Applied Soft Computing xxx (2017) xxx–xxx

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Learning enhanced differential evolution for tracking optimal decisions in dynamic power systems

Tao Zhu[a,b,c], Yingjie Hao[a], Wenjian Luo[d], Huansheng Ning[a,*]

[a] School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing, China
[b] Software School, University of South China, Hengyang, China
[c] Beijing Key Laboratory of Knowledge Engineering for Materials Science, Beijing, China
[d] School of Computer Science and Technology, University of Science and Technology of China, Hefei, China

ARTICLE INFO

Article history:
Received 16 January 2017
Received in revised form 13 July 2017
Accepted 16 July 2017
Available online xxx

Keywords:
Dynamic optimal power flow
Differential evolution
Learning
Evolutionary dynamic optimization

ABSTRACT

Optimal power flow (OPF) refers to the problem of optimizing the operating decisions such as electric power generation in power systems, which are always subjected to dynamic factors like bus loads. Conventionally, OPF in dynamic environments has been solved by static-oriented optimization methods based on the prediction of the dynamic factors. However, as the dynamics of modern power systems become more and more complex and difficult to predict, research interest of intelligent methods that track the optimal decisions of OPF has been grown recently. Devoted to this objective, a learning enhanced differential evolution (LEDE) is proposed in this paper. LEDE incorporates the idea of nearest-neighbor rule from the field of machine learning, with which decisions of the previous environments are retrieved continually to replace the newly generated individuals of differential evolution. A so-called elitism stochastic ranking strategy is also proposed, used in LEDE to handle constraints of OPF. Experiments are conducted on the dynamic IEEE 30-bus system and IEEE 118-bus system, and the results show the efficiency of LEDE in comparison with other algorithms.

© 2017 Elsevier B.V. All rights reserved.

1. Introduction

Optimal power flow (OPF) is the problem of determining the operating decisions such as electric power generation and transmission in power systems, so that some system performance indices could be optimized [1–3]. In the decision making systems of power systems, OPF is usually formulated mathematically as a constrained optimization problem [1]. At present, optimization techniques of OPF can be divided into two categories [2–4]: mathematical programming based methods and heuristic optimization algorithms [4], such as differential evolution (DE) [5] and particle swarm optimizer (PSO) [6].

In the real world, due to the dynamic changes of the bus loads, network connections, etc., the optimal decisions of OPF could change with time. Conventionally, OPF in dynamic environments (dynamic OPF) has been solved by static-oriented optimization methods based on prediction of load patterns and operation conditions, etc. However in recent years, with more new energy and electrical equipment being involved, the dynamics of the modern power systems are becoming more frequent, complex and difficult to predict [7,8]. Therefore, some researchers argue that intelligent methods that track the optimal decisions of dynamic OPF could be better than the traditional prediction-based approaches in the decision making systems of power systems [7–9].

However, tracking the optima of dynamic OPF is not an easy task. The mathematical programming based methods are fast, but easy to fall into the local optima, and usually require the problems to be continuous or differentiable [2], and hence may be unapplicable to dynamic OPF [7]. In contrast, many heuristic optimization algorithms have good global search ability and can be applied to various kinds of problems [3], but compared with the mathematical programming based methods, these techniques are time-consuming and thus few of them are used in real world power systems [4]. Therefore, how to improve the efficiency of heuristic optimization algorithms is important for the problem of OPF in dynamic environments.

Incorporating machine learning is a promising strategy to improve heuristic optimization methods in dynamic environments [10,11]. In his famous text book [12], Mitchell defined a machine-learning program as follows: a computer program is said to learn from experience *E* with respect to some class of tasks *T* if its per-

* Corresponding author.
E-mail addresses: tzhu@ustb.edu.cn (T. Zhu), hao_ying_jie@163.com (Y. Hao), wjluo@ustc.edu.cn (W. Luo), ninghuansheng@ustb.edu.cn (H. Ning).

http://dx.doi.org/10.1016/j.asoc.2017.07.037
1568-4946/© 2017 Elsevier B.V. All rights reserved.

Please cite this article in press as: T. Zhu, et al., Learning enhanced differential evolution for tracking optimal decisions in dynamic power systems, Appl. Soft Comput. J. (2017), http://dx.doi.org/10.1016/j.asoc.2017.07.037

formance at tasks in T improves with experience E. Similar idea underlies our recent work using memory to enhanced DE, which is devoted to dynamic OPF and termed as MEDE [9]. In MEDE, tracking optima of dynamic OPF is regarded as a series of static optimization tasks over time. When the power systems changes, MEDE reinitializes the DE population with the decisions from the previous experience according the environmental similarity between the current task and the previous tasks.

In this paper, a new algorithm to track optima of OPF in dynamic environments is proposed, termed as learning enhanced DE (LEDE). LEDE combines MEDE with the nearest-neighbor rule from the field of machine learning to reuse decisions of previous tasks. The underlying idea is as follows. In LEDE, the previous tasks and their best decisions are archived over time. Assuming the best decision would change smoothly if the task varies smoothly, the previous static tasks could be divided into clusters according to a certain similarity metric, such that the tasks in the same cluster have similar best decisions. For a new task, the nearest-neighbor rule assigns it and its nearest-neighbor task to the same cluster, and consequently, the best decision of the nearest-neighbor could be similar to that of the new task, and could be helpful to optimize the new task. Therefore, in the operators of LEDE, the nearest-neighbor rule is used to retrieve previous decisions to replace the newly generated individuals. In this way, the search of LEDE is adaptively adjusted by the experience obtained previously.

As for using which individuals to retrieve the neighboring decisions, two schemes are designed for LEDE and compared experimentally in the paper. For the first scheme, the newly generated donor individuals of DE are used to retrieve decisions, and these donor individuals are replaced by the retrieved decisions in DE operators. For the second scheme, the newly generated trial individuals of DE are used. In addition, to handle the constraints of dynamic OPF, an elitism stochastic ranking strategy is also proposed in LEDE.

In the experiments, we instantiate LEDE with the classical DE/rand/1 and a modern advanced DE variant, named DEGL (DE with global and local neighborhoods) [13], respectively. We compared the performance of the instantiated LEDE algorithms and MEDE algorithms on the dynamic IEEE 30-bus system and IEEE 118-bus system with dynamic bus loads, which was introduced in [7]. As an example, the optimization objective is to minimize the real power transmission loss, which is important to maintain the secure state of the power systems. The results show that the LEDE algorithms outperform the corresponding MEDE algorithms, and that LEDE is applicable to not only the classical DE but also some modern advanced DE variants.

The remainder of this paper is outlined as follows. Section 2 introduces the relevant background. The proposed LEDE algorithm is described in detail in Section 3. Section 4 describes the dynamic testing power systems and the experimental settings. The experimental results are presented and analyzed in Section 5. Section 6 concludes this paper and points out some future works.

2. Background

2.1. Dynamic OPF with minimizing real power transmission loss

Minimizing real power transmission loss is a typical objective of OPF, and is very important for maintaining the secure state of the power systems. The objective can be formulated by the following equation:

$$Ploss = \sum_{k=1}^{N_l} G_k[(T_k V_i)^2 + V_j^2 - 2T_k V_i V_j \cos(\theta_i - \theta_j)] \quad (1)$$

where N_l is the number of branches, G_k and T_k the conductance and the tap ratio of the transformer connected to branch k between buses i and j, respectively; V_i and V_j are the voltages magnitudes at buses i and j, θ_i and θ_j the voltage angles of buses i and j, respectively. To achieve this objective, the optimal settings of three types of the controllable variables should be find within the valid ranges, including the generator terminal voltage (V_{G_i}), compensation capacitor capacity (Q_{C_i}) and adjustable transformer ratio (T_i). Therefore a solution (the two terms, decision and solution, can be used interchangeably in this paper) of the OPF problem addressed in this paper can be formulated as follow, where N_G, N_T and N_C are the set of the involved generators, transformers and capacitors, respectively.

$$(V_{G_1} \cdots V_{G_{N_G}}, T_1 \cdots T_{N_T}, Q_{c_1} \cdots Q_{c_{N_C}})$$

Equality constraints to be met are the set of power flow equations.

$$\begin{aligned} 0 &= P_{G_i} - P_{D_i} - V_i \sum_{j=0}^{N_i} V_j (G_{ij} \cos(\theta_i - \theta_j) + B_{ij} \sin(\theta_i - \theta_j)), \quad i \in N_0 \\ 0 &= Q_{G_i} - Q_{D_i} - V_i \sum_{j=0}^{N_i} V_j (G_{ij} \sin(\theta_i - \theta_j) - B_{ij} \cos(\theta_i - \theta_j)), \ i \in N_{PQ} \end{aligned} \quad (2)$$

where P_{D_i} and P_{G_i} are the demanded active power (bus loads) and injected active power at bus i, G_{ij} and B_{ij} the transfer conductance and susceptance between bus i and j, Q_{D_i} and Q_{G_i} the demanded and injected reactive power at bus i, N_i the set of buses adjacent to bus i, N_0 the set of total buses excluding slack bus, N_{PQ} the set of PQ buses.

In addition, the following in equations should be met, which are about the value ranges of the control variables and the intermediate state variables. The inequality constraints of control variables:

$$\begin{cases} Q_{C_i}^{\min} \le Q_{C_i} \le Q_{C_i}^{\max}, \ i \in N_C \\ T_k^{\min} \le T_k \le T_k^{\max}, \ k \in N_T \\ V_{G_i}^{\min} \le V_{G_i} \le V_{G_i}^{\max}, \ i \in N_{PV} \end{cases} \quad (3)$$

where Q_{C_i} and V_{G_i} are the reactive power compensation and the generator voltages at bus i. N_{PV} are the set of numbers of the voltage-controlled (PV) buses. The inequality constraints of state variables are:

$$\begin{cases} \sqrt{P_l^2 + Q_l^2} \le S_l^{\max}, \ l \in N_l \\ Q_{G_i}^{\min} \le Q_{G_i} \le Q_{G_i}^{\max}, \ i \in N_G \\ V_{L,i}^{\min} \le V_{L,i} \le V_{L,i}^{\max}, \ i \in N_{PQ} \end{cases} \quad (4)$$

where $V_{L,i}$ is the voltage magnitude at load bus i. P_l and Q_l are the real and reactive power flow in line l. $S_l^{\max}$ is the maximum limit of the apparent power in branch l. And N_G, N_l are the set of generator buses and branches, respectively.

In [7], dynamic OPF is simulated by varying the loads of some selected buses with a certain probability.

$$P_{D_i} = P_{D_i}^0 + d_i(t), \ i \in cbus \quad (5)$$

where $P_{D_i}^0$ is the basic loads at bus i, $cbus$ is the set of the selected dynamic buses. The variable $d_i(t)$ is as follows, where α is a uniformly random variable.

$$d_i(t) = \alpha P_{D_i}, \alpha \in [-0.2, 0.2] \quad (6)$$

Please cite this article in press as: T. Zhu, et al., Learning enhanced differential evolution for tracking optimal decisions in dynamic power systems, Appl. Soft Comput. J. (2017), http://dx.doi.org/10.1016/j.asoc.2017.07.037

2.2. Differential evolution for OPF

2.2.1. Differential evolution

Differential evolution (DE) is one of the most powerful stochastic real-parameter optimization algorithms in current use, which was proposed by Storn and Price [5], and has a wide range of applications [14,15]. An individual of DE is a real vector, which typically is a candidate solution of the optimization problem. DE works through a simple cycle of several operators, including mutation, crossover and selection. The DE population individuals are termed *target* individuals. At each generation, the mutation operator produces a so-called *donor* individual $\mathbf{v}_i$ for each target individual $\mathbf{x}_i$, where i is the index in the population. Then, with $\mathbf{v}_i$, the crossover operator generates a so-called *trial* individual $\mathbf{u}_i$ for $\mathbf{x}_i$. Finally, the selection operator chooses the winners into the DE population of the next generation between each pair of target and trial individuals.

Different DE variants could have different operators [5,13,16,14]. For examples, the mutation operator of the classical DE/Rand/1 can be described with Eq. (7) [5], while the mutation operator of DEGL [13] can be described with Eq. (8).

$$\mathrm{v_i} = \mathrm{x_{r1}} + F(\mathrm{x_{r2}} - \mathrm{x_{r3}}) \tag{7}$$

$$\mathrm{v_i} = \mathrm{x_i} + \omega F(\mathrm{x_{gbest}} - \mathrm{x_i} + \mathrm{x_{r1}} - \mathrm{x_{r2}}) + (1-\omega) F(\mathrm{x_{nbest}} - \mathrm{x_i} + \mathrm{x_p} - \mathrm{x_q}) \tag{8}$$

In the above two equations, F is a real number controlling the scaling of the individual difference. $\mathbf{x}_{r1}, \mathbf{x}_{r2}, \mathbf{x}_{r3}$ are distinct individuals randomly selected from the population, and are also different from $\mathbf{x}_i$. $\mathbf{x}_{gbest}$ is the best of all the target individuals, while $\mathbf{x}_{nbest}$ is the best among the neighbors of $\mathbf{x}_i$. $\mathbf{x}_p$ and $\mathbf{x}_q$ are randomly chosen distinct neighbors of $\mathbf{x}_i$. ω is a real number ranging from 0 to 1. Please refer to [13] for more details of DEGL.

For both DE/Rand/1 and DEGL, the crossover operator generates a trial individual $\mathbf{u}_i$ for each $\mathbf{x}_i$ by recombining $\mathbf{x}_i$ and $\mathbf{v}_i$ as follows.

$$u_{i,j} = \begin{cases} v_{i,j}, & \text{if } rand \le CR \text{ or } j = j_{rand} \\ x_{i,j}, & otherwise \end{cases} \tag{9}$$

where $u_{i,j}$, $v_{i,j}$, $x_{i,j}$ are the jth element of the $\mathbf{u}_i$, $\mathbf{v}_i$ and $\mathbf{x}_i$. CR represents the crossover probability, j_{rand} is a randomly selected dimension index, which ensures that $\mathbf{u}_i$ gets at least one component from $\mathbf{v}_i$. j_{rand} is instantiated once for each individual per generation.

2.2.2. Individual representation

In [17], applying DE to OPF is investigated. As to DE for the OPF problem described above, an individual consists of generator voltages, the transformer tap ratios, and the reactive power compensation of the shunt reactive power compensators, which is expressed as follows:

$$(V_{G_1} \cdots V_{G_{N_G}}, T_1 \cdots T_{N_T}, Q_{c_1} \cdots Q_{c_{N_C}})$$

The domain of T_i is a set of discrete values in $[T_i^{\min}, T_i^{\max}]$ with step size of ΔT_i. Q_{C_i} is integer. The operators of DE would produce invalid values for T_i and Q_{C_i}. As in [17], the invalid value of Q_{C_i} is converted to the nearest integer no greater than itself, and for T_i, it is modified as follows.

$$T_i = T_i^{\min} + \left\lfloor (T_i - T_i^{\min})/\Delta T_i \right\rfloor \times \Delta T_i$$

2.2.3. Handling constraints

As in [17], the equality constraints in Eq. (2) are met by the load-flow solutions. The inequality constraints of the control variables in Eq. (3) are met by confining the search space of DE. The constraints of the state variables in Eq. (4) are handled in DE with *stochastic ranking*, which is discussed in [17]. In this stochastic selection operator, the target individual $\mathbf{x}_i$ and its trial individual $\mathbf{u}_i$ are compared based on the objective value f alone or overall constraint violation g alone as randomly determined as Algorithm 1. The overall constraint violation is a weighted mean of all the constraints in Eq. (4) [17].

Algorithm 1. Stochastic ranking in DE selection operator.

```
1:  x_i(g+1) ⟵ x_i
2:  if (g(x_i) = g(u_i) = 0) ‖ rand < p_f) then
3:    if f(u_i) ≤ f(x_i) then    // in minimization problem
4:  x_i(g+1) ⟵ u_i;
5:    end if
6:  else
7:    if g(u_i) ≤ g(x_i) then
8:  x_i(g+1) ⟵ u_i;
9:    end if
10: end if
```

2.3. The memory enhanced DE

The memory enhanced differential evolution, proposed in [9], improves DE by using memory to reinitialize DE population once the power system changes. The memory is composed of pairs in form of ⟨**P**, **u**⟩, where **P** is bus loads and **u** the corresponding best solution obtained by DE. Specifically, MEDE employs the so-called *similar retrieval scheme* to select memory solutions the memory and *mean-based immigrants mechanism* to generate memory immigrants, and replaces the worst individuals in the population with the selected and generated solutions.

Algorithm 2. MEDEChangeHandle(pop).

```
1: Add ⟨P, u⟩ into memory, where P and u is the bus loads and the best
   individual of the last task
2: Select K solutions from the memory with the similar retrieval scheme
3: Generate η1 + η2 − K solutions with the mean-based immigrants
   scheme
4: Replace the worst K individuals in pop in terms of the fitness in the last
   task with the selected and generated solutions
5: Evaluate the fitness of the individuals in pop
```

Similar Retrieval Scheme: Let $\mathbf{P}^c$ denote the current bus loads. Given a distance parameter γ, pick out all the pairs that the distance from **P** to $\mathbf{P}^c$ is less than γ. Let N denotes the number of these qualified entries. Rank these pairs according to the distance to $\mathbf{P}^c$. The least one is ranked 0, the next one ranked 1, and so forth. Then, based on the ranks, a selection probability is assigned to each pair according to the following equation.

$$Prob(n) = \frac{2\eta_1}{N+1}\left(1 - \frac{n}{N}\right) \tag{10}$$

where $Prob(n)$ is the selection probability of the pair ranked n. After the selection probabilities are given by the above equation, η_1 pairs are selected from the N entries according to the stochastic universal sampling algorithm [18], and then, the related solutions are retrieved for reinitializing DE population. Note that if $N < \eta_1$, only N distinct solutions will be retrieved.

Mean-Based Immigrants scheme generate an immigrant **x** as follows.

$$\mathbf{x}_d = mean_d + r_d * \mathrm{randn} \tag{11}$$

$$mean_d = \frac{1}{|mem|} \sum_{<\mathbf{P}_D, \mathbf{u}> \in mem} \mathbf{u}_d \tag{12}$$

$$r_d = std\{\mathbf{u}_d \mid <\mathbf{P}_D, \mathbf{u}> \in mem\} \tag{13}$$

where $\mathbf{u}_d$ is the dth element of **u**; *mem* and $|mem|$ is the memory set and its size; *std* means the standard deviation; randn is a standard normally distributed random variable. The pseudo codes for MEDE

Please cite this article in press as: T. Zhu, et al., Learning enhanced differential evolution for tracking optimal decisions in dynamic power systems, Appl. Soft Comput. J. (2017), http://dx.doi.org/10.1016/j.asoc.2017.07.037

to reinitialize DE population once the power system changes are given in Algorithm 2.

3. Learning-enhanced DE algorithm for tracking optima of dynamic power systems

3.1. Incorporate the nearest-neighbor rule

The nearest-neighbor algorithm has conceptual and computational simplicity. Let $D_n = \{x_1, \ldots, x_n\}$ denote a set of n classified instances, and $x_i \in D_n$ be the instance nearest to a test instance x. Then the nearest-neighbor rule assigns x to the class associated with x_i. The nearest-neighbor rule is a sub-optimal procedure; its use will usually lead to an error rate greater than the minimum possible, the Bayes rate. It has been proven that, however, with an unlimited number of instances the error rate is never worse than twice the Bayes rate [19].

Tracking optima in dynamic power systems can be regarded as a series of static optimization tasks over time with different bus loads, and these tasks can be stored as the data set. However, the proposed LEDE does not use the nearest-neighbor rule to classify tasks, but to select, for the new task, useful decisions of previous tasks, so that the new task could be solved by adapting these decisions rather than from scratch. In practice, the variation of the bus loads is always within a certain range or even periodical, and thus, some previous tasks could be very similar. Assuming the best decision changes smoothly if the task varies smoothly, these previous static tasks could be divided into clusters according to a certain similarity metric, such that the tasks in the same cluster have similar best decisions. For the new task, the nearest-neighbor rule assigns it and its nearest-neighbor task into the same cluster, and consequently, the best decision of the nearest-neighbor could be similar to that of the new task, and could be helpful to optimize the new task.

A major issue of using the nearest-neighbor rule is how to measure the task similarity. The similar retrieval scheme of MEDE estimates the task similarity according the Euclidean distance of the loads. This measure sounds reasonable, but lacks solid ground yet. As a complement to MEDE, LEDE uses solution similarity to estimate the task similarity. The underlying assumption is that "similar solutions solve similar problems", which is supported and validated in [20]. In [20], hamming distance serves as the similarity metric among solutions. However, neither hamming distance nor Euclidean distance is suitable for our problem, because for the solutions, the search ranges of different dimensions are in real space and differs greatly. For example, in our experiments on dynamic IEEE 30-bus system, the search range of the generator voltage is from 0.95 to 1.1, while shunt reactive power compensator is from 1 to 30. Euclidean distance biases towards those dimensions that have larger search range. In this paper, we propose to use normalized Euclidean distance as the similarity metric, which is formulated as follow.

$$\mathrm{Sim}(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1} \left(\frac{\mathbf{x}_i - \mathbf{y}_i}{U_i - L_i} \right)^2} \tag{14}$$

where $\mathbf{x}$, $\mathbf{y}$ are two solutions, and U_i, L_i are the upper and lower bounds of the ith dimension, respectively.

For the previous tasks, the best individuals stored in the data set can be used to calculate task similarity. For the current task being solved, the best solution is being updated over time, and therefore, we proposed to use newly generated individuals (donor individuals or trial individuals) to calculate task similarity and retrieve solutions of the nearest-neighbor tasks. Specifically, when the underlying DE generates a new donor (trial) individual, LEDE uses the nearest-neighbor rule to retrieve a solution of a previous task from the data set. If the retrieved solution has never been evaluated for the current task, it replaces the donor (trial) individual in the crossover (selection) operator of DE. This mechanism is adaptive, because which and how many solutions will be retrieved is not fixed, but depends on the underlying DE's search behavior on a specific task.

3.2. Elitism stochastic ranking for handling constraints

Stochastic ranking is a successful constraint handling technique, which was originally proposed with (μ, λ)-ES algorithm [21], where $\mu/\lambda \approx 1/7$. At every generation, (μ, λ)-ES generates λ individual from μ old individuals. Then the λ individuals is stochastically ranked with a bubble-sort-like procedure, and finally, the top μ ranked individuals are selected as the parents of the next generation. In contrast, the selection operator of DE is the competition between only two individuals: a target individual and its trial individual. Due to this difference, although applying stochastic ranking to DE as in Algorithm 1 is effective [17], it leads to a problem as follow.

Algorithm 1 loses better feasible solutions. Suppose $\mathbf{u}_i$ is the trial individual generated for a target individual $\mathbf{x}_i$, and $f(\mathbf{u}_i) < f(\mathbf{x}_i)$, $g(\mathbf{u}_i) > 0$, $g(\mathbf{x}_i) = 0$. When comparing $\mathbf{x}_i$ and $\mathbf{u}_i$ with stochastic ranking, the probability of losing $\mathbf{x}_i$ is p_f, which is typically around 0.475. The worst possible case is that all the feasible elite solutions in the population could be lost within several generations, which would degenerate the optimization progress.

To handle this problem, we propose a so-called *Elitism Stochastic Ranking* strategy for DE to handle constraints of OPF. This strategy ensures that the best target individual competes with its trial individual based on the overall constraint if either of them is infeasible. The best target individual is picked out at the end of every generation in terms of the comparison rule in [22]. With this rule, when the two individual $\mathbf{x}_1$, $\mathbf{x}_2$ are compared, $\mathbf{x}_1$ is better if

- $g(\mathbf{x}_1) = 0$, $g(\mathbf{x}_2) > 0$ or
- $g(\mathbf{x}_1) = g(\mathbf{x}_2) = 0$, $f(\mathbf{x}_1) < f(\mathbf{x}_2)$ or
- $g(\mathbf{x}_2) > g(\mathbf{x}_1) > 0$.

3.3. The pseudo codes

Algorithm 3. Nearest-neighbor ($\mathbf{x}$, *ds*).

```
y ⟵ the nearest-neighbor of x in ds with the similarity metric in Eq. (14);
if y is not marked then
    x ⟵ y;
    Mark y
end if
Return x
```

Algorithm 4. Learning enhanced DE.

```
g ⟵ 1;
Initialize the population pop(g) and other relevant parameters; ds is an empty KD-tree
ibest is the index of the best individual in pop(g)
while the stop criteria is not satisfied do
    if A change of load is detected then
MEDEChangeHandle(pop)
Insert the best solution of the last task into ds
Remove the marks of all the solutions in ds
    end if
    for each x_i ∈ pop(g) do
Generate the donor individual v_i with a mutation operator
if Using donor individuals then    // Strategy of LEd
    v_i ⟵ nearest-neighbor(v_i, ds)
end if
Generate the trial individual u_i with a crossover operator
if Using trial individuals then    // Strategy of LEt
    u_i ⟵ nearest-neighbor(u_i, ds)
```

Please cite this article in press as: T. Zhu, et al., Learning enhanced differential evolution for tracking optimal decisions in dynamic power systems, Appl. Soft Comput. J. (2017), http://dx.doi.org/10.1016/j.asoc.2017.07.037

```
18:    end if
19:    Evaluate f(u_i) and g(u_i);
20:    x_i(g+1) ⟵ x_i
21:    if (g(x_i)=0 & g(u_i)=0) || ((i ≠ ibest) & rand < p_f) then    // Elitism
       Stochastic Ranking
22:      if f(u_i) ≤ f(x_i) then
23:    x_i(g+1) ⟵ u_i;
24:      end if
25:    else
26:      if g(u_i) ≤ g(x_i) then
27:    x_i(g+1) ⟵ u_i;
28:      end if
29:    end if
30:      end for
31:      g ⟵ g+1
32:      Update ibest
33:    end while
```

The pseudo codes of retrieving stored individuals from the data set with the nearest-neighbor rule are given in Algorithm 3. Every stored individual will be retrieved at most one time for a certain static task, therefore after a stored individual is retrieved, it is marked. Once the environment changes, all the marks will be removed (Algorithm 4, Line 8), so that they can be reused again for the new task. Please note Line 21 in Algorithm 3, which ensures that if the $\mathbf{x}_i$ is the best in the $pop(g)$ and $g(\mathbf{u}_i) > 0$, the selection would based on the overall constraint violation.

The pseudo codes of the proposed LEDE are given in Algorithm 4. Two schemes are compared in the experiments. The first scheme uses the newly generated donor individuals to retrieve previous best solutions, and this scheme is denoted as LEd (Line 12–14). In other words, with LEd, new solutions of the current task are generated by hybridizing the retrieved previous individuals with the target individuals through the crossover operator. The second scheme uses the newly generated trial individuals to retrieve the previous best solutions, which is denoted as LEt (Line 16–18). With LEt, the retrieved individuals are directly evaluated as the new solutions of the current task.

Many DE variants work through the same cycle of mutation, crossover and selection, generating donor individuals, trial individual and target individuals in turn. It can be inferred from the pseudo codes in Algorithm 4 that LEDE can be applied to these DE variants easily. Users can simply insert the codes of LEd, LEt and Elitism Stochastic Ranking in the programs of these DE variants, after the mutation, crossover and selection operators, respectively.

The proposed LEDE is a new optimization algorithm for tracking optima of OPF in dynamic environments. In MEDE, the previous solutions are used to reinitialize the DE population when the power system changes, while in LEDE the learning mechanism takes effect in the entire search progress after the population initialization. In addition, although the idea of using solution similarity to measure task similarity comes from [20], LEDE injects previous individuals into DE's population using the nearest-neighbor rule adaptively, while in [20] individuals are injected into the population of genetic algorithm periodically at fixed time points.

It should be pointed out that the usage of nearest neighbors in combination with differential evolution is not new [23–26]. Reference [23] and [24] propose to take advantage of DE to optimize the positioning adjustment problem in Nearest neighbor classification (NNC). Their objective is to enhance the efficiency of NNC, while the objective of our paper is to enhance the efficiency of DE. Both Reference [25] and [26] uses KNN as a fitness predictor to save expensive fitness function calls and thus enhance DE for STATIC optimization problems. In our paper, LEDE is proposed for DYNAMIC OPF problems, and KNN is used to retrieve previous solutions which could be perform well in the current state of the DYNAMIC power systems, which is a new method.

Time efficiency is an important concern in practice. Besides the operations of DE, the extra time complexity caused by LEDE mainly includes retrieving individuals from data set ds and storing the best individuals into ds. The data set ds can be organized in a KD-tree. The average complexity of inserting a solution into a balanced KD-Tree of size n is $O(\log n)$, and the average complexity of finding the nearest-neighbor solution in a balanced KD-tree is $O(\log n)$ too [27]. Suppose there is C system changes and N fitness evaluations during the optimization period, the maximum size of ds is C, and the time complexity of LEDE is $O(C\log(C)) + O(N\log(C))$.

Table 1
Features of the IEEE 30-bus and IEEE 118-bus systems.

Features	30-bus	118-bus
♯ buses	30	118
♯ generators	6	54
♯ transformers	4	9
♯ branches	41	186
♯ equality constraints	60	236
♯ inequality constraints	125	572
♯ control variables	12	75
♯ discrete variables	6	21

4. Dynamic test environment simulation

4.1. Test systems

In order to evaluate the performance of the proposed LEDE, the experiments are carried on two test powers systems: the IEEE 30-bus system and the IEEE 118-bus system, which are from the real world and the main features are listed in Table 1. In [7], the authors modified them by introducing several dynamic buses with varying loads buses, which are depicted in Figs. 1 and 2, respectively. For the dynamic IEEE 30-bus system, $cbus = \{7, 19, 21, 24, 30\}$, and for the dynamic IEEE 118-bus system, $cbus = \{11, 45, 60, 78, 82\}$. The load changes are simulated with load variations applied on the selected variable buses randomly with a given probability at every generation, where the probability is denoted as τ. Three different values of τ are used to simulate the load variations at slightly ($\tau = 0.01$), moderately ($\tau = 0.05$) and severely ($\tau = 0.2$) changing levels respectively. Implementation of the power systems in MATPOWER [28] is adopted in the experiments. All the details of the IEEE 30-bus system and the IEEE 118-bus system can be found in MATPOWER [28].

4.2. Performance measurements

To evaluate the performance of the algorithms, four performance measures are adopted in the experiments [9].

Tracking error refers to the difference between the decisions obtained the algorithm during the optimization and the reference solutions. Reference solutions are the best decisions of each static task obtained offline in advance by a certain algorithm. Because it is difficult to obtain the real optimal OPF decisions, the reference solutions are used to serve as the approximate optimal decisions. Track error is formulated as follow.

$$\mathrm{TE} = \sum_{g=1}^{G} \frac{F_{alg}(g) - F_{ref}(g)}{G} \tag{15}$$

where $F_{ref}(t)$ is the objective value of the reference solutions at generation g, while $F_{alg}(g)$ is the least transmission power loss obtained by the algorithm at generation g; G is the number of generations in a run. If the solutions obtained by the algorithms are better than reference solutions, the track error would be negative.

Feasible time ratio [9] is the ratio of time when feasible solution is find to the whole optimization time span, of which the expression is as follows.

Please cite this article in press as: T. Zhu, et al., Learning enhanced differential evolution for tracking optimal decisions in dynamic power systems, Appl. Soft Comput. J. (2017), http://dx.doi.org/10.1016/j.asoc.2017.07.037

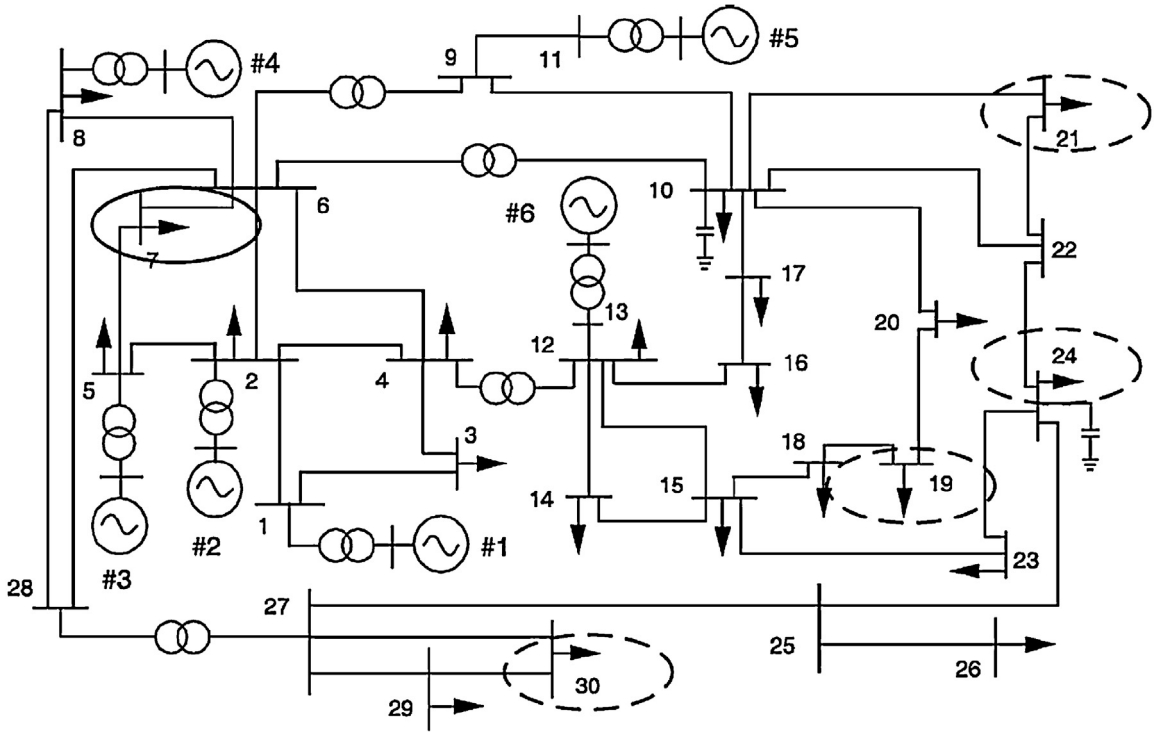

Fig. 1. The dynamic IEEE 30-bus system with varying load buses [7].

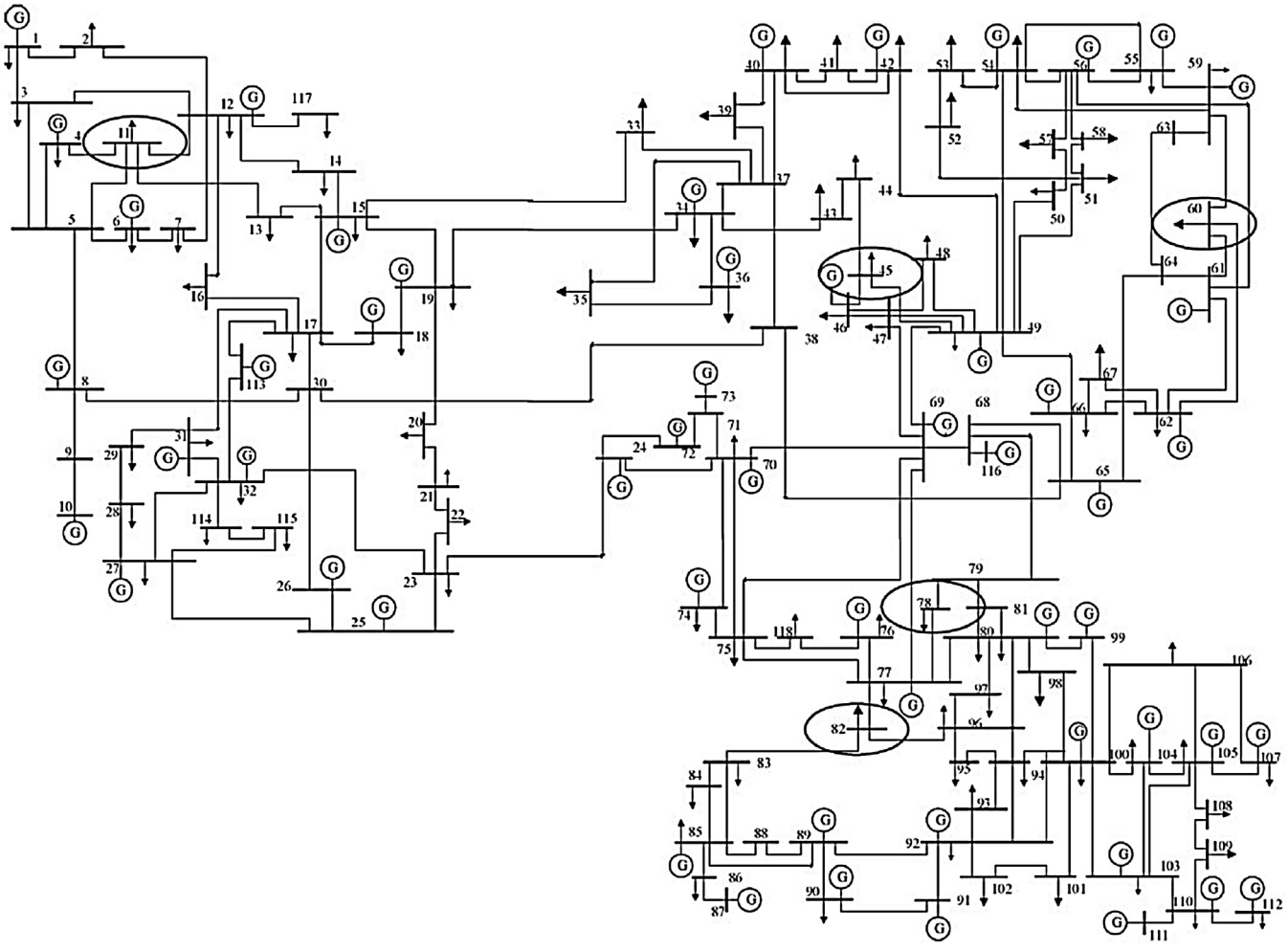

Fig. 2. The dynamic IEEE 118-bus system with varying load buses [7].

Please cite this article in press as: T. Zhu, et al., Learning enhanced differential evolution for tracking optimal decisions in dynamic power systems, Appl. Soft Comput. J. (2017), http://dx.doi.org/10.1016/j.asoc.2017.07.037

$$\mathrm{FTR} = \frac{\sum_{g=1}^{G} fs(g)}{G} \tag{16}$$

where the value of $fs(g)$ is 1 if the algorithm has obtained a feasible solution at generation g, otherwise the value is 0.

The tracking speed reflects how quick the algorithm can find obtain a satisfying solution after the load changes in terms of the average required fitness evaluations, which is described as follows:

$$\mathrm{TS} = \frac{\sum_{i=1}^{N_e} L_i}{N_e} \tag{17}$$

In Eq. (17), N_e is the total number of environments (load changes), and L_i is the number of fitness evaluations until a new feasible and satisfying solution is obtained in the *ith* environment. Whether a solution $\mathbf{x}$ is satisfying or not is determined by Eq. (18).

$$S(\mathbf{x}) = \begin{cases} true; & if \quad \dfrac{f(\mathbf{x}) - F_{ref}}{F_{ref}} < \eta\% \\ false; & if \quad otherwise \end{cases} \tag{18}$$

In the above equation, F_{ref} represents the objective value of the reference solutions at the time instant of $\mathbf{x}$, $f(\mathbf{x})$ objective value of $\mathbf{x}$. Parameter η is usually a small real number.

It is probable for the algorithm to fail to find a satisfying solution in some environments, then *tracking success ratio* is used to evaluate the ratio when the algorithm succeeds, which is formulates as follows.

$$\mathrm{TSR} = \frac{\sum_{e=1}^{N_e} s(e)}{N_e} \tag{19}$$

where N_e means the number of environments in a run. If the algorithm succeeds to obtained a satisfying solution in the eth environment, the value of $s(e)$ is 1, otherwise 0. Please note that tracking success ratio and tracking speed should be used together, and tracking success ratio is the dominant metric.

4.3. Algorithm and parameter settings

In the experiments, we instantiate LEDE and MEDE with DE/Rand/1 (denoted as DER1) and DEGL. The instance algorithms are denoted as LEd-DER1, LEt-DER1, LEd-DEGL, LEt-DEGL, ME-DER1 and ME-DEGL, respectively. Besides, as the benchmark algorithms, the DER1 with randomly population reinitialization is also compared, denoted as RR-DER1. For RR-DER1, once the power system changes, the top 1/4 individuals of last population are reevaluated and the rest are replaced by uniformly randomly generated individuals in the search space.

For all the algorithms, the population size is 20, $CR = 0.9$, $F = 0.7$ and $pf = 0.475$. The parameter ω of DEGL is tuned with the Self-Adaptive scheme in [13]. As to the parameters of the *Similar Retrieval Scheme* and *Mean-Based Immigrants scheme*, γ is set to 0.08, $\eta_1 = 5$, and $\eta_2 = 5$.

For each test case, 15 independent runs are conducted, and the average results and the standard deviation are used for comparison. For each run, there are totally 40,000 fitness evaluations (2000 generations). Both MEDE and LEDE have problem of cold start. Therefore, as in [9], two initialization methods are compared. For the first one, the memory (data set) is initially empty. For the second one, the memory (data set) is initialized with 200 entries obtained in advance, and the algorithms are denoted in the form of "*-(200)", for example, LEd-DEGL(200). The former simulates the case of applying the algorithms to a totally new power system. The latter simulates the case that previous optimization information about the power system are available in the past optimization logs.

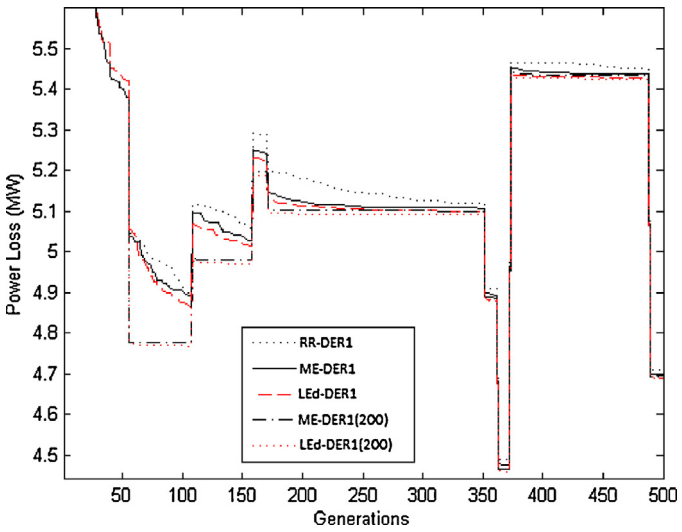

Fig. 3. Online power loss of the algorithms on the dynamic IEEE 30-bus systems with $\tau = 0.01$.

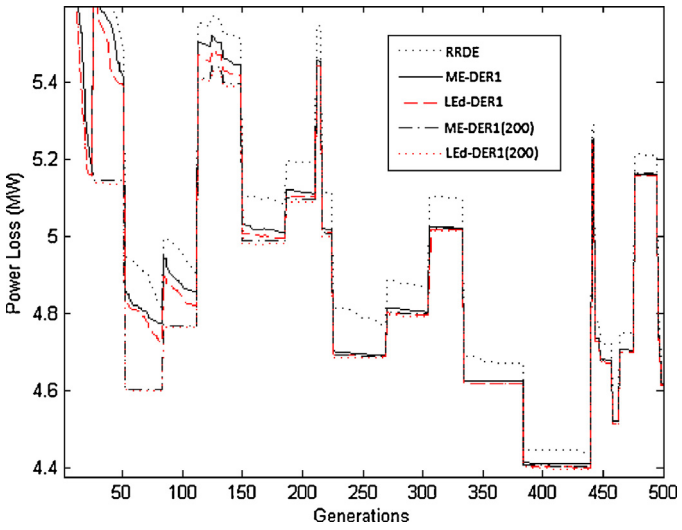

Fig. 4. Online power loss of the algorithms on the dynamic IEEE 30-bus systems with $\tau = 0.05$.

5. Experimental results and analysis

5.1. Online performance analysis

To investigate the online performance of the algorithms, Figs. 3 and 4 illustrate the best power loss achieved by some algorithms at each generation on the dynamic IEEE 30-bus system with $\tau = 0.01$ and 0.05 respectively. The data in the two figures are the average values over 15 runs. A sudden change in the lines indicates that the bus loads change at that time. In this group of experiments, LEDE uses donor individuals. From the figures, we can find the RR-DER1's performance is much worse than the other algorithms, which indicates that the learning strategies used in MEDE and LEDE can enhance DER1's performance significantly.

In the first 350 generations of Fig. 3, there exists a clear difference between the lines of ME-DER1 and LEd-DER1. LEd-DER1 is faster in finding a better solution than ME-DER1. Afterwards, the best power loss achieved by LEd-DER1 and ME-DER1 changes slightly between two changing time points. The reason could be that the best solutions found by LEd-DER1 and ME-DER1 have approached closely to the real optimal solutions. Although the dif-

Please cite this article in press as: T. Zhu, et al., Learning enhanced differential evolution for tracking optimal decisions in dynamic power systems, Appl. Soft Comput. J. (2017), http://dx.doi.org/10.1016/j.asoc.2017.07.037

Table 2
TE and FTR of the algorithms on dynamic IEEE 30-bus systems.

$\tau = 0.01$	TE (MW)	FTR	$\tau = 0.01$	TE (MW)	FTR
ME-DER1	5.90E−2 ± 1.55E−2	9.93E−1 ± 2.87E−3	ME-DEGL	2.68E−2 ± 7.11E−3	9.96E−1 ± 1.60E−3
LEd-DER1	4.81E−2 ± 1.50E−2	9.95E−1 ± 2.71E−3	LEd-DEGL	2.28E−2 ± 6.97E−3	9.97E−1 ± 1.41E−3
LEt-DER1	5.21E−2 ± 1.58E−2	9.95E−1 ± 2.71E−3	LEt-DEGL	2.48E−2 ± 5.03E−3	9.97E−1 ± 1.03E−3
ME-DER1(200)	5.09E−2 ± 1.42E−2	9.93E−1 ± 2.87E−3	ME-DEGL(200)	1.48E−2 ± 4.09E−3	**9.98E−1 ± 6.76E−4**
LEd-DER1(200)	**3.91E−2 ± 1.28E−2**	**9.95E−1 ± 2.71E−3**	LEd-DEGL(200)	1.40E−2 ± 3.93E−3	**9.98E−1 ± 6.76E−4**
LEt-DER1(200)	4.38E−2 ± 1.28E−2	9.95E−1 ± 2.71E−3	LEt-DEGL(200)	**1.39E−2 ± 4.01E−3**	**9.98E−1 ± 6.76E−4**
$\tau = 0.05$	TE (MW)	FTR	$\tau = 0.05$	TE (MW)	FTR
ME-DER1	5.25E−2 ± 9.52E−3	9.95E−1 ± 1.34E−3	ME-DEGL	2.78E−2 ± 7.62E−3	9.97E−1 ± 1.16E−3
LEd-DER1	4.79E−2 ± 1.44E−2	9.95E−1 ± 2.42E−3	LEd-DEGL	2.66E−2 ± 9.03E−3	9.97E−1 ± 1.11E−3
LEt-DER1	5.12E−2 ± 1.40E−2	9.95E−1 ± 2.42E−3	LEt-DEGL	2.73E−2 ± 8.73E−3	9.97E−1 ± 1.05E−3
ME-DER1(200)	4.21E−2 ± 6.19E−3	**9.95E−1 ± 1.22E−3**	ME-DEGL(200)	1.53E−2 ± 4.38E−3	**9.98E−1 ± 6.45E−4**
LEd-DER1(200)	**3.60E−2 ± 1.15E−2**	**9.95E−1 ± 2.45E−3**	LEd-DEGL(200)	1.42E−2 ± 4.30E−3	**9.98E−1 ± 6.45E−4**
LEt-DER1(200)	4.07E−2 ± 1.15E−2	9.95E−1 ± 2.45E−3	LEt-DEGL(200)	**1.41E−2 ± 4.33E−3**	**9.98E−1 ± 6.45E−4**
$\tau = 0.2$	TE (MW)	FTR	$\tau = 0.2$	TE (MW)	FTR
ME-DER1	5.03E−2 ± 8.37E−3	9.94E−1 ± 1.59E−3	ME-DEGL	2.72E−2 ± 8.22E−3	9.98E−1 ± 1.10E−3
LEd-DER1	4.70E−2 ± 1.38E−2	9.94E−1 ± 2.27E−3	LEd-DEGL	2.63E−2 ± 5.21E−3	9.97E−1 ± 1.01E−3
LEt-DER1	4.82E−2 ± 9.85E−3	9.94E−1 ± 2.20E−3	LEt-DEGL	2.50E−2 ± 6.89E−3	9.97E−1 ± 1.38E−3
ME-DER1(200)	3.01E−2 ± 3.37E−3	9.95E−1 ± 7.56E−4	ME-DEGL(200)	1.46E−2 ± 4.25E−3	**9.98E−1 ± 9.61E−4**
LEd-DER1(200)	**2.48E−2 ± 5.12E−3**	**9.96E−1 ± 1.26E−3**	LEd-DEGL(200)	1.37E−2 ± 4.22E−3	9.98E−1 ± 9.67E−4
LEt-DER1(200)	2.74E−2 ± 4.92E−3	**9.96E−1 ± 1.26E−3**	LEt-DEGL(200)	**1.36E−2 ± 4.02E−3**	9.98E−1 ± 9.29E−

ference between lines of ME-DER1 and LEd-DER1 is less significant then, we still observe that the power loss achieved by LEd-DER1 is better than that by ME-DER1. With the initial data set, both ME-DER1(200) and LEd-DER1(200) react fast to the change of bus loads. However, we can also find that LEd-DER1(200) is fast in finding better solutions. Similar results can be found in Fig. 4. These results validate the proposed algorithm.

5.2. Results on the IEEE 30-bus system

The tracking errors and feasible time ratios of the investigated algorithms on the dynamic IEEE 30-bus system are listed in Table 2. The results about the tracking success ratio (TSR) and the corresponding tracking speed (TS) of the algorithms are listed in Table 3.

From the tables, it can be seen that in terms of TE, the average values obtained by the LEDE algorithms are better than the values obtained by the corresponding MEDE algorithms in all the test cases. For examples, where $\tau = 0.01$, the average TE obtained by LEd-DER1 is 4.81E−2 MW, better than 5.90E−2 MW by ME-DER1; where $\tau = 0.05$, the average TE is 1.41E−2 MW for LEt-DEGL(200), better than 1.53E−2 MW for ME-DEGL(200). These results validate the effect of the proposed LEDE, and show that LEDE can be applied to not only classical DEs but also some advanced DE variants to solve dynamic OPF.

From the tables, it can be seen that, the average TE values obtained by DEGL algorithms are better than the values obtained by the corresponding DER1 algorithms in all the test cases. For example, where $\tau = 0.2$, the average TE is 1.36E−2 MW for LEt-DEGL(200), which is better than 2.74E−2 MW for LEt-DER1(200). These results suggest that combining the idea of LEDE with more powerful DE algorithms could solve dynamic OPF problem more efficiently.

From the tables, it seems like that the *LEd* strategy is more suitable for DER1 than the *LEt* strategy, because LEd-DER1 algorithms can obtained better TE values than LEt-DER1 in all the test cases. However, for DEGL, it seems like that the *LEt* is the better, since the average TE values obtained by LEt-DEGL are better than those by LEd-DEGL.

The results also demonstrate that using an initial data set of size 200 significantly improves the performance of the algorithms, which also indicate that more data could help MEDE and LEDE to solve dynamic OPF more efficiently.

From the tables, some interesting trends can be observed with τ increasing. A bigger τ indicates the power system changes more frequently and harder to solve. However, the TE values obtained by the algorithms do not increase with τ, which is somehow unexpected. This can be explained as follows. Please note that the strategy of MEDE is called once the power system changes. If τ increases, the call numbers of MEDE will increase, and the size of the data set will increase too. Consequently, the enhancing effect of MEDE and LEDE will heighten and improve the algorithms performance. Another observed trend is that the enhancing effect of LEDE over MEDE decreases with τ. For example, where $\tau = 0.01$ the TE difference between ME-DER1 and LEd-DER1 is 5.90–4.81 = 1.09E−2 MW, while with $\tau = 0.2$, the difference is 5.03–4.70 = 0.33E−2 MW. This is because the proposed LEDE takes effect in the search period. The greater τ is, the shorter the search period is, and consequently, the less effective the LEDE is. Between TSR and TS, TSR is the dominant metric, therefore the best TS values are not boldfaced in Table 3. From the table, it can be seen that the LEDE algorithms achieve higher TSR than the corresponding MEDE algorithms, which indicates that the tracking ability of MEDE is improved by the proposed learning strategy. For example, when $\tau = 0.01$, $\eta = 0.1$, the TSR of ME-DER1 is 1.93E−1 with TS being 2.96E2, while the TSR of LEd-DER1 is 6.78E1 with TS being 1.08E2. This phenomenon indicates that the LEd-DER1 is able to quickly track the reference optimal solution at a high degree of accuracy at most of the time, which is every important in practice. The difference of ME-DER1(200) and LEd-DER1(200) is even greater. The TSR of ME-DER1(200) is only 1.33E−1 on the case when $\eta = 0.1$ and $\tau = 0.5$, the TSR of LEd-DER1(200) is 8.09E−1, which indicates the efficiency of the proposed learning strategy. For DEGL algorithms, since ME-DEGL and ME-DEGL(200) have achieved a high value of TSR, the enhancing effect of LEDE is less significant.

As to FTR, the results of the algorithms are almost the same, and all the results of the MEDE algorithms exceed 99.3%, and all the results of the LEDE algorithms exceed a high level of 99.5%.

5.3. Results on the IEEE 118-bus system

Table 4 lists the tracking errors and feasible time ratios of the algorithms for all the three dynamic cases of IEEE 118-bus system with $\tau = 0.01$, 0.05 and 0.2 respectively. Compared with the

Please cite this article in press as: T. Zhu, et al., Learning enhanced differential evolution for tracking optimal decisions in dynamic power systems, Appl. Soft Comput. J. (2017), http://dx.doi.org/10.1016/j.asoc.2017.07.037

Table 3
TSR and TS of the algorithms on dynamic IEEE 30-bus system with $\eta = 0.01$.

$\tau = 0.01$	TSR	TS	$\tau = 0.01$	TSR	TS
ME-DER1	1.93E−1 ± 1.09E−1	2.96*E*2 ± 2.23*E*2	ME-DEGL	8.38E−1 ± 7.33E−2	9.58*E*1 ± 9.48*E*1
LEd-DER1	6.78E−1 ± 1.08E−1	1.13*E*2 ± 4.03*E*1	LEd-DEGL	8.69E−1 ± 7.50E−2	6.23*E*1 ± 6.27*E*1
LEt-DER1	4.02E−1 ± 1.26E−1	3.22*E*2 ± 1.82*E*2	LEt-DEGL	8.40E−1 ± 9.94E−2	6.05*E*1 ± 4.75*E*1
ME-DER1(200)	1.33E−1 ± 6.30E−2	3.68*E*2 ± 4.67*E*2	ME-DEGL(200)	9.40E−1 ± 2.58E−2	5.53*E*1 ± 1.53*E*1
LEd-DER1(200)	**8.09E−1 ± 5.11E−2**	9.51*E*1 ± 3.57*E*1	LEd-DER1(200)	**9.56E−1 ± 1.63E−2**	2.80*E*1 ± 6.39
LEt-DER1(200)	2.44E−1 ± 9.89E−2	4.48*E*2 ± 2.71*E*2	LEt-DER1(200)	**9.56E−1 ± 1.63E−2**	2.79*E*1 ± 7.91
$\tau = 0.05$	TSR	TS	$\tau = 0.05$	TSR	TS
ME-DER1	4.57E−1 ± 1.56E−1	3.53*E*1 ± 1.48*E*1	ME-DEGL	8.63E−1 ± 4.35E−2	1.95*E*1 ± 8.70
LEd-DER1	8. E−1 ± 4.28E−2	1.23*E*1 ± 5.14	LEd-DEGL	8.90E−1 ± 5.93E−2	1.48*E*1 ± 9.38
LEt-DER1	6.23E−1 ± 1.11E−1	3.79*E*1 ± 1.91*E*1	LEt-DEGL	8.44E−1 ± 1.87E−1	1.67*E*1 ± 9.38
ME-DER1(200)	1.53E−1 ± 3.48E−2	8.54*E*1 ± 5.96*E*1	ME-DEGL(200)	9.28E−1 ± 2.39E−2	2.51*E*1 ± 7.35
LEd-DER1(200)	**8.96E−1 ± 3.92E−2**	2.06*E*1 ± 6.38	LEd-DEGL(200)	9.72E−1 ± 1.18E−2	1.39*E*1 ± 3.39
LEt-DER1(200)	3.82E−1 ± 1.15E−1	7.69*E*1 ± 3.63*E*1	LEt-DER1(200)	**9.78E−1 ± 9.40E−3**	1.33*E*1 ± 2.14
$\tau = 0.2$	TSR	TS	$\tau = 0.2$	TSR	TS
ME-DER1	6.23E−1 ± 1.72E−1	3.98 ± 1.68	ME-DEGL	6.35E−1 ± 1.69E−1	6.76 ± 3.48
LEd-DER1	7.60E−1 ± 1.23E−1	2.70 ± 1.40	LEd-DEGL	6.45E−1 ± 2.41E−1	9.46 ± 1.45*E*1
LEt-DER1	6.46E−1 ± 1.94E−1	4.83 ± 3.79	LEt-DEGL	8.05E−1 ± 1.05E−1	3.79 ± 3.12
ME-DER1(200)	7.08E−1 ± 4.07E−2	4.16 ± 1.01	ME-DEGL(200)	9.00E−1 ± 1.91E−2	4.64 ± 9.09E−1
LEd-DER1(200)	**9.36E−1 ± 1.44E−2**	2.29 ± 5.51E−1	LEd-DEGL(200)	9.35E−1 ± 1.42E−2	3.79 ± 4.72E−1
LEt-DER1(200)	7.67E−1 ± 8.77E−2	3.86 ± 1.35	LEt-DEGL(200)	**9.44E−1 ± 1.33E−2**	3.87 ± 6.84E−1

Table 4
TE and FTR of the algorithms on dynamic IEEE 118-bus systems.

$\tau = 0.01$	TE (MW)	FTR	$\tau = 0.01$	TE (MW)	FTR
ME-DER1(200)	−8.61 ± 3.07E−1	9.39E−1 ± 0.00	ME-DEGL(200)	−9.85 ± 5.17E−1	9.39E−1 ± 0.00
LEd-DER1(200)	**−9.57 ± 3.87E−1**	9.39E−1 ± 0.00	LEd-DEGL(200)	−9.99 ± 3.86E−1	9.39E−1 ± 0.00
LEt-DER1(200)	−9.43 ± 4.59E−1	9.39E−1 ± 0.00	LEt-DEGL(200)	**−1.03E1 ± 3.47E−1**	9.39E−1 ± 0.00
$\tau = 0.05$	TE (MW)	FTR	$\tau = 0.05$	TE (MW)	FTR
ME-DER1(200)	−3.70 ± 1.59E−1	9.91E−1 ± 2.30E−16	ME-DEGL(200)	−4.35 ± 2.40E−1	9.91E−1 ± 2.30E−16
LEd-DER1(200)	**−3.91 ± 1.61E−1**	9.91E−1 ± 2.30E−16	LEd-DEGL(200)	−4.49 ± 1.25E−1	9.91E−1 ± 2.30E−16
LEt-DER1(200)	−3.78 ± 2.43E−1	9.91E−1 ± 2.30E−16	LEt-DEGL(200)	**−4.50 ± 1.49E−1**	9.91E−1 ± 2.30E−16
$\tau = 0.2$	TE (MW)	FTR	$\tau = 0.2$	TE (MW)	FTR
ME-DER1(200)	−3.46 ± 5.01E−2	9.99E−1 ± 1.15E−16	ME-DEGL(200)	−3.62 ± 1.08E−1	1.00 ± 1.15E−16
LEd-DER1(200)	**−3.59 ± 6.40E−2**	9.99E−1 ± 1.15E−16	LEd-DEGL(200)	−3.75 ± 1.58E−1	1.00 ± 1.15E−16
LEt-DER1(200)	−3.47 ± 5.59E−2	9.99E−1 ± 1.15E−16	DEt-DEGL(200)	**−3.94 ± 8.28E−2**	1.00 ± 1.15E−16

Table 5
TSR and TS of the algorithms on dynamic IEEE 118-bus system with $\eta = 0.01$.

$\tau = 0.01$	TSR	TS	$\tau = 0.01$	TSR	TS
ME-DER1(200)	8.24E−1 ± 8.89E−2	2.04E1 ± 7.36E1	ME-DEGL(200)	9.14E−1 ± 3.76E−2	2.24E1 ± 2.72E1
LEd-DER1(200)	**9.33E−1 ± 3.04E−2**	1.45E1 ± 5.24E1	LEd-DEGL(200)	9.29E−1 ± 2.44E−2	1.28E1 ± 1.58E1
LEt-DER1(200)	9.25E−1 ± 4.14E−2	1.00 ± 0.00	LEt-DEGL(200)	**9.41E−1 ± 1.15E−16**	2.29 ± 3.85
$\tau = 0.05$	TSR	TS	$\tau = 0.05$	TSR	TS
ME-DER1(200)	9.73E−1 ± 1.83E−2	1.02 ± 6.13E−2	ME-DEGL(200)	9.82E−1 ± 1.00E−2	4.12 ± 5.41
LEd-DER1(200)	**9.84E−1 ± 8.72E−3**	1.17 ± 6.41E−1	LEd-DEGL(200)	9.85E−1 ± 5.51E−3	3.65 ± 4.34
LEt-DER1(200)	9.82E−1 ± 1.43E−2	1.00 ± 0.00	LEt-DEGL(200)	**9.88E−1 ± 0.00**	1.36 ± 6.89E−1
$\tau = 0.2$	TSR	TS	$\tau = 0.2$	TSR	TS
ME-DER1(200)	**9.96E−1 ± 2.36E−3**	1.00 ± 0.00	ME-DEGL(200)	9.96E−1 ± 2.92E−3	1.00 ± 0.00
LEd-DER1(200)	**9.96E−1 ± 2.22E−3**	1.01 ± 2.24E−2	LEd-DEGL(200)	9.96E−1 ± 2.96E−3	1.00 ± 0.00
LEt-DER1(200)	9.95E−1 ± **3.93E−3**	1.00 ± 0.00	LEt-DEGL(200)	**9.98E−1 ± 2.30E−16**	1.02 ± 3.86E−2

IEEE 30-bus system, the IEEE 118-bus system is more complex, and more difficult to optimize. However, the conclusions above drawn from the results on IEEE 30-bus systems still hold in this group of experiments. The experiments show that RR-DER1 fails to locate any feasible solution during the entire period of optimization. The track error of the algorithms in this group of cases are all less than 0. According to the definition of tracking error, it means that the solutions obtained by the algorithms are better than the reference solutions. The average values of TE obtained by the LEDE algorithms are better than the corresponding MEDE algorithms in all the three cases, which indicates the superiority of the LEDE.

The results on the tracking success ratio and the tracking speed of the algorithms are listed in Table 5. As to the tracking success ratio, all the MEDE algorithms achieve a level greater than 82%, while LEDE algorithms 92% on almost all the cases. The results of tracking speed show that both MEDE and LEDE could help DE

Please cite this article in press as: T. Zhu, et al., Learning enhanced differential evolution for tracking optimal decisions in dynamic power systems, Appl. Soft Comput. J. (2017), http://dx.doi.org/10.1016/j.asoc.2017.07.037

to quickly find a solution nearly as good as the reference solution.

6. Conclusions and future works

In order to quickly track optimal decisions of OPF in dynamic environments, this paper proposes a learning enhanced DE algorithm (LEDE) for decision making systems of power systems. LEDE combines MEDE with the nearest-neighbor rule to reuse solutions of previous tasks, and employs the elitism stochastic ranking to handle constraints. As to the mechanism of reusing previous solutions, two schemes are designed for LEDE, termed as LEd and LEt. Experiments are conducted on dynamic IEEE 30-bus system and IEEE 118-bus system with varying bus loads. LEDE and MEDE are instantiated with DE/Rand/1 and DEGL to be compared in the experiments. The experimental results indicate the proposed LEDE can enhance MEDE in all the test cases. The results also indicate that LEDE can be applied to not only classical DE but also some advanced DE variants, and that LEd is better than LEt for some DE algorithms and vice versa.

There are still some worthy works to pursue in the future. Similarity metric is a core issue of nearest-neighbor algorithm. In this paper, we use the normalized Euclidean distance to measure the similarity between tasks, which needs to be further investigated. Moreover, this paper shows that enhancing differential evolution by exploiting the optimization records is a promising direction, and hence, more mechanisms incorporating various machine learning algorithms should be investigated. Finally, applying the proposed algorithms to other applications in real world is also one of our future works.

Acknowledgments

This work is supported by the Fundamental Research Funds for the Central Universities (No. 06116073) and the National Natural Science Foundation of China (No. 61573327). Many thanks to Dr. Mallipeddi and his colleagues for their Matlab codes of optimal power flow, and the anonymous reviewers for their valuable comments.

References

[1] H. Dommel, W. Tinney, Optimal power flow solutions, IEEE Trans. Power Appar. Syst. 87 (10) (1968) 1866–1876, http://dx.doi.org/10.1109/TPAS.1968.292150.
[2] M. Irving, Yong-Hua Song, Optimisation techniques for electrical power systems: Part 1. Mathematical optimisation methods, Power Eng. J. 14 (5) (2000) 245–254, http://dx.doi.org/10.1049/pe:20000509.
[3] Yong-Hua Song, M. Irving, Optimisation techniques for electrical power systems: Part 2. Heuristic optimisation methods, Power Eng. J. 15 (3) (2001) 151–160, http://dx.doi.org/10.1049/pe:20010307.
[4] X. Xia, A.M. Elaiw, Optimal dynamic economic dispatch of generation: a review, Electr. Power Syst. Res. 80 (8) (2010) 975–986, http://dx.doi.org/10.1016/j.epsr.2009.12.012.
[5] R. Storn, K. Price, Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, J. Glob. Optim. 11 (4) (1997) 341–359, http://dx.doi.org/10.1023/A:1008202821328.
[6] R.C. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: Proceedings of the Sixth International Symposium on Micro Machine and Human Science, vol. 1, New York, NY, 1995, pp. 39–43.
[7] W. Tang, M. Li, Q. Wu, J. Saunders, Bacterial foraging algorithm for optimal power flow in dynamic environments, IEEE Trans. Circ. Syst. I: Regul. Pap. 55 (8) (2008) 2433–2442, http://dx.doi.org/10.1109/TCSI.2008.918131.
[8] M.S. Li, T.Y. Ji, Q.H. Wu, Y.S. Xue, Stochastic optimal power flow using a paired-bacteria optimizer, IEEE PES General Meeting (2010) 1–7, http://dx.doi.org/10.1109/PES.2010.5589620.
[9] T. Zhu, W. Luo, C. Bu, L. Yue, Accelerate population-based stochastic search algorithms with memory for optima tracking on dynamic power systems, IEEE Trans. Power Syst. 31 (1) (2016) 268–277, http://dx.doi.org/10.1109/TPWRS.2015.2407899.
[10] J. Zhang, Z.H. Zhang, Y. Lin, N. Chen, Y.J. Gong, J.H. Zhong, H.S.H. Chung, Y. Li, Y.H. Shi, Evolutionary computation meets machine learning: a survey, IEEE Comput. Intell. Mag. 6 (4) (2011) 68–75, http://dx.doi.org/10.1109/MCI.2011.942584.
[11] T.T. Nguyen, S. Yang, J. Branke, Evolutionary dynamic optimization: a survey of the state of the art, Swarm Evol. Comput. 6 (2012) 1–24, http://dx.doi.org/10.1016/j.swevo.2012.05.001.
[12] T.M. Mitchell, Machine Learning, McGraw-Hill, Boston, MA, 1997.
[13] S. Das, A. Abraham, U.K. Chakraborty, A. Konar, Differential evolution using a neighborhood-based mutation operator, IEEE Trans. Evol. Comput. 13 (3) (2009) 526–553, http://dx.doi.org/10.1109/TEVC.2008.2009457.
[14] S. Das, P.N. Suganthan, Differential evolution: a survey of the state-of-the-art, IEEE Trans. Evol. Comput. 15 (1) (2011) 4–31, http://dx.doi.org/10.1109/TEVC.2010.2059031.
[15] D.M. Martínez, J.C. Fernández-rodríguez, Artificial intelligence applied to project success: a literature review, Int. J. Interact. Multimed. Artif. Intell. 3 (5) (2015) 77–82, http://dx.doi.org/10.9781/ijimai.2015.3510.
[16] M.G. Epitropakis, D.K. Tasoulis, N.G. Pavlidis, V.P. Plagianakos, M.N. Vrahatis, Enhancing differential evolution utilizing proximity-based mutation operators, IEEE Trans. Evol. Comput. 15 (1) (2011) 99–119, http://dx.doi.org/10.1109/TEVC.2010.2083670.
[17] R. Mallipeddi, S. Jeyadevi, P. Suganthan, S. Baskar, Efficient constraint handling for optimal reactive power dispatch problems, Swarm Evol. Comput. 5 (2012) 28–36, http://dx.doi.org/10.1016/j.swevo.2012.03.001.
[18] J.E. Baker, Reducing bias and inefficiency in the selection algorithm, Proceedings of the Second International Conference on Genetic Algorithms (1987) 14–21.
[19] T.M. Cover, P.E. Hart, Nearest neighbor pattern classification, IEEE Trans. Inf. Theory 3 (1) (1967) 21–27.
[20] S.J. Louis, J. McDonnell, Learning with case-injected genetic algorithms, IEEE Trans. Evol. Comput. 8 (4) (2004) 316–328, http://dx.doi.org/10.1109/TEVC.2004.823466.
[21] T.P. Runarsson, X. Yao, Stochastic ranking for constrained evolutionary optimization, IEEE Trans. Evol. Comput. 4 (3) (2000) 284–294, http://dx.doi.org/10.1109/4235.873238.
[22] K. Deb, An efficient constraint handling method for genetic algorithms, Comput. Methods Appl. Mech. Eng. 186 (2000) 311–338.
[23] I. Triguero, S. García, F. Herrera, IPADE: iterative prototype adjustment for nearest neighbor classification, IEEE Trans. Neural Netw. 21 (12) (2010) 1984–1990, http://dx.doi.org/10.1109/TNN.2010.2087415.
[24] I. Triguero, S. García, F. Herrera, Differential evolution for optimizing the positioning of prototypes in nearest neighbor classification, Pattern Recognit. 44 (4) (2011) 901–916, http://dx.doi.org/10.1016/j.patcog.2010.10.020.
[25] Y. Liu, F. Sun, A fast differential evolution algorithm using k-nearest neighbour predictor, Expert Syst. Appl. 38 (4) (2011) 4254–4258, http://dx.doi.org/10.1016/j.eswa.2010.09.092.
[26] S.Y. Park, J.J. Lee, An efficient differential evolution using speeded-up k-nearest neighbor estimator, Soft Comput. 18 (1) (2014) 35–49, http://dx.doi.org/10.1007/s00500-013-1030-x.
[27] T.H. Cormen, C.E. Leiserson, R.L. Rivest, Introduction to Algorithms, MIT Press/McGraw-Hill, 2009.
[28] R.D. Zimmerman, C.E. Murillo-Sanchez, R.J. Thomas, MATPOWER: steady-state operations planning and analysis tools for power systems research and education, IEEE Trans. Power Syst. 26 (1) (2011) 12–19, http://dx.doi.org/10.1109/TPWRS.2010.2051168.

Please cite this article in press as: T. Zhu, et al., Learning enhanced differential evolution for tracking optimal decisions in dynamic power systems, Appl. Soft Comput. J. (2017), http://dx.doi.org/10.1016/j.asoc.2017.07.037