. In order to say that there is a greater than 50% chance of finding a particle in the vicinity of the center of any of the potential wells, the following criterion must be satisfied [217]:

$$\int_{-|r|}^{|r|} \psi(r)^2 \, dr > 0.5 \tag{4.6}$$

where $-|r|$ and $|r|$ denote the two boundaries of the vicinity. Although we are trying to confine the particle in any one of the two potential wells, the wave function considered here is different from that considered in the case of a single potential well setup (as in traditional QPSO), This is due to the influence of the other well which is taken into consideration when deriving conditions for the confinement of a particle in a single well. Eq. (4.6) can also be written as:

$$\int_{-|r|}^{|r|} \psi(r)^2 \, dr = 0.5g \qquad (1 < g < 2) \tag{4.7}$$

For ease of computation, we now consider that one of the wells is centered at 0. Solving for conditions of confinement of the particle in that well and computing $\int_{-|r|}^{|r|}(\psi(r)^2 \, dr$ for regions $\mathbb{R}2_{0-}$: $r' \in (-r, 0)$ by applying the second condition of Eq. (4.5) and $\mathbb{R}2_{0+}$: $r' \in (0, r)$ by

applying the first condition of Eq. (4.5), we arrive at the equation below:

$$B^2 = \frac{kg}{e^{2kr} - 5e^{-2kr} + 4kr + 4} \tag{4.8}$$

We replace the denominator of the R.H.S. ($e^{2kr} - 5e^{-2kr} + 4kr + 4$) as δ. Thus, we get:

$$\delta = e^{2kr} - 5e^{-2kr} + 4kr + 4 \tag{4.9}$$

Equating B^2 in L.H.S. of equation (4.8) for any two consecutive iterations (assuming it is a constant over iterations as it not a function of time) we get Eq. (4.10), Eq. (4.11) and Eq. (4.12):

$$\frac{g_{iter-1}}{e^{2kr_{iter-1}} - 5e^{-2kr_{iter-1}} + 4kr_{iter-1} + 4} = \frac{g_{iter}}{e^{2kr_{iter}} - 5e^{-2kr_{iter}} + 4kr_{iter} + 4} \tag{4.10}$$

$$\Rightarrow \frac{g_{iter-1}}{\delta_{iter-1}} = \frac{g_{iter}}{\delta_{iter}} \tag{4.11}$$

$$\Rightarrow \delta_{iter} = G.\ \delta_{iter-1} \qquad (0.5 < G < 2) \tag{4.12}$$

G is the ratio (g_{iter}/g_{iter-1}) and can vary from 0.5 to 2 since (1<g<2). To keep a particle moving towards the center of a potential well we find $\delta_{iter} \mid (0.5\ \delta_{iter-1} < \delta_{iter} < 2\ \delta_{iter-1})$. Thus, we find δ_{iter} at the current iteration based on δ_{iter-1} (found in the previous iteration) by adding or subtracting the gradient of δ_{iter-1} multiplied by a learning rate. The governing conditions of finding δ_{iter} depend on the relation of δ_{iter-1} with its version in the previous iteration, i.e., δ_{iter-2} as well as the sign of the gradient of δ_{iter-1} as described in Algorithm 4.1. The learning rate θ is designed as:

$$\theta = (1 - \epsilon)\left(\frac{max.\ iterations - current\ iteration}{max.\ iterations}\right) + \epsilon \tag{4.13}$$

ϵ is a small fraction between [0,1] set by the user. The learning rate θ decreases linearly from 1 to ϵ with the passage of iterations. Once we obtain δ_{iter}, we back-solve Eq. (4.9) to retrieve r_{iter} which denotes a particle's position as well as the potential solution for that particular iteration. We let r_{iter}, i.e. a particle's position in the current iteration maintain a component towards the best position found so far *(gbest)* along with its current solution obtained from

δ_{iter}. Subsequently, a cost function is computed with the solution r_{iter} and if it is better than the best cost found thus far, the cost and the corresponding solution are stored in memory. This process is repeated over the total number of evaluations to find the overall best cost and best solution for the swarm.

4.2 Pseudocode of QDDS

In this section, the pseudocode of the QDDS Algorithm is presented:

Algorithm 4.1. Quantum Double Delta Swarm Algorithm

Initialization Phase

Initialize k
Initialize a small constant λ randomly
Initialize maximum no. of iterations as maxiter
Initialize bestcost
for each particle
    for each dimension
        Initialize positions r_1 and r_2 for iterations 1 and 2
    end for
end for
Generate δ_1 and δ_2 from r_1 and r_2 according to Eq. (4.9)
Set iteration count iter=3

Optimization Phase

while ***(iter<maxiter) and***
    {($\delta_{iter-1} < 0.5 * \delta_{iter-2}$) or ($\delta_{iter-1} > 2 * \delta_{iter-2}$)}
    Find learning rate θ using eq. (4.13)
    Select a particle randomly
    for ***each dimension***
        if ***($\delta_{iter-1} > 2 * \delta_{iter-2}$) and $\nabla\delta_{iter-1}$>0***
            $\delta_{iter} = \delta_{iter-1} - \theta * \nabla\delta_{iter-1} * \lambda$
        elseif ***($\delta_{iter-1} > 2 * \delta_{iter-2}$) and $\nabla\delta_{iter-1}$<0***
            $\delta_{iter} = \delta_{iter-1} + \theta * \nabla\delta_{iter-1} * \lambda$
        elseif ***($\delta_{iter-1} < 0.5 * \delta_{iter-2}$) and $\nabla\delta_{iter-1}$<0***
            $\delta_{iter} = \delta_{iter-1} - \theta * \nabla\delta_{iter-1} * \lambda$
        elseif ***($\delta_{iter-1} < 0.5 * \delta_{iter-2}$) and $\nabla\delta_{iter-1}$>0***
            $\delta_{iter} = \delta_{iter-1} + \theta * \nabla\delta_{iter-1} * \lambda$
        end if
    end for
    Solve r_{iter} from δ_{iter}
    Generate a random number ρ between 0 and 1
    r_{iter}=$\rho * r_{iter} + (1-\rho) * r_{gbest}$
    Compute cost using r_{iter}
    if ***$cost_{iter}$<bestcost***
        bestcost = $cost_{iter}$
        best_solution = r_{iter}
    end if
    iter = iter + 1
end while

4.3 Benchmark Functions

To evaluate the performance of the proposed algorithm, benchmark functions such as Rastrigrin, Rosenbrock, Sphere and Griewank have been considered.

Table 4.1 Benchmark Functions Considered for Testing

Function	Expression	Min
Rastrigrin	$f(x) = An + \sum_{i=1}^{n}[x_i{}^2 - Acos(2\pi x_i)]$, A=10	0
Rosenbrock	$f(x) = \sum_{i=1}^{n-1}[100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2]$	0
Sphere	$f(x) = \sum_{i=1}^{n} x_i^2$	0
Griewank	$f(x) = 1 + \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})$	0

4.4 Simulation Results on the Benchmark Functions

Table 4.2 Results for the Rosenbrock Function using 10 Independent Trials of QDDS

P	Dim	PSO [219]		QPSO [219]		WQPSO [219]		QDDS			
		Iter	Mean ± St. Dev.	Iter	Mean ± St. Dev.	Iter	Mean ± St. Dev.	Iter	Mean ± St. Dev	Best	Worst
20	10	1000	94.1276 ±194.3648	1000	51.9761 ±0.4737	1000	35.8436 ±0.2843	250	**8.8912 ±0.0529**	8.8224	8.9700
	20	1500	204.337 ±293.4544	1500	136.8782 ±0.6417	1500	62.7696 ±0.4860	375	**19.0593 ±0.0991**	18.9071	19.2355
	30	2000	313.734 ±547.2635	2000	157.4707 ±0.8287	2000	70.9525 ±0.4283	500	**29.4457 ±0.0883**	29.2827	29.5926
40	10	1000	71.0239 ±174.1108	1000	17.3177 ±0.1515	1000	16.9583 ±0.1336	250	**8.9030 ±0.0875**	8.7483	9.0642
	20	1500	179.291 ±377.4305	1500	54.0411 ±0.4210	1500	54.2439 ±0.3752	375	**19.1421 ±0.0745**	18.9883	19.2278
	30	2000	289.593 ±478.6273	2000	81.1382 **±0.0319**	2000	57.0883 ±0.3437	500	**29.3855** ±0.1321	29.1769	29.5359
80	10	1000	37.3747 ±57.4734	1000	**7.5755** ±0.2708	1000	10.1650 ±0.2345	250	8.9213 **±0.0670**	8.8569	9.0454
	20	1500	83.6931 ±137.2637	1500	32.9970 ±0.2068	1500	47.0275 ±0.3507	375	**19.0663 ±0.1339**	18.8227	19.2097
	30	2000	202.672 ±289.9728	2000	53.6422 ±0.2616	2000	51.8299 ±0.3103	500	**29.4015 ±0.1079**	29.2291	29.5685

Table 4.3 Results for the Rastrigrin Function using 10 Independent Trials of QDDS

P	Dim	PSO [219]		QPSO [219]		WQPSO [219]		QDDS			
		Iter	Mean ± St. Dev.	Iter	Mean ± St. Dev.	Iter	Mean ± St. Dev.	Iter	Mean ± St. Dev	Best	Worst
20	10	1000	5.5382 ±3.0477	1000	4.8274 **±0.0015**	1000	4.0567 ±0.0094	250	**0.1214** ±0.0592	0.0425	0.2394
	20	1500	23.1544 ±10.4739	1500	16.0519 ±0.0414	1500	12.1102 **±0.0287**	375	**0.5776** ±0.0967	0.4765	0.7564
	30	2000	47.4168 ±17.1595	2000	33.7218 **±0.0114**	2000	23.5593 ±0.0713	500	**1.1709** ±0.2528	0.7499	1.5330
40	10	1000	3.5778 ±2.1384	1000	3.1794 **±6.033e-04**	1000	2.8163 ±0.0083	250	**0.1163** ±0.0577	0.0459	0.2628
	20	1500	16.4337 ±5.4811	1500	10.8824 **±0.0496**	1500	9.7992 ±0.0628	375	**0.5539** ±0.1300	0.3117	0.6847
	30	2000	37.2796 ±14.2838	2000	21.4530 ±0.0949	2000	17.4436 **±0.0034**	500	**1.1440** ±0.2768	0.6379	1.4135
80	10	1000	2.5646 ±1.5728	1000	2.2962 ±0.0130	1000	1.8857 **±0.0118**	250	**0.1397** ±0.0486	0.0543	0.2145
	20	1500	13.3826 ±8.5137	1500	7.8544 **±0.0011**	1500	7.2855 ±0.0032	375	**0.5312** ±0.1331	0.3059	0.7633
	30	2000	28.6293 ±10.3431	2000	15.9474 **±0.0198**	2000	15.0255 ±0.0294	500	**1.3041** ±0.2578	0.9302	1.7447

Table 4.4 Results for the Sphere Function using 10 Independent Trials of QDDS

P	Dim	PSO [219]		QPSO [219]		WQPSO [219]		QDDS			
		Iter	Mean ± St. Dev.	Iter	Mean ± St. Dev.	Iter	Mean ± St. Dev.	Iter	Mean ± St. Dev	Best	Worst
20	10	1000	3.16e-20 ±6.23e-20	1000	1.3909e-41 ±1.4049e-43	1000	**2.2922e-056 ±1.5365e-58**	250	6.2437e-04 ±3.0752e-04	3.0163e-04	0.0014
	20	1500	5.29e-11 ±1.56e-10	1500	3.5103e-22 ±3.5452e-24	1500	**2.9451e-40 ±2.8717e-42**	375	0.0027 ±6.9011e-04	0.0019	0.0039
	30	2000	2.45e-06 ±7.72e-06	2000	5.3183e-14 ±5.3623e-16	2000	**3.9664e-33 ±3.8435e-35**	500	0.0067 ±0.0013	0.0046	0.0085
40	10	1000	3.12e-23 ±8.01e-23	1000	2.5875e-71 ±2.6137e-73	1000	**5.5806e-80 ±5.6370e-82**	250	7.3478e-04 ±2.3036e-04	4.9032e-04	0.0012
	20	1500	4.16e-14 ±9.73e-14	1500	3.7125e-42 ±3.7500e-44	1500	**8.8186e-055 ±7.1785e-57**	375	0.0028 ±8.3546e-04	0.0016	0.0041
	30	2000	2.26e-10 ±5.10e-10	2000	4.2369e-30 ±1.7009e-33	2000	**5.4389e-44 ±2.4132e-45**	500	0.0060 ±8.4108e-04	0.0048	0.0072
80	10	1000	6.15e-28 ±2.63e-27	1000	8.5047e-102 ±7.5974e-104	1000	**4.7144e-106 ±4.7620e-108**	250	5.1027e-04 ±1.4758e-04	3.1457e-04	7.5794e-04
	20	1500	2.68e-17 ±5.24e-17	1500	1.1542e-68 ±1.1585e-70	1500	**2.5982e-74 ±2.6243e-76**	375	0.0026 ±6.0137e-04	0.0020	0.0040
	30	2000	2.47e-12 ±7.16e-12	2000	2.2866e-49 ±2.3070e-51	2000	**2.3070e-51 ±1.9125e-62**	500	0.0055 ±8.8175e-04	0.0043	0.0069

Table 4.5 Results for the Griewank Function using 10 Independent Trials of QDDS

P	Dim	PSO [219]		QPSO [219]		WQPSO [219]		QDDS			
		Iter	Mean ± St. Dev.	Iter	Mean ± St. Dev.	Iter	Mean ± St. Dev.	Iter	Mean ± St. Dev	Best	Worst
20	10	1000	0.09217 ±0.0833	1000	5.5093e-04 ±0.0657	1000	5.6353e-04 ±5.5093e-04	250	**8.0851e-05 ±3.3375e-05**	2.3539e-05	1.2727e-04
	20	1500	0.03002 ±0.03255	1500	**1.0402e-04** ±0.0211	1500	2.1318e-04 ±1.0402e-04	375	1.9821e-04 **±6.2856e-05**	1.2262e-04	2.9547e-04
	30	2000	0.01811 ±0.02477	2000	**1.2425e-04** ±0.0110	2000	2.1286e-04 ±1.2425e-04	500	2.5607e-04 **±6.4991e-05**	1.6388e-04	3.5948e-04
40	10	1000	0.08496 ±0.0726	1000	1.6026e-04 ±0.0496	1000	0.0020 ±1.6026e-04	250	**6.9932e-05 ±3.3276e-05**	3.2751e-05	1.3565e-04
	20	1500	0.02719 ±0.02517	1500	1.7127e-04 ±0.0167	1500	**1.6861e-04** ±1.7127e-04	375	1.9180e-04 **±4.7197e-05**	1.1890e-04	2.7165e-04
	30	2000	0.01267 ±0.01479	2000	3.9088e-05 ±0.0085	2000	**3.6762e-05 ±3.9088e-05**	500	2.3775e-04 ±5.3165e-05	1.5328e-04	3.1541e-04
80	10	1000	0.07484 ±0.07107	1000	3.3744e-04 ±0.0327	1000	1.5281e-04 ±3.3744e-04	250	**7.4205e-05 ±3.2774e-05**	2.9087e-05	1.4314e-04
	20	1500	0.02854 ±0.0268	1500	4.1701e-04 ±0.0168	1500	3.2549e-04 ±4.1701e-04	375	**1.8714e-04 ±5.6483e-05**	1.1359e-04	2.5826e-04
	30	2000	0.01258 ±0.01396	2000	**1.3793e-05** ±0.0106	2000	4.2231e-05 **±1.3793e-05**	500	2.8736e-04 ±4.6883e-05	2.0340e-04	3.6768e-04

We choose the constant k to be 5 and λ to be the product of a random number drawn from a normal distribution with $\mu = 0$ and $\sigma = 0.5$ and a factor of the order of 10^{-3}. ρ is a random number drawn between 0 to 1. The learning rate θ decreases linearly with iterations from 1 to 0.3. All experiments are carried out on two Intel(R) Core (TM) i7-5500U CPU @ 2.40GHz with 8GB RAM and one Intel(R) Core(TM) i7-2600U CPU @ 3.40GHz with 16GB RAM using MATLAB R2017a. 10 trials are carried out and results reported without use of GPUs. Tables 4.2 through 4.5 report performance of the QDDS algorithm on the Rosenbrock, Rastrigrin, Sphere and Griewank functions as well as compare and contrast with performances of PSO, QPSO and Weighted Mean Best QPSO (WQPSO) on the same benchmarks as reported by Xi et al. in [219]. All mean values and standard deviations listed in Tables 4.2 through 4.5 for the algorithms PSO, QPSO and WQPSO have been obtained from the work of Xi et al [219]. These have been used for a comparison of the performance of our algorithm (QDDS) on one-fourth the number of iterations with all other conditions for population and dimension remaining the same.

P and *Dim* represent the number of particles and the dimensionality of the functions. Figs. 4.1 through 4.12 plot the convergence profiling of the above experiments on the stated benchmarks of orders 10, 20 and 30 over 10 independent trials. For purposes of brevity, results of simulations using only population size 20 are reported in this section.

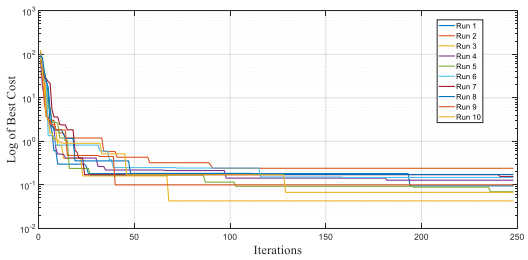


Figure 4.1 Conv. Profiling of Rastrigrin using QDDS (dimension=10, population=20)

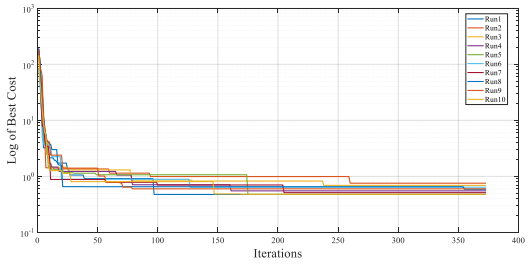


Figure 4.2 Conv. Profiling of Rastrigrin using QDDS (dimension=20, population=20)

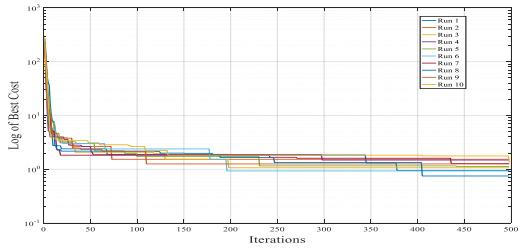


Figure 4.3 Conv. Profiling of Rastrigrin using QDDS (dimension=30, population=20)

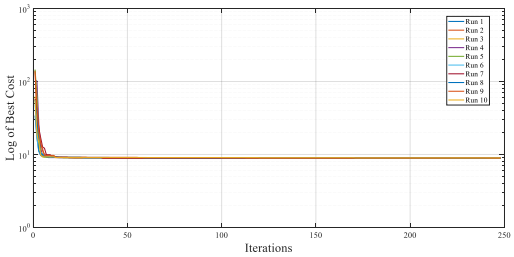


Figure 4.4 Conv. Profiling of Rosenbrock using QDDS (dimension=10, population=20)

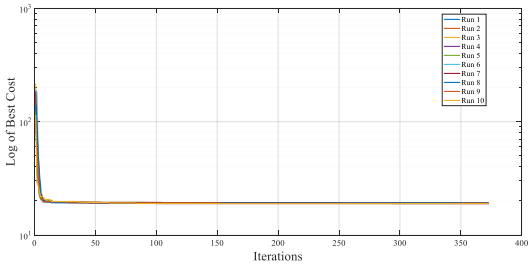


Figure 4.5 Conv. Profiling of Rosenbrock using QDDS (dimension=20, population=20)

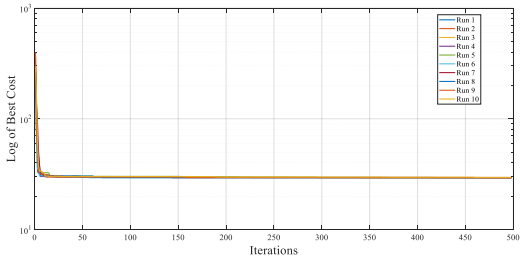


Figure 4.6 Conv. Profiling of Rosenbrock using QDDS (dimension=30, population=20)

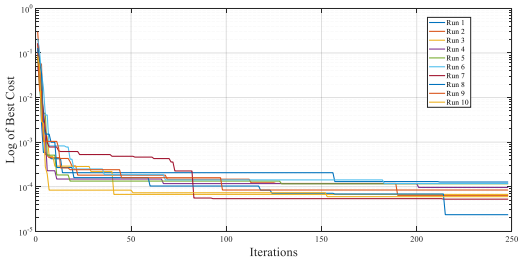


Figure 4.7 Conv. Profiling of Griewank using QDDS (dimension=10, population=20)

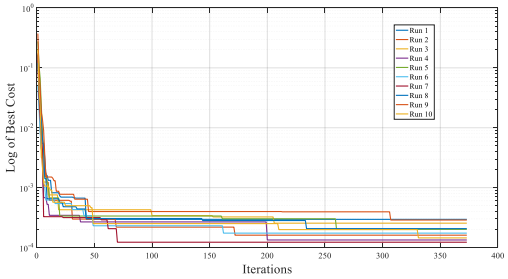


Figure 4.8 Conv. Profiling of Griewank using QDDS (dimension=20, population=20)

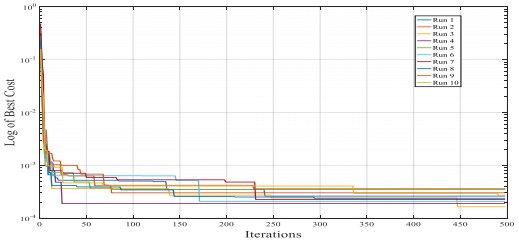


Figure 4.9 Conv. Profiling of Griewank using QDDS (dimension=30, population=20)

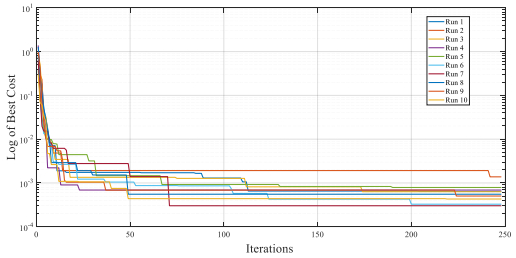


Figure 4.10 Conv. Profiling of Sphere using QDDS (dimension=10, population=20)

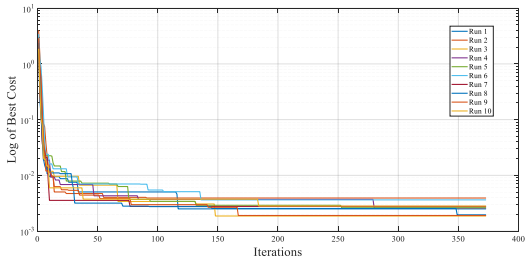


Figure 4.11 Conv. Profiling of Sphere using QDDS (dimension=20, population=20)

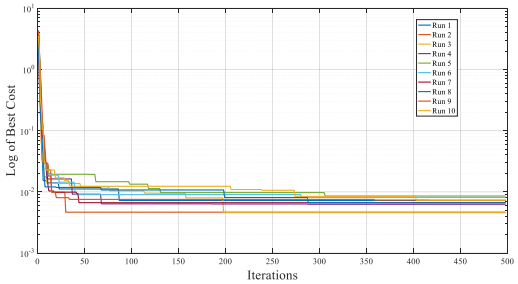


Figure 4.12 Conv. Profiling of Sphere using QDDS (dimension=30, population=20)

4.5 Analysis of Experimental Results

From Table 4.2 it is observed that the QDDS algorithm performs significantly better on the Rosenbrock function of dimensionality 10, 20 and 30 as compared to PSO, QPSO and WQPSO. In Table 4.3 the QDDS algorithm generates mean values which are at least 11.521 times smaller than WQPSO or even smaller in case of QPSO and PSO on the Rastrigrin function of dimensionality 10, 20 and 30. However, the standard deviation values obtained using WQPSO and QPSO are clearly superior to those found using QDDS. The results from

Table 4.4 using the Sphere function indicate the sub-par performance of QDDS with respect to the competitor algorithms. Table 4.5 reports somewhat comparable results in terms of mean cost using WQPSO, QPSO and QDDS, while it is to be noted that QDDS has a standard deviation at least ~160 times smaller than that of QPSO. The convergence profiles in Figures 4.1 through 4.12 point out that QDDS is fairly consistent in its ability to converge to local optima of acceptable quality. It is obvious that the solutions to the problems discussed in the paper are in fact local optima, however the solution qualities corresponding to some of these local optima obtained using QDDS are evidently superior to some related reports in the literature [219][221][226][227]. One way to improve the performance of QDDS may be to not use gradient descent but a problem-independent optima-seeking mechanism in the δ update step of the algorithm.

CHAPTER 5

CHAOTIC QDDS

5.1 Background

The seminal work of Eberhart and Kennedy on flocking induced stochastic, multiparticle swarming resulted in a surge in nature-inspired optimization research, specifically after their highly influential paper Particle Swarm Optimization [1] (PSO) at the International Conference on Neural Networks in Perth, Australia in 1995. This was a landmark moment in the history of swarm intelligence and the following years saw a surge of interest towards the application of nature-inspired methods in approximating engineering problems that were till then either not tractable or simply hard from a computational standpoint. With a steady increase in processor speed and distributed computing abilities over the last couple of decades, gradient-independent approaches have gradually become ever so common. The simple and intuitive equations of motion in PSO are powerful due to simplicity and low computational cost. In this section, a formal transition from the classical model of the canonical PSO to that of quantum-inspired PSO, or the Quantum-behaved PSO (QPSO) is explored. The QPSO model assumes quantum properties in agents and establishes an uncertainty-based position distribution instead of a deterministic one as in the canonical PSO with Newtonian walks. Importantly enough, the QPSO algorithm requires the practitioner to tune only one parameter—the Contraction–Expansion (CE) coefficient—instead of three in PSO. It is worth looking at the dynamics of a PSO-driven swarm to gain a better understanding of singular and double Dirac delta driven quantum swarms, later on

5.2 Revisiting the Classical PSO

Assume $\boldsymbol{x_{i=1..m}} = [x_1 x_2 x_3 \dots x_m]$ is the cohort of *m* particles of dimensionality *n* and $\boldsymbol{v_{i=1..m}} = [v_1 v_2 v_3 \dots v_m]$ are the velocity vectors which denote incremental changes in their positions in the solution hyperspace. Given this knowledge, a canonical PSO-like formulation may be expressed as:

$$v_{ij}(t+1) = w \times v_{ij}(t) + C_1 \times r_1(t) \times \Big(P_{ij}(t) - x_{ij}(t)\Big) + C_2 \times r_2(t) \times \Big(P_{gj}(t) - x_{ij}(t)\Big) \quad (5.1)$$

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1) \quad (5.2)$$

The parameters w, C_1, C_2, r_1, r_2 are responsible for imparting inertia, cognitive, and social weights as well as random perturbations towards the historical best position $P_{ij}(t)$ of any particle (pbest) or $P_{gj}(t)$, that of the swarm as a whole (gbest). The canonical PSO model mimics social information exchange in flocks of birds and schools of fish and is a simple, yet powerful, optimization paradigm. However, it has its limitations: Van den Bergh showed that the algorithm is not guaranteed to converge to globally optimum solutions based on the convergence criteria put forward by Solis and Wet [233]. Clerc and Kennedy demonstrated that the algorithm may converge if particles cluster about a local attractor p lying at the diagonal end of the hyper-rectangle constructed using its cognitive and social velocity vectors [235] (terms 2 and 3 in the right-hand side of equation 5.1, respectively). Proper tuning of the algorithmic parameters and limits on the velocity are usually required to bring about convergent behavior. The interested reader may look at [236][237][238][239] for detailed operating conditions, possible applications, and troubleshooting of issues when working with the PSO algorithm.

5.3 The Quantum-behaved PSO

The local attractor p, introduced by Clerc and Kennedy [235] as the point around which particles should flock in order to bring about swarm-wide convergence can be formally expressed using equation 5.3 and further simplifications lead to a parameter reduced form in equation 5.4. This result is possible of course, after the assumption that c_1 and c_2 may take on any values between 0 and 1.

$$p_{ij}(t+1) = \frac{c_1 P_{ij}(t) + c_2 P_{gj}(t)}{c_1 + c_2} \tag{5.3}$$

$$p_{ij}(t+1) = \varphi P_{ij}(t) + (1-\varphi) P_{gj}(t)\,, \varphi \sim U\,(0,1) \tag{5.4}$$

Drawing insights from this analysis, Sun et al. in [230][231] outlined algorithmic working of Quantum-behaved Particle Swarm Optimization (QPSO). Instead of point representations of a particle, wave functions were used to provide quantitative sense about its state. The normalized probability density function **F** of a particle may be put forward as:

$$\mathbf{F}(X_{ij}(t+1)) = \frac{1}{L_{ij}(t)} exp^{\left(-2\left|p_{ij}(t) - X_{ij}(t+1)\right| / L_{ij}(t)\right)} \tag{5.5}$$

L is the standard deviation of the distribution: it provides a measure of the dynamic range of the search space of a particle in a specific timestep. Using Monte Carlo method, equation (5.5) may be transformed into a recursive, computable closed form expression of particle positions in equation (5.6) below:

$$X_{ij}(t+1) = p_{ij}(t) \pm \frac{L_{ij}(t)}{2} \ln\left(\frac{1}{u}\right) \;\;, u \sim U\,(0,1) \tag{5.6}$$

L is computed as a measure of deviation from the average of all individual personal best particle positions (pbest) in each dimension, i.e., the farther from the average a particle is in a dimension the larger the value of *L* is for that dimension. This average position has been

dubbed the name ‘Mean Best’ or ‘mbest’ and is an agglomerative representation of the swarm as if each member were in its personal best position visited in course of history.

$$mbest(t) = \left[mbest_1(t) mbest_2(t) mbest_3(t) \dots mbest_j(t)\right]$$
$$= \left[\frac{1}{m}\sum_{i=1}^{m} p_{i1}(t) \frac{1}{m}\sum_{i=1}^{m} p_{i2}(t) \frac{1}{m}\sum_{i=1}^{m} p_{i3}(t) \dots \frac{1}{m}\sum_{i=1}^{m} p_{ij}(t)\right] \quad (5.7)$$

Therefore, L may be expressed by including the deviation from $mbest$ by equation (5.8) below. The modulation factor β is known as the Contraction–Expansion (CE) Factor and may be adjusted to control the convergence speed of the QPSO algorithm depending on the application.

$$L_{ij}(t) = 2\beta\left|mbest_j(t) - X_{ij}(t)\right| \quad (5.8)$$

Subsequently plugging the value of L obtained in equation (5.8) into equation (5.6), the position update formulation for QPSO may be re-expressed as the following

$$X_{ij}(t+1) = p_{ij}(t) \pm \beta\left|mbest_j(t) - X_{ij}(t)\right| \ln\left(\frac{1}{u}\right) \quad , u \sim U(0,1) \quad (5.9)$$

Issues such as suboptimal convergence during the application of the QPSO algorithm may arise out of an unbiased selection of weights in the mean best computation as well as the overdependence on the globally best particle in the design of the local attractor p. These issues have also been studied by Xi et al. [240], Sengupta et al. [225], and Dhabal et al. [224]. Xi et al. proposed a differentially weighted mean best [232]: a variant of the QPSO algorithm with a weighted mean best position (WQPSO), which seeks to alleviate the subpar selection of weights in the *mean best* update process. The underlying assumption is that fitter particles stand to contribute more to the *mean best* position and that these particles should be accorded larger weights, drawing an analogy with the correlation between cultural uptick and the contributions of the societal, intellectually elite to it [232]. Xi et al. also put forward [240] a

local search strategy using a 'super particle' with variable contributions from swarm members to overcome the dependence issues during the local attractor design. However, to date no significant study has been undertaken to investigate the effect of more than one spatially co-located basin of attraction around the local attractor, particularly that of multi-well systems. In the next section we seek to derive state expressions of a particle convergent upon one well under the influence of two spatially co-located Dirac delta wells.

5.4 Swarming Under the Influence of Two Delta Potential Wells

The time-independent Schrodinger's wave equation governs the different interpretations of particle behavior:

$$[-\frac{\hbar^2}{2m}\nabla^2 + V(r)]\psi(r) = E\,\psi(r) \tag{5.10}$$

ψ(r), *V(r)*, *m*, *E*, and ħ represent the wave function, the potential function, the reduced mass, the energy of the particle, and reduced Planck's constant, respectively. However, the wave function $\psi(r)$ has no physical significance on its own: its amplitude squared is a measure of the probability of finding a particle. Let us consider a particle under the influence of two delta potential wells experiencing an attractive potential V (Fig. 5.1):

$$V(r) = -\mu\{\delta(r+a) + \delta(r-a)\} \tag{5.11}$$

The centers of the two wells are at $-a$ and a and μ is a constant indicative of the depth of the wells. Under the assumption that the particle experiences no attractive potential, i.e., $V = 0$ in regions far away from the centers, the even solution of the time-independent Schrodinger's equation in equation (5.10) takes the following form.

$$-\frac{\hbar^2}{2m}\frac{d^2}{dr^2}\psi(r) = E\psi(r) \tag{5.12}$$

The even solutions to ψ for $E < 0$ (bound states) in regions $\mathbb{R}1$: $r \in (-\infty, a)$, $\mathbb{R}2$: $r \in (-a, a)$ *and* $\mathbb{R}3$: $r \in (a, \infty)$, taking k to be equal to $(\sqrt{2mE}/\hbar)$ can be expressed as has been proved in Griffiths [223]:

$$\psi_{even}(r) = \begin{cases} \eta_1 exp(-kr) & r > a \\ \eta_2 exp(-kr) + \eta_3 exp(kr) & 0 < r < a \\ \eta_2 exp(kr) + \eta_3 exp(-kr) & -a < r < 0 \\ \eta_1 exp(kr) & r < -a \end{cases} \tag{5.13}$$

The constants η_1 and η_2 described in the above equation are obtained by (a) solving for the continuity of the wave function ψ_{even} at $r = a$ and $r = -a$ and (b) solving for the continuity of the derivative of the wave function at $r = 0$. Thus, ψ_{even} may be rewritten below as has been in Griffiths [223].

$$\psi_{even}(r) = \begin{cases} \eta_2\{1 + \exp(2ka)\}\exp(-kr) & r > a \\ \eta_2\{\exp(-kr) + \exp(kr)\} & -a < r < a \\ \eta_2\{1 + \exp(2ka)\}\exp(kr) & r < -a \end{cases} \tag{5.14}$$

The odd wave function ψ_{odd} does not guarantee that a solution would be found [223]. Additionally, the bound state energy in double well setup is lower than that in a single well setup by approximately a factor of $(1.11)^2 \approx 1.2321$ [222]:

$$E_{bs,Double\ Well} = -(1.11)^2 E_{bs,Single\ Well} \tag{5.15}$$

To study the motional aspect of a particle its probability density function given by the squared magnitude of ψ_{even} is formally expressed. Further, the claim that there is greater than 50% probability of a particle existing in neighborhood of the center of any of the potential wells (assumed centered at 0) boils down to the following criterion being met [230].

$$\int_{-|r|}^{|r|} \psi_{even}(r)^2 \, dr > 0.5 \tag{5.16}$$

$-|r|$ and $|r|$ are the dynamic limits of the neighborhood. Doing away with the inequality, equation (5.16) is rewritten as:

$$\int_{-|r|}^{|r|} \psi_{even}(r)^2 \, dr = 0.5\lambda \qquad (1 < \lambda < 2) \tag{5.17}$$

Equation (5.17) is the criterion for localization around the center of a potential well in a double Dirac delta well.

5.5 The Chaotic QDDS Algorithm

To ease computations, we make the assumption that one of the two potential wells is centered at 0. Then, solving for conditions of localization of the particle in the neighborhood around the center of that well and computing $\int_{-|r|}^{|r|}(\psi(r)^2 \, dr$ for regions $\mathbb{R}2_{0-}$: $r' \in (-r, 0)$ and $\mathbb{R}2_{0+}$: $r' \in (0, r)$, we obtain the relationship below.

$$\eta_2{}^2 = \frac{k\lambda}{\exp(2kr) - 5\exp(-2kr) + 4kr + 4} \tag{5.18}$$

Replacing denominator of the Right Hand Side (R.H.S.) of equation (5.18) i.e. $(\exp(2kr) - 5\exp(-2kr) + 4kr + 4)$ as δ, we rewrite it as

$$\delta = \exp(2kr) - 5\exp(-2kr) + 4kr + 4 \tag{5.19}$$

Equating $\eta_2{}^2$ in the Left Hand Side (L.H.S.) of equation (5.18) for any two consecutive iterations (assuming it is a constant over iterations as it not a function of time) we get equations (5.20), (5.21), and (5.22):

$$\frac{\lambda_t}{\exp(2kr_t) - 5\exp(-2kr_t) + 4kr_t + 4} = \frac{\lambda_{t-1}}{\exp(2kr_{t-1}) - 5\exp(-2kr_{t-1}) + 4kr_{t-1} + 4} \tag{5.20}$$

$$\Rightarrow \frac{\lambda_t}{\delta_t} = \frac{\lambda_{t-1}}{\delta_{t-1}} \tag{5.21}$$

$$\Rightarrow \delta_t = \Lambda \, . \, \delta_{t-1} \qquad (0.5 < \Lambda < 2) \tag{5.22}$$

Λ is the ratio $(\lambda_t/\lambda_{t-1})$ and it may vary between 0.5 to 2 since (1< λ <2). To keep a particle constrained within the vicinity of the center of the potential well, it must meet the following condition.

$$\frac{1}{2}\delta_{t-1} < \delta_t < 2\,\delta_{t-1} \tag{5.23}$$

Thus, we find δ_t for any iteration by utilizing δ_{t-1} , obtained in the immediately past iteration. This is done by accounting for a correction factor in the form of the gradient of δ_{t-1}, multiplied by a learning rate α. The computation of δ_t from δ_{t-1} feeds off the relationship of δ_{t-1} with δ_{t-2} while taking the sign of the gradient of δ_{t-1} into consideration. The procedural details are outlined in Algorithm 5.1. The learning rate α is chosen as a linearly decreasing, time-varying one (LTV) to help facilitate exploration of the solution space early on in the optimization phase and a gradual shift to exploitation as the process evolves. ν is a small fraction between 0 and 1 chosen at will. However, one empirically successful value is 0.3 and we use it in our computations.

$$\alpha = (1-\nu)\left(\frac{maximum\ number\ of\ \ iterations - current\ iteration}{maximum\ number\ of\ \ iterations}\right) + \nu \tag{5.24}$$

Upon computing a value for δ_t, equation (5.19) is solved to retrieve an estimate of r_t, which denotes a candidate position as well as a potential solution at the end of that iteration.

$$r_t \cong Solve[\{\,\delta - (\exp(2kr) - \ 5\exp(-2kr) + 4kr + 4)\} = 0] \tag{5.25}$$

We let r_t, i.e., a particle's position in the current iteration, maintain a component towards the best position found so far (gbest) in addition to its current solution obtained from equation (5.19). Let ρ denote the component towards the gbest position and (1- ρ) be that towards the current solution.

$$r_t^{new} = \rho r_t + (1-\rho) r_{gbest} \tag{5.26}$$

A cost function is subsequently computed and the corresponding particle position is saved if the cost is lowest among all the historical swarm-wide best costs obtained. This process is repeated until the convergence criteria of choice (solution accuracy threshold, computational expense, memory requirements, success rate, etc.) are met.

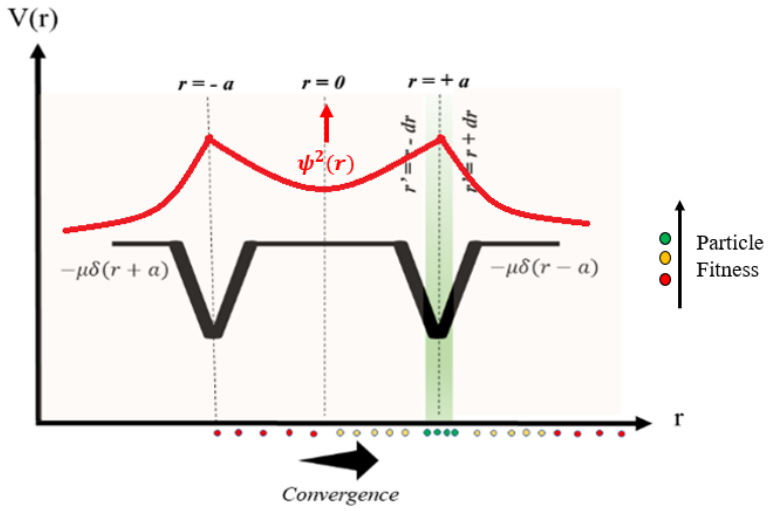


Figure 5.1 The Double Potential Well Setup

5.5.1 Chaotic QDDS on Chebyshev Maps

In this section, we use a Chebyshev chaotic map to generate coefficient sequences for driving the belief ρ in the solution update phase of the QDDS algorithm.

5.5.1.1 Chebyshev Map Driven Solution Update: Motivation

Chaotic metaheuristics necessitate control over the balance between diversification and intensification phases. The diversification phase is carried out by choosing an appropriate chaotic system which performs the extensive search, while the intensification phase is carried out by performing a local search such as gradient descent. It is important that during the initial progression of the search, multiple orbits pass through the vicinity of the local extrema. A large perturbation weight ensures that the strange attractor

of one local extremum intersects the strange attractor of any of the other local extrema [258]. To this end, we generate a sorted sequence which acts as a perturbation source of tapering magnitude using the Chebyshev chaotic map using the recursive relation in Equation (5.27) [245]. There is a relative dearth of studies looking at chaotic perturbations to agent positions to drive them towards socially optimal agent locations. In our approach, we look to facilitate extensive communication among agents by employing larger chaotic weights (diversification phase) in the initial stages and local communication among agents by tapering weights (intensification phase) with the progression of function evaluations. The optimal choice and arrangement of the modulus and sign of the weights generated using the pseudo random number generator or any other method for that matter is subject to change with a change in the application problem and is very much an open question in an exploration–exploitation based search niche. However, the two properties of ergodicity and non-repetition in chaotic time sequences have proved useful in a number of related classical studies [259][261][243] and are key factors supporting the choice of the perturbation weights in this work. Furthermore, the properties of large Lyapunov coefficient (a measure of chaoticity) and space-filling nature of the Chebyshev sequence serve to help avoid stagnation in local extrema and supplement the choice of the type of chaotic map in the studies in this article.

$$\rho_t^{Chebyshev} = \cos\left(t * cos^{-1}\left(\rho_{t-1}^{Chebyshev}\right)\right) \tag{5.27}$$

Equation (5.26) subsequently becomes

$$r_t^{new} = \rho_t^{Chebyshev} * r_{iter} + \left(1 - \rho_t^{Chebyshev}\right) * r_{gbest} \tag{5.28}$$

Fig. 5.2 shows generated weights from a Chebyshev map over 1000 iterations while Fig. 5.3 shows the histogram of the weights. Fig. 5.4 shows a schematic of the C-QDDS workflow.

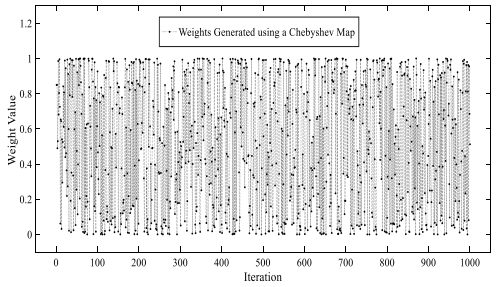


Figure 5.2 Weights ($\rho^{Chebyshev}$) from a Chebyshev chaotic map over 1000 iterations

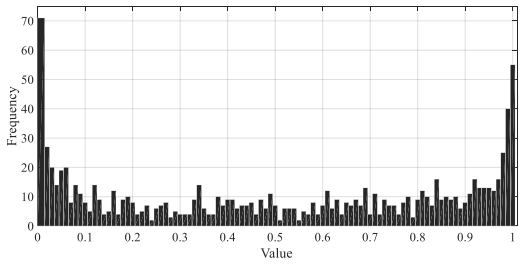


Figure 5.3 Histogram of Weights from the Chebyshev map over 1000 iterations

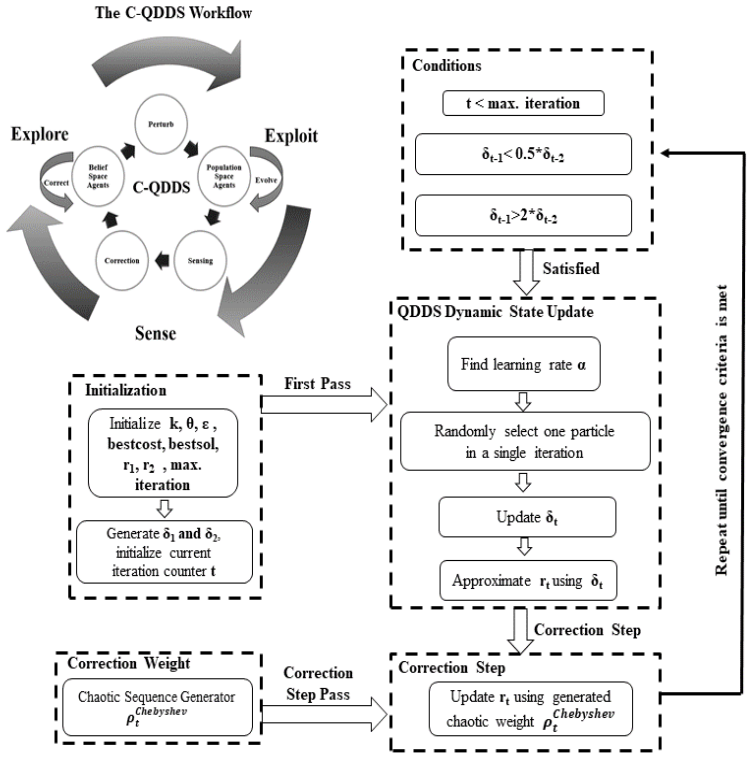


Figure 5.4. Schematic of Chaotic Quantum Double Delta Swarm (C-QDDS) workflow

Table 5.1 General terms used in context of the algorithms and experimentation.

Term	Discussion
Some General Terms	
Population (X)	The collection or 'swarm' of agents employed in the search space
Fitness Function (f)	A measure of convergence efficiency
Current Iteration	The ongoing iteration among a batch of dependent/independent runs
Maximum Iteration Count	The maximum number of times runs are to be performed
Particle Swarm Optimization (PSO)	
Position (X)	Position value of individual swarm member in multidimensional space
Velocity (v)	Velocity values of individual swarm members
Cognitive Accl. Coefficient (C1)	Empirically found scale factor of pBest attractor
Social Accl. Coefficient (C2)	Empirically found scale factor of gBest attractor
Personal Best (pBest)	Position corresponding to historically best fitness for a swarm member
Global Best (gBest)	Position corresponding to best fitness over history for swarm members
Inertia Weight Coefficient (ω)	Facilitates and modulates exploration in the search space
Cognitive Random Perturbation (r_1)	Random noise injector in the Personal Best attractor
Social Random Perturbation (r_2)	Random noise injector in the Global Best attractor
Quantum-behaved PSO	
Local Attractor	Set of local attractors in all dimensions
Characteristic Length	Measure of scales on which significant variations occur
Contraction–Expansion Parameter (β)	Scale factor influencing the convergence speed of QPSO
Mean Best	Mean of personal bests across all particles, akin to leader election in species
Quantum Double-Delta Swarm Optimization (QDDS)	
ρ	Component towards the global best position gbest
$\psi(r)$	Wave function in the Schrodinger's equation
$\psi_{even}(r)$	Even solutions to Schrodinger's Equation for Double Delta Potential Well
$V(r)$	Potential Function
Λ	Limiter
δ_{iter}	Characteristic Constraint
ϵ	A small fraction between 0 and 1 chosen at will
$\mathbb{R}1$: $r \in (-\infty, a)$	Region 1
$\mathbb{R}2$: $r \in (-a, a)$	Region 2
$\mathbb{R}3$: $r \in (a, \infty)$	Region 3
α	Learning Rate
$\rho_{iter}{}^{Chebyshev}$	Component towards global best gbest drawn from Chebyshev map
μ	Depth of the wells
a	Coordinate of wells

5.5.1.2 Pseudocode of C-QDDS

In this section, the pseudocode of the C-QDDS algorithm is presented.

Algorithm 5.1. Chaotic Quantum Double Delta Swarm Algorithm

Initialization Phase

1: *Initialize* **k**
2: *Initialize scale factor* $\boldsymbol{\theta}$ *randomly* ($\approx 10^{-3}$)
3: *Initialize a constant* $\boldsymbol{\varepsilon}$ *between 0 and 1 as the lower bound of* $\boldsymbol{\chi}$
4: *Initialize maximum number of iterations as* **max. iterations**
5: *Initialize the global best cost as* **bestcost** *and global best position as* **bestsol**
6: *for each particle*
7: | *for each dimension*
8: | | *Initialize positions* $\mathbf{r_1}$ *and* $\mathbf{r_2}$ *for iterations 1 and 2*
9: | *end for*
10: *end for*
11: *Generate* $\boldsymbol{\delta_1}$ *and* $\boldsymbol{\delta_2}$ *from* $\mathbf{r_1}$ *and* $\mathbf{r_2}$ *using equation (5.19)*
12: *Set current iteration* $\boldsymbol{t}$ *=3*

Optimization Phase

13: ***while*** (**t** < **max. iterations**) *and* $\{(\boldsymbol{\delta_{t-1}} < \mathbf{0.5} * \boldsymbol{\delta_{t-2}})$ *or* $(\boldsymbol{\delta_{t-1}} > \mathbf{2} * \boldsymbol{\delta_{t-2}})\}$
14: | *Find learning rate* $\boldsymbol{\alpha}$ *using eq. (5.24)*
15: | *Select a particle randomly*
16: | ***for*** *each dimension*
17: | | ***if*** $(\boldsymbol{\delta_{t-1}} > \mathbf{2} * \boldsymbol{\delta_{t-2}})$ *and* $\boldsymbol{\nabla\delta_{t-1}}$>0
18: | | | $\boldsymbol{\delta_t} = \boldsymbol{\delta_{t-1}} - \boldsymbol{\theta} * \boldsymbol{\nabla\delta_{t-1}} * \boldsymbol{\alpha}$
19: | | ***else if*** $(\boldsymbol{\delta_{t-1}} > \mathbf{2} * \boldsymbol{\delta_{t-2}})$ *and* $\boldsymbol{\nabla\delta_{t-1}}$<0
20: | | | $\boldsymbol{\delta_t} = \boldsymbol{\delta_{t-1}} + \boldsymbol{\theta} * \boldsymbol{\nabla\delta_{t-1}} * \boldsymbol{\alpha}$
21: | | ***else if*** $(\boldsymbol{\delta_{t-1}} < \mathbf{0.5} * \boldsymbol{\delta_{t-2}})$ *and* $\boldsymbol{\nabla\delta_{t-1}}$<0
22: | | | $\boldsymbol{\delta_t} = \boldsymbol{\delta_{t-1}} - \boldsymbol{\theta} * \boldsymbol{\nabla\delta_{t-1}} * \boldsymbol{\alpha}$
23: | | ***else if*** $(\boldsymbol{\delta_{t-1}} < \mathbf{0.5} * \boldsymbol{\delta_{t-2}})$ *and* $\boldsymbol{\nabla\delta_{t-1}}$>0
24: | | | $\boldsymbol{\delta_t} = \boldsymbol{\delta_{t-1}} + \boldsymbol{\theta} * \boldsymbol{\nabla\delta_{t-1}} * \boldsymbol{\alpha}$
25: | | ***end if***
26: | ***end for***
27: | *Solve* $\mathbf{r_t}$ *from* $\boldsymbol{\delta_t}$

Chaotic Random Number Generation and Correction Phase

28: | *Generate* $\boldsymbol{\rho} \in [\mathbf{0}, \mathbf{1}]$ *using Chebyshev recurrence:* $\boldsymbol{\rho_t^{Chebyshev}} = \mathbf{cos}\left(\mathbf{t} * \mathbf{cos^{-1}}\left(\boldsymbol{\rho_{t-1}^{Chebyshev}}\right)\right)$
29: | $\mathbf{r_t^{(updated)}} = \left(\boldsymbol{\rho_t^{Chebyshev}}\right)\mathbf{r_t} + \left(\mathbf{1} - \left(\boldsymbol{\rho_t^{Chebyshev}}\right)\right)\mathbf{r_{gbest}}$
30: | *Compute cost using* $\mathbf{r_t^{(updated)}}$
31: | ***if*** $\mathbf{cost_t}$<**bestcost**
32: | | **bestcost** = $\mathbf{cost_t}$
33: | | **bestsol** = $\mathbf{r_t^{(updated)}}$
34: | ***end if***
35: | **t** = **t** +1
36: ***end while***

5.6 Experimental Setup

5.6.1 Benchmark Functions

A suite of the following 23 optimization benchmark functions (F1–F23) are popularly used to inspect the performance of evolutionary optimization paradigms and have been utilized in this work to characterize the behavior of C-QDDS across unimodal and multimodal function landscapes of fixed and varying dimensionality.

Table 5.2 Unimodal test functions considered for testing

Number	Name	Expression	Range	Min
F1	Sphere	$f(x) = \sum_{i=1}^{n} x_i^2$	[−100, 100]	$f(x^*) = 0$
F2	Schwefel's Problem 2.22	$f(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	[−10, 10]	$f(x^*) = 0$
F3	Schwefel's Problem 1.2	$f(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2$	[−100, 100]	$f(x^*) = 0$
F4	Schwefel's Problem 2.21	$f(x) = \max_i \{ \lvert x_i \rvert, 1 \le i \le n \}$	[−100, 100]	$f(x^*) = 0$
F5	Generalized Rosenbrock's Function	$f(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	[−n, n]	$f(x^*) = 0$
F6	Step Function	$f(x) = \sum_{i=1}^{n} (\lfloor x_i + 0.5 \rfloor)^2$	[−100, 100]	$f(x^*) = 0$
F7	Quartic Function i.e., Noise	$f(x) = \sum_{i=1}^{n} i x^4 + \text{random}[0,1)$	[−1.28, 1.28]	$f(x^*) = 0$

Note: x* Globally optimum argument.

Table 5.3 Multimodal test functions considered for testing

Number	Name	Expression	Range	Min
F8	Generalized Schwefel's Problem 2.26	$f(x) = -\sum_{i=1}^{n} (x_i \sin(\sqrt{\lvert x_i \rvert}))$	[−500, 500]	$f(x^*) = -12{,}569.5$
F9	Generalized Rastrigrin's Function	$f(x) = An + \sum_{i=1}^{n} [x_i^2 - A\cos(2\pi x_i)]$, A=10	[−5.12, 5.12]	$f(x^*) = 0$
F10	Ackley's Function	$f(x) = -20 \exp\left(-0.2 \sqrt{\frac{1}{d} \sum_{i=1}^{d} x_i^2} \right) - \exp\left(\sqrt{\frac{1}{d} \sum_{i=1}^{d} \cos(2\pi x_i)} \right) + 20 + \exp(1)$	[−32.768, 32.768]	$f(x^*) = 0$
F11	Generalized Griewank Function	$f(x) = 1 + \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}})$	[−600, 600]	$f(x^*) = 0$
F12	Generalized Penalized Function 1	$f(x) = \frac{\pi}{d} \left\{ 10\sin^2(\pi y_1) + \sum_{i=1}^{n-1} (y_i - 1)^2 [1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2 \right\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$	[−50, 50]	$f(x^*) = 0$
F13	Generalized Penalized Function 2	$f(x) = 0.1 \left\{ \sin^2(3\pi x_1) + \sum_{i=1}^{n-1} (x_i - 1)^2 [1 + \sin^2(3\pi x_{i+1})] + (x_n - 1)^2 [1 + \sin^2(2\pi x_{30})] \right\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$ where $u(x_i, 5, 100, 4) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & -a < x_i < a \\ k(-x_i - a)^m, & x_i < -a \end{cases}$ $y_i = 1 + \frac{1}{4}(x_i + 1)$	[−50, 50]	$f(x^*) = 0$

Note: x* Globally optimum argument.

Table 5.4 Multimodal test functions with fixed dimensions considered for testing

Number	Name	Expression	Range	Min
F14, n = 2	Shekel's Foxholes Function	$f(x) = \left[\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6}\right]^{-1}$ where $a_{ij} = \begin{pmatrix} -32 & -16 & 0 & 16 & 32 & -32 & \dots & 0 & 16 & 32 \\ -32 & -32 & -32 & -32 & -32 & -16 & \dots & 32 & 32 & 32 \end{pmatrix}$	$[-65.536, 65.536]$	$f(x^*) \approx 1$
F15, n = 4	Kowalik's Function	$f(x) = \left[\sum_{i=1}^{11} a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}\right]^2$ Coefficients are defined according to Table F15.	$[-5, 5]$	$f(x^*) \approx 0.0003075$
F16, n = 2	Six-Hump Camel-Back Function	$f(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	$[-5, 5]$	$f(x^*) = -1.0316285$
F17, n = 2	Branin Function	$f(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos x_1 + 10$	$-5 \le x_1 \le 10$, $0 \le x_2 \le 15$	$f(x^*) = 0.398$
F18, n = 2	Goldstein-Price Function	$f(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)][30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	$[-2, 2]$	$f(x^*) = 3$
F19, n = 3	Hartman's Family Function 1	$f(x) = -\sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2\right]$	$0 \le x_j \le 1$	$f(x^*) = -3.86$
F20, n = 6	Hartman's Family Function 2	$f(x) = -\sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2\right]$ Coefficients are defined according to Table F20.1 and F20.2 respectively.	$0 \le x_j \le 1$	$f(x^*) = -3.86$
F21, n = 4	Shekel's Family Function 1	$f(x) = -\sum_{i=1}^{5}[(x - a_i)(x - a_i)^T + c_i]^{-1}$ Coefficients are defined according to Table F21.	$0 \le x_j \le 10$	$f(x^*_{local}) = \frac{1}{c_i}$, $1 \le i \le m$
F22, n = 4	Shekel's Family Function 2	$f(x) = -\sum_{i=1}^{7}[(x - a_i)(x - a_i)^T + c_i]^{-1}$ Coefficients are defined according to Table F22.	$0 \le x_j \le 10$	$f(x^*_{local}) = \frac{1}{c_i}$, $1 \le i \le m$
F23, n = 4	Shekel's Family Function 3	$f(x) = -\sum_{i=1}^{10}[(x - a_i)(x - a_i)^T + c_i]^{-1}$ Coefficients are defined according to Table F23.	$0 \le x_j \le 10$	$f(x^*_{local}) = \frac{1}{c_i}$, $1 \le i \le m$

Note: x^* Globally optimum argument, x^*_{local} Locally optimum argument.

Table 5.5 Coefficients of Kowalik's Function (F15)

Index (i)	a_i	a_{ij}^{-1}
1	0.1957	0.25
2	0.1947	0.5
3	0.1735	1
4	0.1600	2
5	0.0844	4
6	0.0627	6
7	0.0456	8
8	0.0342	10
9	0.0323	12
10	0.0235	14
11	0.0246	16

Table 5.6 Coefficients of Hartman's Functions (F19)

Index (i)	$a_{ij}, j = 1, 2, 3$			c_i	$p_{ij}, j = 1, 2, 3$		
1	3	10	30	1	0.3689	0.1170	0.2673
2	0.1	10	35	1.2	0.4699	0.4387	0.7470
3	3	10	30	3	0.1091	0.8732	0.5547
4	0.1	10	35	3.2	0.038150	0.5743	0.8828

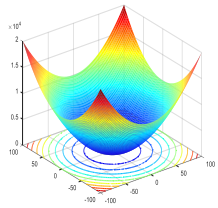
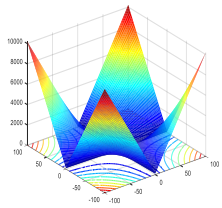
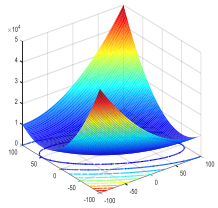
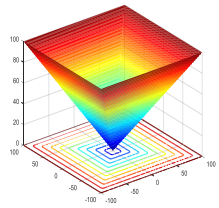
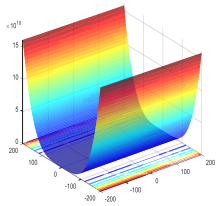
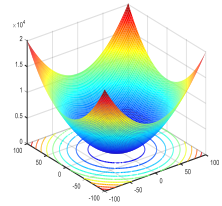
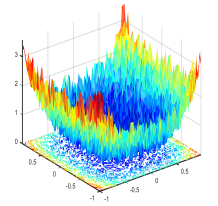
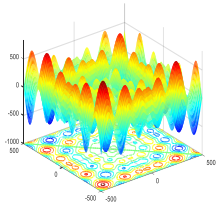
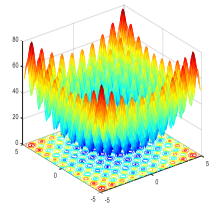
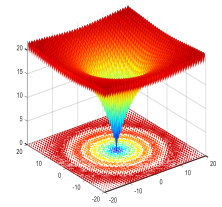
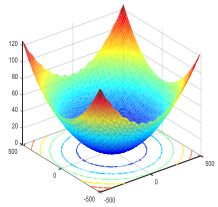
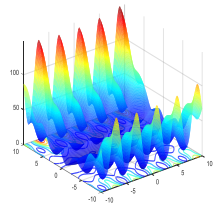
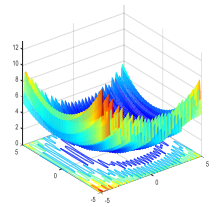
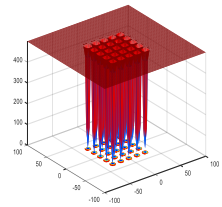
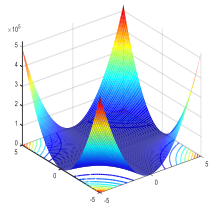
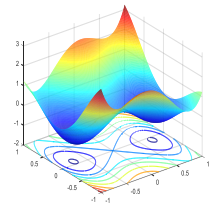
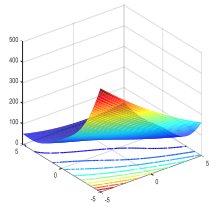
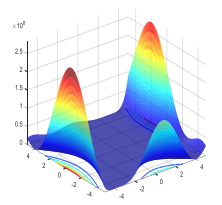
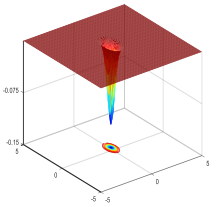
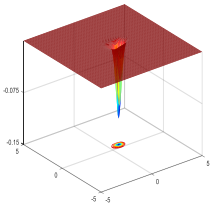
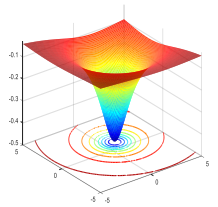
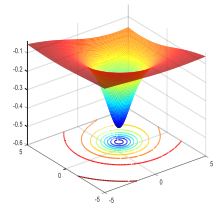
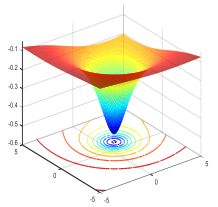
Table 5.7 Coefficients of Hartman's Functions (F20)

Index (i)	$a_{ij}, j = 1, 2, 3$						c_i	$p_{ij}, j = 1, 2, 3$					
1	10	3	17	3.5	1.7	8	1	0.1312	0.1696	0.5569	0.0124	0.8283	0.5886
2	0.5	10	17	0.1	8	14	1.2	0.2329	0.4135	0.8307	0.3736	0.1004	0.9991
3	3	3.5	1.7	10	17	8	3	0.2348	0.1415	0.3522	0.2883	0.3047	0.6650
4	17	8	0.05	10	0.1	14	3.2	0.4047	0.8828	0.8732	0.5743	0.1091	0.0381

Table 5.8 Coefficients of Shekel's Functions (F21–F23)

Index (i)	$a_{ij}, j = 1, \ldots, 4$				c_i
1	4	4	4	4	0.1
2	1	1	1	1	0.2
3	8	8	8	8	0.4
4	6	6	6	6	0.4
5	3	7	3	7	0.4
6	2	9	2	9	0.6
7	5	5	3	3	0.3
8	8	1	8	1	0.7
9	6	2	6	2	0.5
10	7	3.6	7	3.6	0.5

Table 5.9 3D Surface Plots of the Benchmark Functions F1–F23

F1	**F2**	**F3**	**F4**
			
F5	**F6**	**F7**	**F8**
			
F9	**F10**	**F11**	**F12**
			
F13	**F14**	**F15**	**F16**
			
F17	**F18**	**F19**	**F20**
			
F21	**F22**	**F23**	
			

5.6.2 Parameter Settings

We chose the constant *k* to be 5 and θ to be the product of a random number drawn from a zero-mean Gaussian distribution with a standard deviation of 0.5 and a factor of the order of 10^{-3} after sufficient number of trials. The learning rate χ decreases linearly with iterations from 1 to 0.3 according to equation (5.24) as an LTV weight [236]. $\rho^{Chebyshev} \in [0,1]$ is a random number generated using a Chebyshev chaotic map in equation (5.27). All experiments were carried out on two Intel(R) Core (TM) i7-5500U CPUs @ 2.40GHz with 8GB RAM and one Intel(R) Core (TM) i7-2600U CPU @ 3.40GHz with 16GB RAM using MATLAB R2017a. All experiments were independently repeated 30 times in order to account for variability in reported data due to the underlying stochasticity of the metaheuristics used. Clusters from the MATLAB Parallel Computing Cloud were utilized to speed up the benchmarking.

5.7 Experimental Results

Tables 5.10 through 5.12 report performances of the C-QDDS algorithm on the test problems stacked against solution qualities obtained using eight other commonly used, recent nature-inspired approaches: (i) Sine Cosine Algorithm (SCA) [246], (ii) Dragon Fly Algorithm (DFA) [247], (iii) Ant Lion Optimization (ALO) [248], (iv) Whale Optimization Algorithm (WOA) [249], (v) Firefly Algorithm (FA) [250], (vi) Quantum-behaved Particle Swarm Optimization (QPSO) [230] [231], (vii) Particle Swarm Optimization with Damped Inertia ** (PSO-I), and (viii) the canonical Particle Swarm Optimizer (PSO-II) [1]. Each algorithm has been executed for 1000 iterations with 30 independent trials following which their mean, standard deviation, and minimum values are noted. Testing carried out on the functions adhere to the dimensionalities and range constraints specified in Tables 5.2 through

5.9. A total of 50 agents have been introduced in the particle pool, out of which only one agent is picked in each iteration. The rationale for choosing one agent instead of many or all from the pool is to investigate the incremental effect of a single agent's propagation under different nature-inspired dynamical perturbations. The ripple effect caused otherwise, by many sensor reading exchanges among many or all particles, may be delayed when a single particle affects the global pool of particles in one iteration.

** PSO-I utilizes an exponentially decaying inertia weight for exploration–exploitation trade-off

5.7.1 Test Results on Optimization Problems

Table 5.10 Solution quality in unimodal functions in Table 5.2 (30D, 1000 iterations, 30 independent trials)

	Stat	C-QDDS Chebyshev Map	Sine Cosine Algorithm	Dragon Fly Algorithm	Ant Lion Optimization	Whale Optimization	Firefly Algorithm	QPSO	PSO w=0.95*w	PSO No Damping
F1	Mean	1.1956e-06	0.0055	469.8818	**7.8722e-07**	17.3824	3.5794e+04	3.0365e+03	109.5486	110.3989
	Min	5.1834e-07	1.0207e-07	23.9914	**8.9065e-08**	0.6731	3.0236e+04	1.3286e+03	39.3329	42.8825
	Std	**2.8711e-07**	0.0161	474.0822	1.0286e-06	19.6687	3.3373e+03	920.4817	43.3127	54.7791
F2	Mean	0.0051	**3.6862-06**	9.2230	27.8542	0.7846	3.4566e+04	36.4162	4.2299	4.4102
	Min	0.0025	2.7521e-09	**0**	0.0029	0.0745	84.8978	21.7082	2.0290	2.1627
	Std	9.7281e-04	**8.9681e-06**	5.7226	42.2856	0.5303	1.3595e+05	12.5312	1.1111	1.3804
F3	Mean	**1.0265e-04**	3.4383e+03	6.3065e+03	302.3783	1.0734e+05	4.4017e+04	3.0781e+04	4.0409e+03	3.4218e+03
	Min	**1.0184e-05**	27.3442	310.7558	102.7732	5.0661e+04	3.0021e+04	1.8940e+04	2.2416e+03	1.9223e+03
	Std	**6.5905e-05**	3.1641e+03	4.7838e+03	167.7687	4.0661e+04	6.6498e+03	5.9848e+03	994.2550	997.4284
F4	Mean	**3.6945e-04**	12.8867	13.8222	8.8157	66.4261	68.4102	56.5926	12.8272	11.9252
	Min	**1.4162e-04**	1.4477	4.1775	2.0212	17.8904	62.9296	32.6744	10.2302	9.0857
	Std	**9.8034e-05**	8.1625	5.5197	3.0808	21.5187	2.6497	8.2985	1.5793	2.0508
F5	Mean	**28.7211**	60.7787	2.0123e+04	143.9657	1.5976e+03	7.4584e+07	2.1204e+06	6.0590e+03	5.3377e+03
	Min	28.7074	28.0932	44.0682	**20.7989**	39.9132	3.8917e+07	5.0759e+05	655.5618	1.2610e+03
	Std	**0.0077**	55.2793	3.6793e+04	288.1879	3.0458e+03	2.0606e+07	9.1390e+05	4.2558e+03	2.6303e+03
F6	Mean	7.2332	4.2963	488.3942	**6.0117e-07**	30.0158	3.6216e+04	3.6028e+03	107.5196	116.9431
	Min	6.4389	3.3201	17.4978	**8.9390e-08**	0.8531	2.8838e+04	1.8380e+03	45.9374	28.9258
	Std	0.5612	0.4007	309.2795	**6.2634e-07**	44.1595	2.8434e+03	986.7972	47.5633	49.5767
F7	Mean	**0.0037**	0.0289	0.1491	0.0541	0.1265	36.0335	1.4761	0.1737	0.1749
	Min	**4.9685e-04**	0.0010	0.0157	0.0210	0.0177	21.1334	0.3837	0.0697	0.0734
	Std	**0.0023**	0.0472	0.0918	0.0229	0.0993	7.5632	0.7718	0.0561	0.0690

Table 5.11 Solution quality in multimodal functions in Table 5.3 (30D, 1000 iterations, 30 independent trials)

	Stat	C-QDDS Chebyshev Map	Sine Cosine Algorithm	Dragon Fly Algorithm	Ant Lion Optimizer	Whale Optimization	Firefly Algorithm	QPSO	PSO w=0.95*w	PSO No Damping
F8	Mean	-602.2041	-4.0397e+03	-6.001e+03	-5.5942e+03	**-8.5061e+03**	-3.8714e+03	-3.3658e+03	-5.1487e+03	-4.8821e+03
	Best	-975.5422	-4.4739e+03	-8.9104e+03	-8.2843e+03	**-1.0768e+04**	-4.2603e+03	-5.0298e+03	-7.4208e+03	-6.6643e+03
	Std	**160.8409**	214.0523	783.7255	515.1599	895.4642	204.0029	486.0400	766.3330	750.3092
F9	Mean	**2.4873e-04**	8.8907	124.0432	79.9945	116.4796	328.4011	248.0831	57.8114	57.1125
	Best	8.2194e-05	**1.0581e-06**	32.1699	45.7681	0.4305	308.3590	177.8681	19.1318	27.4985
	Std	**6.3770e-05**	16.2284	40.4730	22.2932	88.0344	10.1050	31.9501	15.1644	15.0292
F10	Mean	**8.1297e-04**	10.7873	6.0693	1.6480	1.1419	19.3393	12.3433	4.9951	4.9271
	Best	5.6777e-04	3.4267e-05	**8.8818e-16**	1.7296e-04	0.0265	18.4515	9.7835	3.9874	2.9208
	Std	**8.8526e-05**	9.6938	1.9141	0.9544	0.9926	0.2797	1.8413	0.6230	0.7957
F11	Mean	**8.7473e-08**	0.1770	5.0784	0.0082	1.1735	316.5026	33.5446	2.0669	2.0604
	Best	**3.5705e-08**	2.2966e-05	1.1727	2.5498e-05	0.9839	226.5205	11.9701	1.3636	1.3744
	Std	**2.6504e-08**	0.2195	4.5098	0.0093	0.2340	33.3806	12.5605	0.5989	0.5366
F12	Mean	**0.0995**	991.4301	12.2571	9.4380	642.0404	1.2629e+08	5.6147e+05	6.4329	6.5610
	Best	**0**	0.2878	1.6755	3.4007	0.0442	5.6104e+07	4.0841e+04	1.0266	2.8742
	Std	**0.2621**	5.4201e+03	13.5218	3.9121	3.5039e+03	4.4034e+07	6.9761e+05	2.7882	3.0003
F13	Mean	**0.0105**	3.1940	1.5156e+04	0.0133	2.3405e+03	2.8867e+08	3.5568e+06	38.1945	39.0369
	Best	**0**	1.8776	5.6609	2.7212e-07	0.3813	1.3101e+08	6.8216e+05	12.7653	15.4619
	Std	0.0576	2.2922	6.0811e+04	**0.0163**	1.2373e+04	8.1766e+07	2.4393e+06	15.2922	27.6751

Table 5.12 Solution quality in multimodal functions in Table 5.4 (fixed dim, 1000 iterations, 30 trials)

Fn	Stat	C-QDDS Chebyshev Map	Sine Cosine Algorithm	Dragon Fly Algorithm	Ant Lion Optimizer	Whale Optimization	Firefly Algorithm	QPSO	PSO w=0.95*w	PSO No Damping
F14, n=2	Mean	3.6771	1.3949	**1.0311**	1.2299	4.2524	1.0519	2.3561	2.7786	3.7082
	Best	1.0056	**0.9980**	**0.9980**	**0.9980**	**0.9980**	**0.9980**	0.9981	**0.9980**	**0.9980**
	Std	2.2295	0.8072	**0.1815**	0.4276	3.7335	0.1889	1.7188	2.2246	2.7536
F15, n=4	Mean	**3.7361e-04**	9.1075e-04	0.0016	0.0027	0.0051	0.0024	0.0030	0.0036	0.0034
	Best	3.1068e-04	3.1549e-04	4.7829e-04	4.0518e-04	3.4820e-04	0.0011	7.2169e-04	3.6642e-04	**3.0858e-04**
	Std	**5.0123e-05**	4.2242e-04	0.0014	0.0060	0.0076	0.0012	0.0059	0.0063	0.0068
F16, n=2	Mean	-0.5487	**-1.0316**	**-1.0316**	**-1.0316**	-1.0315	-1.0295	**-1.0316**	**-1.0316**	**-1.0316**
	Best	-1.0315	**-1.0316**	**-1.0316**	**-1.0316**	**-1.0316**	**-1.0316**	**-1.0316**	**-1.0316**	**-1.0316**
	Std	0.4275	1.1863e-05	1.4229e-06	**3.6950e-14**	3.3613e-04	0.0030	1.1009e-04	8.2108e-14	2.7251e-13
F17, n=2	Mean	0.4721	0.3983	**0.3979**	**0.3979**	0.4069	0.4002	0.4000	**0.3979**	**0.3979**
	Best	0.3989	**0.3979**	**0.3979**	**0.3979**	**0.3979**	**0.3979**	**0.3979**	**0.3979**	**0.3979**
	Std	0.0920	4.8435e-04	4.9327e-08	**2.3588e-14**	0.0179	0.0020	0.0043	5.0770e-10	2.1067e-08
F18, n=2	Mean	3.8438	**3**	**3**	**3**	3.9278	3.0402	3.0007	**3.0000**	**3.0000**
	Best	3.0080	**3**	**3**	**3**	**3.0000**	3.0002	**3.0000**	**3.0000**	**3.0000**
	Std	0.9128	5.7657e-06	8.7817e-07	**1.2869e-13**	5.0752	0.0397	0.0017	1.0155e-11	5.8511e-11
F19, n=3	Mean	-3.6805	-3.8547	-3.8625	**-3.8628**	-3.8246	-3.8542	**-3.8628**	**-3.8628**	**-3.8628**
	Best	-3.8587	-3.8626	**-3.8628**	**-3.8628**	**-3.8628**	-3.8625	**-3.8628**	**-3.8628**	**-3.8628**
	Std	0.1942	0.0016	8.8455e-04	**7.5193e-15**	0.0657	0.0066	1.5043e-05	5.2841e-11	9.2140e-11
F20, n=6	Mean	-2.2207	-2.9961	-3.2421	**-3.2705**	-3.0966	-3.0645	-3.2646	-3.2625	-3.2546
	Best	-2.7562	-3.2911	**-3.3220**	**-3.3220**	-3.2610	-3.2436	-3.3219	**-3.3220**	**-3.3220**
	Std	0.29884	0.2060	0.0670	**0.0599**	0.1535	0.0911	0.0605	0.0605	**0.0599**
F21, n=4	Mean	-3.1126	-4.0962	**-9.0360**	-6.7752	-6.5291	-4.3198	-5.8537	-5.3955	-5.4045
	Best	-4.5610	-5.3343	**-10.1532**	**-10.1532**	-9.8465	-7.5958	-10.1474	**-10.1532**	**-10.1532**
	Std	**0.7090**	1.5519	1.9130	2.6824	1.9988	1.4599	3.5651	3.3029	3.4897
F22, n=4	Mean	-3.2009	-3.9949	**-10.0455**	-7.2979	-6.3611	-4.2776	-6.7830	-5.3236	-6.3098
	Best	-4.5933	-7.9241	**-10.4029**	**-10.4029**	-10.2432	-9.2741	-10.3974	**-10.4029**	**-10.4029**
	Std	**0.7098**	2.1774	1.3422	3.0440	2.3852	1.6527	3.5783	3.2000	3.4602
F23, n=4	Mean	-2.3595	-4.6650	**-9.9928**	-7.1691	-5.2592	-4.6959	-7.5372	-7.3175	-5.1501
	Best	-4.2043	-7.7259	**-10.5364**	**-10.5364**	-10.0617	-8.5734	-10.5344	**-10.5364**	**-10.5364**
	Std	**0.8183**	1.5038	1.6439	3.2926	2.5389	1.4647	3.6778	3.7753	3.4033

Table 5.13 Win/tie/loss count among competitors w.r.t. reported global best

Perfor-mance	Metric	C-QDDS Chebyshev Map	Sine Cosine Algorithm	Dragon Fly Algorithm	Ant Lion Optimizer	Whale Optimization	Firefly Algorithm	QPSO	PSO w=0.95*w	PSO No Damping
Win	Mean	**10**	1	3	3	1	0	0	0	0
	Best	**6**	1	2	3	1	0	0	0	1
	Std	**14**	1	1	7	0	0	0	0	0
Tie	Mean	0	2	3	**4**	0	0	2	**4**	5
	Best	0	4	**9**	**9**	5	3	4	**9**	**9**
	Std	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Lose	Mean	**13**	20	17	17	22	23	21	19	18
	Best	17	18	**12**	13	18	20	19	14	13
	Std	**9**	22	22	17	23	23	23	23	23

Table 5.14 Average ranks based on win/tie/loss count among competitors w.r.t. reported global best

Perfor-mance	Metric	C-QDDS Chebyshev Map	Sine Cosine Algorithm	Dragon Fly Algorithm	Ant Lion Optimizer	Whale Optimization	Firefly Algorithm	QPSO	PSO w=0.95*w	PSO No Damping
Win	Mean	**1**	3	2	2	3	4	4	4	4
	Best	**1**	4	3	2	4	5	5	5	4
	Std	**1**	3	3	2	4	4	4	4	4
Tie	Mean	5	4	3	**2**	5	5	4	**2**	**1**
	Best	5	3	2	2	3	4	3	2	**1**
	Std	1	1	1	**1**	1	1	1	1	**1**
Lose	Mean	**1**	5	2	2	7	8	6	4	3
	Best	4	5	**1**	**2**	5	7	6	3	**2**
	Std	**1**	3	3	2	4	4	4	4	3
Average Rank	Mean	**2.333**	4	**2.333**	2	5	5.666	4.666	3.333	2.666
	Best	3.333	4	**2**	**2**	4	5.333	4.666	3.333	2.333
	Std	**1**	2.333	2.333	1.666	3	3	3	3	2.666

5.7.2 Statistical Significance of Results

Table 5.15 Results of two-tailed *t*-test for C-QDDS vs. competitors

Algorithm	C-QDDS vs. SCA	C-QDDS vs. DFA	C-QDDS vs. ALO	C-QDDS vs. WOA	C-QDDS vs. FA	C-QDDS vs. QPSO	C-QDDS vs. PSO-II	C-QDDS vs. PSO-I
Function	**t values ($t_{critical}$ = 2.001717). Null Hypothesis: ($\mu_{\_CQDDS} - \mu_{\_Competitor}$) > 0**							
F1	-1.8707	-5.4287	2.094532	-4.84055	-58.7456	-18.0684	-13.8533	-11.0385
F2	28.69263	-8.82265	-3.60728	-8.05108	-1.39261	-15.9148	-20.8264	-17.4788
F3	-5.95188	-7.22065	-9.87189	-14.4592	-36.2554	-28.1704	-22.2608	-18.7903
F4	-8.64702	-13.7155	-15.6724	-16.9076	-141.411	-37.3523	-44.4852	-31.8485
F5	-3.17636	-2.99135	-2.19031	-2.8213	-19.825	-12.7079	-7.76098	-11.0552
F6	23.32769	-8.52117	70.59491	-2.82556	-69.7487	-19.9572	-11.5478	-12.12
F7	-2.92082	-8.67254	-11.9943	-6.77163	-26.0926	-10.4491	-16.5837	-13.5823
F8	70.32003	36.96027	50.66345	47.58374	68.92735	29.56635	31.80233	30.54904
F9	-3.0006	-16.7868	-19.6538	-7.24698	-178.004	-42.529	-20.8808	-20.8139
F10	-6.09462	-17.3651	-9.45308	-6.29659	-378.696	-36.7146	-43.9082	-33.9102
F11	-4.41671	-6.1678	-4.82933	-27.4681	-51.933	-14.6277	-18.9028	-21.0311
F12	-1.00178	-4.92371	-13.0453	-1.00347	-15.7087	-4.40833	-12.3869	-11.7511
F13	-7.60459	-1.36509	-0.25619	-1.03608	-19.337	-7.98647	-13.6763	-7.72376
F14	5.271808	6.47901	5.904436	-0.72462	6.426319	2.570191	1.562548	-0.04808
F15	-6.9162	-4.79494	-2.12362	-3.40618	-9.2411	-2.4381	-2.80494	-2.43761
F16	6.187023	6.187023	6.187023	6.18574	6.159965	6.187023	6.187023	6.187023
F17	4.393627	4.417501	4.417501	3.810236	4.27956	4.287797	4.417501	4.417501
F18	5.063193	5.063193	5.063193	-0.08922	4.81742	5.058984	5.063193	5.063193
F19	4.912978	5.133083	5.141597	3.849854	4.896216	5.141597	5.141597	5.141597
F20	11.70106	18.26704	18.86578	14.28008	14.7933	18.75247	18.71474	18.58005
F21	3.157566	15.90258	7.230404	8.823441	4.074111	4.13039	3.701433	3.525209
F22	1.898948	24.69127	7.179345	6.955444	3.278706	5.378252	3.547072	4.820763
F23	7.375911	22.76815	7.76455	5.953976	7.627314	7.526917	7.029854	4.366702
Significantly better	**9**	**12**	**10**	**11**	**12**	**13**	**13**	**13**
Significantly worse	**11**	**10**	**12**	**8**	**10**	**10**	**9**	**9**

Table 5.16 Cohen's d-values for C-QDDS v/s competitors

Algorithm	C-QDDS vs. SCA	C-QDDS vs. DFA	C-QDDS vs. ALO	C-QDDS vs. WOA	C-QDDS vs. FA	C-QDDS vs. QPSO	C-QDDS vs. PSO-II	C-QDDS vs. PSO-I
Function	**Cohen's d-values, where** $d = \frac{\mu_{C-QDDS} - \mu_{Competitor}}{\sqrt{\frac{s^2_{\mu\_CQDDS} + s^2_{\mu\_Competitor}}{2}}}$							
F1	-0.483	-1.4017	0.5408	-1.2498	-15.1681	-4.6652	-3.5769	-2.8501
F2	7.4084	-2.278	-0.9314	-2.0788	-0.3596	-4.1092	-5.3773	-4.513
F3	-1.5368	-1.8644	-2.5489	-3.7333	-9.3611	-7.2736	-5.7477	-4.8516
F4	-2.2327	-3.5413	-4.0466	-4.3655	-36.5121	-9.6443	-11.486	-8.2233
F5	-0.8201	-0.7724	-0.5655	-0.7285	-5.1188	-3.2812	-2.0039	-2.8544
F6	6.0232	-2.2002	18.2275	-0.7296	-18.009	-5.1529	-2.9816	-3.1294
F7	-0.7542	-2.2392	-3.0969	-1.7484	-6.7371	-2.698	-4.2819	-3.5069
F8	18.1566	9.5431	13.0812	12.2861	17.797	7.634	8.2113	7.8877
F9	-0.7748	-4.3343	-5.0746	-1.8712	-45.9603	-10.9809	-5.3914	-5.3741
F10	-1.5736	-4.4836	-2.4408	-1.6258	-97.7789	-9.4797	-11.3371	-8.7556
F11	-1.1404	-1.5925	-1.2469	-7.0922	-13.4091	-3.7769	-4.8807	-5.4302
F12	-0.2587	-1.2713	-3.3683	-0.2591	-4.056	-1.1382	-3.1983	-3.0341
F13	-1.9635	-0.3525	-0.0661	-0.2675	-4.9928	-2.0621	-3.5312	-1.9943
F14	1.3612	1.6729	1.5245	-0.1871	1.6593	0.6636	0.4034	-0.0124
F15	-1.7858	-1.238	-0.5483	-0.8795	-2.386	-0.6295	-0.7242	-0.6294
F16	1.5975	1.5975	1.5975	1.5972	1.5905	1.5975	1.5975	1.5975
F17	1.1344	1.1406	1.1406	0.9838	1.105	1.1071	1.1406	1.1406
F18	1.3073	1.3073	1.3073	-0.023	1.2439	1.3062	1.3073	1.3073
F19	1.2685	1.3254	1.3276	0.994	1.2642	1.3276	1.3276	1.3276
F20	3.0212	4.7165	4.8711	3.6871	3.8196	4.8419	4.8321	4.7973
F21	0.8153	4.106	1.8669	2.2782	1.0519	1.0665	0.9557	0.9102
F22	0.4903	6.3753	1.8537	1.7959	0.8466	1.3887	0.9158	1.2447
F23	1.9045	5.8787	2.0048	1.5373	1.9694	1.9434	1.8151	1.1275

Table 5.17 Hedges' g-values for C-QDDS vs. competitors

Algorithm	C-QDDS vs. SCA	C-QDDS vs. DFA	C-QDDS vs. ALO	C-QDDS vs. WOA	C-QDDS vs. FA	C-QDDS vs. QPSO	C-QDDS vs. PSO-II	C-QDDS vs. PSO-I
Function	**Hedge's g-values, where** $g = \frac{\mu_{\text{C-QDDS}} - \mu_{\text{Competitor}}}{\sqrt{\frac{(n_1-1)*s^2_{\mu\_\text{CQDDS}} + (n_2-1)*s^2_{\mu\_\text{Competitor}}}{n_1+n_2-2}}}$							
F1	-0.6716	-1.949	0.752	-1.7378	-21.0904	-6.4867	-4.9735	-3.9629
F2	10.301	-3.1674	-1.2951	-2.8905	-0.5	-5.7136	-7.4768	-6.2751
F3	-2.1368	-2.5923	-3.5441	-5.1909	-13.0161	-10.1135	-7.9919	-6.7459
F4	-3.1044	-4.924	-5.6266	-6.07	-50.768	-13.4099	-15.9706	-11.434
F5	-1.1403	-1.074	-0.7863	-1.0129	-7.1174	-4.5623	-2.7863	-3.9689
F6	8.3749	-3.0593	25.3443	-1.0145	-25.0405	-7.1648	-4.1457	-4.3513
F7	-1.0487	-3.1135	-4.3061	-2.4311	-9.3676	-3.7514	-5.9537	-4.8761
F8	25.2457	13.2691	18.1887	17.0831	24.7457	10.6146	11.4173	10.9674
F9	-1.0773	-6.0266	-7.0559	-2.6018	-63.9052	-15.2683	-7.4964	-7.4724
F10	-2.188	-6.2342	-3.3938	-2.2606	-135.956	-13.181	-15.7636	-12.1742
F11	-1.5857	-2.2143	-1.7337	-9.8613	-18.6446	-5.2516	-6.7863	-7.5504
F12	-0.3597	-1.7677	-4.6834	-0.3603	-5.6396	-1.5826	-4.4471	-4.2187
F13	-2.7301	-0.4901	-0.0919	-0.3719	-6.9422	-2.8672	-4.9099	-2.773
F14	1.8927	2.3261	2.1197	-0.2602	2.3072	0.9227	0.5609	-0.0172
F15	-2.4831	-1.7214	-0.7624	-1.2229	-3.3176	-0.8753	-1.007	-0.8751
F16	2.2212	2.2212	2.2212	2.2208	2.2115	2.2212	2.2212	2.2212
F17	1.5773	1.5859	1.5859	1.3679	1.5364	1.5394	1.5859	1.5859
F18	1.8177	1.8177	1.8177	-0.032	1.7296	1.8162	1.8177	1.8177
F19	1.7638	1.8429	1.846	1.3821	1.7578	1.846	1.846	1.846
F20	4.2008	6.558	6.773	5.1267	5.3109	6.7324	6.7188	6.6704
F21	1.1336	5.7092	2.5958	3.1677	1.4626	1.4829	1.3288	1.2656
F22	0.6817	8.8645	2.5775	2.4971	1.1771	1.9309	1.2734	1.7307
F23	2.6481	8.174	2.7876	2.1375	2.7383	2.7022	2.5238	1.5677

5.7.3 Fraction of Successful Runs v/s Cost Range

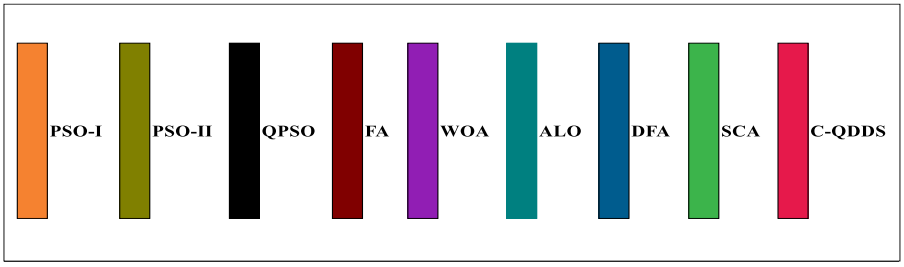
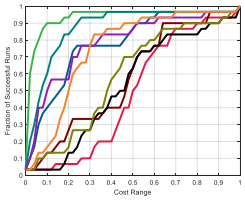
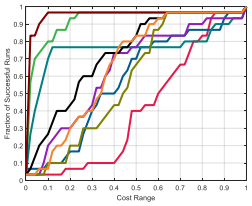
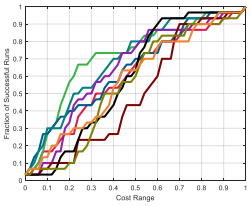
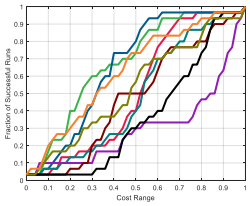
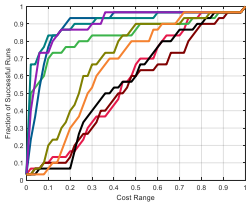
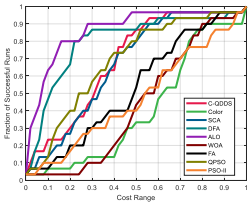
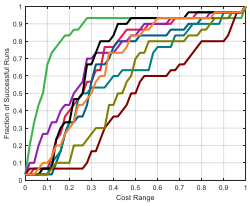
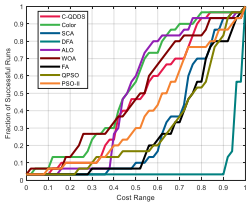
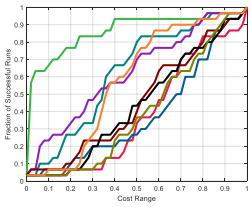
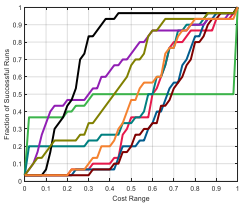
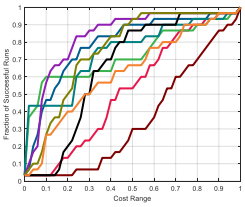
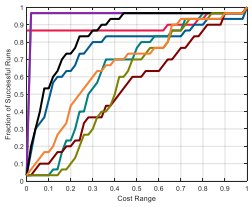
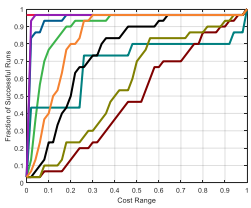
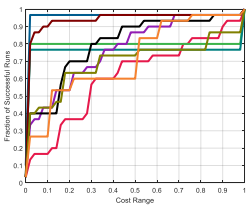
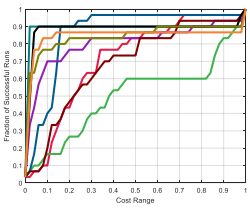
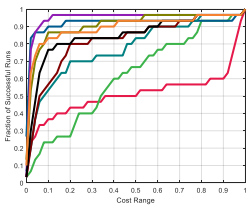
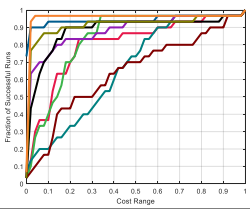
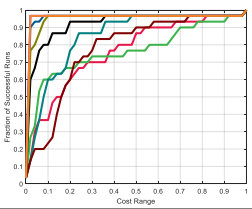
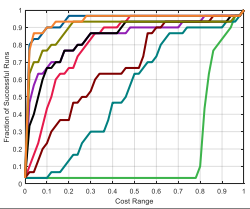
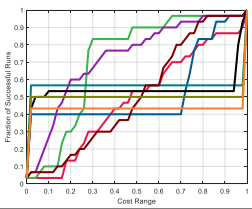
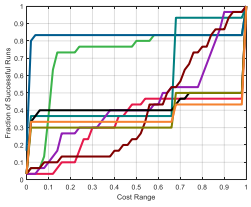
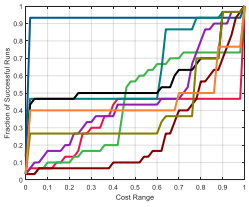
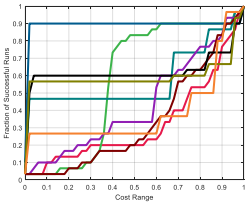
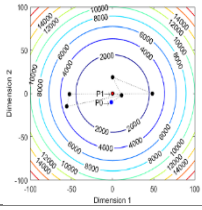
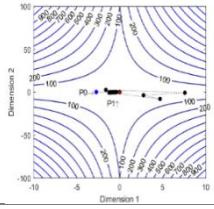
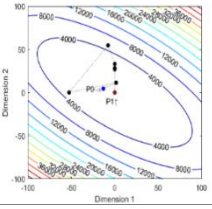
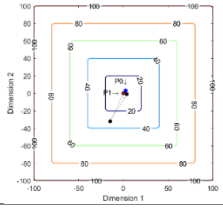
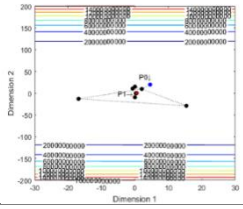
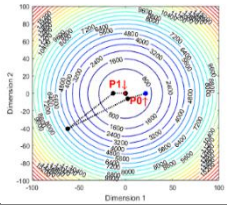
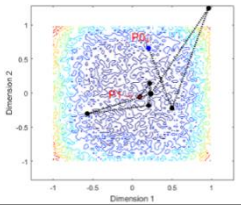
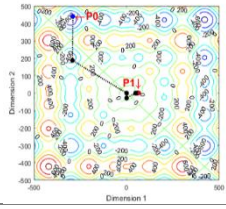
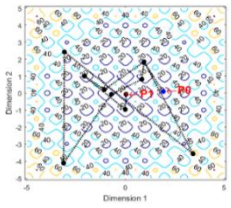
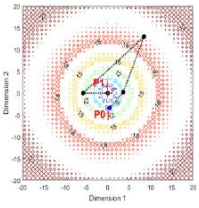
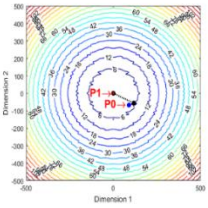
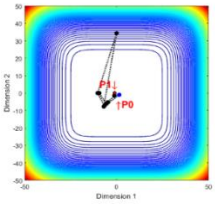
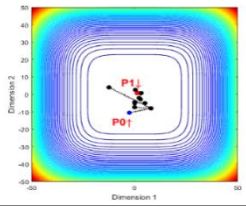
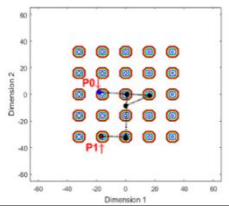
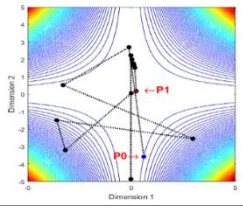
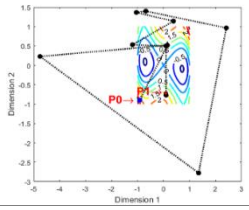
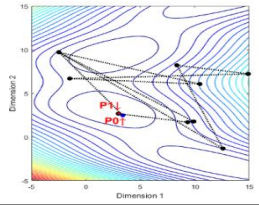
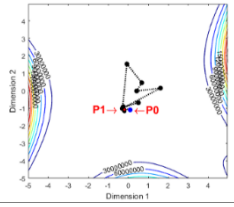
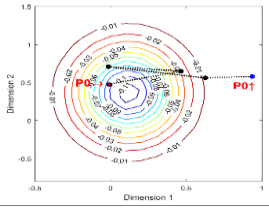
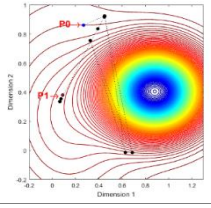
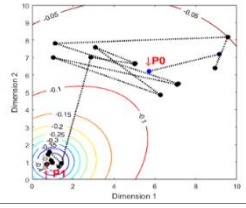
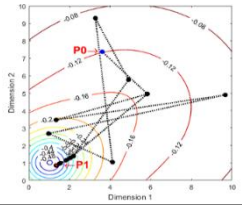
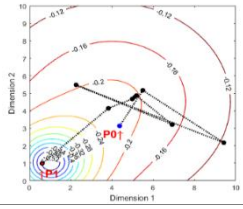


Table 5.18 Precision plots (fraction of successful runs v/s. cost range) for 23 functions

F1	**F2**	**F3**	**F4**
			
F5	**F6**	**F7**	**F8**
			
F9	**F10**	**F11**	**F12**
			
F13	**F14**	**F15**	**F16**
			
F17	**F18**	**F19**	**F20**
			
F21	**F22**	**F23**	
			

5.7.4 Trajectory Tracking of Best Performing Agents Over Time

Table 5.19 Trajectory of the best solutions for the 23 benchmark functions

F1	**F2**	**F3**	**F4**
			
F5	**F6**	**F7**	**F8**
			
F9	**F10**	**F11**	**F12**
			
F13	**F14**	**F15**	**F16**
			
F17	**F18**	**F19**	**F20**
			
F21	**F22**	**F23**	
			

5.8 Analysis of Experimental Results

Tables 5.10–5.12 report the solution qualities obtained on the suite of test functions F1–F23 followed by Tables 5.13–5.17 in which the win/tie/loss counts, average ranks, and results of statistical significance tests such as that of a two-tailed *t*-test and Cohen's d and Hedge's g values are reported. From Tables 5.10–5.12 one can make the observation that C-QDDS has a distinctive advantage over the other algorithms in terms of quality of optima found, outperforming competitors in unimodal functions as F3–F5, F7, and multimodal ones such as F9–F13. However, solution quality drops for the multimodal functions F14–F23, with the agents getting stuck in local minima. One interpretation is that since communication between particles is limited when only one agent is drawn in an iteration, it will take a considerably large number of iterations for promising regions to be found. Alternatively, because the QDDS mechanism is based on gradient descent, saddle points and valleys introduce stagnation which is difficult to break out of. A two-tailed Student's *t*-test with significance level $\Theta = 0.05$ in Table 5.15 is used to accept or reject the hypothesis that the performance of the C-QDDS algorithm is significant when compared to any of the other approaches. It is observed that in general, C-QDDS provides superior solution quality when applied to problems in Tables 5.10–5.11 and that the difference is statistically significant at $\Theta = 0.05$. A measure of the effect sizes is provided in Table 5.16 through the computation of Cohen's d values, however to account for the correction Hedge's g values have also been reported in Table 5.17.

In Table 5.18, the number of successful executions against the obtained cost range for any algorithm is demonstrated for all test functions. The horizontal axis represents a value equivalent to the sum of the lowest cost obtained during the 30 runs of an algorithm and a

fraction of the cost range i.e., (maximum cost) – (minimum cost), ranging from 0.1 through 1 at intervals of 0.1. The vertical axis is the cumulative number of trials that resulted in solutions with lower cost than the corresponding horizontal axis value. For example, the vertical axis value at the horizontal tick of 0.1 is the number of trials having cost values less than $[(\text{minimum cost}) + 0.1 \times \{(\text{maximum cost}) - (\text{minimum cost})\}]$. These curves are a measure of the variability of the algorithmic solutions within their reported cost ranges and an indicator of how top-heavy or bottom-heavy they are. It is important to note that the cost range for each algorithm is different on every test function execution and as such the curves are merely meant for an intuitive understanding of the variability of the solutions and not intended to provide any basis for comparison among the algorithms. Algorithms having the least standard deviation among the cohort are expected to have a uniform density of solutions in the cost range and as such should follow a roughly linear relationship between the variables in the horizontal and vertical axes. It may be noted that C-QDDS, which roughly follows this relationship, indeed has the least standard deviation in many cases, specifically for 14 of the 23 functions as illustrated in Table 5.13. This is in congruence with the convergence profiles of QDDS in Figures 1–12 of [229] which point out that QDDS is fairly consistent in its ability to converge to local optima of acceptable quality in certain problems.

Table 5.19 shows the trajectory evolution of the global best position across the functional iterations for each test case using C-QDDS. For ease of visualization, the contours of the 30-dimensional functions as well as the obtained gbest, i.e., global best solutions are plotted using only the first two dimensions. P_0 represents the initial gbest position and P_1 represents the gbest position upon convergence, given the convergence criteria. The interim gbest position transitions are shown by dotted lines. The solutions to the 23 test problems

outlined in the paper are local minima, however the quality of solutions that the C-QDDS and QDDS algorithm provide to some of these problems are markedly better than those reported in some studies in the literature [232] [221] [251] [252]. A logical next-step to improve the optima seeking capability of the QDDS/C-QDDS approach is to introduce a problem-independent random walk in the δ recomputing step of the algorithm instead of using gradient descent.

5.9 Notes on Convergence of the Algorithm

In this section, we discuss the convergence characteristics of the QDDS algorithm by formulating the algorithmic objective as an optimization problem and proving hypotheses adherence under certain weak assumptions. We start by considering following problem $\mathbb{C}$.

$\mathbb{C}$**:** Provided there is a function $\boldsymbol{f}$ from $\mathbb{R}^{\boldsymbol{n}}$ to $\mathbb{R}$ and that $\boldsymbol{S}$ is a subset of $\mathbb{R}^{\boldsymbol{n}}$, a solution $\boldsymbol{x}$ in $\boldsymbol{S}$ is sought such that $\boldsymbol{x}$ minimizes $\boldsymbol{f}$ on $\boldsymbol{S}$ or finds an acceptable approximation of the minimum of $\boldsymbol{f}$ on $\boldsymbol{S}$.

A conditioned approach to solving $\mathbb{C}$ was proposed by Solis and Wet [233] which we describe below. The rest of the proof follows logically from [233] as has also been shown by Van den Bergh in [253] and Sun et al. in [254].

Algorithm 5.2. A conditioned approach to solving $\mathbb{C}$ **[233]**

1: Initialize x^0 in S and set $e = 0$

2: Generate ξ^e from the sample space $(\mathbb{R}^n, \mathbb{B}, \mathbb{T}_e)$

3: Update $x^{e+1} = £(x^e, \xi^e)$, choose $\mathbb{T}_{e+1}$, set $e = e+1$ and repeat Step 1.

The mapping £ is the optimization algorithm and should satisfy the following two hypotheses $\mathbb{H}^{\boldsymbol{1}}$ and $\mathbb{H}^{\boldsymbol{2}}$ in order to theoretically be globally convergent.

Hypothesis $\mathbb{H}^1$:

$$f(£(x,\xi)) \leq f(x) \text{ and if } \xi \in S \text{ then } f(£(x,\xi)) \leq f(\xi) \tag{5.29}$$

The sequence $f(x_e)_{e=1}^{\infty}$generated by £ must monotonically reach a stable value, i.e. the infimum, for the mapping to be a globally convergent one.

Hypothesis $\mathbb{H}^2$:

For any Borel subset *A* of *S* with $\vartheta(A) > 0$, it can be proved that

$$\prod_{k=0}^{\infty}\{1 - \mathbb{T}_e(A)\} = 0 \tag{5.30}$$

This means that if there exists a subset *A* of *S* with positive volume then the chance that upon generating random samples ξ^e it will repeatedly miss *A* is zero. Guided random search methods are conditioned, which implies $\mathbb{T}_e$ depends on x^0, x^1,..., x^{e-1} generated in the preceding iterations. Therefore, $\mathbb{T}_e(A)$ is a conditional probability measure.

Definition $\mathbb{D}^1$:

Values close to the essential infimum σ is generated by a set of points having a non-zero ϑ measure.

$$\sigma = \inf\{t: \vartheta[x \in S \mid f(x) < t] > 0\} \tag{5.31}$$

Definition $\mathbb{D}^2$:

The acceptable solution range $\mathfrak{R}_{\varepsilon,\mathfrak{S}}$ for $\mathbb{P}$ is constructed around the essential infimum σ with step size ε and bounded support $\mathfrak{S}$.

$$\mathfrak{R}_{\varepsilon,\mathfrak{S}} = \begin{cases} x \in S \mid f(x) < \sigma + \varepsilon\,, & \sigma \in (-\infty, \infty) \\ x \in S \mid f(x) < \mathfrak{S}, & \sigma \; is \; infinite \end{cases} \tag{5.32}$$

Theorem $\mathbb{T}^1$:

The Global Convergence Theorem for Random Search Algorithms states that when $\mathbb{H}^1$ and $\mathbb{H}^2$ are satisfied on a measurable subset of $\mathbb{R}^n$ for a measurable function f, the probability that the conditioned sequence $\{x_e\}_{e=1}^{\infty}$ generated by the algorithm lies within the acceptable solution range $\mathfrak{N}_{\varepsilon,\mathfrak{S}}$ for $\mathbb{P}$ is one.

$$lim_{e\to\infty} P(x_e \in \mathfrak{N}_\varepsilon) = 1 \tag{5.33}$$

Proposition $\mathbb{P}^1$:

The QDDS algorithm satisfies **Hypothesis $\mathbb{H}^1$**.

Let us consider the solution update stage of the QDDS algorithm. If a new solution is generated such that its fitness is better than the ones recorded so far (global best), it replaces the best solution and is stored in memory.

$$x_{i,e+1} = £\,(x_{i,e}) \tag{5.34}$$

$$update\ (gbest, x_e) = \begin{cases} gbest, & fit(new) < fit(best) \\ x_e, & otherwise \end{cases} \tag{5.35}$$

This implies sequence $\{fit(gbest_e)\}_{e=1}^{\infty}$ is monotonically decreasing and $fit(£\,(x_e, gbest_e)) \leq fit(x_e)$. So $\mathbb{H}^1$ is satisfied.

Proposition $\mathbb{P}^2$:

The QDDS algorithm satisfies **Hypothesis $\mathbb{H}^2$**.

Recall that in equation (5.14) the even solutions to the double delta potential well setup take on the form given below.

$$\psi_{even}(r) = \begin{cases} \eta_2(1+e^{2ka})e^{-kr} & r > a \\ \eta_2(e^{-kr}+e^{kr}) & -a < r < a \\ \eta_2(1+e^{2ka})e^{kr} & r < -a \end{cases} \tag{5.36}$$

$$\lambda(r_{i,j,t}) = \psi^2{}_{even,i,j,t}(r) = \begin{cases} {\eta_2}^2\left(e^{-2kr} + e^{2k(2a-r)} + 2e^{4k(a-r)}\right) & r > a \\ {\eta_2}^2(e^{-2kr} + e^{2kr} + 2) & -a < r < a \\ {\eta_2}^2\left(e^{2kr} + e^{2k(2a+r)} + 2e^{4k(a+r)}\right) & r < -a \end{cases} \quad (5.37)$$

$\psi^2{}_{even,i,j,t}(r)$ is a measure of the probability density function of a particle in a particular dimension and integrating it across all dimensions yields the corresponding cumulative distribution function $\Lambda_{i,t}(Set)$:

$$\Lambda_{i,t}(Set) = \int_{Set}\left\{\prod_{j=1}^{d}\lambda\left(r_{i,j,t}\right)\right\} dr_{i,1,t}dr_{i,2,t}\dots dr_{i,D,t} \quad (5.38)$$

Observe that when $r \to \pm\infty$, the probability measure $\psi^2{}_{even}(r)$ goes to zero for $r \in (-\infty, -a) \cup (a, \infty)$ and is bounded for the region $-a < r < a$.

$$\lim_{r\to\pm\infty}\lambda\left(r_{i,j,t}\right) = 0 \quad (5.39)$$

$$\therefore 0 < \Lambda(Set) < 1 \quad (5.40)$$

$$\Lambda_t(Set) = \cup_{i=1}^{n}\Lambda_{i,t} \quad (5.41)$$

$$\therefore \prod_{t=0}^{\infty}\{1 - \Lambda_t(Set)\} = 0 \quad (5.42)$$

Thus, $\mathbb{H}^2$ is also satisfied. This in turn implies that theorem $\mathbb{T}^{\mathbf{1}}$, which is the global convergence algorithm for random search algorithms, is also satisfied and that £ is globally convergent.

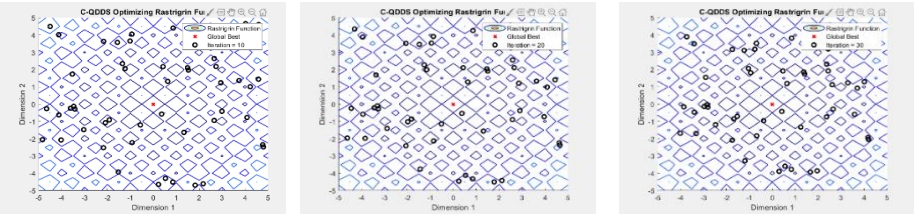
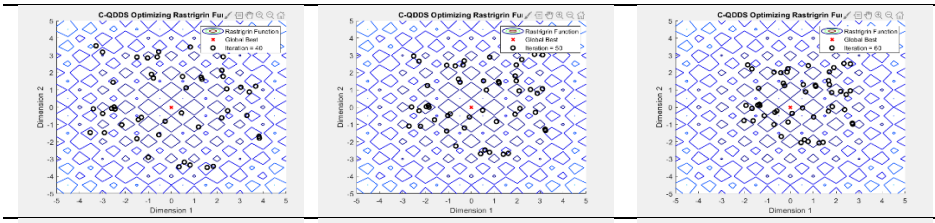
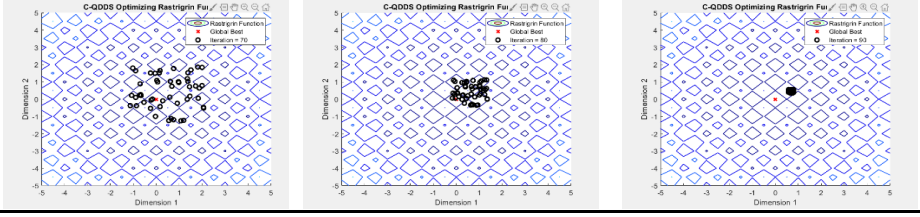
CHAPTER 6

VISUALIZATIONS OF SWARMING BEHAVIOR

6.1 Case Study using Simulations on the Rastrigrin Function (F9)

The C-QDDS approach is used on Rastrigrin Function using **50 particles** and **100 iterations** to demonstrate a visualization of how multiple agents evolve under the C-QDDS motion model. Fig. 6.1 shows that cost reduces as the swarm converges to promising regions.

Table 6.1 Visualization of Swarming Phenomenon over Rastrigrin Function (F9)

	**Iteration 10-30**
	**Iteration 40-60**
	**Iteration 70-90**

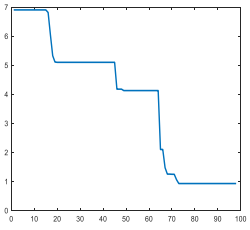
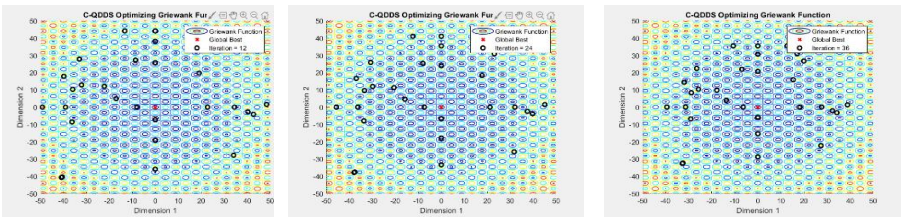
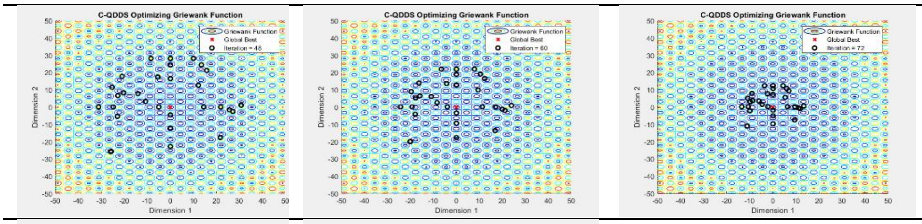
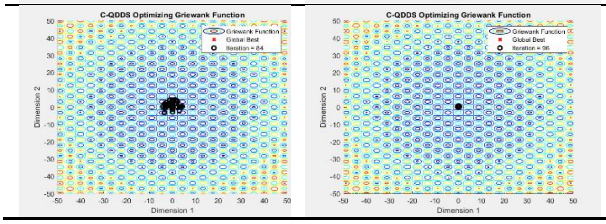


Figure 6.1. Cost v/s. Iterations over 100 iterations for Rastrigrin Function (F9)

6.2 Case Study using Simulations on the Griewank Function (F11)

The C-QDDS approach is used on the Griewank Function using **30 particles** and **100 iterations** to demonstrate a visualization of how multiple agents evolve under the C-QDDS motion model. Fig. 6.2 shows that cost reduces as the swarm converges to promising regions.

Table 6.2 Visualization of Swarming Phenomenon over Griewank Function (F11)

	**Iteration 12-36**
	**Iteration 48-72**
	**Iteration 84-96**

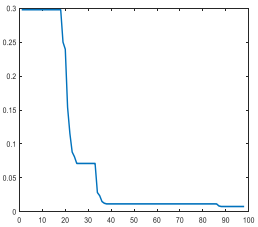
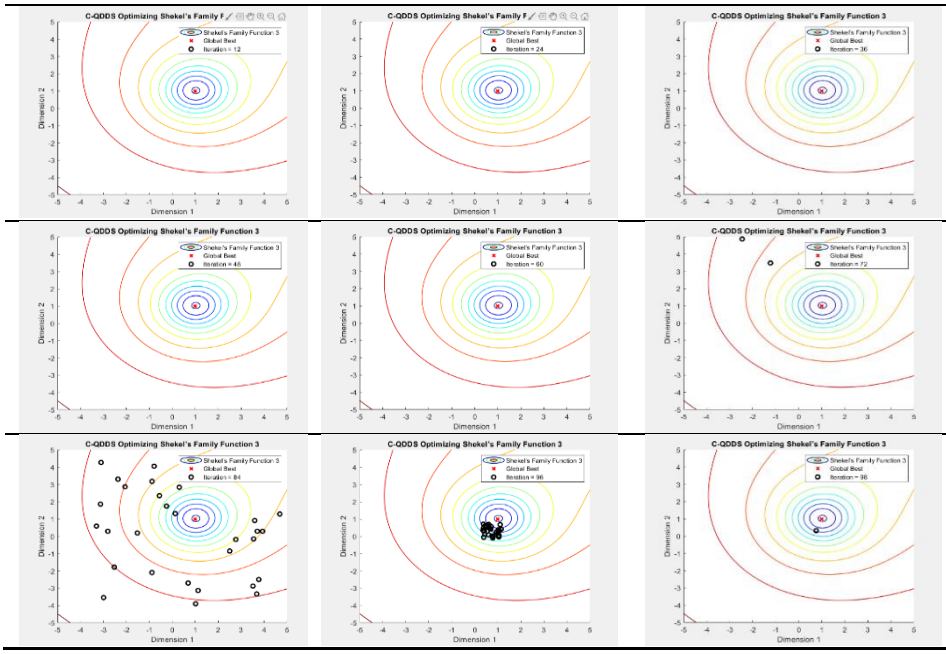


Figure 6.2. Cost v/s. Iterations over 100 iterations for Griewank Function (F11)

6.3 Case Study using Simulations on the Shekel's Family Function 3 (F23)

The C-QDDS approach is used on the Shekel's Family Function 3 **(F23)** using **30 particles** and **100 iterations** to demonstrate how multiple agents evolve under the C-QDDS motion model. Fig. 6.2 shows that cost reduces as the swarm converges to promising regions.

Table 6.3 Visualization of Swarming Phenomenon over Shekel's Family Function 3 (F23)

	**Iteration 12-36**
	Iteration 48-72
	Iteration 84-96

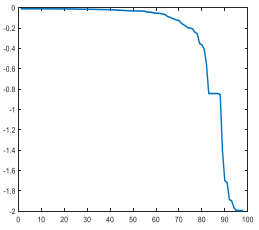


Figure 6.3.Cost v/s. Iterations over 100 iterations for Shekel's Family Function 3(F23)

CHAPTER 7

PROOF OF CONCEPT: APPLICATIONS

7.1 High Dimensional Finite Impulse Response (FIR) Filter Design

This subsection outlines the design procedure of a multidimensional low-pass Finite Impulse Response (FIR) filter as proposed in [224] [225]. The ideal filter response $H_d(e^{j\omega})$ and system transfer function $H(z)$ governing the design of the filter are given by Eqs. (7.1) and (7.2) respectively:

$$H_d(e^{j\omega}) = 1 \qquad 0 \leq \omega \leq \omega_p$$
$$= 0 \qquad \omega_s \leq \omega \leq \pi \tag{7.1}$$

$$H(z) = \sum_{n=0}^{N} h(n) z^{-n} \qquad n = 0,1,\ldots.N \tag{7.2}$$

Here *h(n)* denotes the filter's impulse response and *N* is order of the filter having *N+1* coefficients. The normalized passband and stopband edge frequencies are ω_p and ω_s respectively and E_p and E_s are errors in passband and stopband given by Eqs. (7.3) and (7.4).

$$E_p = \frac{1}{\pi}\int_0^{\omega_p} (1 - H(\omega))^2 d\omega \tag{7.3}$$

$$E_s = \frac{1}{\pi}\int_{\omega_s}^{\pi} H(\omega)^2 d\omega \tag{7.4}$$

A cost function γ of choice used in [224] is chosen for the minimization objective:

$$\gamma = \eta E_p + (1 - \eta) E_s \qquad 0 \leq \eta \leq 1 \tag{7.5}$$

The main focus of the optimization routine is to find a balanced response that seeks to minimize the cost function γ, thereby minimizing the passband and stopband errors. The trade-off between reducing E_p and E_s is controlled by selection of a weight $\eta \in [0,1]$ as per the user's design condition.

7.1.1 Simulation Results for the FIR Filter Design Problem

The following subsection illustrates the design of a multidimensional low-pass Finite Impulse Response (FIR) filter using a population size of 1000 and an iteration count of 250 and 500 for filter orders 10 and 20 respectively. The passband and stopband edges are set at 0.3π and 0.6π.

Table 7.1 Simulations for 10-Dimensional FIR Filter Design using 10 Independent Trials of QDDS

Δ (dB)	Mean Cost	St. Dev	Best Cost	Worst Cost
-13.6466	1.4817e-05	1.6762e-05	5.7632e-08	4.5367e-05

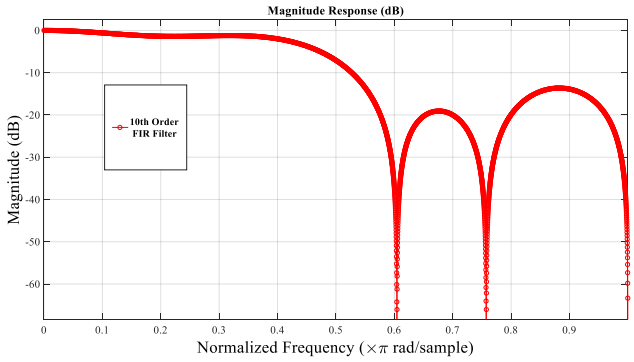


Figure 7.1 Response of the 10-Dimensional FIR Filter

Table 7.2 Best 10-Dimensional Filter Coefficients (10 Trials)

Filter Coefficients	h(1) = h(10)	0.070824792496751651
	h(2) = h(9)	-0.063184376757871669
	h(3) = h(8)	-0.038806613903081974
	h(4) = h(7)	0.013227497402604124
	h(5) = h(6)	0.39889122413816075

Table 7.3 Simulations for 20-Dimensional FIR Filter Design using 10 Independent Trials of QDDS

Δ (dB)	Mean Cost	St. Dev	Best Cost	Worst Cost
-17.7398	7.1306e-05	9.9875e-05	1.6055e-06	3.1458e-04

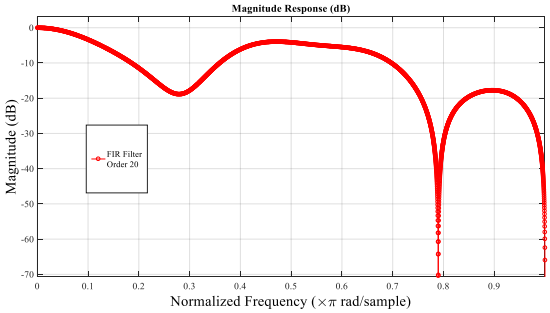


Figure 7.2 Response of the 20-Dimensional FIR Filter

Table 7.4 Best 20-Dimensional Filter Coefficients (10 Trials)

Filter Coefficients	h(1) = h(20)	0.011566963779404912
	h(2) = h(19)	0.0077331878563942523
	h(3) = h(18)	-0.0094736298940968737
	h(4) = h(17)	-0.0068424142182682956
	h(5) = h(16)	0.024047530227972496
	h(6) = h(15)	0.04099248691610477
	h(7) = h(14)	0.14983102243854188
	h(8) = h(13)	0.0057626071427242216
	h(9) = h(12)	-0.0038505536917844913
	h(10) = h(11)	0.28023279944300716

Tables 7.1 and 7.3 list the maximum stopband attenuation and the mean, standard deviation, best and worst cost values when using QDDS for 10 trials. Tables 7.2 and 7.4 report the set of best filter coefficients for orders 10 and 20 whereas Figures 7.1 and 7.2 plot the corresponding filter responses. The FIR filter responses record a maximum stopband attenuation of -13.6466 dB and -17.7398 dB on dimensions 10 and 20 for a QDDS implementation using gradient descent.

7.2 Automatic Generation of Neural Network Architectures

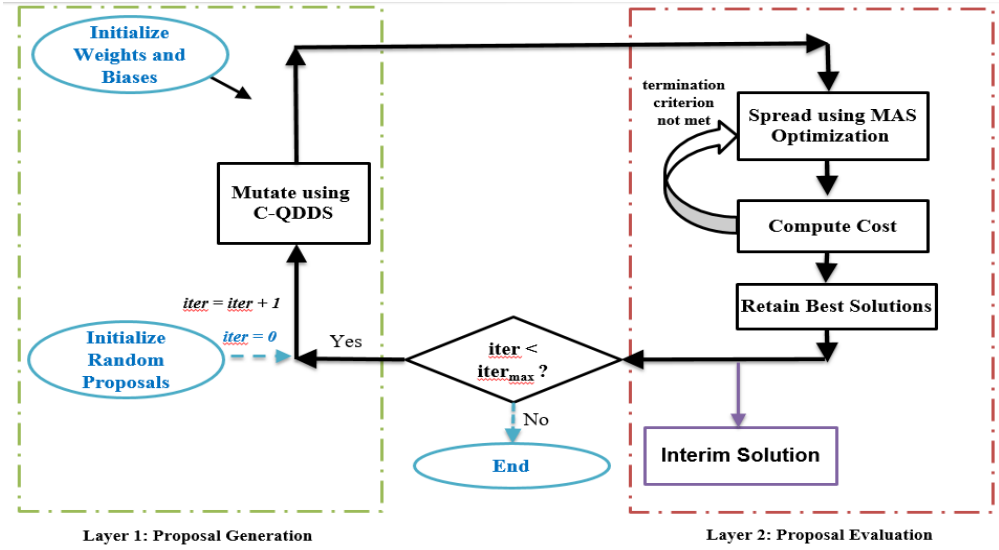


Figure 7.3 A Bi-layered C-QDDS Based NN Architecture Generation Mechanism

7.2.1 Description of the Architecture Setup:

Consider the schematic in Figure 7.3 where layer 1 corresponds to the proposal generation scheme and layer 2 corresponds to the neural network weights and biases optimization phase. In layer 1, the C-QDDS algorithm takes as input the number of desired layers of the network, the minimum and maximum number of units permissible in each layer, the type of chaotic map used in the generation process, number of particles (proposals) used and the number of iterations. Layer 1 of the C-QDDS architecture then generates a number of proposals of network architectures and mutates it using the C-QDDS mechanism and passes it on to Layer 2 where its corresponding weights and biases are optimized using a multi-agent (swarm-based) approach. This process in repeated over a number of iterations with an underlying bias towards the global best architecture of a particular iteration in the successive ones. The global best is identified by the C-QDDS layer by computing a MSE/accuracy/similarity score among the pool of proposals in the ongoing iteration. It is

critically important to generate an initial proposal which spans a practical range of layer unit proposals given a minimum and maximum number of units in each layer. From a pure optimization standpoint, the lower the error in optimizing the weights and biases the better the performance of the optimization algorithm is. In other words, overfitting the training data implies efficiency of the underlying optimizer. This is different from a traditional machine learning outlook, where there needs to be a judicious cut-off in lowering training error beyond a certain extent, since the testing error and in turn the generalization error increases. The expectation is that, compared to a baseline without the C-QDDS architecture mutation in layer 1, the evolution will lead to a final solution that has a higher accuracy/lower absolute error in the training process.

7.2.2 Illustration 1:

In this illustration, a 3-layer (hidden) deep multi-layer-perceptron [228] is generated using a linearly decreasing learning rate (tapers linearly from 0.9 to 0.3 over course of iterations) using a Chebyshev map on the IRIS classification dataset [264]. After computation completes for 200 iterations, the best MSE value obtained is 0.0269, the best classification rate obtained is 99.33% and based on these results the optimum architecture recommended is [10;6;10] that is 10 units in hidden layer 1, 6 units in hidden layer 2 and 10 units in hidden layer 3 given the input constraint of a minimum of 5 units and a maximum of 15 units per layer. The optimum recommendation of the network architecture would also depend of the sample proposal generations in every C-QDDS layer iteration. Table 7.5 lists the hyperparameters used and Table 7.6 shows the simulation results.

Table 7.5 Hyperparameter Choices for Illustration 1

Hyperparameter	Value
Number of Particles in MAS Optimization	50
Maximum Number of Iterations	200
Inertia Weight	2
Number of Particles in C-QDDS (Proposals)	50
Minimum Number of Nodes in Any Layer	5
Maximum Number of Nodes in Any Layer	15
Number of Iterations in C-QDDS Phase	50
Type of Chaotic Map Used	Chebyshev
Number of Layers in the Neural Network Architecture	3
Learning Rate in C-QDDS Phase	Linearly decreasing from 0.9 to 0.3 (Exploration-Exploitation)

Table 7.6 Performance Metrics for Testing Illustration 1

Performance Metric	Value
Global Best Mean Squared Error	0.0269
Global Best Classification Rate	99.33%
Architecture Proposed as Optimum subj. to Constraints	[10;6;10]

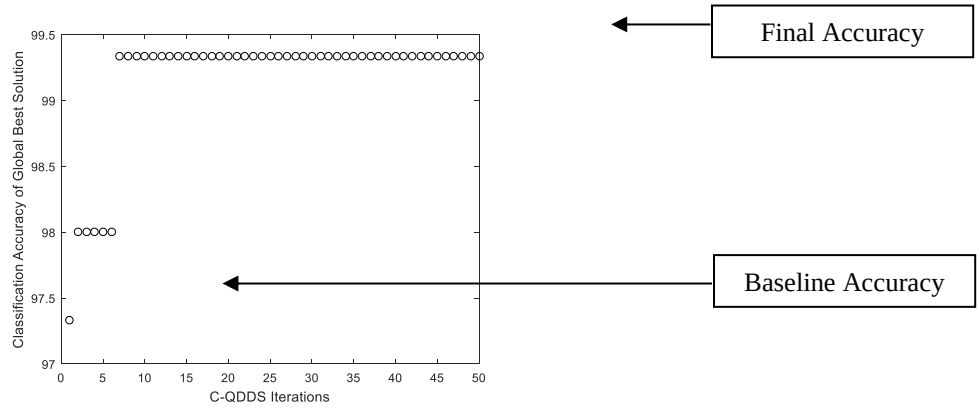


Figure 7.4 Evolution of Classification Rate over C-QDDS Iterations for Adaptive Learning Rate

7.2.3 Illustration 2:

In this illustration, a 3-layer (hidden) deep multi-layer-perceptron is generated using a constant exploratory learning rate (fixed at a high value of 0.9) using a Chebyshev map on the IRIS classification dataset [264]. After computation completes for 200 iterations,

the best MSE value obtained is 0.0527, the best classification rate obtained is 98.6667% and based on these results the optimum architecture recommended is [11;9;7] that is 11 units in layer 1, 9 units in layer 2 and 7 units in layer 3 given the input constraint of a minimum of 5 units and a maximum of 15 units per layer. The optimum recommendation of the network architecture in this case too, would depend on the sample proposal generations in every C-QDDS layer iteration. Table 7.7 lists the hyperparameters used and Table 7.8 shows the simulation results.

It can be noted by comparing Fig. 7.4 and 7.5 that the initial exploration and subsequent exploration facilitated by the linearly decreasing learning rate in Illustration 1 results in a marginally better classification rate of 99.3333% over 98.6667%.

Table 7.7 Hyperparameter Choices for Illustration 2

Hyperparameter	Value
Number of Particles in MAS Optimization	50
Maximum Number of Iterations	200
Inertia Weight	2
Number of Particles in C-QDDS (Proposals)	50
Minimum Number of Nodes in Any Layer	5
Maximum Number of Nodes in Any Layer	15
Number of Iterations in C-QDDS Phase	50
Type of Chaotic Map Used	Chebyshev
Number of Layers in the Neural Network Architecture	3
Learning Rate in C-QDDS Phase	0.9 (Exploratory)

Table 7.8 Performance Metrics for Testing Illustration 2

Performance Metric	Value
Global Best Mean Squared Error	0.0527
Global Best Classification Rate	98.6667%
Architecture Proposed as Optimum subj. to Constraints	[11;9;7]

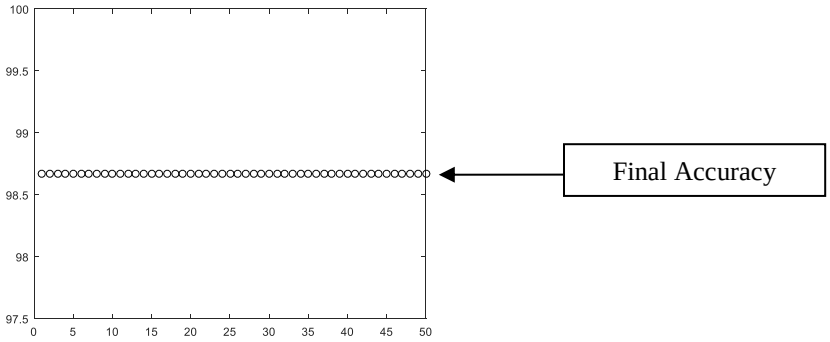


Figure 7.5 Evolution of Classification Rate over C-QDDS Iterations for Constant Learning Rate

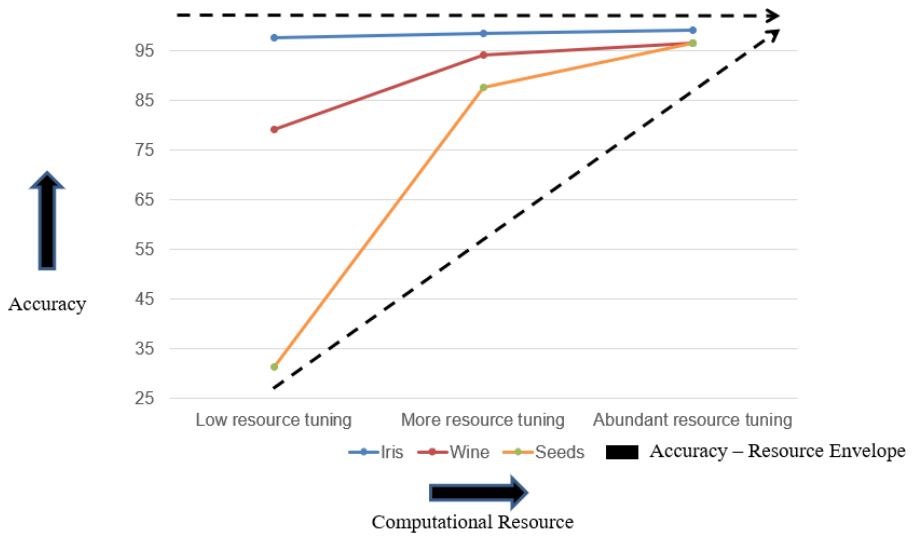


Figure 7.6 Accuracy v/s. Computational Resource Tuning

Figure 7.6 shows classification accuracy as a function of computational resource tuning for the IRIS, WINE and SEEDS classification datasets from the UCI Machine Learning Repository [242]. It is observed that classification accuracy improves steadily as the hyperparameters in layer 1 (C-QDDS driven architecture mutation) and layer 2 (gradient-free training of weights and biases) are fine tuned to fit the problem at hand. For example, it is possible to overfit a data distribution with abundant resource tuning in the form of number of layers of hidden nodes, time-varying learning rates, number of nodes in each hidden layer,

number of particles in the candidate architecture generation stage, number of particles involved in the training process, computation budget etc. Using a 3-layer deep network, the accuracy on the WINE data is 94.3820% whereas a 6-layer deep network results in a 96.6292% accuracy. A 6-layer deep network on the SEEDS data results in a 96.6667% accuracy. The trend noticed is an experimental validation of the No Free Lunch theorems in learning.

CHAPTER 8

CONCLUSIONS

8.1 Future Research Directions in Particle Swarm Intelligence

Two decades of exciting developments in the particle swarm paradigm has seen many exciting upheavals and successes alike. The task of detecting an essential infimum/supremum among the presence of many local optima, the arbitrary nature of the search space and the intractability of using conventional mathematical abstractions on a wide range of objective functions coupled with little or no guarantees apriori about any optima being found make the search process a challenging undertaking. However, Particle Swarm Optimizers have had their fair share of success stories – they can be used on any objective function: continuous or discontinuous, tractable or intractable, even for those where initialization renders solution quality to be sensitive as evidenced in case of their deterministic counterparts. However, some pressing issues which are listed below merit further work by the PSO community.

Parameter Sensitivity: Solution quality of metaheuristics like PSO are sensitive to their parametric evolutions. This means that the same strategy of parameter selection does not work for every problem.

Convergence to local optima: Unless the basic PSO is substantially modified to take into account the modalities of the objective function, more often than not it falls prey to local optima in the search space for sufficiently complex objective functions.

Subpar performance in multi-objective optimization for high dimensional problems: Although niching techniques render acceptable solutions for multimodal functions in both static and dynamic environments, the solution quality falls sharply when the dimensionality of the problem increases.

Ensemble optimizers, although promising, do not address the underlying shortcomings of the basic PSO. Theoretical issues such as the particle explosion problem, loss of particle diversity as well as stagnation to local optima deserve the attention of researchers so that a unified algorithmic framework with more intelligent self-adaptation and less user-specified customizations can be realized for future applications.

8.2 Analysis of the QDDS Mechanism

The QDDS algorithm is based on a quantum double Dirac delta potential well model and is an extension of the conventional QPSO (Type I and Type II) which are modeled after a singular Dirac delta potential well. The intuitively simple iterative updates of QDDS lead the swarm towards fitter regions of the search space in conjunction with reaching for regions of lower energy for a particle under influence of a spatially co-located attractive double delta potential. The current form, however is prone to delivering suboptimal results because of the use of a gradient descent scheme in the $\boldsymbol{\delta}$ update phase. In addition to this, time complexity of the algorithm is markedly high due to the computationally heavy numerical approximation of r_{iter} from δ_{iter}. The effects of using different social/cognition attractors as well as multi-scale particle topologies remain to be investigated and a thorough characterization of initialization schemes versus numerical accuracy is a logical follow-up. Overall, despite high computational overhead QDDS produces acceptable solutions for some problems (Tables 4.2-4.3, 4.5) and not for some others, as seen in Table 4.4. However, the fact QDDS is based

on a legitimate optimization phenomenon in quantum physics and that it produces quality solutions on classical benchmarks warrants resource expenditure in exploring better how it works.

This work derives a new swarming mechanism inspired from quantum mechanical considerations in a double delta potential well and explores its inner workings. The mechanism (QDDS) is formally derived and a computable form is put forward followed by experiments to determine its accuracy in finding global optima. This is achieved by approaching some classic benchmark functions such as Rosenbrock, Rastrigrin, Sphere and Griewank. Experiments provide insight into the performance of the approach on Rosenbrock, Rastrigrin and Griewank functions on which it appears to perform better, while not performing as well in the Sphere function. While the present version of the algorithm is computationally expensive and stagnates occasionally, there is room for improvement in both areas. Future studies would aim at addressing these issues in addition to performing statistical significance tests to quantify the performance of the approach over an extensive suite of separable and non-separable functions of much higher dimensionality. However, QDDS captures a relatively unexplored approach in blending swarm intelligence and optimization and extends the literature on quantum-inspired computational intelligence.

8.3 Analysis of the C-QDDS Mechanism

C-QDDS is an extension of QDDS in a double Dirac delta potential well setup and uses a Chebyshev map driven solution update. The evolutionary behavior of QDDS is simple to follow from an intuitive point of view and guides the particle set towards lower energy configurations under the influence of a spatially co-located attractive double delta potential. The current formulation is susceptible to getting trapped in suboptimal results because of the

use of a gradient descent scheme in the δ_t computation phase. The algorithm is expensive in terms of time complexity because of a numerical approximation of r_t from δ_t in the transcendental equation involved. As outlined in [229], the impact of cognition and social attractors, initial tessellation configurations, multiscale topological communication schemes and correction (update) processes need to be studied to provide more insightful comments into the optimization of the workflow itself, specifically the stagnation issue and the high time complexity. In summary, the use of additional chaotic sequences in the heuristic evolution of QDDS based on this commonly used approximation abstraction from quantum physics remains to be further explored. Further, the snowball effect on the dynamics due to the selection of varying number of agents and selective communication among them over a user-defined number of generations is a thrust area gaining prominence [255][256]. As we continue to further our understanding of how emergent properties arise out of simple, local-level interactions at the lowest hierarchical levels, we may expect the evolutionary computation community to increasingly consider scale-free interactions among atomic agents on top of the existing, already rich body of research on biomimicry. The proposed paradigm is well-suited for application in single-objective unimodal/multimodal optimization problems such as those discussed in [236][225][224][257][260] along the lines of digital filtering, fuzzy-clustering, scheduling, routing, feature engineering etc. and those illustrated in Chapter 5. The QDDS and, subsequently, C-QDDS approaches build on a growing corpus of algorithms hybridizing quantum swarm intelligence and global optimization and adds to the existing collection of nature-inspired optimization techniques.

BIBLIOGRAPHY

[1] Kennedy, J., Eberhart, R., “Particle swarm optimization”, Proc. IEEE Int. Conf. Neural Network, 1995.

[2] Holland, J.H., “Adaptation in Natural and Artificial Systems”, University of Michigan Press, 1975.

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

[4] Sun, J., Feng, B., Xu, W.B., “Particle swarm optimization with particles having quantum behavior”, IEEE Proceedings of Congress on Evolutionary Computation, pp. 325–331, 2004.

[5] Sun, J., Xu,W.B., Feng, B., “A global search strategy of quantum-behaved particle swarm optimization.”, Cybernetics and Intelligent Systems Proceedings of the 2004 IEEE Conference, pp. 111–116, 2004.

[6] Reeves, W.T., “Particle systems – a technique for modelling a class of fuzzy objects”, ACM Trans. Graphics 2(2):91-108.

[7] Reynolds, C.W., “Flocks, herds, and schools: a distributed behavioral model”, Computer Graphics, 21(4): 25-34 (Proc SIGGRAPH ‘87).

[8] Y. Shi, R.C. Eberhart, “Parameter selection in particle swarm optimization”, Proceedings of 7th International Conference on Computation Programming VII, 1998.

[9] Y. Shi, R. Eberhart, “A modified particle swarm optimizer”, IEEE, 1998, pp. 69–73.

[10] Eberhart, R.C., Shi, Y., “Particle Swarm Optimization: Developments, Applications and Resources”, Proceedings of IEEE Congress on Evolutionary Computation, Volume 1, IEEE Press, May 2001, pp. 27-30.

[11] Suganthan, P.N., “Particle Swarm Optimiser with Neighborhood Operator”, Proceedings of the IEEE Congress on Evolutionary Computation, IEEE, 1999, pp. 1958-1962.

[12] Ratnaweera, A., Halgamuge, S., Watson, H., “Particle Swarm Optimization with Self-Adaptive

Acceleration Coefficients", Proceedings of the First International Conference on Fuzzy Systems and Knowledge Discovery, 2003, pp. 264-268.

[13] Zheng, Y., Ma, L., Zhang, L., Qian, J., "On the Convergence Analysis and Parameter Selection in Particle Swarm Optimization", Proceedings of the International Conference on Machine Learning and Cybernetics, vol. 3, pp. 1802-1807, Nov. 2003.

[14] Zheng, Y., Ma, L., Zhang, L., Qian, J., "Empirical Study of Particle Swarm Optimizer with Increasing Inertia Weight", Proceedings of the IEEE Congress on Evolutionary Computation, pp. 221-226, IEEE, 2003.

[15] Naka, S., Genji, T., Yura, T., Fukuyama, Y., "Practical Distribution State Estimation using Hybrid Particle Swarm Optimization", IEEE Power Engineering Society Winter Meeting, vol. 2, pp. 815-820, Jan. 2001.

[16] Clerc, M., "Think Locally, Act Locally: The Way of Life of Cheap-PSO, an Adaptive PSO", Technical Report, http://clerc.maurice.free.fr/pso/, 2001.

[17] Shi, Y., Eberhart, R.C., "Fuzzy Adaptive Particle Swarm Optimization", Proceedings of the IEEE Congress on Evolutionary Computation, vol. 1, pp. 101-106, IEEE, May 2001.

[18] Eberhart, R.C., Simpson, P.K., Dobbins, R.W., "Computational Intelligence PC Tools", Academic Press Professional, First Edition, 1996.

[19] Clerc, M., Kennedy, J., "The Particle Swarm-Explosion, Stability and Convergence in a Multidimensional Complex Space", IEEE Transactions on Evolutionary Computation, 6(1):pp. 58-73, 2002.

[20] Clerc, M., "The Swarm and the Queen: Towards a Deterministic and Adaptive Particle Swarm Optimization", Proceedings of the IEEE Congress on Evolutionary Computation, vol. 3, pp. 1951-1957, Jul. 1999.

[21] Eberhart, R.C., Shi, Y., "Comparing Inertia Weights and Constriction Factors in Particle Swarm Optimization", Proceedings of the IEEE Congress on Evolutionary Computation, vol. 1, pp. 84-88, Jul. 2000.

[22] Kennedy, J., “The Particle Swarm: Social Adaptation of Knowledge”, Proceedings of the IEEE International Conference on Evolutionary Computation, pp. 303-308, Apr. 1997.

[23] Carlisle, A., Dozier, G., “Adapting Particle Swarm Optimization to Dynamic Environments”, Proceedings of the International Conference on Artificial Intelligence, pp. 429-434, 2000.

[24] Stacey, A., Jancic, M., Grundy, I., “Particle Swarm Optimization with Mutation”, Cong. Evolutionary Computation, CEC 2003, IEEE, Dec 2003, pp. 1425-1430.

[25] Jie, X., Deyun, X., “New Metropolis Coefficients of Particle Swarm Optimization”, 2008 Chinese Control and Decision Conference, Yantai, Shandong, 2008, pp. 3518-3521.

[26] Kirkpatrick, S., Gelatt, C., Vecci, M., “Optimization by Simulated Annealing”, Science 220:671-680, 1983.

[27] Ratnaweera, A., Halgamuge, S., Watson, H., “Particle Swarm Optimization with Time Varying Acceleration Coefficients”, Proceedings of the International Conference on Soft Computing and Intelligent Systems, pp. 240-255, 2002.

[28] Kennedy, J., Mendes, R., "Population structure and particle swarm performance" , Proceedings of the 2002 Congress on Evolutionary Computation, CEC'02, 2002.

[29] Kennedy, J., “Small Worlds and Mega-Minds: Effects of Neighbourhood Topology on Particle Swarm Performance”, Proceedings of the IEEE Congress on Evolutionary Computation, vol. 3, pp. 1931-1938, Jul. 1999.

[30] Kennedy, J., Mendes, R., “Population Structure and Particle Swarm”, Proceedings of the IEEE Congress on Evolutionary Computation, pp. 1671-1676, IEEE, 2002.

[31] Mendes, R., Kennedy, J., Neves, J., “Watch thy Neighbour or How the Swarm can Learn from its Environment”, Proceedings of the IEEE Swarm Intelligence Symposium, pp. 88-94, Apr. 2003.

[32] Liu, Q., Wei, W., Yuan, H., Zhan, Z.H., Li, Y., “Topology selection for particle swarm optimization”, Inf. Sci. 363 (2016) 154–173.

[33] van den Bergh, F., “An Analysis of Particle Swarm Optimizers”, PhD Thesis, Department of

Computer Science, University of Pretoria, Pretoria, South Africa, 2002.

[34] van den Bergh, F., Engelbrecht, A.P., “A Study of Particle Swarm Optimization Particle Trajectories”, Information Sciences, 2005.

[35] Trelea, L.C., “The Particle Swarm Optimization Algorithm: Convergence Analysis and Parameter Selection”, Information Processing Letters, 85(6):317-325, 2003.

[36] Robinson, J., Sinton, S., Rahmat-Samii, Y.,“Particle Swarm, Genetic Algorithm, and Their Hybrids: Optimization of a Profiled Corrugated Horn Antenna”, Proceedings of the IEEE Antennas and Propagation Society International Symposium and URSI National Radio Science Meeting, vol. 1, pp. 314-317, June 2002.

[37] Shi, X., Lu, Y., Zhou, C., Lee, H., Lin, W., Liang, Y., “Hybrid Evolutionary Algorithms Based on PSO and GA”, Proceedings of the IEEE Congress on Evolutionary Computation, vol. 4, pp. 2393-2399, Dec. 2003.

[38] Yang, B., Chen, Y., Zhao, Z., “A hybrid evolutionary algorithm by combination of PSO and GA for unconstrained and constrained optimization problems”, Proceedings of the IEEE International Conference on Control and Automation, 2007, pp. 166–170.

[39] Li, T., Xu, L., Shi, X.W., “A hybrid of genetic algorithm and particle swarm optimization for antenna design”, PIERS online 4 (2008) 56–60.

[40] Valdez, F., Melin, P., Castillo, O., “Evolutionary method combining particle swarm optimization and genetic algorithms using fuzzy logic for decision making”, Proceedings of the IEEE International Conference on Fuzzy Systems, 2009, pp. 2114–2119.

[41] Krink, T., Løvbjerg, M., “The lifecycle model: combining particle swarm optimization, genetic algorithms and hill climbers,” Proceedings of the Parallel Problem Solving From Nature (2002) 621–630.

[42] Conradie, E., Miikkulainen, R., “C. Aldrich, Intelligent process control utilising symbiotic memetic neuro-evolution”, Proceedings of the IEEE Congress on Evolutionary Computation 1 (2002) 623–628.

[43] Grimaldi, E.A., Grimacia, F., Mussetta, M., Pirinoli, P., Zich, R.E., "A new hybrid genetical – swarm algorithm for electromagnetic optimization", Proceedings of International Conference on Computational Electromagnetics and its Applications, Beijing, China, 2004, pp. 157–160.

[44] Juang, C – F. "A hybrid of genetic algorithm and particle swarm optimization for recurrent network design", IEEE Transactions On Systems, Man, And Cybernetics-Part B: Cybernetics 34 (2004) 997–1006.

[45] Settles, M., Soule, T., "Breeding swarms: a GA/PSO hybrid, Proceedings of Genetic and Evolutionary Computation Conference (2005)", 161–168.

[46] Jian, M., Chen, Y., "Introducing recombination with dynamic linkage discovery to particle swarm optimization", Proceedings of the Genetic and Evolutionary Computation Conference (2006) 85–86.

[47] Esmin, A.A., Lambert-Torres, G., Alvarenga, G.B., "Hybrid evolutionary algorithm based on PSO and GA mutation, Proceedings of 6th International Conference on Hybrid Intelligent Systems", 2006, pp. 57–62.

[48] Kim, H., "Improvement of genetic algorithm using PSO and Euclidean data distance, International Journal of Information Technology", 12 (2006) 142–148.

[49] Mohammadi, A., Jazaeri, M., "A hybrid particle swarm optimization-genetic algorithm for optimal location of SVC devices in power system planning", Proceedings of 42nd International Universities Power Engineering Conference, 2007, pp.1175–1181.

[50] Gandelli, A., Grimaccia, F., Mussetta, M., Pirinoli, P., Zich, R.E., "Development and Validation of Different Hybridization Strategies between GA and PSO", Proc. of the 2007 IEEE Congress on Evolutionary Computation, Sept. 2007, Singapore, pp. 2782–2787.

[51] Kao, Y.T., Zahara, E., "A hybrid genetic algorithm and particle swarm optimization for multimodal functions", Applied Soft Computing 8 (2008) 849–857.

[52] Premalatha, K., Natarajan, A.M., "Discrete PSO with GA operators for document clustering", International Journal of Recent Trends in Engineering 1 (2009) 20–24.

[53] Abdel-Kader, R.F., “Genetically improved PSO algorithm for efficient data clustering”, Proceedings of International Conference on Machine Learning and Computing, 2010, pp. 71–75.

[54] Kuo, R. J., Hong, C. W., “Integration of genetic algorithm and particle swarm optimization for investment portfolio optimization,” Applied Mathematics and Information Sciences, vol. 7, no. 6, pp. 2397–2408, 2013.

[55] Ghamisi, P., Benediktsson, J. A., “Feature selection based on hybridization of genetic algorithm and particle swarm optimization,” IEEE Geoscience and Remote Sensing Letters, vol. 12, no. 2, pp. 309–313, 2015.

[56] A. Benvidi, S. Abbasi, S. Gharaghani, M.D. Tezerjani, S. Masoum, Spectrophotometric determination of synthetic colorants using PSO-GA-ANN, Food Chem. 220 (2017) 377–384.

[57] Yu, S., Wei, Y-M., Wang, K. A., “PSO–GA optimal model to estimate primary energy demand of China”. Energy Policy 2012;42:329–40.

[58] Moussa, R., Azar, D., “A PSO-GA approach targeting fault-prone software modules”, The Journal of Systems and Software, 132 (2017) 41-49.

[59] Nik, A. A., Nejad, F. M., Zakeri, H., “Hybrid PSO and GA approach for optimizing surveyed asphalt pavement inspection units in massive network”, Automation in Construction 71 (2016) 325-345.

[60] Garg, H., “A hybrid PSO-GA algorithm for constrained optimization problems”, Applied Mathematics and Computation, Elsevier. 2016;274:292-305.

[61] Zhang, Q., Ogren, R.M., Kong, S.C., “A comparative study of biodiesel engine performance optimization using enhanced hybrid PSO–GA and basic GA”, Appl Energy 2016, 165, 676–84.

[62] Li, C., Zhai, R., Liu, H., Yang, Y., Wu, H., “Optimization of a heliostat field layout using hybrid PSO-GA algorithm”, Applied Thermal Engineering 128 (2018), 33-41.

[63] Price, K., Storn, R., “Differential evolution – a simple and efficient adaptive scheme for global

optimization over continuous spaces", Technical Report, International Computer Science Institute, Berkeley, 1995.

[64] Hendtlass, T., "A Combined Swarm differential evolution algorithm for optimization problems", Proceedings of 14th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Lecture Notes in Computer Science, vol. 2070, Springer Verlag, 2001, pp. 11–18.

[65] Zhang, W.J., Xie, X.F., "DEPSO: hybrid particle swarm with differential evolution operator", Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (SMCC), Washington DC, USA, 2003, pp. 3816–3821.

[66] Talbi, H., Batouche, M., "Hybrid particle swarm with differential evolution for multimodal image registration", Proceedings of the IEEE International Conference on Industrial Technology 3 (2004) 1567–1573.

[67] Hao, Z. -F., Gua, G. -H., Huang, H., "A particle swarm optimization algorithm with differential evolution", Proceedings of Sixth International Conference on Machine Learning and Cybernetics, 2007, pp. 1031–1035.

[68] Das, S., Abraham, A., Konar, A., "Particle swarm optimization and differential evolution algorithms: technical analysis, applications and hybridization perspectives", Ying Liu et al. (Eds.), Advances of Computational Intelligence in Industrial Systems, Studies in Computational Intelligence, Springer Verlag, Germany, 2008, pp. 1–38.

[69] Luitel, B., Venayagamoorthy, G. K., "Differential evolution particle swarm optimization for digital filter design," in Proc. IEEE Cong. Evol. Comput., Hong Kong, China, Jun. 2008, pp. 3954–3961.

[70] Vaisakh, K. et al., "Differential evolution particle swarm optimization algorithm for reduction of network loss and voltage instability", IEEE World Congress on Nature and Biologically Inspired Computing, pp. 391-396, Dec. 2009.

[71] Huang, H., Wei, Z. H., Li, Z. Q., Rao, W. B., "The back analysis of mechanics parameters

based on DEPSO algorithm and parallel FEM," in Proc. Int. Conf. Comput. Intell. Natur. Comput., Wuhan, China, Jun. 2009, pp. 81–84.

[72] James G. Malone, "Automated Mesh Decomposition and Concurrent Finite Element Analysis for Hypercube Multiprocessor Computers", Computer Methods in Applied Mechanics and Engineering, 70, 1988, pp. 27-58.

[73] Charbel Farhat, "implementation Aspects of concurrent finite element computations in Parallel Computations and their Impact on computational mechanics", ASME New York, 1987.

[74] Rehak, D.R., Baugh, J.W., "Alternative Programming Techniques for Finite Element Program Development", Proc. IABSE Colloquium on Expert Systems in Civil Engineering, Bergamo, Italy, 1989.

[75] Logozzo, F., "Modular static analysis of object-oriented Languages", PhD thesis, Ecole Polytechnique, June 2004.

[76] Xu, R., Xu, J., Wunsch II, D. C., "Clustering with differential evolution particle swarm optimization," in Proc. IEEE Cong. Evol. Comput., Barcelona, Spain, Jul. 2010, pp. 1–8.

[77] Xiao, L., Zuo, X., "Multi-DEPSO: a DE and PSO Based Hybrid Algorithm in Dynamic Environments", WCCI 2012 IEEE World Congress on Computational Intelligence June, 10-15, 2012- Brisbane, Australia.

[78] Junfei, H., Liling, M.A., Yuandong, Y.U., "Hybrid Algorithm Based Mobile Robot Localization Using DE and PSO", Proceedings of 32nd International Conference on Control and Automation, Xi'an, China, July 26-28, 2013:5955–5959.

[79] Sahu, B. K., Pati, S., Panda, S., "Hybrid differential evolution particle swarm optimisation optimised fuzzy proportional– integral derivative controller for automatic generation control of interconnected power system", IET Generation, Transmission & Distribution 8, no. 11 (2014): 1789-1800.

[80] Seyedmahmoudian, M. et al., "Simulation and hardware implementation of new maximum power point tracking technique for partially shaded PV system using hybrid DEPSO method",

IEEE Trans Sustain energy 2015; 6:850-62.

[81] Gomes, P. V., Saraiva, J. T., “Hybrid Discrete Evolutionary PSO for AC Dynamic Transmission Expansion Planning,” in IEEE Energycon conference, 2016.

[82] Boonserm, P., Sitjongsataporn, S., "A robust and efficient algorithm for numerical optimization problem: DEPSO-Scout: A new hybrid algorithm based on DEPSO and ABC," 2017 International Electrical Engineering Congress, Pattaya, Thailand, 2017, pp. 1-4.

[83] Zhao, F.Q. et al., “A hybrid algorithm based on PSO and simulated annealing and its applications for partner selection in virtual enterprises”, Advances in Intelligent Computing, PT 1. Proc. Lect. Notes Comput. Sci. 2005;3644:380–5.

[84] G. Yang, D. Chen, and G. Zhou, “A new hybrid algorithm of particle swarm optimization,” in Lecture Notes in Computer Science, vol. 4115, pp. 50–60, 2006.

[85] Gao, H. et al., “Training RBF neural network with hybrid particle swarm optimization”, In: Wang, J., Yi, Z., urada, J.M., Lu, B. urada, J.M., Lu, B.-L., Yin, H. (eds.) ISNN 2006. LNCS, vol. 3971, pp. 577–583. Springer, Heidelberg (2006).

[86] Lichman, M., (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[87] Chu, S.C., Tsai, P., Pan, J.S., “Parallel Particle Swarm Optimization Algorithms with Adaptive Simulated Annealing”, Studies in Computational Intelligence Book Series, Springer Berlin Heidelberg, 31:261-279, 2006.

[88] Sadati, N., Amraee, T., Ranjbar, A., “A global particle swarm-based-simulated annealing optimization technique for under-voltage load shedding problem”, Applied Soft Computing 9 (2009) 652 – 657.

[89] Ge, H., Du, W., Qian, F., "A Hybrid Algorithm Based on Particle Swarm Optimization and Simulated Annealing for Job Shop Scheduling," Third International Conference on Natural Computation (ICNC 2007), Haikou, 2007, pp. 715-719.

[90] Song, X., Cao, Y., Chang, C., "A Hybrid Algorithm of PSO and SA for Solving JSP," 2008

Fifth International Conference on Fuzzy Systems and Knowledge Discovery, Shandong, 2008, pp. 111-115.

[91] Dong, X. et al, "A hybrid discrete PSO-SA algorithm to find optimal elimination orderings for Bayesian networks," 2010 2nd International Conference on Industrial and Information Systems, Dalian, 2010, pp. 510-513.

[92] Shieh, H.-L., Kuo, C.-C., Chiang, C.-M., "Modified particle swarm optimization algorithm with simulated annealing behavior and its numerical verification," Applied Mathematics and Computation, vol. 218, no. 8, pp. 4365–4383, 2011.

[93] Zhang, X.-F., Koshimura, M., Fujita, H. and Hasegawa, R. 2011. An efficient hybrid particle swarm optimization for the job shop scheduling problem. In: Proceedings of the 2011 IEEE International Conference on Fuzzy Systems. pp. 622-626.

[94] Idoumghar, L., Melkemi, M., Schott, R., Aouad, M.I., "Hybrid PSO-SA Type Algorithms for Multimodal Function Optimization and Reducing Energy Consumption in Embedded Systems," Applied Computational Intelligence and Soft Computing, vol. 2011, Article ID 138078, 12 pages, 2011.

[95] Tajbakhsh, A., Eshghi, K., Shamsi, A., "A hybrid PSO-SA algorithm for the travelling tournament problem", Eur J Ind Eng 6:2–25, 2012.

[96] T. Niknam, M. R. Narimani, and M. Jabbari, "Dynamic optimal power flow using hybrid particle swarm optimization and simulated annealing," International Transactions on Electrical Energy Systems, vol. 23, no. 7, pp. 975–1001, 2013.

[97] P. C. Ma, F. Tao, Y. L. Liu, L. Zhang, H. X. Lu and Z. Ding, "A hybrid particle swarm optimization and simulated annealing algorithm for job-shop scheduling," 2014 IEEE International Conference on Automation Science and Engineering (CASE), Taipei, 2014, pp. 125-130.

[98] Sudibyo, S., Murat, M.N., Aziz, N., "Simulated Annealing Particle Swarm Optimization (SA-PSO): particle distribution study and application in Neural Wiener-based NMPC, The 10th

Asian Control Conference (2015).

[99] Wang, X., Sun, Q., "The Study of K-Means Based on Hybrid SA-PSO Algorithm", 2016 9th International Symposium on Computational Intelligence and Design (ISCID), Hangzhou, 2016, pp. 211-214.

[100] Javidrad, F., Nazari, M., "A new hybrid particle swarm and simulated annealing stochastic optimization method", Applied Soft Computing, 60, 634-654, 2017.

[101] Metropolis, N. et al. "Equations of state calculations by fast computing machines", J. Chem. Phys. 21 (1953) 1087–1092.

[102] P. Li, N. Cui, Z. Kong and C. Zhang, "Energy management of a parallel plug-in hybrid electric vehicle based on SA-PSO algorithm", 2017 36th Chinese Control Conference (CCC), Dalian, 2017, pp. 9220-9225.

[103] Colorni, A., Dorigo, M., et al, Distributed Optimization by Ant Colonies, actes de la première conférence européenne sur la vie artificielle, Paris, France, Elsevier Publishing, 134-142, 1991.

[104] Shelokar, P.S., et al, Particle swarm and ant colony algorithms hybridized for improved continuous optimization, In Applied Mathematics and Computation, Volume 188, Issue 1, 2007, Pages 129-142.

[105] Kaveh, A., Talatahari, S., A particle swarm ant colony optimization for truss structures with discrete variables, In Journal of Constructional Steel Research, Volume 65, Issues 8–9, 2009, Pages 1558-1568.

[106] Kaveh, A., Talatahari, S., Particle swarm optimizer, ant colony strategy and harmony search scheme hybridized for optimization of truss structures, In Computers & Structures, Volume 87, Issues 5–6, 2009, Pages 267-283.

[107] Niknam, T., Amiri, B., An efficient hybrid approach based on PSO, ACO and k-means for cluster analysis, Applied Soft Computing, Volume 10, Issue 1, 2010, Pages 183-197.

[108] Chen, S. M., Chien, C., Solving the traveling salesman problem based on the genetic

simulated annealing ant colony system with particle swarm optimization techniques, In Expert Systems with Applications, Volume 38, Issue 12, 2011, Pages 14439-14450.

[109] Xiong, W., Wang, C., "A novel hybrid clustering based on adaptive ACO and PSO," 2011 International Conference on Computer Science and Service System (CSSS), Nanjing, 2011, pp. 1960-1963.

[110] Kıran, M. S. et al., "A novel hybrid approach based on Particle Swarm Optimization and Ant Colony Algorithm to forecast energy demand of Turkey", In Energy Conversion and Management, Volume 53, Issue 1, 2012, Pages 75-83.

[111] Huang, C., et al., Hybridization strategies for continuous ant colony optimization and particle swarm optimization applied to data clustering, In Applied Soft Computing, Volume 13, Issue 9, 2013, Pages 3864-3872.

[112] Mahi, M., et al., A new hybrid method based on Particle Swarm Optimization, Ant Colony Optimization and 3-Opt algorithms for Traveling Salesman Problem, In Applied Soft Computing, Volume 30, 2015, Pages 484-490.

[113] Kefi, S., Rokbani, N., Krömer, P., Alimi, A. M., A New Ant Supervised PSO Variant Applied to Traveling Salesman Problem, The 15th International Conference on Hybrid Intelligent Systems (HIS), Seoul, November 16-18, South Korea, pp. 87-101, 2015.

[114] Lazzus, J. A., et al., Application of particle swarm+ant colony optimization to calculate the interaction parameters on phase equilibria, Journal of Engineering Thermophysics, 2016, Vol. 25, No. 2, pp. 216–226.

[115] Mandloi, M., Bhatia, V., A low-complexity hybrid algorithm based on particle swarm and ant colony optimization for large-MIMO detection, In Expert Systems with Applications, Volume 50, 2016, Pages 66-74.

[116] Indadul, K., et al., Coordinating Particle Swarm Optimization, Ant Colony Optimization and K-Opt Algorithm for Traveling Salesman Problem, Mathematics and Computing: Third International Conference, ICMC 2017, Haldia, India, January 17-21, 2017, Proceedings,

2017, Springer Singapore, pp 103-119

[117] Liu, Y., et al., The Container Truck Route Optimization Problem by the Hybrid PSO-ACO Algorithm, Intelligent Computing Theories and Application: 13th International Conference, ICIC 2017, Liverpool, UK, August 7-10, 2017, Proceedings, Part, 2017, Springer International Publishing, pp 640-648.

[118] Junliang, L., et al., A Hybrid Algorithm Based on Particle Swarm Optimization and Ant Colony Optimization Algorithm, Smart Computing and Communication: First International Conference, SmartCom 2016, Shenzhen, China, December 17-19, 2016, Proceedings, 2017, Springer International Publishing, pp 22-31.

[119] Yang, X.S., Deb, S., "Cuckoo Search via Lévy flights," 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC), Coimbatore, 2009, pp. 210-214.

[120] Ghodrati, A., Lotfi, S., "A hybrid cs/ga algorithm for global optimization", In Proceedings of the International Conference on Soft Computing for Problem Solving (SocProS 2011) December 20-22, 2011, pages 397–404. Springer, 2012.

[121] Nawi, N.M. et al., "Neural network training by hybrid accelerated cuckoo particle swarm optimization algorithm", In: International Conference on Neural Information Processing, 2014, November, (pp. 237-244). Springer International Publishing.

[122] Enireddy, V., Kumar, R.K., "Improved cuckoo search with particle swarm optimization for classification of compressed images", Sadhana, vol.4, no.8, pp.2271-2285, 2015.

[123] Ye, Z., Wang, M., Wang, C., Xu, H., "P2P traffic identification using support vector machine and cuckoo search algorithm combined with particle swarm optimization algorithm", in: Frontiers in Internet Technologies, Springer, 2014, pp. 118–132.

[124] Li, X.T. and Yin, M. H., "A particle swarm inspired cuckoo search algorithm for real parameter optimization," Soft Computing, vol. 20, no. 4, pp. 1389–1413, 2016.

[125] Chen, J.F. et al., "Training Artificial Neural Networks by a Hybrid PSO-CS Algorithm", Algorithms, volume 8, 2015, pp. 292-308.

[126] Guo, J. et al., “Hybrid Optimization Algorithm of Particle Swarm Optimization and Cuckoo Search for Preventive Maintenance Period Optimization,” Discrete Dynamics in Nature and Society, vol. 2016, Article ID 1516271, 12 pages, 2016.

[127] Chi, R., Su, Y., Zhang, D. et al., "A hybridization of cuckoo search and particle swarm optimization for solving optimization problems", Neural Comput & Applic (2017). https://doi.org/10.1007/s00521-017-3012-x.

[128] Dash, J. et al., “Optimal design of linear phase multi-band stop filters using improved cuckoo search particle swarm optimization”, Appl. Soft Comput. 52 (2017) 435–445.

[129] Karaboga, D., Basturk, B., “A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm”, Journal of Global Optimization, 2007, vol 39, pp 459-471.

[130] Shi, X., et al., “An integrated algorithm based on artificial bee colony and particle swarm optimization”, In: 2010 sixth international conference on natural computation (ICNC), vol 5, pp 2586–2590

[131] El-Abd, M., "A hybrid ABC-SPSO algorithm for continuous function optimization," 2011 IEEE Symposium on Swarm Intelligence, Paris, 2011, pp. 1-6.

[132] Kıran, M.S., Gündüz, M., “A recombination-based hybridization of particle swarm optimization and artificial bee colony algorithm for continuous optimization problems”, In Applied Soft Computing, Volume 13, Issue 4, 2013, pp 2188-2203

[133] Xiang, Y., et al., “A particle swarm inspired multi-elitist artificial bee colony algorithm for real-parameter optimization,” Computational Optimization and Applications, vol. 57, no. 2, pp. 493–516, 2014.

[134] Vitorino, L. N., et al., “A mechanism based on Artificial Bee Colony to generate diversity in Particle Swarm Optimization”, In Neurocomputing, Volume 148, 2015, Pages 39-45.

[135] Lin, K., Hsieh, Y., “Classification of medical datasets using SVMs with hybrid evolutionary algorithms based on endocrine-based particle swarm optimization and artificial bee colony

algorithms", J. Med. Syst. 39, 119 (2015)

[136] Zhou, F., Yang, Y., "An Improved Artificial Bee Colony Algorithm Based on Particle Swarm Optimization and Differential Evolution", in Intelligent Computing Theories and Methodologies: 11th International Conference, ICIC 2015, Springer International Publishing, pp 24-35.

[137] Li, Z., et al., "PS–ABC: A hybrid algorithm based on particle swarm and artificial bee colony for high-dimensional optimization problems", In Expert Systems with Applications, Volume 42, Issue 22, 2015, Pages 8881-8895.

[138] Sedighizadeh, D., Mazaheripour, H., "Optimization of multi objective vehicle routing problem using a new hybrid algorithm based on particle swarm optimization and artificial bee colony algorithm considering Precedence constraints", In Alexandria Engineering Journal, 2017.

[139] Farmer, J. D., Packard, N. H. and Perelson, A., "The Immune System, Adaptation, and Machine Learning", Physica D 22(1-3) (1986): 187-204.

[140] Bersini H., Varela F.J., (1991) "Hints for adaptive problem solving gleaned from immune networks", In: Parallel Problem Solving from Nature, PPSN 1990, Lecture Notes in Computer Science, vol 496, Springer, Berlin, Heidelberg.

[141] Forrest, S. et al, 1994, "Self-Nonself Discrimination in a Computer", Proceeding of 1994 IEEE Symposium on Research in Security and Privacy, Los Alamos, CA: IEEE Computer Society Press, 1994.

[142] Kephart, J. O. (1994). "A biologically inspired immune system for computers". Proceedings of Artificial Life IV: The Fourth International Workshop on the Synthesis and Simulation of Living Systems. MIT Press. pp. 130–139.

[143] Yang, X. S., "A new metaheuristic bat-inspired algorithm," in Nicso 2010: Nature Inspired Cooperative Strategies, pp. 65–74, Springer, Berlin, Germany, 2010.

[144] Yang, X.S., (2009) "Firefly Algorithms for Multimodal Optimization", In: Stochastic

Algorithms: Foundations and Applications. SAGA 2009. Lecture Notes in Computer Science, vol 5792. Springer, Berlin, Heidelberg.

[145] Krishnanand, K. N., Ghose, D., “Multimodal Function Optimization using a Glowworm Metaphor with Applications to Collective Robotics”, Proceedings of the 2nd Indian International Conference on Artificial, December 20-22, 2005, pp. 328-346.

[146] Zhao, F., Li, G., Yang, C., Abraham, A., Liu, H., “A human–computer cooperative particle swarm optimization based immune algorithm for layout design”, Neurocomputing 132 (2014).

[147] El-Sherbiny, M. M., Alhamali, R. M., “A hybrid particle swarm algorithm with artificial immune learning for solving the fixed charge transportation problem,” Computers & Industrial Engineering, vol. 64, pp. 610–620, 2013.

[148] Pan,T.S., Dao, T.K., Nguyen, T.T., Chu, S.C. (2015), “Hybrid Particle Swarm Optimization with Bat Algorithm”. In: Genetic and Evolutionary Computing. Advances in Intelligent Systems and Computing, vol 329. Springer, Cham.

[149] Manoj, S., Ranjitha, S., Suresh, H. N., "Hybrid BAT-PSO optimization techniques for image registration," 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), Chennai, 2016, pp. 3590-3596.

[150] Xia, X., Gui, L., He, G., Xie, C., Wei, B., Xing, Y., Wu, R., Tang, Y., “A hybrid optimizer based on firefly algorithm and particle swarm optimization algorithm”, In Journal of Computational Science, 2017, , ISSN 1877-7503.

[151] Arunachalam, S., Agnes Bhomila T., Ramesh Babu M., (2015) “Hybrid Particle Swarm Optimization Algorithm and Firefly Algorithm Based Combined Economic and Emission Dispatch Including Valve Point Effect”, In: Swarm, Evolutionary and Memetic Computing. SEMCCO 2014. Lecture Notes in Computer Science, vol 8947. Springer, Cham.

[152] Shi, Y., Wang, Q. and Zhang, H., "Hybrid ensemble PSO-GSO algorithm," 2012 IEEE 2nd

International Conference on Cloud Computing and Intelligence Systems, Hangzhou, 2012, pp. 114-117.

[153] Liu, H., Zhou, F., "PSO algorithm based on GSO and application in the constrained optimization", Proceedings of the 2nd International Conference on Computer Science and Electronics Engineering (ICCSEE 2013), Advances in Intelligent Systems Research, AISR, ISSN: 1951-6851, volume: 34.

[154] Gies, D. and Y. Rahmat-Samii. "Reconfigurable array design using parallel particle swarm optimization", Antennas and Propagation Society International Symposium, IEEE, 2003.

[155] Schutte, J.F, Reinbolt, J.A, Fregly, B.J, Haftka, R.T, George, A.D, "Parallel Global Optimization with the Particle Swarm Algorithm", Int J Numer Meth Eng 61(13):2296–2315, John Wiley and Sons Ltd, Great Britain, 2004.

[156] Venter, G., Sobieszczanski-Sobieski, J., "A parallel particle swarm optimization algorithm accelerated by asynchronous evaluations", 6th world congresses of structural and multidisciplinary optimization, 30 May–03 June, 2005.

[157] Chang, J-F., Chu, S-C., Roddick, J.F., Pan, J.S., "A parallel particle swarm optimization algorithm with communication strategies", J Information Sci Eng 21:809–818, 2005.

[158] Waintraub, M., Schirru, R., Pereira, C. M. N. A., "Multiprocessor modeling of parallel Particle Swarm Optimization applied to nuclear engineering problems," Progress in Nuclear Energy, vol. 51, no. 6, pp. 680–688, 2009.

[159] Rymut, B., Kwolek, B., "GPU-supported object tracking using adaptive appearance models and Particle Swarm Optimization", Proceedings of the 2010 international conference on Computer vision and graphics: Part II, ICCVG'10, Springer-Verlag, Berlin, Heidelberg, 227–234, 2010.

[160] Gordon, N.J., Salmond, D.J., Smith, A.F.M., "Novel Approach to Nonlinear/Non-Gaussian Bayesian State Estimation", IEE Proc., F, Radar Signal Process. 140 (2), pp 107-113, 1993.

[161] Zhang, J., Pan, T.-S., Pan, J.-S., "A parallel hybrid evolutionary particle filter for nonlinear

state estimation," in Proc. Robot, Vis. Signal Process. First Int. Conf., Nov. 2011, pp. 308–312.

[162] Chen, R.-B., Hsu, Y.-W., Hung, Y., Wang, W., "Discrete particle swarm optimization for constructing uniform design on irregular regions," Computational Statistics & Data Analysis, vol. 72, pp. 282–297, 2014.

[163] Awwad, O., Al-Fuqaha, A., Ben Brahim, G., Khan, B., Rayes, A. "Distributed topology control in large-scale hybrid RF/FSO networks: SIMT GPU-based particle swarm optimization approach," International Journal of Comm. Systems, vol. 26, no. 7, pp. 888–911, 2013.

[164] Qu, Jianhua., Liu, Xiyu., Sun, Minghe., Qi, Feng., "GPU-Based Parallel Particle Swarm Optimization Methods for Graph Drawing," Discrete Dynamics in Nature and Society, vol. 2017, Article ID 2013673, 15 pages, 2017.

[165] Zhou, Y., Tan, Y., "GPU-based parallel particle swarm optimization",Proceedings of IEEE Congress on Evolutionary Computation (CEC 2009), 2009, pp. 1493–1500.

[166] Mussi, L., Daolio, F., Cagnoni, S., "Evaluation of parallel particle swarm optimization algorithms within the CUDA™ architecture", In Information Sciences, Volume 181, Issue 20, 2011, Pages 4642-4657, ISSN 0020-0255.

[167] Parsopoulos, K.E., Plagianakos, V.P., Magoulas, G.D., Vrahatis, M.N., "Improving particle swarm optimizer by function "stretching" ", Nonconvex Optimization and Applications, vol. 54, ch. 3, pp. 445– 457, Kluwer Academic Publishers, The Netherlands, 2001.

[168] Parsopoulos, K.E., Plagianakos, V.P., Magoulas, G.D., Vrahatis, M.N., "Stretching technique for obtaining global minimizers through particle swarm optimization", in: Proceedings of the Workshop on Particle Swarm Optimization, Indianapolis, IN, 2001.

[169] Parsopoulos, K.E., Vrahatis, M.N., "Modification of the particle swarm optimizer for locating all the global minima, Artificial Neural Networks and Genetic Algorithms", Computer Science series, Springer, Wien, 2001, pp. 324–327.

[170] Parsopoulos, K.E., Vrahatis, M. N., “On the computation of all global minimizers through particle swarm optimization”, IEEE transactions on evolutionary computation 8 (2004), no. 3, 211–224.

[171] Brits, R., Engelbrecht, A.P., van den Bergh, F., “Solving systems of unconstrained equations using particle swarm optimization”, Proceedings of the IEEE 2002 Conference on Systems, Man, and Cybernetics (Tunisia), 2002.

[172] Brits, R., Engelbrecht, A.P., van den Bergh, F., “A niching particle swarm optimizer”, Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning (SEAL’02) (Singapore), vol. 2, November 2002, pp. 692–696.

[173] X, Li., “Adaptively choosing neighbourhood bests using species in a particle swarm optimizer for multimodal function optimization”, In: Deb, K., et al. (eds.) GECCO 2004. LNCS, vol. 3102, pp. 105–116. Springer, Heidelberg (2004).

[174] Bird, S., “Adaptive techniques for enhancing the robustness and performance of speciated psos in multimodal environments”, Phd thesis. Ph.D. dissertation, RMIT University, Melbourne, Australia (2008).

[175] Bird, S., Li, X. “Adaptively choosing niching parameters in a PSO” In: Cattolico, M. (ed.) Proceedings of Genetic and Evolutionary Computation Conference, GECCO 2006, Seattle, Washington, USA, July 8-12, pp. 3–10. ACM, New York (2006).

[176] Li, X.: Multimodal function optimization based on fitness-euclidean distance ratio. In: Thierens, D. (ed.) Proc. of Genetic and Evolutionary Comp. Conf. 2007, pp. 78–85 (2007).

[177] Kennedy, J.: Stereotyping: Improving particle swarm performance with cluster analysis. In: Proc. of IEEE Int. Conf. Evolutionary Computation, pp. 303–308 (2000).

[178] Passaro, A., Starita, A.: Particle swarm optimization for multimodal functions: a clustering approach. J. Artif. Evol. App. 2008, 1–15 (2008).

[179] Schwarz, G.: Estimating the dimension of a model. Annals of Statistics 6(2), 461– 464 (1978).

[180] Blackwell, T.M., Branke, J.: Multi-swarm optimization in dynamic environments. In: Raidl, G.R., Cagnoni, S., Branke, J., Corne, D.W., Drechsler, R., Jin, Y., Johnson, C.G., Machado, P., Marchiori, E., Rothlauf, F., Smith, G.D., Squillero, G. (eds.) EvoWorkshops 2004. LNCS, vol. 3005, pp. 489–500. Springer, Heidelberg (2004).

[181] Bird, S., Li, X.: Using regression to improve local convergence. In: Proceedings of the 2007 IEEE Congress on Evolutionary Computation, Singapore, pp. 1555–1562 (2007).

[182] Parrott, D., Li, X.: Locating and tracking multiple dynamic optima by a particle swarm model using speciation. IEEE Trans. on Evol. Comput. 10(4), 440–458 (2006).

[183] Li, X.: Niching without niching parameters: Particle swarm optimization using a ring topology. IEEE Trans. on Evol. Comput. 14(1), 150–169 (2010).

[184] Afshinmanesh, F., Marandi, A., Rahimi-Kian, A., "A novel binary particle swarm optimization method using artificial immune system", IEEE, pp 217–220, 2005.

[185] Deligkaris, K.V., Zaharis, Z.D., Kampitaki, D.G., Goudos, S.K., Rekanos, I.T., Spasos, M.N., "Thinned planar array design using Boolean PSO with velocity mutation", IEEE Trans Magn 45(3):1490–1493, 2009.

[186] Chen, W., Zhang, J., Chung, H., Zhong, W., Wu, W., Shi, Y., "A novel set-based particle swarm optimization method for discrete optimization problems," IEEE Trans. Evol. Comput., vol. 14, no. 2, pp. 278–300, 2010.

[187] Gong, Y., Zhang, J., Liu, O., Huang, R., Chung, H., and Shi, Y., "Optimizing vehicle routing problem with time windows: A discrete particle swarm optimization approach," IEEE Trans. Syst. Man Cybern., vol. 42, no. 2, pp. 254–267, 2012.

[188] Solomon, M., "Algorithms for the vehicle routing and scheduling problems with time window constraints," Operations Research 35, 254–265, 1987.

[189] Kitayama, S., Arakawa, M., Yamazaki, K., "Penalty function approach for the mixed discrete nonlinear problems by particle swarm optimization", Structural Multidisciplinary Optimization 32 (2006) 191–202.

[190] Nema, S., Goulermas, J., Sparrow, G., Cook, P., “A hybrid particle swarm branch-and-bound (HPB) optimizer for mixed discrete nonlinear programming”, IEEE Trans. on Systems, Man, and Cybernetics, Part A, vol.38, no.6, pp.1411-1424, 2008.

[191] Sun, C., Zeng, J., Pan, J., Zhang, Y., "PSO with Constraint-Preserving Mechanism for Mixed-Variable Optimization Problems," 2011 First International Conference on Robot, Vision and Signal Processing, Kaohsiung, 2011, pp. 149-153.

[192] Chowdhury, S., Zhang, J., Messac, A., “Avoiding premature convergence in a mixed-discrete particle swarm optimization (MDPSO) algorithm” In: 53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, No. AIAA 2012-1678. Honolulu, Hawaii: AIAA; 2012.

[193] Laskari, E., Parsopoulos, K., and Vrahatis, M., “Particle swarm optimization for integer programming,” in Proc. IEEE Congr. Evol. Comput., May 2002, vol. 2, pp. 1582–1587.

[194] Yare, Y., and Venayagamoorthy, G. K., "Optimal Scheduling of Generator Maintenance Using Modified Discrete Particle Swarm Optimization," Proceedings of the Symposium on Bulk Power System Dynamics and Control - VII. Revitalizing Operational Reliability, 2007 iREP, Institute of Electrical and Electronics Engineers (IEEE), Aug 2007.

[195] Eajal, A. A., and El-Hawary, M. E., “Optimal capacitor placement and sizing in unbalanced distribution systems with harmonics consideration using particle swarm optimization,” IEEE Trans. Power Del. , vol. 25, no. 3, pp. 1734–1741, Jul. 2010.

[196] Phung, M.D., Quach, C.H., Dinh, T.H., Ha, Q., “Enhanced discrete particle swarm optimization path planning for UAV vision-based surface inspection”, In Automation in Construction, Volume 81, 2017, Pages 25-33, ISSN 0926-5805.

[197] Gong, M.G., Yan, J.N., Shen, B., Ma, L.J., Cai, Q., “Influence maximization in social networks based on discrete particle swarm optimization”. Inform Sci 2016;367–368:600–14.

[198] Aminbakhsh, S., Sonmez, R., “Discrete particle swarm optimization method for the large-scale discrete time–cost trade-off problem”, In Expert Systems with Applications, Volume

51, 2016, Pages 177-185, ISSN 0957-4174.

[199] Li, L., Jiao, L., Zhao, J., Shang, R., Gong, M., “Quantum-behaved discrete multi-objective particle swarm optimization for complex network clustering”, In Pattern Recognition, Volume 63, 2017, Pages 1-14, ISSN 0031-3203.

[200] Girvan, M., Newman, M.E.J., "Community structure in social and biological networks", Proc. Natl. Acad. Sci. 99 (12) 7821–7826, 2002.

[201] Ates, A., Alagoz, B. B., Kavuran, G., Yeroglu, C., “Implementation of fractional order filters discretized by modified Fractional Order Darwinian Particle Swarm Optimization”, in Measurement, Volume 107, 2017, Pages 153-164, ISSN 0263-2241.

[202] Du, W., Li, B., “Multi-strategy ensemble particle swarm optimization for dynamic optimization”, Inf. Sci. 178 (2008) 3096–3109.

[203] Wolpert, D.H., Macready, W.G., “No free lunch theorems for optimization”, IEEE Trans. Evol. Comput. 1 (1997) 67–82.

[204] Engelbrecht, A.P., “Heterogeneous particle swarm optimization”, in: Swarm Intell, Springer, 2010, pp. 191–202.

[205] Lynn, N., Suganthan, P. N., “Ensemble particle swarm optimizer”, In Applied Soft Computing, Volume 55, 2017, Pages 533-548, ISSN 1568-4946.

[206] Shirazi, M.Z., Pamulapati, T., Mallipeddi, R., Veluvolu, K.C., (2017) “Particle Swarm Optimization with Ensemble of Inertia Weight Strategies”. In: Advances in Swarm Intelligence. ICSI 2017. Lecture Notes in Computer Science, vol 10385. Springer, Cham.

[207] Lynn, N., Suganthan, P.N., “Heterogeneous comprehensive learning particle swarm optimization with enhanced exploration and exploitation”. Swarm Evol. Comput. 24, 11–24, 2015.

[208] Mirjalili, S, Mirjalili, S.M., Lewis, A., Grey Wolf Optimizer, Advances in Engineering Software, Volume 69, 2014, Pages 46-61, ISSN 0965-9978.

[209] Rashedi, E., Nezamabadi-pour, H., Saryazdi, S., GSA: A Gravitational Search Algorithm,

Information Sciences, Volume 179, Issue 13, 2009, Pages 2232-2248, ISSN 0020-0255.

[210] Mirjalili, S. and Hashim, S. Z. M., A new hybrid PSOGSA algorithm for function optimization, 2010 International Conference on Computer and Information Application, Tianjin, 2010, pp. 374-377.

[211] Xia, X., Gui, L., He, G., Xie, C., Wei, B., Xing, Y., Wu, R., Tang, Y., A hybrid optimizer based on firefly algorithm and particle swarm optimization algorithm, Journal of Computational Science, Volume 26, 2018, Pages 488-500, ISSN 1877-7503.

[212] Y. D. Sergeyev, D. E. Kvasov, and M. S. Mukhametzhanov, ‘‘On the efficiency of nature-inspired metaheuristics in expensive global optimization with limited budget,’’ Sci. Rep., vol. 8, Jan. 2018, Art. no. 453, doi: 10.1038/s41598-017-18940-4

[213] D.E. Kvasov, M.S. Mukhametzhanov, Metaheuristic vs. deterministic global optimization algorithms: The univariate case, Applied Mathematics and Comput. 318 (2018) 245–259.

[214] Kvasov, D.E., Mukhametzhanov, M.S., (2016) One-dimensional global search: nature-inspired vs. lipschitz methods. AIP Conf Proc 1738(1):400012.

[215] Gaviano, M., Kvasov, D.E., Lera, D., Sergeyev, Y.D., 2003. Algorithm 829: Software for generation of classes of test functions with known local and global minima for global optimization. ACM Trans. Math. Softw. 29, 4 (December 2003), 469-480.

[216] Sergeyev, Y.D., Kvasov, D.E., Global search based on efficient diagonal partitions and a set of Lipschitz constants, SIAM J. Optim. 16 (3) (2006) 910–937.

[217] Sun, J., Feng, B., Xu, W.B., “Particle swarm optimization with particles having quantum behavior.”, IEEE Proc. of Congress on Evolutionary Computation, pp. 325–331, 2004.

[218] Sun, J. et al. “A global search strategy of quantum behaved particle swarm optimization.”, Cybernetics and Intelligent Systems Proc. of the 2004 IEEE Conference, pp. 111–116, 2004.

[219] Xi, M., Sun, J., Xu, W., “An improved quantum-behaved particle swarm optimization algorithm with weighted mean best position”, Applied Mathematics and Computation, Volume 205, Issue 2, 2008, pp. 751-759.

[220] Eberhart, R., Shi, Y., “Comparison between genetic algorithms and particle swarm optimization.”, Proc. 7th Ann. Conf. Evolutionary Computation, San Diego, 2000.

[221] Xie, Z., Liu, Q., Xu, L., “A New Quantum-Behaved PSO: Based on Double δ-Potential Wells Model”, Proc. of 2016 Chinese Intelligent Systems Conference, CISC 2016, LNEE, vol 404. Springer, Singapore, 2016, pp 211-219.

[222] Basak, S., Lecture Notes, P303 (PE03) Quantum Mechanics I, National Institute of Science Education and Research, http://www.niser.ac.in/~sbasak/p303_2010/06.09.pdf.

[223] Griffiths, David J. (2005), Introduction to Quantum Mechanics, 2nd Edition; Pearson Education - Problem 2.27.

[224] Dhabal, S. and Sengupta, S., "Efficient design of high pass FIR filter using quantum-behaved particle swarm optimization with weighted mean best position," Proceedings of the 2015 Third International Conference on Computer, Communication, Control and Information Technology (C3IT), Hooghly, 2015, pp. 1-6.

[225] Sengupta, S. and Basak, S., "Computationally efficient low-pass FIR filter design using Cuckoo Search with adaptive Levy step size," 2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC), Jalgaon, 2016, pp. 324-329.

[226] Han,P., Yuan, S., Wang, D., “Thermal System Identification Based on Double Quantum Particle Swarm Optimization”, Intelligent Computing in Smart Grid and Electrical Vehicles, ICSEE 2014, LSMS 2014, Communications in Computer and Information Science, vol 463. Springer, Berlin, Heidelberg, 2014, pp 125-137.

[227] Jia, P., Duan, S., Yan, J., "An enhanced quantum-behaved particle swarm optimizatin based on a novel computing way of local attractor", Information 2015, 6, 633–649.

[228] Sengupta, S. et. al., A Review of Deep Learning with Special Emphasis on Architectures, Applications and Recent Trends, arXiv:1905.13294 [cs.LG].

[229] Sengupta, S.; Basak, S.; Peters, R.A. QDDS: A Novel Quantum Swarm Algorithm Inspired

by a Double Dirac Delta Potential. Proceedings of 2018 IEEE Symposium Series on Computational Intelligence (in press), arXiv 2018, arXiv:1807.02870.

[230] Sun, J.; Feng, B.; Xu, W.B. Particle swarm optimization with particles having quantum behavior. In Proceedings of the IEEE Congress on Evolutionary Computation, Portland, OR, USA, 19–23 June 2004; pp. 325–331.

[231] Sun, J.; Xu, W.B.; Feng, B. A global search strategy of quantum behaved particle swarm optimization. In Proceedings of the 2004 IEEE Conference on Cybernetics and Intelligent Systems, Singapore, 1–3 December 2004; pp. 111–116.

[232] Xi, M.; Sun, J.; Xu, W. An improved quantum-behaved particle swarm optimization algorithm with weighted mean best position. Appl. Math. Comput. 2008, 205, 751–759.

[233] Solis, F.J.; Wets, R.J.-B. Minimization by random search techniques. Math. Oper. Res. 1981, 6, 19–30.

[234] Sengupta, S.; Basak, S.; Peters, R.A., II. Chaotic Quantum Double Delta Swarm Algorithm Using Chebyshev Maps: Theoretical Foundations, Performance Analyses and Convergence Issues. J. Sens. Actuator Netw. 2019, 8, 9.

[235] Clerc, M.; Kennedy, J. The particle swarm: Explosion, stability, and convergence in a multi-dimensional complex space. IEEE Trans. Evol. Comput. 2002, 6, 58–73.

[236] Sengupta, S.; Basak, S.; Peters, R.A., II. Particle Swarm Optimization: A Survey of Historical and Recent Developments with Hybridization Perspectives. Mach. Learn. Knowl. Extr. 2018, 1, 157–191.

[237] Khare, A.; Rangnekar, S. A review of particle swarm optimization and its applications in Solar Photovoltaic system. Appl. Soft Comput. 2013, 13, 2997–3006.

[238] Zhang, Y.; Wang, S.; Ji, G. A Comprehensive Survey on Particle Swarm Optimization Algorithm and Its Applications. Math. Probl. Eng. 2015, 2015, 931256.

[239] Ab Wahab, et al. A Comprehensive Review of Swarm Optimization Algorithms. PLoS ONE 2015, 10, e0122827.

[240] Xi, M.; Wu, X.; Sheng, X.; Sun, J.; Xu, W. Improved quantum-behaved particle swarm optimization with local search strategy. J. Algorithms Comput. Technol. 2017, 11, 3–12.

[241] Fisher, R.A. "The use of multiple measurements in taxonomic problems" Annual Eugenics, 7, Part II, 179-188 (1936).

[242] Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[243] Wang, G.-G.; Guo, L.; Gandomi, A.H.; Hao, G.-S.; Wang, H. Chaotic krill herd algorithm. Inf. Sci. 2014, 274, 17–34.

[244] Reynolds, C., Boids Background and Update, http://www.red3d.com/cwr/boids

[245] He, D.; He, C.; Jiang, L.; Zhu, H.; Hu, G. Chaotic characteristic of a one-dimensional iterative map with infinite collapses. IEEE Trans. Circuits Syst. 2001, 48, 900–906.

[246] Mirjalili, S. SCA: A Sine Cosine Algorithm for solving optimization problems, Knowledge-Based Systems. 2016, 96, 120–133, doi:10.1016/j.knosys.2015.12.022.

[247] Mirjalili, S. Dragonfly algorithm: A new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. Neural Comput. Appl. 2016, 24, 1053–1073.

[248] Mirjalili, S. The ant lion optimizer. Adv. Eng. Softw. 2015, 83, 80–98.

[249] Mirjalili, S.; Lewis, A. The Whale Optimization Algorithm. Adv. Eng. Softw. 2016, 95, 51–67. doi:10.1016/j.advengsoft.2016.01.008.

[250] Yang, X.-S. Firefly algorithms for multimodal optimization. In Stochastic Algorithms: Foundations and Applications, SAGA 2009; Sapporo, Japan; Lecture Notes in Computer Sciences; 2009; Volume 5792, pp. 169–178.

[251] Han, P.; Yuan, S.; Wang, D. Thermal System Identification Based on Double Quantum Particle Swarm Optimization. In Proceedings of the Intelligent Computing in Smart Grid and Electrical Vehicles, ICSEE 2014, LSMS 2014, Communications in Computer and

Information Science, Shanghai, China, 20–23 September 2014; Springer: Berlin/Heidelberg, Germany, 2014; Volume 463, pp. 125–137.

[252] Jia, P.; Duan, S.; Yan, J. An enhanced quantum-behaved particle swarm optimization based on a novel computing way of local attractor. Information 2015, 6, 633–649.

[253] Van den Bergh, F.; Engelbrecht, A. A convergence proof for the particle swarm optimiser. Fundam. Inf. 2010, 105, 341–374.

[254] Fang, W.; Sun, J.; Chen, H.; Wu, X. A decentralized quantum inspired particle swarm optimization algorithm with cellular structured population. Inf. Sci. 2016, 330, 19–48.

[255] Gao, Y. et al. Selectively-informed particle swarm optimization. Sci. Rep. 2015, 5, 9295.

[256] Liu, C.; Du, W.B.; Wang, W.X. Particle Swarm Optimization with Scale-Free Interactions. PLoS ONE 2014, 9, e97822.

[257] Sengupta, S.; Basak, S.; Peters, R.A. Data Clustering using a Hybrid of Fuzzy C-Means and Quantum-behaved Particle Swarm Optimization. In Proceedings of the 2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 8–10 January 2018; pp. 137–142.

[258] Tatsumi, K.; Tetsuzo, T. A perturbation based chaotic system exploiting the quasi-newton method for global optimization. Int. J. Bifur. Chaos 2017, 27, 1750047.

[259] Coelho, L.; Mariani, V.C. Use of chaotic sequences in a biologically inspired algorithm for engineering design optimization. Expert Syst. Appl. 2008, 34, 1905–1913.

[260] S. Basak, S. Sengupta and A. Dubey, "Mechanisms for Integrated Feature Normalization and Remaining Useful Life Estimation Using LSTMs Applied to Hard-Disks," 2019 IEEE International Conference on Smart Computing (SMARTCOMP), Washington, DC, USA, 2019, pp. 208-216.

[261] Gandomi, A.; Yang, X.-S.; Talatahari, S.; Alavi, A. Firefly algorithm with chaos. Commun. Nonlinear Sci. Numer. Simul. 2013, 18, 89–98. doi:10.1016/j.cnsns.2012.06.009.