2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

Artificial Bee Colony Induced Multi-objective Optimization in Presence of Noise

Pratyusha Rakshit[1], Amit Konar[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 aims at designing new strategies to extend traditional Non-dominated Sorting Bee Colony algorithm to proficiently obtain Pareto-optimal solutions in presence of noise on the fitness landscapes. The first strategy, referred to as adaptive selection of sample-size, is employed to balance the trade-off between accurate fitness estimate and computational complexity. The second strategy is concerned with determining statistical expectation, instead of conventional averaging of fitness-samples as the measure of fitness of the trial solutions. The third strategy attempts to extend Goldberg's approach to examine possible placement of a slightly inferior solution in the optimal Pareto front using a more statistically viable comparator. Experiments undertaken to study the performance of the extended algorithm reveal that the extended algorithm outperforms its competitors with respect to four performance metrics, when examined on a test-suite of 23 standard benchmarks with additive noise of three statistical distributions.**

Keywords*— *multi-objective optimization; artificial bee colony; noise-handling; non-dominated sorting.

I. INTRODUCTION

The fitness functions in a multi-objective optimization (MOO) capture the mathematical relationship between the input (measurement) and the output (estimator) variables, characterizing functionality of a physical system. Hence the objectives in a MOO problem are functions of both measurement and estimator variables. However, the measurement variables of a real-world problem often are contaminated with noise because of noisy ambience, poor sensor characteristics and/or faulty measurement procedure. It in turn persuades noise in the trial solutions in a MOO attempting to optimize the noisy objectives. Consequently, an infiltration of noise in the measurement variables induces inaccuracies in the estimator variables or fitness functions or both. Hence in noisy fitness landscape, a quality trial solution may be deceived due to its poor (noisy) fitness estimates and may be discarded from the optimal Pareto front, while a deceptive solution with illusive good fitness may be promoted to the next generation. The paper addresses the issues of uncertainty management (regarding selection of qualitative trial solutions) in MOO in presence of noise.

"Sampling" is one of the most popular approaches to improve fitness measurement in noisy MOO problem [1]. Among the other noisy MOO techniques, periodic re-evaluation of archived solutions [2], probabilistic Pareto ranking [3], and extended averaging approach [4] need special mentioning.

The present paper addresses the issues of uncertainty management in ranking trial solutions by incorporating the following three policies. First, the sample-size of the fitness of each trial solution is adapted with the fitness variance in their local neighborhood to competently balance the accuracy in fitness estimation and run-time. Second, the paper introduces a novel strategy to evaluate expected fitness of the trial solutions from the fitness samples for fitness estimation. The third policy taken up here is an extension of Goldberg's original work [5] which states that if a good member occupies the optimal Pareto front, a weaker member can also occupy the same front, if their difference in mean fitness is less than the average of their fitness variances, scaled by a "neighborhood restriction factor". In our extension, the mean is substituted by expected value and the variance of each trial solution is replaced by the pooled variance of the two competitor solutions, supposing that the sample-sizes of two members are different.

Any traditional MOO algorithm, such as Non-dominated Sorting Bee Colony (NSBC) [6], Differential Evolution for Multi-objective Optimization (DEMO) [7], Multi-objective Particle Swarm Optimization (MOPSO) [8], Non-dominated Sorting Genetic Algorithm-II (NSGA-II) [9] can be extended by the proposed approach in the present context. However, here we selected NSBC for its simplicity in coding, fewer control parameters, good accuracy and fast speed of convergence.

Experiments have been undertaken to investigate the efficacy of the proposed approach for noisy optimization by adding three different types of stochastic noise: i) Gaussian noise with mean zero and increasing variance; ii) Poisson noise distribution with increasing variance and iii) random noise with positive and negative excursions of the noise amplitude within ± 25% of the fitness function values. The performance of the proposed noisy MOO algorithm realized with NSBC (hereafter referred to as Noisy Non-dominated Sorting Bee Colony—NNSBC) is compared with traditional NSBC [6], modified NSGA-II [5], NSGA-II with α-dominance operator (NSGA-II-A) [10], elitist Evolutionary Multi-Agent System (elEMAS) [11], Noise-Tolerant Strength Pareto Evolutionary Algorithm (NT-SPEA) [2], and Differential Evolution for Multi-objective Optimization with Random Scale-Factor and Threshold Selection (DEMO-RSF-

TS), DEMO with Noise [17], extended MOPSO [17] and extended NSGA-II [17] on the noisy version of a set of 23 benchmark functions [13]. The performance of traditional DEMO, MOPSO and NSGA-II are also compared with their amended versions realized with the proposed noise handling strategies. Experiments reveal that the proposed algorithm outperforms other algorithms by four important aspects, namely *inverted generational distance*, *spacing*, *error ratio* and *hyper-volume ratio*.

The paper is divided into five sections. Section II introduces the NSBC algorithm. Section III provides the noise handling mechanism in NNSBC. A comprehensive discussion on the experimental settings for the benchmarks and simulation results are presented in section IV. Conclusions are given in section V.

II. NON-DOMINATED SORTING BEE COLONY

The following definitions are referred to frequently to explain a MOO.

Definition 1. In a MOO problem, a trial solution $\vec{X}_i$ is said to **dominate** other solution $\vec{X}_j$, denoted by $\vec{X}_i \prec \vec{X}_j$, if $\vec{X}_i$ is no worse than $\vec{X}_j$ in all fitness functions and for at least one fitness measure $\vec{X}_i$ is strictly better than $\vec{X}_j$.

Definition 2. Let P be a set of solutions to a MOO problem. $P' \subseteq P$ is called the **non-dominated set** of solutions if the members of P' are not dominated by any member of P.

Definition 3. Let fit_1 and fit_2 be two fitness functions in a MOO problem, and $\vec{X}$, $\vec{X}_i$ and $\vec{X}_j$ are the members of a non-dominated list of solutions. Furthermore, $\vec{X}_i$ and $\vec{X}_j$ are the nearest neighbors of $\vec{X}$ in the fitness landscape. **Crowding distance** of $\vec{X}$ delineates the perimeter of a hypercube formed by considering its nearest neighbors, i.e., $\vec{X}_i$ and $\vec{X}_j$, at the vertices of the fitness landscape, given by $\left|fit_1(\vec{X}_i) - fit_1(\vec{X}_j)\right| + \left|fit_2(\vec{X}_i) - fit_2(\vec{X}_j)\right|$.

One of the most popular members of the MOO family is Non-dominated Sorting Bee Colony (NSBC) [6]. An overview of the main steps of the NSBC algorithm is presented next.

A. Initialization

NSBC starts with a population P_t of NP, D-dimensional food sources $\vec{X}_i(t) = \{x_{i,1}(t), x_{i,2}(t), ..., x_{i,D}(t)\}$ at the current generation $t = 0$ by uniformly randomizing in the range $\vec{X}^{\min} = \{x_1^{\min}, x_2^{\min}, ..., x_D^{\min}\}$ and $\vec{X}^{\max} = \{x_1^{\max}, x_2^{\max}, ..., x_D^{\max}\}$, and its k-th fitness function $fit_k(\vec{X}_i(0))$ is evaluated with i= [1, NP] and k= [1, n].

B. Employed Bee Phase

An employed bee obtains a new food source $\vec{X}'_i(t) = \{x_{i,1}(t), ..., x'_{i,j}(t), ..., x_{i,D}(t)\}$ in the neighborhood of $\vec{X}_i(t) = \{x_{i,1}(t), ..., x_{i,j}(t), ..., x_{i,D}(t)\}$ by computing $x_{i,j}'(t)$ parameter using the following expression for $j \in$ [1, D].

$$x'_{i,j}(t) = x_{i,j}(t) + rand_{i,j}(-1,1) \times (x_{i,j}(t) - x_{k,j}(t)) \qquad (1)$$

where $rand_{i,j}$(-1, 1) is a uniform random variable in [-1, 1] and k is randomly selected from the range [1, NP] but $k \neq i$. The k-th fitness function $fit_k(\vec{X}'_i(t))$ is evaluated for the food source $\vec{X}'_i(t)$ with i= [1, NP] and k= [1, n].

C. Selection by Employed Bee

If $\vec{X}'_i(t)$ dominates $\vec{X}_i(t)$, the bee memorizes $\vec{X}'_i(t)$ and forgets $\vec{X}_i(t)$. However if $\vec{X}_i(t)$ and $\vec{X}'_i(t)$ are non-dominated, she keeps both solutions in her memory P_t. This step is repeated for i= [1, NP] and hence, a population of food sources is obtained with size N in between NP and 2NP.

D. Non-dominated Sorting

The population P_t, thus obtained, is sorted into a number of Pareto fronts according to non-domination. All the non-dominated solutions of the current population are ranked one (named optimal Pareto front *Front_Set*(1)). The second front is formed by identifying the non-dominated solutions from the set P_t –*Front_Set*(1). The ranking process thus continues until all the non-dominated sets are identified and ranked as *Front_Set*(1), *Front_Set*(2), *Front_Set*(3), and so on.

E. Truncation of the extended Population

The parent population P'_t (of size NP) for the onlooker bee phase is constructed by selecting the non-dominated sets of solutions from P_t (of size N) according to the ascending order of their Pareto ranking. Let *Front_Set*(l) be the set beyond which no other set can be accommodated in P'_t (i.e., by adding *Front_Set*(l) its size exceeds NP). Then the solutions in *Front_Set*(l) are sorted in descending order of crowding distance CD. To ensure diversity in population the solutions in *Front_Set*(l) with the highest crowding distances are included in P'_t until its size becomes NP.

F. Probability Calculation

Let Set_i denotes the set of all food sources that are dominated by $\vec{X}_i(t)$ based on $fit_k(\vec{X}_i(t))$ for k= [1, n]. The probability of each food source $\vec{X}_i(t)$ to be selected by the onlooker bee is calculated by (2).

$$prob\,(i) = Set_i \,/\, NP \qquad (2)$$

G. Onlooker Bee Phase

An onlooker bee evaluates the fitness of food sources from all employed bees and probabilistically selects a food

source $\vec{X}_i(t)$ based on its probability *prob*(*i*), as calculated by the expression (2). After that, onlooker bee also produces a modification on the position as described in (1). The population P'_t of size between *NP* and 2*NP* is formed by following the principle as stated in section II. C. Then using the methodology of non-dominated sorting and crowding distance, the non-dominated food sources are found out from P'_t to form the resulting population P_{t+1} of size *NP*.

H. 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 position by the scout.

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

III. Optimization in Presence of Noise

The paper attempts to handle noise in NSBC by taking repeated samples of the fitness functions for each food source and then consider the expected value of the measured fitness samples as the fitness estimate. It is apparent that a large sample-size enhances the quality of estimated fitness (of a trial solution), at the cost of additional overhead in computational complexity. The trade-off (between quality and complexity) is balanced here by means of adaptive selection of sample-size in NSBC. It relies on the underlying premise that the fitness variance of a selected sub-population around a given food source anticipates a measure of the possible creeping of noise in the local neighborhood of the given food source. Hence, it is necessary to estimate the noise-level in the small neighborhood of each food source by assessing the variance of the (probably noisy) fitness measure of the food sources located at the small neighborhood around it. If the variance is high (i.e., above a user-defined threshold), the sample-size of the given food source is increased.

Let,

$\vec{X}_i = \{x_{i1}, x_{i2}, ..., x_{iD}\}$ be a *D*-dimensional food source with *n* noisy fitness functions,

$fit_k(\vec{X}_i)$ be the *k*-th fitness function of $\vec{X}_i$ for *k*= [1, *n*],

$fit_{k,l}(\vec{X}_i)$ be the *l*-th observed sample of $fit_k(\vec{X}_i)$,

$s_{k,i}$ be the used sample size of $fit_k(\vec{X}_i)$,

$v_{k,i}$ be the fitness variance in the neighborhood of $\vec{X}_i$ for the *k*-th fitness

$v_k^{\max}$ be the maximum fitness variance in the neighborhood of all food sources for the *k*-th objective

In a *D*-dimensional search space the neighborhood of a food source $\vec{X}_i$ is formed by selecting the food sources (from a population of size *NP*) lying within a hyperspace bounded by $\Delta\vec{X}_i^{\min} = \{x_{i,1}-\Delta x_1,\ x_{i,2}-\Delta x_2, \ldots,\ x_{i,D}-\Delta x_D\}$ and $\Delta\vec{X}_i^{\max} = \{x_{i,1}+\Delta x_1,\ x_{i,2}+\Delta x_2, \ldots,\ x_{i,D}+\Delta x_D\}$. The neighborhood of two or more solutions, however, may be overlapped. Here $\Delta x_j = (x_j^{\max} - x_j^{\min})/NP$ for *j*= [1, *D*].

The functional relationship of sample-size, $s_{k,i}$ against fitness variance in local neighborhood, $v_{k,i}$ (Fig. 1) here is selected as a non-linearity of the following form.

$$s_{k,i} = \frac{s_{\max} + s_{\min}}{2} + \frac{s_{\max} - s_{\min}}{\pi} \times \arctan(v_{k,i} - Th_k) \quad (3)$$

where Th_k denotes the (globally selected) threshold for the *k*-th fitness variance in local neighborhood, $s_{\min}$ and $s_{\max}$ represent predefined minimum and maximum sample-size, and $s_{av} = (s_{\max}+s_{\min})/2$ respectively. The lower quartile of the fitness variances in the neighborhood of each member in the population is favored as the global threshold. Hence the sample-size increases even when the noise-level is even one-fourth of its largest possible occurrence.

Traditional noisy optimization algorithms [1], [3] consider an average of the fitness samples for a given trial solution as the effectual measurement of the fitness. An averaging presumes equal probability of occurrence of all fitness samples, thereby, providing a poor fitness estimate. This, however, can be surmounted by considering expected fitness of the samples for a given food source. Let, $E_k(\vec{X}_i)$ and $V_k(\vec{X}_i)$ be the expected value and variance of the samples of $fit_k(\vec{X}_i)$, for *k*= [1, *n*]. We first determine the minimum and the maximum values of the observed fitness samples, say $fit_{k-\min}(\vec{X}_i)$ and $fit_{k-\max}(\vec{X}_i)$ respectively. Now the entire range $[fit_{k-\min}(\vec{X}_i), fit_{k-\max}(\vec{X}_i)]$ is divided into *q* intervals of equal length as shown in Fig. 2. The number of samples residing in the *j*-th interval (specified by [*j*-min, *j*-max]) is denoted by n_j for *j*= [1, *q*]. Then the probability of occurrence of fitness samples in the *j*-th interval, p_j, is calculated as follows.

$$p_j = n_j / s_{k,i} \quad (4)$$

We now obtain $E_k(\vec{X}_i)$ by (5) and $V_k(\vec{X}_i)$ by (6).

$$E_k(\vec{X}_i) = \sum_{j=1}^{q} p_j \times ((j\_\max + j\_\min)/2) \quad (5)$$

$$V_k(\vec{X}_i) = \sum_{j=1}^{q} p_j \times ((j\_\max + j\_\min)/2)^2 - (E_k(\vec{X}_i))^2 \quad (6)$$

Although adaptive sample-size selection strategy attempts to diminish the impact of noise in MOO problem, it cannot guarantee that the fitness-based ranking adopted for placement of the members in the Pareto fronts (during Non-dominated Sorting) to be always accurate. To give a relatively worse-looking member a possibility to occupy the optimal Pareto front, Goldberg proposed the following

strategy [5]. The strategy verifies whether the difference between the sample means of individual members is less than the scaled sum of their variances, i.e.,

$$|\mu_k(\vec{X}_i)-\mu_k(\vec{X}_j)|<K\sqrt{(V_k(\vec{X}_i)-V_k(\vec{X}_j))/2} \quad (7)$$

where, $\mu_k(\vec{X}_i)$ denotes the mean sample value of $fit_k(\vec{X}_i)$ and K is called 'neighborhood restriction parameter'. Here, Goldberg's method is amended as follows:

1. The expectation being a better measure of the statistical fitness, we substitute mean by expectation of the fitness samples for individual food source.
2. The sample-size of two food sources being different, here, we replace variance in (7) by the pooled variance given as follows.

$$P\_V_k(\vec{X}_i,\vec{X}_j)=\frac{(s_{k,i}-1)\times V_k(\vec{X}_i)+(s_{k,j}-1)\times V_k(\vec{X}_j)}{s_{k,i}+s_{k,j}-2} \quad (8)$$

Incorporation of above mentioned points in (7), yields a new criterion (9) for analyzing Pareto co-ranking of two members of the population.

$$|E_k(\vec{X}_i)-E_k(\vec{X}_j)|<K\sqrt{P\_V_k(\vec{X}_i,\vec{X}_j)\left(\frac{1}{s_{k,i}}+\frac{1}{s_{k,j}}\right)}, \text{ for } k=[1,n] \quad (9)$$

To keep the slightly worse solutions an opportunity to enter the optimal Pareto front during the early exploration phase of the NNSBC and restrict their ingress gradually, expression (10) is employed to design K that captures the above requirement automatically.

$$K = C(1-\sigma^2)^t \quad (10)$$

Here, σ^2 is the variance of the noise in the fitness function.

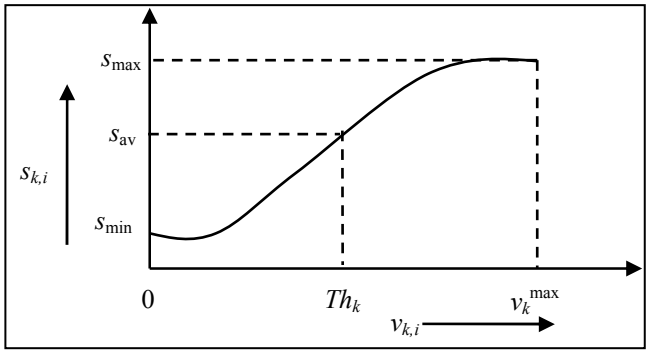

Fig. 1. The non-linearity used to adapt sample size with fitness variance in local neighborhood

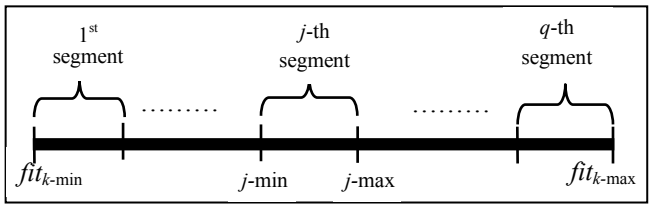

Fig. 2. Fitness intervals in the sample space

The pseudo code of NNSBC is given below.

Procedure NNSBC

Begin

1. Initialize a population P_t of NP, D-dimensional food sources $\vec{X}_i(t)$ at generation t=0 with $trial_i$=0 for i= [1, NP].
2. Evaluate $fit_{k,l}(\vec{X}_i(t))$ for i= [1, NP], k= [1, n] and l= [1, s_{min}].
3. Evaluate $E_k(\vec{X}_i(t))$ and $V_k(\vec{X}_i(t))$ using (5) and (6) respectively from the measured fitness samples for i= [1, NP] and k= [1, n].
4. **While** termination condition is not reached **do**

   Begin

   //Employed Bee Phase

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

   4.2. Determine the k-th fitness variance $v_{k,i}$ in the local neighborhood of $\vec{X}_i'(t)$ for i= [1, NP] and k= [1, n].

   4.3. Sort $v_{k,i}$ for i= [1, NP] in ascending order and select the lower quartile as Th_k for k= [1, n].

   4.4. Determine sample size $s_{k,i}$ of the k-th fitness of $\vec{X}_i'(t)$ using (3) for i= [1, NP] and k= [1, n].

   4.5. Evaluate $E_k(\vec{X}_i'(t))$ and $V_k(\vec{X}_i'(t))$ using (5) and (6) respectively from $fit_{k,l}(\vec{X}_i'(t))$ for i= [1, NP], k= [1, n] and l= [1, $s_{k,i}$].

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

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

   Else $P_t \leftarrow P_t \bigcup \vec{X}_i'(t)$;

   End If

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

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

   4.8. For each $\vec{X}_i(t) \in$ *Front_Set*(1) and $\vec{X}_j(t) \notin$ *Front_Set*(1), place $\vec{X}_j(t)$ in *Front_Set*(1) if (9) is satisfied.

   4.9. Truncate P_t of size $NP<N<2NP$ to P'_t using crowding distance criterion and the principle as stated in section II.E.

   //Onlooker Bee Phase

   4.10. Select a food source $\vec{X}_i(t) \in P'_t$ based on its probability of selection *prob*(i) for calculated using (2) for i= [1, NP].

   4.11. Repeat from 4.1 to 4.9 with $P_{t+1} \leftarrow P'_t$ in step 4.9.

   4.12. Reinitialize food source with maximum *trial* value exceeding "limit" by a **scout**.

   4.13. $t \leftarrow t$+1.

   End While

End

IV. Experiments and Results

The performance of the proposed NNSBC algorithm is examined here with respect to noisy-version of 23 CEC-2009

recommended multi-objective benchmark functions [13]. The noisy version of the k-th objective $f_k(\vec{X})$ for k= [1, n] is given by

$$f_{k-noisy}(\vec{X}) = f_k(\vec{X}) + \eta_k \tag{11}$$

where η_k is the injected stochastic noise amplitude that follows certain probability density function (PDF). The following three variants of η_k are considered in this paper.

1. **Gaussian:** η_k has a **Gaussian PDF**, which is given by

$$f(\eta_k) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(\eta_k - m)^2}{2\sigma^2}} \tag{12}$$

where m and σ^2 denote the mean and variance of the Gaussian PDF. The injection of Gaussian noise η_k here is performed using a well-known technique called Box-Muller method [14].

2. **Poisson:** η_k has a **Poisson PDF**, which is given by

$$f(\eta_k) = \frac{\lambda^{\eta_k} . e^{-\lambda}}{\eta_k !} \tag{13}$$

where λ represents mean (as well as variance) in Poisson distribution. We used Knuth's algorithm [15] to inject η_k into the fitness landscape following Poisson distribution.

3. **Random:** Lastly we consider η_k to be a **random** noise with maximum noise amplitude within ±25% of the fitness amplitudes. Here, random noise is generated using linear congruential pseudo random number generator [16].

Comparative Framework: The MOO algorithms used for the comparative study include DEMON (Differential Evolution for Multi-objective Optimization with Noise) [17], NSGA-II with α-dominance operator (NSGA-II-A) [10], elitist Evolutionary Multi-Agent System (elEMAS) [11], extended MOPSO [17], extended NSGA-II [17], modified NSGA-II [5], Noise-Tolerant Strength Pareto Evolutionary Algorithm (NT-SPEA) [2], DEMO-RSF-TS [12] [17] and traditional NSBC [6]. Similarly, traditional DEMO, MOPSO and NSGA-II are also extended with the noise handling strategies introduced in this paper to develop three new members of our comparative framework, denoted as Noisy DEMO (NDEMO), Noisy MOPSO (NMOPSO) and Noisy NSGA-II (NNSGA-II).

Parameter Settings: For all the algorithms the population size is kept at 50 and the maximum function evaluations (FEs) is set as 100000, 300000 and 500000 for 10-*D*, 30-*D* and 50-*D* problems respectively. To make the comparison fair, the population for all the algorithms (over all problems tested) is initialized using the same random seeds. We employ the best parametric set-up for all these algorithms as prescribed in their respective sources. In our proposed NNSBC algorithm, the minimum and maximum sample-size are considered to be 2 and 30 respectively with limit of 50.

A. Performance Metrics

Inverted Generational Distance (IGD): Let P^* be a set of uniformly distributed points along the optimal Pareto front (in the objective space). Let A be an approximate set to the Pareto front, the *IGD* metric can be defined as follows.

$$IGD(A, P^*) = \frac{\Sigma_{v \in P^*} d(v, A)}{|P^*|} \tag{14}$$

Here $d(v, A)$ is the minimum Euclidean distance between v and the points in A. A lower value of *IGD* is required to ensure that the approximate Pareto front, obtained by the proposed MOO, is very close to the optimal Pareto front.

Spacing (S): The metric measures the range variance of the neighboring vectors in the non-dominated vectors found by the algorithm. It is defined as follows.

$$S = \sqrt{\frac{1}{M-1}\sum_{i=1}^{M}(\bar{d} - d_i)^2}, \bar{d} = \frac{1}{M}\sum_{i=1}^{M} d_i \tag{15}$$

Here $d_i = \min_{j=1, j\neq i}^{M} \sum_{k=1}^{n} \left| E_k(\vec{X}_i) - E_k(\vec{X}_j) \right|$ with M as the number of non-dominated vectors found by the method. Here $\vec{X}_i$ and $\vec{X}_j$ are non-dominated vectors. A value of zero of this metric signifies all members of the approximate Pareto front are equidistantly spaced.

Error Ratio (ER): This metric is defined as follows.

$$ER = \frac{\sum_{i=1}^{M} e_i}{M}, \quad e_i = \begin{cases} 0, \text{ if } \vec{X}_i \in A \text{ and } \vec{X}_i \in P^* \\ 1, \text{ if } \vec{X}_i \in A \text{ and } \vec{X}_i \notin P^* \end{cases} \tag{16}$$

In ideal case, this metric should have a zero value indicating that all the non-dominated solutions in the approximate Pareto front belong to the optimal Pareto front.

Hyper Volume Ratio (HVR): It is defined as follows.

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

Here $HV(A)$ and $HV(P^*)$ symbolize the hyper-volume of the approximate Pareto front A and optimal Pareto front P^* respectively. The size of the objective space covered by a set of non-dominated solutions S is envisaged as its hyper-volume $HV(S)$. In ideal case, the $HVR(A)$ indicator attains its maximum value 1 if and only if its non-dominated vectors in the objective space are identical with the members of the optimal Pareto-front P^*.

B. Analysis and Results

The mean and standard deviation (within parenthesis) of *IGD* metric for 50 independent runs (each with 300000 FEs)

of each of the thirteen algorithms are presented in Table-I with η_k to be the Gaussian noise (mean=0 and variance, σ^2=0.4). The best solution in each case has been shown in bold. Although all the experiments are performed for all three variants of noise with noise variance $\sigma^2 \in [0, 1]$ and also for 10, 30 and 50-dimensional problems, we report here the results for 30 dimensions only and for a finite value of σ^2 to save space. Please note that the results omitted follow a similar trend like those reported in Table-I.

Since all the algorithms commence with the same initial population over each problem instance, we use paired t-tests to compare the means of the results produced by the best and the second best algorithms. The statistical significance level of the difference of the means of NNSBC and the best of the remaining twelve algorithms is presented in the 15-th column of Table-I. Here '+' indicates the t-value of 49 degrees of freedom is significant at a 0.05 level of significance by two-tailed test, '–' means the difference of means is not statistically significant and 'NA' stands for Not Applicable, referring to the cases for which two or more algorithms achieve the best accuracy results.

The simulation results in Table-I shows that NNSBC outperforms its competitors over 21 cases out of 23 benchmark instances. Out of these 21 cases, for 20 instances the difference between the mean of *IGD*s of NNSBC and its nearest competitor is statistically significant. In one case (UF7), NDEMO achieves the best average *IGD*-value outperforming NNSBC, which remains the second best algorithm.

Fig. 3 shows the evolution of the average *S*-metric values of the population with the noise variance in all the thirteen algorithms keeping the number of generations to be fixed at 300000 for Poisson noise. The plot of the average *ER*-metric against problem dimension (within [10, 50]) is given in Fig. 4. The convergence characteristics of three difficult test functions with different settings of noise variance are shown in Fig. 5 in terms of the *IGD*, *S* and *ER* metric values of the median run of each algorithm. A close scrutiny of the results in Table-I and the plots reveal that NNSBC is much more competent and capable to capture the optima in the noisy fitness landscape than its competitor algorithms on all the test instances, irrespective of the form and variance of noise.

Two non-parametrical statistical tests, known as Friedman and Iman-Davenport tests are also performed on the mean of *HVR* metric for 50 independent runs of each of the thirteen algorithms. With the level of significance α=0.05, both the statistics show significant differences in operators with test values of 275.7054 and 20591.88, respectively, and $p<0.001$. The rankings obtained by Friedman procedure in Table-II highlight NNSBC as the best algorithm, so Bonferroni-Dunn post-hoc analysis is performed with NNSBC as the control method. Here, the critical difference (*CD*) comes as 2.639. The interpretation of this measure is that the performance of two algorithms is significantly different, only if the corresponding average Friedman ranks differ by at least by *CD*, which is depicted in Fig. 6. It can be perceived that only for NDEMO and NMOPSO, the null hypothesis cannot be rejected with any of the tests for α = 0.05. The other ten algorithms, however, may be regarded as significantly worse than the NNSBC.

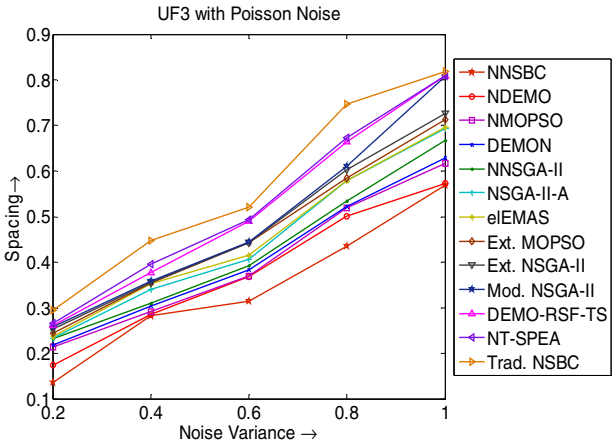

Fig. 3. Plot of average *S* metric values with Poisson noise variance σ^2 for 300000 FEs for UF3

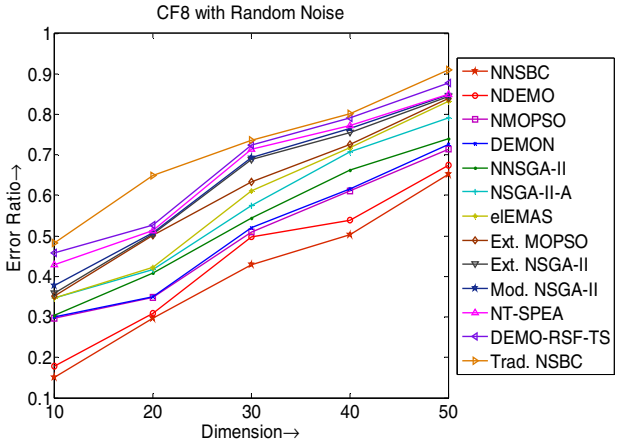

Fig. 4. Plot of average *ER* metric values with problem dimension for random noise and 300000 FEs for CF8

TABLE II. AVERAGE RANKINGS FROM FRIEDMAN'S TEST

Algorithm	Ranking
NNSBC	**1.109**
NDEMO	1.891
NMOPSO	3.000
DEMON	4.000
NNSGA-II	5.000
NSGA-II-A	6.000
elEMAS	7.000
Extended MOPSO	8.000
Extended NSGA-II	9.000
Modified NSGA-II	10.00
DEMO-RSF-TS	11.00
NT-SPEA	12.00
Traditional NSBC	13.00
Critical Difference α=0.05	2.639

V. CONCLUSION

We proposed a novel approach to uncertainty management in the noisy fitness landscape of NSBC algorithm. The merit of the proposed work lies in the following counts: i) adaptive selection of sample-size in the local neighborhood of individual food source, ii) expectation of the fitness samples, and iii) extending Goldberg's method

using a more statistical basis of comparison. The experiments undertaken reveal that the proposed extension, called NNSBC, outperforms its competitor algorithms with respect to four standard metrics: *IGD*, *S*, *ER* and *HVR* in presence of three different noise distributions (Gaussian, Poisson and random noise of limited amplitude) on the standard 23 benchmark functions.

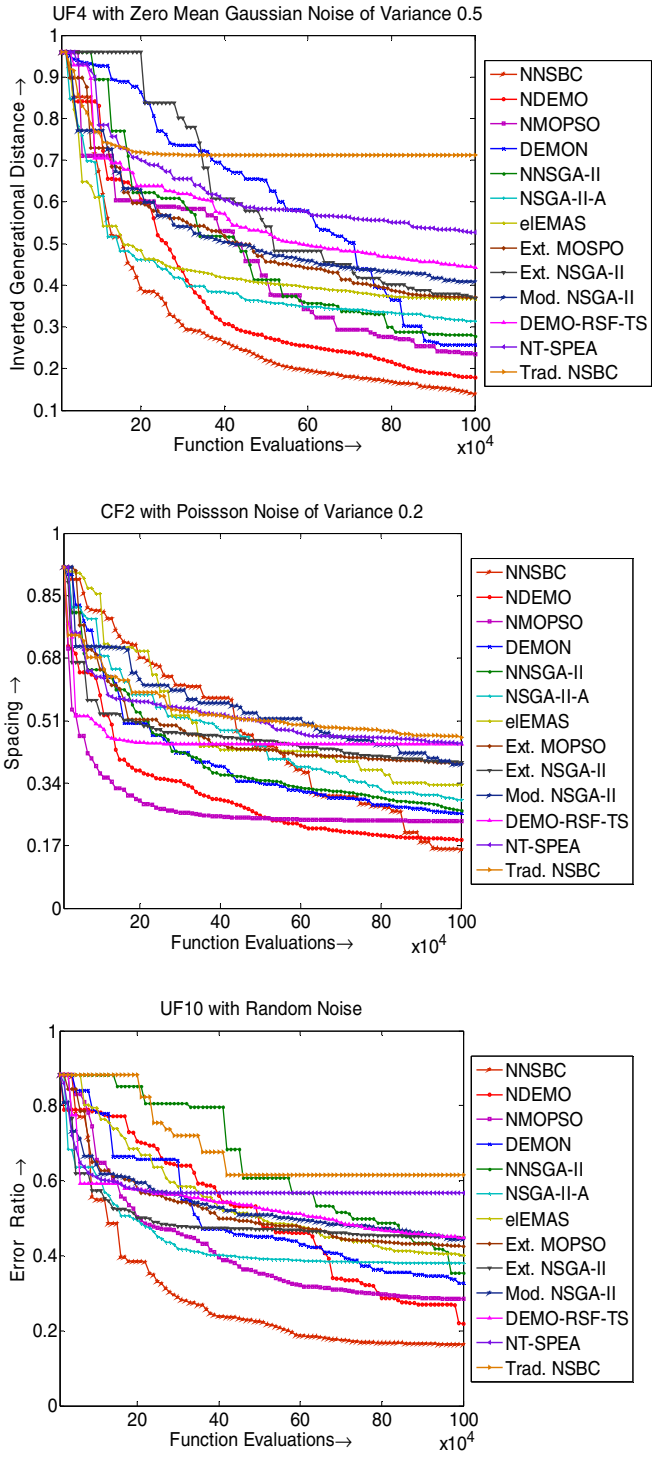

Fig. 5. Plot of average *IGD*, *S* and *ER* metric values with FEs for different settings of noise

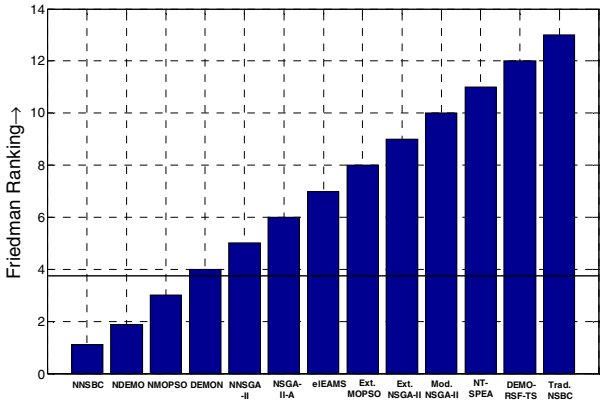

Fig. 6. Graphical representation of Bonferroni-Dunn's procedure considering NNSBC as control method

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 greatly acknowledged for the present work.

REFERENCES

[1]. B. L. Miller, *Noise, Sampling, and Efficient Genetic Algorithms*, Ph. D. dissertation, Dept. of Computer Science, Univ. Illinois at Urbana-Champaign, Urbana, IL, 1997, available as TR 97001.

[2]. D. Buche, P. Stall, R. Dornberger, and P. Koumoutsakos, "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, 2002, pp. 460-473.

[3]. E. J. Hughes, "Evolutionary Multi-objective Ranking with Uncertainty and Noise," in *Proceedings of Evolutionary Multi-Criterion Optimization*, vol. 1993, 2001.

[4]. A. Singh, *Uncertainty Based Multi-objective Optimization of Groundwater Remediation Design*, M.S. thesis, Univ. Illinois at Urbana-Champaign, Urbana, IL, 2003.

[5]. M. Babbar, A. Lakshmikantha, D.E. Goldberg, "A Modified NSGA-II to solve Noisy Multi-objective Problems," in *Proceedings of Conference on Genetic Evolutionary Computation*, 2003.

[6]. P. Rakshit, A. K. Sadhu, P. Bhattacharya, A. Konar, and R. Janarthanan, "Multi-robot Box-Pushing using Non-dominated Sorting Bee Colony Optimization Algorithm," in *Proceedings of Swarm, Evolutionary and Memetic Computing*, Lecture Notes in Computer Science, vol. 7076, 2011, pp. 601-609.

[7]. T. Robic, and B. Philipic, "DEMO: Differential Evolution for Multiobjective Optimization," in *Proceedings of the third International Conference on Evolutionary Multi-Criterion Optimization*, Coello Coello, C.A., Aguirre, A.H., Zitzler, E., Eds.; Springer Lecture Notes in Computer Science: Guanajuato, Mexico, vol. 3410, 2005, pp. 520-533.

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

[9]. K. Deb, A. P. S. Agarwal, and T. Meyarivan, "A Fast and Elitist Multi-objective Genetic Algorithm: NSGA II," *IEEE Transactions on Evolutionary Computation*, vol. 2, 1998, pp. 162-197.

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

[11]. L. Siwik, and S. Natanek, "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.

[12]. S. Das, A. Konar, and U. K. Chakraborty, "Improved Differential Evolution Algorithms for Handling Noisy Optimization Problems," in *Proceedings of IEEE Congress of Evolutionary Computation*, vol. 2, September, 2005, pp. 1691–1698.

[13]. Q. Zhang, A. Zhou, S. Zhao, P. N. Suganthan, W. Liu, and S. Tiwari, *Multi-objective 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, 2008.

[14]. G. E. P. Box, and M. E. Muller, "A note on the generation of random deviates," *the Annals of Mathematical Statistics*, vol. 29, 1958, pp. 610-611.

[15]. D. E. Knuth, *Seminumerical Algorithms*, the Art of Computer Programming, vol. 2, 1969, Addison Wesley.

[16]. J. Bolte, *Linear Congruential Generators*, Wolfram Demonstrations Project.

[17]. P. Rakshit, A. Konar, S. Das, L. C. Jain, and A. K. Nagar, "Uncertainty Management in Differential Evolution Induced Multi-objective Optimization in Presence of Measurement Noise," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. PP, no. 99, October, 2013, pp. 1-16.

TABLE I. MEAN IGD VALUES OVER 50 INDEPENDENT RUNS GAUSSIAN NOISE WITH MEAN=0 AND VARIANCE σ^2=0.4

Functions	NNSBC	NDEMO	NMOPSO	DEMON	NNSGA-II	NSGA-II-A	elEAMS	Extended MOPSO	Extended NSGA-II	Modified NSGA-II	NT-SPEA	DEMO-RSF-TS	Traditional NSBC	Statistical Significance
UF1	**0.0082**	0.0103	0.0111	0.0111	0.0118	0.0205	0.0277	0.0309	0.0569	0.0583	0.0718	0.0741	0.0861	+
	(0.000)	(0.000)	(0.000)	(0.000)	(0.000)	(0.031)	(0.000)	(0.046)	(0.515)	(0.4717)	(0.056)	(0.001)	(0.067)	
UF2	**0.0051**	0.0063	0.0068	0.0069	0.0076	0.0191	0.0216	0.0254	0.0462	0.0466	0.0510	0.0550	0.0586	+
	(0.000)	(0.000)	(0.001)	(0.000)	(0.001)	(0.013)	(0.000)	(0.010)	(0.015)	(0.015)	(0.018)	(0.002)	(0.019)	
UF3	**0.0595**	0.0730	0.0817	0.0828	0.0927	0.1171	0.1599	0.1614	0.2497	0.2508	0.2624	0.2832	0.2871	+
	(0.002)	(0.003)	(0.007)	(0.008)	(0.011)	(0.015)	(0.007)	(0.007)	(0.016)	(0.016)	(0.017)	(0.017)	(0.017)	
UF4	**0.0421**	0.0517	0.0525	0.0526	0.0535	0.0556	0.0567	0.0568	0.0640	0.0650	0.0749	0.0814	0.0816	+
	(0.000)	(0.000)	(0.000)	(0.000)	(0.000)	(0.001)	(0.000)	(0.000)	(0.001)	(0.001)	(0.001)	(0.001)	(0.001)	
UF5	**0.3266**	0.4009	0.4128	0.4132	0.4209	0.4776	0.4875	0.4938	0.5366	0.5457	0.6338	0.6667	0.6727	+
	(0.000)	(0.000)	(0.003)	(0.006)	(0.0086	(0.011)	(0.009)	(0.007)	(0.016)	(0.016)	(0.021)	(0.014)	(0.024)	
UF6	**0.0801**	0.1237	0.0963	0.1263	0.1763	0.2033	0.2284	0.2458	0.3181	0.3207	0.3462	0.3489	0.3550	+
	(0.000)	(0.003)	(0.000)	(0.005)	(0.007)	(0.001)	(0.010)	(0.000)	(0.001)	(0.001)	(0.001)	(0.019)	(0.002)	
UF7	0.0210	**0.0171**	0.0363	0.0367	0.0515	0.1103	0.1579	0.1592	0.1763	0.1767	0.1811	0.1832	0.1845	–
	(0.008)	**(0.008)**	(0.009)	(0.009)	(0.015)	(0.010)	(0.018)	(0.009)	(0.013)	(0.013)	(0.013)	(0.026)	(0.014)	
UF8	**0.2018**	0.2477	0.5547	0.5723	0.1626	0.1398	0.1723	0.2246	0.2794	0.2817	0.3046	0.3219	0.3388	+
	(0.008)	(0.009)	(0.010)	(0.012)	(0.016)	(0.019)	(0.018)	(0.018)	(0.023)	(0.023)	(0.023)	(0.024)	(0.026)	
UF9	**0.0487**	0.0598	0.0805	0.0845	0.0967	0.1699	0.1664	0.2037	0.3075	0.3114	0.3492	0.3727	0.3799	+
	(0.000)	(0.000)	(0.001)	(0.003)	(0.006)	(0.000)	(0.007)	(0.000)	(0.000)	(0.000)	(0.001)	(0.024)	(0.001)	
UF10	**0.4720**	0.5793	0.6112	0.6226	0.7254	0.8938	0.9706	0.9793	1.9858	2.0273	2.4271	2.6090	2.6251	+
	(0.015)	(0.019)	(0.029)	(0.037)	(0.045)	(0.020)	(0.046)	(0.018)	(0.020)	(0.020)	(0.021)	(0.067)	(0.024)	
UF11	**0.2810**	0.3449	0.3583	0.3613	0.3779	0.4922	0.4934	0.4948	0.5136	0.5146	0.5251	0.5320	0.5332	+
	(0.000)	(0.001)	(0.002)	(0.008)	(0.017)	(0.029)	(0.021)	(0.024)	(0.033)	(0.033)	(0.036)	(0.030)	(0.036)	
UF12	**510.20**	626.23	697.16	711.35	760.49	726.73	735.48	739.62	767.56	768.23	774.73	786.10	804.21	+
	(0.007)	(0.009)	(0.198)	(0.270)	(0.324)	(0.216)	(0.499)	(0.327)	(0.381)	(0.384)	(0.413)	(0.644)	(0.663)	
UF13	**2.3230**	2.8513	3.0269	3.0563	3.1549	3.0461	3.7010	3.8725	4.2427	4.2545	4.3680	4.5156	4.5249	+
	(0.061)	(0.075)	(0.094)	(0.112)	(0.154)	(0.172)	(0.226)	(0.163)	(0.214)	(0.214)	(0.215)	(0.338)	(0.218)	
CF1	**0.0086**	0.0105	0.0105	0.0115	0.0140	0.0122	0.0135	0.0138	0.0574	0.0591	0.0764	0.0843	0.0850	+
	(0.000)	(0.000)	(0.000)	(0.000)	(0.000)	(0.001)	(0.000)	(0.001)	(0.001)	(0.001)	(0.001)	(0.000)	(0.001)	
CF2	**0.0055**	**0.0055**	0.0085	0.0089	0.0112	0.0275	0.0233	0.0243	0.0379	0.0386	0.0463	0.0513	0.0522	NA
	(0.000)	**(0.000)**	(0.000)	(0.0003)	(0.000)	(0.021)	(0.000)	(0.021)	(0.023)	(0.023)	(0.024)	(0.002)	(0.026)	
CF3	**0.0702**	0.0862	0.1156	0.1214	0.1417	0.1060	0.1056	0.1173	0.1969	0.1988	0.2173	0.2497	0.3012	+
	(0.000)	(0.000)	(0.002)	(0.005)	(0.006)	(0.005)	(0.007)	(0.018)	(0.020)	(0.020)	(0.021)	(0.016)	(0.028)	
CF4	**0.0396**	0.0486	0.0519	0.0524	0.0543	0.0505	0.0519	0.0612	0.08118	0.0818	0.0879	0.0958	0.0963	+
	(0.000)	(0.000)	(0.000)	(0.000)	(0.001)	(0.008)	(0.001)	(0.008)	(0.014)	(0.014)	(0.014)	(0.001)	(0.015)	
CF5	**0.3335**	0.4094	0.4310	0.4331	0.4568	0.4327	0.4337	0.4344	0.4808	0.4873	0.5503	0.5921	0.5932	+
	(0.000)	(0.000)	(0.007)	(0.007)	(0.007)	(0.063)	(0.008)	(0.062)	(0.068)	(0.068)	(0.071)	(0.015)	(0.086)	
CF6	**0.0817**	0.1003	0.1520	0.1537	0.1866	0.1918	0.2003	0.2053	0.2388	0.2459	0.3150	0.3408	0.3455	+
	(0.000)	(0.001)	(0.005)	(0.008)	(0.016)	(0.000)	(0.018)	(0.000)	(0.001)	(0.001)	(0.001)	(0.028)	(0.002)	
CF7	**0.0212**	0.0260	0.0637	0.0673	0.1470	0.0536	0.0598	0.0767	0.1470	0.1495	0.1743	0.1770	0.1828	+
	(0.000)	(0.000)	(0.010)	(0.010)	(0.008)	(0.008)	(0.021)	(0.007)	(0.008)	(0.008)	(0.011)	(0.026)	(0.016)	
CF8	**0.0653**	0.0802	0.0802	0.0968	0.1125	0.0990	0.0975	0.1096	0.2643	0.2683	0.3075	0.3261	0.3385	+
	(0.002)	(0.002)	(0.002)	(0.003)	(0.006)	(0.000)	(0.006)	(0.000)	(0.000)	(0.000)	(0.000)	(0.011)	(0.000)	
CF9	**0.0633**	0.0777	0.1148	0.1170	0.1521	0.0925	0.1540	0.2098	0.2680	0.2705	0.2949	0.3133	0.3313	+
	(0.007)	(0.009)	(0.009)	(0.010)	(0.013)	(0.007)	(0.014)	(0.006)	(0.007)	(0.007)	(0.010)	(0.020)	(0.011)	
CF10	**0.4880**	0.5990	0.7831	0.8187	0.9262	0.6881	0.8508	0.9265	2.0502	2.0796	2.3623	2.5003	2.8120	–
	(0.013)	(0.016)	(0.042)	(0.043)	(0.062)	(0.015)	(0.062)	(0.014)	(0.015)	(0.015)	(0.017)	(0.086)	(0.020)	