

Economic Load Dispatch using Intelligent Optimization with Fuzzy Control

Johnny C.Y. Lai

Centre for Signal
Processing, Dept. of
Electronic & Information
Engg., The Hong Kong
Polytechnic University,
Hung Ham, Hong Kong
08900438r@polyu.edu.hk

Frank H.F. Leung

Centre for Signal
Processing, Dept. of
Electronic and Information
Engg., The Hong Kong
Polytechnic University,
Hung Ham, Hong Kong
enfrank@inet.polyu.edu.hk

Sai-Ho Ling

Centre for Health
Technologies, Faculty of
Engineering and
Information Technology,
University of Technology,
Sydney, NSW, Australia
steve.ling@uts.edu.au

Edwin C. Shi

Centre for Signal
Processing, Dept. of
Electronic and Information
Engg., The Hong Kong
Polytechnic University,
Hung Ham, Hong Kong
forseeking@gmail.com

Abstract— In this paper, Differential Evolution (DE) that incorporates fuzzy control and k-nearest neighbors algorithm is proposed to tackle the economic load dispatch problem. To provide the self-terminating ability, a technique called Iteration Windows (IW) is introduced to govern the number of iteration in each searching stage during the optimization. The size of IW is controlled by a fuzzy controller, which uses the information provided by the k-nearest neighbors system to analyze the population during the searching process. The controller keeps controlling the IW till the end of the searching process. A wavelet based mutation process is embedded in the DE searching process to enhance the searching performance. The weight F of DE is also controlled by the fuzzy controller to further speed up the searching process. The proposed method is employed to solve the Economic Load Dispatch with Valve-Point Loading (ELD-VPL) Problem. It is shown empirically that the proposed method can terminate the searching process with a reasonable number of iteration and performs significantly better than the conventional methods in terms of convergence speed and solution quality.

Keywords—Fuzzy Control, Economic Load Dispatch, Differential Evolution

I. INTRODUCTION

Differential Evolution (DE) [1] is a population based stochastic optimization algorithm that searches the solution space for the best solution by using a group of particles. It uses the weighted difference between two particles to determine the third particle. Because of the population based searching strategy, DE is less possibly getting trapped in local optima [3]. DE guides the particles to move toward the best solution by using evolutionary operations. Comparing with other optimization algorithms, DE is relatively easy to implement. It requires fewer parameters for tuning, and has a relatively fast convergence speed. It can also offer a high degree of exploration for the population, and has been successfully applied in many optimization problems such as data clustering [4], power plant control [5], optimization of non-linear functions [6], etc.

On applying DE to real-world problems, the maximum number of iteration is usually needed to be provided as the stopping criterion of the algorithm. Different problems might require different numbers of iteration and there is no strong proof for the best number of iteration for a given problem. The maximum number of iteration often is determined by trial and error and/or the user's experience. To tackle this problem, much research work have been done to test the convergence criteria for different search methods [7][8]. In this paper, a new technique called Iteration Windows is embedded into the searching process of DE. On searching, the searching stage of DE can be analyzed based on the distribution of the population in the search domain. By monitoring the distribution of the population, we can get an idea of the state of the searching process. One method for analyzing the population behavior is clustering. In this paper, an improved DE that incorporates fuzzy control and k-nearest neighbors to determine the terminating condition is proposed to handle the problem of Economic Load Dispatch with Valve-Point Loading (ELD-VPL).

ELD is a method to schedule or control the power generator outputs with respect to the load demands, and to operate the power system economically [9]. The goal is to minimize the operation cost of the power generating system. Because of the valve-point loadings and rate limits, the input-output characteristics of modern generators are nonlinear by nature. As a result, the characteristics of ELD-VPL problems are multimodal, discontinuous, and highly nonlinear. Owing to its complexity, we cannot determine the maximum number of iteration easily.

This paper is organized as follows: Section II presents the operation of the DE with self-terminating ability. The problem formulation of ELD-VPL is given in Section III. Experimental study and analysis are given in Section IV. A conclusion will be drawn in Section V.

II. DE WITH SELF-TERMINATING ABILITY

First, a population of solution vectors will be randomly generated over the solution space. They move on the solution space by addition, subtraction and component swapping; until the population converges to the best solution after a number of iteration. The pseudo code for the standard DE (SDE) process is shown in Fig.1. In this paper, DE with self-terminating ability using k -nearest neighbors and fuzzy control is proposed. The pseudo code is shown in Fig. 2.

```

begin
  Initialize the population;
  While (not the maximum number of iteration)
  {
    Mutation operation by equation (2);
    Crossover operation by equation (3);
    Evaluation of the fitness function;
    Selection of the best vector by equation (4);
  }
end

```

Figure 1. Pseudo code for SDE.

```

begin
  Initialize the population;
  While (control not equal to 0) do
  {
    While (Iteration window not equal to 0)
    {
      Mutation operation by equation (2);
      Crossover operation by equation (3);
      Modifying the trial population vectors by
      equation (5);
      Evaluation of the fitness function;
      Select the best vector by equation (4);
      Iteration window = Iteration window - 1;
    }
    Analyze the population by  $k$ -NN by equation (10);
    Count the number of local clusters;
    Determine the new  $k$ , the new  $F$  and iteration
    window size by the fuzzy controller;
    If iteration window size > 6 then control = 1;
    If iteration window size ≤ 6 then control = 0;
  }
end

```

Figure 2. Pseudo code for the proposed system.

A. Standard Differential Evolution (SDE)

DE attempts to control a set of population of N_p vectors for each generation of evolution, with each vector contains D elements of parameters. Let $P_{x,g}$ be the population of the current generation g , and $\mathbf{x}_{i,g}$ be the i -th vector in this population:

$$P_{x,g} = (\mathbf{x}_{i,g}), i = 0, 1, \dots, N_p - 1; g = 0, 1, \dots, g_{\max} \quad (1)$$

$$\mathbf{x}_{i,g} = (x_{j,i,g}), j = 0, 1, \dots, D - 1.$$

where $g_{\max}$ is the maximum number of generation. The population should be initialized as uniformly and randomly

distributed over the solution space, which should have a specified boundary. DE generates a mutated vector $\mathbf{v}_{i,g}$ for each target vector $\mathbf{x}_{i,g}$ by using the mutation operation:

$$\mathbf{v}_{i,g} = \mathbf{x}_{i,g} + F \cdot (\mathbf{x}_{r_1,g} - \mathbf{x}_{r_2,g}) \quad (2)$$

where F is the scaling factor, which is a user-defined value; r_1 and r_2 are two different integers randomly generated from $\{0, 1, \dots, N_p - 1\}$. After the mutation operation, uniform crossover is employed to enhance the differential mutation search strategy and increase the diversity of the perturbed parameter vectors. Each vector element pair $x_{j,i,g}$ and $v_{j,i,g}$ generates a new trial vector element $u_{j,i,g}$:

$$\mathbf{u}_{i,g} = (u_{j,i,g}) = \begin{cases} (v_{j,i,g}) & \text{if } (rand_j(0,1) \leq Cr) \\ (x_{j,i,g}) & \text{otherwise.} \end{cases} \quad (3)$$

where $Cr \in [0, 1]$ is the crossover rate, which is a user-defined value that controls the number of parameters that are copied from the mutant. $rand_j(0,1)$ generates a random value between 0 and 1 for the j -th parameter. The algorithm also ensures $u_{j,i,g}$ gets at least one parameter value as $x_{j,i,g}$ [1]. Then the population is updated by comparing each trial vector to the corresponding target vector $\mathbf{x}_{i,g}$. For finding the global minimum, if the fitness value of the trial vector is lower than that the target vector, DE will replace the target vector in the next generation; otherwise the target vector retains its original place in the population for at least one generation. The selection operation is therefore given by:

$$\mathbf{x}_{i,g+1} = \begin{cases} \mathbf{u}_{i,g} & \text{if } f(\mathbf{u}_{i,g}) \leq f(\mathbf{x}_{i,g}) \\ \mathbf{x}_{i,g} & \text{otherwise.} \end{cases} \quad (4)$$

where $f(\cdot)$ is the fitness function. When the condition to stop further evolution is satisfied, for example, the preset maximum number of iteration has been reached, the algorithm ends with the best solution as the final solution.

B. Wavelet mutation in crossover operation

To enhance the searching performance, we can embed a wavelet mutation in the DE crossover operation [19], which exhibits a fine-tuning ability. Let the i -th vector after crossover for the wavelet mutation be $\mathbf{u}_{i,g} = (u_{0,i,g}, u_{1,i,g}, \dots, u_{D-1,i,g})$. Its element value is inside the vector element's boundary $[para_{\min}^j, para_{\max}^j]$. The mutated crossover vector is given by $\bar{\mathbf{u}}_{i,g} = (\bar{u}_{0,i,g}, \bar{u}_{1,i,g}, \dots, \bar{u}_{D-1,i,g})$, and

$$\bar{u}_{j,i,g} = \begin{cases} u_{j,i,g} + \sigma \times (para_{\max}^j - u_{j,i,g}) & \text{if } \sigma > 0 \\ u_{j,i,g} + \sigma \times (u_{j,i,g} - para_{\min}^j) & \text{if } \sigma \leq 0 \end{cases}, \quad (5)$$

$$\sigma = \frac{1}{\sqrt{a}} e^{-\left(\frac{\phi}{a}\right)^2 / 2} \cos\left(5\left(\frac{\phi}{a}\right)\right), \quad (6)$$

where

$$a = e^{-\ln(\lambda) \times \left(1 - \frac{t}{T}\right)^{\zeta_{wm}} + \ln(\lambda)}. \quad (7)$$

T is the total number of iteration and t is the current iteration number; ζ_{wm} is the shape parameter of the monotonic increasing function, λ is the upper limit of the parameter a . At the early stage, a larger value of $|\sigma|$ gives a larger searching space; at the later stage, $|\sigma|$ is smaller and the algorithm gives a smaller searching space for fine-tuning. After the operation of the wavelet mutation, the population is updated by comparing each trial vector $\bar{\mathbf{u}}_{i,g}$ to the corresponding target vector $\mathbf{x}_{i,g}$ using the method of standard DE as given by (4). A new population is generated and the same DE evolution process is repeated.

C. K-Nearest Neighbors (K-NN)

The k -nearest neighbor algorithm (k -NN) algorithm is used to identify the number of groups of vectors (clusters) in the searching domain. It is a method for classifying and grouping data based on closest training data samples in a multidimensional feature space, and a class label is assigned with each training data sample [10][11][12]. By counting the majority class of its neighbors, a data element can be classified with a class label. The value of k is used to control the number of training data samples being considered in one group.

In the proposed system, a user-defined threshold T is introduced for controlling the size of the cluster. Firstly, the entire population is evaluated by the k -NN algorithm, where k is a value controlled by a fuzzy controller. Then, the k nearest vectors would be determined by k -NN. The total distance between those nearest vectors and the current test vector is:

$$D_{total} = \sum_{i=1}^k d_i(\mathbf{p}, \mathbf{q}_i) \quad (8)$$

where $d_i(\cdot)$ is the Euclidean distance between two points; $\mathbf{p}$ is the current testing point and $\mathbf{q}_i$ is one of the k nearest elements in the searching domain. The Euclidean distance is given by:

$$\begin{aligned} d_i(\mathbf{p}, \mathbf{q}_i) &= \sqrt{(p_1 - q_{i1})^2 + (p_2 - q_{i2})^2 + \cdots + (p_n - q_{in})^2} \\ &= \sqrt{\sum_{j=1}^n (p_j - q_{ij})^2} \end{aligned} \quad (9)$$

The average distance $d_{average}$ of a group (cluster) is:

$$d_{average} = \frac{D_{total}}{k} \quad (10)$$

If $d_{average} < T$, a local cluster is obtained. If $d_{average} \geq T$, a cluster is not present. The system will count the number of clusters obtained, which is passed to the fuzzy controller to determine the size of the next iteration window.

D. Iteration Windows

In the proposed system, the DE algorithm will first run for a small number of iteration (i.e. the first stage), and then the evolution will be paused (Fig. 3). After going through this iteration window, the population will be saved. The population distribution information (i.e. the number of local clusters) obtained from k -NN will be passed to the fuzzy controller to determine the size of the next-stage iteration window. One advantage of using the iteration window is that it can reduce the computation power for reaching the final solution, since the fuzzy controller is not operating during the evolution within a stage. It operates between stages only. Moreover, the total number of iteration is governed by the fuzzy controller instead of fixed by the user.

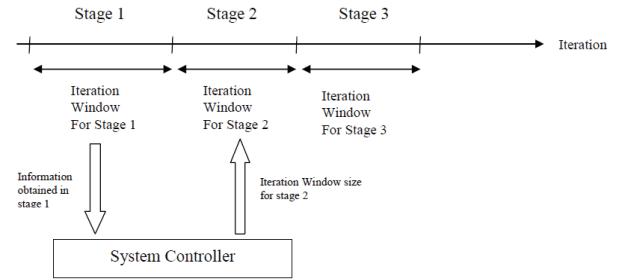


Figure 3. Operation of Iteration Windows.

The relation of the next iteration window's size (the next number of iteration) to the population distribution could be described by some heuristic rules. As a result, the fuzzy controller is modeling this relation. Hence, the total number of iteration could be controlled with respect to the nature of the problem. It does not require the user to determine the total number of iteration, but the system has its own ability to stop the system when it has reliably converged to the real optimum point.

E. Fuzzy Controller

A fuzzy controller realizes fuzzy inference [13]. Its applications include climate control [14], medicine [15], relational database [16], scheduling [17], etc. The process of fuzzy inference involves membership functions, logical operations and fuzzy if-then rules. Implementing fuzzy inference systems can be in Mamdani-type and Sugeno-type [18]. The Mamdani-type fuzzy inference method is used in this paper to construct the fuzzy controller. It uses linguistic control rules obtained from experienced human experts to perform decision making. The output variables must be fuzzy sets. To map the fuzzy set to some crisp output value, the system needs to do defuzzification. In this paper, the method

of center of gravity (COG) is used to perform defuzzification. Fig. 4 shows how the fuzzy inference system works.

The evolution of population-based searching algorithms, including DE, will not be the same for different experiments of a given problem because the algorithms themselves contain a lot of uncertainties. On analyzing the population distribution in the searching process, the derivation of the next iteration window's size from the population distribution should best be described as some heuristic rules. The aforementioned fuzzy controller can then be employed to determine the window's size together with the value of k for the k -NN in the next stage. The fuzzy controller takes the current k value and the population distribution information (i.e. the number of local clusters obtained) provided by k -NN as inputs, and maps this information into the fuzzy membership functions. The fuzzy controller also controls the value of F in the DE operation to enhance the searching process. An overview of the proposed system is shown in Fig. 5.

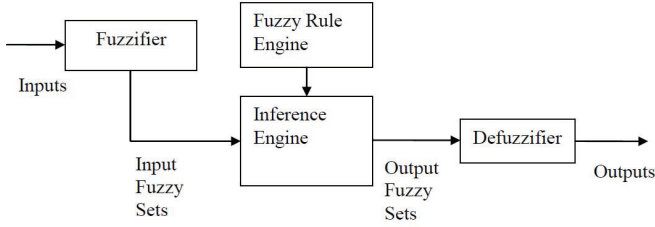


Figure 4. Operation of fuzzy inference system.

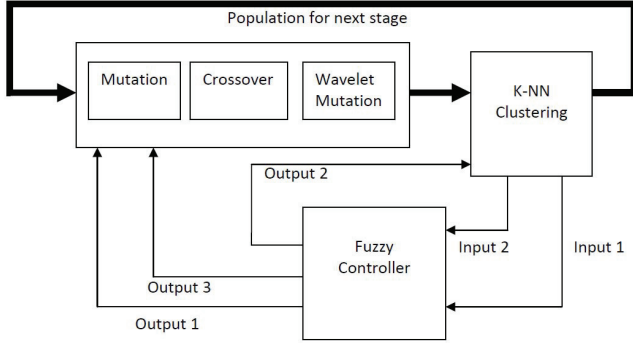


Figure 5. Overview of the proposed system.

III. ELD-VPL PROBLEM

The Economic Load Dispatch with Valve-Point Loading (ELD-VPL) is a method to control or schedule a group of power generator outputs with respect to the load demands, and operate a power system economically so as to minimize the operation cost of the power generating system. The problem can be formulated into the following objective function:

$$\min \sum_{i=1}^n C_i(P_{L_i}) \quad (11)$$

where $C_i(P_{L_i})$ is the operation fuel cost of generator i , and n denotes the number of generators. The problem is subject to balance constraints and generating capacity constraints:

$$D_L = \sum_{i=1}^n P_{L_i} - P_{Loss} \quad (12)$$

$$P_{L_{i,min}} \leq P_{L_i} \leq P_{L_{i,max}}, i=1,2,\dots,n \quad (13)$$

where D_L is the load demand, P_{L_i} is the output power of the i -th generator, P_{Loss} is the transmission loss, and $P_{L_{i,max}}$ and $P_{L_{i,min}}$ are the maximum and minimum output power of the i -th generator, respectively. The operation fuel cost function with valve-point loadings of the generators is given by

$$C_i(P_{L_i}) = a_i P_{L_i}^2 + b_i P_{L_i} + c_i + \left| e_i \times \sin(f_i \times (P_{L_{i,min}} - P_{L_i})) \right| \quad (14)$$

where a_i , b_i , and c_i are the coefficients of the cost curve of the i -th generator, and e_i and f_i are the coefficients of the valve point loadings. The generating units with multivalve steam turbines exhibit a greater variation in the fuel cost functions. The valve-point effects introduce ripples in the heat-rate curves. To solve the ELD-VPL problem by using the proposed system, the solution representation of elements in the population is defined as follows:

$$\mathbf{P} = [P_{L_1} P_{L_2} P_{L_3} \dots P_{L_{n-1}}] \quad (15)$$

where n is the number of generators. From (11), we have

$$P_{L_n} = D_L - \sum_{i=1}^{n-1} P_{L_i} + P_{Loss} \quad (16)$$

In this paper, the power loss is not considered, so

$$P_{L_n} = D_L - \sum_{i=1}^{n-1} P_{L_i} \quad (17)$$

To ensure that P_{L_n} falls within the range $[P_{L_{n,min}}, P_{L_{n,max}}]$,

$$\text{If } P_{L_n} > P_{L_{n,max}} \begin{cases} P_{L_1} = P_{L_1} + (P_{L_n} - P_{L_{n,max}}) \\ P_{L_n} = P_{L_{n,max}} \end{cases} \quad (18)$$

$$\text{If } P_{L_n} < P_{L_{n,min}} \begin{cases} P_{L_1} = P_{L_1} - (P_{L_n} - P_{L_{n,min}}) \\ P_{L_n} = P_{L_{n,min}} \end{cases} \quad (19)$$

It should be noted that (18) and (19) are used to control each individual power generator. If the value of P_{L_i} is outside the constraint boundary (i.e. the safety range), the exceeding portion of the power required will be shared by other generators to ensure that the output power of all individual generators are working within the safety range.

IV. EXPERIMENT AND RESULT

Different kinds of DE methods are applied to a 40-generator system, which is adopted as an example in [19]. The load demand (D) of the problem is 10500 MW. Standard DE [20], DE/local-to-best/1 [20], DE/rand/1 with per-vector-dither [22] and the proposed system are used to solve the ELD-VPL problem. The following simulation conditions are used:

- The shape parameter of the wavelet mutation (ζ_{wm}): It is chosen by trial and error. $\zeta_{wm} = 1$ is used in this paper;
- The parameter λ in (7): 10000;
- The number of population vectors: $N_p = 30$;
- Initial population: It is generated uniformly at random.
- Crossover rate: $C_r = 0.5$;
- The mutation weight factor (For standard DE, DE/local-to-best/1, DE/rand/1 with per-vector-dither): $F = 0.6$;
- The number of iteration: 2000;
- Threshold T for k -NN: 5;
- Membership functions: As shown in Fig. 6 and Fig. 7;
- Fuzzy rule: Listed in Table 1. (Input 1 is the current number of local clusters; Input 2 is the k value of the current iteration window; Output 1 is the next F value for DE; Output 2 is k for the next stage; Output 3 is the next iteration window size.)

TABLE 1. THE FUZZY RULE TABLE

1	If (Input1 is Too Low) then (Output1 is Large).
2	If (Input1 is Low) then (Output1 is Average).
3	If (Input1 is Average) then (Output1 is Low).
4	If (Input1 is Large) then (Output1 is Too Low).
5	If (Input1 is Too Large) then (Output1 is Too Low).
6	If (Input1 is Too Low) or (Input2 is Too Low) then (Output1 is Large).
7	If (Input1 is Low) or (Input2 is Low) then (Output1 is Large).
8	If (Input1 is Average) or (Input2 is Average) then (Output1 is Average).
9	If (Input1 is Large) or (Input2 is Large) then (Output1 is Too Low).
10	If (Input1 is Too Large) or (Input2 is Too Large) then (Output1 is Too Low).
11	If (Input1 is Too Low) then (Output2 is Too Low).
12	If (Input1 is Low) then (Output2 is Low).
13	If (Input1 is Average) then (Output2 is Average).
14	If (Input1 is Large) then (Output2 is Large).
15	If (Input1 is Too Large) then (Output2 is Too Large).
16	If (Input1 is Too Low) then (Output3 is Too Large).
17	If (Input1 is Low) then (Output3 is Large).
18	If (Input1 is Average) then (Output3 is Average).
19	If (Input1 is Large) then (Output3 is Large).
20	If (Input1 is Too Large) then (Output3 is Too Large).
21	If (Input1 is Too Low) or (Input2 is Too Low) then (Output3 is too Large).
22	If (Input1 is Low) or (Input2 is Low) then (Output3 is Large).
23	If (Input1 is Average) or (Input2 is Average) then (Output3 is Average).
24	If (Input1 is Large) or (Input2 is Large) then (Output3 is Low).
25	If (Input1 is Too Large) or (Input2 is Too Large) then (Output3 is Too Low).

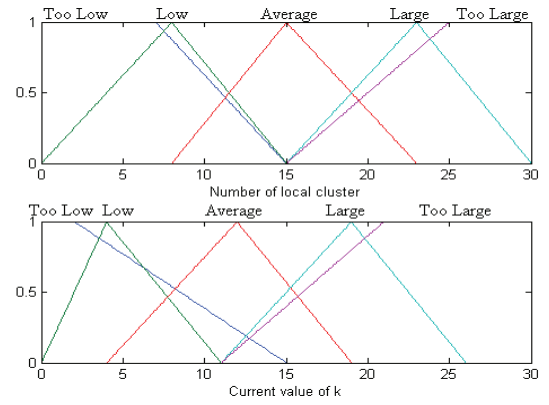


Figure 6. Output Membership Functions.

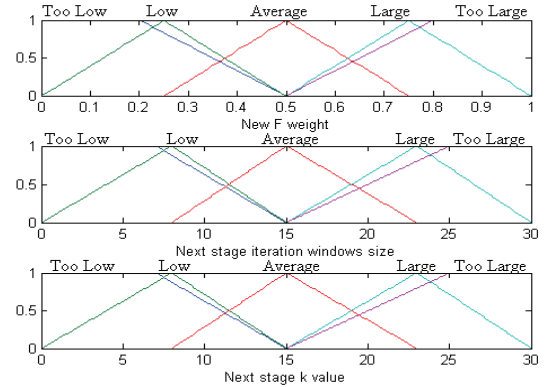


Figure 7. Input Membership Functions.

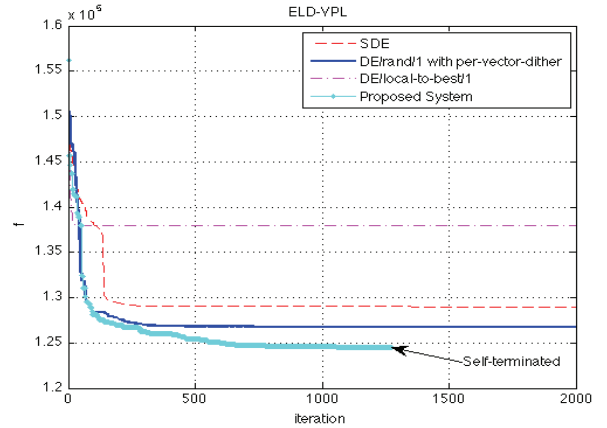


Figure 8. Fitness curve of ELD Problem

The dimension of ELD-VPL problem is 39. The result in terms of the cost value is shown in Table 2. All results shown are averaged data out of 50 trials. The convergence curves of different tested methods are shown in Fig. 8. We can see that the proposed system performs much better than the other methods in terms of convergence rate and cost value. The proposed system can offer the best (minimum) cost among the methods, which is \$124559.87. The proposed system is able to terminate itself at the 1376-th iteration.

The comparison between the Delta method with standard DE, Delta method with DE/local-to-best/1, Delta method with

DE/rand/1 with per-vector-dither and the proposed system with Iteration Window is shown in Fig. 9. The Delta method is a common method for terminating the searching process in evolution algorithms. In the experiments, if the fitness value of the current iteration is less than the 10^{-6} of the previous 10-th iteration, the searching process will be terminated. It shows that when using the Delta method, most of the algorithms will terminate very early with a very poor result, while the proposed system can terminate the searching process with much better solution quality.

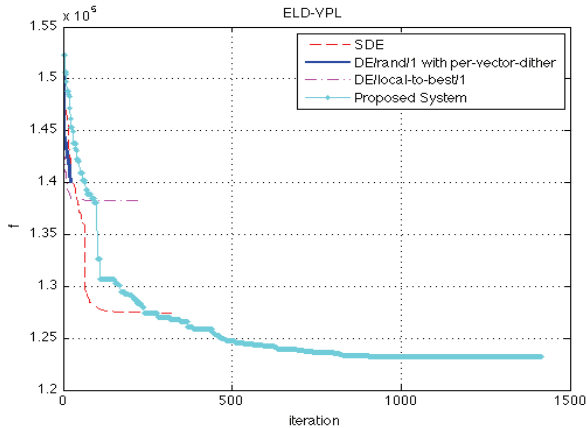


Figure 9. Comparison between the Delta method and the proposed system.

TABLE II. THE RESULT

	Proposed System	standard DE	DE/local-to-best/1	DE/rand/1 with per-vector-dither
Costs	124559.87	128982.54	137924.53	126753.64

V. CONCLUSION

In this paper, DE that incorporates fuzzy control and k -nearest neighbor algorithm has been proposed to handle the ELD-VPL Problem. To provide the self-terminating ability, the technique called Iteration Windows has been introduced to govern the number of iteration in each searching stage during the optimization. The size of the iteration windows is controlled by a fuzzy controller, which uses the information provided by the k -NN algorithm to analyze the population during the searching process. The controller keeps controlling the iteration windows until the end of the searching process. The wavelet based mutation process is embedded in the DE searching process to enhance the searching performance. The F weight of DE is also controlled by the fuzzy controller to further speed up the searching process. For the ELD-VPL problem, it is shown empirically that the proposed method outperforms significantly the conventional methods in terms of convergence speed and solution quality. The proposed system can terminate itself with a reasonable number of iteration.

ACKNOWLEDGMENT

The work described in this paper was substantially supported by a grant from The Hong Kong Polytechnic University (Project No. RP9L).

REFERENCES

- [1] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer "Self-Adapting Control Parameters in Differential Evolution: Comparative Study on Numerical Benchmark Problems," *IEEE Trans. Evol. Comput.*, vol. 10, no. 6, pp 646-657, Dec. 2006.
- [2] R. Storn and K. Price "Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, pp. 341–359, 1997.
- [3] U.K. Chakraborty (Ed.), *Advances in Differential Evolution*. Springer, Heidelberg, 2008.
- [4] S. Paterlini and T. Krink, "High performance clustering with differential evolution," in *Proc. IEEE Congress on Evolutionary Computation*, vol. 2, 2004, pp. 2004–2011.
- [5] J.H. van Sickel, K.Y. Lee, and J.S. Heo, "Differential evolution and its applications to power plant control," in *Proc. Intelligent Systems Applications to Power Systems 2007*, (ISAP 2007), 5-8 Nov. 2007, pp.1 – 6.
- [6] B. Babu and R. Angira, "Optimization of non-linear functions using evolutionary computation," in *Proc. 12th ISME International Conference on Mechanical Engineering*, India, 2001, pp. 153–157.
- [7] D. Greenhalgh and S. Marshall, "Convergence criteria for genetic algorithms," *SIAM J. Computer*, vol. 30, no. 1, pp. 260-282, 2000.
- [8] H. Aytug and G.J. Koehler, "New stopping criterion for genetic algorithms," *European Journal of Operational Research*, Vol. 126, Issue 3, pp. 662-674, 2000.
- [9] P.H. Chen and H.C. Chang, "Large-scale economic dispatch by genetic algorithms," *IEEE Trans. Power Syst.*, vol. 10, no. 1, pp. 117–124, Feb. 1995.
- [10] D. Bremner, E. Demaine, J. Erickson, J. Iacono, S. Langerman, P. Morin, and G. Toussaint, "Output-sensitive algorithms for computing nearest-neighbor decision boundaries," *Discrete and Computational Geometry*, vol. 33, no. 4, pp. 593-604, 2005.
- [11] T.M. Cover and P.E. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21-27, 1976.
- [12] G.T. Toussaint, "Geometric proximity graphs for improving nearest neighbor methods in instance-based learning and data mining," *International Journal of Computational Geometry and Applications*, vol. 15, no. 2, pp. 101-150, April 2005.
- [13] G.J. Klir and T.A. Folger, *Fuzzy Sets, Uncertainty, and Information*. Prentice-Hall, 1988.
- [14] M.M. Eftekhari and L.D. Marjanovic. "Application of fuzzy control in naturally ventilated buildings for summer conditions," *Energy and Buildings*, vol. 35, issue 7, Aug. 2003.
- [15] S. Zahan, C. Micheal, and S. Nikolakeas. *Fuzzy and Neuro-Fuzzy Systems in Medicine, chapter Fuzzy Expert Systems for Myocardial Ischemia Diagnosis*. CRC, 1999.
- [16] P. Bosc and O. Pivert, "Extending SQL retrieval feature for the handling of flexible queries," *Fuzzy Information Engineering: A Guided Tour of Applications*, pp 233 -251. Wiley, 1997.
- [17] B. Grabot, L. Geneste, and A. Dupeux, "Tuning of fuzzy rules for multiobjective scheduling," *Fuzzy Information Engineering: A Guided Tour of Applications*, pp. 695-703. Wiley, 1997.
- [18] Mamdani, E.H. and S. Assilian, "An experiment in linguistic synthesis with a fuzzy logic controller," *International Journal of Man-Machine Studies*, vol. 7, no. 1, pp. 1-13, 1975.
- [19] S.H. Ling, H.H.C. Iu, K.Y. Chan, H.K. Lam, C.W. Yeung, and F.H.F. Leung, "Hybrid particle swarm optimization with wavelet mutation and its industrial applications," *IEEE Trans. Syst., Man and Cybern., Part B: Cybernetics*, vol. 38, no. 3, pp. 743-763, Jun. 2008.
- [20] X. Yao and Y. Liu, "Evolutionary programming made faster," *IEEE Trans. Evolutionary Computation*, vol. 3, no. 2, pp. 82-102, July 1999.
- [21] S. Rahnamayan, H.R. Tizhoosh, and M.M.A. Salama, "Opposition-based differential evolution," *IEEE Trans. Evol. Comput.*, vol. 12, no. 1, pp 64-79, Feb. 2008.
- [22] Y. Ao, and H. Chi, "Experimental study on differential evolution strategies," in *Proc. Global Congress on Intelligent Systems.*, vol. 2, pp. 19-24, May. 2009.