Network Stochastic Programming for Valuing Reservoir Storage

by

John David Emmert

B.A. Mathematics: Civil Engineering, Carroll College, 2002

A thesis submitted to the

University of Colorado in partial fulfillment

of the requirements for the degree of

Master of Science

Department of Civil, Environmental, and Architectural Engineering

2005

This thesis entitled:

Network Stochastic Programming for Valuing Reservoir Storage

written by John D. Emmert

has been approved for the Department of Civil, Environmental, and

Architectural Engineering

_____

Balaji Rajagopalan

_____

Edith Zagona

Date_____

The final copy of this thesis has been examined by the

signators, and we find that both the content and the form

meet acceptable presentation standards of scholarly work in

the above mentioned discipline.

Emmert, John (M.S., Civil, Environmental, and Architectural Engineering)

Network Stochastic Programming for Valuing Reservoir Storage

Thesis directed by Assistant Professor Balaji Rajagopalan

The stochastic nature of hydrologic inflows complicates the simultaneous optimization of hydropower for a multi-stage, multi-reservoir system. The expected value of hydropower must be simultaneously optimized over all time steps and scenarios. Previous research (Stochastic Programming with Recourse) has represented alternative inflow scenarios with a tree structure that branches after each time step and grows exponentially. This research replaces the tree structure with a network structure that grows only linearly as the number of time steps modeled increases. Benders Decomposition allows both the tree structure and the network structure to be optimized as a series of subproblems - one for each arc in the scenario structure. Using this network representation, an 8-week network stochastic programming model of the Tennessee River Basin converged rapidly to an upper bound on hydropower value. This research also provided preliminary results for a lower bound calculation as well as a goal programming solution. The network structure may also be a suitable replacement for the tree structure in stochastic applications outside of hydropower optimization.

Acknowledgements

The completion of this thesis would not have been possible without the support of many people. I would like to thank my advisors, Balaji Rajagopalan, Edith Zagona and Ken Strzepek for their feedback and support. I would also like to especially thank Tim Magee for his guidance and devotion of time throughout this project.

I would like to acknowledge the Tennessee Valley Authority (TVA) for providing funding for this project. Specifically, I am grateful to Suzanne Biddle for her communication, feedback and willingness to provide data regarding the Tennessee Valley when needed.

In addition, I would like to thank the Center for Advanced Decision Support in Water and Environmental Systems (CADSWES) for also providing funding and the use of their resources. I am indebted to all of the staff at CADSWES for their technical advice and support.

Finally, I would to thank my wife Anna, as well as the rest of my family, for enduring with me throughout this project and always being there for encouragement.

Contents

Tables

Figures

Chapter 1

INTRODUCTION

Reservoirs are operated to serve two primary purposes: (1) to generate hydroelectric power, and (2) to manage natural fluctuations in flows. The benefits of managing natural fluctuations in flows are many, including provision of water supplies, water quality management, flood control, drought management, recreational use, navigation, and many others. Organizations may operate their reservoir systems differently, but these two purposes lie at the center of most operational policies. For example, the U.S. Army Corps of Engineers (USACE) has played a dominant role in constructing and operating major reservoir systems for navigation and flood control, while the U.S Bureau of Reclamation (USBR) water resources program was founded upon demands for irrigation and hydroelectric power (USACE, 1991). The Tennessee Valley Authority (TVA) operates as part of the TVA Act passed by Congress in 1933. According to this act, TVA's reservoir system must be used to "… regulate the streamflow primarily for the purposes of promoting navigation and controlling floods," and so far as may be consistent with such purposes, "…for the generation of electric energy…" (Shane and Gilbert, 1982). Although these organizations each operate under differing policies, they share the similar purpose of generating hydroelectric power and managing the natural fluctuations of flows.

Hydroelectric power is one of the most reliable, efficient, and economic renewable energy resources available (www.tva.gov). However, hydroelectric power is also a limited resource. The quantity of water available for hydroelectric production depends upon limited amounts of rainfall or snowmelt, and water stored in hydroelectric reservoirs must be managed accordingly. Reservoirs serve to manage natural fluctuations in flows by controlling releases and storing water during wet periods and utilizing stored water during dry periods. The most extreme scenarios of wet and dry periods come in the form of floods and droughts, and reservoirs serve to minimize the impacts of these events and create a dependable water and power supply.

In most basins, the majority of water that flows into the reservoir system arrives seasonally. For example, in most of the basins in the Rocky Mountains, approximately 70 percent of the annual inflow into a basin occurs during April, May, June, and July due to snowmelt runoff (Regonda, 2004) with very little contribution coming from storm events throughout the remainder of the year. In other basins, such as the Yakima basin in Washington, the inflow may occur bi-annually during spring and fall wet seasons (Stapleton, 2004). Regardless of the timing of the inflow events, the demand for water as a source of electricity is independent of runoff seasons, so reservoir operators attempt to manage these inflows in such a manner that hydroelectric power demands can be met throughout the year.

Reservoir operators are aware of these fairly dependable cycles and operate their reservoir systems to account for these cycles. According to Gilbert (1985), TVA operates their reservoir system on an annual cycle. In the Tennessee River basin, the

winter and spring seasons are wet and have a higher danger of flooding. To account for this, the reservoirs are drawn to their lowest levels in December in preparation for the winter flood season. In January, TVA begins to fill their reservoirs gradually, but maintains relatively low storage levels, except during a flood, until the spring fill season. Then, beginning in April, TVA begins to aggressively fill the reservoirs and realizes the highest reservoir levels in late spring or early summer. These high reservoir levels are managed throughout the dry summer and fall seasons to supplement the natural streamflows for navigation, power production, and water supply.

For long periods of drought that may occur in a basin, reservoir operators attempt to store enough water so that demands can be met throughout the duration of the drought. If reservoirs are allowed to get to a point where the water supply is not enough to meet power demand, less flexible and more expensive power sources must be utilized. By operating reservoirs effectively, reservoir operators are able to provide a safe and dependable water and power supply, regardless of inflow conditions.

1.1 Hydropower as a Flexible Power Supply

The control and flexibility associated with hydroelectric power plants vary greatly from reservoir to reservoir, but all offer an excellent alternative or compliment to power supplies such as fossil fuel and nuclear power plants. According to Jacobs et al. (1995), hydroelectric plants are operated in a range of manners as follows. *Storage* hydro plants are the most common and flexible type of hydroelectric plant. Storage hydro plants are large, dammed reservoirs that are equipped with relatively

easily adjustable gates to control the flow of water through turbines. On the other end of the spectrum are *run-of-the-river* reservoirs that store little water and generally operate as necessitated by current streamflow conditions. Between these two extremes are a variety of hydro plants whose operations are being controlled differently for a wide variety of reasons.

Jacobs et al. (1995) explain that in many basins, pumped storage hydro plants can be used to increase peak energy production without an increase in basin inflows. Pumped storage units operate by pumping water into an uphill reservoir during periods of low power demand, making the water available to be released for generation during peak generation periods. These plants provide an economical approach to storing electricity.

In addition to the variety of hydro plants, many power companies also operate fossil fuel and nuclear plants to generate electricity and use hydropower as a supplement to these power supplies. For example, the Pacific Gas and Electric Company (PG&E) generates power from a variety of sources including fossil-fueled plants, hydroelectric powerhouses, and one nuclear power plant (www.pge.com). TVA operates 11 fossil fuel plants, 6 nuclear plants, 29 hydroelectric dams, and one pumped storage unit (www.tva.gov). In 2003, fossil fuel plants provided 60 percent of TVA's total power output, nuclear plants produced 29 percent, and hydroelectric plants produced 11 percent (www.tva.gov).

The fossil fuel and nuclear plants in these systems are very important to the power production of the basins but provide a less flexible source of energy. Fossil fuel and nuclear plants generally have slow ramp up times and are very inefficient in

varying power output if fluctuations in power demand arise. For this reason, these sources tend to provide a base level of power, with limited flexibility. In contrast, hydropower plants, particularly those associated with large reservoirs, are used to meet fluctuations in demand, unless a reservoir is in, or near, flood control. Flexibility in hydroelectric pants means they can supply electricity for power demand on relatively short notice and, unlike fossil fuel or nuclear power, can be brought on line almost instantaneously (Gilbert, 1985). Thus, for utilities such as PG&E and TVA, hydropower provides a very attractive way to smooth out the daily peaks in the power load and to provide a readily available reserve to be used in the event of extreme load increases or a loss of generating capacity elsewhere in the system (Gilbert, 1985). It would be very expensive and inefficient to attempt to meet these demands with fossil fuel or nuclear power. In fact, according to Wunderlich (1989), $1 per megawatt-hour (MWh) in hydroelectric operation cost can replace $20 to $90 per MWh power generation cost from other power sources. This makes hydropower a very attractive source of electricity.

1.2 Variations in Power Value

One of the issues associated with the operation of hydroelectric power plants is the fact that the economic value of hydropower is not constant. The value of hydropower is a function of demand and supply, and the goal of hydro plant operators is to generate power during peak demand when hydropower is most valuable. According to Jacobs et al. (1995), PG&E attempts to obtain the maximum value from its hydroelectric resources by minimizing spills and timing hydroelectric generation so that electricity is produced when it is most valuable. It is also necessary to

maintain enough water in storage to capture the flexibility benefits described earlier. This strategy is not unique to PG&E's operation and is, essentially, the strategy of any hydro plant that wishes to operate its reservoir system with consideration to the economic value of their hydroelectric power.

To some degree, the variation in power demand is predictable; however, there are certainly factors that make this demand uncertain. Seasonally it is expected that during the hot summer months, more residential, industrial, and commercial sites will be operating air conditioning units. This high demand for power makes hydropower very valuable. However, during these hot summer months there are often periods of cooler weather, which typically are accompanied with storm events. During these events the air temperature is cooler and the use of air conditioning decreases, causing less demand for power. In the cold winter months, there is a high demand for power for heating purposes. In some basins, such as the Tennessee River Basin, air temperature increases before storm events causing less demand for heating power. These physical patterns typically cause inflows to arrive when power is not in highest demand. When the supply of available water in the system is increasing, the demand for hydropower is lower. In this sense, power demand amplifies the effects of variations in water supply.

The demand for power on a weekly timescale is also somewhat predictable. Generally, more power is required during the week when the majority of commercial and industrial entities are operating at full capacity. During the weekend, when a large number of businesses are not operating, the demand for power is much less. On a daily timescale, the demand for power fluctuates in a similar manner. During

certain periods of the day there may be great demand for industrial and commercial use and at other times, generally mornings and evenings, there may be more residential use. During the night when most people are sleeping, there is much less demand for power. However, there are situations on these time scales that make the variation in power demand difficult to predict. For example, there can be substantial fluctuations in temperature over the course of a day from storms, cloud cover, or any number of meteorological reasons. Also, it is quite possible for unforeseen outages to occur somewhere in the system, which will increase demand significantly in other areas of the system.

Along with being a function of demand, the value of water is affected by other factors. Many power companies purchase electricity from outside suppliers. If the market value for electricity is cheap, the benefit of producing electricity internally through hydropower is not as great. It may be more economical to save the water in storage for use when market values are higher. The amount of water available in the system also plays a significant role in determining the value of water. The long-term value of water is both a function of water available in storage and uncertain hydrologic inflows in the future. Most hydro plant operators attempt to quantify the value of water in some manner, and operations models serve as valuable tool for achieving that goal.

1.3 Modeling for Operation Planning

In an effort to maximize the value of hydropower, many hydropower companies have developed models to schedule their hydropower operations. Gilbert (1985) and Wunderlich (1989) describe a series of models that TVA developed to

plan the operation of a large system of reservoirs. The goal of these models is more efficient use of hydropower, better strategic planning, and policy formulation (Gilbert, 1985). Gilbert describes two primary models developed by TVA; known as the *Economy Guide Development Model* and the *Weekly Scheduling Model*. The Economy Guide Development Model was proposed as a long-term planning model to derive economy guide curves of the expected future power generation cost (Gilbert, 1985). The Weekly Scheduling Model was designed to be run weekly as an aid to real time operation (Gilbert, 1985). Jacobs et al. (1995) developed a long-term model for PG&E called *Stochastic Optimal Coordination of River-basin and Thermal Electric Systems* (SOCRATES). [An extensive review of multiple purpose modeling and analysis approaches is provided in USACE (1991).] These models serve as a tool for hydro-plant operators to analyze outcomes of operation decisions prior to making any release decision.

1.4 Generate Now or Save for Later

When considering the variations in the value of hydropower, reservoir operators must answer the following question: Is it better to generate hydroelectricity now or save the water to release later when the value of power may be greater? This decision must be made within the context of other operating criteria such as maintaining minimum or maximum storage levels, minimum flow requirements, recreation considerations, etc. For example, operating objectives for the TVA basin, as presented by Shane and Gilbert (1982) include:

- Flood Prevention
- Navigation

- Water Supply
- Power Generation
- Water Quality Releases
- Recreation
- Reservoir Balancing

While this list is specific to the TVA basin, it is a good indication of the types of objectives that must be considered for reservoir management in any basin.

Another issue that makes this operation decision difficult is the fact that the entire basin is interconnected, so releases made at upstream reservoirs will have an effect on downstream reservoirs. In the Tennessee River basin, the farthest upstream reservoirs, those in the mountainous regions, are the largest and operate as storage hydro plants. The reservoirs farther down in the basin can be smaller and operate more closely to run-of-the-river-reservoirs. In these situations, large releases made at upstream reservoirs may cause spill in the lower reservoirs, which could result in loss of potential hydropower production. In other areas such as the Colorado River, exactly the opposite situation exists. There are extremely large reservoirs in the downstream areas, and a relatively small amount of water is stored upstream. The fact that typical systems, such as those on the Colorado and Tennessee River) are largely mismatched, reservoir decisions cannot be managed independently, and it is necessary to develop a system-wide release policy.

With all of these considerations, it can be difficult to decide if power should be generated now or later. If the short-term economic benefits of hydropower production are only considered, the maximum amount of water will be released,

leaving very little remaining water for future operation. This occurs because hydropower production is the most economical source of power and, if the long-term operation is not considered, stocks of potential hydroelectric energy are depleted now for their immediate economic benefit. In this scenario, if low inflow volumes occur, it then may be necessary to use more expensive thermal generation or potentially not meet demand requirements (Pereira, 1989). Instead, short-term use of water for hydropower generation must trade off against the value of using the water in the future. A common approach to quantify this tradeoff is to separate the short-term and future values.

The short-term value of water is relatively certain and is based on the market values or on a reduction in the cost of alternative power generation (e.g. fossil fuel or nuclear power to meet demand). Also, in the short-term, load requirements, reservoir levels, and inflows into the system are known with some certainty making the value of water at the current time easier to determine. However, the future value of water is less certain because future inflows to the system are not known. The future inflow into the system affects the volume of water available in storage. The value of water in storage is essentially a function of supply and, in general, as reservoir storage increases, the incremental value of additional stored water decreases. For example, if inflow values are low, which has the effect of lowering reservoir levels, water will be allocated for use only during peak generation periods. In contrast, if inflow values are high, more water will be allocated for use during off peak generation, when water is less valuable. Also, during periods of high inflow, the water in storage is more likely to be spilled which provides no economic benefit.

Although greater storage values cause the incremental value of water to decrease for the reasons mentioned above, this is partially offset by the fact that more power is produced per unit of water released as reservoir elevation increases. This actually causes a small increase in the marginal value of water at high reservoir elevations. Together, however, these factors still cause the incremental value of additional water to decrease as storage increases, but the change is relatively slow and the curvature of value as a function of storage is relatively small. Figure 1 shows a schematic exaggeration of this curve, and Figure 2 shows a to scale representation of this curve for a TVA reservoir using their current equations for estimating the future value of stored water.

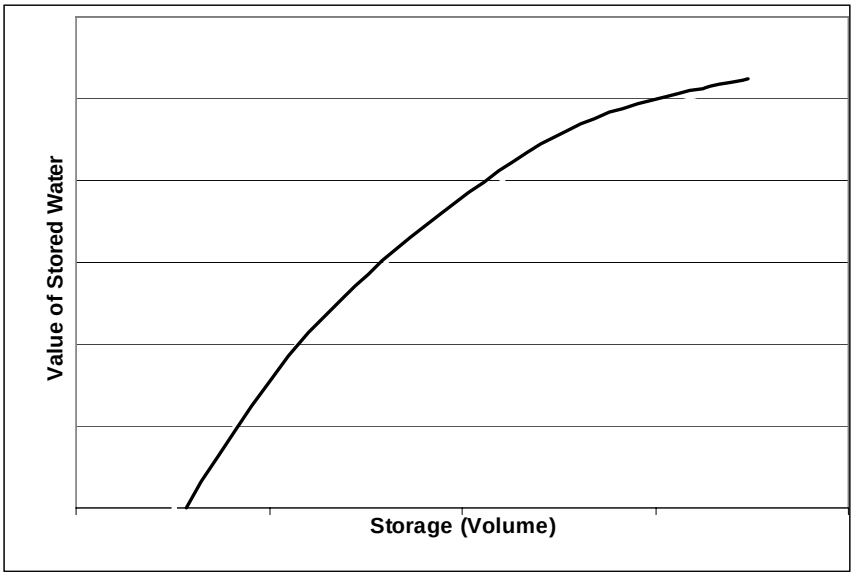

Figure 1: Value of Water vs. Storage

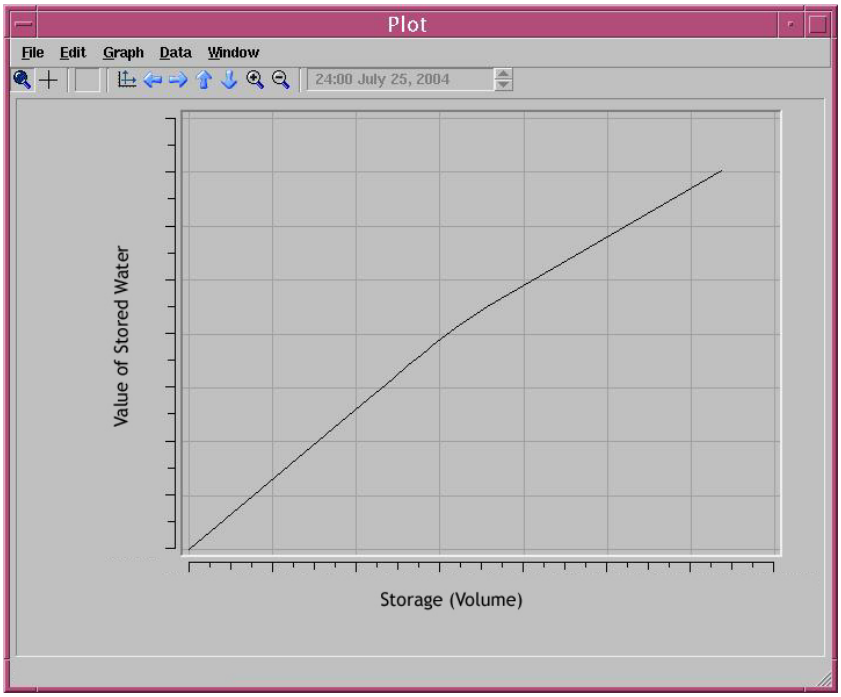

Figure 2: Value of Water vs. Storage

1.5 Stochastic Nature of Inflow

Because the future value of water depends, in part, on uncertain future hydrologic inflows, operations planning models must account for this uncertainty. In order to account for these future inflow values, a hydroelectric schedule must anticipate future inflow values and hedge against the consequences of extreme streamflow scenarios (Jacobs et al., 1995). Some models use a deterministic approach, in which expected streamflow value is used and others may use an ensemble of possible hydrologic inflow values that are referred to throughout this paper as hydrologic scenarios.

One of the major drawbacks to using the expected streamflow values in a deterministic model is that these models do not hedge against the consequences of extreme streamflow scenarios. Deterministic models underestimate the economic effects at the extremes and produce an "optimistic" operation schedule that, in the case of having an extreme inflow scenario, can lead to high economic losses (Pereira and Pinto, 1985). Prior to the development of the SOCRATES model, PG&E used such a deterministic approach but PG&E, as well as many other hydroelectric operators, recognized these shortcomings and began to implement models that incorporate stochastic streamflow scenarios.

In contrast to deterministic models, which generate operation schedules based on perfect knowledge of the future, stochastic models consider a number of possible inflow scenarios that could potentially occur (Watkins et al., 1999). Because it is impossible to have perfect forecasts for the future inflow scenarios, there is an inherent uncertainty in the future inflow values making them stochastic (Pereira and Pinto, 1985). Models that consider stochastic scenarios produce a more conservative operation schedule because the effects of the extremes are considered. The result is that reservoir operating levels tend to remain more toward the middle of standard operating conditions so that extreme streamflow scenarios can be managed more economically.

Along with the inherent uncertainty of future streamflow conditions, there is also some correlation of streamflows from one time period to the other. For a variety of physically explainable reasons, there is a correlation between current and previous inflows at a range of timescales. Watkins et al. (1999) show that model results can be

improved by using a scenario generation technique that preserves this correlation of inflows. The correlation of inflows means that wet conditions currently are more likely to lead to wet conditions in the future and dry conditions are more likely to lead to dry conditions. This correlation can be due to localized climatic high-pressure systems, large-scale climate conditions, or the persistence of base flow in the basin.

On a short time scale, it is somewhat possible to predict the upcoming weather in a basin and have an idea for the amount of moisture that will fall in the upcoming weeks. It is fairly easy to track high-pressure systems and know what kind of effect these systems will have when they arrive in any given basin. Localized storm systems also contribute to the runoff and recharge of the river basin.

On the annual time scale, there are large-scale indicators, such as El Nino or the Southern Oscillation Index that can provide an indication of the amount of moisture that may fall in the upcoming year (Grantz, 2003). Studies have shown that the strength of the signal received from these indicators gives some ability to predict inflows in the upcoming water year and in some cases can provide fairly accurate forecasts (Grantz, 2003). In the southeastern United States, these signals can also give an indication of the severity of the upcoming hurricane seasons. Severe hurricane seasons can bring a tremendous amount of water to a basin and severely affect reservoir operations.

The amount of water that has arrived in the system previously can also affect the base inflow in a river basin. Base inflow is generally a function of prior precipitation, which can persist for long periods of time. If the previous year happened to be an extremely wet year, streamflow values may remain fairly high due

to base inflow, even if the current year is somewhat dry. Also, storm events may produce greater streamflow values because less water is being absorbed into the soil. The opposite may occur in dry situations where a storm event mostly recharges soil moisture and very little water actually reaches the hydroelectric system.

The fact that future inflow values are uncertain, with some ability to be forecast, as well as correlated from time period to time period, means that for developing long-term operation plans, it is necessary to account for these characteristics. To do so, models have been developed that incorporate these characteristics. The inflow scenario frameworks of these models are typically known as *multi-stage stochastic networks*.

1.6 Solving as a Multi-stage Stochastic Network

Most long-term planning models attempt to optimize the reservoir scheduling operations. Often the goal is either to determine a generation schedule for each plant in a system that minimizes the expected operating cost along the planning period (Pereira and Pinto, 1985) or to maximize the avoided cost of using more expensive forms of energy such as market purchases, fossil fuels, or nuclear power (Biddle, 1999). Due to the fact that these models incorporate future inflows, uncertainty in streamflow values is generally considered.

Optimizing reservoir scheduling with regard to uncertainty in inflows is generally classified as Stochastic Optimization. Historically, two approaches have dominated the stochastic optimization of reservoirs: Stochastic Dynamic Programming (SDP) and Stochastic Programming with Recourse (SPR). The literature is not always clear about this distinction, and the titles of articles can be

misleading. Where Birge (1997) uses the terminology SPR as described here, Pereira (1989) and Pereira and Pinto (1985, 1991) use the terminology Stochastic Dual Dynamic Programming (SDDP), which is easily confused with SDP.

In applying these stochastic optimization methods, the problem is broken down into a multi-stage stochastic network. This multi-stage network is divided into time periods, or stages (e.g. weekly, monthly or annually), and possible streamflow scenarios. Each branch in the network represents a potential streamflow scenario. At each time period, any number of branches can be considered at each transition. Figure 3 shows a representation of this multi-stage stochastic network in which three possible scenarios (high (h), normal (m), and low (l) flow) are considered at each stage. This figure represents the traditional structure in the SPR methodology but is similar to that of SDP.

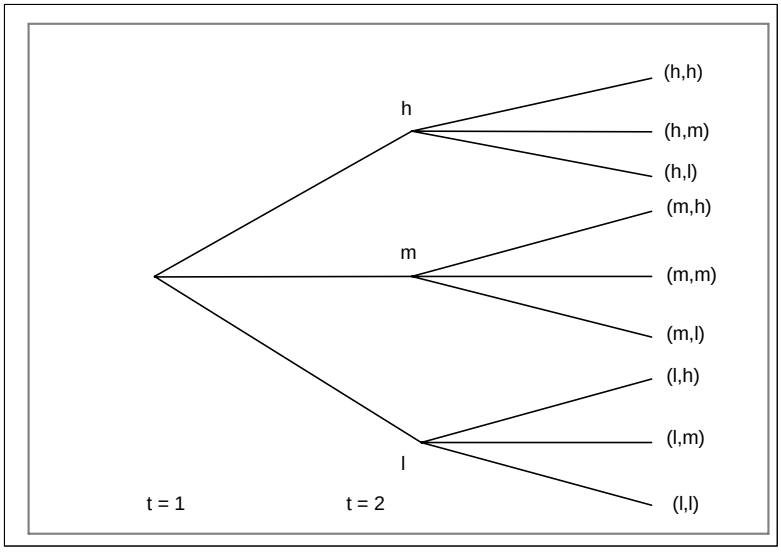

Figure 3: Multi-stage Stochastic Network

A large share of the research done in long-term planning models has been SDP. A typical SDP problem represents the "state" of a reservoir system by a vector of possible reservoir storage levels, and the "state space" is discretized into a finite number of storage values (Pereira and Pinto, 1985). This technique has proven successful for systems that have a small number of reservoirs, but has been more difficult to implement as the number of reservoirs increases. Bellman (1952) ascribes this situation to the well known "curse of dimensionality" that limits the application of dynamic programming in general. Many methods have been developed to circumvent this limitation in SDP [References to these methods can be found in Stedinger et al. (1984), Tejada-Guibert et al. (1995) and Labadie (2004)] which are discussed in more detail later in this paper. In general, these techniques have increased the number of reservoirs that can be modeled but have not eliminated the curse of dimensionality.

In contrast, SPR has received relatively less application for reservoir scheduling. With SPR, it is possible to decompose the multiple-period, multiple-scenario problems into smaller one period subproblems that can be solved more easily. For example, in Figure 3, one problem can be solved for each streamflow scenario with some information sharing between stages. The information that is generally shared between stages is reservoir levels and information to generate approximations of the future value of water. These approximations of the future value of water are generated using shadow prices obtained from the solutions to the optimization problem and are interpreted as Bender's (Bender, 1962) in a stochastic multi-stage decomposition (Pereira and Pinto, 1991). By using these piecewise

approximations of the future value of water, the application of SPR is less sensitive to the number of reservoirs than is an SDP algorithm because with SPR it is not necessary to break down the network into a finite number of states (Pereira and Pinto, 1991). However, the difficulty in applying SPR has been that the number of possible scenario combinations increases exponentially with the number of time periods modeled (Pereira and Pinto, 1985; Jacobs et al., 1995).

This thesis presents a proposed methodology, called *Network Stochastic Programming* (NSP), to deal with the shortcomings of SDP and SPR. The methodology is developed within the SPR framework and limits the combinatorial explosion of scenarios as the number of time periods increases. The concepts of SDP and SPR, including the benefits and shortcomings of each, are explained in more detail. Case study analyses of the TVA Tennessee River basin are presented followed by conclusions and discussions of the results from application of the proposed methodology.

Chapter 2

PREVIOUS WORK IN STOCHASTIC PROGRAMMING

The objective of stochastic programming algorithms, both SDP and SPR, is to maximize the current period objective function plus the expected value of future stages. Stochastic programming algorithms are typically separated into stages, which subdivide the problem by time period, and possible streamflow scenarios. The availability of limited amounts of hydroelectric energy, in the form of stored water in the system, makes the operational problem very complex because it creates a link between an operation decision in a given stage and the future consequences of this decision (Pereira and Pinto, 1985). Stochastic Dynamic Programming and Stochastic Programming with Recourse are two techniques that have been developed to solve such problems.

2.1 Stochastic Dynamic Programming

This section, which describes the stochastic dynamic programming approach to water resource operation planning and the motivation behind developing models using the stochastic programming with recourse approach, is not required reading to understand the technique being developed in this thesis. The organization of the discussion begins with a general explanation of dynamic programming, followed by a

simple mathematical description of the SDP algorithm, a survey of work that has been done using the SDP algorithm, and ending with a discussion of the motivation to find improved techniques.

2.1.1 *General Characteristics of Dynamic Programming Problems*

Hillier and Lieberman (2001) describe dynamic programming as a useful mathematical technique for making a sequence of interrelated decisions and for systematically determining the optimal combination of decisions. The dynamic programming approach involves decomposing a complex problem into a series of simpler problems that are solved sequentially, while transmitting information from one stage of the problem to the next using state concepts (USACE, 1991).

The general characteristics of dynamic programming problems are (USACE, 1991):

First, the problem can be divided into stages, with a policy decision required at each stage. In hydroelectric systems, the stages typically represent different points in time (i.e., determining reservoir releases for each time interval), but can also represent different points in space (i.e., releases from different reservoirs), or different activities (i.e., releases for different project purposes or water users).

Second, each stage has a finite number of states associated with the beginning of that stage. In general, the states are the various possible conditions in which the system might be at in that stage of the problem. In hydroelectric systems, if there is no correlation between current flows and previous flows, reservoir storage is a useful state. However, because there is generally a correlation of flow between time periods, at least two classes of state variables should be included: the reservoir

storage level and some information about the “hydrologic trend” in the system (Pereira and Pinto, 1991).

A third characteristic of dynamic programming algorithms is that the effect of a policy decision of each stage of the problem is to transform the current state to a state associated with the beginning of the next stage. For example, if the policy decision is how much water to release from the reservoir at the current stage, this decision transforms the amount of water stored in the reservoir from the current amount to a new amount in the next stage.

Another characteristic of dynamic programming is that a return function indicating the benefit of the transformation is associated with each potential state transformation. For example, in hydroelectric systems the future value function may be the value of energy in storage. The future value function is calculated for discretized values of storage. Figure 4 illustrates this calculation for the future value function. The horizontal axis is the discretized storage levels, and the vertical axis is the function characterized by the set of corresponding future values.

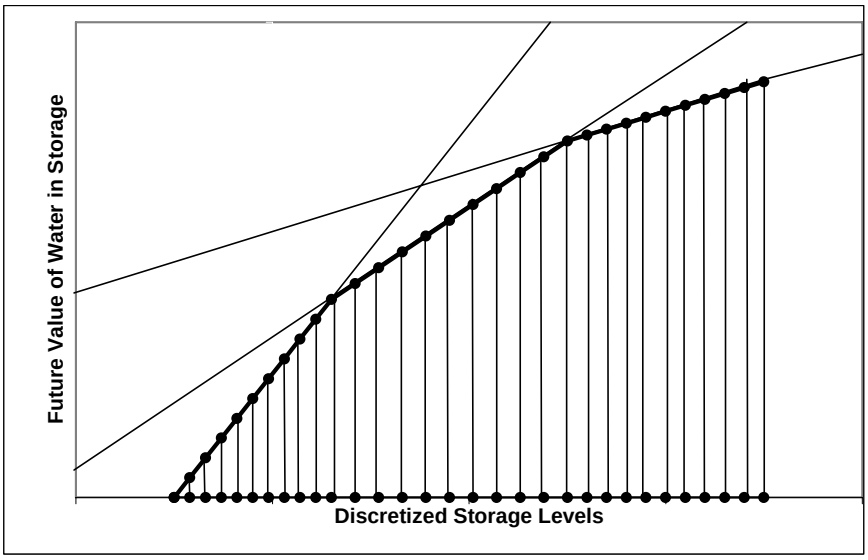

Figure 4: Discretization Scheme in Dynamic Programming

Hillier and Lieberman (2001) explain a fifth characteristic of dynamic programming that is really what makes these algorithms so attractive. This characteristic is what is known as the principle of optimality for dynamic programming. This principle states that given the current state, an optimal policy for the remaining stages is independent of the policy decisions chosen in previous stages. Therefore, the optimal immediate decision depends on only the current state and not how that state was arrived at. The state encompasses all prior relevant information from previous solutions, and the problem only needs to be optimized from that state forward.

These characteristics of dynamic programming problems provide a way to reduce a large, complicated problem to a series of smaller, tractable problems if the state space is relatively small. Because the number of discretized states is limited, as

the number of stages being examined increases, the problem does not grow too rapidly. This element is very beneficial because it allows hydroelectric models that have a large number of timesteps to be solved. A visual representation of an SDP network is shown in Figure 5.

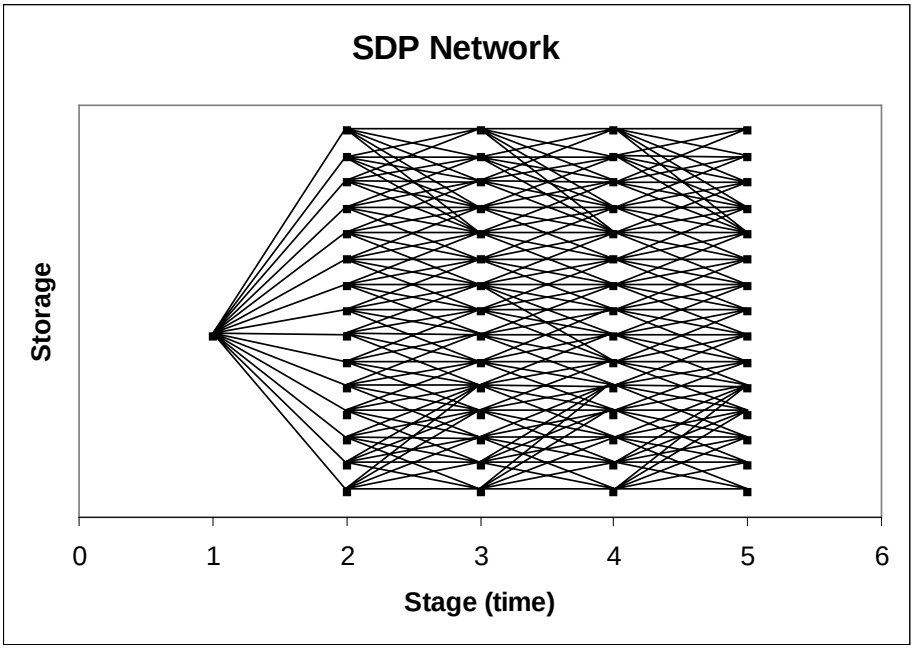

Figure 5: SDP Network

2.1.2 *Mathematical Representation of SDP Algorithms*

For explanatory purposes, suppose that the objective is to maximize the expected sum of the contributions for each stage. Assuming the simplest case in which no hydrologic state is considered, the stochastic dynamic recursion function would be of the form (Tejada-Guibert et al., 1995):

$$f_t(X_t) = E_{Q_t}\{\max_{R_t}[B_t(X_t, Q_t, R_t) + f_{t+1}(X_{t+1})]\} \qquad \textbf{(Eq. 1)}$$

subject to

$$\max[X_{i,t} + Q_{i,t} - X_{i,t}^{\max}, 0] \le R_{i,t} \le X_{i,t} + Q_{i,t} - X_{i,t}^{\min}$$

$$\forall t = 1, \ldots, T-1, T$$
$$\forall i = 1, \ldots, n-1, n$$

where

n = number of reservoirs

T = number of stages in planning horizon

t = current stage in planning horizon

Q_t = inflow into the system in period t

X_t = vector of storages at the beginning of stage t

R_t = vector of target releases during stage t

X_{t+1} = vector of ending storage in stage t ($X_{i,t+1} = X_{i,t} + Q_{i,t} - R_{i,t}$ for i = 1,…n)

$B_t(X_t, Q_t, R_t)$ = benefit from system operation in the current stage t

$F_t(X_t)$ = expected future return from the optimal operation of the system from stage t to the end of the planning horizon given that the system begins in period t in state (X_t)

2.1.3 *Survey of SDP Models and Enhancements*

Loucks et al. (1981), present Stochastic Dynamic Programming as an effective technique for plan formulation and evaluation in water resources systems that contain inherent uncertainties in economic and hydrologic variables. SDP has been a very useful technique for calculating single reservoir operating policies. Labadie (2004) presents a list of researchers that have successfully implemented SDP for single reservoir problems. Labadie's list includes Stedinger et al. (1984), who successfully applied SDP to a dam in the Nile River System, and Huang et al. (1991), who

successfully applied SDP to the Feitsui Reservoir in India. A more extensive list of papers that includes 14 researchers who have applied SDP to single reservoir systems is provide in Stedinger et. al (1984).

While SDP is an attractive algorithm for problems that have a small state space, these problems can get extremely large if the state space is of higher dimension due to the well-known "curse of dimensionality" (Bellman, 1952). For example, if a state vector is used to describe the storage level of each reservoir and some "hydrologic trend" information (as suggested by Pereira and Pinto, 1989) and each component of the state vector is discretized into X intervals, there are X^{2N} discretized states in each stage. In other words, the number of states increases exponentially with the number of state variables in the problem, greatly increasing computational effort. For example, suppose X = 20 intervals of discretization:

1 reservoir $\Rightarrow 20^2$ = 400 states

2 reservoirs $\Rightarrow 20^4$ = 160,000 states

3 reservoirs $\Rightarrow 20^6$ = 64 million states

4 reservoirs $\Rightarrow 20^8$ = 25 billion states

5 reservoirs $\Rightarrow 20^{10}$ = 10 trillion states

(from Pereira, 1989)

The implication is that the solution of the SDP problem is computationally challenging for even relatively small reservoir systems. This explosion of state space can cause SDP algorithms to be very unattractive when dealing with large reservoir systems.

Improvements to Stochastic Dynamic Programming models have been examined in the past. The motivation is to develop ways to solve systems with a

large state space within the context of SDP algorithms and not be limited so greatly by the "curse of dimensionality." Kelman et al. (1990) developed a technique called Sampling Stochastic Dynamic Programming that captures the complex temporal and spatial structure of the streamflow process by using a large number of sample streamflow sequences. However, this method mitigates, but does not eliminate, the dimensionality problems of SDP and has not been applied to multireservoir systems.

Another approach developed to alleviate the dimensionality problems of SDP is to aggregate all the reservoirs in the system into a single representative reservoir. TVA used such an approach with their Economy Guide Development Program by using energy in storage to describe the system state (Shane and Gilbert, 1982). In order to relate energy in storage to individual reservoir storage levels, TVA assumed that the optimal distribution of energy in storage maximizes the capacity of the entire system. In this case, TVA was successful in developing a 52-week model. Labadie (2004) presents a list of other models that incorporate similar aggregation techniques beginning with Hall in 1970 and ending with Saad et al. in 1996. Stedinger et al. (1984) also present two papers that have incorporated aggregation techniques. These techniques have been successful in developing multireservoir, multi-stage problems. However, the problem with state aggregation methods is the often unacceptable loss of information that occurs during the aggregation process (Labadie, 2004).

An expanded list of SDP models and improvements to standard SDP models is contained in Stedinger et al. (1984), Tejada-Guibert et al. (1995), and Labadie (2004). All of these papers provide references to SDP models, but none of these SDP models are able to completely circumvent the curse of dimensionality. To further

address the dimensionality problems of SDP, another technique has been developed known as stochastic programming with recourse.

2.2 Stochastic Programming with Recourse (SPR)

This section describes the stochastic programming with recourse (SPR) approach to water resource operation planning. A general description of SPR is followed by a mathematical description of the SPR algorithm and a two-stage example of the SPR mathematics. A survey of work using SPR and the shortcomings of SPR are presented along with a proposed methodology to avoid these shortcomings.

In the past, SPR has also been referred to as Dual Dynamic Programming (Pereira, 1989). However, this expression has been used in reference to other methods as well (Yang and Read, 1999).

2.2.1 *General Description of SPR*

In contrast to SDP, SPR is able to handle a large number of reservoirs relatively easily. SPR avoids the curse of dimensionality by attempting to construct an approximation of the future value function (shown in Figure 4) directly from shadow price information from the solution to the optimization problem (Pereira and Pinto, 1989). Most SPR solution procedures that have been developed in the past are based on Bender's Decomposition (Bender, 1962), and these future value approximations are typically referred to as *Bender's cuts*.

Historically, the SPR network is represented by a tree structure (Figure 6). The arcs in the tree structure represent a possible hydrologic scenario that may occur

for the entire basin. In this traditional representation, there is a single hydrologic scenario considered for the node at t = 1. The outcome of this hydrologic scenario results in some set of initial conditions to be considered at the nodes for time period t = 2. Each arc that precedes an individual node represents the hydrologic scenario that is considered for the initial conditions of the node.

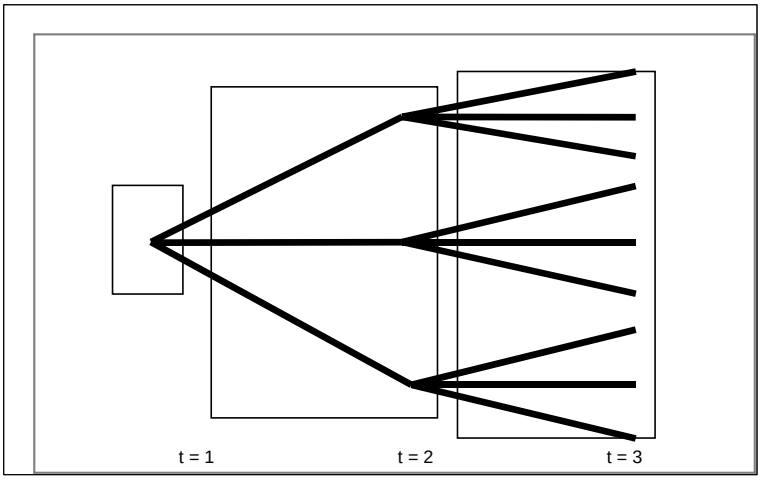

Figure 6: Multi-stage Stochastic Network

For ease of explanation throughout this paper, this traditional tree can be mapped slightly differently as illustrated in Figure 7. This representation of the SPR is used in the framework of this research. The tree in Figure 7 represents the exact same problem as in Figure 6, just mapped differently. In Figure 7, the node at time t = 1 represents the initial conditions of the system. The single arc between t = 1 and t = 2 represents the hydrologic scenario that is considered for the first week of the model. This hydrologic scenario results in some set of initial conditions for time t = 2. During the second week, three hydrologic scenarios may occur, each resulting in

some initial conditions to begin time t = 3. This mapping can continue throughout the rest of the planning horizon to t = T.

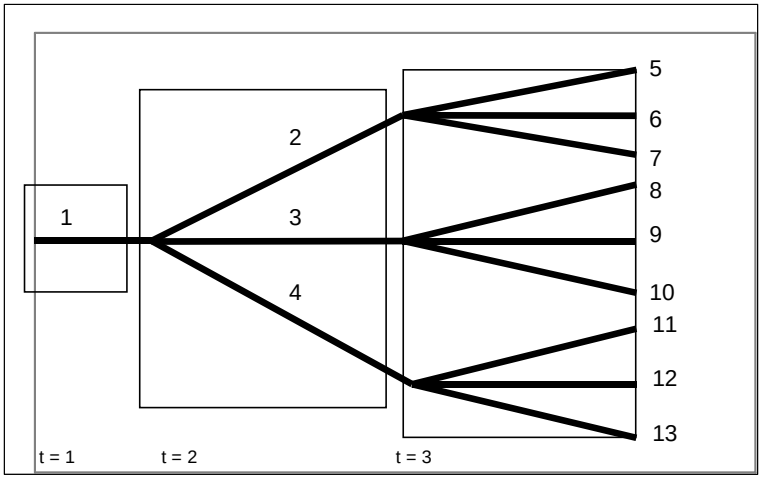

Figure 7: Alternate Representation of the Multi-stage Stochastic Network

The difficulty is that a relatively small number of hydrologic scenarios result in an extremely large-scale linear program. Bender's Decomposition alleviates this difficulty by allowing the problem to be broken down into a series of smaller subproblems, with one linear program (LP) to be solved at each arc of the tree. In these models, the first stage releases are made with consideration of the potential scenarios in the future. Only the first stage decisions are actually implemented, since future scenarios are not known with certainty. Following the implementation of the first stage decisions, the problem is reformulated with the next period decisions and solved over the remainder of the operational horizon (Labadie, 2004).

The essence of the solution process for a multi-stage Bender's Decomposition is as follows. Each time an LP is solved, it is solved with a set of initial storage

values. After the solution is complete, the "current" LP has information to send to both previous and future LPs. The future LPs receive the ending storage levels from the current problem. The future problem can be solved using these values as initial storage values. For example, when a problem is solved for arc 1 in Figure 7, the ending storage value for this arc then becomes the initial storage value for arcs 2, 3 and 4.

The previous LPs receive the incremental value (also referred to as shadow prices, dual variables, or simplex multipliers) of increasing initial storage for the problem just solved, which is equivalently the incremental value of increasing the final storage in the previous problem. In other words, when a problem is solved for arc 2, information can be gathered from the LP to determine the incremental change in the stage 2 objective if there were slightly different initial storage values for this problem. These values are used to create constraints on the previous stage problems, which are known as the Bender's cuts in the solution. The specifics of this cut generation are described in the following section.

In this tree structure, no cuts are produced for the arc at time t = 1. This is because there are no arcs preceding this time period that could result in different initial conditions for time t = 1. The initial conditions for this time period are input into the model. At time t = T, the ending storage levels are not passed to the next time period because there are no more arcs in the network beyond this stage. The last time period also contains no Bender's cuts to constrain the future value estimate. Typically some boundary condition or estimate of the future value is input for the last

time period. In addition, the solutions in time period T are optimal given the initial storage levels.

The solution to the SPR algorithm is an iterative procedure. One sequencing of the iterations that has been successful in previous research and that is used in this research has been termed "fastpass" (Jacobs et al., 1995). As the algorithm moves forward from t = 1 to t = T, the cuts are passed back to the previous time period. When the algorithm reaches the end of the planning horizon (t = T), it reverses direction and moves back to t = 1. As the algorithm moves backward, problems are solved using the same initial conditions as during the forward iteration, but there are now additional constraints in the form of cuts for the problem to use. Each time the algorithm makes an iteration, more cuts are added until the cuts sufficiently approximate the future value of water. This process continues until a solution is converged upon for the first time period.

2.2.2 *Mathematical Description of SPR*

The SPR solution optimizes the expected value over a set of all possible scenarios that could occur. In the SPR tree shown in Figure 7, there is one hydrologic scenario that occurs during the first stage. Following this hydrologic scenario, three possible scenarios could occur (scenarios 2, 3, 4) in the second stage. Each of these scenarios can then be followed by three more scenarios (scenarios 5, 6, 7,…13) as shown.

Each of these scenarios that could potentially occur at a given time period have a given probability based on the predecessor scenario as well as a unique hydrologic inflow representation. In other words, any arc following a scenario

represents a different hydrologic inflow value than the other arcs following the same predecessor scenario (i.e., scenario 5 is different than scenarios 6 and 7). The scenarios that the arcs represent are conditioned on the path that leads to that particular scenario. For example, arc 5 is determined by the path 1 to 2 to 5 and arc 13 is determined by the path 1 to 4 to 13.

In order to represent these scenarios in a mathematical model, each arc can be represented by the notation ω_t. The notation ω_t of a given arc represents each scenario that could occur at time period t. This scenario ω_t is a representation of each scenario that could occur at the given stage based on what happened in the previous stages. For example, in the third stage, shown in Figure 7, there are 9 possible scenarios that could occur, each representing a different scenario that could occur over the entire planning horizon. These scenarios could therefore be represented by $\omega[1,2,3\ldots9]_t$, where $t = 3$.

The solution to a multi-stage stochastic network is classified as large scale due to the existence of multiple interconnected reservoirs and the need for multi-period optimization (Pereira and Pinto, 1985). The mathematical formulation of this large-scale problem can be represented by:

$$\begin{aligned} &\max_{x_t} \sum_{\omega_T} p(\omega_T) \sum_t c_t x_t \\ &s.t. A_1 x_1 = b_1 \\ &\quad A_t x_t = b_t(\omega_t) + B_t x_{t-1} \quad \forall \omega_t, 2 \le t \le T \\ &\quad x_t \ge 0 \quad 1 \le t \le T \end{aligned} \qquad \textbf{(Eq. 2)}$$

where

c_t = a vector of stage t objective function coefficients
x_t = a vector of stage t period decision variables
ω_t = scenario index
p = probability of particular scenario

A_t = *vector of stage t constraint coefficients*
b_t = *vector of stage t right hand side constraints*
B_t = *correlation coefficient, dependency on* x_{t-1}

The objective function is the expected value of the total profit over all time periods. The first block of constraints represents the first period constraints. The second block of constraints is the constraints for all time periods under all scenarios. The solution of the problem in Equation 2 involves computing all the equation variables simultaneously over all time periods in the network represented in Figure 7. At each stage in the network, the vectors x_t, A_t, c_t, etc. could represent thousands of variables, coefficients and constraints. The solution of this problem is too computationally difficult to actually be implemented. However, Bender's decomposition allows this problem to be broken down into smaller subproblems, sub(ω_t), that are much simpler to solve.

2.2.3 *Bender's Decomposition*

In general terms, the objective of a stochastic programming subproblem is to maximize the current period objective function plus the expected value of the future stages. The future periods contain several possible hydrologic inflows that may occur. The expected value of the future stages is calculated as the sum of the probability of each scenario times the predicted value of the future stages. The stochastic solution is maximizing the expected value over a number of scenarios and, as a result, sacrifices the maximum objective for each individual scenario in order to obtain a robust solution over all scenarios (Birge and Loveaux, 1997).

$$
\begin{aligned}
&Max \;\; c_t x_t + \sum_{\omega \in scenarios} P_\omega f_\omega \\
&A_t x_t = b_t(\omega_t) + B_t x'_{t-1} \\
&x_t \geq 0 \\
&f_\omega \leq Obj'_\omega + \sum_r \pi'_{r,\omega}(S_r - S'_r) \forall \omega' \mid \omega_t
\end{aligned}
\qquad \textbf{(Eq. 3)}
$$

where

P_ω= *Probability of particular scenario*
f_ω = *future value under scenario ω (Bender's cut)*
$\pi'_{r,\omega}$ = *dual price from stage t+1 solution*
S_r = *storage at end of stage 1 (variable)*
S'_r = *storage at beginning of stage t+1 from a previous problem*
x'_{t-1} = *a vector of values that are fixed at specific values during this subproblem (storage carryover from previous stage)*

The current period decision variables, x_t, are constrained by typical short-term constraints (i.e., mass balance, reservoir levels, turbine release, etc.). The future values, f_ω, are constrained by a series of inequalities that have been referred to previously as cuts. Each cut is the result of a previous solution of a different optimization problem for time period t +1. Each future period problem is indexed by the symbol (′) and is solved for a particular hydrologic scenario ω and with storage at the end of period t for each reservoir *r* momentarily fixed at S'_r. The future period optimization problem has an optimal solution with value Obj'_ω and incremental storage values for each reservoir $\pi'_{r,\omega}$. Together these values from the future period optimization problem can be used to produce a cut for the current period problem $f_\omega \leq Obj'_\omega + \sum_r \pi'_{r,\omega}(S_r - S'_r) \forall \omega' \mid \omega_t$.

Thus, if the first period ending storage S_r happens to equal S'_r for all

reservoirs, then the future value of scenario ω, f_ω, is limited to Obj'_ω. In this case, the cut is exactly predicting the second period value of the first period solution if scenario ω is realized. If the first period storages are different from these values, the cut becomes an upper bound on the second period solution. The cut tends to be a good estimate for a set of storage values close to the fixed values in the cut.

This approach utilizes the flatness of the value of storage function shown in Figure 1. Because the function changes slowly, a relatively small number of cuts from future period solutions can provide a good estimate of the current period objective function for a large range of storage values. While the above explanation is for a two-stage decision process, it can be extended to a multi-stage process. Each stage has a short-term decision problem, alternative future hydrologic inflows, and cuts for each hydrologic alternative.

2.2.4 *Two-Stage Example*

For example, suppose a two-stage situation with two possible hydrologic inflows of equal probability with one scenario being a low inflow that produces a low value of the second stage objective function and the other scenario is a high inflow producing a higher stage two objective function. The estimate of the second stage objective function is simply the weighted average (or in the case of equal probability, just the average) of these objective functions. Figure 8 illustrates these objective functions for a single reservoir.

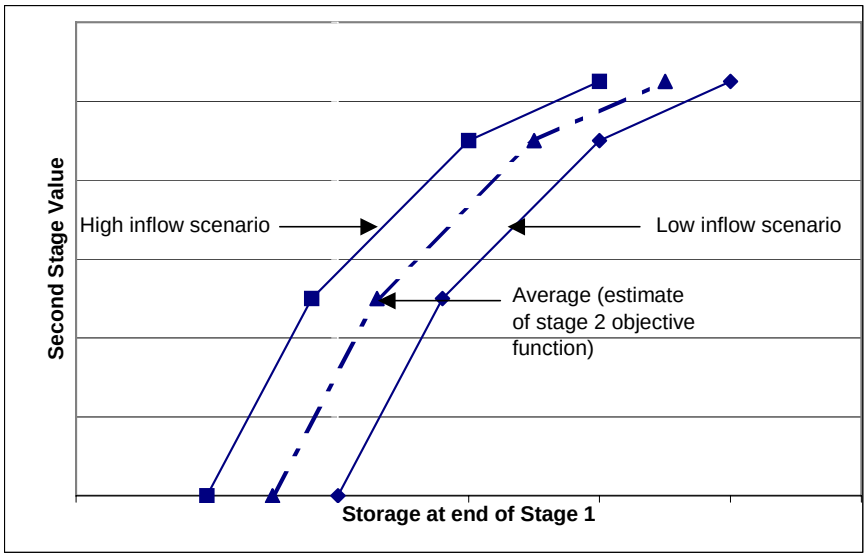

Figure 8: Future Stage Objective Function

As can be noted on Figure 8, the stage 2 objective function is a piecewise linear representation of the true objective function. The piecewise functions are obtained from the shadow price solutions of the stage 2 optimization problem. Each linear segment corresponds to a Bender's cut in a stochastic, multi-stage framework (Pereira and Pinto, 1991). The cut represents the future value of water, or gradient, at the storage levels that were used to generate the cut. As storage levels change, the gradient changes, and the cut becomes a weaker approximation of the objective function value. At some point, enough cuts are generated so that there is very little improvement in the solution. In some instances, the program is initialized with a reasonable set of cuts to reduce computation time (Morton, 1996).

For example, suppose the program begins with two initial cuts marked with solid lines as shown in Figure 9. These two cuts provide an estimate of the second stage benefit as a function of the ending storage for the first stage. If the reservoir

storage at the end of the first stage is 3, the cuts imply an estimate that the second stage has a value of 8. Specifically, the estimate is an upper bound on the actual stage 2 objective function for this scenario. When stage 2 is actually solved at this storage level, it turns out that this is an overestimate. The actual value is 7 and a new cut is generated (the dashed line). In general, any point may be overestimated by the existing cuts because any subset of the full set of cuts tends to overestimate the stage 2 objective function. (If the actual solution to the stage 2 objective is the same as the stage 1 estimate, then all the necessary cuts have been generated and the solution has been found.)

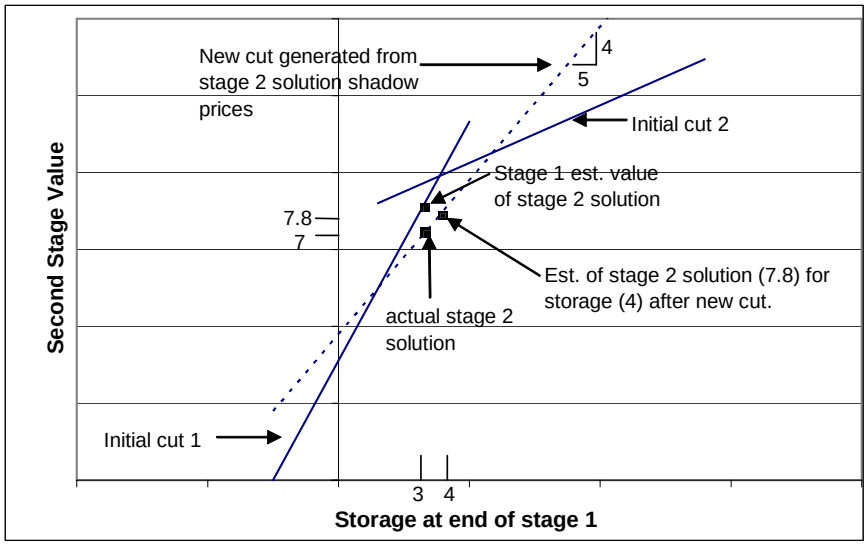

Figure 9: Cut Generation

The new solution also produces a value of storage (or shadow price) of 0.8. This shadow price is the slope of the dashed line representing a new cut in the stage 2 objective function. This shadow price represents the incremental increase in

the stage 2 objective with an increase in the storage at the end of stage 1. In Figure 9, the new cut has a slope of 0.8 and a stage 2 objective function value of 7 at a stage 1 storage of 3. Thus the new cut is $f_i \le 7 + .8(S - 3)$.

After adding this cut to the stage 1 problem, suppose the stage 1 solution had an ending storage of 4. The estimated objective for stage 2 would be 7 + 0.8(4 – 3) = 7.8 ($f_\omega \le Obj'_\omega + \sum_r \pi'_{r,\omega}(S_r - S'_r) \forall \omega' \mid \omega_t$). The true objective value at a storage of 4 is less than or equal this value as shown by the constraint $f_i \le 7 + .8(4 - 3)$.

In this two-stage example, the actual objective is to maximize the first period objective function plus the expected value of future periods, $c_1 x_1 + \sum_{\omega \in scenarios} P_\omega f_\omega$.

Figure 10 provides and illustration of how the stage 2 cuts affect the stage 1 objective function and solution. The lower solid line is the stage 1 objective function (c_1x_1), which increases as ending stage 1 storage decreases (and releases increase). The dashed line represents the averaged cuts for the stage 2 scenarios ($\sum_{\omega \in scenarios} P_\omega f_\omega$). The top solid line combines both curves ($c_1 x_1 + \sum_{\omega \in scenaios} P_\omega f_\omega$) and represents the combined stage 1 and 2 objective. This objective is used to solve for the optimal end of stage 1 storage.

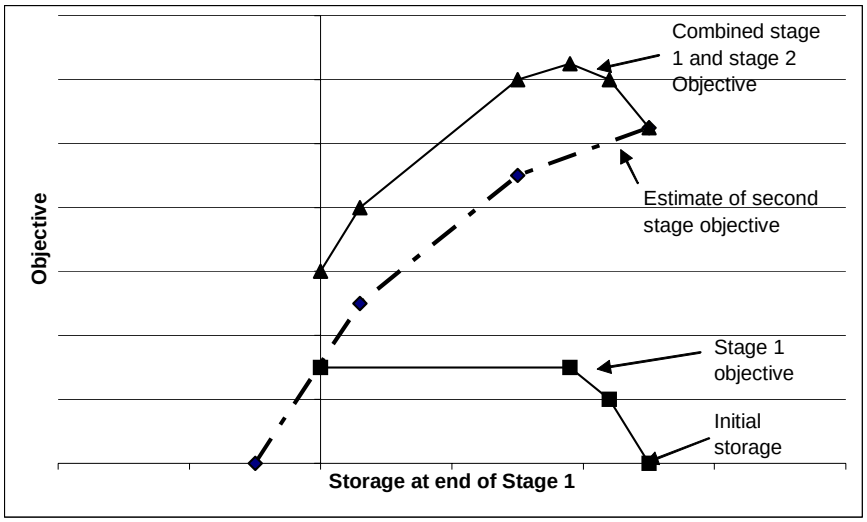

Figure 10: Combined Stage 1 and Stage 2 Objective

Therefore, the optimal release decision in stage 1 is the release decision that results in the optimal combined stage 1 and stage 2 objective.

2.2.5 *Gradient Form of Bender's Decomposition*

The constraint $f_\omega \le Obj'_\omega + \sum_r \pi'_{r,\omega}(S_r - S'_r) \forall \omega' \mid \omega_t$ in Equation 3 represents this research conceptually. However, a reformulation of the constraints in terms of gradients is useful for comparing this research to previous research.

In Equation 3, some components of b_t may be deterministic while others are stochastic. In the framework of this research, the hydrologic inflows represent the components of b_t that are stochastic and all other components are deterministic. In general, the correlation coefficient B_t should have limited dependence on x_{t-1}. In this research, the only portion of x_{t-1} that is of interest is the storage that remains at the end of the previous stage. Thus, B_t is non-zero only for ending storage variables of x_{t-1} and zero for all other variables.

Equation 3 illustrates the approach of this research and can be written in a more general matrix form. The cut in equation 3:

$$f_\omega \le Obj'_\omega + \sum_r \pi'_{\omega,r}(S_r - S'_r)$$

can be rewritten as:

$$f_\omega \le Obj'_\omega - \sum_r \pi'_{\omega,r}(S'_r) + \sum_r \pi'_{\omega,r}(S_r) \qquad \textbf{(Eq. 4)}$$

where

$Obj'_\omega - \sum_r \pi'_{\omega,r}(S'_r)$ is a constant.

The collection of constants for different constraints can be replaced with a vector $g_{t,\omega_{t+1}}$. The term $\sum_r \pi'_{\omega,r}(S_r)$ in each constraint can be replaced by $G_{t,\omega_{t+1}} S_r$, where $G_{t,\omega_{t+1}}$ is a matrix and each row replaces $\sum_r \pi'_{\omega,r}$ and represents the gradient change at the specified storage level S_r. More generally it is possible to write $G_{t,\omega_{t+1}} x_t$, where $G = 0$ except for the case when $x_t = S_r$. This allows the cut to be written in the form:

$$f_\omega \le g_{t,\omega_{t+1}} - G_{t,\omega_{t+1}} x_t \quad \forall \omega_{t+1} \mid \omega_t \qquad \textbf{(Eq. 5)}$$

Now, the subproblem presented in Equation 3 can be written in the general form of:

$$\begin{aligned} \max_{x_t, f_{t,\omega_{t+1}}} \quad & z_t = c_t x_t + \sum_{\omega_{t+1}} p(\omega_{t+1} \mid \omega_t) f_{t,\omega_{t+1}} \\ s.t.\ & A_t x_t = b_t(\omega_t) + B_t x'_{t-1} \\ & ef_{t,\omega_{t+1}} \le g_{t,\omega_{t+1}} + G_{t,\omega_{t+1}} x_t \quad \forall \omega_{t+1} \mid \omega_t \\ & x_t \ge 0 \end{aligned} \qquad \textbf{(Eq. 6)}$$

where

G and g are from valid cut inequalities generated during the decomposition process

g = cut intercepts
G = cut gradients for x_t
e = a vector of 1's for every cut that is generated for scenario ω_{t+1}

Thus, the last block of constraints includes one block for each scenario in the next time period. The term $B_t x'_{t-1}$ in Equation 6 is omitted for the first stage because there is no carryover term from the previous stage. For $t = T$, the last block of equations in Equation 6 are eliminated because there are no scenarios beyond stage T to generate cuts for that stage.

The future value at stage T is generally represented by some boundary condition. In the case studies used in this research, the stage T objective function implicitly includes $\sum_r FV_r(S_{Tr})$, which is the future value of water as a function of storage for each reservoir. This estimate of future value of water is the value that TVA currently uses and therefore makes sense as a boundary condition in the TVA models used here as case studies (presented in Chapter 4). Previous research has used alternative boundary conditions for stage T. In cases where the end of the planning horizon is at a period where the reservoirs should be drained in order to prepare for filling, a future value of zero is input as the stage T boundary. This has the effect of draining the reservoir at stage T (Jacobs, et al., 1995).

The dual of Equation 6 is shown in equation 7.

$$\begin{aligned} \min_{\pi_{t,\omega_t},\alpha_{t,\omega_t}} \quad & z_t = \pi_{t,\omega_t}\left(b_t(\omega_t) + B_t x'_{t-1}\right) + \sum_{\omega_{t+1}|\omega_t} \alpha_{t,\omega_{t+1}} g_{t,\omega_{t+1}} \\ s.t. \quad & \pi_{t,\omega_t} A_t - \alpha_{t,\omega_{t+1}} G_{t,\omega_{t+1}} \geq c_t \\ & e^T \alpha_{t,\omega_{t+1}} = p(\omega_{t+1} \mid \omega_t) \\ & \alpha_{t,\omega_{t+1}} \geq 0 \end{aligned} \qquad \textbf{(Eq. 7)}$$

In the dual formulation, the π variables are the dual prices on this period's constraints ($A_t x_t = b_t(\omega_t) + B_t x'_{t-1}$) in Equation 6 and the α variables are the dual prices on the future value constraints. The solution of the dual is useful because the shadow prices π_{t,ω_t} are generated immediately upon solving the dual and can then be passed to the predecessor scenario. The dual prices from the first stage are not needed because dual price information is not being passed to the predecessor scenario. The stage T dual of the subproblem is again simpler because, at stage T, no future scenarios exist. This simplification can be seen in Equation 8.

$$\min_{\pi_{T,\omega_T}} \; z_T = \pi_{t,\omega_T}\left(b_T(\omega_T) + B_T x'_{t-1}\right)$$
$$s.t. \; \pi_{t,\omega_T} A_T \geq c_T \qquad \textbf{(Eq. 8)}$$

In this formulation of the model, the original problem and each subproblem are bounded and have a finite number of solutions. Therefore, the gradient vector *G* and the cut intercept vector *g* are finite for each subproblem. At the time of solving any given subproblem, only a portion of the cuts from *G* and *g* may be present. In general, only a small fraction of all possible cuts are present. The cuts presented above implicity use a 0 value for any α variable corresponding to a missing cut. This is a feasible, if possibly less than optimal, dual solution.

Theorem: The cuts in Equation 6 are valid.

This theorem has been proven elsewhere (Bender, 1962; Infanger and Morton, 1996) An outline of the proof is provided below. The proof of the validity of these cuts is by recursion, proving validity for T and working backwards. The proof for a generic step t is as follows:

An individual cut is equivalent to:

$$\pi_{t,\omega_t} B_t x'_{t-1} + \pi_{t,\omega_t} b_t(\omega_t) + \sum_{\omega_{t+1}|\omega_t} \alpha_{t,\omega_{t+1}} g_{t,\omega_{t+1}} \geq f_{t-1,\omega_t}$$

Proof of validity can be shown by duality through a series of intermediate inequalities:

$$\pi_{t,\omega_t} B_t x'_{t-1} + \pi_{t,\omega_t} b_t(\omega_t) + \sum_{\omega_{t+1}|\omega_t} \alpha_{t,\omega_{t+1}} g_{t,\omega_{t+1}} \geq \pi_{t,\omega_t} A_t x_t$$

$$+ \sum_{\omega_{t+1}|\omega_t} e^T \alpha_{t,\omega_{t+1}} f_{t,\omega_{t+1}} - \sum_{\omega_{t+1}|\omega_t} \alpha_{t,\omega_{t+1}} G_{t,\omega_{t+1}} x_t$$

because x_t is feasible in the primal problem ($[\pi_{t,\omega_t}, \alpha_{t,\omega_{t+1}}]$ x constraints in Equation 6),

$$\pi_{t,\omega_t} A_t x_t - \sum_{\omega_{t+1}|\omega_t} \alpha_{t,\omega_{t+1}} G_{t,\omega_{t+1}} x_t + \sum_{\omega_{t+1}|\omega_t} e^T \alpha_{t,\omega_{t+1}} f_{t,\omega_{t+1}} \geq c_t x_t$$

$$+ \sum_{\omega_{t+1}|\omega_t} p(\omega_{t+1} \mid \omega_t) f_{t,\omega_{t+1}}$$

because the dual prices, π_{t,ω_t} and $\alpha_{t,\omega_{t+1}}$, are feasible (constraints in Equation. 7 x [x_t, $f_{t,\omega t+1}$]).

Finally,

$$c_t x_t + \sum_{\omega_{t+1}|\omega_t} p(\omega_{t+1} \mid \omega_t) f_{t,\omega_{t+1}} = f_{t-1,\omega_t}$$

by the definition of future value.

A more thorough and involved description of Bender's decomposition, is provided in Bender's (1962) and Infanger and Morton (1996).

The important point to note from this proof is that **a dual solution need only be <u>feasible</u> to generate a valid cut.** As illustratede in chapter 3, using this fact allows cuts to be shared: either used directly for multiple subproblems or slightly modified for other subproblems.

2.2.6 *Previous SPR Models*

SPR has been applied to a wide variety of Stochastic Optimization problems, including hydroelectric scheduling. According to Jacobs et al. (1995) Van Slyke and Wets first described how to apply Bender's Decomposition to the two-stage problem. Birge then extended the work of Van Slyke and Wets to the multi-stage setting. This multi-stage extension has been applied to hydroelectric scheduling in Brazil (Pereira, 1989; Pereira and Pinto, 1985, 1991), California (Jacobs et al., 1995), Norway (Rotting and Gjelsvik, 1992), the Highland Lakes area in central Texas (Watkins et al., 1999), as well as many other basins. A more complete list of such applications can be found in a recent state-of-the-art review (Labadie, 2004), and in any of the papers cited above.

Pereira and Pinto (1985) formulated hydroelectric scheduling as an SPR model and then solved it using Bender's Decomposition. This allowed for the replacement of the classic dynamic programming state variables with an iterative approximation of the future value component at every stage, thereby eliminating the "curse of dimensionality" (Velasquez et al., 1999). In their 1985 paper, Pereira and Pinto were able to determine a generation schedule for each plant in a hydroelectric system that minimizes the expected operation cost along the planning period. Using the Bender's Decomposition approach, they were able to generate a hydroelectric schedule for 37 reservoirs in a Brazilian basin. (Of these reservoirs, 16 were constrained to a constant storage, and the remainder solved for storage using Bender's cuts). The system operation was calculated over a five-stage planning horizon with two alternate inflow scenarios considered per stage. The system was tested with three different combinations of streamflow sequences and convergence occurred within 12

to 14 iterations. Pereira and Pinto also extended the model to run with three alternate inflow scenarios per stage over a four-stage planning horizon. Convergence occurred within 15 iterations.

Several authors have extended this research to make it more compatible to a larger number of stages. The primary challenge of the SPR solution is the number of subproblems that must be solved explodes exponentially as the length of the planning horizon increases (see Figure 6). For example, if three possible hydrologic scenarios are considered at each node of the SPR tree, the number of scenarios to be considered increases in the following manner:

2nd Stage:	3 scenarios
3rd Stage:	9 scenarios
4th Stage:	27 scenarios
5th Stage:	81 scenarios
6th Stage	243 scenarios
7th Stage	729 scenarios
8th Stage	2187 scenarios

This explosion of the tree structure in the SPR framework often limits the number of stages that can be considered throughout the planning horizon.

Pereira (1989) extended their previous work (Pereira and Pinto, 1985) and applied the name Stochastic Dual Dynamic Programming to the computational scheme developed in that paper. The improvement presented was a way to avoid the "combinatorial explosion" that occurs with the scenario branching in traditional SPR networks. This combinatorial explosion can be avoided for certain stochastic streamflow models. In his 1989 paper, Pereira focuses mostly on the case of independent, or uncorrelated, streamflow realizations where the realization of the current stage scenario has no correlation with previous stages. In this case, each cut

generated in the current stage is valid for the entire previous stage and not only for the predecessor scenario. This is valid because if the stochastic scenarios are independent, then the future cost functions do not depend on the current scenario and, hence, cuts generated for a particular scenario are also valid for any other scenario at the same stage (Infanger et al., 1996). Because of this feature in models that exhibit interstage independency, Pereira (1989) shows to simulate the sequential decision process for all possible scenarios is not necessary, but to use a sufficiently large sample to estimate the expected operational cost is adequate. Pereira also acknowledges the value of this in models that do exhibit interstage dependency, but he did not apply the algorithm to this case. Pereira and Pinto (1991) improve their network sampling procedure using Monte-Carlo simulations. While all three of these papers by Pereira and by Pereira and Pinto describe an attractive technique for problems with many reservoirs, Velasquez et al. (1999) showed that these three papers (Pereira, 1989; Pereira and Pinto, 1985, 1991) were exactly correct for only two stages and were missing a term to be exactly correct for multiple stages. Velasquez et al. (1999) update the description of the algorithm proposed by Pereira and Pinto by including this additional term.

Infanger and Morton (1996) present a methodology for sharing cuts in multi-stage stochastic linear programs with interstage dependency. They show that models that have certain parametric distributions of flows can easily convert cuts generated from a particular scenario to cuts for other scenarios. The ability to share cuts provides increased efficiency in these models by accelerating convergence and allows for sampling of the stochastic tree. A detailed explanation of this cut-sharing model,

as well as that presented by Pereira (1989) is included in the next chapter. The remaining challenge, which is the focus of this research, is the case of uncorrelated, non-parametric flows.

In addition, two specific applications of SPR suggest the importance of accurate models. Jacobs et al. (1995) apply an SPR algorithm to a generation scheduling system under development at PG&E called *Stochastic Optimal Coordination of River-basin and Thermal Electric Systems* (SOCRATES). In this particular model, Jacobs et al. introduce upper and lower bounds on reservoirs, lakes, and rivers attributable to physical constraints, as well as minimum flow requirements for issues such as maintaining fish habitat. This model is divided into subperiods due to the fact that the value of electricity varies significantly between weekday-peak, weekday-off-peak, weekend-peak, and weekend-off-peak time periods.

Watkins et al. (1999) developed a multi-stage stochastic programming model for the management of the Highland Lakes by the Lower Colorado River Authority in central Texas. Watkins et al. were able to show that the amount of water to contract for the coming year is highly dependent on the initial reservoir storage levels. Watkins et al. illustrated that model results can be improved using scenario generation techniques that preserve the correlation of historical flows.

Morton (1996) developed performance enhancements for Bender's Decomposition algorithm for solving multi-stage stochastic linear programs. Morton enhanced the traditional Bender's decomposition algorithm using *warm start* basis selection, *advanced start* procedures, *multicut* procedures, and *tree traversing strategies*. According to Morton (1996) *warm start* techniques obtain "good" initial

basic feasible solutions for subproblems based on optimal basis information from previous subproblem solutions. *Advanced start* procedures generate preliminary cuts prior to initiating a formal Bender's algorithm. The *multicut* Bender's decomposition algorithm returns one cut for each descendant of a particular subproblem instead of a single aggregate cut. *Tree traversing strategies* prescribe the order in which subproblems of the decision tree are solved.

In his 1996 paper, Morton demonstrated that beginning with good initial conditions and preliminary cuts can reduce the computation time of the algorithm. The multicut procedure has the disadvantage of requiring more decision variables but typically takes fewer iterations than the aggregate cut algorithms. Morton (1996) also explored several options for tree traversing strategies and showed that the "fastpass" algorithm is an efficient and robust choice, but other choices may be comparable. These strategies were all tested on the PG&E model used by SOCRATES.

2.2.7 *Network Stochastic Programming*

This research introduces the concept of Network Stochastic Programming (NSP), and it builds on the previous research cited in this chapter. NSP shares three features with Periera and Pinto (1991) and Infanger and Morton (1996): Bender's Decomposition, assumptions about the stochastic structure of inflows to prevent the exponential growth of the scenario tree, and using dual feasibility to share cuts from different subproblems. Like Watkins et al. (1999), NSP also allows for flow correlation without using parametric methods. Jacobs et al. (1995) believed that detailed time step modeling was important, and TVA's prior experience suggests that detailed time step modeling (6-hour time step) is necessary for the entire model in

order to accurately capture realistic operations in their basin. Following these suggestions, NSP is separated into weekly stages, but each stage is solved for 6-hour subperiods. Based on Morton's (1996) research, NSP also incorporates performance enhancing techniques such as the multicut formulation and the fastpass tree traversing strategy.

NSP also shares a feature with SDP: both algorithms define state variables and solve a series of problems for each state and each time period. For this reason, the graph of an NSP network appears quite similar to the graph of an SDP network. However, the states in NSP differ from each other in terms of alternative inflow forecasts for each point in time rather than predetermined storage states. This is a significant difference; eliminating reservoir storage as a state variable dramatically reduces the dimension of the state space for basins that have many reservoirs. Despite the obvious graphical similarity and similar language (e.g., states and stages), NSP and SDP are very different algorithms

While previous research has concentrated on bounds and cuts based on duality theory, relatively little attention has been paid to the generation of primal solutions and convergence. The two approaches that have been used are to calculate a primal solution for every node of the tree or to use Monte Carlo simulation and a probabilistic statement of convergence based on the distribution of inflows.

In contrast, for this thesis, a formulation based on column generation to generate implicit primal solutions for any future scenario without exploring every node of the tree is used. By doing so a *duality gap* is defined for every node explored and its descendants. Of course, the duality gap for the first stage is a duality gap for

the entire problem. Thus, each primal solution is guaranteed to be within some percentage of optimality, and the algorithm can be terminated whenever the solution is within a predetermined optimality tolerance. In addition, by choosing subproblems containing large duality gaps, the overall gap can be methodically reduced either by finding improved solutions or by eliminating other solutions.

Chapter 3

NETWORK STOCHASTIC PROGRAMMING

The premise of Network Stochastic Programming (NSP) is that the exponential growth of the scenario tree that makes traditional SPR so difficult is also unnecessary. A reasonable hydrologic forecast does not uniquely depend on the entire history of inflows to any given point in time. Instead, we propose that a *hydrologic forecast state* can be defined as a set of alternative inflow forecasts with probabilities for each forecast. The following assumption is central to this proposal.

> **Assumption:** a relatively small number of discrete forecast states can approximately capture the forecast differences of alternative scenarios, and actual inflow scenarios can be mapped into these states with relatively small forecast errors.

In other words, there may be many alternative forecasts, but a small number of states can summarize the effect of the past on the forecasts. Of course, with a relatively small number of states, each state typically includes several years from the historical record and may be reached in several different ways in the future. This assumption leads immediately to a network representation of hydrologic forecast states rather than a tree structure.

A visual representation of what this network may look like is presented in Figure 11. In this network, there is one node for each of the three hydrologic forecast

states for each time period (stage) with a transitional hydrologic inflow arc between states. Each arc has a probability and a vector of hydrologic inflows for each reservoir. For example, if each arc corresponds to one week and if each week is modeled at a 6-hour time step, each arc has a vector of 28 hydrologic inflows for each reservoir. It is possible for there to be multiple arcs or no arcs between a pair of states in adjacent stages. The details of the NSP algorithm and the definition of the state representation and the state transformation arcs are explained in greater detail in this chapter.

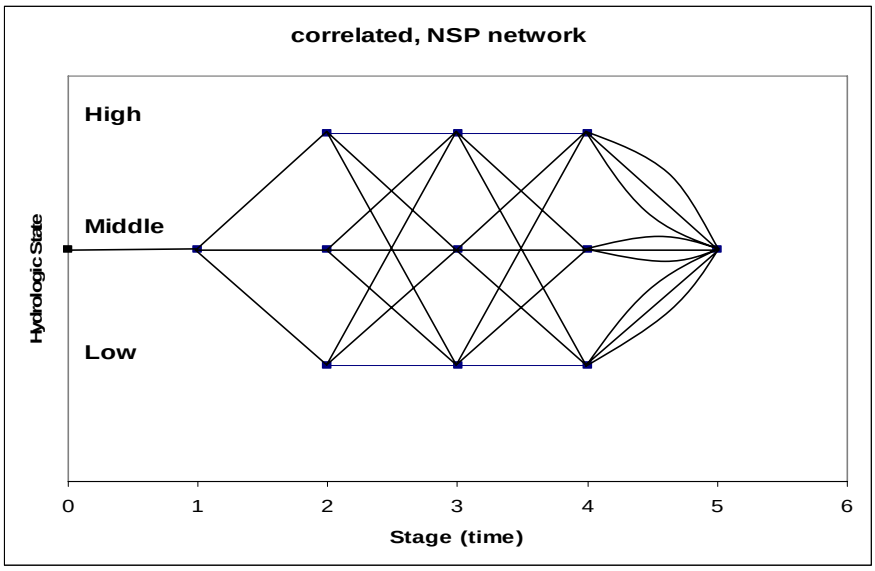

Figure 11: NSP Network

A reasonable criticism is that this network model is inferior to other approaches for generating synthetic inflow scenarios. Furthermore, the network model may be an overly simplistic approximation for many of the other purposes of synthetic inflow scenarios. These are fair criticisms, but they are outweighed in the case of stochastic optimization by the need to have a branching process and a way to

limit the growth of the scenario tree. From this perspective, NSP is a step towards more realistic flow models that can still be optimized with the added bonus of producing an optimality tolerance.

We present NSP within the mathematical framework of SPR. The mathematical descriptions are very similar. The primary difference between NSP and SPR is that NSP eliminates the combinatorial explosion of the scenario tree seen in traditional SPR by using hydrologic forecast states to collapse the scenario tree to a network. NSP can be described from the traditional SPR perspective as a cut sharing mechanism, but a drastic one where certain arcs share all of their cuts with each other. With this complete sharing of cuts, it seems simpler to represent the scenarios as a network. If the scenario tree is collapsed to a constant number of states for each time period, the network only grows linearly as the number of stages increases, rather than exponentially.

This chapter presents NSP in more detail. It begins with a brief comparison to work done in the field of SPR that has attempted to reduce this tree explosion. Next, it describes the definition of hydrologic state used in this paper, as well as potential alternatives and encouragement to explore other state definitions. A representation of NSP with no temporal correlation is be presented first, followed by a more realistic model that considers temporal correlation. Further explanation of the NSP network is also presented as well as a description of the implementation of NSP into RiverWare. A model of the TVA basin is used as a case study in this thesis, and many of the explanations and descriptions of the network pertain to that basin specifically) but there should be no limitation in extending this algorithm to other basins). In this

TVA model, a stage in the NSP network represents one week, and the terms stage and week are used interchangeably throughout the remainder of the paper.

3.1 Cut Sharing - SPR and NSP

The general mathematical description of Bender's Decomposition presented in Chapter 2 includes as special cases NSP, classic SPR, and previous research to circumvent the problems associated with exponential growth of the scenario tree. In this section, we use this description to explain how NSP and other algorithms have solved the problem of exponential tree growth. In the process, we provide an overview of the cut sharing in NSP, but explain the process in greater detail later.

Pereira (1989) showed that, in the case of uncorrelated flows, a cut generated for one scenario is valid for every scenario in the previous stage, not just the immediate predecessor scenario. Because the expected future cost functions do not depend on the current scenario, there is really only one approximate future cost function in each stage (Pereira, 1989). Therefore, each cut generated is actually valid for all of the scenarios in the entire previous stage and not only for the predecessor scenario. Infanger (1996) validates this statement and states that if future cost functions do not depend on the current scenario, cuts generated for a particular scenario are also valid for any other scenario at the same stage.

Figure 12 represents a two-stage scenario tree in the traditional SPR representation. Each arc (i.e., arc 1, 2 and 3) of this tree structure represents a similar problem in that the future value functions of all arcs are the same. Because the future scenarios in the third week are the same, regardless of the stage 2 scenario, the SPR tree shown in Figure 12 could be "collapsed" into a network as shown in Figure 13.

Within the framework of the NSP definition, the uncorrelated case can be thought of as having a single state per week, where all hydrologic scenarios produce the same state. Pereira (1989) presented cut sharing as a mechanism for improving performance of scenario tree optimization but did not suggest a collapsed graph like those shown in Figure 13. The collapsed graph provides a better visual representation of the fact that each arc shares the same future value functions, albeit one that is solved for many different initial storage vectors in the course of optimizing the stochastic optimization problem.

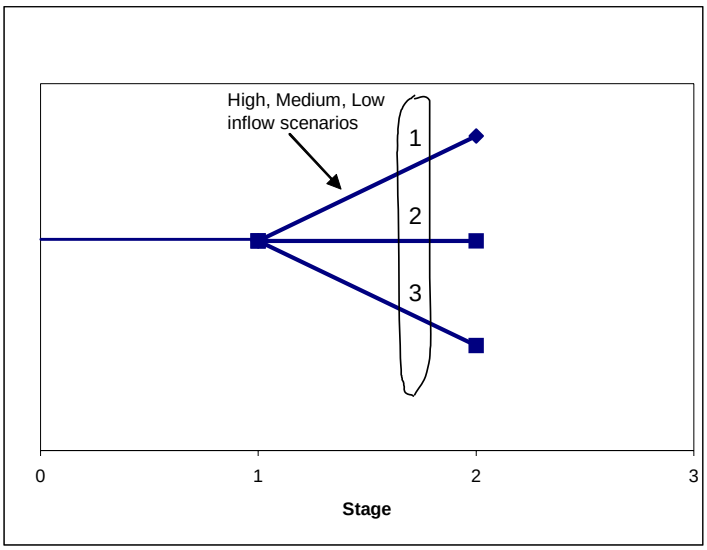

Figure 12: Special Structure of the NSP Network

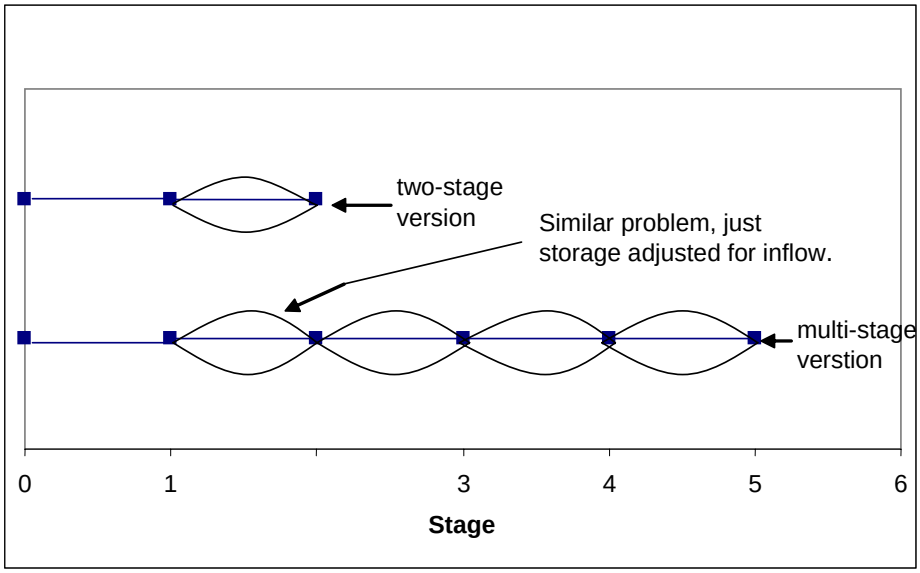

Figure 13: Uncorrelated NSP Network

A second form of cut sharing is also possible in this model. A cut generated for one scenario in the current stage can be converted to a similar cut for all of the scenarios in the current stage. The validity of this cut sharing is proven in previous work via duality theory (Pereira, 1989; Infanger, 1996) and is best explained upon examination of the dual solution to the current subproblem as presented earlier in Equation 7:

$$\min_{\pi_{t,\omega_t},\alpha_{t,\omega_t}} \quad z_t = \pi_{t,\omega_t}\left(b_t(\omega_t) + B_t x'_{t-1}\right) + \sum_{\omega_{t+1}|\omega_t} \alpha_{t,\omega_{t+1}} g_{t,\omega_{t+1}}$$

$$s.t. \; \pi_{t,\omega_t} A_t - \alpha_{t,\omega_{t+1}} G_{t,\omega_{t+1}} \geq c_t$$

$$e^T \alpha_{t,\omega_{t+1}} = p(\omega_{t+1} \mid \omega_t)$$

$$\alpha_{t,\omega_{t+1}} \geq 0$$

Examination of the dual problem shows that, because the probability of the future scenario has no dependency on the current scenario, $p(\omega_{t+1} \mid \omega_t) = p(\omega_{t+1})$, the only difference between any two subproblems with the same future value function is the

coefficient $b_t(\omega_t) + B_t x'_{t-1}$ in the objective function. Thus, a dual feasible solution for one subproblem is feasible for all subproblems at that stage. As discussed in Section 2.2.2, a dual solution need only be feasible to generate a valid cut; optimality is not necessary. A cut for one subproblem can be converted to a valid cut for another subproblem by adjusting for the difference in objective functions: $\pi_{t,\omega_t}\left(b_t(\omega'_t) - b_t(\omega_t)\right)$ for each previous scenario. The exact cuts are shown below in Section 3.3.

Infanger (1996) extended the cut sharing strategy to models in which the stochastic scenario b_t (hydrologic inflow scenarios in this research) has some dependency on the previous stage in the form of an AR(1) model where $b_t = R_t b_{t-1} + \eta_t$, with R_t being a matrix of correlation coefficients and η_t representing a vector of random uncorrelated portion of the flow. Using this type of simple correlation, Infanger was able to break the model down into an uncorrelated flow portion and a correlated flow portion in which the correlated flow portion existed only in the objective function of the dual problem and the uncorrelated flow existed only in the constraint list. This structure of the problem allowed cuts to be shared between stages by adjusting the cuts using the correlated portion of the flow. (Details of this cut sharing methodology are described in Infanger, 1996.) Infanger extended this model to general ARMA correlated models and also included the carryover term B_t as a stochastic parameter.

Infanger developed this model as a general SPR model, not one developed specifically for water resources management. In water resources management, a parametric correlation in hydrologic inflows is generally too simple of a

representation, and modeling the carryover term B_t as a random variable is unnecessary because whatever water is remaining at the end of one stage is precisely the amount of water remaining at the beginning of the next stage.

Both models mentioned above were developed so that a sufficient number of cuts could be added to the solution without having to solve every scenario in the tree. By sharing cuts, these models were able to sample only a portion of the tree using Monte Carlo simulation.

NSP differs from the previous research in a variety of ways. First, the previous models assume that the stochastic variation in the hydrologic scenarios is of the form $\Omega_t = \sum_1 x ... x \sum_t$. In other words, all the scenarios leaving one node are the same scenarios that are leaving a different node in that stage. NSP does not require this assumption. (If any two states have the same scenarios, with the same probabilities and ending states they can be collapsed into a single state.) Second, previous research assumes a parametric process for determining inflow correlation, and NSP uses a non-parametric approach to map historical inflows into the network without using a parametric distribution of flows. Finally, the lower bound calculation in previous models is determined either statistically or by solving an exponential number of subproblems. The lower bound in NSP can actually be computed but does not require an exponential number of solutions.

The cut sharing in an NSP model is similar to the cut sharing in the uncorrelated model of Pereira and Pinto (1991). The main difference is that NSP requires one subproblem for <u>each</u> hydrologic state in each time period, while Periera and Pinto had one subproblem for a single state for each time period. However, in

both approaches, all of the arcs coming into a node share the same future value functions. Thus, cuts generated by any arc leaving a given node are valid for all of the arcs coming into the node.

The second form of cut sharing in the uncorrelated model also generalizes to a correlated NSP network. Suppose, for example, that two arcs, A and B, enter the same node for the current time period. A cut generated by solving arc A is passed back to all of the arcs that directly precede A using the reasoning in the established above. Because arc B shares the same potential future with arc A, the dual prices for the cut for arc A are valid for arc B as well, and the cut from arc A is converted to a cut for arc B by the same process used in the uncorrelated case: adjust for the difference in objective functions: $\pi_{t,\omega_t}\left(b_t(\omega'_t) - b_t(\omega_t)\right)$. This new cut can then be shared with all the predecessors of arc B. This technique differs from the uncorrelated case in that the predecessors of arcs A and B are frequently different.

3.2 Stochastic Modeling of Inflows

3.2.1 *Creating arcs through the historical record*

In developing the NSP network, approximately 100 years of historical data from the TVA basin were used to create hydrologic scenarios that are statistically similar to the historical record in that the scenarios maintain the spatial and temporal correlation of the flows. In this network, hydrologic states are defined as functions of previous flows for each stage. Once the hydrologic states have been defined, each historical flow is mapped into exactly one state for each stage with a transitional arc between states for each historical flow. The probability of moving from one state at the current stage to another state at the future stage is simply the number of historical

arcs that make that state transition divided by the total number of historical traces leaving the state at the current stage:

$$P = x/N$$

where :

P = probability of a given state transition

x = number of historical traces making that transition

N = total number of historical traces leaving the current state

In the network, the sum of the probabilities of the arcs leaving a given state should be one.

3.2.2 *Defining Hydrologic States*

Although the definition of the hydrologic state is a very important area of research, it is not the primary focus of this work. Here, an intentionally simple definition of hydrologic state has been chosen, and further exploration of the definition is a topic for future research. Many other definitions of the hydrologic state are compatible with the NSP network.

In this research, the hydrologic state is defined as a function of previous inflows. Specifically, the hydrologic state is defined as *the sum of total inflow volume into the entire watershed in the previous week*. To create the NSP network, a specified number of states are chosen for each week, and the state is determined by a specified percentile range. For example, if three hydrologic states were assigned to a given week, the states could potentially represent the top, middle, and bottom third of previous flows. The network is not limited by the number of states or the percentile assignments that could be used at each stage (i.e., 3 states represented by top 20%, middle 60%, bottom 20% or 5 states each at 20%).

Developing the NSP network in this manner preserves spatial correlation by using actual historical flows for each scenario. It also potentially preserves temporal correlation throughout the network to the extent that the hydrologic states are well chosen.

The network setup is shown in Figure 14. This network is broken up into five stages (each stage representing a week) with three hydrologic states at each stage of the network (representing, high, medium and low states). In most basins, the short-term forecast of hydrologic inflow is fairly accurate. In particular, in TVA's basin, forecasts for one week are assumed to be quite accurate and future flows are less certain (Gilbert, 1985). For this reason, there is only one hydrologic scenario that is considered for the first week in the network. Based on the historical record of flows for this particular week, week 1's forecasted scenario could fall into any one of the three states considered. Figure 14 shows the week 1 forecast falling into a middle state.

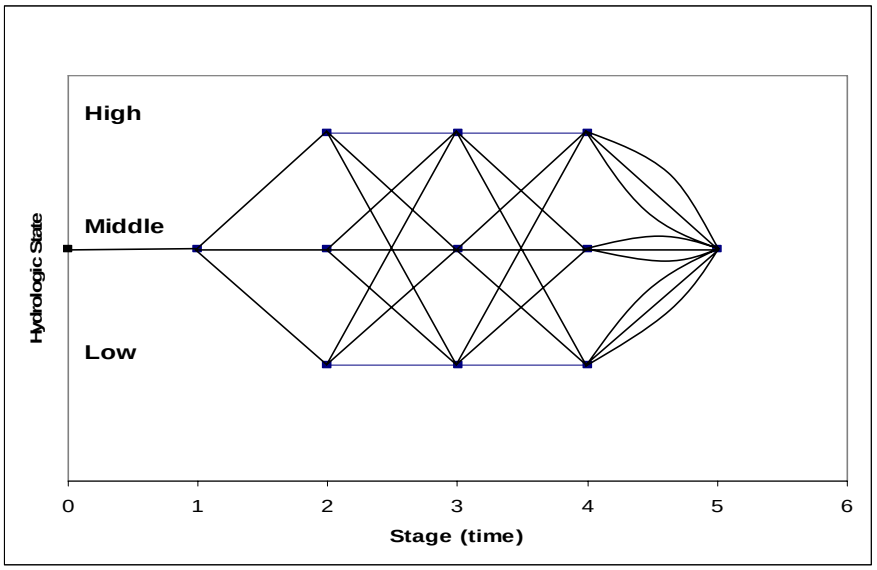

Figure 14: Creating the NSP network

The probability of the week 2 transitions can now be determined based on the week 1 state. For week 2, the number of historical traces for that time period that follow a middle flow value for the previous week is determined. Knowing the total number of week 2 historical traces, it is possible to determine how many of these historic hydrologic scenarios then transition into a high, middle, or low state for week two. For example, if there were a total of 45 historical flow values for week 2 following a middle state for week 1 and 15 of those historical values transitioned from a middle state in week 1 to a high state in week two, the probability of that particular state transition is 15/45 or 33%. It is possible that none of the historical traces follow a particular state transition. For example, if the network was very highly correlated it would be unlikely that a high flow state in week one would be followed by a low state value in week 2.

This state transformation is then carried out for week 3, week 4, etc., developing transition probabilities for each arc in the network. The state representation in the final stage is different than the other stages because at this stage it is assumed that the future value of water is the same under all scenarios. The details of this assumption were mentioned briefly in Chapter 2 and are explained later.

3.2.3 *Alternative Definitions of Hydrologic State*

Using just the basin's previous week flow values as the definition of hydrologic state in the NSP network is an admittedly simple assumption. It is possible to use any number of alternate functions to define the hydrologic state in the network, and more sophisticated approaches could improve the performance of NSP.

The alternatives that could be considered involve both temporal and spatial adjustments. Temporally, one could examine previous flows in different numbers of weeks, average of recent weeks, exponential smoothing of recent weeks, etc. In addition, base flow separation could enhance the accuracy of the model. Spatially, one option would be to divide the flows into various subbasins. In the TVA basin there is a significant distance between the upper and lower basins and significantly different geography. Separate states for each basin could potentially provide a better representation of the streamflow characteristics.

Another alternative approach would be to define the hydrologic states based on physical criteria such as those used for Extended Streamflow Prediction (ESP) (Watkins and Wei, 2004). The states could be tailored to a specific basin, and the alternative hydrologic scenarios could be similar to those generated via ESP methods. Each scenario would transition from one physical state to another and provide a series of hydrologic inflows for each reservoir. The definition of the physical state might have to be simplified to keep the number of alternative physical states reasonably small.

Defining alternative hydrologic state is not attempted in this thesis. However, results of the stochastic network solutions are more reliable as the network more closely describes the true physical characteristics of the basin. For the purposes of this research purposes, the focus is the development of the NSP algorithm. However, it could be possible that in some basins, previous inflows sufficiently capture the hydrologic state of the system.

3.2.4 *Reduction in Arcs – Efficiency Requirement*

When solving the NSP network, an individual linear program is solved several times for each arc in the network. For this reason, the number of arcs in the network greatly affects the efficiency of the algorithm. In this research, some historical streamflow scenarios that duplicate the transition between the same pair of states are removed from the network. Figure 14 illustrates this removal in that there is only one arc connecting any two states. The arc connecting any two hydrologic states represents a single historic streamflow scenario. The aim is to represent the historical record as accurately as possible with a limited number of arcs per week by choosing good representative streamflow scenarios for each state transition. If the number of arcs used to represent each state transition were not limited, the number of arcs examined at each stage in the network would be equal to the number of hydrologic scenarios in the historical record for that time period.

The number of arcs present in each stage can be any number. The network is not limited to having only a single arc connecting any two states. However, during the arc reduction process, at least one historical arc is retained per state pair (if one exists in the historical record) and the rest of the arcs are assigned to each state pair roughly proportional to the number of historical arcs that exist between each state pair. For each state pair, the arcs that best represent the historical streamflow scenarios are the ones retained. For example, if only one arc is retained for a given pair, the median historical value for that state transition is chosen. If five arcs happen to be retained for a given state pair, the historical scenarios that roughly represent the percentiles of 10%, 30%, 50%, 70% and 90% are chosen. When choosing the retained arcs for a given state pair, it is necessary to adjust the probability of the

retained arcs so that the state-to-state transition probability is the same as before the arc reduction.

For example, suppose again that there were a total of 45 historical scenarios in this network for week 2 that followed a middle hydrologic state in week 1. Earlier for explanation purposes, it was suggested that 15 historical scenarios made the transformation from a middle state in week 1 to a high state in week 2. In addition to this state transformation, assume that there are five historical scenarios making a middle to middle state transformation, and 25 making a middle to low state transformation. In this case, if there were four hydrologic scenarios to be used in week 2, one arc would be assigned to the top and middle state pairs, and two arcs would be assigned to the low state transformation because this state transformation has the highest number of historical arcs. In this case, because the probability of making this transition is roughly 56% (25/45), each arc in this state pair would be assigned a probability of roughly 28% (56%/2). This arc assignment example for week 2 is illustrated in Figure 15.

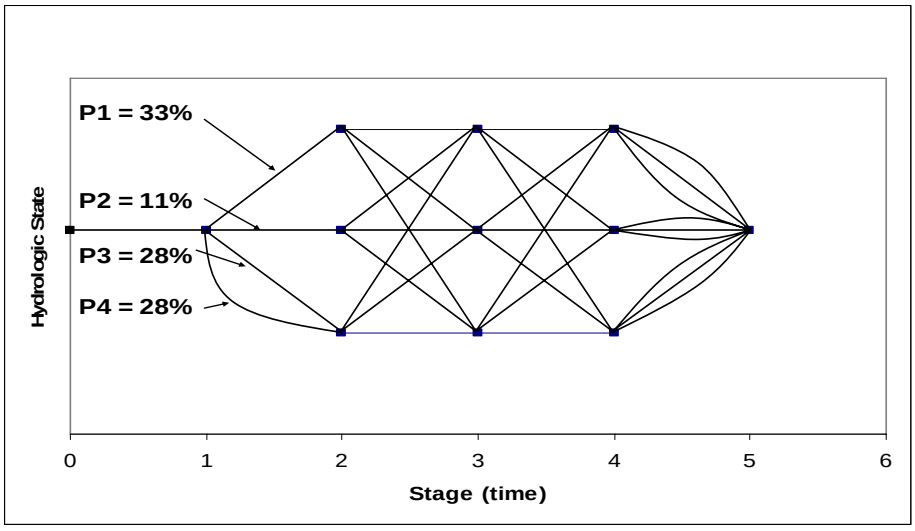

Figure 15: Arc Assignment

Alternatives to the approach mentioned above were not explored in this research. However, there are other possibilities to consider when attempting to limit the number of arcs that are optimized. For example, problems could only be solved for the historical traces that actually occurred.

3.3 Scenario networks of the NSP algorithm

The network of the NSP algorithm takes advantage of this state representation to limit the size of the network as the number of stages increases. In this section, two alternative limitations on traditional SPR trees are used to further explain the network. In the first case it is assumed that the inflows at the current stage are uncorrelated with previous stage flows. However, the spatial correlation of the flows is preserved. The second case allows both spatial and temporal correlation of the inflows. Specifically, in the second case, it is assumed that the system can be described as being in one of a finite number of hydrologic states as defined

previously. In this case, the probability of an inflow vector depends only on the hydrologic state. The hydrologic inflows are sufficient to describe the transition from the hydrologic state in one stage to the hydrologic state in the next stage. The second case is a much more reasonable assumption for most watersheds given the correlation of inflows, but the uncorrelated case is both conceptually and computationally simpler and provides a good basis for the second state explanation.

Previously, the process of how cuts can be shared was illustrated; by adjusting for the difference in objective functions in the dual problem in general matrix form: $\pi_{t,\omega_t}\left(b_t(\omega'_t) - b_t(\omega_t)\right)$. In the following sections, the cut sharing technique specifically associated with this research conceptually using the primal form of the subproblem presented in Equation 3 is discussed.

3.3.1 *Uncorrelated Flows*

In the case of uncorrelated flows, special structure of the problem considerably reduces the number of cuts required to generate a sufficient approximation of the future value function. Figure 12 represents a two-stage scenario tree in the traditional SPR representation. Each arc of this tree structure represents a similar problem in that the future value function of each arc is the same. Pereira (1989) showed that in this case, a cut generated in stage t + 1 is a valid cut for all stage t scenarios, allowing the network to be represented as shown in Figure 13. In addition to this type of cut sharing that was explained earlier, duality theory also shows that a cut solved for any scenario in stage t, also produces a similar cut for the remainder of the stage t scenarios, just adjusted by the difference in inflow vectors. More specifically, suppose we solve a problem in period t+1 for inflow scenario i, at

reservoir r, which has inflows, $I_{t+1,r,i}$ and with initial storage, $S_{t,r,k}$ and generate a cut, k:

$$f_{t,i} \le Obj_{t,i,k} + \sum_r \pi_{t,r,k,i}(S_{t,r} - S_{t,r,k})$$

It is also possible to generate a valid cut for any other scenario, i' just by shifting it by the difference in inflows:

$$f_{t,i'} \le Obj_{t,i,k} + \sum_r \pi_{t,r,k,i}(S_{t,r} - S_{t,r,k} + I_{t+1,r,i'} - I_{t+1,r,i})\forall_{i'}$$

Therefore, sharing cuts in this manner for the model represented in Figure 13 would generate three cuts to be used by the stage 1 solution, each time an individual stage 2 scenario is solved. This would produce a total of nine cuts each time stage 2 was solved (rather than the three that would otherwise be generated).

In summary, if the hydrologic inflows were not temporally correlated, the network can be simplified to a single node for each time period. The arcs are still solved multiple times for alternate initial storage values (i.e., one or more storage values for each arc entering a node). At optimality, the initial week has one optimal release schedule that optimizes the short-term value and the expected long-term value of water. If one of these scenarios after the first period is realized, it is possible to solve the second period arcs using the associated inflows and the initial storages that the first period solution provided. The second stage problem is to optimize the second period value and the expected long-term value. The optimal solution exactly corresponds to one of the cuts in the first stage LP. As the algorithm steps forward through stages, alternately adding an inflow scenario and optimizing a time period, each solution corresponds to a cut in the previous stage. A graphical representation for a two-stage and multi-stage problem of this type is shown in Figure 13. Within

the framework of the NSP definition, the uncorrelated case can be thought of has having a single state per week. All hydrologic scenarios fall into the same state.

3.3.2 *Correlated Flows*

The uncorrelated case is not realistic for most watersheds. Most watersheds have temporal and spatial correlation in the hydrologic inflows, and previous research (Watkins et al., 1999; Yang and Read, 1999) has shown that models that ignore this fact under represent extreme scenarios in the hydrologic record. It is possible to capture this correlation with a set of hydrologic states for each time period as defined earlier. The NSP network assumes that there are a relatively small number of hydrologic states that can differentiate the probabilities of future inflows. As discussed earlier, in this research each state is defined as a function of the previous week's hydrologic inflows.

A network representation of the historical record would include one arc for each state transition. This network may have multiple arcs between a pair of nodes. The number of arcs is proportional to the probability of making a particular transition between nodes, as illustrated in Figure 15. This network is different than a typical SPR problem because it is not necessary to consider the entire historical record. Rather, there are several paths that can yield the same hydrologic state. For each state in a given time period, there is a transitional probability of moving to a state at the next time period. The probability of moving into a state in the next time period is dependent only on the current hydrologic state and not necessarily on how that state was reached. In essence, the correlated NSP network represents a collapsed SPR tree that has different past scenarios leading to the same current hydrologic state. By

designing the network in this manner, if each time period has a constant number of hydrologic states, then the size of the network grows linearly with the number of time periods. (A linear increase in the number of states as a function of time would result in quadratic growth of the network, but this option in the was not explored in the current research.) This NSP network is precisely the network of LPs that is optimized in this work. A representation of this network can be seen in Figure 16.

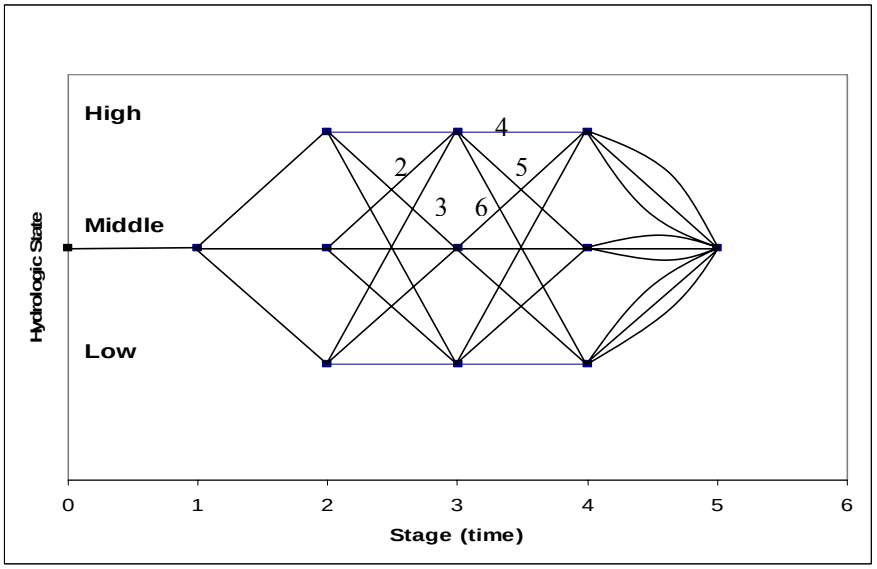

Figure 16: Correlated NSP Network

In a similar fashion as the uncorrelated case, special structure of the correlated network also allows for increased cut sharing. In the uncorrelated network, all arcs at stage t end at the same node and, therefore, share the same future value functions. In the correlated network, a subset of all of the arcs at stage t end at the same node and, therefore, a cut generated for any arc at stage t + 1 is a valid cut for the subset of arcs in stage t that end at that node. For example, in Figure 16, a cut generated in stage 4 for arc number 4, 5 or 6 is a valid cut in stage 2 for arcs 1, 2 and 3.

In addition to this type of cut sharing, for any cut k that is generated at stage t, it is also possible to generate a feasible cut for any other scenario ending at the same node as the scenario that generated cut k by shifting cut k by the difference in inflows in the same manner as in the uncorrelated case:

$$f_{t,i'} \leq Obj_{t,i,k} + \sum_{r} \pi_{t,r,k,i}(S_{t,r} - S_{t,r,k} + I_{t+1,r,i'} - I_{t+1,r,i})\forall_{i'}$$

For example, a cut generated for arc number 1 at stage 2 in Figure 16 would produce a feasible cut for arcs 2 and 3 but not any other arc in stage 2. The increased cut sharing in the correlated case should also allow for sufficient approximation of the future value function without solving every possible subproblem. Approaches to avoid solving every possible subproblem are discussed in detail in Section 3.7 below.

3.4 Boundary Conditions

In solving stochastic programming algorithms, there are some boundary conditions that must be considered. The first boundary condition that exists in both previous research and the NSP algorithm is the initial reservoir storage levels at time t = 0. These storage levels are simply the current reservoir storages in the basin.

A second boundary condition that was mentioned briefly in Chapter 2 must be set at the final stage of the network, stage T, in order to represent the future value of water beyond the end of the planning horizon. Previous research has used differing choices for this boundary condition. Jacobs, et al. (1995) developed SOCRATES in such a manner that the end of the planning horizon is at a period where the reservoirs should be drained in order to prepare for filling and therefore set the stage T future value of water equal to zero. This has the effect of draining the reservoir at the end of stage T.

Watkins et al. (1999) also place zero value on the water supply beyond the end of the planning horizon for some reservoirs in their basin. On other reservoirs, an ending target storage value is set as a constraint. The reservoirs that have zero future value are drained at the end of stage T in the same manner as those in SOCRATES.

In the case studies used in this research, the stage T objective function implicitly includes $\sum_r FV_r(S_{Tr})$, otherwise known as the "value of project storage" (VPS), which is the future value of water as a function of storage for each reservoir. This estimate of future value of is currently used by TVA in their one week optimization models, and serves as a logical boundary condition in the models used here as case studies. The visual representation of this for a single reservoir was shown in Figure 2. Another alternative to this boundary condition would be to set a range of possible ending storage values for each reservoir in the basin.

3.5 Lower Bounds

In this algorithm, the upper bound is being created by the addition of cuts in the stochastic program. The second piece of the algorithm that must be considered is producing a lower bound on the network.

Previous methods for determining lower bounds have generally involved two strategies. The details of these methods are not presented here but can be found in Pereira (1989), Jacobs et al. (1995), and Infanger and Morton (1996). The first method involves calculating a feasible objective function for every arc in the traditional SPR scenario tree. The network representation of NSP could use this method by solving a subproblem for every waiting storage vector at each node on the forward pass; however, this approach is not reasonable because of the exponential

growth in the number of subproblems to solve at each stage (see discussion in Section 3.7.1 and Table 1 in that section).

The second method applied in previous research uses the Monte Carlo Simulation strategies (discussed in Chapter 2 and the beginning of Chapter 3) to reduce the number of subproblems solved in the scenario tree. The lower bounds computed by applying this strategy are determined using a probabilistic argument based on the number of scenarios sampled during the forward simulation. With this probabilistic argument, a tolerance can be set using the standard deviation to determine when the solution is close enough to optimality.

The technique used to determine a lower bound in the NSP algorithm is not be determined statistically. Rather, the lower bound is actually computed based on feasible solutions but does not require an exponential number of objective functions to be solved as in the first method mentioned above. This is made possible by the network structure present in the NSP algorithm. Conceptually, this lower bound can be visualized as in shown Figure 17.

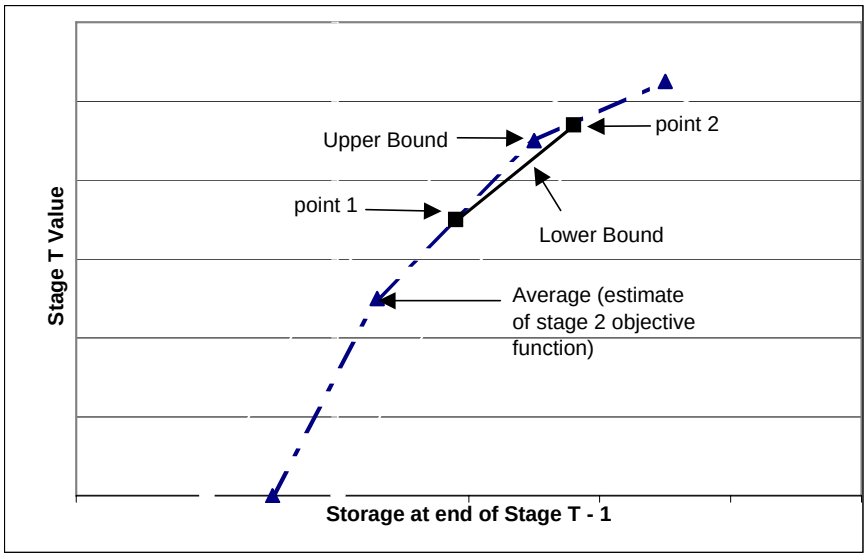

Figure 17: Calculating a Lower Bound

Due to the boundary conditions present in stage T, subproblems solved for arcs in this stage are different from the other stages. They create both a true optimal solution for that stage for a given initial storage vector (the future is not estimated by cuts) and valid cuts for previous stages. For explanation purposes, suppose two of the cuts generated to give an estimate of the stage 2 objective function are produced using point 1 and point 2 in Figure 17. Theoretically, if a line is drawn connecting point 1 and point 2, this is a lower bound on the objective function. As mentioned earlier, the estimate of the objective function produced by the cuts is an over-estimate of the true objective function. Once a lower bound has been established, it is possible to continue to sample the scenario tree until the difference between the upper and lower bound is small enough to be near acceptable optimality. This lower bound is a feasible, but perhaps less than optimal solution.

Because the stage T subproblems create a true optimal solution for that stage for a given initial storage vector, the algorithm can be initialized with a lower bound for each waiting storage vector in the last stage equal to the optimal solution:

$$LB_a(S_k) = \max cx + \sum_r CSV_r(ES_r)$$

where

S_k is a known starting storage vector for the node and the beginning of the arc
$LB_a(S_{k'})$ is the lower bound for arc a, based on a starting storage vector
c is a vector of short term costs for arc a
x is a vector of short term variables for arc a
CSV_r is the Cumulative Storage Value, a piecewise linear value of storage in reservoir r
ES_r is a variable, the ending storage for reservoir r on arc a

The next step is to calculate a lower bound for the waiting storage vector at a node (i.e., the expected value of the lower bounds for the arcs after that node):

$$LBN_n(S_k) = \sum_a p_a LB_a(S_k)$$

where

$LBN_n(S_k)$ is the lower bound for node n, *preceding* arc a and
p_a is the probaility of arc a *following* node n.

This process can be generalized to arcs and nodes that are not in the last stage. The arcs that precede each node can now calculate a lower bound for any initial storage vector by limiting the ending storage vector to be a convex combination of the storage vectors that have lower bounds at this node. By constraining the ending storage vector, a true optimal solution may have been excluded. Thus, the optimal objective function value is a lower bound on the optimal value. Conceptually, the line connecting points 1 and 2 in Figure 17 represents this convex combination. However, Figure 17 represents only a single reservoir.

The linear program is as follows:

$$LB_a(S_{k'}) = \max cx + \sum_k LBN_n(S_k)\lambda_k$$

$$\text{s.t.} \sum_k S_{rk}\lambda_k = ES_r \quad \forall r \in \text{CutRes}$$

$$\sum_k \lambda_k = 1, \quad \lambda_k \geq 0 \ \forall k$$

and all of the other constraints used thus far

where

S_k and $S_{k'}$ are known starting storage vectors for time t and t - 1 respectively,

S_{rk} are the individual resevoir storages in vector S_k

$LBN_n(S_k)$ is the lower bound for node n, *following a*, and

λ_k is a variable, the weighting factor for a convex combination of storage vectors.

CutRes is the set of reservoirs that are being optimized.

The complete algorithm for calculating a lower bound can now be stated as:

1. Solve a forward pass of the existing stochastic programming algorithm, creating storage vectors (S_k) at each node (n) of the network.
2. Use the optimal solutions for the arcs in the last stage as lower bounds for those arcs, $LB_a(S_k)$.
3. Calculate lower bounds for each node, $LBN_n(S_k)$, based on the expected value of following arcs.
4. Calculate lower bounds for the preceding stage arcs by limiting the final storage to a convex combination of storage vectors for the ending node.
5. Repeat steps 3 and 4 until solving for the arc in the first stage.

The result of the algorithm is an implicit operating plan for each arc for certain storage vectors for any set of arcs that might be realized. Each ending storage vector (and thus each initial storage) is a convex combination of storage vectors for which lower bounds have already been calculated. The operating plan for any of the arcs

leaving that node is the same convex combination of short-term operating plans associated with using those storage vectors as starting storages:

$$x_a(IS) = \sum_k x_a(S_k)\lambda_k^*$$

$$\sum_k S_{rk}\lambda_k^* = IS_r \quad \forall r \in \text{CutRes}$$

where

$x_a(S)$ is the optimal plan for storage S,
IS is the vector of initial storage levels,
λ_k^* is the optimal weighting of storage vectors, S_k, found during the calculation of the lower bound from the previous arc.

The linear programs in step 3 may be infeasible in theory. In practice, calculation of the bound is abandoned if any LP is infeasible. If a lower bound is infeasible, it is no longer possible to compute a total lower bound for the network because an arc lower bound cannot be passed back to the node to calculate the node lower bound. However, on future iterations, enough information should become available to eventually generate lower bound.

If lower bounds are calculated for a given arc and waiting storage vector, the bounds and associated storage vectors should be retained; any of the previous lower bound solutions can be used as part of a convex combination. This is similar to retaining cuts from previous solutions.

3.6 Iteration of the Stochastic Programming Algorithm

Previous research has investigated the most efficient iteration procedure for the stochastic programming algorithm. Morton (1996) and Jacobs et al. (1995) present several options for iteration: shuffle, cautious, ε-shuffle, ε-cautious, and fastpass. Their experience showed that fastpass was the most efficient and robust

iteration procedure and that same procedure is applied here. As mentioned in Chapter 2, the fastpass procedure is as follows.

The algorithm moves forward from stage 1 to stage T, with what is termed the "forward pass," and solves each subproblem at a given stage. The cuts from a subproblem are passed back to the previous time period. When the algorithm reaches the end of the planning horizon (stage T), it reverses direction and moves back to the first stage on a "backward pass." As the algorithm moves backward, problems are solved using the same initial conditions as during the forward pass, but there are now additional constraints in the form of cuts for the problem to use. The backward pass also produces cuts to pass back to the previous time period, but it does not pass storage vectors forward to the future time period. (In fact, during the backward pass after a storage vector is used, it is cleared and a new storage vector is used on the next iteration). Each time the algorithm makes an iteration more cuts are added until the cut sufficiently approximates the future value of water. This process continues until a solution is converged upon for the first time period. (Discussion of convergence is provided later in this thesis.)

From an implementation perpective, the forward and backward passes can be represented by a set of loops. A simplified version of the forward and backward pass algorithm would have the following structure.

Forward Pass:

```
t = state number
k = hydrologic state variable
w = waiting storage vector
h = hydrologic scenario

for ( t = 1 to T )
{
    for ( k = 1 to number of hydrologic states for time t )
```

```
{
    for ( Upper  Bound then Lower Bound )
    NOTE: a lower bound is not computed on the forward pass
    in all situations (explanation later).
    {
        for ( w = a subset of waiting storage vectors at state k )
        NOTE: for t = 1, the  storage vector is always the initial
        storage vector for the entire run.
        NOTE: if a subproblem is solved for every waiting
         storage vector at state k, the NSP algorithm has the
         same exponential growth as the traditional SPR
         algorithm  (explanation to avoid this is later).
         {
            for ( h = 1 to # of hydrologic scenarios at state k)
            {
                Solve the subproblem for scenario h.
                NOTE: if this is the Upper Bound, pass a new cut backward
                and the ending storage vector forward.
                NOTE: if this is the Lower Bound, pass the "duality Gap"
                forward  (explanation later).
            }
        }
    }
}
```

Backward Pass:

The backward pass is the same as the forward pass except:

1. The backward pass moves from t = T to t = 1.
2. The storage vector is cleared after it is used.
3. If the current subproblem is a lower bound, it is passed back to the previous period to compute the total lower bound (explanation later).

In this loop setting, the tightest loop is incremented first. Once this loop reaches the end, it is reset to its initial value and the next outer loop is incremented.

The first time that a forward pass is performed there are no cuts available to provide guidance on the future value of water estimates for a given time period. Therefore, for the first initial pass, the value of project storage that TVA currently uses for their future value estimates is applied. Note that this value is also the estimate that is used as a boundary condition for the future value at stage T. Also, because the future value on the first forward pass is being estimated in this manner, we cannot generate valid cuts cannot be generated during this pass. Once the

algorithm reaches stage T and begins the backward pass, the future value is based on the cuts, and cuts can now be produced to estimate the future value of storage. Cuts can also be generated for all subsequent forward and backward passes. This estimate on the first forward pass simply initializes the algorithm and provides some reasonable storage vectors to begin producing cuts. In the future, it may be possible to use results from previous runs to establish estimates on the first forward pass.

As the iteration proceeds, cuts and lower bounds tend to become irrelevant for earlier iterations. The newer cuts and lower bounds tend to be used instead. After enough iterations, the algorithm should eventually converge. Details of convergence are discussed later.

3.7 Reducing Waiting Storage Vectors

During a forward pass of the fastpass algorithm, the solution of each subproblem at stage t results in a vector of ending storage values. This ending storage vector becomes a vector of initial storage values at stage t + 1 for the descendant subproblems. In the traditional SPR scenario tree, each subproblem has one vector of initial storage values waiting to be optimized because a single scenario precedes each node (see Figure 7). The nodes single predecessor scenario produces this storage vector.

In the NSP network, each subproblem may have several immediate predecessors and one *waiting storage vector* from each of these ancestors. For example, in Figure 16 scenarios 1, 2 and 3 all terminate at the same node. This would result in three initial storage vectors to be solved for arcs 4, 5 and 6. In fact, if a subproblem is solved for every possible waiting storage vector in the scenario

network, the number of subproblems solved at each stage grows exponentially, and the only advantage over the traditional SPR tree is in the increased cut sharing. However, because each individual subproblem shares a similarity in structure when determining the objective function, it is not necessary to generate the objective function cuts for every possible waiting storage value. This is due to the cut sharing benefits of the NSP algorithm discussed previously. On the other hand, it is necessary to sample enough of the network to produce a good approximation of the objective function.

The presence of multiple waiting storage vectors at each node results in the algorithmic choice of which storage vector to use as the initial storage vector for a given subproblem. In this research, two methods for choosing a waiting storage vector at a node are implemented. The first approach, which is similar to the Monte Carlo simulation of previous research, is to randomly select one storage vector to solve at each node.

3.7.1 *Choosing a Waiting Storage Vector Via the Duality Gap*

A second option is to score each of the waiting storage vectors with a criteria and choose the vector with the best score. The "score" used here is the duality gap for a given storage vector (i.e., the difference between the current upper bound and lower bound for that vector). The procedure to choose a waiting storage vector via the duality gap is based on the lower bound computation discussed previously. This duality gap is determined by first solving an arc for a given waiting storage vector using the cuts as the future value estimate. The ending storage vector of this solution is passed to the next time period along with an associated upper bound. A lower

bound is then computed for this same arc using the same initial waiting storage vector by constraining the ending storage in the manner described above to compute a lower bound. This lower bound is then subtracted from the upper bound for the waiting storage vector associated with this arc. The waiting storage vector now has an associated gap between the upper and lower bound, and the storage vector with the largest gap is used in the next forward pass.

The only requirement to computing a gap in this manner is that a feasible solution for a lower bound has been found for all arcs in future stages. Because a lower bound is not computed until the backward pass, the first forward pass cannot choose waiting storage vectors in this manner. In practice, on the first forward pass the random method is used for selecting a waiting storage. Also, on subsequent iterations, if a lower bound was not feasible for any future arcs, again the random method is used for choosing a waiting storage. This should not occur often but is more likely during early iterations.

Selection of the vector with the largest gap was chosen because it has the largest potential to result in new information that should either reduce the upper bound or increase the lower bound. However, when choosing the storage vector in this manner, it becomes necessary to compute both an upper and lower bound for each arc. This doubles the number of subproblems solved at each stage. However, choosing the storage vector in this manner could have the potential advantage of causing the algorithm to converge with fewer iterations.

The number of subproblems that need to be solved at stage n of a forward pass for each alternative, as presented, is shown in Table 1.

Table 1: Number of solutions for a forward pass in week n of a 10-week solution

Nodes per Week* = 3 Branches per Node = 3										
Week	1	2	3	4	5	6	7	8	9	10
Algorithm										
All Waiting Storages	1	3	9	27	81	243	729	2187	6561	19683
Random	1	3	9	9	9	9	9	9	9	9
Duality Gap	2	6	18	18	18	18	18	18	18	18
Nodes per Week* = 5 Branches per Node = 5										
Week	1	2	3	4	5	6	7	8	9	10
Algorithm										
All Waiting Storages	1	5	25	125	625	3125	15625	78125	390625	1953125
Random	1	5	25	25	25	25	25	25	25	25
Duality Gap	2	10	50	50	50	50	50	50	50	50

*Week 1 & 2 have a single node

As shown in Table 1, reducing the number of waiting storages significantly decreases the number of subproblems solved at each stage of the network, especially beyond the fourth stage.

3.7.2 *Iteration: When to Calculate Lower Bounds*

The calculation of a lower bound requires one solution for each waiting storage vector in the network during a backward pass. During the forward pass, an arc in the network is solved only for a single waiting storage vector. For the network shown in Figure 16, this would result in three waiting storage vectors to consider at each node (one for every arc entering a node). On the forward pass, a solution is calculated only once for each arc, but when computing a lower bound on the backward pass, a lower bound solution is calculated for all three waiting storages at each node. However, because the number of problems solved during the forward pass

has been limited, the number of solutions required to calculate a lower bound is still a linear function of the number of stages. The number of lower bound subproblems required to be solved at stage n of a 10-week solution is shown in Table 2. Note that an upper bound solution is still solved once for each arc during the backward pass, so the total number of solutions solved at each stage is also shown in Table 2.

Table 2: Number of lower bound solutions for a backward pass in week n of a 10-week solution

Nodes per Week* = 3 Branches per Node = 3										
Week	1	2	3	4	5	6	7	8	9	10
Solutions										
Lower Bound	1	3	9	27	27	27	27	27	27	27
Total	2	6	18	36	36	36	36	36	36	36
Nodes per Week* = 5 Branches per Node = 5										
Week	1	2	3	4	5	6	7	8	9	10
Algorithm										
Lower Bound	1	5	25	125	125	125	125	125	125	125
Total	2	10	50	150	150	150	150	150	150	150

*Week 1 & 2 have a single node

Again, it is possible that during the backward pass, the computation of a lower bound for any given arc could be infeasible. If this occurs, the iteration procedure quits attempting to compute a lower bound for the network and waits until a subsequent pass to compute a lower bound.

3.7.3 *Convergence*

All optimization solvers, as well as previous stochastic programming algorithms, terminate when the solution is within a given optimality tolerance.

Specifically, stochastic programming algorithms terminate when upper and lower bounds are within a specified tolerance level:

$$UB - LB \leq Gap\ Tolerance$$

Generally, the benefit of reaching true optimality does not outweigh the additional cost and time associated with doing so. Therefore, it is acceptable to terminate computation when the upper and lower bounds are within some tolerance level.

Above the mathematical formulation for computing a lower bound for the NSP network was described. Computing the lower bound requires a feasible solution to every subproblem in the future stages. As shown each time a lower bound is computed on a backward pass in a three-branch network, a total of 36 subproblems would be solved at most stages (Table 2) as opposed to 9 (Table 1) if subproblems were solved only for a single waiting storage vector. While the total number of subproblems solved grows linearly as the number of stages increases, there are still more subproblem solutions than if only an arc for a single waiting storage vector were solved (Table 1 vs. Table 2).

For this reason, in practice, a lower bound is not calculated on every pass. Instead, a total lower bound is computed with user-determined frequency. The frequency needed to adequately compute a total lower bound is not known *a priori* but must be identified by experimentation.

Ultimately, the concern in solving this NSP problem is deciding at what point enough problems have been solved and enough cuts generated to guarantee optimality. After the forward and backward pass, the algorithm could evaluated if the first period solution is sufficiently close to optimality. If it is sufficiently close to

optimality, a reasonable first period solution has been found and the iteration procedure stops. Otherwise, another forward and backward pass is performed. For determining optimality in case studies used in this research, the appropriate gap between upper and lower bounds is uncertain.

In this research 13 iterations were performed and the results of the upper and lower bound computations were examined. In these test cases, the appropriate convergence criterion is not known *a priori,* and this should also be determined by experimentation.

3.8 Goal Programming

Estimation of the future value function through Bender's cuts assumes a convex shape of the future value function at all storage levels. If only a single objective function is considered (maximizing the economic benefit of hydropower) this is always the case. The initial test cases presented in this paper use only a single objective to analyze the validity of the NSP algorithm.

In practice, realistic problems would likely use Preemptive Goal Programming constraints in the model. These goal programming constraints consider multiple objectives such as minimum flows, minimum storages, maximum fluctuations, etc. This research considers one multi-objective test case as a more realistic operations study. When considering multiple objectives, it is possible for there to be points in the curve that are non-convex. These points in the curve exist where there are competing objectives between minimum (or maximum) storage levels and release decisions. At these storage levels, release decisions are not optimal from an economic standpoint because the solution is attempting to meet the minimum (or

maximum) storage levels. In the case of minimum storage levels, instead of making the optimal economic release decision, the solution releases the minimum flows in order to meet the storage objective.

At the storage values where this occurs, the shape of the objective function diverges from the optimal objective curve, as shown in Figure 18. As the releases are held back to meet these storage values, the maximum economic return of hydroelectric releases is larger than the economic return of reducing flows to meet storage constraints. As the storage drops well below the storage values in these goal programming constraints, the minimum releases become closer to the true optimal release decision and the curve eventually returns to the optimal curve. The same thing occurs at the other end of the curve with maximum releases and high storage values.

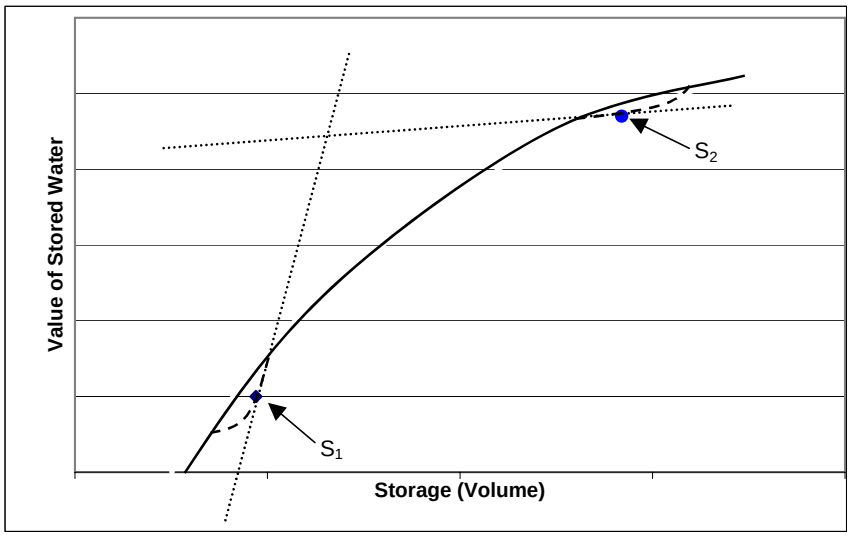

Figure 18: Goal Programming Curve

If a cut is produced for values of storage just below the minimum value, S_1, or just above the maximum value, S_2, the cuts eliminate a large portion of the future value estimate. To account for this problem, when a given subproblem ends with storage values below the minimum goal or above the maximum goal, these storage values are adjusted to be just inside of the convex region, and a cut is produced along the dotted lines in Figure 18.

This is a reasonable choice to make because the goal of the stochastic programming algorithm is to keep the system in normal operating conditions. By generating a cut for the future value estimate that is guaranteed to be within the convex region, the feasible region is limited to the normal operation range. Also at the extreme levels (which are rare), the system is not likely to be operating under economic considerations.

It is possible that several locations in the future value curve have this nonconvex structure. For example, in TVA's models, their goal programming constraint sets have prioritized goals for top and bottom of daily operation zones, balance guide levels, flood guide levels, etc. The region in which the storage level is to be constrained is the inner most gap between high and low storages. Specifically, for the reservoirs for which cuts are being generated for in this research, this region is the area between the balancing guide and the flood guide level. If other goals existed inside of this region, it would be possible to adjust the storages to be within that range.

3.9 Implementation into a Hydrologic Modeling Tool: RiverWare

The NSP algorithm presented in this paper is implemented in a hydrologic modeling tool known as RiverWare. RiverWare is a generalized river basin decision support system that allows for site-specific construction of basin models using a point and click interface (Zagona et al., 2001). Optimization solutions in RiverWare are based on Preemptive Goal Programming that automatically linearizes the decision variables and utilize the very powerful commercial linear program solver CPLEX (Eschenbach et al., 2001)

By implementing this algorithm into RiverWare, it is possible to take advantage of work that has already been done. Primarily, it is easy to build relatively realistic reservoir basin models to test the NSP algorithm. TVA has already constructed optimization models of their basin in RiverWare and has been very cooperative with sharing models, supplying data, and providing feedback on model construction. These models already contain constraint sets, economic data, individual reservoir power calculations, etc. Without taking advantage of previous work in RiverWare, the development of a basin model would be a very difficult task.

Also, if the NSP algorithm proves to be a valuable tool, it is likely that it could be implemented as a commercial product in RiverWare with follow-on work, code enhancements, etc. Developing the algorithm in the RiverWare framework makes this transition more possible.

Implementation of this algorithm into RiverWare involves an extensive amount of coding in C++. The code for this project utilizes a large portion of existing C++ code written for RiverWare as a commercial project and also introduces an extensive amount of new code written for research purposes. New code added to the

RiverWare C++ library includes, but is not limited to, new user-interface objects known as slots for inputting data regarding the NSP network, user-selectable methods for determining the specifics of the NSP algorithm (i.e., duality gap vs. random waiting storages, lower bound computations, etc.), a new time controller to perform the forward and backward pass of the NSP algorithm, logic to appropriately iterate through the NSP network, data objects to store data, etc. A sample of some of the slots added to RiverWare for this research is shown in Figure 19.

The efficiency criterion in this research is based on reducing the number of subproblems to be solved. In implementing this algorithm into RiverWare, no consideration is made to reduce the computation time of an individual subproblem. Future work could make this improvement in efficiency as well.

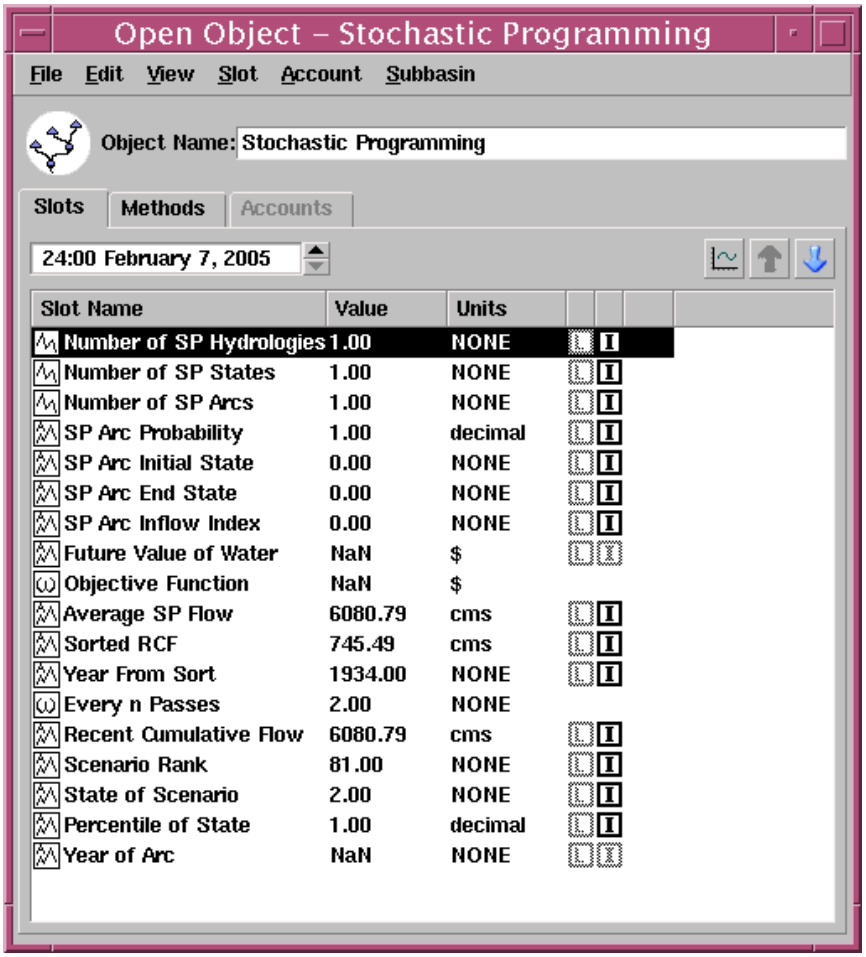

Figure 19: Slots Added to RiverWare for the NSP Algorithm

3.9.1 *Modeling Assumptions for Algorithm Implementation*

In order to implement this algorithm using an existing RiverWare model, it is necessary to make several assumptions regarding the physical conditions of the system. First, the NSP algorithm does not generate Bender's cuts for every reservoir in the basin. Instead, the focus is on generating cuts for the largest reservoirs in the system that have the greatest impact on the economic operation of the basin. The

remainder of the reservoirs are considered run-of-river reservoirs that do not have much potential for their long-term storage levels to vary and whose operation is driven more by inflow conditions and short-term economic value (rather than long-term economic considerations). For these reservoirs, the storage value is fixed at the beginning and end of each stage to a target value. However, these reservoirs are allowed to fluctuate between the start and end dates of the individual subproblems.

It was also necessary to make assumptions regarding the modeling of lagged reaches. In the NSP algorithm, when the model progresses from stage t to stage t + 1, the only data that are moved forward to use in the solution to the next stage is the ending storage vector for the large reservoirs mentioned above. However, for reaches that require some lagged inflow values to solve for outflow, no data are retained from dates previous to the beginning of stage t + 1. To circumvent this problem, the lagged inflow values are set equal to zero for the appropriate number of timesteps before the beginning of stage t + 1. Also, for the sake of consistency, when solving stage t, the lagged inflow values for the appropriate dates at the end of the stage are set equal to zero. In theory, such flows could be incorporated into the cuts rather than being fixed.

Another difficulty in the physical modeling of the basin is in regards to reservoirs that do not have a direct relationship between pool elevation and storage (known as sloped storage reservoirs). As much as possible, these reservoirs are treated in the same manner as run-of-river reservoirs mentioned above by setting target values at the beginning and end of each stage. However, these reservoirs are slightly more difficult to constrain in this manner because there is no direct

relationship between pool elevation and storage. Depending on the inflow conditions into the reservoir, for a given pool elevation, the storage may vary. Also, for these reservoirs, in order to compute storage at a current timestep, lagged inflows must be considered in a similar manner as lagged reaches. To circumvent this problem as much as possible, the lagged inflow values are constrained prior the beginning of a stage. Specifically, reasonable flows are chosen for the time of year that the model considers. The pool elevations were constrained at the beginning of each stage as a target, and RiverWare was allowed to solve for storage value. At the end of the stage, only the pool elevation was constrained, but the outflow and storages were allowed to vary. This could cause some minor discrepancy because the ending conditions of stage t do not exactly coincide with the initial conditions at stage t + 1. These discrepancies should be minor.

All of the above assumptions were made so as to be able to maintain ending conditions of one stage as the beginning conditions of the next. While these assumptions may not be perfect from a physical modeling standpoint, they should have limited effect on the outcome of the results and are necessary for analyzing the NSP algorithm. Future research could examine more realistic assumptions from a physical modeling perspective but should have no effect from an algorithm implementation viewpoint.

Chapter 4

CASE STUDIES USINGTHE TENNESSEE RIVER BASIN

The Network Stochastic Programming algorithm is tested using the Tennessee River Basin. The Tennessee Valley Authority (TVA) has been using optimization in RiverWare to aid in operation of their basin for over a decade, and TVA's existing RiverWare models to are utilized in this research. This chapter introduces some of the general characteristics of TVA's basin. The sources for the information regarding their system include: www.tva.gov, Biddle (1999, 2005), Shane and Gilbert (1982), and Gilbert (1985).

This chapter also explains the alternative NSP networks that were analyzed in this research, as well as the motivation behind the cases chosen.

Formatted: Bullets and Numbering

4.1 TVA: General Characteristics

TVA is the largest public power provider in the United States. On average they generate 16,000 GWH/year. They operate 49 dams and reservoirs and provide power to an area of nearly 80,000 square miles including all of Tennessee and parts of Mississippi, Kentucky, Alabama, Georgia, North Carolina and Virginia (Figure 20).

TVA's power facilities include 11 fossil fuel plants, 29 hydroelectric dams, 3 nuclear plants, 6 combustion turbine plants, a pumped-storage facility, and 17,000

miles of transmission lines. The operation of their basin is considered multi-objective, and the operation plan supplies year-round commercial navigation, reduced risk of flooding, affordable and reliable electricity, improved water supply, improved water quality, and recreation opportunities. The basin receives nearly 50 inches of rainfall and 22 inches of runoff annually. The average monthly runoff in the basin is shown in Figure 21.

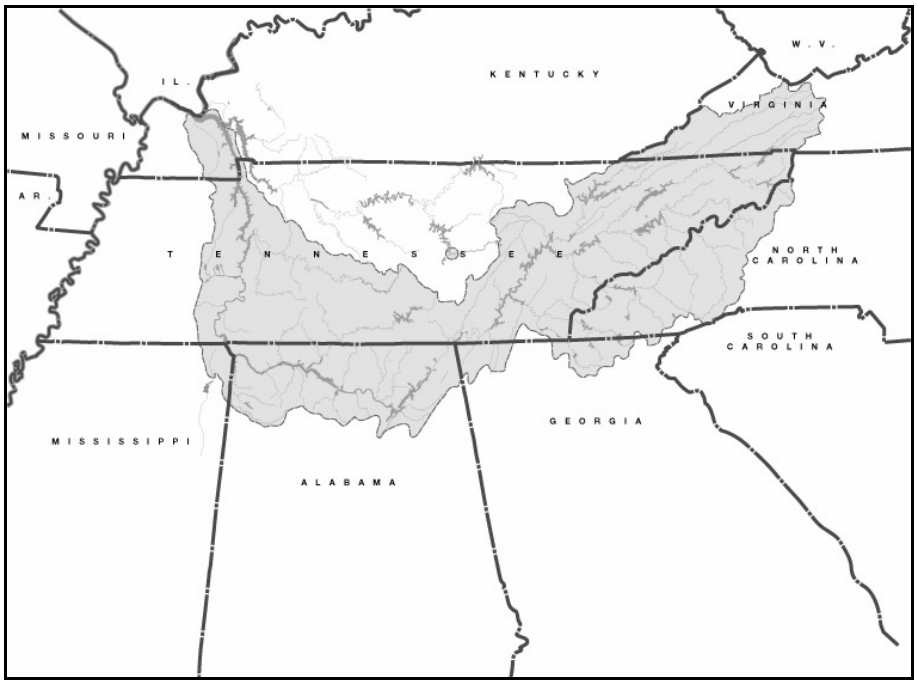

Figure 20: Map of the Tennessee Valley

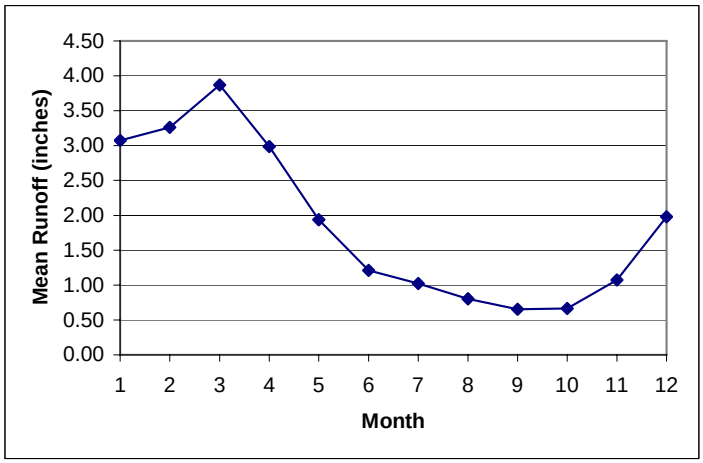

Figure 21: Mean Monthly Runoff in the Tennessee Basin

The Tennessee River Basin begins in the Smoky Mountains of Tennessee and North Carolina. The reservoirs in this region provide most of the dynamic storage in the basin because they are generally deep reservoirs that allow for a wide range of potential operating head. The turbines in these reservoirs typically generate electricity less frequently than the reservoirs further downstream because water is held back to be used when the demand is greatest.

As the water flows out of the Smoky Mountains and heads into the Tennessee Valley, it flows through a series of large sloped storage reservoirs. These reservoirs have the appearance of big, wide rivers that are relatively shallow, especially compared to the storage reservoirs of the Smoky Mountains. These shallow reservoirs provide less dynamic storage but carry a tremendous volume of water. These reservoirs generate electricity more frequently, including more off peak generation, because if the turbines are not open, the water is likely to spill and be wasted. Since these are almost run-of-the-river reservoirs, whatever water comes into the reservoir must go out relatively quickly, either as spill or through turbines. The

percent utilization of the hydroplants at upstream storage reservoirs and the downstream sloped storage reservoirs are shown in Table 3. With the exception of only a few of the upstream reservoirs, the downstream reservoirs generate electricity more frequently.

Table 3: Percent utilization of hydroplants in the TVA basin

Upstream Res.	% Utilization	Downstream Res.	% Utilization
GREAT FALLS	52%	FT. LOUDOUN	59%
NORRIS	43%	WATTS BAR	52%
MELTON HILL	23%	CHICKAMAUGA	66%
SOUTH HOLSTON	29%	NICKAJACK	65%
WATAUGA	22%	GUNTERSVILLE	64%
WILBUR	23%	WHEELER	41%
BOONE	21%	WILSON	45%
FT. PAT HENRY	31%	PICKWICK	59%
CHEROKEE	26%	KENTUCKY	65%
DOUGLAS	31%		
FONTANA	36%		
CHATUGE	24%		
NOTTELY	22%		
HIWASSEE	19%		
APALACHIA	69%		
BLUE RIDGE	32%		
OCOEE 3	69%		
OCOEE 2	55%		
OCOEE 1	40%		
TIMS FORD	23%		

The physical makeup of this system is such that the downstream slope power reservoirs act as a bottleneck to the upstream reservoirs. Water that is released from the upstream reservoirs in the Smoky Mountains eventually flows through the sloped power reservoirs downstream. In fact, if the upstream reservoirs release large

amounts of water, the sloped power reservoirs are likely to spill water even if the turbines are operating at full capacity.

4.2 Optimization in TVA

Currently, TVA uses RiverWare for two different optimization models. The first is a two day model with an hourly timestep. The second model, which is the one used herein as the basis for NSP, is a 1- to 2-week model with a 6-hour timestep. This model is used to develop a weekly hydropower generation schedule that utilizes the flexibility of the TVA reservoir system to reduce the total power supply cost (Biddle, 1999). This model is a deterministic optimization model that considers only a single hydrologic scenario for the upcoming week. However, this model can be run several times per day if the forecast for the future hydrologic scenario changes.

In this model, TVA utilizes preemptive goal programming (Eschenbach et al., 2001) with potentially over 900 goal programming constraints depending on which season the model is considering (Biddle, 2005). The model uses linear goal programming to define physical constraints on the reservoir system, such as backwater profiles, as well as user-defined prioritized constraints including flood control, navigation, recreation, water quality, etc. (Biddle, 1999). The recommended hydropower generation schedule maximizes the short-term avoided operating cost plus the total future value of water remaining in storage. The avoided operating cost at each reservoir is determined by replacing more expensive forms of power generation, such as market purchases or fossil fuels, with less expensive hydropower (Biddle, 1999).

The RiverWare models that TVA uses in practice contain a large amount of data, including hourly hydropower values for the short-term optimization period, expected future hydropower value of stored energy beyond the short-term, guide curves, minimum operating levels, etc (Biddle, 1999). The layout of this model is shown in Figure 22.

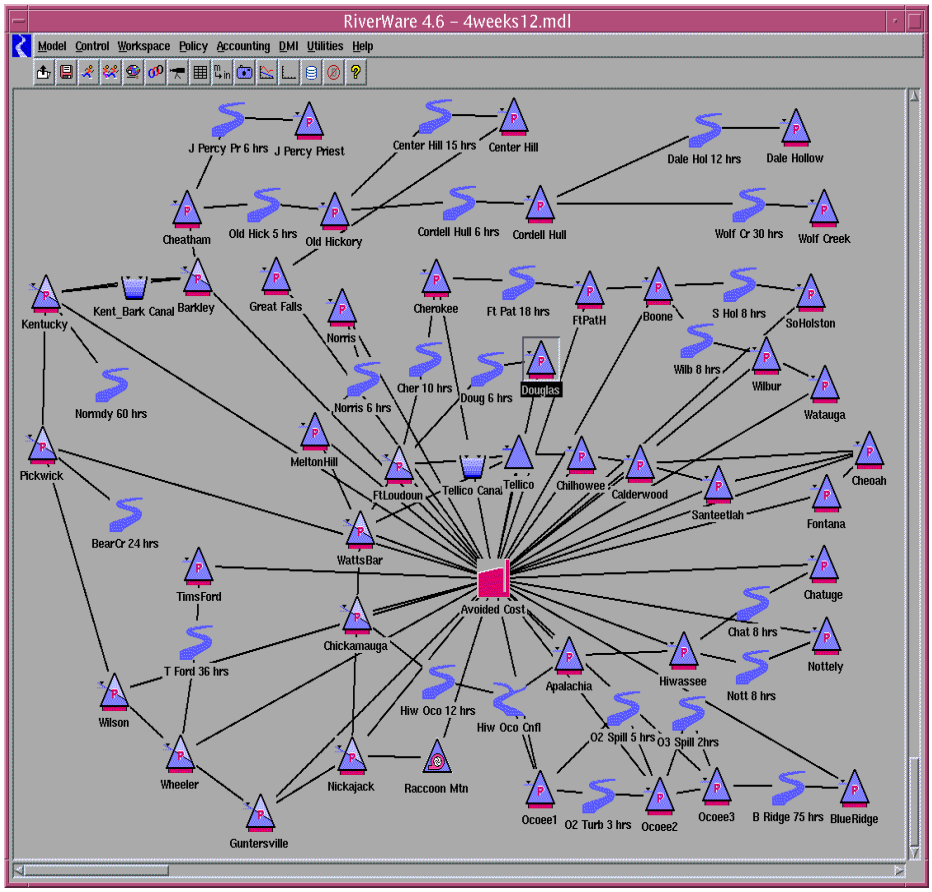

Figure 22: TVA's RiverWare Optimization Model

As a current estimate of the future value of stored energy beyond the short-term, TVA produces a single future value curve per reservoir (Figure 2), known as the

value of project storage (VPS). As discussed previously, the value of project storage is determined as a function of storage in each reservoir. The summation of the value of water remaining in storage in each reservoir at the end of the planning horizon provides the cumulative future value of water remaining in the basin, $\sum_r FV_r(S_r)$.

TVA determines this VPS value for each reservoir by using end-of-season target elevations, expected future hydrologic inflows, and expected market value of hydropower. The VPS estimates are somewhat stochastic in that they do consider alternative historical hydrologic inflows, but they use relatively simple averaging of the historical record to produce these curves. In addition, these VPS curves provide an estimate of the future value of water in a single reservoir without consideration of the water remaining in storage for other reservoirs. Biddle (1999) provides a more detailed explanation of TVA's use of RiverWare.

4.3 Case Studies Performed

Several case studies are used to evaluate the NSP algorithm. In choosing these case studies, the intention is to show a variety of different NSP networks to demonstrate that NSP can produce reasonable reservoir operation schedules without encountering the exponential growth seen in traditional SPR. Implementation of the NSP algorithm in itself avoids the exponential explosion, and the results from these case studies will demonstrate that the NSP algorithm also provides acceptable solutions in a reasonable number of iterations. In addition, these case studies will allow comparison of solutions for alternative NSP network setups to determine the factors to which the NSP algorithm is sensitive.

In this research, although the model shown in Figure 22 is used, the focus of is primarily on optimizing the short-term avoided cost plus the long-term expected value, and in most of the runs, the other goal programming constraints are not considered. This choice was made primarily to determine if the concept of the NSP algorithm is acceptable, however, one goal programming case study is used to test NSP under more realistic operating conditions. One long-term goal of this research is to potentially replace TVA's current estimate of expected future hydropower value of stored energy beyond the short-term.

All of the case studies in this research are divided into one-week stages. An individual subproblem is a one-week model that is solved using 6-hour timesteps. TVA uses a 6-hour modeling timestep because it is adequate to capture the variation in power value for purposes of weekly planning in their basin. At a larger timestep, too much approximation is required, and the model cannot sufficiently capture the details of the daily operation. In practice, TVA further refines the results from the 6-hour model using the hourly model over 2 days. In other basins, it may be worth the computational effort to model the entire week with an hourly timestep.

Therefore, in case studies used in this research, each arc corresponds to one week, and each week is modeled at a 6-hour time step, meaning that each arc represents a vector of 28 hydrologic inflows for each reservoir. All other data in the model, input as well as computed, exist in this form.

Because the reservoirs in the upper basin have more dynamic storage, the NSP algorithm generated Bender's cuts for some of the storage reservoirs in the upper basin and constrains the beginning and ending storages for the remainder of the

reservoirs at each stage as (discussed in Chapter 3). Due to the physical characteristics of the TVA basin discussed in Section 4.1, this is an acceptable modeling assumption. The five largest storage reservoirs in TVA's basin have been chosen as the reservoirs for which Bender's cuts are generated. These reservoirs are Cherokee, Douglas, Hiwassee, Norris and Fontana. In theory, more reservoirs could be included for producing Bender's cuts, with the expectation that convergence might be slightly slower. However, the algorithm would remain unchanged.

The test cases begin on February 7 and cover a duration of 1 to 8 weeks. This time period is considered the beginning of the spring fill season in the TVA basin and the largest hydrologic inflows arrive in the sixth, seventh and eighth weeks. NSP networks that consider more stages should anticipate these large inflows in the future and adjust the reservoir releases accordingly.

All test cases that are setup as correlated NSP networks consider 3 states per stage. The historic hydrologic inflows are mapped to these states (as discussed in Section 3.2). Only a single scenario was used to represent each state-to-state transition. The 3 hydrologic states represent a low, middle, and high hydrologic state with the low state representing the 40^{th} percentile (0.0 to 0.4), the middle state representing the succeeding 40^{th} percentile (0.4 to 0.8), and the high state representing the highest 20^{th} percentile (0.8 to 1.0). These percentiles were chosen ad hoc upon a brief visual examination of the historical data, but no further analysis was done to determine these states.

4.3.1 *Research Questions*

The primary questions to be answered in this research revolve around NSP as a potential stochastic algorithm for reservoir scheduling. Below are the primary questions identified regarding NSP.

1. *Does the upper bound converge on a solution, and if so, does it converge in a reasonable number of iterations?*

   Answering this question will verify that conceptually, the NSP algorithm works as anticipated. The upper bound should converge rapidly due to the increased cut sharing in the NSP algorithm. Each alternative NSP network used as a test case will address this question. To address this question, an uncorrelated 8-week model, several correlated 8-week models, a correlated 6-week model, and a correlated 4-week model were used.

2. *Is there a difference between the correlated and uncorrelated models?*

   Answering this question allows an evaluation to determine if NSP is sensitive to correlation in the models. In the framework of NSP, the uncorrelated model has a single state at each state. Solving the uncorrelated network using NSP turns out to be the same cut sharing algorithm that Infanger and Morton (1996) proposed for "SLP-T with stochastic parameters exhibiting interstage independence." To answer this question, a single 8-week uncorrelated model is used and the results are compared with several 8-week correlated models.

3. *Does the solution depend on the number of stages?*

   Answering this question will provide a determination if the NSP solutions are dependent on the number of stages in the network. Previous

research in stochastic programming with recourse suggests that the solution is not dependent on the number of stages (Jacobs et al., 1995). To answer this question, a correlated 4-week model, a correlated 6-week model, and several correlated 8-week models, are compared.

4. *How do the stochastic solutions compare with TVA's current VPS future value estimate?*

   This question is posed in order to compare results of this research to an existing method that is used in practice and to provide a basis to determine if the results from NSP are reasonable. To make this comparison, one deterministic model is solved using TVA's current future value estimate and the results are compared to the NSP networks (discussed in Question 1 above).

5. *Is it possible to compute a lower bound to guarantee acceptable optimality?*

   This question is posed in order to determine if the proposed method for computing a lower bound is effective as implemented. Previous research that limits the exponential growth of the scenario tree has been able to determine a lower bound only statistically. The proposed method herein should compute an actual lower bound. This question is answered using all the NSP networks discussed in Question 1.

6. *Is there improved efficiency when choosing waiting storages based on the duality gap?*

   Answering this question provides a determination if choosing waiting storage vectors selectively within the NSP algorithm improves the rate of

convergence. In order to evaluate this question, a correlated 8-week model that selects waiting storage vectors using the duality gap is compared with three correlated 8-week models that randomly select waiting storage vectors.

7. *Can the NSP algorithm converge on a solution when considering goal programming?*

   Answering this question we provide a determination if the NSP algorithm works when goal programming constraints are considered. Because TVA uses goal programming constraints in their optimization models, this provides a realistic test case for them. For this evaluation, a correlated 6-week model with goal programming constraints is used.

A comprehensive list of the test cases examined to answer these questions is provided in Table 4. These test cases should allow for sufficient analysis of the NSP algorithm and reveal strengths and weaknesses of the current algorithm.

Table 4: Test cases performed to analyze the NSP algorithm

	Weeks	Cut Reservoirs	States/Stage	Arcs/Stage**	Storage Vector Selection	Passes	LB comp Frequency	Reason
VPS	1	5	N/A	N/A	N/A	N/A	N/A	BASE
CASE 1	8	5	1	9	RAND	13	4	uncorrelated
CASE 2	8	5	3	9	RAND	13	4	RAND/cor/LB
CASE 3	8	5	3	9	RAND	13	2	RAND/cor/LB
CASE 4	8	5	3	9	RAND	13	2*	RAND/cor/LB
CASE 5	8	5	3	9	GAP	13	2*	GAP/cor/LB
CASE 6	6	5	3	9	RAND	13	1	6 v. 8-week/LB
CASE 7	4	5	3	9	RAND	13	1	4 v. 8-week/LB
CASE 8	6	5	3	9	RAND	13	N/A	Goal Programming

***the lower bound algorithm was modified slightly for these test cases.

****there is one arc for stage 1, three arcs for stage 2, and nine arcs for each successive stage.

Chapter 5

RESULTS AND DISCUSSIONS

In this chapter, the results of each individual run are presented first. The results include:

- End of stage 1 storage values for each cut reservoir as a measure of how much water is released in the first week.
- Short-term objective function (stage 1 avoided cost)
- Total objective function (combined stage 1 avoided cost plus future value)
- Upper bound convergence
- Lower bound convergence when possible
- Reduced cost (a.ka. marginal value, shadow price) of increased storage for each cut reservoir at the end of the first stage for selected cases

These results show (1) whether or not NSP can generate a solution in a reasonable number of iterations, and (2) whether or not NSP produces reasonable solutions for the short-term release schedule. After presenting these run results individually, the lower bound implementation is discussed along with selecting the waiting storage vectors using the duality gap (presented in Section 5.2). In Section 5.3, the upper bound solutions are compared in several ways. In Section 5.4, a comparison of solutions is made in order to analyze the benefits of an accurate future

value assumption. Finally, in Section 5.5 a very simple sensitivity analysis is provided.

When comparing differences between total objective function values in many of the results in this chapter, the differences are very small when expressed as a percent difference. For example, the difference in the total objective function values between using the VPS future value estimate and the 8-week NSP stochastic networks ranges from .00016 to .00025% when combining short-term and long-term objective functions (as discussed later in Section 5.3). Based on these differences it is possible to make an erroneous conclusion: the choice of algorithm does not matter. However, this same example also illustrates the differences are significant; if these differences are extrapolated over the course of 52 weeks, the total dollar difference ranges from approximately $2,000,000 to $2,500,000. (This extrapolation over 52 weeks is a rough estimate and only illustrates a *potential* benefit. It is possible (in fact, likely) that the differences presented in Section 5.3 could be either larger or smaller if the test cases considered different seasons, hydrologic states, network setups, or any other perturbations in addition to those considered in these test cases.)

Two factors explain why the percent differences are so small.

1. The quantity of water in storage in these large reservoirs is extremely large and therefore, the value of water in storage is orders of magnitude larger than any rational release.
2. The structures of the short-term objective functions are identical. The only possible changes are due to the long-term value of water.

Formatted: Bullets and Numbering

5.1 Individual Runs

The results from each individual test case are presented below. In these test cases, the total objective function values using the VPS solutions appear much different than the total objective function values from the NSP network. The reason for this difference is the VPS solutions optimize only the short-term objective from the stage 1 solution plus the total value of water in storage. The NSP solutions implicitly include a short-term objective function value for each stage in the network plus the total value of water left in storage at the end of the planning horizon. For this same reason, 4-, 6-, and 8-week NSP networks appear to have much different total objective function values. However, this does not necessarily imply significant difference in the first stage release decisions.

5.1.1 *VPS Case*

The results for this case are from the TVA deterministic model using their current VPS future value estimate (presented in Table 5). The output of this deterministic model provides a base solution to which the other solutions can be compared.

Table 5: VPS results

	End of Week 1 Storage (10^8 m^3)	Reduced Cost ($/$10^3$ m^3)
Cherokee	9.488089	9.431141
Douglas	4.029506	8.062289
Hiwassee	2.492492	11.776052
Norris	16.847609	9.210425
Fontana	10.895824	27.202103

Total Objective Function Value (10^3)	Short-term Objective Function Value (10^3)
158,389	16,495

Formatted: Bullets and Numbering

5.1.2 *Case 1*

Case 1 is an 8-week NSP stochastic network. This network is uncorrelated and, therefore, has only one state per stage. The waiting storage vectors are chosen randomly. Computation of a lower bound was attempted every fourth iteration, but a lower bound is never computed successfully for this case (which will be discussed later). Therefore, only the convergence for the upper bound is presented in Figure 23. The results of this test case are presented in Table 6.

Table 6: Case 1, 8-week, uncorrelated, random network results

	End of Week 1 Storage (10^8 m^3)	Reduced Cost ($/$10^3$ m^3)
Cherokee	9.263667	7.951616
Douglas	3.727313	6.90675
Hiwassee	2.2468868	22.202485
Norris	16.77872	10.502421
Fontana	10.78107	7.922123

Total Objective Function Value (10^3)	Short-term Objective Function Value (10^3)
312,653	17,547

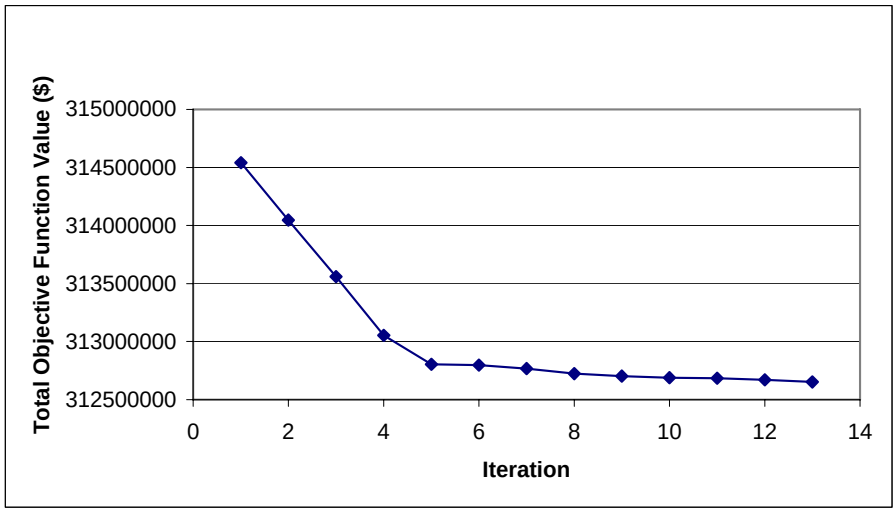

Figure 23: Upper Bound Convergence for Case 1

Convergence of the upper bound appears to occur within five iterations with very little change in the objective function after the fifth iteration.

5.1.3 *Case 2*

Case 2 is an 8-week NSP stochastic network. This network is correlated with three states at each stage. This is the only difference between Case 2 and Case 1. The waiting storage vectors are chosen randomly. The lower bound is computed every fourth iteration. As in Case 1, a lower bound was not computed successfully (which will be discussed later). Therefore, only the convergence for the upper bound is presented in Figure 24. The results of this test case are shown in Table 7.

Table 7: Case 2, 8-week, correlated, random network results

	End of Week 1 Storage (10^8 m^3)	Reduced Cost ($/$10^3$ m^3)
Cherokee	9.446894	8.529282
Douglas	3.80326	7.415767
Hiwassee	2.333013	23.72331
Norris	16.77872	11.009091
Fontana	10.79902	8.370175

Total Objective Function Value ($$10^3$)	Short-term Objective Function Value ($$10^3$)
304,464	17,210

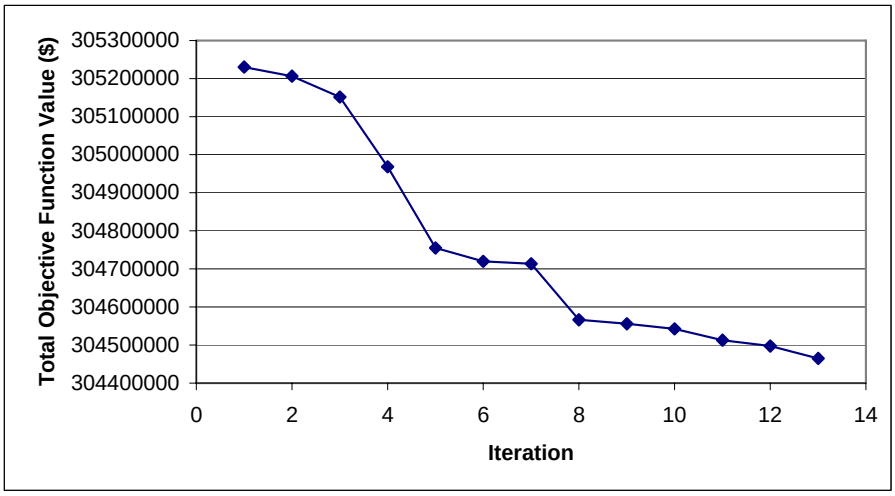

Figure 24: Upper Bound Convergence for Case 2

Convergence in Case 2 occurs after eight iterations but there is some decrease occurring in the objective function value beyond that.

Formatted: Bullets and Numbering

5.1.4 *Case 3*

Case 3 is an 8-week NSP stochastic network. This network is correlated with three states at each stage. The waiting storage vectors are chosen randomly. The

lower bound is computed every other iteration, as opposed to every fourth iteration as in Case 2. However, a total lower bound was not computed successfully for this test case. Therefore, only the convergence for the upper bound is presented in Figure 25. The results of this test case are presented in Table 8.

Table 8: Case 3, 8-week, correlated, random network results

	End of Week 1 Storage (10^8 m^3)
Cherokee	9.358805
Douglas	3.922726
Hiwassee	2.301008
Norris	16.778721
Fontana	10.799887

Total Objective Function Value (10^3)	Short-term Objective Function Value (10^3)
304,379	17,229

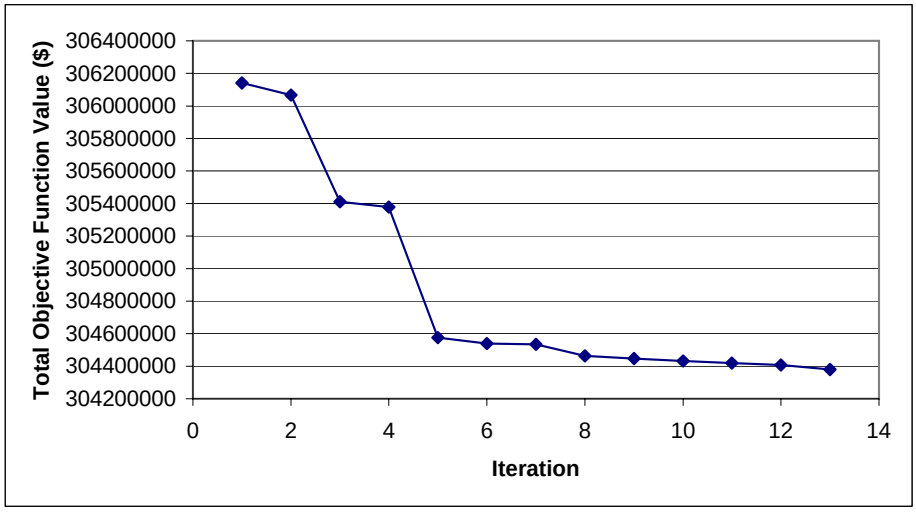

Figure 25: Upper Bound Convergence for Case 3

Convergence appears to occur within 8 iterations. There is some leveling off after the fifth iteration but the objective function drops again after the seventh iteration.

Formatted: Bullets and Numbering

5.1.5 *Case 4*

Case 4 is an 8-week NSP stochastic network. This network is correlated with three states at each stage. The waiting storage vectors are chosen randomly. The lower bound is computed every other iteration. A slight change was made in the lower bound algorithm for this test case after Cases 1 through 3 were run. The convex combination of waiting storages was adjusted to also include the origin from Figure 17. This modification should increase the feasible region when computing a lower bound problem and improve the possibility of computing a lower bound. This modification is the only difference between Case 4 and Case 3. The results for Case 4 are presented in Table 29. The convergence of the upper bound is presented in Figure 26.

Table 9: Case 4: 8-week, correlated, random network results

	End of Week 1 Storage (10^8 m^3)
Cherokee	9.488089
Douglas	3.797147
Hiwassee	2.256071
Norris	16.778721
Fontana	10.799029

Total Objective Function Value ($\$10^3$)	Short-term Objective Function Value ($\$10^3$)
304,549	17,263

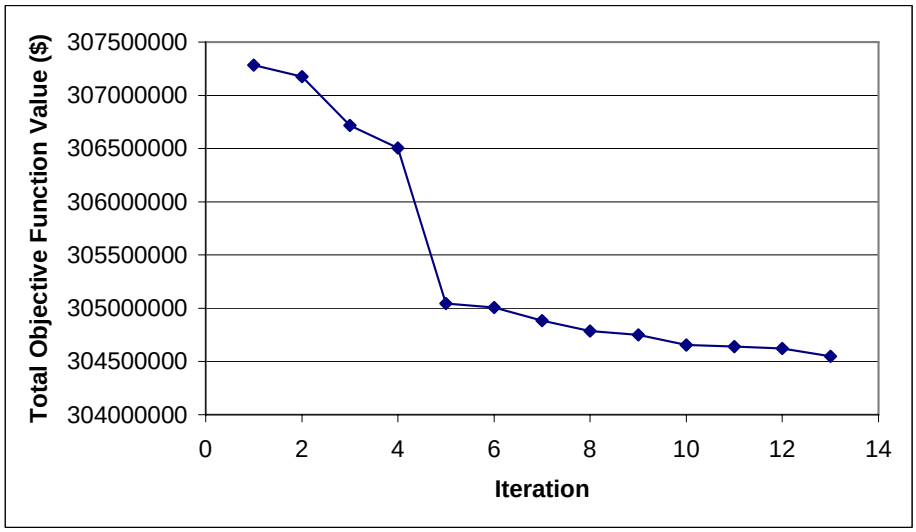

Figure 26: Upper Bound Convergence for Case 4

After approximately eight iterations, the upper bound is changes only slightly; however, some decrease in the upper bound continues to the thirteenth iteration.

For this network, a lower bound was successfully computed. After 13 iterations the upper and lower bound are not particularly close and neither are converging at a significant rate. Also, the scale at which the lower bound increases is much different than the scale at which the upper bound decreases. It would appear that the solution has not converged after 13 iterations. These results are shown in Figure 27.

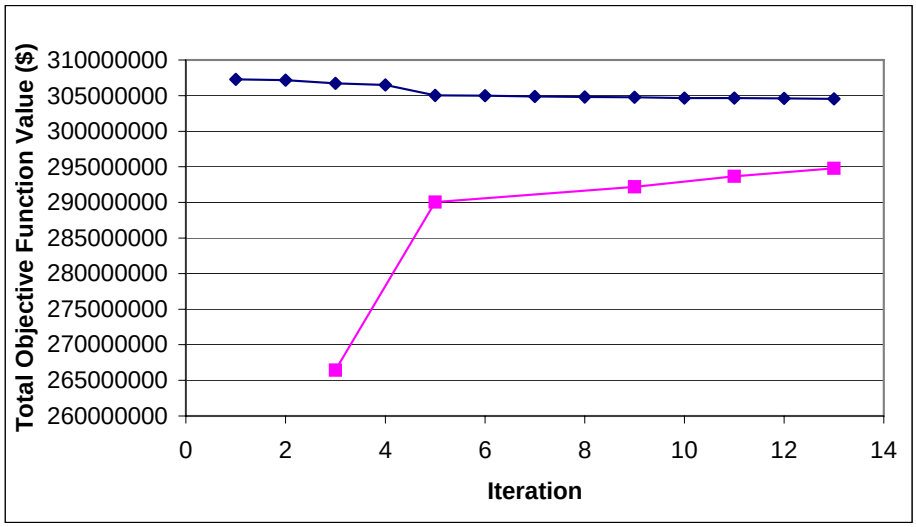

Figure 27: Upper and Lower Bound Convergence for Case 4

Because a lower bound is successfully computed in this test case, the gap between the upper and lower bound is computed. The gap is presented here by computing the lower bound as a percentage of the upper bound and is shown in Table 10.

Table 10: Case 4, upper and lower bound comparisons

Lower Bound Objective Function Value (10^3)	Upper Bound Objective Function Value (10^3)	Gap LB/UB * 100 (%)
294,784	304,549	96.8

Formatted: Bullets and Numbering

5.1.6 *Case 5*

Case 5 is an 8-week NSP stochastic network. This network is correlated with three states at each stage. The waiting storage vectors are chosen based on the duality gap, instead of randomly as in prior cases. The lower bound is computed every other iteration. The same change of including the origin was made in the lower bound algorithm for this test as in Case 4. The results of Case 5 are presented in Table 11 and the upper bound convergence is presented in Figure 28.

Table 11: Case 5, 8-week, correlated, gap network results

	End of Week 1 Storage (10^8 m^3)	Reduced Cost ($/$10^3$ m^3)
Cherokee	9.355591	8.06938
Douglas	3.691462	6.926875
Hiwassee	2.219752	21.71432
Norris	16.778721	10.333205
Fontana	10.734569	8.253574

Total Objective Function Value (10^3)	Short-term Objective Function Value (10^3)
305,046	17,630

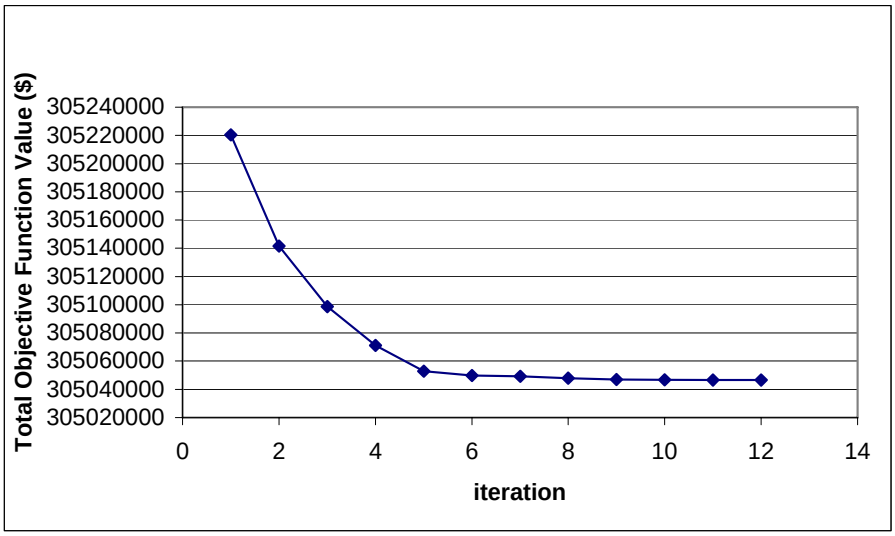

Figure 28: Upper Bound Convergence for Case 5

From Figure 28, it appears that the upper bound converges after roughly five iterations. However, it should be noted that this is the same network setup as in Cases 2, 3 and 4 and should have approximately the same total objective function after 13 iterations. Instead, this test case converges to a larger upper bound. Also, the change in the upper bound from the first to the fifth iteration is small when compared to the change in the upper bound for Cases 2, 3 and 4.

A lower bound was successfully computed for this test case. However, similar to Case 4, after 13 iterations the upper and lower bound are not particularly close and neither is changing at a significant rate. Because the upper and lower bounds are not changing significantly, it is most likely that the solution is getting locked into solving the same trace of the network for all iterations. Further explanation of this issue will be presented later, but this could explain why the lower

bound remains relatively constant and the upper bound never converges to the same solution as in Cases 2, 3 and 4.

The gap is presented by computing the lower bound as a percentage of the upper bound and is shown in Table 12.

Table 12: Case 5, upper and lower bound comparison

Lower Bound Objective Function Value (10^3)	Upper Bound Objective Function Value (10^3)	Gap LB/UB * 100 (%)
293,005	305,046	96.05

5.1.7 *Case 6*

Case 6 is a 6-week NSP stochastic network. This network is correlated with three states at each stage. Computation of a lower bound is attempted every other iteration prior to making the algorithm change (as was made in Cases 4 and 5), and a lower bound was successfully computed on the final iteration. The waiting storage vectors are chosen randomly. This model is used to compare the difference in the upper bound solutions between the 6-week and 8-week correlated models (Cases, 2, 3 and 4). The results of Case 6 are presented in Table 13 and the convergence of the upper and lower bound is presented in Figure 29.

Table 13: Case 6, 6-week, correlated, random network results

	End of Week 1 Storage (10^8 m^3)
Cherokee	9.488089
Douglas	3.689715
Hiwassee	2.424008
Norris	16.778721
Fontana	10.815095

Total Objective Function Value (10^3)	Short-term Objective Function Value (10^3)
264,198	17,119

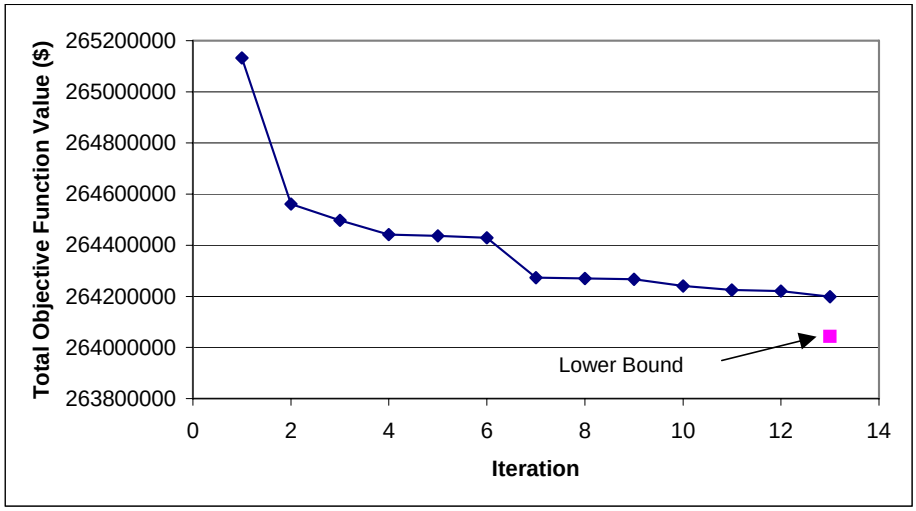

Figure 29: Upper Bound Convergence for Case 6

Convergence in this model appears to occur after roughly seven iterations, but there is still a small decrease in the upper bound after further iterations.

The gap is presented by computing the lower bound as a percentage of the upper bound and is shown Table 14. The upper and lower bounds are much closer here than those in the 8-week models (96.05 – 96.8%).

Table 14: Case 6, upper and lower bound comparison

Lower Bound Objective Function Value (10^3)	Upper Bound Objective Function Value (10^3)	Gap LB/UB * 100 (%)
264,042	264,198	99.94

Formatted: Bullets and Numbering

5.1.8 *Case 7*

Case 7 is a 4-week NSP stochastic network. This network is correlated with three states at each stage. Computation of a lower bound is attempted every iteration prior to making the algorithm change made (as was made in Cases 4 and 5), and a lower bound was computed. The waiting storage vectors are chosen randomly. This model is used to compare the difference in the upper bound solutions between the 4-week, 6-week (Case 6), and 8-week (Cases 2, 3 and 4) models. The results of Case7 are presented in Table 15 and convergence of the upper and lower bound is presented in Figure 30.

Table 15: Case 7, 4-week, correlated, random network results

	Storage (10^8 m^3)
Cherokee	9.488089
Douglas	3.769278
Hiwassee	2.492492
Norris	16.858541
Fontana	10.889437

Total Objective Function Value (10^3)	Short-term Objective Function Value (10^3)
223,335	16,725

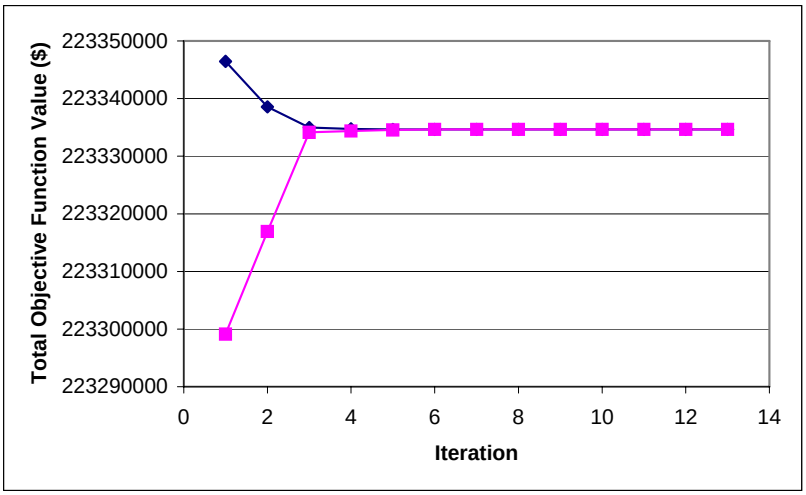

Figure 30: Convergence for Case 7

This test case converges within three to four iterations when looking at both the change in the upper bound objective function as well as the difference between the upper and lower bounds.

The gap is presented by computing the lower bound as a percentage of the upper bound and is shown in Table 16. For all practical purposes, the upper and lower bounds are identical.

Table 16: Case 7, upper and lower bound comparison

Lower Bound Objective Function Value (10^3)	Upper Bound Objective Function Value (10^3)	Gap LB/UB * 100 (%)
223,335	223,335	100

5.1.9 *Case 8*

Case 8 is a 6-week NSP stochastic network. This network is correlated with three states at each stage. The waiting storage vectors are chosen randomly. This model uses a subset of TVA's goal programming constraints. This run is not used for

comparison purposes because the goal is to see if the NSP algorithm will produce a reasonable solution using goal programming.

Table 17: Case 8, 6-week, correlated, random, goal programming network results

	End of Week 1 Storage (10^8 m^3)
Cherokee	9.70729
Douglas	4.745804
Hiwassee	2.583223
Norris	17.222983
Fontana	10.815665

Total Objective Function Value (10^3)	Short-term Objective Function Value (10^3)
205,493	13,453

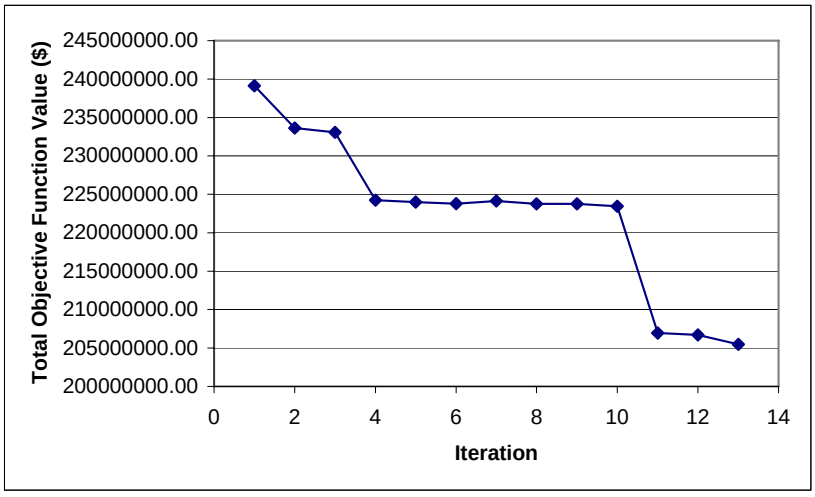

Figure 31: Upper Bound Convergence for Case 8

After the fifth iteration, the upper bound begins to level off but there is another large drop in the upper bound after the eleventh iteration. The final results after 13 iterations appear to be reasonable, but it is not obvious as to what causes the

big jumps in the upper bound solution or whether additional jumps might occur with additional iterations. The test cases that do not consider goal programming constraints (particularly Cases 2, 3 and 4) have similar jumps early in the iterative procedure but then level off. More experimentation is required to determine if this test case would level off after more iterations.

5.2 Discussion of Lower Bound and Gap Algorithm

Below is a discussion of the results of the lower bound implementation as well as of selection of waiting storage vectors based on the duality gap. To describe the concept of choosing waiting storages based on the duality gap, the terminology "gap algorithm" is used throughout this discussion.

5.2.1 *Implementation Problems with Lower Bound*

The results presented in Section 5.1 show that the implementation of the lower bound algorithm requires further work, particularly in the 6- to 8-week models. The distance between the upper and lower bound does not reduce significantly on successive iterations (Cases 4 and 5) and in many instances, the lower bound is not computed at all (Case 1, 2 and 3). This occurs because the lower bound subproblems are not feasible for some scenarios when constraining the ending storages. A small change was made to the algorithm in the middle of testing to increase the chances of computing feasible lower bound subproblems by including the origin as part of the convex combination of waiting storage vectors, which resulted in some improvement in the lower bound computation (Cases 4 and 5). Additional enhancements similar to

the one above could aid in the implementation of the lower bound but were not explored in this research.

From a theoretical perspective, the lower bound algorithm shows promise because of the successful computation of a lower bound in the 4-week model (Case 7) and somewhat successful in computation of a lower bound in the 6-week model (Case 6) and an in the 8-week model (Cases 4 and 5) after making an algorithmic change. However, in the 6- and 8-week models, the lower bound does not converge as rapidly as expected and more work needs to be done from an implementation perspective to work out some algorithmic and/or implementation oversights that could aid in this convergence.

5.2.2 *Inconclusive Gap Algorithm Results*

The existing problems with the lower bound implementation cause problems when using the gap algorithm to choose waiting storage vectors. Because the gap algorithm chooses a waiting storage vector based on the difference in the upper and lower bound, it is necessary for both an upper and lower bound to be computed correctly. The 8-week model that selects the waiting storage vectors using the gap algorithm (Case 5) has the same network setup as the 8-week models that select the waiting storage vectors using the random algorithm (Cases 2, 3 and 4). For this reason, the expectation was that each of these test cases would eventually converge on the same solution, but this was not the case. The final solution using the gap algorithm is larger ($305,046,000) than the final solutions using the random algorithm ($304,379,000 to $304,549,000). Because these objective function values are so large, the percent difference between these values does not appear to be

significant. However, Figure 32 (in Section 5.3) shows clearly that the two algorithms are not converging on the same solution.

This problem occurs for a particular reason. Because the gap between the upper and lower bound is not decreasing significantly on successive iterations, the solution procedure appears to get locked into solving for the waiting storage vector for the same arc on every iteration. By definition, the gap algorithm will choose the waiting storage vector that has the biggest gap at a given node, but if the total gap does not change significantly on successive iterations, it is likely that the gap at a given node is not changing significantly and therefore solving for the same scenario trace on every iteration. In particular, these results show that the algorithm is choosing the waiting storage vector that produces the largest subproblem objective function value and is therefore producing an overestimate of the final objective function. This final solution is a feasible solution to the stochastic network but not necessarily an accurate solution.

Because of the existing problems in the lower bound implementation, it is premature to make any conclusions regarding the gap algorithm. However, future work could resolve these problems and allow for further analysis of the gap algorithm.

5.2.3 *Future Research to Improve Lower Bounds and Gap Algorithm*

In order for any definite conclusions to be made regarding the lower bound and gap algorithm, further research is needed. First, more testing could be done to determine exactly where the problems exist and reveal if the problem is an algorithmic or implementation oversight. Additionally, as the algorithm functions

currently, during an attempt to compute the lower bound for the entire network, if a lower bound subproblem that is infeasible is encountered, computing the total lower bound during the current iteration is abandoned. It is possible to handle these infeasible subproblems more robustly so that a lower bound can be computed. It could also be possible to construct the lower bound problems so that they are not infeasible. An attempt was made in this research to include the origin as part of the convex combination of waiting storage vectors, and this approach improved the computation of the lower bound. Additionally, other approaches could be explored.

Fixing current implementation problems with the lower bound computation is expected to fix the problems that were observed with the gap algorithm. If the gap between the upper and lower bound changed more rapidly, it would be less likely that the gap algorithm would follow the same scenario path on each iteration and would pick waiting storages from a different predecessor arc on each iteration. However, changes could also be made to the gap algorithm to force this situation to not occur. For example, if the algorithm chooses the same waiting storage vector on several iterations it could be forced to choose a different waiting storage vector. It could also be possible to periodically choose waiting storage vectors randomly so as to guarantee some variation in the network solution.

5.3 Comparisons of Test Cases

Results of the test cases presented in Section 5.1 are discussed and compared in the following sections. In Section 5.3.1, the upper bound convergence of all test cases is reviewed and then the upper bound solutions of the 8-week correlated

random networks are discussed in detail. In Section 5.3.2, the short-term objective values are compeared in order to answer some of the questions posed in Chapter 4.

Table 18 contains the results for all the test cases. Throughout this section portions of the table will be repeated and compared for runs that differ in only one attribute. Each case number can be mapped to its specific attributes as presented in Table 4 (Chapter 4) but for ease of explanation, each test case will be referred to by its identifying attributes rather than by its case number when appropriate throughout this section.

The results in Table 18 include the end of stage 1 storage values for each cut reservoir, the short-term objective function value, and the total objective function value for all test cases studied. Cases 1 through 5 are the 8-week NSP networks. Case 1 is an uncorrelated network, and Cases 2 through 5 are correlated networks. In Cases 2, 3 and 4, the waiting storage vectors are chosen randomly, and in Case 5 the duality gap is used to choose waiting storage vectors. Case 6 is a 6-week NSP network, and Case 7 is a 4-week NSP network. Case 8 is a 6-week network with goal programming constraints.

Table 18: Comparison of run results

Storage in 10^8 m^3

	VPS	CASE 1	CASE 2	CASE 3	CASE 4	CASE 5	CASE 6	CASE 7	CASE 8
Cherokee	9.488089	9.263667	9.446894	9.358805	9.488089	9.355591	9.488089	9.488089	9.70729
Douglas	4.029506	3.727313	3.80326	3.922726	3.797147	3.691462	3.689715	3.769278	4.745804
Hiwassee	2.492492	2.246887	2.333013	2.301008	2.256071	2.219752	2.424008	2.492492	2.583223
Norris	16.84761	16.77872	16.77872	16.77872	16.77872	16.77872	16.77872	16.85854	17.22298
Fontana	10.89582	10.78107	10.79902	10.799887	10.79903	10.73457	10.8151	10.88944	10.81567
Short-term Value (10^3)	16,495	17,548	17,210	17,229	17,263	17,629	17,119	16,725	13,454
Total Objective (10^3)	**158,389**	**312,653**	**304,464**	**304,379**	**304,549**	**305,046**	**264,198**	**223,335**	**205,493**

5.3.1 *Upper Bound Convergence*

5.3.1.1 Upper Bound Convergence of NSP Algorithm

The results presented in Section 5.1 show that the upper bound does converge on a solution within a reasonable number of iterations. Increased cut sharing in the NSP algorithm was expected to lead to rapid convergence of the upper bound and the results confirm this. The upper bounds show only a small change in objective value after four to nine iterations for varying stage numbers. The results from the three 8-week test cases (Cases 2, 3 and 4) that randomly select the waiting storage vector are particularly encouraging because all three solutions are converging on a similar solution, albeit via different paths.

5.3.1.2 Upper Bound Convergence of 8-week Random Solutions

The three 8-week random correlated models (Cases 2, 3 and 4) use the same network setup (stages, states, arcs, etc.) and differ only in the waiting storage vectors chosen using the random algorithm. The cuts used to produce the future value estimate of one test case are valid cuts for any other test case that has the same network. In essence, these cuts just represent traces of the network that may vary from case to case, particularly in test cases where the waiting storage vectors are chosen randomly. The cuts generated from Case 2, 3 and 4 were combinded, and solved a new linear program for stage 1 was solved using CPLEX. In essence, combining these three cut sets creates a "mega-cut set" that contains three times as many cuts as any individual test case. This provides an even better estimate of the true future value function and allows an evaluation to determine if the test cases are truly converging after 13 iterations.

The results of this evaluation are shown in Table 19.

Table 19: Combined cut set results

	End of Week 1 Storage (10^8 m^3)
Cherokee	9.355591
Douglas	3.91735
Hiwassee	10.799887
Norris	2.301008
Fontana	16.778721

Total Objective Function Value (10^3)	Short-term Objective Function Value (10^3)
304,364	17,205

The objective function solution from this evaluation is plotted in Figure 32. Also shown in Figure 32 is the convergence of each of the three runs (Case 2, 3 and 4) that were used to generate this combined cut set, as well as the cut set from the 8-week solution that selects the waiting storage using the duality gap. In Figure 33 the final solution of the three runs (Case 2, 3 and 4) is presented along with the solution from the combined cut set in order to show the differences in the solutions at a more visible scale.

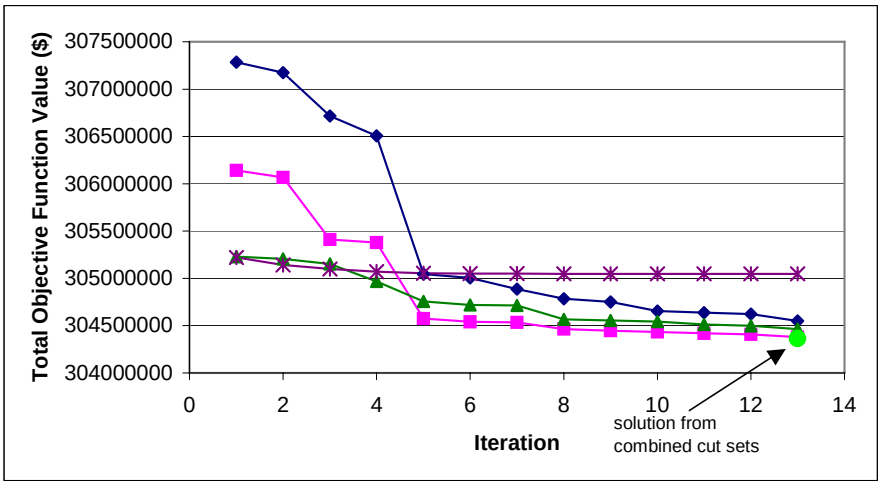

Figure 32: Convergence of Case 2, 3, 4 and 5 Along with the Solution from the Combined Cut Sets

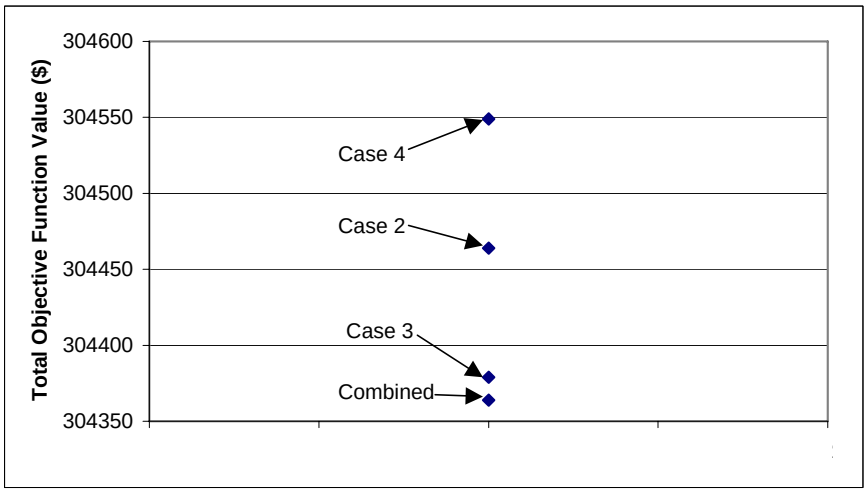

Figure 33: Difference in Final Objective Between Case 2, 3 and 4 and the Solution from the Combined Cut Sets

Figure 32 shows that the three solutions for the 8-week random algorithm models converge on a similar solution. The solution for the 8-week gap algorithm should also be converging on this same solution but clearly is not for reasons

mentioned previously. For this reason, the cut set from the gap algorithm has not been included in the mega-cut set.

In Table 20, the final objective function values for each of the three runs are presented as a percentage of the solution from the combined cuts sets. Also included in Table 20 is the total objective function value from the 8-week correlated model that uses the gap algorithm (Case 5) to show that it did not converge on the same solution as did the random algorithm models. The combined cut set is chosen as 100%, and the other solutions are all larger than this value.

Table 20: Objective function comparisons for Case 2, 3 and 4 along with the combined cut set

Case 5 Total Objective Function Value (10^3)	Case 2 Total Objective Function Value (10^3)	Case 3 Total Objective Function Value (10^3)	Case 4 Total Objective Function Value (10^3)	Combined Total Objective Function Value (10^3)
305,046	304,464	304,379	304,549	304,364

Case 5 Percentage of Combined Objective (%)	Case 2 Percentage of Combined Objective (%)	Case 3 Percentage of Combined Objective (%)	Case 4 Percentage of Combined Objective (%)	Combined Pecentage of Combined Objective (%)
100.224	100.033	100.005	100.061	100

The three test cases (Cases 2, 3 and 4) used to produce this additional constraint set have the same network setup and randomly select the waiting storage vectors. Because of the randomness in the solution procedure, the solutions converge with different paths, but after several iterations begin to converge on the same solution. In addition, the solution for the problem with the combined cut sets is lower than any of the three solutions used to generate this cut set. This is the

expected outcome because there are now more constraints in the form of cuts to approximate the upper bound.

Table 21 presents how close each solution (Case 2, 3 and 4) is to the combined cut set solution. This table shows that the solution from Case 3 is much closer to the solution for the mega-cut set than are Cases 2 and 4. This implies that the cuts from Cases 2 and 4 have relatively little additional information about the future value of water.

Even though these three models converge on similar objective function values, Table 18 shows that the end of week 1 storage value for each reservoir is somewhat different in each case. In Table 22 these differences can be analyzed more closely. The standard deviations reveal that the largest differences are at Cherokee, Douglas and Hiwassee with relatively little difference at Norris and Fontana. The coefficients of variation are extremely small, reflecting the fact that any reasonable change in reservoir storage in one week is miniscule in comparison to the total storage in these reservoirs.

Additionally, as shown in Table 23, the total volume of storage in the five reservoirs is very similar in all three cases (Case 2, 3 and 4). Thus, these solutions differ primarily in the choice of which reservoir from which to release water. This result reflects that in some instances, shifting water/energy from one reservoir to another has little effect on the solutions. This result is consistent with previous research and was the motivation for TVA's aggregate reservoir SDP model (discussed in Chapter 2).

Table 21: Difference between individual NSP solutions and mega-cut set solution

	3 Cut Soln ($10^8 m^3$)	Case 2 ($10^8 m^3$)	- 3 Cut Soln ($10^4 m^3$)	Case 3 ($10^8 m^3$)	- 3 Cut Soln ($10^4 m^3$)	Case 4 ($10^8 m^3$)	- 3 Cut Soln ($10^4 m^3$)
Cherokee	9.35559	9.44689	913	9.35881	32	9.48809	1325
Douglas	3.91735	3.80326	-1141	3.92273	54	3.79715	-1202
Hiwassee	2.30101	2.33301	320	2.30101	0	2.25607	-449
Norris	16.77872	16.77872	0	16.77820	-5	16.77872	0
Fontana	10.79989	10.79902	-9	10.79989	0	10.79903	-9
Total			83		81		-335

Table 22: Comparison of ending storage in NSP networks

	Case 2 ($10^8 m^3$)	Case 3 ($10^8 m^3$)	Case 4 ($10^8 m^3$)	Mean ($10^8 m^3$)	St. Dev. ($10^4 m^3$)	COV
Cherokee	9.44689	9.35881	9.48809	9.4313	660	0.00700
Douglas	3.80326	3.92273	3.79715	3.8410	708	0.01843
Hiwassee	2.33301	2.30101	2.25607	2.2967	387	0.01683
Norris	16.77872	16.77820	16.77872	16.7785	3	0.00002
Fontana	10.79902	10.79989	10.79903	10.7993	5	0.00005

Table 23: Total remaining volume comparison for all test cases

	VPS	CASE 1	CASE 2	CASE 3	CASE 4	CASE 5	CASE 6	CASE 7	CASE 8
Total Volume ($10^8 m^3$)	43.75352	42.79766	43.16091	43.16115	43.11906	42.7801	43.19563	43.49784	45.07497
% of Case 8*	0.971	0.949	0.958	0.958	0.957	0.949	0.958	0.965	1.000

*there is no significance to using case 8, this is just a means of comparing the values easily.

5.3.2 *Comparing Short-term Objective Values and Ending Storage Levels*

In this section the short-term objective values and the end of stage 1 storages values are compared between varying test cases. Several of the questions regarding NSP posed in Chapter 4 are reviewed and answered.

5.3.2.1 How do Solutions Between Correlated and Uncorrelated Networks Compare?

Table 18 shows that in the 8-week stochastic networks (Cases 1 through 5), the uncorrelated case (Case 1) has a larger short-term objective function than do the correlated networks that use the random algorithm (Cases 2, 3 and 4) and a slightly smaller short-term objective than does the correlated network that uses the gap algorithm (Case 5). However, as discussed earlier, the gap algorithm seems to over-estimate the objective function because it is selecting waiting storages based on the duality gap, and the current lower bound algorithm needs more work. The uncorrelated network has lower end of stage 1 storage values for the selected cut reservoirs, indicating a larger release during the first week. The cuts produced in the uncorrelated network imply a higher future value of water than do the cuts from the correlated network. The conjecture here is that because the uncorrelated model has less persistence of flows between stages, there is a lower likelihood to move toward extreme scenarios and water can be managed more easily without encountering losses such as spill.

5.3.2.2 How is the Solution Affected by the Number of Stages in the Network?

In our test cases, the correlated networks do show a difference in solutions based on the number of stages considered. The 4-week model (Case 7) has the

smallest short-term objective function value and the highest end of stage 1 storage values. The 6-week model (Case 6) has a higher short-term objective function value than does the 4-week model and lower end of stage 1 storage values. The 8-week models (Cases 2, 3 and 4) have the highest short-term objective function value and the lowest end of stage 1 storage values. In summary, the short-term objective function value increases and the end of stage 1 storage values decrease with additional stages. The NSP networks anticipate large inflows in the future and adjust the stage 1 releases accordingly

A conclusion is not drawn based on these results. In theory, the NSP solution could also hold back water in anticipation of low future inflows, but this situation was not explored. It is possible that the trend regarding the number of stages is due to the algorithm considering more stages in the network, but the trend is most likely due to larger inflows in weeks 6, 7 and 8.

Additional experiments are necessary to determine if the algorithm itself depends on the number of stages considered.

5.3.2.3 How do the NSP Solutions Compare with the VPS Solutions?

The results in Table 18 show that, in general, the end of week 1 storage values for all cut reservoirs using the VPS solution are larger than are the end of week 1 storage values for the NSP solutions. In some cases, the storage values are the same for some reservoirs, but the NSP values are never greater than the VPS values. Also, the short-term avoided cost using the VPS solution is lower than the short-term avoided cost for the NSP solutions. Both of these results indicate that the VPS solution places more value on using the water in the future than using the water now.

The NSP solutions release more water in the current stage, potentially in anticipation of large inflows that may arrive in future weeks. As mentioned in Chapter 4, the VPS estimates do consider future inflows based on averaging of historical hydrologic flows but the conjecture is that the averaging of scenarios underestimates large correlated future inflows.

Another way to view the differences between the VPS solution and the NSP solutions is to convert the storage volume differences to differences in the average outflow during the first stage. Table 24 presents the solution from the combined cut set of the three 8-week NSP networks as a representative sample of the NSP solutions to compare the outflow differences.

This comparison is done by computing the differences between the end of stage 1 storage values for the NSP solution and the VPS solution to get the differences in volumes, then divide by the number of seconds in a week to get the average outflow difference as a flow rate. In Table 24, column 1 contains the ending storage values for the combined cut set NSP solution, column 2 is the VPS solution, column 3 is the difference in storage levels, and column 4 is the average outflow difference.

Table 24: Average outflow differences between VPS and combined cut set NSP solution

	Combined Cut Set Solution ($10^8 m^3$)	VPS ($10^8 m^3$)	Difference ($10^4 m^3$)	Outflow Difference (cms)
Cherokee	9.35559	9.488089	1325	21.91
Douglas	3.91735	4.029506	1122	18.54
Hiwassee	2.30101	2.492492	1915	31.66
Norris	16.77872	16.84761	689	11.39
Fontana	10.79989	10.89582	959	15.86

As discussed earlier, the NSP solutions produce larger releases in the first stage. The results in Table 24 support this by showing that the solutions for the 8-week correlated random networks recommend an average increased outflow over the course of the first week of approximately 21.91 cms for Cherokee, 18.54 cms for Douglas, 31.66 cms for Hiwassee, 11.39 cms for Norris, and 15.86 cms for Fontana.

5.3.2.4 Can the NSP Algorithm Converge on a Solution when Considering Goal Programming Constraints?

As can be seen in Table 18, NSP does converge on a reasonable solution when goal programming constraints are considered. However, Figure 31 shows that the goal programming upper bound appears to converge after 4 iterations but that there is another large drop after the tenth iteration. Further analysis is required to determine if there is another drop in the upper bound beyond the thirteenth iteration.

5.4 Comparing Combined Short-term and Long-term Objective Values

Many of the test cases in this research are based on different stochastic models of future inflows. The VPS model assumes a deterministic value of ending storage levels. Case 1 assumes an uncorrelated stochastic network. Cases 2 through 5 assume an 8-week correlated network. Case 6 considers a 6-week network. Case 7 considers a 4-week network, and Case 8 considers goal programming constraints. These different stochastic models lead to different objective functions. Thus, a normal comparison of objective values is not possible. Each group of cases --VPS, uncorrelated, 8-week correlated, 6-week correlated, 4-week correlated and goal programming -- attempt to optimize their assumed objective. To the extent that one

of these assumptions is correct, the associated solution and convergence produces a near-optimal solution and the superiority of that solution can be believed.

With this limitation in mind, the best possible comparison of solutions from different groups is to evaluate them using the objective functions from alternative groups. The expectation is that a near-optimal solution for any objective function is better than a solution based on another objective. However, this is not guaranteed because none of the objectives were optimized to an absolute optimal solution.

The comparisons in this section are presented in tabular form with the rows representing the optimal short-term solution for a given case and the columns representing approximate alternative long-term objective functions, or criteria, for evaluating the solutions. The objective functions are only approximate because the final cut set is used to approximate the future value.

The mechanics of evaluating the solution for one test case using the objective function for another test case are as follows:

1. Solve the stage 1 optimization problem: short-term avoided cost + long-term future value. This is the final stage 1 solution for any selected test case.
2. Store the associated short-term value for the stage 1 solution.
3. Store the ending storage values for all cut reservoirs from stage 1 solution.
4. Solve a new linear program using the short-term value and storage vector from steps 2 and 3 paired with the long-term estimate from any other test case that has been solved.

For example, suppose we have solved one test case has been solved using the VPS estimate as the future value, and another test case has been solved using the NSP

network described in Case 1. To compare the future value estimate of these two cases, it is possible to store the VPS short-term objective value and ending storage values and solve an LP using the Case 1 future value constraints. Combining short-term and long-term values in this manner provides a gauge of the difference in the total objective function if a different future value estimate is used. Theoretically, each method to determine the short-term value will have the greatest combined objective using the same future value for which the problem was originally optimized; if all possible cuts were present, this result is guaranteed. The analysis here focuses on the difference in the combined objective value using a different future value estimate. The comparison can be thought of in this manner: if a decision is made for the short-term based on an assumption regarding the future, what would be the outcome if a different representation of the future value is actually the true representation?

This comparison can be achieved using the CPLEX solver. When RiverWare solves an optimization problem, it saves the problem as a text file that can be read by CPLEX. This allows editing of the file and it is possible to separate the short-term solution from the long-term solution, and solve a new problem using CPLEX directly.

This comparison was conducted with selected test cases. First, the comparison was made of representative samples of the 8-week networks to see how the NSP solutions compare with the VPS solutions. The correlated gap algorithm model (Case 5), the uncorrelated random algorithm model (Case 1), and the correlated random algorithm model (Case 2), were selected as the representative samples. In theory, Case 5 and Case 2 can be compared directly because they share

the same stochastic network. However, Case 2 and Case 5 are compared using their individual cut sets because Case 5 converged on a false optimum.

The results of this comparison are shown in Table 25. In the tables below, the test cases across the top represent the alternative future value criteria, and the cases on the left represent alternative solutions. The values combine the short-term and long-term objective values for each solution. For example, if the short-term value for the VPS solution and the long-term value from the uncorrelated random algorithm model are compared, the VPS row on the left intersects with the RAND/uncor column across the top at 312,555.

Table 25: Objective function comparison between VPS and 8-week NSP solutions

[Units: $\$10^3$]

	VPS	GAP/cor	RAND/uncor	RAND/cor
VPS	**158,389**	304,942	312,555	304,415
GAP/cor	158,261	**305,046**	312,652	304,457
RAND/uncor	158,298	305,043	**312,653**	304,451
RAND/cor	158,348	305,035	312,644	**304,464**

For simpler analysis, the difference in these objective function values is presented in Table 26.

Table 26: Additional objective function comparison between VPS and 8-week NSP solutions

[Units: $\$10^3$]

	VPS	GAP/cor	RAND/uncor	RAND/cor
VPS	0	-104	-98	-49
GAP/cor	-128	0	-1	-7
RAND/uncor	-91	-3	0	-13
RAND/cor	-41	-11	-9	0

As expected, when a short-term solution is paired with its respective long-term solution (i.e., VPS-VPS), that combination produces the greatest total objective. However, the results of most interest are the outcome of making a stage 1 decision (i.e., VPS recommended release schedule) if the true future value function is a different function (i.e., an 8-week correlated NSP network).

These results show that if the NSP stage 1 release schedule were implemented but the true future value estimate was the VPS estimate, the total objective function value would be roughly $41,000 to $128,000 less than if the NSP future value represented the true future value. On the other hand, if the VPS short-term solution were implemented, but the NSP network more correctly captures the future value estimate, the total objective function would be roughly $49,000 to $104,000 less than if the VPS future value function represented the true future value.

The correlated random solutions (Case 2) paired with the VPS solutions produce the smallest difference in the total objective function. Table 18 shows that the correlated random (Case 2) short-term objective function and end of stage 1 storage levels are more similar to the VPS short-term objective and end of stage 1 storage levels than are the uncorrelated (Case 1) and correlated gap algorithm (Case 5) solutions. For this reason, it makes sense that the combination of these two solutions produces the closest total objective. Additionally, the NSP solutions when paired are much closer to each other than when paired with the VPS solution

As discussed earlier, the uncorrelated network (Case 1) and the correlated network using the gap algorithm (Case 5) seem to over-estimate the total objective function value. Table 25 shows that combining the uncorrelated model's short-term

objective with the correlated gap long-term objective (and vice versa) produces very similar total objective functions (approximate differences of roughly $1,000 to $3,000). This result is surprising because the correlated gap algorithm network is identical to the correlated random network and should produce solutions closer to these solutions.

This comparison is also performed for the 8-week, 6-week and 4-week networks in order to analyze how the number of stages affects the NSP solution. Case 2 is chosen as the representative 8-week model. The results of this comparison are shown in Table 27.

Table 27: Objective function comparison for alternate number of stages

[Units: 10^3]

	VPS	4-week	6-week	8-week
VPS	**158,389**	223,313	264,157	304,415
4-week	158,386	**223,335**	264,189	304,446
6-week	158,357	223,321	**264,198**	304,462
8-week	158,348	223,314	264,195	**304,464**

Again, for simpler analysis, the difference in these objective function values is shown in Table 28.

Table 28: Additional objective function comparison for alternate number of stages

[Units: 10^3]

	VPS	4-week	6-week	8-week
VPS	**0**	**-22**	**-41**	**-49**
4-week	**-3**	**0**	**-9**	**-18**
6-week	**-32**	**-14**	**0**	**-2**
8-week	**-41**	**-21**	**-3**	**0**

These results show that as the number of weeks increases, the solutions deviate more from the VPS solutions. This result makes sense because, as can be seen in Table 18, the 4-week (Case 7) short-term release schedule is closer to the VPS solution than are the 6-week (Case 6) and 8-week (Cases 2, 3 and 4) release schedules. As mentioned in Chapter 4, the largest inflows during the time period that these networks represent arrive in weeks 6, 7 and 8. As these results show, the NSP solutions with more weeks release more water in the first stage than in the VPS solutions, likely in anticipation of larger future inflows.

5.5 Sensitivity Analysis

For an additional comparison, selected test cases are used to compare the effect of changing the end of stage 1 storage levels on the marginal value of water (reduced cost). These results are shown in Table 29.

Table 29: Reduced cost comparison for selected test cases

	VPS Reduced Cost ($/10^3 m^3)	CASE 1: RAND/uncor Reduced Cost ($/10^3 m^3)	CASE 2: RAND/cor Reduced Cost ($/10^3 m^3)	CASE 5: GAP/cor Reduced Cost ($/10^3 m^3)
Cherokee	9.431141	7.951616	8.529282	8.06938
Douglas	8.062289	6.90675	7.415767	6.926875
Hiwassee	11.776052	22.202485	23.72331	21.71432
Norris	9.210425	10.502421	11.009091	10.333205
Fontana	27.202103	7.922123	8.370175	8.253574

There is no obvious trend in these results other than biggest differences between the VPS and NSP solutions are at Hiwassee and Fontana.

Chapter 6

SUMMARY AND CONCLUSIONS

6.1 Answers to Questions Regarding NSP

This research has shown that the NSP algorithm produces reasonable results, with possible improvement over TVA's existing VPS solution, depending on what assumption about the future value is the most accurate representation of the future value of water in an individual basin. Particularly successful was the computation of an upper bound on the NSP network. The upper bound solutions generate reasonable release schedules and converge on a solution quite rapidly due to increased cut sharing in the NSP network. The results from the test cases answer the first question posed regarding NSP: the upper bound solution does converge on a solution and does so in a reasonable number of iterations (approximately 4 to 9 iterations) for the networks of varying stage length used in this research.

Some previous work has shown that SPR is not dependent on the number of stages in the network (Jacobs et al., 1995). For the test cases used in this research, fewer stages resulted in smaller stage 1 releases. However, this is not to say that this occurrence is a result of the algorithm considering additional stages; rather it is more likely due to the large differences in inflows in weeks 6, 7 and 8 as opposed to earlier

weeks. As mentioned previously, the time period considered in these test cases is the beginning of the spring fill season in the TVA basin, with the largest inflows arriving in weeks 6, 7 and 8. The 6-week and 8-week models anticipate these large future inflows and release more water in stage 1 than does the 4-week model. It is possible that if weeks 6, 7 and 8 did not have such large inflows, that the release recommendations from the 4-week network would be closer to release recommendations for the 6-week and 8-week networks showing limited dependency on the number of stages. This characteristic of the high inflows arriving in weeks 6, 7 and 8 is a seasonal phenomenon that occurs during the spring snowmelt in the TVA basin. If alternative seasons were considered where there was not such significant difference in flows between the early and late weeks, it is likely that considering more weeks would not produce such noticeable differences. However, the ability to anticipate these large future inflows in a multi-stage network is attractive, especially if one is confident in the future forecasts.

The model results presented also answer the remaining questions posed regarding the NSP algorithm. In the research test cases, there is a difference in results between the uncorrelated and correlated networks. Specifically, the uncorrelated model has a larger total objective function value and recommends larger stage 1 releases than do the correlated networks. Additionally, the VPS solutions recommend even smaller releases than do the NSP solutions.

Further research could determine to what extent the results depend on the state definition. As discussed previously, the previous weeks inflow is an admittedly simple state definition. A more complex, and perhaps realistic, state definition could

potentially show different patterns. Similarly, future research could determine if these results hold up for alternative network setups (additional stages, states, arcs, etc.).

6.2 Limited Success of the Lower Bound

Implementation of the lower bound algorithm was not entirely successful. More testing needs to be done to determine what problems exist in the current implementation of the algorithm and what enhancements could be made to make the implementation more useful. Future work is also necessary to determine if the proposed approach to computing a lower bound in the NSP algorithm is a viable approach. It appears that the approach is viable because of the success with the 4-week model and limited success with the longer models. Currently, any conclusions about the lower bound would be premature.

The current state of the lower bound implementation also means it is premature to make any conclusions regarding selecting waiting storages based on the duality gap.

6.3 Ready for Improved State Definition

It has been demonstrated from an algorithmic perspective, that NSP is able to produce reasonable release schedules using this study's simple state definition. It would be very useful to study more realistic state definitions. From a practical implementation perspective, if states could be defined that better represent the physical characteristics of the basin, and if more effort could be put into forecasting

scenario probabilities, then the results would be more accurate, and larger improvement over the VPS solution might be realized.

6.4 Additional Future Work

In addition to the future work mentioned above, there are also additional properties of the NSP network that would be interesting to explore. For this research, a relatively small sample of NSP networks were selected in order to determine if the NSP algorithm produces a reasonable solution and to analyze those factors to which the network is sensitive. Future work could explore additional network setups that contain more states per stage, more arcs per stage, differing stage lengths, different time periods, additional cut reservoirs, or any number of alternative network representations. With the addition of additional cut reservoirs, it is expected that convergence might be slightly slower but the algorithm would remain unchanged. It would also be useful to give attention to analyzing the NSP algorithm using goal programming.

Other work that could be explored involves the boundary conditions at the end of the planning horizon. In this research, TVA's VPS estimates were used. However, when TVA produces these VPS curves, a different curve is produced for every week of the year. In this study the week 1 VPS curve is used for all stochastic networks regardless of the number of weeks the stochastic network considered. A more accurate choice is to use the VPS curve that would actually be used if the VPS model is implemented at the end of the planning horizon. Such a change could change the differences in the VPS solutions and the NSP solutions. For basins other than the

Tennessee Valley, alternative boundary conditions such as target storage values could also be explored.

Future work could also enhance this research to a commercial implementation level. As mentioned earlier, the focus of this research was to increase efficiency based on the number of subproblems solved, with no work done to improve individual subproblem solution time. Currently, each subproblem is created from scratch and takes roughly 30 to 55 seconds on a fast Windows machine. Because, these problems share a similarity in structure, it would be possible to save the warm basis of these problems and dramatically reduce solution time. Based on previous experience, it is believed that the solution time could be reduced to 1 to 5 seconds.

Additional improvements could also be made regarding the use of this algorithm in RiverWare. The work done in RiverWare for this research focuses mostly on NSP from a research perspective. The GUI interface for creating and viewing the stochastic network and the solutions could be improved.

6.5 Final Conclusions

In this research we have shown that Network Stochastic Programming successfully produces reasonable reservoir release schedules without running into the exponential growth of scenario trees traditionally associated with Stochastic Programming with Recourse algorithms by reducing the scenario tree to a network of hydrologic states. We showed that the NSP algorithm converges on a solution quickly with relatively little iteration. With additional research regarding the definition of the hydrologic state of the NSP network and accurate modeling of the stochastic hydrology, this algorithm could be applied to TVA's operation models, as

well as any other basin that wishes to optimize reservoir scheduling. In our test cases, the percent difference of the final objective function solutions was relatively small when compared to TVA's existing VPS solutions due to extremely large value of the large volume of water remaining in storage after one week. However, extrapolated over the course of an entire year, the benefit of the NSP algorithm has the *potential* of being millions of dollars, which makes the NSP algorithm a procedure worth considering for stochastic reservoir scheduling.

REFERENCES

Bellman, R.E., 1952. On the Theory of Dynamic Programming. *in* Proceedings of the National Academy of Sciences, USA, 38, 716-719.

Bender, J.F., 1962. Partitioning Procedures for Solving Mixed Variables Programming Problems. *Numerische Mathematik,* 4, 238-252.

Biddle, S.H., 1999. RiverWare Applications at TVA, *in* Proceedings of American Society of Civil Engineers *WaterPower '99*, Las Vegas, Nevada, July 6−9, 1999.

Biddle, S.H., 2005. Optimizing TVA's Hydro System Using RiverWare. *in* Proceedings of RiverWare Users Group Meeting, Boulder, CO, March 1-2, 2005.

Birge, J.R., and F. Louveaux, 1997. *Introduction to Stochastic Programming*. Springer-Verlag, New York.

Condevaux-Lanloy, C., and E. Fragniere, 1998, SETSTOCH: A Tool for Multistage Stochastic Programming with Recourse. Logilab Technical Report, Department of Management Studies, University of Geneva, Switzerland, August 1998..

Eschenbach, E.A., T. Magee, E. Zagona, M. Goranflo, and R. Shane, 2001. Multiobjective Operations of Reservoir Systems via Goal Programming. *Journal of Water Resources Planning and Management,* 127, 108-120.

Faber, B.A., and J.R. Stedinger, 2001. Reservoir optimization using sampling SDP with ensemble streamflow prediction (ESP) forecasts. *Journal of Hydrology*, 249, 113-133.

Gilbert, K.C., 1985. Management Science in Reservoir Operation. *Applications of Management Science,* 4, 107-130.

Grantz, K.A., 2003. Using Large-Scale Climate Information to Forecast Seasonal Streamflow in the Truckee and Carson Rivers. Master's Thesis, Department of Civil, Environmental, and Architectural Engineering, University of Colorado, Boulder.

Hillier, F.S., and G.J. Lieberman, 2001. *Introduction to Operations Research.* McGraw-Hill, New York..

Huang, W., R Harboe, and J.J. Bogardi, 1991. Testing Stochastic Dynamic Programming Models Conditioned on Observed or Forecasted Inflows. *Journal of Water Resources Planning and Management,* 117, 28-36.

Infanger, G., and D.P. Morton, 1996. Cut Sharing for Multistage Stochastic Linear Programs with Interstage Dependency. *Mathematical Programming* 75, 241-256.

Jacobs, J., G. Freeman, J. Grygier, D. Morton, G. Schultz, K. Staschus, and J. Stedinger, 1995. Stochastic Optimal Coordination of River-Basin and Thermal Electric Systems (SOCRATES): A System for Scheduling Hydroelectric Generation Under Uncertainty. *Annals of Operation Research*, 59, 99-133.

Kelman, J., J.R. Stedinger, L.A. Cooper, E. Hsu, and S.Yuan, 1990. Sampling Stochastic Dynamic Programming Applied to Reservoir Operation. *Water Resources Research*, 26, 447-454.

Labadie, J.W., 2004. Optimal Operation of Multireservoir Systems: State-of-the-Art Review. *Journal of Water Resources Planning and Management,* 130, 93-111.

Loucks, D.P., J.R. Stedinger, and D.A. Haith, 1981. *Water Resource Systems Planning and Analysis.* Prentice Hall, Inc., Englewood Cliffs, New Jersey.

Morton, D.P., 1996. An Enhanced Decomposition Algorithm for Multistage Stochastic Hydroelectric Scheduling. *Annals of Operation Research,* 64, 211-235.

Pacific Gas and Electric Company, Web Page: http://www.pge.com/.

Periera, M.V.F., 1989. Optimal Stochastic Operations Scheduling of Large Hydroelectric Systems. *International Journal of Electrical Power & Energy Systems*, 11,161-169.

Periera, M.V.F., and L.M. Pinto, 1985. Stochastic Optimization of a Multireservoir Hydroelectric System – A Decomposition Approach. *Water Resources Research*, 21, 779-792.

Pereira, M.V.F., and L.M. Pinto, 1991. Multi-Stage Stochastic Optimization Applied to Energy Planning. *Mathematical Programming*, 52, 359-375.

Periera, M.V.F., G.C. Oliveira, C.G. Costa, and J. Kelman, 1984, Stochastic Streamflow Models for Hydroelectric Systems. *Water Resources Research*, 20, 379-390.

Regonda, S.K., 2004. Ph.D. candidate, University of Colorado, personal communication, September 2004.

Rotting, T.A., and A. Gjelsvik, 1992. Stochastic Dual Dynamic Programming for Seasonal Scheduling in the Norwegian Power System. *IEEE Trans. Power Systems*, 7, 273-279.

Stedinger, J.R., B.F. Sule, and D.P. Loucks, 1984. Stochastic Dynamic Programming Models for Reservoir Operation Optimization. *Water Resources Research*, 20, 1499-1505.

Shane, R. M. and Gilbert, K. C. (1982), "TVA Hydro Scheduling Model", J. Water Resources Planning and Management, 108

Stapleton, 2004. Ph.D. candidate, University of Colorado, personal communication, September 2004.

Tejada-Guibert, J.A., S.A. Johnson, and J.R. Stedinger, 1995. The Value of Hydrologic Information in Stochastic Dynamic Programming Models of a Multireservoir System. *Water Resources Research,* 31, 2571-2579.

Tennessee Valley Authority*, Energy, Environment, and Economic Development Web Page*: http://www.tva.gov/

U.S. Army Corps of Engineers, 1991. Optimization of Multiple-Purpose Reservoir System Operations: A Review of Modeling and Analysis Approaches. Hydrologic Engineering Center, January 1991.

Velasquez, J., 2002. GDDP: Generalized Dual Dynamic Programming Theory. *Annals of Operations Research,* 117, 21-31.

Velasquez, J., P.J. Restepo, and R. Campo, 1999. Dual Dynamic Programming: A note on Implementation. *Water Resources Research*, 35, 2269-2271.

Watkins, D.W., D.C. McKinney, L.S. Lasdon, S.S. Nielsen, and Q.W. Martin, 1999. A Scenario-Based Stochastic Programming Model for Water Supplies from the Highland Lakes. *in* International Transactions in Operational Research 1999.

Watkins, D.W., and W. Wei, 2004. Scenario-Tree Framework for Streamflow Forecasting and Water Resources Decision Support. *in* transactions of the World Water Congress 2004.

Wunderlich, W.O., 1989. Experiences with water management model development and use. *in* Proceedings of the Baltimore Symposium, May, 1989. IAHS Publ. No. 180.

Yang, M., and E.G. Read, 1999, A Constructive Dual DP for a Reservoir Model with Correlation. *Water Resources Research,* 35, 2247-2257.

Zagona, E.A., T.J. Fulp, R. Shane, T. Magee, and H. Morgan Goranflow, 2001. RiverWare: A Generalized Tool for Complex Reservoir System Modeling. *Journal of the American Water Resources Association,* 37, 913-929.