Available online at www.sciencedirect.com
SCIENCE DIRECT®
Electric Power Systems Research 72 (2004) 299–308
ELECTRIC POWER SYSTEMS RESEARCH
www.elsevier.com/locate/epsr

# Unit commitment by genetic algorithm with specialized search operators

Grzegorz Dudek*

*Institute of Electrical Power Engineering, Technical University of Czestochowa, Al. Armii Krajowej 17, Czestochowa 42-200, Poland*

Received 10 September 2003; received in revised form 9 January 2004; accepted 24 April 2004
Available online 14 July 2004

**Abstract**

An approach for solving the unit commitment problem based on genetic algorithm with new search operators is presented. These operators, specific to the problem, are mutation with a probability of bit change depending on load demand, production and start-up costs of the generating units and transposition. The method incorporates time-dependent start-up costs, demand and reserve constraints, minimum up and down time constraints and units power generation limits. Repair algorithms or penalty factors in the objective function are applied to the infeasible solutions. Numerical results showed an improvement in the solution cost compared to the results obtained from genetic algorithm with standard operators and other techniques.


*Keywords:* Unit commitment; Power generation dispatch; Genetic algorithms; Evolutionary computation; Combinatorial optimization

## 1. Introduction

Unit commitment (UC) is an important problem in the daily operation and planning of the power system. The objective of UC is to determine the optimal set of generating units to be in service during each interval of the scheduling period (a day or a week ahead), to meet system demand and reserve requirements at minimal production cost, subject to satisfying a large set of operating constraints. The solution of the UC problem is really a complex optimization problem with both discrete (unit commitment) and continuous (generation levels) variables. Generation levels for each feasible combination of units can be obtained by the economic dispatch procedure.

The optimal solution to the problem can be found by exhaustive enumeration of all feasible combinations of generating units. The computer execution time for this method is usually too immense for practical systems. Research efforts have concentrated on efficient, suboptimal UC algorithms which can be applied to realistic power systems. The solution methods being used to solve the UC problem can be grouped as follows [1,2]:

- heuristic methods such as priority list;
- classical optimization methods such as: dynamic programming, Lagrangian relaxation, branch-and-bound, linear programming, integer programming;
- artificial intelligence methods such as: expert systems, neural networks, simulated annealing, genetic algorithms.

The priority list method is easy to implement and the simplest of the UC methods. This method specifies the order in which units start up or shut down. The classical average full load cost index can be used to determine the priority commitment order. The quality of the solution is usually far from optimal due to the incomplete search of the solution space.

Many classical methods (dynamic and integer programming, branch-and-bound) suffer from the "curse of dimensionality" because the problem size and the solution time increase rapidly with the number of generating units to be committed. To reduce the search space several approaches have been developed. Most approaches are based on the priority list technique (dynamic programming–sequential combination, dynamic programming–truncated combination [3,4]). Lagrangian relaxation is considered the most realistic and efficient way for large-scale systems. Lagrangian relaxation has higher computational efficiency and is more flexible for handling different types of constraint compared with other approaches. However, because of the dual nature of the algorithm, its primary difficulty is associated

* Tel.: +48 34 3250 896; fax: +48 34 3250 803.
*E-mail address:* dudek@el.pcz.czest.pl (G. Dudek).

0378-7796/$ – see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.epsr.2004.04.014

with obtaining solution feasibility. Furthermore, the optimal value of the dual problem is not generally equal to that of the primal (original) problem.

In the expert system approach, the knowledge of experienced power system operators and UC experts is combined to create an expert system rule base. However, a great deal of operator interaction is required in this approach, making it inconvenient and time-consuming.

Neural networks (most often multilayer perceptrons), based on a database holding typical load curves and corresponding UC schedules, are trained to recognise the most economical UC schedule associated with the pattern of the current load curve [5,6]. If the neural network solution is not feasible for the entire UC period, it will be used as an initial starting point for a near-optimal solution.

There are many uncertainties involved in the planning and operation of power systems. Recently, there have been attempts to solve the UC problem using a possibilistic approach [7,8]. The key factors such as load demand and reserve margin are treated as fuzzy variables. A fuzzy decision system has been developed to select the units to be on or off based on these fuzzy variables.

Simulated annealing is a powerful technique to solve combinatorial optimization problems such as UC [9,10]. A complicated mathematical model of the problem under study is not needed with this method. The starting point can be any given solution and it will attempt to improve it. The final solution does not strongly depend on the initial solution; it has been theoretically proved to converge with the optimum solution, but although it does not need large computer memory, the convergence time of UC by simulated annealing is a limiting factor.

Genetic algorithms (GA) represent a class of stochastic adaptive search techniques and these are different from the above-mentioned methods. They are global optimization techniques that work with a coding of the parameter set, with both discrete and continuous functions. GA search from a population of points and they use probabilistic transition rules. A simple GA implementation using the standard crossover and mutation operators can locate near-optimal solutions. However, by adding problem-specific operators and by the proper choice of variables and their representation, satisfactory solutions to the UC problem can be obtained. In power systems GA have been recently applied for the solution of the unit commitment problem [11–13].

Artificial intelligence methods such as GA are still in development and seem to offer a promising solution to the UC problem. This article presents a GA with specialized operators to solve the UC problem. New effective operators are mutation and transposition. The fitness function is constructed as the summation of the objective function and penalty terms for some constraint violations. A repair algorithm is also used for infeasible solutions. The combinatorial optimization sub-problem is solved using the GA while the economic dispatch problem is solved via the conventional lambda-iteration method.

## 2. The mathematical model of unit commitment

The UC problem can be mathematically formulated as follows:*Objective function*:

$$F = \sum_{t=1}^{T}\sum_{i=1}^{N} \{\alpha_i(t)C_i[P_i(t)] + \alpha_i(t)[1-\alpha_i(t-1)]SC_i(t_{\text{off}i})\} \tag{1}$$

*Constraints*:

(a) Load balance

$$\forall t : \sum_{i=1}^{N}[\alpha_i(t)P_i(t)] = D(t) \tag{2}$$

(b) Unit power generation limits

$$\forall i, t : \alpha_i(t)P_{\text{min}i} \le P_i(t) \le \alpha_i(t)P_{\text{max}i} \tag{3}$$

(c) Set of unit power generation limits

$$\forall t : \sum_{i=1}^{N}[\alpha_i(t)P_{\text{min}i}] \le D(t) \tag{4}$$

$$\forall t : \sum_{i=1}^{N}[\alpha_i(t)P_{\text{max}i}] \ge D(t) + R(t) \tag{5}$$

(d) Minimum up/down time

$$\forall i : t_{\text{off}i} \ge t_{\text{down}i} \tag{6}$$

$$\forall i : t_{\text{on}i} \ge t_{\text{up}i} \tag{7}$$

where the variable production cost of unit $i$ at time $t$ $C_i[P_i(t)]$ is conventionally approximated by the quadratic function:

$$C_i(P_i) = a_i P_i^2 + b_i P_i + c_i \tag{8}$$

and the start-up cost of unit $i$ $SC_i(t_{\text{off}i})$ is expressed as a function of the number of hours the unit has been down:

$$SC_i(t_{\text{off}i}) = e_i \exp(-g_i t_{\text{off}i}) + f_i \exp(-h_i t_{\text{off}i}) \tag{9}$$

To take into account the costs connected with unit outage in time period $t$, in the event that it remains in an off state to the end of time period $T$, it is assumed that:

- unit start-up costs are evenly distributed over the number of hours of unit down time;
- unit start-up occurs at time period $\tau$, subtracted from the end of the optimization period $T$ ($\tau \in \{1, 2, 3, \dots\}$).

Taking these assumptions into account, unit start-up costs in time period $T$ (staying in down time until the end of time period $T$) are calculated using the formula:

$$SC_i(T-t) = \frac{SC_i(T-t+\tau)}{T-t+\tau}(T-t) \tag{10}$$

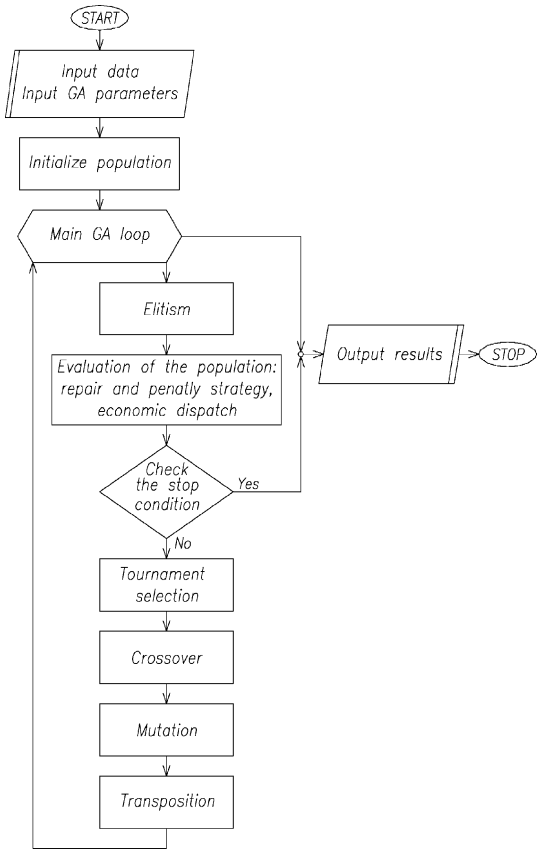


Fig. 1. GA flowchart.

## 3. The proposed genetic algorithm approach

The GA implementation consists of initialization, economic dispatch and cost calculations, reproduction, crossover, mutation, transposition and elitism. A flowchart of the algorithm is given in Fig. 1.

### 3.1. Representation

GA searches the solution space through the evolution of a population of candidate solutions. Each individual of the population is represented by a binary string. Each bit in the string represents the on/off status of the $i$th unit at the $t$th hour, $\alpha_i(t)$. For $N$ units and $T$ hours the string has $N \times T$ bits.

### 3.2. Economic dispatch and cost calculations: the procedure with infeasible individuals

Since the production cost is a quadratic function (convex and continuous), the economic dispatch problem is solved using a lambda-iteration method [14], based on the principle of equal incremental cost. Lambda-iteration method is used for various generating unit schedules obtained by the GA. Generation levels $P_i(t)$ determined in this procedure are used to calculate unit production costs (Eq. (8)) and the objective function (Eq. (1)).

If solutions violate a set of unit power generation limit constraints(Eq. (4) or (5)), the following repair algorithm is applied [15]. Let $\Omega_1(t)$ be a set of units in on status at the $t$th hour and let $\Omega_0(t)$ be a set of units in off status at the $t$th hour (these sets are determined by the solution string (Section 3.1)). If constraint (4) is not met at the $t$th hour, one unit $x$ is chosen from the $\Omega_1(t)$ and its status is reset to off at the $t$th hour (e.g. a bit in the solution string representing the status of unit $x$ at the $t$th hour is changed from 1 to 0). Similarly, if constraint (5) is not met, the status of one unit $y$ from the $\Omega_0(t)$ is changed. Repair can be greedy—$x$ is the most economical unit, e.g. one having the lowest incremental cost at full load, whereas $y$ is the least economical unit, or random—units $x$ and $y$ are chosen at random from sets $\Omega_1(t)$ and $\Omega_0(t)$, respectively. The repair algorithm is activated for every hour $t$, until the moment constraints (4) and (5) are met.

The repaired version of strings can be used either for evaluation only, or it can also replace the original strings. A so-called 5%-rule [16] states that in many combinatorial optimization problems, an evolutionary computation technique with repair algorithm provides the best result when 5% of repaired strings replace their infeasible originals.

For solutions which violate the minimum up/down time constraint (6) or (7), a penalty function is created [17]:

$$F' = M\left\{1 + m\sum_{i=1}^{N}[g(i) + h(i)]\right\} \tag{11}$$

where $g(i)$ is calculated as follows:

$$g(i) = \sum_{k=1}^{n_{\text{down}i}} \{\beta_i(k)[t_{\text{down}i} - t_{\text{off}i}(k)]\} \tag{12}$$

$\beta_i(k)$ is expressed as follows:

$$\beta_i(k) = \begin{cases} 1 & \text{if } t_{\text{off}i}(k) < t_{\text{down}i} \\ 0 & \text{if } t_{\text{off}i}(k) \geq t_{\text{down}i} \text{v} \tau_{\text{on}i}(k) > T \end{cases} \tag{13}$$

$h(i)$ is given by:

$$h(i) = \sum_{k=1}^{n_{\text{up}i}} \{\gamma_i(k)[t_{\text{up}i} - t_{\text{on}i}(k)]\} \tag{14}$$

$\gamma_i(k)$ is calculated as follows:

$$\gamma_i(k) = \begin{cases} 1 & \text{if } t_{\text{on}i}(k) < t_{\text{up}i} \\ 0 & \text{if } t_{\text{on}i}(k) \geq t_{\text{up}i} \text{v} \tau_{\text{off}i}(k) > T \end{cases} \tag{15}$$

The substitute cost function (11) ensures a worse valuation of individuals violating constraint (6) or (7) from feasible individuals. This function is linearly dependent on the level of violation of constraints (6) and (7). At the starting phase of the evolution process the level of violation of these constraints is minimized. At a certain point in the process individuals that are feasible according to these constraints start to appear and become the majority in the population. Because binary tournament, not proportional selection (e.g.

roulette wheel selection) is used, the feasible individuals do not strongly dominate, which allows the avoidance of a premature convergence of the population into a superindividual.

### 3.3. Stopping criterion

The main GA loop is terminated when there is no significant improvement in the solution after a pre-specified number of generations or when the maximum number of generations is reached.

### 3.4. Selection, crossover, elitism

The tournament selection method is used with tournament sizes of 2. Tournament selection is simple to implement and has none of the disadvantages of the roulette wheel selection method (it does not require scaling of the fitness function, and the fitness function values can be negative). An elitism strategy is also used which copies the best parent individual into the next population. The recombination method is one-point crossover, multi-point crossover or uniform crossover [18]. Crossover occurs with probability $p_c$.

### 3.5. Mutation

In the classic mutation method, the probability of bit mutation (on or off state) does not depend on the unit production cost, its start-up cost, or load demand. Therefore this operator will turn off economical units at peak load as well as less economical units at minimum value of the load curve with the same probability. This leads the algorithm to "wander" and results in a much less effective search of the solution space. In the proposed method of mutation, probability of mutation is made dependent on the necessity of meeting the load demand of the number of units, cost of unit production and its start-up costs [17]. The probability of a bit change from 0 to 1, depending on the number of units necessary to meet load demand at moment $t$, is calculated in the formula:

$$p_{\mathrm{up1}}(t) = q_1 + (1 - q_1)\frac{n_{\min}(t) + n_{\max}(t)}{2N} \qquad 1 \le t \le T \tag{16}$$

where $q_1 \le p_{\mathrm{up1}} \le 1$.

If through $L_1$ we denote a list of units sorted in decreasing order in terms of their upper generation limit $P_{\max}$, then the minimum number of units necessary to meet load demand is obtained by summing the $P_{\max}$ of succeeding units from list $L_1$ until the sum exceeds load demand and the spinning reserve:

$$n_{\min}(t) = \min_{\sum_{i=L_1(1)}^{L_1(n)} P_{\max i} \ge D(t)+R(t)} \{n\} \tag{17}$$

where $n$ is the auxiliary variable which denotes the number of units.

The maximum number of units necessary to meet load demand is obtained as follows: a list $L_2$ of units sorted in ascending order in terms of their lower generation limit $P_{\min}$ is introduced. The $P_{\min}$ of succeeding units from list $L_2$ is summed which gives the maximum number of units $n$, at which the sum does not exceed load demand:

$$n_{\max}(t) = \max_{\sum_{i=L_2(1)}^{L_2(n)} P_{\min i} \le D(t)} \{n\} \tag{18}$$

Parameter $q_1$ in formula (16) has the function of limiting the range of probability $p_{\mathrm{up1}}$.

The probability of a bit change from 0 to 1 is dependent on unit production costs as follows:

$$p_{\mathrm{up2}}(i) = q_2 + (1 - q_2)\frac{u_i - u_{\min}}{u_{\max} - u_{\min}} \qquad 1 \le i \le N \tag{19}$$

where:

$$u_i = \frac{\min_{j=1,2,\ldots,N}\{C_j(P_{\max j})/P_{\max j}\}}{C_i(P_{\max i})/P_{\max i}} \tag{20}$$

$$u_{\min} = \min_{i=1,2,\ldots,N}\{u_i\} \tag{21}$$

$$u_{\max} = \max_{i=1,2,\ldots,N}\{u_i\} \tag{22}$$

and $q_2 \le p_{\mathrm{up2}} \le 1$.

The probability $p_{\mathrm{up2}}$ has the minimum value, equal to $q_2$ for units of the greatest production cost per unit at maximum load, and the maximum value, equal to 1, for units of the lowest production cost per unit at maximum load.

The sum of both probabilities of bit mutation, representing the state of unit $i$ at moment $t$, from 0 to 1, is proposed by calculating the formula:

$$p_{\mathrm{up}}(i,t) = \begin{cases} 1 & \text{if } p_{\mathrm{up1}}(t) + p_{\mathrm{up2}}(i) - \dfrac{1+q_2}{2} > 1 \\ 0 & \text{if } p_{\mathrm{up1}}(t) + p_{\mathrm{up2}}(i) - \dfrac{1+q_2}{2} < 0 \\ p_{\mathrm{up1}}(t) + p_{\mathrm{up2}}(i) - \dfrac{1+q_2}{2} & \text{otherwise} \end{cases} \tag{23}$$

where $\max\left(0, q_1 - \frac{1-q_2}{2}\right) \le p_{\mathrm{up}} \le 1$.

An analogous probability of a bit change from 1 to 0, denoting a unit being turned off, is dependent on the number of units necessary to meet load demand according to the formula:

$$p_{\mathrm{down1}}(t) = 1 - (1 - r_1)\frac{n_{\min}(t) + n_{\max}(t)}{2N} \qquad 1 \le t \le T \tag{24}$$

where $r_1 \le p_{\mathrm{down1}} \le 1$.

The probability of a bit change from 1 to 0, depending on the production costs of unit $i$ are obtained using the formula:

$$p_{\mathrm{down2}}(i) = 1 - (1 - r_2)\frac{u_i - u_{\min}}{u_{\max} - u_{\min}} \qquad 1 \le i \le N \tag{25}$$

where $r_2 \le p_{\mathrm{down2}} \le 1$.

For units of the lowest production cost per unit at maximum load the probability $p_{down2}$ assumes the maximum value, equal to 1, whereas for units of the highest production cost per unit at maximum load, it assumes the minimum value of $r_2$.

The dependence of the probability of bit mutation from 1 to 0 on unit $i$ start-up cost is defined by the formula:

$$p_{down3}(i) = r_3 + (1 - r_3)\frac{v_i - v_{min}}{v_{max} - v_{min}} \qquad 1 \le i \le N \quad (26)$$

where:

$$v_i = \frac{\min_{j=1,2,\dots,N}\{SC_j(t_{off\,x})\}}{SC_i(t_{off\,x})} \quad (27)$$

$$v_{min} = \min_{i=1,2,\dots,N}\{v_i\} \quad (28)$$

$$v_{max} = \max_{i=1,2,\dots,N}\{v_i\} \quad (29)$$

and $r_3 \le p_{down3} \le 1$.

The probability $p_{down3}$ assumes a minimum value of $r_3$ for units with the highest start-up costs after down time $t_{off\,x}$, and a maximum value of 1 for units with the lowest start-up costs.

The sum of probabilities of bit mutation, representing the state of unit $i$ at moment $t$, from 1 to 0 is proposed by calculating the formula:

$$p_{down}(i,t) = \begin{cases} 1 \text{ if } p_{down1}(t) + p_{down2}(i) + p_{down3}(i) \\ \quad -\dfrac{1+r_2}{2} - \dfrac{1+r_3}{2} > 1 \\ 0 \text{ if } p_{down1}(t) + p_{down2}(i) + p_{down3}(i) \\ \quad -\dfrac{1+r_2}{2} - \dfrac{1+r_3}{2} < 0 \\ p_{down1}(t) + p_{down2}(i) + p_{down3}(i) \\ \quad -\dfrac{1+r_2}{2} - \dfrac{1+r_3}{2} \text{ otherwise} \end{cases} \quad (30)$$

where $\max\left(0, r_1 - \frac{1-r_2}{2} - \frac{1-r_3}{2}\right) \le p_{down} \le 1$.

The values of parameters $q_1$, $q_2$, $r_1$, $r_2$, $r_3$ and $t_{off\,x}$ are chosen heuristically. For $q_1 = 1$ probability $p_{up1}(t)$ does not depend on the load demand and is equal to 1 for each unit $i$ and each hour $t$. While $q_1 = 0$ the probability $p_{up1}(t)$ is the most diversified, dependent on the load demand within time period $T$, e.g. for $q_1 = 0$ and load data from Table 4 considered in the second example (Section 4.2) the $p_{up1}$ changes in the range from 0.6667 (for the minimum load demand at time $t = 4$) to 0.9583 (for the peak load demand at time $t = 18$). The larger the value of $q_1$ ($0 \le q_1 \le 1$) the more the range of the $p_{up1}(t)$ narrows and nears 1, which means a reduction in the influence of the load demand value on the probability of unit start-up. The greatest "selective pressure" is acquired for $q_1 = 0$ and such a value is recommended.

The component $p_{up2}(i) - (1 + q_2)/2$ in formula (23) signifies the correction added to probability $p_{up1}(t)$ which allows the differentiation of the probability of unit start-up from unit production costs. This correction changes in the range from $-(1 + q_2)/2$ (for the unit of the highest production cost) to $(1 + q_2)/2$ (for the unit of the lowest production cost). If $q_2 = 1$ the probability of unit start-up $p_{up}(i, t)$ is not dependent on the production costs, whereas if $q_2 = 0$ this dependence is the greatest—the correction assumes values from the range $[-0.5, 0.5]$. The value of $q_2 = 0.8$ gives the range of correction $[-0.1, 0.1]$ and seems to be a reasonable compromise.

The formula (30) includes corrections differentiating the probability of the unit shut-down $p_{down}(i, t)$ from unit production costs: $p_{down2}(i) - (1 + r_2)/2$ and unit start-up costs: $p_{down3}(i) - (1 + r_3)/2$. The influence of the production costs on $p_{down}(i, t)$ is the greatest if $r_2 = 0$ (then the correction range is from $-0.5$ for the unit of the lowest production cost to 0.5 for the unit of the highest production cost). The influence of the start-up costs on $p_{down}(i, t)$ is the greatest if $r_3 = 0$ (then the correction range is from $-0.5$ for the unit of the highest start-up cost after down time $t_{off\,x}$ to 0.5 for the unit of the lowest start-up cost after down time $t_{off\,x}$). If $r_2 = r_3 = 0.9$ the range of each correction is $[-0.05, 0.05]$ (jointly for both corrections $[-0.1, 0.1]$), which means an equal influence of production and start-up costs on the probability of unit shut-down.

The parameter $t_{off\,x}$ means expected unit down time. If start-up costs change for each unit uniformly, i.e. $v_i$ is constant for each unit, apart from down time $t_{off\,x}$ (just like in the second example, Section 4.2) this parameter is not important. If the start-up curves (Eq. (9)) cross, it is safer not to take into account the start-up cost assuming $r_3 = 1$. In other cases, for different $t_{off\,x}$ different probabilities $p_{down3}(i)$ are obtained, but if the unit order with respect to start-up cost is constant for different $t_{off\,x}$, the unit order with respect to values of $p_{down3}(i)$ is constant as well.

As in the case of many others GA parameters (e.g. population size, probability of crossover and mutation) there are no hard rules for setting up the above parameters of mutation method. In accordance with what was written above the advisable values of these parameters are $q_1 = r_1 = 0$, $q_2 = 0.8$, $r_2 = r_3 = 0.9$, $t_{off\,x} = 8$ if $v_i$ is constant for each unit, apart from down time $t_{off\,x}$ or $q_1 = r_1 = 0$, $q_2 = r_2 = 0.8$, $r_3 = 1$ in other cases. The probabilities $p_{up}$ and $p_{down}$ for the example defined in Section 4.2 are shown in Fig. 2.

If the bit $b(i, t)$ chosen for mutation, which represents the state of unit $i$ at moment $t$, changes its value from 0 to 1 (from 1 to 0) and the bits representing the state of unit $i$ at neighbouring moments $(t - 1)$ and $(t + 1)$ have the same value as bit $b(i, t)$ before mutation, then the probability of a change in the state of unit $i$ for these moments is analysed. If $p_{up}(i, t - 1) > p_{up}(i, t + 1)$ (or $p_{down}(i, t - 1) > p_{down}(i, t + 1)$ in the case of outage) then the value of bit $b(i, t - 1)$ and succeeding bits $b(i, t - 2)$, $b(i, t - 3)$, …, is changed, on condition that they are of the same value as bit $b(i, t)$ before mutation. A bit with the opposite value finishes this process. If $p_{up}(i, t - 1) < p_{up}(i, t + 1)$ (or $p_{down}(i, t - 1) < p_{down}(i, t + 1)$) then bits $b(i, t + 1)$, $b(i, t + 2)$ and so on, are changed analogously. A change in succeeding bits

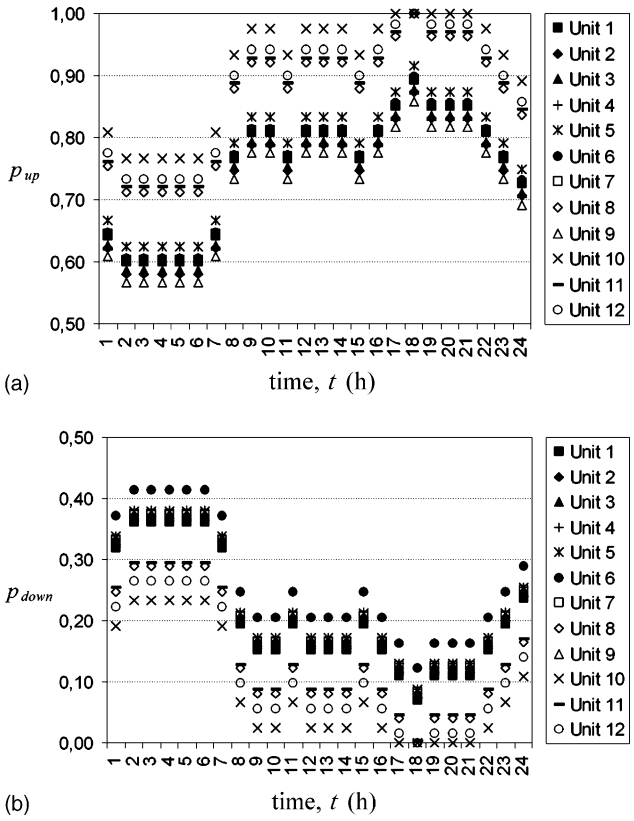


Fig. 2. (a) The probability of a bit change from 0 to 1, $p_{up}$; and (b) the probability of a bit change from 1 to 0, $p_{down}$, for the problem defined in Section 4. Assumed: $q_1 = r_1 = 0$, $q_2 = 0.8$, $r_2 = r_3 = 0.9$, $t_{off\,x} = 8$.

of the same value means a change in the off state or on state of units. This mechanism, suggested in [9] as a solution to the problem of UC using simulated annealing, allows for the avoidance of cases of multiple changes in the on state or off

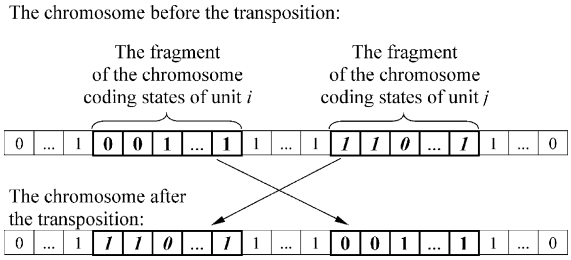


Fig. 3. An illustration of the transposition.

state of units in period $T$ and quickens the convergence of GA.

### 3.6. Transposition

A transposition operation is introduced [17] which exchanges fragments of the chromosomes that encode the states (during period $T$) of two randomly chosen units. This operation is shown in Fig. 3.

This transposition can considerably help the evolution process, particularly in the last phase, penetrating the local minimums by changing the work states of pairs of units.

## 4. Application examples

The genetic algorithm for the UC problem described above was implemented in Matlab and has been applied to a practical power system with 3 and 12 units. The scheduling time horizon for all cases is 24 h. These experiments were done on a personal computer with a Pentium III 800 MHz processor.

The unit and load data can be found in Tables 3 and 4 and in Appendix A.

Table 1
Comparison of GA variants for the 3-unit test system

| GA variant | Greedy repair | Random repair | Standard mutation | Proposed mutation | Transposition | One-point crossover | Multi-point crossover | Uniform crossover | $f_{opt}$ | $n_{opt}$ | $\sigma_{opt}$ | $t_{opt}$ (s) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | 1 | 0.5 | | | × | | | 0.13 | 1300 | 455 | 64 |
| 2 | | 1 | 0.5 | | 0.25 | × | | | 1.00 | 1435 | 397 | 67 |
| 3 | 1 | | 0.5 | | | × | | | 0.87 | 866 | 355 | 41 |
| 4 | 1 | | 0.5 | | 0.25 | × | | | 1.00 | 970 | 336 | 45 |
| 5 | 0.05 | | 0.5 | | 0.25 | × | | | 1.00 | 880 | 356 | 44 |
| 6 | 1 | | 0.5 | | 0.5 | × | | | 1.00 | 845 | 303 | 41 |
| 7 | 1 | | 0.25 | | 0.25 | × | | | 1.00 | 905 | 451 | 46 |
| 8 | | 1 | | 0.5 | | × | | | 0.90 | 2600 | 2600 | 122 |
| 9 | | 1 | | 0.5 | 0.25 | × | | | 1.00 | 870 | 338 | 48 |
| 10 | 1 | | | 0.5 | | × | | | 1.00 | 280 | 63 | 16 |
| 11 | 1 | | | 0.5 | 0.25 | × | | | 1.00 | 270 | 89 | 13 |
| 12 | 0.05 | | | 0.5 | 0.25 | × | | | 1.00 | 370 | 118 | 22 |
| 13 | 1 | | | 0.5 | 0.5 | × | | | 1.00 | 330 | 101 | 19 |
| 14 | 1 | | | 0.25 | 0.25 | × | | | 1.00 | 345 | 86 | 20 |
| 15 | 1 | | | 0.5 | 0.25 | | 2 | | 1.00 | 302 | 132 | 45 |
| 16 | 1 | | | 0.5 | 0.25 | | 5 | | 1.00 | 335 | 140 | 32 |
| 17 | 1 | | | 0.5 | 0.25 | | | 0.1 | 1.00 | 308 | 200 | 29 |
| 18 | 1 | | | 0.5 | 0.25 | | | 0.3 | 1.00 | 321 | 112 | 31 |
| 19 | 1 | | | 0.5 | 0.25 | | | 0.5 | 1.00 | 272 | 111 | 25 |

Table 2
The optimal power sharing $P_{opt}$ (MW) of example 1

| Hour | $P_1$ | $P_2$ | $P_3$ |
|---|---|---|---|
| 1 | 252.75 | 0.00 | 234.75 |
| 2 | 238.74 | 0.00 | 221.26 |
| 3 | 239.25 | 0.00 | 221.75 |
| 4 | 233.64 | 0.00 | 216.36 |
| 5 | 235.81 | 0.00 | 218.44 |
| 6 | 243.83 | 0.00 | 226.17 |
| 7 | 253.01 | 0.00 | 234.99 |
| 8 | 317.09 | 0.00 | 296.66 |
| 9 | 236.95 | 211.52 | 219.53 |
| 10 | 237.54 | 212.52 | 220.11 |
| 11 | 232.33 | 207.07 | 215.10 |
| 12 | 244.71 | 219.03 | 227.01 |
| 13 | 250.85 | 224.98 | 232.92 |
| 14 | 250.85 | 224.98 | 232.92 |
| 15 | 222.95 | 197.98 | 206.07 |
| 16 | 234.04 | 208.72 | 216.74 |
| 17 | 283.44 | 256.52 | 264.29 |
| 18 | 307.59 | 279.88 | 287.53 |
| 19 | 292.66 | 265.43 | 273.16 |
| 20 | 289.93 | 262.79 | 270.53 |
| 21 | 277.39 | 250.65 | 258.46 |
| 22 | 253.67 | 227.70 | 235.63 |
| 23 | 215.01 | 190.31 | 198.43 |
| 24 | 187.50 | 180.00 | 180.00 |

### 4.1. First example

The first example includes three generating units (unit nos. 1, 2 and 3 from Table 3) and load data from Table 4. Table 1 shows the comparison of results obtained for various variants of GA with greedy or random repair, with or without transposition including:

1. standard binary mutation and one-point crossover (rows 1–7);
2. the proposed method of mutation and one-point crossover (rows 8–14);
3. the proposed method of mutation and multi-point crossover or uniform crossover (rows 15–19).

For each variant of GA the following were defined:

- the frequency of optimum solution $f_{opt}$, found by GA;
- the average number of evaluations necessary to find the optimum solution $n_{opt}$, and their standard deviation $\sigma_{opt}$,
- the average computational time necessary to find the optimum solution $t_{opt}$.

The above values are defined according to 30 executions of the algorithm and GA parameters outlined below.

The following GA parameters were used in the first example:

- population size: $P_{size} = 50$;
- maximum no. of generations: $N_g = 200$;
- probability of crossover: $p_c = 0.9$;
- parameters of the proposed mutation: $q_1 = r_1 = 0$, $q_2 = 0.8$, $r_2 = r_3 = 0.9$, $t_{offx} = 8$;
- constant $M$ in formula (11):

$$M = T\sum_{i=1}^{N} C_i(P_{maxi}) \tag{31}$$

- parameter $m$ in formula (11), when (disproportional) tournament selection is used, must only fulfill the constraint $m > 0$.

The remaining GA parameters were variable, i.e.:

- the probability of replacement in repair algorithms: $p_r = 0.05$ or 1;
- the expected number of individual mutations: $n_m = 0.25$ or 0.5;
- the expected number of individual transpositions: $n_t = 0$, 0.25 or 0.5;
- the number of cut points in multi-point crossover: $n_{cp} = 2$ or 5;
- probability of swapping in uniform crossover: $p_s = 0.1$, 0.3 or 0.5.

These parameters were entered into the appropriate cells in Table 1. For example, row 1 shows the GA variant with

Table 3
Characteristics and initial state of units

| Unit | Initial status[a] (h) | $a$ (\$/MW$^2$ h) | $b$ (\$/MW h) | $c$ (\$/h) | $e$ (\$) | $f$ (\$) | $g$ (h$^{-1}$) | $h$ (h$^{-1}$) |
|---|---|---|---|---|---|---|---|---|
| 1 | On/-24 | 0.004531 | 7.3968 | 643.24 | −2889.45 | 5466.28 | 0.3680 | −0.0112 |
| 2 | On/-4 | 0.004683 | 7.5629 | 666.27 | −2893.81 | 5474.51 | 0.3680 | −0.0112 |
| 3 | On/-4 | 0.004708 | 7.4767 | 672.77 | −2888.84 | 5465.13 | 0.3680 | −0.0112 |
| 4 | On | 0.004880 | 7.4742 | 686.58 | −2882.77 | 5453.66 | 0.3680 | −0.0112 |
| 5 | On | 0.004214 | 7.2995 | 601.53 | −2863.94 | 5418.07 | 0.3680 | −0.0112 |
| 6 | On | 0.004582 | 7.3102 | 641.99 | −2843.13 | 5378.74 | 0.3680 | −0.0112 |
| 7 | On | 0.004267 | 7.5494 | 609.07 | −2876.16 | 5441.15 | 0.3680 | −0.0112 |
| 8 | On | 0.003572 | 6.6577 | 531.63 | −2903.29 | 5492.22 | 0.3680 | −0.0112 |
| 9 | On | 0.004788 | 7.7184 | 678.40 | −2892.73 | 5472.47 | 0.3680 | −0.0112 |
| 10 | On | 0.003485 | 6.2115 | 503.60 | −2928.65 | 5540.14 | 0.3680 | −0.0112 |
| 11 | On | 0.003658 | 6.5492 | 528.19 | −2894.88 | 5476.32 | 0.3680 | −0.0112 |
| 12 | On | 0.003671 | 6.4137 | 527.81 | −2915.53 | 5515.34 | 0.3680 | −0.0112 |

[a] "On" indicates unit is in the on-state, "-$x$" indicates unit is in the off-state for $x$ hours. In the first example units 1, 2 and 3 are in the on-state, whereas in the second example they are in the off-state.

Table 4
Load demand $D$ (MW)

| Hours | $D$ (first example) | $D$ (second example) |
|---|---|---|
| 1 | 487.50 | 1950.00 |
| 2 | 460.00 | 1840.00 |
| 3 | 461.00 | 1844.00 |
| 4 | 450.00 | 1800.00 |
| 5 | 454.25 | 1817.00 |
| 6 | 470.00 | 1880.00 |
| 7 | 488.00 | 1952.00 |
| 8 | 613.75 | 2455.00 |
| 9 | 668.00 | 2672.00 |
| 10 | 669.75 | 2679.00 |
| 11 | 654.50 | 2618.00 |
| 12 | 690.75 | 2763.00 |
| 13 | 708.75 | 2835.00 |
| 14 | 708.75 | 2835.00 |
| 15 | 627.00 | 2508.00 |
| 16 | 659.50 | 2638.00 |
| 17 | 804.25 | 3217.00 |
| 18 | 875.00 | 3500.00 |
| 19 | 831.25 | 3325.00 |
| 20 | 823.25 | 3293.00 |
| 21 | 786.50 | 3146.00 |
| 22 | 717.00 | 2868.00 |
| 23 | 603.75 | 2415.00 |
| 24 | 547.50 | 2190.00 |

the probability of replacement in the random repair algorithm equal to 1, a standard mutation with expected individual mutation equal to 0.5, one-point crossover, without transposition.

The optimum solution, $P_{opt}$, was calculated by enumerating all possible combinations of the generating units, shown in Table 2. The cost of this solution is: total cost = \$179116, production cost = \$173282, start-up cost = \$5834.

Table 1 shows that greedy repair is better than random repair. No important differences were observed in the working of the algorithm with a probability of replacement in repair algorithms at a value of 0.05 or 1. The proposed mutation allows the obtainment of an optimum solution in nearly every case, in a considerably quicker time than using standard mutation. All the crossover methods gave similar results, but one-point crossover functioned the fastest. In all cases transposition raised the efficiency of the algorithm, in some cases considerably (e.g. compare variants 1, 2, 3 and 4 in Table 1).

### 4.2. Second example

The second example includes 12 generating units (Table 3) and load data from Table 4. On the basis of preliminary experiments and results from the first example the following GA parameters were assumed:

- population size: $P_{size} = 100$;
- maximum no. of generations: $N_g = 1000$;
- probability of replacement in repair algorithm: $p_r = 1$;
- expected number of individual mutations: $n_m = 0.5$;
- expected number of individual transpositions: $n_t = 0.25$;
- greedy repair;
- one-point crossover;
- $p_c$, $q_1$, $r_1$, $q_2$, $r_2$, $r_3$, $t_{offx}$, $M$, $m$ are assumed to be of the same value as in the first example.

Table 5
The best power sharing $P_{best}$ (MW) of example 2

| Hour | $P_1$ | $P_2$ | $P_3$ | $P_4$ | $P_5$ | $P_6$ | $P_7$ | $P_8$ | $P_9$ | $P_{10}$ | $P_{11}$ | $P_{12}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.00 | 0.00 | 0.00 | 180.00 | 180.00 | 180.00 | 180.00 | 282.01 | 0.00 | 350.00 | 290.27 | 307.72 |
| 2 | 0.00 | 0.00 | 0.00 | 180.00 | 180.00 | 180.00 | 180.00 | 253.57 | 0.00 | 323.91 | 262.49 | 280.03 |
| 3 | 0.00 | 0.00 | 0.00 | 180.00 | 180.00 | 180.00 | 180.00 | 254.57 | 0.00 | 324.95 | 263.47 | 281.01 |
| 4 | 0.00 | 0.00 | 0.00 | 180.00 | 180.00 | 180.00 | 180.00 | 243.50 | 0.00 | 313.60 | 252.66 | 270.24 |
| 5 | 0.00 | 0.00 | 0.00 | 180.00 | 180.00 | 180.00 | 180.00 | 247.78 | 0.00 | 317.98 | 256.84 | 274.40 |
| 6 | 0.00 | 0.00 | 0.00 | 180.00 | 180.00 | 180.00 | 180.00 | 263.63 | 0.00 | 334.22 | 272.32 | 289.83 |
| 7 | 0.00 | 0.00 | 0.00 | 180.00 | 180.00 | 180.00 | 180.00 | 282.69 | 0.00 | 350.00 | 290.93 | 308.38 |
| 8 | 0.00 | 0.00 | 0.00 | 234.78 | 292.61 | 267.92 | 259.69 | 350.00 | 0.00 | 350.00 | 350.00 | 350.00 |
| 9 | 0.00 | 0.00 | 238.06 | 229.95 | 287.03 | 262.78 | 254.18 | 350.00 | 0.00 | 350.00 | 350.00 | 350.00 |
| 10 | 0.00 | 0.00 | 239.40 | 231.25 | 288.53 | 264.16 | 255.66 | 350.00 | 0.00 | 350.00 | 350.00 | 350.00 |
| 11 | 0.00 | 0.00 | 227.70 | 219.96 | 275.45 | 252.14 | 242.75 | 350.00 | 0.00 | 350.00 | 350.00 | 350.00 |
| 12 | 0.00 | 0.00 | 255.51 | 246.80 | 306.53 | 280.72 | 273.44 | 350.00 | 0.00 | 350.00 | 350.00 | 350.00 |
| 13 | 0.00 | 0.00 | 269.32 | 260.12 | 321.97 | 294.91 | 288.68 | 350.00 | 0.00 | 350.00 | 350.00 | 350.00 |
| 14 | 0.00 | 0.00 | 269.32 | 260.12 | 321.97 | 294.91 | 288.68 | 350.00 | 0.00 | 350.00 | 350.00 | 350.00 |
| 15 | 0.00 | 0.00 | 206.60 | 199.59 | 251.88 | 230.46 | 219.47 | 350.00 | 0.00 | 350.00 | 350.00 | 350.00 |
| 16 | 0.00 | 0.00 | 231.54 | 223.66 | 279.74 | 256.08 | 246.98 | 350.00 | 0.00 | 350.00 | 350.00 | 350.00 |
| 17 | 0.00 | 281.04 | 288.69 | 278.81 | 343.60 | 314.81 | 310.05 | 350.00 | 0.00 | 350.00 | 350.00 | 350.00 |
| 18 | 0.00 | 284.45 | 292.07 | 282.07 | 347.38 | 318.28 | 313.78 | 350.00 | 261.97 | 350.00 | 350.00 | 350.00 |
| 19 | 0.00 | 260.02 | 267.77 | 258.63 | 320.23 | 293.31 | 286.97 | 350.00 | 238.07 | 350.00 | 350.00 | 350.00 |
| 20 | 0.00 | 255.55 | 263.33 | 254.34 | 315.27 | 288.75 | 282.06 | 350.00 | 233.70 | 350.00 | 350.00 | 350.00 |
| 21 | 0.00 | 235.03 | 242.92 | 234.64 | 292.46 | 267.78 | 259.54 | 350.00 | 213.63 | 350.00 | 350.00 | 350.00 |
| 22 | 0.00 | 195.52 | 203.62 | 196.73 | 248.55 | 227.40 | 216.18 | 350.00 | 180.00 | 350.00 | 350.00 | 350.00 |
| 23 | 0.00 | 180.00 | 180.00 | 180.00 | 190.18 | 180.00 | 180.00 | 314.16 | 0.00 | 350.00 | 350.00 | 339.00 |
| 24 | 0.00 | 180.00 | 180.00 | 180.00 | 180.00 | 180.00 | 180.00 | 251.05 | 0.00 | 321.34 | 260.03 | 277.58 |

The following results were obtained:

- the minimum, maximum and average costs of the best solutions found by the algorithm in 10 runs were \$644951, \$646229, and \$645264, respectively;
- the standard deviation of the costs of the best solutions found by the algorithm in 10 runs was \$381;
- the frequency of the best solution $P_{best}$ found by GA: 0.2;
- the average number of evaluations necessary to find the best solution $P_{best}$: 70650;
- the average computational time necessary to find the best solution $P_{best}$: 2.25 h.

The best solution found by the algorithm is shown in Table 5.

For comparison, calculations were done using the simulated annealing algorithm, Monte Carlo method and the heuristic method of limit time characteristics [19], which was used for many years in the Polish Electrical Power System. In simulated annealing, the features representation is the same as for GA, but the solutions are generated by a change of one bit in the base solution (standard binary mutation). In the Monte Carlo method, points in the solution space are randomly chosen from the uniform distribution, remembering the best solution. The number of evaluations of the cost function in these algorithms has been set at 100 000, similar to the proposed GA algorithm, and the calculations for every algorithm are done 10 times. The costs of the best solutions found by these algorithms are: simulated annealing = \$702379, the Monte Carlo method—no acceptable solution, the heuristic method = \$665634.

## 5. Conclusions

The proposed genetic algorithm for the Unit Commitment problem gives a stable and acceptable solution that is near optimal. The difference between the cost of the best and the worst solutions found in 10 runs of the algorithm in the second example was 0.20% (\$1278). The advantages the algorithm achieved by the introduction of genetic operators specific to the problem were: mutation that makes the probability of bit change dependent on load demand, production and start-up costs of units, as well as the transposition searching through local minimums. The solutions obtained for the second example are better by 3–3.2% from those obtained with the help of the heuristic method used by the Polish Electrical Power System.

The calculation time, which is rather long here, can be limited by implementing the algorithm in a programming environment that is faster than Matlab, and doing the calculations in a parallel machine environment.

The author's present work on this problem is concentrated on the construction of genetic algorithms with alternative methods of variable representation and genetic operators.

## Acknowledgements

This work was supported by the Polish State Committee for Scientific Research under Grant 8T10B03921.

## Appendix A

$\forall t$: $R(t) = 0{,}05 D_{max}$;
$\forall i$: $P_{min i} = 180$ MW, $P_{max i} = 350$ MW;
$\forall i$: $t_{up i} = 5$ h, $t_{down i} = 5$ h.

## Appendix B. List of symbols

| | |
|---|---|
| $a_i$, $b_i$, $c_i$ | production cost function parameters of unit $i$ |
| $C_i[P_i(t)]$ | variable production cost of unit $i$ at time $t$ (\$/h) |
| $D(t)$ | load demand at the $t$th hour (MW) |
| $e_i$, $f_i$, $g_i$, $h_i$ | start-up cost function parameters of unit $i$ |
| $f_{opt}$ | the frequency of optimum solution found by GA |
| $g(i)$ | the discreet function defining the level of constraint (6) violation |
| $h(i)$ | the discreet function defining the level of constraint (7) violation, $i = 1, 2, \ldots, N$ (unit index) |
| $M$ | the constant greater than the maximum cost of feasible individuals calculated in formula (1) |
| $m$ | the parameter controlling the influence of the constraint violation level on the cost function value |
| $N$ | number of units |
| $n_{down i}$ | the number of periods in which unit $i$ is in continuous off state during the optimization period $T$ |
| $n_{min}(t)$, $n_{max}(t)$ | the minimum and maximum number of units necessary to meet load demand at moment $t$ |
| $n_{opt}$ | the average number of evaluations necessary to find the optimum solution by GA |
| $n_{up i}$ | the number of periods in which unit $i$ is continuously in on state in the optimization period $T$ |
| $P_i(t)$ | power generation of unit $i$ at time $t$ (MW) |
| $P_{min i}$, $P_{max i}$ | lower/upper generation limit of unit $i$ |
| $p_{up}$, $p_{down}$ | the probability of a bit change from 0 to 1 or 1 to 0, respectively |

| | |
|---|---|
| $p_{up1}$, $p_{down1}$ | the probability of a bit change from 0 to 1 or 1 to 0, respectively, dependent on the number of units necessary to meet load demand |
| $p_{up2}$, $p_{down2}$ | the probability of a bit change from 0 to 1 or 1 to 0, respectively, dependent on unit production costs |
| $q_1$, $q_2$, $r_1$, $r_2$ | the lowest borderline value of probabilities $p_{up1}$, $p_{up2}$, $p_{down1}$, $p_{down2}$, respectively |
| $R(t)$ | spinning reserve requirement at the $t$th hour (MW) |
| $SC_i(t_{off\,i})$ | start-up cost of unit $i$ ($) |
| $T$ | number of hours in the study period, $t = 1, 2, \ldots, T$, time index (h) |
| $t_{oni}$, $t_{off\,i}$ | time periods during which unit $i$ is continuously on/off (h) |
| $t_{off\,i}(k)$ | the down time of unit $i$ during $k$th period of off state |
| $t_{oni}(k)$ | the up time of unit $i$ during $k$th period of on state |
| $t_{opt}$ | the average computational time necessary to find the optimum solution by GA |
| $t_{up\,i}$, $t_{down\,i}$ | minimum up/down time of unit $i$ |
| $v_i$ | the relation between the start-up cost after down time $t_{off\,x}$ of the unit with the lowest start-up cost to the start-up cost of unit $i$ |
| $v_{min}$, $v_{max}$ | the minimum and maximum variable $v_i$ values for $i = 1,2, \ldots, N$ |

*Greek letters*

| | |
|---|---|
| $\alpha_i(t)$ | on/off status of the $i$th unit at the $t$th hour, $\alpha_i\ (t) \in \{0, 1\}$ |
| $\beta_i\ (k)$ | the binary variable, which is 1 when unit $i$ violates constraint (6) during $k$th period of off state, or 0 when the constraint is not violated or it is not possible to state the down time (e.g. when the unit stays down until the end of optimization period $T$) |
| $\gamma_i\ (k)$ | the binary variable, which is 1 when unit $i$ violates constraint (7) during the on period $k$, or 0 when the constraint is not violated or it is not possible to state the on period (e.g. when the unit stays on until the end of optimization period $T$) |
| $\sigma_{opt}$ | the standard deviation of the number of evaluations necessary to find the optimum solution by GA |
| $\tau_{off\,i}(k)$ | the outage time of unit $i$ after on period $k$ |
| $\tau_{oni}(k)$ | the start-up hour of unit $i$ after the $k$th period of down time |

## References


[1] G.B. Sheblé, G.N. Fahd, Unit commitment literature synopsis, IEEE Trans. Power Syst. 9 (1) (1994) 128–136.

[2] S. Sen, D. Kothari, Optimal thermal generating unit commitment: a review, Electrical Power Energy Syst. 20 (7) (1998) 443–451.

[3] C.K. Pang, H.C. Chen, Optimal short-term thermal unit commitment, IEEE Trans. Power Apparatus Syst. PAS 95 (4) (1976) 1336–1342.

[4] C.K. Pang, G.B. Sheblé, F. Albuyeh, Evaluation of dynamic programming based methods and multiple area representation for thermal unit commitments, IEEE Trans. Power Apparatus Syst. PAS 100 (3) (1981) 1212–1218.

[5] R. Nayak, J. Sharma, A hybrid neural network and simulated annealing approach to the unit commitment problem, Comput. Ind. Eng. 30 (4) (2000) 851–870.

[6] M. Wong, T. Chung, Y. Wong, An evolving neural network approach in unit commitment solution, Microprocess. Microsyst. 24 (2000) 251–262.

[7] N. Padhy, S. Paranjothi, V. Ramachandran, A hybrid fuzzy neural network—expert system for a short term unit commitment problem, Microelectron. Reliab. 37 (5) (1997) 733–737.

[8] N. Padhy, Unit commitment using hybrid models: a comparative study for dynamic programming, expert system, fuzzy system and genetic algorithms, Electrical Power Energy Syst. 23 (2000) 827–836.

[9] A. Mantawy, Y. Abdel-Magid, S. Selim, A simulated annealing algorithm for unit commitment, IEEE Trans. Power Syst. 13 (1) (1998) 197–204.

[10] S. Wong, An enhanced simulated annealing approach to unit commitment, Electrical Power Energy Syst. 20 (5) (1998) 359–368.

[11] D. Dasgupta, D.R. McGregor, Thermal unit commitment using genetic algorithms, IEE Proc. Gener. Transm. Distrib. 141 (5) (1994) 459–465.

[12] S.A. Kazarlis, A.G. Bakirtzis, V. Petridis, A genetic algorithm solution to the unit commitment problem, IEEE Trans. Power Syst. 11 (1) (1996) 83–92.

[13] A. Mantawy, Y. Abdel-Magid, S. Selim, Integrating genetic algorithms, tabu search, and simulated annealing for the unit commitment problem, IEEE Trans. Power Syst. 14 (3) (1999) 829–836.

[14] A. Wood, B. Wollenberg, Power Generation, Operation, and Control, Wiley, New York, 1996.

[15] Z. Michalewicz, et al., Evolutionary algorithms for constrained engineering problems, Comput. Ind. Eng. 30 (4) (1996) 851–870.

[16] D. Orvosh, L. Davis, Using a genetic algorithm to optimize problems with feasibility constraints, in: Fogel, D. (Ed.), Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE Press, Orlando, FL, 1994.

[17] G. Dudek, Economic dispatch and unit commitment using evolutionary algorithms, Ph.D. Dissertation (in Polish), Dept. Elect. Eng., Technical University of Czestochowa, Poland, 2003.

[18] T. Bäck, D.B. Fogel, Z. Michalewicz (Eds.), Handbook of Evolutionary Computation, IOP Publishing Ltd and Oxford University Press, 1997.

[19] M. Toroñ, A method of selecting the optimal set of generating units in the power systems, Ph.D. Dissertation (in Polish), Dept. Elect. Eng., Technical University of Wroclaw, Poland, 1962.