applied sciences

MDPI

Article

Parallel Compact Differential Evolution for Optimization Applied to Image Segmentation

Xiao Sui [1], Shu-Chuan Chu [1,2], Jeng-Shyang Pan [1,*] and Hao Luo [3]

[1] College of Computer Science and Engineering, Shandong University of Science and Technology, No. 579, QianWanGang Road, Qingdao 266590, China; sdustxiaosui@gmail.com (X.S.); scchu0803@gmail.com (S.-C.C.)
[2] College of Science and Engineering, Flinders University, 1284 South Road, Clovelly Park, SA 5042, Australia
[3] School of Aeronautics and Astronautics, Yuquan Campus, Zhejiang University, No. 38, ZheDa Road, Hangzhou 310027, China; luohao@zju.edu.cn
* Correspondence: jspan@cc.kuas.edu.tw

Received: 17 February 2020; Accepted: 18 March 2020; Published: 24 March 2020

Abstract: A parallel compact Differential Evolution (pcDE) algorithm is proposed in this paper. The population is separated into multiple groups and the individual is run by using the method of compact Differential Evolution. The communication is implemented after predefined iterations. Two communication strategies are proposed in this paper. The first one is to replace the local optimal solution by global optimal solution in all groups, which is called optimal elite strategy (oe); the second one is to replace the local optimal solution by mean value of the local optimal solution in all groups, which is called mean elite strategy (me). Considering that the pcDE algorithm does not need to store a large number of solutions, the algorithm can adapt to the environment with weak computing power. In order to prove the feasibility of pcDE, several groups of comparative experiments are carried out. Simulation results based on the 25 test functions demonstrate the efficacy of the proposed two communication strategies for the pcDE. Finally, the proposed pcDE is applied to image segmentation and experimental results also demonstrate the superior quality of the pcDE compared with some existing methods.

Keywords: parallel compact differential evolution (pcDE); compact differential evolution (cDE); image segmentation; communication strategies

1. Introduction

Differential Evolution (DE) has prove to be an useful optimization algorithm applied in many areas [1–3]. In many practical application scenarios, it may be less likely to use high-performance computing equipment due to various reasons, but it is necessary to solve varieties of optimization problems. The traditional optimization algorithm often needs to store a lot of solutions, and to calculate each solution's corresponding value. The huge amount of calculation conflicts with the performance of the computing equipment. Whereas, the compact evolutionary algorithm (cEA) which belongs to the estimation of distribution algorithm is proposed to overcome this dilemma. The compact evolutionary algorithm uses only one individual solution with the statistical characteristics of the population to optimize the related problem. Therefore, the method demands less computing power and memory compared with traditional methods.

Several versions of compact evolutionary algorithm have been proposed. The compact genetic algorithm (cGA) [4] is the first compact concept of the genetic algorithm (GA), which only processes one solution, but it gets the similar result when compared with the traditional genetic algorithm. The convergence of compact genetic algorithm is proved in [5] and an extended compact genetic

algorithm (ecGA) can be found in [6]. The study of the scalability for extended compact genetic algorithm can be found in [7] and the compact genetic algorithm is also applied to the training of natural network [8].

A compact differential evolution (cDE) algorithm is proposed in [9]. The principle of compact differential evolution algorithm is similar to that of compact genetic algorithm, but there are still two obvious differences. The first is survivor selection scheme. cDE adopts one-to-one generation logic and selects survivors by comparing the fitness of parents and offspring, which is different from the typical selection mechanism of the genetic algorithm. The second difference lies in that the number of DE search operations is limited, which will most likely result in failure when efforts are made to find the highest quality solution (see [10–12]). To solve this problem, randomness is added to the search logic of the basic DE algorithm, and cDE introduces certain randomness because it uses the statistical characteristics of the population. This feature not only makes it easier for the algorithm to find better solutions, but also reduces stability.

In this paper, we propose an improved version of cDE, which is called parallel compact differential evolution algorithm (pcDE). In cDE, only one individual is generated in per iteration, so it is easy to cause a large experimental error. The stability of the cDE algorithm is not strong, and the experimental results fluctuate widely. In the pcDE algorithm, multiple groups are parallelized, and the results of different groups are fused through the communication strategies after fixed number of iterations so as to obtain a reliable solution. Hence, the stability and efficiency of the algorithm are enhanced. It is found that the performance of pcDE is better than that of cDE.

Image segmentation is still a hot topic in image processing and computer vision research. The goal of image segmentation is to separate the target from its background, which is the basis of image classification and recognition. Frequently-used image segmentation methods include threshold segmentation, region tracking, and edge detection. The method of threshold segmentation is the most frequently-used image segmentation method. At present, there are many threshold selection methods, like maximum inter-class variance method (Otsu method) [13], best histogram entropy method (KSW entropy method) [14], and minimum error thresholding method [15] etc. Most existing methods use the one-dimensional gray histogram to select segmentation threshold. However, these segmentation methods will cause a lot of errors when the signal-to-noise ratio (SNR) of image decreases. Therefore, the image segmentation method based on a two-dimensional gray histogram appeared. This method is called two-dimensional maximum entropy threshold segmentation [16,17]. It uses gray information and the related information of the neighborhood space, so its effect is significantly improved by comparing with the traditional method. Different algorithms are used to select the segmentation threshold, and the results are analyzed and compared. The analysis of the experimental results shows that the threshold found by the pcDE algorithm is better for image segmentation comparing with some existing methods.

2. Background

For the optimization problem, we employ some kind of optimization algorithm to get the optimal approximate solution, which is usually the minimum value of the objective function $f(X)$, that is, min $f(X)$, where $X = (x_1, x_2, x_3, ...x_D)$ is the vector we want to solve in the solution space D ($x_i \in [x_i^{min}, x_i^{max}]$). Optimization algorithms could be applied in a wide range of areas, like wireless sensor networks [18–24], transportation optimization issues [25], high-dimensional expensive problems [26] etc. The search range of the problem is arranged to be [−1,1]. To obtain the true target solution, we need to perform the following transformations:

$$x_i = (x_i^{'} + 1)(x_i^{max} - x_i^{min})/2 + x_i^{min} \quad (1)$$

where $x_i \in [x_i^{min}, x_i^{max}]$, $x_i^{'} \in [-1,1]$.

2.1. DE Algorithm

The original definition of DE can be found in [10,27]. A Gaussian bare-bones differential evolution algorithm can be found in [28]. The steps of DE algorithm include:

1. Initialization: $\{X_i(0)|x_{i,j}(0) \in [x_{i,j}^{min}, x_{i,j}^{max}], i \in [1, Np] \cap N^*, j \in [1, D] \cap N^*\}$, where $X_i(0)$ represents the i-th individual of the 0-th generation, and $x_{i,j}$ represents the j-th dimension of the i-th individual. Initialize the population through an uniform distribution method:

$$x_{i,j}(0) = x_{i,j}^{min} + rand(0,1) * (x_{i,j}^{max} - x_{i,j}^{min}) \tag{2}$$

   rand () represents a random number.
2. Variation: The DE algorithm implements the mutation of individuals through a difference mechanism. A frequently-used difference mechanism is to randomly select two individuals from the population and add their scaled vector difference to the individual vector to be mutated:

$$X_{off}^{'}(gth+1) = X_t(gth) + F * (X_r(gth) - X_s(gth)) \tag{3}$$

   where $X_t(gth)$ is the individual to be mutated in the g-th generation, $X_r(gth)$ and $X_s(gth)$ are two individuals randomly selected from g-th generation. $F \in [0, 2]$ as a constant, represents the scaling factor. $r, s, t \in [1, \text{Np}]$ and $r \neq s \neq t$. $X_{off}^{'}(gth+1)$ is a temporary offspring produced by mutation. This difference mechanism is called rand/1/bin. Other difference mechanisms are proposed in [27].
3. Crossover: The binomial crossover mechanism is implemented for each individual of the g-th generation population $\{x_i(gth)\}$ and its temporary offspring $\{xoff_i^{'}(gth+1)\}$ to obtain the final offspring. The formula is as follows:

$$xoff_{i,j}^{'}(gth+1) = \begin{cases} xoff_{i,j}^{'}(gth+1), & if \quad rand(0,1) \leq Cr \\ x_{i,j}(gth), & otherwise \end{cases} \tag{4}$$

   where $Cr \in [0, 1]$ is a predefined constant, which is called the crossover rate.
4. Selection: The DE algorithm employs a greedy algorithm to choose individuals for the next generation:

$$x_i(gth+1) = \begin{cases} xoff_i(gth+1), & if \quad fit(xoff_i(gth+1)) \geq fit(x_i(gth)) \\ x_i(gth), & otherwise \end{cases} \tag{5}$$

   $fit()$ is a fitness function. In the process of evolution, the individual with higher fitness value will be preserved. The pseudocode of the DE algorithm is shown in Algorithm 1.

2.2. Compact Differential Evolution Algorithm

The cDE algorithm inherits the probability vector (PV) and elites mechanism from rcGA [9,29]. In cDE, PV is an $n * 2$ matrix and is not a vector [30]:

$$PV = [\mu, \sigma] \tag{6}$$

where n is the dimension of the problem. μ and σ are $n * 1$ vectors. μ_i and σ_i respectively represent the mean value and standard deviation of the normal distribution. The normal probability distribution function (PDF) is truncated on the interval $[-1,1]$. To make the area of the probability distribution function equal to 1, we normalize the height of the probability distribution function.

$$PDF_{\mu_i,\sigma_i}(x) = \frac{e^{-\frac{-(x-\mu_i)^2}{2\sigma_i^2}}\sqrt{\frac{2}{\pi}}}{\sigma_i(erf(\frac{\mu_i+1}{\sqrt{2}\sigma_i}) - erf(\frac{\mu_i-1}{\sqrt{2}\sigma_i}))} \tag{7}$$

where $erf()$ is the error function [31]. According to the description in [32], cumulative distribution function (CDF) can be constructed from Chebyshev polynomials. The specific steps of sampling can be expressed as follows:

1. Construct a PDF with PV as a parameter.
2. Construct a cumulative distribution function through Chebyshev polynomials.
3. Generate a random number r over the (0, 1) interval, which is uniformly distributed.
4. Get the corresponding x through inverse cumulative distribution function (ICDF) according to r.

Algorithm 1: DE algorithm pseudocode (rand/1/bin)

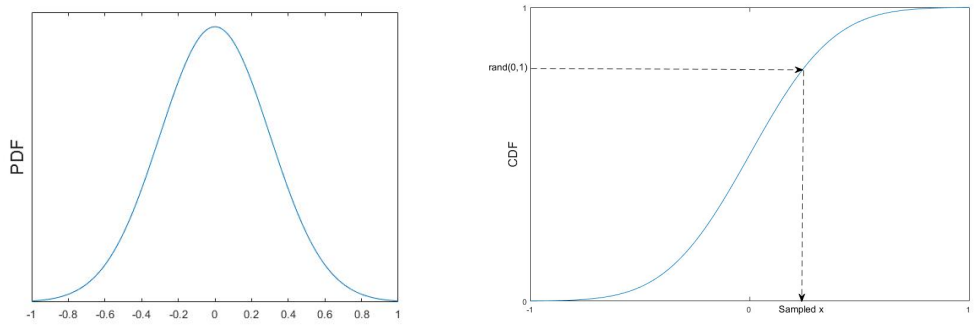
```
//Initialization
generate Np individuals pseudo-randomly
while budget condition do
    for k=1:Np do
        calculate the fit(x_k)
    for k=1:Np do
        //Mutation
        randomly select three individuals x_r, x_s, and x_t
        compute x'_off = x_t + F * (x_r - x_s);
        //Crossover
        x_off = x'_off
        for i= 1:n do
            if rand(0, 1)>Cr then
                x_off[i] = x_k[i]
        //Selection
        if fit(x_off) ≥ fit(x_k) then
            save index for replacement x_k = x_off
    perform replacements
```

The sketch map of the sampling mechanism is shown in Figure 1.

At the beginning of the cDE algorithm, μ is initialized to 0, σ is initialized to 10. A solution is sampled from the solution space as elite. During each iteration, a new temporary offspring will be generated through the new PV. The temporary offspring will be compared with the elite. The one with higher fitness will be recorded as the winner, while the one with lower fitness as a loser. PV update rules are as follows:

$$\mu_i(gth+1) = \mu_i(gth) + \frac{(winner_i - loser_i)}{Np} \tag{8}$$

$$\sigma_i(gth+1) = \sqrt{(\sigma_i(gth))^2 + (\mu_i(gth))^2 - (\mu_i(gth+1))^2 + \frac{(winner_i^2 - loser_i^2)}{Np}} \tag{9}$$



(**a**) PDF (**b**) CDF

Figure 1. Sampling mechanism.

There are two different versions of cDE, which are called persistent elitism compact differential evolution (pe-cDE) and non-persistent elitism compact differential evolution (ne-cDE) (see [9]). For pe-cDE algorithm, the elite is replaced by the temporary offspring when the elite is loser, while for the ne-cDE algorithm, if there is no replacement after the η (a predefined constant representing the threshold) comparison, whatever the state of elite's fitness is, the elite will be replaced by temporary offspring. The pseudocode of pe-cDE and ne-cDE are shown in Algorithms 2 and 3.

Algorithm 2: pe-cDE algorithm pseudocode (rand/1/bin)

$t = 0$
$//PV$ Initialization
for *i=1:n* **do**
    Initialize $\mu[i] = 0$
    Initialize $\sigma[i] = 10$
generate *elite* by PV
while *budget condition* **do**
    //Mutation
    generate 3 individuals x_r, x_s, x_t by PV
    compute $x'_{off} = x_t + F * (x_r - x_s)$
    //Crossover
    $x_{off} = x'_{off}$
    for *i=1:n* **do**
        if *rand(0, 1)>Cr* **then**
            $x_{off}[i] = elite[i]$
    //Selection
    $[winner, loser]$ =compete$(x_{off}, elite)$
    if $x_{off} == winner$ **then**
        $elite = x_{off}$
    $//PV$ Update
    for *i=1:n* **do**
        $\mu^{t+1}[i] = \mu^t[i] + (winner[i] - loser[i])/Np$
        $\sigma^{t+1}[i] = \sqrt{(\sigma^t[i])^2 + (\mu^t[i])^2 - (\mu^{t+1}[i])^2 + (winner[i]^2 - loser[i]^2)/Np}$
    $t = t + 1$

Algorithm 3: ne-cDE algorithm pseudocode (rand/1/bin)

```
t = 0
η = 5
counter = 0
//PV Initialization
same as pe-cDE
while budget condition do
    //Mutation
    same as pe-cDE
    //Crossover
    same as pe-cDE
    //Selection
    [winner, loser] =compete(x_off, elite)
    counter = counter + 1
    if x_off == winner or counter ≥ η then
        elite= x_off
        counter = 0
    //PV Update
    same as pe-cDE
    t = t + 1
```

3. Parallel Compact Differential Evolution

In this section, we propose a parallel cDE algorithm (pcDE). The purpose of parallel processing is to perform calculations on multiple processors at the same time to produce the same results. Parallelism can improve the speed of search and convergence, and find better solutions faster. In [33], parallel particle swarm optimization (PPSO) and three different communication strategies are simultaneously proposed. A new communication strategy for paralleling gray wolf optimization is proposed in [34]. Parameter adaptive DE(PaDE) is proposed in [35], and parallel heterogeneous meta-heuristic is proposed in [36]. It can be seen that the effect of parallelization is much better than that of non-parallelization.

This paper also proposes two parallel strategies. One is to replace the elites in all groups with the most adaptable elites and replace the corresponding PV value at the same time, called the optimal elite strategy(oe); the other is to average the elites in all groups and replace the PV with the average value of PV in all groups at the same time, called the mean elite strategy(me). The difference mechanism rand/1/bin is adopted in this paper, and two different communication strategies are tested under persistent elitism version. The algorithm flow is as follows:

1. Divide the problem into g independent groups, and initialize the PV in each group, where $\mu_i = 0, \sigma_i = 1, i \in [1, n]$.
2. According to the method proposed in Section 2, each group generates an elite.
3. Each group performs mutation and crossover.
4. According to the persistent elitism version, each group compares the generated offspring with the elite, and obtains a new elite and updates the PV according to the rules mentioned in Section 2.
5. Every time after θ (a predefined constant representing the threshold) iterations, the groups communicate with each other. Taking the optimal elite strategy as an example, it firstly calculates the fitness of the elites in each group and then selects the elite with the highest fitness to replace the other elites. PV will also be replaced with the optimal elite's PV .
6. Repeat 2–5 until the program ends.

The pseudocode of pcDE (optimal elite strategy) is shown in Algorithm 4, and the pseudocode of pcDE (mean elite strategy) is shown in Algorithm 5, $m \in [1, g]$.

Algorithm 4: pcDE (optimal elite) algorithm pseudocode (rand/1/bin)

```
t = 0
g = 3
//Groups Initialization
for i=1:g do
    Initialize Group[i].PV
    generate Group[i].elite by Group[i].PV
while budget condition do
    //Mutation
    parallel for every group do
        generate 3 individuals Group[m].x_r, Group[m].x_s, Group[m].x_t by Group[m].PV
        compute Group[m].x'_off = Group[m].x_t + F * (Group[m].x_r - Group[m].x_s)
        //Crossover
        Group[m].x_off = Group[m].x'_off
        for i=1:n do
            if rand(0, 1)>Cr then
                Group[m].x_off[i] = Group[m].elite[i]
        //Selection
        [winner, loser] =compete(Group[m].x_off, Group[m].elite)
        if Group[m].x_off == winner then
            Group[m].elite = Group[m].x_off
        //PV Update
        for i=1:n do
            Group[m].μ^{t+1}[i] = Group[m].μ^t[i] + (winner[i] - loser[i])/Np
            temp = (Group[m].σ^t[i])^2 + (Group[m].μ^t[i])^2 - (Group[m].μ^{t+1}[i])^2
            Group[m].σ^{t+1}[i] = sqrt(temp + (winner[i]^2 - loser[i]^2)/Np)
    Find the group with the most adaptable solution and record the group number as I
    if mod(t, θ) == 0 then
        for i=1:g do
            Group[i].PV = Group[I].PV
            Group[i].elite = Group[I].elite
    t = t + 1
```

Algorithm 5: pcDE (mean elite) algorithm pseudocode (rand/1/bin)

```
t = 0
g = 3
//Groups Initialization
same as pcDE(optimal elite)
while budget condition do
    //Mutation
    parallel for every group do
        same as ppe-cDE(optimal elite)
        //Crossover
        same as ppe-cDE(optimal elite)
        //Selection
        same as ppe-cDE(optimal elite)
        //PV Update
        same as ppe-cDE(optimal elite)
    if mod(t, θ) == 0 then
        for i=1:n do
            Initialize PVsum[i] = 0
            Initialize elitesum[i] = 0
        for i=1:g do
            PVsum = PVsum + Group[i].PV
            elitesum = elitesum + Group[i].elite
        for i=1:g do
            Group[i].PV = PVsum/g
            Group[i].elite = elitesum/g
    t = t + 1
```

4. Numerical Results

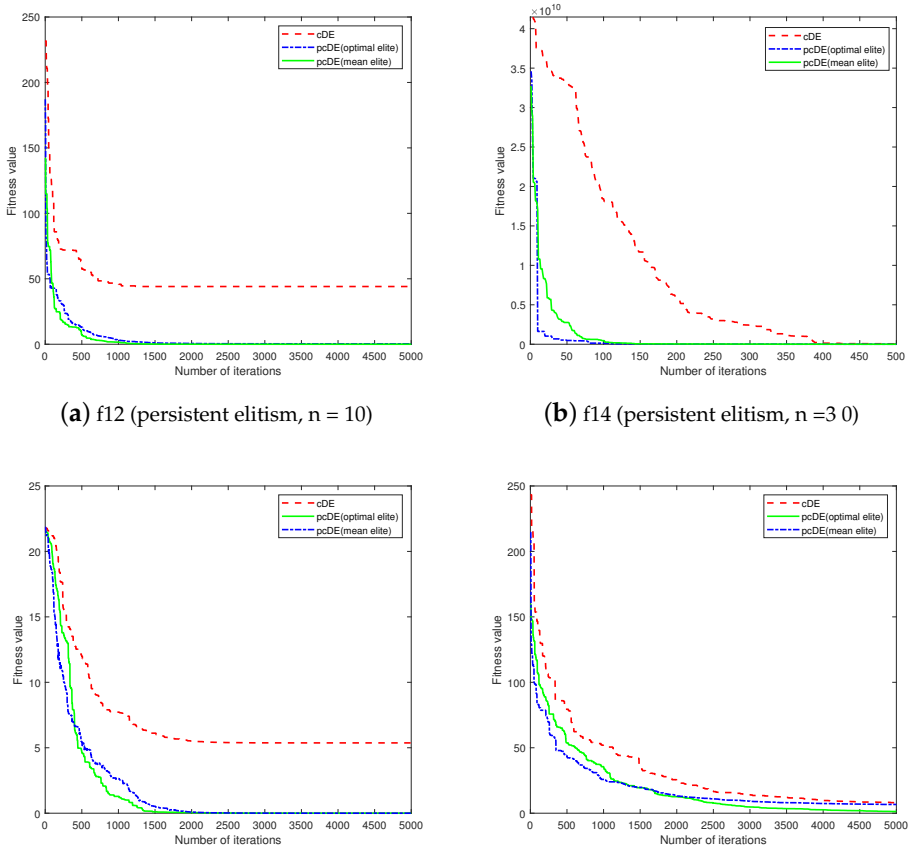
Figure 2 shows the test function used in this paper, with f1, f2, f8, f9, f15–f17 coming from [37], f3–f7, f10, f19 from [38], f11–f14, f20 from [39], f18, f21–f25 from [40].

For each test function, we test it with the previously mentioned algorithms. f1–f17 are tested with $n = 10$ and $n = 30$; f18–f25 are tested in the specified dimensions. Each algorithm runs independently for 10 times. The number of parallel groups is set to 3. Each test is performed in $5000 * n$ iterations. The number of virtual population Np is set as $2 * n$. The mean of 10 independent experiments is used as the experimental result. Table 1 shows the experimental results of two parallel strategies (persistent elitism version, F = 0.1, Cr = 0.1, rand/1/bin). Table 2 shows the experimental results of two parallel strategies (non-persistent elitism version, F = 0.1, Cr = 0.1, rand/1/bin). The best results are highlighted in boldface. From Table 1, we can see that in the 17 tests with n = 10, pcDE achieves the best results of 15 tests, while cDE only achieves 2; when n = 30, the result is the same; in the 8 tests with n = various, pcDE achieves the best results of 6 tests, while cDE achieves only 2. From Table 2, we can see that in the 17 tests with n = 10, pcDE achieves the best results of 16 tests, while CDE only achieves 1; when n = 30, pcDE achieves the best results of 14 tests, while cDE achieves only 3; in the 8 tests with n = various, pcDE achieves the best results of all tests. From the above data, we can see that the performance of pcDE is significantly better than that of cDE. Figure 3 shows the performance trends of pcDE and cDE algorithms. We also test the effects of two communication strategies on experimental results under three different schemes. The three schemes are:1.rand/1/bin, 2.best/1/bin, 3.rand-to-best/1/bin. Table 3 shows the test results, where '+' represents that the performance of the

algorithm after parallelism is better than the original algorithm, '−' represents the opposite. Figure 4 shows performance trends of pcDE and cDE algorithms under three different strategies.

number	name	number	name
1	Shifted sphere function	13	Generalized penalized function 1
2	Shifted schwsfel's problem 1.2	14	Generalized penalized function 2
3	Rosenbrock's function	15	Schwefel's problem 2. 6 with Global Optimum on Bounds
4	Shifted Ackley's function	16	Shifted rotated Weierstrass function
5	Shifted rotated Ackley's function	17	Schwefel's problem 2. 13
6	Shifted Griewank's function	18	Kowalik's function
7	Shifted rotated Griewank's function	19	Six-hump camel-back function
8	Shifted Rastrigin's function	20	Branin function
9	Shifted rotated Rastrigin's function	21	Hartman's function 1
10	Shifted noncontinuous Rastrigin's function	22	Hartman's function 2
11	Schwefel problem 2. 22	23-25	Shekel's family
12	Schwefel problem 2. 21		

Figure 2. Test function.



(**a**) f12 (persistent elitism, n = 10)

(**b**) f14 (persistent elitism, n =3 0)

(**c**) f4 (non-persistent elitism, n = 10)

(**d**) f12 (non-persistent elitism, n = 30)

Figure 3. Performance trends pcDE vs. cDE.

Table 1. Final average fitness ± standard deviation(persistent elitism).

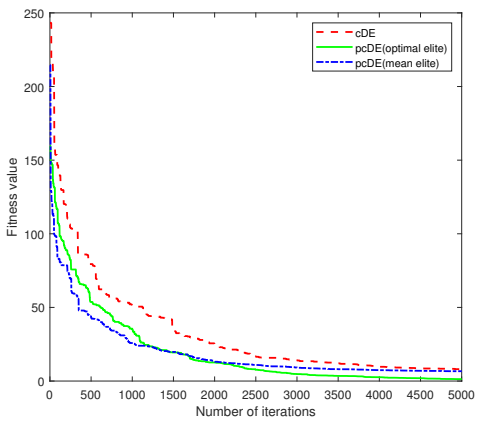
Function	cDE	pcDE (Optimal Elite)	pcDE (Mean Elite)
		n = 10	
1	$6.652 \times 10^{-2} \pm 2.00 \times 10^{-1}$	$2.424 \times 10^{-14} \pm 1.18 \times 10^{-14}$	$\mathbf{8.056 \times 10^{-18} \pm 5.13 \times 10^{-18}}$
2	$1.261 \times 10^{-2} \pm 3.24 \times 10^{-2}$	$\mathbf{5.385 \times 10^{-12} \pm 6.72 \times 10^{-12}}$	$9.841 \times 10^{0} \pm 8.58 \times 10^{0}$
3	$7.420 \times 10^{1} \pm 1.60 \times 10^{2}$	$4.144 \times 10^{1} \pm 5.88 \times 10^{1}$	$\mathbf{2.089 \times 10^{1} \pm 2.72 \times 10^{1}}$
4	$3.998 \times 10^{0} \pm 8.00 \times 10^{0}$	$4.547 \times 10^{-9} \pm 8.49 \times 10^{-9}$	$\mathbf{9.988 \times 10^{-10} \pm 5.68 \times 10^{-10}}$
5	$1.064 \times 10^{-1} \pm 9.29 \times 10^{-2}$	$\mathbf{4.885 \times 10^{-2} \pm 3.64 \times 10^{-2}}$	$2.491 \times 10^{-1} \pm 1.15 \times 10^{-1}$
6	$1.792 \times 10^{-16} \pm 7.72 \times 10^{-17}$	$\mathbf{1.411 \times 10^{-16} \pm 2.47 \times 10^{-32}}$	$1.342 \times 10^{-9} \pm 7.02 \times 10^{-10}$
7	$8.881 \times 10^{-1} \pm 9.13 \times 10^{-1}$	$\mathbf{8.518 \times 10^{-1} \pm 7.60 \times 10^{-1}}$	$2.361 \times 10^{0} \pm 1.47 \times 10^{0}$
8	$8.955 \times 10^{-1} \pm 6.96 \times 10^{-1}$	$\mathbf{1.027 \times 10^{-10} \pm 1.09 \times 10^{-10}}$	$4.601 \times 10^{0} \pm 5.37 \times 10^{0}$
9	$4.467 \times 10^{1} \pm 1.06 \times 10^{1}$	$3.831 \times 10^{1} \pm 1.63 \times 10^{1}$	$\mathbf{1.851 \times 10^{1} \pm 8.15 \times 10^{0}}$
10	$1.700 \times 10^{0} \pm 1.19 \times 10^{0}$	$\mathbf{2.000 \times 10^{-1} \pm 4.00 \times 10^{-1}}$	$5.100 \times 10^{0} \pm 1.04 \times 10^{0}$
11	$\mathbf{0.000 \times 10^{0} \pm 0.00 \times 10^{0}}$	$\mathbf{0.000 \times 10^{0} \pm 0.00 \times 10^{0}}$	$1.448 \times 10^{-11} \pm 3.77 \times 10^{-12}$
12	$8.391 \times 10^{-8} \pm 2.44 \times 10^{-7}$	$\mathbf{1.293 \times 10^{-13} \pm 3.01 \times 10^{-13}}$	$7.883 \times 10^{-1} \pm 5.60 \times 10^{-1}$
13	$7.280 \times 10^{-21} \pm 1.89 \times 10^{-20}$	$\mathbf{4.627 \times 10^{-21} \pm 8.78 \times 10^{-21}}$	$4.858 \times 10^{-21} \pm 4.37 \times 10^{-21}$
14	$8.588 \times 10^{-21} \pm 2.54 \times 10^{-20}$	$\mathbf{2.515 \times 10^{-21} \pm 7.31 \times 10^{-21}}$	$3.552 \times 10^{-21} \pm 5.94 \times 10^{-21}$
15	$1.103 \times 10^{4} \pm 2.88 \times 10^{3}$	$\mathbf{2.553 \times 10^{3} \pm 1.13 \times 10^{3}}$	$1.567 \times 10^{4} \pm 7.88 \times 10^{2}$
16	$8.291 \times 10^{0} \pm 1.96 \times 10^{0}$	$\mathbf{8.099 \times 10^{0} \pm 1.61 \times 10^{0}}$	$2.696 \times 10^{1} \pm 2.45 \times 10^{0}$
17	$\mathbf{2.031 \times 10^{2} \pm 2.53 \times 10^{2}}$	$2.505 \times 10^{3} \pm 3.54 \times 10^{3}$	$6.684 \times 10^{2} \pm 7.85 \times 10^{2}$
		n = 30	
1	$1.825 \times 10^{2} \pm 5.25 \times 10^{2}$	$2.782 \times 10^{1} \pm 8.33 \times 10^{1}$	$\mathbf{4.601 \times 10^{-16} \pm 2.95 \times 10^{-16}}$
2	$8.721 \times 10^{1} \pm 1.87 \times 10^{2}$	$\mathbf{1.836 \times 10^{-6} \pm 5.44 \times 10^{-6}}$	$9.082 \times 10^{2} \pm 3.95 \times 10^{2}$
3	$1.668 \times 10^{2} \pm 1.98 \times 10^{2}$	$7.906 \times 10^{1} \pm 7.62 \times 10^{1}$	$\mathbf{4.823 \times 10^{1} \pm 3.67 \times 10^{1}}$
4	$3.047 \times 10^{-9} \pm 4.40 \times 10^{-9}$	$\mathbf{7.489 \times 10^{-10} \pm 7.01 \times 10^{-10}}$	$3.319 \times 10^{-9} \pm 8.63 \times 10^{-10}$
5	$8.948 \times 10^{-2} \pm 3.63 \times 10^{-2}$	$\mathbf{3.334 \times 10^{-2} \pm 2.01 \times 10^{-2}}$	$3.300 \times 10^{-1} \pm 8.92 \times 10^{-2}$
6	$1.180 \times 10^{-15} \pm 2.01 \times 10^{-15}$	$\mathbf{4.232 \times 10^{-16} \pm 0.00 \times 10^{0}}$	$2.306 \times 10^{-7} \pm 8.90 \times 10^{-8}$
7	$2.171 \times 10^{-2} \pm 2.39 \times 10^{-2}$	$\mathbf{1.378 \times 10^{-2} \pm 1.20 \times 10^{-2}}$	$1.208 \times 10^{0} \pm 9.71 \times 10^{-2}$
8	$9.950 \times 10^{-2} \pm 2.98 \times 10^{-1}$	$\mathbf{1.287 \times 10^{-10} \pm 6.95 \times 10^{-11}}$	$4.804 \times 10^{-1} \pm 4.04 \times 10^{-1}$
9	$2.372 \times 10^{2} \pm 8.38 \times 10^{1}$	$2.086 \times 10^{2} \pm 7.00 \times 10^{1}$	$\mathbf{1.151 \times 10^{2} \pm 4.77 \times 10^{1}}$
10	$1.210 \times 10^{1} \pm 2.98 \times 10^{0}$	$\mathbf{4.300 \times 10^{0} \pm 1.35 \times 10^{0}}$	$1.010 \times 10^{1} \pm 1.22 \times 10^{0}$
11	$\mathbf{0.000 \times 10^{0} \pm 0.00 \times 10^{0}}$	$\mathbf{0.000 \times 10^{0} \pm 0.00 \times 10^{0}}$	$2.985 \times 10^{-10} \pm 4.36 \times 10^{-11}$
12	$5.433 \times 10^{0} \pm 1.21 \times 10^{0}$	$\mathbf{1.046 \times 10^{-2} \pm 2.25 \times 10^{-2}}$	$3.663 \times 10^{0} \pm 1.59 \times 10^{0}$
13	$\mathbf{5.513 \times 10^{-22} \pm 5.71 \times 10^{-22}}$	$1.075 \times 10^{-21} \pm 1.70 \times 10^{-21}$	$1.411 \times 10^{-20} \pm 5.95 \times 10^{-21}$
14	$1.955 \times 10^{-22} \pm 1.39 \times 10^{-22}$	$\mathbf{4.474 \times 10^{-23} \pm 2.81 \times 10^{-23}}$	$1.656 \times 10^{-20} \pm 6.59 \times 10^{-21}$
15	$1.329 \times 10^{4} \pm 2.16 \times 10^{3}$	$\mathbf{9.367 \times 10^{3} \pm 1.03 \times 10^{3}}$	$1.970 \times 10^{4} \pm 9.12 \times 10^{2}$
16	$3.031 \times 10^{1} \pm 3.41 \times 10^{0}$	$2.973 \times 10^{1} \pm 1.99 \times 10^{0}$	$\mathbf{2.933 \times 10^{1} \pm 2.50 \times 10^{0}}$
17	$4.361 \times 10^{3} \pm 3.78 \times 10^{3}$	$\mathbf{3.079 \times 10^{3} \pm 2.54 \times 10^{3}}$	$1.642 \times 10^{4} \pm 1.18 \times 10^{4}$
		n various	
18	$6.540 \times 10^{-3} \pm 5.03 \times 10^{-3}$	$\mathbf{5.065 \times 10^{-3} \pm 6.53 \times 10^{-3}}$	$5.317 \times 10^{-3} \pm 1.50 \times 10^{-3}$
19	$3.333 \times 10^{2} \pm 7.44 \times 10^{2}$	$-7.316 \times 10^{-1} \pm 8.81 \times 10^{-1}$	$\mathbf{-1.032 \times 10^{0} \pm 8.26 \times 10^{-8}}$
20	$1.529 \times 10^{0} \pm 1.44 \times 10^{0}$	$\mathbf{4.154 \times 10^{-1} \pm 4.17 \times 10^{-2}}$	$5.507 \times 10^{-1} \pm 4.40 \times 10^{-1}$
21	$\mathbf{-3.859 \times 10^{0} \pm 8.18 \times 10^{-3}}$	$-3.785 \times 10^{0} \pm 2.32 \times 10^{-1}$	$-3.863 \times 10^{0} \pm 3.97 \times 10^{-16}$
22	$-3.274 \times 10^{0} \pm 5.82 \times 10^{-2}$	$-3.263 \times 10^{0} \pm 5.94 \times 10^{-2}$	$\mathbf{-3.298 \times 10^{0} \pm 4.76 \times 10^{-2}}$
23	$\mathbf{-6.147 \times 10^{0} \pm 3.39 \times 10^{0}}$	$-5.609 \times 10^{0} \pm 2.45 \times 10^{0}$	$-4.419 \times 10^{0} \pm 2.96 \times 10^{0}$
24	$-4.982 \times 10^{0} \pm 2.94 \times 10^{0}$	$-5.581 \times 10^{0} \pm 3.23 \times 10^{0}$	$\mathbf{-6.911 \times 10^{0} \pm 3.55 \times 10^{0}}$
25	$-3.424 \times 10^{0} \pm 2.23 \times 10^{0}$	$\mathbf{-7.477 \times 10^{0} \pm 3.76 \times 10^{0}}$	$-5.178 \times 10^{0} \pm 3.53 \times 10^{0}$

Table 2. Final average fitness ± standard deviation (non-persistent elitism).

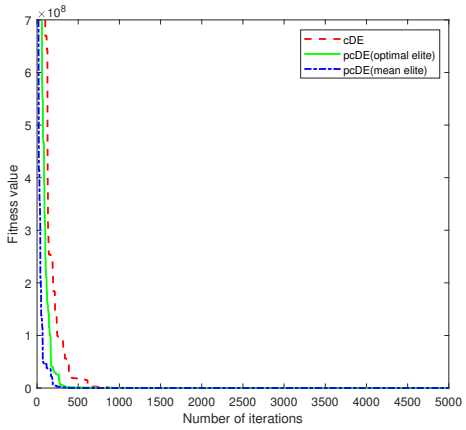
Function	cDE	pcDE (Optimal Elite)	pcDE (Mean Elite)
		n = 10	
1	$4.435 \times 10^{-5} \pm 1.33 \times 10^{-4}$	$3.894 \times 10^{-14} \pm 1.23 \times 10^{-14}$	$\mathbf{1.573 \times 10^{-17} \pm 1.36 \times 10^{-17}}$
2	$1.777 \times 10^{3} \pm 1.70 \times 10^{3}$	$\mathbf{1.235 \times 10^{-9} \pm 3.70 \times 10^{-9}}$	$6.089 \times 10^{2} \pm 3.80 \times 10^{2}$
3	$8.042 \times 10^{7} \pm 2.41 \times 10^{8}$	$\mathbf{1.803 \times 10^{2} \pm 3.85 \times 10^{2}}$	$3.112 \times 10^{2} \pm 4.56 \times 10^{2}$
4	$7.714 \times 10^{0} \pm 9.46 \times 10^{0}$	$1.833 \times 10^{0} \pm 5.50 \times 10^{0}$	$\mathbf{1.450 \times 10^{-9} \pm 1.04 \times 10^{-9}}$
5	$2.388 \times 10^{-1} \pm 1.31 \times 10^{-1}$	$\mathbf{8.729 \times 10^{-2} \pm 7.61 \times 10^{-2}}$	$4.450 \times 10^{-1} \pm 2.23 \times 10^{-1}$
6	$3.797 \times 10^{-14} \pm 6.97 \times 10^{-14}$	$\mathbf{1.411 \times 10^{-16} \pm 2.47 \times 10^{-32}}$	$3.720 \times 10^{-9} \pm 1.75 \times 10^{-9}$
7	$1.054 \times 10^{0} \pm 1.06 \times 10^{0}$	$\mathbf{5.912 \times 10^{-1} \pm 2.95 \times 10^{-1}}$	$1.906 \times 100 \pm 1.00 \times 10^{0}$
8	$6.965 \times 10^{-1} \pm 6.37 \times 10^{-1}$	$\mathbf{1.188 \times 10^{-10} \pm 1.68 \times 10^{-10}}$	$6.666 \times 10^{0} \pm 7.69 \times 10^{0}$
9	$4.875 \times 10^{1} \pm 1.87 \times 10^{1}$	$5.104 \times 10^{1} \pm 2.45 \times 10^{1}$	$\mathbf{4.338 \times 10^{1} \pm 3.47 \times 10^{1}}$
10	$9.385 \times 10^{2} \pm 2.80 \times 10^{3}$	$\mathbf{4.000 \times 10^{-1} \pm 6.63 \times 10^{-1}}$	$6.000 \times 10^{0} \pm 1.79 \times 10^{0}$
11	$1.993 \times 10^{-3} \pm 5.98 \times 10^{-3}$	$\mathbf{0.000 \times 10^{0} \pm 0.00 \times 10^{0}}$	$1.483 \times 10^{-11} \pm 2.69 \times 10^{-12}$
12	$1.234 \times 10^{2} \pm 5.18 \times 10^{1}$	$4.489 \times 10^{1} \pm 2.92 \times 10^{1}$	$\mathbf{1.894 \times 10^{0} \pm 9.73 \times 10^{-1}}$
13	$4.789 \times 10^{5} \pm 1.44 \times 10^{6}$	$\mathbf{7.524 \times 10^{-22} \pm 2.23 \times 10^{-21}}$	$1.206 \times 10^{-20} \pm 2.67 \times 10^{-20}$
14	$3.355 \times 10^{-2} \pm 5.22 \times 10^{-2}$	$\mathbf{2.428 \times 10^{-23} \pm 3.69 \times 10^{-23}}$	$3.350 \times 10^{-21} \pm 3.81 \times 10^{-21}$
15	$1.539 \times 10^{4} \pm 1.65 \times 10^{3}$	$\mathbf{4.718 \times 10^{3} \pm 3.05 \times 10^{3}}$	$1.719 \times 10^{4} \pm 5.84 \times 10^{2}$
16	$\mathbf{8.104 \times 10^{0} \pm 1.90 \times 10^{0}}$	$8.153 \times 10^{0} \pm 1.46 \times 10^{0}$	$2.813 \times 10^{1} \pm 3.49 \times 10^{0}$
17	$1.742 \times 10^{3} \pm 3.12 \times 10^{3}$	$\mathbf{5.582 \times 10^{2} \pm 1.61 \times 10^{3}}$	$1.469 \times 10^{3} \pm 1.53 \times 10^{3}$
		n = 30	
1	$1.484 \times 10^{-11} \pm 3.34 \times 10^{-12}$	$8.738 \times 10^{-12} \pm 1.36 \times 10^{-12}$	$\mathbf{7.044 \times 10^{-16} \pm 2.90 \times 10^{-16}}$
2	$2.861 \times 10^{3} \pm 1.55 \times 10^{3}$	$\mathbf{2.157 \times 10^{-1} \pm 3.23 \times 10^{-1}}$	$4.242 \times 10^{3} \pm 1.18 \times 10^{3}$
3	$7.753 \times 10^{2} \pm 1.50 \times 10^{3}$	$\mathbf{2.342 \times 10^{1} \pm 3.13 \times 10^{1}}$	$3.136 \times 10^{2} \pm 3.96 \times 10^{2}$
4	$\mathbf{8.781 \times 10^{-10} \pm 1.75 \times 10^{-10}}$	$1.874 \times 10^{-9} \pm 3.21 \times 10^{-9}$	$5.286 \times 10^{-9} \pm 8.20 \times 10^{-10}$
5	$3.419 \times 10^{-1} \pm 7.07 \times 10^{-2}$	$\mathbf{7.626 \times 10^{-2} \pm 4.47 \times 10^{-2}}$	$1.337 \times 10^{0} \pm 5.62 \times 10^{-1}$
6	$5.540 \times 10^{-16} \pm 2.76 \times 10^{-16}$	$\mathbf{4.232 \times 10^{-16} \pm 0.00 \times 10^{0}}$	$4.030 \times 10^{-7} \pm 1.01 \times 10^{-7}$
7	$1.881 \times 10^{-2} \pm 1.06 \times 10^{-2}$	$\mathbf{1.379 \times 10^{-2} \pm 1.25 \times 10^{-2}}$	$1.230 \times 10^{0} \pm 9.22 \times 10^{-2}$
8	$7.503 \times 10^{-10} \pm 2.30 \times 10^{-10}$	$\mathbf{1.503 \times 10^{-10} \pm 1.67 \times 10^{-10}}$	$1.010 \times 10^{0} \pm 9.76 \times 10^{-1}$
9	$2.830 \times 10^{2} \pm 8.07 \times 10^{1}$	$1.797 \times 10^{2} \pm 4.65 \times 10^{1}$	$\mathbf{9.134 \times 10^{1} \pm 2.68 \times 10^{1}}$
10	$2.138 \times 10^{3} \pm 6.36 \times 10^{3}$	$\mathbf{4.900 \times 10^{0} \pm 1.92 \times 10^{0}}$	$1.530 \times 10^{1} \pm 2.45 \times 10^{0}$
11	$0\mathbf{.000 \times 10^{0} \pm 0.00 \times 10^{0}}$	$\mathbf{0.000 \times 10^{0} \pm 0.00 \times 10^{0}}$	$3.821 \times 10^{-10} \pm 5.27 \times 10^{-11}$
12	$1.806 \times 10^{2} \pm 5.88 \times 10^{1}$	$9.494 \times 10^{1} \pm 5.47 \times 10^{1}$	$\mathbf{5.294 \times 10^{0} \pm 1.32 \times 10^{0}}$
13	$1.037 \times 10^{-2} \pm 3.11 \times 10^{-2}$	$\mathbf{3.284 \times 10^{-22} \pm 6.52 \times 10^{-22}}$	$3.027 \times 10^{-20} \pm 8.22 \times 10^{-21}$
14	$\mathbf{1.896 \times 10^{-22} \pm 1.46 \times 10^{-22}}$	$3.769 \times 10^{-22} \pm 9.26 \times 10^{-22}$	$4.130 \times 10^{-20} \pm 3.21 \times 10^{-20}$
15	$1.707 \times 10^{4} \pm 1.47 \times 10^{3}$	$\mathbf{1.005 \times 10^{4} \pm 1.25 \times 10^{3}}$	$2.241 \times 10^{4} \pm 8.27 \times 10^{2}$
16	$3.033 \times 10^{1} \pm 4.37 \times 10^{0}$	$\mathbf{2.764 \times 10^{1} \pm 3.90 \times 10^{0}}$	$2.830 \times 10^{1} \pm 3.44 \times 10^{0}$
17	$8.176 \times 10^{3} \pm 5.21 \times 10^{3}$	$\mathbf{3.101 \times 10^{3} \pm 1.89 \times 10^{3}}$	$2.839 \times 10^{4} \pm 2.10 \times 10^{4}$
		n various	
18	$1.537 \times 10^{-2} \pm 1.31 \times 10^{-2}$	$\mathbf{3.471 \times 10^{-3} \pm 2.45 \times 10^{-3}}$	$5.098 \times 10^{-3} \pm 2.78 \times 10^{-3}$
19	$1.256 \times 10^{2} \pm 3.53 \times 10^{2}$	$-5.670 \times 10^{-1} \pm 5.79 \times 10^{-1}$	$\mathbf{-9.500 \times 10^{-1} \pm 2.45 \times 10^{-1}}$
20	$5.077 \times 10^{0} \pm 6.41 \times 10^{0}$	$2.343 \times 10^{0} \pm 5.42 \times 10^{0}$	$\mathbf{4.655 \times 10^{-1} \pm 1.96 \times 10^{-1}}$
21	$-3.595 \times 10^{0} \pm 3.39 \times 10^{-1}$	$\mathbf{-3.863 \times 10^{0} \pm 2.25 \times 10^{-5}}$	$-3.554 \times 10^{0} \pm 3.79 \times 10^{-1}$
22	$-3.113 \times 10^{0} \pm 4.32 \times 10^{-1}$	$-3.286 \times 10^{0} \pm 5.45 \times 10^{-2}$	$\mathbf{-3.298 \times 10^{0} \pm 4.76 \times 10^{-2}}$
23	$-3.868 \times 10^{0} \pm 3.44 \times 10^{0}$	$-6.155 \times 10^{0} \pm 3.38 \times 10^{0}$	$\mathbf{-6.402 \times 10^{0} \pm 3.75 \times 10^{0}}$
24	$-3.278 \times 10^{0} \pm 2.80 \times 10^{0}$	$\mathbf{-6.291 \times 10^{0} \pm 3.47 \times 10^{0}}$	$-5.917 \times 10^{0} \pm 3.70 \times 10^{0}$
25	$-3.284 \times 10^{0} \pm 2.63 \times 10^{0}$	$-5.200 \times 10^{0} \pm 3.61 \times 10^{0}$	$\mathbf{-5.989 \times 10^{0} \pm 3.73 \times 10^{0}}$

Table 3. Two communication strategies under three different schemes.

Function	Scheme 1: Rand/1/Bin		Scheme 2: Best/1/Bin		Scheme 3: Rand-to-Best/1/Bin	
	pcDE (Optimal Elite) and pcDE	pcDE (Mean Elite) and pcDE	pcDE (Optimal Elite) and pcDE	pcDE (Mean Elite) and pcDE	pcDE (Optimal Elite) and pcDE	pcDE (Mean Elite) and pcDE
			n = 30			
1	+	+	+	−	+	+
2	+	−	+	−	+	−
3	+	+	−	−	−	−
4	+	−	+	+	+	+
5	+	−	+	−	+	+
6	+	−	+	−	+	+
7	+	−	+	−	+	−
8	+	−	+	−	+	−
9	+	+	−	+	+	+
10	+	+	+	−	−	−
11	−	−	+	−	+	+
12	+	+	+	+	+	+
13	−	−	+	−	+	+
14	+	−	+	−	+	+
15	+	−	+	−	−	−
16	+	+	+	−	+	+
17	+	−	+	−	+	−
			n various			
18	+	+	+	+	+	+
19	+	+	+	+	+	+
20	+	+	+	+	+	+
21	−	−	+	+	+	+
22	+	−	+	+	−	−
23	−	−	−	+	+	+
24	+	+	+	+	−	+
25	+	+	+	+	+	+

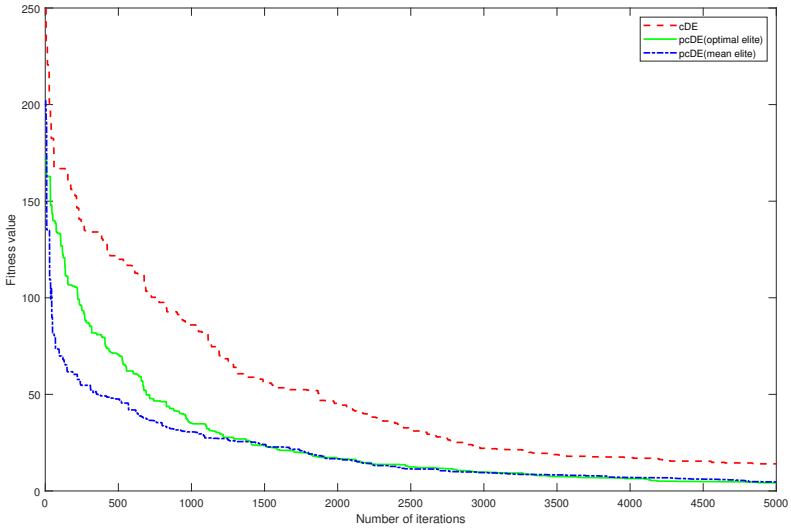


(**a**) f12 (scheme 1, n = 30)



(**b**) f14 (scheme 3, n = 30)

Figure 4. *Cont.*



(c) f12 (scheme 2, n = 30)

Figure 4. Performance trends of pcDE and cDE algorithms under three different strategies.

5. Case of Study: Parallel Compact Differential Evolution for Image Segmentation

In this section, we employ the pcDE algorithm to implement image threshold segmentation. The differences between pcDE and traditional methods are compared. For the image Img(x, y), we suppose that its size is $width * height$ and the gray level is L, the neighborhood average gray value g of $(2 * n + 1)^2$ at the (x, y) point is:

$$g(x,y) = \frac{1}{(2*n+1)^2} \sum_{i=-n}^{n} \sum_{j=-n}^{n} Img(x+i, y+j) \tag{10}$$

We use the point gray value and $(2 * n + 1) * (2 * n + 1)$ neighborhood average gray value to establish a two-dimensional gray histogram F(x, y), then segment F(x, y) with a two-dimensional threshold (s, t). Let $N_{i,j}$ be the number of pixels whose gray value is i and neighborhood average gray value is j, $P_{i,j}$ be the probability, then

$$P_{i,j} = \frac{N_{i,j}}{width * height} \tag{11}$$

$\{P_{i,j}, i, j = 1, 2, 3...L\}$ is the two-dimensional gray histogram of the image, which is shown in Figure 5. Figure 5 shows that the peak of the probability of the point-neighbor average gray is mainly distributed near the diagonal of the plane, and the overall appearance is bimodal. It is because the target and background account for the largest proportion of all pixels, and the gray value of the pixels inside the target area and the background area are relatively uniform. The gray values of the points are similar to their neighborhood average gray value, so they are concentrated near the diagonal. The two peaks respectively represent target and background. Away from the diagonal, the height of the peak drops sharply. This part reflects the image noise, edges and other information. A two-dimensional histogram XOY plan view is shown in Figure 6. Region A and B correspond to the target and background, and region C and D to the edges and noise. The optimal threshold is determined by the pcDE algorithm to maximize the amount of information that truly represents the target and background.

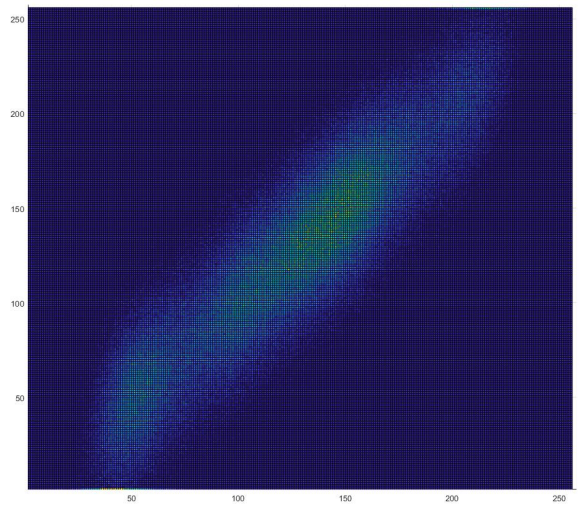


Figure 5. Two-dimensional gray histogram.

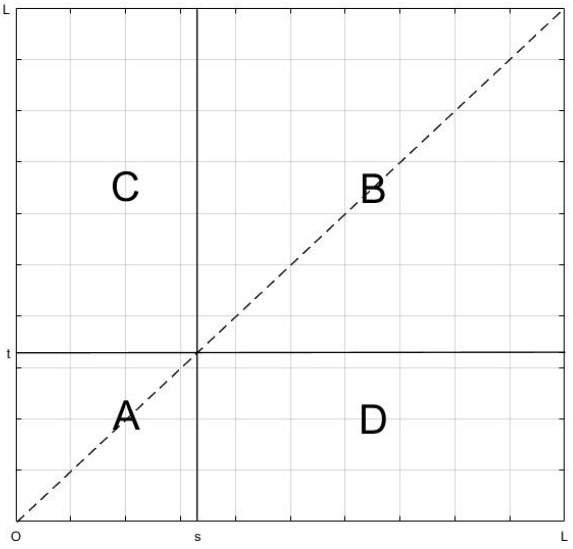


Figure 6. Two-dimensional histogram XOY plan view.

Assuming that the target region and the background region have different probability distributions, we normalize each region with the posterior probability of the target and the background region, so that the entropy of each region is additive. We suppose the threshold is (s, t), then the discrete two-dimensional entropy is:

$$H = -\sum_{i}\sum_{j} P_{i,j} lg P_{i,j} \tag{12}$$

The two-dimensional entropy function of the target and the background is defined as:

$$\phi(s,t) = H_A/P_A + log_{10}[P_A(1-P_A)] + (H_L - H_A)/(1-P_A) \tag{13}$$

where $P_A = \sum_{i=1}^{s}\sum_{j=1}^{t} P_{i,j}$, $H_A = -\sum_{i=1}^{s}\sum_{j=1}^{t} P_{i,j} lg P_{i,j}$. H_L is the entropy when (s, t) takes the maximum gray value. The calculation method is the same as H_A. Using the formula (13) as the

evaluation function, the pcDE algorithm is used to calculate the fitness value at different thresholds. Finally we get the optimal threshold. Table 4 shows the final fitness and standard deviation of the Lena images (Figure 7) tested with four algorithms, s and t represent the final two-dimensional segmentation threshold (s,t∈[0,255]). The performance comparison among these four algorithms is shown in Figure 8. In the following section, we segment a medical image of lung and compare the experimental results of Otsu method, minimum error thresholding method, and pcDE. The main steps include:

1. Image preprocessing. Enhance the contrast of the image to achieve better segmentation.
2. Calculate the optimal threshold for image segmentation through different methods to get the binarized image.
3. Use morphological methods to process images to extract targets.
4. Mark target contour.

Table 4. Final fitness and standard deviation of the Lena images tested with four algorithms.

Algorithm	Mean	Standard Deviation	s	t
cDE(persistent elitism)	1.433×10^{1}	2.972×10^{-2}	122	118
pcDE(persistent elitism)	1.435×10^{1}	1.947×10^{-3}	103	105
cDE(non-persistent elitism)	1.423×10^{1}	6.654×10^{-2}	115	112
pcDE(non-persistent elitism)	1.435×10^{1}	8.321×10^{-3}	106	107

Figure 7. Lena image.

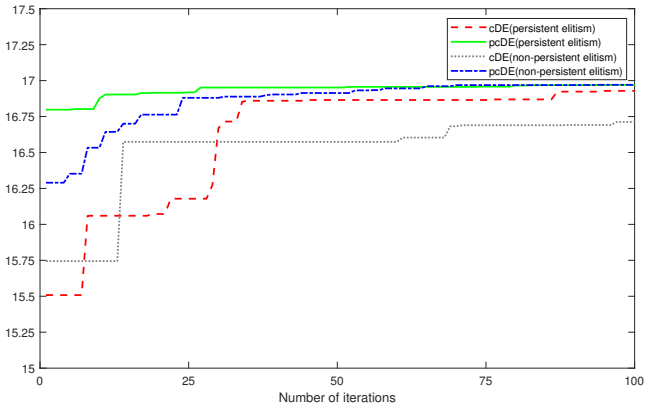
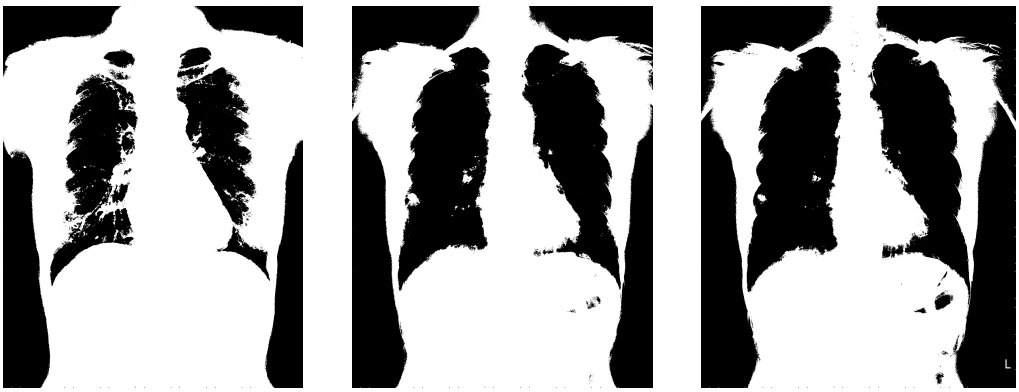


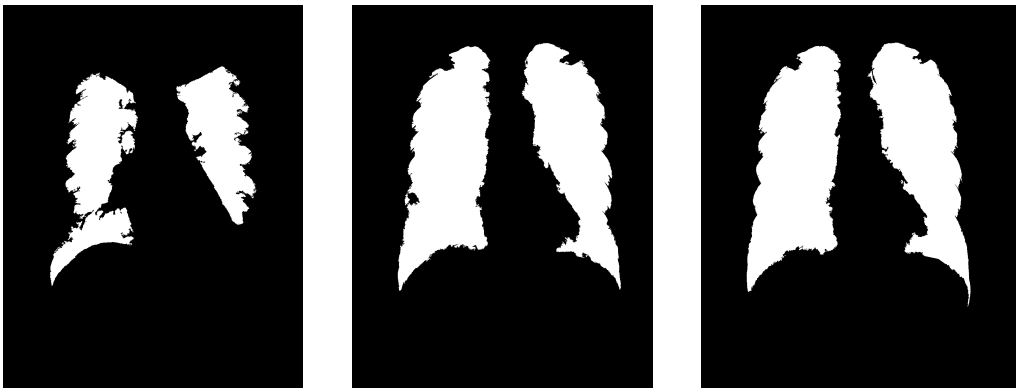
Figure 8. Performance trends of four algorithms.

Figure 9 shows the experimental result, in which (a)–(c) are the images after threshold segmentation, and (d) and (e) are the images after morphological processing, and (g) is the original image, and (h) is the contrast-enhanced image. Figure 10 shows the contour marked image. From the

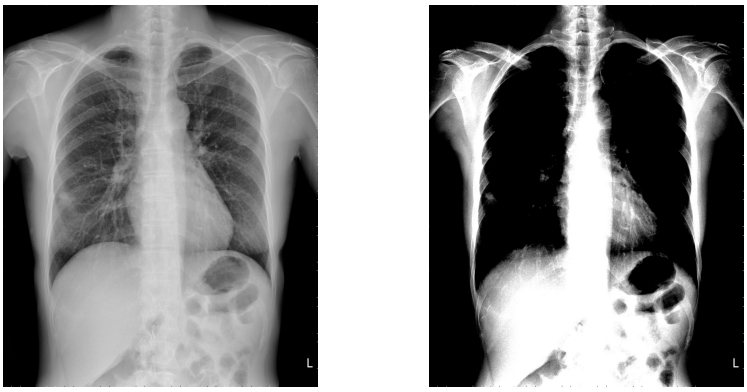
above experimental results, we can see that the image segmented via the pcDE algorithm is much closer to the reality, compared with other algorithms.



(**a**) Otsu (**b**) minimum error thresholding (**c**) pcDE



(**d**) Otsu (**e**) minimum error thresholding (**f**) pcDE



(**g**) original image (**h**) contrast-enhanced image

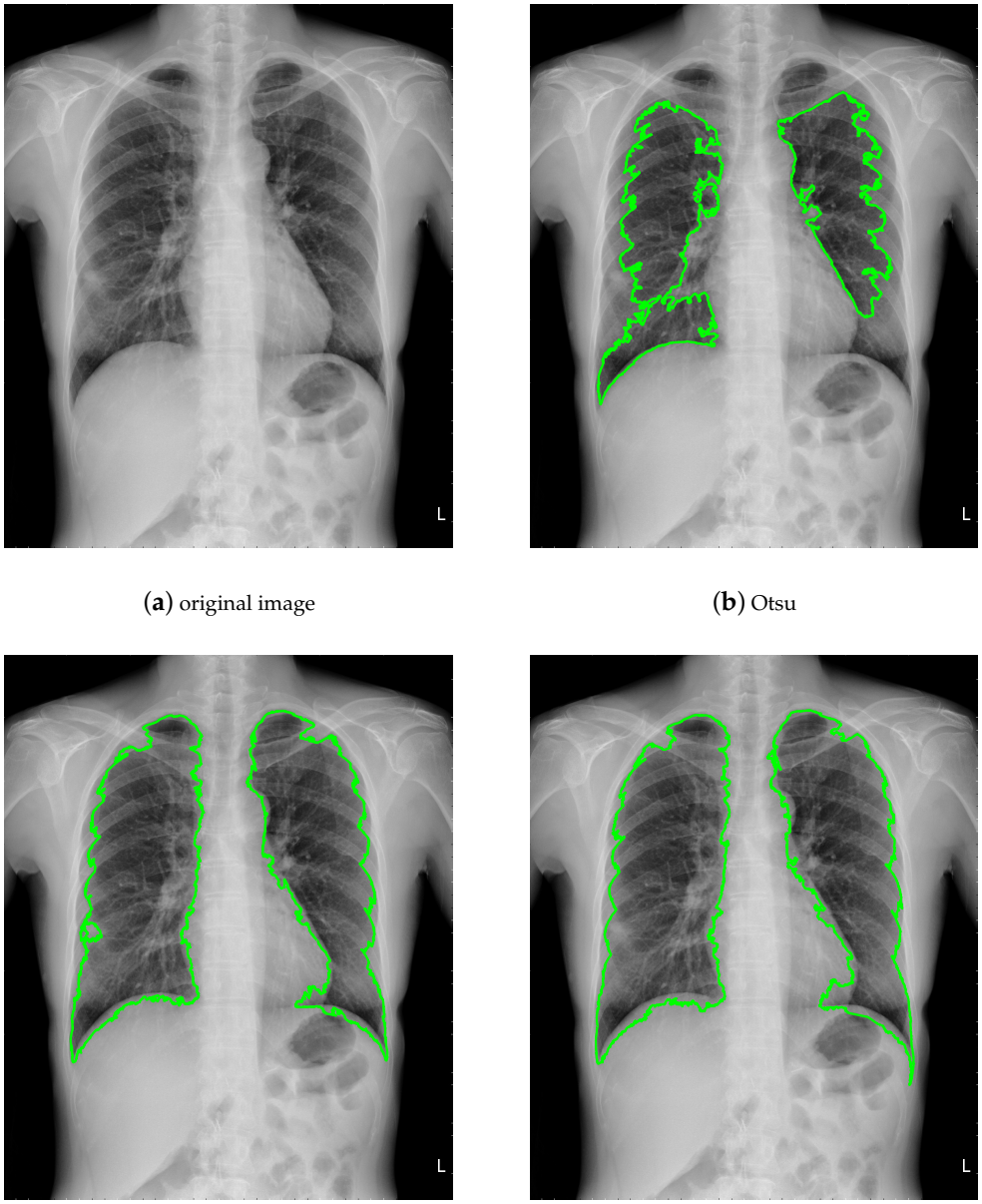
Figure 9. (**a**–**c**) image after threshold segmentation, (**d**–**f**) image after morphological processing, (**g**) original image, (**h**) contrast-enhanced image.

On the basis of the above experiments, we add Gaussian noise($\mu = 1.00 \times 10^{-5}, \sigma = 1.00 \times 10^{-1}$) to the original picture and re-run the experiment. Figure 11 shows the experimental result, in which (a)–(c) are the images after threshold segmentation, and (d) and (e) are the images after morphological processing, and (g) is the original image, and (h) is the noisy image. Figure 12 shows the contour marked image.

Several frequently-used metrics in medical image segmentation are: Dice coefficient (DICE), volumetric overlap error (VOE), relative volume difference (RVD), average symmetric surface distance (ASD), and maximum symmetric surface distance (MSD) (see [41,42]) etc. This paper uses DICE as a metric to evaluate the quality of image segmentation results. DICE is the most frequently-used method. When the segmentation effect is optimal, the value of DICE is 1. The mathematical definition is as follows:

$$DICE = \frac{2|R_{gt} \cap R_{seg}|}{|R_{gt}| + |R_{seg}|}, DICE \in [0,1] \tag{14}$$

R_{gt}: Represent the result of ground truth segmentation (Figure 13).
R_{seg}: Represent the result of algorithm segmentation.



(**a**) original image (**b**) Otsu

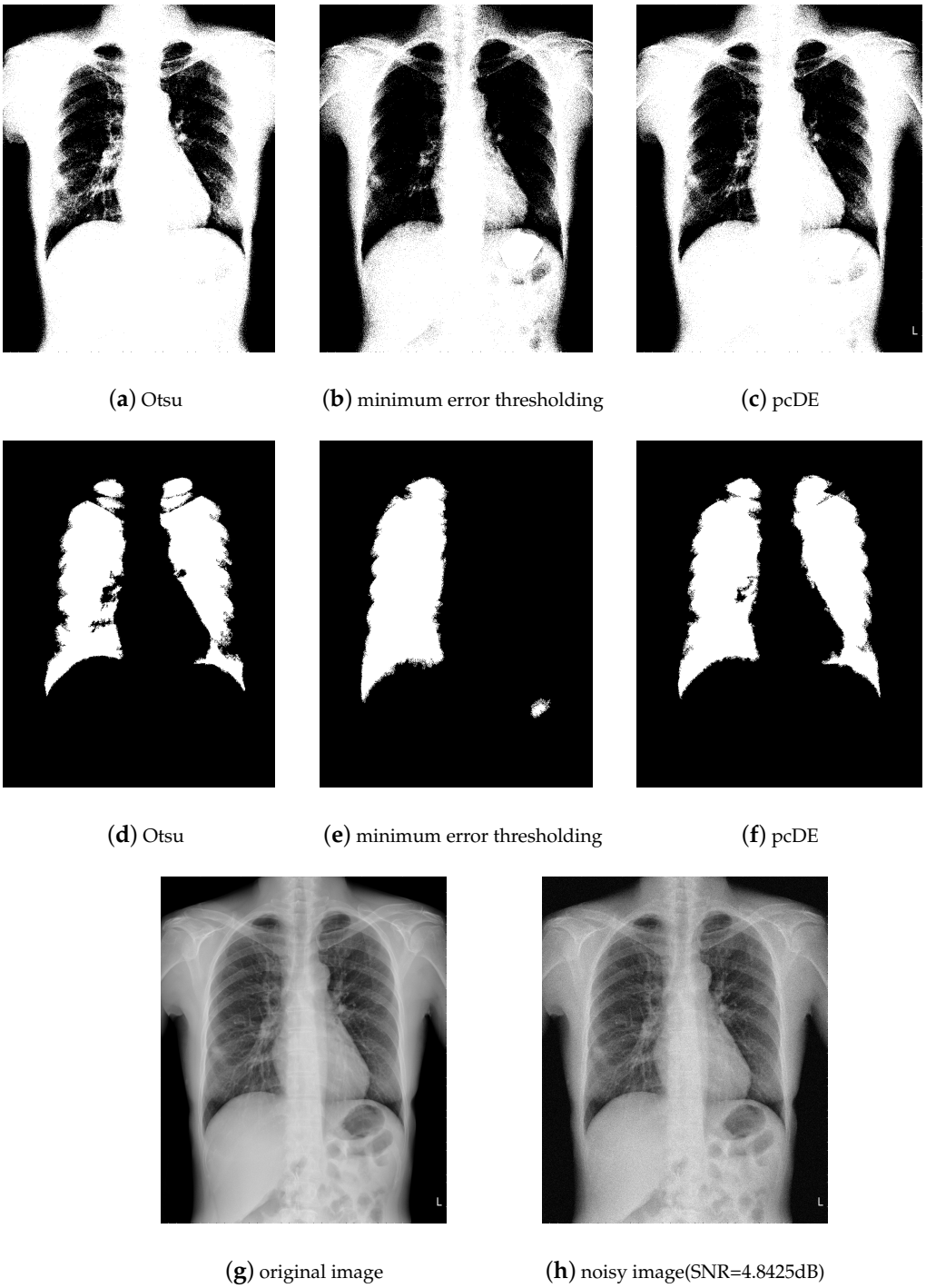
(**c**) minimum error thresholding (**d**) pcDE

Figure 10. Contour marked image.

Table 5 shows the experimental numerical results.

Table 5. DICE values obtained by different image segmentation methods.

Segmentation Method	Original Image	Noisy Image
Otsu	0.9210	0.9531
minimum error thresholding	0.9732	0.9093
pcDE	0.9738	0.9684

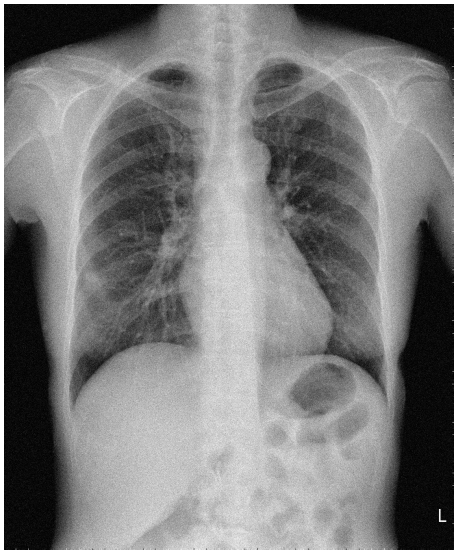


(**a**) Otsu (**b**) minimum error thresholding (**c**) pcDE

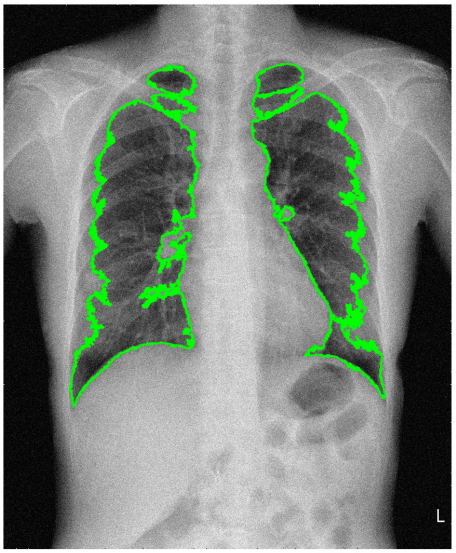
(**d**) Otsu (**e**) minimum error thresholding (**f**) pcDE

(**g**) original image (**h**) noisy image(SNR=4.8425dB)

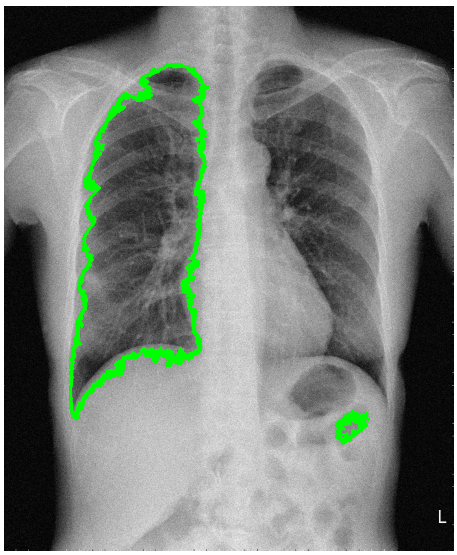
Figure 11. (**a**–**c**) image after threshold segmentation, (**d**–**f**) image after morphological processing, (**g**) original image, (**h**) noisy image.



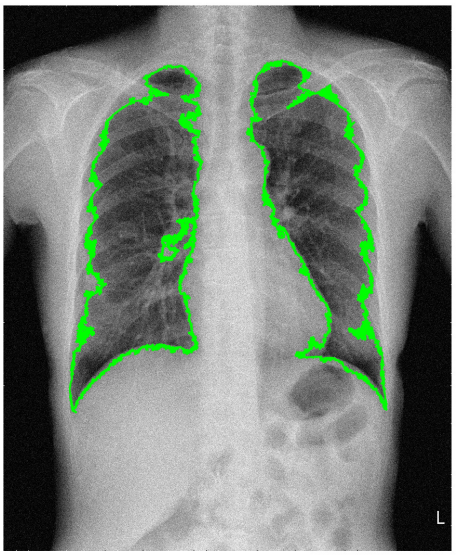
(**a**) noisy image(SNR=4.8425dB)



(**b**) Otsu



(**c**) minimum error thresholding



(**d**) pcDE

Figure 12. Contour marked image.

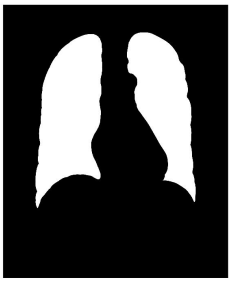


Figure 13. Ground truth segmentation.

Through analyzing the experimental results, we can see that, in the absence of noise, the minimum error thresholding and pcDE algorithm have a better segmentation effect, with Otsu's segmentation effect being weak. After that Gaussian noise is added, the pcDE algorithm can still achieve better segmentation effect, while the minimum error thresholding segmentation effect becomes unsatisfactory. Considering the two situations, the pcDE performs better than the other two methods with or without noise, and could be extensive in prospects for application.

6. Conclusions

This paper proposes a parallel compact DE algorithm. The parallel communication strategy achieves faster convergence speed and can find a better solution in a short time. Two different parallel communication strategies are proposed and compared, which are optimal elite strategy and mean elite strategy. In general, the former is better than the later one. Compared with the previous algorithms, the parallel algorithm has obvious advantages in terms of performance and stability. It is applied to image threshold segmentation. Compared with the previous method, we can see that the threshold found by the pcDE algorithm has a better segmentation effect, which has a positive impact on subsequent image processing operations. It can adapt to some weak computing power environment, such as medical equipment, micro robot and so on. In these environments, pcDE algorithm can still maintain good performance and stability, and achieve a balance between accuracy and speed. It has a good application prospect for those optimization problems that need to be solved in the environment with weak computing power.

Author Contributions: Conceptualization, S.-C.C. and J.-S.P.; Formal analysis, X.S.; Methodology, H.L.; Software, X.S.; Supervision, S.-C.C. and J.-S.P.; Writing—original draft, X.S.; Writing—review & editing, J.-S.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Natural Science Foundation of China (grant number NSF 61872085) and Natural Science Foundation of Fujian Province (grant number 2018J01638).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Storn, R.; Price, K. Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [CrossRef]
2. Meng, Z.; Pan, J.S. HARD-DE: Hierarchical archive based mutation strategy with depth information of evolution for the enhancement of differential evolution on numerical optimization. *IEEE Access* **2019**, *7*, 12832–12854. [CrossRef]
3. Chu, S.C.; Xue, X.; Pan, J.S.; Wu, X. Optimizing Ontology Alignment in Vector Space. *J. Internet Technol.* **2020**, *21*, 15–22.
4. Harik, G.R.; Lobo, F.G.; Goldberg, D.E. The compact genetic algorithm. *IEEE Trans. Evolut. Comput.* **1999**, *3*, 287–297. [CrossRef]
5. Rastegar, R.; Hariri, A. A step forward in studying the compact genetic algorithm. *Evolut. Comput.* **2006**, *14*, 277–289. [CrossRef]
6. Harik, G. *Linkage Learning via Probabilistic Modeling in the ECGA*; IlliGAL Report 99010; Illinois Genetic Algorithms Laboratory: Urbana, IL, USA, 1999.
7. Sastry, K.; Goldberg, D.E.; Johnson, D. Scalability of a hybrid extended compact genetic algorithm for ground state optimization of clusters. *Mater. Manuf. Process.* **2007**, *22*, 570–576. [CrossRef]
8. Gallagher, J.C.; Vigraham, S. A modified compact genetic algorithm for the intrinsic evolution of continuous time recurrent neural networks. In Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation, New York, NY, USA, 9–13 July 2002, pp. 163–170.
9. Mininno, E.; Neri, F.; Cupertino, F.; Naso, D. Compact differential evolution. *IEEE Trans. Evolut. Comput.* **2010**, *15*, 32–54. [CrossRef]
10. Neri, F.; Tirronen, V. Recent advances in differential evolution: A survey and experimental analysis. *Artif. Intell. Rev.* **2010**, *33*, 61–106. [CrossRef]

11. Caponio, A.; Kononova, A.V.; Neri, F. Differential evolution with scale factor local search for large scale problems. In *Computational Intelligence in Expensive Optimization Problems*; Springer: Berlin, Germany, 2010; pp. 297–323.
12. Weber, M.; Neri, F.; Tirronen, V. Distributed differential evolution with explorative–exploitative population families. *Genet. Program. Evol. Mach.* **2009**, *10*, 343. [CrossRef]
13. Otsu, N. A threshold selection method from gray-level histograms. *IEEE Trans. Syst. Man Cybern.* **1979**, *9*, 62–66. [CrossRef]
14. Kapur, J.N.; Sahoo, P.K.; Wong, A.K. A new method for gray-level picture thresholding using the entropy of the histogram. *Comput. Vis. Graph. Image Process.* **1985**, *29*, 273–285. [CrossRef]
15. Kittler, J.; Illingworth, J. Minimum error thresholding. *Pattern Recognit.* **1986**, *19*, 41–47. [CrossRef]
16. Hongfu, C.G.Z. 2-D maximum entropy method of image segmentation based on genetic algorithm. *J. Comput. Aided Des. Comput. Graph.* **2002**, *6*, 8.
17. Wu, Y.Q.; Pan, Z.; Wu, W.Y. Maximum Entropy Image Thresholding Based on Two-Dimensional Histogram Oblique Segmentation. *Pattern Recognit. Aritif. Intell.* **2009**, *6*, 162–168.
18. Pan, J.S.; Kong, L.; Sung, T.W.; Tsai, P.W.; Snášel, V. A clustering scheme for wireless sensor networks based on genetic algorithm and dominating set. *J. Internet Technol.* **2018**, *19*, 1111–1118.
19. Nguyen, T.T.; Pan, J.S.; Dao, T.K. An improved flower pollination algorithm for optimizing layouts of nodes in wireless sensor network. *IEEE Access* **2019**, *7*, 75985–75998. [CrossRef]
20. Wang, J.; Ju, C.; Gao, Y.; Sangaiah, A.K.; Kim, G.J. A PSO based energy efficient coverage control algorithm for wireless sensor networks. *Comput. Mater. Contin.* **2018**, *56*, 433–446.
21. Wang, J.; Gao, Y.; Liu, W.; Sangaiah, A.K.; Kim, H.J. An improved routing schema with special clustering using PSO algorithm for heterogeneous wireless sensor network. *Sensors* **2019**, *19*, 671. [CrossRef]
22. Wang, J.; Gao, Y.; Liu, W.; Wu, W.; Lim, S.J. An asynchronous clustering and mobile data gathering schema based on timer mechanism in wireless sensor networks. *Comput. Mater. Contin.* **2019**, *58*, 711–725. [CrossRef]
23. Chen, C.M.; Wang, K.H.; Yeh, K.H.; Xiang, B.; Wu, T.Y. Attacks and solutions on a three-party password-based authenticated key exchange protocol for wireless communications. *J. Ambient Intell. Humaniz. Comput.* **2019**, *10*, 3133–3142. [CrossRef]
24. Wu, C.I.; Kung, H.Y.; Chen, C.H.; Kuo, L.C. An intelligent slope disaster prediction and monitoring system based on WSN and ANP. *Expert Syst. Appl.* **2014**, *41*, 4554–4562. [CrossRef]
25. Chen, C.H.; Lee, C.A.; Lo, C.C. Vehicle localization and velocity estimation based on mobile phone sensing. *IEEE Access* **2016**, *4*, 803–817. [CrossRef]
26. Sun, C.; Jin, Y.; Cheng, R.; Ding, J.; Zeng, J. Surrogate-assisted cooperative swarm optimization of high-dimensional expensive problems. *IEEE Trans. Evolut. Comput.* **2017**, *21*, 644–660. [CrossRef]
27. Das, S.; Suganthan, P.N. Differential evolution: A survey of the state-of-the-art. *IEEE Trans. Evolut. Comput.* **2010**, *15*, 4–31. [CrossRef]
28. Wang, H.; Rahnamayan, S.; Sun, H.; Omran, M.G. Gaussian bare-bones differential evolution. *IEEE Trans. Cybern.* **2013**, *43*, 634–647. [CrossRef]
29. Mininno, E.; Cupertino, F.; Naso, D. Real-valued compact genetic algorithms for embedded microcontroller optimization. *IEEE Trans. Evolut. Comput.* **2008**, *12*, 203–219. [CrossRef]
30. Pan, J.S.; Lee, C.Y.; Sghaier, A.; Zeghid, M.; Xie, J. Novel systolization of subquadratic space complexity multipliers based on toeplitz matrix–vector product approach. *IEEE Trans. Very Large Scale Integr. VLSI Syst.* **2019**, *27*, 1614–1622. [CrossRef]
31. Gautschi, W. Error function and Fresnel integrals. *Handb. Math. Funct.* **1972**, *55*, 297–308.
32. Cody, W.J. Rational Chebyshev approximations for the error function. *Math. Comput.* **1969**, *23*, 631–637. [CrossRef]
33. Chang, J.F.; Roddick, J.F.; Pan, J.S.; Chu, S. A parallel particle swarm optimization algorithm with communication strategies. *Inf. Sci. Eng.* **2005**,*21*, 809–818.
34. Pan, T.S.; Dao, T.K.; Chu, S.C. A Communication Strategy for Paralleling Grey Wolf Optimizer. In *International Conference on Genetic and Evolutionary Computing*; Springer: Berlin, Germany, 2015; pp. 253–262.
35. Meng, Z.; Pan, J.S.; Tseng, K.K. PaDE: An enhanced Differential Evolution algorithm with novel control parameter adaptation schemes for numerical optimization. *Knowl. Based Syst.* **2019**, *168*, 80–99. [CrossRef]
36. Pan, J.S.; Hu, P.; Chu, S.C. Novel Parallel Heterogeneous Meta-Heuristic and Its Communication Strategies for the Prediction of Wind Power. *Processes* **2019**, *7*, 845. [CrossRef]

37. Suganthan, P.N.; Hansen, N.; Liang, J.J.; Deb, K.; Chen, Y.P.; Auger, A.; Tiwari, S. *Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization*; KanGAL Report 2005005; Kanpur Genetic Algorithms Laboratory: Kanpur, India, 2005.
38. Qin, A.K.; Huang, V.L.; Suganthan, P.N. Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE Trans. Evol. Comput.* **2008**, *13*, 398–417. [CrossRef]
39. Vesterstrom, J.; Thomsen, R. A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems. In Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No. 04TH8753), Portland, OR, USA, 19–23 June 2004; Volume 2, pp. 1980–1987.
40. Yao, X.; Liu, Y.; Lin, G. Evolutionary programming made faster. *IEEE Trans. Evolut. Comput.* **1999**, *3*, 82–102.
41. Chang, H.H.; Zhuang, A.H.; Valentino, D.J.; Chu, W.C. Performance measure characterization for evaluating neuroimage segmentation algorithms. *Neuroimage* **2009**, *47*, 122–135. [CrossRef] [PubMed]
42. Taha, A.A.; Hanbury, A. Metrics for evaluating 3D medical image segmentation: Analysis, selection, and tool. *BMC Med. Imaging* **2015**, *15*, 29. [CrossRef]

© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).