*Int. J. Appl. Math. Comput. Sci., 2004, Vol. 14, No. 3, 411–421*

# SCHEDULING ELECTRIC POWER GENERATORS USING PARTICLE SWARM OPTIMIZATION COMBINED WITH THE LAGRANGIAN RELAXATION METHOD

HUSEYIN HAKAN BALCI*, JORGE F. VALENZUELA*

* Department of Industrial and Systems Engineering
Auburn University, Auburn, AL, 36849, USA
e-mail: balcihh@auburn.edu, jvalenz@eng.auburn.edu

This paper describes a procedure that uses particle swarm optimization (PSO) combined with the Lagrangian Relaxation (LR) framework to solve a power-generator scheduling problem known as the unit commitment problem (UCP). The UCP consists of determining the schedule and production amount of generating units within a power system subject to operating constraints. The LR framework is applied to relax coupling constraints of the optimization problem. Thus, the UCP is separated into independent optimization functions for each generating unit. Each of these sub-problems is solved using Dynamic Programming (DP). PSO is used to evolve the Lagrangian multipliers. PSO is a population based search technique, which belongs to the swarm intelligence paradigm that is motivated by the simulation of social behavior to manipulate individuals towards better solution areas. The performance of the PSO-LR procedure is compared with results of other algorithms in the literature used to solve the UCP. The comparison shows that the PSO-LR approach is efficient in terms of computational time while providing good solutions.

**Keywords:** particle swarm optimization, unit commitment, Lagrange relaxation

## 1. Introduction

Electric power industry is changing rapidly and the traditional monopolistic environment is moving to a competitive power supply market. Determining the operating strategies to meet the demand for electricity for a specific planning horizon is one of the most important concerns under the current commercial pressure. The Unit Commitment Problem (UCP) is to determine the schedule and production amount of generating units within a power system subject to machine and operating constraints (Wood and Wollenberg, 1996). This problem is a mixed combinatorial and continuous optimization problem in which the states of the units, on or off, and generation amounts are determined under constraints (Sheble and Fahd, 1994). The generic UCP is to minimize the total operational cost and is subject to minimum up- and down-time constraints, crew constraints, unit capability limits, generation constraints and reserve constraints. The most popular techniques to solve the UCP have been the priority list (Lee, 1988), dynamic programming (DP) (Su and Hsu, 1991), integer programming (Garver, 1963), Langrangian relaxation (LR) (Virmani *et al.*, 1989), and stochastic programming (Takriti *et al.*, 1996). Metaheuristic approaches such as genetic algorithms (GA), simulated annealing (SA) (Suzannah, 1998), and tabu search (Xiaomin and Shahidehpour, 1997) have also been used since the beginning of the last decade. Recently, Particle Swarm Optimization (PSO) has been used to solve the UCP (Ting *et al.*, 2003). Hybrid methods such as memetic algorithms (Valenzuela and Smith, 2002) and decommitment with priority list and LR (Tseng *et al.*, 2000) have been shown to be effective in solving large UCPs.

In this paper we propose a new approach to solve the UCP, a hybrid method that uses a combination of PSO (Kennedy and Eberhart, 1995) and LR (Fischer, 1973). We use the abbreviation "PSO-LR" to refer to our approach. The LR framework is based on dual optimization and it decomposes a problem into one master and more manageable sub-problems. In our approach, the sub-problems are solved using DP. The connection between sub-problems is maintained by the Lagrange multipliers, which are updated in our method by the PSO. The PSO approach is motivated from the social behavior of bird flocking and fish schooling. Kennedy and Eberhart introduced PSO in 1995 in terms of social and cognitive behavior. The PSO has been widely used as a problem-solving method in engineering and computer science. Some examples of PSO application are: the training of a neural network to predict the type of human tremor (Eberhart and Hu, 1999), optimizing a computer controlled milling machine (Tandon, 2000), generating interactive and improvised music (Blackwell and Bentley, 2002), and assigning tasks in distributed computing systems (Salman *et al.*, 2002). Recently, in the

area of electric power systems, PSO seems to be gaining popularity. The PSO has been used to solve the optimal power flow problem (Abido, 2002), the reactive power and voltage control problem (Yoshida *et al.*, 2001), and the distribution state estimation problem (Naka *et al.*, 2003). Other applications of PSO are listed on the website `www.swarmintelligence.org`.

The remaining sections of this paper are organized as follows: Section 2 gives a description of the unit commitment problem. Section 3 includes a brief review of the PSO algorithm. Section 4 defines our methodology and tuning efforts. Section 5 shows the implementation of the proposed algorithm on test problems. Section 6 summarizes the results.

## 2. Unit Commitment Problem

The UCP involves the determination of the state (ON/OFF) of power generating units for each time period, as well as the power output levels subject to the system and generating unit's operating constraints. The standard UCP for $N$ generating units and $T$ time periods is formulated as follows (Wood and Wollenberg, 1996):

$$\min \sum_{t=1}^{T} \sum_{i=1}^{N} \left[ C_i(Q_{i,t}) U_{i,t} + S_{i,t}(1 - U_{i,t-1}) U_{i,t-1} \right] \quad (1)$$

subject to

1. Demand constraint

$$\sum_{i=1}^{N} U_{i,t} Q_{i,t} = d_t, \quad t = 1, 2, \ldots, T.$$

2. Power reserve constraint

$$\sum_{i=1}^{N} U_{i,t} Q_i^{\max} \geq d_t + r_t, \quad t = 1, 2, \ldots, T.$$

The power reserve is used in the case of a unit failure or an unexpected increase in the demand.

3. Capacity constraints

$$U_{i,t} Q_i^{\min} \leq Q_{i,t} \leq U_{i,t} Q_i^{\max}, \quad t = 1, 2, \ldots, T, \quad i = 1, 2, \ldots, N.$$

4. Minimum up/down time

$$\left.\begin{aligned} (x_{t-1,i}^{\mathrm{up}} - T_i^{\mathrm{up}})(U_{t-1,i} - U_{t,i}) \geq 0 \\ (x_{t-1,i}^{\mathrm{down}} - T_i^{\mathrm{down}})(U_{t,i} - U_{t-1,i}) \geq 0 \end{aligned}\right\}$$
$$t = 1, 2, \ldots, T, \quad i = 1, 2, \ldots, N,$$

where

$C_i(Q_{i,t})$: Cost of producing $Q_{i,t}$ units of power by unit $i$ at time period $t$, measured in \$/h,

$S_{i,t}$: Start up cost of unit $i$ at time period $t$, measured in \$,

$d_t$: Power demand at time period $t$, measured in MW,

$r_t$: Power reserve at time period $t$, measured in MW,

$T_i^{\mathrm{up}}$: Minimum up time for unit $i$, measured in hours,

$T_i^{\mathrm{down}}$: Minimum down time for unit $i$, measured in hours.

Decision Variables:

$Q_{i,t}$: Power produced by unit $i$ at time period $t$, measured in MW,

$U_{i,t}$: Up or down status for unit $i$ at time period $t$,

$$U_{i,t} = \begin{cases} 1 & \text{means that unit } i \text{ is on for time period } t, \\ 0 & \text{means that unit } i \text{ is off for time period } t. \end{cases}$$

State Variables:

$x_{t,i}^{\mathrm{up}}$: Number of consecutive uptime periods until time period $t$, measured in hours,

$x_{t,i}^{\mathrm{down}}$: Number of consecutive downtime periods until time period $t$, measured in hours.

The start up cost occurs when a unit is turned on, but it depends on how long the unit has been off. If the unit has been off for a long time, a cold start up cost is applied. If the unit has been off for a short time, a hot start up cost is applied. The start up cost is represented by a step function:

$$S_{i,t} = \begin{cases} S_{h,i} & \text{if } x_{t,i}^{\mathrm{down}} \leq t_i^{\mathrm{cold}}, \\ S_{c,i} & \text{otherwise}, \end{cases}$$

where $t_i^{\mathrm{cold}}$ is the maximum time for the cold start-up.

### 2.1. Solving the UCP by the Lagrangian Relaxation Method

The UCP has commonly been formulated as a large scale, nonlinear and mixed-integer combinatorial optimization problem with many constraints (Orero and Irving, 1997; Sheble and Fahd, 1994). The LR method has been applied successfully to the UCP for years and has been demonstrating its good performance on handling the UCP (Bertsekas *et al.*, 1983; Muckstadt and Koenig, 1977).

The UCP has two kinds of constraints: separable and coupling constraints. Separable constraints such as capacity and minimum up and down time constraints are related with one single unit. On the other hand, coupling constraints involve all units. A change in one unit affects the other units. The demand and power reserve constraints are examples of coupling constraints. The LR framework relaxes the coupling constraints and incorporates them into the objective function by a dual optimization procedure. Thus, the objective function can be separated into independent functions for each unit, subject to unit capacity and minimum up and down time constraints. The resulting Lagrange function of the UCP is as follows:

$$\mathcal{L}(\boldsymbol{Q},\boldsymbol{U},\boldsymbol{\lambda},\boldsymbol{\beta}) = F(\boldsymbol{Q},\boldsymbol{U}) + \sum_{t=1}^{T}\lambda_t\Big(d_t - \sum_{i=1}^{N}U_{i,t}Q_{i,t}\Big) + \sum_{t=1}^{T}\beta_t\Big(d_t + r_t - \sum_{i=1}^{N}U_{i,t}Q_i^{\max}\Big), \quad (2)$$

where

$$F(\boldsymbol{Q},\boldsymbol{U}) = \sum_{t=1}^{T}\sum_{i=1}^{N}\big[C_i(Q_{i,t}) + S_{i,t}(1-U_{i,t-1})\big]U_{i,t}$$

and the minimization of the $\mathcal{L}$ function is subject to Constraints 3 and 4 of Eqn. (1). For the sake of simplicity, we have used the symbol $\boldsymbol{Q}$, without the subscripts $i$ and $t$, to denote any appropriate vector of elements $Q_{i,t}$. The same is also valid for the symbols $\boldsymbol{U}$, $\boldsymbol{\lambda}$, and $\boldsymbol{\beta}$. The LR approach requires minimizing the Lagrange function given in (2):

$$q(\boldsymbol{\lambda},\boldsymbol{\beta}) = \min_{Q,U}\mathcal{L}(\boldsymbol{Q},\boldsymbol{U},\boldsymbol{\lambda},\boldsymbol{\beta}). \quad (3)$$

Since $q(\boldsymbol{\lambda},\boldsymbol{\beta})$ provides a lower bound for the objective function of the original problem, the LR method requires to maximize the objective function over the Lagrangian multipliers:

$$q^*(\boldsymbol{\lambda},\boldsymbol{\beta}) = \max_{\lambda,\beta}q(\boldsymbol{\lambda},\boldsymbol{\beta}).$$

After eliminating constant terms such as $\lambda_t d_t$ and $\beta_t(d_t + r_t)$ in Eqn. (2), Eqn. (3) can be written as

$$q(\boldsymbol{\lambda},\boldsymbol{\beta}) = \min_{\boldsymbol{Q},\boldsymbol{U}}\sum_{i=1}^{N}\Big(\sum_{t=1}^{T}\big[C_i(Q_{i,t})U_{i,t} + S_{i,t}(1-U_{i,t})U_{i,t} - \lambda_t Q_{i,t}U_{i,t} - \beta_t Q_i^{\max}U_{i,t}\big]\Big),$$

which is subject to unit capacity and minimum up and down time constraints. Notice that these constraints pertain to individual units only. Thus, the master problem is separable by individual units, i.e., solving a sub-problem for each of the units can solve the master problem. If we choose the values for the Lagrange multipliers, they could be treated as fixed numbers. Therefore, for given values of $\boldsymbol{\lambda}$ and $\boldsymbol{\beta}$, the optimization problem for unit $i$ is

$$q_i(\boldsymbol{\lambda},\boldsymbol{\beta}) = \min_{Q,U}\sum_{t=1}^{T}\big[C_i(Q_{i,t}) + S_{i,t}(1-U_{i,t}) - \lambda_t Q_{i,t} - \beta_t Q_i^{\max}\big]U_{i,t}. \quad (4)$$

The complete Lagrangian relaxation procedure is described next.

### 2.2. Lagrangian Relaxation Procedure

The Lagrangian procedure to solve the UCP starts with initializing the Lagrange multipliers with values that try to make $q(\boldsymbol{\lambda},\boldsymbol{\beta})$ larger (Carter and Price, 2001). Next, they are considered fixed and the Lagrange function $(\mathcal{L})$ is minimized by adjusting $Q_{i,t}$ and $U_{i,t}$. This minimization is done separately for each unit, and different techniques such as LP and dynamic programming can be used. The solutions for the $N$ independent sub-problems are used in the master problem to find a new set of Lagrange multipliers. This iterative procedure continues until a duality gap criterion is met. The duality gap is used as a measure of convergence. If the relative duality gap between the primal and the dual solutions is less than a specific tolerance, it is considered that the optimum has been reached. The procedure then ends with finding a feasible UC schedule.

*Updating the Lagrangian multipliers using the sub-gradient method*

The multipliers can be updated by using a sub-gradient method with a scaling factor and tuning constants, which are determined heuristically (Bertsekas, 1982; Sheble and Fahd, 1994). This method is explained as follows: A vector $\psi$ is called a sub-gradient of $\mathcal{L}(\cdot)$ at $\overline{\lambda}$ if

$$\mathcal{L}(\lambda) \le \mathcal{L}(\overline{\lambda}) + (\lambda - \overline{\lambda})^T\psi.$$

If the sub-gradient is unique at a point $\lambda$, then it is the gradient at that point. The set of all sub-gradients at $\lambda$ is called the sub-differential, $\partial\mathcal{L}(\lambda)$, and is a closed convex set. A necessary and sufficient condition for optimality in sub-gradient optimization is $0 \in \partial\mathcal{L}(\lambda)$. In this paper, we used the sub-gradient optimization algorithm to generate a sequence of points using the rule

$$\lambda_t^{k+1} = \lambda_t^k + \alpha\psi^k,$$

where $\psi^k$ is any sub-gradient of $\mathcal{L}(\cdot)$ at $\lambda_t^k$. The step size, $\alpha$, has to be chosen carefully to achieve good performance by the algorithm. Here $\psi^k$ is calculated as follows:

$$\partial\mathcal{L}(\lambda_t^k) = d_t - \sum_{i=1}^{N}U_{i,t}Q_{i,t}.$$

Then

$$\lambda_t^{k+1} = \lambda_t^k + \alpha \left( d_t - \sum_{i=1}^{N} U_{i,t} Q_{i,t} \right).$$

*Finding a Feasible Solution to the Primal Problem*

Since a solution to the dual problem may not be feasible to the primal problem, a feasible solution is constructed by economic dispatching of units that have the variable $U_{i,t}$ equal to 1 (a state equal to "ON") in the dual problem solution. Therefore, at every time period, say time $t$, the following optimization problem (known as the Economic Dispatch) is solved:

$$\begin{aligned} &\min \ \ \mathrm{TC} = \sum_{i=\Omega_t} C_i\left(Q_{i,t}\right) \\ &\text{s.t.} \ \ \sum_{i=\Omega_t} Q_{i,t} = d_t \ \text{ and } \ Q_{i,t}^{\min} \le Q_{i,t} \le Q_{i,t}^{\max}, \end{aligned}$$

where $\Omega_t$ is the subscript set of all units committed at time $t$ in the dual solution. Figure 1 shows a flow diagram of the complete LR procedure.

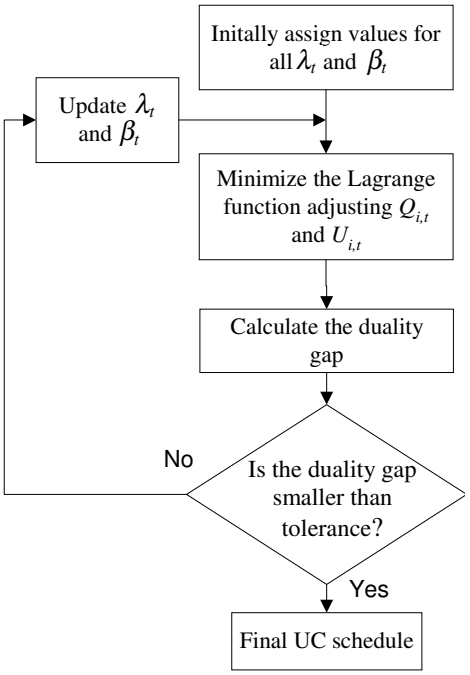


Fig. 1. Lagrange relaxation procedure for the UCP.

### 2.3. Dynamic Programming Method for Solving the Sub-Problem

The Dynamic Programming (DP) method consists in implicitly enumerating feasible schedule alternatives and comparing them in terms of operating costs. To solve the sub-problem for unit $i$ using dynamic programming, we define the function $R_t(U_t)$ by the following equation:

$$R_t(U_t) = \min_{Q_t} \left\{ \left( C(Q_t) - \lambda_t Q_t - \beta_t Q^{\max} \right) U_t \right\}, \quad (5)$$

which is subject to the capacity constraint $U_t Q^{\min} \le Q_t \le U_t Q^{\max}$. Equation (5) denotes the minimum cost at hour $t$ excluding start-up costs. Here, we have omitted the subscript $i$ to simplify the notation. If the unit at time $t$ is off $(U_t = 0)$, the solution of (5) is $Q_t = 0$. On the other hand, if the unit at time $t$ is on $(U_t = 1)$, the solution of (5) is calculated by taking the first derivative with respect to $Q_t$ within the range $[Q_t^{\min},\ Q_t^{\max}]$. If the unit has a quadratic cost function as shown below:

$$C(q) = a + bq + cq^2,$$

the optimal value of $Q_t$ in (5) is given by the following expression:

$$Q_t = \max\left\{ Q^{\min}, \min\left\{ Q^{\max}, \frac{\lambda_t - b}{2c} \right\} \right\} U_t.$$

We also define the recursive function $\mathrm{F}_t(x_t)$ to be the minimum accumulated cost until hour $t$ of operating the generator in state $x_t$ during hour $t$. Thus, the expression for hour $t$ is

$$\mathrm{F}_t(x_t) = \min_{U_t} \left\{ \mathrm{R}_t(U_t) + U_t(1 - U_{t-1})\mathrm{S}_t(x_t) + \mathrm{F}_{t-1}(x_{t-1}) \right\},$$

which is subject to the operating constraints. Since the costs of decisions taken before and during time 0 are irrelevant, the accumulated cost at time 0 is set to zero $(F_0(x_0) = 0)$.

### 2.4. 2-Unit Example

A sample problem is created with two units for three hours of scheduling (Table 1). We assume that there are no reserve constraints $(\beta = 0)$. Lagrange multipliers for each hour are assumed to be 11, 6 and 13 \$/MWh respectively. The dynamic programming procedure is used to find the optimum schedule.

Table 2 shows alternative schedules for Unit 1 and their Lagrangian function values. Due to minimum up/down time constraints, Schedules 2, 4 and 6 are infeasible. The other schedules do not violate the minimum up and down time constraints, and therefore they are all feasible. The lowest cost schedule is number 7, which results in the minimum cost, $-698.68$. Thus, with the given Lagrange multipliers, the best schedule is to turn Unit 1 off at time 1 and turn it back on at time 3 and to produce 220 MW at that hour. Figure 2 shows a graphic representation of Table 2. The amount of power produced and cost are shown in the figure for each period.

Table 1. Unit characteristics (Bard, 1988).

| Characteristic | Unit 1 | Unit 2 |
|---|---|---|
| Maximum capacity (MW) | 220 | 200 |
| Minimum capacity (MW) | 80 | 50 |
| Cost function ($/h) | $400 + 7.654Q + 0.0016Q^2$ | $175 + 7.054Q + 0.00515Q^2$ |
| Minimum up time (hour) | 3 | 2 |
| Minimum down time (hour) | 2 | 2 |
| Initial state (off-line($-$) or on-line($+$)) | 5 | $-1$ |

Table 2. Lagrangian function for Unit 1.

| No. | $U_{1,1}$ | $U_{1,2}$ | $U_{1,3}$ | $Q_{1,1}$ | $Q_{1,2}$ | $Q_{1,3}$ | $q_1(\lambda,\beta)$ |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 220 | 108.125 | 220 | $-576.06$ |
| 2 | 0 | 1 | 1 | 0 | infeasible | infeasible | — |
| 3 | 1 | 1 | 0 | 220 | 108.125 | 0 | 122.61 |
| 4 | 1 | 0 | 1 | 220 | infeasible | infeasible | — |
| 5 | 1 | 0 | 0 | 220 | 0 | 0 | $-258.68$ |
| 6 | 0 | 1 | 0 | 0 | infeasible | infeasible | — |
| 7 | 0 | 0 | 1 | 0 | 0 | 220 | **$-698.68$** |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

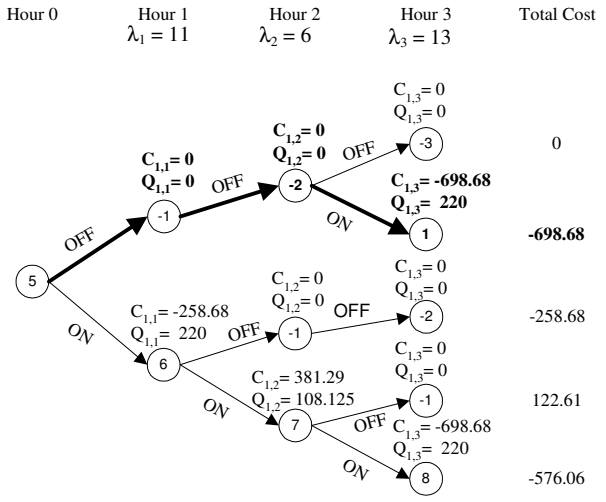


Fig. 2. State transition diagram for Unit 1.

Similarly, for Unit 2, the state transition diagram emphasizing the dynamic programming solution is shown in Fig. 3. The optimum schedule is OFF, OFF and ON, respectively.

## 3. Particle Swarm Optimization for the UCP

Particle Swarm Optimization is a computation technique introduced by Kennedy and Eberhart in 1995, which was inspired by the social behavior of bird flocking or fish

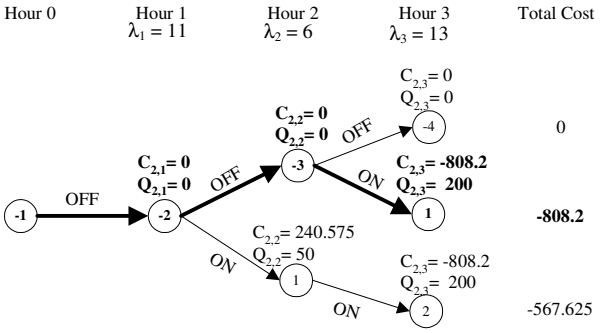


Fig. 3. State diagram for Unit 2.

schooling (Reynolds, 1987). They theorize that the process of cultural adaptation can be summarized in terms of three principles: evaluate, compare and imitate. An organism, a bird in PSO, evaluates its neighbors, compares itself to others in the population and then imitates only those neighbors who are superior. So they behave with two kinds of information: their own experience and the knowledge of how the other individuals around them have performed (Kennedy and Eberhart, 2001).

The PSO approach has some similarities to GA and evolutionary algorithms. PSO has a population of individuals that move through the $D$-dimensional search space and each individual has a velocity that acts as an operator to obtain a new set of individuals. Individuals, called *particles*, adjust their movements depending on both their own experience and the population's experience. At each iteration, a particle moves towards a direction computed from the best visited position and the best visited position of all particles in its neighborhood. Among several variants of PSO, the global variant considers the neighborhood as the whole population, called the *swarm*, which permits the global sharing of information.

In PSO, the $p$-th particle is represented as $\boldsymbol{\lambda}_p = \{\lambda_{p1}, \lambda_{p2}, \ldots, \lambda_{pD}\}$, where $\lambda_{pj}$ is the value of the $j$-th coordinate in the $D$-dimensional space. The best particle of the swarm is represented by the symbol $\boldsymbol{G} = \{g_1, g_2, g_3, \ldots, g_D\}$ and the best visited position of the $p$-th particle is represented as $\boldsymbol{P}_p = \{p_{p1}, p_{p2}, \ldots, p_{pD}\}$. The rate of the position change, which is the velocity for particle $p$, is represented as $\boldsymbol{V}_p = \{v_{p1}, v_{p2}, \ldots, v_{pD}\}$.

The position of a particle changes according to its velocity, which is adjusted at each iteration. Particle $p$ is repositioned according to the $d$-coordinate of its velocity, which is calculated as follows (Shi and Eberhart, 1999):

$$v_{pd} = wv_{pd} + c_1 \,\mathrm{rand}(0,1)(p_{pd} - \lambda_{pd}) + c_2 \,\mathrm{rand}(0,1)(g_d - \lambda_{pd}).$$

The factor $w$ is the inertia weight and is similar to the effect of temperature in simulated annealing. If the inertia weight is large, the search becomes more global, while for smaller inertia the search becomes more local. The coefficients $c_1$ and $c_2$ are learning factors, which help particles to accelerate towards better areas of the solution space.

The velocity of each dimension has upper and lower limits, $V_{\max}$ and $V_{\min}$, and they are defined by the user. The new position of a particle is updated as follows:

$$\lambda_{pd} = \lambda_{pd} + v_{pd}.$$

The PSO algorithm can start with a population of particles with random positions or with a population of particles created heuristically. In PSO, a single particle is a solution in the search space. All particles have fitness values, which are evaluated by the fitness function to be optimized, and velocities, which direct the flying of the particles. The PSO algorithm that maximizes the fitness is described by the pseudo code below (Kennedy and Eberhart, 2001):

*Pseudo code of PSO*

Initialize population
While (number of generations, or the stopping criterion is not met)

For $p = 1$ to number of particles
If the fitness of $\boldsymbol{\lambda}_p$ is greater than the fitness of $\boldsymbol{P}_p$ then
Update $\boldsymbol{P}_p = \boldsymbol{\lambda}_p$
For $k \in$ Neighborhood of $\boldsymbol{\lambda}_p$
If the fitness of $\boldsymbol{\lambda}_k$ is greater than that of $\boldsymbol{G}$ then
Update $\boldsymbol{G} = \boldsymbol{\lambda}_k$
Next $k$
For each dimension $d$
$v_{pd} = wv_{pd} + c_1 \,\mathrm{rand}(0,1)(p_{pd} - \lambda_{pd}) + c_2 \,\mathrm{rand}(0,1)(g_d - \lambda_{pd})$
if $v_{pd} \notin (V_{\min}, V_{\max})$ then $v_{pd} = \max(\min(V_{\max}, v_{pd}), V_{\min})$
$\lambda_{pd} = \lambda_{pd} + v_{pd}$
Next $d$
Next $p$
Next generation until criterion

## 4. Proposed Methodology

In this section, the implementation of PSO-LR to solve the UCP is explained. The PSO approach is used to search for the Lagrange multipliers $\lambda$ for each hour in a similar fashion as in (Shi and Krohling, 2002). The sub-gradient approach (with step size equal to 0.0001) is used to search for the Lagrange multipliers $\beta$ for each hour. For a given set of hourly Lagrange multipliers of a particle $p$, the schedules $U_{p,i,t}$ and $Q_{p,i,t}$ are computed using forward dynamic programming. The values of $\lambda_{p,t}, v_{p,t}, Q_{p,i,t}, U_{p,i,t}$, and $\beta_{p,t}$ are stored in matrices $\boldsymbol{\lambda}, \boldsymbol{V}, \boldsymbol{Q}, \boldsymbol{U}$, and $\boldsymbol{\beta}$, respectively. Figure 4 shows matrices $\boldsymbol{\lambda}$ and $\boldsymbol{V}$ and Fig. 5 shows sub-matrices of $\boldsymbol{Q}$ and $\boldsymbol{U}$ for a particle $(p = 1)$.

Lagrange multipliers

$$\boldsymbol{\lambda} = \begin{bmatrix} \lambda_{11} & \lambda_{12} & \lambda_{13} & \cdots & \lambda_{1T} \\ \lambda_{21} & \lambda_{22} & \lambda_{23} & \cdots & \lambda_{2T} \\ \lambda_{31} & \lambda_{32} & \lambda_{33} & \cdots & \lambda_{3T} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \lambda_{P1} & \lambda_{P2} & \lambda_{P3} & \cdots & \lambda_{PT} \end{bmatrix}$$

Velocities

$$\mathbf{V} = \begin{bmatrix} v_{11} & v_{12} & v_{13} & \cdots & v_{1T} \\ v_{21} & v_{22} & v_{23} & \cdots & v_{2T} \\ v_{31} & v_{32} & v_{33} & \cdots & v_{3T} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ v_{P1} & v_{P2} & v_{P3} & \cdots & v_{PT} \end{bmatrix}$$

Fig. 4. Encoding of the Lagrange multipliers and velocities of particles.

A particle, one row in matrix $\boldsymbol{\lambda}$, represents a set of the Lagrange multipliers corresponding to each hour. The fitness of a particle $p$ is defined as

$$F_p = \sum_{t=1}^{T}\sum_{i=1}^{N}\left[C_{i,t}\left(Q_{p,i,t}\right) + S_{i,t}\left(1 - U_{p,i,t-1}\right)\right]U_{p,i,t} + \sum_{t=1}^{T}\lambda_{p,t}\left[d_t - \sum_{i=1}^{N}Q_{p,i,t}\right] + \sum_{t=1}^{T}\beta_{p,t}\left[r_t - \sum_{i=1}^{N}Q_i^{\max}U_{p,i,t}\right].$$

In the PSO approach, the fitness of a particle is interpreted as the worth of its location in the search space. Thus,

Power Amounts

$$\mathbf{Q}_1 = \begin{bmatrix} Q_{11}\ Q_{12}\ Q_{13}\ \ldots\ Q_{1T} \\ Q_{21}\ Q_{22}\ Q_{23}\ \ldots\ Q_{2T} \\ Q_{31}\ Q_{32}\ Q_{33}\ \ldots\ Q_{3T} \\ \ldots\ \ldots\ \ldots\ \ldots\ \ldots \\ Q_{N1}\ Q_{N2}\ Q_{N3}\ \ldots\ Q_{NT} \end{bmatrix}$$

Unit states

$$\mathbf{U}_1 = \begin{bmatrix} U_{11}\ U_{12}\ U_{13}\ \ldots\ U_{1T} \\ U_{21}\ U_{22}\ U_{23}\ \ldots\ U_{2T} \\ U_{31}\ U_{32}\ U_{33}\ \ldots\ U_{3T} \\ \ldots\ \ldots\ \ldots\ \ldots\ \ldots \\ U_{N1}\ U_{N2}\ U_{N3}\ \ldots\ U_{NT} \end{bmatrix}$$

Fig. 5. Encoding of power amounts and unit states for particle one $(p = 1)$.

as the matrices $\boldsymbol{\lambda}$ and $\boldsymbol{\beta}$ are updated in each iteration, the swarm moves in the solution space toward the optimal point. The best previous position for each particle and the best position for the swarm are kept as $\boldsymbol{P}_p$ and $\boldsymbol{G}$, respectively. The pseudo code for the proposed method is shown below:

*Main Procedure of PSO-LR*

Initialize population
For each particle $p$
  Compute $U_{p,i,t}$ and $Q_{p,i,t}$ using $\boldsymbol{\lambda}_p$
  Compute fitness function, $F_p$
  If $F_p$ is less than the fitness of $\boldsymbol{P}_p$, then $\boldsymbol{P}_p = \boldsymbol{\lambda}_p$.
  If $F_p$ is less than the fitness of $G$, then $G = \boldsymbol{\lambda}_p$.
  Update $V_p$
  Update $\boldsymbol{\lambda}_p$ and $\boldsymbol{\beta}_p$
Next $p$

*Illustrative Example*

The problem defined in Section 2 is extended with a load of 220, 240 and 180 megawatt (MW) at each time period, and used here to illustrate the PSO-LR. The problem involves three Lagrange multipliers (one for each hour). Here, a particle consists of three Lagrange multipliers. To show how the particles search for the optimal solution, we create two independent swarms. In Fig. 6, we show the movement of the swarms in two dimensions, $\lambda_2$ and $\lambda_3$. The first dimension, which is $\lambda_1$ for the first hour, is not shown. Each swarm is composed of 8 particles. One swarm (shown with lines in Fig. 6) starts at the upper-right corner of the solution space of $\lambda_2$ and $\lambda_3$, and the other swarm (shown with dots in Fig. 6) starts at the bottom-left corner. The figure shows the initial positions and the positions after the 9-th iteration. The location of the optimal solution for the test problem is depicted by a square.

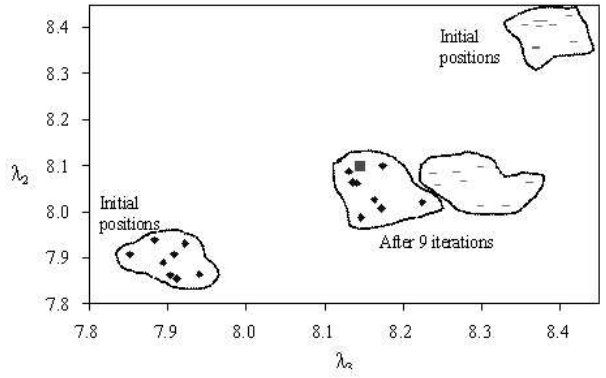


Fig. 6. Movement of two swarms after nine iterations.

### 4.1. Parameter Tuning

The PSO-LR algorithm is tuned using a small test problem taken from (Valenzuela and Smith, 2002). The problem consists of four generating units and a time horizon of 8 hours. The system data are given in Tables 3 and 4. The optimal solution is known to be \$74,675 (computed using enumeration).

Table 3. Test system.

| | Unit 1 | Unit 2 | Unit 3 | Unit 4 |
|---|---|---|---|---|
| $Q^{\max}$ (MW) | 300 | 250 | 80 | 60 |
| $Q^{\min}$ (MW) | 75 | 60 | 25 | 20 |
| $a_0$ | 684.74 | 585.62 | 213.00 | 252.00 |
| $a_1$ | 16.83 | 16.95 | 20.74 | 23.60 |
| $a_2$ | 0.0021 | 0.0042 | 0.0018 | 0.0034 |
| Max marginal cost (\$/MWh) | 18.09 | 19.05 | 21.03 | 24.01 |
| Min marginal cost (\$/MWh) | 17.15 | 17.45 | 20.83 | 23.74 |
| $T_i^{\text{up}}$ (h) | 5 | 5 | 4 | 1 |
| $T_i^{\text{down}}$ (h) | 4 | 3 | 2 | 1 |
| $S_{h,i}$ (\$) (hot start) | 500 | 170 | 150 | 0 |
| $S_{c,i}$ (\$) (cold start) | 1100 | 400 | 350 | 0.02 |
| $t_i^{\text{cold}}$ (h) | 5 | 5 | 4 | 0 |
| Initial state (h) | 8 | 8 | −5 | −6 |

Table 4. Load and reserve.

| Hour | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| Demand (MW) | 450 | 530 | 600 | 540 | 400 | 280 | 290 | 500 |
| Reserve (MW) | 45 | 53 | 60 | 54 | 40 | 28 | 29 | 50 |

In our approach, we set the number of particles equal to 20. Since the Lagrangian multipliers $\lambda$ can also be regarded as the marginal costs of supplying the demand at each hour, the value of $V_{\max}$ is set to the system maximum marginal cost, i.e., $V_{\max}$ is set to 24.01. The value of $V_{\min}$ is set to $-24.01$. The initial values of the matrix $\boldsymbol{\lambda}$ are randomly selected from a uniform distribution between the system minimum and maximum marginal costs. The values of the matrix $\boldsymbol{\beta}$ are initially set to zero. The parameters $w$, $c_1$ and $c_2$ are tuned by running PSO-LR with different combinations of values for these parameters (see Table 5). PSO-LR is run ten times for each combination, and each run is terminated after 50 iterations. The mean and the number of times that PSO-LR found the optimal solution for each combination are shown in Table 5.

Table 5. PSO-LR tuning results.

| $w$ | $c_1$ | $c_2$ | Average over 10 runs | No. of times optimum is found |
|---|---|---|---|---|
| | 1 | 1 | 75,012 | 0 |
| | 1 | 2 | 75,012 | 0 |
| 0.3 | 2 | 1 | 75,012 | 0 |
| | 2 | 2 | 74,709 | 9 |
| | 1 | 1 | 75,012 | 0 |
| | 1 | 2 | 74,945 | 2 |
| 0.6 | 2 | 1 | 74,878 | 4 |
| | 2 | 2 | 74,675 | 10 |
| | 1 | 1 | 74,775 | 7 |
| | 1 | 2 | 74,742 | 8 |
| 0.9 | 2 | 1 | 74,675 | 10 |
| | 2 | 2 | 74,742 | 8 |

The combinations of parameters $(w, c_1, c_2)$ that provided the best results were (0.6, 2, 2) and (0.9, 2, 1). For these two sets of values the optimal solution was found in all ten runs. We choose these two combinations to further test our PSO-LR algorithm. We denote these two versions as PSO-LR$^1$ and PSO-LR$^2$ for the parameter set (0.6, 2, 2) and (0.9, 2, 1), respectively.

## 5. Computation Results

After tuning, we compare the performance of PSO-LR with other approaches. For the comparison, we use a 10-unit problem taken from the literature (Kazarlis *et al.*, 1996). They solved this problem using DP, LR, and GA. The same problem was also solved by Valenzuela and Smith (2002) using DP, LR, and a memetic algorithm (MA), and by Ting *et al.* (2003) using PSO. The optimal solution is known to be \$565,825. The system, load, and power reserve data of this problem are given in Tables 6 and 7.

In Table 8, we show the best and worst solutions obtained using DP, LR, MA, GA, PSO and our approach (PSO-LR). In this table, the DP, LR, and MA results are from (Valenzuela and Smith, 2002), the GA results are from (Kazarlis *et al.*, 1996), and the PSO results are from (Ting *et al.*, 2003). To compare the relative speed of the hardware utilized, we give the SPECfp95 of the computer system utilized in running each algorithm. For example, Sun Ultra 2 is roughly seven times faster than HP Apollo 720 and two times slower than Dell Dimension 4100.

Notice that for the 10-unit problem, PSO-LR$^2$ found a solution that is very close to the optimum, and it is also better than the best solution found by LR and PSO. In order to have a fair comparison regarding the computational effort, we have estimated the total CPU time of the different algorithms as they were run using a Dell Dim 4100 computer system. The total CPU time of an algorithm is computed by multiplying the number of runs by the CPU time of one run and the result multiplied by a speed factor to obtain the total CPU time. The results are shown in Table 9. This table includes also a measure of the quality of the best solution. This quantity measures the closeness of the best solution to the optimal solution and it is calculated by the following equation:

$$\text{Solution quality} = 100\left(1-\frac{\text{Best solution} - \text{Optimal solution}}{\text{Optimal solution}}\right).$$

Notice that both PSO-LR$^1$ and PSO-LR$^2$ provide high quality solutions in one sixth of the computation time of the other algorithms.

To further test our approach, we use expanded versions of the problem of (Kazarlis *et al.*, 1996). These new problems are created by repeating each unit of the original problem 2, 4, 6, 8 and 10 times, respectively. Thus, test problems with 20, 40, 60, 80 and 100 units are obtained. Hourly load and reserve amounts are scaled by the same factors. These problems were also solved by Kazarlis *et al.* (1996) using GA and Valenzuela and Smith (2002) using LR and MA. In Table 10 we show the best results obtained by LR, MA, GA and PSO-LR for these six test problems. In Table 11, we show the total scaled CPU time as the algorithms were run using a Dell Dim 4100 computer system. The results show that both PSO-LR$^1$ and PSO-LR$^2$ are able to find very good solutions in much smaller times than any of the other algorithms. In Fig. 7, we have plotted the results shown in Table 11. To observe the rate of the increase in PSO-LR in greater detail, in Fig. 8 we show the CPU time versus the problem size. Notice that the rate of the increase in the CPU time of PSO-LR is polynomial with respect to the problem size while the GA is exponential.

Table 6. System data (Kazarlis *et al.*, 1996).

| Unit $i$ | $Q^{\max}$ MW | $Q^{\min}$ MW | $a_0$ | $a_1$ | $a_2$ | $T_i^{\text{up}}$ hour | $T_i^{\text{down}}$ hour | $S_{h,i}$ | $S_{c,i}$ | $t_i^{\text{cold}}$ hour | Initial state |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 455 | 150 | 1000 | 16.19 | 0.00048 | 8 | 8 | 4500 | 9000 | 5 | 8 |
| 2 | 455 | 150 | 970 | 17.26 | 0.00031 | 8 | 8 | 5000 | 10000 | 5 | 8 |
| 3 | 130 | 20 | 700 | 16.60 | 0.00200 | 5 | 5 | 550 | 1100 | 4 | −5 |
| 4 | 130 | 20 | 680 | 16.50 | 0.00211 | 5 | 5 | 560 | 1120 | 4 | −5 |
| 5 | 162 | 25 | 450 | 19.70 | 0.00398 | 6 | 6 | 900 | 1800 | 4 | −6 |
| 6 | 80 | 20 | 370 | 22.26 | 0.00712 | 3 | 3 | 170 | 340 | 2 | −3 |
| 7 | 85 | 25 | 480 | 27.74 | 0.00079 | 3 | 3 | 260 | 520 | 2 | −3 |
| 8 | 55 | 10 | 660 | 25.92 | 0.00413 | 1 | 1 | 30 | 60 | 0 | −1 |
| 9 | 55 | 10 | 665 | 27.27 | 0.00222 | 1 | 1 | 30 | 60 | 0 | −1 |
| 10 | 55 | 10 | 670 | 27.79 | 0.00173 | 1 | 1 | 30 | 60 | 0 | −1 |

Table 7. Load for the test problem (Kazarlis *et al.*, 1996).

| Hour | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| Load (MW) | 700 | 750 | 850 | 950 | 1000 | 1100 | 1150 | 1200 |
| Reserve (MW) | 70 | 75 | 85 | 95 | 100 | 110 | 115 | 120 |
| Hour | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| Load (MW) | 1300 | 1400 | 1450 | 1500 | 1400 | 1300 | 1200 | 1050 |
| Reserve (MW) | 130 | 140 | 145 | 150 | 140 | 130 | 120 | 105 |
| Hour | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| Load (MW) | 1000 | 1100 | 1200 | 1400 | 1300 | 1100 | 900 | 800 |
| Reserve (MW) | 100 | 110 | 120 | 140 | 130 | 110 | 90 | 80 |

Table 8. Results obtained by different approaches.

| Approach | Best solution | Worst solution | Number of runs | Population size | CPU time (sec) | Hardware used | SPECfp95 |
|---|---|---|---|---|---|---|---|
| DP | 565,827 | na | 1 | na | 177 | Sun Ultra 2 | 14.70 |
| LR | 566,107 | 566,817 | 10 | na | 54 | Sun Ultra 2 | 14.70 |
| GA | 565,827 | 570,032 | 20 | 50 | 221 | HP Apollo 720 | 2.02 |
| MA | 565,827 | 566,861 | 10 | 50 | 61 | Sun Ultra 2 | 14.70 |
| PSO | 574,153 | 647,305 | nr | 20 | nr | nr | — |
| PSO-LR[1] | 565,869 | 566,793 | 10 | 20 | 4.2 | Dell Dim 4100 | 30.9 |
| PSO-LR[2] | 566,297 | 567,143 | 10 | 20 | 4.5 | Dell Dim 4100 | 30.9 |

*nr: Not reported by the author; **na: Not applicable; ***CPU time is per run.

Table 9. Best solution quality and CPU time.

| Approach | Quality of best solution | Reported total CPU time (sec) | CPU speed factor | Scaled total CPU time (sec) |
|---|---|---|---|---|
| DP | 100.000 | 177 | 0.476 | 84 |
| LR | 99.951 | 540 | 0.476 | 257 |
| GA | 100.000 | 4,420 | 0.065 | 289 |
| MA | 100.000 | 610 | 0.476 | 290 |
| PSO | 98.529 | nr | — | — |
| PSO-LR[1] | 99.992 | 42 | 1.000 | 42 |
| PSO-LR[2] | 99.917 | 45 | 1.000 | 45 |

*nr: Not reported by the author.

Table 10. Best solution for large problems.

| Problem size | LR | GA | MA | PSO-LR[1] | PSO-LR[2] |
|---|---|---|---|---|---|
| 20 | 1,128,362 | 1,126,243 | 1,128,192 | 1,128,072 | 1,128,281 |
| 40 | 2,250,223 | 2,251,911 | 2,249,589 | 2,251,116 | 2,252,330 |
| 60 | 3,374,994 | 3,376,625 | 3,370,820 | 3,376,407 | 3,377,718 |
| 80 | 4,496,729 | 4,504,933 | 4,494,214 | 4,496,717 | 4,499,347 |
| 100 | 5,620,305 | 5,627,437 | 5,616,314 | 5,623,607 | 5,623,607 |

Table 11. Scaled total CPU time (in seconds).

| Problem size | LR | GA | MA | PSO-LR[1] | PSO-LR[2] |
|---|---|---|---|---|---|
| 10 | 257 | 289 | 290 | 42 | 45 |
| 20 | 514 | 958 | 538 | 91 | 96 |
| 40 | 1066 | 3526 | 1032 | 213 | 218 |
| 60 | 1594 | 7635 | 2740 | 360 | 384 |
| 80 | 2122 | 13122 | 3159 | 543 | 595 |
| 100 | 2978 | 20570 | 6365 | 730 | 856 |

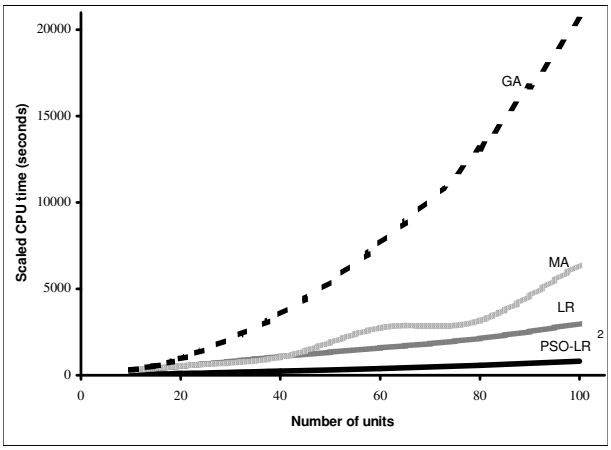


Fig. 7. Scaled total CPU time.

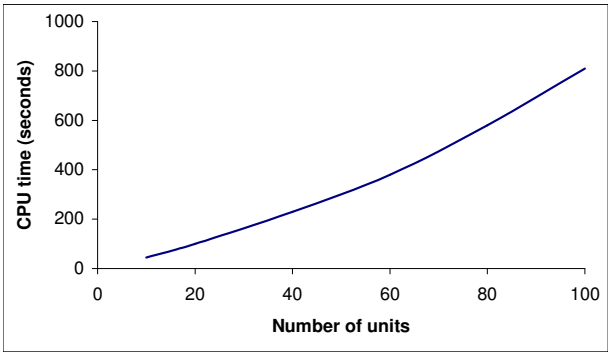


Fig. 8. CPU time versus problem size of PSO-LR.

## 6. Conclusions

In this paper, a procedure that combines PSO and LR to solve the unit commitment problem has been proposed. The results obtained after solving six instances of the UCP showed that PSO-LR is computationally efficient in solving these problems. After scaling down the CPU time of the other algorithms, PSO-LR was on average five times faster than LR, seven times faster than MA, and seventeen times faster than GA, and it provided solutions that are comparable to these approaches. In terms of the solution quality, the PSO-LR provided a "best solution" with a lower cost than GA for problem sizes larger than 20 units, and than LR for problem sizes 20 and 80 units. PSO-LR also provided a "best solution", for the problem size of 10 units, with a much lower cost than using PSO alone. The results show that both PSO-LR[1] and PSO-LR[2] are able to find very good solutions, which indicates that the proposed approach is very robust regarding parameter setting.

## Acknowledgment

This research was partly supported by the National Science Foundation under the grant ECS–0245650.

## References

Abido M.A. (2002): *Optimal power flow using particle swarm optimization*. — Int. J. Electr. Power Energy Syst., Vol. 24, No. 7, pp. 563–571.

Bard J.F. (1988): *Short-term scheduling of thermal-electric generators using lagrangian relaxation*. — Opers. Res., Vol. 36, No. 5, pp. 756–766.

Bertsekas D.P. (1982): *Constrained Optimization and Lagrange Multiplier Methods*. — New York: Academic Press.

Bertsekas D.P., Lauer G.S., Sandell N.R. Jr. and Posbergh T.A. (1983): *Optimal short-term scheduling of large scale power systems*. — IEEE Trans. Automat. Contr., Vol. 28, No. 1, pp. 1–11.

Blackwell T.M. and Bentley P. (2002): *Don't push me! Collision avoiding swarms*. — Proc. Congress *Evolutionary Computation*, Honolulu, USA, pp. 1691–1696.

Carter M.W. and Price C.C. (2001): *Operations Research: A Practical Introduction*. — Boca Raton: CRC Press.

Eberhart R.C. and Hu X. (1999): *Human tremor analysis using particle swarm optimization*. — Proc. Congress *Evolutionary Computation*, Piscataway, NJ, pp. 1927–1930.

Fischer M.L. (1973): *Optimal solution of scheduling problems using lagrange multipliers: Part I*. — Opers. Res., Vol. 21, No. 5, pp. 1114–1127.

Garver L.L. (1963): *Power generation scheduling by integer programming—Development of theory*. — AIEE Trans., No. 2, pp. 730–735.

Kazarlis S.A., Bakirtzis A.G. and Petridis V. (1996): *A genetic algorithm solution to the unit commitment problem*. — IEEE Trans. Power Syst., Vol. 11, No. 1, pp. 83–90.

Kennedy J. and Eberhart R.C. (1995): *Particle swarm optimization*. — Proc. IEEE Int. Conf. *Neural Networks*, Perth, Australia, pp. 1942–1948.

Kennedy J. and Eberhart R.C. (2001): *Swarm Intelligence*. — San Francisco: Morgan Kaufmann.

Lee F.N. (1988): *Short term unit commitment—A new method*. — IEEE Trans. Power Syst., Vol. 3, No. 2, pp. 421–428.

Muckstadt J.A. and Koenig S.A. (1977): *An application of Lagrangian relaxation to scheduling in power generating systems*. — Opers. Res., Vol. 25, No. 3, pp. 387–403.

Naka S., Genji T., Yura T. and Fukuyama Y. (2003): *A hybrid particle swarm optimization for distribution state estimation*. — IEEE Trans. Power Syst., Vol. 18, No. 1, pp. 60–68.

Orero S.O. and Irving M.R. (1997): *A combination of the genetic algorithm and Lagrangian relaxation decomposition techniques for the generation unit commitment problem*. — Electr. Power Syst. Res., Vol. 43, No. 3, pp. 149–156.

Reynolds C.W. (1987): *Flocks, herds and schools: A distributed behavioral model*. — Comput. Graph., Vol. 21, No. 4, pp. 25–34.

Salman A., Ahmad I. and Al-Madani S. (2002): *Particle swarm optimization for task assignment problem*. — Microprocess. Microsyst., Vol. 26, No. 8, pp. 363–371.

Sheble G.B. and Fahd G.N. (1994): *Unit commitment literature synopsis*. — IEEE Trans. Power Syst., Vol. 9, No. 1, pp. 128–135.

Shi Y. and Eberhart R.C. (1999): *Empirical study of particle swarm optimization*. — Proc. Congress *Evolutionary Computation*, Piscataway, NJ, pp. 1945–1950.

Shi Y. and Krohling R.A. (2002): *Co-evolutionary particle swarm optimization to solve min-max problems*. — Proc. IEEE Congress *Evolutionary Computation*, Honolulu, Hawaii, USA, pp. 1682–1687.

Su C.C. and Hsu Y.Y. (1991): *Fuzzy dynamic programming: An application to unit commitment*. — IEEE Trans. Power Syst., Vol. 6, No. 3, pp. 1231–1237.

Suzannah Y.W.W. (1998): *An enhanced simulated annealing approach to unit commitment*. — Int. J. Electr. Power Energy Syst., Vol. 20, No. 5, pp. 359–368.

Takriti S., Birge J.R. and Long E. (1996): *A stochastic model for the unit commitment problem*. — IEEE Trans. Power Syst., Vol. 11, No. 3, pp. 1497–1508.

Tandon V. (2000): *Closing gap between CAD/CAM and optimized CNC and milling*. — M.Sc. Thesis, Purdue School of Engineering and Technology, Purdue University, Indiana, USA.

Ting T.-O., Rao M.V.C., Loo C.K. and Ngu S.S. (2003): *Solving unit commitment problem using hybrid particle swarm optimization*. — J. Heuristics, Vol. 9, No. 6, pp. 507–520.

Tseng C.L., Li C.A. and Oren S.S. (2000): *solving the unit commitment problem by a unit decommitment method*. — J. Optim. Theory Applics., Vol. 105, No. 3, pp. 707–730.

Valenzuela J. and Smith A. (2002): *A seeded memetic algorithm for large unit commitment problems*. — J. Heuristics, Vol. 8, No. 2, pp. 173–195.

Virmani S., Adrian E.C., Imhof K. and Mukherjee S. (1989): *Implementation of a Lagrangian relaxation based unit commitment problem*. — IEEE Trans. Power Syst. Vol. 4, No. 4, pp. 373–1380.

Wood A.J. and Wollenberg B.F. (1996): *Power Generation, Operation and Control, 2-nd Ed.*. — New York: Wiley.

Xiaomin B. and Shahidehpour S.M. (1997): *Extended neighborhood search algorithm for constrained unit commitment*. — Int. J. Electr. Power Energy Syst., Vol. 19, No. 5, pp. 349–356.

Yoshida H., Kawata K., Fukuyama Y. and Nakanishi Y. (2001): *A particle swarm optimization for reactive power and voltage control considering voltage security assessment*. — IEEE Trans. Power Syst., Vol. 15, No. 4, pp. 1232–1239.