Q-Learning Induced Artificial Bee Colony for Noisy Optimization

Pratyusha Rakshit[1], Amit Konar[2]
[1]*Basque Center for Applied Mathematics, Bilbao, Spain*
[1, 2]*Department of Electronics and Telecommunication Engineering Jadavpur University, Kolkata, India*
[1]pratyushar1@gmail.com, [2]konaramit@yahoo.co.in

Atulya K. Nagar
Department of Mathematics and Computer Science, Liverpool Hope University, United Kingdom
nagara@hope.ac.uk

***Abstract*— The paper proposes a novel approach to adaptive selection of sample size for a trial solution of an evolutionary algorithm when noise of unknown distribution contaminates the objective surface. The sample size of a solution here is adapted based on the noisy fitness profile in the local surrounding of the given solution. The fitness estimate and the fitness variance of a sub-population surrounding the given solution are jointly used to signify the degree of noise contamination in its local neighborhood (LN). The adaptation of sample size based on the characteristics of the fitness landscape in the LN of a solution is realized here with the temporal difference Q-learning (TDQL). The merit of the present work lies in utilizing the reward-penalty based reinforcement learning mechanism of TDQL for sample size adaptation. This sidesteps the prerequisite setting of any specific functional form of relationship between the sample size requirement of a solution and the noisy fitness profile in its LN. Experiments undertaken reveal that the proposed algorithms, realized with artificial bee colony, significantly outperform the existing counterparts and the state-of-the-art algorithms.**

Keywords*— *artificial bee colony; noise-handling; temporal difference Q-learning; reinforcement learning; sampling.

I. INTRODUCTION

The reliance of real world optimization problems on *evolutionary algorithms* (EAs) has been widely observed over the past decades. The real world system characteristic is captured by the objective function of an EA, which is optimized for ensuring optimal utilization of system resources. However, the efficacy of the traditional EAs degrades with creeping of noise in the measurement of input data of the real world systems, primarily due to degraded sensor characteristics and/or noisy ambience. Edging of noise in the input measurement also induces inaccuracy in the assessment of objective function value (often called *fitness*) of a population member of an EA. Noisy fitness measurements of quality (and poor) solutions deceive the evolutionary selection operator by rejecting (and accepting) them over the generations. Consequently, the traditional EA fails to track the true global optimum in the presence of noise in the fitness landscape. The intricacy of the problem increases with noise of unknown stochastic distribution contaminating multimodal fitness landscape. It has thus called the amendment of the traditional EAs to overcome the illusive effect of noise. This class of optimization algorithm devised to alleviate the problem is referred to as *noisy optimization* [1], [2].

The well-known strategies used in noisy optimization algorithms include sampling [3-9], effective estimation of fitness [10], [11], dynamic population sizing [12], [13], improved search dynamics [14], [15] and robust selection operator [16], [17]. Among these, the most significant stratagem is sampling, where the fitness of a solution is periodically evaluated for a number of times, called *sample size*. An aggregate measure of these *fitness samples* thus obtained is used as the fitness estimate of the given solution. The multiple fitness evaluation of a solution increases the accuracy in its fitness assessment in the presence of noise of unknown distribution.

Evidently, a large sample size ensures fitness accuracy but at the cost of computational overhead. To overcome this impasse, dynamic sampling policies are developed in the recent past [3-9]. Among the existing strategies, sequential sampling [3], progress based dynamic sampling strategy [4], noise analysis selection [5], and optimal computing budget allocation [6] need special mentioning. The major constraint of the existing strategies is to overlook the level of contamination of noise in the local neighbourhood of a solution to determine its sampling requirement.

This issue has been addressed by the authors' previous works [7-9]. The variance of fitness estimates of a sub-population surrounding the given solution is used to capture the degree of creeping of noise in its local neighbourhood. It is referred to as *local neighbourhood fitness variance* (LNFV). Different monotonically increasing functional forms have been used to model the relationship between LNFE and the sample size required for a solution [7-9]. Experiments undertaken reveal that a specific functional form cannot guarantee optimal assignment of sample size to a solution for all possible distribution of noise in the fitness landscape.

The paper proposes an alternative approach to bypass the need of a particular functional form to determine the sample size of a solution based on its LNFV. In addition to the LNFV, the policy also adapts sample size based on the *local neighborhood fitness estimate* (LNFE). The LNFV and LNFE are jointly used to capture the noise-contaminated fitness profile in the local vicinity of a solution. The reward and penalty based reinforcement learning mechanism of *temporal difference Q-learning* (TDQL) [18], [19] is used to prudently assign sample size (similar to action of TDQL) to a solution based on its local fitness profile, jointly captured by its

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

respective LNFV and LNFE (similar to states of TDQL). If the assignment of a sample size n to a solution by the TDQL increases LNFE (for maximization problem) and simultaneously reduces the respective LNFV, the selection is rewarded. Otherwise, the selection is penalized. These reward and penalties are encoded in the Q-values which in turn guide population of next generations to judiciously select their sample sizes.

In the present work, the *artificial bee colony* [20] is selected as the EA. The performance of the proposed noisy single objective optimization algorithm, referred to as *Q-learning induced noisy bee colony* (QLNBC), is compared with four state-of-the-art techniques [21-24]. The proposed adaptive sampling strategy is also used to replace the sampling policy recommended in [25] and results in a new competitor in the comparative framework, called *extended* QLNBC. The comparative performance is analyzed with respect to *function error value* metric while optimizing noisy versions of 28 CEC'2013 benchmark functions [26]. The benchmark functions are contaminated with additive noise samples taken from Gaussian [27], Poisson [28], Rayleigh [29], exponential [30], and random distribution [31]. Experiments undertaken reveal that the proposed realizations outperform other algorithms in a statistically significant manner. The Wilcoxon test, the Friedman and Iman-Davenport non-parametric tests and the Bonferroni-Dunn post-hoc analysis [32] are undertaken to arrive at the conclusion.

The paper is divided into six sections. Section II overviews the traditional ABC and the TDQL algorithms. A brief outline of the existing sample size adaptation policies is given in section III. Section IV provides the noise handling mechanism used to extend the ABC. Simulation results are reported in section V. Section VI concludes the paper.

II. PRELIMINARIES

A. Artificial Bee Colony Algorithm

Artificial bee colony (ABC) [20] is a population-based meta-heuristic algorithm. An overview of the main steps of the ABC algorithm is presented next for maximization problem.

1. Initialization: ABC begins with a population $\mathbf{P}(t)$ of S, D-dimensional real-valued solutions $\{\vec{Z}_1(t), \vec{Z}_2(t), ..., \vec{Z}_S(t)\}$, randomly initialized within the search bound of the given optimization problem at generation $t = 0$. The fitness value $fit(\vec{Z}_i(0))$ of $\vec{Z}_i(0)$ is evaluated for $i = 1, 2, \ldots, S$.

2. Employed Bee Phase: In this phase, a randomly chosen parameter ($j \in \{1, 2, \ldots, D\}$ of a solution $\vec{Z}_i(t)$ is modified to generate an offspring solution $\vec{Z}'_i(t)$ following (1) for $i = 1, 2, \ldots, S$. The remaining D–1 components of $\vec{Z}'_i(t)$ are identical to that of $\vec{Z}_i(t)$.

$$z'_{i,j}(t) = z_{i,j}(t) + F \times (z_{i,j}(t) - z_{k,j}(t)) \qquad (1)$$

Here $\vec{Z}_k(t)$ is a solution selected randomly from $\mathbf{P}(t) \setminus \vec{Z}_i(t)$ and F represents the scale factor in $(-1, 1)$. $\vec{Z}_i(t)$ is replaced by $\vec{Z}'_i(t)$, if $fit(\vec{Z}'_i(t)) \geq fit(\vec{Z}_i(t))$.

3. Probability of Selection by Onlookers: This step is concerned with assigning a high probability of selection to a solution with better fitness for the subsequent step. In other words, the selection probability of $\vec{Z}_i(t)$ with fitness $fit(\vec{Z}_i(t))$ is calculated using (2) for $i = 1, 2, \ldots, S$.

$$ps_i = fit(\vec{Z}_i(t)) \Big/ \sum_{j=1}^{S} fit(\vec{Z}_j(t)) \qquad (2)$$

4. Onlooker Bee Phase: The q-th onlooker probabilistically selects a solution $\vec{Z}_i(t) \in \mathbf{P}(t)$ based on ps_i and employs (1) to generate a new solution $\vec{Z}'_i(t)$. $\vec{Z}'_i(t)$ replaces $\vec{Z}_i(t)$ in the next generation t+1 provided $fit(\vec{Z}'_i(t)) \geq fit(\vec{Z}_i(t))$. This step is repeated for all onlooker bees with $q = 1, 2, \ldots, S$.

5. Scout Bee Phase: This step deals with random re-initialization of a population member (within predefined search boundary) remaining unchanged for a predefined number of generations, called '*limit*'.

6. Convergence: After each evolution cycle, steps 2 to 5 are repeated until terminating criterion is satisfied.

B. Temporal Difference Q-Learning

Q-learning is a well-known member of the class of reinforcement learning [18], [19]. In Q-learning, an agent at a given state $\tau_i \in \{\tau_1, \tau_2, \ldots, \tau_M\}$ selects and executes an action α_j from a set of possible N actions $\{\alpha_1, \alpha_2, \ldots, \alpha_N\}$ and eventually receives an immediate reward/penalty $r(\tau_i, \alpha_j)$ from its environment. It helps the agent to learn a control policy to reach a definite goal by increasing the possibility of its correct response.

Let,

$Q(\tau_i, \alpha_j)$ be the quality factor of the state-action pair $<\tau_i, \alpha_j>$ encoding the total reward that the agent acquires by execution of an action α_j at state τ_i.

$\delta(\tau_i, \alpha_j)$ be the transition function that returns the next state $\tau_k \in \{\tau_1, \tau_2, \ldots, \tau_M\}$ of the agent due to execution of action α_j at state τ_i.

To ensure unbiased selection of actions in the initial phase, equal Q-value is assigned to each state-action pair. At a particular instant of Q-learning, the agent occupies a state $\tau_i \in \{\tau_1, \tau_2, \ldots, \tau_M\}$, consults the Q-values $Q(\tau_i, \alpha_l)$, for $l = 1, 2, \ldots, N$ to select and execute an action $\alpha_j \in \{\alpha_1, \alpha_2, \ldots, \alpha_N\}$. The agent consequently receives an immediate reward $r(\tau_i, \alpha_j)$ and moves to the new state τ_k using δ-transition rule. The Q-value $Q(\tau_i, \alpha_j)$ is then updated based on the immediate reward $r(\tau_i, \alpha_j)$ and the cumulative reward expected by the agent in subsequent state-transitions from its next state τ_k. The currently updated Q-value is used to guide the agent to judiciously select its actions at state $\tau(i)$ in subsequent instants. Now, the next state $\tau_k = \delta(\tau_i, \alpha_j)$ is considered as the initial state, and the steps of action selection, receiving immediate reward, state-transition and Q-value update are repeated forever.

Evidently, the reinforcement learning policy used to update the Q-values plays the vital role to ensure the quality performance of Q-learning. According to the classical Q-learning, the updated Q value at state τ_i after execution of action α_j is given by (3).

$$Q(\tau_i,\alpha_j) = r(\tau_i,\alpha_j) + \gamma \max_{\alpha'} Q(\delta(\tau_i,\alpha_j),\alpha') = r(\tau_i,\alpha_j) + \gamma \max_{\alpha'} Q(\tau_k,\alpha') \quad (3)$$

Here, $\gamma \in (0, 1]$ denotes the discounting factor used to signify the importance of the cumulative future reward $\max_{\alpha'} Q(\tau_k,\alpha')$. A factor of 0 implies an opportunist agent by considering only the immediate rewards, while γ approaching 1 will make the agent strive for a long-term high reward.

The *temporal difference Q-learning* (TDQL) [18], [19] is a modified version of the classical Q-learning. In TDQL, the current Q-value at state-action pair $<\tau_i, \alpha_j>$ is also used to update its value following (4).

$$Q(\tau_i,\alpha_j) \leftarrow (1-\beta)\times Q(\tau_i,\alpha_j) + \beta\times(r(\tau_i,\alpha_j) + \gamma \max_{\alpha'} Q(\delta(\tau_i,\alpha_j),\alpha') \quad (4)$$

The learning rate $\beta \in (0, 1)$ determines the degree of dominance of the newly acquired reward over the old information. Evidently, the agent stop learning with $\beta = 0$, while $\beta = 1$ allows the agent to consider the latest information only.

III. Overview of Existing Sampling Policies

The primary objective of a noisy optimization problem is to estimate the true fitness of a candidate solution when the fitness landscape is contaminated with noise of unknown stochastic distribution. The infiltration of noise in the fitness measure of a quality solution may deprive it from promotion to the next generation population. On the contrary, a true inferior solution with deceptive noisy fitness value may occupy the next generation population.

To avoid this camouflaged presence of noise in the fitness measure of an offspring $\vec{Z}_i'$ (generated from its parent $\vec{Z}_i$ through employed/onlooker bee phase), its fitness $fit(\vec{Z}_i')$ is repeatedly evaluated for $n(\vec{Z}_i')$ times, called *sample size*. The strategy is referred to as *sampling*. An aggregate measure of the resulting $n(\vec{Z}_i')$ number of *fitness samples* of $fit(\vec{Z}_i')$ is then used as the *effective fitness estimate* $\overline{fit}(\vec{Z}_i')$ [7-9]. The accuracy of the estimate $\overline{fit}(\vec{Z}_i')$ is ascertained from the *fitness variance* $V(\vec{Z}_i')$ capturing the spread of $n(\vec{Z}_i')$ samples of $fit(\vec{Z}_i')$ away from the fitness estimate $\overline{fit}(\vec{Z}_i')$ [7-9]. The evaluations of effective fitness estimate $\overline{fit}(\vec{Z}_i')$ and fitness variance $V(\vec{Z}_i')$ are omitted here due to space constraint. Interested readers may refer to [7-9] for the necessary background.

Evidently, proper selection of sample size is an essential step of noisy optimization problem to ensure accurate fitness measures of population members. Authors' previous works [7-9] are concerned with adaptive sampling strategy where each candidate solution $\vec{Z}_i'$ is assigned with a sample size $n(\vec{Z}_i')$ based on noise-contamination level in its *local neighborhood* (LN). As the true distribution of noise in the fitness landscape is unknown, the fitness variance $\vartheta(\vec{Z}_i')$ of the solutions in the LN of $\vec{Z}_i'$ is used to model the noise-contamination level in the respective LN. $\vartheta(\vec{Z}_i')$ is referred to as *local neighbourhood fitness variance* (LNFV) of $\vec{Z}_i'$. Intuitively, a large (or a small) $\vartheta(\vec{Z}_i')$ indicates a large (or small) degree of intrusion of noise in the LN of $\vec{Z}_i'$ demanding a large (or small) sample size $n(\vec{Z}_i')$. This in turn balances the trade-off between the degree of accuracy in the fitness estimate and the runtime.

The sample size $n(\vec{Z}_i')$ is modeled as a monotonically non-decreasing function of $\vartheta(\vec{Z}_i')$ in the existing works [7-9]. Symbolically,

$$n(\vec{Z}_i') = g(\vartheta(\vec{Z}_i')) . \quad (5)$$

Here $g(.)$ denotes the monotonically non-decreasing function used to predict sample size $n(\vec{Z}_i')$ based on LNFV $\vartheta(\vec{Z}_i')$. Different forms of $g(.)$ have been adopted in [7-9].

The existing approaches suffer from two limitations.

First, the methods predict sample size of an offspring $\vec{Z}_i'$ based on its LNFV only, ignoring the varying convexity of the fitness landscape of its LN. Specifically, for a maximization problem, if $\vec{Z}_i'$ falls in an LN with large fitness estimate, a large $n(\vec{Z}_i')$ is required to guarantee the true quality of $\vec{Z}_i'$. Otherwise, a small value of $n(\vec{Z}_i')$ is used to reduce the runtime overhead. Hence, the first aim of the present work is to predict sample size $n(\vec{Z}_i')$ of $\vec{Z}_i'$ jointly based on its LNFV $\vartheta(\vec{Z}_i')$ and *local neighbourhood fitness estimate* (LNFE) $\mu(\vec{Z}_i')$. $\vartheta(\vec{Z}_i')$ and $\mu(\vec{Z}_i')$ jointly capture the fitness profile of the LN of $\vec{Z}_i'$. Mathematically,

$$n(\vec{Z}_i') = g(\vartheta(\vec{Z}_i'), \mu(\vec{Z}_i')) . \quad (6)$$

Second, the judicious selection of the functional form $g(.)$ greatly influences the efficacy of the sampling strategy. A specific functional form of $g(.)$ being biased to the nature of stochastic distribution of noise in the fitness landscape, may not always predict the appropriate sample size required for all possible stochastic distribution of noise. This gridlock has been overcome here by employing TDQL-induced adaptive sampling policy which detours the need for a specific functional form of $g(.)$.

IV. Proposed Method

This section elaborates the steps used to employ the TDQL-induced adaptive sampling policy.

1. Local Neighborhood Formation

The first step is concerned with dividing the search space into a number of local neighborhoods (LNs). First, for each candidate solution, we compute the following composite measure

$$\phi(\vec{Z}_i) = \left(1 - \frac{\overline{fit}(\vec{Z}_i)}{\sum_{j=1}^{S} \overline{fit}(\vec{Z}_j)}\right) \times \frac{\vartheta(\vec{Z}_i)}{\sum_{j=1}^{S} \vartheta(\vec{Z}_j)} \quad (7)$$

for i = 1, 2, …, S. For a maximization problem, more the fitness measure $\overline{fit}(.)$ and small the fitness variance $\vartheta(.)$, small is the composite measure $\varphi(.)$. The entire population is then sorted in ascending order of their respective composite measures. The candidate at the top most position of the sorted population **P** is selected as the first local guide $\vec{Z}^{1,lg}$ of the first LN $\boldsymbol{\psi}_1$. The LN $\boldsymbol{\psi}_1$ with local guide $\vec{Z}^{1,lg}$ is formed by the sub-population lying within the hyperspace bounded by $[\Delta\vec{Z}^{1,l}, \Delta\vec{Z}^{1,h}]$, where

$$\Delta\vec{Z}^{1,l} = \{z_1^{1,lg} - \Delta z_1, z_2^{1,lg} - \Delta z_2, ..., z_D^{1,lg} - \Delta z_D\} \quad (8.a)$$

$$\Delta\vec{Z}^{1,h} = \{z_1^{1,lg} + \Delta z_1, z_2^{1,lg} + \Delta z_2, ..., z_D^{1,lg} + \Delta z_D\} \quad (8.b)$$

and $\Delta z_j = (z_j^h - z_j^l)/S$ for j = 1, 2, …, D. (8.c)

Here, z_j^l and z_j^h respectively denote the minimum and maximum search boundary along the j-th dimension for j = 1, 2, …, D. To construct the second LN $\boldsymbol{\psi}_2$, the solution at the top most position of the sorted list {**P**−$\boldsymbol{\psi}_1$} is recorded as the second local guide $\vec{Z}^{2,lg}$. The subordinates of $\vec{Z}^{2,lg}$ lying within $[\Delta\vec{Z}^{2,l}, \Delta\vec{Z}^{2,h}]$ (following (8)) are included in the second LN $\boldsymbol{\psi}_2$.

This is repeated until each member of **P** is assigned to an LN and finally, the D-dimensional search space is segmented into a number of LNs. Let the number of LNs thus identified in the current generation be denoted by C.

2. Fitness Estimate and Fitness Variance in Local Neighborhood

The fitness profile of each LN (discovered in the last step) here is characterized by the effective fitness estimate and the fitness variance of their constituent population members, however, with different weights. Let $\vec{Z}_l^j$ be the l-th member of the j-th LN $\boldsymbol{\psi}_j$ with its effective fitness estimate $\overline{fit}(\vec{Z}_l^j)$ for l = 1, 2, …, $|\boldsymbol{\psi}_j|$. Moreover, let ξ_l^j be the average of effective fitness estimates of all population members of $\boldsymbol{\psi}_j$, however excluding $\overline{fit}(\vec{Z}_l^j)$. Mathematically,

$$\xi_l^j = \frac{\sum_{k=1, k\neq l}^{|\boldsymbol{\psi}_j|} \overline{fit}(\vec{Z}_k^j)}{|\boldsymbol{\psi}_j| - 1} \quad (9)$$

for l = 1, 2, …, $|\boldsymbol{\psi}_j|$. Next, we determine the weight of $\overline{fit}(\vec{Z}_l^j)$, given by

$$w_l = \exp\left(-\left|\overline{fit}(\vec{Z}_l^j) - \xi_l^j\right|\right). \quad (10)$$

The design philosophy is based on the supposition that larger the value of $\left|\overline{fit}(\vec{Z}_l^j) - \xi_l^j\right|$, greater is the degree of disagreement between effective fitness estimate of $\vec{Z}_l^j$ and other members of LN $\boldsymbol{\psi}_j$, indicating possibility of $\overline{fit}(\vec{Z}_l^j)$ to be noisy. Thus its significance towards calculation of LNFE of the entire LN $\boldsymbol{\psi}_j$ must be reduced by assigning a small weight w_l. It is apparent from (10) that with an increase in $\left|\overline{fit}(\vec{Z}_l^j) - \xi_l^j\right|$, w_l tends to be reduced to zero. After evaluating the weights $0 < w_l < 1$ for all members of $\boldsymbol{\psi}_j$ for l = 1, 2, …, $|\boldsymbol{\psi}_j|$, we obtain the *local neighborhood fitness estimate* (LNFE) of LN $\boldsymbol{\psi}_j$, say μ_j, as

$$\mu_j = \sum_{l=1}^{|\boldsymbol{\psi}_j|} \left(w_l \times \overline{fit}(\vec{Z}_l^j)\right) \Big/ \sum_{l=1}^{|\boldsymbol{\psi}_j|} w_l \,. \quad (11)$$

Finally, the *local neighborhood fitness variance* (LNFV) of LN $\boldsymbol{\psi}_j$ is computed following (12).

$$\vartheta_j = \left(\sum_{l=1}^{|\boldsymbol{\psi}_j|} w_l (\overline{fit}(\vec{Z}_l^j) - \mu_j)^2\right) \Big/ \left(\sum_{l=1}^{|\boldsymbol{\psi}_j|} w_l\right) \,. \quad (12)$$

Once we obtain μ_j and ϑ_j for all neighborhoods j = 1, 2, …, C in a particular generation, the normalized LNFE $\mu_{j\text{-}norm} \in [0, 1]$ and LNFV $\vartheta_{j\text{-}norm} \in [0, 1]$ of $\boldsymbol{\psi}_j$ are computed using (13) and (14) respectively for j = 1, 2, …, C.

$$\mu_{j-norm} = \mu_j \Big/ \sum_{k=1}^{C} \mu_k \quad (13)$$

$$\vartheta_{j-norm} = \vartheta_j \Big/ \sum_{k=1}^{C} \vartheta_k \quad (14)$$

3. Design of Q-Table

The proposed adaptive sampling policy accesses a three-dimensional Q-table to determine sample size of a solution from the pool {2, 3, …, N} where N denotes the maximum sample size. The solution is assigned to a state-pair in the Q-table based on its normalized LNFE and LNFV. The first dimension of the Q-table represents the possible set of M states of population members based on their normalized LNFE μ_{norm}, denoted by {$\tau_1(1)$, $\tau_1(2)$, …, $\tau_1(M)$}. Similarly, the second dimension of the Q-table {$\tau_2(1)$, $\tau_2(2)$, …, $\tau_2(L)$} encodes L possible states of the candidate solutions based on their normalized LNFV ϑ_{norm}. The columns of the Q-table denote uniformly quantized values of the sample size {2, 3, …, N}, to be used for the periodic fitness evaluation of a solution.

The state $\tau_1(m) \in \{\tau_1(1), \tau_1(2), \ldots, \tau_1(M)\}$ represents normalized LNFEs in the range $[(m-1)/M, m/M)$ while the state $\tau_2(l) \in \{\tau_2(1), \tau_2(2), \ldots, \tau_2(L)\}$ symbolizes normalized LNFVs in the range $[(l-1)/L, l/L)$. In other words, a candidate solution $\vec{Z}$, belonging to the LN $\boldsymbol{\psi}_j$, is assigned to a state-pair

$<\tau_1(m), \tau_2(l)>$ if the corresponding normalized LNFE $\mu_{j\text{-}norm} \in [(m-1)/M, m/M)$ and the respective normalized LNFV $\vartheta_{j\text{-}norm} \in [(l-1)/L, l/L)$ for $m \in \{1, 2, \ldots, M\}$ and $l \in \{1, 2, \ldots, L\}$. The component $Q(<\tau_1(m), \tau_2(l)>, \alpha)$ of the Q-table denotes the Q-value correponding to a sample size $\alpha \in \{2, 3, \ldots, N\}$ at a specific state-pair $<\tau_1(m), \tau_2(l)>$ for $m \in \{1, 2, \ldots, M\}$ and $l \in \{1, 2, \ldots, L\}$.

4. Identification of Local Neihborhood of an Offspring

Next, the employed/onlooker bee phase of ABC is employed to generate an offspring $\vec{Z}_i'$ from its respective parent $\vec{Z}_i$. To determine sample size for periodic fitness evaluation of $\vec{Z}_i'$, its LN $\boldsymbol{\psi}_j$ is identified for $k \in \{1, 2, \ldots, C\}$, by finding out the local guide $\vec{Z}^{k,lg}$ with the minimum distance from $\vec{Z}_i'$ among all C local guides.

5. State-pair Assignment of a Trial Vector

The normalized LNFE $\mu_{k\text{-}norm}$ and LNFV $\vartheta_{k\text{-}norm}$ are used to assign $\vec{Z}_i'$ to a specific state-pair in the Q-table. If $\mu_{k\text{-}norm} \in [(m-1)/M, m/M)$ and $\vartheta_{k\text{-}norm} \in [(l-1)/L, l/L)$, $\vec{Z}_i'$ is assigned to the state-pair $<\tau_1(m), \tau_2(l)>$ for $m \in \{1, 2, \ldots, M\}$ and $l \in \{1, 2, \ldots, L\}$.

6. Adaptive Selection of Sample Size using Temporal Difference Q-Learning

Once $\vec{Z}_i'$ is assigned to a state-pair $<\tau_1(m), \tau_2(l)>$, for $m \in \{1, 2, \ldots, M\}$ and $l \in \{1, 2, \ldots, L\}$, the TDQL is employed to select a sample size from $\{2, 3, \ldots, N\}$ based on the Q-values $Q(<\tau_1(m), \tau_2(l)>, \alpha)$ for $\alpha = 2, 3, \ldots, N$. The reward and penalty obtained during the selection of sample size α at state-pair $<\tau_1(m), \tau_2(l)>$ in the previous generations are used to adapt the Q-values $Q(<\tau_1(m), \tau_2(l)>, \alpha)$ in the subsequent generations. If $Q(<\tau_1(m), \tau_2(l)>, \alpha) \geq Q(<\tau_1(m), \tau_2(l)>, \alpha')$ for $\alpha' = 2, 3, \ldots, N$, it signifies that the selection of particular sample size α was rewarded many times before in the evolution process. The learning experience thus selects the sample size $n(\vec{Z}_i') = \alpha$ with the highest Q-value. The probability of selection of $n(\vec{Z}_i') = \alpha$ from the pool $\{2, 3, \ldots, N\}$ at the state-pair $<\tau_1(m), \tau_2(l)>$ is given by

$$p(m,l,\alpha) = \frac{Q(\langle \tau_1(m), \tau_2(l) \rangle, \alpha)}{\sum_{\alpha'=2}^{N} Q(\langle \tau_1(m), \tau_2(l) \rangle, \alpha')} \quad (15)$$

for $m = 1, 2, \ldots, M$ and $l = 1, 2, \ldots, L$. To maintain adaptation and learning in all Q's present in all possible state-pairs, the Roulette-choice strategy [19] is employed for selection of sample size. The strategy selects a sample size α from the pool $\{2, 3, \ldots, N\}$ for an offspring $\vec{Z}_i'$ at state-pair $<\tau_1(m), \tau_2(l)>$ if the following condition holds where rand(0, 1) denotes a random number uniformly distributed in (0, 1).

$$\sum_{\alpha'=1}^{\alpha-1} p(m,l,\alpha') \leq \text{rand}(0,1) \leq \sum_{\alpha'=1}^{\alpha} p(m,l,\alpha') \quad (16)$$

7. State-Transition

The sample size $n(\vec{Z}_i') = \alpha$ selected for $\vec{Z}_i'$ in the last step is used for its periodic fitness evaluation, effective fitness estimate $\overline{fit}(\vec{Z}_i')$ and fitness variance $V(\vec{Z}_i')$, using the approaches proposed in [7-9]. The evaluated $\overline{fit}(\vec{Z}_i')$ and $V(\vec{Z}_i')$ are now used to evaluate the new normalized LNFE and LNFV of $\boldsymbol{\psi}_k$, the LN of $\vec{Z}_i'$, using (11)-(14). Let the currently obtained values of normalized LNFE and LNFV be denoted by μ_{k-norm}^{cur} and ϑ_{k-norm}^{cur}. If $\mu_{k-norm}^{cur} \in [(x-1)/M, x/M)$ and $\vartheta_{k-norm}^{cur} \in [(y-1)/L, y/L)$, for $x \in \{1, 2, \ldots, M\}$ and $y \in \{1, 2, \ldots, L\}$, then $\vec{Z}_i'$ is assigned to the new state-pair $<\tau_1(x), \tau_2(y)>$ from its old state-pair $<\tau_1(m), \tau_2(l)>$. Symbolically,

$$\delta(\langle \tau_1(m), \tau_2(l) \rangle, \alpha) = \langle \tau_1(x), \tau_2(y) \rangle \quad (17)$$

8. Reward and Penalty based Update of Q-Table

If $\mu_{k-norm}^{cur} > \mu_{k-norm}$ (for maximization problem) or $\vartheta_{k-norm}^{cur} < \vartheta_{k-norm}$, the selection of sample size $\alpha \in 2, 3, \ldots, N$ at the state-pair $<\tau_1(m), \tau_2(l)>$ is rewarded with the respective reward, given below.

If $\mu_{k-norm}^{cur} > \mu_{k-norm}$ and $\vartheta_{k-norm}^{cur} \geq \vartheta_{k-norm}$

$$r(\langle \tau_1(m), \tau_2(l) \rangle, \alpha) = \mu_{k-norm}^{cur} - \mu_{k-norm}. \quad (18.a)$$

If $\mu_{k-norm}^{cur} \leq \mu_{k-norm}$ and $\vartheta_{k-norm}^{cur} < \vartheta_{k-norm}$

$$r(\langle \tau_1(m), \tau_2(l) \rangle, \alpha) = \vartheta_{k-norm} - \vartheta_{k-norm}^{cur}. \quad (18.b)$$

If $\mu_{k-norm}^{cur} > \mu_{k-norm}$ and $\vartheta_{k-norm}^{cur} < \vartheta_{k-norm}$

$$r(\langle \tau_1(m), \tau_2(l) \rangle, \alpha) = (\mu_{k-norm}^{cur} - \mu_{k-norm}) + (\vartheta_{k-norm} - \vartheta_{k-norm}^{cur}). \quad (18.c)$$

If $\mu_{k-norm}^{cur} \leq \mu_{k-norm}$ and $\vartheta_{k-norm}^{cur} \geq \vartheta_{k-norm}$, then the selection is penalized with

$$r(\langle \tau_1(m), \tau_2(l) \rangle, \alpha) = -K \quad (19)$$

with K as a positive constant value, however, small.

The corresponding Q-value is then updated with

$$Q(\langle \tau_1(m), \tau_2(l) \rangle, \alpha) \leftarrow (1-\beta) \times Q(\langle \tau_1(m), \tau_2(l) \rangle, \alpha) + \beta \times (r(\langle \tau_1(m), \tau_2(l) \rangle, \alpha) + \gamma \max_{\alpha'} Q(\langle \tau_1(x), \tau_2(y) \rangle, \alpha') \quad (20)$$

Update the normalized LNFE and LNFV of $\boldsymbol{\psi}_k$ as

$$\mu_{k-norm} \leftarrow \mu_{k-norm}^{cur} \text{ and } \vartheta_{k-norm} \leftarrow \vartheta_{k-norm}^{cur}. \quad (21)$$

V. Experiments and Results

A. Benchmark Functions and Performance Metric

Noisy versions of 28 single objective benchmark functions proposed in CEC'2013 [26] are used here to compare the performance of the QLNBC algorithm with its contenders in

the presence of noise (in the fitness landscape) following Gaussian [27], Poisson [28], Rayleigh [29], exponential [30] and random [31] distribution. *Function error value* (FEV) is considered as the performance metric for the comparative analysis. For a particular benchmark function *f*, FEV is represented by

$$FEV(f) = \left| f^* - f^{med} \right| \tag{22}$$

where f^{med} denotes the median of the best fitness values obtained by an algorithm over R runs (say 50) and f^* represents the true optimum value of *f*. Smaller the FEV, better is the performance of an algorithm.

B. Comparative Framework

The comparative framework is formed by considering a few widely popular noisy single objective optimization algorithms (NSOOAs), including *learning automata induced noisy bee colony* (LANBC) [21], *memetic for uncertainties DE* (MUDE) [22], *subset based LA incorporated particle swarm optimization* (LAPSO) [23], and *noise tolerant genetic algorithm* (NTGA) [24]. In the present work, the sampling strategy of *extended* LANBC [25] is replaced with the proposed TDQL induced adaptive sampling policy while keeping other strategies embedded in *extended* LANBC unchanged. The new contender algorithm thus developed is referred to as *extended* QLNBC henceforth and is regarded as new member of the comparative framework.

C. Results and Performance Analysis

Comparative performance analysis of the proposed algorithms (QLNBC and *extended* QLNBC) with their contenders is undertaken in this section. The performance analysis is carried out for all possible noise distribution in the fitness landscape. However, a few results are reported for space restriction.

The median FEV values obtained by all seven contenders over 50 independent runs are reported in Table-I with corresponding interquartile range (within parenthesis). Noise samples taken from zero mean Gaussian distribution of variance 0.57 are used to additively contaminate the benchmark functions to test the accuracy of the algorithms. Statistical significance of superiority of one algorithm over others is assessed by Wilcoxon rank sum test [32] with a significance level of 0.05. The test is undertaken between the best algorithm (providing the minimum median FEV value) and the rest and the respective *p*-value thus obtained is reported in the third bracket for each benchmark instance. Here NA denotes *not applicable* cases of comparing the best algorithm with itself. The null hypothesis is concerned with statistically equivalent performance of the best algorithm and each of the remaining contenders. The null hypothesis is rejected on obtaining the corresponding *p*-value less than 0.05.

It is evident from Table-I that *extended* QLNBC outperforms its contenders in 19 cases out of 28 benchmark functions. Out of these 19 cases, 17 cases reveal statistically significant superiority of the *extended* QLNBC to its contenders. In case of f10 and f15, *extended* QLNBC marginally outperforms *extended* LANBC, though not significantly. However, *extended* QLNBC is outperformed by *extended* LANBC in case of f13 and f23. QLNBC supersedes its extended version for f20. The performance difference is however statistically insignificant. For f01-f03, f17, f19 and f26, both *extended* QLNBC and *extended* LANBC achieve the same accuracy. The interquartile range obtained by *extended* LANBC for f17 and f19 is however more than that of *extended* QLNBC. Similar performance of QLNBC and its extended counterpart is also noted for f01 and f02.

TABLE I-A: FEVS OF NSOOAS IN PRESENCE OF ZERO MEAN GAUSSIAN NOISE (OF VARIANCE =0.57) IN FITNESS LANDSCAPE f01 TO f08

Functions	Ext. QLNBC	Ext. LANBC	QLNBC	LANBC	MUDE	LAPSO	NTGA
f01	**0.00e+000 (0.00e+000) NA**	**0.00e+000 (0.00e+000) NA**	**0.00e+000 (0.00e+000) NA**	**0.00e+000 (0.00e+000) NA**	1.89e-007 (1.25e-008) [3.86e-005]	2.13e-007 (3.31e-008) [1.94e-005]	3.95e-007 (6.16e-008) [4.30e-006]
f02	**0.00e+000 (0.00e+000) NA**	**0.00e+000 (0.00e+000) NA**	**0.00e+000 (0.00e+000) NA**	5.67e-004 (5.56e+001) [1.45e-003]	7.95e-004 (1.24e+002) [3.91e-004]	1.27e-003 (9.09e+002) [5.48e-006]	1.61e-003 (1.10e+004) [2.28e-006]
f03	**0.00e+000 (0.00e+000) NA**	**0.00e+000 (0.00e+000) NA**	7.21e-003 (8.33e+001) [4.86e-002]	7.90e-003 (1.64e+002) [4.83e-002]	1.08e-002 (2.16e+002) [4.50e-002]	6.69e-002 (2.51e+002) [1.53e-002]	1.48e-001 (2.68e+002) [6.47e-003]
f04	**5.17e-005 (2.99e-005) NA**	1.05e-004 (7.83e-005) [3.01e-003]	3.62e-004 (5.22e-004) [2.67e-003]	3.74e-003 (6.30e-004) [2.41e-003]	5.01e-003 (9.01e-004) [3.04e-004]	6.84e-003 (1.13e-001) [1.23e-004]	1.26e-002 (1.64e+000) [2.73e-005]
f05	**0.00e+000 (0.00e+000) NA**	1.51e-008 (4.82e-004) [4.74e-005]	9.05e-008 (1.34e-003) [1.67e-005]	1.99e-006 (3.48e-003) [4.61e-007]	2.16e-006 (3.84e-003) [4.24e-007]	1.47e-005 (4.55e-003) [4.09e-007]	5.16e-005 (5.61e-003) [1.14e-007]
f06	**1.55e+001 (6.65e-006) NA**	2.86e+001 (7.48e-002) [1.43e-003]	3.10e+001 (1.45e+000) [1.23e-004]	3.83e+001 (3.02e+000) [4.10e-005]	4.05e+001 (6.46e+000) [1.95e-006]	4.08e+001 (7.38e+000) [1.56e-006]	4.27e+001 (9.04e+000) [2.30e-007]
f07	**3.71e+000 (3.25e+000) NA**	3.43e+001 (5.72e+000) [1.10e-003]	3.56e+001 (6.69e+000) [3.17e-005]	3.75e+001 (9.16e+000) [2.40e-005]	4.48e+001 (9.18e+000) [1.33e-006]	4.78e+001 (9.71e+000) [7.97e-006]	6.37e+001 (1.10e+001) [7.52e-007]
f08	**2.08e+001 (2.96e-002) NA**	2.09e+001 (3.58e-002) [3.56e-002]	2.10e+001 (4.23e-002) [1.47e-002]	2.10e+001 (4.64e-002) [1.03e-002]	2.10e+001 (4.65e-002) [5.38e-003]	2.10e+001 (5.19e-002) [5.15e-003]	2.10e+001 (6.20e-002) [4.93e-003]

TABLE I-B: FEVS OF NSOOAS IN PRESENCE OF ZERO MEAN GAUSSIAN NOISE (OF VARIANCE =0.57) IN FITNESS LANDSCAPE f09 TO f28

Functions	**Ext. QLNBC**	Ext. LANBC	QLNBC	LANBC	MUDE	LAPSO	NTGA
f09	**1.16e+001** **(1.27e+000)** **NA**	1.66e+001 (2.27e+000) [5.94e-002]	3.40e+001 (2.29e+000) [5.57e-002]	4.37e+001 (2.46e+000) [5.07e-002]	4.90e+001 (2.86e+000) [4.18e-002]	4.98e+001 (3.33e+000) [1.48e-002]	5.05e+001 (3.60e+000) [9.40e-004]
f10	**5.14e-008** **(4.12e-005)** **NA**	5.35e-004 (7.22e-002) [5.10e-002]	7.55e-004 (1.19e-003) [2.36e-004]	5.14e-003 (3.64e-003) [3.02e-006]	1.25e-002 (1.15e-002) [2.59e-006]	1.75e-002 (1.28e-002) [3.22e-006]	2.57e-002 (1.37e-002) [3.03e-007]
f11	**7.88e-001** **(1.27e-003)** **NA**	3.27e+000 (4.21e-002) [2.97e-003]	5.50e+000 (9.88e-002) [1.26e-003]	5.74e+000 (1.21e-001) [3.42e-004]	7.43e+000 (3.21e-001) [3.35e-004]	1.54e+001 (1.92e+000) [9.09e-005]	1.69e+001 (2.59e+000) [3.95e-006]
f12	**9.72e+000** **(1.86e+000)** **NA**	2.69e+001 (2.78e+000) [3.41e-004]	4.20e+001 (7.71e+000) [2.50e-004]	6.63e+001 (9.46e+000) [1.86e-004]	8.55e+001 (2.09e+001) [2.47e-005]	9.65e+001 (2.27e+001) [2.02e-006]	1.32e+002 (3.26e+001) [3.05e-007]
f13	8.05e+001 (7.92e+000) [9.93e-003]	**4.55e+001** **(2.68e+000)** **NA**	9.26e+001 (1.89e+001) [9.78e-003]	1.41e+002 (1.99e+001) [2.20e-004]	1.48e+002 (2.27e+001) [1.79e-004]	1.84e+002 (3.07e+001) [2.99e-004]	1.96e+002 (3.37e+001) [2.78e-005]
f14	**1.37e+001** **(5.73e+000)** **NA**	2.06e+001 (8.55e+000) [4.91e-003]	2.75e+001 (1.03e+001) [3.79e-003]	2.19e+002 (7.45e+001) [1.57e-004]	5.38e+002 (1.83e+002) [1.46e-005]	7.48e+002 (2.65e+002) [4.52e-006]	7.90e+002 (4.48e+002) [7.25e-006]
f15	**1.09e+003** **(2.83e+002)** **NA**	2.18e+003 (3.67e+002) [5.47e-002]	2.28e+003 (4.32e+002) [4.39e-002]	4.36e+003 (4.38e+002) [6.97e-003]	6.35e+003 (5.31e+002) [4.97e-004]	6.47e+003 (5.51e+002) [5.84e-005]	7.42e+003 (6.30e+002) [3.89e-006]
f16	**3.84e-001** **(3.21e-002)** **NA**	7.84e-001 (4.23e-002) [1.26e-003]	1.07e+000 (1.68e-001) [5.16e-004]	1.46e+000 (1.96e-001) [4.89e-004]	1.63e+000 (2.03e-001) [3.22e-004]	1.72e+000 (2.35e-001) [5.01e-005]	2.12e+000 (2.81e-001) [5.33e-006]
f17	**4.36e+001** **(2.58e-003)** **NA**	**4.36e+001** **(3.17e-002)** **NA**	5.07e+001 (4.10e-001) [6.95e-003]	5.40e+001 (1.12e+000) [9.15e-004]	5.65e+001 (1.77e+000) [5.02e-005]	6.55e+001 (2.37e+000) [3.47e-005]	7.37e+001 (4.29e+000) [2.84e-007]
f18	**6.60e+001** **(1.17e+001)** **NA**	1.09e+002 (1.33e+001) [1.27e-002]	1.26e+002 (1.61e+001) [5.24e-002]	1.30e+002 (2.87e+001) [6.73e-004]	1.48e+002 (3.33e+001) [2.10e-004]	1.53e+002 (3.64e+001) [2.77e-004]	1.67e+002 (4.48e+001) [2.58e-005]
f19	**2.13e+000** **(2.91e-001)** **NA**	**2.13e+000** **(3.51e-001)** **NA**	3.38e+000 (4.69e-001) [1.26e-004]	3.42e+000 (4.89e-001) [3.20e-005]	3.84e+000 (6.74e-001) [3.39e-006]	3.89e+000 (7.24e-001) [1.08e-006]	4.33e+000 (7.96e-001) [8.63e-007]
f20	1.89e+001 (4.44e-001) [5.54e-002]	1.91e+001 (5.39e-001) [3.91e-003]	**1.40e+001** **(3.86e-001)** **NA**	1.91e+001 (5.84e-001) [3.41e-004]	1.94e+001 (6.55e-001) [2.75e-004]	2.01e+001 (7.85e-001) [2.01e-004]	2.05e+001 (7.96e-001) [1.36e-005]
f21	**2.28e+002** **(1.07e+001)** **NA**	2.38e+002 (4.03e+001) [3.03e-004]	2.45e+002 (1.83e+002) [1.01e-004]	2.96e+002 (2.27e+002) [6.27e-005]	3.72e+002 (2.73e+002) [5.42e-005]	5.08e+002 (2.88e+002) [1.17e-006]	6.30e+002 (3.25e+002) [8.20e-007]
f22	**2.43e+001** **(1.46e+001)** **NA**	2.79e+001 (2.57e+001) [2.62e-003]	4.37e+001 (4.07e+001) [1.01e-003]	2.79e+002 (4.93e+001) [6.66e-004]	3.38e+002 (2.66e+002) [6.49e-005]	5.70e+002 (3.55e+002) [3.03e-007]	5.97e+002 (3.61e+002) [1.41e-007]
f23	1.85e+003 (3.63e+002) [1.48e-002]	**8.04e+002** **(3.26e+002)** **NA**	2.91e+003 (5.35e+002) [1.07e-003]	6.51e+003 (5.52e+002) [1.11e-004]	7.23e+003 (6.52e+002) [1.00e-005]	8.25e+003 (7.36e+002) [2.41e-006]	8.79e+003 (8.98e+002) [3.16e-007]
f24	**2.23e+002** **(2.23e+000)** **NA**	2.26e+002 (5.53e+000) [5.60e-003]	2.36e+002 (9.34e+000) [5.53e-003]	2.80e+002 (1.18e+001) [1.25e-003]	2.82e+002 (1.40e+001) [3.75e-005]	2.84e+002 (1.70e+001) [2.21e-005]	3.07e+002 (1.95e+001) [4.29e-007]
f25	**2.25e+002** **(3.97e+000)** **NA**	3.01e+002 (8.01e+000) [4.89e-005]	3.05e+002 (1.01e+001) [4.60e-005]	3.14e+002 (1.05e+001) [2.98e-005]	3.26e+002 (1.21e+001) [9.21e-006]	3.37e+002 (1.47e+001) [2.77e-006]	3.60e+002 (1.83e+001) [7.83e-007]
f26	**2.28e+001** **(3.68e+000)** **NA**	**2.28e+001** **(3.68e+000)** **NA**	1.92e+002 (2.38e+001) [5.17e-003]	2.40e+002 (3.83e+001) [2.47e-004]	2.49e+002 (5.88e+001) [2.07e-005]	2.65e+002 (6.71e+001) [2.05e-006]	2.78e+002 (9.08e+001) [9.50e-007]
f27	**3.91e+002** **(3.85e+001)** **NA**	6.87e+002 (9.80e+001) [4.15e-003]	9.19e+002 (1.36e+002) [2.42e-003]	9.27e+002 (1.38e+002) [2.61e-004]	9.56e+002 (1.39e+002) [1.37e-005]	1.12e+003 (1.76e+002) [9.96e-006]	1.38e+003 (1.88e+002) [1.93e-006]
f28	**3.83e+002** **(1.65e-004)** **NA**	4.22e+002 (3.55e-001) [4.11e-003]	4.30e+002 (2.03e+001) [2.20e-003]	4.38e+002 (8.67e+001) [1.55e-003]	4.58e+002 (3.57e+002) [5.97e-004]	4.58e+002 (4.17e+002) [2.85e-005]	5.55e+002 (4.36e+002) [3.90e-005]

VI. CONCLUSION

The paper proposes a novel approach to utilize the benefit of the reinforcement learning (RL) technique to judiciously assign sample size to solutions for their periodic fitness evaluation in a noisy fitness landscape. The sample size is determined based on the noisy fitness profile in the surrounding of the solutions. The fitness estimate and fitness variance of a sub-population surrounding a solution are used to capture the fitness profile in the local neighborhood of the given solution. The proposed approach detours the requirement of any specific functional form to model the relationship between the sample size of a solution and its noisy local fitness profile. TDQL is selected here as a RL mechanism to assign a small (and a large) sample size to a solution in a more (and a less) noisy zone, captured by its LNFE and LNFV.

A relative comparison of the proposed algorithms (QLNBC and *extended* QLNBC) with five state-of-the-art algorithms reveals that the proposed algorithms outperform most of their competitors with respect to FEV metric. Noisy versions of a set of 28 CEC'2013 benchmark functions have been used to arrive at the foregoing conclusions. Additive noise samples taken from Gaussian, Poisson, Rayleigh, exponential and random distributions are used to realize the noisy versions of the benchmark functions. The Wilcoxon rank-sum test, the Friedman and the Iman-Davenport non-parametric tests are undertaken to affirm the statistical significance of the reported results.

REFERENCES

[1] P. Rakshit, A. Konar, and S. Das, "Noisy Evolutionary Optimization algorithms-A comprehensive survey," *Swarm and Evolutionary Computation*, vol. 33, 2017, pp. 18-45.

[2] Y. Jin, and J. Branke, "Evolutionary optimization in uncertain environments-a survey," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 3, 2005, pp. 303-317.

[3] A. D. Pietro, *Optimising evolutionary strategies for problems with varying noise strength*, Ph. D. Thesis, University of Western Australia, 2007.

[4] F. Siegmund, A. H. C. Ng, and K. Deb, "Hybrid dynamic resampling for guided evolutionary multi-objective optimization," in *Proceedings of Evolutionary Multi-Criterion Optimization*, Springer International Publishing, 2015, pp. 366-380.

[5] G. Iacca, F. Neri, and E. Mininno, "Noise analysis compact differential evolution," *International Journal of Systems Science*, vol. 43, no. 7, 2012, pp. 1248-1267.

[6] C. H. Chen, J. Lin, E. Yücesan, and S. E. Chick, "Simulation budget allocation for further enhancing the efficiency of ordinal optimization," *Discrete Event Dynamic Systems*, vol. 10, no. 3, 2000, pp .251-270.

[7] P. Rakshit, A. Konar, S. Das, L. C. Jain, and A. K. Nagar, "Uncertainty management in differential evolution induced multiobjective optimization in presence of measurement noise," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 44, no. 7, 2014, pp. 922-937.

[8] P. Rakshit, and A. Konar, "Differential evolution for noisy multiobjective optimization," *Artificial Intelligence*, vol. 227, 2015, 165-189.

[9] P. Rakshit, and A. Konar, "Extending multi-objective differential evolution for optimization in presence of noise," *Information Sciences*, vol. 305, 2015, pp. 56-76.

[10] H. Kita, and Y. Sano, "Genetic algorithms for optimization of uncertain functions and their applications," in *Proceedings of SICE Annual Conference*, vol. 3, 2003, pp. 2744-2749.

[11] L. T. Bui, H. A. Abbass, and D. Essam, "Fitness inheritance for noisy evolutionary multi-objective optimization," in *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation*, ACM, 2005, pp. 779-785.

[12] K. C. Tan, T. H. Lee, and E. F. Khor, "Evolutionary algorithms with dynamic population size and local exploration for multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 5, no. 6, 2001, pp. 565-588.

[13] H. G. Beyer, D. V. Arnold, and S. M. Nieberg, "A new approach for predicting the final outcome of evolution strategy optimization under noise," *Genetic Programming and Evolvable Machines*, vol. 6, no. 1, 2005, pp. 7-24.

[14] C. K. Goh, and K. C. Tan, "An investigation on noisy environments in evolutionary multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 3, 2007, pp. 354-381.

[15] Rahnamayan S., H. R. Tizhoosh, and M. Salama, "Opposition-based differential evolution for optimization of noisy problems," in *Proceedings of IEEE Congress on Evolutionary Computation*, 2006, pp. 1865-1872.

[16] J. Branke, and C. Schmidt, "Selection in the presence of noise," in *Proceedings of Genetic and Evolutionary Computation*, Springer Berlin Heidelberg, 2003, pp. 766-777.

[17] B. L. Miller, and D. E. Goldberg, "Genetic algorithms, selection schemes, and the varying effects of noise," *Evolutionary Computation*, vol. 4, no. 2, 1996, pp. 113-131.

[18] T. Mitchell, "Machine Learning", *McGraw Hill*, 1997.

[19] P. Rakshit, A. Konar, P. Bhowmik, I. Goswami, S. Das, L. C. Jain, and A. K. Nagar, "Realization of an Adaptive Memetic Algorithm Using Differential Evolution and Q-Learning: A Case Study in Multirobot Path Planning," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 43, no. 4, July, 2013.

[20] B. Basturk., D. Karaboga, "An artificial bee colony (ABC) algorithm for numeric function optimization," in *Proceedings of the IEEE Swarm Intelligence Symposium*, 2006.

[21] P. Rakshit, A. Konar, and A. K. Nagar, "Learning Automata Induced Artificial Bee Colony for Noisy Optimization," in *IEEE Congress on Evolutionary Computation*, 2017, pp. 984 - 991.

[22] E. Mininno, and F. Neri, "A memetic differential evolution approach in noisy optimization," *Memetic Computing*, vol. 2, no. 2, 2010, pp. 111-135.

[23] J. Q.Zhang, L. W. Xu, J. Ma, M. C. Zhou, "A learning automata-based particle swarm optimization algorithm for noisy environment," in *IEEE Congress on Evolutionary Computation*, 2015, pp. 141-147.

[24] H. Jang, R. Choe, and K. R. Ryu, "Deriving a robust policy for container stacking using a noise-tolerant genetic algorithm," in *ACM Research in Applied Computation Symposium*, 2012, pp. 31-36.

[25] P. Rakshit, A. Konar, and A. K. Nagar, "Modified selection and search in learning automata based artificial bee colony in noisy environment," in *IEEE Congress on Evolutionary Computation*, 2019, pp. 3173-3180.

[26] J. J.Liang, B. Y. Qu, P. N. Suganthan, and A. G. H. Díaz, *Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization*, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Nanyang Technological University, Singapore, Technical Report 201212, 2013.

[27] G. E. P. Box, and M. E. Muller, "A note on the generation of random normal deviates," *The annals of mathematical statistics*, vol. 29, 1958, pp. 610-611.

[28] D. E. Knuth, *Seminumerical Algorithms, The art of computer programming*, vol. 2, 1981.

[29] W. Hörmann, J. Leydold, and G. Derflinger, "General principles in random variate generation," in *Automatic Nonuniform Random Variate Generation*, Springer Berlin Heidelberg, 2004, pp. 13-41.

[30] G. Marsaglia, and W. W. Tsang, "The ziggurat method for generating random variables," *Journal of Statistical Software*, vol. 5, no. 8, 2000, pp. 1-7.

[31] J. Bolte, *Linear congruential generators*, Wolfram Demonstrations Project.

[32] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 1, 2011, pp. 3-18.