Journal of Heuristics, 8: 173–195, 2002

A Seeded Memetic Algorithm for Large Unit Commitment Problems

JORGE VALENZUELA* AND ALICE E. SMITH
Department of Industrial and Systems Engineering, Auburn University, 207 Dunstan Hall, Auburn University, AL 36849-5346, USA
email: jvalenz@auburn.edu

Received September 9, 1999; Revised May 10, 2000

Abstract

The paper shows that the use of a memetic algorithm (MA), a genetic algorithm (GA) combined with local search, synergistically combined with Lagrangian relaxation is effective and efficient for solving large unit commitment problems in electric power systems. It is shown that standard implementations of GA or MA are not competitive with the traditional methods of dynamic programming (DP) and Lagrangian relaxation (LR). However, an MA seeded with LR proves to be superior to all alternatives on large problems. Eight problems from the literature and a new large, randomly generated problem are used to compare the performance of the proposed seeded MA with GA, MA, DP and LR. Compared with previously published results, this hybrid approach solves the larger problems better and uses less computational time.

Key Words: unit commitment, electrical power generation, genetic algorithm, Lagrangian relaxation, memetic algorithm

1. Introduction

The Unit Commitment Problem (UCP) is to determine a minimal cost turn-on and turn-off schedule of a set of electrical power generating units to meet a load demand while satisfying a set of operational constraints (Sheble and Fahd, 1994). The production costs include fuel, startup, shutdown, and no-load costs. Constraints include capacity reserve, minimum up/down time, maximum power flow in the transmission lines and operating limits. The UCP is a mixed combinatorial and continuous optimization problem, which is very complex to solve because of its enormous dimension, a non-linear objective function, and a large number of constraints. Well known traditional techniques such as integer programming (IP) (Dillon et al., 1978; Garver, 1963), dynamic programming (DP) (Lowery, 1996; Snyder, Powel, and Rayburn, 1987), branch and bound (Cohen and Yoshimura, 1983), Benders' decomposition (Baptistella and Geromel, 1980), and Lagrangian relaxation (LR) (Bard, 1988; Zhuang and Galiana, 1988) have been used to solve the UCP. More recently, metaheuristic approaches have been used such as simulated annealing (SA) (Zhuang and Galiana, 1990), tabu search (Xiaomin, Shahidehpour, and Erkeng, 1996), and genetic algorithms (GA)

*Author to whom all correspondence should be addressed.

(Kazarlis, Bakirtzis, and Petridis, 1996; Orero and Irving, 1998). Other problem-specific heuristics can be found in Lee (1980), Sheble (1990) and Tong, Shahidepour, and Ouyang (1991).

This paper puts forth a new approach for solving large UCP, a memetic algorithm (Gen and Cheng, 1997; Radcliffe, 1994) that combines a GA seeded with LR results with local search. The term memetic algorithm (MA) comes from the notion of a meme, a unit that can be modified by thought or experience before being passed along (Dawkins, 1976). This is different from a gene, which is passed unaltered by the being's experience. Radcliffe (1994) first formally defined a memetic algorithm as one that integrates local search as part of the reproductive mechanism. Results show that this approach can yield excellent schedules at computational times that allow timely management of electrical power systems.

2. Unit commitment problem formulation

As is true for many systems, an electric power system experiences cycles. The demand for electricity is higher during the daytime and lower during the late evening and early morning. This cyclical demand requires that utility companies plan for generation of power on an hourly basis. The problem is first to decide which of the available units to turn on, and then to determine an economical dispatch schedule of the units. Determining an optimal economical dispatch schedule of a set of generating units to meet a load demand while satisfying a set of operational constraints is called the unit commitment problem (UCP). The following notation is used:

System parameters:
- $CF_i(p)$: Cost of producing p units of power by unit i
- SU_i: Start up cost of unit i
- $u(t)$: Load at time t (demand)
- $r(t)$: Power reserve at time t (in case of unit failures)

Decision variables:
- $P_i(t)$: Amount of power produced by unit i at time t
- $v_i(t)$: Control variable of unit i at time t

$$v_i(t) = \begin{cases} 0 & \text{if unit } i \text{ is off at time } t \\ 1 & \text{if unit } i \text{ is on at time } t \end{cases}$$

Auxiliary variables:
- $x_i(t)$: Consecutive time that unit i has been up (+) or down (−) at time t
- $I(x)$: Logic function defined by

$$I(x) = \begin{cases} 0 & \text{if } x \text{ is false} \\ 1 & \text{if } x \text{ is true} \end{cases}$$

The objective of the standard UCP is to minimize the sum of two cost terms. The first term is the cost of the power produced by the generating units, which depends on the amount

of fuel consumed. The second term is the start-up cost of the generating units, which for thermal units, depends on the prevailing temperature of the boilers.

2.1. *Fuel cost*

For a given set of N committed units at hour t, the total fuel cost, at that particular hour, is minimized by economically dispatching the units subject to the following constraints:

(a) The total generated power must be equal to the demand (also called load).
(b) The power produced by each unit must be within certain limits (minimum and maximum capacity).

This problem is called Economic Dispatch (ECD) and it can be stated as follows (the subscript t is omitted for simplicity):

$$\begin{aligned} \min \quad & CF = \sum_{i=1}^{N} CF_i(P_i) \\ \text{Subject to} \quad & 1) \ \sum_{i=1}^{N} P_i = u \\ & 2) \ P_i^{\min} \le P_i \le P_i^{\max} \quad i = 1, 2, \ldots, N \end{aligned}$$

2.2. *Start-up cost*

The start-up costs relate to turning a unit on. If the thermal unit has been off for a long period, a cold start-up cost will be incurred. If the unit has been recently turned off (temperature of the boiler is still high), a hot start-up cost is applied. Two functions are commonly used to model start-up costs as a function of the temperature: two-step (Kazarlis, Bakirtzis, and Petridis, 1996) and exponential functions (Bard, 1988; Turgeon, 1978).

(a) Two-step function

$$S(t) = \begin{cases} S_h & \text{if } -x(t) \le t_{\text{cold start}} \\ S_c & \text{otherwise} \end{cases}$$

$t_{\text{cold start}}$ is the number of hours that it takes for the boiler to cool down. The S_h and S_c costs are the start-up costs incurred for a hot and cold start, respectively.

(b) Exponential function

$$S(t) = b_0\left(1 - e^{-\frac{\max(0, -x(t-1))}{\tau}}\right) + b_1$$

Start-up costs are incurred only when a transition from state off to on occurs, which can be expressed as follows:

$$CS(t) = S(t)v(t)(1 - v(t-1))$$

2.3. Objective function

Consequently, the objective function of the unit commitment problem for N generating units and T hours can be written as follows:

$$\min \sum_{t=1}^{T} \sum_{i=1}^{N} [CF_i(P_i(t))v_i(t) + CS_i(t)]$$

Subject to the constraints

(a) Demand:

$$\sum_{i=1}^{N} v_i(t)P_i(t) = u(t) \quad t = 1, \ldots, T$$

(b) Capacity limits:

$$v_i(t)P_i^{\min} \leq P_i(t) \leq v_i(t)P_i^{\max} \quad t = 1, \ldots, T; i = 1, \ldots, N$$

(c) Minimum uptime and minimum downtime:[1]

$$v_i(t) \geq I(1 \leq x_i(t-1) \leq t_{\text{up}} - 1) \quad t = 1, \ldots, T; i = 1, \ldots, N$$
$$v_i(t) \leq 1 - I(-t_{\text{down}} + 1 \leq x_i(t-1) \leq -1) \quad t = 1, \ldots, T; i = 1, \ldots, N$$

(d) Power reserve:

$$\sum_{i=1}^{N} v_i(t)P_i^{\max} \geq u(t) + r(t) \quad t = 1, \ldots, T$$

The total amount of power available at each hour must be greater than the load demanded. The reserve power available, denoted by $r(t)$, is used when a unit fails or an unexpected increase in load occurs.

3. The genetic approach

A genetic algorithm (GA) is an adaptive search based on natural selection, reproduction and mutation (Gen and Cheng, 1997; Goldberg, 1989). The following sections define the genetic encoding and operators that are used in the GA, the MA and the seeded MA.

3.1. Solution encoding

A solution of the UCP is determined by both the on-off schedule, $v_i(t)$ (0-1 values), of the units at each hour and the power, $P_i(t)$ (real values), generated by each individual unit

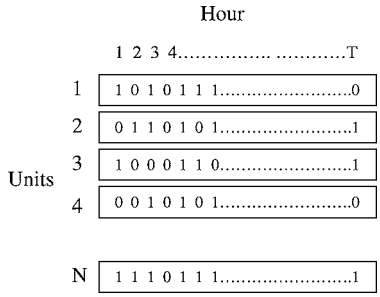


Figure 1. Encoding an on-off schedule (matrix **V**).

every hour. In this approach, the values of $v_i(t)$ are generated by the GA, while the optimum values of $P_i(t)$ are computed by solving a nonlinear program with $2N$ constraints. The on-off schedule of the units is stored as an integer-matrix **V** and the power generated as a real-matrix **P** with dimension $N \times T$. Figure 1 illustrates an example of encoding an on-off schedule. To save computer memory, only one matrix **P** is stored, which corresponds to the GA solution for which the fitness value is being computed. Using this encoding, the computer memory required for storing a population of 50 solutions of a power generation system with 100 units and a time horizon of 24 hours is approximately 118 Kbytes ($50 \times 100 \times 24/1024$ Kbytes). An alternative encoding, used in Mantawy, Abdel-Magid, and Selim (1997), is to represent each row of the matrix **V** by its corresponding decimal number. Although this can obtain a significant reduction in computer memory, the processing time increases since every time the fitness function and constraints are evaluated the decimal numbers need to be translated to their corresponding 0-1 values. A discussion of the matrix representation in GA can be found in Whitley (1997), however this focuses on encoding permutations rather than zero/one entries.

3.2. Fitness function

The fitness of a solution **V** is evaluated as the total production cost plus a penalty for those constraints that are violated:

$$\text{Fitness}(\mathbf{V}) = \text{Total\_fuel\_cost}(\mathbf{V}) + \text{Total\_start\_up cost}(\mathbf{V}) + \text{Penalty}(\mathbf{V})$$

To evaluate the total fuel cost of a given solution **V**, the optimal value of **P** needs to be determined by solving T different (one at each hour) ECD sub-problems. An ECD sub-problem consists of economically dispatching those units whose entries in **V** are equal to "1". Using the Lagrange multiplier, the ECD sub-problem at hour t can be stated as the following system of N equations:

$$\frac{\partial}{\partial P_i(t)} \left\{ \sum_{i=1}^{N} CF_i(P_i(t)) + \lambda(t) \left[u(t) - \sum_{i=1}^{N} P_i(t) \right] \right\} v_i(t) = 0 \quad i = 1, \ldots, N$$

Subject to the constraints

$$v_i(t)P_i^{\min} \le P_i(t) \le v_i(t)P_i^{\max} \quad i = 1, \ldots, N$$

Assuming that the fuel cost can be modeled by a quadratic function (Bard, 1988; Kazarlis, Bakirtzis, and Petridis, 1996; Turgeon, 1978; Wood and Wollenberg, 1996), the fuel cost of producing p MW of power using generating unit i is found by:

$$CF_i(p) = a_{0i} + a_{1i}p + a_{2i}p^2 \quad (\$/\text{h}) \tag{1}$$

The coefficient terms are determined by the price of the fuel ($/Mbtu) and the amount of fuel required per unit of power generated (Mbtu/MW). The Lambda-iteration method, described in (Wood and Wollenberg, 1996), is used to solve this system of equations. The procedure has complexity O(N) and is as follows:

Lambda-iteration procedure

1. Set initial values for λ(t) and Δ
2. Repeat
   - 2.1 For $i = 1$ to N
     - 2.1.1. Calculate P_i(t) from $\dfrac{dCF_i(P_i(t))}{dP_i(t)} = \lambda(t)$
   - 2.2 Calculate $\varepsilon = u(t) - \sum_{i=1}^{N} P_i(t)$
   - 2.3 if $(\varepsilon < 0)$ then set $\lambda(t) = \lambda(t) - \Delta$
     else set $\lambda(t) = \lambda(t) + \Delta$
   - 2.4 Set $\Delta = \Delta/2$
3. Until $|\varepsilon| \le$ *Tolerance*

Once the optimal values of $P_i(t)$ are found, the total fuel cost is computed by adding the fuel cost of all generating units over the time horizon T. The total start-up cost is calculated by adding the start up costs of those units that change their states from "0" to "1".

Finally, the penalty term is evaluated by adding the penalties of unmet load and unmet minimum up/down time constraints. The penalty due to unmet load is computed by penalizing the unmet load and power reserve requirements with a unit figure of m $/MWH:

$$LP = m\left\{\sum_{t=1}^{T}\max\left\{0, u(t) + r(t) - \sum_{i=1}^{N} v_i(t)P_i^{\max}\right\} + \sum_{t=1}^{T}\max\left\{0, u(t) - \sum_{i=1}^{N} v_i(t)P_i(t)\right\}\right\}$$

The penalty due to unmet minimum up/down time constraints is a dynamic penalty that is minimal at the early stages of evolution and grows with each generation. This allows solutions that satisfy the load requirements and do not meet the minimum up/down time constraints to evolve into feasible solutions. The unmet minimum up/down time penalty is

calculated as follows:

$$Min\,up\,|\,down\,penalty = k \times n\sqrt{ngen}$$

where n is the total number of minimum up/down time constraints violated, *ngen* is the number of generations and k is a user defined constant.

3.3. Parent selection

Tournament selection is used for selecting parents with the first parent the better of two solutions in tournament and the second parent uniformly randomly chosen from the population. While it is more common to have both parents selected by a tournament, this increases selective pressure that can lead to convergence to local optima. By choosing one parent randomly (essentially a tournament of size one), the selection pressure is reduced which tends to diversify the population. Blickle (1997) provides a good discussion of selection intensity relative to tournaments of various sizes.

3.4. Crossover

After using the standard two-point crossover operator and obtaining deficient results, a window crossover operator was devised. This crossover operator is quite similar to that of Beasley (1997), who also uses a matrix encoding. He applies three types of crossover operators—whole row, whole column and square-patch. This latter type is essentially the same as the window crossover. This crossover works by randomly choosing a window size. If the window size is smaller than the remaining portion of the matrix, a child is created with the window portion from the worse parent and the remaining portion from the better parent. This is reversed if the window is larger than the remaining matrix entries, i.e., the child has more material from the better parent, a version of parameterized uniform crossover (Spears and DeJong, 1991). Figure 2 shows an example of the crossover operator.

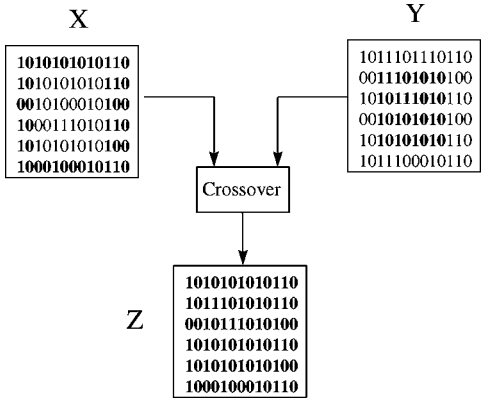


Figure 2. Window crossover operator.

3.5. Mutation

A standard mutation operator, where a bit of an offspring is flipped with probability 0.001, is used.

3.6. Evolution of current population

At the beginning of the algorithm an initial population of size p is randomly generated. Then parents are chosen from the initial population to procreate a new population of p solutions throughout crossover and mutation. The offspring are set in competition for survival with the current population using a probabilistic version of $(\mu + \lambda)$ selection (Schwefel, 1995). The probabilistic aspect helps prevent premature convergence to a local minimum but still maintains selection pressure. The contest for survival is as follows:

Replacing current population procedure

1. Define X as a member of the current population and Y as an offspring.
2. For each member X do
   - 2.1. Select randomly a member Y from the offspring.
   - 2.2. If fitness(Y) < fitness(X)
     - Add X to the new generation.
     - Else
     - Add Y to the new generation.
   - 2.3. Remove X from current population.
   - 2.4. Remove Y from eligible offspring.
3. Endfor.

3.7. Parameter tuning

The algorithm was tuned using a small test problem (Wood and Wollenberg, 1996) consisting of four units and a time horizon of eight hours and adding a quadratic fuel cost term. The new system has an optimal solution of \$74,675 (found by enumeration). The system data is given in Tables 1 and 2.

The GA was run using $m = \$200$ MWH and a population size of 50 and terminated after no improvement in 100 consecutive generations. Because the GA response was quite variable over 20 replications (best solution \$74,675 and worst solution \$99,215), two operators were added. First, a *repair operator* was added. This operator repairs solutions that are infeasible regarding the minimum up/down constraints. The process of fixing a solution is done by evaluating the state of each unit, $x_i(t)$. As it was previously defined $x_i(t)$ denotes the consecutive time that unit i has been up $(+)$ or down $(-)$ at time t. For instance, $x_2(18)$ equal to -6 means that at hour 18 unit 2 has been down for 6 hours. The state of a unit is evaluated starting from hour 0. If at a given hour t the minimum up or downtime constraint is violated, the state (on/off) of the unit at that hour is reversed and $x_i(t)$ is updated. The process continues until the last hour has been reached. Figure 3 illustrates an example of applying this repair operator to unit 7.

Table 1. Test system (Wood and Wollenberg, 1996).

	Unit 1	Unit 2	Unit 3	Unit 4
$P^{\max}$ (MW)	300	250	80	60
$P^{\min}$ (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
t_{up} (h)	5	5	4	1
t_{down} (h)	4	3	2	1
S_{h} ($) (hot start)	500	170	150	0.00
S_{c} ($) (cold start)	1100	400	350	0.02
$t_{\text{cold start}}$ (h)	5	5	4	0
Initial state (h)	8	8	−5	−6

Table 2. Load and reserve (Wood and Wollenberg, 1996).

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

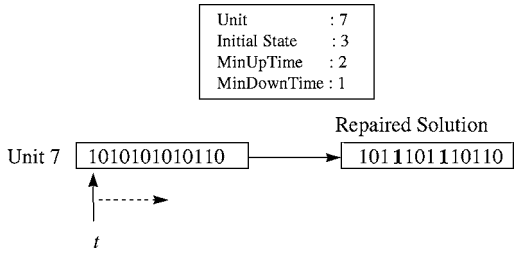


Figure 3. Repair operator.

A *swap mutation operator* was also added. The swap operator uses the average full-load costs (AFLC) of the generating units to perform a swap of unit states. The AFLC of a unit is defined as the cost per unit of power ($/MW) when the generator is at its full capacity. When the fuel cost is given by Eq. (1), AFLC can be expressed as:

$$AFLC = \frac{a_0}{P^{\max}} + a_1 + a_2 P^{\max} \quad (\$/\text{MW})$$

The generating units are ranked by their AFLC in ascending order. This means that for any two generating units i and j where $i < j$ the inequality $\text{AFLC}_i \leq \text{AFLC}_j$ holds. Units with lower AFLC should have higher priority to be dispatched. At a given hour, the operator

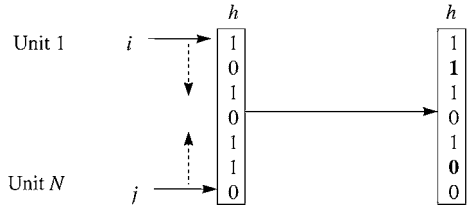


Figure 4. Swap operator.

probabilistically swaps the states of two units i and j only if the unit i is ranked better than unit j ($i < j$) and the state of the units are "off" and "on", respectively. This operator is a modified version of the priority list dispatching heuristic (Lee, 1991), where the generating units are committed in a pre-determined order.

Swap Operator Procedure

1. For $h = 1$ to T do
   1.1. Set $i = 1$.
   1.2. While ($i < N$)
      1.2.1. if (unit i at hour h is OFF) then
         1.2.1.1. Search for the most expensive unit j that is ON.
         1.2.1.2. With probability 0.7 unit i is turned ON and unit j is turned OFF.
      1.2.2. Set $i = i + 1$;
   1.3. Endwhile.
2. Endfor.

In figure 4 an example of applying the swap operator to units 2 and 6 at hour h is illustrated.

The repair and swap operators are applied with probabilities p_r and p_s, respectively. The minimum, maximum, and average of the best solutions over twenty replications is found in Table 3 for values of p_r and p_s equal to 0.3, 0.7, and 1.0. The combination $p_r = 0.7$ and $p_s = 0.3$ provided the best results, though the search is not very sensitive to changes in p_r and p_s.

4. The local search

A modified Lamarckian approach was used to augment the GA, creating an MA. At each generation, two local searches, 1-Opt and 2-Opt, were applied to the best solution of the new generation. 1-Opt flips one bit of the solution matrix **V**, whereas 2-Opt switches the states of two bits. The 2-Opt operator first searches for improvement by switching the states of two units at each hour. Then, the search continues by switching the states of a unit at two different hours. As soon as a better solution is found the local search is stopped and the modified solution replaces the original solution in the new generation. To perform the search more efficiently only changes that maintain solution feasibility regarding the minimum up and downtime constraints are attempted. Additionally, to avoid reiterative local search, local

Table 3. Parameter tuning results.

p_r	p_s	Best	Average	Worst
0.3	0.3	74675	74959	75012
0.7	0.3	74675	74909	75012
1.0	0.3	74675	74976	75012
0.3	0.7	74675	74992	75012
0.7	0.7	75008	75009	75012
1.0	0.7	74675	74991	75012
0.3	1.0	75008	75009	75012
0.7	1.0	75008	75008	75012
1.0	1.0	75008	75008	75012

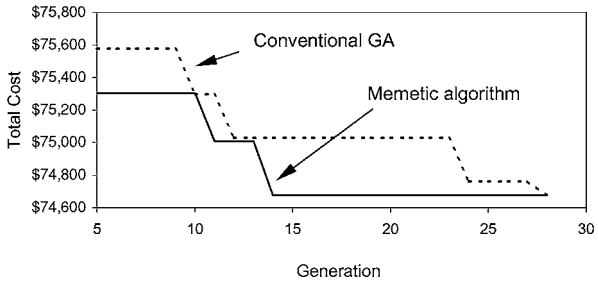


Figure 5. Typical performance of the MA versus the GA.

search is applied only if the solution is better than the best solution found so far. Figure 5 shows results obtained by including these elitist local-search operators. It can be observed that the memetic algorithm requires fewer generations to converge than the conventional GA.

5. Experimental results

After tuning, the MA was tested with eight problems taken from the literature (Bard, 1988; Kazarlis, Bakirtzis, and Petridis, 1996; Turgeon, 1978). The first three problems consist of ten units and a time horizon of 24 hours, and the data is given in Appendix A. The optimal solutions to these problems were found using DP with complete enumeration. The functions used to compute the start-up costs of the units are different for each problem.

Start-up Cost Functions

Problem 1 (Bard, 1988): $S(t) = b_0\left(1 - e^{-\frac{\max(0,-x(t-1))}{\tau}}\right) + b_1$

Problem 2 (Turgeon, 1978): $S(t) = b_0(1 - b_1 e^{-b_3 t})$

Problem 3 (Kazarlis et al., 1996): $S(t) = \begin{cases} S_c & \text{if } -x(t) \leq t_{\text{cold start}} \\ S_h & \text{otherwise} \end{cases}$

Table 4. Test results of traditional approaches.

			DP cost ($)	LR (5,000 iterations) cost ($)			
Problem	*N*	Search space	Optimum	Best	Average	Worst	CV (%)
P1	10	1.7E72	540904	541278	541605	541656	0.021
P2	10	1.7E72	59478	59485	59486	59491	0.004
P3	10	1.7E72	565827	566107	566493	566817	0.060
P4	20	2.9E144	NA	1128362	1128395	1128444	0.004
P5	40	8.3E288	NA	2250223	2250223	2250223	0.000
P6	60	2.4E433	NA	3374994	3374994	3374994	0.000
P7	80	6.9E577	NA	4496729	4496729	4496729	0.000
P8	100	2.0E722	NA	5620305	5620305	5620305	0.000

The other five problems are scaled versions of problem 3. The data of each unit of problem 3 was repeated 2, 4, 6, 8 and 10 times to obtain five problems with 20, 40, 60, 80, and 100 units, respectively. The hourly loads and reserve requirements were scaled by the same factors.

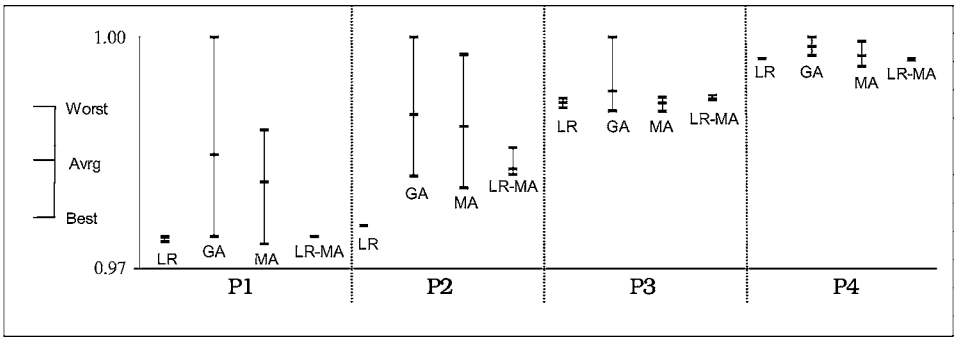
The results—best, average, worst and coefficient of variation (CV)—of the computational experiments are shown in Tables 4–6 with computational effort compared in Table 7. Table 4 shows best solutions for each of the eight problems obtained by traditional methods, DP and LR. The LR method was stopped when the procedure completed 5000 iterations. At each replication, the Lagrange multipliers were randomly initialized to make a fair comparison with the GA and MA procedures. Table 5 shows the results of the GA and MA and Table 6 shows the results of seeding the MA with the best solution obtained from the LR method. In this case, the LR was run once with fixed initial values for the Lagrange multipliers. The LR procedure was stopped after 100 iterations and the best solution was used to seed ten replications of the MA. Figures 6(a) and (b) illustrate for each problem

Table 5. Test results of GA and MA.

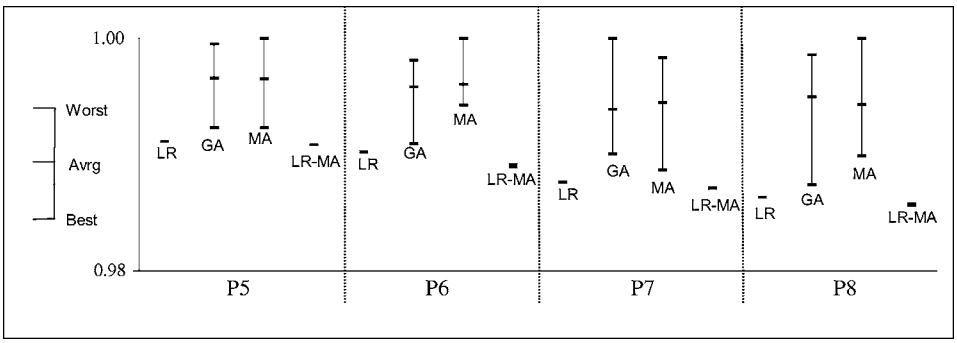
	Genetic algorithm cost ($)				Memetic algorithm cost ($)			
Problem	Best	Average	Worst	CV (%)	Best	Average	Worst	CV (%)
P1	541656	547527	555997	0.88	541108	545591	549290	0.61
P2	59882	60364	60977	0.74	59788	60271	60838	0.65
P3	565866	567329	571336	0.26	565827	566453	566861	0.07
P4	1128876	1130160	1131565	0.08	1127254	1128824	1130916	0.11
P5	2252909	2262585	2269282	0.28	2252937	2262477	2270361	0.29
P6	3377393	3394044	3401847	0.23	3388676	3394830	3408275	0.18
P7	4507692	4525204	4552982	0.21	4501449	4527779	4545305	0.33
P8	5626362	5669362	5690086	0.33	5640543	5665803	5698039	0.33

Table 6. Test results of using LR to seed the MA.

Problem	LR (100 iterations)	Best ($)	Average ($)	Worst ($)	CV (%)	LR + MA CPU time (sec.)
P1	541656	541656	541656	541656	0.000	2 + 39
P2	60100	59892	59936	60100	0.123	2 + 126
P3	567663	566686	566787	567022	0.028	2 + 59
P4	1129633	1128192	1128213	1128403	0.006	5 + 108
P5	2250223	2249589	2249589	2249589	0.000	14 + 203
P6	3374994	3370595	3370820	3371272	0.005	16 + 560
P7	4496729	4494214	4494378	4494439	0.001	17 + 647
P8	5621796	5616314	5616699	5616900	0.003	21 + 1317



(a)



(b)

Figure 6. (a) Results standardized by the worst value obtained by each method. (b) Results standardized by the worst value obtained by each method.

Table 7. Average CPU time of one replication (in seconds).

Problem	DP	LR	GA	MA	Combined LR and MA
P1	255	59	131	101	41
P2	207	55	209	161	128
P3	177	54	113	84	61
P4	–	108	374	287	113
P5	–	224	1600	1063	217
P6	–	335	3889	2772	576
P7	–	446	6995	5145	664
P8	–	626	15068	10463	1338

the results standardized by the worst value obtained by each method. The CPU times reported in Table 7 were averaged over ten replications on a Sun workstation. Recall that the GA and MA are terminated after 100 generations with no improvement in the best solution. While the MA takes marginally longer per generation because of the local search aspect,[2] it converges much more quickly resulting in significantly less overall computational effort.

Table 8 shows results from the GA approach of Kazarlis, Bakirtzis, and Petridis (1996). It can be seen that for the larger problems (P5 through P8), the MA seeded by LR of this paper provides better results at a reduction in computational time.[3] This is important because real UC problems are large with many generators to schedule. Smaller problems are interesting only for academic study and can reasonably be solved exactly using DP. The modest computational times of the seeded MA will allow this scheduling to be done in a timely manner, e.g., day to day. Looking at the worst results columns of Tables 6 and 8 show that the MA/LR hybrid is significantly less variable over random number seed and dominates the approach of Kazarlis, Bakirtzis, and Petridis (1996) for all problems.

To further gauge the performance of the hybrid approach and to ensure there was no bias in scaling the larger problems of a smaller problem, a new, randomly generated problem

Table 8. Results reported in [8].

Problem	Best	Worst	GA CPU Time (sec.)
P3	565825	570032	221
P4	1126243	1132059	733
P5	2251911	2259706	2697
P6	3376625	3384252	5840
P7	4504933	4510129	10036
P8	5627437	5637914	15733

Table 9. Test results for the large randomly generated problem.

Approach	Best ($)	Average ($)	Worst ($)	CV (%)	CPU Time (sec.)
LR	5774659	5779826	5783435	0.045	454
LR + MA	5770672	5774407	5776947	0.037	18 + 2022

with 100 units was created (the problem data is given in Appendix B). Table 9 shows the performance of the LR seeded MA against the LR run for 5000 iterations over ten seeds. It can be seen that the results of this problem are similar to those of the earlier problems, with the best performance of the hybrid approach improving 0.07% over LR with a manageable increase in computation time.

6. Discussion

Several heuristic methods for solving the UC problem have been described. The results obtained of solving nine instances of the UC problem showed that for problems of size equal or less to 10 units the DP approach is the best choice (the optimal solution can be found in less than four minutes). For problems of realistic size, consisting of 20 or more units, the combined use of the LR and MA methods provides the best results. For the larger problems, this method returned better solutions than those previously seen in the literature while reducing computational time significantly (after allowing for hardware differences).

While the results showed that the MA without seeding provides, in less time, comparable solutions to the GA, neither the MA nor the GA is as effective as the LR approach or the combination of LR and MA. Running the LR procedure for a short time (100 iterations, just enough to obtain a good feasible solution) and then using this solution to seed the MA is dependable, efficient and effective. For the larger problems, the solutions obtained using this approach were superior, even in the worst runs, to the results obtained by running the LR method much longer (5,000 iterations). While the MA was seeded with one copy of the LR solution, the approach was not sensitive to the number of copies and performed almost as well with five copies.

Because the schedules generated by the UCP are used continuously, any reduction in cost can result in significant savings over an annual period. Additionally, this cost reduction has positive environmental effects as fuel costs are minimized. As an example consider 1999 figures published by PJM Interconnection which manages the largest centrally dispatched electric system in North America (PJM Interconnections 1999). In 1999, the electric energy traded was $1.18 billion. PJM currently uses a central unit commitment algorithm run twice a day to schedule about 500 units. By moving from an LR approach to a seeded MA approach, 0.07% might be saved (the results on the largest problem reported here), saving over $800 thousand per annum. Furthermore, this approach is feasible from a computational viewpoint.

Appendix A

Table A1. System data for test problem P1 (Bard, 1988).

Unit	P^{max} MW	P^{min} MW	a_0	a_1	a_2	t_{up} hour	t_{down} hour	b_0	b_1	τ hour	Initial state hour
6	200	50	175	7.054	0.00515	2	2	1360	750	2	−1
2	400	130	400	7.654	0.00160	3	2	1460	650	3	5
8	375	110	400	7.762	0.00171	1	3	1370	550	3	−1
4	420	130	420	8.431	0.00150	1	3	1480	650	4	−2
10	250	75	200	8.149	0.00452	2	1	1180	625	2	7
9	850	275	725	8.162	0.00128	4	3	2200	950	4	2
3	600	165	600	8.752	0.00147	2	4	2100	950	4	1
7	750	250	600	9.121	0.00131	3	4	2300	950	4	6
1	1000	300	820	9.023	0.00113	5	4	2050	825	4	−4
5	700	225	540	9.223	0.00234	4	5	2100	900	3	−8

Table A2. Load for test problem P1 (Bard, 1988).

Hour	0	1	2	3	4	5	6	7
Load MW	1025	1000	900	850	1025	1400	1970	2400
Reserve MW	85	85	65	55	85	110	165	190
Hour	8	9	10	11	12	13	14	15
Load MW	2850	3150	3300	3400	3275	2950	2700	2550
Reserve MW	210	230	250	275	240	210	200	195
Hour	16	17	18	19	20	21	22	23
Load MW	2725	3200	3300	2900	2125	1650	1300	1150
Reserve MW	200	220	250	210	170	130	100	90

Table A3. System data for test problem P2 (Turgeon, 1978).

Unit	P^{max} MW	P^{min} MW	a_0	a_1	a_2	t_{up} hour	t_{down} hour	b_0	b_1	τ hour	Initial state hour
10	200	75	82	1.2136	0.00148	5	2	227	0.641	0.11	6
1	60	15	15	1.4	0.0051	5	2	85	0.588	0.2	6
5	150	50	29	1.54	0.00212	5	2	113	0.639	0.18	6
4	120	25	32	1.4	0.00382	5	2	94	0.65	0.18	6
2	80	20	25	1.5	0.00396	5	2	101	0.594	0.2	6
3	100	30	40	1.35	0.00393	5	2	114	0.57	0.2	6
8	150	50	100	1.3285	0.00135	5	2	282	0.749	0.09	6
7	520	250	105	1.3954	0.00127	5	2	267	0.749	0.09	6
6	280	75	72	1.35	0.00261	5	2	176	0.568	0.15	6
9	320	120	49	1.2643	0.00289	5	2	187	0.617	0.130	6

Table A4. Load for test problem P2 (Turgeon, 1978).

Hour	0	1	2	3	4	5	6	7
Load MWH	1459	1372	1299	1285	1271	1314	1372	1314
Reserve MWH	146	137	130	129	127	131	137	131
Hour	8	9	10	11	12	13	14	15
Load MWH	1271	1242	1197	1182	1154	1138	1124	1095
Reserve MWH	127	124	120	118	115	114	112	110
Hour	16	17	18	19	20	21	22	23
Load MWH	1066	1037	993	978	963	1022	1081	1459
Reserve MWH	107	104	99	98	96	102	108	146

Table A5. System data for test problems P3, P4, P5, and P6 (Kazarlis, Bakirtzis, and Petridis, 1996).

Unit	$P^{\max}$ MW	$P^{\min}$ MW	a_0	a_1	a_2	t_{up} hour	t_{down} hour	S_{h} $	S_{c} $	$t_{\text{cold start}}$ hour	Initial state hour
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 A6. Load for test problems P3, P4, P5, and P6 (Kazarlis, Bakirtzis, and Petridis, 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

Appendix B

Table B1. System data for random test problem.

Unit	P^{max} MW	P^{min} MW	a_0	a_1	a_2	t_{up} hour	t_{down} hour	S_h $	S_c $	$t_{cold\ start}$ hour	Initial state hour
1	382	47	689	19.84	0.00621	5	6	4500	8555	5	6
2	155	38	920	23.66	0.00476	4	4	4267	7202	2	−5
3	304	95	782	21.36	0.00316	1	2	3768	6923	5	2
4	477	97	520	22.50	0.00260	6	8	2886	7304	3	7
5	143	33	393	24.39	0.00550	7	2	4521	8634	1	8
6	301	134	788	20.86	0.00343	5	1	3009	5830	4	6
7	417	35	882	15.88	0.00584	1	5	3220	4340	2	2
8	458	66	690	17.13	0.00345	4	3	3999	5662	1	−4
9	393	135	702	16.03	0.00282	2	6	2915	7468	0	−3
10	499	137	812	20.31	0.00459	4	1	3375	5530	4	4
11	244	51	561	23.42	0.00317	5	3	1291	5002	2	−4
12	218	97	432	20.97	0.00358	8	3	3653	6265	2	9
13	490	136	777	21.73	0.00694	6	5	4244	8031	2	1
14	228	23	948	19.65	0.00581	7	5	1401	4014	3	−6
15	274	87	706	20.88	0.00131	8	5	815	3367	3	9
16	109	108	638	17.00	0.00406	3	1	3732	5075	1	4
17	157	54	784	23.47	0.00346	4	7	3665	4036	2	5
18	437	111	535	20.72	0.00327	8	1	2575	6337	5	−2
19	473	140	605	23.26	0.00612	2	4	50	1352	2	−5
20	256	121	405	25.27	0.00098	7	1	1587	4281	2	−2
21	240	38	599	23.52	0.00569	2	2	1616	4411	2	3
22	470	95	560	26.44	0.00186	6	6	336	3075	1	−7
23	351	58	407	20.19	0.00424	6	3	379	1158	4	−4
24	388	12	899	26.33	0.00694	6	6	4297	5667	2	7
25	345	99	915	16.53	0.00310	2	3	4615	5423	1	−4
26	271	136	470	26.30	0.00331	5	7	3667	7946	4	6
27	225	130	779	17.14	0.00438	8	2	2384	7235	1	−3
28	256	29	936	16.78	0.00102	2	4	2753	6944	1	−5
29	417	63	623	23.18	0.00694	7	3	1648	2363	4	8
30	213	60	451	16.53	0.00684	4	7	4607	6409	0	−8
31	386	13	835	21.32	0.00657	2	7	3072	7289	2	3
32	217	98	424	17.39	0.00634	3	1	603	1894	0	4
33	322	64	610	22.32	0.00166	4	2	3503	3619	2	−3
34	236	135	408	20.00	0.00313	7	6	1473	4901	3	8
35	370	131	452	20.28	0.00471	6	4	37	4946	1	−5

(*Continued on next page*.)

Table B1. (*Continued*).

Unit	P^{max} MW	P^{min} MW	a_0	a_1	a_2	t_{up} hour	t_{down} hour	S_h $	S_c $	$t_{cold\ start}$ hour	Initial state hour
36	260	108	371	24.50	0.00524	5	2	3883	8413	3	−3
37	422	23	519	18.21	0.00508	4	3	4354	7119	2	2
38	330	97	936	15.67	0.00191	4	3	1811	3318	5	5
39	48	29	720	21.40	0.00515	6	3	2067	4705	5	2
40	381	132	907	16.60	0.00212	5	4	1917	4468	3	6
41	486	137	518	18.75	0.00331	5	1	1231	2535	2	6
42	214	53	483	20.31	0.00162	8	6	3298	3418	1	9
43	396	78	976	21.74	0.00419	4	5	4524	7826	2	5
44	93	31	428	23.85	0.00451	6	5	977	1979	4	−6
45	217	155	845	15.69	0.00325	7	7	669	3083	1	8
46	179	125	799	25.25	0.00602	8	7	4349	5062	4	−8
47	470	74	755	15.32	0.00443	2	6	1569	2277	1	−7
48	322	90	977	15.50	0.00466	7	4	1922	5392	5	8
49	369	12	695	22.67	0.00285	6	5	2691	4567	4	−6
50	451	108	575	22.16	0.00404	6	6	3419	5135	4	7
51	266	114	531	18.86	0.00157	7	4	392	5120	4	−2
52	304	53	484	27.48	0.00321	7	5	3411	7616	1	2
53	165	135	905	22.81	0.00152	8	3	4279	8558	1	9
54	471	78	784	19.14	0.00473	2	2	3733	7912	4	−3
55	211	133	498	23.28	0.00447	2	7	2195	5927	2	3
56	458	91	433	15.17	0.00320	4	6	2580	6702	2	−7
57	227	67	734	26.18	0.00602	6	7	685	3471	4	−8
58	179	98	894	17.40	0.00277	7	2	2206	2533	3	6
59	432	125	674	18.17	0.00362	6	3	3698	6956	0	7
60	418	62	549	17.51	0.00231	3	1	3639	6446	5	−1
61	349	12	661	16.29	0.00159	2	2	1242	3405	3	3
62	266	37	719	18.82	0.00275	4	3	1806	2442	1	5
63	466	153	413	24.39	0.00461	7	4	1886	3815	5	−5
64	413	18	450	25.49	0.00065	4	5	2284	6256	3	−6
65	467	70	949	17.00	0.00212	7	8	3320	7581	1	8
66	132	55	475	15.31	0.00085	7	3	895	5484	1	−4
67	231	50	703	25.48	0.00343	4	4	412	4594	1	4
68	231	131	467	21.90	0.00164	2	3	3407	3837	2	3
69	319	90	765	27.58	0.00304	5	7	289	5083	1	−8
70	249	122	646	17.21	0.00159	3	7	4368	6089	0	−8
71	280	26	471	17.59	0.00437	5	1	1248	2315	3	6

(*Continued on next page*.)

Table B1. (*Continued*).

Unit	P^{max} MW	P^{min} MW	a_0	a_1	a_2	t_{up} hour	t_{down} hour	S_h $	S_c $	$t_{cold\ start}$ hour	Initial state hour
72	321	79	390	15.73	0.00179	4	2	1933	6059	2	−3
73	332	56	520	26.38	0.00618	2	1	4138	7278	4	3
74	176	120	489	26.80	0.00496	2	6	2601	3902	4	−7
75	414	147	810	22.17	0.00620	3	4	2331	6087	4	−5
76	430	113	446	19.85	0.00388	6	8	1584	2808	5	3
77	427	109	500	21.28	0.00684	4	1	1051	5194	1	5
78	263	97	732	25.53	0.00096	8	2	2973	4790	5	−2
79	257	33	909	24.76	0.00369	7	5	4598	6042	3	−6
80	124	50	973	25.32	0.00663	4	3	2370	3179	5	−4
81	465	94	545	23.54	0.00071	4	1	1083	1835	0	5
82	277	125	373	21.63	0.00254	8	8	2198	6011	2	9
83	384	136	623	17.82	0.00692	6	5	975	3907	1	−6
84	301	92	468	24.55	0.00262	6	2	352	1213	2	2
85	325	31	876	22.57	0.00319	5	2	1545	2325	4	6
86	317	107	762	18.88	0.00228	3	4	4161	7442	4	4
87	151	53	552	24.21	0.00288	7	2	1607	2870	5	−3
88	354	146	455	24.86	0.00417	5	5	1164	3126	3	−6
89	494	155	735	24.26	0.00065	7	4	1867	3011	3	8
90	158	48	650	26.02	0.00516	6	1	4977	8535	1	7
91	257	82	686	23.00	0.00259	2	4	3212	4399	4	3
92	178	130	918	26.98	0.00586	4	3	2131	5694	2	5
93	493	131	517	20.60	0.00379	6	5	2745	6245	1	7
94	132	107	701	24.51	0.00164	2	7	2469	5422	5	3
95	451	74	580	27.51	0.00559	1	4	3969	4991	4	2
96	492	143	946	25.90	0.00382	2	4	718	1479	1	3
97	292	18	818	23.33	0.00578	7	6	2627	6847	0	−7
98	171	90	692	15.86	0.00159	6	3	1980	5238	3	−4
99	60	58	614	25.99	0.00299	7	2	725	4210	2	−3
100	328	66	975	15.39	0.00172	5	2	2836	4906	2	5

Table B2. Load data for random test problem.

Hour	Load MW	Reserve MW
0	6712	671
1	8138	814
2	8892	889

(*Continued on next page.*)

Table B2. (*Continued*).

Hour	Load MW	Reserve MW
3	9891	989
4	10547	1055
5	11895	1189
6	10959	1096
7	11480	1148
8	12856	1286
9	14235	1424
10	14106	1411
11	14922	1492
12	14815	1482
13	12077	1208
14	12397	1240
15	10355	1035
16	10277	1028
17	10819	1082
18	11135	1114
19	13322	1332
20	12325	1233
21	10984	1098
22	9546	955
23	8764	876

Acknowledgment

Part of this research was supported by the U.S. National Science Foundation grant ECS 96 32702.

Notes

1. The minimum uptime and minimum downtime constraints state that a unit that is running must be up for at least t_{up} hours and a unit that is down must stay down for at least t_{down} hours. The minimum uptime constraints arise from physical considerations associated with thermal stress on the units and are designed to prevent equipment fatigue. The minimum downtime constraints, on the other hand, are based on economic considerations intended to prevent excessive maintenance and repair costs due to frequent unit cycling.
2. The local search is done less than once per generation as it is only applied to solutions better than any yet seen.
3. The Sun Ultra 2 with dual 200 MHz UltraSPARC CPU has a SPECfp95 of 14.7 while the HP Apollo 9000 model 720 used in Kazarlis, Bakirtzis, and Petridis (1996) has a SPECfp95 of 2.02; the hardware used in this paper is roughly 7 times faster than that of Kazarlis, Bakirtzis, and Petridis (1996).

References

Baptistella, L.F. and J.C. Geromel. (1980). "A Decomposition Approach to Problem of Unit Commitment Schedule for Hydrothermal Systems." *IEEE Proc*. 127(6), part D, 250.

Bard, J.F. (1988). "Short-Term Scheduling of Thermal-Electric Generators using Lagrangian Relaxation." *Operations Research* 36(5), 756–766.

Beasley, D. (1997). "Designing a Reduced-Complexity Algorithm for Quaternion Multiplication." In T. Baeck, D.B. Fogel, and Z. Michalewicz (eds.), *Handbook of Evolutionary Computation*. Bristol, UK: IOP Publishing Ltd. and Oxford University Press, Chap. G1.1.

Blickle, T. (1997). "Tournament Selection." In T. Baeck, D.B. Fogel, and Z. Michalewicz (eds.), *Handbook of Evolutionary Computation*. Bristol, UK: IOP Publishing Ltd. and Oxford University Press, Chap. C2.3.

Cohen, A.I. and M. Yoshimura. (1983). "A Branch and Bound Algorithm for Unit Commitment." *IEEE Trans. Power App. Syst*. PAS-102, 444–451.

Dawkins, R. (1976). *The Selfish Gene*. Oxford, UK: Oxford University Press.

Dillon, T.S., K.W. Edwin, H.D. Kochs, and R.J. Taud. (1978). "Integer Programming Commitment with Probabilistic Reserve Determination." *IEEE Trans. Power App. Syst*. PAS-97(6), 2154–2166.

Garver, L.L. (1963). "Power Generation Scheduling by Integer Programming—Development of Theory." *IEEE Trans. Power App. Syst*. PAS-82, 730–735.

Gen, M. and R. Cheng. (1997). *Genetic Algorithms & Engineering Design*. New York: John Wiley & Sons, 31–34.

Goldberg, D.E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison Wesley.

Kazarlis, S.A., A.G. Bakirtzis, and V. Petridis. (1996). "A Genetic Algorithm Solution to the Unit Commitment Problem." *IEEE Trans. on Power Systems* 11(1), 83–90.

Lee, F.N. (1980). "Short-Term Unit Commitment—A New Method." *IEEE Trans. on Power Systems* 3(2), 625–633.

Lee, F.N. (1991). "The Application of Commitment Utilization Factor (CUF) to Thermal Unit Commitment." *IEEE Trans. on Power Systems* 6(2), 691–698.

Lowery, P.G. (1996). "Generating Unit Commitment by Dynamic Programming." *IEEE Trans. Power App. Syst*. PAS-85(5), 422–426.

Mantawy, A.H., Y.L. Abdel-Magid, and S.Z. Selim. (1997). "A New Genetic Algorithm Approach for Unit Commitment." *Genetic Algorithms in Engineering Systems*: *Innovations and Applications*, IEE Conference Publication 0537-9989 1997 No. 446, Sept. 1997, pp. 215–220.

Orero, S.O. and M.R. Irving. (1998). "A Genetic Algorithm Modeling Framework and Solution Technique for Short Term Optimal Hydrothermal Scheduling." *IEEE Trans. on Power Systems* 13(2), 501–516.

PJM Interconnections web site, www.pjm.com, under Market and System Data, 1999 System Statistics.

Radcliffe, N.J. (1994). "Formal Memetic Algorithms." In T. Fogarty (ed.), *Evolutionary Computing*. Springer Lecture Notes in Computer Science, Vol. 865, pp. 250–263.

Schwefel, H.-P. (1995). *Evolution and Optimum Seeking*. New York: John Wiley & Sons.

Sheble, G.B. (1990). "Solution of the Unit Commitment Problem by the Method of Unit Periods." *IEEE Trans. on Power Systems* 5(1), 257–260.

Sheble, G.B. and G.N. Fahd. (1994). "Unit Commitment Literature Synopsis." *IEEE Trans. on Power Systems* 9(1), 128–135.

Snyder, W.L., H.D. Powel, and J. C. Rayburn. (1987). "Dynamic Programming Approach to Unit Commitment." *IEEE Trans. on Power Systems* 2(2), 339–350.

Spears, W.M. and K.A. DeJong. (1991). "On the Virtues of Parameterized Uniform Crossover." In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pp. 230–236.

Tong, S.K., S.M. Shahidepour, and Z. Ouyang. (1991). "A Heuristic Short-Term Unit Commitment." *IEEE Trans. on Power Systems* 6(3), 1210–1216.

Turgeon, A. (1978). "Optimal Scheduling of Thermal Generating Units." *IEEE Trans. on Automatic Control* AC-23(6), 100–105.

Whitley, D. (1997). "Permutations." In T. Baeck, D.B. Fogel, and Z. Michalewicz (eds.), *Handbook of Evolutionary Computation*. Bristol, UK: IOP Publishing Ltd. and Oxford University Press, Chap. C1.4.

Wood, A.J. and B.F. Wollenberg. (1996). *Power Generation, Operation, and Control*. New York: John Wiley & Sons.

Xiaomin, B., S.M. Shahidehpour, and Y. Erkeng. (1996). "Constrained Unit Commitment by Using Tabu Search Algorithm." In *Proceedings of the International Conference on Electrical Engineering*, Vol. 2, pp. 1088–1092.

Zhuang, F. and F.D. Galiana. (1988). "Towards a More Rigorous and Practical Unit Commitment by Lagrangian Relaxation." *IEEE Trans. on Power Systems* 3, 763–770.

Zhuang, F. and F.D. Galiana. (1990). "Unit Commitment by Simulated Annealing."*IEEE Trans. on Power Systems* 5(1), 311–317.