Extended Differential Evolution Algorithm for Worst-Case Value Minimization Problems

Kiyoharu Tagawa and Taiki Suenaga

Abstract—In many real-world optimization problems, a wide range of uncertainties have to be taken into account. Therefore, real-world optimization problems can be regarded as uncertain optimization problems. In uncertain optimization problems, objective function values observed are corrupted by noise. A number of evolutionary algorithms reported to solve uncertain optimization problems use the average of randomly sampled values as an estimate of the true objective function value. In this paper, a predicted upper bound of the noisy objective function's values is used to evaluate the quality of solutions of uncertain optimization problems. This is because we should prepare for the worst possible pass in real-world optimization problems. In accordance with the new criterion of solutions, uncertain optimization problems are formulated as worst-case value minimization problems in which the predicted upper bound of the noisy objective function's values is minimized. An extended variant of a recently-developed evolutionary algorithm, namely Differential Evolution (DE), is also proposed to solve the worst-case value minimization problems effectively. In order to allocate the computing budget to promising solutions, the extended DE adopts two pruning techniques, namely prediction interval and cutoff point, which can judge and discard hopeless solutions only by a single sampling.

Keywords—Uncertain optimization problem, probabilistic method, prediction interval, evolutionary algorithm, differential evolution.

I. Introduction

IN MANY real-world optimization problems, a wide range of uncertainties have to be taken into account. Therefore, they can be regarded as uncertain optimization problems. In uncertain optimization problems, objective function values observed are corrupted by noise. A number of Evolutionary Algorithms (EAs) have been reported to solve uncertain optimization problems [1], [2], [3], [4], [5]. Conventional EAs for uncertain optimization problems use the average of randomly sampled values as an estimate of the true objective function value. However, because noise is inevitable in real-world optimization problems, we should prepare for the worst possible pass in uncertain optimization problems [6], [7].

In this paper, we provide a new class of uncertain optimization problems. First of all, we suppose that the objective function is subject to noise, and the underlying distribution of noise also depends on the decision variables. Therefore, the new class of uncertain optimizations problem can be regarded as a general form of the noisy problem in which the robustness of solutions is required too. We also define a new criterion of solutions based on the statistical estimation of the worst objective function value. In accordance with the new criterion of solutions, we formulate uncertain optimization problems as worst-case value minimization problems in which a predicted upper bound of the noisy objective function's values is minimized. Multiple sampling is a well-known method to overcome noisy and robust problems [1], [2]. Multiple sampling suggests evaluating the same solution for N times and approximating the true objective function value by averaging. On the other hand, we use the set of randomly sampled N values to predict the worst value of the noisy objective function.

Differential Evolution (DE) [8], [9], [10] is a recently-developed EA. DE is arguably one of the most powerful stochastic real-parameter optimization algorithms in current use. Due to its simple but powerful searching capability, DE has been used successfully in many scientific and engineering applications [11], [12], [13], [14], [15]. As well as other EAs, DE attracts attention as a method not only for deterministic optimization problems, but also for uncertain optimization problems because of is characteristics of

1) direct optimization method that uses only objective function values,
2) stochastic search for global optimization,
3) self-averaging nature of population-based search.

Consequently, a number of variants of DE have been reported for conventional uncertain optimization problems as we will mention those variants later [3], [4], [5], [16], [17].

In this paper, an extended DE is proposed for solving new uncertain optimization problems, namely worst-case value minimization problems, in which the predicted worst value of noisy objective function is minimized. The worst value is predicted by multiple sampling that evaluates the same solution N times. In order to allocate the computing budget to promising solutions, the extended DE adopts two pruning techniques, namely prediction interval and cutoff point. Due to the two pruning techniques, the extended DE can judge and discard hopeless solutions only by a single sampling.

The organization of this paper is as follows. After this introduction, Section II attempts to provide an overview of the related work. Conventional uncertain optimization problems are categorized into four classes. Existing approaches to addressing different uncertainties are also presented and discussed. Section III provides the fundamentals of mathematical statistics for deriving the prediction interval of noisy objective function's value, followed by the formulation of a new class of uncertain optimization problems, namely worst-case value minimization problems. Section IV describes the cardinal DE that can be applied directly to the worst-case value minimization problem too. Section V proposes an extended DE that adopts prediction interval and cutoff point to allocate the computing budget to promising solutions. Through numerical

K. Tagawa is with School of Science and Engineering, Kinki University, Higashi-Osaka 577-8502 Japan (e-mail: tagawa@info.kindai.ac.jp).

T. Suenaga is with Graduate School of Science and Engineering Research, Kinki University, ditto.

experiments conducted on various benchmark problems, the performance of the extended DE is demonstrated in Section VI. Finally, concluding remarks are made in Section VII.

II. Related Work

Evolutionary Algorithms (EAs) often used to solve various optimization problems in the presence of a wide range of uncertainties. Generally, uncertain optimization problems can be categorized into the following four classes [2].

- **Noisy problem:** The objective function of the optimization problem is subject to noise. Noise in objective function may come from many different sources such as sensory measurement errors or randomized simulations [1], [3], [4], [5], [18]. Mathematically, a noisy objective function can be described as follows:

$$F(\vec{x}) = f(\vec{x}) + \varepsilon, \tag{1}$$

where $\vec{x} \in \Re^D$ is a vector of decision variables that can be changed by the algorithm, $f(\vec{x}) \in \Re$ is a time-invariant objective function, ε is additive noise which is often assumed to be normally distributed with zero mean and variance σ^2: $\varepsilon \sim \mathcal{N}(0,\ \sigma^2)$. Therefore, the noisy objective function $F(\vec{x}) \in \Re$ is also normally distributed with mean $\mu(\vec{x}) = f(\vec{x})$ and variance σ^2 as

$$F(\vec{x}) \sim \mathcal{N}(\mu(\vec{x}),\ \sigma^2). \tag{2}$$

Ideally, EAs should work on the expected objective function $f(\vec{x})$ and not be failed due to the presence of noise. However, during optimization, the only measurable objective function value is stochastic $F(\vec{x}) = f(\vec{x}) + \varepsilon$. Therefore, in practice, the the expected objective function value $f(\vec{x})$ is often approximated by an averaged sum of a number of random samples as follows:

$$F(\vec{x},\ N) = \frac{1}{N}\sum_{n=1}^{N}(f(\vec{x}) + \varepsilon_n), \tag{3}$$

where N is the number of samples.

- **Robust problem:** The decision variables are subject to perturbations or changes after the optimal solution has been determined. This kind of uncertainties concerns production tolerances and limitations, or actuator imprecision acting directly on the decision variables [19], [20], [21]. A common requirement is that a solution $\vec{x} \in \Re^D$ should still work satisfactorily when the design variables change slightly. Therefore, in this class of uncertain optimization problems, an objective function can be described as

$$F(\vec{x}) = f(\vec{x} + \vec{\varepsilon}), \tag{4}$$

where the probability distribution of the possible disturbance $\vec{\varepsilon} = (\varepsilon_1, \cdots, \varepsilon_D) \in \Re^D$ are often assumed to be independent of each other and normally distributed. Because an analytical closed form of the effective objective function in (4) is usually not available, it is often approximated using Monte Carlo integration as

$$F(\vec{x},\ N) = \frac{1}{N}\sum_{n=1}^{N} f(\vec{x} + \vec{\varepsilon}_n). \tag{5}$$

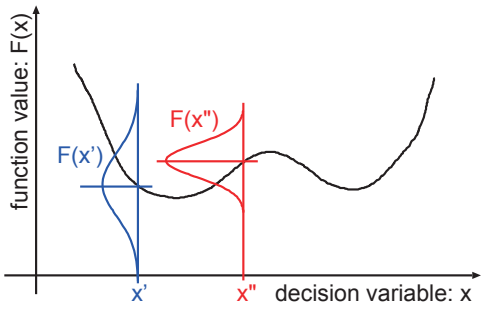


Fig. 1. Image of noisy objective function ($D = 1$)

- **Surrogate problem:** When the objective function is very expensive to evaluate, or an analytical function is not available, objective functions are often approximated based on data generated from experiments or simulations [16], [22]. The approximated objective function is often known as meta-model [23]. A meta-model is usually used together with the original objective function.
- **Dynamic problem:** The objective function is deterministic at any point in time, but is dependent on time. As a consequence, the optimum also changes over time. Thus, optimization algorithm should be able to continuously track the changing optimum rather than requiring a repeated restart of the optimization process [17], [24], [25]. The challenge here is to reuse information from previous environments to speedup optimization after a change.

III. Problem Formulation

A number of real-world applications can be formulated as a real-parameter optimization problem. In an orthodox optimization problem, the global optimal solution is a D-dimensional real-parameter vector $\vec{x} = (x_1, \cdots, x_j, \cdots, x_D) \in \Re^D$ that minimizes a deterministic objective function value. However, in many real-world optimization problems, objective function values observed are corrupted by noise. For many real-world optimization problems, we can suppose that the noisy objective function value $F(\vec{x})$ is distributed normally as

$$F(\vec{x}) \sim \mathcal{N}(\mu(\vec{x}),\ \sigma(\vec{x})^2), \tag{6}$$

where both the mean $\mu(\vec{x})$ and the variance $\sigma(\vec{x})^2$ of the normal distribution depend on the solution $\vec{x} \in \Re^D$.

In noisy problems, the variance σ^2 of noisy objective function values usually takes a constant. Fig. 1 shows an image of the new noisy objective function $F(\vec{x})$ in (6). As you can see in Fig. 1, not only the mean $\mu(\vec{x})$ but also the variance $\sigma^2(\vec{x})$ need to be minimized for improving the quality of solutions $\vec{x} \in \Re^D$. Thus, the definition of $F(\vec{x})$ in (6) can be regarded as a general form of noisy objective functions in (2).

Now, we think about any solution $\vec{x} \in \Re^D$. A prediction interval for a future observation $F \in \Re$ of $F(\vec{x})$ is calculated statistically for a given significant level α as

$$\mu(\vec{x}) - z_{\alpha/2}\,\sigma(\vec{x}) \le F \le \mu(\vec{x}) + z_{\alpha/2}\,\sigma(\vec{x}), \tag{7}$$

where $z_{\alpha/2}$ denotes the $\alpha/2$-quantile of the standard normal distribution $\mathcal{N}(0,\ 1)$ for which the probability becomes

$$P\left(-z_{\alpha/2} \leq \frac{F-\mu(\vec{x})}{\sigma(\vec{x})} \leq z_{\alpha/2}\right) = (1-\alpha). \tag{8}$$

Therefore, the future observation F of $F(\vec{x})$ falls in the prediction interval in (7) with the probability $(1-\alpha)$.

Because the mean $\mu(\vec{x})$ and the variance $\sigma(\vec{x})^2$ are usually unknown in real-world applications, we have to use the sample mean and the sample variance instead of them. First of all, we take a sample set $\{F_1, \cdots, F_n, \cdots, F_N\}$ of the observed $F(\vec{x})$ values. Thereby, the sample mean is calculated as

$$F(\vec{x},\ N) = \frac{1}{N}\sum_{n=1}^{N} F_n, \tag{9}$$

where $F(\vec{x},\ N)$ is used as estimate for the mean $\mu(\vec{x})$.

The sample variance is also calculated as

$$s(\vec{x},\ N)^2 = \frac{1}{N-1}\sum_{n=1}^{N} (F_n - F(\vec{x},\ N))^2, \tag{10}$$

where $s(\vec{x},\ N)^2$ is used as estimate for the variance $\sigma(\vec{x})^2$.

As shown in (6), the underlying distribution is the normal distribution. By using the sample mean and the sample variance instead of the mean and the variance, the normal distribution can be approximated by Student's t-distribution [26]. We have already obtained the sample set $\{F_1, \cdots, F_n, \cdots, F_N\}$. Let F_{N+1} be the $(N+1)$-th sample, or the future observation of $F(\vec{x})$. Then, the following ancillary statistic yields Student's t-distribution with $N-1$ degrees of freedom:

$$\frac{F_{N+1} - F(\vec{x},\ N)}{s(\vec{x},\ N)\sqrt{1+\dfrac{1}{N}}} \sim \mathcal{T}(N-1). \tag{11}$$

In the same way with (7), the prediction interval in which F_{N+1} will fall can be derived from (11) as follows [27]:

$$L(\vec{x},\ N) \leq F_{N+1} \leq U(\vec{x},\ N), \tag{12}$$

$$\begin{cases} L(\vec{x},\ N) = F(\vec{x},\ N) - \beta\, s(\vec{x},\ N), \\ U(\vec{x},\ N) = F(\vec{x},\ N) + \beta\, s(\vec{x},\ N), \end{cases} \tag{13}$$

$$\beta = t(N-1,\ \alpha/2)\sqrt{1+\frac{1}{N}}, \tag{14}$$

where $t(N-1,\ \alpha/2)$ denotes the $\alpha/2$-quantile of the Student's t-distribution with $N-1$ degrees of freedom, which is derived from a significant level α and the number of samples N.

The probability of F_{N+1} falling in the prediction interval described in (12) is $(1-\alpha)$. For example, if we want to calculate the 95% prediction interval for the future observation F_{N+1}, we choose the significant level as $\alpha = 0.05$ in (14).

In conventional noisy or robust problems [1], [2], the sample mean $F(\vec{x},\ N)$ in (9) is minimized for the optimal solution $\vec{x} \in \Re^D$. However, we consider the worst possible value of $F(\vec{x})$ caused by $\vec{x} \in \Re^D$ because noise is inevitable in real-world optimization problems. Therefore, we seek a solution $\vec{x} \in \Re^D$ that minimizes the upper bound $U(\vec{x},\ N)$ of the prediction interval in (12). Besides, we suppose that each of the decision variables $x_j \in \Re$ is limited to the range between the lower $\underline{x}_j$ and the upper $\overline{x}_j$ bounds. Thereby, the worst-case value minimization problem is formulated as

$$\left[\begin{array}{ll} \text{minimize} & U(\vec{x},\ N), \\ \text{subject to} & \underline{x}_j \leq x_j \leq \overline{x}_j,\ j = 1, \cdots, D, \end{array}\right. \tag{15}$$

where, to calculate the upper bound $U(\vec{x},\ N)$, a significant level α and the number of samples N are given in advance.

IV. Differential Evolution (DE)

DE can be applied directly to the optimization problem in (15). Like other EAs, DE holds N_P tentative solutions of the optimization problem, which are referred to as individuals, in the population $\mathbf{P}$. The i-th individual $\vec{x}_i \in \mathbf{P}$ is represented by a vector of decision variables $x_{j,i} \in \Re$ as follows:

$$\vec{x}_i = (x_{1,i}, \cdots, x_{j,i}, \cdots, x_{D,i}), \tag{16}$$

where $\underline{x}_j \leq x_{j,i} \leq \overline{x}_j$, $j = 1, \cdots, D$, and $i = 1, \cdots, N_P$.

Let $\mathrm{rand}_j[0,\ 1]$ be the random number generator that returns a uniformly distributed random number in the range $[0,\ 1]$. Thereby, an initial population $\vec{x}_i = (x_{1,i}, \cdots, x_{D,i}) \in \mathbf{P}$ is generated randomly within the range $[\underline{x}_j, \overline{x}_j]$ as

$$x_{j,i} = \mathrm{rand}_j[0,\ 1]\,(\overline{x}_j - \underline{x}_j) + \underline{x}_j, \tag{17}$$

where $j = 1, \cdots, D$ and $i = 1, \cdots, N_P$.

In order to generate a candidate for a new individual of the population $\mathbf{P}$, DE uses a unique strategy. The strategy of DE is defined by a series of three genetic operators, namely reproduction selection, differential mutation, and crossover [8]. Even though various strategies have been proposed [10], a basic strategy named "DE/rand/1/exp" is used.

First of all, each individual $\vec{x}_i \in \mathbf{P}$ is assigned to the "target vector" in turn. Except for the target vector, three other distinct individuals, say $\vec{x}_{r1}$, $\vec{x}_{r2}$, and $\vec{x}_{r3} \in \mathbf{P}$ $(i \neq r1 \neq r2 \neq r3)$, are selected randomly from the population $\mathbf{P}$. By using the above three individuals, the differential mutation generates a new vector $\vec{v} \in \Re^D$ called the "mutated vector" as

$$\vec{v} = \vec{x}_{r1} + S_F\,(\vec{x}_{r2} - \vec{x}_{r3}), \tag{18}$$

where the scale factor $S_F \in \Re$ is a control parameter.

In the basic strategy of DE, exponential crossover between the mutated vector $\vec{v}$ and the target vector $\vec{x}_i$ generates a candidate for an individual $\vec{u} \in \Re^D$ called the "trial vector". Each component u_j of the trial vector $\vec{u} \in \Re^D$ is inherited from either the mutated vector $\vec{v}$ or the target vector $\vec{x}_i$.

Algorithm 1 provides the pseudo-code describing the procedure of the basic strategy of DE, i. e., "DE/rand/1/exp". The subscript $j_r \in [1,\ D]$ at the 1st line is selected randomly, which ensures that $\vec{u}$ differs from the existing $\vec{x}_i \in \mathbf{P}$ for at least one component $u_{j_r} \in \Re$. In lines 2-5 of the pseudo-code, some elements of the trial vector $\vec{u}$ is generated by the differential mutation as shown in (18). The crossover rate $C_R \in [0,\ 1]$ at the 6th line is also a control parameter. In lines 6-8 of the pseudo-code, the other elements of the trial vector $\vec{u}$ is inherited from the target vector $\vec{x}_i \in \mathbf{P}$. Finally, if a component u_j of the trial vector $\vec{u}$ is made out of the range $[\underline{x}_j,\ \overline{x}_j]$, it will be returned to the range in lines 10-17.

Algorithm 1 : STRATEGY (DE/rand/1/bin)

1: $j := j_r$;
2: **repeat**
3: $u_j := x_{j,r1} + S_F\,(x_{j,r2} - x_{j,r3})$;
4: $j := (j \bmod D) + 1$;
5: **until** $\mathrm{rand}_j[0,\,1] < C_R \ \wedge \ j \neq j_r$;
6: **while** $j \neq j_r$ **do**
7: $u_j := x_{j,i}$;
8: $j := (j \bmod D) + 1$;
9: **end while**
10: **for** $j := 1$ to D **do**
11: **if** $u_j < \underline{x}_j$ **then**
12: $u_j := x_{j,r1} + \mathrm{rand}_j[0,\,1]\,(\underline{x}_j - x_{j,r1})$;
13: **end if**
14: **if** $u_j > \overline{x}_j$ **then**
15: $u_j := x_{j,r1} + \mathrm{rand}_j[0,\,1]\,(\overline{x}_j - x_{j,r1})$;
16: **end if**
17: **end for**

Algorithm 2 : DIFFERENTIAL EVOLUTION (DE)

1: **for** $i := 1$ to N_P **do**
2: $\vec{x}_i :=$ RANDOMLY_GENERATE();
3: $U(\vec{x}_i,\,N) :=$ EVALUATE_N_TIME($\vec{x}_i$);
4: **end for**
5: **repeat**
6: **for** $i := 1$ to N_P **do**
7: $\vec{u} :=$ STRATEGY($\vec{x}_i,\,\vec{x}_{r1},\,\vec{x}_{r2},\,\vec{x}_{r3} \in \mathbf{P}$);
8: $U(\vec{u},\,N) :=$ EVALUATE_N_TIME($\vec{u}$);
9: **if** $U(\vec{u},\,N) \leq U(\vec{x}_i,\,N)$ **then**
10: $\vec{x}_i := \vec{u}$;
11: $U(\vec{x}_i,\,N) := U(\vec{u},\,N)$;
12: **end if**
13: **end for**
14: **until** a termination condition is satisfied;
15: Output $\vec{x}_b \in \mathbf{P}$ with the minimum $U(\vec{x}_b,\,N)$;

Algorithm 2 provides the pseudo-code of the cardinal DE applied directly to the worst-case value minimization problem in (15). First of all, in lines 1-4 of the pseudo-code, an initial population $\vec{x}_i \in \mathbf{P}$ $(i = 1, \cdots, N_P)$ is generated randomly as shown in (17). Besides, the value of $U(\vec{x}_i,\,N)$ is evaluated for each $\vec{x}_i \in \mathbf{P}$ by taking N samples of $F(\vec{x}_i)$ values.

By using the strategy shown in Algorithm 1, the trial vector $\vec{u}$ is generated in the 7th line. DE evaluates all individuals based on the upper bound of the prediction interval. Therefore, to calculate the value of $U(\vec{u},\,N)$ for $\vec{u}$ in the 8th line, the noisy function $F(\vec{u})$ has to be evaluated for N times.

The newborn trial vector $\vec{u}$ is compared to the existing target vector $\vec{x}_i \in \mathbf{P}$ in the 9th line. If $\vec{u}$ is not worse than $\vec{x}_i \in \mathbf{P}$, $\vec{x}_i \in \mathbf{P}$ is replaced instantly by $\vec{u}$ in lines 10-11. Otherwise, the trial vector $\vec{u}$ is discarded. Consequently, the excellent trial vector $\vec{u}$ can be used soon to generate offspring.

The procedure in lines 5-14 is repeated until a termination condition is satisfied. As the termination condition, the total number of noisy function's observations is usually used. Finally, the individual $\vec{x}_b \in \mathbf{P}$ with the minimum $U(\vec{x}_b,\,N)$ value is obtained as the best solution for the worst-case value minimization problem in the 15th line of Algorithm 2.

V. EXTENDED VARIANTS OF DE

Since multiple sampling of noisy function values could be very expensive in most of the applications, it is desired to reduce the number of samples. In order to obtain solutions for worst-case value minimization problems effectively, we propose an extended DE. First of all, in order to explain the principal techniques used by the extended DE, namely prediction interval and cutoff point, we present two immature versions. After that, we describe a complete version.

A. DE with Prediction (DE-P)

The first extended DE is called DE with Prediction (DE-P). DE-P uses the prediction interval in (12) to skip the multiple sampling for the trial vector $\vec{u}$. As well as DE, DE-P generates a new trial vector $\vec{u}$ by the strategy. However, DE-P evaluates $F(\vec{u},\,1)$ instead of $U(\vec{u},\,N)$ for $\vec{u}$. From (9), $F(\vec{u},\,1)$ denotes a single sample value of the noisy function $F(\vec{u})$. Then DE-P compares the trial vector $\vec{u}$ with the target vector $\vec{x}_i$. If the value of $F(\vec{u},\,1)$ doesn't fall in the prediction interval made from $\vec{x}_i$, $\vec{u}$ must be different from $\vec{x}_i$ in short odds. Besides, if $F(\vec{u},\,1) > U(\vec{x}_i,\,N)$ holds, $\vec{u}$ is inferior to $\vec{x}_i$. Therefore, DE-P can omit to evaluate the value of $U(\vec{u},\,N)$.

Algorithm 3 provides the pseudo-code of DE-P. First of all, in lines 1-4 of the pseudo-code, an initial population $\vec{x}_i \in \mathbf{P}$ is generated randomly and evaluated in the same way with DE. By using the strategy shown in Algorithm 1, the trial vector $\vec{u}$ is generated in the 7th line. The value of $F(\vec{u},\,1)$ is evaluated only by a single sample of the noisy function in the 8th line. If $F(\vec{u},\,1) \leq U(\vec{x}_i,\,N)$ holds in the 9th line, the value of $U(\vec{u},\,N)$ is evaluated for $\vec{u}$ by taking N samples of $F(\vec{u})$ in the 10th line. After that, the trial vector $\vec{u}$ is compared to the target vector $\vec{x}_i \in \mathbf{P}$ in the same way with DE.

Algorithm 3 : DE WITH PREDICTION (DE-P)

1: **for** $i := 1$ to N_P **do**
2: $\vec{x}_i :=$ RANDOMLY_GENERATE();
3: $U(\vec{x}_i,\,N) :=$ EVALUATE_N_TIME($\vec{x}_i$);
4: **end for**
5: **repeat**
6: **for** $i := 1$ to N_P **do**
7: $\vec{u} :=$ STRATEGY($\vec{x}_i,\,\vec{x}_{r1},\,\vec{x}_{r2},\,\vec{x}_{r3} \in \mathbf{P}$);
8: $F(\vec{u},\,1) :=$ EVALUATE_ONE_TIME($\vec{u}$);
9: **if** $F(\vec{u},\,1) \leq U(\vec{x}_i,\,N)$ **then**
10: $U(\vec{u},\,N) :=$ EVALUATE_N_TIME($\vec{u}$);
11: **if** $U(\vec{u},\,N) \leq U(\vec{x}_i,\,N)$ **then**
12: $\vec{x}_i := \vec{u}$;
13: $U(\vec{x}_i,\,N) := U(\vec{u},\,N)$;
14: **end if**
15: **end if**
16: **end for**
17: **until** a termination condition is satisfied;
18: Output $\vec{x}_b \in \mathbf{P}$ with the minimum $U(\vec{x}_b,\,N)$;

Algorithm 4 : DE WITH CUTOFF (DE-C)

```
1: for i := 1 to N_P do
2:    x_i := RANDOMLY_GENERATE();
3:    F(x_i, 1) := EVALUATE_ONE_TIME(x_i);
4:    if F(u, 1) ≤ γ then
5:       U(x_i, N) := EVALUATE_N_TIME(x_i);
6:    else
7:       U(x_i, N) := ∞;
8:    end if
9: end for
10: repeat
11:    for i := 1 to N_P do
12:       u := STRATEGY(x_i, x_r1, x_r2, x_r3 ∈ P);
13:       F(u, 1) := EVALUATE_ONE_TIME(u);
14:       if F(u, 1) ≤ γ then
15:          U(u, N) := EVALUATE_N_TIME(u);
16:          if U(u, N) ≤ U(x_i, N) then
17:             x_i := u;
18:             F(x_i, 1) := F(u, 1);
19:             U(x_i, N) := U(u, N);
20:          end if
21:       else
22:          if F(u, 1) ≤ F(x_i, 1) then
23:             x_i := u;
24:             F(x_i, 1) := F(u, 1);
25:             U(x_i, N) := ∞;
26:          end if
27:       end if
28:    end for
29: until a termination condition is satisfied;
30: Output x_b ∈ P with the minimum U(x_b, N);
```

Algorithm 5 : DE WITH PREDICTION & CUTOFF (DE-PC)

```
1: for i := 1 to N_P do
2:    x_i := RANDOMLY_GENERATE();
3:    F(x_i, 1) := EVALUATE_ONE_TIME(x_i);
4:    if F(u, 1) ≤ γ then
5:       U(x_i, N) := EVALUATE_N_TIME(x_i);
6:    else
7:       U(x_i, N) := ∞;
8:    end if
9: end for
10: repeat
11:    for i := 1 to N_P do
12:       u := STRATEGY(x_i, x_r1, x_r2, x_r3 ∈ P);
13:       F(u, 1) := EVALUATE_ONE_TIME(u);
14:       if F(u, 1) ≤ γ then
15:          if F(u, 1) ≤ U(x_i, N) then
16:             U(u, N) := EVALUATE_N_TIME(u);
17:             if U(u, N) ≤ U(x_i, N) then
18:                x_i := u;
19:                F(x_i, 1) := F(u, 1);
20:                U(x_i, N) := U(u, N);
21:             end if
22:          end if
23:       else
24:          if F(u, 1) ≤ F(x_i, 1) then
25:             x_i := u;
26:             F(x_i, 1) := F(u, 1);
27:             U(x_i, N) := ∞;
28:          end if
29:       end if
30:    end for
31: until a termination condition is satisfied;
32: Output x_b ∈ P with the minimum U(x_b, N);
```

B. DE with Cutoff (DE-C)

The second extended DE is called DE with Cutoff (DE-C). DE-C uses a new control parameter called cutoff point $\gamma \in \Re$ ($\gamma > 0$) to skip the multiple sampling for many individuals. Fundamentally, DE-C uses a single sample value $F(\vec{x}_i, 1)$ for handling every individual $\vec{x}_i \in \mathbf{P}$. Besides, if some excellent individuals $\vec{x}_i$ satisfy the condition $F(\vec{x}_i, 1) \leq \gamma$, DE-C evaluates the values of $U(\vec{x}_i, N)$ for them.

Algorithm 4 provides the pseudo-code of DE-C. In lines 1-9 of the pseudo-code, an initial population $\vec{x}_i \in \mathbf{P}$ is generated randomly. Then the values of $U(\vec{x}_i, N)$ are evaluated only for the superior ones $\vec{x}_i \in \mathbf{P}$ that satisfy the condition $F(\vec{x}_i, 1) \leq \gamma$. For the other ones $\vec{x}_i \in \mathbf{P}$, $U(\vec{x}_i, N)$ are initialized to infinity (∞). By using the strategy, the trial vector $\vec{u}$ is generated in the 12th line. The value of $F(\vec{u}, 1)$ is obtained only by a single sample of the noisy function $F(\vec{u})$ in the 13th line. If $F(\vec{u}, 1) \leq \gamma$ holds in the 14th line, $U(\vec{u}, N)$ is evaluated for $\vec{u}$ by taking N samples of $F(\vec{u})$ in the 15th line. After that, the trial vector $\vec{u}$ is compared to the target vector $\vec{x}_i \in \mathbf{P}$ in the same way with DE. On the other hand, if $F(\vec{u}, 1) > \gamma$ holds in the 14th line, the trial vector $\vec{u}$ is compared to the target vector $\vec{x}_i \in \mathbf{P}$ in lines 22-26. The inferior trial vector $\vec{u}$ has a chance to survive if it can beat the target vector $\vec{x}_i \in \mathbf{P}$ such as $F(\vec{u}, 1) \leq F(\vec{x}_i, 1)$.

C. DE with Prediction & Cutoff (DE-PC)

The third extended DE is called DE with Prediction & Cutoff (DE-PC). DE-PC is the complete version of the extended DE proposed in this paper. DE-PC is a good combination between DE-P and DE-C. Therefore, as well as DE-C, the proposed DE-PC also uses the cutoff point $\gamma \in \Re$ ($\gamma > 0$) to skip the multiple sampling for many individuals. DE-PC evaluates the values of $U(\vec{x}_i, N)$ only for the excellent ones $\vec{x}_i \in \mathbf{P}$ that satisfy $F(\vec{x}_i, 1) \leq \gamma$. DE-PC is almost the same as DE-C. However, DE-PC differs from DE-C in the way to evaluate the trial vector $\vec{u}$. For evaluating $U(\vec{u}, N)$, DE-PC demands the trial vector $\vec{u}$ to satisfy both of the two conditions, namely $F(\vec{u}, 1) \leq \gamma$ and $F(\vec{u}, 1) \leq U(\vec{x}_i, N)$.

Algorithm 5 gives the pseudo-code of DE-PC. First of all, in lines 1-9 of the pseudo-code, an initial population $\vec{x}_i \in \mathbf{P}$ is generated randomly and evaluated in the same way with DE-C. By using the strategy shown in Algorithm 1, the trial vector $\vec{u}$ is generated in the 12th line. The value of $F(\vec{u}, 1)$ is obtained only by a single sample of the noisy function $F(\vec{u})$ in the 13th line. If $F(\vec{u}, 1) \leq \gamma$ holds in the 14th line, the trial vector $\vec{u}$ is compared to the target vector $\vec{x}_i \in \mathbf{P}$ in the same way with DE-P. Therefore, as stated above, the trial

vector $\vec{u}$ has to satisfy two conditions, namely $F(\vec{u},\, 1) \leq \gamma$ and $F(\vec{u},\, 1) \leq U(\vec{x}_i,\, N)$, before $U(\vec{u},\, N)$ is evaluated for $\vec{u}$. On the other hand, if $F(\vec{u},\, 1) > \gamma$ holds in the 14th line, the trial vector $\vec{u}$ is compared to the target vector $\vec{x}_i \in \mathbf{P}$ in lines 24-28. As well as PD-C, the inferior trial vector $\vec{u}$ still has the chance to survive if $F(\vec{u},\, 1) \leq F(\vec{x}_i,\, 1)$ holds.

The procedure in lines 10-31 of the pseudo-code of DE-PC is repeated until a termination condition is satisfied. Finally, the individual $\vec{x}_b \in \mathbf{P}$ with the minimum $U(\vec{x}_b,\, N)$ value is obtained as the best solution for the worst-case value minimization problem in the bottom line of Algorithm 5.

VI. Numerical Experiments

A. Benchmark Problems

By using deterministic functions $f_p(\vec{x})$ and $g(\vec{x})$, an instance of noisy objective function $F(\vec{x})$ in (6) is defined as

$$F_p(\vec{x}) = f_p(\vec{x}) + \kappa\, g(\vec{x})\, \varepsilon, \tag{19}$$

where κ ($\kappa > 0$) is a gain factor and additive noise ε is normaly distributed such as $\varepsilon \sim \mathcal{N}(0,\, 1)$.

For the first deterministic function $f_p(\vec{x})$ in (19), we employ the well-known test functions shown in Appendixes A and B [9]. Appendix A provides four uni-modal test functions: f_p ($p = 1,\, \cdots,\, 4$). On the other hand, Appendix B provides four multi-modal test functions: f_p ($p = 5,\, \cdots,\, 8$).

The second deterministic function $g(\vec{x})$ in (19) is given as

$$g(\vec{x}) = \frac{1}{D} \sum_{j=1}^{D} \sin\left(\pi \left(\frac{x_j - \underline{x}_j}{\overline{x}_j - \underline{x}_j}\right)\right). \tag{20}$$

From (20), $g(\vec{x})$ is an uni-modal function such as

$$g(\vec{x}) = \begin{cases} 0, & \text{if } \forall\, j:\ x_j = \underline{x}_j, \\ 1, & \text{if } \forall\, j:\ x_j = (\overline{x}_j + \underline{x}_j)/2, \\ 0, & \text{if } \forall\, j:\ x_j = \overline{x}_j. \end{cases} \tag{21}$$

Consequently, the noisy objective function $F_p(\vec{x})$ defined by (19) is also distributed normally as follows:

$$F_p(\vec{x}) \sim \mathcal{N}(f_p(\vec{x}),\ \kappa^2\, g(\vec{x})^2), \tag{22}$$

where both the mean $f_p(\vec{x})$ and the variance $\kappa^2\, g(\vec{x})^2$ of the normal distribution depend on the solution $\vec{x} \in \Re^D$.

For calculating the objective function $U_p(\vec{x},\, N)$ of the worst-case value minimization problem defined in (15) from the observed values F_n ($n = 1,\, \cdots,\, N$) of $F_p(\vec{x})$, the significant level α and the number of samples N in (14) are chosen, respectively, as $\alpha = 0.05$ and $N = 30$. Besides, the dimension of the solution $\vec{x} \in \Re^D$ is chosen as $D = 20$.

B. Experimental Setup

The control parameters of all algorithms, namely DE, DE-P, DE-C, and DE-PC, are chosen as $S_F = 0.5$, $C_R = 0.9$, and $N_P = 100$, 200. A cutoff point $\gamma = 50$ is chosen for DE-C and DE-PC. As the termination condition, the total number of noisy function's observations is limited to 3×10^5. Thereby, all the algorithms are applied, respectively, to each of benchmark problems 30 times. Incidentally, all algorithms are coded by Java and executed on a personal computer (CPU: Intel Core$^{\mathrm{TM}}$i7@3.33[GHz]; OS: Microsoft Windows 7).

TABLE I
Upper and Lower Bounds by Best Solution ($N_P = 100$)

(a) Sphere function: f_1

κ		DE	DE-P	DE-C	DE-PC
1.0	U	371.718	3.709	3.198	**2.780**
	L	367.590	−0.575	−1.096	**−1.212**
2.0	U	338.214	7.043	5.651	**5.659**
	L	329.955	−1.644	−2.245	**−2.779**
4.0	U	393.355	13.595	11.504	**11.219**
	L	376.844	−3.474	−5.368	**−5.965**

(b) Hyper-Ellipsoid function: f_2

κ		DE	DE-P	DE-C	DE-PC
1.0	U	54.291	3.281	2.994	**2.661**
	L	50.293	−0.882	−1.137	**−1.257**
2.0	U	55.895	6.183	5.687	**5.259**
	L	47.894	−2.125	−2.691	**−2.795**
4.0	U	58.564	12.058	10.755	**10.617**
	L	42.614	−4.265	−4.994	**−5.887**

(c) Rosenbrock function: f_3

κ		DE	DE-P	DE-C	DE-PC
1.0	U	75.996	19.289	18.971	**18.273**
	L	71.920	15.061	14.942	**14.156**
2.0	U	77.009	22.573	22.311	**21.363**
	L	65.077	14.357	13.987	**13.259**
4.0	U	81.377	28.925	28.683	**27.608**
	L	65.077	12.913	11.990	**10.961**

(d) Schwefel's Ridge function: f_4

κ		DE	DE-P	DE-C	DE-PC
1.0	U	3500.078	4.203	6.556	**3.221**
	L	3496.388	0.183	2.414	**−0.904**
2.0	U	3472.947	8.183	8.907	**6.128**
	L	3465.581	0.014	0.554	**−2.170**
4.0	U	3340.477	15.056	14.404	**11.723**
	L	3325.674	−1.696	−2.157	**−4.280**

(e) Ackley function: f_5

κ		DE	DE-P	DE-C	DE-PC
1.0	U	19.811	**17.787**	19.811	**17.787**
	L	16.147	**14.099**	16.147	**14.099**
2.0	U	23.068	**22.844**	23.068	**22.844**
	L	19.481	19.729	**19.481**	19.729
4.0	U	23.427	**23.104**	23.427	**23.104**
	L	19.937	20.077	**19.937**	20.077

(f) Griewank function: f_6

κ		DE	DE-P	DE-C	DE-PC
1.0	U	7.404	3.847	3.986	**3.815**
	L	3.279	−0.146	−0.131	**−0.414**
2.0	U	9.740	6.669	6.858	**6.559**
	L	1.490	−1.236	−1.618	**−1.884**
4.0	U	15.757	12.447	12.819	**11.630**
	L	−0.722	−4.071	−4.267	**−4.580**

(g) Rastrigin function: f_7

κ		DE	DE-P	DE-C	DE-PC
1.0	U	126.694	14.225	16.402	**5.213**
	L	122.942	9.982	12.294	**1.059**
2.0	U	129.076	22.145	20.333	**11.499**
	L	121.597	14.005	12.150	**3.451**
4.0	U	131.566	35.800	29.739	**22.589**
	L	116.572	19.999	13.030	**6.033**

(h) Salomon function: f_8

κ		DE	DE-P	DE-C	DE-PC
1.0	U	7.261	**5.555**	7.261	**5.555**
	L	3.187	**1.401**	3.187	**1.401**
2.0	U	10.503	**9.409**	10.503	**9.409**
	L	2.419	**0.973**	2.419	**0.973**
4.0	U	17.316	**15.791**	17.316	**15.791**
	L	1.520	**0.544**	1.520	**0.544**

C. Results

Table I shows the results of experiments conducted on the benchmark problems from test functions f_p ($p = 1, \cdots, 8$). The population size is chosen as $N_P = 100$ for all algorithms, namely DE, DE-P, DE-C, and DE-PC. Table I describes the upper (U) and the lower (L) bounds in (12) calculated from the best solution $\vec{x}_b \in \mathbf{P}$ obtained by respective algorithms. The results in Table I are averaged over 30 independent runs per the function f_p and per the gain factor κ used in (19). Besides, the best result in each case is described in bold.

From Table I, the smallest upper bound is obtained by the proposed DE-PC in every case. On the other hand, the largest upper bound is provided by DE in every case. The effect of prediction interval can be confirmed by comparison of DE and DE-P. The effect of cutoff point can be also confirmed by comparison of DE and DE-C. Furthermore, the great effect of the combination between the prediction interval and the cutoff point can be confirmed by the excellent performance of DE-PC. From Table I, the cutoff point seems to be more effective than the prediction interval for uni-modal test functions except one test function: f_4. On the other hand, the prediction interval seems to be more effective than the cutoff point for multi-modal test functions except one test function: f_7.

Table II shows the results of experiments conducted on the benchmark problems from test functions f_p ($p = 1, \cdots, 8$) too. The population size is chosen as $N_P = 200$ for all algorithms. Table II describes the upper (U) and the lower (L) bounds calculated from the best solution obtained by respective algorithms in the same way with Table I. The results shown in Table I and Table II resemble each other. The smallest upper bound is obtained by the proposed DE-PC in every case. The largest upper bound is also provided by DE in every case. The effect of prediction interval can be confirmed by comparison of DE and DE-P. The effect of cutoff point can be also confirmed by comparison of DE and DE-C. From Table II, the cutoff point is more effective than the prediction interval for all uni-modal test functions: f_p ($p = 1, \cdots, 4$). However, the prediction interval is also effective for many multi-modal test functions except one test function: f_7.

Fig. 2 shows some examples of the convergence of the upper bound $U(\vec{x}_b, N)$ achieved by the best solution $\vec{x}_b \in \mathbf{P}$ in the population for different algorithms and for benchmark problems in which the gain factor $\kappa = 1.0$ is used. The population size is chosen as $N_P = 100$ for all algorithms. The vertical axis in Fig. 2 denotes the value of $U(\vec{x}_b, N)$, while the horizontal axis denotes the number of noisy function's observations. The values plotted in Fig. 2 are averaged over 30 independent runs. Fig. 3 also shows some examples of the upper bound $U(\vec{x}_b, N)$ achieved by the best solution $\vec{x}_b \in \mathbf{P}$ in the population in the same way with Fig. 2, where the population size is chosen as $N_P = 200$ for all algorithms.

From Fig. 2 and Fig. 3, the effectiveness of the proposed two techniques can be confirmed in uni-modal functions: f_1, f_2, f_3, and f_4. The best solution obtained by DE-C converges quickly in the early stage of the search. Therefore, the cut-off point is very effective to accelerate the extended DE. However, the cut-off point becomes ineffective in the middle of the

TABLE II
UPPER AND LOWER BOUNDS BY BEST SOLUTION ($N_P = 200$)

(a) Sphere function: f_1

κ		DE	DE-P	DE-C	DE-PC
1.0	U	4877.076	53.183	6.301	**3.903**
	L	4873.060	48.888	2.140	**−0.183**
2.0	U	4790.676	67.114	9.222	**7.146**
	L	4782.641	58.837	0.955	**−1.233**
4.0	U	4789.448	81.077	14.530	**12.813**
	L	4773.375	63.964	−2.624	**−3.290**

(b) Hyper-Ellipsoid function: f_2

κ		DE	DE-P	DE-C	DE-PC
1.0	U	839.263	17.795	5.468	**3.567**
	L	835.506	13.807	1.325	**−0.305**
2.0	U	746.969	22.527	8.322	**6.814**
	L	739.420	14.369	0.058	**−1.498**
4.0	U	914.192	32.631	14.052	**12.235**
	L	899.032	16.829	−2.337	**−3.495**

(c) Rosenbrock function: f_3

κ		DE	DE-P	DE-C	DE-PC
1.0	U	407.159	27.696	22.637	**19.970**
	L	403.338	23.455	18.591	**15.845**
2.0	U	414.639	33.189	25.863	**23.552**
	L	407.030	24.830	17.159	**15.226**
4.0	U	391.586	43.293	31.184	**29.896**
	L	376.231	26.244	14.815	**13.293**

(d) Schwefel's Ridge function: f_4

κ		DE	DE-P	DE-C	DE-PC
1.0	U	8176.660	88.494	15.258	**5.186**
	L	8173.404	84.423	11.269	**1.137**
2.0	U	8164.842	93.784	16.932	**9.430**
	L	8158.319	85.827	9.075	**1.396**
4.0	U	8067.288	105.167	21.421	**16.172**
	L	8054.371	88.344	5.964	**−0.237**

(e) Ackley function: f_5

κ		DE	DE-P	DE-C	DE-PC
1.0	U	21.814	**21.349**	21.814	**21.349**
	L	18.740	**17.966**	18.740	**17.966**
2.0	U	23.285	**23.076**	23.285	**23.076**
	L	**19.363**	19.371	**19.363**	19.371
4.0	U	24.720	**24.182**	24.720	**24.182**
	L	**18.557**	19.033	**18.557**	19.033

(f) Griewank function: f_6

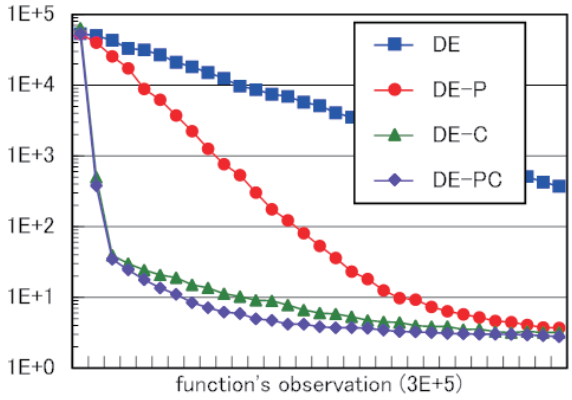
κ		DE	DE-P	DE-C	DE-PC
1.0	U	46.016	6.404	6.781	**4.688**
	L	41.999	2.177	2.592	**0.648**
2.0	U	49.000	10.744	9.166	**8.196**
	L	40.973	1.963	0.958	**−0.331**
4.0	U	49.927	17.582	15.299	**14.638**
	L	33.833	1.181	−1.158	**−2.992**

(g) Rastrigin function: f_7

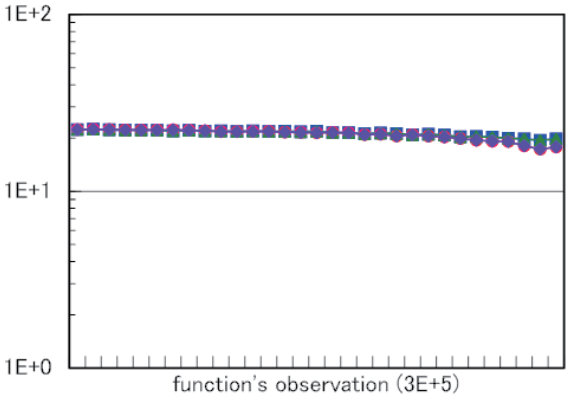
κ		DE	DE-P	DE-C	DE-PC
1.0	U	168.667	49.838	26.991	**18.938**
	L	165.124	45.693	22.809	**14.803**
2.0	U	172.310	60.147	28.768	**23.884**
	L	165.169	51.935	20.674	**15.498**
4.0	U	181.734	76.588	35.459	**33.560**
	L	167.701	61.221	19.278	**16.799**

(h) Salomon function: f_8

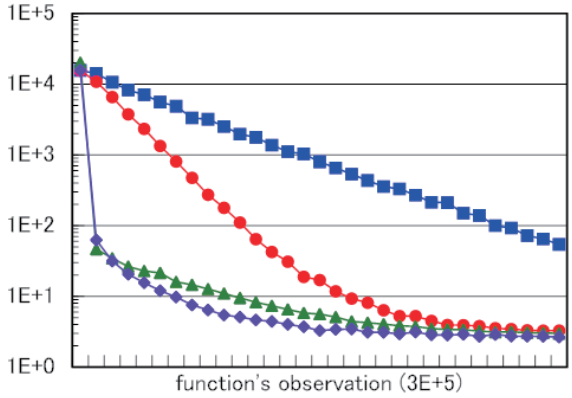
κ		DE	DE-P	DE-C	DE-PC
1.0	U	11.239	**8.275**	11.239	**8.275**
	L	7.302	**4.241**	7.302	**4.241**
2.0	U	14.839	**12.622**	14.839	**12.622**
	L	7.107	**4.646**	7.107	**4.646**
4.0	U	21.131	**19.660**	21.131	**19.660**
	L	6.196	**4.576**	6.196	**4.576**



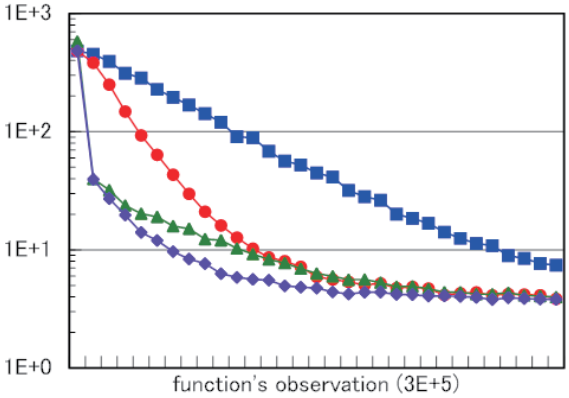
(a) Sphere function: f_1



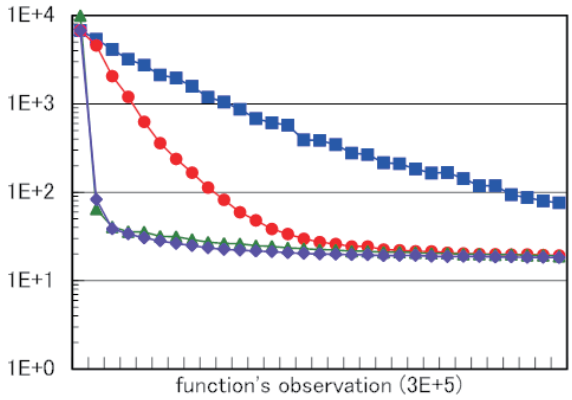
(e) Ackley function: f_5



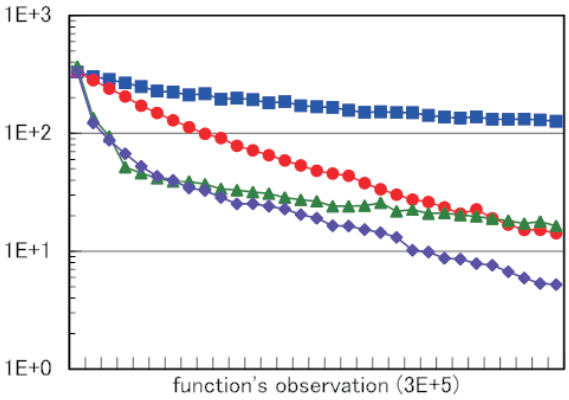
(b) Hyper-Ellipsoid function: f_2



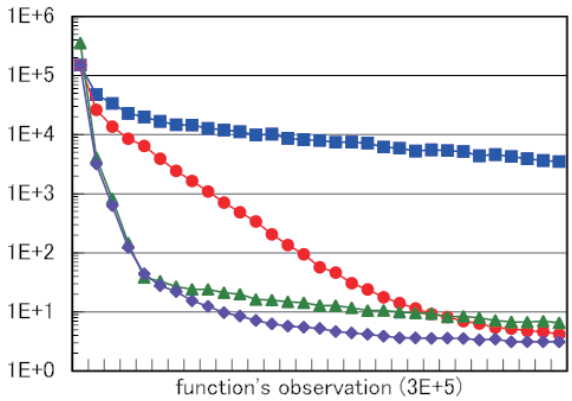
(f) Griewank function: f_6



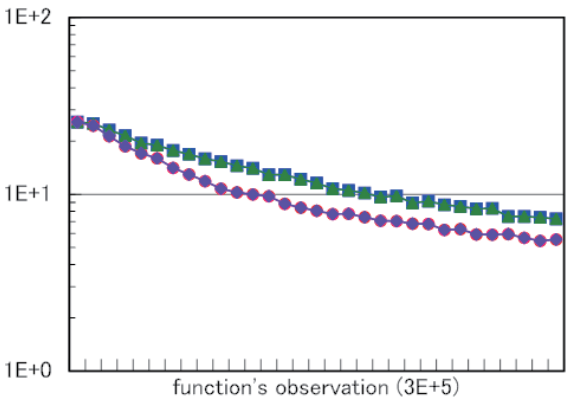
(c) Rosenbrock function: f_3



(g) Rastrigin function: f_7

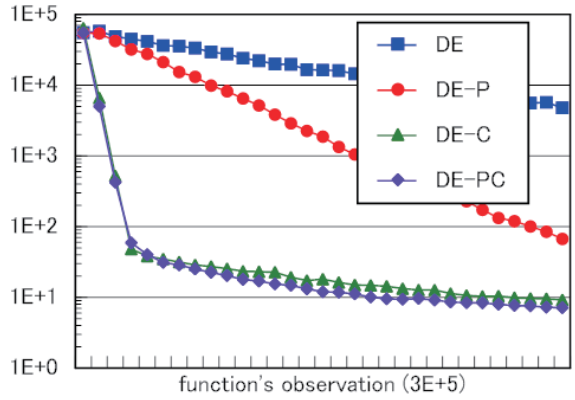


(d) Schwefel's Ridge function: f_4

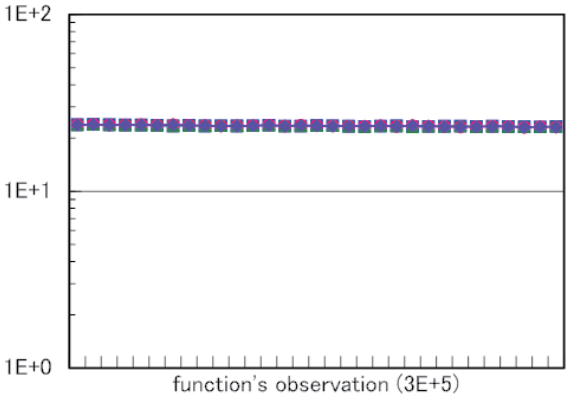


(h) Salomon function: f_8

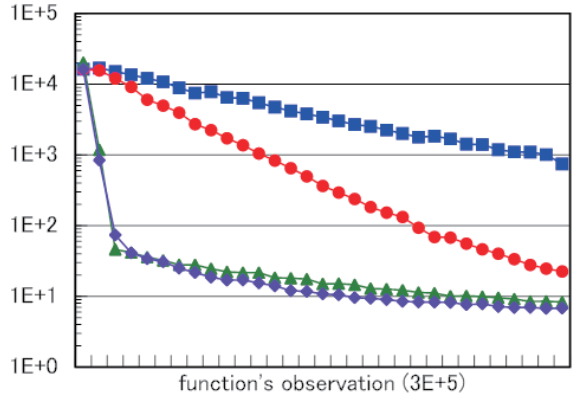
Fig. 2. Convergence of the upper bound achieved by the best solution in the population ($N_P = 100$)



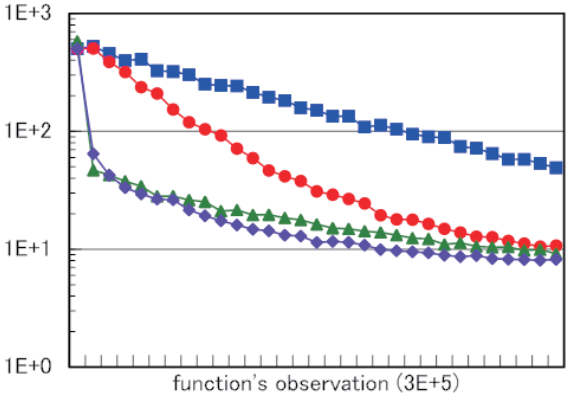
(a) Sphere function: f_1



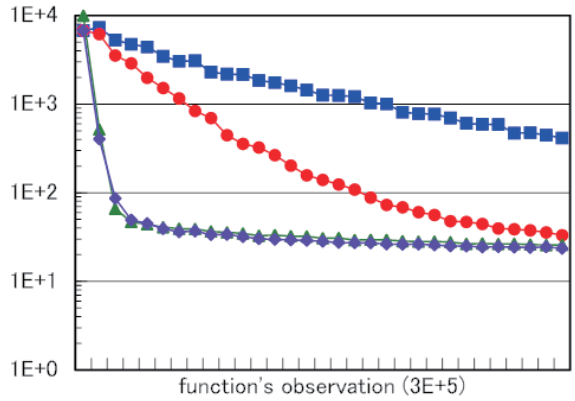
(e) Ackley function: f_5



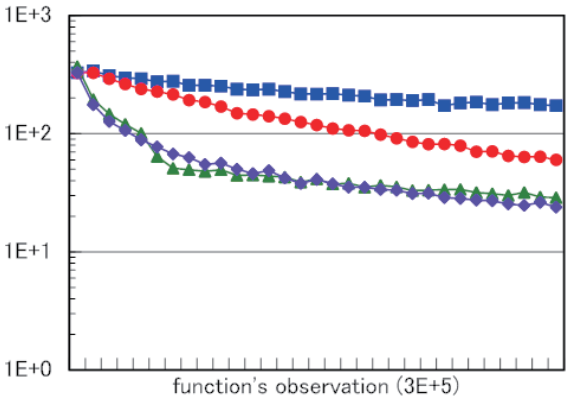
(b) Hyper-Ellipsoid function: f_2



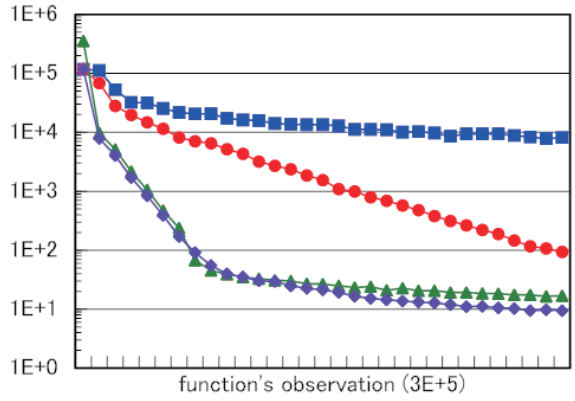
(f) Griewank function: f_6



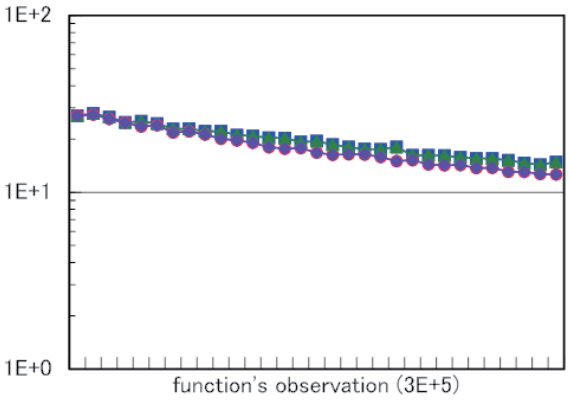
(c) Rosenbrock function: f_3



(g) Rastrigin function: f_7



(d) Schwefel's Ridge function: f_4



(h) Salomon function: f_8

Fig. 3. Convergence of the upper bound achieved by the best solution in the population ($N_P = 200$)

search. Contrary to DE-C, the best solution obtained by DE-P converges slowly but steadily until the end of the search. Therefore, the prediction interval is effective in the whole of the search. Consequently, the proposed DE-PC using both of the two techniques outperforms DE-P and DE-C in all uni-modal functions. On the other hand, the best solution obtained by DE is hard to converge in all uni-modal functions.

As well as uni-modal functions, the effectiveness of the proposed two techniques can be confirmed in some multi-modal functions: f_6 and f_7. However, for the other multi-modal functions: f_5 and f_8, the effectiveness of the proposed techniques is not able to be confirmed enough. Since both f_5 and f_8 have a huge number of local minima, the proposed techniques might fail to prune away those local minima.

VII. Conclusion

Since noise is inevitable in real-world applications, they can be regarded as uncertain optimization problems. In order to prepare for the worst possible pass in real-world optimization problems, a new class of uncertain optimization problems, i.e., the worst-case value minimization problem, was formulated in this paper. In the worst-case value minimization problem, a predicted upper bound of the noisy objective function's values was minimized by tuning decision variables.

For solving the worst-case value minimization problem effectively, an extended DE named DE-PC was proposed. The cardinal DE could be applied directly to the worst-case value minimization problem. However, in order to evaluate each solution based on the upper bound of noisy objective function's values, multiple sampling was required. Therefore, DE-PC adopted two pruning techniques, namely prediction interval and cutoff point, both of which could judge and discard hopeless solutions only by a single sampling.

In order to verify the effects of the pruning techniques, two immature versions of DE-PC, namely DE-P and DE-C, were also presented. DE-P adopted only prediction interval, while DE-C adopted only cutoff point. From the results of numerical experiments conducted on eight benchmark problems derived, respectively, from four uni-modal and four multi-modal test functions, it could be confirmed that both DE-P and DE-C were more efficient than the cardinal DE. Furthermore, it was shown that the proposed DE-PC always outperformed the cardinal DE, DE-P, and DE-C in all benchmark problems.

The main drawback of the proposed DE-PC is that it needs a new control parameter, namely the cutoff point γ. Therefore, further research is required to suggest an appropriate γ value for a wide range of worst-case value minimization problems. Another interesting aspect which will be considered is to create an adaptive control technique of the cutoff point γ.

Appendix A
Uni-Modal Test Functions

Uni-modal test functions have a single minimum.

- Sphere function

$$f_1(\vec{x}) = \sum_{j=1}^{D} x_j^2,$$
$$-100 \le x_j \le 100,\ j = 1, \cdots, D.$$

- Hyper-Ellipsoid function

$$f_2(\vec{x}) = \sum_{j=1}^{D} 2^j\, x_j^2,$$
$$-5.12 \le x_j \le 5.12,\ j = 1, \cdots, D.$$

- Rosenbrock function

$$f_3(\vec{x}) = \sum_{j=1}^{D-1} (100\,(x_{j+1} - x_j^2)^2 + (x_j - 1)^2),$$
$$-2.048 \le x_j \le 2.048,\ j = 1, \cdots, D.$$

- Schwefel's Ridge function

$$f_4(\vec{x}) = \sum_{k=1}^{D} \left(\sum_{j=1}^{k} x_j \right)^2,$$
$$-65.536 \le x_j \le 65.536,\ j = 1, \cdots, D.$$

Appendix B
Multi-Modal Test Functions

Multi-modal test functions have more than one local minimum, where one of them is a global minimum.

- Ackley function

$$f_5(\vec{x}) = -20 \exp\left(-0.2 \sqrt{\frac{1}{D} \sum_{j=1}^{D} x_j^2}\right) - \exp\left(\frac{1}{D} \sum_{j=1}^{D} \cos(2\,\pi\, x_j)\right) + 20 + e,$$
$$-32.768 \le x_j \le 32.768,\ j = 1, \cdots, D.$$

- Griewank function

$$f_6(\vec{x}) = \frac{1}{4000} \sum_{j=1}^{D} x_j^2 - \prod_{j=1}^{D} \cos\left(\frac{x_j}{\sqrt{j}}\right) + 1,$$
$$-600 \le x_j \le 600,\ j = 1, \cdots, D.$$

- Rastrigin function

$$f_7(\vec{x}) = \sum_{j=1}^{D} (x_j^2 - 10\,\cos(2\,\pi\, x_j) + 10),$$
$$-5.12 \le x_j \le 5.12,\ j = 1, \cdots, D.$$

- Salomon function

$$f_8(\vec{x}) = -\cos(2\,\pi\, y) + 0.1\, y + 1,$$
$$y = \sqrt{\sum_{j=1}^{D} x_j^2},$$
$$-100 \le x_j \le 100,\ j = 1, \cdots, D.$$

Acknowledgment

This work was supported in part by the Grant-in-Aid for Scientific Research (C) (Project No. 24560503) from Japan Society for the Promotion of Science (JSPS).

REFERENCES

[1] J. Branke and C. Schmidt, Sequential sampling in noisy environments, *Proc. of PPSN VIII*, LNCS 3242, Springer, 2004, pp. 202–211.

[2] Y. Jin and J. Branke, Evolutionary optimization in uncertain environments – a survey, *IEEE Trans. on Evolutionary Computation*, Vol. 9, No. 3, 2005, pp. 303–317.

[3] S. Das, A. Konar, and U. K. Chakraborty, Improved differential evolution algorithms for handling noisy optimization problems, *Proc. of IEEE Congress of Evolutionary Computation*, 2005, pp. 1691–1698.

[4] S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama, Opposition-based differential evolution for optimization of noisy problems, *Proc. of IEEE Congress of Evolutionary Computation*, 2006, pp. 6756–6763.

[5] B. Liu, X. Zhang, and H. Ma, Hybrid differential evolution for noisy optimization, *Proc. of IEEE Congress of Evolutionary Computation*, 2008, pp. 587–592.

[6] A. Dharchoudhury and S. M. Kang, Performance-constrained worst-case variability minimization of VLSI circuits, *Proc. of IEEE 30th Conference on Design Automation*, 1993, pp. 154–158.

[7] K. J. Antreich, H. E. Graeb, and C. U. Wieser, Circuit analysis and optimization driven by worst-case distance, *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 13, No. 1, 1994, pp. 57–71.

[8] R. Storn and K. Price, Differential evolution - a simple and efficient heuristic for global optimization over continuous space, *Journal of Global Optimization*, Vol. 4, No. 11, 1997, pp. 341–359.

[9] K. V. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution - A Practical Approach to Global Optimization*, Springer, 2005.

[10] S. Das and P. N. Suganthan, Differential evolution: a survey of the state-of-the art, *IEEE Trans. on Evolutionary Computation*, Vol. 15, No. 1, 2011, pp. 4–31.

[11] R. Storn, System design by constraint adaptation and differential evolution, *IEEE Trans. on Evolutionary Computation*, Vol. 3, No. 1, 1999, pp. 22–34.

[12] U. K. Chakraborty (Ed.), *Advances in Differential Evolution*, Springer, 2008.

[13] L. Kouril, M. Pospisilik, M. Adamek, and R. Jasek, Application of differential evolution for audio transformers optimization, *International Journal of Circuits, Systems and Signal Processing*, Issue 3, Vol. 6, 2012, pp. 231–240.

[14] Y. Tang, Parameter estimation of wiener model using differential evolution algorithm, *International Journal of Circuits, Systems and Signal Processing*, Issue 5, Vol. 6, 2012, pp. 315–323.

[15] K. Tagawa, Two-stage optimum design method for surface acoustic wave duplexers using differential evolution algorithms, *International Journal of Systems Applications, Engineering & Development*, Issue 2, Vol. 7, 2013, pp. 103–111.

[16] R. Mallipeddi and M. Lee, Surrogate model assisted ensemble differential evolution algorithm, *Proc. of IEEE Congress of Evolutionary Computation*, 2012, pp. 1–8.

[17] M. Suciu, R. Lung, N. Gasko, and D. Dumitrescu, Differential evolution for discrete-time large dynamic games, *Proc. of IEEE Congress of Evolutionary Computation*, 2013, pp. 2108–2113.

[18] A. Alwadie, Efficient algorithms for noise estimation in electrical power line communications, *WSEAS Trans. on Communications*, Issue 11, Vol. 11, 2012, pp. 405–414.

[19] H. B. Beyer and B. Sendhoff, Evolution strategies for robust optimization, *Proc. of IEEE Congress of Evolutionary Computation*, 2006, pp. 1346–1353.

[20] R. Matu, R. Prokop, and L. Peka, Parametric and unstructured approach to uncertainty modelling and robust stability, *International Journal of Mathematical Models and Methods in Applied Sciences*, Issue 6, Vol. 5, 2011, pp. 1011-1018.

[21] T. Riouch and R. E. Bachtiri, Robust control of the active and reactive power exchanged with the rotor of the DFIG and the grid, *WSEAS Trans. on Environment and Development*, Issue 1, Vol. 9, 2013, pp. 35–45.

[22] J. Peter and M. Marcelet, Comparison of surrogate models for turbo machinery design, *WSEAS Trans. on Fluid Mechanics*, Issue 1, Vol. 3, 2008, pp.10-17.

[23] Y. Jin and B. Sendhoff, Fitness approximation in evolutionary computation: a survey, *Proc. of Genetic and Evolutionary Computation Conference*, 2002, pp. 1105–1112.

[24] H. Ishibuchi, K. Hoshino, and Y. Nojima, Strategy evolution in a spatial IPD game where each agent is not allowed to play against itself, *Proc. of IEEE Congress of Evolutionary Computation*, 2012, pp. 688–695.

[25] W. Xu and J. Ma, Study on the dynamic model of a duopoly game with delay in insurance market, *WSEAS Trans. on Mathematics*, Issue 7, Vol. 11, 2012, pp. 599–608.

[26] D. J. Sheskin, *Handbook of Parametric and Nonparametric Statistical Procedures*, Fifth Edition, CRC Press, 2011.

[27] D. D. Wackerly, W. Mendenhall, and R. L Scheaffer, *Mathematical Statistics with Applications*, 7th Edition, Thomson Learning, Inc. 2008.

Kiyoharu Tagawa received the Master's degree and Ph.D. degree from Kobe University, Kobe, Japan, in 1993 and 1997 respectively. From 2005 to 2007, he served as an Associate Professor of the Faculty of Engineering, Kobe University.

He is currently a Professor of the School of Science and Engineering, Kinki University, Higashi-Osaka, Japan. His research interests include concurrent computing, evolutionary algorithms, the design of efficient meta-heuristics, and their application for solving complex real-world optimization problems.

Taiki Suenaga received the Bachelor's degree of Engineering from the Faculty of Engineering, Kinki University, Hiroshima, Japan, in 2013.

He is currently a graduate student of the Graduate School of Science and Engineering Research, Kinki University, Higashi-Osaka, Japan. His research interests include optimization algorithms, probabilistic method, and differential evolution.