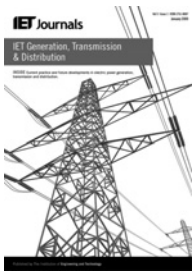
www.ietdl.org

Published in IET Generation, Transmission & Distribution
Received on 25th February 2012
Revised on 28th August 2012
Accepted on 11th October 2012
doi: 10.1049/iet-gtd.2012.0096




ISSN 1751-8687

# Binary neighbourhood field optimisation for unit commitment problems

*Zhou Wu, Tommy W.S. Chow*

*Department of Electronic Engineering, City University of Hong Kong, Kowloon Tong, Kowloon, Hong Kong*
*E-mail: wuzhsky@gmail.com*

**Abstract:** Inspired by the neighbourhood cooperation, a new discrete optimisation algorithm is proposed. The so-called binary neighbourhood field optimisation (BNFO), utilises the attractive field of the superior neighbour and the repulsive field of the inferior neighbour. As a kind of local search, BNFO is able to deliver promising results efficiently within acceptable computational time. BNFO is applied to solve the unit commitment problem (UCP), whose objective is to minimise the operation cost of the generation units over the scheduling horizon. After numerical tests on several benchmark UCP cases, the obtained costs are less expensive compared with conventional Lagrangian relaxation, genetic algorithm, evolutionary programming, particle swarm optimisation and differential evolutionary. BNFO can converge to promising results with less computation times, especially for the large-scale UCPs.

## 1 Introduction

The unit commitment problem (UCP) refers to the scheduling of start-up and shutdown operation of the generation units for satisfying the forecasted demand over a short term period (1–7 days) in future. UCP plays a major role in the daily operation planning of power systems, especially in the framework of the deregulated power markets. UCP can be regarded as an optimisation task to minimise the total operation cost of a power system during the scheduling horizon, subject to a number of system and unit constraints. The overall optimisation problem can be divided into two sub-problems. One is the mixed integer non-linear programming problem of determining the on/off status of the generating units hourly in a short term period; the other is the quadratic programming problem of economic load dispatching (ELD) among the committed units. It is a complicated procedure to find the optimal solution that optimises both sub-problems; the difficulty grows in proportion to the number of units and constraints.

Many methods have been applied to solve UCP in the past decades. The major methods include priority list (PL) method [1], dynamic programming (DP) [2], Lagrangian relaxation (LR) [3–5] and mixed-integer linear programming [6, 7]. In the PL method, baseline load units are committed at first and peak load units at last. PL is fast, but easily obtaining trapped in the local optimum with relatively high costs. The DP method has been widely used in UCP, but its disadvantage is the 'curse of dimensionality', which has limited its application to very small-size system. The LR methods focus on finding a feasible solution of the primal objective function, while minimising the 'duality gap', that is, the difference of the primal and the dual objective function. Their main disadvantage is the difficulty of searching the feasible solutions.

Apart from the aforementioned conventional methods, there are many meta-heuristic methods applied in UCP, such as simulated annealing (SA) [8], tabu search (TS) [9], artificial neural network [10], genetic algorithm (GA) [11, 12], evolutionary programming (EP) [13–15], particle swarm optimisation (PSO) algorithm [16–18] and differential evolution (DE) [19]. Other effective methods have hybridised certain heuristic and meta-heuristic to improve the resulted performance of UCP, such as the hybrid EP and TS method [20] and the hybrid PSO and TS method [21]. These meta-heuristic methods have become more popular than the conventional methods, because they can explore the whole search space with the ability of escaping from local optima, especially when the problems are difficult and multimodal. Global search algorithms, such as GA, PSO and DE, need a large amount of computational time to find the global optimum for the large scale UCP, because they usually have random choices of direction to approach possibly good solutions.

The mechanism of GA is based on the principles of natural evolution. In GA, the offspring solutions are generated from the parent solutions with three operators – selection, crossover and mutation [22]. Each individual has the chance to be selected into the breeding pool according to its fitness value. Then the crossover and mutation operations are used to reproduce offspring. For the crossover, individuals in the breeding pool are recombined to explore the search space. For the mutation, the individual randomly exploits its surrounding region to generate the offspring. As GA can use binary encoding method and its three operators can be generally used in the discrete case, GA has been applied into discrete optimisation problems.

PSO algorithm emulates the forging behaviour in the bird flock and fish school [23]. In PSO, each member, called a

particle, learns from the best particle in the population. Each particle adapts its positions towards the best position of the population and its own best position. Because of the attraction of the two poles, each particle can escape out of its surrounding region to explore the search space. DE is another popular global search algorithm, which generates offspring individuals by perturbing parents with differential vectors of two random individuals [24]. In DE, the search step is self-adapted along the evolving process because of the balance between the exploitation and the exploration. It is worth noting that PSO and DE are both based on the vector field in the continuous space. Their operators cannot be directly applied to discrete optimisation. Some research works have focused on their extensions of discrete optimisation, such as projecting the discrete space into the continuous space [25], converting numerical operators (' + ', ' − ') into logical operators ('and', 'or') [26].

However, the global search algorithms such as GA and PSO, often fail in solving some difficult problems in the given computation time. The main reason is their lack of local search during the global exploration. A new meta-heuristic algorithm called neighbourhood field optimisation (NFO) has been proposed to balance exploitation and exploration in order to accelerate the convergence speed [27]. The NFO algorithm is inspired by the local cooperation behaviour in the biology world, which is modelled as the neighbourhood field. This paper newly proposes a binary NFO (BNFO) algorithm for the combinational and discrete optimisation problems to enhance the local search. BNFO can find the global optimal efficiently with fast convergence speed. BNFO is applied to solve a typical discrete problem, that is, UCP. The proposed method is tested on the UCP systems with the number of units in the range of 10–100. Our simulation results show that BNFO is more effective and accurate than other methods, such as SA, GA, PSO and DE.

This paper is organised as follows: Section 2 provides the mathematical formulations of UCP; Section 3 introduces the background of the NFO algorithm and presents the newly proposed BNFO; Section 4 gives the details to solve UCP with BNFO; Section 5 gives the numerical simulations and the statistical comparisons with SA, GA, PSO and DE; and the paper is concluded in Section 6.

## 2 Formulation of UCP

The objective of UGP is to find an optimal combinational set of units' on/off status, which minimises the total costs over the scheduling horizon. The total costs include fuel costs, start-up costs and shutdown costs as

$$\min \sum_{t=1}^{T} \sum_{i=1}^{N} f_i\left(P_i^t\right) u_i^t + \mathrm{ST}_i^t u_i^t \left(1 - u_i^{t-1}\right) + \mathrm{SD}_i u_i^{t-1} \left(1 - u_i^t\right) \tag{1}$$

where the first component is the fuel costs, the second component is start-up costs and the third component is shutdown costs. In (1), $T$ is the total scheduling period, $N$ is the number of generators, $P_i^t$ is power output of unit $i$ at time $t$, $u_i^t$ is on/off status of unit $i$ at time $t$ ($u_i^t = 1$ means the unit is on and $u_i^t = 0$ means the unit is off), $\mathrm{ST}_i^t$ is start-up cost of unit $i$ at time $t$, $\mathrm{SD}_i$ is shutdown cost of unit $i$. The fuel cost of unit $i$ is usually approximated by a quadratic function as

$$f_i\left(P_i^t\right) = a_i + b_i P_i^t + c_i \left(P_i^t\right)^2 \tag{2}$$

where $a_i$, $b_i$ and $c_i$ are the cost coefficients. The start-up cost is dependent upon the operating time. If the off time before start-up is less than a required period, the cost is called hot start cost, otherwise, the cost is called cold start cost. The start-up cost can be defined as

$$\mathrm{ST}_i^t = \begin{cases} \text{hot start cost, if } Td_i \le T\mathrm{off}_i^t \le Td_i + Tc_i \\ \text{cold start cost, if } T\mathrm{off}_i^t > T_{i,\mathrm{down}} + Tc_i \end{cases} \tag{3}$$

where $Td_i$ is the required minimal down-time (MDT) for unit $i$, $Tc_i$ is cold start time of unit $i$, $T\mathrm{off}_i^t$ is the continuously off time of unit $i$ up to time $t$.

UCP has many equality and inequality constraints, including system's constraints and unit's constraints as follows [28].

### 2.1 System power balance

The total output power of all committed units needs to meet the system power demand as

$$\sum_{i=1}^{N} P_i^t u_i^t = P_{\mathrm{D}}^{\mathrm{t}} \tag{4}$$

where $P_{\mathrm{D}}^t$ is system load demand at time $t$.

### 2.2 Unit power output limits

Each unit has a range of power output, which is represented as

$$P_{i,\min} \le P_i^t \le P_{i,\max} \tag{5}$$

where $P_{i,\min}$ is the minimal power output of unit $i$ and $P_{i,\max}$ is the maximal power output of unit $i$.

### 2.3 System spinning reserve requirements

Spinning reserve requirements are necessary in the power system for avoiding the load interruption. The reserved value means the capability of generating excessive power outputs beyond the forecasted demand,

$$\sum_{i=1}^{N} u_i^t P_{i\max} \ge P_{\mathrm{D}}^t + P_{\mathrm{R}}^t \tag{6}$$

where $P_{\mathrm{R}}^t$ is the spinning reserve at time $t$, $P_{i\max}$ is the maximal power output of unit $i$.

### 2.4 Unit minimal up-time (MUT) and MDT

A unit must be on or off for a certain number of hours so that the unit does not change its no/off status frequently for its health. This constraint is used to enlarge the unit's life-time and efficiency

$$\begin{cases} T\mathrm{on}_i^t \ge Tu_i \\ T\mathrm{off}_i^t \ge Td_i \end{cases} \tag{7}$$

where $Tu_i$ is the required MUT of unit $i$, $Td_i$ is the required

MDT of unit $i$. $Ton_i^t$ and $Toff_i^t$ are the consecutive hours that unit $i$ has been on and off at time $t$, respectively, which can be calculated as

$$Ton_i^t = \begin{cases} Ton_i^{t-1}, & \text{if } u_i^t = 1 \\ 0, & \text{if } u_i^t = 0 \end{cases}, \quad Toff_i^t = \begin{cases} Toff_{if}^{t-1}, & \text{if } u_i^t = 0 \\ 0, & \text{if } u_i^t = 1 \end{cases} \tag{8}$$

### 2.5 Ramp up rate and ramp down rate

For unit $i$, the ramp up rates constrain the maximal increase of power output in two successive time periods, and the ramp down rates constrain the maximal power decrease of the unit. The constraints of the ramp up and down rates need to be satisfied as

$$\begin{cases} P_i^t - P_i^{t-1} \le \mathrm{RU}_i \\ P_i^{t-1} - P_i^t \le \mathrm{RD}_i \end{cases}, \quad \text{if } u_i^{t-1} = 1 \text{ and } u_i^t = 1 \tag{9}$$

where $\mathrm{RU}_i$ is the maximal ramp up rate of unit $i$, $\mathrm{RD}_i$ is the ramp down rate of unit $i$.

## 3 Neighbourhood field optimisation algorithms

### 3.1 Neighbourhood field optimisation algorithm

Some previous work about hybrid local search and self-organising algorithms were proposed to emulate the local phenomenon of nature [29, 30]. In biological world, individuals often communicate with their neighbours within the limited range of seeing and hearing. They are apt to collect information from their surrounding regions, and exchange the information with their neighbours. So individuals in NFO algorithm are mostly affected by the local environment rather than the global environment. In NFO, each individual is updated under the concept of 'learning from the neighbours', that is following superior neighbours and diverging from inferior neighbours [21]. The detailed procedure of NFO algorithm for a minimisation problem $f(x)$ is illustrated as follows:

1. *Initialisation:* randomise the initial $np$ individuals, which are sampled uniformly in the search space.
2. *Localisation:* for each individual $\boldsymbol{x_{i,G}}$ at the generation $G$, find the superior neighbour $\boldsymbol{xc_{i,G}}$ and the inferior neighbour $\boldsymbol{xw_{i,G}}$ (in the search space) as

$$\begin{cases} \boldsymbol{xc}_{i,G} = \arg \min\limits_{f(x_{k,G}) < f(x_{i,G})} \left\| \boldsymbol{x}_{k,G} - \boldsymbol{x}_{i,G} \right\| \\ \boldsymbol{xw}_{i,G} = \arg \min\limits_{f(x_{k,G}) > f(x_{i,G})} \left\| \boldsymbol{x}_{k,G} - \boldsymbol{x}_{i,G} \right\| \end{cases} \tag{10}$$

where $\boldsymbol{xc_{i,G}}$ is the superior neighbour with the function value smaller than $f(x_{i,G})$ and $\boldsymbol{xw_{i,G}}$ is the inferior neighbour with a larger function value (for minimisation problems). $\|\cdot\|$ is the distance evaluation (Euclidean distance is used). If $\boldsymbol{x_{i,G}}$ is in the best individual in the population, $\boldsymbol{xc_{i,G}}$ is defined as $\boldsymbol{x_{i,G}}$. If $\boldsymbol{x_{i,G}}$ is in the worst individual in the population, $\boldsymbol{xw_{i,G}}$ is defined as $\boldsymbol{x_{i,G}}$.

3. *Mutation:* perturb each individual as

$$\boldsymbol{v}_{i,G} = \boldsymbol{x}_{i,G} + \alpha\, \boldsymbol{rd}_1 (\boldsymbol{xc}_{i,G} - \boldsymbol{x}_{i,G}) + \alpha\, \boldsymbol{rd}_2 (\boldsymbol{xc}_{i,G} - \boldsymbol{xw}_{i,G}) \tag{11}$$

where $\boldsymbol{xc_{i,G}}$ is the superior neighbour, $\boldsymbol{xw_{i,G}}$ is the inferior neighbour, $\boldsymbol{rd}_1$ and $\boldsymbol{rd}_1$ are two random vectors in [0, 1], $\alpha$ is the learning rate. $v_{i,G}$ is the obtained mutant vector.
4. *Crossover:* recombine the mutation vector with the target vector $x_i$.

$$u_{j,i,G} = \begin{cases} v_{j,i,G}, & \text{if rand}(0, 1) \le Cr \text{ or } j = j_{\text{rand}} \\ x_{j,i,G}, & \text{otherwise} \end{cases} \tag{12}$$

where $j = 1, 2, \ldots, D$ is the dimension index; $Cr$ is the crossover probability; rand(0, 1) is a uniformly distributed random number in the scale of [0, 1]; $j_{\text{rand}}$ is a random component to accept the new mutant vector so that the trial vector is different from the target vector.
5. *Selection:* in the next generation the individual will be updated as the better one between $\boldsymbol{x_i}$ and $u_i$ as (13).

$$\boldsymbol{x}_{i,G+1} = \begin{cases} \boldsymbol{u}_{i,G}, & \text{if } f(\boldsymbol{u}_{i,G}) \le f(\boldsymbol{x}_{i,G}) \\ \boldsymbol{x}_{i,G}, & \text{otherwise} \end{cases} \tag{13}$$

6. If the stopping criteria are not satisfied, go to step 2.

As the neighbours considered in NFO are close with the updated individual with an adjusting distance, NFO can adapt the search behaviour between the local exploitation and the global exploration. In NFO, the neighbourhood field can approximate a descent direction of the objective function shown in Fig. 1. The neighbourhood field is close to the inverse gradient direction [20]. Therefore NFO can effectively find the global optimum, and enhance the diversity of the population at the same time.

### 3.2 Binary neighbourhood field optimisation

NFO has been firstly applied in continuous problems [20]. There is no work of neighbourhood field applied in discrete optimisation problems before, but some real world applications are discrete optimisation problems. For example, UCP is a binary optimisation problem with 0–1 decision variables, which determines the on/off status of generating units in each time period. When applying NFO in UCP, NFO must be extended to a binary version. In this

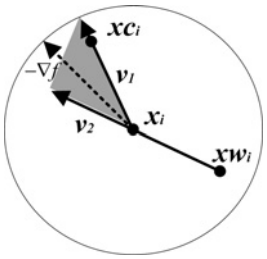



**Fig. 1** *Direction of neighbourhood field in NFO in a two-dimensional example*

The differential vectors of $xc_i$–$x_i$ and $x_i$–$xw_i$ are denoted as $v_1$ and $v_2$
The shaded area between $v_1$ and $v_2$ is likely to include the inverse direction of the gradient at $x_i$, which is denoted as the dashed arrow.




doi: 10.1049/iet-gtd.2012.0096

**Algorithm: Binary Neighbourhood Field Optimisation**

**Definition:**

*np*: the population size;
*d*: dimension of the problem;
*X*: the decision matrix with the size of *np***d*;
*Z*: the function value vector with the size of *np**1;
*Mg*: the maximum number of generations for stopping criterion.

**Begin**

***for*** $j \leftarrow 1$ to $n$ // **initialisation**
Randomise the $j$th individual $\boldsymbol{x}_j$.
Repair $\boldsymbol{x}_j$ if it violates the problem's constraints.
Calculate $\boldsymbol{x}_j$'s fitness value $z_j$.
***endfor***
$G=1$;
***while*** $G \leq Mg$ // **main iteration**
***for*** $i \leftarrow 1$ to $np$
Locate $\boldsymbol{x}_{i,G}$ in $X$ to obtain its superior neighbour $\boldsymbol{xc}_{i,G}$ and inferior neighbour $\boldsymbol{xw}_{i,G}$.
$r_{a1}=rand<\alpha$; // **mutation**
$r_{a2}=rand<\alpha$;
$v_1 = r_{a1}$ & xor($\boldsymbol{x}_{i,G}$, $\boldsymbol{xc}_{i,G}$);
$v_2 = r_{a2}$ & xor($\boldsymbol{xw}_{i,G}$, $\boldsymbol{xc}_{i,G}$);
$\boldsymbol{v}_{i,G}$ = xor($\boldsymbol{x}_{i,G}$, $v_1 \mid v_2$);
Repair $\boldsymbol{v}_{i,G}$ if it violates the problem's constraints
Crossover the mutant $\boldsymbol{v}_{i,G}$ with $\boldsymbol{x}_{i,G}$ to generate the trial vector $\boldsymbol{u}_{i,G}$.
Select the trial vector $\boldsymbol{u}_{i,G}$ if it is better than $\boldsymbol{x}_{i,G}$, otherwise select $\boldsymbol{x}_{i,G}$.
***endfor***
***endwhile***
Output the best solution in the population.
**End**

**Fig. 2** *Pseudo-code of BNFO*

paper, we extend NFO to BNFO for discrete optimisation problems.

In BNFO, the individuals are coded as bit strings. Owing to this different coding method, the location and mutation need to be revised accordingly. In the location, the Hamming distance is used instead of the Euclidian distance to find the nearest neighbours. In the mutation, the mutant vector is obtained by reformulating (14) as

$$\begin{aligned} \boldsymbol{v}_{i,G} \ominus \boldsymbol{x}_{i,G} &= \boldsymbol{\alpha}_{r1} \otimes (\boldsymbol{xc}_{i,G} \ominus \boldsymbol{x}_{i,G}) \oplus \boldsymbol{\alpha}_{r2} \otimes (\boldsymbol{xc}_{i,G} \ominus \boldsymbol{xw}_{i,G}) \\ \boldsymbol{v}_{i,G} &= \boldsymbol{x}_{i,G} \ominus [\boldsymbol{\alpha}_{r1} \otimes (\boldsymbol{xc}_{i,G} \ominus \boldsymbol{x}_{i,G}) \oplus \boldsymbol{\alpha}_{r2} \otimes (\boldsymbol{xc}_{i,G} \ominus \boldsymbol{xw}_{i,G})] \end{aligned} \tag{14}$$

where $\ominus$ denotes 'XOR' operator, $\otimes$ denotes 'AND' operator, and $\oplus$ denotes 'OR' operator. $\alpha_{r1}$ and $\alpha_{r2}$ are random binary integer vectors generated by

$$\begin{cases} \boldsymbol{\alpha}_{r1} = \text{rand}_1 < \alpha \\ \boldsymbol{\alpha}_{r2} = \text{rand}_2 < \alpha \end{cases} \tag{15}$$

where $\text{rand}_1$ and $\text{rand}_2$ are random vectors uniformly distributed in [0, 1] and $\alpha$ is the learning rate ($0 < \alpha < 1$).

The other parts of BNFO are the same with the original NFO. The details of BNFO are shown in Fig. 2. The figure has listed the pseudo-code of BNFO for general discrete problems. It is clear that BNFO has a similar structure with the structure of continuous NFO except the location and mutation.

## 4 Application to UCP

In the UCP's applications, BNFO is used to optimise the unit-scheduling problem in the first step, and the Lambda-iteration method [31] is used to solve the economic load dispatch problem in the second step. These two steps run iteratively until the algorithm meets the stopping criterion. Optimising the first sub-problem of unit-scheduling is more difficult than the other sub-problem of ELD. So this paper mainly discusses how to model BNFO for the first sub-problem, and the second sub-problem is solved by the traditional Lambda-iteration method. These two sub-problems are optimised iteratively until the algorithm meets the stopping criterion shown in Fig. 3.

### 4.1 Initialisation

For the application of BNFO to UCP, each individual is encoded as a binary string, in which a '1' at a certain location indicates that the unit is on at that particular hour,

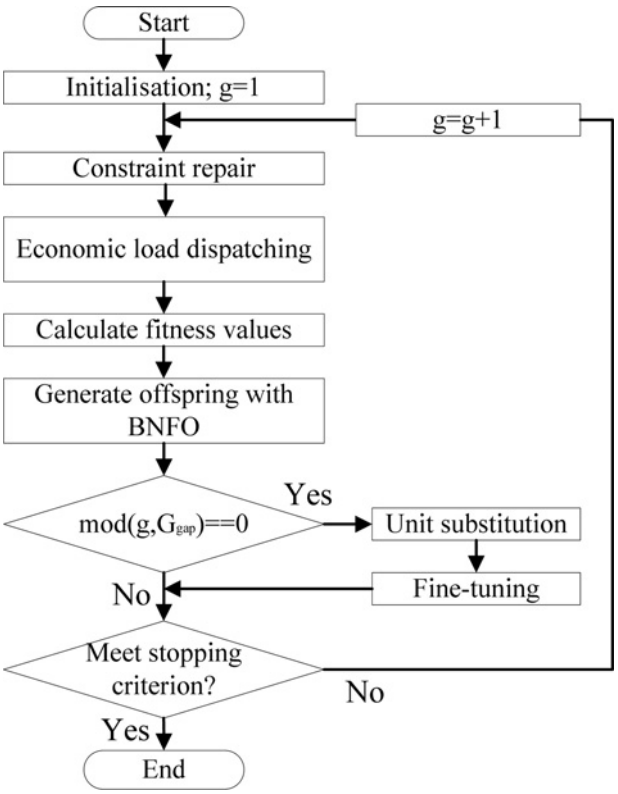



**Fig. 3** *Flowchart of the proposed method for UCP*

while a '0' indicates that the unit is off. Over the scheduling period $T$, the on/off status of all the $N$ units can be expressed as an integer matrix $\boldsymbol{U}$ as

$$\boldsymbol{U} = \begin{bmatrix} u_1^1 & u_1^2 & \cdots & u_1^T \\ u_2^1 & u_2^2 & \cdots & u_2^T \\ \vdots & \vdots & \vdots & \vdots \\ u_N^1 & u_N^2 & \cdots & u_N^T \end{bmatrix}$$

where $u_i^t$ is the on/off status of unit $i$ at time $t$. In the initialisation process, the integer matrices are uniformly randomised as either '0' or '1'. By converting each matrix, the individual in BNFO is a vector with length $N*T$ as

$$\boldsymbol{x_i} = \left[u_1^1, u_1^2, \ldots, u_1^{\mathrm{T}}, u_2^1, u_2^2, \ldots, u_2^{\mathrm{T}}, u_N^1, u_N^2, \ldots, u_N^{\mathrm{T}}\right]$$

### 4.2 Constraints repair

The initialised vectors or the offspring vectors may violate the constraints of UCP mentioned in the Section 2, which includes the spinning reserve requirement, MUT, MDT and power balance limits. In this paper, these constraints are repaired under the following steps:

1. *MUT and MDT constraints:* The vectors are checked whether they violate the MUT and down-time constraints. Although the power demand may change frequently in the scheduling period, frequent changing of the units' status need to be avoided. For the vectors violating the MUT and MDT constraints, we repair them using a rule-based heuristic method used in [17] as follows Fig. 4:

Fig. 5 has given the flowchart of above pseudo-code.

```
Begin
  for j=1 to N
    for t=1 to T
      if u_j^t==1 and u_j^{t-1}==0 and Toff_i^{t-1} < Td_i
        u_j^t=0;
      endif
      if u_j^t==0 and u_j^{t-1}==1 and Ton_i^{t-1} < Tu_i
        u_j^t=1;
      endif
    endfor
  endfor
End
```

**Fig. 4** *Rule-based heuristic method [17]*

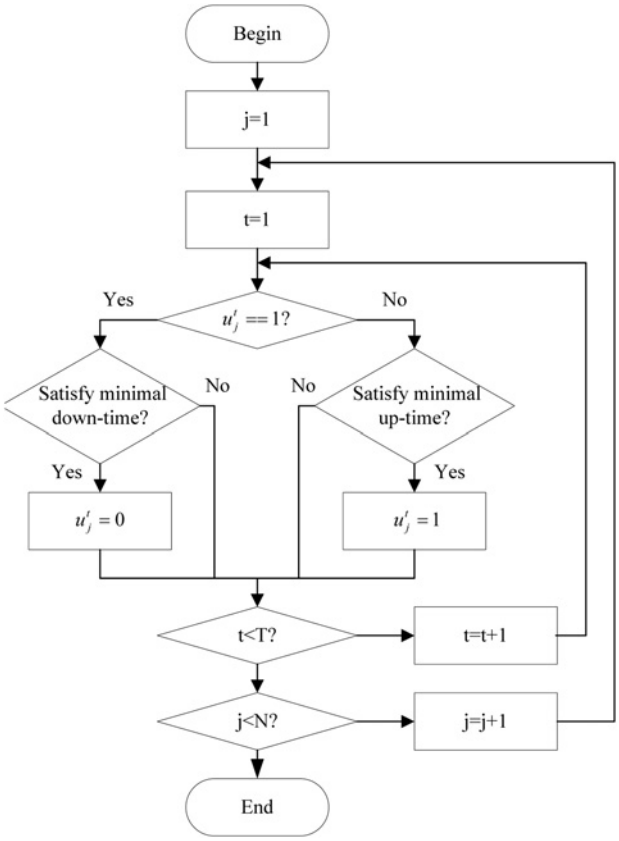



**Fig. 5** *Flowchart of repairing MUT and down-time constraints*

2. *Spinning reserve constraint:* The vectors are checked whether they violate the spinning reserve constraint. The amount of excessive spinning reserve is checked at each hour as

$$\Delta^t = \sum_{i=1}^{N} u_i^t P_{i\max} - P_D^t - P_R^t, \quad (t = 1, 2, \ldots, T) \qquad (16)$$

If the spinning reserve is not sufficient ($\Delta^t < 0$), the uncommitted units need to be turned on according to their efficiency rank list. The efficiency of unit $i$ can be evaluated by the average cost, which is the cost per unit of power ($/MW) when the unit is at its full capacity. The efficiency of unit $i$ can be formulated as

$$e_i = \frac{f_i(P_{i\max})}{P_{i\max}} = \frac{a_i}{P_{i\max}} + b_i + c_i P_{i\max} \qquad (17)$$




doi: 10.1049/iet-gtd.2012.0096

**Begin**
  **for** $t$=1 to $T$
    $\Delta^t = \sum_{i=1}^{N} u_i^t P_{i\max} - P_D^t - P_R^t$;
    **if** $\Delta^t < 0$
      Stochastic sort the uncommitted units in ascending order of the average cost to obtain a list ***ss*** .
      $g$ =1;
      **while** $\Delta^t < 0$
        $u_{ss_g}^t$ =1;
        $\Delta^t = \sum_{i=1}^{N} u_i^t P_{i\max} - P_D^t - P_R^t$
        $g = g + 1$;
      **endwhile**
    **endif**
  **endfor**
  Update the on/off time *Ton* and *Toff* .
  /*start to check and repair the minimal on/off time constraints. */
  **for** $j$=1 to $N$
    **for** $t$=1 to $T$
      **if** $u_j^t$ ==0 and $u_j^{t-1}$ ==1 12
        **if** $Ton_i^{t-1} < Tu_i$
          $u_j^t$ =1;
        **endif**
        **if** $t + Td_j - 1 \le T$ and $Toff_i^{t+Td_i-1} < Td_i$
          $u_j^t$ =1;
        **endif**
        **if** $t + Td_j - 1 > T$ and $\sum_{k=t}^{T} u_i^t > 0$
          $u_j^t$ =1;
        **endif**
      **endif**
    **endfor**
  **endfor**
**End**

**Fig. 6** *Heuristic-based method*

In the following constraints' handling, we always need to generate a units' list with the average costs in ascending order. A stochastic sorting method has been employed in this paper as follows. First, randomly select two units and then add the more efficient unit in the end of list (in ascending order). Second, for the remaining units out of the list, do the first step and then place the selected unit sequentially in the list until the list is full. For the solutions that violate spinning reserve, the uncommitted units are sorted stochastically in an ascending order of the average cost. The following repair method is conducted according to the obtained list.

We introduce a heuristic-based method to repair the vectors for satisfying the spin reserve constraints as in Fig. 6:

Fig. 7 has given the flowchart of above pseudo-code.

3. *Excessive spinning reserve constraint:* After repairing the time and spinning reserve constraints as mentioned above, some redundant units may be turned on. This is called excessive spinning reserve, which is not desirable because of the high operation cost. We use a heuristic-based algorithm to de-commit the redundant units. Stochastic sort the committed units in descending order of the average cost. Starting from the committed units in the first place of the list, the units will be shut down if the decision does not violate the constraints of the spinning reserve and MUT. The flow chart of repairing the excessive spinning reserve constraint is given in Fig. 8. The detailed procedure of handling the excessive spinning reserve is as follows Fig. 9:

### 4.3 Economic load distribution

When obtaining the feasible scheduling solutions, the fitness function value in (1) is still unknown because the fuel cost cannot be calculated without dispatching the load. The ELD is an independent optimisation problem, which can be regarded as a quadratic programming. ELD problem can be easily solved by some numerical techniques. We use Lambda-iteration method to optimise ELD problem [19, 28]. Given the solution $x_t$ at time $t$, Lambda-iteration method searches an optimal power outputs $P_i^t$ for all unit to minimise the fuel cost $\sum_{i=1}^{N} f_i(P_i^t) u_i^t$.

### 4.4 Unit substitution

A heuristic search method called unit substitution is used to find a better solution locally, based on the obtained solution by the BNFO. The unit substitution runs repeatedly at the end of each $G_{\text{gap}}$ iterations shown in Fig. 3. Generally, generation units can be classified into three types: base load units, intermediate load units and peak load units. Base load units have low operation cost, high startup cost and long MUT and MDT requirements. Intermediate load units have medium operating cost, medium startup cost, and medium MUT and MDT requirements. Peak load units have high operation cost, low startup cost and short MUT and MDT requirements. Owing to these properties, peak load units could frequently turned on/off, unlike intermediate load units in the peak load

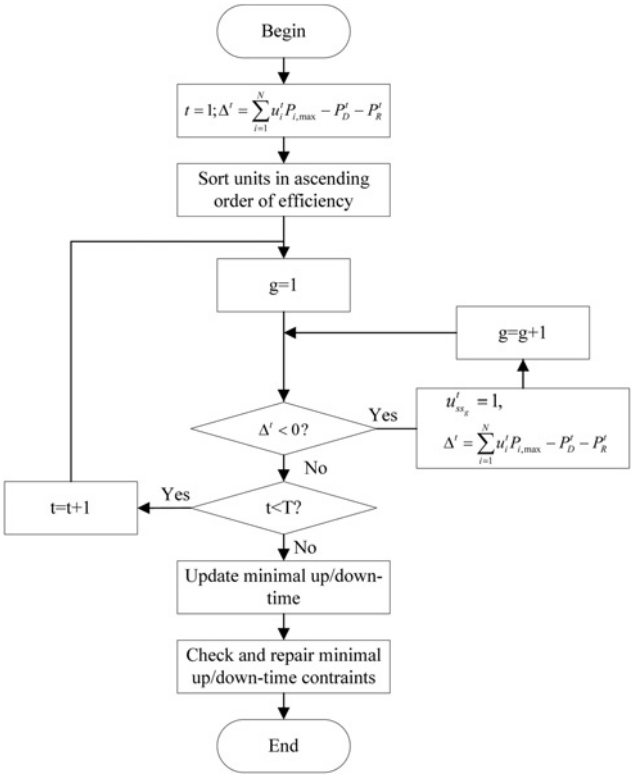



**Fig. 7** *Flowchart of repairing the spinning reserve constraint*

periods. If the intermediate units are committed in the peak load periods, they cannot be turned off after the peak hour because of the MUT constraint. These may cause excessive spinning reserve and waste of money. Therefore if possible, those intermediate load units could be de-committed in the peak load periods and replaced by some peak load units to satisfy the spinning reserve requirement.

A unit substitution method modified from [5] is utilised. For each peak period, starting from 2 h after peak hour, if the excessive spinning reserve exists, the unit substitution will be carried out. The unit substitution method searches the inefficient intermediate load units for de-commitment, and commits the most efficient peak load units instead. If the solution after the unit substitution does not violate the constraints and has lower cost than the previous one, it will be accepted as the new individual in the next generation.

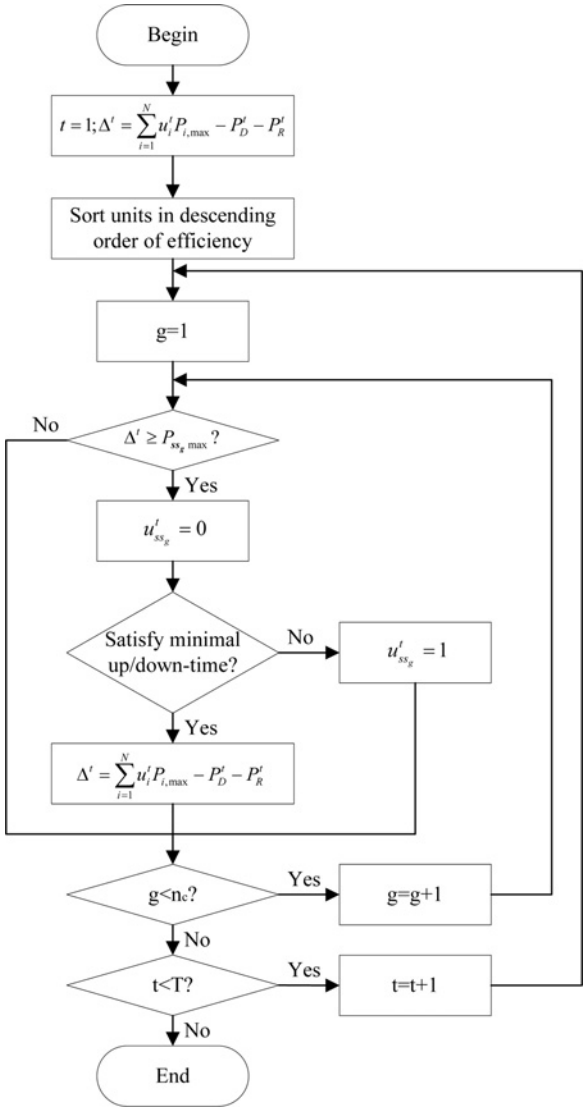



**Fig. 8** *Flowchart of repairing the excessive spinning reserve constraint*

**Begin**
**for** $t$=1 to $T$
$\Delta^t=\sum_{i=1}^{N}u_i^tP_{i,\max}-P_D^t-P_R^t;$
Stochastic sort the committed units in descending order of average cost to obtain a list ***ss*** with length $n_c$.
**for** $g$=1 to $n_c$
**if** $\Delta^t\geq P_{ss_g\max}$
$u_{ss_g}^t=0;$
Check whether the de-committed unit violates minimal up-time and down-time constraints.
**if** violating the time constraints
$u_{ss_g}^t=1;$
**endif**
$\Delta^t=\sum_{i=1}^{N}u_i^tP_{i\max}-P_D^t-P_R^t;$
**endif**
**endfor**
**endfor**
**End**

**Fig. 9** *Detailed procedure of handling the excessive spinning reserve*




doi: 10.1049/iet-gtd.2012.0096

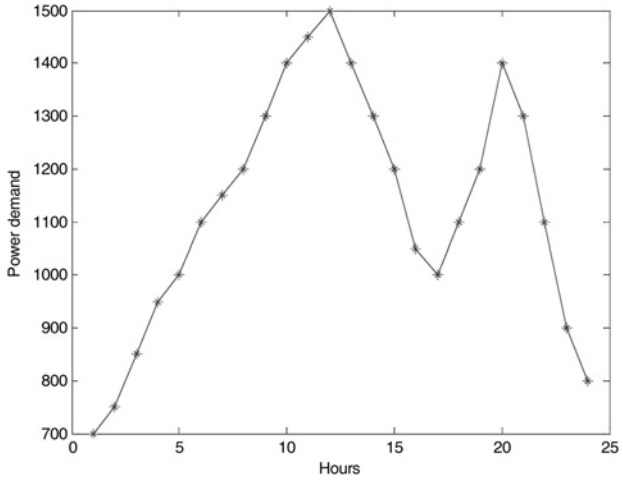



**Fig. 10** *Demanded power of the ten units*

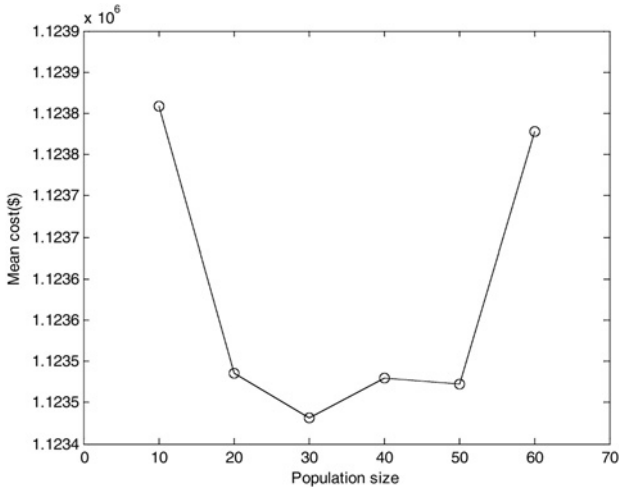



**Fig. 11** *Mean cost on the 20-unit system with different population sizes*

### 4.5 Fine tuning

For the obtained best solution, once certain zones are changed from uncommitted to committed, the specified units may change from the cold start-up to the hot start-up. This kind of zone is called grey zone [19]. As the hot start-up cost is much lower than the cold start-up cost, it is desirable that the units can start up with hot start-up cost if possible. Therefore grey zone can be found out for commitment if the modification can reduce the total cost. To reduce the total cost, the grey zone modification algorithm is utilised to fine tune the best solution as follows.

*Step 1:* The best solution in the population is chosen for fine tuning.

*Step 2:* Search the grey zones. If $T\text{off}_i^t = T\text{clod}_i + Td_i + 1$ and $u_i^{t+1} = 1$, $u_i^t$ is a grey zone.

*Step 3:* If grey zones exist, randomly select one grey zone and modify it to be committed.

*Step 4:* Economic load dispatch for the modified solution, and calculate its total cost.

*Step 5:* If the modified solution is better than the initial solution, the modified solution will replace the initial one go to step 1 until the procedure has repeated $n_{ft}$ times.

## 5 Numerical tests

We test our proposed algorithm on several benchmark UCP problems with different size of units, that is, 10, 20, 40, 60, 80 and 100 units [11]. The scheduling period is 24 h. The power demand and the unit characteristic of the 10-unit system are given in [11], which is shown in Fig. 10 and Table 1.

In Table 1, the initial status means the unit's on/off time before scheduling. A positive initial status indicates the number of hours the unit has been already on, and a negative status indicates the number of hour the unit has been already off. For 20, 40, 60, 80 and 100-unit systems, the system data are obtained by duplicating the base case (10-unit system), whereas the load demands are adjusted in proportion to the

**Table 1** Ten-unit system data

| Unit | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| $P_{\max}$, MW | 455 | 455 | 130 | 130 | 162 | 80 | 85 | 55 | 55 | 55 |
| $P_{\min}$, MW | 150 | 150 | 20 | 20 | 25 | 20 | 25 | 10 | 10 | 10 |
| $a_i$ | 1000 | 970 | 700 | 680 | 450 | 370 | 480 | 660 | 665 | 670 |
| $b_i$ | 16.19 | 17.26 | 16.6 | 16.5 | 19.7 | 22.26 | 27.74 | 25.92 | 27.27 | 27.79 |
| $c_i$ | 0.00048 | 0.00031 | 0.002 | 0.00211 | 0.00398 | 0.00712 | 0.00079 | 0.00413 | 0.00222 | 0.00713 |
| MUT, h | 8 | 8 | 5 | 5 | 6 | 3 | 3 | 1 | 1 | 1 |
| MDT, h | 8 | 8 | 5 | 5 | 6 | 3 | 3 | 1 | 1 | 1 |
| $\text{cost}_{\text{hot}}$ | 4500 | 5000 | 550 | 560 | 900 | 170 | 260 | 30 | 30 | 30 |
| $\text{cost}_{\text{cold}}$ | 9000 | 10 000 | 1100 | 1120 | 1800 | 340 | 520 | 60 | 60 | 60 |
| $T_{\text{cold}}$ | 5 | 5 | 4 | 4 | 4 | 2 | 2 | 0 | 0 | 0 |
| initial status | 8 | 8 | −5 | −5 | −6 | −3 | −3 | −1 | −1 | −1 |

**Table 2** Parameter evaluation of BNFO

| Cr | $\alpha = 0.1$ | $\alpha = 0.2$ | $\alpha = 0.3$ | $\alpha = 0.4$ | $\alpha = 0.5$ | $\alpha = 0.6$ |
|---|---|---|---|---|---|---|
| 0.1 | 1 123 879 | 1 123 431 | 1 123 796 | 1 123 583 | 1 123 678 | 1 123 904 |
| 0.2 | 1 123 958 | 1 123 537 | 1 123 926 | 1 124 053 | 1 123 574 | 1 124 137 |
| 0.3 | 1 123 797 | 1 123 683 | 1 123 722 | 1 124 101 | 1 124 304 | 1 124 141 |
| 0.4 | 1 124 100 | 1 123 735 | 1 124 343 | 1 124 146 | 1 124 438 | 1 124 302 |
| 0.5 | 1 124 100 | 1 123 998 | 1 124 062 | 1 124 191 | 1 124 400 | 1 124 381 |
| 0.6 | 1 124 115 | 1 123 940 | 1 124 129 | 1 124 250 | 1 124 407 | 1 124 392 |
| 0.7 | 1 124 228 | 1 123 869 | 1 124 363 | 1 124 416 | 1 124 358 | 1 124 405 |

system size. In all cases, the spinning reserve requirements are assumed to be 10% of the hourly demand. For each test case, 50 independent trials are conducted to evaluate the statistical results and the median convergence characteristics. Numerical tests have been coded in MATLAB on a computer with Core 2 CPU Q9650 and 4G RAM. In our algorithm, the maximum number of function evaluations is set to 20 000; $G_{gap}$ is 10 for the unit substitution; and $n_{ft}$ is set to 10 in the fine tuning. Other parameters population size *np*, learning rate $\alpha$ and the crossover probability *Cr* are evaluated in the following part.

### 5.1 Parameter analysis

The evaluation of parameter setting can illustrate their effects to the final results. There are mainly two parameters in NFO: learning rate $\alpha$ and the crossover probability *Cr*. The effect of $\alpha$ is to control the scale of search region, while *Cr* is used to control the convergence speed. For users, the two parameters need to be insensitive and problem independent. In this experiment, we have evaluated how $\alpha$ and *Cr* affect the performance of BNFO. Different settings of $\alpha$ and *Cr* are tested on the 20-unit UCP problem. In this experiment, the learning rate $\alpha$ is set to 0.1, 0.2, .0.3, 0.4, 0.5, 0.6 and 0.7. For each learning rate, *Cr* is set to 0.1, 0.2, 0.3, 0.4, 0.5 and 0.6 for comparisons. The population size is set to 30 in this experiment. With each pair of parameters, the mean values of 50 trial results are listed in Table 2. In this table, it can be noted that BNFO is not sensitive to these parameters and can obtain acceptable results for different settings. Furthermore, we find that the optimal parameters lie in the scale $\alpha \in [0.1, 0.5]$ and $Cr \in [0.1, 0.3]$. If $\alpha$ is larger, NFO may obtain trapped in the local minimum

**Table 3** Numerical comparison

| Method | Best cost | Mean cost | Worst cost | Std. dev. cost | Difference, % | Mean time, s |
|---|---|---|---|---|---|---|
| 10-unit | | | | | | |
| GA | 565 825 | – | 570 032 | – | 0.74 | 221 |
| EP | 564 551 | 565 352 | 566 231 | – | 0.30 | 100 |
| SA | 565 828 | 565 988 | 566 260 | – | 0.08 | 3 |
| DE | 563 977 | 564 028 | 564 241 | 103 | 0.05 | 3.6 |
| IPSO | 563 954 | 564 162 | 564 579 | 0 | 0.11 | – |
| IQEA | 563 977 | 563 977 | 563 977 | 0 | 0 | 15 |
| QBPSO | 563 977 | 563 977 | 563 977 | 0 | 0 | 18 |
| BNFO | 563 938 | 563 938 | 563 938 | 0 | 0 | 4 |
| 20-unit | | | | | | |
| GA | 1 126 243 | – | 1 132 059 | – | 0.52 | 733 |
| EP | 1 125 494 | 1 127 257 | 1 129 793 | – | 0.38 | 340 |
| SA | 1 126 251 | 1 127 955 | 1 129 112 | – | 0.25 | 17 |
| DE | 1 123 988 | 1 124 339 | 1 124 539 | 243 | 0.05 | 71 |
| IPSO | 1 125 279 | – | 1 127 643 | – | 0.21 | – |
| IQEA | 1 123 890 | 1 124 320 | 1 124 504 | 126 | 0.05 | 42 |
| QBPSO | 1 123 297 | 1 123 981 | 1 124 294 | 377 | 0.09 | 50 |
| BNFO | 1 123 297 | 1 123 431 | 1 123 563 | 112 | 0.0002 | 29 |
| 40-unit | | | | | | |
| GA | 2 251 911 | – | 2 259 706 | – | 0.35 | 2697 |
| EP | 2 249 093 | 2 252 612 | 2 256 086 | – | 0.31 | 1176 |
| SA | 2 250 063 | 2 252 125 | 2 254 539 | – | 0.20 | 88 |
| DE | 2 245 631 | 2 245 877 | 2 246 457 | 297 | 0.04 | 153 |
| IPSO | 2 248 163 | – | 2 252 117 | – | 0.18 | – |
| IQEA | 2 245 151 | 2 246 026 | 2 246 701 | 378 | 0.07 | 132 |
| QBPSO | 2 242 957 | 2 244 657 | 2 245 941 | 674 | 0.13 | 158 |
| BNFO | 2 242 957 | 2 243 241 | 2 244 237 | 367 | 0.005 | 92 |
| 60-unit | | | | | | |
| GA | 3 376 625 | – | 3 384 252 | – | 0.23 | 5840 |
| EP | 3 371 611 | 3 376 255 | 3 381 012 | – | 0.28 | 2267 |
| SA | – | – | – | – | – | – |
| DE | 3 366 502 | 3 367 166 | 3 367 612 | 383 | 0.03 | 257 |
| IPSO | 3 370 979 | – | 3 379 125 | – | 0.24 | – |
| IQEA | 3 365 003 | 3 365 667 | 3 366 223 | 309 | 0.04 | 273 |
| QBPSO | 3 361 980 | 3 363 763 | 3 365 707 | 796 | 0.11 | 328 |
| BNFO | 3 361 527 | 3 662 137 | 3 363 251 | 321 | 0.0004 | 193 |
| 80-unit | | | | | | |
| GA | 4 504 933 | – | 4 510 129 | – | 0.12 | 10 036 |
| EP | 4 498 479 | 4 505 536 | 4 512 739 | – | 0.32 | 3584 |
| SA | 4 498 076 | 4 501 156 | 4 503 987 | – | 0.13 | 405 |
| DE | 4 488 225 | 4 489 253 | 4 490 252 | 642 | 0.05 | 377 |
| IPSO | 4 495 032 | – | 4 508 943 | – | 0.31 | – |
| IQEA | 4 486 963 | 4 487 985 | 4 489 286 | 501 | 0.05 | 453 |
| QBPSO | 4 482 085 | 4 485 410 | 4 487 168 | 1054 | 0.11 | 554 |
| BNFO | 4 482 085 | 4 485 131 | 4 485 633 | 523 | 0.0008 | 331 |
| 100-unit | | | | | | |
| GA | 5 627 437 | – | 5 637 914 | | 0.19 | 15 733 |
| EP | 5 623 885 | 5 633 800 | 5 639 148 | | 0.27 | 6120 |
| SA | 5 617 876 | 5 624 301 | 5 628 506 | | 0.19 | 696 |
| DE | 5 608 603 | 5 609 174 | 5 610 160 | 700 | 0.03 | 485 |
| IPSO | 5 619 284 | – | 5 633 021 | | 0.24 | – |
| IQEA | 5 606 022 | 5 607 561 | 5 608 525 | 578 | 0.04 | 710 |
| QBPSO | 5 602 486 | 5 604 275 | 5 606 178 | 952 | 0.07 | 883 |
| BNFO | 5 602 433 | 5 603 120 | 5 605 678 | 561 | 0.0006 | 528 |




doi: 10.1049/iet-gtd.2012.0096

because of the excessive search. In contrast, if $Cr$ is larger, NFO is more difficult to converge to the global optimum regardless of the learning rate. In our comparison studies, $\alpha$ and $Cr$ are set as the optimal values 0.2 and 0.1, respectively.

The population size is determined through experiments for the 20-unit system with different population sizes. In this experiment, learning rate $\alpha$ and the crossover probability $Cr$ are set to 0.2 and 0.1 (as optimal values). We have evaluated the NFO's performance with population sizes 10, 20, 30, 40, 50 and 60. The mean values after 50 trials on each size are shown in Fig. 11. Under the same number of function evaluations, a small size means a large number of generations and a large size means a small number of generations. In Fig. 11, it can be noted that the solution quality is improved when increasing the population size at first. When the size is larger than 30, the solution quality turns worse and worse when increasing the population size. The reason may be that a relatively small population size makes the population easily converge to local optimum, and a relatively large population size increases the burden of computation time. The optimal population size is 30, which is used in the following comparison studies.

### 5.2 Comparison study

For the six UCPs, the best, mean, worst costs and the standard deviations obtained by improved quantum evolutionary algorithm (IQEA) [15], quantum-inspired binary PSO (QBPSO) [17], DE [19] and BNFO in 50 trials are summarised in Table 3. We also analyse the statistical difference as the percentage of deviation, that is, (Best–Worst)/Best × 100%. Our results are compared with the reported results using SA [8]; GA [11]; EP [13] and the improved PSO (IPSO) [16]. Furthermore, we have implemented three latest proposed algorithms, the IQEA [15]; and a QBPSO [17] and the enhanced DE [19]. The parameters of IQEA, QBPSO and DE follow the optimal settings recommended in their original papers. For fair comparisons, the maximal number of function evaluations in each algorithm is set to 20 000, which is the same as the setting in NFO.

From Table 3, it can be noted that BNFO performs superior to the compared algorithms, in terms of solution quality and CPU times especially on the large unit system. For the 10-unit system, the BNFO can find all solutions with the lowest costs and no deviation. For the 20-, 40-, 60-, 80- and 100-Unit systems, BNFO is competitive with the lowest mean cost compared with other algorithms. In terms of best cost, mean cost and worst cost, BNFO is better than GA, EP, SA, DE, IPSO and IQEA on all the UC problems. Compared with QBPSO, BNF shows more robust with lower mean cost, worst cost and smaller deviation. The best costs of BNFP and QBSO are almost the same.

CPU time may reflect the difficulty of practical implementations (especially online) when the number of unit increases. The mean CPU time shown in Table 3 may not be directly comparable because of different computers used. Therefore it is still substantial to compare BNFO with some recent algorithms [16–19] because of same level of CPU speed (better than Pentium IV). In Table 3, the CPU times of BNFO are much smaller than those of other algorithms except SA. Furthermore, it is worth noting that the CPU times of BNFO increase approximately linear with respect to the system size of UCP, which is favourable for large-scale UCP applications.

It can be concluded that the results of BNFO are more accurate and with smaller deviation compared with other algorithms. This is significant to minimise the cost of energy, and to maximise the efficiency of coal resource.

## 6 Conclusion

A new algorithm BNFO has been proposed to solve discrete optimisation problems, which is inspired by the local cooperation behaviour in biology. In BNFO, each individual is affected by the superior neighbour and the inferior neighbour. This effect has been modelled as neighbourhood field, which is approximately on the descending direction of the fitness function. BNFO is efficiently applied to solve the UCP. The propose method is a combination of BNFO and the conventional Lambda-iteration method, which includes some other constraints repairing heuristics. The total production costs of BNFO over the scheduled period are less expensive than the GA, EP, PSO and DE algorithms especially on the large number of generation units.

## 7 References


1 Senju, T., Miyagi, T., Saber, A.: 'Emerging solution of large-scale unit commitment problem by stochastic priority list', *Electr. Power Syst. Res.*, 2006, **76**, (5), pp. 283–292
2 Yi, J., Labadie, J.W., Stitt, S.: 'Dynamic optimal unit commitment and loading in hydropower system', *J. Water Resour. Plan. Manage.*, 2003, **129**, (5), pp. 388–398
3 Li, C., Johnson, R.B., Svoboda, A.J., Tseng, C., Hsu, E.: 'A robust unit commitment algorithm for hydro-thermal optimization', *IEEE Trans. Power Syst.*, 1998, **13**, pp. 1051–1056
4 Redondo, N.J., Conejo, A.J.: 'Short-term hydro-thermal coordination by Lagrangian relaxation: solution of the dual problem', *IEEE Trans. Power Syst.*, 1999, **14**, pp. 89–95
5 Ongsakul, W., Petcharaks, N.: 'Unit commitment by enhanced adaptive Lagrangian relaxation', *IEEE Trans. Power Syst.*, 2004, **19**, (1), pp. 621–628
6 Venkatesh, B., Jamtsho, T., Gooi, H.: 'Unit commitment – a fuzzy mixed integer linear programming solution', *IET Gener. Transm. Distrib.*, 2007, **1**, (5), pp. 836–846
7 Carrion, M., Arroyo, J.M.: 'A computationally efficient mixed-integer linear formulation for the thermal unit commitment problem', *IEEE Trans. Power Syst.,* 2006, **21**, (3), pp. 1371–1378
8 Simopoulos, D.N., Kavatza, S.D., Vournas, C.D.: 'Unit commitment by an enhanced simulated annealing algorithm', *IEEE Trans. Power Syst.*, 2006, **21**, (1), pp. 68–76
9 Rajan, C., Mohan, M.: 'An evolutionary programming-based tabu search method for solving the unit commitment problem', *IEEE Trans. Power Syst.*, 2004, **19**, (1), pp. 577–585
10 Swarup, K.S., Simi, O.: 'Neural computation using discrete and continuous Hopfield networks for power system economic dispatch and unit commitment', *Neurocomputing*, 2006, **70**, (1), pp. 119–129
11 Kazarlis, S.A., Bakirtzis, A.G., Petridis, V.: 'A genetic algorithm solution to the unit commitment problem', *IEEE Trans. Power Syst.*, 1996, **11**, (1), pp. 83–92
12 Swarup, K.S., Yamashiro, S.: 'Unit commitment solution methodology using genetic algorithm', *IEEE Trans. Power Syst.*, 2002, **17**, (1), pp. 87–91
13 Juste, K.A., Kita, H., Tanaka, E., Hasegawa, J.: 'An evolutionary programming solution to the unit commitment problem', *IEEE Trans. Power Syst.*, 1999, **14**, (4), pp. 1452–1459
14 Chen, H., Wang, X.: 'Cooperative coevolutionary algorithm for unit commitment', *IEEE Trans. Power Syst.*, 2002, **16**, (1), pp. 128–133
15 Jeong, Y.W., Park, J.B., Shin, J.R., Lee, K.Y.: 'A thermal unit commitment approach using an improved quantum evolutionary algorithm', *Electr. Power Compon. Syst.*, 2009, **37**, (7), pp. 770–786
16 Zhao, B., Guo, C.X., Bai, B.R., Cao, Y.J.: 'An improved particle swarm optimization algorithm for unit commitment', *Electr. Power Energy Syst.*, 2006, **28**, (7), pp. 482–490
17 Jeong, Y.W., Park, J.B., Jang, S.H., Lee, K.Y.: ' A new quantum-inspired binary PSO: application to unit commitment problems for power systems', *IEEE Trans. Power Syst.*, 2002, **25**, (3), pp. 1486–1495

18 Yuan, X., Nie, H., Su, A., Wang, L., Yuan, Y.: 'An improved binary particle swarm optimization for unit commitment problem', *Expert Syst. Appl.*, 2009, **36**, pp. 8049–8055

19 Yuan, X., Su, A., Nie, H., Yuan, Y., Wang, L.: 'Application of enhanced discrete differential evolution approach to unit commitment problem', *Energy Convers. Manage.*, 2009, **50**, pp. 2449–2456

20 Rajan, C.C.A., Mohan, M.R.: 'An Evolutionary programming-based tabu search method for solving the unit commitment problem', *IEEE Tran. Power Syst.*, 2004, **19**, (1), pp. 577–585

21 Victoire, T.A.A., Jeyakumar, A.E.: 'Unit commitment by a tabu-search-based hybrid-optimisation technique', *IEE Proc. Gener. Transm. Distrib.*, 2005, **152**, (4), pp. 563–574

22 Goldberg, D.E.: 'Genetic algorithms in search, optimization, and machine learning' (Addison-Wesley Professional, 1989, 1st edn.)

23 Eberhart, R.C., Kennedy, J.: 'A new optimizer using particle swarm theory'. Proc. sixth Int. Symp. on Micro machine and Human Science, Japan, October 1995, pp. 39–43

24 Storn, R., Price, K.: 'Differential evolution – a simple and efficient heuristic for global optimization over continuous space', *J. Global Optim.*, 1997, **11**, (4), pp. 341–359

25 Kennedy, J., Eberhart, R.C.: 'A discrete binary version of the particle swarm algorithm'. Proc. IEEE Int. Conf. on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation, October 1997, vol. 5, pp. 4104–4108

26 Gong, T, ., T., Tuson, A.: 'Differential evolution for binary encoding', *Soft Comput. Ind. Appl.*, 2007, **ASC 39**, pp. 251–262

27 Wu, Z., Chow, T.W.S.: 'Neighborhood field optimization', *J. Soft. Comput.*, DOI 10.1007/s00500-012-0955-9

28 Eslamian, M., Hosseinian, S.H., Vahidi, B.: 'Bacterial foraging-based solution to the unit-commitment problem', *IEEE Trans. Power Syst.*, 2009, **24**, (3), pp. 1478–1488

29 Xu, L., Chow, T.W.S.: 'Self-organizing potential field network: A new optimization algorithm', *IEEE Trans. Neural Netw.*, 2010, **21**, (9), pp. 1482–1495

30 Wu, S., Chow, T.W.S.: 'Self-organizing and self-evolving neurons: a new neural network for optimization', *IEEE Trans. Neural Netw.*, 2007, **18**, (2), pp. 385–396

31 Wood, A., Wollenberg, B.: 'Power generation, operation and control' (John Wiley & Sons, New York, 1996)





doi: 10.1049/iet-gtd.2012.0096