Type-2 Fuzzy Induced Non-dominated Sorting Bee Colony for Noisy Optimization

Pratyusha Rakshit[1], Amit Konar[2]
Dept. of Electronics & Telecommunication Engineering
Jadavpur University, Kolkata, India.
[1]konaramit@yahoo.co.in, [2]pratyushar1@gmail.com.

Atulya K. Nagar
Dept. of Mathematics and Computer Science
Liverpool Hope University, Liverpool, UK.
nagara@hope.ac.uk.

***Abstract*—A novel multi-objective optimization algorithm is introduced in the paper to proficiently obtain Pareto-optimal solutions in the noisy fitness landscapes. First, a non-linear functional relationship between the fitness variance in the local neighborhood of a trial solution and the sample size for its periodic fitness evaluation is proposed. The second strategy is concerned with determining defuzzified centroidal value of the noisy fitness samples, instead of their conventional averaging, as the effective fitness measure of the trial solutions. Finally, to ensure the diversity of quality solutions in the noisy fitness landscapes, a new selection criterion induced by the crowding distance measure and the distribution pattern of noisy fitness samples is formulated. Experiments undertaken to validate the performance of the extended algorithm affirm its superiority to its contenders with respect to hyper volume ratio, when examined on a test suite of 23 standard benchmarks contaminated with additive noise of five statistical distributions.**

Keywords— non-dominated sorting bee colony; noise-handling; sampling; type-2 fuzzy set; stochastic selection.

I. INTRODUCTION

Multi-Objective Optimization (MOO) refers to jointly optimizing two or more objective functions [1] of a complex physical system. The objective functions usually return unique values, often called fitness, for the variables in their argument. However, in many real world engineering problems, it has been observed that even though the measurements of the variables remain constant, different fitness values are returned by the repeated evaluations of the objective functions at the same set of parameter values because of noise induced dynamic variation of the fitness landscapes. This class of problem is referred to as noisy optimization problem [2]. In noisy MOO, the fitness estimates being noisy, there might be a dismissal of a quality solution from the optimal Pareto front because of its noisy poor fitness estimates, while a deceptive solution with illusive good fitness may be passed on to the next generation [2].

There exist quite a few papers to extend traditional MOO algorithms to capture the Pareto optima in presence of noise in the fitness landscapes [2-8]. The common policy used to deal with the problem is to exercise sampling (fitness re-evaluation of the same trial solution) [3] to improve accuracy in the fitness estimates in presence of noise. But it suffers from the increased cost of the algorithm associated with the estimation of the sample size for repeated evaluations. In [4], a statistical comparator is modeled to permit apparently inferior solutions to enter the optimal Pareto front by comparing the distance between the mean values of the fitness samples of two individuals with the scaled average of their fitness variances. However the constraint of the paper lies in the viewpoint of uniform sampling. In [2], a linear relationship between the sample size of a trial solution and the fitness variance in its local neighborhood is employed for non-uniform sampling. Further statistical expectation of fitness samples is regarded as the effective fitness estimate of the trial solutions. However, the performance of the algorithm declines in case of multimodal fitness landscapes contaminated with noise. Among the other popular works, the introduction of the principles of noise-aware α-dominance [5], noise tolerant dominance dependent lifetime [6], 'soft' selection strategy [7], and surrogate model based approach [8] need special mentioning. The paper amends the existing approaches of noisy MOO by the following counts.

1. The possibility of placement of an inferior solution in the optimal Pareto front is usually diminished by its periodic evaluation (sampling). Traditional approaches utilize fixed sample size, irrespective of the local noise variance of the trial solutions [3], [5]. A monotonic non-decreasing functional form of relationship between the sample size and the fitness variance in the neighborhood (signifying the local noise variance) of a trial solution is exploited with an aim to allow more (less) sample size for trial solutions with larger (smaller) local noise variance.

2. Traditional noisy MOO algorithms estimate fitness of a trial solution by taking the average of the corresponding fitness samples [4]. Unfortunately, the amplitude of noise associated with different fitness samples vary apparently randomly. Hence the uncertainty in the estimation of the respective fitness cannot be efficiently captured by considering equal probability of all the fitness samples. The variation between different fitness samples in an interval in the sample space of an individual trial solution can be regarded as an *intra-personal level uncertainty* which is here handled by an *Interval Type*-2 *Fuzzy Set* [9] based model.

3. The diversity of trial solutions of the selected lowest rank Pareto front is established in traditional MOO algorithms by introducing the concept of crowding distance [1], [2]. However, it may lead to sub-optimal or misleading set of non-dominated solutions in the noisy environment even being accompanied by sampling [4]. This paper considers both crowding distance and a probabilistic estimate of

reliability on the fitness samples of a trial solution for its promotion from the selected lowest rank Pareto front to the next generation. The strategy thus ensures both diversity and quality of solutions in presence of noise.

Experiments have been undertaken to examine the potency of the proposed noisy MOO algorithm realized with Non-dominated Sorting Bee Colony (NSBC) algorithm [10] (hereafter called Noisy NSBC with Stochastic selection-NNSBCS) by contaminating the fitness landscapes by noise samples taken from five different types of distribution, including i) zero mean Gaussian, ii) Poisson, iii) Rayleigh, iv) exponential noise with increasing variance and v) random noise of amplitude within ±25% of the true fitness values. Experiments undertaken reveal that NNSBCS outperforms its competitors [2], [5-8], [11-15] while optimizing noisy versions of 23 recommended benchmark functions [16] with respect to *hyper volume ratio* metric.

The paper is divided into six sections. Section II provides an overview of type-2 fuzzy sets and the traditional NSBC algorithm. Section III provides the noise handling mechanism introduced in NNSBCS. Experimental settings for the benchmarks are explained in section IV. The simulation strategies and experimental results are given in section V. Section VI concludes the paper.

II. PRELIMINARIES

A. Type-2 Fuzzy Sets

Definition 1: A *Type*-1 *fuzzy set A* defined on a universe of discourse *X*, is expressed as a two tuple [9], given by

$$A = \{(x, \mu_A(x)) \mid \forall x \in X\} \quad (1)$$

Here $\mu_A(x)$ known as *membership function* is a crisp number in [0, 1] for a generic element $x \in X$.

Definition 2: A *Type*-2 *fuzzy set* $\tilde{A}$ is described by (2).

$$\tilde{A} = \{((x, \mu_A(x)), \mu_{\tilde{A}}(x, \mu_A(x))) \mid x \in X, \mu_A(x) \in J_x \subseteq [0,1]\} \quad (2)$$

where $\mu_{\tilde{A}}(x, \mu_A(x)) \in [0, 1]$ is known as the *Type*-2 *membership function* corresponding to the generic element $x \in X$ with its primary membership $\mu_A(x) \in J_x \subseteq [0, 1]$.

Definition 3: $\tilde{A}$ is referred to as *Interval Type*-2 *Fuzzy Set* (*IT2FS*) provided

$$\mu_{\tilde{A}}(x, \mu_A(x)) = 1 \;\; \forall x \in X, \; \forall \mu_A(x) \in J_x \subseteq [0,1] \quad (3)$$

Definition 4: The *Footprint of Uncertainty* (FOU), representing the uncertainty in the primary memberships of a Type-2 fuzzy set $\tilde{A}$,is a region bounded by two curves, called the *Lower* and the *Upper Membership Functions*, denoted by $\underline{\mu}_{\tilde{A}}(x)$ and $\overline{\mu}_{\tilde{A}}(x)$, respectively symbolizing the minimum and the maximum of the membership functions of the Type-1 fuzzy sets [9] embedded in the FOU all *x*.

B. Non-dominated Sorting Bee Colony (NSBC) Optimization

An overview of the main steps of the NSBC algorithm for jointly minimizing *N* objectives is presented next.

1. Initialization: NSBC involves a population P_t of *NP*, *D*-dimensional food sources (candidate solutions) $\vec{Z}_i(t) = \{z_{i,1}(t), z_{i,2}(t), ..., z_{i,D}(t)\}$ at the current generation t = 0 randomly initialized in the range $[\vec{Z}^{\min}, \vec{Z}^{\max}]$ where $\vec{Z}^{\min} = \{z_1^{\min}, z_2^{\min}, ..., z_D^{\min}\}$ and $\vec{Z}^{\max} = \{z_1^{\max}, z_2^{\max}, ..., z_D^{\max}\}$. The *k*-th objective function $f_k(\vec{Z}_i(0))$ of $\vec{Z}_i(0)$ is evaluated for *i*= [1, *NP*] and *k*= [1, *N*].

2. Employed Bee Phase: An employed bee discovers a new food source $\vec{Z}'_i(t) = \{z_{i,1}(t), ..., z'_{i,j}(t), ..., z_{i,D}(t)\}$ in the neighborhood of the original one $\vec{Z}_i(t) = \{z_{i,1}(t), ..., z_{i,j}(t), ..., z_{i,D}(t)\}$ by determining $z'_{i,j}(t)$ as

$$z'_{i,j}(t) = z_{i,j}(t) + rand_{i,j}(-1,1) \times (z_{i,j}(t) - z_{l,j}(t)) \quad (4)$$

where $rand_{i,j}(-1, 1)$ is a uniform random variable in [−1, 1] and *j* and *l* are randomly selected indices from [1, *D*] and [1, *NP*] respectively but $l \neq i$. The *k*-th objective function $f_k(\vec{Z}'_i(t))$ is evaluated for the food source $\vec{Z}'_i(t)$ for *i*= [1, *NP*] and *k*= [1, *N*].

3. Selection by Employed Bee: If $\vec{Z}'_i(t)$ dominates $\vec{Z}_i(t)$ (i.e., if $f_k(\vec{Z}'_i(t)) \le f_k(\vec{Z}_i(t))$ for k =[1, *N*] and $f_l(\vec{Z}'_i(t)) < f_l(\vec{Z}_i(t))$ for at least one $l \in$ [1, *N*]), $\vec{Z}'_i(t)$ replaces $\vec{Z}_i(t)$ in the memory of the employed bee. However if $\vec{Z}_i(t)$ and $\vec{Z}'_i(t)$ are non-dominated, both solutions are kept in her memory P_t. This step is reiterated for *i*= [1, *NP*] and hence, a population of food sources is achieved with size $|P_t| \in [NP, 2NP]$.

4. Non-dominated Sorting: All the non-dominated food sources of P_t are assigned with the highest rank of one and are included in the optimal Pareto front, *Front_Set*(1). The second front *Front_Set*(2) is formed by the non-dominated food sources of the set $\{P_t – Front\_Set(1)\}$. Continuation of this Pareto ranking process eventually identifies all the non-dominated sets and rank them as *Front_Set*(1), *Front_Set*(2), *Front_Set*(3), and so on.

5. Truncation of Extended Population: The non-dominated set of food sources are passed from P_t to the parent population P'_t for the onlooker bee phase starting from *Front_Set*(1). Let by adding *Front_Set*(*l*) to P'_t, $|P'_t|$ exceeds *NP*. Then the food sources in *Front_Set*(*l*) are sorted in descending order of crowding distance *CD*, revealing the perimeter of a hypercube formed by their nearest neighbors at the vertices in the fitness landscapes. To ensure diversity in population, the food sources in *Front_Set*(*l*) with the highest crowding distances are prioritized for being included in P'_t until $|P'_t|$ becomes *NP*.

6. Probability Calculation: Let Set_i represents the set of all food sources that are dominated by $\vec{Z}_i(t)$ based on $f_k(\vec{Z}_i(t))$ for *k*= [1, *N*]. The probability of $\vec{Z}_i(t)$ to be selected by the onlooker bee is calculated by (5) for *i*= [1, *NP*].

$$prob(i) = |Set_i| / NP \quad (5)$$

7. Onlooker Bee Phase: An onlooker bee probabilistically selects a food source $\vec{Z}_i(t)$ based on its probability *prob*(*i*) using (5) and produces a modification on the food source position as using (4). The population P'_t of size in [*NP*, 2*NP*] is then constructed by following the principle of section II.B.3. Then using the crowding distance induced non-dominated sorting, the non-dominated food sources are identified from P'_t to develop the next generation population P_{t+1} of size NP.

8. Scout Bee Phase: If a position of a food source cannot be improved further through a predefined number of cycles called 'limit', it is abandoned and is replaced by a randomly reinitialized food source position by the scout.

After each evolution, we repeat from step 2 until termination condition for convergence is satisfied.

III. Optimization in Presence of Noise

This section extends traditional NSBC algorithm by the following counts to upgrade its performance in the presence of stochastic noise [2].

A. *Adaptive Selection of Sample Size (ASSS)*

Traditional noisy MOO employs fixed sample size, disregarding the non-uniform distribution of noise in the parametric space of population. Here, we introduce an adaptive selection of sample size $s_k(\vec{Z}_i)$ corresponding to $f_k(\vec{Z}_i)$ of a food source $\vec{Z}_i$ as a monotonically non-decreasing function of fitness variance $y_k(\vec{Z}_i)$ in its local neighborhood for k = [1, N] following the model of [17] within suitable bounds [$s^{\min}$, $s^{\max}$] using

$$s_k(\vec{Z}_i) = s^{\min} + (s^{\max} - s^{\min}) \times (1 - \exp(-y_k(\vec{Z}_i))) \text{ for } k = [1, N]. \quad (6)$$

It is pictorially represented in Fig. 1, where $y_k^{\max}$ denotes the maximum fitness variance in the neighborhood of all food sources in the current population for the k-th objective with k =[1, N]. The monotonicity property of the functional form in (6) ensures that for small fitness variance near zero, the sample size is small, and for increasing fitness variance the sample size also increases with a largest value $s^{\max}$ at $y_k^{\max}$.

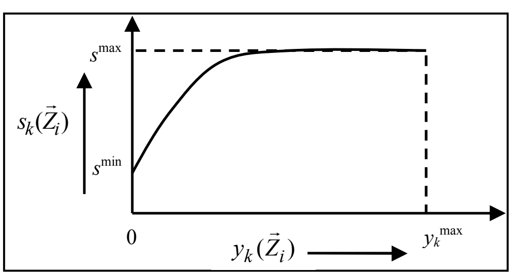

Fig. 1 The non-linearity used to adapt sample size with fitness variance in the local neighborhood

The local neighborhood of a D-dimensional food source $\vec{Z}_i$ is obtained by sub-dividing the search range [$z_j^{\min}$, $z_j^{\max}$] into equally spaced intervals Δz_j for j= [1, D], and then identifying the food sources lying in the hyperspace $[\Delta\vec{Z}_i^{\min}, \Delta\vec{Z}_i^{\max}]$, where

$$\Delta\vec{Z}_i^{\min} = \{z_{i,1}-\Delta z_1, z_{i,2}-\Delta z_2, \ldots, z_{i,D}-\Delta z_D\} \quad (7.1)$$

$$\Delta\vec{Z}_i^{\max} = \{z_{i,1}+\Delta z_1, z_{i,2}+\Delta z_2, \ldots, z_{i,D}+\Delta z_D\} \quad (7.2)$$

and $$\Delta z_j = (z_j^{\max} - z_j^{\min})/NP \quad \text{for } j = [1, D]. \quad (7.3)$$

After the food sources in the local neighborhood N_i around $\vec{Z}_i$ are recognized, the fitness variance in N_i is determined by

$$y_k(\vec{Z}_i) = \frac{\sum_{\forall \vec{Z}_j \in N_i} (s_k(\vec{Z}_j) \times V_k(\vec{Z}_j))}{\sum_{\forall \vec{Z}_j \in N_i} s_k(\vec{Z}_j)} \quad (8)$$

where $V_k(\vec{Z}_j)$ represents a quantitative measure of the spread of the samples of $f_k(\vec{Z}_j)$ for k= [1, N], which is discussed in the next section.

B. *Uncertainty Management in Fitness Evaluation using Centroidal Fitness Estimation (CFE)*

Because of non-uniform variation in fitness measures of the samples of $f_k(\vec{Z}_i)$ in the sample space, an uncertainty creeps into the estimate of $f_k(\vec{Z}_i)$. In order to correctly assess $f_k(\vec{Z}_i)$, the variation of $f_k(\vec{Z}_i)$ is modeled here as an Interval Type-2 Fuzzy Set (IT2FS). This is done in four steps.

Step 1: First we partition the span of $f_k(\vec{Z}_i)$, denoted by $[f_k^{\min}(\vec{Z}_i), f_k^{\max}(\vec{Z}_i)]$, into q intervals of equal length (Fig. 2).

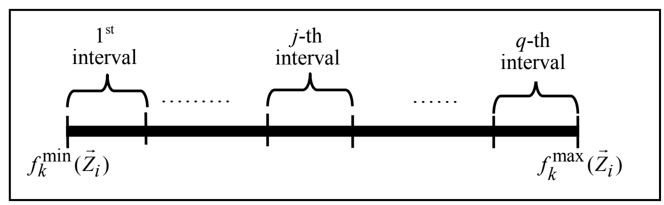

Fig. 2 Fitness intervals in the sample space

Step 2: The $f_k(\vec{Z}_i)$ in each interval is represented by a Gaussian Type-1 Membership Function (T1MF) with mean and variance equal to the respective mean and variance of the fitness samples in the selected interval.

Step 3: Taking union of all such q T-1 Gaussian MFs, each one for one interval, we construct an IT2FS $\mu_{\tilde{A}}(f_k(\vec{Z}_i))$ (Fig. 3), the Footprint of Uncertainty (FOU) of which represents the possible uncertainty in the fitness sample space of $f_k(\vec{Z}_i)$. The FOU is bounded by the UMF $\overline{\mu}_{\tilde{A}}(f_k(\vec{Z}_i))$ and the LMF $\underline{\mu}_{\tilde{A}}(f_k(\vec{Z}_i))$ [9]. The UMF is then approximated as a flat-top MF to maintain convexity characteristics.

Step 4: We perform IT2 defuzzification to determine the left and right end point centroids [9], C_l and C_r respectively, using (9) where l represents the left switching point from UMF to LMF and r denotes the right switching point from LMF to UMF for n distinct points in the interval $[f_k^{\min}(\vec{X}_i), f_k^{\max}(\vec{X}_i)]$. Here w_m symbolizes the m-th sampled point in the same interval with $w_1 = f_k^{\min}(\vec{Z}_i)$ and $w_n = f_k^{\max}(\vec{Z}_i)$. The centroid C of the IT2FS is then obtained by (10) which represents the centroidal fitness of $f_k(\vec{Z}_i)$, denoted by $\overline{f}_k(\vec{Z}_i)$, considering the entire sample space. The entire procedure is performed for each objective, i.e., k= [1, N].

Step 5: Next the median values of fitness samples of each of the q intervals are sorted in ascending order of the magnitude.

The lower and upper quartile from the sorted list are identified and recorded as $Q_{k,0.25}(\vec{Z}_i)$ and $Q_{k,0.75}(\vec{Z}_i)$ respectively.

$$C_l = \frac{\sum_{m=1}^{l} w_m \cdot \overline{\mu}_{\tilde{A}}(w_m) + \sum_{m=l+1}^{n} w_m \cdot \underline{\mu}_{\tilde{A}}(w_m)}{\sum_{m=1}^{l} \overline{\mu}_{\tilde{A}}(w_m) + \sum_{m=l+1}^{n} \underline{\mu}_{\tilde{A}}(w_m)}$$
$$C_r = \frac{\sum_{m=1}^{r} w_m \cdot \underline{\mu}_{\tilde{A}}(w_m) + \sum_{m=r+1}^{n} w_m \cdot \overline{\mu}_{\tilde{A}}(w_m)}{\sum_{m=1}^{r} \underline{\mu}_{\tilde{A}}(w_m) + \sum_{m=r+1}^{n} \overline{\mu}_{\tilde{A}}(w_m)} \quad (9)$$

$$C = (C_l + C_r)/2 \quad (10)$$

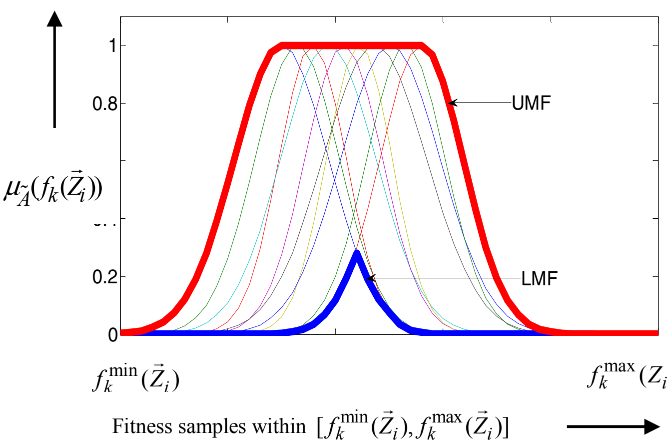

Fig. 3 IT2FS formation from primary memberships of fitness samples lying within the range $[f_k^{\min}(\vec{Z}_i), f_k^{\max}(\vec{Z}_i)]$

The inter-quartile range corresponding to $f_k(\vec{Z}_i)$ is then used as a measure of the spread of sample values of $f_k(\vec{Z}_i)$ away from $\overline{f}_k(\vec{Z}_i)$, denoted by $V_k(\vec{Z}_i)$, for k= [1, N] as follows.

$$V_k(\vec{Z}_i) = Q_{k,0.75}(\vec{Z}_i) - Q_{k,0.25}(\vec{Z}_i) \quad (11)$$

To eliminate the impact of the extreme values of the noisy fitness samples, inter-quartile range is referred to as a measure of the spread of samples of $f_k(\vec{Z}_i)$ from the respective centroidal measurement $\overline{f}_k(\vec{Z}_i)$, instead of sample variance.

C. Probabilistic Selection (PS) during Truncation of Extended Population

The possible promotion of a food source $\vec{Z}_i$ to the next generation (from the same rank candidate pool) in presence of noise depends on two important issues. First, greater the crowding distance measure $CD(\vec{Z}_i)$, higher is the chance of selecting $\vec{Z}_i$ from its competitors residing in the same front. The uncertainty involved in the fitness measurement is taken care of by the second criterion— the probability of non-occurrence of rare samples of $f_k(\vec{Z}_i)$. A higher value of this probability ensures proximity of the measured fitness samples to $\overline{f}_k(\vec{Z}_i)$ in the sample space. The intermittent samples, far away from $\overline{f}_k(\vec{Z}_i)$, are supposed to result from the contamination effect of noise and evidently, their occurrence produces a longer tail in the sample distribution. The contamination effect of noise on fitness samples here is realized using the quartile skewness $\gamma_{k,i}$, defined by (12), which provides a robust measure of the degree of asymmetry of the distribution of fitness samples of $f_k(\vec{Z}_i)$ with respect to the centroidal fitness estimate $\overline{f}_k(\vec{Z}_i)$ for k= [1, N].

$$\gamma_{k,i} = \frac{(Q_{k,0.25}(\vec{Z}_i) - \overline{f}_k(\vec{Z}_i)) - (\overline{f}_k(\vec{Z}_i) - Q_{k,0.75}(\vec{Z}_i))}{Q_{k,0.75}(\vec{Z}_i) - Q_{k,0.25}(\vec{Z}_i)} \quad (12)$$

It is apparent that for $\gamma_{k,i}$ approaching −1 (or $\gamma_{k,i}$ approaching +1), the frequency of occurrence of the fitness samples lying on the left (or right) tail of the fitness samples distribution, far away from $\overline{f}_k(\vec{Z}_i)$, are extremely small and may be regarded as noisy samples. Hence it is expected that for $f_k(\vec{Z}_i)$ being less affected by noise, the measured fitness samples will be very close to $\overline{f}_k(\vec{Z}_i)$ and hence $\gamma_{k,i} \approx 0$. This observation has motivated us to define the probability of non-occurrence of rare samples, providing a measure of the degree of reliability on the samples of $f_k(\vec{Z}_i)$, as follows.

$$p_{k,i} = 1 - |\gamma_{k,i}| \quad (13)$$

The normalized measure of $CD(\vec{Z}_i)$, denoted by $\overline{CD}(\vec{Z}_i)$, is given in (14), where $\vec{Z}_i$ and $\vec{Z}_j$ lie in the same Pareto front.

$$\overline{CD}(\vec{Z}_i) = CD(\vec{Z}_i) \Big/ \sum_{\forall j} CD(\vec{Z}_j) \quad (14)$$

Now, treating $\overline{CD}(\vec{Z}_i)$ like probability and presuming that $p_{k,i}$ for k=[1, N] and $\overline{CD}(\vec{Z}_i)$ are independent, we define the selection probability of $\vec{Z}_i$ for the next generation as

$$ps_i = \prod_{k=1}^{N} p_{k,i} \times \overline{CD}(\vec{Z}_i). \quad (15)$$

It reveals that an increase in either $p_{k,i}$ for k=[1, N] or $\overline{CD}(\vec{Z}_i)$ or both ensure an increase in ps_i.

Procedure NNSBCS

Input: MOO problem, stopping criterion, minimum ($s^{\min}$) and maximum ($s^{\max}$) sample sizes, number of intervals q and 'limit'.

Output: Optimal Pareto set *Front_Set*(1).

Begin

1. Initialize a population P_t of NP, D-dimensional food sources $\vec{Z}_i(t)$ at generation t=0 with $trial_i$=0 for i= [1, NP]. Evaluate $f_{k,l}(\vec{Z}_i(t))$ for i= [1, NP], k= [1, N] and l= [1, $s^{\min}$].
2. Evaluate $\overline{f}_k(\vec{Z}_i(t))$ and $V_k(\vec{Z}_i(t))$ using (10) and (11) respectively from the measured fitness samples for i= [1, NP] and k= [1, N].
3. **While** stopping criterion is not reached **do**

   Begin

   //Employed Bee Phase

   3.1. Produce a new food source $\vec{Z}_i'(t)$ using (4) for i= [1, NP].

   3.2. Determine $y_k(\vec{Z}_i'(t))$ for i= [1, NP] and k= [1, N] using (8).

3.3. Determine $s_k(\vec{Z}'_i(t))$ using (6) for i= [1, NP] and k= [1, N].

3.4. Evaluate $\bar{f}_k(\vec{Z}'_i(t))$ and $V_k(\vec{Z}'_i(t))$ using (10) and (11) respectively from $f_{k,l}(\vec{Z}'_i(t))$ for i= [1, NP], k= [1, N] and l= [1, $s_k(\vec{Z}'_i(t))$].

3.5. **If** $\vec{Z}'_i(t) \prec \vec{Z}_i(t)$ **Then** replace $\vec{Z}_i(t)$ with $\vec{Z}'_i(t)$; $trial_i$= 0;

Else If $\vec{Z}_i(t) \prec \vec{Z}'_i(t)$ **Then** $trial_i$= $trial_i$ +1;

Else $P_t \leftarrow P_t \cup \vec{Z}'_i(t)$;

End If

Repeat the step for i= [1, NP].

3.6. Sort P_t in to subsequent Pareto fronts *Front_Set* using non-dominated sorting principle.

3.7. /* **Stochastic Selection from Same Rank Solutions** */

(a). Set $P'_t \leftarrow NULL$ and $l \leftarrow 1$.

(b). **Repeat**

$P'_t \leftarrow P'_t \cup Front\_Set(l)$; $l \leftarrow l+1$;

Until $|P'_t| + |Front\_Set(l)| > NP$.

(c). Calculate the selection probability of the food sources in *Front_Set*(l) using (15) and sort them in descending order of selection probability.

(d). Set $P'_t \leftarrow P'_t \cup$top ($NP$ – $|P'_t|$) food sources of *Front_Set*(l) with highest selection probability.

//Onlooker Bee Phase

3.8. Select a food source $\vec{Z}_i(t) \in P'_t$ based on its probability of selection $prob(i)$ calculated using (5) for i= [1, NP].

3.9. Repeat from 3.1 to 3.7 with $P_t \leftarrow P'_t$ (in steps 3.1 to 3.6) and $P'_t \leftarrow P_{t+1}$ (in step 3.7).

3.10. Reinitialize food source with maximum *trial* value exceeding 'limit' by a **scout**.

3.11. $t \leftarrow t+1$.

End While

End

IV. Experimental Settings

A. Benchmark Functions and Noise Models

The performance of the proposed NNSBCS algorithm is validated here with respect to noisy version of 23 multi-objective benchmark functions [16] recommended in CEC'2009. Among these benchmarks UF11-UF13 are the extended and rotated versions of two immensely popular test suites, Deb-Thiele-Laumanns-Zitzler (DTLZ) and one test function of the Walking Fish group (WFG) test suite. The noisy version of $f_k(\vec{Z})$ for k= [1, N] is given by

$$f_{k-noisy}(\vec{Z}) = f_k(\vec{Z}) + \eta_k \qquad (16)$$

where η_k is the amplitude of stochastic noise samples, taken from five probability distribution functions (PDFs), regarded in this paper.

1. **Gaussian:** The injection of Gaussian noise η_k here is realized using a well-known technique called Box-Muller method [2].

2. **Poisson:** Knuth's algorithm [2] is used to inject η_k into the fitness landscapes following Poisson distribution.

3. **Rayleigh:** The injection of Rayleigh noise η_k on $f_k(\vec{Z})$ here is performed using inverse transform sampling [18].

4. **Exponential:** We have used Ziggurat method [19] to inject η_k into $f_k(\vec{Z})$ following exponential distribution.

5. **Random:** Lastly we consider η_k to be a **random** noise with maximum noise amplitude within ±25% of the true fitness amplitudes. Here, random noise is generated using linear congruential pseudo random number generator [2].

B. Comparative Framework and Parameter Settings

The MOO algorithms used for the comparative study include Noisy NSBC (NNSBC) [11], Differential Evolution for Multi-objective Optimization with Noise (DEMON) [2], NSGA-II with α-dominance operator (NSGA-II-A) [5], Confidence based Dynamic Resampling (CDR) [12], elitist Evolutionary Multi-Agent System (elEMAS) [7], Multi-Objective Evolutionary Algorithm with Robust Features (MOEA-RF) [13], Noise-Tolerant Strength Pareto Evolutionary Algorithm (NT-SPEA) [6] and Pareto front-Efficient Global Optimization (ParEGO) [8]. Similarly, traditional DEMO [2], Multi-Objective Particle Swarm Optimization (MOPSO) [20], Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D) [14] and S Metric Selection Evolutionary Multiobjective Optimization Algorithm (SMS-EMOA) [15] are also extended with the noise handling strategies introduced in this paper to develop four new members of our comparative framework, denoted as NDEMOS, NMOPSOS, NMOEA/DS and NSMS-EMOAS respectively. For all the algorithms, the population size is kept at 50 and the maximum function evaluations (FEs) is set as $10^4 \times D$ for D-dimensional problem. To make the comparison fair, the population for all the algorithms is initialized using the same random seeds. The best parametric set-up is employed for all these algorithms as prescribed in their respective sources. In our proposed NNSBCS algorithm, s^{min} and s^{max} are considered to be 2 and 30 respectively with 'limit' of 50 and q equals to 10.

C. Performance Metric

Hyper Volume Ratio (*HVR*): It is defined as follows [2].

$$HVR(A) = \frac{HV(A)}{HV(P^*)} \qquad (17)$$

Here $HV(A)$ and $HV(P^*)$ represent the hyper volume of the approximate Pareto front A and optimal Pareto front P^* respectively. The hyper volume of a set S of non-dominated solutions, denoted by $HV(S)$, signifies the size of the objective space covered by S. If the non-dominated vectors of A are identical with the members of P^*, $HVR(A)$ indicator achieves its maximum and ideal value 1.

V. Experiments and Results

A. Effectiveness of Different Extensions of NNSBCS

In section III, traditional NSBC is extended by incorporating three mutually independent strategies namely ASSS, CFE and PS. The seven groups of strategies are considered for the experiments, including NSBC-ASSS, NSBC-CFE, NSBC-PS, NSC-ASSS-CFE, NSBC-ASSS-PS, NSBC-CFE-PS and NNSBCS. The mean and the standard deviation (within parentheses) of the best-of-run *HVR* metric values for 50 independent runs of each of the seven variants of the traditional NSBC, are presented in Table-I for 30-dimensional benchmark functions [16], each contaminated with zero mean Gaussian noise of variance σ^2=0.75. The best metric value obtained in each case has been marked in bold. A non-parametrical statistical test, known as Friedman two-way analysis of variances by ranks [2], is also performed on the mean of *HVR* metric values as reported in Table-I. The null hypothesis here, states that all the algorithms are equivalent, so their individual ranks should be equal. The last row of Table-I summarizes the rankings obtained by Friedman procedure. With the level of significance α=0.05, the Friedman statistic exhibits significant differences in the performance of the competitor algorithms with a test value of 138. The results highlight NNSBCS as the best algorithm.

B. Comparative Analysis of NNSBCS

The comparative analysis of the relative performance of the proposed NNSBCS with its competitors is summarized in this section. Although all the experiments are performed for all five variants of noise with noise variance $\sigma^2 \in$ [0, 1] and also for problem dimensions in [10, 100], we report here the results for a finite values of σ^2 and for specific problem dimensions only to save space. The mean and standard deviation (within parenthesis) of *HVR* metric for 50 independent runs (each with 30×10^4 FEs for 30-dimensional problem) of each of the thirteen algorithms are presented in Table-II with η_k to be the Poisson noise (mean=variance, σ^2=0.25). The best metric value obtained in each case has been shown in bold. The simulation results in Table-II show that NNSBCS has outperformed its competitors over 16 cases out of 23 benchmark functions. NMOEA/DS, which achieves the second best rank, has obtained the best average *HVR* value outperforming NNSBCS in cases of UF5, UF12, CF7 and CF9. In case of benchmark functions UF13 and CF4, the performance of NNSBCS is comparable to that of NMOEA/DS. NDEMOS has obtained the best average *HVR* value outperforming NNSBCS in one case (UF8).

C. Effect of Varying Noise Variance

Fig. 4 shows the evolution of the average *HVR* metric values of the population with the exponential noise variance for all the thirteen algorithms keeping the number of generations to be fixed at 50×10^4 (for problem dimension *D*=50). It is evident from Fig. 4 that all the noisy MOO algorithms gradually lose their efficacy in attaining the *HVR* metric values close to the ideal value one with increasing variance of noise samples contaminating the fitness landscapes. However, NNSBCS has emerged as the most effective algorithm in obtaining approximate Pareto front with higher values of *HVR* metric, even when noise is a predominant factor.

D. Effect of Varying Problem Dimension

The evolution of the average *HVR* metric values obtained by all competitor algorithms against problem dimension (within [10, 100]) is plotted in Fig. 5 for random distribution of noise. Fig. 5 envisages that *HVR* metric is non-increasing function of dimension *D* of the parametric space for specific noise settings. This phenomenon elucidates that more complex terrain needs to be explored by the candidate solutions to obtain the Pareto optima with an increase in *D*, thereby decreasing *HVR* metric values. However for all instances NNSBCS achieves the best rank among its contestants by providing the highest values of the above metric.

E. Non-parametric Statistical Analysis of Performance

Friedman and Iman-Davenport non-parametrical statistical tests [2] are performed on the mean of *HVR* metric values as reported in Table-II. Table-III summarizes the rankings obtained by Friedman procedure, highlighting NNSBCS as the best algorithm. With the level of significance α of 0.05, both the Friedman and Iman-Davenport statistic show significant differences in performance of the contender algorithms with test values as presented in Table-III and $p < 0.001$. In the post-hoc analysis we have applied the Bonferroni-Dunn test [2] over the ranking results of Friedman procedure. The critical difference [2] for the test is calculated as 3.2913 for these data. The performance of two algorithms is considered to be significantly different only if their corresponding average Friedman ranks differ by at least a critical difference, which is depicted in Fig. 6. It portrays that all the algorithms, except NMOEA/DS, NDEMOS and NSMS-EMOAS, may be regarded as significantly inferior to NNSBCS with a level of significance $\alpha = 0.05$.

TABLE I. RANKING OF DIFFERENT EXTENSIONS OF NNSBCS WITH RESPECT TO *HVR* METRIC VALUES FOR ZERO MEAN GAUSSIAN NOISE OF σ^2=0.75

Func.	NSBC-ASSS	NSBC-CFE	NSBC-PS	NSBC-ASSS-CFE	NSBC-ASSS-PS	NSBC-CFE-PS	NNSBCS
UF1	0.6609 (0.308)	0.7348 (0.300)	0.7211 (0.306)	0.8165 (0.208)	0.7470 (0.247)	0.8228 (0.154)	**0.8559 (0.144)**
UF2	0.7652 (0.387)	0.8103 (0.270)	0.7807 (0.364)	0.8454 (0.243)	0.8181 (0.257)	0.8564 (0.151)	**0.8643 (0.116)**
UF3	0.6455 (0.336)	0.7094 (0.091)	0.6581 (0.187)	0.8002 (0.090)	0.7550 (0.090)	0.8073 (0.077)	**0.8881 (0.068)**
UF4	0.8021 (0.396)	0.8427 (0.372)	0.8066 (0.378)	0.8704 (0.222)	0.8541 (0.224)	0.8779 (0.184)	**0.8980 (0.144)**
UF5	0.7427 (0.168)	0.7645 (0.126)	0.7447 (0.166)	0.8198 (0.081)	0.7651 (0.119)	0.8752 (0.080)	**0.8976 (0.043)**
UF6	0.7103 (0.308)	0.7213 (0.211)	0.7198 (0.240)	0.7951 (0.168)	0.7413 (0.176)	0.8352 (0.143)	**0.8584 (0.108)**
UF7	0.7761 (0.348)	0.8058 (0.284)	0.7865 (0.307)	0.8235 (0.133)	0.8179 (0.206)	0.8624 (0.075)	**0.8761 (0.057)**
UF8	0.6751 (0.308)	0.7393 (0.184)	0.6794 (0.193)	0.7761 (0.157)	0.7689 (0.176)	0.8127 (0.091)	**0.8134 (0.089)**
UF9	0.6865 (0.374)	0.7587 (0.249)	0.7127 (0.268)	0.8101 (0.171)	0.7881 (0.203)	0.8568 (0.161)	**0.8912 (0.047)**
UF10	0.7271 (0.343)	0.7502 (0.305)	0.7279 (0.334)	0.8109 (0.157)	0.7612 (0.268)	0.8277 (0.134)	**0.8717 (0.081)**
UF11	0.7106 (0.276)	0.7622 (0.259)	0.7311 (0.260)	0.7958 (0.201)	0.7909 (0.243)	0.8041 (0.107)	**0.8274 (0.099)**
UF12	0.7585 (0.379)	0.7708 (0.339)	0.7632 (0.375)	0.8348 (0.237)	0.8189 (0.328)	0.8825 (0.204)	**0.9107 (0.162)**
UF13	0.7166 (0.379)	0.7846 (0.346)	0.7304 (0.364)	0.8563 (0.279)	0.8491 (0.282)	0.8652 (0.242)	**0.8812 (0.235)**
CF1	0.6649 (0.232)	0.6941 (0.133)	0.6742 (0.155)	0.7380 (0.056)	0.6951 (0.070)	0.8323 (0.052)	**0.8606 (0.018)**
CF2	0.7263 (0.346)	0.8069 (0.260)	0.7280 (0.269)	0.8209 (0.199)	0.8127 (0.208)	0.8659 (0.080)	**0.8691 (0.075)**
CF3	0.6645 (0.214)	0.6936 (0.192)	0.6773 (0.214)	0.7386 (0.060)	0.7087 (0.146)	0.7523 (0.048)	**0.7823 (0.042)**
CF4	0.6679 (0.400)	0.7541 (0.304)	0.7125 (0.347)	0.7714 (0.261)	0.7710 (0.301)	0.8680 (0.241)	**0.8779 (0.234)**
CF5	0.6817 (0.328)	0.7416 (0.318)	0.7162 (0.326)	0.7553 (0.096)	0.7499 (0.173)	0.8586 (0.082)	**0.8903 (0.064)**
CF6	0.6532 (0.319)	0.6733 (0.283)	0.6634 (0.311)	0.7967 (0.248)	0.7490 (0.253)	0.8989 (0.219)	**0.9028 (0.156)**
CF7	0.6006 (0.361)	0.6420 (0.248)	0.6080 (0.259)	0.6917 (0.210)	0.6597 (0.216)	0.7715 (0.199)	**0.8178 (0.182)**
CF8	0.7475 (0.381)	0.7950 (0.332)	0.7670 (0.374)	0.8307 (0.239)	0.8124 (0.274)	0.8459 (0.223)	**0.8483 (0.173)**
CF9	0.7040 (0.317)	0.7701 (0.246)	0.7276 (0.307)	0.8433 (0.175)	0.8292 (0.207)	0.8648 (0.174)	**0.8991 (0.145)**
CF10	0.7388 (0.398)	0.7740 (0.265)	0.7712 (0.278)	0.7894 (0.215)	0.7840 (0.265)	0.7900 (0.071)	**0.8564 (0.051)**
Fried-man Rank	7	5	6	3	4	2	1

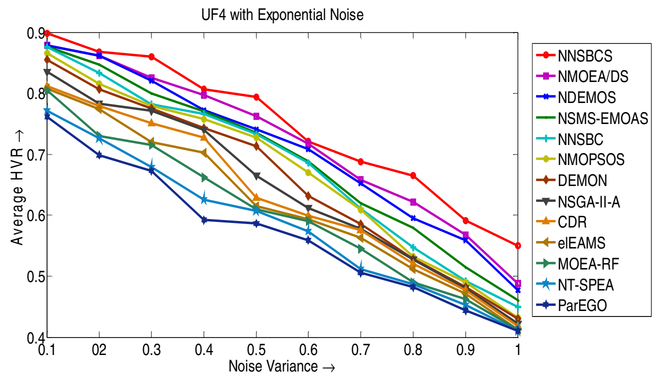

Fig. 4 Plot of average *S* metric values with exponential noise variance σ^2 for UF4 with 500000 FEs (30-*D* problem)

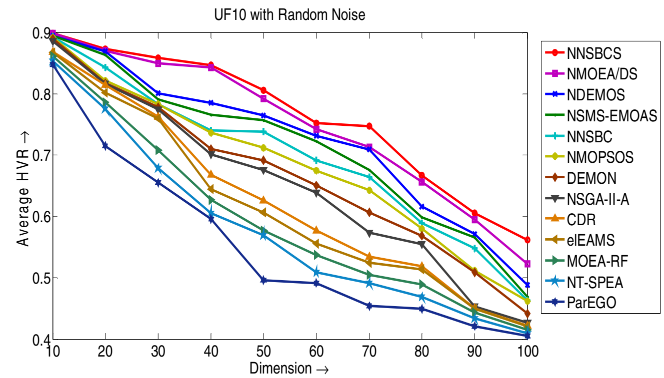

Fig. 5 Plot of average *ER* metric values with problem dimension for random (limited amplitude) noise contaminating UF10 respectively

TABLE II. MEAN *HVR* VALUES OVER 50 INDEPENDENT RUNS WITH POISSON NOISE OF MEAN=VARIANCE σ^2=0.25

Functions	NNSBCS	NMOEA/ DS	NDEMOS	NSMS-EMOAS	NNSBC	NMOPS OS	DEMON	NSGA-II-A	CDR	elEAMS	MOEA-RF	NT-SPEA	ParEGO
UF1	**0.8665 (0.013)**	0.8659 (0.019)	0.8635 (0.020)	0.8465 (0.023)	0.8330 (0.035)	0.7638 (0.058)	0.7250 (0.229)	0.6536 (0.273)	0.61570 (0.315)	0.5660 (0.350)	0.5160 (0.350)	0.4667 (0.396)	0.4541 (0.408)
UF2	**0.9072 (0.011)**	0.9033 (0.013)	0.8666 (0.014)	0.8533 (0.036)	0.8290 (0.041)	0.8256 (0.050)	0.7571 (0.220)	0.7326 (0.296)	0.7168 (0.318)	0.6801 (0.342)	0.5905 (0.349)	0.5703 (0.377)	0.4732 (0.377)
UF3	**0.8998 (0.071)**	0.8296 (0.072)	0.8195 (0.073)	0.8108 (0.075)	0.7898 (0.081)	0.7499 (0.085)	0.6472 (0.142)	0.6112 (0.167)	0.6111 (0.181)	0.5810 (0.186)	0.5807 (0.235)	0.4932 (0.279)	0.4612 (0.293)
UF4	**0.9137 (0.012)**	0.9125 (0.069)	0.9009 (0.089)	0.8995 (0.130)	0.8955 (0.140)	0.8940 (0.152)	0.7858 (0.194)	0.7700 (0.198)	0.7692 (0.220)	0.7356 (0.233)	0.6734 (0.347)	0.5762 (0.409)	0.4780 (0.416)
UF5	0.9053 (0.025)	**0.9137 (0.019)**	0.9010 (0.046)	0.8893 (0.048)	0.8849 (0.055)	0.8785 (0.106)	0.7537 (0.228)	0.7166 (0.255)	0.7052 (0.276)	0.6739 (0.347)	0.6716 (0.350)	0.5236 (0.373)	0.4569 (0.413)
UF6	**0.8728 (0.024)**	0.8623 (0.027)	0.8612 (0.029)	0.8583 (0.053)	0.8524 (0.055)	0.8424 (0.111)	0.6728 (0.192)	0.6706 (0.221)	0.6444 (0.253)	0.5523 (0.303)	0.5457 (0.326)	0.5068 (0.365)	0.4861 (0.375)
UF7	**0.8893 (0.001)**	0.8773 (0.005)	0.8728 (0.028)	0.8726 (0.028)	0.8723 (0.028)	0.8563 (0.035)	0.78318 (0.121)	0.7308 (0.141)	0.6753 (0.153)	0.6473 (0.172)	0.6283 (0.186)	0.5627 (0.224)	0.4806 (0.347)
UF8	0.8612 (0.059)	0.8663 (0.037)	**0.8725 (0.028)**	0.8101 (0.047)	0.7980 (0.129)	0.7786 (0.196)	0.7174 (0.264)	0.6645 (0.274)	0.6162 (0.278)	0.6138 (0.281)	0.5168 (0.299)	0.4859 (0.367)	0.4697 (0.368)
UF9	**0.9059 (0.037)**	0.9052 (0.037)	0.8899 (0.045)	0.8498 (0.052)	0.8444 (0.075)	0.8073 (0.126)	0.6897 (0.273)	0.6839 (0.312)	0.6793 (0.325)	0.6393 (0.346)	0.6329 (0.362)	0.5511 (0.380)	0.5050 (0.419)
UF10	**0.9188 (0.081)**	0.9186 (0.083)	0.9141 (0.089)	0.8988 (0.113)	0.8836 (0.129)	0.8605 (0.178)	0.7060 (0.208)	0.6901 (0.215)	0.6527 (0.229)	0.6258 (0.349)	0.6087 (0.387)	0.4673 (0.412)	0.4564 (0.416)
UF11	**0.8396 (0.000)**	0.7990 (0.000)	0.7915 (0.007)	0.7906 (0.009)	0.7711 (0.012)	0.7460 (0.050)	0.7094 (0.071)	0.6989 (0.175)	0.6905 (0.205)	0.6409 (0.207)	0.5532 (0.233)	0.5170 (0.399)	0.4745 (0.409)
UF12	0.9114 (0.035)	**0.9143 (0.022)**	0.8750 (0.035)	0.8620 (0.043)	0.8554 (0.059)	0.8421 (0.105)	0.7470 (0.183)	0.7456 (0.223)	0.7185 (0.286)	0.6284 (0.310)	0.6280 (0.336)	0.5641 (0.356)	0.4510 (0.414)
UF13	**0.8878 (0.075)**	**0.8878 (0.075)**	0.8658 (0.075)	0.8588 (0.124)	0.8180 (0.134)	0.7722 (0.163)	0.7175 (0.278)	0.7159 (0.310)	0.6748 (0.333)	0.6691 (0.343)	0.6146 (0.371)	0.5265 (0.383)	0.5185 (0.388)
CF1	**0.8835 (0.020)**	0.8742 (0.032)	0.8656 (0.038)	0.8395 (0.061)	0.8320 (0.078)	0.7849 (0.109)	0.6286 (0.236)	0.6277 (0.260)	0.5688 (0.287)	0.5669 (0.313)	0.5647 (0.322)	0.4899 (0.368)	0.4751 (0.410)
CF2	**0.8946 (0.032)**	0.8910 (0.040)	0.8769 (0.060)	0.8643 (0.079)	0.8265 (0.095)	0.8216 (0.132)	0.7687 (0.214)	0.6998 (0.217)	0.6924 (0.258)	0.6168 (0.261)	0.5979 (0.292)	0.4930 (0.416)	0.4746 (0.418)
CF3	**0.9134 (0.009)**	0.8429 (0.021)	0.8313 (0.022)	0.8090 (0.027)	0.8014 (0.028)	0.7952 (0.059)	0.6475 (0.195)	0.6441 (0.207)	0.6284 (0.208)	0.5999 (0.235)	0.5066 (0.249)	0.4847 (0.377)	0.4803 (0.412)
CF4	**0.8909 (0.015)**	**0.8909 (0.015)**	0.8522 (0.020)	0.8252 (0.023)	0.7581 (0.032)	0.7415 (0.155)	0.6736 (0.230)	0.6563 (0.238)	0.6231 (0.287)	0.5829 (0.310)	0.5464 (0.311)	0.4752 (0.393)	0.4628 (0.400)
CF5	**0.9168 (0.002)**	0.9075 (0.244)	0.8067 (0.554)	0.8058 (0.210)	0.8038 (0.101)	0.7934 (0.116)	0.6619 (0.168)	0.6523 (0.190)	0.6176 (0.268)	0.5727 (0.273)	0.5721 (0.294)	0.5017 (0.347)	0.4945 (0.371)
CF6	**0.9151 (0.013)**	0.9099 (0.035)	0.8902 (0.071)	0.8862 (0.096)	0.8542 (0.105)	0.7155 (0.139)	0.6553 (0.228)	0.6366 (0.243)	0.6261 (0.270)	0.5930 (0.314)	0.5867 (0.361)	0.5639 (0.384)	0.5363 (0.399)
CF7	0.8653 (0.012)	**0.8661 (0.000)**	0.8467 (0.016)	0.8449 (0.028)	0.8042 (0.049)	0.7926 (0.079)	0.6172 (0.249)	0.5965 (0.251)	0.5853 (0.253)	0.5672 (0.298)	0.5452 (0.364)	0.4759 (0.406)	0.4599 (0.410)
CF8	**0.8946 (0.022)**	0.8828 (0.022)	0.8605 (0.022)	0.8444 (0.036)	0.8251 (0.047)	0.8239 (0.068)	0.7379 (0.147)	0.7133 (0.202)	0.6927 (0.255)	0.6166 (0.297)	0.5913 (0.340)	0.5032 (0.383)	0.4535 (0.417)
CF9	0.8520 (0.015)	**0.9154 (0.002)**	0.8497 (0.075)	0.8199 (0.078)	0.8152 (0.115)	0.8029 (0.141)	0.6902 (0.231)	0.6854 (0.250)	0.6176 (0.280)	0.6060 (0.303)	0.5317 (0.336)	0.5176 (0.365)	0.4630 (0.403)
CF10	**0.9127 (0.159)**	0.9103 (0.160)	0.8970 (0.163)	0.8708 (0.163)	0.8663 (0.180)	0.8405 (0.189)	0.6629 (0.309)	0.662 (0.310)	0.6527 (0.311)	0.6316 (0.332)	0.5670 (0.356)	0.4662 (0.397)	0.4564 (0.410)

TABLE III. AVERAGE RANKINGS OBTAINED THROUGH FRIEDMAN'S TEST

Algorithm	Ranking
NNSBCS	**1.304**
NMOEA/DS	1.783
NDEMOS	2.913
NSMS-EMOAS	4.000
NNSBC	5.000
NMOPSOS	6.000
DEMON	7.000
NSGA-II-A	8.000
CDR	9.000
elEMAS	10.00
MOEA-RF	11.00
NT-SPEA	12.00
ParEGO	13.00
Friedman Statistic	275.0371
Iman-Davenport Statistic	6284.246

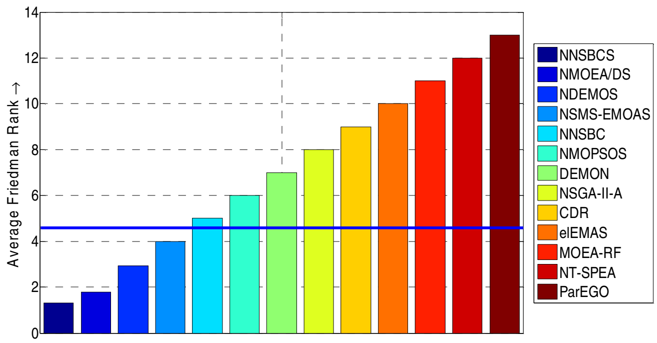

Fig. 6 Graphical representation of Bonferroni-Dunn's procedure with NNSBCS as control method using the results given in Table-III corresponding to the values of *HVR* metric in Table-II

VI. CONCLUSION

A novel approach is proposed to handle noise in the fitness landscapes of NSBC algorithm for efficiently obtaining the global optima. First, a non-linear functional form is developed to competently model the adaptation of the sample size for periodic fitness evaluation of a trial solution with the fitness variance in its local neighborhood. The second strategy is concerned with the evaluation of defuzzified centroidal value of the noisy fitness samples based on their membership value belonging to a definite interval in the entire sample space. The third strategy enhances the robustness in selecting a trial solution from its contenders of the same front based on its crowding distance measure and skewness of its fitness samples distribution.

A comparative study of the proposed NNSBCS algorithm is undertaken with twelve contenders with respect to the noisy version of a test bench of 23 CEC'2009 objective functions on the basis of *HVR* metric. Statistical significance of the results has been judged with the non-parametric Friedman test, Iman-Davenport statistic and Bonferroni–Dunn post hoc analysis. The supremacy of the proposed NNSBCS algorithm over its contenders in a statistically significant manner is substantiated by the experimental study undertaken in presence of five different stochastic distributions of noise samples contaminating the objective spaces.

Acknowledgment

Funding by Council of Scientific and Industrial Research, India (for awarding Senior Research Fellowship to the first author) and funding by University Grants Commission, India, University of Potential Excellence Program (Phase II) in Cognitive Science, Jadavpur University are gratefully acknowledged for the present work.

REFERENCES

[1]. Deb K., Pratap A., Agarwal S., and Meyarivan T., "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, April, 2002, pp. 182-197.

[2]. Rakshit P., Konar A., Das S., Jain L. C., and Nagar A. K., "Uncertainty Management in Differential Evolution Induced Multi-objective Optimization in Presence of Measurement Noise," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 44, no. 7, July, 2014, pp. 922 – 937.

[3]. Stagge P., "Averaging Efficiently in the Presence of Noise," in *the fifth International Conference on Parallel Problem Solving from Nature*, Lecture Notes in Computer Science, Springer-Verlag, vol. 1498, 1998, pp. 188-197.

[4]. Babbar M., Lakshmikantha A., Goldberg D. E., "A Modified NSGA-II to Solve Noisy Multi-objective Problems," in *Genetic and Evolutionary Computation Conference*, AAAI, Chicago, 2003, Late-Breaking Papers, pp. 21-27.

[5]. Boonma P., and Suzuki J., "A Confidence-based Dominance Operator in Evolutionary Algorithms for Noisy Multiobjective Optimization Problems," in *Proceedings of 21st IEEE International Conference on Tools with Artificial Intelligence*, November, 2009, pp. 387-394.

[6]. Buche D., Stall P., Dornberger R., and Koumoutsakos P., "Multiobjective Evolutionary Algorithm for the Optimization of Noisy Combustion Processes," *IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews*, vol. 32, no. 4, November, 2002, pp. 460-473.

[7]. Siwik L., and Natanek S., "Elitist Evolutionary Multi-Agent System in Solving Noisy Multi-objective Optimization Problems," in *Proceedings of IEEE Congress on Evolutionary Computation*, June, 2008, pp. 3319-3326.

[8]. Knowles J., and Hughes E. J., "Multiobjective Optimization on a Budget of 250 Evaluations," in *Evolutionary Multi-Criterion Optimization*, Lecture Notes in Computer Science, Springer-Verlag Berlin Heidelberg, vol. 3410, 2005, pp. 176-190.

[9]. Mendel J. M., *Uncertain Rule-Based Fuzzy Logic Systems: Introduction and New Directions*, Upper Saddle River, NJ: Prentice-Hall, 2001.

[10]. Rakshit P., Konar A., Chowdhury A., Kim E., and Nagar A. K., "Multi-objective Evolutionary Approach of Ligand Design for Protein-Ligand Docking Problem", in *Proceedings of IEEE Congress on Evolutionary Computation*, June, 2013, pp. 237-244.

[11]. Rakshit P., Konar A., and Nagar A. K., "Artificial Bee Colony Induced Multi-objective Optimization in Presence of Noise", in *Proceedings of IEEE Congress on Evolutionary Computation*, 2014, pp. 3176-3183.

[12]. Syberfeldt A., Ng A., John R. I., and Moore P., "Evolutionary Optimisation of Noisy Multi-objective Problems using Confidence-Based Dynamic Resampling," *European Journal of Operational Research*, vol. 204, no. 3, 2010, pp. 533-544.

[13]. Goh C. K., and Tan K. C., "An Investigation on Noisy Environments in Evolutionary Multiobjective Optimization," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 3, 2007, pp. 354-381.

[14]. Zhang Q., and Li H., "MOEA/D: A Multiobjective Evolutionary Algorithm based on Decomposition," *IEEE Transactions on Evolutionary Computation,* vol. 11, no. 6, 2007, pp. 712-731.

[15]. Beume N., Naujoks B., and Emmerich M., "SMS-EMOA: Multiobjective Selection based on Dominated Hypervolume," *European Journal of Operational Research*, vol. 181, no. 3, 2007, pp. 1653-1669.

[16]. Zhang Q., Zhou A., Zhao S., Suganthan P. N., Liu W., and Tiwari S., "Multiobjective Optimization Test Instances for the CEC 2009 Special Session and Competition," *Working Report, CES-887,* School of Computer Science and Electrical Engineering, University of Essex, Colchester, UK and Nanyang Technological University, Singapore, Special Session on Performance Assessment of Multi-Objective Optimization Algorithms, Technical Report 2008, revised on April, 20, 2009.

[17]. Zitzler E., and Künzli S., "Indicator-based Selection in Multiobjective Search," in *Parallel Problem Solving from Nature-PPSN VIII*, Springer Berlin Heidelberg, January, 2004, pp. 832-842.

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

[19]. Marsaglia G., and Tsang W. W., "The Ziggurat Method for Generating Random Variables," *Journal of Statistical Software*, vol. 5, no. 8, 2000, pp. 1-7.

[20]. Coello C. C. A., and Lechuga M., "MOPSO: A Proposal for Multiple Objective Particle Swarm Optimization," in *Proceedings of IEEE Congress of Evolutionary Computation*, vol. 2, May, 2002, pp. 1051–1056.