Fall 1991

Coordinated strategic planning of hydroelectric energy usage using Stochastic Dynamic Programming with decoupled production cost simulation

Duane David Highley

Department:

COORDINATED STRATEGIC PLANNING OF HYDROELECTRIC ENERGY USAGE USING STOCHASTIC DYNAMIC PROGRAMMING WITH DECOUPLED PRODUCTION COST SIMULATION

by

DUANE DAVID HIGHLEY, 1961-

A THESIS

Presented to the Faculty of the Graduate School of the

UNIVERSITY OF MISSOURI-ROLLA

In Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE IN ENGINEERING MANAGEMENT

1991

Approved by

Y. Bill Omurtag, Advisor

Stephen A. Raper, Co-Advisor

John B. Kincaid

ABSTRACT

The method of Stochastic Dynamic Programming is applied to the problem of Long-term hydrothermal coordination for a five reservoir system in the midwest. The model that has been developed will optimize the monthly reservoir elevations to produce the lowest expected energy production cost. The production cost calculations are decoupled from the Dynamic Programming loop, resulting in shorter execution times. Constraints are included for maximum and minimum reservoir elevations, maximum monthly reservoir rise and drop, and maximum and minimum monthly energy production. These constraints help to reduce the state space and shorten execution time. An additional constraint is modeled to control the reliability of the reservoir management schedule that is produced. Schedules with higher reliability also have higher costs.

The model is demonstrated on a typical planning problem: to optimize the lake elevations over the fall, winter, and spring. The stochastic dynamic programming model reduced the expected energy production cost by $680,000 over this period, as compared to the rule curve based methods currently in use.

This research shows that, with the computing power now available, the direct stochastic solution for a five-reservoir system is now feasible. Execution times were under two hours on a 486 computer running at 33 MHz.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

PREFACE

This model did not just create itself. I have many people to thank for making my job of developing it easier. I am thankful for the support given by Dr. Bill Omurtag and Dr. Stephen Raper of UMR. They have made my graduate program an almost pleasant experience. At Associated Electric Cooperative, David Dockery and James Brake were very helpful when it came time to express my ideas as Pascal code. Finally I must thank Jim McNabb, Director of Engineering at Associated Electric, for challenging me to find an answer to this problem.

I. INTRODUCTION

A. BACKGROUND

As a wholesale electric power supplier, Associated Electric Cooperative, Inc. provides the total energy requirements for 40 retail distribution cooperatives across Missouri and three in southern Iowa. To supply this electrical load, the cooperative owns and operates over 2400 MW of coal-fired generating capacity, and over 50 MW of oil-fired peaking capacity. This fossil-fueled generation is sometimes called "thermal" generation.

Supplementing this thermal generation, the cooperative also has operational control of over 750 MW of hydroelectric generation, which is available through the Southwestern Power Administration (SWPA). SWPA is a federal power marketing agency under the United States Department of Energy. SWPA has the responsibility of marketing the federal hydroelectric capacity to preference customers in the Midwest and Southwest United States.

Through its high-voltage transmission grid, the cooperative also has access to large amounts of capacity and energy from 17 neighboring utilities. Through these interconnections, the cooperative can purchase and sell bulk energy at market-based prices. The revenues from these sales and purchases will reduce the net energy cost to the cooperative's customers.

Figure 1 illustrates the cooperative's service area and the location of its primary generating facilities. As shown in the figure, the hydroelectric capacity is comprised of five separate projects. Four of these projects possess large amounts of storage capacity in their reservoirs, allowing their hydraulic inflow to be held back for later use.

By storing the water, it can be saved for use at times when energy has its highest value.

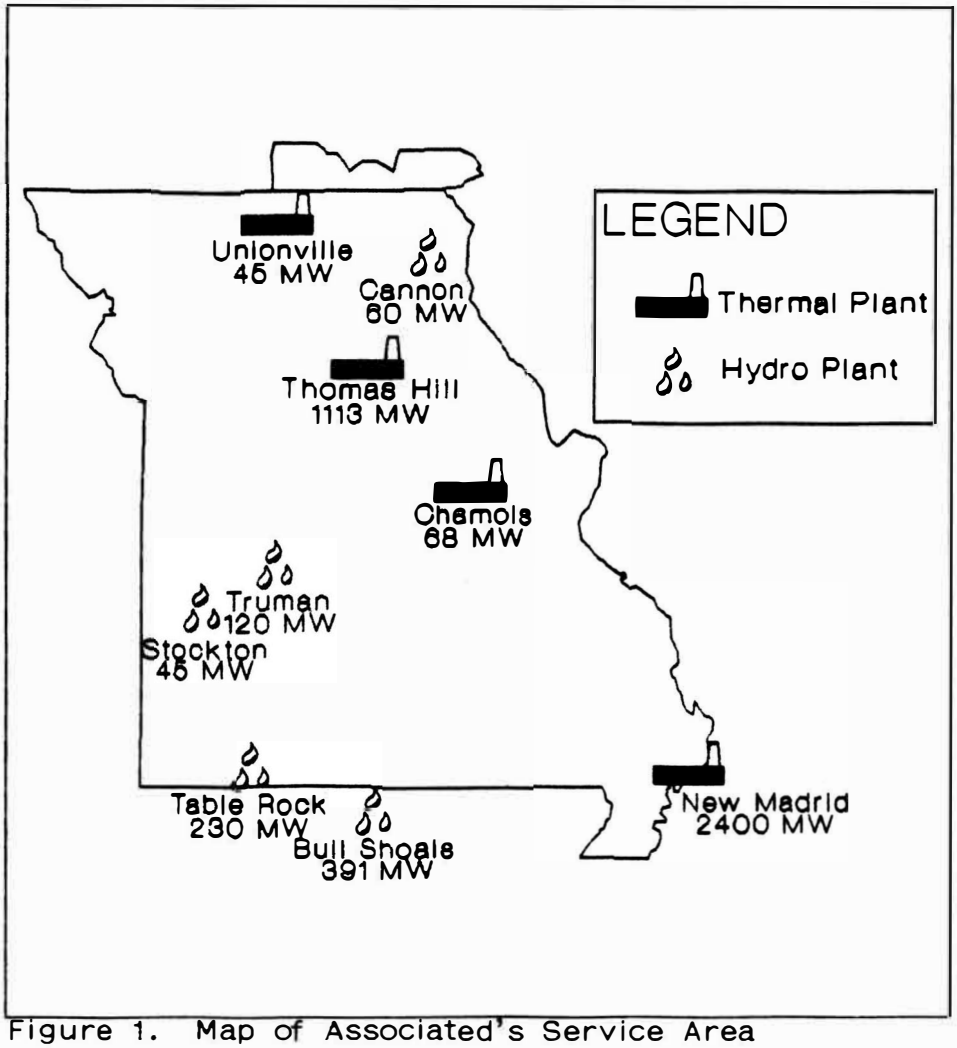

Figure 1. Map of Associated's Service Area

Some projects are connected to each other in a cascade fashion, as Figure 2 shows. Releases from the upstream projects will influence the storage and elevation at downstream projects. This complicates the coordination of water usage at the five projects.

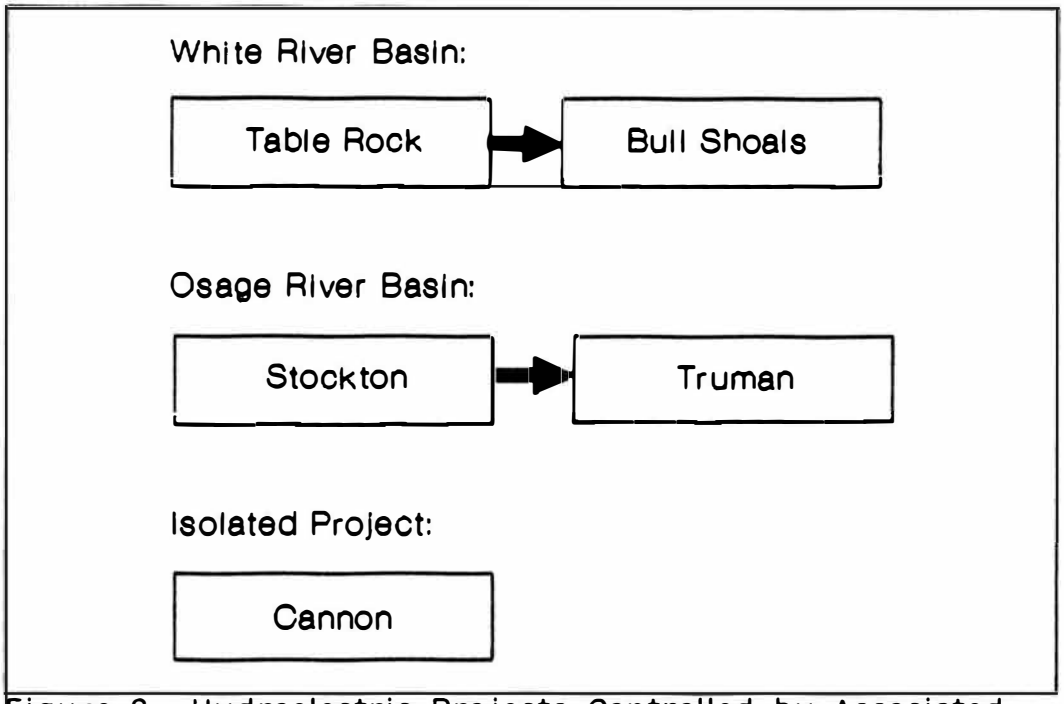

Figure 2. Hydroelectric Projects Controlled by Associated

The cooperative currently pays SWPA only 4.2 mills/kWh for the hydroelectric energy received, which is inexpensive when compared to the cost of energy from thermally-fired generating plants. The "baseload" thermally-fired generating units have marginal energy production costs that vary from 10 mills per kWh to over 18 mills/kWh. These marginal costs are almost entirely due to the cost of fuel, usually coal. The oil-fired "peaking" units have marginal energy production costs in excess of 75 mills/kWh, and are therefore used as infrequently as possible.

The cost for energy that can be purchased from neighboring utilities varies constantly as energy market conditions change. The costs change because of changing weather patterns and generating equipment availability. Energy costs are higher when the weather is

extreme, either hot or cold. Costs also increase when "baseload" generating units become unavailable due to unexpected equipment failures.

The cooperative wishes to maximize the benefit that could be received from the inexpensive, but limited, hydroelectric energy. This can be accomplished by using hydroelectric energy to displace the most expensive purchases and thermally generated energy. The value of the energy that can be displaced by hydroelectric energy changes throughout the year, making the coordination process more difficult. Since the supply of water is limited, the question becomes: when should it be used for generation? And when should the water be held back for future use? The answers can be found by approaching the problem as a specialized inventory management problem, where the inventory to be managed is the supply of water held in each lake.

B. CONSTRAINTS IN HYDROELECTRIC ENERGY USAGE

If the storage volume of each hydroelectric project was unlimited, the optimal coordination of the hydro energy with thermal energy would be simple. However, there are several constraints that limit the cooperative's flexibility in scheduling the use of hydroelectric energy.

Constraints in Power Pool Size. The usable storage volume (power pool) at each reservoir is limited in size, limiting the amount of water that can be saved for future use. Because of the limited power pool size, risk is present. There is a risk that large unexpected inflow will cause the lake to be filled and water spilled over the top of the dam. Water which is spilled is wasted, since it cannot be used to generate electrical energy. There is another risk that unexpectedly small inflow

will not replenish the water that is being used, resulting in an empty power pool. This means that hydroelectric energy may not be available at the times when it would have the greatest value.

Variations in Hydraulic Conversion Efficiency. The efficiency of each hydro project varies with the lake's elevation. More energy is produced per unit volume of water as the reservoir elevation (and storage volume) is increased.

Limitations in Maximum and Minimum Water Usage. The maximum amount of water that can be used at each project in a month is limited by the U. S. Army Corps of Engineers. A maximum reservoir draw down of 4.5 feet per month is specified. The minimum usage is also determined by the Corps to provide "fish water," which is a minimally acceptable stream flow to provide a proper aquatic habitat downstream from the project.

Additional Seasonal Restrictions. At some projects, seasonal restrictions are placed on the maximum usage and maximum power pool elevation to mitigate other adverse environmental consequences of hydroelectric generation, such as low dissolved oxygen levels downstream.

C. THE RANDOM NATURE OF HYDRAULIC INFLOW

Besides these constraints and restrictions on the usage of hydroelectric energy, the optimal coordination with thermal generation is made even more difficult by the unpredictable nature of hydraulic inflow. The inflow varies with weather conditions (rainfall and runoff) and can be approximated by a random variable.

The statistical distribution of future inflow into a particular lake can be estimated from historical stream flow readings that are maintained by the U. S. Army Corps of Engineers. The Corps has monthly inflow readings for lakes in the Missouri-Arkansas region since 1928. Figure 3 graphically shows the historical monthly inflow into Stockton lake from 1928 to 1987 (60 years). The inflow patttern for the other lakes is similar.

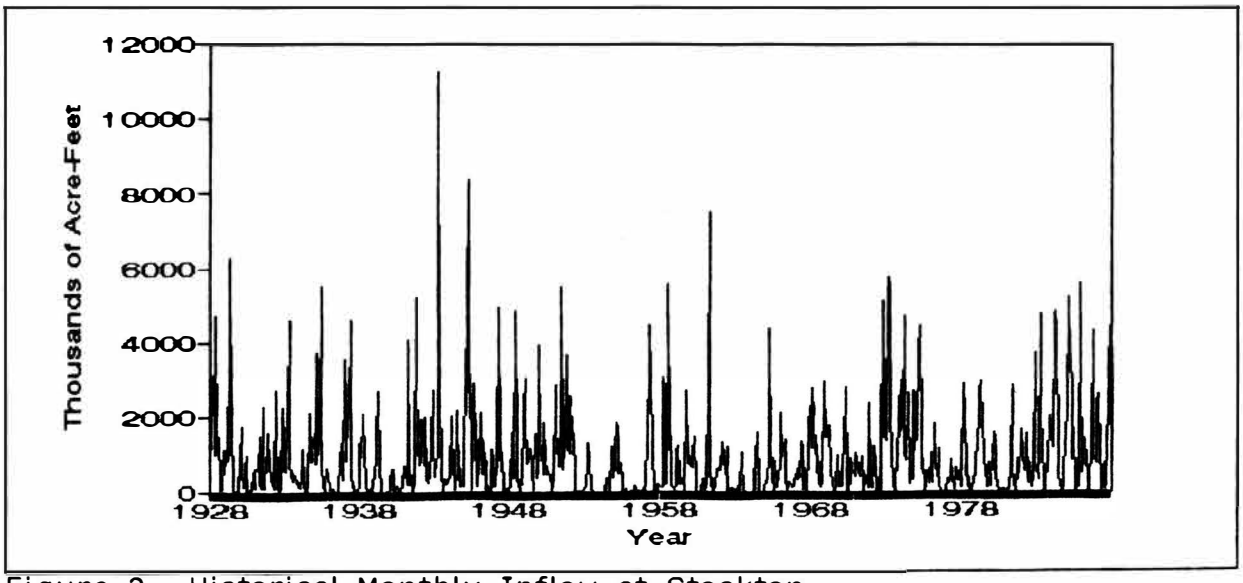

Figure 3. Historical Monthly Inflow at Stockton

The graph shows that there has been a tremendous variation in hydraulic inflow from month to month. Because the variance of inflow is so large for these lakes, it must be given full consideration in any optimization scheme.

D. THE PROBLEM STATEMENT

Given all the constraints and uncertainties associated with the limited, but valuable hydroelectric energy, the problem statement is: what should the optimal monthly elevation target be for each lake, to provide the lowest system wide energy production cost, recognizing that the inflow of water is uncertain? If the lakes are operated too low, there is a risk of insufficient hydroelectric energy being available to displace the most expensive purchased and thermal energy. Conversely, if the lakes are operated too high, the risk increases that water will be spilled, or used at times when it has a comparatively low value.

The cooperative does not currently have a method to coordinate the hydro and thermal energy while recognizing these constraints. Instead, it relies upon a heuristic method that is based on historical inflow statistics. The weakness of this method is that there is no direct consideration of the avoided thermal generating costs in the planning process. It is based solely on reliability, without regard for costs.

Currently, the cooperative produces a total of 14 million MWh of electrical energy per year. In an average-inflow year, the hydroelectric capacity will provide over 1.5 million MWh of the total energy requirements of the cooperative. If only 1% of the hydro energy could be used more economically, by displacing 18 mill/kWh energy instead of 10 mill/kWh energy, the annual savings would be: (1.5 million MWh) x (.01) x (18 $/MWh - 10 $/MWh) = $ 120,000.

Given the magnitude of these potential savings, a considerable research effort, directed toward finding a means of solving this hydrothermal coordination problem, can be easily justified.

E. THE OBJECTIVE OF THIS WORK

In this thesis a computer model will be created. This model will consider the performance and interrelationship among the five hydroelectric projects available to Associated Electric Cooperative. The randomly variable inflow and the time-varying electrical energy production costs will also be used by the model. As an output, the model will provide the optimal monthly target lake elevations. These elevations will be optimized to yield the lowest expected energy production costs over the optimization period.

II. LITERATURE REVIEW

Many techniques have been used in an attempt to solve the long-term hydrothermal coordination problem. Wood and Wollenberg (1984) devote an entire chapter of their text to the subject. Their chapter is a good primer in hydrothermal coordination, starting off with the basics and covering the three main methods currently in use: the method of lagrange multipliers, gradient search techniques, and dynamic programming.

They emphasize that all hydrothermal electric power systems are unique due to the unique character of each hydroelectric reservoir and power plant. For this reason, no one approach is appropriate for solving every coordination problem. They discuss the three primary methods and direct the reader to perform further research.

For a more detailed study of hydrothermal coordination, the reader should begin with the works of Yakowitz (1982) and Yeh (1985). Yakowitz traces the development of dynamic programming as it has been applied to many types of water resource problems, including long-term hydrothermal coordination. Yeh considered several different optimization techniques that have been applied to various water resource operations and management problems, including the dynamic programming technique.

For specific applications of optimization techniques to electrical coordination, many papers will be found in the IEEE Transactions on Power Systems (formerly IEEE Transactions on Power Apparatus and Systems). By reviewing these papers, one can see how the optimization models that have been used have grown more complex and exact over time, as more computing power has become available.

A survey of the literature reveals several techniques that have been used for long-term hydrothermal coordination: calculus-based methods, lagrange multipliers, gradient search techniques, linear programming, integer programming, network flow algorithms, progressive optimality algorithms, and dynamic programming. In addition, there are two techniques that show promise but have not yet been applied to hydrothermal coordination: simulated annealing and genetic algorithms. All these techniques will be reviewed in the following sections.

A. CALCULUS BASED METHODS

In calculus-based methods, the objective and constraint equations are written as continuous, differentiable functions. These equations are manipulated and partial differentiation is performed on the cost function. The minimum of the cost function is found by solving for the result when the derivative is zero.

Two examples of applications of this technique were found, both of them by El-Hawary and Christensen (in 1972 and 1973). Because of the limitations of their technique, it can only be useful for very simple and unrealistic power systems.

Because calculus is used, all of the constraints and the objective function must be expressed as continuous and differentiable functions. Real-world constraints are not always continuous and differentiable, so the true constraints must be approximated or simplified. By approximating these values, error is introduced into the optimization process.

Because of the mathematical complexity of their method, they have chosen to limit themselves to only linear constraint equations. The

assumption of linearity does not reflect the true performance of most hydroelectric projects, which are non-linear in nature. Also, the costs associated with the thermal generating plants are very non-linear. If non-linear functions had been used, the mathematics would become unwieldy.

The final problem with their method is that it cannot include a random variable. The hydraulic inflows are assumed to be deterministic, that is, they are known with certainty in advance. In some areas of the world, hydraulic inflow is provided by melting snowpack. This type of inflow can be forecasted with great precision. Under these circumstances, the assumption of deterministic inflows is reasonable, but in the midwest, where inflow is provided entirely by rainfall and runoff, this is a very poor assumption.

B. METHODS USING LAGRANGE MULTIPLIERS

In the method of lagrange multipliers (also know as lagrange relaxation) an iterative technique is used to find the optimal hydrothermal coordination schedule. The hydroelectric energy is given a fictitious value (known as the lagrange multiplier) that is initialized to a constant. By varying the multiplier, the usage of hydroelectric energy is varied. If too much energy is used and a constraint is violated, the multiplier is increased. This will create a higher apparent value, which serves to reduce the usage of the resource. The multipliers are varied in an iterative fashion until all the available hydroelectric energy is used and no constraints are violated.

As the solution proceeds, the cost decreases with each iteration. When the total cost reaches a limit or is decreasing at a very slow pace, the iterations cease and the optimal solution is said to be found.

Several researchers have applied this technique to long-term hydrothermal coordination. Bonaert et al. (1972), Wan and Larson (1984), Duncan et al. (1985), Andersson et al. (1986), and Lyra et al. (1990) have all demonstrated the use of lagrange multipliers in the optimization of a single reservoir system with deterministic inflows.

There are many problems with the lagrange multiplier method. Because it is an iterative technique, it can be very slow to converge, if it converges at all. Making the proper initial guess for the multipliers is important, to get the method started moving in the right direction toward the minimum cost. If the method starts wrong, it may not converge, or worse yet, it may converge to a point that is not the lowest cost. This is known as being trapped in a local minima of the cost function.

Related to the problems of convergence is the problem of execution time. It may take a very long time for a computer to find the correct minimum cost solution. With the increasing power of computers, this is now becoming less of a problem.

This method can be very difficult to apply to the simultaneous coordination of multiple hydroelectric projects. Soares et al. (1980) demonstrated the method on a multi-project cascaded hydro system with deterministic inflow. It can be done, but the use of lagrange multipliers with multiple projects only exacerbates the convergence and execution time concerns discussed above.

Another limitation of the lagrange multiplier method is its complete inability to optimize with a random variable. Lyra et al. (1984) claim to have included a stochastic (random) variable for inflow in a lagrange solution, but in reality they did not. In actuality, they found the average inflow for each monthly period before performing the lagrangian solution. Therefore, their optimization was done using a deterministic variable for inflow rather than a stochastic one.

C. GRADIENT SEARCH TECHNIQUES

Besides dynamic programming, gradient search techniques are the most frequently applied to hydrothermal coordination. The gradient technique is a search technique that approximates the direct solution that can be found using calculus. In order to apply this method, the constraint equations must be continuous and differentiable. This requirement makes it difficult to apply gradient techniques to realistic problems.

The gradient method works by starting from an initial feasible solution and moving in the steepest direction toward the minimum. The steepest direction is found by taking the partial derivative of each variable. In an iterative approach, the gradient technique moves in small steps toward the minimum, stopping when no further improvement can be found or after a predetermined number of iterations.

Several researchers have demonstrated successful gradient solutions to the hydrothermal problem: Saha and Khaparde (1978), Diaz and Fontane (1989), Tejada-Guibert et al. (1990), and Tong and Shahidehpour (February 1990). Diaz and Fontane observed that the gradient method was superior to the linear programming method in

solving their problem for a system of lakes in Argentina. Tejada-Guibert et al. used a commercially available gradient optimization package (MINOS) to solve their problem, which illustrates the maturity of this optimization technique.

One problem with the gradient technique is its tendency toward getting stuck in local minima. Several researchers have proposed heuristic solutions, known as "branch and bound" techniques, to this problem. By using heuristic rules, the algorithm is directed away from local minima and (hopefully) toward the global minimum. This method has been demonstrated by Carvalho and Soares (1987), Carneiro et al. (1990), and Tong and Shahidehpour (May 1990). While this is an improvement over straight gradient methods, it still does not guarantee optimality in the solution.

A major weakness of the gradient technique is its difficulty in handling realistic constraints. Besides the differentiability requirement, Wood and Wollenberg observe that the gradient technique suffers as more constraints are added. They also say the gradient technique is slow to converge when multiple hydroelectric plants are coordinated.

Another limitation of this method is its complete inability to optimize with a stochastic variable. Although Christensen and Soliman claim to have developed a "stochastic" gradient based technique, in reality it is deterministic. They have simply substituted the "average" expected value of hydraulic inflow for each month in a traditional gradient structure.

D. LINEAR PROGRAMMING

Linear programming (also known as the simplex method) is a well understood optimization technique. It was applied to short-term hydrothermal coordination by Pereira and Pinto (1982). The primary limitation of this method is its restriction to only linear (or piecewise linear) constraints and objective functions. Also, it cannot easily handle multiple projects or stochastic variables.

Pereira and Pinto recommend the use of stochastic dynamic programming for the solution of the long-term hydrothermal coordination problem.

E. INTEGER PROGRAMMING

Integer Programming has seen much application in operations research. Pereira and Pinto (1983) attempted to apply it to the mid-term hydrothermal coordination problem. This method suffers from many of the same limitations of linear programming: it is difficult to handle multiple projects, and inflows must be deterministic (not stochastic). In addition, it does not allow real-valued constraints. They must be expressed in integer form.

Recently, Trezos (1991) reported on the successful application of integer programming to a system with multiple cascaded hydroelectric projects. This demonstrates that multiple projects can now be optimized with this technique, although no one has yet demonstrated the use of random inflow in an integer programming scheme.

F. NETWORK FLOW ALGORITHMS

Network flow algorithms are related to integer programming, but without the restriction to integer-valued constraints. The solution that is found is not guaranteed to be optimal, since the algorithm can become stuck in local minima.

Sjelvgran et al. (1983) demonstrated the use of a network flow model in the solution of a multi-project hydroelectric power system. In their paper they remark that the network flow approach is superior to integer programming. The primary weakness of this approach is its inability to include a random inflow variable.

G. PROGRESSIVE OPTIMALITY ALGORITHMS

The progressive optimality algorithm is also know as the "greedy" algorithm. It is a predecessor to simulated annealing. The method is described by Barnard and Skillcorn (1988) in their text on Pascal programming. In progressive optimality, a random change is made to a variable. If the change results in a lower cost, it is kept. Otherwise, another random change is made and the results are checked again. This process continues until no further improvement can be found. The method is computationally very efficient in terms of memory and CPU usage, but it may never converge. Even if it does converge to a minimum cost solution, the results are likely to be a local minimum rather than the global minimum that was sought.

Nanda and Bijwe (1981) applied this method to hydrothermal coordination with cascaded plants and many realistic constraints. It could even be used to solve a system with stochastic inflows. But the

convergence and sub-optimal solution problems of the progressive optimality algorithm outweigh these advantages.

H. SIMULATED ANNEALING

Simulated annealing is a technique similar to the progressive optimality algorithm, but it is more likely to converge and is more successful at reaching the global minimum. By using an exponential "cooling schedule" based on Newton's law of cooling, the algorithm will allow some random changes in variables even if they cause an increase in cost. This is done to help bump the solution process away from local minima and (hopefully) toward the global minimum.

The simulated annealing method will allow for the inclusion of multiple hydroelectric projects with many non-linear constraints. Stochastic variables could even be included for inflow. The greatest weakness of the method is the need for the researcher to provide the proper cooling schedule.

If the algorithm is cooled too fast, there is a high probability that the solution that is found will be sub-optimal. If the algorithm is cooled too slowly, the execution time grows and solutions will be long in coming. There is currently no know way to determine the proper cooling schedule in advance, and the best schedule will vary with each new problem. It is usually necessary to compare the solution found with simulated annealing to one from another method to confirm that the cooling schedule is appropriate.

While no one has yet applied this technique to hydrothermal coordination, Zhuang and Galiana (1990) demonstrated its use in the related problem of thermal generating unit commitment. There is

currently much interest in this technique and its applications will certainly grow.

I. GENETIC ALGORITHMS

The genetic algorithm is somewhat related to simulated annealing. It is a search technique that mimics the process of natural selection in genetics. Like simulated annealing, it involves a random process for selecting variables. But unlike simulated annealing, there is no need to specify a cooling schedule. The algorithm directs itself toward the optimal solution by learning from the history of its prior decisions, just as genes provide information about the parentage of the offspring. By combining the best elements of two good parent solutions, the offspring solution is hoped to be as good or even better.

No one has yet applied genetic algorithms to the hydrothermal coordination problem, but Goldberg (1989) has written a text explaining the method in detail. Genetic algorithms are claimed to allow non-linear constraints, multiple objective variables, and stochastic variables while producing near-optimal results. There is no guarantee that the optimal solution will be found, however.

J. DYNAMIC PROGRAMMING

Dynamic programming is a widely applied optimization technique. It is a directed search technique that allows multiple objective variables and multiple constraints. In fact, as more constraints are added, the efficiency of the algorithm actually improves. Dynamic programming (DP) can even optimize with stochastic variables, resulting in the technique known as Stochastic Dynamic Programming (SDP). The biggest benefit of

DP is that it is not distracted by local minima. If the algorithm solves, the solution is guaranteed to be optimal.

Dynamic programming relies on the researcher's ability to break the problem up into a series of independent, sequential decisions. If this can be done, then DP can be applied. In the hydrothermal coordination problem, the state variable is the elevation of each reservoir. This state variable is optimized sequentially, starting in the first month and moving forward in time.

The biggest limitation of DP is the so called "curse of dimensionality." As the number of state variables increases, the efficiency of the DP solution technique deteriorates. As the dimension of the problem increases, the execution times and computer memory requirements increase. Yakowitz has stated that DP with deterministic inflow cannot be used for systems with more than three lakes because of the increasing dimensionality. Similarly, Yeh has said that Stochastic DP cannot be applied to a system with more than one lake, due to the higher dimensionality with stochastic variables.

DP has been applied by many researchers in recent years. Deterministic solutions have been demonstrated for systems with one reservoir by Arvanitidis and Rosing (1970). At that time, computing power would not allow them to consider systems with multiple reservoirs. The dimensionality of systems with multiple reservoirs was reduced by combining the reservoirs into a single composite model. This allowed the researchers to make general recommendations about the amount of water to keep in storage each month, although they could not determine how the water should be allocated among the lakes.

Other researchers have explored different techniques for reducing the dimensionality of their problems. Bonaert et al. (1972) applied the technique of successive approximation to deterministic DP. More recently, Chang et al. (1990) has demonstrated a similar method. In successive approximation, a multi-plant system is solved for one reservoir at a time using DP. This process repeats, with a single reservoir optimized as the others are held fixed. Over time, the process should converge. The problem with this approach is that sometimes it does not converge, and if a solution is found it may be sub-optimal.

Bean et al. (1987) experimented with a technique for aggregating the states in the solution process. While this will accelerate the solution time, it also may lead to sub-optimal solutions.

Other researchers have attempted to combine the DP approach with other optimization methods. Foufoula et al. (1988) combined DP with a gradient search to reduce the execution time. Unfortunately, this method also introduces the weaknesses of gradient methods into the solution process. Li and Zhou (1990) combined successive approximation DP with a lagrange multiplier approach. Again, the limitations of the lagrange multipliers are introduced into the problem. Finally, Bannister and Kaye (1991) combined DP with linear programming. This limits the constraints to be linear or piecewise linear functions, and has not been demonstrated for systems of multiple projects. None of these combination techniques are guaranteed to converge and none of them guarantee global optimality.

A major shortcoming of every DP example discussed so far is that they have not used a random variable to represent hydraulic inflow. Many researchers claim to have used stochastic dynamic programming in

their coordination problems, but often they are simply using deterministic DP with the average expected value of inflow. This has been done by Quintana and Chikhani (1981), Neto et al. (1985), and Contaxis and Kavatza (1990). By using the expected value of inflow within the DP loop, one is optimizing with the average inflow rather than optimizing with the uncertain inflow. For lakes in the midwest, where inflow is very uncertain, it is crucial to consider the impact of stochastic inflow on the operation of hydroelectric reservoirs.

A few researchers have demonstrated truly stochastic dynamic programming solutions to the hydrothermal coordination problem. Viramontes and Hamilton (1978) performed stochastic DP with a single reservoir system. The hydraulic inflow was represented as a random variable in the DP loop. To improve the performance of the method, the production cost of the thermal system was done ahead of time outside the DP loop. This pre-computed production cost information was saved and made available to the DP optimization, resulting in greatly improved performance. By decoupling the production costing problem from the DP reservoir problem, a true stochastic problem of higher dimension could be solved.

The stochastic solution to a single reservoir was also performed by Weiner and Ben-Zvi (1982). Their method added another state variable to represent the prior month's inflow, which in strongly correlated in the Nile River. Because midwestern lakes have a very weak monthly inflow correlation, this additional effort would be unnecessary.

As computing power grew, so did the applications of SDP. Sherkat et al. (1985) performed SDP with a two-reservoir system. To

avoid the curse of dimensionality, the two reservoirs were optimized using the successive approximation technique described earlier. The authors comment that, as of 1985, the true stochastic DP solution to a system of more than three lakes is "computationally infeasible."

This comment is echoed in the more recent work of Hernandez et al. (1991). The authors have used SDP to optimize a system with two aggregated lakes, and say the straightforward application of SDP is "computationally infeasible for systems with more than three reservoirs because of dimensionality." For this reason they advocate the use of successive approximation with lake aggregation, though these techniques may not converge, or may converge to local minima, and do not provide for the detailed modeling of individual lake constraints.

Because the hydroelectric plants in Missouri have uncertain inflow, with uniquely varying, non-linear constraints, the direct application of stochastic dynamic programming appears to hold the greatest promise for success. For the Associated Electric Cooperative system, it is desirable to coordinate five different reservoirs simultaneously. No researcher has yet created a model that will meet these requirements and produce a reliably optimal solution within a reasonable computing time.

III. MODEL DEVELOPMENT

Of the methods discussed in the current literature, stochastic dynamic programming (SDP) offers the greatest promise of finding an optimal solution to the long-term hydrothermal coordination problem. I have decided to use this method to find a solution to the five-reservoir hydrothermal coordination problem.

The general dynamic programming method is an approach to problem solving, rather than a specific step-by-step process. As such, each DP algorithm must be custom tailored to the specific problem at hand. DP methods are particularly well suited for solving multi-period or multistage problems involving sequenced decisions. Examples of these problems include the "shortest path" traveling salesman problem, "knapsack" or packing problems, production and inventory management problems, and resource allocation problems. The hydrothermal coordination problem can be considered as a specialized inventory management problem, where the "inventory" being managed is the water in each storage reservoir, and the future delivery of water is uncertain.

The Stochastic Dynamic Programming method (SDP) was chosen for several reasons. It will allow many constraints to be modeled realistically, as non-linear functions with integer or real values. This guarantees that the optimal solutions that are produced will be realistically achievable. Adding constraints to a DP problem actually improves the solution process by restricting the state space and shortening the solution time.

Another advantage of DP is the ability to model the hydraulic inflow as a stochastic variable. Using this information, the SDP method

will produce the globally optimal solution for all lakes simultaneously coordinated with one another.

The main disadvantage of the SDP technique is its execution time, which can be long when many lakes and stochastic variables are considered. In this method the production cost analysis has been decoupled from the SDP optimization loop, using the technique of Viramontes and Hamilton (1978). By decoupling the detailed production cost simulation process, computational efficiency has been greatly improved. This allows a very detailed analysis and simulation of production costs to be performed before the optimization is started, without sacrificing computing time during the optimization.

The pre-computed cost information is brought into the SDP algorithm and can be used quickly and repeatedly without sacrificing detail in the production cost simulation. This is an advantage over the simplified and unrealistic methods of production costing used by some researchers. By decoupling the production cost analysis, the benefits of detailed cost modeling can be combined with a high speed SDP optimization.

A. APPLICATION OF STOCHASTIC DYNAMIC PROGRAMMING TO HYDROTHERMAL OPTIMIZATION

In order to apply dynamic programming, the problem must first be broken down into "stages." In solving sequential problems involving time, the stages may be represented by days, weeks, or months.

Within each stage, the problem is further divided into "states." A state describes a decision variable in the problem, for instance, each state may represent a lake's current elevation, in discrete one-foot

increments. Here, the lake's level is the state variable that the DP model is being asked to optimize, subject to the defined constraints.

In a multi-lake optimization, the state variable will actually be a state vector: an ordered set of lake levels, one for each lake.

The output from the DP model will be the optimal "policy." A policy is the set of states in each stage. In the hydrothermal coordination problem, the optimal policy is the set of states (monthly lake levels) which result in the lowest expected production cost.

Essential to the theory of the DP method is the "Theorem of Optimality," which states that an optimal policy must contain only optimal subpolicies.

This theorem is based on the "Principle of Optimality," which can be stated as follows: if a given state is part of the optimal path, then the optimal path from the first state to the given state is also part of the optimal path. The impact of this theorem is that a large sequential decision problem can be broken down into several small decision problems. Dynamic programming works by solving a large problem one stage at a time. The optimal policy is found by adding together all the optimal subpolicies.

B. EQUATIONS USED IN THE SDP ALGORITHM

The mathematical form of the DP approach can be expressed as a recursive relationship in the following form.

$$f_i(v_i) = \min_{v \in V} \sum_{x_i} p(x_i)[C(v_i, i, v_{i-1}) + f_{i-1}(v_{i-1})]$$

where:

f_i = minimum expected cost from beginning stage to stage i

v_i = end of month lake level in stage (month) i

V = set of all lake levels (states), may be state vectors

X = set of all points in the inflow probability density function

x_i = an inflow estimate from the p. d. f. for month (stage) i

p = the probability of occurrence of a point on the inflow p. d. f.

C = the cost function, the production cost for given volumes in month i

This general DP equation includes a random variable, the hydraulic inflow x. The stochastic DP algorithm uses this variable to compute the expected cost of each state transition. The resulting optimal policy developed by this DP problem is the policy with the minimum expected cost. Of course, this may not be the lowest cost plan for a given set of monthly inflows, but it is the most robust policy in the long-run.

For the lake optimization model developed in this project, the above DP formula is solved subject to the following constraint equations.

$$vmin_i \leq v_i \leq vmax_i$$

This equation defines the reservoir's upper and lower allowable elevation for each month. These power pool limits may change seasonally.

$$(v_i - v_{i-1}) \leq maxrise_i$$

This equation establishes the maximum monthly increase in reservoir elevation. It is desirable to limit the algorithm's search to states that can be achieved with reasonable levels of inflow.

$$(v_{i-1} - v_i) \leq maxdrop_i$$

This equation establishes the maximum monthly decrease in reservoir elevation. This rate of decrease is determined by the U. S. Army Corps of Engineers.

$$Storage(v_i) - Storage(v_{i-1}) \leq MinInflow_i$$

This constraint requires the monthly increase in elevation (and thus the increase in storage) to be less than the lowest expected monthly inflow. The lowest expected inflow is based on a reliability constraint which can be modified by the user.

A flowchart of the stochastic DP model developed for this problem is shown in Figure 4. Details of certain parts of the flowchart are shown in Figures 5 and 6.

Given the constraints shown above, the water releases can be determined from the following equation.

$$x_i - (Storage(v_i) - Storage(v_{i-1})) = VolumeReleased$$

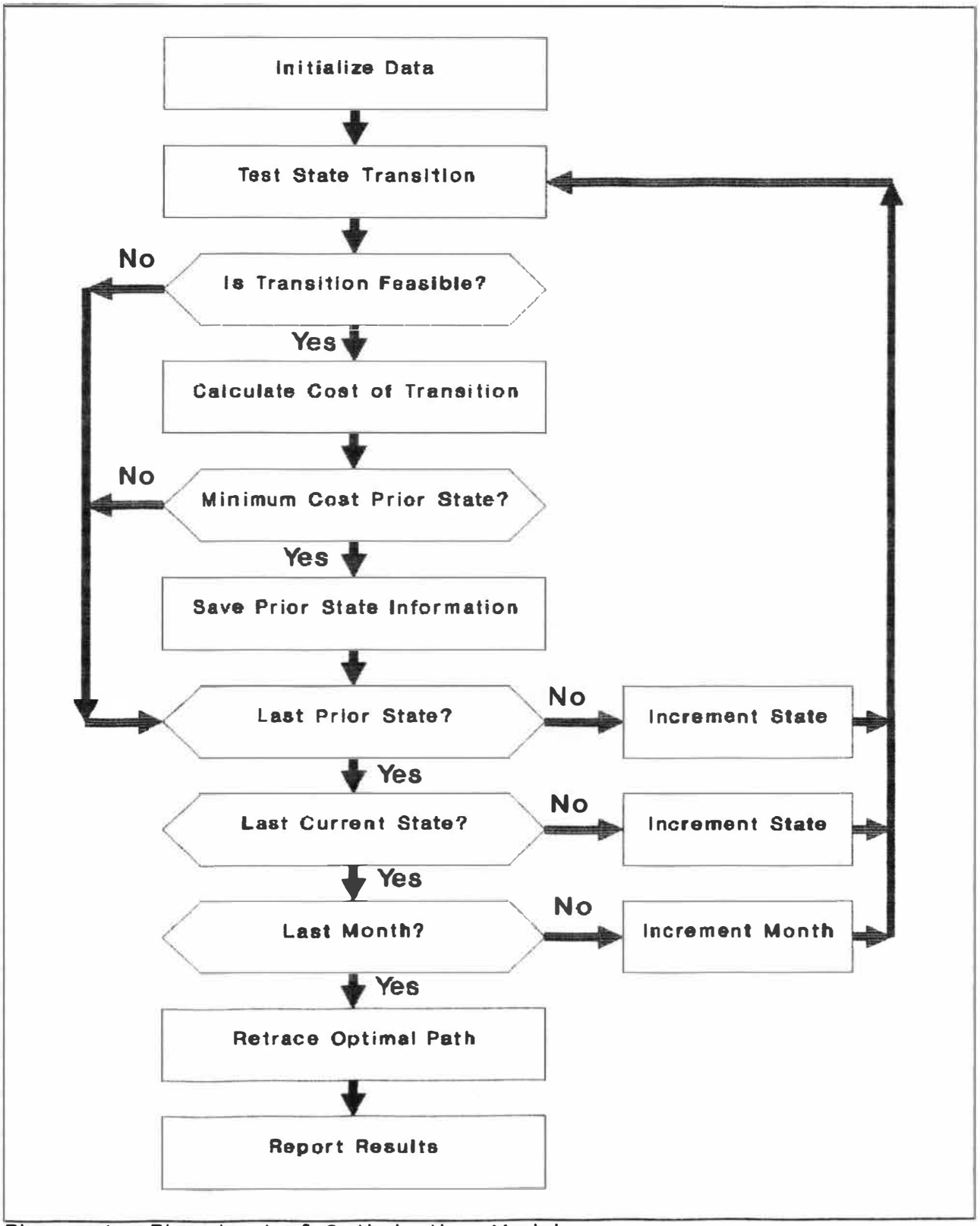

Figure 4. Flowchart of Optimization Model

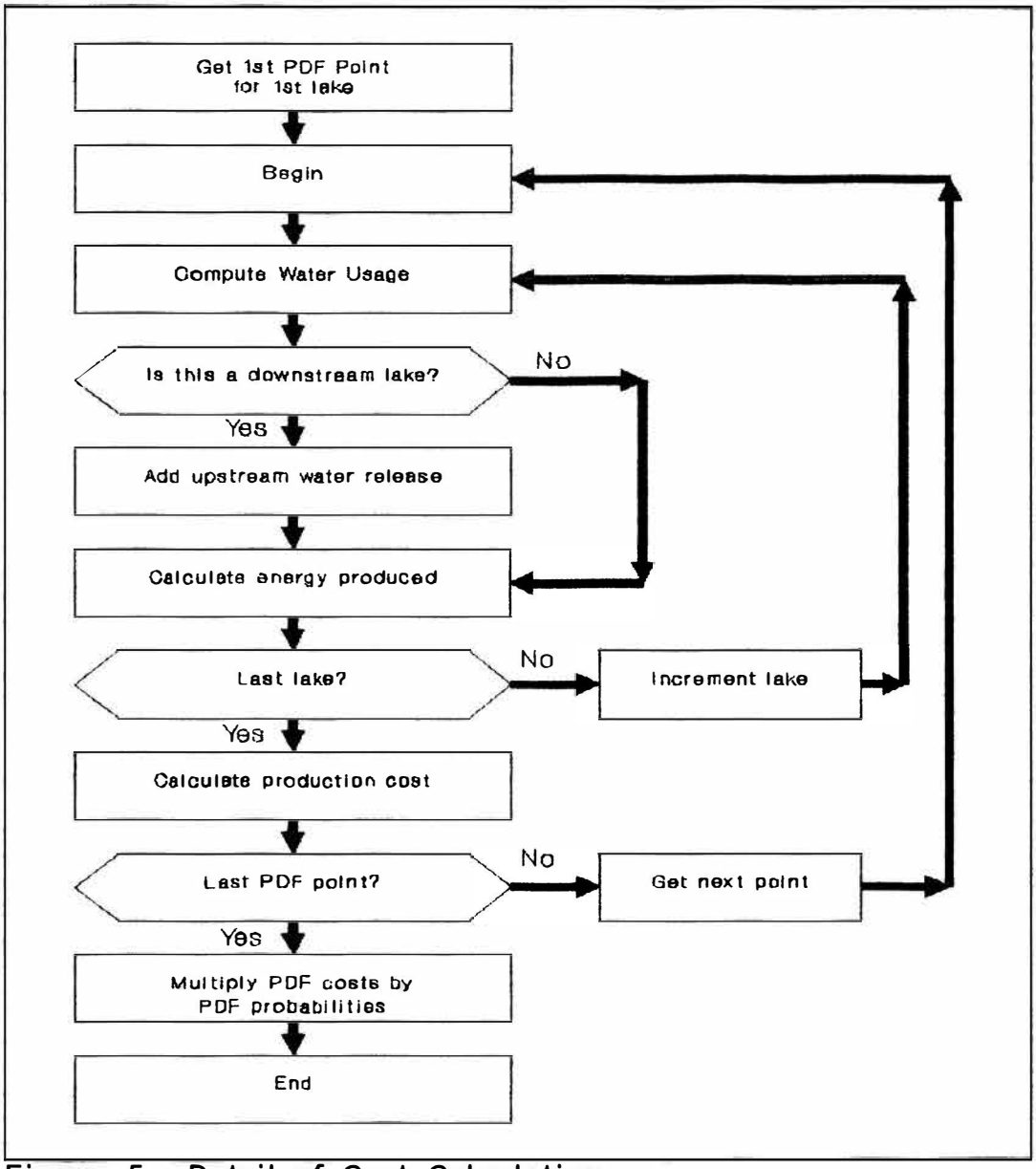

Figure 5. Detail of Cost Calculation

Water releases in any month are converted to energy at the average conversion ratio for the month, as determined by the starting and ending lake elevations. Once the energy has been calculated, an additional pair of constraints is added as follows.

$$MinGWh \leq Energy \leq MaxGWh$$

This allows limits to be placed on minimum and maximum energy production at a given lake. Minimum energy levels must sometimes be produced to provide "fish water," or an acceptable aquatic habitat downstream from the hydroelectric project. Maximum energy levels are required to properly account for forced spills which may occur during periods of extremely high inflow. Energy production will be limited to the maximum even if more water is moving downstream.

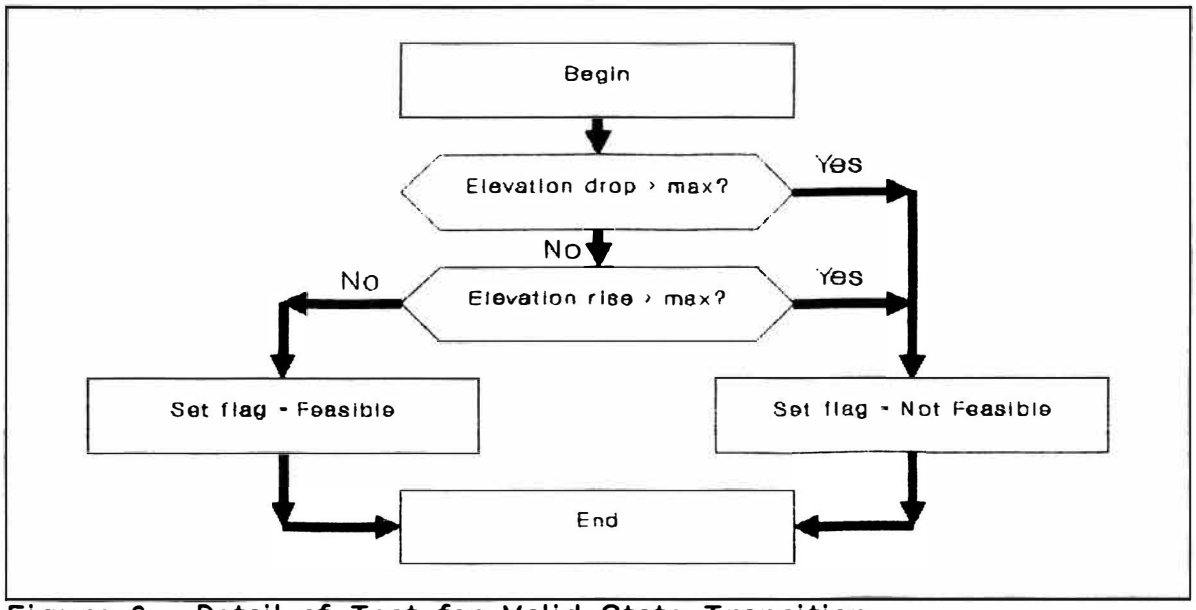

Figure 6. Detail of Test for Valid State Transition

Once the amount of generated energy has been determined, the system production cost can be calculated. In order to reduce the model's execution time, the detailed production cost calculations have

been decoupled from the DP loop, using the method of Viramontes and Hamilton (1978). Before running the SDP optimization, cost curves are created for each month of the forecast period. These curves relate the total system production cost to the amount of hydroelectric energy available for the month. An example of one of these curves is shown in Figure 7. The cost is shown for February of 1991. It can be seen from the figure that these cost curves are non-linear. It would be an oversimplification to fit these cost curves to a continuous function of some type. Because the DP method is able to optimize with non-linear and even discontinuous costs, the optimal system operating policy can still be found.

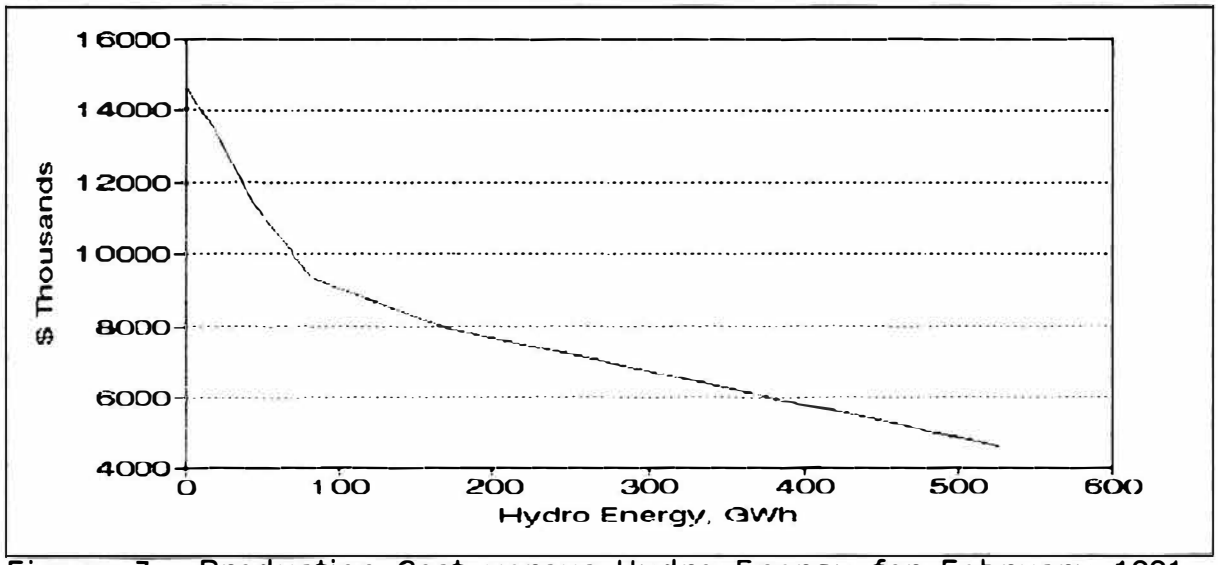

Figure 7. Production Cost versus Hydro Energy for February 1991

These system production cost computations are performed once for each of ten equally likely inflow scenarios, as derived from intervals on

the inflow probability density function (pdf). The cost values are multiplied by the corresponding pdf interval probability, and the ten values are added together to determine the expected cost for a particular state transition in a given month.

The expected cost is computed for each feasible state transition between months. The lowest accumulated cost to each state is saved, along with information about the prior state. Once this has been done for every feasible state in every month, the dynamic programming algorithm retraces the optimal path backwards, from the lowest cost state in the final month to the starting state in the first month.

It is important to note that the implicit assumption in this DP framework is that the monthly inflow from one lake to another is correlated. This means that when one lake has high inflow, all the others are assumed to have high inflow also. An additional assumption implicit in the model is the independence of monthly inflows on one another. This means that a high inflow in one month does not imply a greater probability of a high inflow in the next month. Both of these implicit assumptions are usually true for lakes in the midwest.

C. IMPLEMENTATION OF THE SDP ALGORITHM

The model was written in Turbo Pascal. This strongly typed language reduces the chances of programming errors by requiring a strictly structured program format. The availability of long, descriptive variable names helps to make the code somewhat self-documenting. Certain extensions, such as "record" variable types, greatly simplify the implementation of this type of model.

Pascal is supported on most computers, and Turbo Pascal (for IBM PC's or compatibles) is very similar to standard Pascal, so very few changes are required when transporting this program from one machine to another.

The code was written with extensive use of "constants" for key parameters. In this manner, future changes in the program will be more easily accommodated, for instance, it will be simple to change the program to include different lakes in the optimization process. The Pascal code for the model is included in Appendix A.

Early runs indicated that the program will always minimize production cost by emptying the lakes in the final two or three months of simulation. This "end effect" has been observed by several DP researchers. It was avoided by requiring the user to specify the ending level at the end of the study period.

IV. APPLICATION

The program to implement a stochastic dynamic programming solution for the multi-reservoir system has been written and debugged. It will find the optimal monthly target lake elevations for up to five lakes. The physical characteristics of each lake are stored in separate files, since this data does not change very often. An additional file is used for data that directs the optimization process. This data is likely to change often. Also, the pre-computed production cost information is stored in a file. This way many different cost scenarios can be used for different optimization studies.

A. INPUT DATA THAT REMAINS CONSTANT

Appendix B lists the input data for each lake. The individual data tables are discussed below.

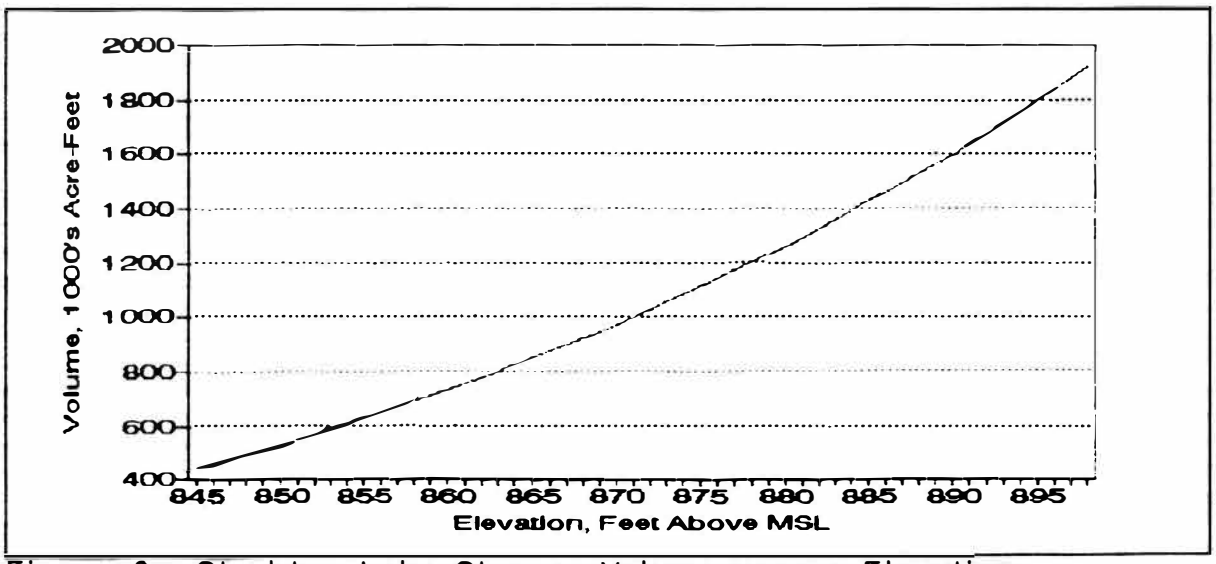

Figure 8. Stockton Lake Storage Volume versus Elevation

Lake Storage Volume. Figure 8 illustrates the near-linear relationship between lake storage volume and lake elevation at the Stockton reservoir. This data is unique for each hydroelectric project.

Lake Head Curves. Figure 9 shows the non-linear relationship between energy conversion efficiency and lake elevation for the Stockton reservoir. This curve is unique for every one of the hydroelectric projects.

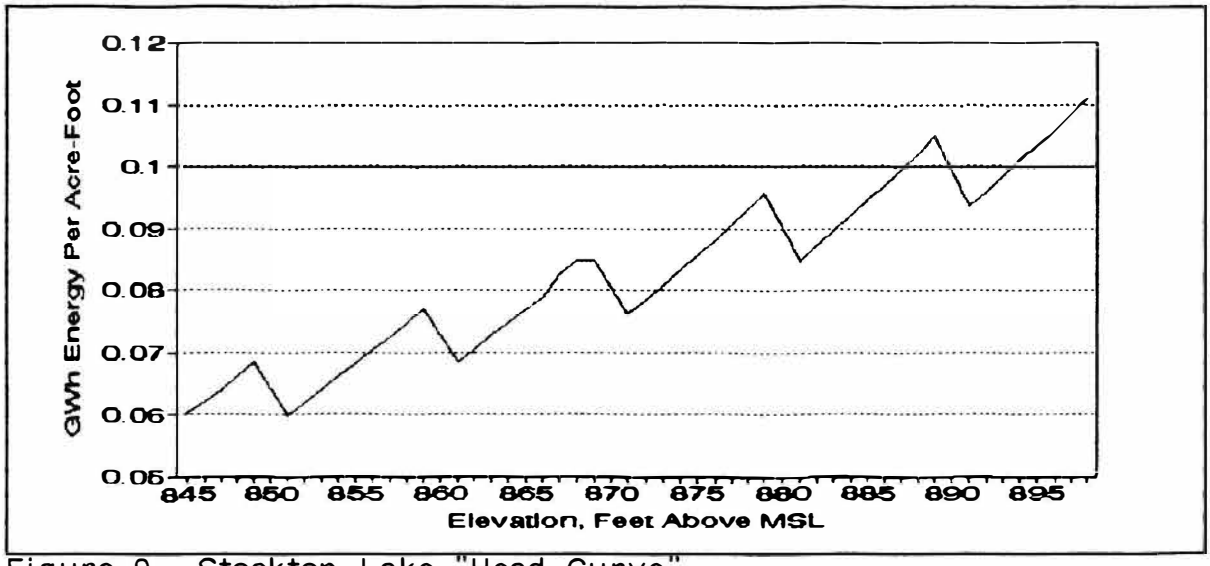

Figure 9. Stockton Lake "Head Curve"

Inflow Probability Density Function. Figure 10 shows the inflow data provided for the Stockton reservoir. It is represented as a ten-point probability density function, so ten different inflow levels are shown for each month. This data is unique for every hydroelectric reservoir.

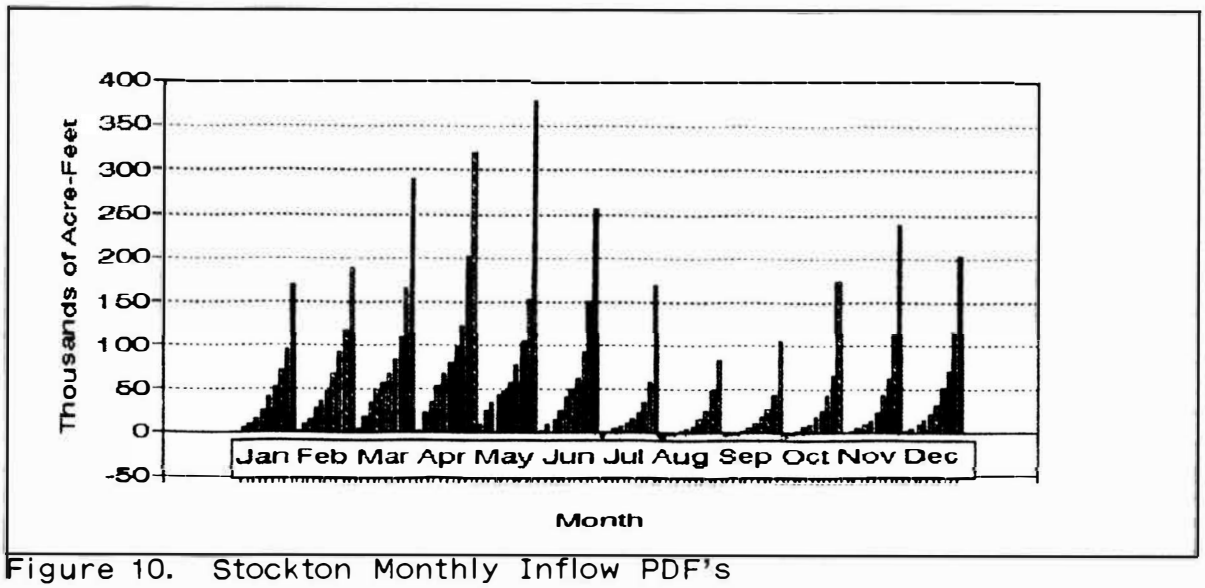

Figure 10. Stockton Monthly Inflow PDF's

Detailed Inflow Data for Reliability Calculations. The sorted historical inflow for each reservoir is also provided to the program. It is used to set a minimum level of reliability for the optimization solution. The level is set by specifying the minimum amount of inflow that can be expected in any given month, as a probability percentage.

B. INPUT DATA THAT CAN CHANGE

Appendix C contains an example input data file. This data is defined by the user and changes with each individual planning case.

Titles. The user can specify a title to identify each planning scenario uniquely.

Study Period. The user can specify the beginning month of the optimization period and the number of months in the period. The program can optimize for any period from one month to three years.

Reliability of the Optimized Schedule. This entry defines the minimum expected probability of inflow over the study period. By modifying this parameter, the user can determine the expected value of increased risk. A riskier schedule will have a lower reliability, but it also will have a lower expected cost. It is up to the user to decide what level of reliability is adequate.

Minimum and Maximum Probability Range for the Stochastic Optimization. This pair of parameters determines the range of the probability density function that the SDP algorithm will use in its search. Usually this is set from 1 to 10 to cover the entire ten-point probability density function. Sometimes the user may want to optimize for a particular range of inflow, for instance, he may want to optimize the lakes for a range of small inflows. This could be done by setting the probability range to 1 through 4.

Individual Lake Parameters. A series of parameters is provided for each of the hydroelectric projects.

- Beginning and Ending Elevation. This defines the elevation at the beginning and end of the study period. It is stated in feet above mean sea level.

- Maximum Rise and Minimum Drop. These items define the limits of the reservoir's elevation change in any given month. They are given in feet.

- Maximum and Minimum Energy Production. This pair of data defines the monthly energy production limits for each reservoir. Normally the minimum monthly production is allowed to be zero, although a higher amount could be specified. The maximum limit can be used to

constrain the energy production during periods of low dissolved oxygen levels. This information is given in Gigawatt-hours.

• Maximum and Minimum Monthly Elevations. This data restricts the SDP search space for each reservoir. This user should set these limits to reflect operational limitations of the reservoirs. For instance, the maximum power pool elevation varies seasonally at some projects.

Monthly Production Cost Information. A separate file is used to provide the monthly cost information to the program. This information is given as production cost (in thousands of dollars) versus the amount of hydroelectric energy produced (in Gigawatt-hours). Figure 7 shows an example production cost curve.

C. RUNNING THE PROGRAM

The program is call HOBIE (Hydro Optimization By Iterative Evaluation). To run the program, the user enters HOBIE at the DOS prompt. He will be prompted for a cost file name, an input data file name, and a name for the output file. As the program runs, a progress report is sent to the terminal screen. If the optimization process fails, which sometimes happens when the process is over-constrained, a descriptive error message is given to the user. Otherwise, the program completes its optimization and writes the results to the output file.

The program has been run for a four lake and a five lake system. For four lakes, the program will run in about 10 minutes on a 33 MHz 486-DX computer. With five lakes, the dimensionality is increased, and the program runs in about one and a half hours. Since the problem being solved is a long-term plan, this execution time is more than adequate.

V. RESULTS

Using the input data shown in Appendix B and C, the SDP model was run to solve a typical strategic planning problem. The model was used to plan the strategy for operating the five lakes from August of 1991 to July of 1992. From prior work it was known that it is optimal for the lakes to be full (with maximum stored energy) in July as the summer peak demand season approaches. The only question was how to manage the reservoir levels from August through the fall, winter and spring to provide the lowest expected energy production cost.

The model was run on a 486-DX computer running at 33 MHz, and it took about one and a half hours to reach a solution. The output from the model is shown in Appendix D.

Historically, the cooperative has managed the reservoir levels using a "rule curve." This curve dictates the target lake elevation for each month, based solely on reliability. The SDP model was run in a simulation mode to find the expected energy production cost when using rule curves to manage the lakes. The results of the two methods are compared in Table I, below.

Table I: Results of Example Study

	Optimized with SDP	Managed Using Rule Curve	Difference
Expected Energy Production Cost	$101,118,000	$101,800,000	$682,000
Expected Energy Lost due to Spill	136 GWh	129 GWh	7 GWh
Expected Energy Purchased due to Insufficient Inflow	52 GWh	136 GWh	-84 GWh

The optimal lake levels that were found by the model are shown for the two largest lakes, Table Rock and Bull Shoals, in Figure 12 and Figure 13. It can be seen from the figures that the SDP model did not draw the lakes down as much in the winter months as the rule curve would suggest.

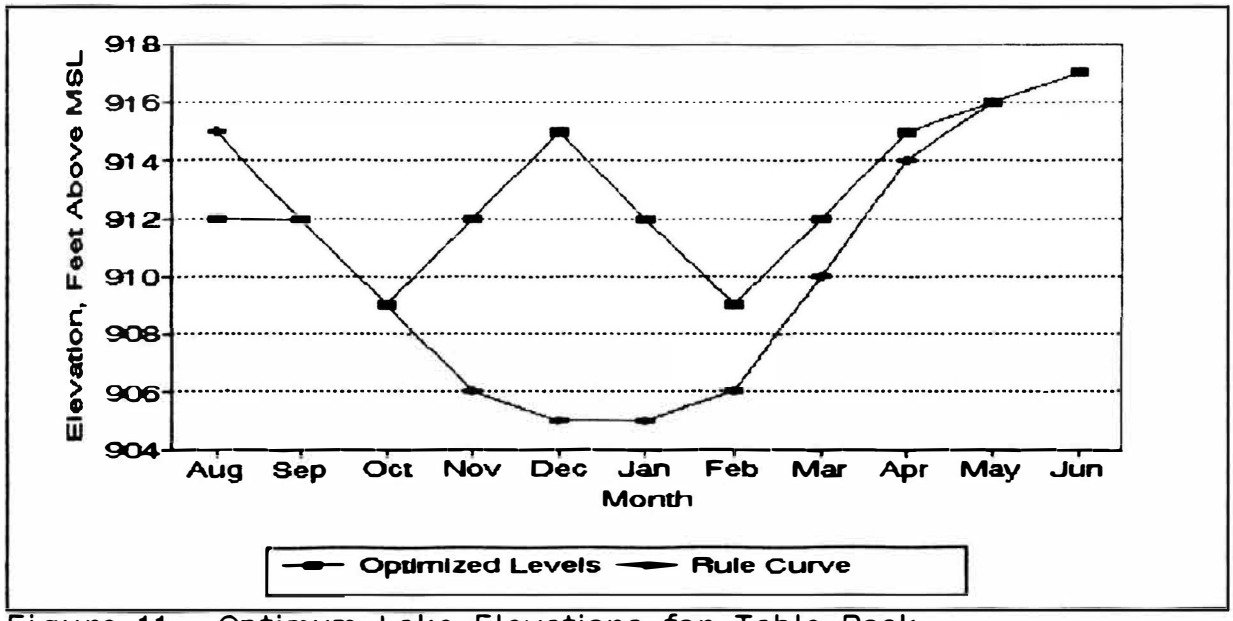

Figure 11. Optimum Lake Elevations for Table Rock

The model has produced a solution that has expected savings of $682,000 when compared to the rule curve method. Although this is only 0.7% of the total system production cost, it is still a number large enough to get the attention of Associated Electric's management. Based on these results, an effort is underway to continue development of this model.

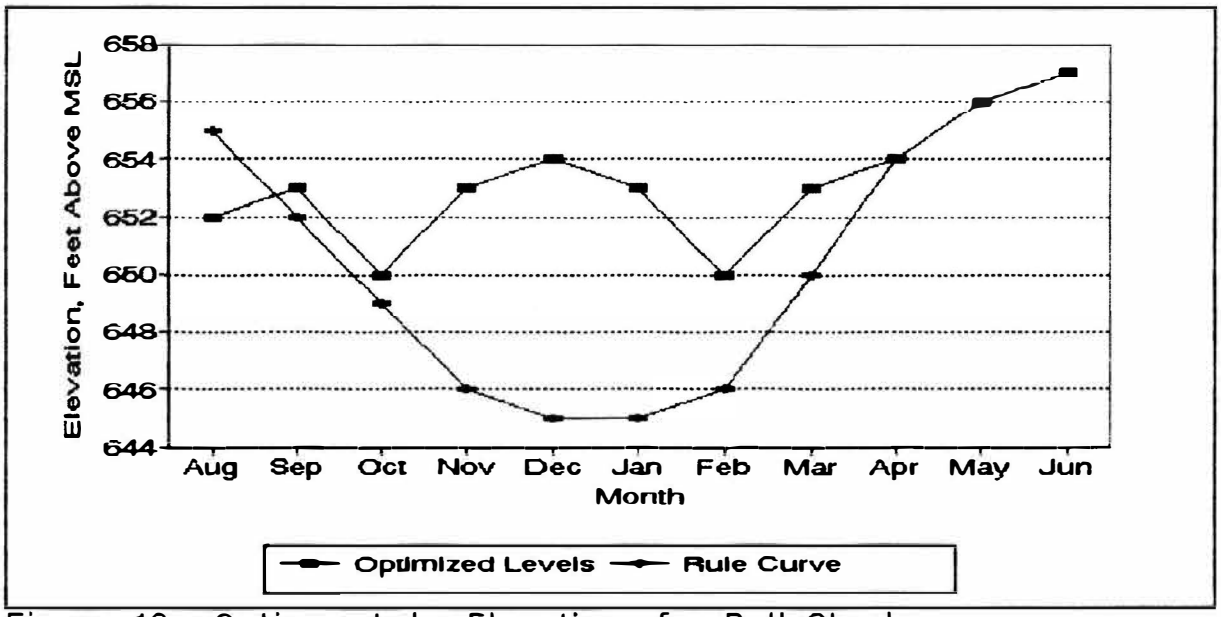

Figure 12. Optimum Lake Elevations for Bull Shoals

VI. CONCLUSIONS

The model has met its objective. It can take data for the five lakes and produce a management plan that meets all the constraints and saves money. The SDP method is an obvious improvement over the existing rule curve management method, producing expected savings of well over half a million dollars per year.

Many researchers have struggled with SDP, and some have said that it couldn't be used for systems with more than three lakes, but this research has proven that a direct solution can be found for a five lake system. A straightforward SDP algorithm was used. It was not compromised by adding odd, unproven heuristic rules. Extensions to SDP that may have caused it to produce sub-optimal solutions were not attempted, and the stochastic nature of the data was maintained without resorting to using simple average values for inflow.

This research was successful for three reasons. 1. By decoupling the detailed production cost analysis from the SDP loop, execution time was reduced without sacrificing accuracy. 2. Many realistic constraints were modeled. Additional constraints reduce the feasible state space and actually reduce SDP solution times. 3. Most importantly, computing power has grown rapidly in the last few years, making bigger problems easier to tackle.

The five-lake optimization that solves in 1.5 hours on a 486/33 MHz computer takes almost four hours on a 386/33 MHz with co-processor. This difference represents about one year's advance in computing power. These advances will continue to make yesterday's infeasible problem feasible for direct solution.

Although the SDP algorithm is working for Associated Electric, it is not without its weaknesses. The user must be careful not to over-constrain the problem or no feasible solutions may be possible. Work is underway on an expert system that will help the user overcome this problem if it occurs.

Perhaps the greatest weakness of this SDP algorithm is buried in the basic structure of the SDP approach. There are two assumptions that are implicit in any SDP solution to the hydrothermal coordination problem. First, because SDP solves a problem in sequential stages, it must be assumed that inflow is not correlated from one month to another. In reality, monthly inflows are weakly correlated.

A second assumption is that, within a given month, the inflows are fully correlated from one lake to another. This assumption is based in the consideration of stochastic inflows, where the algorithm loops over a ten-point probability density function of inflow. The lowest inflow is considered for every lake, then the next lowest, and so on. This means that all lake inflows are assumed to be fully correlated. In reality, the correlation of these inflows can be classified as moderate to strong. The influence of these assumptions on the optimality of the SDP solution is not currently known.

A solution process that can simulate operations over a sequential historical time period would preserve these correlations and could overcome the limitations of SDP. Simulated Annealing and Genetic Algorithms are two techniques that could be used. So far no one has applied these methods to the hydrothermal coordination problem, but it would make a good area for further study.

APPENDIX A

PASCAL CODE

```
PROGRAM HOBIE5;
{
  Version 006
}
USES Goodies;
CONST
  MaxSimulationPeriods = 36; {to limit the maximum simulation length}
  PeriodsPerCycle = 12; {for looping on 12 month data}
  MaxNumCostPts = 12; {for piecewise-linear cost curve representation}
  MaxNumSegments = MaxNumCostPts - 1;
  DebugLevel = 2;   {Maximum level is 4, 0 is no debugging trace
statements}
  TableRock  = 1;
  BullShoals = 2;
  Stockton   = 3;
  Truman     = 4;
  Cannon     = 5;
  FirstLake = TableRock;
  LastLake  = Cannon;    {fewer lakes solve faster}
  StaticDataPath = '\HYDRO\DATA\';
  MaxCurves = 5;
TYPE
  Real = Single;
  ValidLakes = FirstLake..LastLake;
  ValidPeriods = 0..MaxSimulationPeriods+1;
  CyclePeriods = 1..PeriodsPerCycle;
  PDF_Intervals = 1..10;
  LakeLevels = Integer;
  LakeTables = array[0..99] of real;  {used for various lookups}
  StatePtrs = ^SystemStates;
  SystemStates = record
    ParentPtr : StatePtrs;
    ExpectedCost : Real;
    ExpectedSpill : Real;
    ExpectedUnsrvd : Real;
    Level : array[ValidLakes] of LakeLevels;
  end;
  LakeRcds = record
    Name : string[12];
    AbbrevName : string[2];
    MaxElevation,
    MinElevation,
    MaxRise,
    MaxDrop       : array[CyclePeriods] of LakeLevels;
    MinInflow     : array[CyclePeriods] of Real;
    MaxGWh        : array[CyclePeriods] of Real;
    MinGWh        : array[CyclePeriods] of Real;
    Inflow        : array[CyclePeriods,PDF_Intervals] of Real;
    AcftUse       : Real;       {modified frequently in FeasiblePath}
    BaseLevel,                  {Base of following lookup tables}
    MaxLevel      : LakeLevels; {Maximum of lookup tables that follow}
    StorageVolume,
    Head          : LakeTables;
```

```
  end; {LakeRcds record}
  PtrRcds = record
    OfsWrd,
    SegWrd : word;
  end;
  Curves = 1..MaxCurves;

VAR
  NumCostPts,
  NumSegments,
  Segment      : byte;
  SegmentGWh,
  SegmentCost : array[Curves,CyclePeriods, 1..MaxNumCostPts ] of real;
  SegmentSlope: array[Curves,CyclePeriods, 1..MaxNumSegments] of real;
  CurveWeight : array[Curves] of byte;
  State        : SystemStates;
  StateFile    : file of SystemStates;
  PeriodPtr   : array[ValidPeriods] of StatePtrs;{includes 0,init value}
  PreviousState,
  PossibleState : SystemStates;
  Lake,
  CyclePeriod,
  Period : Integer;
  PreviousStatePtr,
  StatePtr,
  NextPtr,
  FirstPtr,
  PastLastPtr  : StatePtrs;
  Done,
  StillLooping : boolean;
  PctReliability : real;
  LakeRcd : array[ValidLakes] of LakeRcds;
  MinExpectedCost : real; {used in MAIN for replacing non-optimal paths}
  SimulationPeriods,     {number of months to simulate}
  FirstPeriod : integer; {first month to simulate}
  EndingLevel : array[ValidLakes] of LakeLevels;
  BaseStatesPtr : StatePtrs;
  MaxStatePtr  : StatePtrs;
  StateHeapOfs : LongInt;
  TitleLine1,
  TitleLine2 : string[80];
  fo : text; { output file }
  OutputFname,
  ProdCostFname,
  InputFname : string;
  NumCurves : byte;
  LoProbability,
  HiProbability : byte;
  FirstCtr,
  PastLastCtr,
  NextCtr : longint;
  PossibleCtr : longint;
  MaxRiseCtr,
```

```
  MaxDropCtr,
  MinInfCtr,
  MinGWhCtr : array [ValidLakes] of longint;
  PrevMinVldElv,
  PrevMaxVldElv : array[ValidLakes] of LakeLevels;
{
                    I n i t S t a t e H e a p
}
PROCEDURE InitStateHeap;
BEGIN {InitStateHeap}
  {Compute and save away base and max number of save-states}
  BaseStatesPtr := HeapPtr;
  MaxStatePtr    := HeapEnd;
END;  {InitStateHeap}
{
                      I n c S t a t e P t r
}
PROCEDURE IncStatePtr( var StatePtr : StatePtrs );
var PtrRcd : PtrRcds absolute StatePtr;
BEGIN {IncStatePtr}
  inc( PtrRcd.OfsWrd, SizeOf(SystemStates) );
  if ( PtrRcd.OfsWrd > $000F { $7FFF } )
  then with PtrRcd do begin
    SegWrd := SegWrd + ( OfsWrd SHR 4 );
    OfsWrd := OfsWrd AND $000F;
    if (SegWrd >= PtrRcds(MaxStatePtr).SegWrd) then begin
      writeln('$$$ERROR: Maximum States space overflowed!!!');
      HALT;
    end;
  end;
  (*
  inc( LongInt(StatePtr), SizeOf(SystemStates) );
  if (ofs(StatePtr^) > $3FFF)
  then StatePtr := ptr( seg(StatePtr^)+(ofs(StatePtr^) SHR 4),
ofs(StatePtr^) AND $000F);
  *)
END;  {IncStatePtr}
{
                        I n i t L a k e s

    Initializes lake data, reading from a file (lake_n.dat, n=1..5) for:
      max_drop, ft, by lake, by period
      max_elevation, ft, by lake, by period
      min_elevation, ft, by lake, by period
      storage_volume, acre-ft, in 1 ft intervals, by lake
      head_curves, GWh/acre-ft, in 1 ft intervals, by lake

    Reading from another file (inflow_n.dat, n=1..5) for:
      inflow, acre-ft/mo, by lake, by month, as a P.D.F. in
      "PDF_Intervals"

  Reading from another file (*.dat) for:
  hydro_value_curve, production cost as a piecewise linear cost curve
```

```
    for Segment := 1 to NumCostPts do
      for p := 1 to PeriodsPerCycle do begin
        readln(f, SegmentGWh[Curve,p,Segment], SegCost);
        SegmentCost[Curve,p,Segment] := SegCost / 1000.0;
      end;
    NumSegments := NumCostPts -1;
    for p := 1 to PeriodsPerCycle do
      for Segment := 1 to NumSegments do begin
        DY := SegmentCost[Curve,p,Segment] -
           SegmentCost[Curve,p,Segment+1];
        DX := SegmentGWh[Curve,p,Segment] -
           SegmentGWh[Curve,p,Segment+1];
        if DX = 0 then begin
         writeln('Vertical segment detected (Curve ',Curve,
                 ',Seg ',Segment,',Period ',p,').');
         Halt;
        end;
        SegmentSlope[Curve,p,Segment] := DY / DX;
      end;
  end; { for NumCurves }
  close(f);                {close production cost file}
  TotalWeight := 0;
  for Curve := 1 to NumCurves do
    TotalWeight := TotalWeight + CurveWeight[Curve];
  if TotalWeight <> 100 then begin
    writeln('Total weight of curves must be equal to 100 percent.');
    Halt;
  end;

END; {InitLakes}
{
                    G e t M i n I n f l o w

    determines the maximum rise allowed from prev period to curr period
    based on the minimum inflow expected with a given probability level

}
PROCEDURE GetMinInflow;
  VAR
    Slope,
    Intercept,
    RealRecord : Real;
    Points1,
    Points2 : array[CyclePeriods] of Real;
    Lake,
    p,
    j    : integer;
    YearsOfData,
    FirstRecord,
    NextRecord      : integer;
    f : Text;
    LakeNumberStr : string[1];
BEGIN {GetMinInflow}
```

```
  {
    interpolate the values in the pre-sorted inflow table
    1st value = number of years of data
    subsequent records are inflow values for nperiods (acre-ft/period)
    Map the integer values into the range from 0 to 100%
    Interpolate to the desired level
    100% reliability = 1st record = minimum historically observed inflow
  }
  for Lake := FirstLake to LastLake do with LakeRcd[Lake] do begin
    str(Lake,LakeNumberStr);
    Assign(f,StaticDataPath+'Mininf_'+LakeNumberStr+'.dat');
    Reset(f);       {open minimum inflow file for lake n}
    readln(f,YearsOfData);
    Slope := (1. - YearsOfData)/100.;
    Intercept := YearsOfData;
    RealRecord := Slope*PctReliability + Intercept;
    FirstRecord := trunc(RealRecord);
    NextRecord := FirstRecord + 1;
    Slope := (RealRecord - FirstRecord) / (NextRecord - FirstRecord);

    if (DebugLevel > 3) then writeln(' $$$DEBUG:
RealRec=',RealRecord:6:3,
         ' FirstRec=',FirstRecord:4,' Slope=',Slope:6:3);

    for j := 1 to FirstRecord-1 do readln(f);     {eat FirstRecord-1
records}
    for p := 1 to PeriodsPerCycle do read(f,Points1[p]);{read next 2
records}
    for p := 1 to PeriodsPerCycle do read(f,Points2[p]);
    close(f);

    for p := 1 to PeriodsPerCycle do
     MinInflow[p] := (Points2[p] - Points1[p]) * Slope + Points1[p];

  end; {for lake}
END; {GetMinInflow}
{
                    F e a s i b l e P a t h

    1. Check if path is valid from prev state to current state, subject
     to lake rise or drop does not exceeding the maximum allowed.
     (Rise governed by minimum inflow subject to specified reliability.)
     (Drop determined by maximum allowed drawdown.)
    2. If valid path, determine the expected energy to be obtained.
       (Energy limited by MaxGWh for limiting usage in wet months,
       also for modeling D. O. restrictions or maintenance deratings.)
    3. Determine the value of this expected energy from the value curve.
 }
FUNCTION FeasiblePath( var PathCost, GWhSpilled, GWhUnsrvd : Real ) :
boolean;
  VAR
    Probability,
    Lake         : integer;
```

```
    LevelChange : LakeLevels;
    Cost,
    ThisVolume,
    UpstreamRelease,
    ThisGWhSpilled,
    TotalGWhSpilled,
    ThisGWhUnsrvd,
    TotalGWhUnsrvd,
    ThisGWh,
    TotalGWh    : real;
    Curve : byte;
    AcFt2GWh : array [FirstLake..LastLake] of real;
BEGIN {FeasiblePath}
  FeasiblePath := false;
  with PossibleState do begin

    {test for allowable rise or drop from previous period}
    Lake := FirstLake;
    repeat
      with LakeRcd[Lake] do begin
        AcFt2GWh[Lake] :=
             ( Head [ PreviousState.Level[Lake] - BaseLevel ]
             + Head [ Level[Lake] - BaseLevel ]
             ) / 2;
        LevelChange := Level[Lake] - PreviousState.Level[Lake];
        if(LevelChange > MaxRise[CyclePeriod])
          or
          (LevelChange < -MaxDrop[CyclePeriod])
          then Exit;

        ThisVolume :=  StorageVolume [ PreviousState.Level[Lake]
              - BaseLevel ]
              + MinInflow[CyclePeriod] ;
        if (Lake = BullShoals) or (Lake = Truman) then begin
    {for downstream chained lakes, include potential upstream releases
     assuming "minimum" upstream inflow as defined by user}
          UpstreamRelease :=
              StorageVolume[PreviousState.Level[Lake-1]
              - LakeRcd[Lake-1].BaseLevel ]
              - StorageVolume[Level[Lake-1] - LakeRcd[Lake-1].BaseLevel]
              + LakeRcd[Lake-1].MinInflow[CyclePeriod];
          ThisVolume := ThisVolume + UpstreamRelease;
        end; {if Lake}
        if (ThisVolume < StorageVolume [ Level[Lake] - BaseLevel ] )
        then Exit;
        ThisVolume := ThisVolume - StorageVolume[Level[Lake]-BaseLevel];
        ThisGWh := ThisVolume * AcFt2GWh[Lake];
        if ThisGWh < MinGWh[Lake]
        then Exit;
      end; { with LakeRcd }
      inc(Lake);
    until Lake > LastLake;
```

```
    FeasiblePath := TRUE;

    {find useage in acre-feet, then GWh for each lake,
     include effects of chained lakes on each other (dependent inflow)}

    PathCost := 0.0;
    GWhUnsrvd := 0.0;
    GWhSpilled := 0.0;
    Probability := LoProbability;
    repeat
      TotalGWh := 0.0;
      TotalGWhUnsrvd := 0.0;
      TotalGWhSpilled := 0.0;
      Lake := FirstLake;
      repeat
        with LakeRcd[Lake] do begin
          AcFtUse := StorageVolume[PreviousState.Level[Lake]
                     - BaseLevel ]
                     - StorageVolume[Level[Lake] - BaseLevel ]
                     + inflow[CyclePeriod,Probability];

      {for chained projects, add the upstream lakes discharge to inflow}
          If(Lake = BullShoals) or (Lake = Truman) then
                     AcFtUse := AcFtUse +  LakeRcd[Lake-1].AcFtUse;

        {for all projects, if PDF is negative, show zero energy useage}
          If(AcFtUse < MinInflow[CyclePeriod]) then
            ThisGWhUnsrvd := AcFt2GWh[Lake] * (MinInflow[CyclePeriod] -
              AcFtUse)
          else
            ThisGWhUnsrvd := 0.0;
          if(AcFtUse < 0.0) then AcFtUse := 0.0;
          ThisGWh := AcFt2GWh[Lake] * AcftUse;
          If(ThisGWh > MaxGWh[Lake]) then begin
            ThisGWhSpilled := ThisGWh - MaxGWh[Lake];
            ThisGWh := MaxGWh[Lake];
          end else ThisGWhSpilled := 0.0;
          TotalGWh         := TotalGWh + ThisGWh;
          TotalGWhUnsrvd   := TotalGWhUnsrvd + ThisGWhUnsrvd;
          TotalGWhSpilled  := TotalGWhSpilled + ThisGWhSpilled
        end; { with LakeRcd }
        inc(Lake);
      until Lake > LastLake;

      Cost := 0;
      for Curve := 1 to NumCurves do begin
        {check that total energy produced is within cost curve bounds}
        if(TotalGWh > SegmentGWh[Curve,CyclePeriod,NumCostPts]) then
begin
          writeln(' **ERROR in cost calculation (FeasiblePath) ');
          writeln('         Too much energy in period ',CyclePeriod:3);
        halt;
        end; {if}
```

```
        Segment := 1;
        while (TotalGWh > SegmentGWh[Curve,CyclePeriod,Segment+1]) do
          Inc(Segment);

        Cost := Cost + (CurveWeight[Curve] /100 ) *
               ( SegmentCost[Curve,CyclePeriod,Segment]
               + SegmentSlope[Curve,CyclePeriod,Segment]
               * (TotalGWh - SegmentGWh[Curve,CyclePeriod,Segment]) );

      end; { for NumCurves }
      GWhUnsrvd := GWhUnsrvd + TotalGWhUnsrvd;
      GWhSpilled := GWhSpilled + TotalGWhSpilled;
      PathCost := PathCost + Cost;
      inc(Probability);
    until Probability > HiProbability;
    {Average the cost over the 10 PDF intervals}
    PathCost := PathCost/(HiProbability-LoProbability+1);
    GWhUnsrvd := GWhUnsrvd/(HiProbability-LoProbability+1);
    GWhSpilled := GWhSpilled/(HiProbability-LoProbability+1);
  end; {with PossibleState do begin}
END; {FeasiblePath}
{
                      P a t h  C o m p u t e d

  Performs:  1st Copy PreviousStatePtr^ to PreviousState working record

             2nd check for path validity from PreviousState -to-
             PossibleState;

             3rd if valid, compute estimated cost of path from
             PreviousState -to- PossibleState and return TRUE
             else return FALSE

}

FUNCTION PathComputed : boolean;
  VAR
    Lake : Integer;
    Result : Boolean;
    Cost,
    GWhUnsrvd,
    GWhSpilled : Real;
BEGIN {PathComputed}
  {get working copy of previous state for execution efficiency}
  {RETRIEVE OLD STATE RECORD}
  PreviousState := PreviousStatePtr^;

  if (DebugLevel >= 4) then begin
    write('Comparing: (');
    with PreviousState do
    for Lake := FirstLake to LastLake do write(Level[Lake]:4);
    write(') -to- (');
    with PossibleState do
```

```
    for Lake := FirstLake to LastLake do write(Level[Lake]:4);
    write(')');
  end; {Debugging trace printout}

  {Do ExpectedCost calculation here}
  Result := FeasiblePath(Cost,GWhSpilled,GWhUnsrvd);
  if (Result)
  then with PossibleState do begin
    {accumulate cost along path, and link parenthood}
    ExpectedCost := PreviousState.ExpectedCost + Cost;
    ExpectedSpill := PreviousState.ExpectedSpill + GWhSpilled;
    ExpectedUnsrvd := PreviousState.ExpectedUnsrvd + GWhUnsrvd;
    ParentPtr := PreviousStatePtr;
  end; {if Result}

  if (DebugLevel >= 4) then begin
    if (Result)
    then writeln(' ExptdCost:',PossibleState.ExpectedCost:8:2)
    else writeln(' Invalid');
  end; {Debugging trace printout}

  PathComputed := Result;
END;  {PathComputed}
{
                R e s e t S y s t e m
}
PROCEDURE ResetSystem;
  VAR IP : Integer;
BEGIN {ResetSystem}
  PeriodPtr[1] := NextPtr;
  for IP := 2 to MaxSimulationPeriods+1 do PeriodPtr[IP] := nil;
END;  {ResetSystem}
{
             I n i t i a l i z e S y s t e m
}
PROCEDURE InitializeSystem;
  VAR Lake : Integer;
      f : text;
      p : integer;
      fnam : string;
BEGIN {InitializeSystem}

  Assign(Input,''); Reset(Input);     {Allow input redirection}
  Assign(Output,''); Rewrite(Output); {Allow output redirection}

  For Lake := FirstLake to LastLake do begin
    case Lake of
      TableRock:  begin
        LakeRcd[Lake].Name := 'TableRock';
        LakeRcd[Lake].AbbrevName := 'TR';
      end;
      BullShoals: begin
        LakeRcd[Lake].Name := 'BullShoals';
```

```
        LakeRcd[Lake].AbbrevName := 'BS';
      end;
      Stockton:   begin
        LakeRcd[Lake].Name := 'Stockton';
        LakeRcd[Lake].AbbrevName := 'ST'
      end;
      Truman:     begin
        LakeRcd[Lake].Name := 'Truman';
        LakeRcd[Lake].AbbrevName := 'TM';
      end;
      Cannon:     begin
        LakeRcd[Lake].Name := 'Cannon';
        LakeRcd[Lake].AbbrevName := 'CA';
      end;
    else writeln(' Error:  Invalid Lake Specified in FirstLake,
LastLake')
    end; {case}
  end; {for Lake}

  InitLakes;   {read in data files}
  with PossibleState do begin
    ParentPtr := nil;
    ExpectedCost := 0.0;
    ExpectedSpill := 0.0;
    ExpectedUnsrvd := 0.0;
    if InputFname = '' then begin
      writeln('Enter Filename for lake starting levels & reliability:');
      readln(InputFname);
    end;
    assign(f,InputFname);
    reset(f);
    if OutputFname = '' then begin
      writeln('Enter Filename for output:');
      readln(OutputFname);
    end;
    assign(fo,OutputFname);
    rewrite(fo);
    readln(f,TitleLine1);
    readln(f,TitleLine2);
    writeln(fo,TitleLine1);
    writeln(fo,TitleLine2);
    writeln(fo);
    writeln(fo,'Production Cost file name:  ',ProdCostFname);
    writeln(fo,'Lake information file name: ',InputFname);
    readln(f,FirstPeriod,SimulationPeriods);
    readln(f,PctReliability);
    if (PctReliability < 1.0) OR (PctReliability > 100.0) then begin
      writeln(PctReliability,' does not fall in the range [1..100].');
      Halt;
    end;
    readln(f,LoProbability,HiProbability);
    if (NOT (LoProbability in [1..10])) OR
       (NOT (HiProbability in [1..10])) OR
```

```
       (LoProbability > HiProbability)
    then begin
      writeln('[',LoProbability,'..',HiProbability,
              '] is not a valid probability range.');
      writeln('Try a range within [1..10].');
      Halt;
    end;
    for Lake := FirstLake to LastLake do with LakeRcd[Lake] do begin
      readln(f,Level[Lake],EndingLevel[Lake]);
      for p := 1 to PeriodsPerCycle do read(f, MaxRise[p]);readln(f);
      for p := 1 to PeriodsPerCycle do read(f, MaxDrop[p]);readln(f);
      for p := 1 to PeriodsPerCycle do read(f, MaxGWh[p]);readln(f);
      for p := 1 to PeriodsPerCycle do read(f, MinGWh[p]);readln(f);
      for p := 1 to PeriodsPerCycle do read(f,
MaxElevation[p]);readln(f);
      for p := 1 to PeriodsPerCycle do read(f,
MinElevation[p]);readln(f);
      if DebugLevel > 2 then begin
        writeln(' MaxDrop:');
        for p := 1 to PeriodsPerCycle do write(MaxDrop[p]:4);writeln;
        writeln(' MaxElevation:');
        for p := 1 to PeriodsPerCycle do
write(MaxElevation[p]:4);writeln;
        writeln(' MinElevation:');
        for p := 1 to PeriodsPerCycle do
write(MinElevation[p]:4);writeln;
      end; { if DebugLevel > 2 }
    end; {for}
  end; {with}

  {SAVE FIRST STATE RECORD}
  {OPEN SCRATCH FILE FOR SYSTEM STATES}
  InitStateHeap;
  StatePtr := BaseStatesPtr;
  StatePtr^ := PossibleState;
  PeriodPtr[0] := StatePtr;
  NextPtr := StatePtr;
  IncStatePtr(NextPtr);
  PastLastCtr := 1;
  NextCtr := 2;
  ResetSystem;
END;  {InitializeSystem}
{
                  R e p o r t R e s u l t s
}
PROCEDURE ReportResults;
  VAR StatePtr,
      MinStatePtr : StatePtrs;       {Used in minimum search}
      MinExpectedCost : Real;
      MinPath : array[ValidPeriods] of SystemStates; {Will hold min path
state}
      Lake, IP : Integer;            {Used in looping statements}
      Period : Integer;
```

```
      PrevCost,
      AccumCost : Real;
BEGIN {ReportResults}
  {First locate minimum state in last period}
  FirstPtr := PeriodPtr[SimulationPeriods];
  PastLastPtr := PeriodPtr[SimulationPeriods+1];
  MinStatePtr :=FirstPtr;  {Initialize first search state}

  PreviousState := FirstPtr^;

  MinExpectedCost := PreviousState.ExpectedCost;
  StatePtr := FirstPtr;
  while (StatePtr <> PastLastPtr) do begin

    PossibleState := StatePtr^;

    if (PossibleState.ExpectedCost < MinExpectedCost)
    then begin
      MinExpectedCost := PossibleState.ExpectedCost;
      PreviousState := PossibleState;
      MinStatePtr := StatePtr;
    end; {if}
    IncStatePtr( StatePtr );
  end;

  {Now, backtrack through ParentPtrs from this final solution}
  if (DebugLevel >= 1)
  then writeln(' $$$DEBUG: Backtrack indices of minimum path states');
  Period := SimulationPeriods;
  repeat
    MinPath[Period] := MinStatePtr^;
    if (DebugLevel >= 2)
    then writeln('    Period=',Period:3,'
StatePtr=',Ptr2Str(MinStatePtr));
    MinStatePtr := MinPath[Period].ParentPtr;
    Period := Period - 1;
  until (MinStatePtr = nil);

  {Next, print results}
  writeln(fo);
  writeln(fo,'SUCCESS, minimum ', SimulationPeriods , ' month expected
cost is:',
    MinExpectedCost:10:3);
  write(fo,'Reliability = ',PctReliability:4:0,' %');
  writeln(fo,'   Inflow probability range :
 ,LoProbability,'-',HiProbability);
  writeln(fo,'First Period = ',FirstPeriod:4);
  writeln(fo);
  {Print to screen}
  writeln;
  writeln('SUCCESS, minimum ', SimulationPeriods , ' month expected cost
is:',
    MinExpectedCost:10:3);
```

```
  write('Reliability = ',PctReliability:4:0,' %');
  writeln('   Inflow probability range :
',LoProbability,'-',HiProbability);
  writeln('First Period = ',FirstPeriod:4);
  writeln;
{$DEFINE Columns}
{$IFDEF Columns}
  Period := FirstPeriod;
  PrevCost := 0;
  for Lake := FirstLake to LastLake do begin
    write('     ');
    write(fo,'     ');
  end;
  writeln('       Monthly  Accum.  Spilled  Unserved');
  writeln(fo,'       Monthly  Accum.  Spilled  Unserved');
  for Lake := FirstLake to LastLake do begin
    write('     ');
    write(fo,'     ');
  end;
  writeln('          Cost    Cost   Energy    Energy');
  write(' Mo Pd');
  writeln(fo,'          Cost    Cost   Energy    Energy');
  write(fo,' Mo Pd');
  for Lake := FirstLake to LastLake do with LakeRcd[Lake] do begin
    write(AbbrevName:5);
    write(fo,AbbrevName:5);
  end;
  writeln('    (M$)    (M$)   (GWhs)    (GWhs)');
  writeln(fo,'    (M$)    (M$)   (GWhs)    (GWhs)');
  for IP := 1 to SimulationPeriods do begin
    write(Period:3,IP:3);
    write(fo,Period:3,IP:3);
    if Period = 12 then Period := 1 else inc(Period);
    for Lake := FirstLake to LastLake do with LakeRcd[Lake] do begin
      write(MinPath[IP].Level[Lake]:5);
      write(fo,MinPath[IP].Level[Lake]:5);
    end;
    AccumCost := MinPath[IP].ExpectedCost;
    writeln((AccumCost-PrevCost):8:1,AccumCost:8:1,
            ' ',MinPath[IP].ExpectedSpill:8:1,'
',MinPath[IP].ExpectedUnsrvd:8:1);
    writeln(fo,(AccumCost-PrevCost):8:1,AccumCost:8:1,
            ' ',MinPath[IP].ExpectedSpill:8:1,'
',MinPath[IP].ExpectedUnsrvd:8:1);
    PrevCost  := AccumCost;
  end;
  writeln;
{$ELSE}
  Period := FirstPeriod;
  for IP := 1 to SimulationPeriods do begin
    write(fo,Period:7);
    write(Period:7); { to screen }
    if (Period = 12)
```

```
    then Period := 1
    else inc(Period);
  end;
  writeln(fo);
  writeln; { to screen }
  for Lake := FirstLake to LastLake do with LakeRcd[Lake] do begin
    write(fo,Name:12);
    write(Name:12); { to screen }
    Period := 0;
    for IP := 1 to SimulationPeriods do begin
      write(fo,MinPath[IP].Level[Lake]:7);
      write(MinPath[IP].Level[Lake]:7); { to screen }
      if (Period = (PeriodsPerCycle-1))
      then begin
        writeln; { to screen }
        write('            '); { to screen }
        Period := 0;
      end
      else inc(Period);
    end; {for}
    writeln(fo);
    writeln; { to screen }
  end; {Lakes Loop}
  write(fo,' Accum_Cost:');
  write(' Accum_Cost:'); { to screen }
  for IP := 1 to SimulationPeriods do begin
    write(fo,MinPath[IP].ExpectedCost:7:1);
    write(MinPath[IP].ExpectedCost:7:1); { to screen }
  end;
  writeln(fo);
  writeln(fo);
  writeln; { to Screen }
  writeln; { to Screen }
{$ENDIF}
END;  {ReportResults}

procedure GetCommandLineParameters;
var i : word;
begin   { GetCommandLineParameters }
  ProdCostFname := '';
  InputFname    := '';
  OutputFname   := '';
  if ParamCount > 0 then ProdCostFname := ParamStr(1);
  if ParamCount > 1 then InputFname    := ParamStr(2);
  if ParamCount > 2 then OutputFname   := ParamStr(3);
end;    { GetCommandLineParameters }
{
                    C h e c k C o n s t r a i n t s

    1. Check if path is valid from prev state to current state, subject
to
       lake rise or drop does not exceeding the maximum allowed, minimum
       inflow, and minGWh.
```

```
        (Rise governed by minimum inflow subject to specified
reliability.)
        (Drop determined by maximum allowed drawdown.)
 }
procedure CheckConstraints;
  VAR
    Probability,
    Lake         : integer;
    LevelChange : LakeLevels;
    Cost,
    ThisVolume,
    UpstreamRelease,
    ThisGWh : real;
    AcFt2GWh : real;
BEGIN {CheckConstraints}
  PreviousState := PreviousStatePtr^;
  with PossibleState do begin
    inc(PossibleCtr);
    {test for allowable rise or drop from previous period}
    Lake := FirstLake;
    repeat
      if PreviousState.Level[Lake] < PrevMinVldElv[Lake]
      then PrevMinVldElv[Lake] := PreviousState.Level[Lake];
      if PreviousState.Level[Lake] > PrevMaxVldElv[Lake]
      then PrevMaxVldElv[Lake] := PreviousState.Level[Lake];
      with LakeRcd[Lake] do begin
        AcFt2GWh :=
             ( Head [ PreviousState.Level[Lake] - BaseLevel ]
             + Head [ Level[Lake] - BaseLevel ]
             ) / 2;
        LevelChange := Level[Lake] - PreviousState.Level[Lake];
        {
          Check MaxRise constraint
        }
        if (LevelChange > MaxRise[CyclePeriod]) then
          inc(MaxRiseCtr[Lake]);
        {
          Check MaxDrop constraint
        }
        if (LevelChange < -MaxDrop[CyclePeriod]) then
          inc(MaxDropCtr[Lake]);

        ThisVolume :=  StorageVolume [ PreviousState.Level[Lake]
              - BaseLevel ]
              + MinInflow[CyclePeriod] ;
        if (Lake = BullShoals) or (Lake = Truman) then begin
        {for downstream chained lakes, include potential upstream
releases
         assuming "minimum" upstream inflow as defined by user}
          UpstreamRelease :=
              StorageVolume[PreviousState.Level[Lake-1]
              - LakeRcd[Lake-1].BaseLevel ]
              - StorageVolume[Level[Lake-1] - LakeRcd[Lake-1].BaseLevel]
```

```
            + LakeRcd[Lake-1].MinInflow[CyclePeriod];
          ThisVolume := ThisVolume + UpstreamRelease;
        end; {if Lake}
        {
          Check MinInf constraint
        }
        if (ThisVolume < StorageVolume [ Level[Lake] - BaseLevel ] )
        then begin
          inc(MinInfCtr[Lake]);
          inc(MinGWhCtr[Lake]);
        end
        else begin
          ThisVolume := ThisVolume -
            StorageVolume[Level[Lake]-BaseLevel];
          ThisGWh := ThisVolume * AcFt2GWh;
          {
            Check MinGWh constraint
          }
          if ThisGWh < MinGWh[Lake]
          then inc(MinGWhCtr[Lake]);
        end;
      end; { with LakeRcd }
      inc(Lake);
    until Lake > LastLake;
  end; { with PossibleState }
END; {CheckConstraints}

procedure Diagnostics;
begin   { Diagnostics }
  writeln;
  writeln('$$$Diagnosing failure to get from computing period
',Period-1,' to ',Period,'.');
  writeln(fo,' Diagnosing failure to get from computing period
',Period-1,' to ',Period,'.');

  if (DebugLevel >= 2)
  then begin
    writeln(' Previous period states to be searched are from: ',
              Lng2Str(FirstCtr),' -to- ',Lng2Str(PastLastCtr-1));
    writeln(fo,' Previous period states to be searched are from: ',
              Lng2Str(FirstCtr),' -to- ',Lng2Str(PastLastCtr-1));
  end;
  with PossibleState do begin

    {for the final simulation period, force min and max lake levels to
     the desired ending levels supplied by the user}
    if(Period = SimulationPeriods) then
      for Lake := FirstLake to LastLake do begin
        LakeRcd[Lake].MinElevation[CyclePeriod]:=EndingLevel[Lake];
        LakeRcd[Lake].MaxElevation[CyclePeriod]:=EndingLevel[Lake];
    end; {for lake}

    {Reset all lakes to bottom, begin cycling 1st lake again}
```

```
    for Lake := FirstLake to LastLake do begin
      Level[Lake] := LakeRcd[Lake].MinElevation[CyclePeriod];
      MaxRiseCtr[Lake] := 0;
      MaxDropCtr[Lake] := 0;
      MinInfCtr[Lake]  := 0;
      MinGWhCtr[Lake]  := 0;
      PrevMinVldElv[Lake]  := FirstPtr^.Level[Lake];
      PrevMaxVldElv[Lake]  := FirstPtr^.Level[Lake];
    end;
    Done := false;
    PossibleCtr := 0;
    repeat  {until done}

{ Compare this possible state to all previous states for minimum path}
      PreviousStatePtr := FirstPtr;
      while (PreviousStatePtr <> PastLastPtr) do begin
        CheckConstraints;
        IncStatePtr(PreviousStatePtr);
      end; {while}

      { Increment to next lakes state }
      Lake := FirstLake;
      StillLooping := false;
      repeat
      Level[Lake] := Level[Lake] + 1;
        with LakeRcd[Lake] do begin
          if (Level[Lake] > MaxElevation[CyclePeriod]) then begin
          if (Lake <> LastLake) then begin
            Level[Lake] := MinElevation[CyclePeriod];
            Lake := Lake + 1;
            StillLooping := true;
          end
          else begin
            Done := true;
            StillLooping := false;
            end;
        end
          else StillLooping := false;
        end; {with LakeRcd[Lake]}
      until (not StillLooping);

    until (Done);

  { report to screen }

    writeln('Diagnostics report');
    writeln;
    writeln('Number of valid previous states : ',PastLastCtr-FirstCtr);
    writeln('Number of state transitions attempted : ',PossibleCtr);
    writeln;
    write('                      ');
    for Lake := FirstLake to LastLake do with LakeRcd[Lake] do begin
      write(AbbrevName:8);
```

```
    end;
    writeln;

    write('Previous Elevations ');
    for Lake := FirstLake to LastLake do begin
      write(' ',PrevMinVldElv[Lake]:3,'-',PrevMaxVldElv[Lake]:3);
    end;
    writeln;

    write('Power Pool Range    ');
    for Lake := FirstLake to LastLake do begin
      write('
',LakeRcd[Lake].MinElevation[CyclePeriod]:3,'-',LakeRcd[Lake].MaxElevati
on[Lake]:3);
    end;
    writeln;

    write('Max Rise Violations ');
    for Lake := FirstLake to LastLake do begin
      write(MaxRiseCtr[Lake]:8);
    end;
    writeln;

    write('Max Drop Violations ');
    for Lake := FirstLake to LastLake do begin
      write(MaxDropCtr[Lake]:8);
    end;
    writeln;

    write('Min Inf Violations  ');
    for Lake := FirstLake to LastLake do begin
      write(MinInfCtr[Lake]:8);
    end;
    writeln;

    write('Min Gwh Violations  ');
    for Lake := FirstLake to LastLake do begin
      write(MinGwhCtr[Lake]:8);
    end;
    writeln;

  { report to output file }

    writeln(fo,'Diagnostics report');
    writeln(fo);
    writeln(fo,'Number of valid previous states :
',PastLastCtr-FirstCtr);
    writeln(fo,'Number of state transitions attempted : ',PossibleCtr);
    writeln(fo);
    write(fo,'                      ');
    for Lake := FirstLake to LastLake do with LakeRcd[Lake] do begin
      write(fo,AbbrevName:8);
    end;
```

```
    writeln(fo);

    write(fo,'Previous Elevations ');
    for Lake := FirstLake to LastLake do begin
      write(fo,' ',PrevMinVldElv[Lake]:3,'-',PrevMaxVldElv[Lake]:3);
    end;
    writeln(fo);

    write(fo,'Power Pool Levels   ');
    for Lake := FirstLake to LastLake do begin
      write(fo,' ',LakeRcd[Lake].MinElevation[CyclePeriod]:3,'-',
                   LakeRcd[Lake].MaxElevation[CyclePeriod]:3);
    end;
    writeln(fo);

    write(fo,'Max Rise Violations ');
    for Lake := FirstLake to LastLake do begin
      write(fo,MaxRiseCtr[Lake]:8);
    end;
    writeln(fo);

    write(fo,'Max Drop Violations ');
    for Lake := FirstLake to LastLake do begin
      write(fo,MaxDropCtr[Lake]:8);
    end;
    writeln(fo);

    write(fo,'Min Inf Violations  ');
    for Lake := FirstLake to LastLake do begin
      write(fo,MinInfCtr[Lake]:8);
    end;
    writeln(fo);

    write(fo,'Min Gwh Violations  ');
    for Lake := FirstLake to LastLake do begin
      write(fo,MinGwhCtr[Lake]:8);
    end;
    writeln(fo);

    Close(fo);
  end; {with PossibleState}
end;    { Diagnostics }
{
                      M A I N
}
BEGIN {main}
  GetCommandLineParameters;
  InitializeSystem;

  GetMinInflow;   {for all cycle periods, subject to percent
reliability}

  if (DebugLevel >= 1) then begin
```

```
    writeln;
    writeln(' $$$DEBUG: Generating and evaluating paths between
periods');
  end;

  CyclePeriod := FirstPeriod; {tracks actual cyclical period value}
  for Period := 1 to SimulationPeriods do begin
    if (DebugLevel >= 2) then begin
      writeln('  Computing Period: ',Period:3,' Cycle Period:
',CyclePeriod:3);
    end;

    FirstPtr := PeriodPtr[Period-1];
    PastLastPtr  := PeriodPtr[Period];

    FirstCtr := PastLastCtr;
    PastLastCtr := NextCtr;

    if (FirstPtr = nil) then begin
      writeln('$$$ERROR, Period =',(Period-1):3,
        ' had no valid possible states, execution was suspended!');
      Halt;
    end;
    if (DebugLevel >= 2)
    then writeln(' Previous period states to be searched are from: ',
                Lng2Str(FirstCtr),' -to- ',Lng2Str(PastLastCtr-1));
    with PossibleState do begin

      {for the final simulation period, force min and max lake levels to
       the desired ending levels supplied by the user}
      if(Period = SimulationPeriods) then
        for Lake := FirstLake to LastLake do begin
          LakeRcd[Lake].MinElevation[CyclePeriod]:=EndingLevel[Lake];
          LakeRcd[Lake].MaxElevation[CyclePeriod]:=EndingLevel[Lake];
      end; {for lake}

      {Reset all lakes to bottom, begin cycling 1st lake again}
      for Lake := FirstLake to LastLake do
        Level[Lake] := LakeRcd[Lake].MinElevation[CyclePeriod];
      Done := false;
      repeat  {until done}

        { Compare this possible state to all previous states for minimum
path}
      StatePtr := nil;   {Set after 1st valid path found}
        PreviousStatePtr := FirstPtr;
        while (PreviousStatePtr <> PastLastPtr) do begin
        if (DebugLevel >= 3) then begin
          write('[',Ptr2Str(PreviousStatePtr),',',Ptr2Str(NextPtr),']
');
        end; {if}
        if (PathComputed) then begin
          {Path was valid}
```

```
        if (StatePtr = nil) then begin
          {FIRST VALID PATH TO THIS STATE}
          StatePtr := NextPtr;
          StatePtr^ := PossibleState;
          MinExpectedCost := PossibleState.ExpectedCost;
        end {then}
      else begin
        if (PossibleState.ExpectedCost < MinExpectedCost)
        then begin
            {REPLACE OLD STATE RECORD}
            StatePtr^ := PossibleState;
            MinExpectedCost := PossibleState.ExpectedCost;
          end; {then begin}
      end; {else begin}
    end; {if}
      IncStatePtr(PreviousStatePtr);
    end; {while}

    if (StatePtr <> nil) then begin
      IncStatePtr(NextPtr);  {had a live one!}
      inc(NextCtr);
    end;

  { Increment to next lakes state }
  Lake := FirstLake;
    StillLooping := false;
    repeat
    Level[Lake] := Level[Lake] + 1;
      with LakeRcd[Lake] do begin
        if (Level[Lake] > MaxElevation[CyclePeriod]) then begin
        if (Lake <> LastLake) then begin
          Level[Lake] := MinElevation[CyclePeriod];
          Lake := Lake + 1;
          StillLooping := true;
        end
        else begin
          Done := true;
          StillLooping := false;
          end;
      end
        else StillLooping := false;
      end; {with LakeRcd[Lake]}
    until (not StillLooping);

  until (Done);

end; {with PossibleState}
if (NextPtr = PeriodPtr[Period])
then begin
  writeln('$$$ERROR, Period=',Period:3,
  ' had no valid possible states, execution was suspended!');
  Diagnostics;
  Halt;
```

```
    end
    else PeriodPtr[Period+1] := NextPtr;
    if(CyclePeriod = PeriodsPerCycle)
      then CyclePeriod := 1
      else CyclePeriod := CyclePeriod + 1;
  end; {for Period}

  ReportResults;
  (*
  close(StateFile);
  *)
  close(fo);
  writeln('Normal end of Lake Optimizing Program');

END.  {main}
```

APPENDIX B

INPUT DATA

TABLE ROCK LAKE PHYSICAL CHARACTERISTICS

ELEVATION FEET ABOVE MSL	ELECTRIC CAPACITY MEGAWATTS	ENERGY STORED GWH	VOLUME 1000'S ACRE-FT
881.000	200.000	0.000	1520.500
882.000	202.000	4.096	1548.000
884.000	204.500	12.536	1604.000
886.000	207.500	21.316	1662.000
888.000	212.000	30.456	1721.000
890.000	215.500	39.954	1783.000
892.000	219.500	49.820	1845.000
894.000	222.500	60.071	1910.000
896.000	226.000	70.711	1976.000
898.000	230.000	81.756	2044.000
900.000	230.000	93.212	2114.000
902.000	230.000	105.106	2186.000
904.000	230.000	117.466	2259.000
906.000	230.000	130.294	2335.000
908.000	230.000	143.607	2413.000
910.000	230.000	157.396	2493.000
912.000	230.000	171.688	2575.000
914.000	230.000	186.512	2659.000
915.000	230.000	194.121	2702.000
916.000	230.000	201.863	2745.000
918.000	230.000	217.750	2833.000
920.000	230.000	234.177	2923.000
922.000	230.000	251.177	3016.000
924.000	230.000	268.767	3111.000
926.000	230.000	286.949	3208.000
928.000	230.000	305.753	3308.000
930.000	230.000	325.205	3410.000
932.000	230.000	345.298	3514.000
934.000	230.000	366.045	3621.000
936.000	230.000	387.475	3731.000
938.000	230.000	409.608	3843.000
940.000	230.000	432.468	3958.000
942.000	230.000	456.087	4075.000

BULL SHOALS LAKE PHYSICAL CHARACTERISTICS

ELEVATION FEET ABOVE MSL	ELECTRIC CAPACITY MEGAWATTS	ENERGY STORED GWH	VOLUME 1000'S ACRE-FT
628.500	322.500	0.000	2045.000
630.000	330.000	7.634	2096.000
632.000	335.500	18.125	2166.000
634.000	341.200	28.986	2237.000
636.000	347.000	40.233	2309.000
638.000	352.300	51.879	2384.000
640.000	358.300	63.942	2460.000
642.000	364.000	76.428	2539.000
644.000	369.800	89.336	2618.000
646.000	375.500	102.689	2700.000
648.000	381.000	116.513	2784.000
650.000	387.000	130.819	2870.000
651.000	389.666	138.156	2914.000
652.000	391.000	145.618	2958.000
654.000	391.000	160.919	3048.000
656.000	391.000	176.730	3139.000
658.000	391.000	193.057	3233.000
660.000	391.000	209.912	3329.000
662.000	391.000	227.312	3428.000
664.000	391.000	245.278	3528.000
666.000	391.000	263.836	3630.000
668.000	391.000	283.012	3735.000
670.000	391.000	302.833	3843.000
672.000	391.000	323.304	3953.000
674.000	391.000	344.446	4066.000
676.000	391.000	366.291	4181.000
678.000	391.000	388.841	4299.000
680.000	391.000	412.102	4419.000
682.000	391.000	436.065	4542.000
684.000	391.000	460.745	4667.000
686.000	391.000	486.173	4796.000
688.000	391.000	512.383	4927.000
690.000	391.000	539.407	5060.000
692.000	391.000	567.249	5197.000
694.000	391.000	595.922	5337.000
696.000	391.000	625.449	5479.000
698.000	391.000	656.252	5625.000
700.000	391.000	688.393	5778.000
702.000	391.000	721.537	5933.000
704.000	391.000	755.706	6093.000

STOCKTON LAKE PHYSICAL CHARACTERISTICS

ELEVATION FEET ABOVE MSL	ELECTRIC CAPACITY MEGAWATTS	ENERGY STORED GWH	VOLUME 1000'S ACRE-FT
845.000	40.000	0.000	442.930
846.000	40.000	0.995	459.380
848.000	40.000	3.098	492.280
850.000	40.000	5.354	525.180
852.000	40.000	7.769	565.480
854.000	40.000	10.348	605.780
856.000	40.000	13.097	646.080
858.000	40.000	16.020	686.380
860.000	40.000	19.122	726.680
862.000	40.000	22.416	774.780
864.000	40.000	25.911	822.880
866.000	40.000	29.615	870.980
867.000	40.000	31.548	895.030
868.000	40.000	33.587	919.080
870.000	40.000	37.674	967.180
872.000	40.000	42.055	1024.580
874.000	40.000	46.704	1081.980
876.000	40.000	51.629	1139.380
878.000	40.000	56.837	1196.780
880.000	40.000	62.335	1254.180
882.000	40.000	68.142	1322.580
884.000	40.000	74.277	1390.980
886.000	40.000	80.748	1459.380
888.000	40.000	87.563	1527.780
890.000	40.000	94.729	1596.180
892.000	40.000	102.264	1676.580
894.000	40.000	110.184	1756.980
896.000	40.000	118.497	1837.380
898.000	40.000	127.213	1917.780

TRUMAN LAKE PHYSICAL CHARACTERISTICS

ELEVATION FEET ABOVE MSL	ELECTRIC CAPACITY MEGAWATTS	ENERGY STORED GWH	VOLUME 1000'S ACRE-FT
698.000	184.000	000.000	822.258
700.000	184.000	003.012	907.812
702.000	184.000	006.387	999.133
704.000	184.000	010.187	1097.268
706.000	184.000	014.500	1203.818
707.000	184.000	016.877	1260.698
710.000	184.000	025.069	1448.760
712.000	184.000	031.512	1589.557
714.000	184.000	038.816	1743.256
716.000	184.000	047.034	1910.018
718.000	184.000	056.220	2090.003
720.000	184.000	066.430	2283.370
722.000	184.000	077.740	2490.670
724.000	184.000	090.295	2713.577
726.000	184.000	104.287	2954.490
728.000	184.000	119.924	3215.809
730.000	184.000	137.426	3499.934
732.000	184.000	156.973	3808.441
734.000	184.000	178.689	4141.909
736.000	184.000	202.687	4500.737
738.000	184.000	229.085	4885.327
740.000	184.000	258.002	5296.077
742.000	184.000	289.657	5734.753
744.000	184.000	324.302	6203.440
746.000	184.000	362.123	6703.179
748.000	184.000	403.307	7235.008

CANNON LAKE PHYSICAL CHARACTERISTICS

ELEVATION FEET ABOVE MSL	ELECTRIC CAPACITY MEGAWATTS	ENERGY STORED GWH	VOLUME 1000'S ACRE-FT
590.000	53.300	0.000	295.388
592.000	55.100	1.381	321.337
594.000	57.000	2.885	348.687
595.200	58.000	4.509	362.886
598.000	60.500	6.289	407.674
600.000	62.300	8.203	439.401
602.000	64.100	10.267	472.662
604.000	66.000	12.492	507.522
606.000	67.000	14.884	543.994
608.000	68.300	17.463	582.339
610.000	70.000	20.245	622.607
612.000	70.000	23.238	664.861
614.000	70.000	26.457	709.184
616.000	70.000	29.911	755.629
618.000	70.000	33.609	804.223
620.000	70.000	37.563	854.997
622.000	70.000	41.784	908.007
624.000	70.000	46.283	963.280
626.000	70.000	51.097	1021.019
628.000	70.000	56.249	1081.688
630.000	70.000	61.741	1145.045
632.000	70.000	67.596	1211.226
634.000	70.000	73.834	1280.386
636.000	70.000	80.475	1352.592
638.000	70.000	87.547	1428.055

APPENDIX C

EXAMPLE INPUT FILE

```
OPTIMIZED LAKE LEVELS FOR AUGUST THROUGH JUNE
USING LATEST PRODUCTION COST FORECASTS 8/91
8 11        first month(8/91), number of months to simulate(to 6/92)
 70         percent reliability
1 10        low inflow probability, high inflow probability
915 917     beginning level, ending level, TABLE ROCK
   3   3   3     3   3   3     3   3   3     3   3   3     MaxRise
   3   3   3     3   3   3     3   3   3     3   3   3     MaxDrop
 118 118 118   118 118 118   118 104  70    45  41 114     MaxGWh
   0   0   0     0   0   0     0   0   0     0   0   0     MinGWh
 915 915 915   915 916 917   917 917 917   917 917 915     MaxElev
 905 905 905   905 905 905   905 905 905   905 905 905     MinElev
655 657     beginning level, ending level, BULL SHOALS
   3   3   3     3   3   3     3   3   3     3   3   3     MaxRise
   3   3   3     3   3   3     3   3   3     3   3   3     MaxDrop
 199 199 199   199 199 199   199 199 199   199 199 199     MaxGWh
   0   0   0     0   0   0     0   0   0     0   0   0     MinGWh
 654 654 654   654 656 657   656 656 656   654 654 654     MaxElev
 649 649 649   649 649 649   649 649 649   649 649 649     MinElev
865 867     beginning level, ending level, STOCKTON
   3   3   3     3   3   3     3   3   3     3   3   3     MaxRise
   3   3   3     3   3   3     3   3   3     3   3   3     MaxDrop
  23  23  23    23  23  23    23  23  23    23  23  23     MaxGWh
   0   0   0     0   0   0     0   0   0     0   0   0     MinGWh
 867 867 867   867 867 867   867 867 867   867 867 867     MaxElev
 861 861 861   861 861 861   861 861 861   861 861 861     MinElev
706 706     beginning level, ending level, TRUMAN
   3   3   3     3   3   3     3   3   3     3   3   3     MaxRise
   3   3   3     3   3   3     3   3   3     3   3   3     MaxDrop
  61  61  61    61  61  61    61  61  61    61  61  61     MaxGWh
   0   0   0     0   0   0     0   0   0     0   0   0     MinGWh
 706 706 706   706 706 706   706 706 706   706 706 706     MaxElev
 706 706 706   706 706 706   706 706 706   706 706 706     MinElev
610 606     beginning level, ending level, CANNON
   3   3   3     3   3   3     3   3   3     3   3   3     MaxRise
   3   3   3     3   3   3     3   3   3     3   3   3     MaxDrop
  30  30  30    30  30  30    30  30  30    30  30  30     MaxGWh
   0   0   0     0   0   0     0   0   0     0   0   0     MinGWh
 606 606 606   606 606 606   606 608 606   606 606 606     MaxElev
 598 598 598   598 598 598   598 598 598   598 598 598     MinElev
```

APPENDIX D

EXAMPLE OUTPUT FILE

OPTIMIZED LAKE LEVELS FOR AUGUST THROUGH JUNE
USING LATEST PRODUCTION COST FORECASTS 8/91

Production Cost file name: average.dat
Lake information file name: example.inp

SUCCESS, minimum 11 month expected cost is: 101.118
Reliability = 70 % Inflow probability range : 1-10
First Period = 8

Mo	Pd	TR	BS	ST	TM	CA	Monthly Cost (M$)	Accum. Cost (M$)	Accum. Spilled Energy (GWhs)	Accum. Unserved Energy (GWhs)
8	1	912	652	864	706	604	12.0	12.0	0.0	0.1
9	2	912	653	864	706	604	8.8	20.8	0.0	1.8
10	3	909	650	864	706	603	9.3	30.0	2.7	2.2
11	4	912	653	864	706	603	10.0	40.1	8.0	6.2
12	5	915	654	864	706	603	10.8	50.9	14.4	10.8
1	6	912	653	864	706	600	12.8	63.7	15.9	12.2
2	7	909	650	864	706	600	10.3	74.0	20.7	15.2
3	8	912	653	865	706	602	7.9	81.9	43.6	25.9
4	9	915	654	866	706	604	4.8	86.7	75.1	38.9
5	10	916	656	867	706	606	5.4	92.0	115.5	46.6
6	11	917	657	867	706	606	9.1	101.1	135.7	52.2

BIBLIOGRAPHY

Andersson, Stig and Denis Sjelvgren, "A Probabilistic Production Costing Methodology for Seasonal Operations Planning of a Large Hydro and Thermal Power System," IEEE Transactions on Power Systems, Vol. PWRS-1, November 1986.

Arvanitidis, Nicolaos and Jakob Rosing, "Composite Representation of a Multireservoir Hydroelectric Power System," IEEE Transactions on Power Apparatus and Systems, Vol. PAS-89, February 1970.

Arvanitidis, Nicolaos and Jakob Rosing, "Optimal Operation of Multireservoir Systems Using a Composite Representation," IEEE Transactions on Power Apparatus and Systems, Vol. PAS-89, February 1970.

Bannister, C. and R. Kaye, "A Rapid Method for Optimization of Linear Systems with Storage," Operations Research, March 1991.

Barnard, David T. and David B. Skillcorn. Pascal for Engineers. Boston: Allyn and Bacon, Inc., 1988.

Bean, James C., John R. Birge and Robert L. Smith, "Aggregation in Dynamic Programming," Operations Research, March-April 1987.

Bonaert, Axel P., Ahmed H. El-Abiad and Anitti J. Koivo, "Optimal Scheduling of Hydro-Thermal Power Systems," IEEE Transactions on Power Apparatus and Systems, Vol. PAS-91, January/February 1972.

Bonaert, Axel P., Ahmed H. El-Abiad and Antti J. Koivo, "Effects of Hydro-Dynamics on Optimum Scheduling of Thermo-Hydro Power Systems," IEEE Transactions on Power Apparatus and Systems, Vol. PAS-91, July/August 1972.

Borland International. Turbo Pascal User's Guide, Turbo Pascal Reference Guide. Scotts Valley, California: Borland International, 1988.

Carneiro, A. A. F. M., S. Soares and P. S. Bond, "A Large Scale Application of an Optimal Deterministic Hydrothermal Scheduling Algorithm," IEEE Transactions on Power Systems, Vol. PWRS-5, February 1990.

Carvalho, M. F. and S. Soares, "An Efficient Hydrothermal Scheduling Algorithm," IEEE Transactions on Power Systems, Vol. PWRS-2, August 1987.

Christensen, G. S. and S. A. Soliman, "On the Application of Functional Analysis to the Optimization of the Production of Hydroelectric Power," IEEE Transactions on Power Systems, Vol. PWRS-2, November 1987.

Chang, Shi-Chund, Chun-Hung Chen, I-Kong Fong and Peter B. Luh, "Hydroelectric Generation Scheduling with an Effective Differential Dynamic Programming Algorithm," IEEE Transactions on Power Systems, Vol. PWRS-5, August 1990.

Chang, Yih-Long and Robert S. Sullivan. Quant Systems. Englewood Cliffs, New Jersey: Prentice-Hall, 1989.

Contaxis, G. C. and S. D. Kavatza, "Hydrothermal Scheduling of a Multireservoir Power System with Stochastic Inflows," IEEE Transactions on Power Systems, Vol. PWRS-5, August 1990.

Cook, Thomas M. and Robert A. Russell. Introduction to Management Science. Englewood Cliffs, New Jersey: Prentice-Hall, 1989.

Diaz, Gustavo and Darrell Fontane, "Hydropower Optimization via Sequential Quadratic Programming," Journal of Water Resources Planning and Management, Vol. 115, No. 6, November 1989.

Devore, Jay L. Probability and Statistics for Engineering and the Sciences. Monterey, California: Brooks/Cole Publishing Company, 1982.

Dreyfus, Stuart E. and Averill M. Law. The Art and Theory of Dynamic Programming. New York: Academic Press, 1977.

Duncan, R. A., G. E. Seymore, D. L. Streiffert and D. J. Engberg, "Optimal Hydrothermal Coordination for Multiple Reservoir River Systems," IEEE Transactions on Power Apparatus and Systems, Vol. PAS-104, May 1985.

El-Hawary, Mohamed E. and Gustav S. Christensen, "Functional Optimization of Common-Flow Hydro-Thermal Systems," IEEE Transactions on Power Apparatus and Systems, Vol. PAS-91, September/October 1972.

El-Hawary, Mohamed E. and Gustav S. Christensen, "Extensions to Functional Optimization of Common-Flow Hydro-Thermal Systems," IEEE Transactions on Power Apparatus and Systems, Vol. PAS-92, January/February 1973.

Foufoula, Efi and Peter Kitanidis, "Gradient Dynamic Programming for Stochastic Optimal Control of Multidimensional Water Resources Systems," Water Resources Research, Vol. 24, No 8, August 1988.

Hernandez, H., J. Diaz and G. Sanchez, "Operations Planning of Colombian Hydro-Thermal Interconnected System," IEEE Transactions on Power Systems, Vol. 6, No. 2, May 1991.

Li, Chao-an, Rui Yang and Jing-Yan Zhou, "Stochastic Optimization of Interconnected Multireservoir Power Systems," IEEE Transactions on Power Systems, Vol. 5, No. 4, November 1990.

Lyra, C., H. Tavares and E. Medina, "Optimization with Staircase Structure: An Application to Generation Scheduling," Computers and Operations Research, February 1990.

Lyra, C., H. Tavares and S. Soares, "Modelling and Optimization of Hydrothermal Generation Scheduling," IEEE Transactions on Power Apparatus and Systems, Vol. PAS-103, August 1984.

Nanda, J. and P. R. Bijwe, "Optimal Hydrothermal Scheduling with Cascaded Plants Using Progressive Optimality Algorithm," IEEE Transactions on Power Apparatus and Systems, Vol. PAS-100, April 1981.

Neto, Tristao da A. Aripe, Mario V. F. Pereira and Jerson Kelman, "A Risk-Constrained Stochastic Dynamic Programming Approach to the Operation Planning of Hydrothermal Systems," IEEE Transactions on Power Apparatus and Systems, Vol. PAS-104, February 1985.

Ouchi, Glenn I. "Spreadsheets: Fitting Curves and Plotting Functions with 1-2-3," Computers in Science, November/December 1987, pp. 66-72.

Pereira, M. V. F. and L. M. V. G. Pinto, "Application of Decomposition Techniques to the Mid- and Short-Term Scheduling of Hydrothermal Systems," IEEE Transactions on Power Apparatus and Systems, Vol. PAS-102, November 1983.

Pereira, M. V. F. and L. M. V. G. Pinto, "A Decomposition Approach to the Economic Dispatch of Hydrothermal Systems," IEEE Transactions on Power Apparatus and Systems, Vol. PAS-101, October 1982.

Quintana, V. H. and A. Y. Chikhani, "A Stochastic Model for Mid-Term Operation Planning of Hydro-Thermal Systems with Random Reservoir Inflows," IEEE Transactions on Power Apparatus and Systems, Vol. PAS-100, March 1981.

Ross, Sheldon. Introduction to Stochastic Dynamic Programming. Orlando: Academic Press, 1983.

Sachdeva, S. S., "Bibliography on Optimal Reservoir Drawdown for the Hydroelectric-Thermal Power System Operation," IEEE Transactions on Power Apparatus and Systems, Vol. PAS-101, June 1982.

Saha, T. N. and S. A. Khaparde, "An Application of a Direct Method to the Optimal Scheduling of Hydrothermal System," IEEE Transactions on Power Apparatus and Systems, Vol. PAS-97, May/June 1978.

Sherkat, V. R., R. Campo, K. Moslehi and E. O. Lo, "Stochastic Long-Term Hydrothermal Optimization for a Multireservoir System," IEEE Transactions on Power Apparatus and Systems, Vol. PAS-104, August 1985.

Sjelvgren, Denis, Stig Andersson and T. S. Dillon, "Optimal Operations Planning in a Large Hydro-Thermal Power System," IEEE Transactions on Power Apparatus and Systems, Vol. PAS-102, November 1983.

Stedinger, J., B. Sule and D. Loucks, "Stochastic Dynamic Programming Models for Reservoir Operation Optimization," Water Resources Research, Vol 20, No. 11, November 1984.

Soares, S., C. Lyra and H. Tavares, "Optimal Generation Scheduling of Hydrothermal Power Systems," IEEE Transactions on Power Apparatus and Systems, Vol. PAS-99, May/June 1980.

Tejada-Guibert, J. Alberto, Jery R. Stedinger and Konstantin Staschus, "Optimization of Value of CVP's Hydropower Production," Journal of Water Resources Planning and Management, Vol. 116, No. 1, January 1990.

Tong, S. K. and S. M. Shahidehpour, "Hydrothermal Unit Commitment with Probabilistic Constraints Using Segmentation Method," IEEE Transactions on Power Systems, Vol. PWRS-5, February 1990.

Tong, S. K. and S. M. Shahidehpour, "An Innovative Approach to Generation Scheduling in Large-Scale Hydro-Thermal Power Systems with Fuel Constrained Units," IEEE Transactions on Power Systems, Vol. PWRS-5, May 1990.

Trezos, Thanos, "Integer Programming Application for Planning of Hydropower Production," Journal of Water Resources Planning and Management, Vol. 117, No. 3, May/June 1991.

Viramontes, F. A. and H. B. Hamilton, "Optimal Long Range Hydro Scheduling in the Integrated Power System," IEEE Transactions on Power Apparatus and Systems, Vol. PAS-97, January/February 1978.

Wan, S. H. and R. E. Larson, "Marginal Cost Method for Deterministic Hydro Scheduling," IEEE Transactions on Power Apparatus and Systems, Vol. PAS-103, June 1984.

Weiner, Dan and Arie Ben-Zvi, "A Stochastic Dynamic Programming Model for the Operation of the Mediterranean-Dead Sea Project," Water Resources Research, Vol. 18, No. 4, August 1982.

Winston, Wayne L. Operations Research: Applications and Algorithms. Boston: Duxbury Press, 1987.

Wood, Alan J. and Bruce F. Wollenberg, Power Generation, Operation, and Control. New York: John Wiley and Sons, 1984.

Yakowitz, Sidney, "Dynamic Programming Applications in Water Resources," Water Resources Research, Vol. 18, No. 4, August 1982.

Yeh, William, "Reservoir Management and Operations Models: A State-of-the-Art Review," Water Resources Research, Vol. 21, No. 12, December 1985.

Zhuang, F. and F. D. Galiana, "Unit Commitment by Simulated Annealing," IEEE Transactions on Power Systems, Vol. PWRS-5, February 1990.

VITA

Duane David Highley was born April 23, 1961 in Highland, Illinois. He received his primary education at various schools throughout the midwest, and completed his secondary education in Springfield, Missouri. Following High School, he was awarded a National Merit Scholarship and enrolled in the pre-engineering program at Southwest Missouri State University in Springfield. Upon completion of this program, he transferred to the University of Missouri-Rolla. While there, he was awarded a UMR Power Engineering Scholarship. He earned a Bachelor of Science degree in Electrical Engineering in May 1983, graduating magna cum laude as a divisional honors student in the Electrical Engineering department.

Following graduation, Duane was employed by Associated Electric Cooperative, Inc. in Springfield, Missouri, where he currently holds the position of Senior Planning Engineer. He is responsible for resource planning, fuel management, and financial forecasts. Duane received the cooperative's "Communicator of the Year" award in 1987 for his presentations to various audiences.

Duane became a registered professional engineer in Missouri in August, 1988. He is a member of Eta Kappa Nu, IEEE, and MSPE.

Duane returned to graduate school in 1986. Since then, he has co-authored papers on resource planning for the Pennsylvania Electric Association (1989), the American Power Conference (1990), and the IEEE Computer Applications in Power magazine (1991).

Duane has been married since 1982 and has three children.