Optimal scheduling of thermal generating units in electric power systems

Narsimha Misra

University of Nevada, Las Vegas

Follow this and additional works at: https://digitalscholarship.unlv.edu/rtds

Repository Citation

https://digitalscholarship.unlv.edu/rtds/328
INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6” x 9” black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.
Optimal scheduling of thermal generating units in electric power systems

Misra, Narsimha, M.S.
University of Nevada, Las Vegas, 1993
Optimal Scheduling of Thermal Generating Units in Electric Power Systems

by

Narsimha Misra

A thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science
in
Electrical Engineering

Department of Electrical and Computer Engineering
University of Nevada, Las Vegas

December, 1993
The thesis of Narsimha Misra for the degree of Master of Science in Electrical Engineering is approved.

Chairperson, Yahia Baghzouz, Ph.D.

Examining Committee Member, Rama Venkat, Ph.D.

Examining Committee Member, Ashok Iyer, Ph.D.

Graduate Faculty Representative, Laxmi Gewali, Ph.D.

Graduate Dean, Ronald W. Smith, Ph.D.

University of Nevada, Las Vegas

December 1993
ABSTRACT

The Unit Commitment Problem (UCP) in electric power system problem that consists of finding the startup and shutdown schedule of generating units over a period of time (e.g., 24 hrs) so that the operating cost is minimized.

The UCP is often characterized by its prohibitive computational time and memory space requirement. The thesis investigates some computational aspects of the problem in an effort to improve the CPU time as well as the quality of the solution. Two algorithms that show significant improvement over existing methods are presented: One is based on the dynamic programming approach and designed for implementation on high performance computing machines with vector and parallel processing capabilities. The other is based on genetic algorithm techniques and designed for implementation on regular engineering workstations or fast personal computers.

Finally, the effect of transmission losses on the quality of the optimal scheduling and the computational time are investigated. Simulation results on 26- and 44-unit power systems are presented to illustrate the effectiveness of the proposed algorithms.
Contents

ABSTRACT .................................................................................................................... iii

LIST OF FIGURES ....................................................................................................... vi

ACKNOWLEDGMENTS .............................................................................................. vii

NOMENCLATURE ....................................................................................................... viii

1 INTRODUCTION .................................................................................................. 1

2 PROBLEM FORMULATION .............................................................................. 4

3 SOLUTION METHODS ....................................................................................... 7
   3.1 Supercomputer Algorithm ............................................................................ 7
      3.1.1 Supercomputer Features .................................................................... 7
      3.1.2 Dynamic Programming .................................................................... 8
      3.1.3 Vectorization ....................................................................................... 10
      3.1.4 Parallelization .................................................................................... 10
   3.2 Genetic Algorithm ......................................................................................... 13
      3.2.1 Introduction ....................................................................................... 13
      3.2.2 Application ........................................................................................ 16
List of Figures

3.1 Dynamic Programming Algorithm for Unit Commitment Program ........ 9

3.2 Flow Chart for Computing Start-Up and Generation Costs (a) Conventional
    Algorithm, (b) Modified Algorithm ..................................................... 11

3.3 Basic Genetic Algorithm ................................................................. 14

3.4 Flow Chart for GA Based Algorithm for Unit Commitment Problem ... 17

4.1 Flow Chart for Optimal Power Flow ............................................... 22

5.1 CPU Time of Proposed Algorithm for different Computer Codes and Win-
    dow Sizes ............................................................................................ 29

5.2 CPU Times of Vector/Parallel Code for Conventional and Modified Algo-
    rithms .................................................................................................. 29

5.3 Variation of Production Cost with Window Size .............................. 30

5.4 Fuel Cost for System A ................................................................. 32

5.5 Computation time for System A ...................................................... 33

5.6 Fuel Cost for System B ................................................................. 33

5.7 Computation time for System B ...................................................... 34
ACKNOWLEDGMENTS

My greatest appreciation goes to my committee chair and advisor, Dr. Yahia Baghzouz for his guidance, support, constant encouragement, patience in clarifying most of my doubts.

I would also like to thank Dr. Ashok Iyer for the support and encouragement while teaching me linear systems theory. I would like to thank Dr. Rama Venkat for agreeing to be a member of the graduate committee. I would like to thank Dr. Laxmi Gewali for all the help that he offered in computer algorithms.

Last but not the least, I would like to acknowledge the support of Cray Research Inc., who, through the National Supercomputing Center for Energy and Environment, Las Vegas, supported me financially through my master’s program.
Nomenclature

\[ G_{m,n} \] : power generated by unit \( n \) in period \( m \),

\[ C_i \] : Production cost function of unit \( i \),

\[ P^t_i \] : Power generated by unit \( i \) in period \( t \),

\( (u^t_i) \) : Commitment state (1/0) of unit \( i \) in period \( t \),

\[ S_i \] : Start-up cost function of thermal unit \( i \),

\[ X^t_i \] : Time duration for which unit \( i \) has been OFF,

\( O_{m,n} \) : \[ \begin{cases} 1 & \text{if unit } n \text{ is started in period } m, \\ 0 & \text{otherwise.} \end{cases} \]

\[ L_m \] : forecasted load demand for period \( m \),

\[ G^\text{min}_n \] : minimum generation capacity of unit \( n \),

\[ G^\text{max}_n \] : maximum generation capacity of unit \( n \),

\( t^\text{on}_{m,n} \) : time (hours) that unit \( n \) has been continuously

\hspace{2cm} \text{ON up to and including period } m,

\( t^\text{off}_{m,n} \) : time (hours) that unit \( n \) has been continuously

\hspace{2cm} \text{OFF up to and including period } m,

\[ T^\text{on}_n \] : minimum up time (hours) of unit \( n \),

\[ T^\text{off}_n \] : minimum down time (hours) of unit \( n \),

\[ R^t_i \] : maximum spinning reserve of unit \( i \) in period \( t \),

\[ R^t \] : total spinning reserve capacity of the system in period \( t \),

\[ \text{FCOST}(m,i) \] : total production cost for state \( i \) in period \( m \),

\[ \text{PCOST}(m,i) \] : generation cost for state \( i \) in period \( m \),
SCOST(m,i) : startup cost for state i in period m,
SUP(n) : startup cost of unit n.
$F_{ij}$ : Power flow in line $ij$,
$P^\text{max}_{ij}$ : Maximum power flow in line $ij$,
$NG_n$ : Total generation at node n,
$NC_n$ : Maximum generation allowed at node n,
$V_n$ : Voltage at node n,
$V_n^{\text{min}}$ : Minimum permissible voltage at node n,
$V_n^{\text{max}}$ : Maximum permissible voltage at node n,
$TL_t$ : Transmission Losses in period $t$,
$J$ : Jacobian Matrix,
$\bar{S}_{kj}$ : Complex power flow from node $k$ to node $j$,
$P_i$ : Real Power in node $i$,
$Q_i$ : Reactive Power in node $i$,
$v_i$ : Magnitude of the voltage at node $i$,
$\delta_i$ : Angle of the voltage at node $i$,
$Y_{kj}$ : Complex value of the $kj$ element of the admittance matrix,
$|Y_{ij}|$ : Magnitude of the $ij$ element of the admittance matrix,
$\theta_{ij}$ : Angle of the $ij$ element of the admittance matrix,
$\Delta P_i$ : Change in $P_i$,
$\Delta Q_i$ : Change in $Q_i$. 
Chapter 1

INTRODUCTION

Over the years, electric power utilities have tried to solve the problem of economically scheduling the generating units with only partial success. The problem of long-term and short-term scheduling, also called the Unit Commitment Problem (UCP), involves scheduling the start-up and shut-down information of the generating units present in a power system over a study period. The objective of the problem is to minimize the total cost by loading units in such a manner that the theoretically desirable equal incremental cost is achieved. Loading is constrained by the system and unit constraints. The problem can be described as a non-linear, large-scale, mixed integer optimization problem. Just like many other optimization problems, one could attempt to solve this problem either in its primal form or its dual form [1].

The exact solution of the UCP can be obtained by complete enumeration. The dynamic and integer programming based methods are designed to solve the problem in its primal form. A primal solution method resembles decision making in a regulated environment. Its advantage is the maintenance of solution feasibility and its disadvantage is the curse of dimensionality. Several modifications of the basic dynamic programming and integer
programming techniques have been developed to reduce the computational time [2]–[5]. The techniques basically differ from one another in the approximation used to reduce the problem dimensionality. In spite of the reduction in computation time, the proposed algorithms are complicated in terms of programming and often generate a suboptimal solution.

Extensive research has been done to solve the UCP using alternative methods. Priority listing [6], and artificial intelligence techniques such as expert systems and neural networks [7] – [9] are highly heuristic. The Lagrangian relaxation method [10], [11] tries to solve the problem in its dual form. Using predetermined hourly prices over the study period, the scheduling of each thermal unit is made individually to maximize profit. The schedules are then iteratively combined by adjusting the hourly prices. The advantage of this method is the problem decomposition resulting from the dual formulation, and the disadvantages are the difficulty associated with global coordination and the restrictions on the kinds of constraints and cost functions that can be used.

Two areas which have gained popularity as techniques to solve computationally intensive optimization problems in electric power systems are supercomputers and genetic algorithms. With the recent developments in supercomputers, there has been a significant effort in applying vector/parallel computation techniques to various problems associated with design and operation of power systems. Problems like load flow studies [13], [14], steady state security analysis [15], and transient analysis [16], have already been implemented on supercomputers with encouraging results.

There has also been considerable research done in applying genetic algorithm techniques to obtain solutions to problems in electric power systems. The genetic algorithm technique
has been used to solve problems like load flow [17], loss minimization in distribution systems [18], optimal capacitor selection for radial distribution systems [19], and calculation of worst case distribution harmonics, with very promising results.

This thesis reports the results of the investigation of the application of supercomputers to the UCP, and the application of genetic algorithm technique to the UCP. The algorithm used on the supercomputer is based on the dynamic programming method, but is modified to take advantage of the architecture of vector and parallel processors. The algorithm is tested on a power systems with 26 thermal units using a 24-hour load forecast. The genetic algorithm technique is applied on two different power systems, one with 26 thermal units, and two, with 44 thermal units, over two different 24-hour forecasted loads. Finally, the effect of transmission losses on the quality of the optimal scheduling and computational time are presented.

This thesis is divided into 6 chapters. The Introduction in Chapter 1 gives an overview of the thesis. Problem Formulation in Chapter 2 gives a detailed description of the mathematical formulation of the problem and the constraints associated with it. Solution Methods in Chapter 3, is divided into 2 sections. The supercomputer algorithm section describes the supercomputer architecture and modification of the dynamic programming technique to take advantage of the vector and parallel processing capabilities of the supercomputer. The genetic algorithm section describes the genetic algorithm as it is applied to the UCP. Chapter 4 is devoted to reformulating the UCP with transmission losses as constraints. Chapter 5 details the results of the tests performed to check the performance of the proposed algorithms. Chapter 6 ends the thesis report with a conclusion on the accomplishments made.
Chapter 2

PROBLEM FORMULATION

The unit commitment problem with \( N \) generating units and \( M \) time intervals can be stated as follows: Minimize the overall cost function \( F \), (refer to Nomenclature for symbols not defined in text),

\[
F = \sum_{i=1}^{N} \sum_{t=1}^{M} [G_i(p_t) + u_t^i(1 - u_{t-1}^i)S_i(X_t^i)]. \tag{2.1}
\]

The start-up cost can vary widely depending on how long the unit was turned OFF since the last time it was running. The general constraints that are placed on a unit commitment problem are load demand, spinning reserve, minimum up and down time constraints and the must-run or base units.

1) Load Demand Constraint: The load demand constraint is placed on the unit commitment problem in order to make sure that the forecasted load demand is taken into consideration when the scheduling of the units takes place. The total generating capacity of the committed units in a state should be greater than the forecasted load demand for that particular period, i.e.,

\[
\sum_{i=1}^{N} u_t^i G_n^{max} \geq L_m. \tag{2.2}
\]
2) **Generation Capacity Constraint:** The scheduled generation from a unit in a given period should be within its maximum and minimum generating capacities.

\[
G_{m,n}^{\text{min}} \leq G_{m,n} \leq G_{m,n}^{\text{max}}.
\]  

(2.3)

3) **Spinning Reserve Constraint:** Spinning reserve is the term used to describe the total amount of generation available from all units synchronized (i.e., spinning) on the system minus the present load. Spinning reserve must be carried so that the loss of one or more units does not cause too far a drop in system frequency. Mathematically,

\[
\sum_{i=1}^{N} u_i^t R^i \geq R^t.
\]  

(2.4)

4) **Must Run Units:** The must run or base units are the units that should always be on-line. These units are generally very large generating units with very high start up and shut down costs. They have a very low $/MWh ratio and therefore are kept on-line.

5) **Minimum Up Time Constraint:** Once the unit is turned ON, it should not be turned OFF immediately.

\[
(t_{m-1,n}^{\text{on}} - T_n^{\text{on}})(I_{m-1,n} - I_{m,n}) \geq 0.
\]  

(2.5)

6) **Minimum Down Time Constraint:** Once the machine is turned OFF, there is a minimum time before it can be turned back ON.

\[
(t_{m-1,n}^{\text{off}} - T_n^{\text{off}})(I_{m-1,n} - I_{m,n}) \geq 0.
\]  

(2.6)

6) **Crew Constraint:** If a plant consists of two or more units, they cannot be turned ON or
OFF at the same time.

Other than the constraints that have been mentioned above, there are other constraints imposed on a practical power system which are usually taken onto consideration in the UCP. Examples include area reserve constraint, and unit maximum contribution to serve constraint [18]. Furthermore, the startup cost is generally dependent upon the shut-down time. These constraints and the time dependency of $S_n$ are not considered in this thesis in order to maintain the relative simplicity of the algorithms. However, these constraints can be included in the formulation without affecting the basic structure of the algorithms.
Chapter 3

SOLUTION METHODS

This chapter describes the supercomputer algorithm and the genetic algorithm techniques that are used for solving the UCP with transmission losses ignored.

3.1 Supercomputer Algorithm

This section presents dynamic programming-based supercomputer algorithm for solving unit commitment problem. Methods for modifying the basic algorithm in order to take full advantage of vector and parallel processors are discussed. For clarity, a brief description of the features of the computing machine being used and a review of the basic dynamic programming algorithm are presented.

3.1.1 Supercomputer Features

The CRAY Y-MP2/216 at the NSCEE is a 2-processor machine. It has a clock speed of 6.0ns corresponding to a frequency of 167 MHz. The machine can execute 2 floating point instructions in one clock cycle which means that it can run at 333 MFlops per CPU. It has a main memory capacity of 16 Mwords (each word is 64-bits in length, each processor
has independent vector hardware). The machine has eight vector registers in each of its processors. The length of the vector registers is 64 words, i.e., a vector which has 64 elements will be able to take full advantage of the vector registers. Vector operations allow simultaneous operations on the elements of arrays and permit improvements in machine productivity of an order of magnitude or more depending on the specific computation. A vector calculation can approach a speed of one result per clock period with longer vector lengths. Furthermore, in many instances the result from one functional unit can be sent directly to the input of another functional unit, allowing mathematical operations to be chained together to provide a rate of up to two floating point results per clock period. The longer the vector, the faster the computation.

The main memory of 16 Mwords is shared by the two processors so the variables are actually located at one place and each processor uses them whenever needed. There are, however, two kinds of variables: shared variables and private variables. The shared variables are the variables that are common to both processors. When one processing unit is using this variable it is not possible for any other processing unit to use that variable. On the other hand, the private variables share the name of the variable but each processing unit has a different value and location for the variable.

3.1.2 Dynamic Programming

The dynamic programming algorithm for solving the UCP is a systematic search of all the feasible states. A state is a combination of the status of all the units in the system. A feasible state is a state for which the system and local constraints are satisfied. The search is recursively done and a decision is made for each step so that the objective of obtaining the
Figure 3.1: Dynamic Programming Algorithm for Unit Commitment Program

minimal total cost over the whole study period is achieved. The flow chart of the dynamic programming algorithm for the UCP with \( M \) periods is shown in Figure 3.1 (refer to Nomenclature in Nomenclature).

In a large power system, there are two variables that have to be heuristically predetermined in the algorithm in order to reduce the enormous computation time [18]. One is the number of saved states from the previous period, which is often called the window size (\( X \)). The other variable is the size of the search area for the current period (\( Y \)). These variables are also shown in Figure 3.1.
3.1.3 Vectorization

For each period, the conventional dynamic programming algorithm searches for a feasible state and then calculates the startup and generation costs (SCOST and PCOST) for that state as shown in Figure 3.2(a). The generation cost is found by using standard iterative methods, such as the lambda-iteration method or the second-order gradient method.

As indicated in Figure 3.2(a), the computation of PCOST and SCOST is performed iteratively in a DO loop. The number of states for which the computation of generation cost and startup cost of a system with \( N \) units is \( 2^N \). When the conventional dynamic programming is implemented on the supercomputer, the compiler cannot envision the long vector loops in the program because of the short vector loops formed by the inner DO loops as indicated in Figure 3.2(a). Consequently, the vectorized code will perform very poorly.

Since the compiler used to generate code in the supercomputer optimizes only the innermost loop, the algorithm in its present shape cannot take the greatest advantage of the vector hardware present in the supercomputer. To correct this deficiency, the algorithm is modified by making the larger loop \( Y \) as an inner loop as shown in Figure 3.2(b). The resulting large size of the inner vector \( (Y = 2^N) \) is then optimized by the compiler efficiently. By design, the modified program should run much faster than the conventional dynamic programming method.

3.1.4 Parallelization

The flowchart of the UCP (Figure 3.1) indicates that the iterations for calculating the expenditure for each period have to follow a certain sequence (i.e., the schedule of period \( m \) can be computed only after schedule of period \( m - 1 \) is computed). It is also evident that
Figure 3.2: Flow Chart for Computing Start-Up and Generation Costs (a) Conventional Algorithm, (b) Modified Algorithm
the computation of the cost incurred from starting up generating units and dispatching them in period \( m \) can be performed in parallel. This observation allows the use of the Parallel State Algorithm (PSA) [17] for solving the unit commitment problem on a multiprocessor machine like the CRAY Y-MP2/216. The PSA is outlined below:

(a) Each processing unit \( p, p=1,...,P \), computes SCOST and PCOST in its search area.

(b) Processing unit \( p \) sends the minimum cost in its search area to a specified processing unit to calculate overall minimum cost.

(c) Steps (a) and (b) are repeated for all \( M \) periods.

The above algorithm can be applied very efficiently on a parallel processing machine. Since the minimum cost is calculated for each period, each of the processors has to check for minimum cost after each state. While the present minimum cost is read from the memory of the individual processors every time to check for minimum cost, the minimum cost is saved very few times (i.e., only whenever the cost is less than the previously saved minimum cost). The reading is done entirely locally on each processor whereas writing requires communication with the main memory. The net result is that the vast majority of operations on the shared object do not require communication with the main memory. As a consequence, the processing is highly efficient. An almost linear speed up in computing time can be expected with an increase in the number of processors, when using the Parallel State Algorithm [17].
3.2 Genetic Algorithm

3.2.1 Introduction

Genetic algorithms are based on the process of natural selection, mating and evolution and the idea of the survival of the fittest. The most powerful feature of the genetic algorithms is that they can solve extremely difficult problems with little knowledge of the complex nature of the problem. They achieve this by encoding solutions to the problem into chromosomes. The chromosomes are evaluated for their fitness (worth) by using an evaluation function (usually the objective function). Genetic algorithms are based on the heuristic assumptions that the best solutions will be found in regions of the parameter space containing a relatively high proportion of good solutions and that these regions can be explored by the genetic operators of selection, crossover and mutation.

Genetic algorithms offer a number of advantages: (a) They search from a set of designs and not from a single design. (b) They are not derivative based. (c) They work with discrete and continuous parameters and they explore and exploit the parameter space. [24]. The flowchart shown in Figure 3.3 describes the basic structure of a genetic algorithm, which can be divided into three components, an evaluation module, a population module, and a reproduction module [23].

Evaluation Module

The Evaluation Module measures the worth of a given chromosome. The evaluation module actually decodes the chromosome by operating on the bit strings (representation of chromosomes in strings containing 0's and 1's), so that those with higher evaluations tend
to reproduce more often. The interaction of a chromosome with the evaluation function provides a measure of fitness that the genetic algorithm uses when carrying out reproduction. It is important to note that nowhere except in the evaluation function is there any information in the genetic algorithm about the problem to be solved.

Population Module

The Population Module contains a population of chromosomes and techniques for creating and manipulating that population. It contains information about the chromosomes
(representation technique), information about creating a starting population (initialization
technique), method for deleting chromosomes to replace with new chromosomes (deletion
 technique), and method used for selecting parents (parent selection technique). The pop-
ulation module interacts with the evaluation module during a run, in that whenever a new
set of children has been produced, the population module asks the evaluation module to
evaluate each child before that child is placed in the population.

Reproduction Module

The Reproduction Module contains techniques for creating new chromosomes during
reproduction. In each reproduction event, the reproduction module gets two parents from
the population module, applies a reproduction function and sends the two children to the
population module.

The one-point crossover is an example of a reproduction function. It can be explained
easily through a simple illustration. Take two parents A and B, with 8-bit strings:

\[
\begin{align*}
A &= 0101 \quad 1101 \\
B &= 0111 \quad 0110
\end{align*}
\]

applying one-point crossover after 4-bits (the crossover point can vary), we exchange the bit
values of both the parents after the fourth bit. The two children at result from this crossover
operation are,

\[
\begin{align*}
C &= 0101 \quad 0110 \\
D &= 0111 \quad 1101
\end{align*}
\]

There are various other schemes that can be used for reproduction, e.g., mutation, two point
crossover, etc [26]
3.2.2 Application

This section presents the application of genetic algorithm to the unit commitment problem. The flowchart describing the algorithm is shown in Figure 3.4.

The chromosomes (states) can be represented as bit strings with the bit values representing the ON-OFF status of the units. Therefore the length of the bit strings in each state corresponds to the number of units in the power system under consideration. The initial states are formed by randomly generating bit strings and checking for their feasibility (initialization, Step I). Bit strings are added to the initial population only if they satisfy the constraints.

The evaluation module (Step II) consists of the objective function which calculates the fuel cost (startup and generation cost) for that state. It also computes the minimum fuel cost for that generation. The equation for the calculation of fuel cost is given by:

\[
FCOST = \sum_{n=1}^{N} C_n (G_{m,n} + O_{m,n} S_n),
\]

(3.1)

The fitness of a state, which is a part of the evaluation module, is calculated by using the following equation:

\[
F(c) = \frac{D(c)}{\sum_{c=1}^{NC} D(c)},
\]

(3.2)

where \(D(c)\) is the difference between the fuel cost of state \(c\) and the maximum fuel cost in the population. \(NC\) is the number of states in a generation (size of the population).

New states (Reproduction Step III) are created by mating current states. The parent selection process that is used in producing new states is the roulette wheel selection process
Figure 3.4: Flow Chart for GA Based Algorithm for Unit Commitment Problem
[16], which is like a pie chart with the minimum cost state having the maximum piece of the pie. A random number is generated and the state assigned to that portion of the pie is chosen as the parent.

When replacing current states with new states, elitist approach is followed, which copies the best states of the current generation into the next generation. This makes sure that the best solutions are not eliminated in the recombination process.

Penalty factors are added to the fitness values of states which do not satisfy the constraints, thereby eliminating the possibility of their reproduction. The unit constraints need to be checked for satisfaction only during the very beginning when the initialization process of generating new states (Step I). The process of checking for constraint satisfaction need not be carried in the reproduction module because the new states are generated from the old states that have already satisfied all the unit constraints.
Chapter 4

UCP INCLUDING TRANSMISSION LOSSES

Inclusion of transmission losses into the UCP has received very little support in the research community and has potential for vast savings for the electric power utility industry although the power losses amount to almost 2% of the total load demand. This is a significant value considering that the load demand under consideration is of the order of thousands of Megawatts. If the power losses are taken into account in generation scheduling, the commitment schedule might change giving an entirely different schedule, and a better solution to the minimum cost objective function is determined. [27], [28] and [29] have tried to include the transmission losses into the economic dispatch problem.

4.1 Problem Reformulation

The UCP including transmission losses can be formulated as described in 2.1, with the addition of losses into the total generation. The problem constraints generally considered include all the constraints described in Chapter 2, and also:
a) **Load Demand Constraint:** The load demand constraint has to be reformulated in the unit commitment problem including transmission losses in order to make sure that the forecasted load demand and also the transmission losses are taken into consideration when scheduling of the generating units takes place. The total generation capacity of the committed units in a state should be equal to the sum of forecasted load demand and transmission losses for that particular period. The transmission losses depend on the solution to the power flow equations of the network.

\[ \sum_{n=1}^{N} G_{m,n} = L_m + T L_t. \]  \hspace{1cm} (4.1)

b) **Optimal Power Flow Constraints:** The power transfer in a transmission line cannot exceed a pre-determined limit, in order to protect the line from overloading. The power generated at a node is limited by a pre-determined limit. This is necessary in order to take into account the fuel availability constraint, wherein the amount of fuel available at a particular node for a given interval cannot exceed a specific amount thereby limiting the amount of power generated at that node. The voltage at a node in a given interval is bounded by a maximum and minimum value.

i) **power transfer constraint,**

\[ F_{ij} \leq F_{ij}^{\text{max}}, \]  \hspace{1cm} (4.2)
ii) node generation constraint,

\[ NG_n \leq NC_n, \quad (4.3) \]

iii) voltage limit constraint,

\[ V_{n}^{\text{min}} \geq V_n \leq V_{n}^{\text{max}}. \quad (4.4) \]

Inclusion of transmission losses into the UCP requires the power flow equations as additional constraints. The optimal power flow has to be solved for each feasible state (a state where the operating constraints are satisfied) and the transmission losses for that state have to be calculated. This requires that the jacobian matrix be formed and inverted at each state.

Transmission losses can be computed by running an optimal power flow program that minimizes the fuel cost, while determining the active and reactive power outputs of the generators, and transmission losses associated with the dispatch. The flow chart for this program is the same as the one in 3.1, for the addition of the cost of losses in the fuel cost function.

4.2 Power Flow Solution

The optimal power flow portion of the problem under consideration requires that the power flow equations be solved. The power flow equations can be arranged in matrix form and the inverse of the matrix needs to be computed in order to solve the equations. The
Figure 4.1: Flow Chart for Optimal Power Flow
matrix is popularly known as the Jacobian matrix, and the solution method used is called the Newton-Raphson Solution to the power flow equations. All the busses excluding the swing bus are regarded as PQ busses and the swing bus voltage is assigned a constant value while allowing the active and reactive power outputs of that bus are variable.

The jacobian matrix is partitioned into four sub-matrices, i.e.,:

\[
J = \begin{bmatrix}
\frac{\partial P}{\partial \delta} & \frac{\partial P}{\partial V} \\
\frac{\partial Q}{\partial \delta} & \frac{\partial Q}{\partial V}
\end{bmatrix}, \quad (4.5)
\]

The elements of these sub-matrices are defined as follows:

\[
\frac{\partial P_i}{\partial \delta_i} = \sum_{j=1}^{N} v_i v_j |Y_{ij}| \sin(\delta_i - \delta_j - \theta_{ij}) \quad (4.6)
\]

\[
\frac{\partial P_i}{\partial \delta_k} = -v_i v_k |Y_{ik}| \sin(\delta_i - \delta_k - \theta_{ik}) \quad (k \neq i) \quad (4.7)
\]

\[
\frac{\partial P_i}{\partial v_i} = -\sum_{j=1}^{N} v_j |Y_{ij}| \cos(\delta_i - \delta_j - \theta_{ij}) - 2 v_i |Y_{ii}| \cos(-\theta_{ii}) \quad (4.8)
\]

\[
\frac{\partial Q_i}{\partial \delta_i} = -\sum_{j=1}^{N} v_i v_j |Y_{ij}| \cos(\delta_i - \delta_j - \theta_{ij}) \quad (4.9)
\]

\[
\frac{\partial Q_i}{\partial \delta_k} = v_i v_k |Y_{ik}| \cos(\delta_i - \delta_k - \theta_{ik}) \quad (k \neq i) \quad (4.10)
\]

\[
\frac{\partial Q_i}{\partial v_i} = -\sum_{j=1}^{N} v_j |Y_{ij}| \sin(\delta_i - \delta_j - \theta_{ij}) - 2 v_i |Y_{ii}| \sin(-\theta_{ii}) \quad (4.11)
\]
\[
\frac{\partial Q_i}{\partial v_k} = -v_i|Y_{ik}| \sin(\delta_i - \delta_k - \theta_{ik}). \quad (k \neq i) \tag{4.13}
\]

Once feasible schedule is found with losses ignored, the power flow routine is performed with \(\delta\) and \(V\) initialized to 0.0 and 1.0 pu, respectively. The changes in \(P_i\) and \(Q_i\) are calculated using the mismatch equations given below:

\[
\Delta P_i = -\sum_{j=1}^{N} |Y_{ij}|v_j|v_i| \cos(\delta_i - \delta_j - \theta_{ij}) + P_i \tag{4.14}
\]
\[
\Delta Q_i = -\sum_{j=1}^{N} |Y_{ij}|v_j|v_i| \sin(\delta_i - \delta_j - \theta_{ij}) + Q_i \tag{4.15}
\]

The new values of \(\delta\) and \(V\) are then calculated by:

\[
\begin{bmatrix}
\delta^{(l+1)} \\
V^{(l+1)}
\end{bmatrix}
= \begin{bmatrix}
\delta^{(l)} \\
V^{(l)}
\end{bmatrix} - J^{-1} \begin{bmatrix}
\Delta P \\
\Delta Q
\end{bmatrix}. \tag{4.16}
\]

Finally, the power loss in the line connecting nodes \(j\) and \(k\) is computed from the real part of the complex powers defined by:

\[
\overline{S}_{kj} = \overline{Y}_{kj}^* (v_k^{L\delta_k} - v_j^{L\delta_j})^* v_k^{L\delta_k}, \tag{4.17}
\]
\[
\overline{S}_{jk} = \overline{Y}_{jk}^* (v_j^{L\delta_j} - v_k^{L\delta_k})^* v_j^{L\delta_j}. \tag{4.18}
\]

### 4.3 Supercomputer Implementation

The supercomputer used for implementing the algorithm above is the CRAY Y-MP2E/216 described in Chapter 3. The most time consuming portion of the unit com-
mitment problem including transmission losses is the optimal power flow portion where the economic dispatch is determined with losses included. As seen in Figure 4.1, the optimal power flow includes solving the power flow problem. The main difficulties encountered in applying a vector processor to the solution of a power flow is the sparsity of the power flow matrices which result in very short vector when the non zero terms are gathered into a packed vector format. With the sparsity typically found in power system matrices, there are insufficient number of terms in the packed vectors to allow the vector processor to run efficiently.

The “AC” load flow method is used to solve the power flow problem, and requires the use of cosine and sine operations in the computation of the elements in the jacobian matrix. To reduce the use of cosine and sine operations, all values used in computations are stored in complex form. Voltages are stored in both polar and complex forms. When any of these voltages is updated, the complex form is immediately computed and stored via the use of cosine and sine operations, resulting in faster vector operations.

The inversion of the jacobian matrix was performed using LDU decomposition, and using the forward/backward substitution method (also called the inverse factors method). The jacobian is factorized in LDU form. By setting $W = L^{-1}$ and $\bar{W} = U^{-1}$ the steps of forward and backward substitution can be restated as

$$z = Wb; \quad x = \bar{W}y. \quad (4.19)$$

Since the jacobian matrix is symmetric, $\bar{W} = W^T$. Matrix $L$ can be written as the product of $n$ matrices $L_i$, $n$ being the order of matrix $L$ and of any of the $L_i$. Each $L_i$ is
equal to the $n$-order identity matrix with the exception of the $i$-th column which is equal to the corresponding column of $L$. The $W$ matrix can be written as

$$W = L_n^{-1} L_{n-1}^{-1} L_{n-2}^{-1} \ldots L_1^{-1},$$  \hspace{1cm} (4.20)

where $L_i^{-1}$ is obtained by simply reversing the sign of the nonzero off-diagonal elements of $L_i$. If $p$ partitions are considered, matrix $W$ can be rewritten as

$$W = W_{(p)} W_{(p-1)} W_{(p-2)} \ldots W_{(1)}, \hspace{1cm} W_{(p)} = \prod_{k \in S_p} L_k^{-1}.$$  \hspace{1cm} (4.21)

The use of matrix $W$ allows the replacement of the strictly sequential forward and backward solution algorithms with matrix-vector multiplications. In this way all the multiplication operations relative to each column of $W$ can be executed by a single vectorizable DO loop.
Chapter 5

NUMERICAL EXAMPLES

The proposed algorithms are tested using different electric power systems. The supercomputer algorithm is tested using a 26-thermal unit power systems with 24-hour forecasted load [9]. The genetic algorithm is tested using two systems: the 26-thermal unit power system above and, a 44-thermal unit power system with a 24-hour forecasted load [26] for each of the systems. The quality of the optimal scheduling solution while including transmission losses is tested on a 22 unit, 10 node, 14 transmission line test system [28]. The results of the tests are presented in the following sections.

5.1 Supercomputer Algorithm

The program is written in Fortran 77 and is run of a CRAY Y-MP2/216 in four different codes: a) scalar code (uses only one processor without vector hardware), b) scalar/parallel code (uses both processors without vector hardware), c) vector code (uses only one processor with vector hardware), d) vector/parallel code (uses both processors with vector hardware).

The size of the search area of the current period is kept at its maximum value \(Y = 2^{26} = 67,108,864\) so that the optimal solution for that period is reached. The number of
feasible states, however, is only a fraction of the total search area (it is found to average nearly 6% in this particular system) and minimal effort is spent eliminating the infeasible states. The window size is varied from $X = 1$ to $X = 7$ to determine the change in quality of the solution and the affect of varying the window size on the CPU time. The computation times are summarized in 5.1. When compared to the performance of the scalar code, the scalar/parallel code, vector code and vector/parallel code resulted in nearly 50%, 94% and 97% reduction in CPU time.

To demonstrate its effectiveness, the execution time of the modified program has been compared to that of the conventional program. It has been found that the scalar and scalar/parallel codes required nearly the same amount of CPU time for both algorithms. This implies that there is very little addition to the number of operations performed by the modified programs. But in the vector and vector/parallel codes, the modified program outperformed the conventional program by a factor of 3 as shown in Figure 5.2. The CPU time reduction is primarily due to the very large increase in vector length (from 26 to $2^{26}$).

The quality of the solution is found to improve very little with increase in window size, as illustrated in Figure 5.3. It is interesting to note that there is only a 0.2% decrease in the fuel cost as the window size is increased form 1 to 7, at the expense of 7 times more computation time.

For comparison purposes, the problem was also solved using classical priority list method. This method required only 0.1 seconds of CPU time but the optimum cost of production is found to be $20,000 higher than that obtained by the proposed algorithm when using a window size of $X=1$.

The proposed algorithm was also run on a Sun Sparcstation 2 to determine the relative
Figure 5.1: CPU Time of Proposed Algorithm for different Computer Codes and Window Sizes

Figure 5.2: CPU Times of Vector/Parallel Code for Conventional and Modified Algorithms
magnitude of the computation time required by such a high performance workstation. It is found that the program ran 115 times slower on the Sun Sparcstation 2. However, this can be a misleading figure because the CRAY and the Sun Sparcstation 2 are two different machines. The price/performance ratio of the supercomputer when compared to that of a serial machine cannot justify the purchase of a supercomputer for solving only the problem under consideration.

5.2 Genetic Algorithm

The genetic algorithm is tested on two systems. System A, a 26-thermal unit power system with a 24-hour forecasted load, and System B, a 44-thermal unit power system with a 24-hour forecasted load. The programs were written in C language, (because of its better random number generator functions), and run on a Sparcstation 1.
There are five variables that need to be preset before compiling the program: (i) the size of population in each generation, which is varied from 100 chromosomes to 500 chromosomes in steps of 40, (ii) the number of generations (number of iterations) in each period, which is kept constant at 10 in all the simulations, (iii) the crossover point in the reproduction technique, which is also kept constant through all the simulations, (iv) the number of best chromosomes that need to be copied to the next generation, which is set at 1 all through the simulations. (v) the number of states that are saved and used to obtain the optimal solution in the next period. This variable is varied from 1 to 7 for both systems, A and B.

Penalty factors are imposed on the chromosomes that do not meet the constraints. A very large value of $10,000,000.00 is assigned to the fuel cost for all the chromosomes that do not satisfy the constraints. The fuel cost for such chromosomes is not calculated. The assigning of a large value for the fuel cost gives the chromosome a very small fitness value thereby, making its chances of reproduction negligible.

To compare the performance of the GA in terms of speed and quality of solution, the scheduling was also carried out using the dynamic programming algorithm. Recall that, in the dynamic programming method, there are two variables that need to be pre-specified. One, X, which is the number of states saved in each period, and two, Y, which is the search space in each period. In both the examples considered in this paper, the variable X (fifth variable in GA) is varied from 1 to 7. The variable Y, is set as the complete search space in that period.

Figures 5.4 and 5.6 compare the variation of the fuel cost with the variation in the number of states saved in each period for the genetic algorithm and the dynamic algorithm.
for systems A and B, respectively, while, Figures 5.5 and 5.7 compare the variation of the computation time with the variation in the number of states saved in each period for the genetic algorithm and the dynamic programming algorithm for systems A and B, respectively.

The reason why the GA performs better than the dynamic programming algorithm in terms of computational speed is that the GA does not have to search the entire state space for the optimal solution. The reason why there is such a large variation in the solution quality with the variation in saved states is that the GA does not obtain the same solution as the dynamic programming solution for each period. It must be mentioned here that the improvement in the quality of solution for other systems might not resemble those obtained for the example systems considered in this paper, but, the decrease in the computation time can be obtained in similar scale and magnitude.
Figure 5.5: Computation time for System A

Figure 5.6: Fuel Cost for System B
5.3 UCP including Transmission Losses

A 22 generating unit, 10 node, 14 transmission line power system is used as a test example. The load demand is forecasted over a period of one week (168 study intervals). The power system data and load data are listed in the Nomenclature.

The program was run on CRAY with vectorization and on a CONVEX for comparison purposes. The amount of computational time taken by the program is listed in Table I. It is noted that the time taken by the CRAY to execute the program is about 17.6% of the time taken by the CONVEX. The schedule was found for a study period of 168 hours (1 week). If the study period is reduced to the normally used 24 hours, then the time taken for execution of the program would reduce to 4300.16 secs (1.194 hours) on a single processor and to approximately 36 minutes using 2 cpus. The solution time can be further improved by using sparse matrix methods for inverting the jacobian matrix. It was observed that the
problem was not solvable on a Sun sparcsation because of the computational limits on the system.

Table I. CPU time for UCP with Transmission Losses.

<table>
<thead>
<tr>
<th>Vector Code on CRAY (min)</th>
<th>Vector Code on CONVEX (min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>501.68</td>
<td>2843.73</td>
</tr>
</tbody>
</table>
Chapter 6

CONCLUSIONS

The thesis presented two algorithms for solving the UCP in electric power systems. The first method utilizes the vector and parallel processing capabilities in supercomputers. The algorithm is based on dynamic programming, but is modified to take advantage of the vector and parallel processing capabilities of the supercomputers. Simulation results indicate a significant reduction in CPU time when compared to the conventional method. The second method adapts the genetic algorithm technique to solve the UCP on workstations or fast personal computers. The algorithm is shown to be very efficient. It provides a near optimal solution, while considerably reducing the computation time.

The thesis also investigates the effect of including transmission losses in the UCP as constraints. It is observed that the time requirements for solving the UCP with transmission losses are very high. The proposed algorithms hold a great deal of promise and could result in providing better results and therefore, more savings for the electric power utilities.
Bibliography


