A Probabilistic-Driven Search Algorithm for solving a Class of Optimization Problems

Thong Nguyen Huu
Department of mathematics-information, HCMC University of Pedagogy
280, An Duong Vuong, Ho Chi Minh city, Viet Nam
Email: thong_nh2002@yahoo.com

Abstract: In this paper we introduce a new numerical optimization technique, a Probabilistic-Driven Search Algorithm. This algorithm has the following characteristics: 1) In each iteration of loop, the algorithm just changes the value of k (k<n, n is the number of variables of the solution of the problem) variables to find a new solution better than the current one; 2) In each variable of the solution of the problem, the algorithm finds the value of each digit from left digit to right digit; 3) The algorithm uses probability to control the implementation of the algorithm to perform two tasks mentioned above in a very special connection. We build two applications of Probabilistic-Driven Search algorithm, PDS algorithm 1 for solving single-objective optimization problems and PDS algorithm 2 for solving multi-objective optimization problems. We test this approach by implementing the algorithms on some benchmark optimization problems and we find very good and stable results.

Key words: Optimization, Probability, Algorithm.

1. Introduction

In the field of evolutionary computation, there are many popular approaches for solving optimization problems, such as genetic algorithm, ant algorithm or particle swarm optimization. We have following remarks:

- Suppose that the solution of optimization problems has n variables. These approaches often simultaneously change values of n variables on each iteration in search of optimal solutions. Meanwhile in some cases, if we only need to change the value of k ($1 \le k < n$) variables then it has the ability to find a better solution than the current one.
- Suppose that every variable of the solution of optimization problems has m digits. The role of left digit is more important than the role of right digit for assessing values of objective functions, but evolutionary algorithms remove the difference of the role of the digits.

The evolutionary algorithms are classified into types of metaheuristic algorithms. The metaheuristic algorithm uses a loop and in each iteration of loop it can try to improve the convergence of the solution. In the approach of this paper, we build the kind of metaheuristic algorithms and these algorithms use probability to overcome two problems mentioned above.

In this paper, we build a Search Technique and compute its complexity. The technique has two following characters:

- In each iteration of loop, the technique selects randomly k ($1 \le k < n$) variables of the current solution x to transform x into a new solution y.

- The technique finds the values of digits from left digits to right digits of variables of a solution step by step.

We build a Changing Procedure that uses probabilities to drive the changes of every solution according to two conditions above for finding a better solution than the current one. We build two applications of the Changing Procedure, Probabilistic-Driven Search (PDS) algorithm 1 for solving single-objective optimization problems and PDS algorithm 2 for solving multi-objective optimization problems.

2. The model of optimization problems

We consider two models of optimization problems, a model of single-objective optimization problem and a model of multi-objective optimization problem.

2.1. The model of single-objective optimization problem

We consider a model of single-objective optimization problem as follows:

$$\begin{aligned} &\textit{Minimize} \quad f(x) \\ &\textit{subject to} \quad g_j(x) \le 0 \quad (j = 1, \ldots, r) \\ &\textit{where} \quad x = (x_i), a_i \le x_i \le b_i \ (a_i, b_i \in R, 1 \le i \le n). \end{aligned}$$

where g_j ($1 \le j \le r$) are real valued functions.

2.2. The model of multi-objective optimization problem

A general multi-objective optimization problem can be described as follows:

$$\begin{aligned} &\textit{Minimize} \quad f_k(x) \quad (k = 1, \ldots, s) \\ &\textit{subject to} \quad g_j(x) \le 0 \quad (j = 1, \ldots, r) \\ &\textit{where} \quad x = (x_i), a_i \le x_i \le b_i \ (a_i, b_i \in R, 1 \le i \le n). \end{aligned}$$

where g_j ($1 \le j \le r$) are real valued functions.

A solution x is said to dominate another solution y if and only if

$$\begin{aligned} &f_k(x) \le f_k(y), \ \forall k \in \{1, ..., s\} \\ &\textit{and } f_i(x) < f_i(y) \textit{ for at least one } i \in \{1, ..., s\} \end{aligned}$$

A solution that is not dominated by any other solutions is called a Pareto optimal solution. Let S be a set of Pareto optimal solutions, S is called Pareto optimal set. The set of objective function values corresponding to the variables of S is called Pareto front.

3. The Changing Procedure

We consider a class of optimization problems having the character as follows: there exists a fixed number k ($1 \le k < n$) that is independent of the size n of the problem such that if we only need to change values of k variables then it has the ability to find a better solution than the current one, let us call it O_k.

3.1. The Search Technique and its complexity

We suppose a solution x of an optimization problem has n variables, and every variable x_i ($1 \le i \le n$) has m digits that are listed from left to right $x_{i1}, x_{i2}, \ldots, x_{im}$ (x_{ij} is an integer, $0 \le x_{ij} \le 9$, $1 \le j \le m$). Let L be a number of iterations such that The Search Technique has an ability to find correct values of j-th digits of n variables of a solution. The Search Technique that finds an

optimal solution of the single-objective optimization problem or a Pareto optimal solution of the multi-objective optimization problem is described with general steps as follows:

S1. Select randomly a feasible solution x;

S2. j←1 (determine j-th digit);

S3. i←1 (start counting variable of the loop);

S4. y←x;

S5. Select randomly k variables of solution y and change randomly values of j-th digits of these k variables;

S6. If (y is better than x) then x←y;

S7. If (i<L) then i← i+1 and return S4;

S8. If (j<m) then j← j+1 and return S3;

S9. The end of the Search Technique;

In S6, the condition (y is better than x), it means that (f(y)<f(x)) for the single-objective optimization problem, or (y dominates x) for the multi-objective optimization problem.

The Search Technique finds values of each digit from left digit to right digit one by one. Consider j-th digit, on each iteration the Search Technique selects randomly k variables, changes randomly values of j-th digits of these k variables to find a better solution than the current one. Let A be the event such that the technique can find correct values of j-th digits of k variables on each iteration. The probability of A is

$$p_A = \Pr(A) = \left(\frac{k}{n} \times \frac{1}{10}\right)^k = \frac{k^k}{10^k n^k}$$

Let X be a number of occurrences of A in N iterations, X has the binomial distribution:

$$\Pr(X = x) = C_N^x \left(p_A\right)^x \left(1 - p_A\right)^{N-x} \quad (x = 0,1,...,N)$$

Because p_A is very small and N is very large, so the binomial distribution approximates to the Poisson distribution as follows

$$\Pr(X = x) \approx \frac{\lambda^x}{x!} e^{-\lambda}$$

where the parameter $\lambda = Np_A$ and expectation $E(X) = \lambda$.

Because the solution has n variables, we select an average number of iterations such that the event A occurs at least n/k times. We have

$$E(X) > \frac{n}{k} \Rightarrow N.p_A > \frac{n}{k} \Rightarrow N > \frac{n}{k.p_A} = \frac{10^k n^{k+1}}{k^{k+1}}$$

Because every variable has m digits, so the average number of iterations for finding correct values of a solution the first time is

$$m\left(\frac{10^k}{k^{k+1}}\right) n^{k+1} = \left(\frac{10^k m}{k^{k+1}}\right) n^{k+1}$$

On each iteration, the technique performs k jobs with complexity O(1). Because k is a fixed number, so the complexity of the Search Technique is $O(n^{k+1})$.

3.2. The absorbing Markov chain of the Search Technique

Without loss of generality we suppose that the solution has n variables, every variable has m=5 digits. Let E_0 be the starting state of a solution that is selected randomly, E_i (i=1,2,…,5) be the state of the solution having n variables with correct values that are found for digits from 1-th digit to i-th digit ($1 \le i \le 5$). We have (X_n; n=0,1,2,…) is a Markov chain with states {E_0, E_1, E_2, E_3, E_4, E_5}. Let p be the probability of event in which i-th digits of n variables have correct values. According to section 3.1, we have

$$p = \frac{k^{k+1}}{10^k n^{k+1}}$$

Set q=1-p, the transition matrix is then

$$P = \left(p_{ij}\right)_{i,\, j=\overline{0,5}} = \begin{bmatrix} q & p & 0 & 0 & 0 & 0 \\ 0 & q & p & 0 & 0 & 0 \\ 0 & 0 & q & p & 0 & 0 \\ 0 & 0 & 0 & q & p & 0 \\ 0 & 0 & 0 & 0 & q & p \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

The states E_i ($0 \le i \le 4$) are transient states, and the state E_5 is an absorbing state. The Markov chain is an absorbing Markov chain and its model as follows:

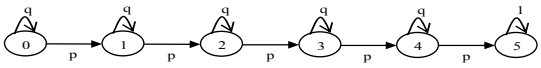

Fig. 1 The model of absorbing Markov chain of the Search Technique

Absorption Probabilities: Let u_{i5} ($0 \le i \le 4$) be the probability that the absorbing chain will be absorbed in the absorbing state E_5 if it starts in the transient state E_i ($0 \le i \le 4$). If we compute u_{i5} in term of the possibilities on the outcome of the first step, then we have the equations

$$u_{i5} = p_{i5} + \sum_{j=0}^{4} p_{ij} u_{j5} \quad (0 \le i \le 4)$$

$$\Rightarrow \; u_{05} = u_{15} = u_{25} = u_{35} = u_{45} = 1$$

Here the result tells us that, starting from state i ($0 \le i \le 4$), there is probability 1 of absorption in state E_5.

3.3. The Changing Procedure

We built the Changing Procedure that transforms a solution x into a new solution y by using two following techniques:

- In each iteration, the procedure selects randomly k ($1 \le k < n$) variables of the current solution x to change their values and transform x into a new solution y.
- We increase speed of the convergence for finding the values of digits from left digits to right digits of variables of a solution by using probabilities to drive the process of changes of values of digits.

3.3.1. Probabilities of changes

We suppose that every variable x_i ($1 \le i \le n$) of a solution of the problem (single-objective or multi-objective optimization problem) has m digits that are listed from left to right x_{i1}, x_{i2},…, x_{im} (x_{ij} is an integer and $0 \le x_{ij} \le 9$, $1 \le j \le m$).

We consider j-digit of a variable x_i. We suppose the values of left digits x_{ik} (k=1, 2, …, j-1) are correct, we have to fix the values of these left digits and change the value of j-th digit to find

a correct value of j-th digit. Because the value of j-digit is changed, the values of digits x_{ik} (k=j+1,…, m) can be changed or can not be changed. Let A_j be an event such that the j-digit is selected to change its value (1≤j≤m). We consider an following event to find a correct value of j-digit:

$$\overline{A_1}\,\overline{A_2}\ldots\overline{A_{j-1}}A_j \quad (1 \le j \le m)$$

We have following remarks:

Remark 1: The role of left digit is more important than the role of right digit of a variable for assessing values of objective functions. Hence we should find the values of digits from left digits to right digits one by one. We consider events

$$B_1B_2B_3\ldots B_m$$

where $B_j = A_j \text{ or } \overline{A_j} \quad (1 \le j \le m)$

We classify these events according to typical events in the table below:

Event	Frequency	Probability
$A_1B_2B_3\ldots B_m$	2^{m-1}	$\frac{2^{m-1}}{2^m}=\frac{1}{2}$
$\overline{A_1}A_2B_3\ldots B_m$	2^{m-2}	$\frac{2^{m-2}}{2^m}=\frac{1}{2^2}$
$\overline{A_1}\,\overline{A_2}A_3B_4\ldots B_m$	2^{m-3}	$\frac{2^{m-3}}{2^m}=\frac{1}{2^3}$
⋮	⋮	⋮
$\overline{A_1}\,\overline{A_2}\ldots\overline{A_{m-1}}A_m$	1	$\frac{1}{2^m}$

Table 1. Frequencies and probabilities of events

The probability of selecting j-digit from m digits is

$$\frac{1}{2^j} \quad (1 \le j \le m)$$

We have a set of probabilities for selecting digits as follows:

$$\left(\frac{1}{2},\frac{1}{4},\ldots,\frac{1}{2^m}\right)$$

It means that number of searches for correct values of left digits is more than number of searches for correct values of the right digits.

Remark 2: Let p_j be the probability of the event A_j (1≤j≤m). In some iteration we have a below event occurring:

$$\overline{A_1}\,\overline{A_2}\ldots\overline{A_{j-1}}A_j \quad (1 \le j \le m) \Rightarrow \begin{cases} \Pr(\overline{A_1}) = \ldots = \Pr(\overline{A_{j-1}}) = 0 \\ \Pr(A_j) = 1 \\ \Pr(\overline{A_{j+1}}) = \ldots = \Pr(\overline{A_m}) = \frac{1}{2} \end{cases}$$

Hence we have probabilities of changes after selecting j-digit as follows:

$$p_1 = 0, \ldots, p_{j-1} = 0, p_j = 1, p_{j+1} = \frac{1}{2}, \ldots, p_m = \frac{1}{2}$$

Remark 3: According to papers [15], we consider two digits a_{j-1} and a_j ($2 \le j \le m$). Let r_1, r_2 and r_3 be probabilities of events below:

- r_1: probability of choosing a random integer number between 0 and 9 for j-th digit.
- r_2: probability of j-th digit incremented by one or a certain value (+1,…,+5).
- r_3: probability of j-th digit decremented by one or a certain value (-1,…,-5).

If the value of a_{j-1} is correct, we have the probability so that a_j can find its correct value as follows:

$$r_1 \frac{1}{10} + r_2 \frac{1}{100} + r_3 \frac{1}{100}$$

Because of $r_1+r_2+r_3=1$, this probability is maximum if $r_1=1$, $r_2=r_3=0$.

If the value of a_{j-1} is not correct, we have the probability so that a_{j-1} and a_j can find their correct values as follows:

$$r_1 0 + r_2 \frac{1}{100} + r_3 \frac{1}{100}$$

Because of $r_1+r_2+r_3=1$, this probability is maximum if $r_1=0$, $r_2=r_3=0.5$

We have the average probabilities r_1, r_2 and r_3 of both two cases as follows:

$$r_1=0.5, r_2=r_3=0.25$$

Probabilities of the other cases for finding correct values of three, four digits side by side are very small; hence we do not consider these cases.

In summary we have three sets of probabilities:

- Select j-th digit ($1 \le j \le m$) of m digits corresponding to the probabilities:

$$\left(\frac{1}{2}, \frac{1}{4}, \ldots, \frac{1}{2^m}\right)$$

- After selecting the j-th digit we set the probabilities for change:

$$p_1 = 0, \ldots, p_{j-1} = 0, p_j = 1, p_{j+1} = \frac{1}{2}, \ldots, p_m = \frac{1}{2}$$

- When a digit is allowed to change, we have the probabilities for choosing values as follows: $r_1=0.5$, $r_2=r_3=0.25$

In next section we use three sets of probabilities above to build the changing procedure that transforms a solution x into a new solution y.

3.3.2. The changing procedure

Without loss of generality we suppose that a solution of the problem has n variables, every variable has m digits, one digit is displayed to the left of the decimal point and m-1 digits are displayed to the right of the decimal point. We use a function *random(num)* that returns a random number between 0 and (num-1). The Changing Procedure changing values of a solution x via probability to create a new solution y is described as follows:

The Changing Procedure

Input: a solution x

Output: a new solution y

S1. y←x;

S2. Select j-th digit according to probabilities

$$\left(\frac{1}{2},\frac{1}{4},\ldots,\frac{1}{2^m}\right)$$

S3. Set

$$p_1 = 0, \ldots, p_{j-1} = 0, p_j = 1, p_{j+1} = \frac{1}{2}, \ldots, p_m = \frac{1}{2}$$

S4. Select randomly k variables of solution y and call these variables y_i ($1 \le i \le k$). The technique for changing values of these variables is described as follows:

For i=1 to k do
Begin_1
y_i=0;
For j=1 to m do
Begin_2
If (a random event with probability p_j occurs) then
If (a random event with probability r_1 occurs) then $y_i=y_i+random(10)*10^{1-j}$;
Else
If (a random event with probability r_2 occurs) then $y_i= y_i+(x_{ij} +1)*10^{1-j}$;
Else $y_i= b*y_i+(x_{ij} -1)*10^{1-j}$;
Else $y_i= y_i +x_{ij}*10^{1-j}$;
End_2
If ($y_i<a_i$) then $y_i=a_i$; If ($y_i>b_i$) then $y_i=b_i$;
End_1;

S5. Return y; S6. The end of Changing Procedure;

The Changing Procedure has the following characteristics:

- The central idea of the Changing Procedure is that variables of the solution x are separated into discrete digits, and then they are changed with the guide of probabilities and combined to a new solution y.
- Because the role of left digit is more important than the role of right digit for assessing values of objective functions. The Procedure finds values of each digit from left digits to right digits of every variable with the guide of probabilities and the newly-found values may be better than the current ones (according to probabilities).
- The parameter k: In practice, we do not know the true value of k for each problem. According to statistics of many experiments, the best thing is to use k in the ratio as follows:
  - $n \ge 5$, k is an integer chosen randomly from 1 to 5.
  - n>6, k is chosen as follows:
    - k is an integer chosen randomly from 1 to n / 2 with probability 20% (find the best peak of a hill to prepare to climb).
    - k is an integer chosen randomly from 1 to 4 with probability 80% (climbing the hill or optimizing the solution).

We have two applications of the Changing Procedure, PDS algorithm 1 for solving single-objective optimization problems and PDS algorithm 2 for solving multi-objective optimization problems.

4. PDS algorithm 1 for solving single-objective optimization problems

4.1. General steps of the PDS algorithm 1

We apply the Changing Procedure to build PDS algorithm 1 for solving single-objective optimization problems. The PDS algorithm 1 uses only one solution in each execution of the algorithm 1, so the starting solution affects the rate of convergence of the algorithm. We improve the speed of convergence by implementing the algorithm 1 in two phases. PDS algorithm 1 is described with general steps as follows:

Phase 1: Let M1 be a number of solutions that are generated randomly, and M2 be a number of iterations of each solution. In Phase 1, generate randomly M1 of solutions and each solution is optimized by M2 of iterations. Then choose a best solution for phase 2.

S1. Select a random feasible solution x; S2. L1←1;
S3. Select a random feasible solution y; S4. L2←1;
S5. Use the Changing Procedure to transform the solution y into a new solution z;
S6. If the solution z is not feasible then return S5;
S7. If f(z)<= f(y) then y←z;
S8. If L2<M2 then L2←L2+1 and return S5;
S9. If f(y)<=f(x) then x←y;
S10. If L1<M1 then L1←L1+1 and return S3;
S11. Return the solution x for phase 2;

Phase 2: Optimize the solution of Phase 1 to find an optimal solution.

S12. Use the Changing Procedure to transform the solution x into a new solution y;
S13. If y is not a feasible solution then return S12;
S14. If f(y)<=f(x) then x←y;
S15. If the condition of stop is not satisfied then return S12;
S16. The end of PDS algorithm 1;

Stop condition is the desired value of the objective function or after a number of certain iterations.

Example A: Michalewicz's function 12 [17]

$$f(x) = -\sum_{i=1}^{n} \sin(x_i) \left[\sin\left(\frac{i.x_i^2}{\pi} \right) \right]^{2m}, \; m = 10,$$
$$0 \le x_i \le \pi \;\; (i = 1, 2, ..., n)$$

There are n! local optimums in the domain of the problem. The best known values are listed below:

- n=2: The global optimum is f*≈-1.801
- n=5: The global optimum is f*≈-4.6877
- n=10: The global optimum is f*≈-9.66

To solve this problem, Yang [17] used the standard version of genetic algorithms, particle swarm optimization and a new Bat Algorithm. The following results show the number of objective function evaluations in the form of mean ± the standard deviation (success rate of the algorithm in finding the global optima). The simulations have been carried out using Matlab on a standard 3GHz desktop computer with the running time less than 5 seconds.

Function	Genetic algorithm	Particle swarm optimization	Bat Algorithm
Michalewicz's (n=16)	89325 ± 7914(95%)	6922 ± 537(98%)	4752 ± 753(100%)

Table 2. The experimental results of [17]

Yang chose the accuracy nearly the value of the objective function of an optimal an optimal solution, and he counted the number of times of algorithms to achieve the accuracy. But we choose the way to compare two algorithms by comparing their optimal values and best solutions (mean and standard deviation). We choose m=7. Because the digit x_i only affects the term containing x_i, so we choose k = 1. Applying PDS algorithm 1 to solving Michalewicz's function 12 with the average number of iterations is theoretically:

$$\left(\frac{10^k m}{k^{k+1}}\right) n^{k+1} = \left(\frac{10^1 7}{1^{1+1}}\right) n^{1+1} = 70n^2$$

Using Borland C++ for Dos on a standard 2GHz desktop computer. Each problem is tested 30 times. The following results list some of the optimal solutions found by PDS algorithm 1:

- n=2 (average running time: 2 seconds): x=(2.202908, 1.570798); f(x)=-1.8013018986
- n=5 (average running time: 2 seconds): x=(2.202908, 1.570798, 1.284993, 1.923061, 1.720472); f(x)=-4.6876560154
- n=10 (average running time: 3 seconds): x=(2.202908, 1.570798, 1.284993, 1.923061, 1.720472, 1.570798, 1.454416, 1.756089, 1.655719, 1.570798); f(x)=-9.6601492066
- n=16 (average running time: 4 seconds): x=(2.202908, 1.570798, 1.284993, 1.923061, 1.720472, 1.570798, 1.454416, 1.756089, 1.655719, 1.570798, 1.497731, 1.696618, 1.630078, 1.570798, 1.517548, 1.666066); f(x)=-15.6418619840
- n=20 (average running time: 20 seconds)**:** x=(2.202908, 1.570798, 1.284993, 1.923061, 1.720472, 1.570798, 1.454416, 1.756089, 1.655719, 1.570798, 1.497731, 1.696618, 1.630078, 1.570798, 1.517548, 1.666066, 1.616331, 1.570798, 1.528909, 1.647458); f(x)=-19.6370106044
- n=50 (average running time: 40 seconds)**:** x=(2.202908, 1.570798, 1.284993, 1.923061, 1.720472, 1.570798, 1.454416, 1.756089, 1.655719, 1.570798, 1.497731, 1.696618, 1.630078, 1.570798, 1.517548, 1.666066, 1.616331, 1.570798, 1.528909, 1.490201, 1.607759, 1.834277, 1.670094, 1.503931, 1.601904, 1.444935, 1.541437, 1.513663, 1.597650, 1.345169, 1.438562, 1.520921, 1.685598, 1.570798, 1.548197, 1.613840, 1.591883, 1.570798, 1.466657, 1.531026, 1.512240, 1.643895, 1.476654, 1.606101, 1.443468, 1.570798, 1.620154, 1.537726, 1.521954, 1.570798); f(x)=-49.5135196111
- n=100 (average running time: 60 seconds)**:** x=(2.202908, 1.570798, 1.284993, 1.923061, 1.720472, 2.027482, 1.454416, 1.756089, 1.655717, 1.570798, 2.009253, 1.696618, 1.630078, 1.570798, 1.517548, 1.666066, 1.616331, 1.570798, 1.528909, 1.490201, 1.607759, 1.570798, 1.536274, 1.503931, 1.473548, 1.570798, 1.655761, 1.513663, 1.597650, 1.570798, 1.545256, 1.520921, 1.594419, 1.570798, 1.454282, 1.613840, 1.591883, 1.570798, 1.466657, 1.684488, 1.663821, 1.494129, 1.624668, 1.606101, 1.588155, 1.637669, 1.553998, 1.397633, 1.521954, 1.632420, 1.491814, 1.600720, 1.643222, 1.449784, 1.613069, 1.542494, 1.584516, 1.570798, 1.446039, 1.544397,

1.583621, 1.620677, 1.558282, 1.546060, 1.676020, 1.570798, 1.461495, 1.593730, 1.582140, 1.570798, 1.559697, 1.634938, 1.623701, 1.612693, 1.720722, 1.549992, 1.735555, 1.610572, 1.600345, 1.551039, 1.618553, 1.608654, 1.561307, 1.589388, 1.580011, 1.570798, 1.487333, 1.588548, 1.579598, 1.639125, 1.596480, 1.553631, 1.612458, 1.603871, 1.595407, 1.554350, 1.546317, 1.538407, 1.625383, 1.555011); f(x)=-99.2965315337

Example B: G2 problem [18].

Maximize

$$f(x)=\left|\frac{\sum_{i=1}^{n}\cos^4(x_i)-2\prod_{i=1}^{n}\cos^2(x_i)}{\sqrt{\sum_{i=1}^{n}ix_i^2}}\right|$$

subject to

$$g_1(x)=0.75-\prod_{i=1}^{n}x_i\le 0;\quad g_2(x)=\sum_{i=1}^{n}x_i-7.5n\le 0;\quad 0\le x_i\le 10\quad (i=1,\ldots,n)$$

The best known value is f (x) =0.803619 for n = 20.

The problem G2 has many local optimums and it is not of class O_k. Since we do not know the value of k, therefore we choose k which is a random number from 1 to 10 in each of iteration. We use Borland C++ for Dos on a standard 2GHz desktop computer. The problem is tested 30 times.

n=20 (Running time: 200 seconds): x=(3.162490, 3.128278, 3.094777, 3.061452, 3.027940, 2.993882, 2.958670, 2.921863, 0.494812, 0.488386, 0.482356, 0.476655, 0.471323, 0.466238, 0.461416, 0.456845, 0.452439, 0.448258, 0.444206, 0.440348)

g_1(x)=-0.0000000002; g_2(x)=-120.0673660000; f(x)=0.8036191026;

4.2. The model of nonlinear Equations System

To cite a few instances of single-objective optimization problems, we consider system of equations and apply PDS algorithm 1 to solving nonlinear equations system. A general nonlinear equations system can be described as follows:

$$\begin{cases} f_1(x_1,x_2,...,x_n)=0 \\ f_2(x_1,x_2,...,x_n)=0 \\ \quad\vdots \\ f_m(x_1,x_2,...,x_n)=0 \end{cases}$$

$$a_i\le x_i\le b_i,\ a_i,b_i\in R\quad (i=1,...,n)$$

where f_j ($1\le j\le m$) are nonlinear functions.

4.3. Popular approaches for solving nonlinear Equations System

There are several standard known techniques to solve nonlinear equations system. Probably the most popular techniques are the Newton-type techniques sush as: trust-region method [2], Broyden method [1], secant method [4], Halley method [14].

In the field of evolutionary computation, recently Grosan et al. [7] have transformed the system of equations into a multi-objective optimization problem as follows:

$$\begin{aligned} &Minimize \quad abs(f_1(x_1, x_2, ..., x_n)) \\ &Minimize \quad abs(f_2(x_1, x_2, ..., x_n)) \\ &\qquad \vdots \\ &Minimize \quad abs(f_n(x_1, x_2, ..., x_n)) \\ &a_i \le x_i \le b_i, a_i, b_i \in R \quad (i = 1, ..., n) \end{aligned}$$

and they use an evolutionary computation technique for solving this multi-objective optimization problem. It is to be noted that solutions found by this approach are Pareto optimal solutions.

4.4. PDS algorithm 1 for solving Equations System

Because there are many equality constraints, the system of equations usually has no solution x such that $f_j(x)=0$ ($1 \le j \le m$). Thus we find an approximate solution of simultaneous equations such that $|f_j(x)|<\varepsilon$ ($1 \le j \le m$) with ε is an arbitrary small positive number. In order to do so, we transform the system of equations into a single-objective optimization problem as follows:

$$\begin{aligned} &Minimize \ \varepsilon(x) = \max\{|f_1(x)|, |f_2(x)|, \ldots, |f_n(x)|\} \\ &x = (x_1, x_2, \ldots, x_n), \\ &a_i \le x_i \le b_i, \ a_i, b_i \in R \quad (i = 1, ..., n) \end{aligned}$$

We use PDS algorithm 1 to solve the single-object optimization problem. In next sections, we use two examples and six benchmark problems for nonlinear equations systems to examine the PDS algorithm 1. Using PC, Celeron CPU 2.20GHz, Borland C++ 3.1. We performed 30 independent runs for each problem. The results for all test problems are reported in Tables.

4.5. Two examples

We consider two examples used by Effati and Nazemi [6], it is to be noted that the techniques of Effati and Nazemi are only applied for two equations systems. PDS algorithm 1 is compared with Newton's method, the Secant method, Broyden's method, Effati's method and evolutionary approach [7]. Only systems of two equations were considered by Effati and Nazemi.

Example 1:

$$\begin{cases} f_1(x_1, x_2) = \cos(2x_1) - \cos(2x_2) - 0.4 = 0 \\ f_2(x_1, x_2) = 2(x_2 - x_1) + \sin(2x_2) - \sin(2x_1) - 1.2 = 0 \end{cases}$$

Method	Solution	Functions values
Newton	(0.15, 0.49)	(-0.00168, 0.01497)
Secant	(0.15, 0.49)	(-0.00168, 0.01497)
Broyden	(0.15, 0.49)	(-0.00168, 0.01497)
Effati	(0.1575, 0.4970)	(0.005455, 0.00739)
Evolutionary approach [7]	(0.15772, 0.49458)	(0.001264, 0.000969)
PDS Algorithm 1	(0.156520, 0.493376)	(-0.0000005815, -0.0000008892)

Table 3. Comparison of results for example 1.

	ε(x)
Min	0.0000008892
Max	0.0000008892
Average	0.0000008892
Median	0.0000008892
Standard deviation	0

Table 4. Statistical table of 30 independent runs of PDS algorithm 1 for example 1.

Example 2:

$$\begin{cases} f_1(x_1, x_2) = e^{x_1} + x_1 x_2 - 1 = 0 \\ f_2(x_1, x_2) = \sin(x_1 x_2) + x_1 + x_2 - 1 = 0 \end{cases}$$

Method	Solution	Functions values
Effati	(0.0096, 0.9976)	(0.019223, 0.016776)
Evolutionary approach [7]	(-0.00138, 1.0027)	(-0.00276,-0.0000637)
PDS Algorithm 1	(0.0, 1.0)	(0, 0)

Table 5. Comparison of results for example 2.

	ε(x)
Min	0
Max	0
Average	0
Median	0
Standard deviation	0

Table 6. Statistical table of 30 independent runs of PDS algorithm 1 for example 2.

Two examples 1 and 2 are the problems of small size and have no local optimal points. The algorithm 1 should be easy to find a global optimal solution and has the standard deviation of 0.

4.6. Six benchmark problems

Six problems of nonlinear equations systems considered in the following sections are as follows: Interval Arithmetic, Neurophysiology Application, Chemical Equilibrium Application, Kinematic kin2, Combustion Application and Economics Modeling Application.

4.6.1. Problem 1: Interval Arithmetic Benchmark

The Interval Arithmetic Benchmark [9] [11] [8] is described as follows:

$$\begin{cases} f_1(x) = x_1 - 0.25428722 - 0.18324757x_4x_3x_9 = 0 \\ f_2(x) = x_2 - 0.37842197 - 0.16275449x_1x_{10}x_6 = 0 \\ f_3(x) = x_3 - 0.27162577 - 0.16955071x_1x_2x_{10} = 0 \\ f_4(x) = x_4 - 0.19807914 - 0.15585316x_7x_1x_6 = 0 \\ f_5(x) = x_5 - 0.44166728 - 0.19950920x_7x_6x_3 = 0 \\ f_6(x) = x_6 - 0.14654113 - 0.18922793x_8x_5x_{10} = 0 \\ f_7(x) = x_7 - 0.42937161 - 0.21180486x_2x_5x_8 = 0 \\ f_8(x) = x_8 - 0.07056438 - 0.17981208x_1x_7x_6 = 0 \\ f_9(x) = x_9 - 0.34504906 - 0.19612740x_{10}x_6x_8 = 0 \\ f_{10}(x) = x_{10} - 0.42651102 - 0.21466544x_4x_8x_1 = 0 \end{cases}$$

$$-2 \le x_i \le 2 \ (i = 1,...,10)$$

There are 8 solutions of paper [7] for the Interval Aithmetic Benchmark; these solutions are Pareto optimal solutions found by evolutionary computation approach. We choose the solution 1 that is displayed below to compare with the solutions found by PDS algorithm 1.

	Evolutionary approach [7]	**PDS algorithn 1**		**Evolutionary approach [7]**	**PDS algorithn 1**
$\mathbf{x_1}$	0.046491	0.257833	$\mathbf{f_1(x)}$	-0.2077959241	-0.0000003959
$\mathbf{x_2}$	0.101357	0.381097	$\mathbf{f_2(x)}$	-0.2769798847	-0.0000001502
$\mathbf{x_3}$	0.084058	0.278745	$\mathbf{f_3(x)}$	-0.1876863213	0.0000000010
$\mathbf{x_4}$	-0.138846	0.200669	$\mathbf{f_4(x)}$	-0.3367887114	0.0000000365
$\mathbf{x_5}$	0.494391	0.445251	$\mathbf{f_5(x)}$	0.0530391321	-0.0000004290
$\mathbf{x_6}$	-0.076069	0.149184	$\mathbf{f_6(x)}$	-0.2223730535	0.0000000763
$\mathbf{x_7}$	0.247582	0.432010	$\mathbf{f_7(x)}$	-0.1816084752	0.0000002966
$\mathbf{x_8}$	-0.017075	0.073403	$\mathbf{f_8(x)}$	-0.0874896386	0.0000002231
$\mathbf{x_9}$	0.000367	0.345967	$\mathbf{f_9(x)}$	-0.3447200367	0.0000001704
$\mathbf{x_{10}}$	0.148112	0.427326	$\mathbf{f_{10}(x)}$	-0.2784227490	-0.0000002774
			ε(x)		0.0000004290

Table 7. Comparison of results for Interval Aithmetic Benchmark.

	ε(x)
Min	0.0000004290
Max	0.0000004290
Average	0.0000004290
Median	0.0000004290
Standard deviation	0

Table 8. Statistical table of 30 independent runs of PDS algorithm 1 for Interval Arithmetic Benchmark.

Interval Benchmark Aithmetic has one global optimal point. Generating multiple random solutions and optimizing solutions in certain of iteration. The algorithm selects a solution having the fastest rate of convergence. The algorithm optimizes the solution and finds a global optimal solution with standard deviation of 0.

4.6.2. Problem 2: Neurophysiology Application

The Neurophysiology Application [16] is described as follows:

$$\begin{cases} f_1(x) = x_1^2 + x_3^2 - 1 = 0 \\ f_2(x) = x_2^2 + x_4^2 - 1 = 0 \\ f_3(x) = x_5 x_3^3 + x_6 x_4^3 - c_1 = 0 \\ f_4(x) = x_5 x_1^3 + x_6 x_2^3 - c_2 = 0 \\ f_5(x) = x_5 x_1 x_3^2 + x_6 x_4^2 x_2 - c_3 = 0 \\ f_6(x) = x_5 x_1^2 x_3 + x_6 x_2^2 x_4 - c_4 = 0 \end{cases}$$

$$-10 \le x_i \le 10 \ (i = 1,...,6)$$

The constants c_i can be randomly chosen. In our experiments, we considered $c_i = 0$ (i= 1, . . . , 4). There are 12 solutions of paper [7] for the Neurophysiology Application problem; these solutions are Pareto optimal solutions found by evolutionary computation approach. We choose the solution 1 that is displayed below to compare with solutions found by PDS algorithm 1. There are many solutions found by PDS algorithm 1 and we choose two typical solutions to display them in the table below:

	E. A. [7]	Solution 1	Solution 2		E. A. [7]	Solution 1	Solution 2
x_1	-0.8282192996	0.703475	0.820345	$f_1(x)$	-0.3139636071	-0.0000000060	0.0000000722
x_2	0.5446434961	0.667647	0.820345	$f_2(x)$	-0.1206333343	0.0000000091	0.0000000722
x_3	-0.0094437659	0.710720	0.571869	$f_3(x)$	0.0652332757	0.0000000000	0.0000000000
x_4	0.7633676230	0.744478	0.571869	$f_4(x)$	0.0123681793	0.0000000000	0.0000000000
x_5	0.0199325983	0.000000	-2.689698	$f_5(x)$	0.0465408323	0.0000000000	0.0000000000
x_6	0.1466452805	0.000000	2.689698	$f_6(x)$	0.0330776356	0.0000000000	0.0000000000
				$\varepsilon(x)$		0.0000000091	0.0000000722

Table 9. Comparison of results for Neurophysiology Application.

	$\varepsilon(x)$
Min	0.0000000091
Max	0.0000005529
Average	0.0000001870
Median	0.0000001084
Standard deviation	0.0000001755

Table 10. Statistical table of 30 independent runs of PDS algorithm 1 for Neurophysiology Application.

Application Neurophysiology has many local optimal points. After a number of iterations, the algorithm selects multiple optimal solutions with standard deviation is 0.0000001755.

4.6.3. Problem 3: Chemical Equilibrium Application

The chemical equilibrium system [10] [8] is described as follows:

$$\begin{cases} f_1(x) = x_1x_2 + x_1 - 3x_5 = 0 \\ f_2(x) = 2x_1x_2 + x_1 + x_2x_3^2 + R_8x_2 - Rx_5 + 2R_{10}x_2^2 + R_7x_2x_3 + R_9x_2x_4 = 0 \\ f_3(x) = 2x_2x_3^2 + 2R_5x_3^2 - 8x_5 + R_6x_3 + R_7x_2x_3 = 0 \\ f_4(x) = R_9x_2x_4 + 2x_4^2 - 4Rx_5 = 0 \\ f_5(x) = x_1(x_2 + 1) + R_{10}x_2^2 + x_2x_3^2 + R_8x_2 + R_5x_3^2 + x_4^2 - 1 + R_6x + R_7x_2x_3 + R_9x_2x_4 = 0 \end{cases}$$

$$-10 \le x_i \le 10 \ (i = 1, ..., 5)$$

There are 12 solutions of paper [7] for the Chemical Equilibrium Application problem; these solutions are Pareto optimal solutions found by evolutionary computation approach. We choose the solution 1 that is displayed below to compare with solutions found by PDS algorithm 1. There are many solutions found by PDS algorithm 1 and we choose two typical solutions to display them in the table below:

	E. A. [7]	Solution 1	Solution 2		E. A. [7]	Solution 1	Solution 2
x_1	-0.0163087544	0.011212	0.010762	$f_1(x)$	-0.1525772444	0.0038723421	0.0036961619
x_2	0.2613604709	9.155043	9.579740	$f_2(x)$	-0.3712483541	-0.0038723448	-0.0036961549
x_3	0.5981559224	0.125929	0.123221	$f_3(x)$	-0.0265535274	0.0038688806	0.0036932686
x_4	0.8606983883	0.857346	0.857893	$f_4(x)$	-0.2784694038	0.0038717720	0.0034008286
x_5	0.0440020125	0.036662	0.036721	$f_5(x)$	-0.1168649340	-0.0018247861	-0.0007101592
				$\varepsilon(x)$		0.0038723448	0.0036961619

Table 11. Comparison of results for Chemical Equilibrium Application.

	$\varepsilon(x)$
Min	0.0036961619
Max	0.0052934327
Average	0.0042505633
Median	0.0040423263
Standard deviation	0.0005960323

Table 12. Statistical table of 30 independent runs of PDS algorithm 1 for Chemical Equilibrium Application.

Chemical Equilibrium Application has many local optimal points. After a number of iterations, the algorithm 1 selects multiple optimal solutions with standard deviation is 0.0005960323

4.6.4. Problem 4: Kinematic Application

The kinematic application kin2 [12][8] describes the inverse position problem for a six-revolute-joint problem in mechanics. The equations describe a denser constraint system and are given as follows:

$$\begin{cases} f_i(x) = x_i^2 + x_{i+1}^2 - 1 = 0 \\ f_{4+i}(x) = a_{1i}x_1x_3 + a_{2i}x_1x_4 + a_{3i}x_2x_3 + a_{4i}x_2x_4 + a_{5i}x_2x_7 + a_{6i}x_5x_8 + a_{7i}x_6x_7 + a_{8i}x_6x_8 + \\ \qquad a_{9i}x_1 + a_{10i}x_2 + a_{11i}x_3 + a_{12i}x_4 + a_{13i}x_5 + a_{14i}x_6 + a_{15i}x_7 + a_{16i}x_8 + a_{17i} = 0 \\ 1 \le i \le 4; \end{cases}$$

$$-10 \le x_j \le 10 \ (j = 1,...,8)$$

The coefficients a_{ki}, $1 \le k \le 17$, $1 \le i \le 4$, are given in the table below:

a_{ki}		i			
		1	2	3	4
k	1	- 0.249150680	+0.125016350	-0.635550077	+1.48947730
	2	+1.609135400	-0.686607360	-0.115719920	+0.23062341
	3	+0.279423430	-0.119228120	-0.666404480	+1.32810730
	4	+1.434801600	-0.719940470	+0.110362110	-0.25864503
	5	+0.000000000	-0.432419270	+0.290702030	+1.16517200
	6	+0.400263840	+0.000000000	+1.258776700	-0.26908494
	7	- 0.800527680	+0.000000000	-0.629388360	+0.53816987
	8	+0.000000000	-0.864838550	+0.581404060	+0.58258598
	9	+0.074052388	-0.037157270	+0.195946620	-0.20816985
	10	- 0.083050031	+0.035436896	-1.228034200	+2.68683200
	11	- 0.386159610	+0.085383482	+0.000000000	-0.69910317
	12	- 0.755266030	+0.000000000	-0.079034221	+0.35744413
	13	+0.504201680	-0.039251967	+0.026387877	+1.24991170
	14	- 1.091628700	+0.000000000	-0.057131430	+1.46773600
	15	+0. 000000000	-0.432419270	-1.162808100	+1.16517200
	16	+0.049207290	+0.000000000	+1.258776700	+1.07633970
	17	+0.049207290	+0.013873010	+2.162575000	-0.69686809

Table 13. Coefficients a_{ki} for the kinematic application kin2.

There are 10 solutions of paper [7] for the Kinematic Application Kin2 problem; these solutions are Pareto optimal solutions found by evolutionary computation approach. We choose the solution 1 that is displayed below to compare with solutions found by PDS algorithm 1. There are many solutions found by PDS algorithm 1 and we choose two typical solutions to display them in the table below:

	E. A. [7]	Solution 1	Solution 2
x_1	-0.0625820337	0.953447	0.958991
x_2	0.7777446281	-0.301560	-0.283426
x_3	-0.0503725828	0.953447	0.958991
x_4	0.3805368959	0.301561	-0.283448
x_5	-0.5592587603	0.953447	0.958984
x_6	-0.6988338865	0.010363	-0.136180
x_7	0.3963927675	0.094760	0.856105
x_8	0.0861763643	-0.099564	-0.198128

	E. A. [7]	Solution 1	Solution 2
$f_1(x)$	-0.3911967825	-0.0000003846	-0.0000059644
$f_2(x)$	-0.3925758964	-0.0000003846	-0.0000059644
$f_3(x)$	-0.8526542738	0.0000002185	0.0000065068
$f_4(x)$	-0.5424213099	0.0000002185	-0.0000069190
$f_5(x)$	0.7742116224	0.0000004085	0.0000069818
$f_6(x)$	-0.3828834764	-0.0000002465	0.0000069099
$f_7(x)$	-0.7843806421	0.0000005466	-0.0000070056
$f_8(x)$	0.4655985543	-0.0000004874	0.0000060584
$\varepsilon(x)$		0.0000005466	0.0000070056

Table 14. Comparison of results for Kinematic Application kin2.

	$\varepsilon(x)$
Min	0.0000005466
Max	0.2580684087
Average	0.0443614392
Median	0.0052189659
Standard deviation	0.079379872

Table 15. Statistical table of 30 independent runs of PDS algorithm 1 for the kinematic application kin2.

Kinematic kin2 application has many local optimal points. After a number of iterations, the algorithm 1 selects multiple optimal solutions with standard deviation is 0.079379872.

4.6.5. Problem 5: Combustion Application

The combustion problem for a temperature of 3000 ◦C [13][8] is described by the following system of equations:

$$\begin{cases} f_1(x) = x_2 + 2x_6 + x_9 + 2x_{10} - 10^{-5} = 0 \\ f_2(x) = x_3 + x_8 - 3.10^{-5} = 0 \\ f_3(x) = x_1 + x_3 + 2x_5 + 2x_8 + x_9 + x_{10} - 5.10^{-5} = 0 \\ f_4(x) = x_4 + 2x_7 - 10^{-5} = 0 \\ f_5(x) = 0.5140437.10^{-7} x_5 - 2x_1^2 = 0 \\ f_6(x) = 0.1006932.10^{-6} x_6 - 2x_2^2 = 0 \\ f_7(x) = 0.7816278.10^{-15} x_7 - x_4^2 = 0 \\ f_8(x) = 0.1496236.10^{-6} x_8 - x_1 x_3 = 0 \\ f_9(x) = 0.6194411.10^{-7} x_9 - x_1 x_2 = 0 \\ f_{10}(x) = 0.2089296.10^{-14} x_{10} - x_1 x_2^2 = 0 \end{cases}$$

$$-10 \le x_i \le 10 \ (i = 1,...,10)$$

There are 8 solutions of paper [7] for the Combustion Application; these solutions are Pareto optimal solutions found by evolutionary computation approach. We choose the solution 1 that is displayed below to compare with solutions found by PDS algorithm 1. There are many

solutions found by PDS algorithm 1 and we choose two typical solutions to display them in the table below:

	E. A. [7]	Solution 1	Solution 2		E. A. [7]	Solution 1	Solution 2
x_1	-0.0552429896	0.000353	0.000003	$f_1(x)$	0.0274133880	0.0000000000	0.0000000000
x_2	-0.0023377533	0.000190	0.000486	$f_2(x)$	0.0841848522	0.0000000000	0.0000000000
x_3	0.0455880930	-0.000537	0.242296	$f_3(x)$	0.1482418892	0.0000000000	0.0000000000
x_4	-0.1287029472	0.000000	0.000020	$f_4(x)$	0.0839188566	0.0000000000	0.0000000000
x_5	0.0539771728	0.710649	1.332765	$f_5(x)$	-0.0030517851	-0.0000000881	0.0000000685
x_6	-0.0151036079	-0.030582	1.163017	$f_6(x)$	-0.0000109317	-0.0000000753	-0.0000003553
x_7	0.1063159019	0.000005	-0.000005	$f_7(x)$	-0.0165644486	0.0000000000	-0.0000000004
x_8	0.0386267592	0.000567	-0.242266	$f_8(x)$	0.0025184283	0.0000001896	-0.0000007631
x_9	-0.1144905135	-2.905380	-2.519984	$f_9(x)$	-0.0001291516	-0.0000002470	-0.0000001576
x_{10}	0.0872294353	1.483182	0.096737	$f_{10}(x)$	0.0000003019	0.0000000000	0.0000000000
				$\varepsilon(x)$		0.0000002470	0.0000007631

Table 16. Comparison of results for Combustion Application.

	$\varepsilon(x)$
Min	0.0000002470
Max	0.0000376137
Average	0.0000126718
Median	0.0000091598
Standard deviation	0.0000110185

Table 17. Statistical table of 30 independent runs of PDS algorithm 1 for Combustion Application.

Combustion Application has many local optimal points. After a number of iterations, the algorithm 1 selects multiple optimal solutions with standard deviation is 0.0000110185.

4.6.6. Problem 6: Economics Modeling Application

The Economics Modeling Application problem [13] is described by the following system of equations:

$$\begin{cases} f_k(x) = \left(x_k + \sum_{i=1}^{n-k-1} x_i x_{i+k} \right) x_n - c_k = 0 \qquad (1 \le k \le n-1) \\ f_n(x) = \sum_{i=1}^{n-1} x_i + 1 = 0 \end{cases}$$

$$-10 \le x_i \le 10 \ \ (i = 1,..., n)$$

The constants c_k ($1 \le k \le n-1$) can be randomly chosen. We choose the value 0 for the constants and the case of n=20 equations in our experiments.

There are 4 solutions of paper [7] for the Economics Modeling Application problem; these solutions are Pareto optimal solutions found by evolutionary computation approach. We choose

the solution 1 that is listed below to compare with solutions found by PDS algorithm 1. Here the solution 1 of [7]:

x=(-0.1639324, -0.3813209, 0.2242448, -0.0755094, 0.1171098, 0.0174083, -0.0594358, -0.2218284, 0.1856304, -0.2653962, -0.3712114, -0.3440810, -0.1060168, 0.0218564, -0.2028748, 0.0533728, -0.0587111, 0.0057098, -0.0149290, -0.0004102);

$f_1(x)$ = 0.0000194318; $f_2(x)$= 0.0000973461; $f_3(x)$=-0.0001201028; $f_4(x)$=-0.0000239671; $f_5(x)$ =-0.0000561734; $f_6(x)$=-0.0000389625; $f_7(x)$= 0.0000390795; $f_8(x)$= 0.0000931186; $f_9(x)$ =-0.0001293920; $f_{10}(x)$=0.0000501015; $f_{11}(x)$=0.0001008920; $f_{12}(x)$=0.0001601619; $f_{13}(x)$=0.0000063289; $f_{14}(x)$=-0.0000079648; $f_{15}(x)$=0.0000766372; $f_{16}(x)$=-0.0000235752; $f_{17}(x)$=0.0000221321; $f_{18}(x)$=-0.0000033461; $f_{19}(x)$=0.0000061239; $f_{20}(x)$=-0.6399149000;

There are many solutions found by PDS algorithm 1 and we choose three typical solutions and have them reported in the table below:

	Solution 1	**Solution 2**	**Solution 3**
$\mathbf{x_1}$	0.611228	0.027417	5.302852
$\mathbf{x_2}$	1.082497	2.996639	0.035055
$\mathbf{x_3}$	6.830700	-1.462483	2.212260
$\mathbf{x_4}$	-5.082635	1.065133	-0.874504
$\mathbf{x_5}$	3.330180	0.869460	1.236087
$\mathbf{x_6}$	1.765048	1.411347	-1.646429
$\mathbf{x_7}$	-3.169329	-5.308277	0.205364
$\mathbf{x_8}$	5.596410	-2.958699	3.330769
$\mathbf{x_9}$	2.001166	1.490692	-4.515966
$\mathbf{x_{10}}$	1.731434	-0.705740	-2.944068
$\mathbf{x_{11}}$	-0.880434	0.119057	1.496767
$\mathbf{x_{12}}$	-5.275206	1.112881	-0.240641
$\mathbf{x_{13}}$	-2.052474	-1.354802	-1.709052
$\mathbf{x_{14}}$	-9.662985	-0.423468	-1.888176
$\mathbf{x_{15}}$	3.184984	2.007751	-0.873979
$\mathbf{x_{16}}$	1.093321	-1.565469	1.410222
$\mathbf{x_{17}}$	-0.457790	0.339037	0.178445
$\mathbf{x_{18}}$	-5.270496	1.009605	-0.646229
$\mathbf{x_{19}}$	3.624381	0.329919	-1.068777
$\mathbf{x_{20}}$	0.000000	0.000000	0.000000
ε(x)	0.0000000000	0.0000000000	0.0000000000

Table 18. Three solutions for Economics Modeling Application found by PDS algorithm 1. Three solutions above have $g_i(x)$=0.0 ($1 \le i \le n$=20) and 10 digits after decimal point are zero.

	ε(x)
Min	0.0000000000
Max	0.0000000000
Average	0.0000000000
Median	0.0000000000
Standard deviation	0.0000000000

Table 19. Statistical table of 30 independent runs of PDS algorithm 1 for Economics Modeling Application.

Economics Application Modeling has many global optimal points. After a number of iterations, the algorithm 1 selects multiple optimal solutions with standard deviation is 0.0000000000.

	Example 1	Example 2	Problem 1	Problem 2	Problem 3	Problem 4	Problem 5	Problem 6
E. A. [7]	5.14	5.09	39.07	28.90	32.71	221.09	151.12	640.92
PDS alg.	5	5	30	30	30	30	30	20

Table 20. Comparison of the running times (second) of evolutionary approach [7] and PDS algorithm 1.

Remarks:

- For each problem, solutions that are found by PDS algorithm 1 dominate solutions of [7]. That means, solutions of [7] are dominated and NOT Pareto optimal solutions!
- PDS algorithm 1 is very efficient for solving equations systems. The algorithm 1 has the abilities to overcome local optimal solutions and to obtain global optimal solutions.

5. PDS algorithm 2 for solving multi-objective optimization problems

5.1. General steps of the PDS algorithm 2

We need to set three parameters S,M and L as follows:

- Let S be the set of Pareto optimal solutions to find
- Let M be a number of Pareto optimal solutions which the algorithm has the ability to find
- After generating a random feasible solution X, set L is the number so large that after repeated L times the algorithm has an ability to find a Pareto optimal solution that dominates the solution X.

The PDS algorithm 2 for solving multi-objective optimization problems is described with general steps as follows:

S1. i←1 (determine i-th solution);
S2. Select a randomly generated feasible solution X_i;
S3. j←1 (create jth loop);
S4. Use the Changing Procedure to transform the solution X_i into a new solution Y;
S5. If (Y is not feasible) then return S4.
S6. If (X_i is dominated by Y) then $X_i \leftarrow Y$;
S7. If j<L then j←j+1 and return S4;
S8. Put X_i on the set S;
Remove the solution in the set S which is dominated by another;
Remove overlapping solutions in the set S;
Set | S | = i;
S9. If i<M then i←i+1 and return S2;
S10. The end of PDS algorithm 2;

Remarks: After generating a random feasible solution x, the algorithm repeats L times to find a solution that dominates the solution x. Thus each of the solutions works independently of the other solutions. The changes of solutions are driven by probabilities. Every solution has an ability to change and directs its position to a point of the Pareto front.

5.2. Examples: Seven optimization test cases of DTLZ problems

In order to assess the performance of PDS algorithm 2 for solving multi-objective optimization problems, the algorithm will be benchmarked by using seven optimization test cases developed by Deb et al. [3]. The problems are minimization problems with M=3 objectives.

DTLZ1:

$$Min\ f_1(x)=\frac{1}{2}x_1x_2(1+g(X_M));\ Min\ f_2(x)=\frac{1}{2}x_1(1-x_2)(1+g(X_M));Min\ f_3(x)=\frac{1}{2}(1-x_1)(1+g(X_M))$$

$$g(X_M)=100(|X_M|+\sum_{x\in X_M}(x_i-0.5)^2-\cos(20\pi(x_i-0.5)))$$

$$0\le x_i\le 1\ (i=1...7),\ x=(x_1,...,x_7),\ X_M=(x_3,...,x_7).$$

DTLZ2:

$$Min\ f_1(x)=(1+g(X_M))\cos(x_1\pi/2)\cos(x_2\pi/2);Min\ f_2(x)=(1+g(X_M))\cos(x_1\pi/2)\sin(x_2\pi/2);$$

$$Min\ f_3(x)=(1+g(X_M))\sin(x_1\pi/2);$$

$$g(X_M)=\sum_{x_i\in X_M}(x_i-0.5)^2$$

$$0\le x_i\le 1(i=1...12),x=(x_1,...,x_{12}),\ X_M=(x_3,...,x_{12}).$$

DTLZ3: As DTLZ2, except the equation for g is replaced by the one from DTLZ1.

DTLZ4: As DTLZ2, except x_i is replaced by x_i^α where $\alpha>0$ (i=1,2)

DTLZ5: As DTLZ2, except x_2 is replaced by

$$\frac{1+2g(X_M)x_2}{2(1+g(X_M))}$$

DTLZ6: As DTLZ5, except the equation for g is replaced by

$$\sum_{i\in X_M}x_i^{0.1}$$

DTLZ7:

$$Min\ f_1(x)=x_1;Min\ f_2(x)=x_2;Min\ f_3(x)=(1+g(X_M))h(f_1(x),f_2(x),g(X_M))$$

$$g(X_M)=1+\frac{9}{|X_M|}\sum_{x_i\in X_M}x_i;$$

$$h(f_1(x),f_2(x),g(X_M))=M-\sum_{i=1}^{M-1}\left[\frac{f_i(x)}{1+g(X_M)}(1+\sin(3\pi f_i(x)))\right]$$

$$0\le x_i\le 1\ (i=1...22),x=(x_1,...,x_{22}),\ X_M=(x_3,...,x_{22}).$$

Because the memory of computer is limited, we divide the Pareto surface into four parts as follows:

- Part 1: $f_1(x) \leq 0.5$ and $f_2(x) \leq 0.5$
- Part 2: $f_1(x) \leq 0.5$ and $f_2(x) \geq 0.5$
- Part 3: $f_1(x) \geq 0.5$ and $f_2(x) \leq 0.5$
- Part 4: $f_1(x) \geq 0.5$ and $f_2(x) \geq 0.5$

Set L=30000 and M=700, we apply PDS algorithm 2 to finding 700 Pareto optimal solutions for each part. We use two digits after decimal point for all problems. It takes 120 seconds to implement PDS algorithm 2 for finding Pareto surface of each problem. Here the Pareto surfaces of illustrative examples are found by PDS algorithm 2.

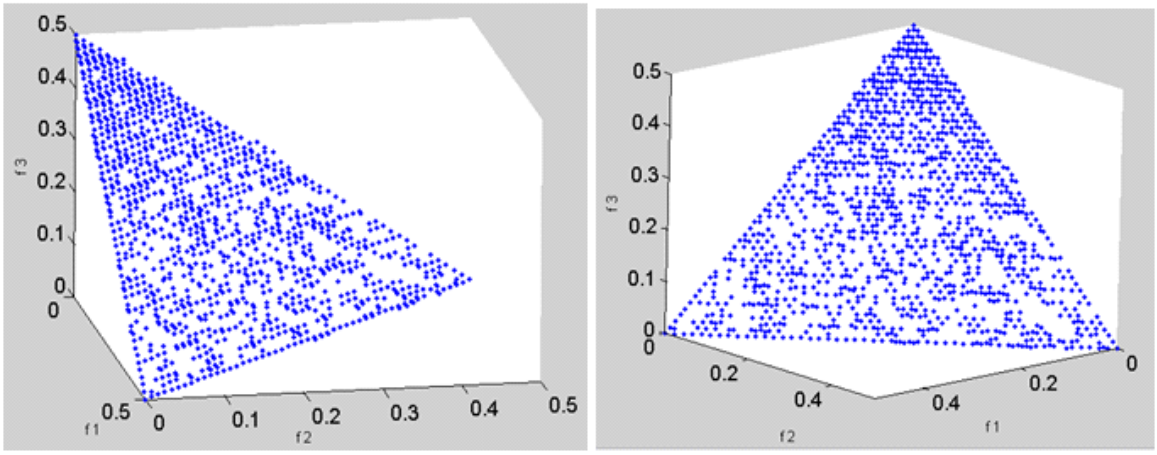

Fig.2. Pareto surface of DTLZ1

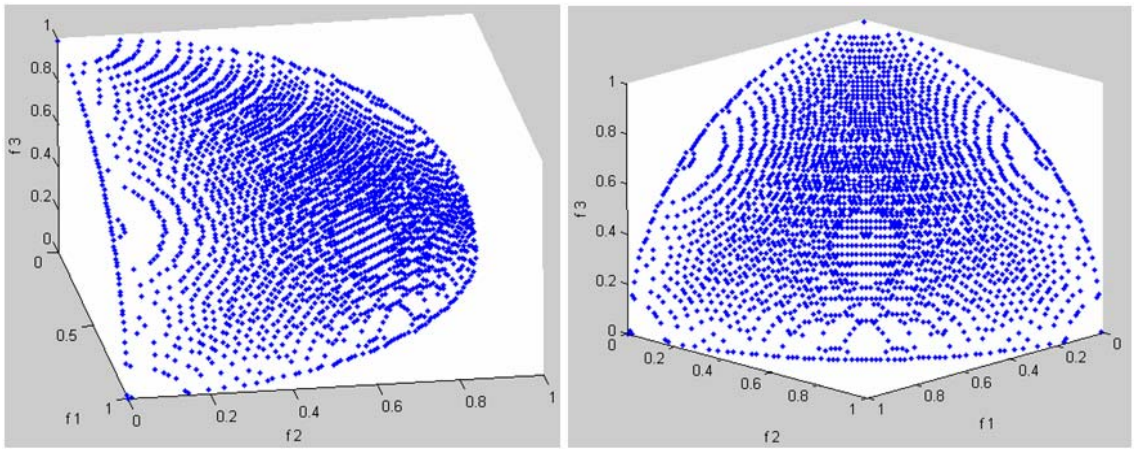

Fig.3. Pareto surface of DTLZ2

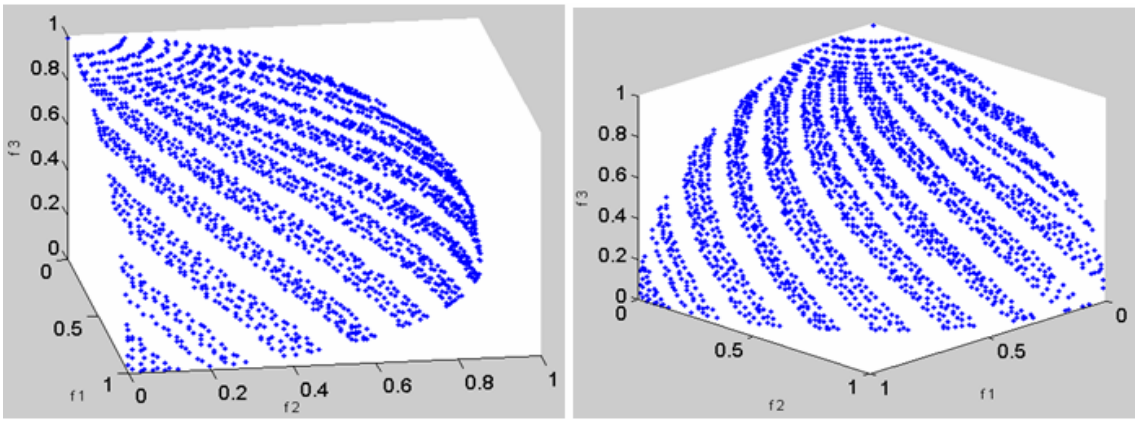

Fig.4. Pareto surface of DTLZ3

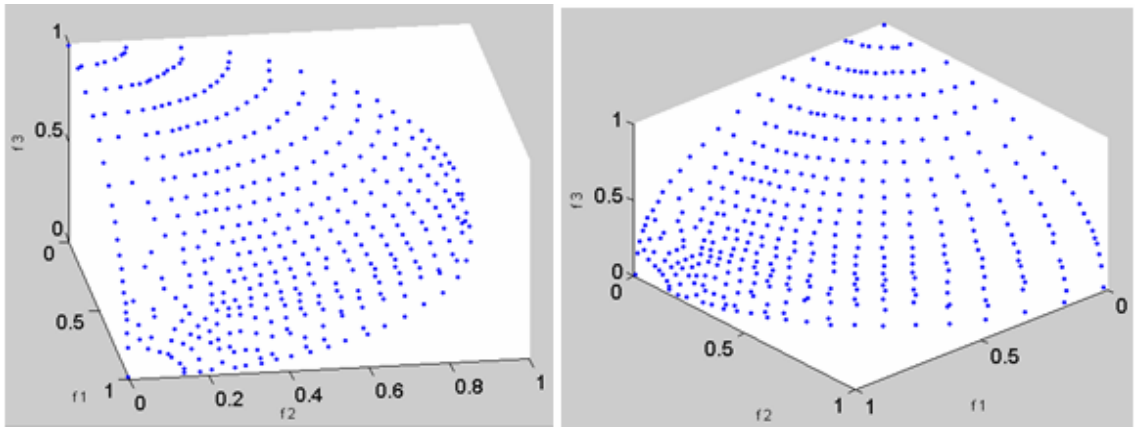

Fig. 5. Pareto surface of DTLZ4 (α=10)

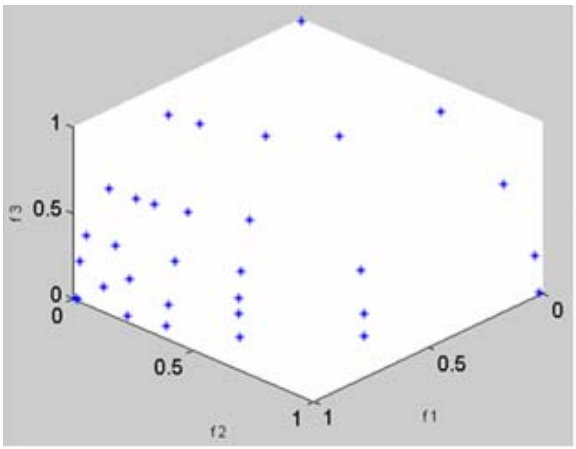

Fig. 6. Pareto surface of DTLZ4 (α=50)

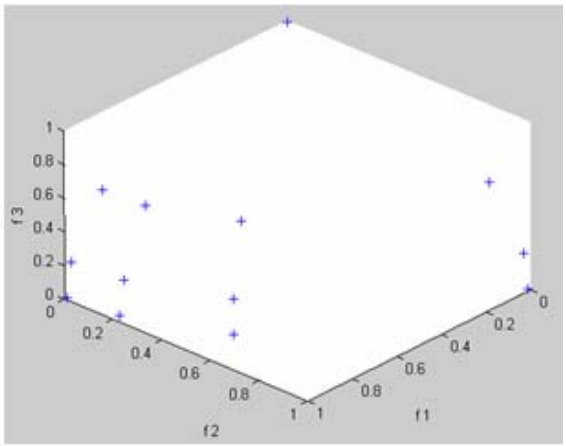

Fig. 7. Pareto surface of DTLZ4 (α=100)

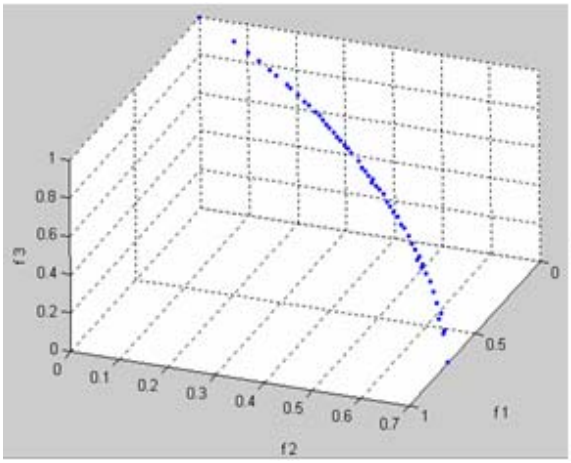

Fig.8. Pareto surface of DTLZ5

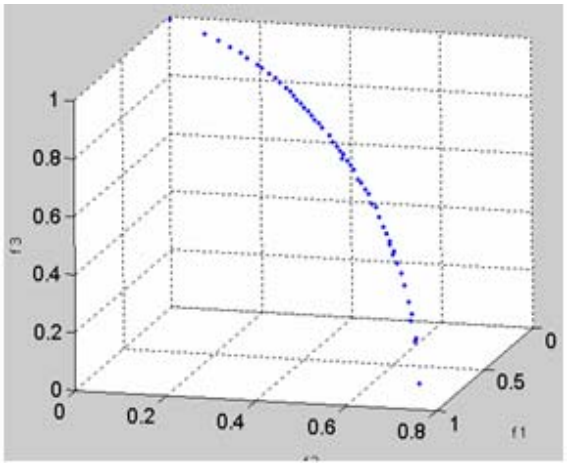

Fig.9. Pareto surface of DTLZ6

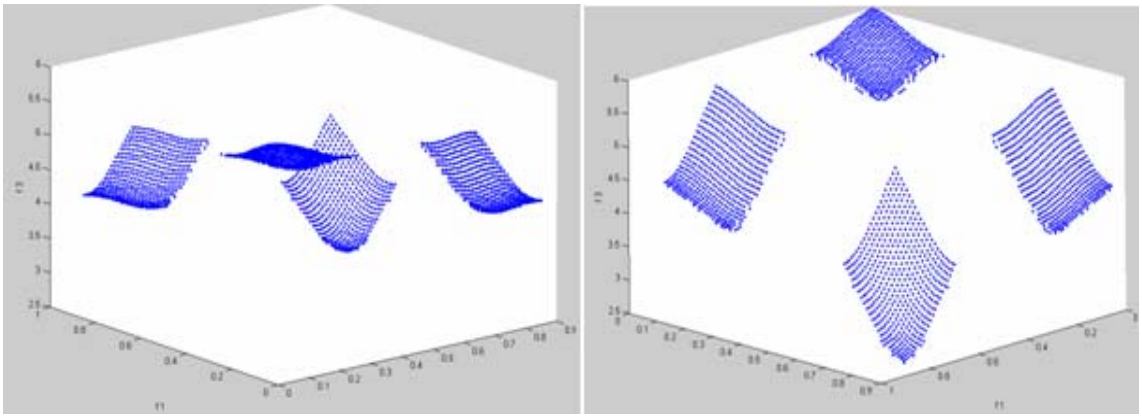

Fig. 10. Pareto surface of DTLZ7

Now we use three digits after decimal point for problem DTLZ4 and the Pareto front is found by PDS algorithm 2 with α=100 as follows:

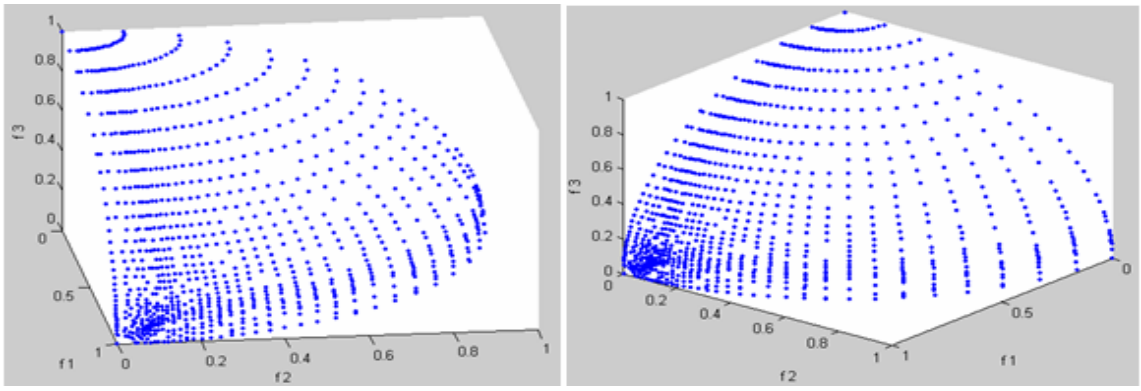

Fig. 11. Pareto surface of DTLZ4 (α=100, 3 digits after decimal point)

Remarks:

- PDS algorithm 2 has an ability to find a large number of Pareto optimal solutions and these solutions could express the concentrated regions of Pareto front.

- PDS algorithm 2 has an ability to maintain diversity and the overall distribution of solutions on Pareto front is acceptable.

6. Conclusions

In this paper, we consider a class of optimization problems having the following characteristics: there exists a fixed number k ($1 \le k < n$) which does not depend on the size n of the problem such that just randomly changing the values of k variables; we may find a new solution that is better than the current one, we call it the class of optimization problems O_k. Next we build the Search Technique based on two conditions:

- In each iteration of loop, the technique selects randomly k ($1 \le k < n$) variables of the current solution x to transform x into a new solution y.
- The technique finds the values of digits of variables from left digits to right digits step by step.

We compute the complexity of the Search Technique; especially we use the absorbing Markov Chain to argue the convergence of the Search Technique.

We design a Changing Procedure that it transforms a solution x into a new solution y so that both single-objective optimization problems and multi-objective optimization problems can use it. The Changing Procedure uses probabilities to drive the changes of values of solutions according to two conditions above for finding a better solution than the current one. We build two applications of the Changing Procedure, Probabilistic-Driven Search algorithm 1 for solving single-objective optimization problems and Probabilistic-Driven Search algorithm 2 for solving multi-objective optimization problems. We test PDS algorithm 1 and 2 by implementing the algorithms on benchmark optimization problems, and we find very good and stable results. The algorithms only find a solution in each of iteration of loop. Each solution works independently of each other and the solutions themselves change under the control of probability.

For application of PDS algorithm 1 we transform the nonlinear equations system into a single-objective optimization problem. PDS algorithm 1 is very efficient for solving nonlinear equations systems. PDS algorithm 1 has the abilities to overcome local optimal solutions and to obtain global optimal solutions.

PDS algorithm 2 has the following advantages:

- There is no population or swarm, the algorithm is very simple and fast.
- The changes of values of solutions are driven by the probabilities. Every solution operates by itself and independently with other solutions. Every solution has an ability to change and directs its position to a point of the Pareto front.
- The PDS algorithm 2 could find a large number of Pareto optimal solutions and the overall distribution of solutions on Pareto front is acceptable.

In practice, we do not know the true value of parameter k for each problem. According to statistics of many experiments, the best thing is to use k in the ratio 10%-60% of n with $n \ge 10$.

Many optimization problems have very narrow feasible domains that require the algorithm having an ability to search values of two or more consecutive digits simultaneously to find a feasible solution. We study this case and the results will be reported in the next paper. We also compare Search via Probability algorithm of papers [15] with PDS algorithm 1 of this paper for solving engineering optimization problems. Based on the idea of search probabilities, we

can think of developing and expanding to a search model with the probabilities of search objects that are not real numbers.

References

[1] Broyden C. G., "A class of methods for solving nonlinear simultaneous equations," Math. Comput., vol. 19, no. 92, pp. 577–593, Oct. 1965.

[2] Conn A. R., Gould N. I. M., and Toint P. L., Trust-Region Methods. Philadelphia, PA: SIAM, 2000.

[3] Deb K., Thiele L., Laumanns M. and Zitzler E., Scalable Multi-Objective Optimization Test Problems. In Congress on Evolutionary Computation (CEC 2002), pages 825–830. IEEE Press, 2002.

[4] Denis J. E. and Wolkowicz H., "Least change secant methods, sizing, and shifting," SIAM J. Numer. Anal., vol. 30, pp. 1291–1314, 1993.

[5] Denis J. E., On Newton like methods, Numer. Math., vol. 11, no. 4, pp. 324–330, May 1968.

[6] Effati S. and Nazemi A. R., A new method for solving a system of the nonlinear equations, Appl. Math. Comput., vol. 168, no. 2, pp. 877–894, 2005.

[7] Grosan C., Abraham A., A New Approach for Solving Nonlinear Equations Systems. IEEE Transactions on Systems, Man, and Cybernetics, Part A 38(3), 698-714, 2008.

[8] Hentenryck P. V., McAllester D. and Kapur D., Solving polynomial systems using a branch and prune approach, SIAM J. Numer. Anal., vol. 34, no. 2, pp. 797–827, Apr. 1997.

[9] Hong H. and Stahl V., Safe starting regions by fixed points and tightening, Computing, vol. 53, no. 3/4, pp. 323–335, Sep. 1994.

[10] Meintjes K. and Morgan A. P., "Chemical equilibrium systems as numerical test problems," ACM Trans. Math. Softw., vol. 16, no. 2, pp. 143–151, Jun. 1990.

[11] Moore R. E., Methods and Applications of Interval Analysis. Philadelphia, PA: SIAM, 1979.

[12] Morgan A. P., Computing all solutions to polynomial systems using homotopy continuation, Appl. Math. Comput., vol. 24, no. 2, pp. 115–138, Nov, 1987.

[13] Morgan A. P., Solving Polynomial Systems Using Continuation for Scientific and Engineering Problems. Englewood Cliffs, NJ: Prentice-Hall,1987.

[14] Ortega J. M. and Rheinboldt W. C., Iterative Solution of Nonlinear Equations in Several Variables. New York: Academic, 1970.

[15] Thong N. H. and Hao T. V., Search via Probability Algorithm for Engineering Optimization Problems, In Proceedings of XIIth International Conference on Applied Stochastic Models and Data Analysis (ASMDA2007), Chania, Crete, Greece, 2007. In book: Recent Advances in Stochastic Modeling and Data Analysis, editor: Christos H. Skiadas, publisher: World Scientific Publishing Co Pte Ltd, 454 – 463, 2007.

[16] Verschelde J., Verlinden P. and Cools R., Homotopies exploiting Newton polytopes for solving sparse polynomial systems, SIAM J. Numer. Anal., vol. 31, no. 3, pp. 915–930, Jun. 1994.

[17] Yang X.-S., A New Metaheuristic Bat-Inspired Algorithm, in: Nature Inspired Cooperative Strategies for Optimization (NICSO 2010) (Eds. J. R. Gonzalez et al.), SCI 284, 65-74 (2010).

Website

[18] http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO_files/Page2502.htm