INFORMATION TO USERS

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

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

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

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

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

A Splitting Algorithm for Multistage Stochastic Programming with Application to Hydropower Scheduling

by

David H. Salinger

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

Doctor of Philosophy

University of Washington

1997

Approved by _____
(Chairperson of Supervisory Committee)

_____

_____

_____

Program Authorized
to Offer Degree Applied Mathematics

Date 23 May, 1997

In presenting this dissertation in partial fulfillment of the requirements for the Doctoral degree at the university of Washington, I agree that the Library shall make its copies freely available for inspection. I further agree that extensive copying of this dissertation is allowable only only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. copyright law. Requests for copying or reproduction of this dissertation my be referred to University Microfilms, 1490 Eisenhower Place, P.O. Box 975, Ann Arbor, MI 48106, to whom the author has granted "the right to reproduce and sell (a) copies of the manuscript in microform and/or (b) copies of the manuscript made from microform."

Signature David H Sachs

Date 23 May, 1997

University of Washington

Abstract

A Splitting Algorithm for Multistage Stochastic Programming with Application to Hydropower Scheduling

by David H. Salinger

Chairperson of Supervisory Committee: *Professor R. Tyrrell Rockafellar*
Dept. of Applied Mathematics

With water in short supply and utility deregulation imminent, the problem of long-term scheduling on a hydroelectric power system has become increasingly important.

Many of the algorithms developed for this large-scale multistage stochastic problem have relied on the dynamic programming algorithm; and so have suffered from the "curse of dimensionality." The current fashion is to use a Benders decomposition type algorithm. These have proven successful for the linear or piecewise linear problem, but due to the costs involved, a nonlinear model is more realistic.

Here, we present a new algorithm for the *nonlinear* (convex) multistage stochastic programming problem (MSP); one that is reasonable for the large-scale (*e.g.* hydro scheduling) problem and is easily parallelizable.

Development of the algorithm draws on many areas of optimization theory including: duality theory, saddle points, subgradients, scenario trees, Fenchel conjugacy, dynamic programming and splitting methods for monotone operators.

The algorithm is based on an application of Spingarn's operator splitting method to the saddle point problem associated with the MSP. The splitting method imposes a decomposability which results in two main subproblems to be solved at each itera-

tion. One is reformulated as an unconstrained linear-quadratic dynamic programming problem and is solved via a linear feedback loop solution extended to the scenario tree structure. The other subproblem is separable into sub-subproblems for each decision state. In the specialization to the hydropower setting, the main sub-subproblem is solved utilizing properties of the Fenchel conjugate function.

The algorithm is employed for a test problem developed from data provided by the Pacific Gas and Electric Co. consisting of the *equivalent* of 176 powerhouses and 174 controlled spillways on 94 reservoirs for a 24 month planning horizon. This results in an optimization problem in 165,200 control variables (water release) and 44,368 state variables (water storage).

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

I wish to thank my advisor, Terry Rockafellar, for his infinite patience and even greater wisdom in guiding me up the mountain that became this dissertation project. Early stages resembled a North Cascades bushwhack while later stages had more of the elegance of a high alpine ridge climb.

I would like to thank the other members of my thesis committee, professors Loyce Adams, Jim Anderson and especially Jim Burke for their efforts and council throughout.

Thanks go also to "unofficial committee member" Dr. Gary L. Schultz, Team Lead, Systems Engineering/Operations Research at the Pacific Gas and Electric Co. in San Francisco for providing the data which made this project come alive for me and for providing important technical advice.

I wish to thank my parents, Evelyn and Gerhard Salinger, for their love and support and for teaching me that I can accomplish anything I set my mind to.

The support of my many friends was of the utmost importance. Thanks to my mountaineering/skiing pals for sharing countless days of beauty and adventure in the North Cascades and elsewhere. Thanks to my city friends and school friends for making Seattle, the University of Washington and the Applied Mathematics department great places to be.

Chapter 1

INTRODUCTION

In any system where humans have some control, the desire is to run that system *optimally.* This leads to questions about how to define optimality for that system. Further questions may be raised about how to optimize when there are aspects of the system which can not be predicted with certainty. With these and other modeling considerations comes also the difficulty of solution techniques for the resulting stochastic optimization problem.

An example of such a system is long term scheduling on a hydroelectric power generation system. With water in short supply and utility deregulation imminent, this has become an increasingly important problem.

Hydropower systems researchers since the 1960's have been developing models and algorithms for optimal scheduling on a hydroelectric power system (cf. reviews by Yakowitz, 1982 and Yeh, 1985). At the same time, mathematical programming researchers in and out of the hydro field have been developing algorithms for two-stage and, more recently, multistage stochastic programming.

Early power system research concentrated on single reservoir models and on the dynamic programming algorithm for solving these problems (Review in Section 3.3). Through the 1980's, many of the algorithms were based on variations on deterministic and stochastic dynamic programming. But the notorious "curse of dimensionality," arising from discretization of the state space, meant that a 5 reservoir/powerhouse problem was considered extremely large computationally. More recently, research has

focused on a stochastic programming approach where it is unnecessary to discretize the state space. This work is returning solutions for much larger and more sophisticated power systems models. In particular, Benders decomposition type algorithms have seen much success for the linear and piecewise linear problems (*e.g.* Pereira and Pinto, 1985, 1991, and Jacobs et al., 1995). But it is well known that the costs involved are not linear. Thus an algorithm for the nonlinear case is desired.

In this paper, we present a new algorithm for the nonlinear (convex) multistage stochastic programming problem; one that is practical for the large-scale problem (*e.g.* hydro scheduling). Development of the algorithm draws on many areas of optimization theory including: duality theory, saddle points, subgradients, scenario trees, Fenchel conjugacy, dynamic programming and splitting methods for monotone operators.

The algorithm is then specialized for the hydropower scheduling problem and is implemented for a test problem consisting of the equivalent of 176 powerhouses on 94 reservoirs with 174 additional controlled water spillways over a 24 month planning horizon. The test problem was constructed from data provided by the Pacific Gas and Electric Co. (PG&E) in Northern California. The resulting optimization problem has 165,200 control variables (water release) and 44,368 state variables (water storage).

Hydropower

A hydropower system can consist of a network of power generation plants, each often with a connected reservoir and dam, in series on parallel and converging rivers in a drainage basin or in many basins. The volume of water released through and around each hydro plant can be controlled throughout time. System dynamics dictates water availability at each hydro plant. Constraints on the release and storage of water are set in accordance with system capacities as well as with contractual or congressionally mandated levels; concerning, for example, flood control, irrigation, recreation and fish and wildlife habitat. Due to uncertainties in rain and snow-fall as well as snow-melt,

the monthly natural (uncontrolled) inflow at each reservoir is also uncertain. Streamflow forecasts can be utilized and historical inflow data can be used to construct inflow probability distributions. Considering these and other factors, we desire to operate the power system in an optimal manner. This means, for example, that we wish to set monthly release levels so as to minimize expected costs associated with supplying electrical power.

A defining feature of such models is the desire for a succession of decisions (say monthly release of water through a powerhouse). The problem is not so difficult if all of the inflow information is considered to be known beforehand (the deterministic problem). But, since future inflows can not be known with certainty, we base our decisions, in part, on probabilistic inflow information as well as current storage levels and, possibly, inflow forecasts for the coming month. Future decisions are made as future information is revealed.

The deterministic problem can model benefits of running the system for an "average" or specific (inflow) year, but does not weigh these benefits against the costs incurred by running the system in that way during a different year. The importance of this is compounded by the nonlinear nature of the value of stored water. A deterministic model also does not weigh benefits against costs of system failure such as inability to meet power demand or to anticipate river flooding — events which do not happen in an "average" year. Thus, a stochastic model is desirable.

Stochastic Programming

In stochastic optimization, we desire to make an optimal decision (or succession of decisions) based, in part, on information which can not be known with certainty at the time of the decision. Stochastic programming is characterized by the opportunity, after initial decisions, to make *recourse* decisions in response to additional developments in information. After new information is revealed, a decision is chosen optimally with respect to that information and to expectations on future information.

A dynamical constraint (the *dynamics*) is included to track the effects of previous decisions (through the definition of an auxiliary state vector).

Stochastic programming employs techniques of large scale nonlinear programming. In doing this, the dynamics is thought of as "just another constraint." Thus it is unnecessary to discretize the state space as is done in dynamic programming.

Throughout this work, a scenario tree will be used to structure the information and decisions. The structure of the tree is provided by the discrete probability distribution. The recourse decisions are made in response to the realization of the various outcomes in the distribution. A particularly attractive feature of the scenario tree structure is the ease in representing the expected cost, which we will clearly wish to minimize.

The Algorithm

Concurrent to the construction of a stochastic programming model for hydropower scheduling must be thoughts of an appropriate algorithm. The choices involved are, of course, suggested by the structure of the problem. Fortunately our problem is, by design, convex. Still, constrained nonlinear (and non-quadratic) optimization problems are notoriously difficult to solve; dynamic and stochastic problems are even more so.

Often in constrained problems, one attempts to include constraints in the objective function via penalty formulations or Lagrange multipliers. Here, a *reduced* Lagrangian is used to include the state constraints in the objective. In the implementation of the algorithm for the hydro scheduling problem, an *augmented* reduced Lagrangian is seen to provide better state-constraint compliance.

Once the Lagrangian is formed, it is typical to look for advantage between the associated primal and dual problems. Here, it is the third associated problem, the *saddle point problem*, where we find the structural advantages which key the algorithm.

As is typical in dynamic optimization problems, our problem shows little decomposable structure on the surface. To impose some decomposability and thereby reduce the problem to manageable-sized subproblems, an operator splitting method is employed. The Peaceman-Rachford and Douglas-Rachford splitting methods are well known to the numerical analysis and PDE communities. But these methods are not appropriate for the multivalued and nonlinear operators which arise here. Instead, Spingarn's method for operator splitting (Spingarn, 1983, 1985) is employed.

The splitting results in two main subproblems to be solved at each iteration. One becomes an unconstrained dynamic programming problem with linear dynamics and quadratic cost (with scenario tree information/decision structure). It is solved via a linear feedback loop (extended to the scenario tree structure in Section 2.3.3). The other subproblem becomes a separable convex box-constrained minimization which, in the specialization to the hydro setting, is solved by utilizing Fenchel conjugate functions (cf. Rockafellar 1984) to solve the associated dual problem. It should be noted that the crucial separable structure of the second subproblem arises only through the appropriate splitting of the *saddle point problem* associated with the reduced Lagrangian.

The imposed decomposability yields an algorithmic structure which can be easily adapted to parallel computation. In the hydropower setting, the dynamic subproblem is separable by distinct watershed; yielding a parallel problem for each watershed. The second subproblem, which is to be solved in parallel to the first, is separable into problems for each node of the scenario tree. These can be advantageously grouped for parallel computation.

Outline

Stochastic programming and the scenario tree, Lagrangians and saddle points are discussed in Chapter 2. Related topics in stochastic optimization such as the extension of dynamic programming to a scenario tree structure, Benders decomposition and the

progressive hedging algorithm are also discussed.

In Chapter 3, a stochastic programming network-based model for the hydropower scheduling problem is developed and improvements based on a model in use at PG&E (Jacobs et al., 1995; personal communication with Gary L. Schultz (PG&E), 1995) are discussed. The last section surveys the evolution of stochastic optimization for the hydro scheduling problem as reported in the engineering literature.

Chapter 4 provides a review of the proximal point method and monotone operator splitting methods as well as the development of the new *dynamic splitting* algorithm for the convex multistage stochastic programming problem. Convergence results are also provided.

The algorithm is specialized for the hydropower scheduling problem in Chapter 5. Performance-enhancing modifications are discussed and a test problem, with results, is presented. Algorithm performance is discussed in terms of the duality gap and the norm of constraint violations. The final section compares the duality gap results with those of an infeasible test problem.

Chapter 6 provides conclusions and potential topics for later work.

Contributions

The main accomplishment of this work is the development of an algorithm for the *nonlinear* (convex) multistage stochastic programming problem (Section 4.6); one that is practical for the large-scale problem and is easily adapted for parallel computation.

Other major contributions include: the development of a Fenchel conjugate based algorithm for the form of the separable subproblem arising in the specialization of the dynamic splitting algorithm for the hydro scheduling problem (Section 5.3); and, in the implementation, the development of performance enhancing modifications and parameter settings (Section 5.5).

Major points in the development include: the extension of the linear-quadratic case of dynamic programming to a scenario tree structure (Section 2.3.3); the recon-

ciliation of the typical dynamics in stochastic programming with the typical dynamics used in a forecast-based hydro scheduling model (Section 3.1.5); and the extension and application of Spingarn's Method to the saddle point problem arising from a multistage stochastic programming problem. Optimality conditions for the convex multistage stochastic programming problem and results guaranteeing linear convergence of the algorithm are also provided.

Other important accomplishments include: the implementation of the algorithm for a hydropower scheduling test problem with over 165,000 control variables; and the investigation of the duality gap (Section 5.6) and comparison with duality gap results from an infeasible version of the test problem (Section 5.7).

Chapter 2

SADDLE POINTS AND PROBLEM FORMULATION IN STOCHASTIC PROGRAMMING

This chapter provides review, extension and introduction of some pertinent results on saddle points, on problem formulation for multistage stochastic programming and on dynamic programming. Section 2.1 reviews Lagrangian functions and the associated primal, dual and saddle point problem formulations. Section 2.2 describes the modeling structure of stochastic programming, develops a saddle point problem formulation for the multistage stochastic programming problem and provides necessary conditions for optimality. A new algorithm for the convex case is presented later (in Chapter 4). The algorithm relies in part on a feedback loop solution for the unconstrained linear-quadratic dynamic programming problem. That well-known solution result is extended in Section 2.3 to a scenario tree structure. The stochastic programming (SP) and stochastic DP approaches are also contrasted. Brief descriptions of Benders decomposition and of the progressive hedging algorithm follow in Section 2.4.

First, some definitions.

Definition 2.1 *Consider any extended-real-valued function* $h: \Re^n \to \Re \cup \{+\infty\}$, *then*

(a) The (effective) domain of h *is* dom $h = \{x \in \Re^n | h(x) < \infty\}$.

(b) h *is* proper *if* dom $h \neq \phi$.

(c) h *is said to be* lower semi-continuous *if* $\{x | h(x) \leq \alpha\}$ *is closed for every* $\alpha \in \Re$.

(d) h *is* convex *if for any* $x_1, x_2 \in \Re^n$ *and for every choice of* $\tau \in (0,1)$,

$$h(\tau x_1 + (1-\tau)x_2) \leq \tau h(x_1) + (1-\tau)h(x_2).$$

A proper function is lower semi-continuous if and only if it is *closed*, thus the term closed will also be used throughout.

Rather than a definition for *relative interior* (which would necessitate further definitions), the following characterization (Rockafellar, 1970a) will suffice.

Remark 2.2 (Characterization of Relative Interior) *Let C be convex and non-empty in $\Re^n$. Then $\overline{x} \in$* ri C *(the* relative interior *of C) if and only if, for every $x \in C$, there exists an $\varepsilon > 0$ such that*

$$\overline{x} + \varepsilon(\overline{x} - x) \in C.$$

This condition means that if $x \in C$, every line segment in C having $\overline{x}$ as an endpoint can be prolonged beyond $\overline{x}$ without leaving C.

2.1 Saddle Points

A *saddle point* of a function L over a product set $U \times V$ is a pair $(\overline{u}, \overline{v}) \in U \times V$ such that

$$L(u, \overline{v}) \geq L(\overline{u}, \overline{v}) \geq L(\overline{u}, v) \quad \text{for all } u \in U,\ v \in V.$$

This means that the minimum of $L(u, \overline{v})$ with respect to $u \in U$ is attained at $\overline{u}$, while the maximum of $L(\overline{u}, v)$ with respect to $v \in V$ is attained at $\overline{v}$. Thus, an equivalent saddle point condition is

$$\min_{u \in U} L(u, \overline{v}) = L(\overline{u}, \overline{v}) = \max_{v \in V} L(\overline{u}, v).$$

Saddle points can also be specified as a gradient condition in terms of the normal cones to U and V ($N_X(\overline{x})$, the normal cone to a convex set X at $\overline{x}$, is the set of all vectors normal to X at $\overline{x}$ (cf. Rockafellar, 1970a)):

Proposition 2.3 (Saddle Point Gradient Condition) *Assume that U and V are closed convex subsets in $\Re^n$ and $\Re^m$ and that L is continuously differentiable (C^1). A first-order necessary condition for $(\overline{u}, \overline{v})$ to be a saddle point of L over $U \times V$ is that*

$$-\nabla_u L(\overline{u}, \overline{v}) \in N_U(\overline{u}), \quad \nabla_v L(\overline{u}, \overline{v}) \in N_V(\overline{v}).$$

Further, this gradient condition is a sufficient condition for a saddle point when $L(u, v)$ is convex in u for each $v \in V$ and concave in v for each $u \in U$.

2.1.1 The Ordinary Lagrangian

One context where saddle points are of interest occurs when the function L on $U \times V$ is a Lagrangian function associated with some minimization problem. As a simple example, consider the following minimization problem in *conventional format*:

$$\underset{u \in U_0 \subset \Re^n}{\text{minimize}} \; f_0(u) \;\text{ subject to }\; f_i(u) \begin{cases} \leq 0, & \text{for } i = 1 \ldots s \\ = 0, & \text{for } i = s+1 \ldots m \end{cases} \qquad (\mathcal{P}_0)$$

where the f_i are C^1 and $U_0 \subset \Re^n$ is closed, convex and nonempty.

The ordinary Lagrangian associated with problem $(\mathcal{P}_0)$ is the function

$$L_0(u, v) = f_0(u) + v_1 f_1(u) + \cdots + v_m f_m(u)$$
$$\text{for } \; u \in U_0, \; v \in V_0 = \Re^s_+ \times \Re^{m-s}.$$

Theorem 2.5 will relate the saddle point of L_0 to optimality for problem $(\mathcal{P}_0)$ (via the gradient condition of Proposition 2.3).

Other Lagrangian functions can also be constructed for $(\mathcal{P}_0)$. We will see in Sections 2.1.2 and 2.2.2 various ways in which these can arise. In general, any function L on a product space $U \times V$ is a Lagrangian for some problem.

Definition 2.4 (Basic Constraint Qualification) *(cf. Rockafellar, 1993b) For a feasible solution $\overline{u}$ to problem $(\mathcal{P}_0)$, the* basic constraint qualification *is fulfilled at $\overline{u}$*

if only *the vector* $v = (0, \ldots, 0)$ *satisfies*

$$-[v_1 \nabla f_1(\overline{u}) + \cdots + v_m \nabla f_m(\overline{u})] \in N_{U_0}(\overline{u})$$

$$\text{with} \begin{cases} v_i \geq 0, & \text{for } i \in [1, s] \text{ active at } \overline{u} \\ v_i = 0, & \text{for } i \in [1, s] \text{ inactive at } \overline{u} \\ v_i \text{ free}, & \text{for } i \in [s+1, m]. \end{cases}$$

Theorem 2.5 (Optimality for $(\mathcal{P}_0)$) *(cf. Rockafellar, 1993b) Let* $\overline{u}$ *be a feasible solution to problem* $(\mathcal{P}_0)$.

(Necessary.) If $\overline{u}$ *is locally optimal and the basic constraint qualification is fulfilled at* $\overline{u}$, *then there exists a vector* $\overline{v} = (\overline{v}_1, \ldots, \overline{v}_m)$ *such that*

$$-\nabla_u L(\overline{u}, \overline{v}) \in N_{U_0}(\overline{u}), \quad \nabla_v L(\overline{u}, \overline{v}) \in N_{V_0}(\overline{v}).$$

Equivalently

$$-[\nabla f_0(\overline{u}) + \overline{v}_1 \nabla f_1(\overline{u}) + \cdots + \overline{v}_m \nabla f_m(\overline{u})] \in N_{U_0}(\overline{u})$$

$$\text{with} \begin{cases} \overline{v}_i \geq 0, & \text{for } i \in [1, s] \text{ active at } \overline{u} \\ \overline{v}_i = 0, & \text{for } i \in [1, s] \text{ inactive at } \overline{u} \\ \overline{v}_i \text{ free}, & \text{for } i \in [s+1, m]. \end{cases}$$

(Sufficient.) If such a vector $\overline{v}$ *exists, and* f_0 *and the feasible set are convex, then* $\overline{u}$ *is globally optimal in* $(\mathcal{P}_0)$ *and* $(\overline{u}, \overline{v})$ *is a saddle point of* L_0 *on* $U_0 \times V_0$

Notice that for the convex case of $(\mathcal{P}_0)$ (where $f_0, f_1, \ldots, f_s$ are convex and $f_{s+1}, \ldots, f_m$ are affine on U_0 convex), the necessary condition for optimality (called the *Kuhn-Tucker conditions*) is equivalent to the saddle point condition of Proposition 2.3 for the associated Lagrangian L_0 on $U_0 \times V_0$. Thus, under the constraint qualification, for there to exist a local optima $\overline{u}$ in the convex case of $(\mathcal{P}_0)$, it is necessary that there exist a $\overline{v}$ such that $(\overline{u}, \overline{v})$ is a saddle point of $L_0(u, v)$ on $U_0 \times V_0$. Also, in the convex case, existence of a saddle point $(\overline{u}, \overline{v})$ of L_0 is sufficient to show that $\overline{u}$ is globally optimal in $(\mathcal{P}_0)$.

2.1.2 *Generalized Lagrangians*

For a set $U \in \Re^n$ and a function $f : \Re^n \to \Re \cup \{+\infty\}$, consider the problem

$$\underset{u \in U}{\text{minimize}}\ f(u). \qquad (\mathcal{P})$$

In general, for some set $V \in \Re^m$ a function $L : U \times V \to [-\infty, \infty]$ is a *Lagrangian* for problem $(\mathcal{P})$ when

$$\sup_{v \in V} L(u, v) = f(u) \quad \text{for } u \in U.$$

Clearly, more than one such triple, consisting of L, U, V exists such that L is a Lagrangian associated with problem $(\mathcal{P})$.

As an illustration of the relationship between the primal problem and an associated Lagrangian, revisit problem $(\mathcal{P}_0)$ and the related Lagrangian function L_0 on $U_0 \times V_0$.

It is not hard to see that for $U = U_0$, problems $(\mathcal{P})$ and $(\mathcal{P}_0)$ are equivalent if

$$f(u) = \begin{cases} f_0(u), & \text{if all constraints in } (\mathcal{P}_0) \text{ are satisfied,} \\ +\infty, & \text{otherwise.} \end{cases}$$

Notice also that $\sup_{v \in V_0} L_0(u, v) = f(u)$, thus confirming the assertion that L_0 is a Lagrangian for problem $(\mathcal{P}_0)$ on $U_0 \times V_0$.

Primal Problem Formulation

Above, we were given the (primal) problem of minimizing a function f over a set U. From that we defined, for a set V, a (Lagrangian) function L on $U \times V$. Alternatively, given sets $U \subset \Re^n$ and $V \subset \Re^m$ and *any* function L on $U \times V$, a function f on U can be defined and the associated primal problem can be formulated

$$\underset{u \in U}{\text{minimize}}\ f(u) \quad \text{where } f(u) = \sup_{v \in V} L(u, v). \qquad (\mathcal{P})$$

Dual Problem Formulation

A second optimization problem related to the function L on $U \times V$ arises similarly to the primal problem. The *dual* problem is formulated

$$\underset{v \in V}{\text{maximize}}\ g(v) \quad \text{where } g(v) = \inf_{u \in U} L(u, v). \tag{$\mathcal{D}$}$$

Notice how g is defined on V analogously to f on U in the primal problem $(\mathcal{P})$.

From the relationship $f(u) \geq L(u, v) \geq g(v)$ for $u \in U,\ v \in V$, it is clear that

$$[\inf(\mathcal{P})] \geq [\sup(\mathcal{D})]$$

where $\inf(\mathcal{P})$ and $\sup(\mathcal{D})$ represent the optimal values in problems $(\mathcal{P})$ and $(\mathcal{D})$. We will be especially interested in the case where the optimal values are equal.

Saddle Point Problem Formulation

A third related problem, the saddle point problem, can be formulated from the definition of a saddle point of L on $U \times V$ as

$$\begin{aligned} &\text{find } (\overline{u}, \overline{v}) \in U \times V \\ &\text{such that : } L(\overline{u}, \overline{v}) = \min_{u \in U} L(u, \overline{v}) = \max_{v \in V} L(\overline{u}, v). \end{aligned} \tag{$\mathcal{S}$}$$

If a saddle point exists for L on $U \times V$, the optimal values in $(\mathcal{P})$ and $(\mathcal{D})$ must coincide. Then, the problems $(\mathcal{P})$, $(\mathcal{D})$ and $(\mathcal{S})$ can provide three formulations, each possibly with distinct advantages, for finding $(\overline{u}, \overline{v})$. The algorithm to be presented later relies on a structural advantage found only in the saddle point formulation.

The following proposition explains the relationship between the solutions of problems $(\mathcal{P})$, $(\mathcal{D})$ and $(\mathcal{S})$ and provides a context for the duality theorems which follow.

Proposition 2.6 (Characterization of Saddle Points)

$$(\overline{u}, \overline{v}) \textit{ is a solution to } (\mathcal{S}) \iff \begin{cases} \overline{u} \textit{ is a solution to } (\mathcal{P}), \\ \overline{v} \textit{ is a solution to } (\mathcal{D}), \\ \inf(\mathcal{P}) = \sup(\mathcal{D}). \end{cases}$$

Saddle Point Existence

So, under what conditions does a saddle point exist? The following two theorems (Rockafellar, 1970a) supply useful criteria relating saddle point existence and primal optimality. The second theorem is a consequence of Fenchel's duality theorem.

Theorem 2.7 (Duality Criteria) *Consider a convex-concave, continuous function L on the product $U \times V$ of closed convex sets as well as the associated functions f and g defined in $(\mathcal{P})$ and $(\mathcal{D})$ above.*

(a) Suppose that $-g$ is proper on V and that the set $\{v \in V | g(v) \geq \alpha\}$ is bounded for all $\alpha \in \Re$. Then $\overline{u}$ is optimal in $(\mathcal{P})$ if and only if there exists $\overline{v}$ such that $(\overline{u}, \overline{v})$ is a saddle point of L on $U \times V$.

(b) If in addition, f is proper on U and the set $\{u \in U | f(u) \leq \alpha\}$ is bounded for all $\alpha \in \Re$, then an optimal $\overline{u}$ exists.

Theorem 2.8 (Fenchel Duality Criteria) *Consider the closed proper convex functions $h : \Re^n \to \Re \cup \{+\infty\}$ and $k : \Re^m \to \Re \cup \{+\infty\}$ and let $U =$dom h and $V =$dom k. Define the function $L(u, v) = h(u) - k(v) + v \cdot Mu$ for $u \in U$, $v \in V$, where M is a linear transformation from $\Re^n$ to $\Re^m$. Then, in $(\mathcal{P})$, $f(u) = h(u) + k^*(Mu)$ for $u \in U$ and, in $(\mathcal{D})$, $g(v) = -k(v) - h^*(-M^\top v)$ for $v \in V$ where h^* and k^* are the Fenchel conjugate functions of h and k. Also;*

(a) Suppose that there exists a $u \in$ ri (U) such that $Mu \in$ ri (dom k^*). Then $\overline{u}$ is optimal in $(\mathcal{P})$ if and only if there exists $\overline{v}$ such that $(\overline{u}, \overline{v})$ is a saddle point of L on $U \times V$.

(b) If in addition, there exists a $v \in$ ri (V) such that $-M^\top v \in$ ri (dom h^*), then an optimal $\overline{u}$ exists.

It should be noted that the conditions in part (a) of Theorems 2.7 and 2.8 play the role of the basic constraint qualification (Definition 2.4) in Theorem 2.5.

It can be shown that when h is piecewise linear, the corresponding "ri" (for h and h^*) can be omitted in Theorem 2.8. The same can be shown for k^* and k. Also the "ri" can be omitted for the special case of extended linear-quadratic programming structure (described at the end of Section 2.2.2) (Rockafellar and Wets, 1986).

2.2 *Modeling Structure in Multistage Stochastic Programming*

One of the major results to be presented in this thesis (Chapter 4) is the development of a new algorithm for the constrained convex multistage stochastic programming problem. This section starts by describing modeling structure for these multistage stochastic programming problems (MSP). Much of the presentation follows Rockafellar (1993a). The section concludes by specializing the saddle point theorems for the convex case of MSP.

In stochastic optimization, we desire to make an optimal decision (or sequence of decisions) based on information which can not be known with certainty at the time of the decision. Stochastic programming is characterized by the opportunity, after initial decisions, to make *recourse* decisions in response to additional developments in information. After new information is revealed, a decision is chosen optimally with respect to that information and to expectations on future information. A dynamical constraint (the *dynamics*) is included to track the effects of previous decisions (through the definition of an auxiliary state vector).

2.2.1 The Scenario Tree

The underlying information and decision structure of the multistage stochastic model is the scenario tree (see Figure 2.1). The nodes $i \in \mathcal{I}$ of the tree are the information/decision states (also referred to as states or nodes). Node i^- is the unique predecessor of i. The notation i^+ represents the variable index ranging over the successors of i, while $\{i^+\}$ represents the set of all successors of i. The set of terminal

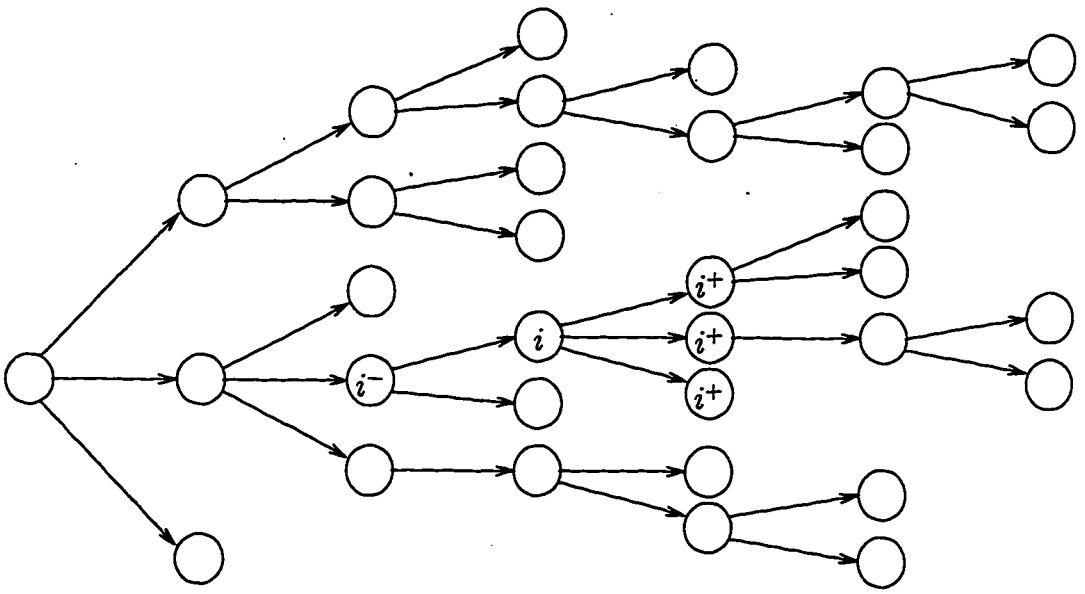

Figure 2.1: Scenario tree.

states (nodes with no following branches) is denoted by $\mathcal{T} \subset \mathcal{I}$.

The transition probability $p_i > 0$ represents the fixed probability of moving from state i^- to state i along a branch of the tree. The *scenario* corresponding to state i is the set of branches leading from the initial state to that state. The scenario probability $\pi_i > 0$ represents the probability of arriving at that state i. It is the product of the transition probabilities along that scenario.

For each information/decision state $i \in \mathcal{I}$, an optimal decision (or control) vector $u_i \in \Re^{n_i}$ is chosen; this defines a mapping $u : i \mapsto u_i$ for all $i \in \mathcal{I}$. We can also view u as a super-vector of controls in $\Re^N = \Pi_{i \in \mathcal{I}} \Re^{n_i}$.

A particularly attractive feature of the scenario tree structure is the ease in representing the *expected cost*, which we may wish to minimize. The expectation is conditioned on the successor states i^+. If $G_i(u_i, u_{i^-}, \ldots, u_0)$ is the cost incurred at state i due to decision u_i and as a function of all previous decisions in that scenario,

then the

$$[\text{Expected Total Cost}] = \sum_{i \in \mathcal{I}} \pi_i G_i(u_i, u_{i^-}, \ldots, u_0).$$

If the G_i are convex in $(u_i, u_{i^-}, \ldots, u_0)$, then the expected cost is a convex function of the policy.

As well as providing an underlying structure for the decisions, the scenario tree also provides a structure for the information. Think of the scenario tree as a discrete probability distribution or as the discretization of a continuous probability distribution of dependent events. The probability of reaching a state i^+ is conditioned on the probability of reaching state i in the first place.

There is opportunity to respond (recourse) to developments of information on the scenario tree. That is, future decisions are made after the observation of information. Notice also that the development of information on the tree is not affected by interim decisions.

Problem Dynamics

As a way of accounting for the effects of past decisions, a state vector $x_i \in \Re^{s_i}$ is defined recursively (by the dynamics) as

$$x_{i+} = f_i(x_i, u_i, b_{i+})$$

where the $b_i \in \Re^{t_i}$ are random variables with discrete probability distribution defined on the scenario tree and where i^+ represents *any* of the states following state i. (The equivalent statement $x_j = f_i(x_i, u_i, b_j),\ \forall j \in \{i^+\}$ will also be used, particularly in representing a sum over members of $\{i^+\}$.) The components of b_i, x_i and u_i represent various aspects of the physical system in question at state $i \in \mathcal{I}$. Here, we will assume a linear form of the dynamics

$$x_{i+} = A_i x_i + B_i u_i + b_{i+}.$$

2.2.2 *The Cost Structure*

In general, the physical system will give rise to various constraints. These are incorporated into the cost objective via a penalty (possibly infinite) or multiplier formulation. Borrowing Rockafellar's (1993a) notation and examples, write this cost for each $i \in \mathcal{I}$ as

$$F_i(x_i, u_i) = r_i \cdot u_i + \varphi_i(u_i) + \psi_i^*(d_i - C_i x_i - D_i u_i) + c_i \cdot x_i$$

where the extended-real-valued penalty functions φ_i and ψ_i^* are convex, closed (lower semi-continuous) and proper and where ψ_i^* is the *Fenchel conjugate* (cf. Rockafellar, 1970a) of ψ_i given by

$$\psi_i^*(z_i) = \sup_{v_i \in \Re^{m_i}} \{v_i \cdot z_i - \psi_i(v_i)\}.$$

The General Multistage Stochastic Programming Problem

Collecting the cost and dynamics structures, the multistage stochastic programming problem (MSP) is formulated as follows:

$$\underset{u \in \Re^N}{\text{minimize}}\ f(u) = \sum_{i \in \mathcal{I}} \pi_i F_i(x_i, u_i) \qquad (\mathcal{P}_{msp})$$

$$\text{where: } F_i(x_i, u_i) = r_i \cdot u_i + \varphi_i(u_i) + \psi_i^*(d_i - C_i x_i - D_i u_i) + c_i \cdot x_i$$

$$x_{i+} = A_i x_i + B_i u_i + b_{i+}, \quad x_0 = b_0$$

where the x_i are viewed as dependent variables defined by the dynamics, where φ_i and ψ_i^* (and ψ_i) are extended-real-valued, convex, closed and proper, where $\pi_i > 0$ is the scenario probability, and where $\Re^N$ is the product of the $\Re^{n_i}$ for all $i \in \mathcal{I}$. We will see below how ψ_i^* arises as the Fenchel conjugate of ψ_i.

What follows are three examples of problem structures which can be formulated in this way. The first of which will be used to illustrate the duality structure. The first two examples will also illustrate aspects of the penalty structure.

Example 1. Linear Programming Structure

Problem $(\mathcal{P}_{msp})$ can exhibit a linear programming structure. For each $i \in I$, define the indicator function $\varphi_i(u_i) = \delta_{\Re^{n_i}_+}(u_i)$ to enforce the nonnegativity of u_i (where $\delta_X(x)$ has a value of 0 if $x \in X$ and $+\infty$ otherwise and where $\Re^n_+$ is the nonnegative orthant of $\Re^n$). Take also $\psi_i(v_i) = \delta_{\Re^{m_i}_+}(v_i)$. Then we will see below that the function $\psi_i^*(d_i - C_i x_i - D_i u_i)$ inflicts an infinite penalty in the minimization if the constraint $d_i - C_i x_i - D_i u_i \leq 0$ is not met (and is zero otherwise).

To gain insight into the duality structure of the above formulation, we examine the following linear program (where x_i is considered to be a function of u_i defined by the dynamics):

$$\begin{aligned} &\underset{u \in \Re^N}{\text{minimize}}\ f_0(u) = \sum_{i \in \mathcal{I}} \pi_i (r_i \cdot u_i + c_i \cdot x_i) \qquad (\mathcal{P}_{lp}) \\ &\text{subject to}: d_i - C_i x_i - D_i u_i \leq 0 \\ &\qquad\qquad u_i \geq 0 \\ &\qquad \text{where}: x_{i+} = A_i x_i + B_i u_i + b_{i+}, \quad x_0 = b_0 \end{aligned}$$

and where $\Re^N$ is the product set of the $\Re^{n_i}$ for all $i \in \mathcal{I}$.

Define $U_i = \{u_i \in \Re^n | u_i \geq 0\}$ and $V_i = \Re^{m_i}_+$. A Lagrangian associated with problem $(\mathcal{P}_{lp})$ (with Lagrange multiplier $v_i \in V_i$ for the linear constraint) is

$$L(u,v) = \sum_{i \in \mathcal{I}} \pi_i \{r_i \cdot u_i + \delta_{U_i}(u_i) + v_i \cdot (d_i - C_i x_i - D_i u_i) - \delta_{V_i}(v_i) + c_i \cdot x_i\}$$

$$\text{where:}\ \ x_{i+} = A_i x_i + B_i u_i + b_{i+}, \ \ x_0 = b_0$$

on $U \times V$ where U and V are each products of the U_i and V_i and where, using the standard convention in this setting, we assume that $\infty - \infty = \infty$.

To derive the primal problem from the Lagrangian, define

$$f(u) = \sup_{v \in \Re^M} L(u,v) \quad \text{on } U$$
$$= \sum_{i \in \mathcal{I}} \pi_i F_i(x_i, u_i)$$

$$\text{where: } F_i(x_i,u_i) = \sup_{v_i \in \Re^{m_i}} \left\{ r_i \cdot u_i + \underbrace{\delta_{\Re^{n_i}_+}(u_i)}_{\varphi_i(u_i)} + v_i \cdot \underbrace{(d_i - C_i x_i - D_i u_i)}_{z_i} - \underbrace{\delta_{\Re^{m_i}_+}(v_i)}_{\psi_i(v_i)} + c_i \cdot x_i \right\}$$

$$x_{i+} = A_i x_i + B_i u_i + b_{i+}, \quad x_0 = b_0.$$

Rewriting F_i gives

$$F_i(x_i,u_i) = r_i \cdot u_i + \varphi_i(u_i) + \underbrace{\sup_{v_i \in \Re^{m_i}} \{v_i \cdot z_i - \psi_i(v_i)\}}_{\psi_i^*(z_i)} + c_i \cdot x_i$$

which is the form of F_i suggested above (where ψ_i^* is the Fenchel conjugate of ψ_i). Collecting terms, we find the formulation of the primal problem $(\mathcal{P}_{lp})$ suggested in $(\mathcal{P}_{msp})$ (with $\varphi_i, \psi_i^*, \psi_i$ as above)

$$\underset{u \in \Re^N}{\text{minimize}}\ f(u) = \sum_{i \in \mathcal{I}} \pi_i F_i(x_i, u_i)$$

$$\text{where: } F_i(x_i,u_i) = r_i \cdot u_i + \varphi_i(u_i) + \psi_i^*(d_i - C_i x_i - D_i u_i) + c_i \cdot x_i$$

$$x_{i+} = A_i x_i + B_i u_i + b_{i+}, \quad x_0 = b_0$$

Here the indicator functions φ_i and ψ_i enforce the nonnegativity of the primal and dual variables. Applying the definition to find ψ_i^*, the Fenchel conjugate of $\psi_i(v_i) = \delta_{\Re^{m_i}_+}(v_i)$, we see that

$$\psi_i^*(d_i - C_i x_i - D_i u_i) = \begin{cases} 0, & \text{if } d_i - C_i x_i - D_i u_i \leq 0 \\ +\infty, & \text{otherwise} \end{cases}$$

as desired (where the vector inequality should be interpreted as holding for all components).

Consider instead, in $(\mathcal{P}_{lp})$, a linear constraint of the form $d_i - C_i x_i - D_i u_i \in \Gamma_i$ where $\Gamma_i = \Pi_{j=1}^{s_i}(-\infty, 0] \times \Pi_{j=s_i+1}^{m_i}\{0\}$. Then, take $V_i = \Pi_{j=1}^{s_i}[0, \infty) \times \Pi_{j=s_i+1}^{m_i}(-\infty, \infty)$ so that

$$\psi_i^*(d_i - C_i x_i - D_i u_i) = \begin{cases} 0, & \text{if } d_i - C_i x_i - D_i u_i \in \Gamma_i \\ +\infty, & \text{otherwise.} \end{cases}$$

Example 2. Piecewise Linear Programming Structure

For $(\mathcal{P}_{msp})$, a piecewise linear programming structure is formed similarly to the LP structure. To formulate the primal problem for this case, define, in $F_i(x_i, u_i)$, the functions

$$\varphi_i(u_i) = \delta_{U_i}(u_i), \quad \psi_i(v_i) = \delta_{V_i}(v_i)$$

where U_i, V_i represent box constraints on u_i and v_i. In the minimization, the conjugate function $\psi_i^*(d_i - C_i x_i - D_i u_i)$ inflicts piecewise linear penalties on $d_i - C_i x_i - D_i u_i$. As an example in one dimension, consider just the j^{th} component $z_{ij} \in \Re$ and the corresponding box constraint, $V_{ij} = [0, \gamma_{ij}]$, on v_{ij}. Then the resulting piecewise linear penalty (associated with that j^{th} component) is

$$\psi_{ij}^*(z_{ij}) = \begin{cases} 0, & \text{if } z_{ij} \leq 0 \\ \gamma_{ij} z_{ij}, & \text{if } z_{ij} \geq 0. \end{cases}$$

Similarly, if $V_{ij} = [\gamma_{ij}^1, \gamma_{ij}^2]$, then

$$\psi_{ij}^*(z_{ij}) = \begin{cases} \gamma_{ij}^1 z_{ij}, & \text{if } z_{ij} \leq 0 \\ \gamma_{ij}^2 z_{ij}, & \text{if } z_{ij} \geq 0. \end{cases}$$

If V_i is the product of finite intervals V_{ij}, then $\psi_i^*(z_i) = \sum_j \psi_{ij}^*(z_{ij})$. (See Rockafellar (1992) for a review of piecewise linear-quadratic penalties.)

Example 3. Extended Linear-Convex Programming Structure

A constrained convex programming structure can be represented in this extended problem $(\mathcal{P}_{msp})$ format with

$$\varphi_i(u_i) = \delta_{U_i}(u_i) + g_i(u_i), \quad \psi_i(v_i) = \delta_{V_i}(v_i) + \frac{1}{2} v_i \cdot Q_i v_i$$

where U_i, V_i are polyhedral constraints on u_i and v_i, g_i is convex and Q_i is positive semi-definite. Here, in the minimization, the conjugate function ψ_i^* inflicts piecewise linear-quadratic penalties on $d_i - C_i x_i - D_i u_i$.

The special case of *extended linear-quadratic programming* occurs for quadratic $g_i(u_i) = \frac{1}{2} u_i \cdot R_i u_i$ where R_i is positive semidefinite.

2.2.3 Problem Formulation; The Primal Problem

The convex programming structure of Example 3 provides a primal formulation of the *convex* multistage stochastic programming problem. Specifically, we will take, for each $i \in \mathcal{I}$, $\varphi_i(u_i) = g_i(u_i) + \delta_{U_i}(u_i)$ with g_i convex and finite on U_i and $\psi_i(v_i) = \delta_{V_i}(v_i)$. The sets U_i and V_i are the product sets of intervals (possibly infinite). Then Γ_i is also a product of intervals (possibly semi-infinite or singleton) related to V_i so that the conjugate

$$\psi_i^*(d_i - C_i x_i - D_i u_i) = \begin{cases} 0, & \text{if } d_i - C_i x_i - D_i u_i \in \Gamma_i \\ \text{linear or infinite penalties}, & \text{otherwise.} \end{cases}$$

With these specifications the convex problem is formulated

$$\underset{u \in \mathbb{R}^N}{\text{minimize}} \; f(u) = \sum_{i \in \mathcal{I}} \pi_i \left\{ r_i \cdot u_i + \varphi_i(u_i) + \psi_i^*(d_i - C_i x_i - D_i u_i) + c_i \cdot x_i \right\} \qquad (\mathcal{P}_{cmsp})$$

$$\text{where:} \quad x_{i+} = A_i x_i + B_i u_i + b_{i+}, \quad x_0 = b_0$$

where x_i is viewed as a dependent variable.

Notice that the term $\psi_i^*(d_i - C_i x_i - D_i u_i)$, in combination with the dynamics in $(\mathcal{P}_{cmsp})$, creates a nonlinear function of u_i, u_{i^-} as well as previous decisions on that

scenario. The algorithm (to be presented in Chapter 4) exploits a certain separable structure in i but needs (at highest order) *bilinear* terms in u_i, u_{i-}. The dual formulation displays similar higher order non-separable terms. But, by formulating the saddle point problem on the reduced Lagrangian, we shall see (in Chapter 4) that this stipulation can be met.

The Expanded Primal Problem

In problem $(\mathcal{P}_{msp})$, the $x_i \in \Re^{s_i}$ are viewed as dependent variables defined by the dynamics. In the following equivalent problem, the x_i are viewed as independent variables in addition to u_i and the dynamics are viewed as additional constraints. Then

$$\underset{u\in\Re^N,\, x\in\Re^S}{\text{minimize}}\ \overline{f}(x,u) = \sum_{i\in\mathcal{I}} \pi_i F_i(x_i,u_i) \qquad (\overline{\mathcal{P}}_{msp})$$

$$\text{subject to: } x_{i+} - \left[A_i x_i + B_i u_i + b_{i+}\right] = 0, \quad x_0 - b_0 = 0$$

$$\text{where: } F_i(x_i,u_i) = r_i{\cdot}u_i + \varphi_i(u_i) + \psi_i^*(d_i - C_i x_i - D_i u_i) + c_i{\cdot}x_i$$

where $\Re^S$ is the product of the $\Re^{s_i}$.

The Full Lagrangian

To formulate a Lagrangian for this expanded problem, introduce multipliers $y_i \in \Re^{s_i}$ for the dynamical constraints and multipliers $v_i \in \Re^{m_i}$ as before for the linear constraints. This *full Lagrangian* is

$$\begin{aligned}\overline{L}(x,u;y,v) \;=\; & \sum_{i\in\mathcal{I}} \pi_i \left\{r_i{\cdot}u_i + \varphi_i(u_i) + v_i{\cdot}(d_i - C_i x_i - D_i u_i) - \psi_i(v_i) + c_i{\cdot}x_i\right\} \\ & + \sum_{i\neq 0} \pi_i y_i{\cdot}\left\{x_i - [A_{i-}x_{i-} + B_{i-}u_{i-} + b_i]\right\} \;+ y_0{\cdot}\{x_0 - b_0\}\,.\end{aligned}$$

on $(X\times U)\times(Y\times V)$ where $X, Y \subset \Re^S$.

As in the examples above, the form chosen for $\psi_i(v_i)$ dictates the type of penalty inflicted on $d_i - C_i x_i - D_i u_i$ in the primal problem. Similarly, the form of $\varphi_i(u_i)$ dictates the type of penalty seen in the dual problem formulation.

The Expanded Dual Problem

A dual problem, similar to the primal problem $(\overline{\mathcal{P}}_{msp})$, can be formulated from the full Lagrangian with a dual dynamics relating y_i and v_i as follows:

$$\underset{v\in\Re^M,\, y\in\Re^S}{\text{maximize}}\ \overline{g}(y,v) = \sum_{i\in\mathcal{I}} \pi_i \Big\{ d_i\cdot v_i - \psi_i(v_i) - \varphi_i^*\big(\big(\sum_{j\in\{i^+\}} p_j B_i^\top y_j\big) + D_i^\top v_i - r_i\big) - b_i\cdot y_i \Big\} \qquad (\overline{\mathcal{D}}_{msp})$$

$$\text{subject to}: \quad y_i - \Big[\big(\sum_{j\in\{i^+\}} p_j A_i^\top y_j\big) + C_i^\top v_i - c_i\Big] = 0, \quad \text{for } i \notin \mathcal{T}$$

$$y_i - \big[C_i^\top v_i - c_i\big] = 0, \qquad \text{for } i \in \mathcal{T}$$

where φ_i^* is the Fenchel conjugate of φ_i.

The Reduced Lagrangian

In the *reduced Lagrangian formulation*, the state variable x_i is interpreted as a dependent variable defined by the dynamics. Similarly, the dual state variable y_i is defined by the dual dynamics. Then $L(u,v) = \overline{L}(x,u;y,v)$ on $U\times V$ where x and y are functions of u and v respectively and

$$L(u,v) = \sum_{i\in\mathcal{I}} \pi_i \left\{ r_i\cdot u_i + \varphi_i(u_i) + v_i\cdot(d_i - C_i x_i - D_i u_i) - \psi_i(v_i) + c_i\cdot x_i \right\}$$

$$\text{where:}\quad x_{i+} = A_i x_i + B_i u_i + b_{i+}, \quad x_0 = b_0.$$

One should recognize that the reduced Lagrangian was the form used in the development of $(\mathcal{P}_{msp})$ as well as the convex case $(\mathcal{P}_{cmsp})$ and examples above.

For completeness, the dual problem can be formulated from the reduced Lagrangian as

$$\underset{v\in\Re^M}{\text{maximize}}\; g(v) = \sum_{i\in\mathcal{I}} \pi_i \left\{ d_i \cdot v_i - \psi_i(v_i) - \varphi_i^* \Big(\big(\sum_{j\in\{i^+\}} p_j B_i^\top y_j \big) + D_i^\top v_i - r_i \Big) - b_i \cdot y_i \right\} \qquad (\mathcal{D}_{msp})$$

$$\text{where: } y_i = \big(\sum_{j\in\{i^+\}} p_j A_i^\top y_j \big) + C_i^\top v_i - c_i, \quad \text{for } i \notin \mathcal{T}$$

$$y_i - \left[C_i^\top v_i - c_i \right] = 0, \qquad \text{for } i \in \mathcal{T}.$$

Of course problems $(\mathcal{D}_{msp})$ and $(\overline{\mathcal{D}}_{msp})$ are equivalent as were the associated primal problems.

We now have the primal $(\mathcal{P}_{msp})$ and dual $(\mathcal{D}_{msp})$ problem formulations associated with the reduced Lagrangian. Because of its special structure we would also like to formulate the associated saddle point problem.

2.2.4 Saddle Point Formulation of the Multistage Stochastic Problem

The saddle point problem for the reduced Lagrangian L on $U \times V$ associated with the convex multistage problem is formulated

$$\text{find } (\overline{u}, \overline{v}) \in U \times V \qquad (\mathcal{S}_{msp})$$

$$\text{such that : } L(\overline{u}, \overline{v}) = \min_{u\in\Re^N} L(u, \overline{v}) = \max_{v\in\Re^M} L(\overline{u}, v)$$

where the restrictions $u \in U$ and $v \in V$ reside within the indicator functions in L and where, as previously mentioned, we use the convention that $\infty - \infty = \infty$.

The following two theorems, which relate the primal optimality for the convex problem $(\mathcal{P}_{cmsp})$ and existence of a saddle point of L on $U \times V$ (i.e. a saddle point solution in $(\mathcal{S}_{msp})$), follow from Theorems 2.7 and 2.8.

Theorem 2.9 (MSP Duality) *Consider the convex problem $(\mathcal{P}_{cmsp})$ and the associated reduced Lagrangian L on $U \times V$;*

(a) Suppose that for each $i \in \mathcal{I}$ there exists a $u_i \in \text{ri } (U_i)$ such that $d_i - C_i x_i - D_i u_i \in \text{ri } (\Gamma_i)$ where $x_{i+} = A_i x_i + B_i u_i + b_{i+}$, $x_0 = b_0$. Then $\overline{u}$ is optimal in $(\mathcal{P}_{cmsp})$ if and only if there exists $\overline{v}$ such that $(\overline{u}, \overline{v})$ is a saddle point of L on $U \times V$.

(b) Further, if for all $\alpha \in \Re$ the set $\{u_i \in U_i | g_i(u_i) + \psi_i^*(d_i - C_i x_i - D_i u_i) < \alpha\}$ is bounded for every $i \in \mathcal{I}$ where $x_{i+} = A_i x_i + B_i u_i + b_{i+}$, $x_0 = b_0$; then an optimal $\overline{u}$ exists.

Proof. In the context of Theorem 2.8, let $h(u) = \sum_{i \in \mathcal{I}} \pi_i (r_i \cdot u_i + \varphi_i(u_i) + c_i \cdot x_i)$ and $k(v) = \sum_{i \in \mathcal{I}} \pi_i \psi_i(v_i)$ and define the affine transformation M such that $v_i \cdot M u_i = \sum_{i \in \mathcal{I}} \pi_i v_i \cdot (d_i - C_i x_i - D_i u_i)$ where $x_{i+} = A_i x_i + B_i u_i + b_{i+}$, $x_0 = b_0$ and where $\varphi_i, \psi_i, U_i, V_i$ are recalled from problem $(\mathcal{P}_{cmsp})$.

Notice that the hypothesis of Theorem 2.8 is fulfilled and that the resulting Lagrangian $L(u, v) = h(u) - k(v) + v_i M u_i$ on $U \times V$ is the reduced Lagrangian associated with problem $(\mathcal{P}_{cmsp})$ where U and V are the product of the U_i and V_i. Then

$$k^*(Mu) = \sum_{i \in \mathcal{I}} \pi_i \big(\psi_i^*(d_i - C_i x_i - D_i u_i) + c_i \cdot x_i\big) \text{ on } U.$$

Applying condition (a) of Theorem 2.8, we arrive at condition (a) here.

Part (b) follows immediately from part (b) of Theorem 2.7. □

The Modified MSP

If, for a particular convex MSP, the condition (a) of Theorem 2.9 is not met, all is not lost. By modifying the MSP to insist that the V_i are bounded, the duality conditions in Theorem 2.10 will provide an alternative to those in Theorem 2.9.

The effect of of this modification is small. Originally, the V_i were allowed to be unbounded (for example $V_i = \Re_+^{m_i}$). As a result, the $\psi_i^*(d_i - C_i x_i - D_i u_i)$ term in the primal problem would inflict an infinite penalty if $d_i - C_i x_i - D_i u_i \notin \Gamma_i$. If instead the V_i are bounded (recall $V_i = \Pi_j [0, \gamma_{ij}]$ in Example 2), then the infinite penalty is replaced by a piecewise linear penalty. This modification will have little effect in

practice. If a bound (γ_{ij}) is found to be too restrictive, it can be increased (in the following run) so that the effect is like the unbounded case.

Theorem 2.10 (Modified MSP Duality) *Consider the reduced Lagrangian L on $U \times V$ associated with the convex problem ($\mathcal{P}_{cmsp}$) as well as the modification that the V_i are bounded boxes.*

(a) Then, $\overline{u}$ is optimal in ($\mathcal{P}_{msp}$) if and only if there exists $\overline{v}$ such that $(\overline{u}, \overline{v})$ is a saddle point of L on $U \times V$.

(b) Additionally, if f (defined in ($\mathcal{P}_{cmsp}$)) is proper on U and the set $\{u \in U | f(u) \leq \alpha\}$ is bounded for all $\alpha \in \Re$, then an optimal $\overline{u}$ exists.

Proof. The theorem follows directly from Theorem 2.7. Notice that the boundedness in part (a) of that theorem is easily satisfied as a result of the problem modification. □

2.2.5 The Augmented Reduced Lagrangian

Another form of the Lagrangian, which will prove useful in Chapter 5, is termed the *augmented* Lagrangian. As a step toward formulating the augmented reduced Lagrangian for problem ($\mathcal{P}_{cmsp}$), an augmented control variable $w_i \in \Re^{m_i}$ is introduced along with the stipulation that

$$w_i - x_i = 0.$$

The terms

$$v_i \cdot (d_i - C_i x_i - D_i u_i) - \psi_i(v_i)$$

of the reduced Lagrangian, possibly corresponding in the associated primal problem to the state constraint

$$d_i - C_i x_i - D_i u_i \in \Gamma_i,$$

are then replaced by

$$v_i \cdot (d_i - C_i w_i - D_i u_i) - \psi_i(v_i) + v_i' \cdot (w_i - x_i)$$

where the multiplier $v_i' \in \Re^{m_i}$ is introduced for the new constraint $w_i - x_i = 0$.

Then an augmented reduced Lagrangian associated with problem $(\mathcal{P}_{cmsp})$ can be formulated

$$L(u, w; v, v') =$$
$$\sum_{i \in \mathcal{I}} \pi_i \{r_i \cdot u_i + \varphi_i(u_i) + v_i \cdot (d_i - C_i w_i - D_i u_i) - \psi_i(v_i) + v_i \cdot (w_i - x_i) + c_i \cdot x_i\}$$
$$\text{where: } x_{i+} = A_i x_i + B_i u_i + b_{i+}, \quad x_0 = b_0$$

on $(U \times \Re^M) \times (V \times \Re^M)$ where φ_i and ψ are defined previously.

An Example

An example, which we will return to later, is given by the case when $d_i = 0$, $C_i = I$, $D_i = 0$ and $\psi_i(v_i) = \max\{\underline{s}_i v_i, \overline{s}_i v_i\}$ with $\underline{s}_i \leq \overline{s}_i$ where the max and inequality are taken component-wise.

In this case, the reduced Lagrangian associated with $(\mathcal{P}_{cmsp})$ is

$$L(u, v) = \sum_{i \in \mathcal{I}} \pi_i \Big\{ r_i \cdot u_i + \varphi_i(u_i) + v_i \cdot x_i - \psi_i(v_i) + c_i \cdot x_i \Big\}$$
$$\text{where: } x_{i+} = A_i x_i + B_i u_i + b_{i+}, \quad x_0 = b_0$$

on $U \times V$.

In the associated primal problem, the terms $v_i \cdot x_i - \psi_i(v_i)$ would have the effect of an infinite penalty if the state constraint

$$\underline{s}_i \leq x_i \leq \overline{s}_i$$

is not met (and be zero otherwise).

The augmented reduced Lagrangian would instead take the form

$$L(u,w:v) = \sum_{i\in\mathcal{I}} \pi_i \{r_i \cdot u_i + \varphi_i(u_i) + v_i \cdot (w_i - x_i) + \delta_{W_i}(w_i) + c_i \cdot x_i\}$$

$$\text{where: } x_{i+} = A_i x_i + B_i u_i + b_{i+}, \quad x_0 = b_0$$

$$W_i = \{w_i \in \Re^{m_i} | \underline{s}_i \leq w_i \leq \overline{s}_i\}$$

on $(U \times W) \times V$ where $W = \Pi_{i\in\mathcal{I}} W_i$.

One advantage of the augmented form is the absence of the non-smooth $\psi_i(v_i)$ term. Another advantage will be seen in Chapter 5 when an operator splitting (defined later) is used; inducing a splitting of the Lagrangian. This advantage will stem from the difference in the way that the terms associated with the state constraint can be grouped under that splitting.

The duality theorems (2.10 and 2.9 with minor modification) hold also for this augmented form of the Lagrangian where $(u,w) \in U \times W$ replaces the primal variable $u \in U$.

2.3 *Dynamic Programming*

Like multistage stochastic programming, dynamic programming (DP) examines situations where decisions are made in stages. The modeling structures of SP and DP share *some* similarities. Their differences will be highlighted below.

This section will introduce dynamic programming from the standpoint of an algorithm for solving problem ($\mathcal{P}_{msp}$) with its underlying scenario tree structure. Dynamic programming on a scenario tree structure has not previously been described in the literature.

This approach will allow a review of DP without getting involved in the measure-theoretic probability theory necessary to examine the expected total cost in general (cf. Bertsekas and Shreve 1978). It will also allow us to examine the DP algorithm (Section 2.3.2), and especially the unconstrained linear-quadratic case (section 2.3.3)

on a scenario tree structure.

In the DP algorithm, the minimization is taken over a function space of policies rather than a vector space as in SP. Despite this difference, the SP solution can be constructed from the DP solution in the end.

2.3.1 The DP Problem on a Scenario Tree

Recall the form of the multistage stochastic problem ($\mathcal{P}_{msp}$)

$$\underset{u \in U \subset \Re^N}{\text{minimize}} f(u) = \sum_{i \in \mathcal{I}} \pi_i F_i(x_i, u_i) \qquad (\mathcal{SP})$$

$$\text{where: } x_{i+} = A_i x_i + B_i u_i + b_{i+}, \quad x_0 = b_0$$

and recall that the solution is viewed as a policy (mapping) $u : i \mapsto u_i$ for $i \in \mathcal{I}$. Thus, the minimization is taken over the *finite-dimensional* set $U = \Pi_{i \in \mathcal{I}} U_i \subset \Re^N$.

In contrast, the DP solution is viewed as a policy (mapping) $u : x_i \mapsto u_i(x_i)$ for $i \in \mathcal{I}$ arising from a minimization over the *infinite-dimensional* (function) space of such mappings. The DP problem is then the minimization, of the expected cost, over the space $\mathbb{P}$ of all such policies with $u_i(x_i) \in U_i$ for $i \in \mathcal{I}$:

$$\underset{u(\cdot) \in \mathbb{P}}{\text{minimize}} \sum_{i \in \mathcal{I}} \pi_i F_i(x_i, u_i(x_i)) \qquad (\mathcal{DP})$$

$$\text{subject to: } x_{i+} = A_i x_i + B_i u_i(x_i) + b_{i+}, \quad x_0 = b_0,$$

where further restrictions on x_i may reside in F_i as infinite penalties.

In a typical DP formulation, x_k is the state of the system at time stage k. On the scenario tree, the pair (i, x_i) is needed to fully specify the state of the system at a corresponding "time stage".

2.3.2 The DP Algorithm

The dynamic programming algorithm works on Bellman's *principle of optimality* (Bellman, 1957). Theorem 2.11, which relies on the following constructions, provides

an optimality principle for the scenario tree based DP.

For any state $i_0 \in \mathcal{I}$, construct the subtree consisting of the set of states $\mathcal{I}^{i_0} \subset \mathcal{I}$ by taking $i_0 \in \mathcal{I}^{i_0}$ and adding states by the rule that if $i \in \mathcal{I}^{i_0}$, then $i^+ \in \mathcal{I}^{i_0}$.

On this subtree $\mathcal{I}^{i_0}$, define the *truncated problem* as the restriction of problem $(\mathcal{DP})$ to $\mathcal{I}^{i_0}$. The initial state for the truncated problem is defined as the state of the system $(\mathcal{DP})$ at state i_0 found by implementing the optimal policy $\{u_i^*(x_i)\}_{i \in \mathcal{I}}$.

Theorem 2.11 (Principle of Optimality for $(\mathcal{DP})$) *Let the policy $\{u_i^*(x_i)\}_{i \in \mathcal{I}}$ be an optimal policy for problem $(\mathcal{DP})$. Then, the policy $\{u_i^*(x_i)\}_{i \in \mathcal{I}^{i_0}}$ is optimal for the truncated problem.*

Proof. Assume that the policy $\{u_i'(x_i)\}_{i \in \mathcal{I}^{i_0}}$ is optimal for the truncated problem. But, since $\{u_i^*(x_i)\}_{i \in \mathcal{I}}$ is an optimal policy for the problem $(\mathcal{DP})$, no improvement could be made by replacing $u_i^*(x_i)$, for $i \in \mathcal{I}^{i_0}$, by $u_i'(x_i)$. (By construction, this replacement would not effect other portions of the policy.) Therefore, $\{u_i^*(x_i)\}_{i \in \mathcal{I}^{i_0}}$ must be an optimal policy for the truncated problem. □

Applying this optimality principle starting from the terminal states and working "backwards in time", we can develop the DP algorithm for the scenario tree based DP. The theorem and proof parallel those presented in Bertsekas (1987) for the basic (not scenario tree based) problem.

Theorem 2.12 (DP Algorithm on a Scenario Tree Structure) *Let $J^*(x_0)$ be the optimal cost in problem $(\mathcal{DP})$. Then $J^*(x_0) = J_0(x_0)$, where the function J_0 is given by the last step of the following algorithm, which proceeds "backwards" from the terminal states T to the initial state 0:*

$$J_j(x_j) = \inf_{u_j(x_j) \in U_j} F_j(x_j, u_j(x_j)), \quad \text{for each } j \in \mathcal{T}$$

$$J_i(x_i) = \inf_{u_i(x_i) \in U_i} \left\{ F_i(x_i, u_i(x_i)) + \sum_{j \in \{i^+\}} p_j J_j(A_i x_i + B_i u_i(x_i) + b_j) \right\} \qquad (\mathcal{CTG})$$

for $i \notin \mathcal{T}$ where $\{i^+\}$ is the set of successor states to state i and p_j is the transition probability.

Furthermore, if $u_i^* = u_i^*(x_i)$ minimizes the right side of $(\mathcal{CTG})$ for each x_i and i, the policy $\{u_i^*(x_i)\}_{i\in\mathcal{I}}$ is optimal for $(\mathcal{DP})$.

Proof. By the principle of optimality, we can expand $J^*(x_0)$ to the following nested form:

$$J^*(x_0) = \inf_{u_0(x_0)\in U_0}\Bigg[F_0(x_0,u_0(x_0)) + \sum_{i\in\{0^+\}} p_i \inf_{u_i(x_i)\in U_i}\Bigg[F_i(x_i,u_i(x_i)) + \sum_{j\in\{i^+\}} p_j \inf_{u_j(x_j)\in U_j}\Big[F_j(x_j,u_j(x_j)) + \cdots$$

$$+ \sum_{m\in\{l^+\}} p_m \inf_{u_m(x_m)\in U_m}\big[F_m(x_m,u_m(x_m)) + \sum_{n\in\{m^+\}=\mathcal{T}} p_n \inf_{u_n(x_n)\in U_n} F_n(x_n,u_n(x_n))\big]\cdots\Big]\Bigg]\Bigg]$$

subject to $x_h = A_k x_k + B_k u_k(x_k) + b_{k+}$ for $h \in \{k^+\}$ and all $k \in \mathcal{I}$.

Starting from the inner minimization, introduce the the cost-to-go functions $J_i(x_i)$ of $(\mathcal{CTG})$ while substituting the dynamics for x_{i+} in the succeeding cost-to-go functions. From this we can obtain

$$J^*(x_0) = J_0(x_0)$$

as desired. Also, since $u_i^*(x_i)$ minimizes the right side of $(\mathcal{CTG})$, the policy $\{u_i^*(x_i)\}_{i\in\mathcal{I}}$ attains the optimal cost. □

Implementation

Ideally, for a given problem, we would like to determine closed-form expressions for J_i. This is possible for the case of an unconstrained DP with linear dynamics and quadratic cost (see Section 2.3.3). But, it is not possible in general, in which case we must resort to numerical solutions on a discretized state space. The minimization in $(\mathcal{CTG})$ is then performed, determining $u_i(x_i)$ for each member in the

discretization of state x_i for each i. On this subject, Bertsekas (1987) makes the following understatement: "For complex multistage problems the computational burden may be prohibitive." This is the so-called "curse of dimensionality." In contrast, u_i in SP is determined only once for each $i \in \mathcal{I}$.

In DP, the solution techniques include the DP algorithm and approximations to the DP algorithm. Convexity, if present, is lost in the course of the algorithm. Solution techniques in SP include any appropriate large-scale mathematical programming techniques. Convexity, if present, can be exploited.

State constraints, while always an added difficulty, are generally unmanageable in the DP algorithm.

The following proposition (Rockafellar, 1993a) further exposes the DP, SP relationship:

Proposition 2.13 *Imagine that the problem of minimization of the total expected cost has been formulated both as a stochastic programming problem ($\mathcal{SP}$) and as a dynamic programming problem ($\mathcal{DP}$) (on a scenario tree so that the structures are consistent). Then*

(a) The optimal values coincide

(b) Any policy which solves ($\mathcal{SP}$) will also solve ($\mathcal{DP}$).

Proof. Notice that the minimization in ($\mathcal{DP}$) is over a larger space than, but includes that, in ($\mathcal{SP}$). Thus, the optimal value in ($\mathcal{SP}$) can not be less than the optimal value in($\mathcal{DP}$).

Assume that $\{u_i^*(x_i)\}_{i\in\mathcal{I}}$ is an optimal policy for ($\mathcal{DP}$). The following construction will be seen to find an optimal solution for the SP:

From the specified initial condition x_0, calculate $\overline{u}_0 = u_0^*(x_0)$. Next, update x_i^* for $i \in \{0^+\}$ in the dynamics. Then calculate the $\overline{u}_i = u_i^*(x_i^*)$ for $i \in \{0^+\}$, update x_j^* for $j \in \{i^+\}$ and calculate the $\overline{u}_j = u_j^*(x_j^*)$, etc. Continue like this until $\overline{u}_i = u_i^*(x_i^*)$ has been specified for all $i \in \mathcal{I}$.

By construction, the policy $\{\overline{u}_i\}_{i\in\mathcal{I}}$ is optimal for $(\mathcal{DP})$. It is also feasible for $(\mathcal{SP})$ and thus must be optimal since the $(\mathcal{SP})$ solution can not improve on the $(\mathcal{DP})$ solution. Thus, the optimal values must coincide and, by the first line in the proof, any policy which solves $(\mathcal{SP})$ will also solve $(\mathcal{DP})$. □

2.3.3 The Linear-Quadratic DP

The special case of unconstrained dynamic programming with quadratic cost and linear dynamics is well known (cf. Bertsekas, 1987) to be one of the few general cases where a closed form ("linear feedback loop") solution can readily be determined. This section extends that result to the DP on a scenario tree structure.

The unconstrained linear-quadratic DP can be formulated as a problem of choosing a policy $\{u_i(x_i)\}_{i\in\mathcal{I}/\mathcal{T}}$ so as to minimize

$$\sum_{i\notin\mathcal{T}} \pi_i \left\{ \frac{1}{2} x_i \cdot Q_i x_i + q_i \cdot x_i + \frac{1}{2} u_i \cdot R_i u_i + r_i \cdot u_i \right\} + \sum_{i\in\mathcal{T}} \pi_i \left\{ \frac{1}{2} x_i \cdot Q_i x_i + q_i \cdot x_i \right\} \quad (\mathcal{DP}_{lq})$$

$$\text{subject to: } x_{i+} = A_i x_i + B_i u_i + b_{i+}, \quad x_0 = b_0$$

where $x_i \in \Re^m$, $u_i(x_i) \in \Re^n$ and where the given matrices A_i, B_i, Q_i, R_i have appropriate dimension and the Q_i are symmetric positive semidefinite and the R_i are symmetric positive definite. Notice that in problem $(\mathcal{DP}_{lq})$ we have assumed that no decision is made at the terminal states $i \in \mathcal{T}$. This is typical of DP formulations, but not necessary as we have seen in problem $(\mathcal{DP})$ and in the DP algorithm.

Theorem 2.14 (Feedback Loop Solution for $(\mathcal{DP}_{lq})$) *A linear feedback loop solution (see Figure 2.2) to problem $(\mathcal{DP}_{lq})$ can be obtained for each $i \in \mathcal{I}/\mathcal{T}$ as*

$$u_i^*(x_i) = -(R_i + B_i^\top K_i^+ B_i)^{-1}(r_i + B_i^\top K_i^+ A_i x_i + B_i^\top l_i^+)$$

where x_i is given by the dynamics and where the symmetric positive semidefinite matrices K_i^+, K_i and the vectors l_i^+, l_i are defined, for $i \in \mathcal{T}$, as $K_i = Q_i$ and $l_i = q_i$

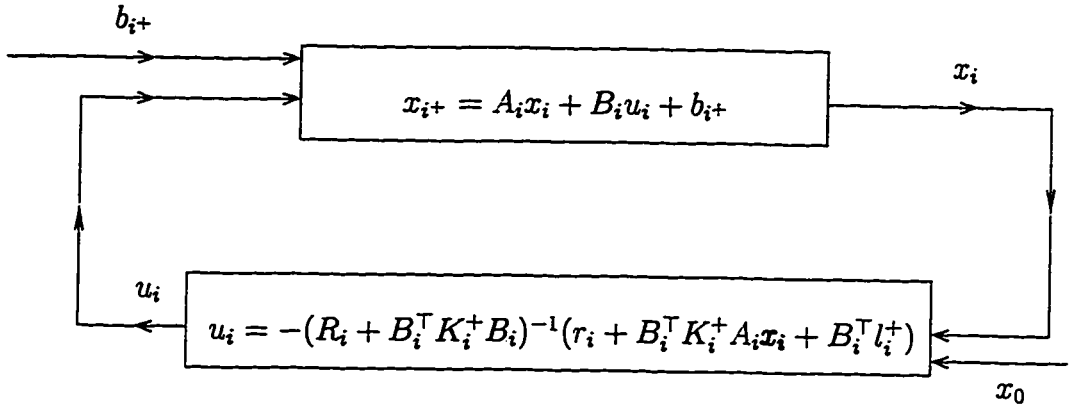

Figure 2.2: Linear feedback loop solution for the unconstrained linear-quadratic DP on the scenario tree.

and recursively defined, for $i \notin \mathcal{T}$, as

$$K_i^+ = \sum_{j \in \{i^+\}} p_j K_j, \qquad l_i^+ = \sum_{j \in \{i^+\}} p_j \left[l_j + K_j b_j \right],$$

$$K_i = Q_i + A_i^\top \left[K_i^+ - K_i^+ B_i (R_i + B_i^\top K_i^+ B_i)^{-1} B_i^\top K_i^+ \right] A_i,$$

$$l_i = q_i + A_i^\top l_i^+ - A_i^\top K_i^+ B_i (R_i + B_i^\top K_i^+ B_i)^{-1} (r_i + B_i^\top l_i^+).$$

Proof. Since Q_i and R_i are symmetric positive (semi)definite, it is easy to see that the K_i^+ and K_i are also symmetric positive semidefinite and $R_i + B_i^\top K_i^+ B_i$ is symmetric positive definite.

Apply the algorithm defined in Theorem 2.5 to problem $(\mathcal{DP}_{lq})$: For each $j \in \mathcal{T}$, write

$$J_j(x_j) = \frac{1}{2} x_j \cdot Q_j x_j + q_j \cdot x_j$$

and notice that it is quadratic in x_j.

Assume that for any $j \in \mathcal{I}$, J_j is a quadratic in x_j of the form

$$J_j(x_j) = \frac{1}{2} x_j \cdot K_j x_j + l_j \cdot x_j + c_j$$

(It is clearly true for all $j \in \mathcal{T}$). Show that if state $i \notin \mathcal{T}$ precedes state j in the scenario tree, then $J_i(x_i)$ has the same quadratic form with K_i and l_i defined as above.

Applying $(\mathcal{CTG})$ gives

$$J_i(x_i) =$$

$$\min_{u_i \in \Re^{m_i}} \left\{ \frac{1}{2} x_i \cdot Q_i x_i + q_i \cdot x_i + \frac{1}{2} u_i \cdot R_i u_i + r_i \cdot u_i + \sum_{j \in \{i^+\}} p_j \left\{ J_j(A_i x_i + B_i u_i + b_j) \right\} \right\}$$

where "min" has replaced "inf" since the minimum will be attained in this, the unconstrained quadratic, case. Expanding the sum (with K_i^+ and l_i^+ as above) gives

$$= \min_{u_i \in \Re^{m_i}} \left\{ \frac{1}{2} x_i \cdot (Q_i + A_i^\top K_i^+ A_i) x_i + \frac{1}{2} u_i \cdot (R_i + B_i^\top K_i^+ B_i) u_i + (q_i + A_i^\top l_i^+) \cdot x_i \right.$$
$$\left. + (r_i + B_i^\top K_i^+ A_i x_i + B_i^\top l_i^+) \cdot u_i + [\text{constant terms}]_i \right\}.$$

Notice that the unconstrained minimand is quadratic in u_i. Then set the gradient to zero yielding the minimizing

$$u_i^*(x_i) = -(R_i + B_i^\top K_i^+ B_i)^{-1}(r_i + B_i^\top K_i^+ A_i \boldsymbol{x_i} + B_i^\top l_i^+)$$

as stated above.

Substituting u_i^* for u_i in $J_i(x_i)$ reveals the quadratic form

$$J_i(x_i) = \frac{1}{2} x_i \cdot K_i x_i + l_i \cdot x_i + c_i$$

with K_i and l_i as above.

Thus, by induction, we can see that the linear feedback loop finds solution to the unconstrained linear-quadratic DP problem $(\mathcal{DP}_{lq})$. □

To employ the algorithm, first calculate the K_i, K_i^+, l_i, l_i^+ starting with $i \in \mathcal{T}$ and working "backwards" along the scenario tree. Then calculate the u_i^* "forwards" (see Figure 2.2) while updating x_i in the dynamics.

Remark 2.15 *To extend this feedback loop solution result to the case where a terminal decision is required, define a set of states $\mathcal{T}'$ following the terminal states $\mathcal{T}$. At these new states $i \in \mathcal{T}$, no decision is made. The feedback loop solution (as derived) can then be applied to the unconstrained linear-quadratic DP on the extended scenario tree $\mathcal{I} \cup \mathcal{T}'$.*

2.4 Related Techniques

Two other stochastic optimization techniques are briefly described in this section. First is the extension of the Benders decomposition algorithm (Benders, 1962) to the multistage and stochastic cases (Birge, 1980, 1985; Wets, 1988). The presentation here will follow a different approach, originally called *stochastic dual dynamic programming*, by Pereira and Pinto (1985, 1991). The Benders decomposition principle can, in theory, be applied to nonlinear problems, but has been found to be practical only for the linear case in large problems. Benders decomposition schemes are the current fashion in the hydropower scheduling setting.

The other technique to be discussed here is the "progressive hedging" algorithm of Rockafellar and Wets (1991).

2.4.1 Benders Decomposition

This section will illustrate the nature of the Benders decomposition schemes for a version of the linear problem $(\mathcal{P}_{lp})$ introduced in Example 1 of Section 2.2.2

$$\begin{aligned} &\underset{u \in \Re_+^{n_i}}{\text{minimize}} \sum_{i \in \mathcal{I}} \pi_i \{r_i \cdot u_i\} && (\mathcal{P}_{lp}) \\ &\text{subject to : } C_i x_i + D_i u_i \geq d_i \\ &\text{where : } x_{i+} = A_i x_i + B_i u_i + b_{i+}, \quad x_0 = b_0. \end{aligned}$$

As is done in Pereira and Pinto (1985, 1991), the Benders scheme will be derived from the dynamic programming algorithm. Proposition 2.13 shows that the DP algorithm can be used to find solution of an SP problem. In DP, the state variables are typically discretized a-priori. This can lead to the "curse of dimensionality." In Benders decomposition, a discretization of the state is constructed as "trial states" as the algorithm progresses.

Applying the cost-to-go construct of the DP algorithm to $(\mathcal{P}_{lp})$, the following

system arises:

$$J_j(x_j) = \inf_{u_j \in \Re_+^{n_j}} \{r_j \cdot u_j\}, \quad \text{for each } j \in \mathcal{T}$$

$$\text{subject to: } C_j x_j + D_j u_j \geq d_j$$

and

$$J_i(x_i) = \inf_{u_i \in \Re_+^{n_i}} \left\{ r_i \cdot u_i + \sum_{j \in \{i^+\}} p_j J_j(A_i x_i + B_i u_i + b_j) \right\} \qquad (\mathcal{CTG})$$

$$\text{subject to: } C_i x_i + D_i u_i \geq d_i$$

for $i \notin \mathcal{T}$ where clearly u_i will depend on the *parameter* x_i and where $\{i^+\}$ is the set of successor states to state i and p_j is the transition probability.

One way to avoid discretization of the state space is to interpolate the $J_i(x_i)$ as a polynomial or with splines (*e.g.* Gal, 1989; Foufoula-Georgiou and Kitandis, 1988). We will see that, in principle, an *exact* piecewise linear representation of $J_i(x_i)$ can be constructed. In the Benders scheme, a relaxed version of this piecewise linear representation will be used to form an approximate cost-to-go function (denoted $\hat{J}_i(x_i)$).

Construction of $\hat{J}_i(x_i)$ for $j \in \mathcal{T}$

For $j \in \mathcal{T}$,

$$J_j(x_j) = \inf_{u_j \in \Re_+^{n_j}} \{r_j \cdot u_j\}$$

$$\text{subject to: } C_j x_j + D_j u_j \geq d_j.$$

Introducing the multiplier $v_j \in \Re_+^{m_i}$ for the constraint, the associated dual problem formulation for J_j is

$$J_j(x_j) = \sup_{v_j \in \Re_+^{m_j}} \{v_j \cdot (d_j - C_j x_j)\}$$

$$\text{subject to: } r_j - D_j^T v_j \leq 0$$

where x_j is thought of as a parameter in the "sup." This dual formulation for $J_j(x_j)$ is a linear programming problem on a polyhedral constraint (which does not depend on the parameter x_j). Thus, the LP must have an optimal solution at a vertex of the feasible region.

Let

$$\mathbf{V}_j = \{v_j^1, v_j^2, \ldots, v_j^{s_j}\}$$

be the set of vertices of the feasible region. Then, in principle, $J_j(x_j)$ can be determined for a specified x_j by enumeration:

$$J_j(x_j) = \max_{h=1,\ldots,s_j} \{v_j^h \cdot (d_j - C_j x_j)\}$$

or equivalently as a minimization over the epigraph of supporting hyperplanes

$$\begin{aligned} J_j(x_j) = & \min \alpha_j \\ & \alpha_j \geq v_j^1 \cdot (d_j - C_j x_j) \\ & \vdots \\ & \alpha_j \geq v_j^{s_j} \cdot (d_j - C_j x_j). \end{aligned}$$

(Thus $J_j(x_j)$ is, in fact, piecewise linear). This structure is the key to the Benders algorithm.

The set of vertices is potentially very large, but only a few of the associated constraints will be active at the optimal solution. Thus, a relaxed set of constraints should suffice. In the Benders algorithm, the constraining hyperplanes are added, as needed, at each iteration. These are the *Benders cuts.*

At each iteration, the approximate cost-to-go function $\hat{J}_j(x_j)$ will be defined as the minimum over the epigraph of a relaxed set of the constraining hyperplanes. First, we will see how the set of current (at an iteration) vertices is determined.

In the algorithm, a set of "trial states" $\hat{x}_j^h, \quad h = 1, \ldots, k$ is stored for each $j \in \mathcal{I}$. (We will see how these arise below.) Essentially, this is state-discretization "on-the-fly" rather than the a-priori choice of states employed in DP.

For each $\hat{x}_j^h$, $h = 1, \ldots, k$ in the set of "trial states," the LP

$$J_j(\hat{x}_j) = \sup_{v_j \in \Re_+^{m_j}} \{v_j \cdot (d_j - C_j \hat{x}_j)\}$$

$$\text{subject to: } r_j - D_j^\top v_j \leq 0.$$

is solved and the resulting solution v_j^h must be, by LP theory, in the set $\mathbf{V}_j$ of vertices of the feasible region.

The current approximate cost-to-go function can then be constructed as

$$\hat{J}_j(x_j) = \min \alpha_j$$

$$\alpha_j \geq v_j^h \cdot (d_j - C_j \hat{x}_j), \quad h = 1, \ldots, k$$

where v_j^h, $h = 1, \ldots, k$ are the multipliers (vertices) which correspond to the current set of "trial states" $\hat{x}_j^h$, $h = 1, \ldots, k$.

Construction of $\hat{J}_i(x_i)$ for $i \notin \mathcal{T}$

For $i \notin \mathcal{T}$, the process is similar.

$$J_i(x_i) = \inf_{u_i \in \Re_+^{n_i}} \left\{ r_i \cdot u_i + \sum_{j \in \{i^+\}} p_j \hat{J}_j (A_i x_i + B_i u_i + b_j) \right\}$$

$$\text{subject to: } C_i x_i + D_i u_i \geq d_i$$

where the J_j on the right-hand-side has been replaced by the current approximate cost-to-go function $\hat{J}_j$.

Equivalently,

$$J_i(x_i) = \inf_{u_i \in \Re_+^{n_i}} \left\{ r_i \cdot u_i + \sum_{j \in \{i^+\}} p_j \alpha_j \right\}$$

$$\text{subject to: } C_i x_i + D_i u_i \geq d_i.$$

$$\alpha_j \geq v_j^h \cdot (d_j - C_j (A_i x_i + B_i u_i + b_j)), \quad j \in \{i^+\},\ h = 1, \ldots, k.$$

As in the terminal stage, multipliers are introduced for the constraints and the dual problem is formulated. The resulting LP is solved for each $\hat{x}_i^h$, $h = 1, \ldots, k$ (in the current set of "trial states"). The resulting multipliers v_i^h, $h = 1, \ldots, k$ form a basis for the approximate cost-to-go function $\hat{J}_i(x_i)$

Before the next iteration, new "trial states" are added to the collection (one for each $i \in \mathcal{I}$) by a forward recursion (or forward Monte Carlo simulation (Pereira and Pinto, 1991; Gorenstin et al., 1992)).

Upper and lower bounds for the optimal value are calculated easily at each iteration and that gap is used as a stopping criterion.

2.4.2 Progressive Hedging

In Section 2.2.1 we saw how the scenario tree (see Figure 2.1) can be used to structure the information and decisions for the SP model. Another way to structure the information/decisions is to represent them on an expanded tree characterized by no further branching after the initial node. Figure 2.3 contrasts the scenario tree structure with this expanded tree structure. Notice that in the expanded tree, a bundle of nodes (encircled by dashed lines) replaces a single node in the scenario tree. Yet each state in the bundle corresponds to identical history and information. Thus, to counter anticipation of later scenario specific developments in information, a *nonanticipativity* constraint is imposed to ensure that equivalent decisions are taken at each state in the bundle. This is the underlying structure of the *progressive hedging* algorithm (Rockafellar and Wets, 1991).

An advantage of progressive hedging is that the stochastic problem can be disaggregated into deterministic problems on each scenario at each iteration. The weighted sum (the expectation) of events on the various scenarios is then minimized subject to the nonanticipativity constraints.

The major drawback of this method is illustrated by the case of a binary tree spanning 9 decision stages and necessitating a computationally reasonable 511 infor-

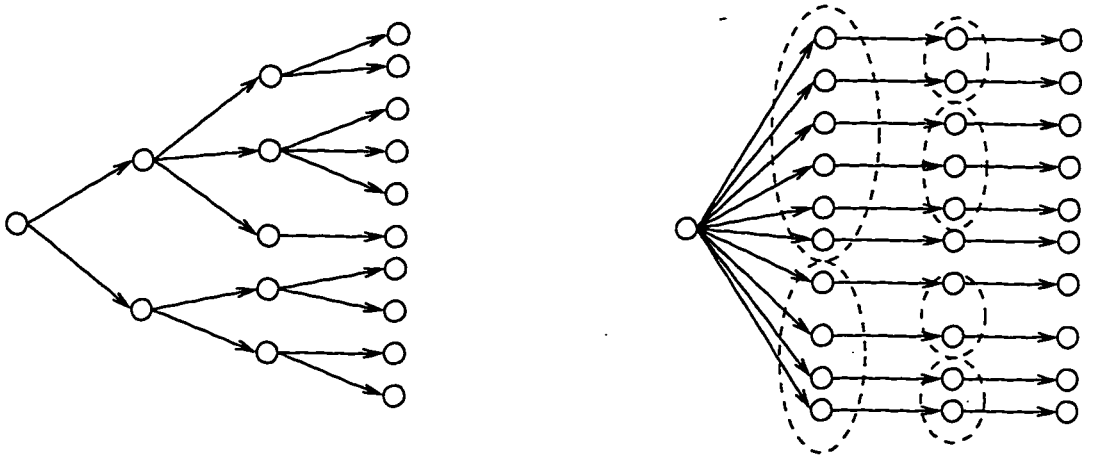

Figure 2.3: A scenario tree vs. the corresponding expanded tree used in progressive hedging.

mation/decision nodes. The corresponding expanded tree would contain 2049 nodes. In this case, progressive hedging would increase the number of decisions by four times while adding countless constraints to what tends already to be a dimensionally challenging problem. The explosion can be worse yet if the tree has more branches or stages.

Chapter 3

HYDROPOWER SCHEDULING MODELING ISSUES

A regional electric power generation system can consist of hydroelectric as well as nuclear, thermal (gas or coal), wind and solar generation systems along with the ability to buy and sell power *off the grid* from and to neighboring producers. Since power can not be effectively stored for later use (except using pump stations — see Section 3.2.1), the problem facing the system manager is: how much power to produce and when to produce it so as to minimize costs (or maximize profit) while meeting the demand for power. When trying to minimize costs, it is natural to look to the optimal operation of the hydropower system. Of the major producers (i.e. excluding wind and solar), hydro is the only one with essentially zero marginal cost of produced power. Since hydropower is fuel (water) limited, its opportunity cost is given by the costs of the resources it displaces.

There are different ways to look at optimal operation of a hydro system. In the Pacific Northwest for example, hydroelectric power generation is scheduled with an eye toward selling power to California as well as meeting regional demands. In California, hydro generation is scheduled so as to minimize costs of obtaining deficit power from other sources (thermal plants, off the grid, etc.). In both cases, the power costs can be thought of as a function of the difference between generated hydropower and total demand; yielding surplus or shortage depending on perspective. Other costs can be incorporated to represent the interests of fish and other water users. These other interests are given varying degrees of priority in different regions.

The models and techniques to be examined here will apply to management of existing resource systems and not (necessarily) to optimal system design or capacity

expansion.

The two power generation scheduling models to be presented share similarities with many of the models in the literature. The first model is more general and could be easily adapted for other resource allocation applications. The algorithm to be presented later would then apply to any situation which can be modeled in that format. The second model is adapted from the one used by the Pacific Gas and Electric company (PG&E) in Northern and central California (Jacobs et al., 1995; personal communication with Gary L. Schultz (PG&E), 1995). This model contains more specific information on the form of the cost function including additional costs in the form of incentives or disincentives to spill water. It also has decision subperiods for dealing with peak and off-peak hours of power demand. These and other differences aside, the "adapted PG&E" model (as well as data) are later written into the essential form of the general model for the purpose of testing the algorithm.

The general hydroelectric power generation scheduling model is introduced in Section 3.1 with review of some of the related modelling literature. The dynamics will be seen to differ form the dynamics introduced in Chapter 2. This difference will be reconciled in Section 3.1.5. Specific improvements of the PG&E model are discussed in Section 3.2. Section 3.3 surveys the evolution of stochastic optimization techniques found in the hydro engineering literature.

3.1 Hydropower Scheduling Models

In both the general model and the adapted PG&E model, the physical aspects of the hydropower system are modeled on a network (the physical network). Reservoirs and junction points are represented as nodes (physical nodes). Controllable releases of water through powerhouses and spillways are represented on arcs. These releases and storages are constrained due to capacity and nonnegativity as well as by governmental decree and contractual obligation. This network structure underlies the physical

portion of many of the models in the literature (though it is rarely described in terms of "nodes" and "arcs").

In both models, and in a very few of the more recent articles (cf. Pereira and Pinto, 1985, 1991; Jacobs et al., 1995), a scenario tree (see Section 2.1) provides the underlying information/decision structure. The probability distribution for the natural (uncontrolled) inflow of water is arrayed on the tree. The control decisions (water releases) are returned for each state $i \in \mathcal{I}$ on the tree. Many of the models in the literature were formulated with an eye toward stochastic dynamic programming. As such, the information structure is typically a discretization of the state space. The decisions are returned as functions of the state variable (discretization).

In both models, the cost functions associated with a release decision are convex functions of that water release. In the adapted PG&E model, this arises as the integral of monotone increasing marginal cost data. In much of the literature, the cost (or power) objective function is assumed to be linear for the sake of ease of solution. Implications of this assumption are discussed subsequently.

Conservation of (water) mass on the system is enforced by the system dynamics (see subsequent discussions). The models are considered to be long time-scale models and are formulated with an eye toward monthly decisions on a one or two-year time horizon.

Combining these aspects of the model with the minimization of the expected total cost, we will arrive at a multistage stochastic programming model (see Section 2.2). In what follows, various aspects of the SP models will be given physical meaning in the hydroelectric power scheduling setting. Some alternative formulations will be discussed.

3.1.1 The Physical Network

In the hydropower generation network, water storage $x_i \in \Re^m$ is taken as the state of the system. For each state $i \in \mathcal{I}$ of the scenario tree, the vector x_i represents the

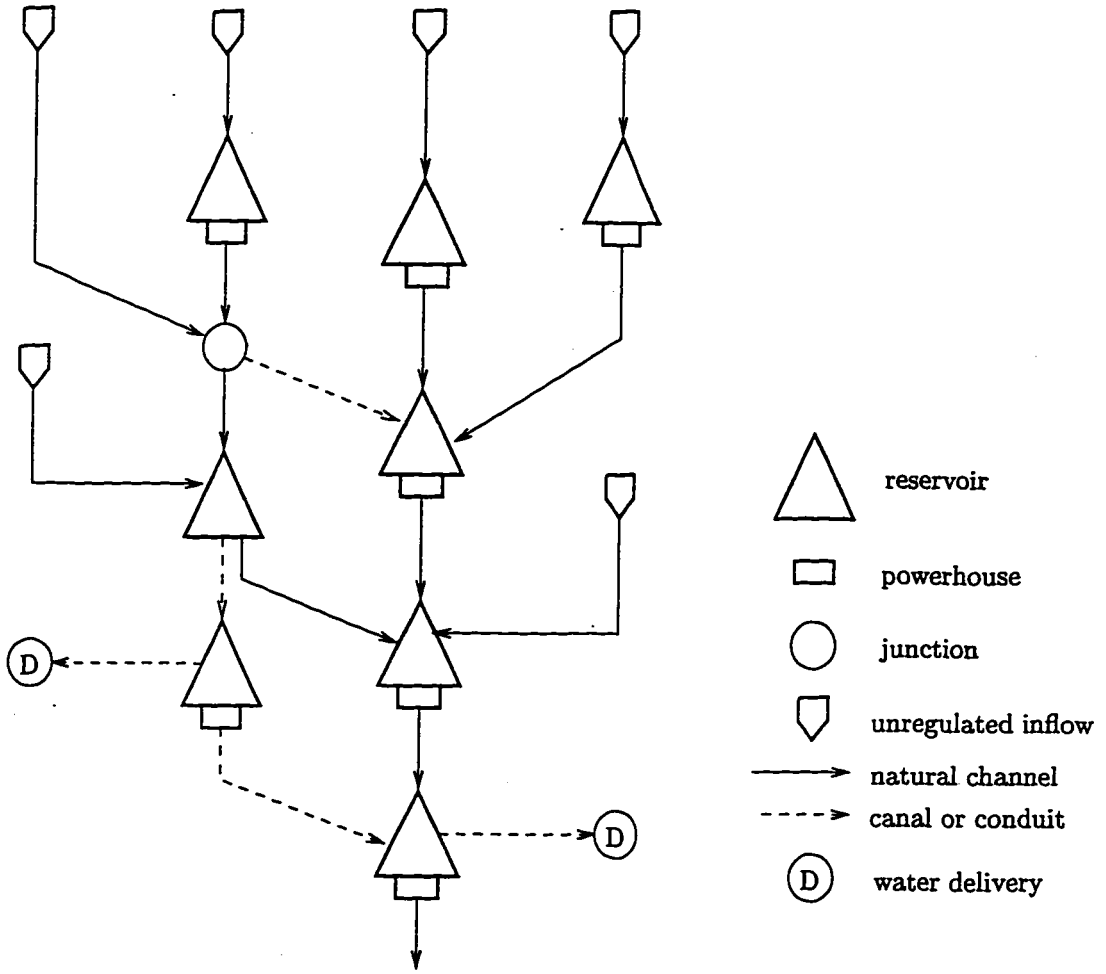

Figure 3.1: Sample schematic of a watershed power generation network.

storage at each of the reservoirs and junctions (physical nodes) $k = 1, \ldots, m$ in the physical network. Controllable releases of water $u_i \in \Re^n$ through powerhouses (turbines) and spillways are represented on arcs $j = 1, \ldots, n$ in the network. Figure 3.1 shows a sample schematic of a watershed power generation network.

The release decisions $u_i \in \Re^n$ are made for each information/decision state $i \in \mathcal{I}$ so as to minimize the expected cost (described next section) of meeting the power demand on the system over the specified time horizon.

The natural inflow $b_i \in \Re^m$ represents the uncontrolled inflow at the physical

nodes $k = 1, \ldots, m$ for each $i \in \mathcal{I}$.

Constraints

Upper and lower bounds on releases $\underline{u}_i \leq u_i \leq \overline{u}_i$ are set in accordance with capacities of powerhouses and spillways, non-negativity of release and congressionally mandated minimum flows on spillways and rivers for fish and other wildlife, irrigation and recreation. Some of the upper bounds are infinite, many of the lower bounds are zero. These bounds define the sets (boxes) $U_i = [\underline{u}_i, \overline{u}_i] \subset \Re^n$ for each $i \in \mathcal{I}$.

Reservoir capacities, flood control and wildlife and recreation interests necessitate a (state) storage constraint $\underline{s}_i \leq x_i \leq \overline{s}_i \in \Re^m$.

3.1.2 The Scenario Tree

The decision to release or (store water) is based in part on the amount of water which will become available in the future. Unfortunately, future inflows of water can not be known with certainty. What is known is many years of recorded inflow data. A scenario tree is used here as an discrete approximation of the continuous probability distribution of these dependent random inflow events. For each state $i \in \mathcal{I}$, the inflow at the physical nodes is represented by $b_i \in \Re^m$ and occurs with probability $p_i > 0$.

In the hydro scheduling model, information and decisions correspond to a particular *time stage* (*e.g.* month). This further structures the scenario tree (see Figure 3.2). For a time stage of one month duration, if state i corresponds to information/decisions for April, then states $j \in \{i^+\}$ correspond to information/decisions for May, and states $k \in \{j^+\}$ for June, *etc.* Time stage duration need not be uniform, but the k^{th} node of every scenario must correspond to the same k^{th} time stage and every scenario has one node per time stage.

The tree is further refined for the particular year using, for example, seasonal streamflow forecast models based on snowpack analysis (cf. Stedinger et al., 1987).

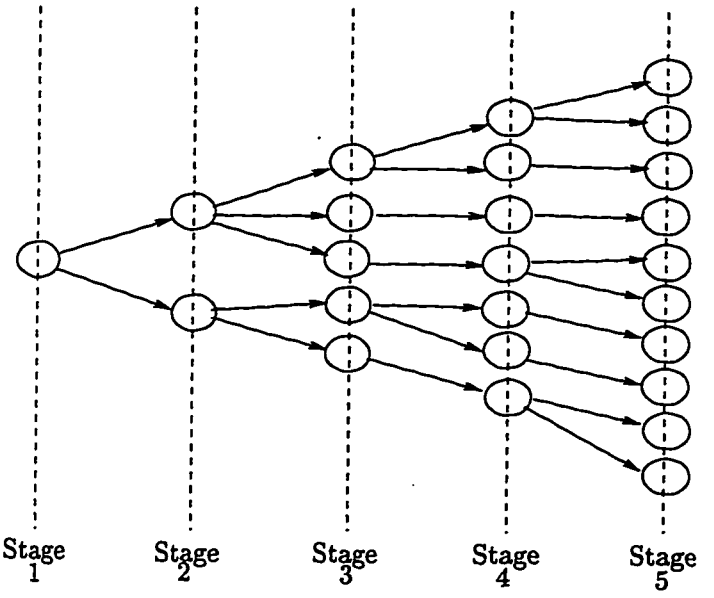

Figure 3.2: Stage-structured scenario tree.

The inflow events are dependent because inflows are correlated over consecutive months. The strength of this month-to-month correlation can help determine the number of branches of the tree. On the PG&E system, for example, the July, August, September inflows are so strongly correlated that a single branch might be used to represent each inflow transition (Jacobs et al., 1995). In other months, three branches might be used, for example, to represent the the probabilities of high, medium and low inflow given the current state $i \in \mathcal{I}$. Pereira and Pinto (1985, 1991) use a binary tree with a 9 month time horizon.

3.1.3 The Cost Function

One advantage of a scenario tree representation of the probability distribution is the ease in representing an expected value on that distribution. Here, we wish to minimize the total expected cost in meeting the power demand. If the cost associated with the decision u_i is represented as $g_i(u_i)$, then the expected value of the total cost can be

written

$$\sum_{i \in \mathcal{I}} \pi_i g_i(u_i)$$

where $\pi_i > 0$ is the scenario probability.

As mentioned in the introduction to this chapter, the cost function actually arises as a function of the "shortage" (= demand − hydro) of power to be made up by means other than hydro in that period. The generated hydropower in the period is in turn a function of release. In most of the literature, an affine cost as a function of release is assumed. This implies that the cost is an affine function of the shortage and that the powerhouse efficiency (ratio of power generation to water release) is constant. Of course these are simplifications to ease the solution process.

The power costs as a function of shortage are however known to be nonlinear. This source of non-linearity has often been neglected because of the difficulty in solving the nonlinear problem. Here, in accordance with data supplied by PG&E, we will assume a nonlinear (convex) cost objective arising from convex costs $g_i(u_i)$.

Powerhouse Efficiency

The powerhouse efficiency can vary with flow (release) and with head (the total height of water above the powerhouse; see Figure 3.3). The dependence on flow occurs mainly at very high or low flows and is "generally insignificant at the time scale of interest" (Jacobs et al., 1995). The maximum flow rate can also depend on head (Ikura and Gross, 1984). If the fluctuation in reservoir level is significant as compared to the head, then the resulting fluctuation in powerhouse efficiency should not be ignored in the model.

Soliman and Christensen (1986) address the variable head problem by assuming an affine relationship between storage and head. Then head, rather than storage, is taken as the state variable in the system. The resulting objective function is

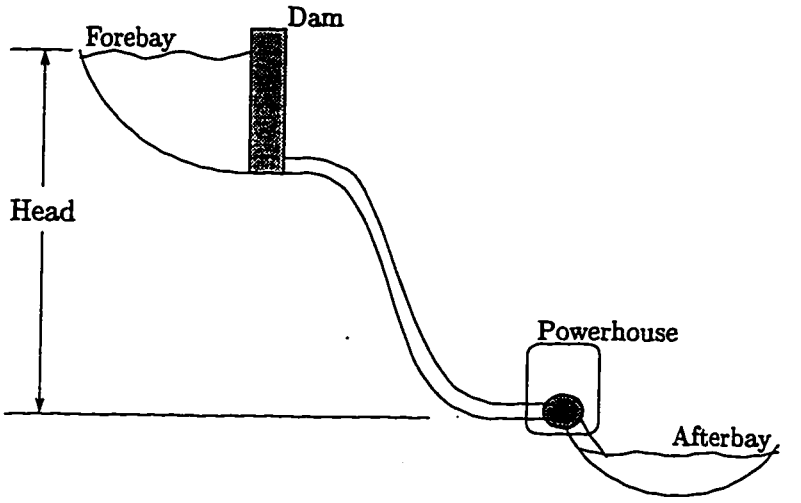

Figure 3.3: Power plant schematic showing "head."

non-separable in state and control (which can greatly increase the difficulty of the problem).

"Fortunately, most of PG&E's reservoirs are high-head" (Jacobs et.al, 1995) and so suffer only minor head variation. Thus the head effects are ignored (*i.e.* powerhouse efficiency is assumed to be constant) in their model as well as nearly all of the models in the literature including the one presented here.

3.1.4 The Dynamics

To keep track of water in the system, a dynamical constraint (the dynamics) is formulated for each river basin in the power generation system. The dynamics adds inflows to and subtracts releases from the storage levels.

In real (continuous) time, water is flowing into and being released from the reservoirs continuously. There is a certain time lag before released water becomes available for release at the next reservoir downstream. Release decisions are continuously being made and storage and release constraints are continuously being observed.

To have any hope of solving the overlying optimization problem, the dynamics must be discretized in time (or, in our model, discretized to fit the informa-

tion/decision structure of the scenario tree). To do this, we must choose an order in which to represent the actions. Is the inflow observed before (as a forecast) the release decision is made; or after? At what point in the process is the storage constrained? Is the time lag between when water is released and when it is available at a downstream reservoir best represented as one time step or zero? If one is not careful in making these choices, unintended consequences can result.

For the hydropower system, the dynamics for each distinct river basin is formulated

$$\underbrace{x_i}_{\text{storage}} = \underbrace{x_{i^-}}_{\text{prev. storage}} + \underbrace{Bu_i}_{\text{inflow-release}} + \underbrace{b_i}_{\text{natural inflow}}, \quad \underbrace{x_0 = b_0}_{\text{initial storage}} \qquad (\text{dyn}_{\text{hp}})$$

where $B \in \Re^{m \times n}$ is the node-arc incidence matrix for the particular river basin. One should immediately notice that *unlike* the dynamics in the models in Chapter 2, the state index of the control (release) u_i matches that of the random variable (natural inflow) b_i. This important difference will be reconciled in Section 3.1.5 so that the developments of Chapters 2 and 4 can be applied to our hydro scheduling problem.

Observed vs. Forecasted Inflows

In this formulation of the dynamics (dyn_{hp}), it is assumed that the inflow b_i for state i is known (forecasted) before the release decision u_i at that state is made. Accurate inflow predictions based on snowpack analysis, streamflow forecasting and climate modeling, justify this assumption. For the purposes of the model, the forecasts are assumed to be realized.

The early stochastic DP models used *observed inflow* as the random variable. More recently, Alercon and Marks (1979), Bras et al. (1983) and Stedinger et al. (1984) started basing release decisions on *forecasted inflow.* Stedinger et al. (1984) found "considerable improvement when using forecasted inflows instead of observed inflow from the previous period as the system state" in a study on the High Aswan Dam in the Nile river basin. However, Hueng et al. (1991) found the opposite result on their

study of the Feitsui river in Taiwan. They admit however that their results may have been "different under different hydrological regimes with better forecasting possibilities." More recently, the use of forecasted inflows has become standard (cf. Pereira and Pinto, 1985 and 1991; Jacobs et al., 1995; Archibald et al., 1996).

In Northern California (PG&E), the monthly streamflows are strongly correlated with snowpack in the Sierra Nevada and so the forecasts are relatively accurate. Thus the present model utilizes inflow forecasts.

Observed Inflows; a Difficulty with State Constraints

If instead we assume that a release decision u_i must be made *before* the inflow b_i is known (observed), the dynamics (for a scenario-tree-based model) would take the form

$$x_{i+} = x_i + Bu_i + b_{i+} \qquad (\text{dyn}_{\text{obs}})$$

like the dynamics of the stochastic programming models in Chapter 2 (with $A_i = I$ and $B_i = B$). Note that for a model not associated with a scenario tree (like most of the DP models in the literature), the dynamics can be written $x_{i+} = x_i + Bu_i + b_i$ where it must be specified whether release is chosen before inflow is *observed* or after inflow is *forecasted.*

Also in Chapter 2, we had a linear constraint of the form $d_i - C_i x_i - D_i u_i \leq 0$. If $D_i \equiv 0$ for all $i \in \mathcal{I}$, then this constraint, like the storage constraint $\underline{s}_i \leq x_i \leq \overline{s}_i$ in the hydro problem, is purely a state constraint. State constraints can be tricky. For example, the combination of the storage (state) constraint with the dynamics (dyn_{obs}) can have the unintended consequence that the implied control constraint

$$\underline{s}_{i+} \leq x_i + Bu_i + b_{i+} \leq \overline{s}_{i+}$$

actually depends upon the discretization of b_{i+} chosen in the problem. In the hydro system, the inclusion of low-probability extreme (high or low) inflows in the dis-

cretization of the distribution for b_{i+} would have the unintended consequence of a more conservative constraint (and possibly a more conservative release policy).

It appears that this sticky issue has previously gone unnoticed (or has been ignored) in the hydro engineering literature.

Time Lags

Another assumption implicit in this formulation of the dynamics (dyn_{hp}) is that the water released through a powerhouse or spillway is immediately available for use down stream in that same time stage.

A minor modification of the dynamics would allow for a lag in time of one (or more) stages between when water is released and when it becomes available downstream

$$x_i = x_{i-} - \underbrace{B^- u_{i-}}_{\text{release}} + \underbrace{B^+ u_i}_{\text{inflow}} + b_i$$

where $B = B^+ - B^-$ (and all entries in B^+ and B^- are either 0 or 1).

Appropriateness of this time lag depends especially on the duration of a time stage in the model. For month-long time stages, time lags are in general not appropriate. On the Columbia River in Washington and Oregon, it can take as little as 3 weeks (depending on season) for water to run the entire controlled course (11 dams) of the river system (personal communication with Pamela Shaw, Columbia Basin Research, 1997). Imposition of a one-stage lag would result in a 10 stage lag between availability of water form the upper to the lower reaches of the system. This would clearly be unacceptable for a month-long time stage. This lag might be more appropriate for a model with a one-day time stage. Turgeon (1981b) employs a time lag in a study on short-term scheduling.

3.1.5 Reconciliation with Dynamics from Chapter 2

The dynamics from the stochastic programming models in Chapter 2 (as in (dyn_{obs})) can be written (with control w_i in place of u_i) as

$$x_{i+} = x_i + Bw_i + b_{i+}. \qquad (\text{dyn}_{\text{ch2}})$$

For comparison, rewrite the dynamics proposed for the hydropower system (with indices shifted):

$$x_{i+} = x_i + Bu_{i+} + b_{i+}. \qquad (\text{dyn}_{\text{hp}})$$

In the dynamics (dyn_{ch2}), the release decision w_i is made before the random inflow b_{i+} is *observed.* In (dyn_{hp}) the decision u_{i+} is made after the inflow b_{i+} is *forecasted.* To reconcile this important difference and so allow the developments of Chapters 2 and 4 to be applied to our hydropower scheduling model, see that the following is true.

Proposition 3.1 *In the dynamics* (dyn_{ch2})*, take*

$$w_i = \begin{pmatrix} u_{j_1} \\ u_{j_2} \\ \vdots \\ u_{j_{h_i}} \end{pmatrix}, \quad \textit{where } \{j_1, j_2, \ldots, j_{h_i}\} = \{i^+\}$$

(where h_i is the number of branches of the scenario tree following state i) and replace B by a block matrix $B_{i+} = [0 \vdots 0 \vdots \cdots \vdots B \vdots \cdots \vdots 0]$ so that $B_j w_i = Bu_j$ for any particular $j \in \{i^+\}$. Then the dynamics (dyn_{ch2}) *is equivalent in effect to* (dyn_{hp})*.*

As an interpretation, think of the decision w_i as a policy which specifies, for each $i \in \mathcal{I}$, a release for each of the possible values of the (discretization of the) inflow forecast b_{i+}. The policy w_i is determined before a particular value of b_{i+} is revealed. This flexibility will be appreciated in Chapter 5 when the dynamic splitting algorithm (Section 4.6) is applied to the hydro scheduling problem.

3.1.6 The Model

For convenience, all aspects of the hydroelectric power generation scheduling SP model are collected here:

$$\begin{aligned}
&\underset{u_i \in U_i,\, i \in \mathcal{I}}{\text{minimize}} \sum_{i \in \mathcal{I}} \pi_i g_i(u_i) \\
&\text{subject to}: \quad x_i = x_{i^-} + B u_i + b_i, \quad x_0 = b_0 && \forall i \in \mathcal{I} \\
&\qquad \underline{u}_i \le u_i \le \overline{u}_i && \forall i \in \mathcal{I} \\
&\qquad \underline{s}_i \le x_i \le \overline{s}_i && \forall i \in \mathcal{I} \\
&\qquad x_i = b_T && \forall i \in \mathcal{T}
\end{aligned}$$

where the information and decision structure is arrayed on the scenario tree and the variables are defined above.

Terminal Storage Targets

Notice the terminal storage targets $x_i = b_T$, $i \in \mathcal{T}$. Since we assume that the inflow is observed (as a forecast) before a release decision is made, the targets can, in principle, be met. We will however think of them as "soft" constraints.

In a model where release decisions precede inflow observation, these targets could not, in principle, be met for all scenarios; they would need to be replaced by terminal costs (penalties) in the objective function.

3.2 The Adapted PG&E model

The essential form of PG&E's model (Jacobs et al., 1995; personal communication with Gary L. Schultz, 1995) can be written into the form of the general SP model presented above. This section will present some specific information about the form of the cost functions as well as improvements such as weekday and weekend peak

and off-peak demand subperiods, reservoir carryover arcs and baseloaded and load following powerhouses. We will also discuss how these fit into the general model.

3.2.1 Peak and Off-peak Subperiods on Weekdays and Weekends

The time stages in the general model are taken to be months although other shorter or longer and even nonuniform periods are acceptable. In PG&E's model, each period (month) is divided into four subperiods. These subperiods represent the aggregate of hours in the month that fall in peak and in off-peak hours on weekdays and on weekends. The main reason for this is that the marginal value of power varies significantly by subperiod.

A consequence of the inclusion of subperiods is that nearly every arc in the physical network is replaced by four arcs. Release decisions are then made for each subperiod. Similarly, any node which represents a junction or small reservoir where no significant storage takes place is replaced by four nodes. This ensures that water entering a junction during a certain subperiod is used during that subperiod. Small reservoirs are also represented by four nodes and have additional (inter-temporal) control arcs which allow water to be transferred between subperiods if those subperiods contain adjacent hours (see Figure 3.4). Large reservoirs have enough buffer that they can handle storage fluctuations and therefore need be represented by just a single node.

Although the inclusion of the subperiods increases the size of the network nearly four times, the nature of the network still fits the form of the general model while much more accurately depicting actual circumstance. These aggregate subperiods are a useful way to capture the essence of peak and off-peak times which would otherwise necessitate two decision periods per *day*.

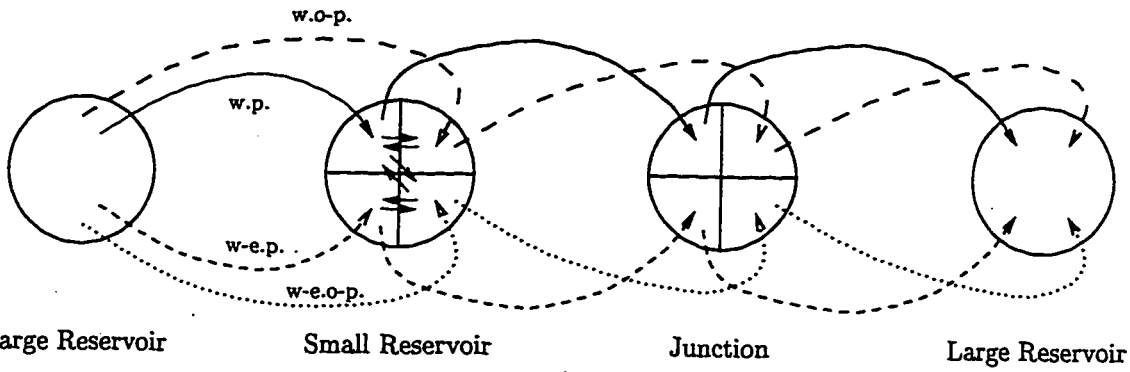

Figure 3.4: An example portion of the physical network showing the expanded treatment of nodes and arcs for subperiods (where "w.p." indicates weekday peak subperiods, "w-e.o-p." indicates weekend off-peak hours, etc.). In the small reservoir, notice the subperiod carryover arcs which allow (small amounts of) water to be held over to a *following* subperiod. The long arcs represent both turbined and spilled releases in the subperiod.

Pump Stations

In the Kings River basin (see test problem in Chapter 5) there is a pump-house (actually just the powerhouse run backwards) which is used to pump water uphill from reservoir two to reservoir one. It takes about $1\frac{1}{2}$ times as much power to pump the water uphill as is produced when the water is released through the powerhouse below reservoir one. The marginal value of power is greater during peak hours than during off-peak hours. In a model without some way to distinguish peak from off-peak hours, it would never be optimal to run that pump-house. Results show that the powerhouse is to be run during peak hours (some months only on weekdays) and the pump-house is to be run only during off-peak hours. This result is an interesting but not surprising testimonial to the necessity of these subperiods.

In this way, a pump station can be thought of as a way to store power for later use. Power produced in off-peak hours can be used to pump water uphill for use in power production during peak hours when the power is more valuable.

3.2.2 Baseloaded and Load-Following Powerhouses

A powerhouse can be classified as *baseloaded* or *load following*. A load following powerhouse can have a different generation level for each subperiod while a baseloaded powerhouse is run at a constant level throughout the stage (Jacobs et al., 1995).

In the PG&E system, every powerhouse can be run as either a baseloaded or a load following powerhouse (where a portion of the turbines in the powerhouse can be designated for each purpose). This dual designation increases the number of powerhouse arcs in the model. Many of the load following powerhouse arcs are also duplicated to model multiple "load shapes" across a subperiod. In the adapted model used here, the load shapes are all taken to be "flat" for simplicity.

3.2.3 Reservoir Carryover Arcs

In PG&E's model, rather than considering reservoir storage to be water that was not released (a state variable), a *carryover* arc is used to represent storage as a release decision from one time stage into the next. Small reservoirs do not have carryover arcs between stages, but do have arcs representing carryover between subperiods in one stage (see Figure 3.4).

Carryover arcs can eliminate the difficulty of dealing with state constraints, but the network must be enlarged so that the carryover arcs can reach from the physical network in one time stage into the physical network in the next. This makes for a very large network.

The model used here will utilize only the inter-subperiod carryover arcs for the small reservoirs. Instead of the inter-stage carryover (storage) arcs for the large reservoirs, the storage into the following stage is tracked by the system state x_i via the dynamics.

3.2.4 The Cost Function

In the general model, the expected cost function is written as

$$\sum_{i \in \mathcal{I}} \pi_i g_i(u_i)$$

where each $g_i(u_i)$ is the convex cost associated with supplying, by means other than hydro, the shortage in power after produced hydro is subtracted from the demand.

The PG&E model has a more specific form of the expected cost function. Here, u_i is partitioned into turbined (^{t}) releases $u^t_{i,\tau}$ and spilled (^{s}) releases $u^s_{i,\tau}$ for each of the four subperiods (τ). Then

$$g_i(u_i) = \sum_{\tau=1}^{4} \sum_{h=1}^{H^\tau} \int_0^{d^h_{i,\tau} - \beta \cdot u^t_{i,\tau}} M^\tau_i(\xi) d\xi + \sum_{\tau=1}^{4} \alpha_{i,\tau} \cdot u^s_{i,\tau}$$

where M^τ_i is the marginal cost of non-hydro power, given as data for each subperiod in each month, $d^h_{i,\tau}$ is the load (power demand) on the system and is given for each sub-subperiod $h = 1, \ldots, H^\tau$ of each subperiod for each month and β is the vector of powerhouse efficiencies (power is assumed a linear function of release). Notice that the integral of the monotone increasing marginal cost data over the shortage, which is affine, produces a convex function.

In the second term in the cost function, $\alpha_{i,\tau}$ is a vector of "costs" representing incentives (negative) or disincentives (positive) to spill for some spillways (fish or irrigation) in some months.

A third term in PG&E's cost function representing "no load costs" is neglected here as its effects are small yet it adds greatly to the complexity of the cost function. These are the costs of keeping unloaded turbines in "spinning reserve" (ready) in case of turbine failure elsewhere.

3.3 Hydro Engineering Literature Review

In his review article on reservoir models and management, Yeh (1985) states that "during the past 20 years, one of the most important advances made in the field of

water resources engineering is the development and adoption of optimization techniques for planning, design and management of complex water resources systems." He goes on to bemoan the gap between mathematical theory and the application of that theory in this setting. Now, 12 years later, that gap appears to be closing.

This section surveys optimization techniques and related modeling advances for the long term hydro scheduling problem. This review will primarily focus on the evolution of (to) multistage stochastic optimization techniques. This includes stochastic dynamic programming techniques and (more recently) stochastic programming techniques.

Yeh's (1985) review paper surveys many variations on stochastic and deterministic linear, nonlinear and dynamic programming. Yakowitz (1982) reviews dynamic programming (DP) models and techniques for a variety of water resource systems. Reznicek and Chang (1991) review stochastic modeling of resource operations.

Many authors agree that Masse (1946, in French) was the first to employ a stochastic DP-like algorithm to the "reservoir inventory problem" (Yakowitz 1982). Another early paper was Little (1955) who considered the operation of a single reservoir as a stochastic DP inventory problem. Through the late 1970's (and beyond) much of the research focused on modeling structure within the DP framework with many authors contributing (cf. Yeh, 1985; Yakowitz, 1982). Howard (1960) and Manne (1962) described the state transitions as a Markov process (rather than random transitions). Su and Deininger (1974) added time dependence of Markov transition probabilities to reflect seasonality. Askew (1974a) combined DP and simulation to maximize expected benefit while constraining the probability of system failure (*i.e.* to meet demand). Askew (1974b) incorporated a penalty for system failure thereby affecting the release policy by reducing expected benefits associated with policies which fail to meet demand. Young (1976) described the mass balance equation (for the deterministic case) similarly to the dynamics currently employed. Some later works (Alercon and Marks (1979), Maidment and Chow (1981), Bras et al. (1983), Stedinger et al.(1984),

Kelman et al. (1990)) examined ways of improving models of reservoir inflows (see "observed vs. forecasted inflows" in Section 3.1.4).

The Curse of Dimensionality

One difficulty which many authors involved in dynamic programming lamented was what Bellman (1957) called the "curse of dimensionality". Recall that to employ the DP algorithm, typically one discretizes the state space. As a result, if the dimension of the state space is increased, there is an exponential increase in computational effort. For the multiperiod hydro scheduling problem, that meant that a power system with 3 – 4 reservoirs was considered *large* from a computational standpoint. Early research often considered just a single reservoir.

Much of the research from 1980 through the present has focused on ways to overcome these dimensionality limitations and so allow consideration of larger power systems. Deterministic techniques can handle larger power systems and are useful in the planning process, but they do not capture the stochastic nature (inflow) of the long-term hydro scheduling problem. Deterministic models, using mean monthly inflows, tend to overestimate system benefits and underestimate costs and losses due to failure (*i.e.* power shortage, flood, *etc.*). A comprehensive analysis should weigh expected performances against costs of failure. Thus, stochastic models and techniques continue to be developed.

Techniques have been developed as modifications of the (stochastic) DP algorithm. In the *parameter iteration method* (Gal, 1979) the control is assumed to be a function of state characterized by a set of parameters which are improved at each iteration by a least squares minimization. The *aggregation-decomposition* method (Turgeon, 1980; Valdes et al, 1992) and a decomposition method in Turgeon (1981) effectively reduce dimensionality by aggregating downstream reservoirs into one composite reservoir; resulting in a sequence of problems with a state dimension of two. These methods rely on approximations which may compromise accuracy for the sake

of modeling a larger power system. In fact, the aggregation-decomposition method is not guaranteed to converge to a global optimum. *Gradient dynamic programming* (Foufoula-Georgiou and Kitanidas, 1988) uses Hermite interpolation of the cost-to-go function to reduce dimensionality by allowing a coarser state discretization. Similarly, Johnson et al. (1988, 1993) use *tensor-product cubic spline* interpolation. These interpolation schemes have been shown to significantly reduce CPU times for stochastic problems with up to 5 state variables (Johnson et al., 1993) but do not appear to be reasonable for significantly larger problems.

Still other techniques do not require state space discretization. This is where the future lies. Papageorgiou (1985) and Mizyed et al. (1992) apply a discrete maximum principle. Ikura and Gross (1984) use nonlinear programming. Chara and Pant (1984) use a *successive variations* technique in which the higher-dimensional problem is tackled as a series of one-dimensional (state) subproblems. But these methods are predominantly for the deterministic problem.

Stochastic Programming

Recall (from Section 2.2) that in stochastic programming, there is no need to discretize the state space. This should, in principle, allow consideration of much larger (power) systems.

Dupacova (1980) applied *two-stage* (recourse) SP to the hydro scheduling problem. But this really is a multistage problem.

The current fashion for multistage SP is a stochastic *Benders decomposition* approach derived from the deterministic Benders decomposition method by Birge (1980) (see also Birge (1985) and Wets (1988)) and derived separately from dynamic programming (stochastic dual DP (SDDP)) by Pereira and Pinto (1985). For these Benders schemes to be practical, the objective function must be linear (or piecewise linear convex). Pereira and Pinto (1985, 1991), Pereira (1989) and Gorenstin et al. (1992) apply their Benders scheme (SDDP) to case studies on portions of the Brazilian power

system involving scheduling for 37 – 44 hydroplants on a 9 month planning horizon. Jacobs et al. (1995) applies a Benders scheme to PG&E's large power system with a 24 month planning horizon (their modeling improvements are described in Section 3.2). They also tested various "tree traversing strategies" and other algorithmic improvements. Archibald et al. (1996) present computational comparisons between Benders and DP approaches. In each of these articles, a scenario tree underlies the information/decision structure.

The *dynamic splitting* SP algorithm, to be introduced in Section 4.6 and applied to the hydro scheduling problem in Chapter 5, is also structured on a scenario tree. It is, however, designed with a *convex* cost objective in mind to more properly model the nonlinear costs. A case study is presented involving scheduling on a portion of PG&E's power system, containing the equivalent of 176 hydroplants and 174 controlled spillways on 94 reservoirs, over a 24 month planning horizon.

Chapter 4

OPERATOR SPLITTING AND APPLICATION TO STOCHASTIC PROGRAMMING

Methods of partitioning or decomposition have long been used in mathematics to *reduce* a problem to one of iteratively solving a collection of smaller or easier problems. These types of methods continue to be emphasized with the advent of parallel computing structures which make it possible to to solve the smaller problems all at the same time. One such class of methods is known as *splitting methods for monotone operators* (or operator splitting methods). The main purpose of these methods is to find a zero of a monotone operator (or sum of monotone operators) by "splitting" the operator into two or more simpler operators and then exploiting the special structure of the resulting subproblems.

A monotone operator (precise definition later) can be thought of as generalizing, to multivalued nonlinear functions, the property of positive semi-definiteness. The gradient or subgradient of a convex function provides the most famous example of a monotone operator. One can easily see that the minimization of the sum of convex functions becomes the problem of finding a zero of the sum of monotone operators. A thorough review of splitting methods for monotone operators can be found in Eckstein (1989).

Starting from the point of view of a steepest descent path for convex minimization and the proximal point method, this chapter will present a brief review of operator splitting methods as well as a more detailed description of Spingarn's splitting method. These schemes are then applied to unconstrained convex optimization as well

as to the saddle point problem. We will then present (in Section 4.6) a new algorithm, called *dynamic splitting*, utilizing Spingarn's method (Spingarn 1983,1985), for the saddle point problem ($\mathcal{S}$) arising from the constrained convex multistage stochastic programming problem ($\mathcal{P}_{cmsp}$) introduced in Section 2.2.3.

It should be noted that application of operator splitting to convex programming is a fairly recent development. Operator splitting methods have been known for much longer in numerical analysis on linear algebra and differential equations. But these operators where predominantly linear and single-valued (Ferziger 1981). Lions and Mercier (1979) extended operator splitting to multivalued monotone operators and applied it to the convex optimization setting. In this paper, operator splitting is applied to convex multistage stochastic programming.

In Chapter 5, the algorithm will be specialized for the hydropower generation scheduling problem introduced in Chapter 3. Previous multistage stochastic programming algorithms for this large scale problem have focused on the linear case via a Benders decomposition approach. The dynamic splitting algorithm is the first to be published for the nonlinear (convex) case, though Pacific Gas and Electric (PG&E) has an unpublished algorithm in use (personal communication with Gary Schultz (1995)).

4.1 *Monotone Operators and Subgradients*

What follows are a few definitions and results from monotone operator theory which are relevant to the subsequent developments.

Definition 4.1 (Monotone Operator) *(a) A multivalued function* $T:\Re^n \rightrightarrows \Re^n$ *is said to be a* monotone operator *if*

$$\langle x - x', y - y' \rangle \geq 0 \quad \text{whenever} \quad y \in T(x),\ y' \in T(x')$$

where $\langle \cdot, \cdot \rangle$ *is the inner product.*

(b) An operator T *is said to be* strongly monotone *if there is a constant* $\mu > 0$ *such*

that

$$\langle x - x', y - y' \rangle \geq \mu \|x - x'\|^2 \quad \text{whenever} \quad y \in T(x),\ y' \in T(x').$$

Equivalently, the mapping $T - \mu I$ *is monotone.*

(c) An operator T *is said to be* maximal monotone *if it is monotone and its graph*

$$G(T) = \{(x, y) \in \Re^n \times \Re^n | y \in T(x)\}$$

is not strictly contained in the graph of any other monotone operator $T' : \Re^n \rightrightarrows \Re^n$.

Definition 4.2 (Subgradient) *A vector* y *is said to be a* subgradient *of a convex function* $h : \Re^n \to \Re \cup \{+\infty\}$ *at a point* x *if*

$$h(x + d) \geq h(x) + \langle d, y \rangle \quad \forall d \in \Re^n.$$

The set of all subgradients of h *at* x *is denoted* $\partial h(x)$. *If* y *is a subgradient of* h *at* x, *we write*

$$y \in \partial h(x).$$

This brings us to the classic result proved by Rockafellar (1966); special cases were known previously to Minty (1962):

Theorem 4.3 (Monotonicity of the Subgradient Operator) *If* $h : \Re^n \to \Re \cup \{+\infty\}$ *is convex, then* $\partial h : \Re^n \rightrightarrows \Re^n$ *is a monotone operator. If* h *is also closed and proper, then* ∂h *is a maximal monotone operator on* $\Re^n$.

Definition 4.4 *Consider any function* $L : \Re^n \times \Re^m \to [-\infty, +\infty]$. *Then*

(a) $L(u, v)$ *is termed a* saddle-function *if it is convex in* u *for all* $v \in \Re^m$ *and concave in* v *for all* $u \in \Re^n$.

(b) A saddle function L *is said to be* proper *if*

$$\left\{u \in \Re^n | L(u, v) < +\infty\ \forall v \in \Re^m\right\} \times \left\{v \in \Re^m | L(u, v) > -\infty\ \forall u \in \Re^n\right\} \neq \emptyset$$

For a saddle function L on $\Re^n \times \Re^m$, Rockafellar (1970b) defines the associated operator $T_L(u,v)$ to be the set of $(w,z) \in \Re^n \times \Re^m$ such that

$$\begin{aligned} L(u',v) - \langle u',w\rangle + \langle v,z\rangle &\geq L(u,v) - \langle u,w\rangle + \langle v,z\rangle \\ &\geq L(u,v') - \langle u,w\rangle + \langle v',z\rangle \end{aligned} \qquad (T_L)$$

for all $u' \in \Re^n$, $v' \in \Re^m$.

The following result, adapted from Rockafellar (1970b), provides conditions for the maximal monotonicity of T_L.

Theorem 4.5 *Let L be a proper saddle-function on $\Re^n \times \Re^m$ such that the function $h_{v'}(u) = L(u,v')$ is lower semi-continuous on $\Re^n$ for all $v' \in \Re^m$ and such that the function $k_{u'}(v) = -L(u',v)$ is upper semi-continuous on $\Re^m$ for all $u' \in \Re^n$. Then T_L is a maximal monotone operator.*

4.2 Steepest Descent Path

To lay a framework for the introduction of operator splitting methods, we will investigate the following unconstrained problem:

$$\underset{x\in\Re^n}{\text{minimize}}\ h(x) \qquad (\mathcal{MIN})$$

where $h:\Re^n \to \Re \cup \{+\infty\}$ is closed proper convex.

Problem $(\mathcal{MIN})$ is equivalent to the problem

$$\text{find } x \in \Re^n \text{ such that } 0 \in \partial h(x) \qquad (\mathcal{Z})$$

under the same conditions on h. (The inclusion is replaced by $\nabla h(x) = 0$ if h is differentiable.)

One way to find a solution of the minimization problem, in principle, is to follow the steepest descent path from some initial guess. Given the initial guess x_0 (see

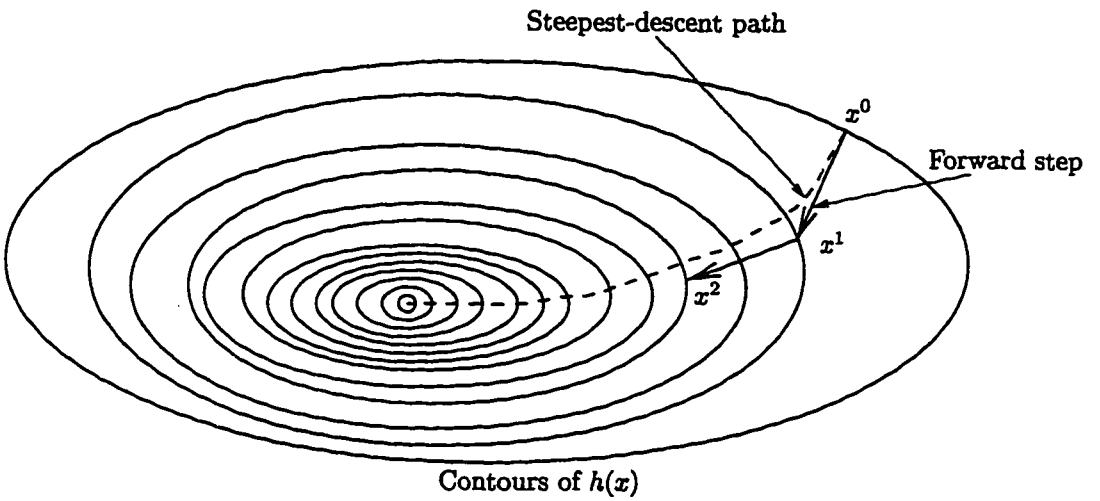

Figure 4.1: Two forward steps. Notice that the forward step is normal to the contour at the initial point of the step. The steepest descent path is normal to all contours.

Figure 4.1), parameterize the steepest descent path (supposing it exists) as $x = x(t)$. An equation for that path, for h differentiable, is then given by

$$\frac{dx}{dt} = -\nabla h(x(t)), \qquad x(0) = x_0.$$

The Gradient Method

Applying a forward Euler scheme (see Figure 4.1), with step sizes c^k, to the differential equation yields

$$x^{k+1} = x^k - c^k \nabla h(x^k)$$

or, in operator terms

$$x^{k+1} = (I - c^k \nabla h)(x^k).$$

This iterative process is known as the gradient method for solving the unconstrained differentiable convex minimization problem $(\mathcal{MIN})$.

For h non-differentiable, this generalizes to the *subgradient* method

$$x^{k+1} \in (I - c^k \partial h)(x^k).$$

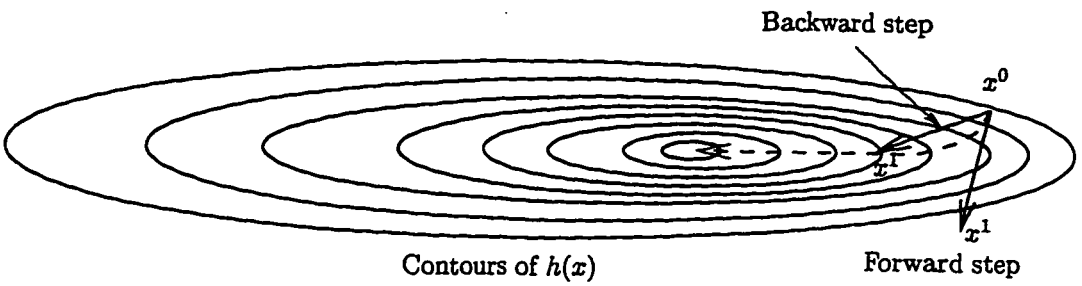

Figure 4.2: An ill-conditioned problem in $\Re^2$. The backward step is normal to the contour at the terminal point of the step while the forward step is normal to the contour at its initial point. Notice how, for this ill-conditioned problem, the forward step not only fails to approximate the steepest-descent path (dashed) but also fails to make an improvement toward a minimizer of h.

The gradient and subgradient methods, when applied to convex minimization, are typically combined with a line search method which samples the objective function in the step direction to choose a step size c^k. In this case, for differentiable convex h, convergence can be superlinear (cf. Bertsekas, 1982).

By examining Figure 4.2, one can imagine the convergence difficulties the forward step could encounter without the aid of a line search method. For example, overstepping (as in the figure) or a very slow "zig-zag" toward the solution for this sort of ill-conditioned problem. In the case where no line search is used, convergence results rely on various restrictive assumptions on step size (cf. Eckstein, 1989) and convergence rates can be sublinear for an ill-conditioned problem.

This type of method can also be applied to the more general problem of finding a zero of a maximal monotone operator:

$$\text{find } x \in \Re^n \text{ such that } 0 \in T(x) \qquad (\mathcal{Z}1)$$

(which is equivalent to problems $(\mathcal{MIN})$and $(\mathcal{Z})$ in the case where $T = \partial h$ and h is closed proper convex.)

If T is *any* maximal monotone operator, then the subgradient method is general-

ized, as the forward step method

$$x^{k+1} \in (I - c^k T)(x^k) \qquad \text{(FWD)}$$

for problem ($\mathcal{Z}1$) as given an initial guess x_0. Of course the inclusion becomes an equality if T is single-valued at x^k.

In the general monotone operator case, there is often no objective function to sample and so a line search rule could not be applied. Also, in operator splitting methods, even as applied to convex minimization, line search will not be appropriate.

4.2.1 Proximal Point Method

Another method can be derived from the steepest descent path equation (written this time for general (nondifferentiable) h)

$$\frac{dx}{dt} \in -\partial h(x(t)), \qquad x(0) = x_0$$

by employing a backward Euler step in place of the forward step. The backward step is known to exhibit better convergence rates, in general, than the forward step while being more difficult, in general, to apply. The backward step is written

$$x^{k+1} \in x^k - c^k \partial h(x^{k+1}).$$

Notice that the difference between the forward and backward steps is the point at which the gradient is evaluated (see Figure 4.2 for geometric differences).

Solving for x^{k+1} gives

$$x^{k+1} \in (I + c^k \partial h)^{-1}(x^k). \qquad \text{(B)}$$

The general case, where T is any *maximal* monotone operator, may be written

$$x^{k+1} \in (I + c^k T)^{-1}(x^k)$$

and is termed the backward step method (or the *proximal point algorithm* (Rockafellar, 1976a)). It turns out (Minty, 1962) that, since T is maximal and c^k is a positive

scalar, the inverse exists and is single-valued. Hence, the inclusion above can be replaced by the equality

$$x^{k+1} = (I + c^k T)^{-1}(x^k). \tag{BWD}$$

Also at issue is how to perform this proximal step, which involves the inverting of an operator.

Remark 4.6 (Proximal Optimization Step) *If $T = \partial h$ (for h closed proper convex), the backward step (BWD) turns out to be equivalent to computing*

$$x^{k+1} = \arg\min_{x \in \Re^n} \left\{ h(x) + \frac{1}{2c^k} \|x - x^k\|^2 \right\} \tag{PRX}$$

(Rockafellar, 1976a).

In practice the parameter c^k is often replaced by a diagonal matrix of parameters c_j^k. This is equivalent to a change of norm.

To prove that this statement is indeed equivalent to the above proximal step (BWD), notice that the minimand is strictly convex and form the inclusion $0 \in \partial\{h(x) + \frac{1}{2c^k}\|x - x^k\|^2\}$. Rearranging gives the inclusion (B), which is equivalent to (BWD) since $T = \partial h$ here.

In considering this iterative approach, one must weigh the fact that a single problem is converted into a sequence of problems. One advantage stems from the fact that the single problem ($\mathcal{MIN}$) was convex while the sequence of problems are each *strongly* convex. "Strong convexity is a boon to good convergence" (Rockafellar, 1976b).

The proximal point algorithm is also a major building block in the development of monotone operator splitting methods.

A nice property of the proximal point algorithm occurs when h is separable. Since the norm-squared is also separable, it turns out that the proximal step itself is separable. This important property is exploited later in the setting of operator splitting.

The following convergence result is due to Rockafellar (1976a).

Theorem 4.7 (Convergence of Proximal Point Method) *Let $\{x^k\}$ be any sequence generated by the proximal point method (BWD), with $c^k \nearrow c \leq \infty$, for solving $0 \in T(\overline{x})$, where T is maximal monotone. If there exists an α such that*

$$\|x - \overline{x}\| \leq \alpha \|w\| \quad \textit{when} \quad x \in T^{-1}(w)$$

for (w, x) close to $(0, \overline{x})$, then x^k converges linearly to a zero of T if one exists. Otherwise $\|x^k\| \to \infty$.

Moreover, there is an index $\overline{k}$ such that

$$\|x^{k+1} - \overline{x}\| \leq \frac{1}{\sqrt{1 + (c/\alpha)^2}} \|x^k - \overline{x}\| \quad \textit{for } k \geq \overline{k}.$$

Notice that by increasing c (or decreasing α) one can increase the rate of linear convergence of the proximal point algorithm. In fact, as $c^k \to \infty$, *superlinear* convergence is achieved.

Rockafellar also provides similar results under inexact calculation of $\{x^k\}$.

4.3 Operator Splitting Methods

With the groundwork of forward and backward steps, the stage is set to investigate operator splitting. The forward step (FWD) is typically easy to implement, especially when T is single-valued, but without the aid of a line search method, convergence can be slow. The backward step (BWD) exhibits better convergence in general. It also is less adversely effected by ill-conditioning and allows much more leeway in selecting step sizes than the forward step. But the backward step can be difficult to implement in some situations.

In operator splitting methods, decomposition of the problem into "smaller" problems can ease the implementation difficulties. Then, the application of backward or

a combination of forward and backward steps to the subproblems can be used to benefit convergence.

Given a maximal monotone operator T, we again (as in problem ($\mathcal{Z}1$)) desire to find $x \in \Re^n$ such that $0 \in T(x)$. Now imagine that T can be "split" so that $T = T_1 + T_2$ where T_1 and T_2 are also maximal monotone. Our problem is then formulated

$$\text{find } x \in \Re^n \text{such that } 0 \in (T_1 + T_2)(x) \qquad (\mathcal{Z}2)$$

In operator splitting methods, we apply backward or forward steps to T_1 and T_2 separately in some combination at each iteration. These splitting schemes should have the property, as the proximal and forward steps do, that a fixed point of the iteration solves problem ($\mathcal{Z}2$).

4.3.1 Forward-Backward Splitting

Imagine now that T_1 and T_2 (maximal monotone) above can be chosen so that T_1 is single-valued and so that the *resolvent* of T_2, which is $(I + c^k T_2)^{-1}$, is much easier to compute than was the resolvent of T (or is possible to compute where the resolvent of T was not). In this situation, it may be useful to try a *forward-backward splitting* scheme, which is formulated

$$x^{k+1} \in (I + c^k T_2)^{-1}(I - c^k T_1)(x^k). \qquad \text{(F-B)}$$

(The inclusion is replaced by an equality if T_1 is single-valued.) Each iteration proceeds in two steps as a forward step for operator T_1 acting on x^k, followed by a backward step for operator T_2 acting on the result. It should be noted that if $\overline{x}$ is a fixed point of this iteration, then $0 \in (T_1 + T_2)(\overline{x})$.

Recent convergence results of Chen and Rockafellar (1997) no longer require strong monotonicity of T_1^{-1} (as in Gabay (1983) and Tseng (1988)) for guaranteed linear convergence. They also prescribe an "optimal step size relative to certain constants associated with the given problem."

4.3.2 Peaceman-Rachford Spitting

A variant on forward-backward splitting is provided by reversing the roles of T_1 and T_2 at each iteration. The *Peaceman-Rachford Splitting* is written

$$x^{k+1} \in (I + c^k T_2)^{-1}(I - c^k T_1)(I + c^k T_1)^{-1}(I - c^k T_2)(x^k). \qquad \text{(P-R)}$$

Lions and Mercier (1979) provide various criteria which guarantee linear convergence. They also describe methods ("tight" or "loose") for choosing a single value of the forward step $(I - c^k T_j)(x)$ when T_j is multivalued. Linear convergence is guaranteed when both T_1 and T_2 are single valued and T_2 is strongly monotone.

Requiring both T_1 and T_2 (and thus T) to be single-valued can limit the suitability of the method in the optimization setting. Another drawback of this method is that it requires the implementation of both backward and forward steps on both operators. In the optimization setting, this can be difficult or impossible.

Still, this class of methods has proven popular (outside of optimization) in the numerical solution of partial differential equations (Ferziger, 1981) where the operators are generally linear and single-valued.

4.3.3 Douglas-Rachford Spitting

Another method is the *Douglas-Rachford Spitting* scheme

$$x^{k+1} \in (I + c^k T_2)^{-1}[(I + c^k T_1)^{-1}(I - c^k T_2) + c^k T_2](x^k) \qquad \text{(D-R)}$$

which Douglas and Rachford (1954) derived from a power-series analysis of a discretization of the heat equation (Eckstein, 1989). Of the splitting methods discussed to here, this is the only one which is shown (Lions and Mercier, 1979) to converge linearly for general maximal monotone choices of T_1 and T_2 (with constant step size $c^k \equiv c$ and with the assumption that there exists $x \in \Re^n$, $y_1 \in T_1(x)$, $y_2 \in T_2(x)$ such that $y_1 + y_2 = 0$). As for (P-R), Lions and Mercier describe methods for ac-

commodating multivaluedness, but the method only seems sensible when T_2 is single valued.

As with Peaceman-Rachford, This method has mostly been applied for single-valued linear operators in a numerical differential equations setting and could prove cumbersome to implement in an optimization setting.

4.3.4 Double-Backward Splitting

Unlike the previous schemes, the *double-backward splitting*, formulated

$$x^{k+1} = (I + c^k T_2)^{-1}(I + c^k T_1)^{-1}(x^k) \qquad \text{(D-B)}$$

does not have the property, in general, that a fixed point of the iteration provides a solution of problem ($\mathcal{Z}2$). Passty (1979) has shown that strict conditions on c^k are required even to ensure merely ergodic convergence (convergence in the mean). The fixed point condition

$$x = (I + cT_2)^{-1}(I + cT_1)^{-1}(x)$$

can be reformulated as

$$0 \in T_1(x) + T_2(x + cT_1(x)).$$

Passty's requirements include $c^k \to 0$. "Then the fixed point condition *approaches*, in some sense, the desired condition $0 \in (T_1 + T_2)(x)$" (Eckstein, 1989).

There is a special case where the double-backward scheme converges for any choices of $c^k > 0$ (cf. Eckstein, 1989). This is the case where T_1 and T_2 are the normal cone operators for a pair of intersecting closed convex sets in $\Re^n$. In fact, in this case, a fixed point of the iteration is seen to be a solution of $0 \in (T_1 + T_2)(x)$.

Of the schemes introduced so far, only double-backward splitting requires only backward steps. One could imagine situations where this would be advantageous (e.g. when both T_1 and T_2 are multivalued). But, the value of the method is dubious

due to its poor convergence rate even under the strict restrictions on step size c^k. Fortunately, Spingarn (1983) has developed a method that requires only backward steps while avoiding these pitfalls.

4.4 *Spingarn's Splitting Method*

One final operator splitting method will be discussed here and in rather more detail. *Spingarn's splitting* method is derived from the proximal point method acting on a specially contrived *partial inverse* operator (to be defined). Thus, Spingarn's method inherits its guaranteed linear convergence from the proximal point method. This section will present a derivation (following Spingarn, 1983) of the method of partial inverses and its application to the problem of finding a zero of the sum of monotone operators.

4.4.1 Method of Partial Inverses

The derivation will be carried out for the two-operator case. In the two-operator case, the derivation can be carried further than the general multi-operator cased, resulting in a form of Spingarn's method which will be most useful subsequently.

Definition 4.8 (Partial Inverse Operator) *Given an operator T and complimentary sets $A, B \subseteq \Re^n$ (i.e. $A = B^\perp$, $B = A^\perp$), define the* partial inverse *operator T_A such that*

$$T_A(x_A + y_B) = x_B + y_A$$

where $y \in T(x)$ and where x_A, x_B, y_A, y_B are the projections of x and y onto A and B.

Remark 4.9 (Monotonicity of T_A) *The partial inverse operator T_A is (maximal) monotone if and only if T is (maximal) monotone (Spingarn, 1983).*

Proposition 4.10 *The following two problems are equivalent:*

(a) Find $x \in A$, $y \in B$ such that $y \in T(x)$.

(b) Find $x \in A$, $y \in B$ such that $0 \in T_A(x+y)$.

Proof. $x \in A$, $y \in B$ implies $x = x_A$, $y = y_B$. Apply the definition of T_A. □

The motivation here is that in trying to find a zero of the sum of monotone operators ($T = T_1 + T_2$), we end up with subproblems like (a) and a natural way to define A and B. To solve the subproblems, operator T_A is defined so that the proximal point method can be applied to the equivalent subproblems (b). In the final formulation (of Spingarn's method) T_A is eliminated and the algorithm is written in terms of separate proximal steps for T_1 and T_2. For the desired decomposition in T_1 and T_2 to arise, we must use $c^k \equiv 1$ in the application of the proximal point method.

Apply the Proximal Point Method to solving $0 \in T_A(x+y)$

Write $z = x + y$ with $x \in A$, $y \in B$, then the proximal point iteration with $c^k \equiv 1$ (and with iteration counter $k, k+1$ suppressed) is

$$z^+ = (I + T_A)^{-1}(z).$$

We desire a method of finding the next iterate $z^+ = x^+ + y^+$ with $x^+ \in A$, $y^+ \in B$ which does not involve T_A explicitly.

To do this, define

$$x' = (I + T)^{-1}(x + y).$$

Then $x + y = x' + y'$ where $y' \in T(x')$.

From the previous lines and then the definition of T_A

$$z = x + y = x' + y' = (x'_A + y'_B) + (x'_B + y'_A) \in (I + T_A)(x'_A + y'_B).$$

But $z \in (I + T_A)(z^+)$ so

$$z^+ = x'_A + y'_B$$

or equivalently, $x^+ = x'_A$ and $y^+ = y'_B$.

Thus, an iteration of the method of partial inverses for problem (a) proceeds (with iteration counter k reintroduced) in two steps; as a proximal step

$$x' = (I+T)^{-1}(x^k + y^k), \quad y' = x^k + y^k - x' \tag{PI}$$

followed by a projection step

$$x^{k+1} = x'_A, \quad y^{k+1} = y'_B.$$

Application of Partial Inverses to problem ($\mathcal{Z}2$); Spingarn's Method

To derive Spingarn's method, the method of partial inverses (PI) is applied to the problem ($\mathcal{Z}2$).

Equivalently, find $x \in \Re^n$, $y \in \Re^n$ such that

$$y \in T_1(x) \text{ and } -y \in T_2(x) \tag{$\mathcal{Z}2'$}$$

(then $0 \in (T_1 + T_2)(x)$).

Define $A = \{(x_1, x_2) | x_1 = x_2 \in \Re^n\}$ and $B = \{(y_1, y_2) | y_1 = -y_2 \in \Re^n\}$ and notice that $A = B^\perp$.

Our problem ($\mathcal{Z}2'$) of finding the zero of the sum of monotone operators can now be formulated as one of finding $(x_1, x_2) \in A$, $(y_1, y_2) \in B$ such that

$$(y_1, y_2) \in (T_1 \times T_2)(x_1, x_2).$$

Then, by Proposition 4.10, the problem may be equivalently formulated in terms of finding $(x_1, x_2) \in A$, $(y_1, y_2) \in B$ such that

$$0 \in T_A\left((x_1, x_2) + (y_1, y_2)\right) \tag{$\mathcal{ZA}$}$$

where T_A is the partial inverse operator corresponding to the operator $T_1 \times T_2$.

Following the method of partial inverses, for a current x $(= x_1 = x_2)$ and $y_1 = -y_2$, first perform the proximal step

$$x_j' = (I + T_j)^{-1}(x + y_j), \quad y_j' = x + y_j - x_j', \qquad j = 1, 2$$

and then project onto A and B to find the new x, y_1, y_2 iterates

$$x^+ = \frac{1}{2}(x_1' + x_2'), \quad y_j^+ = y_j' - \frac{1}{2}(y_1' + y_2'), \quad j = 1, 2$$

This same process could be applied for a splitting of T into any finite sum of operators.

Notice that for (y_1^+, y_2^+) to be in B, it is necessary that $y_2^+ = -y_1^+$ so we define $y^+ = -y_1^+ = y_2^+$.

Finally, replace T_j by λT_j $(j = 1, 2)$ to introduce a parameter for the purpose of speeding convergence $(0 \in (T_1 + T_2)(x) \Leftrightarrow 0 \in (\lambda T_1 + \lambda T_2)(x))$. Notice that λ does not arise as a step size like the parameter c^k in the proximal point algorithm.

Spingarn's splitting method is found by substituting the proximal step into the projection step and regrouping:

$$\begin{aligned} x^{k+1} &= \frac{1}{2}\left[(I + \lambda T_1)^{-1}(x^k - y^k) + (I + \lambda T_2)^{-1}(x^k + y^k)\right] \\ y^{k+1} &= \frac{1}{2}\left[(I + \lambda T_1)^{-1}(x^k - y^k) - (I + \lambda T_2)^{-1}(x^k + y^k)\right] + y^k \end{aligned} \tag{SS}$$

(where the iteration index k has been reintroduced).

In this method, only backward (proximal) steps are performed. Therefore the method may prove useful if the resolvent of maximal monotone T is difficult or impossible to compute but where T can be split so that the resolvents of T_1 and T_2 are significantly easier to compute and such that T_1 and T_2 are maximal monotone. Multivaluedness of T_1 and T_2 poses no added difficulty.

Spingarn (1983) proves the following convergence result based on the fact that the method is derived from direct application of the proximal point method to problem $(\mathcal{ZA})$ where T_A is the partial inverse operator associated with operator $T_1 \times T_2$ (Note: T_A maximal monotone by Remark 4.9).

Theorem 4.11 (Spingarn's Method Convergence) *Suppose that T_1 and T_2 are maximal monotone. In Spingarn's method (SS) with $\lambda > 0$, if a solution of problem ($\mathcal{Z}2'$) exists, then x^k converges linearly to $\overline{x}$ and y^k converges linearly to $\overline{y}$ such that $-\overline{y} \in T_1(\overline{x})$ and $\overline{y} \in T_2(\overline{x})$ (i.e. $0 \in (T_1 + T_2)(\overline{x})$).*

Moreover, if there exists an α such that

$$\|(x+y) - (\overline{x}+\overline{y})\| \leq \alpha\|w\| \quad \textit{when} \quad (x+y) \in T^{-1}(w)$$

for $(w, x+y)$ close to $(0, \overline{x}+\overline{y})$, then there is an index $\overline{k}$ such that

$$\|(x^{k+1}+y^{k+1}) - (\overline{x}+\overline{y})\| \leq \frac{1}{\sqrt{1+(1/\alpha)^2}}\|(x^k+y^k) - (\overline{x}+\overline{y})\| \quad \textit{for } k \geq \overline{k}.$$

The second part of the theorem is a direct consequence of Theorem 4.7. A rate of linear convergence result for x^k (rather than $x^k + y^k$) might be more desirable. Still, this result implies that if α can be decreased, faster convergence should result.

The exact nature of the relationship between λ and α is not obvious. However, it does seem evident and numerical tests concur, that adjusting λ can help speed convergence. This will be discussed further in Section 5.5.

4.5 *Application of Operator Splitting to Unconstrained Optimization*

Recall, from Section 4.2, the connection between finding a zero of a maximal monotone operator and finding a solution of a convex minimization problem. In the previous sections, the discussion has centered around iterative methods, involving forward and backward (proximal) steps, for the solution to the monotone operator problem ($\mathcal{Z}2$). The connection arises when the monotone operators T_j are the gradient or subgradient of convex functions h_j whose sum we desire to minimize.

Recall also (Section 4.2) that the proximal step $(I + c^k T)^{-1}(x^k)$ in optimization is usually computed (PRX) as the

$$\arg\min_{x \in \mathbb{R}^n} \left\{ h(x) + \frac{1}{2c^k}\|x - x^k\|^2 \right\}.$$

Thus, in the forward-backward (F-B) splitting method for the minimization of the sum of convex functions $h_1(x) + h_2(x)$, each iteration would be completed in two steps as

$$y^k \in (I - c^k \partial h_1)(x^k)$$

$$x^{k+1} = \arg\min_{x \in \Re^n} \left\{ h_2(x) + \frac{1}{2c^k} \|x - y^k\|^2 \right\}.$$

Spingarn's Method

Recall Spingarn's method (SS) from the previous section

$$x^{k+1} = \frac{1}{2}\left[(I + \lambda T_1)^{-1}(x^k - y^k) + (I + \lambda T_2)^{-1}(x^k + y^k)\right],$$
$$y^{k+1} = \frac{1}{2}\left[(I + \lambda T_1)^{-1}(x^k - y^k) - (I + \lambda T_2)^{-1}(x^k + y^k)\right] + y^k$$

and notice that unlike the previous methods, the proximal steps are performed in parallel and occur in each of the steps of the method.

Now imagine that T_1 and T_2 arise as the subgradients of convex functions h_1 and h_2 and that we wish to minimize $h = h_1 + h_2$ on $\Re^n$.

To write the algorithm (SS) in a more useful form, represent the proximal terms as

$$\tilde{u}^k = x^k - y^k \quad \text{and} \quad \hat{u}^k = x^k + y^k$$

and define $(u^1)^k$ and $(u^2)^k$ as the result of the proximal steps as follows:

$$(u^1)^k = \arg\min_{u \in \Re^n} \left\{ h_1(u) + \frac{1}{2\lambda} \|u - \tilde{u}^k\|^2 \right\} \qquad \text{(SS')}$$
$$(u^2)^k = \arg\min_{u \in \Re^n} \left\{ h_2(u) + \frac{1}{2\lambda} \|u - \hat{u}^k\|^2 \right\}.$$

Then, the proximal terms are updated as

$$\tilde{u}^{k+1} = (u^1)^k + \frac{1}{2}(\tilde{u}^k - \hat{u}^k)$$
$$\hat{u}^{k+1} = (u^2)^k + \frac{1}{2}(\hat{u}^k - \tilde{u}^k)$$

and the current iterate toward the solution is

$$u^{k+1} = \frac{1}{2}((u^1)^k + (u^2)^k).$$

Similarly other methods for monotone operator splitting, including those mentioned before, can be formulated in the convex minimization setting.

Convergence results for the splitting methods carry over to the optimization applications with conditions on h_1 and h_2 specified so that the conditions on $T_1 = \partial h_1$ and $T_2 = \partial h_2$ in the hypothesis, are met.

Remark 4.12 (Convergence for (SS')) *(a) Recall that to derive (SS') from (SS), T_1 and T_2 arise as subgradients of the convex functions h_1 and h_2. If additionally h_1 and h_2 are closed and proper, then the convergence results of Theorem 4.11 hold also for the method (SS') where $x^{k+1} = u^{k+1}$ and $y^{k+1} = \frac{1}{2}((u^1)^k - (u^2)^k) + y^k$.*

(b) By the convergence of $y^k \to \overline{y}$, we get $(u^1)^k - (u^2)^k \to 0$. With $x^k \to \overline{x}$, this shows that the solution of each proximal subproblem in (SS') converges to the minimizer (if one exists).

Choosing a Splitting Method

So how does one choose between forward-backward splitting, Spingarn's method, and the other splitting schemes? The decision is based on the individual problem structure, which of course stems from earlier decisions on problem formulation and reformulation. The decision on method also depends on our choice of how to "split" the operator (actually, how to split the objective and constraints) and the structure of the resulting functions.

Imagine that h_j is one of the convex functions arising from the splitting of the objective (i.e. $T_j = \partial h_j$ is one of the operators arising from the operator splitting). If h_j is separable, then, because the norm-squared is also separable, the proximal step itself would be separable. For this h_j, the proximal step would likely be favored over

the forward step. One can see that separability is a desirable property to look for when choosing a splitting.

If instead, the function h_j is a composite function (arising for example as a reduced Lagrangian with dynamics) it may be impossible to actually form or evaluate the subgradient at x. For this h_j, the forward step would not be appropriate whereas the proximal step might be.

A differentiable h_j with easily computed gradient (*e.g.* single-valued) might be most appropriately handled by the forward step.

Clearly, different decisions on problem formulation and operator splittings (equal parts art and science) could result in very different algorithms for the same problem.

The Saddle Point Problem

Recall from Section 2.2, the triad of the primal, dual and saddle point problems associated with a Lagrangian L on $U \times V$:

$$\underset{u \in U}{\text{minimize}}\ f(u) \quad \text{where } f(u) = \sup_{v \in V} L(u,v). \tag{$\mathcal{P}$}$$

$$\underset{v \in V}{\text{maximize}}\ g(v) \quad \text{where } g(v) = \inf_{u \in U} L(u,v). \tag{$\mathcal{D}$}$$

$$\text{find } (\overline{u}, \overline{v}) \in U \times V \tag{$\mathcal{S}$}$$

$$\text{such that : } L(\overline{u}, \overline{v}) = \min_{u \in U} L(u, \overline{v}) = \max_{v \in V} L(\overline{u}, v)$$

where u and v are the primal and dual variables.

The splitting methods described earlier could be applied to the primal or adapted to the dual problem formulations. We would like a splitting method for the saddle point problem. This will be seen in the next section to be an advantage due to a certain separability which will arise only in the splitting of the saddle point form of the problem.

By applying the proximal point method (BWD) to the monotone operator T_L

(defined in Section 4.1), Rockafellar (1976b) showed that the proximal point method for the saddle point problem can be formulated

$$(u^{k+1}, v^{k+1}) = \arg\min_{u\in U}\max_{v\in V}\left\{L(u,v) + \frac{1}{2c^k}\|u-u^k\|^2 - \frac{1}{2d^k}\|v-v^k\|^2\right\}. \quad (\mathrm{PRX}_s)$$

Remark 4.13 (Convergence for (PRX$_s$)) *Suppose L is a proper saddle-function on $U\times V$ which satisfies the lower and upper semi-continuity conditions of Theorem 4.5. Then the convergence results of Theorem 4.7 hold also for the saddle point version of the proximal point method (PRX$_s$) where $T = T_L$, $x^k = (u^k, v^k)$ and $d^k = c^k$.*

As mentioned previously, the parameters c^k and d^k are often replaced, in practice, by diagonal matrices of parameters c_j^k and d_j^k. This is equivalent to a change of norm.

Spingarn's Method Extended to the Saddle Point Problem

Imagine that L can be split so that $L = L_1 + L_2$ with L_1 and L_2 convex-concave and L_1 finite. Then the splitting algorithms can similarly be reformulated for the saddle point problem. Since it will be of use later, Spingarn's method for the saddle point problem is formulated by applying (SS) to the maximal monotone operators T_{L_1} and T_{L_2} arising from the saddle-functions L_1 and L_2 (with the iteration-counter k suppressed and with the $\tilde{u}, \hat{u}, \ldots$ notations of the last section) as follows: The proximal steps are

$$(u^1, v^1) = \arg\min_{u\in U}\max_{v\in V}\left\{L_1(u,v) + \frac{1}{2\lambda}\|u-\tilde{u}\|^2 - \frac{1}{2\mu}\|v-\tilde{v}\|^2\right\} \quad (\mathrm{SS}_s)$$

$$(u^2, v^2) = \arg\min_{u\in U}\max_{v\in V}\left\{L_2(u,v) + \frac{1}{2\lambda}\|u-\hat{u}\|^2 - \frac{1}{2\mu}\|v-\hat{v}\|^2\right\},$$

the proximal term updates are

$$(\tilde{u}, \tilde{v})^+ = (u^2, v^2) + \frac{1}{2}((\tilde{u}, \tilde{v}) - (\hat{u}, \hat{v}))$$

$$(\hat{u}, \hat{v})^+ = (u^1, v^1) + \frac{1}{2}((\hat{u}, \hat{v}) - (\tilde{u}, \tilde{v}))$$

and the current iterates toward the primal and dual solutions are

$$u^+ = \frac{1}{2}(u^1 + u^2) \text{ and } v^+ = \frac{1}{2}(v^1 + v^2).$$

Remark 4.14 (Convergence for (SS_s)) *Suppose that L_1 and L_2 are proper saddle-functions on $U \times V$ which satisfy the upper and lower semi-continuity conditions of Theorem 4.5. Then the convergence results of Theorem 4.11 hold also for the saddle point version of Spingarn's method (SS_s) where $T_1 = T_{L_1}$, $T_2 = T_{L_2}$ and $x^{k+1} = (u^+, v^+)$, $y^{k+1} = \frac{1}{2}(u^1 - u^2, v^1 - v^2) - y^k$ with $\mu = \lambda$.*

Moreover, by Remark 4.12(b), $(u^1)^k - (u^2)^k \to 0$ and $(v^1)^k - (v^2)^k \to 0$.

In practice λ and μ can be replaced by diagonal matrices of parameters λ_j and μ_j. Adjustment of λ and μ can lead to faster convergence.

4.6 An Operator Splitting for Multistage Stochastic Programming

This section presents one of the major results of this thesis; the development of the new *dynamic splitting* algorithm for the saddle point problem associated with the convex multistage stochastic programming problem ($\mathcal{P}_{cmsp}$) introduced in Section 2.2.3. The algorithm utilizes Spingarn's method to decompose the saddle point problem into two main subproblems to be solved at each iteration. The dynamic subproblem is solved via a linear feedback loop (see Section 2.3.3). For the separable subproblem, specific information about function structure is needed to fill in the details of a solution method. This is done in Chapter 5 for the case of the hydropower scheduling problem of Chapter 3.

In Section 2.2.3, a saddle point problem was formulated relative to the reduced Lagrangian L associated with the convex multistage stochastic problem ($\mathcal{P}_{cmsp}$). In Section 4.5, Spingarn's method (SS_s) was formulated for the saddle point problem. The idea here is to apply that method to the multistage stochastic problem with the

appropriate choice of splitting $L = L_1 + L_2$ (giving rise to the appropriate operator splitting).

To this end, rewrite the reduced Lagrangian (from Section 2.2.3) associated with the convex problem $(\mathcal{P}_{cmsp})$

$$L(u,v) = \underbrace{\sum_{i\in\mathcal{I}} \pi_i \{c_i \cdot x_i - v_i \cdot C_i x_i\}}_{L_1(u,v)} + \underbrace{\sum_{i\in\mathcal{I}} \pi_i \{r_i \cdot u_i + \varphi_i(u_i) + d_i \cdot v_i - \psi_i(v_i) + v_i \cdot D_i u_i\}}_{L_2(u,v)}$$

$$\text{where: } x_{i+} = A_i x_i + B_i u_i + b_{i+}, \quad x_0 = b_0,$$

on $U \times V$ where $\psi_i(v_i) = \delta_{V_i}(v_i)$ and $\varphi_i(u_i) = g_i(u_i) + \delta_{U_i}(u_i)$ are the convex functions defined in $(\mathcal{P}_{cmsp})$. As mentioned previously, we will assume that $\infty - \infty = \infty$.

The *augmented* reduced Lagrangian (from Section 2.2.4) associated with problem $(\mathcal{P}_{cmsp})$ could also be used here. The following exposition would be more cumbersome for that case. In Chapter 5, the augmented form will be utilized.

Notice that L_1 could also be written in terms of the y_i which would be defined by the dual dynamics. The primal form of L_1 will be used here although the dual form might be advantageous in other situations.

This choice of splitting $(L = L_1 + L_2)$ results from the desire to exploit the decomposable structure of the reduced Lagrangian. Notice the separable structure of L_2, which can be written

$$L_2(u,v) = \sum_{i\in\mathcal{I}} \pi_i l_i(u_i, v_i)$$

$$\text{where: } l_i(u_i, v_i) = r_i \cdot u_i + \varphi_i(u_i) + d_i \cdot v_i - \psi_i(v_i) + v_i \cdot D_i u_i$$

is clearly convex-concave in u_i, v_i. This separable structure makes the monotone operator associated with L_2 a strong candidate for the proximal point step.

Notice also that because of the indicator functions (included in φ_i and ψ_i), L_2 is not differentiable in general. This illustrates the necessity for methods (like proximal point, Spingarn's, forward-backward) which can accommodate *multivalued* monotone operators.

The biaffine structure of

$$L_1(u,v) = \sum_{i \in \mathcal{I}} \pi_i (c_i \cdot x_i - v_i \cdot C_i x_i)$$

$$\text{where : } x_{i+} = A_i x_i + B_i u_i + b_{i+}, \quad x_0 = b_0$$

contains all links between the different decision states. The function L_1 is differentiable, so a forward step for the associated monotone operator could be considered, though the composite nature of L_1 with the dynamics would make the gradient in the forward step difficult to form. One of the keys to the algorithm lies in the method for solving the proximal step arising from L_1.

4.6.1 The Dynamic Splitting Algorithm

We have chosen the saddle point problem formulation of the multistage problem. We have chosen a particular splitting of the reduced Lagrangian due to its decomposable structure. From the structure of the resulting functions, we have chosen proximal steps for both subproblems and will therefore apply Spingarn's method to the saddle point problem. The true justification for these choices comes in hindsight at the point of having found reasonable methods for the subproblems and later from numerical tests showing that the algorithm actually performs as designed (with the help of assorted implementation tricks).

The proximal subproblems at each iteration arising from Spingarn's method (SS_s) for the saddle point problem as applied to our splitting of the reduced Lagrangian

are as follows:

$$(u^1, v^1) = \arg \min_{u \in \Re^N} \max_{v \in \Re^M} \sum_{i \in \mathcal{I}} \pi_i \left\{ c_i \cdot x_i - v_i \cdot C_i x_i + \frac{1}{2\lambda} \|u_i - \tilde{u}_i\|^2 - \frac{1}{2\mu} \|v_i - \tilde{v}_i\|^2 \right\} \tag{$\mathcal{P}1$}$$

$$\text{subject to : } x_{i+} = A_i x_i + B_i u_i + b_{i+}, \quad x_0 = b_0;$$

$$(u^2, v^2) = \arg \min_{u \in \Re^N} \max_{v \in \Re^M} \sum_{i \in \mathcal{I}} \pi_i \left\{ l_i(u_i, v_i) + \frac{1}{2\lambda} \|u_i - \hat{u}_i\|^2 - \frac{1}{2\mu} \|v_i - \hat{v}_i\|^2 \right\} \tag{$\mathcal{P}2$}$$

$$\text{where : } l_i(u_i, v_i) = r_i \cdot u_i + \varphi_i(u_i) + d_i \cdot v_i - \psi_i(v_i) + v_i \cdot D_i u_i.$$

The proximal term updates are

$$(\tilde{u}, \tilde{v})^+ = (u^2, v^2) + \frac{1}{2}((\tilde{u}, \tilde{v}) - (\hat{u}, \hat{v}))$$
$$(\hat{u}, \hat{v})^+ = (u^1, v^1) + \frac{1}{2}((\hat{u}, \hat{v}) - (\tilde{u}, \tilde{v})),$$

and the current iterate toward the primal and dual solutions are

$$u = \frac{1}{2}(u^1 + u^2) \quad \text{and} \quad v = \frac{1}{2}(v^1 + v^2).$$

Remark 4.15 (Convergence Results for Dynamic Splitting) *For the splitting $L = L_1 + L_2$ of problem ($\mathcal{P}_{cmsp}$) defined above, define the operators $T_1 = T_{L_1}$ and $T_2 = T_{L_2}$. Then T_1 and T_2 are maximal monotone. With $\mu = \lambda$ and with $x^{k+1} = (u^+, v^+)$, $y^{k+1} = \frac{1}{2}(u^1 - u^2, v^1 - v^2) - y^k$ at each iteration, the convergence results of Theorem 4.11 hold also for the* dynamic splitting *algorithm as applied to the problem ($\mathcal{P}_{cmsp}$).*

Moreover, by Remark 4.12(b) $(u^1)^k - (u^2)^k \to 0$ and $(v^1)^k - (v^2)^k \to 0$.

It remains to be explained how to solve the subproblems.

4.6.2 *Subproblem* ($\mathcal{P}1$)*: The Dynamic Subproblem*

At each iteration, subproblem ($\mathcal{P}1$) is a saddle point problem in its own right and therefore has an associated primal problem of the form

$$\underset{u\in\mathbb{R}^N}{\text{minimize}}\; f_1(u) = \sup_{v\in\mathbb{R}^M}\left\{\sum_{i\in\mathcal{I}}\pi_i(c_i\cdot x_i - v_i\cdot C_i x_i + \frac{1}{2\lambda}\|u_i-\tilde{u}_i\|^2 - \frac{1}{2\mu}\|v_i-\tilde{v}_i\|^2)\right\}$$
$$\text{where:}\quad x_{i+} = A_i x_i + B_i u_i + b_{i+},\quad x_0 = b_0$$

Noticing that the supremand is concave and unconstrained in v_i for all $i\in\mathcal{I}$, the gradient is set to zero. Solving that equation reveals that

$$\bar{v}_i = \tilde{v}_i - \mu C_i x_i,\quad \forall\, i\in\mathcal{I}.$$

Substituting gives

$$f_1(u) = \sum_{i\in\mathcal{I}}\pi_i\left\{(c_i - C_i\tilde{v}_i)\cdot x_i + \frac{1}{2}\mu(C_i x_i)\cdot C_i x_i + \frac{1}{2\lambda}\|u_i-\tilde{u}_i\|^2\right\},$$

and the resulting minimization, written in a general form (and dropping constant terms), becomes

$$\underset{u\in\mathbb{R}^M}{\text{minimize}}\;\sum_{i\in\mathcal{I}}\pi_i\left\{\frac{1}{2}x_i\cdot Q_i x_i + q_i\cdot x_i + \frac{1}{2}u_i\cdot R_i u_i + r_i\cdot u_i\right\}$$
$$\text{where: } x_{i+} = A_i x_i + B_i u_i + b_{i+},\quad x_0 = b_0$$

and where $Q_i = \mu C_i^{\mathsf{T}} C_i$, $q_i = c_i - C_i\tilde{u}_i$, $R_i = \frac{1}{\lambda}I$, and $r_i = -\frac{1}{\lambda}\tilde{u}_i$.

Formulated in this way, the remaining problem can be thought of as an unconstrained dynamic programming problem on the scenario tree with linear dynamics and quadratic cost. Recall the linear feedback solution for this problem presented in Section 2.3.3. Proposition 2.13 shows that this dynamic programming technique can be used to find a solution to the stochastic programming (sub)problem.

4.6.3 *Subproblem ($\mathcal{P}2$); The Separable Subproblem*

Utilizing the separability of L_2 and the norms, subproblem ($\mathcal{P}2$) can be formulated: For each $i \in \mathcal{I}$

$$(u_i^2, v_i^2) = \arg \min_{u_i \in \Re^{n_i}} \max_{v_i \in \Re^{m_i}} \left\{ l_i(u_i, v_i) + \frac{1}{2\lambda}\|u_i - \hat{u}_i\|^2 - \frac{1}{2\mu}\|v_i - \hat{v}_i\|^2 \right\}$$

$$\text{where: } l_i(u_i, v_i) = r_i \cdot u_i + \varphi_i(u_i) + d_i \cdot v_i - \psi_i(v_i) + v_i \cdot D_i u_i.$$

Since φ_i and ψ_i contain indicator functions for U_i, V_i representing box constraints on u_i, v_i, the problem can be formulated for each $i \in \mathcal{I}$

$$(u_i^2, v_i^2) = \arg \min_{u_i \in U_i} \max_{v_i \in V_i} \left\{ g_i(u_i) + d_i \cdot v_i + v_i \cdot D_i u_i + \frac{1}{2\lambda}\|u_i - \hat{u}_i\|^2 - \frac{1}{2\mu}\|v_i - \hat{v}_i\|^2 \right\}$$

where the $r_i \cdot u_i$ is absorbed into the $g_i(u_i)$.

Further, if v_i is unconstrained (or if u_i is unconstrained) then this saddle point problem should be written in equivalent primal (dual) form. Then the supremum (infimum) problem would be unconstrained in v_i (u_i) and could be solved in closed form (as was done with the supremum in subproblem ($\mathcal{P}1$)).

Without knowledge of the exact nature of g_i as well as U_i and V_i, it would be difficult to be specific about how to continue from here. As mentioned previously, this will be taken up in Chapter 5 for the hydropower scheduling example.

Chapter 5

APPLICATION OF DYNAMIC SPLITTING TO THE HYDRO SCHEDULING PROBLEM

Since the 1960's, power systems researchers have been developing algorithms to solve the hydropower scheduling problem. Most of the algorithms developed through the 1980's focused on coercing or approximating the dynamic programming algorithm to handle this state-constrained and notoriously dimensionally-cursed problem (see literature review, Section 3.3). Algorithms have been developed which could find solution for increasingly sophisticated power scheduling models. At the same time, mathematical programming researchers, in and out of the hydro field, have been developing models and algorithms for the two-stage (recourse programming) and, recently, the multistage stochastic programming problem. In the hydro setting, the successful multistage stochastic programming algorithms have utilized a Benders decomposition approach (cf. Pereira and Pinto, 1985 and 1991; Jacobs et al., 1995). These algorithms have been found to work well for the case of linear or piecewise linear cost objective. But it is well known that the costs involved are not linear. The dynamic splitting algorithm (introduced in Section 4.6) is designed for the *nonlinear* (convex) multistage stochastic programming problem.

In this chapter we apply the dynamic splitting algorithm to the hydro scheduling problem (presented in Sections 3.1 and 3.2). Unlike in Section 4.6, the *augmented* reduced Lagrangian (Section 2.2.5) will be used in formulating the associated saddle point problem. This is advantageous because the terms associated with the state constraint remain intact under the operator splitting of the augmented Lagrangian.

Section 5.1 will recall the hydro scheduling model. Sections 5.2 and 5.3 will present the specialization of the dynamic splitting algorithm for the hydro scheduling problem. Recall that in the algorithm, after the associated saddle point problem is formulated, an operator splitting results in two main subproblems to be solved at each iteration. The dynamic subproblem is solved via the linear feedback loop (of Section 2.3.3). An algorithm for the separable convex box-constrained subproblem, exploiting the structure inherited from the hydro scheduling problem via the Fenchel conjugate function, is developed in Section 5.3. Some important properties of Fenchel conjugate functions are given below.

Section 5.4 introduces the test problem (developed from data supplied by PG&E) and provides some results. Section 5.5 presents performance enhancing modifications. Section 5.6 presents duality gap results and discusses the overall performance of the algorithm. Section 5.7 compares the duality gap results with those for an infeasible version of the test problem.

The Fenchel Conjugate

Suppose F is a closed proper convex function on $\Re^n$. Then

$$F^*(y) = \sup_x \{y \cdot x - F(x)\}$$

is the *Fenchel Conjugate* of $F(x)$. Some important properties of F^* follow:

Proposition 5.1 *Suppose F is a closed proper convex function on $\Re^n$. Then:*

a) F^* is also closed proper convex; and

b) $\partial F^*(y) = [\partial F]^{-1}(y)$.

c) If F is strictly convex, then F^* is differentiable.

(cf. Rockafellar, 1970a, 1984).

5.1 The Hydro Scheduling Model

Recall the hydropower scheduling model of Section 3.1.6 and the improvements of the PG&E model discussed in Section 3.2. Combining, we wish to

$$\begin{aligned} &\underset{u_i \in U_i,\, i \in \mathcal{I}}{\text{minimize}} \sum_{i \in \mathcal{I}} \pi_i g_i(u_i) \\ \text{subject to}: \quad & x_i = x_{i^-} + B u_i + b_i, \quad x_0 = b_0 && \forall i \in \mathcal{I} \\ & \underline{s}_i \le x_i \le \overline{s}_i && \forall i \in \mathcal{I} \\ & x_i = b_T && \forall i \in \mathcal{T}, \end{aligned}$$

where the information and decision structure is arrayed on the scenario tree. All notation is defined in Section 3.1.6. The release u_i can be partitioned into turbined release $u^t_{i,\tau}$ and spilled release $u^s_{i,\tau}$ for each of the four subperiods $\tau = 1, \ldots, 4$ (Section 3.2.1). Then the cost functionals (Section 3.2.4) can be written

$$g_i(u_i) = \sum_{\tau=1}^{4} \sum_{h=1}^{H^\tau} \int_0^{d^h_{i,\tau} - \beta \cdot u^t_{i,\tau}} M^\tau_i(\xi) d\xi + \sum_{\tau=1}^{4} \alpha_{i,\tau} \cdot u^s_{i,\tau}, \quad i \in \mathcal{I}$$

where M^τ_i is the marginal cost of non-hydro power, given as data for each subperiod in each time stage (month), where $d^h_{i,\tau}$ is the load (power demand) on the system and is given for each sub-subperiod $h = 1, \ldots, H^\tau$ of each subperiod in each stage, where β is the vector of powerhouse efficiencies (power is assumed a linear function of release) and where the $\alpha_{i,\tau}$ are imposed costs which encourage or discourage certain spills. Notice that the integral of the monotone increasing marginal cost data over the shortage, which is affine, produces a convex cost as a function of release.

5.2 Dynamic Splitting for the Hydro Scheduling Problem

In this section, the dynamic splitting algorithm is applied to the hydro scheduling problem. Following Section 4.6, the first step toward application of the dynamic

splitting algorithm is formulation of a reduced Lagrangian for the problem. The *augmented* reduced Lagrangian will be used here due to the advantage of this formulation, which stems from the ability to appropriately split the augmented Lagrangian without splitting the terms arising from the state constraint.

The Augmented Reduced Lagrangian

As a step toward formulating the augmented reduced Lagrangian (Section 2.2.5) for the hydro scheduling problem, the state constraints $\underline{s}_i \leq x_i \leq \overline{s}_i,\ i \in \mathcal{I}$ and $x_i = b_T,\ i \in \mathcal{T}$ are reformulated

$$\begin{aligned} x_i - w_i &= 0 \quad && i \in \mathcal{I} \\ \underline{s}_i \leq w_i &\leq \overline{s}_i \quad && i \notin \mathcal{T} \\ w_i &= b_T \quad && i \in \mathcal{T}, \end{aligned}$$

where $w_i \in \Re^m$ is the augmented control variable.

The augmented reduced Lagrangian for the hydro scheduling problem can then be formulated

$$L(u,w;v) = \sum_{i\in\mathcal{I}} \pi_i \Big\{ g_i(u_i) + v_i \cdot (x_i - w_i) + \delta_{U_i}(u_i) + \delta_{W_i}(w_i) \Big\}$$

where: $x_i = x_{i^-} + Bu_i + b_i, \quad x_0 = b_0,$

$$U = \Pi_{i\in\mathcal{I}} U_i, \quad U_i = \{u_i \in \Re^n | \underline{u}_i \leq u_i \leq \overline{u}_i\}$$

$$W = \Pi_{i\in\mathcal{I}} W_i, \quad W_i = \{w_i \in \Re^m | \underline{s}_i \leq w_i \leq \overline{s}_i,\ i \notin \mathcal{T};\ w_i = b_T,\ i \in \mathcal{T}\}$$

$$V = \Pi_{i\in\mathcal{I}} V_i, \quad V_i = \Re^m$$

on $(U \times W) \times V$. Notice that L is a proper saddle function as stipulated in Definition 4.4.

The Operator Splitting

Next, we choose a "splitting" of L (effectively choosing a splitting of the operator T_L (defined in Section 4.1)). On $(U\times W)\times V$,

$$L(u,w;v) = \underbrace{\sum_{i\in\mathcal{I}} \pi_i v_i\cdot(x_i - w_i)}_{L_1(u,w;v)} + \underbrace{\sum_{i\in\mathcal{I}} \pi_i\{g_i(u_i) + \delta_{U_i}(u_i) + \delta_{W_i}(w_i)\}}_{L_2(u,w;v)}$$

$$\text{where: } x_i = x_{i^-} + Bu_i + b_i, \quad x_0 = b_0$$

and where L_1 and L_2 are chosen so as to be proper saddle functions. In the algorithm, this splitting gives rise to two main subproblems, $(\mathcal{P}1)$ and $(\mathcal{P}2)$, to be solved at each iteration:

$$(u^1,w^1,v^1) = \underset{(u,w)\in\Re^N\times\Re^M}{\arg\min}\ \max_{v\in\Re^M} \sum_{i\in\mathcal{I}} \pi_i\Big\{v_i\cdot(x_i - w_i) + \frac{1}{2\lambda}\|u_i - \tilde{u}_i\|^2 + \frac{1}{2\nu}\|w_i - \tilde{w}_i\|^2 - \frac{1}{2\mu}\|v_i - \tilde{v}_i\|^2\Big\} \quad (\mathcal{P}1)$$

$$\text{subject to: } x_i = x_{i^-} + Bu_i + b_i, \quad x_0 = b_0.$$

$$(u^2,w^2,v^2) = \underset{(u,w)\in\Re^N\times\Re^M}{\arg\min}\ \max_{v\in\Re^M} \sum_{i\in\mathcal{I}} \pi_i\Big\{g_i(u_i) + \delta_{U_i}(u_i) + \delta_{W_i}(w_i) + \frac{1}{2\lambda}\|u_i - \hat{u}_i\|^2 + \frac{1}{2\nu}\|w_i - \hat{w}_i\|^2 - \frac{1}{2\mu}\|v_i - \hat{v}_i\|^2\Big\} \quad (\mathcal{P}2)$$

where $(\tilde{u}_i,\tilde{w}_i,\tilde{v}_i)$ and $(\hat{u}_i,\hat{w}_i,\hat{v}_i)$ are the proximal terms in the current iterate and where (u^1,w^1,v^1) and (u^2,w^2,v^2) are the solutions of the subproblems in the current iterate.

Recall from Section 4.6 that the proximal terms are updated at each iteration as

$$(\tilde{u},\tilde{w},\tilde{v})^+ = (u^2,w^2,v^2) + \frac{1}{2}((\tilde{u},\tilde{w},\tilde{v}) - (\hat{u},\hat{w},\hat{v}))$$
$$(\hat{u},\hat{w},\hat{v})^+ = (u^1,w^1,v^1) + \frac{1}{2}((\hat{u},\hat{w},\hat{v}) - (\tilde{u},\tilde{w},\tilde{v})),$$

and that the current iterates toward the primal, augmented and dual solutions are

$$u = \frac{1}{2}(u^1+u^2), \quad w = \frac{1}{2}(w^1+w^2) \text{ and } v = \frac{1}{2}(v^1+v^2).$$

5.2.1 *The Unconstrained Dynamic Subproblem ($\mathcal{P}1$)*

Following Section 4.6.2, we exploit the unconstrained and convex-concave (in w and v) nature of ($\mathcal{P}1$). Setting derivatives (in w and v) to zero, we find that, for each $i \in \mathcal{I}$,

$$v_i^1 = \frac{\mu(x_i - \tilde{w}_i) + \tilde{v}_i}{1 + \mu\nu} \quad \text{and} \quad w_i^1 = \frac{\mu\nu x_i + \nu\tilde{v}_i + \tilde{w}_i}{1 + \mu\nu}.$$

Subproblem ($\mathcal{P}1$) is then reduced to

$$\underset{u \in \mathcal{R}^N}{\text{minimize}} \sum_{i \in \mathcal{I}} \pi_i \left\{ \frac{1}{2} x_i \cdot Q x_i + q_i \cdot x_i + \frac{1}{2} u_i \cdot R u_i + r_i \cdot u_i \right\} \qquad (\mathcal{P}1')$$

$$\text{subject to: } x_i = x_{i^-} + B u_i + b_i, \quad x_0 = b_0$$

for determining u^1 where, for $i \in \mathcal{I}$,

$$Q = \frac{\nu\mu^2 + \mu}{(1 + \nu\mu)^2} I, \quad q_i = \frac{1}{1 + \mu\nu}(\tilde{v}_i - \mu\tilde{w}_i), \quad R = \frac{1}{\lambda} I, \quad r_i = -\frac{1}{\lambda}\tilde{u}.$$

This reduced problem ($\mathcal{P}1'$) is then an unconstrained DP with linear dynamics and quadratic costs and is solved via the linear feedback loop of Section 2.3.3.

Theorem 5.2 (Feedback Loop Solution for Problem ($\mathcal{P}1'$)) *A linear feedback loop solution for problem ($\mathcal{P}1'$) can be obtained for each $i \in \mathcal{I}$ as*

$$u_i^*(x_i) = -(R + B^\top K_i B)^{-1}(r_i + B^\top(l_i + K_i(x_{i^-} + b_i)))$$

where x_i is given by the dynamics and where the symmetric positive definite matrices K_i and the vectors l_i^+, l_i are defined, for $i \in \mathcal{T}$, as $K_i = Q$ and $l_i = q_i$ and recursively defined, for $i \notin \mathcal{T}$, as

$$l_i^+ = \sum_{j \in \{i^+\}} p_j \left[l_j + K_j b_j \right],$$

$$K_i = Q + K_{i^+} - K_{i^+} B (R + B^\top K_{i^+} B)^{-1} B^\top K_{i^+},$$

$$l_i = q_i + l_i^+ - K_{i^+} B (R + B^\top K_{i^+} B)^{-1}(r_i + B^\top l_i^+).$$

Proof. The Proposition follows easily from Theorem 2.14. □

Notice that Q and R do not depend on i. Thus, all K_i in one time stage are equivalent. Moreover, since Q and R do not change from one iteration to the next (as long as λ, μ and ν remain fixed), the inverse $(R+B^\top K_{i+}B)^{-1}$ need be computed (LU factorization) only in the first iteration. Recall from the proof of Theorem 2.14 that $R+B^\top K_{i+}B$ is symmetric positive definite.

Separability

Recall that B is the node-arc incidence matrix. If B is block diagonal (as is the case for a hydropower system consisting of distinct watersheds), then the DP subproblem is block separable. For the hydro scheduling problem, this means that the dynamic subproblem is separable into parallel subproblems for each distinct watershed.

5.3 The Box-Constrained Convex Subproblem ($\mathcal{P}2$)

Following Section 4.6.3 in exploiting the separability of L_2 and the norm-squared in subproblem ($\mathcal{P}2$), we seek for each $i \in \mathcal{I}$,

$$(u_i^2, w_i^2, v_i^2) =$$

$$\underset{(u_i,w_i)\in U_i\times W_i}{\arg\min}\ \max_{v_i\in\Re^M} \left\{ g_i(u_i) + \frac{1}{2\lambda}\|u_i-\hat{u}_i\|^2 + \frac{1}{2\nu}\|w_i-\hat{w}_i\|^2 - \frac{1}{2\mu}\|v_i-\hat{v}_i\|^2 \right\}$$

This subproblem formulation is further separable in the primal, augmented and dual variables. The resulting problems can be written as follows:

The Augmented Variable Subproblem

For each $i \in \mathcal{I}$

$$\underset{\underline{s}_i\le w_i\le \overline{s}_i}{\text{minimize}}\ \frac{1}{2\nu}\|w_i-\hat{w}_i\|^2.$$

The Dual Variable Subproblem

For each $i \in \mathcal{I}$

$$\underset{v_i \in \Re^n}{\text{minimize}} \frac{1}{2\mu}\|v_i - \hat{v}_i\|^2.$$

The Primal Variable Subproblem

$$\underset{u_i \in U_i,\, i \in \mathcal{I}}{\text{minimize}} \left\{ g_i(u_i) + \frac{1}{2\lambda}\|u_i - \hat{u}_i\|^2 \right\}$$

$$\text{where: } g_i(u_i) = \sum_{\tau=1}^{4} \sum_{h=1}^{H^\tau} \int_0^{d^h_{i,\tau} - \beta \cdot u^t_{i,\tau}} M^\tau_i(\xi) d\xi + \sum_{\tau=1}^{4} \alpha_{i,\tau} \cdot u^s_{i,\tau}.$$

The dual and augmented variable problems are solved easily in closed form. The primal variable problem is further separable into *turbined* and *spilled* release problems for each of the four subperiods.

The Spilled Release Subproblem

For each $\tau = 1, \ldots, 4$ in each $i \in \mathcal{I}$

$$\underset{\underline{u}^s_{i,\tau} \le u^s_{i,\tau} \le \overline{u}^s_{i,\tau}}{\text{minimize}} \left\{ \alpha_{i,\tau} \cdot u^s_{i,\tau} + \frac{1}{2\lambda}\|u^s_{i,\tau} - \hat{u}^s_{i,\tau}\|^2 \right\}.$$

The spilled release problem is a box-constrained quadratic minimization and is also easily solved in closed form. The turbined release problem is more challenging.

5.3.1 The Turbined Release Subproblem

The turbined release subproblem ($\mathcal{TRS}$) is formulated (with subscripts suppressed (i.e. u represents $u^t_{i,\tau}$)) for each $\tau = 1, \ldots, 4$ in each $i \in \mathcal{I}$ as

$$\underset{\underline{u} \le u \le \overline{u} \in \Re^{n^t}}{\text{minimize}} \left\{ \sum_{h=1}^{H} \int_0^{d^h - \beta \cdot u} M(\xi) d\xi + \frac{1}{2\lambda}\|u - \hat{u}\|^2 \right\} \qquad (\mathcal{TRS})$$

where n^t is the number of turbined release arcs (powerhouse arcs) in the system.

The key observation in the development of an algorithm for this subproblem is that the whole first term above can be thought of as a function of the scalar value $z = \beta \cdot u$ representing the total hydro-power produced in that subperiod.

Let

$$\phi^h(z) = \int_0^{d^h - z} M(\xi) d\xi \quad \text{and} \quad \Phi(z) = \sum_h^H \phi^h(z)$$

where $z = \sum_{j=1}^{n^t} \beta_j u_j$.

Also let

$$\Psi_j(u_j) = \delta_{[\underline{u}_j, \bar{u}_j]}(u_j) + \frac{1}{2\lambda}|u_j - \hat{u}_j|^2$$

where δ is the indicator function and $u_j \in \Re$ represents the jth component of $u^t_{i,\tau}$.

Problem $(\mathcal{TRS})$ can then be written

$$\underset{(u,z)\in\Re^{n^t+1}}{\text{minimize}} \left\{ \Phi(z) + \sum_{j=1}^{n^t} \Psi_j(u_j) \right\}$$

$$\text{subject to: } z = \sum_{j=1}^{n^t} \beta_j u_j.$$

Recall that M is given as monotone increasing data and therefore ϕ^h is convex. Then clearly Φ as well as the Ψ_j are also convex.

As with the original hydropower problem, we are again confronted by a constrained convex minimization problem. By introducing the multiplier $y \in \Re$, we can (again) formulate the associated Lagrangian

$$L^t(u, z; y) = \Phi(z) + \sum_{j=1}^{n^t} \Psi_j(u_j) + y(z - \sum_{j=1}^{n^t} \beta_j u_j)$$

on $\Re^{n^t+1} \times \Re$. It is easy to see that $(\mathcal{TRS})$ satisfies the basic constraint qualification (Definition 2.4). Thus, by Theorem 2.5, since L^t is convex-concave in $(u, z), y$, the saddle point condition of Proposition 2.3 provides a necessary condition for primal and for dual optimality.

The associated dual problem is seen to be advantageous at this point. The dual problem can be formulated

$$\underset{y\in\Re}{\text{maximize}}\ \Theta(y) \qquad (\mathcal{DTRS})$$

$$\text{where: } \Theta(y) = \min_{(u,z)} \underbrace{\left\{ \Phi(z) + \sum_{j=1}^{n^t} \Psi_j(u_j) + y(z - \sum_{j=1}^{n^t} \beta_j u_j) \right\}}_{L^t(u,z;y)}.$$

Rewriting gives

$$\Theta(y) = -\sum_{j=1}^{n^t} \underbrace{\left\{ \max_{u_j} \{(y\beta_j)u_j - \Psi_j(u_j)\} \right\}}_{\Psi_j^*(\beta_j y)} - \underbrace{\max_{z} \{(-y)z - \Phi(z)\}}_{\Phi^*(-y)}$$

where Ψ_j^* and Φ^* are the Fenchel conjugate functions of Ψ_j and Φ.

The turbined release dual problem $(\mathcal{DTRS})$ has now been reduced to the following unconstrained convex (by Proposition 5.1(a)) minimization problem in *one* scalar variable:

$$\underset{y\in\Re}{\text{minimize}} \left\{ \sum_{j=1}^{n^t} \Psi_j^*(\beta_j y) \ + \ \Phi^*(-y) \right\}$$

which is to be solved for each subperiod of each $i \in \mathcal{I}$ in each iteration.

To minimize, we must find a solution of

$$0 \in \sum_{j=1}^{n^t} \beta_j \partial\Psi_j^*(\beta_j y) \ - \ \partial\Phi^*(-y).$$

The inclusion is solved for y using a secant method. Since Ψ_j is strictly convex and Φ is strictly convex on the crucial portions of its domain, the subgradients are both single-valued on the intervals of interest.

The primal variable u_j is easily recovered as a byproduct of the evaluation of $\partial\Psi_j^*$ in the secant method.

Construction of $\partial\Phi^*$ and $\partial\Psi_j^*$

Notice that

$$\Phi'(z) = \sum_{h=1}^{H} \phi^{h'}(z) = \sum_{h=1}^{H} M(d^h - z)$$

where $M(\xi)$ is tabulated as two columns (ξ and M) of data for each subperiod of each $i \in \mathcal{I}$. Then $\sum_{h=1}^{H} M(d^h - z)$ is constructed as the sum of shifted data and the function $\partial\Phi^* = [\Phi']^{-1}$ (by Proposition 5.1(b)) is then constructed by simply exchanging columns of the tabulation of Φ'.

Similarly, $\partial\Psi_j^* = [\partial\Psi_j]^{-1}$. Figure 5.1 shows the construction of $\partial\Psi_j^*$ (for the case when $\underline{u}_j < u_j < \overline{u}_j$ — the other cases are similar).

5.4 *The Test Problem*

The test problem for the dynamic splitting algorithm was constructed from data supplied by PG&E. In this problem, we wish to optimally schedule releases for two of PG&E's six main watershed power systems on a 24 month planning horizon beginning in January. The Yuba/Bear River basin was chosen because it is the largest basin (most controls) in the system. The Kings River was chosen because of its interesting configuration and because it contains a pump station (see Section 3.2.1) to pump water uphill from the second reservoir to the first. Optimal scheduling is defined as the determination of release policy so as to minimize the costs in supplying (one-third of) the power demand on the system.

The Physical Network

Recall from Section 3.2.1 how the network is expanded to account for peak and off-peak decision subperiods on weekdays and weekends. As a result, the test problem consists of 94 "storage" nodes arising from reservoirs and junctions (23 for Kings, 71 for Yuba) and 350 "release" arcs (101 for Kings, 249 for Yuba). Many of the nodes

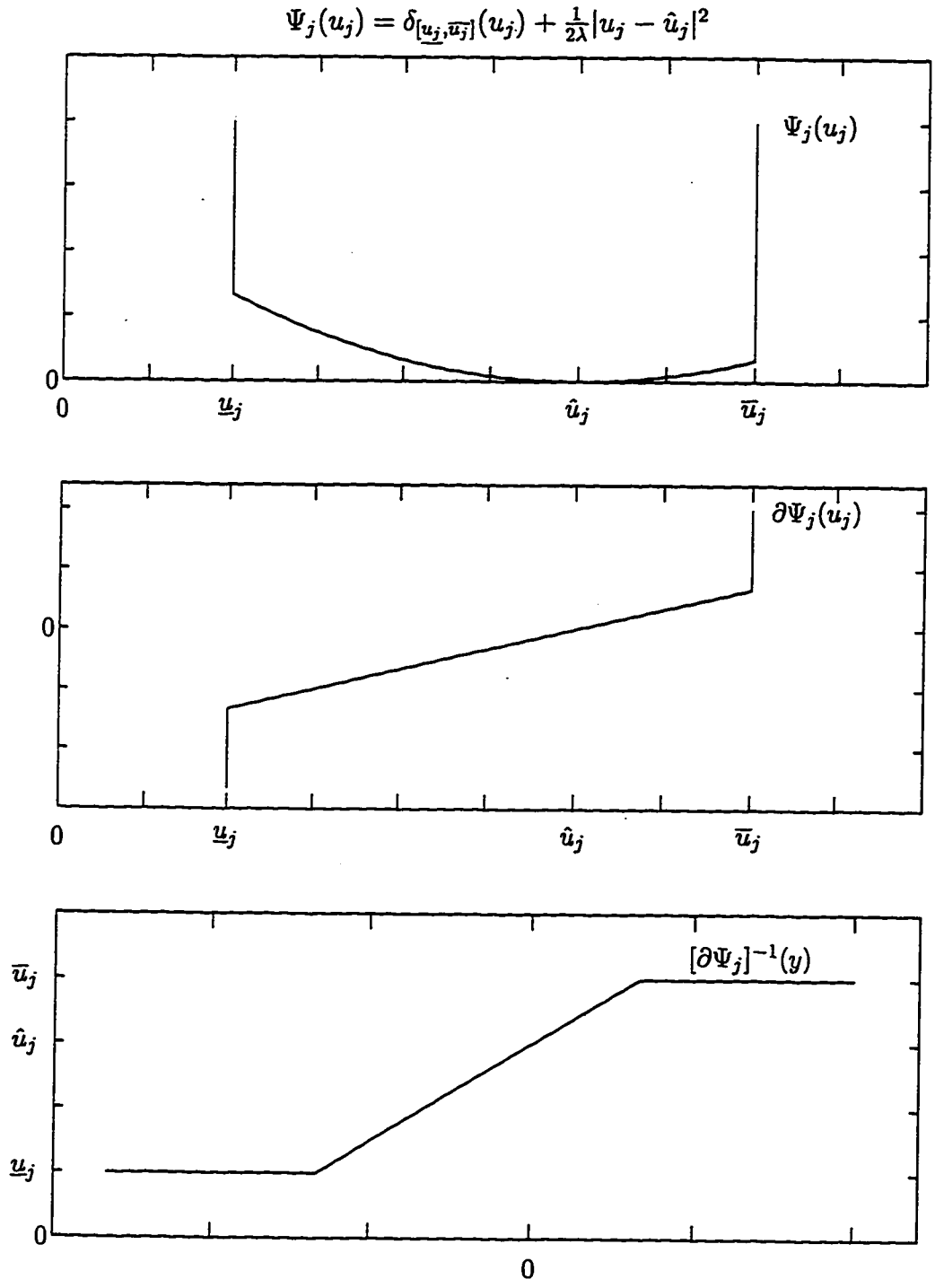

Figure 5.1: Construction of $\partial\Psi_j^* = [\partial\Psi_j]^{-1}$.

actually represent junctions or small reservoirs (see Figure 3.4) where the upper and lower bounds on storage are kept at zero. Special numerical consideration for these nodes will be examined in Section 5.5.

The Scenario Tree

The scenario tree supplied by PG&E has branches for high, medium, and low inflow in each of the first two months and then again at the beginning of the second *water* year. The snowpack which collects in the mountains in late spring is responsible for the inflows through summer and into late fall. Thus the lack of scenario branches through the middle months (as a problem simplification) is justified.

Inflow information is provided for each reservoir at each information/decision state and probabilities are given for the corresponding branches. The inflow data also accounts for evaporation; yielding negative inflows for some reservoirs in some months.

With the 350 release decisions to be made at each of 472 information/decision states, there are 165,200 decision variables in the test problem. Additionally, there are 44,368 state variables, 44,368 augmented variables and over 84,000 dual variables to track as well as 586,000 proximal terms introduced in the solution process. Each of these is to be determined at each iteration.

Other Data

Other problem data includes: powerhouse efficiencies (including a negative efficiency for the pump station); power demands ("load") for each sub-subperiod of each time stage; marginal cost data for each subperiod of each stage; and constraints on the release and storage of water for each subperiod of each time stage.

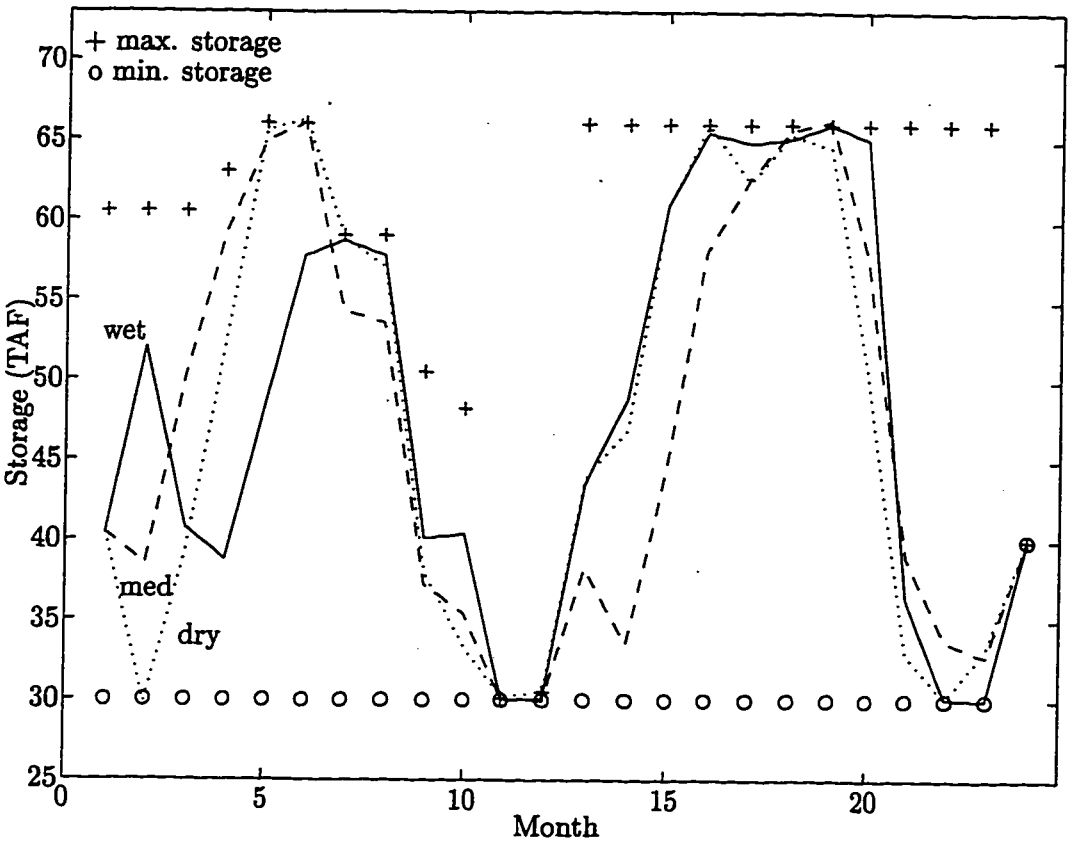

Figure 5.2: End-of-month storage (in thousand acre-feet) at a reservoir on the Yuba River watershed for 3 (of 27) scenarios.

5.4.1 Results

Some example results are shown in Figures 5.2 and 5.3. In Figure 5.2, the end-of-month storage level for a reservoir on the Yuba River is shown for three of the 27 scenarios. The scenario labeled "wet" follows the branches of the scenario tree corresponding to the highest inflow. The "dry" and "med" scenarios follow the lowest and the medium inflow branches of the scenario tree. Notice the terminal storage target at the end of the twenty-fourth month.

Similarly, Figure 5.3 depicts the optimal release of water through a powerhouse

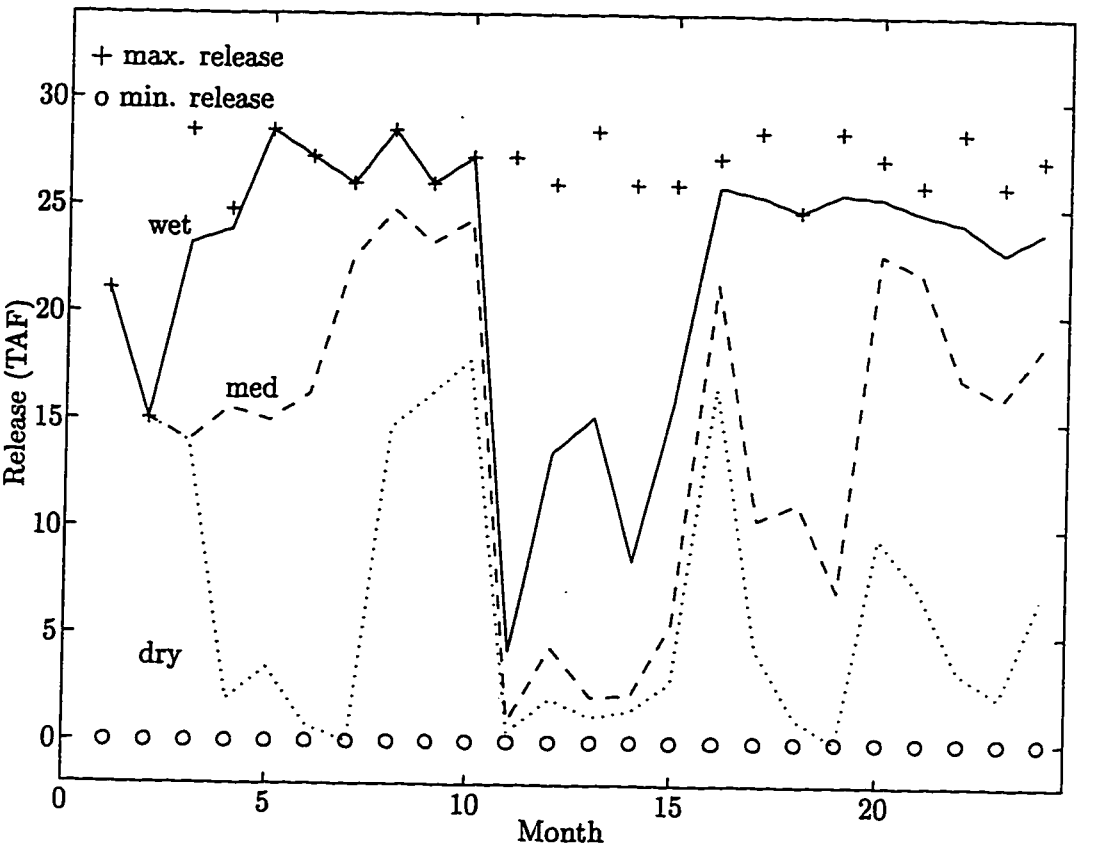

Figure 5.3: Monthly release (in thousand acre-feet) through a powerhouse on the Kings River watershed, during the weekday peak subperiod, for 3 (of 27) scenarios.

on the Kings River, during the weekday peak subperiod, for the same three scenarios.

5.5 *Implementation; Performance-Enhancing Modifications*

This section describes problem modifications made to enhance algorithm performance. These modifications were made in response to difficulties discovered during early testing of the algorithm.

The first of these modifications (described in Section 5.2) was the use of the *augmented* reduced Lagrangian rather than the "non-augmented" form used in the derivation of the algorithm. As mentioned previously, this allowed the terms associated with the state constraint to remain intact under the operator splitting; greatly improving state-constraint compliance.

"Zero Storage Nodes"

In section 3.2.3 it was explained that many of the "storage" nodes are actually junctions or small reservoirs where, in PG&E's model, no month-to-month storage is allowed. Thus the upper *and* lower bounds on many components of the x_i are zero. Compliance in this area was originally found to be lacking.

In response, we partitioned the dynamics

$$x_i = x_{i-} + Bu_i + b_i, \quad x_0 = b_0$$

so that components associated with the "zero storage nodes" could be written

$$B^0 u_i + b_i^0 = 0$$

where the rows of B^0 and components of b_i^0 correspond to the "zero storage" components of x_i.

Introducing a multiplier $v_i^0 \in \Re^{m_z}$ for the additional constraint, the augmented

reduced Lagrangian is reformulated, on $(U \times W) \times (\Re^{m_z} \times \Re^m)$,

$$L(u, w; v^0, v^1) =$$

$$\sum_{i \in \mathcal{I}} \pi_i \Big\{ g_i(u_i) + P^1 v_i^1 \cdot (x_i - w_i) + P^0 v_i^0 \cdot (B^0 u_i + b_i^0) + \delta_{U_i}(u_i) + \delta_{W_i}(w_i) \Big\}$$

$$\text{where: } x_i = x_{i-} + B u_i + b_i, \quad x_0 = b_0$$

where the full dynamics, not a partitioned dynamics, is used to preserve feasibility. Notice the inclusion of penalty parameters P^1 (a diagonal matrix) and P^0 to assist the multipliers. These penalties have the effect of "strengthening" the constraint.

Inclusion of the "zero storage" constraint (with penalty assistance) was seen to greatly influence constraint compliance at the "zero storage nodes." Because "zero storage" feasibility is quickly maintained (rather than slowly approached), control-variable convergence is also greatly enhanced.

Under the operator splitting, the new $P^0 v_i^0 \cdot (B^0 u_i + b_i^0)$ term is included in the dynamic subproblem ($\mathcal{P}1$) (of Section 5.2) resulting only in a change in the coefficients R and r_i in the reduced problem ($\mathcal{P}1'$). No increase in computational complexity results from the addition of this "zero storage" constraint.

Diagonalization of λ, μ and ν

The other improvement to the algorithm (as derived) is the replacement of the parameters λ, μ and ν, in the subproblems; each by a diagonal matrix of parameters (*e.g.* λ_j for $j = 1, \ldots, n$). This is equivalent to a change of norm and allows a much greater flexibility when tailoring the algorithm to a specific problem.

5.5.1 Parameter values

An important aspect of tailoring the algorithm to a particular problem is the choosing of parameters. In hindsight, it is clear that early testing of the algorithm suffered from parameter values that were orders of magnitude away from current "best" estimates.

Through numerical testing, we have found the following ranges of the parameters to be advantageous for this problem: For λ_j and ν_j, values in the 10^{-3} to 10^{-5} range; for μ^0, 1; for μ^1, .1; for P^0 , 10^5; for P^1, 180 – 220 for "storage-able nodes" and 1 for "zero storage nodes"; and for P^T (which replaces P^1 for terminal states), 1 – 5 times P^1.

We find that state constraint compliance at "storage-able" nodes can be very sensitive to the choice of P^1. If P^1 is chosen too small, a constraint can be badly violated; too large and control convergence can be compromised.

The parameter P^0 is very important for "zero storage" node compliance. A P^0 on the order of 10^5 can keep the storage values to very reasonable levels on the order of 10^{-7} or smaller. This also serves to speed convergence in all variables because, as mentioned previously, feasibility is quickly maintained at the "zero storage" nodes, rather than slowly approached.

Some components of u_i (and w_i) will converge more slowly than others. To mitigate this, the programmer should decrease the corresponding components of λ (and ν).

Further testing and refinement of parameter values would certainly prove useful, especially in the area of terminal storage compliance.

5.6 *Algorithm Performance*

There are various criteria for measuring algorithm performance including CPU times and solution accuracy measurements. A standard measure of accuracy of the solution is the *duality gap* which is reported along with the norm of constraint violations. This information can also provide useful stopping criteria for the algorithm. This section presents a formulation for the measure of the duality gap and performance results from the test problem.

5.6.1 *The Duality Gap; Formulation*

The *duality gap* is the gap between the primal and dual objective values at an iteration or at termination of the algorithm. Theorems 2.9 and 2.10 show that, in principle, at the optimal solution (if one exists) the primal and dual objective values should be equal (for our problem). But computationally we are solving a perturbed problem where the constraints are more "soft" than in the original formulation. Thus the duality gap should be calculated for a representation of this perturbed problem. For a measure of feasibility, the norm of the constraint violations will also be calculated.

A Primal Problem

Recall the modified form of the Lagrangian from Section 5.5.1

$$L(u,w;v^0,v^1) = \sum_{i\in\mathcal{I}} \pi_i \Big\{ g_i(u_i) + v_i^1 \cdot (x_i - w_i) + v_i^0 \cdot (B^0 u_i + b_i^0) + \delta_{U_i}(u_i) + \delta_{W_i}(w_i) \Big\}$$

$$\text{where: } x_i = x_{i-} + Bu_i + b_i, \quad x_0 = b_0$$

on $(U \times W) \times (\Re^{m_z} \times \Re^m)$ and where the penalty parameters P^0 and P^1 have been absorbed into the multipliers v_i^0 and v_i^1.

The primal problem can then be formulated

$$\underset{(u_i,w_i)\in U_i\times W_i}{\text{minimize}} \sum_{i\in\mathcal{I}} \pi_i \left\{ g_i(u_i) + \psi_i^* \begin{pmatrix} B^0 u_i + b_i^0 \\ x_i - w_i \end{pmatrix} \right\} \qquad (\mathcal{P})$$

$$\text{where: } x_i = x_{i-} + Bu_i + b_i, \quad x_0 = b_0,$$

$$\psi_i^*(q_i) = \begin{cases} 0, & \text{if } q_i = 0; \\ \infty, & \text{otherwise.} \end{cases}$$

Equivalently, the penalty function could be replaced by the pair of constraints

$$B^0 u_i + b_i^0 = 0, \quad x_i - w_i = 0.$$

A Perturbed Primal Problem

A perturbed primal problem can be formulated

$$\underset{(u_i,w_i)\in U_i\times W_i}{\text{minimize}} \sum_{i\in\mathcal{I}} \pi_i \left\{ g_i(u_i) + v_i^0 \cdot (B^0 u_i + b^0) + v_i^1 \cdot (x_i - w_i) \right\} \qquad (\mathcal{P}')$$

$$\text{where: } x_i = x_{i^-} + Bu_i + b_i, \quad x_0 = b_0$$

where v_i^0 and v_i^1 should be thought of as penalty parameters (instead of as multipliers).

Other perturbed primal problems can be formulated by similarly attaching penalty parameters to the the dynamics or to the box constraints; or by formulating any combination of constraints and penalties.

Problem $(\mathcal{P}')$ most closely resembles the problem being solved computationally at each iteration. Thus, the objective in $(\mathcal{P}')$ will be used in the calculation of the duality gap.

The norm of the violation of the box constraints $u_i \in U_i$ and of the state constraints $\underline{s}_i \leq x_i \leq \overline{s}_i$ will also be calculated.

A Dual Problem

From the Lagrangian L above, the dual problem can be formulated

$$\underset{(v_i^0,v_i^1)\in\Re^{m_z}\times\Re^m}{\text{maximize}} \sum_{i\in\mathcal{I}} \pi_i \left\{ b_i^0 \cdot v_i^0 - b_i \cdot y_i - \varphi_i^*(B^\top y_i - B^{0\top} v_i^0) - \rho_i^*(v_i^1) \right\} \qquad (\mathcal{D})$$

$$\begin{aligned} \text{where: } y_i &= -v_i^1, && i \in \mathcal{T} \\ y_i &= \textstyle\sum_{j\in\{i^+\}} p_j y_j \; - v_i^1, && i \notin \mathcal{T} \\ \varphi_i^*(\sigma_i) &= \sup_{u_i} \left\{ u_i \cdot \sigma_i - g_i(u_i) - \delta_{U_i}(u_i) \right\} \\ \rho_i^*(v_i^1) &= \sup_{w_i} \left\{ w_i \cdot v_i^1 - \delta_{W_i}(w_i) \right\}. \end{aligned}$$

A Perturbed Dual Problem

A problem dual to the perturbed primal problem ($\mathcal{P}'$) can be formulated

$$\underset{(v_i^0, v_i^1) \in \Re^{m_z} \times \Re^m}{\text{maximize}} \sum_{i \in \mathcal{I}} \pi_i \left\{ b_i^0 \cdot v_i^0 - b_i \cdot y_i - w_i \cdot v_i^1 - \left(u_i \cdot (B^\top y_i - B^{0\top} v_i^0) - g_i(u_i) \right) \right\} \quad (\mathcal{D}')$$

$$\text{where: } y_i = -v_i^1, \qquad i \in \mathcal{T}$$

$$y_i = \textstyle\sum_{j \in \{i^+\}} p_j y_j \quad - v_i^1, \quad i \notin \mathcal{T}$$

where w_i and u_i should be thought of as penalty parameters similar to v_i^0 and v_i^1 in the perturbed primal problem. The objective in ($\mathcal{D}'$) will be used in the calculation of the duality gap.

5.6.2 The Duality Gap; Results

Results of the duality gap measurements and the norm of constraint violations are presented in Figures 5.4 – 5.6. Figure 5.4 is a plot of the primal and dual objective values (from ($\mathcal{P}'$) and ($\mathcal{D}'$)) at each iteration as well as the true cost ($\sum_{i \in \mathcal{I}} \pi_i g_i(u_i)$). The duality gap is usually reported as a ratio of

$$\frac{\text{[Primal Objective]} - \text{[Dual Objective]}}{\text{[Primal Objective]}}$$

A plot of this ratio at each iteration is shown in Figure 5.5.

Figure 5.6(a) shows the norm of the (expected value of the) state-constraint violations. Figure 5.6(b) shows the norm of the $u_i \in U_i$ constraint violation. Recall that the solution of the dynamic subproblem is always feasible in the dynamics while the solution of the separable subproblem is always feasible in the box constraints. Recall also that the current solution at each iteration is the average of the solutions of the two subproblems at that iteration. Thus, Figure 5.6(b) also gives an idea of the violation of the dynamical constraint.

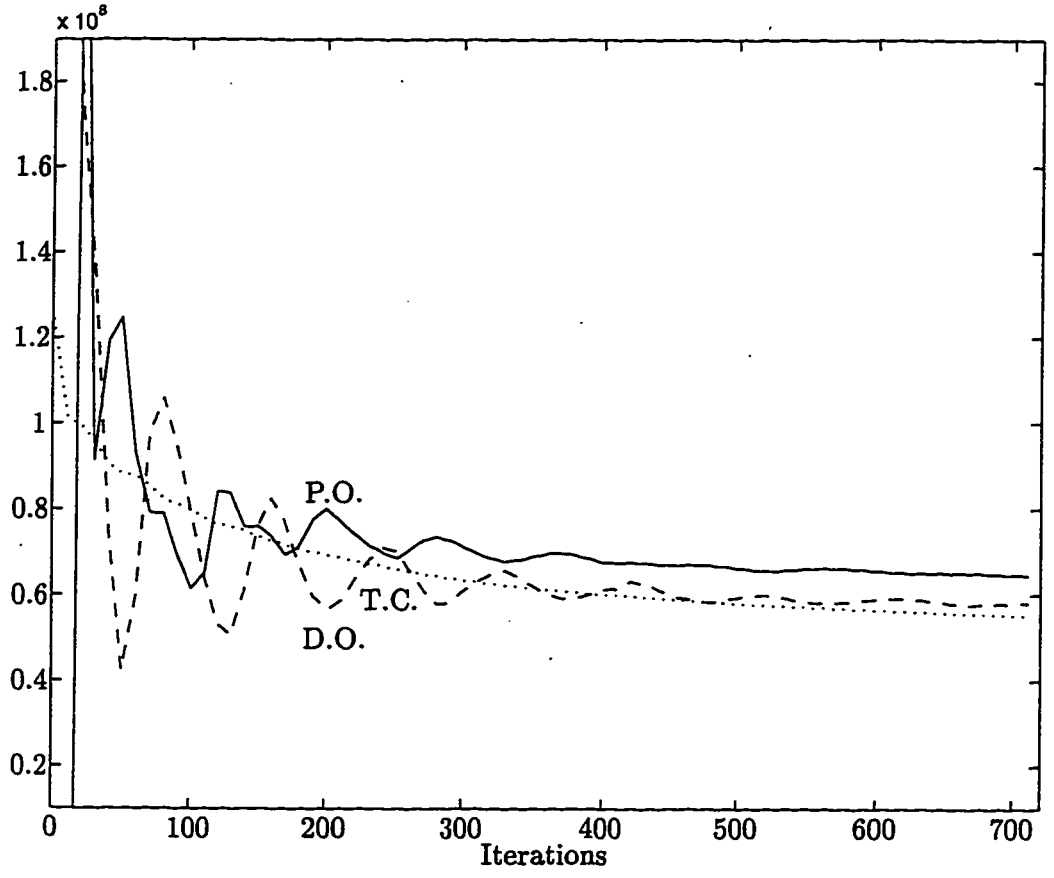

Figure 5.4: The value of the primal (P.O.) and dual (D.O.) objectives and the true cost (T.C.) (dotted line) at each iteration.

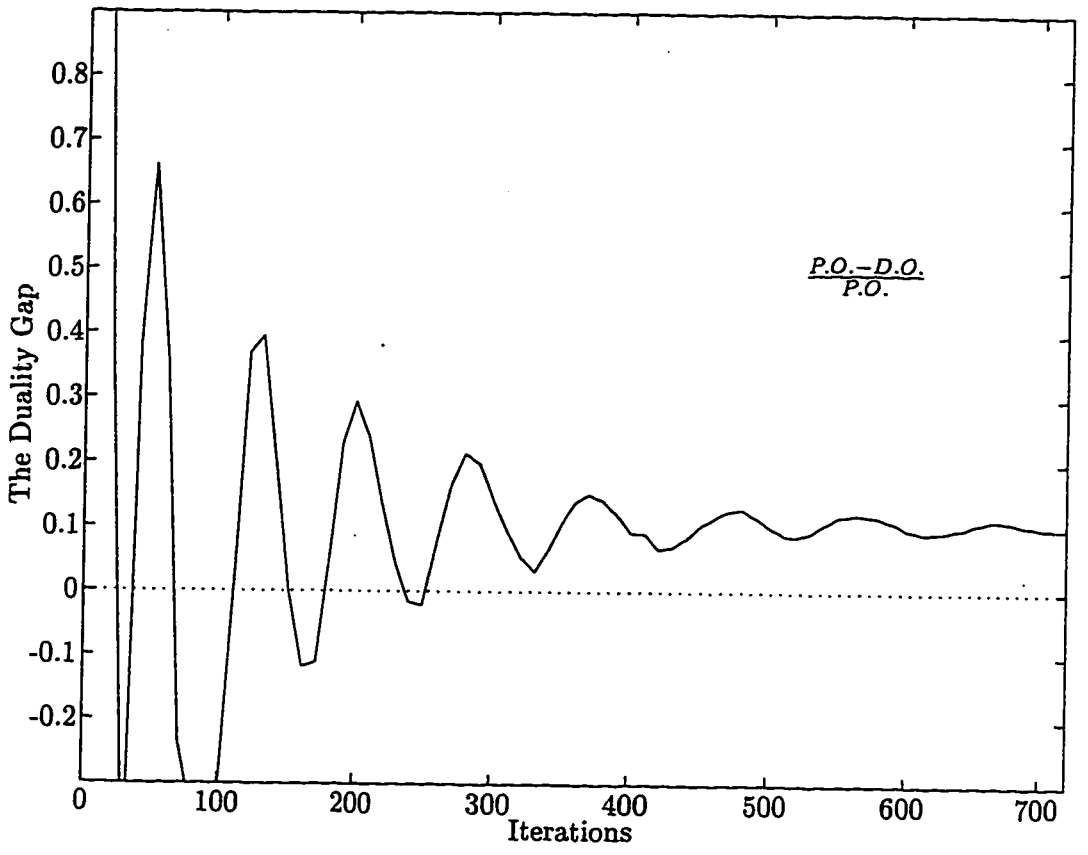

Figure 5.5: The Duality gap at each iteration.

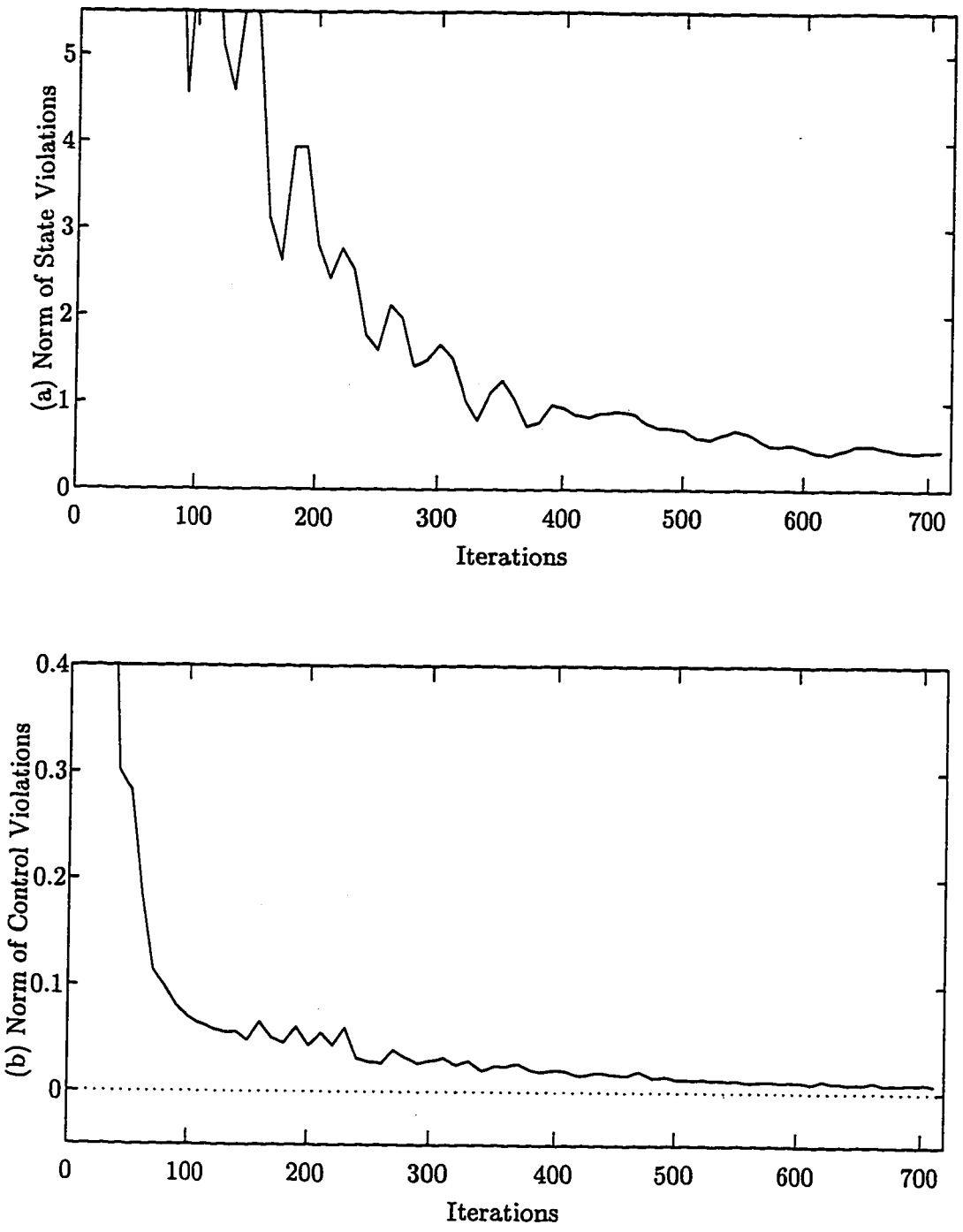

Figure 5.6: (a) The 2-norm of the state-constraint violation at each iteration.
(b) The 2-norm of the primal control-constraint violation at each iteration.

5.6.3 Stopping Criteria

The information in Figures 5.4 – 5.6 should be considered when choosing stopping criteria. Clearly, enough iterations should be used so that the true cost, the duality gap and the norm of constraint violations have "settled down." If at this point, the constraint violations are unacceptably large, then parameter adjustments are indicated rather than further iterations.

From an examination of the figures, it appears that about 450 iterations would be sufficient for this hydro scheduling test problem; though the constraint violations continue to slowly improve during later iterations.

In the hydro problem, it is important that the control constraint violation is small. The state constraints at "storage-able" nodes can be thought of as "soft" by comparison. In tailoring the algorithm to the hydro scheduling test problem, the parameters were chosen in such a way that the convergence of the control variables was placed above state constraint compliance.

5.6.4 CPU Time

The test problem and algorithm were prototyped in Matlab. Running on a DEC Alphastation 500/333 (333MHz EV5 processor), the algorithm used about one minute of CPU time per iteration; on a DEC Alphastation 3000/600 (175MHz EV4 processor), it used about 3.5 minutes of CPU time per iteration; and on an SGI Indigo 2 (150MHz R4400 processor), it needed about 3.2 minutes per iteration.

But, Matlab is notoriously slow for large problems with large loops. If the algorithm were to be implemented using a "faster" language and by taking advantage of the ample opportunity for parallelization, it seems certain that very reasonable run times would result.

5.7 *An Infeasible Test Problem*

This section compares the duality gap results of the previous section with results for an infeasible variation of the test problem (actually the original version of the test problem). Figure 5.7(a) depicts the end-of-month storage at the upper-most reservoir on the Yuba River for three (wet, medium and dry) of 27 scenarios. Notice that the low maximum storage constraints in the 18th and 19th months are badly violated in the dry and medium scenarios and that the storage in those scenarios falls short of the terminal (24th month) storage target. This combination of constraints in concert with the other system constraints presents an infeasible problem for the dryer scenarios. Figure 5.7(b) shows the end-of-month storage for a model where the maximum storage constraints for the 18th and 19th months have been increased (just enough to provide existence of a feasible solution). These low maximum storage constraints are meant to draw down the reservoir; most likely for flood control during the late-spring snow-melt. It seems reasonable to insist on a lesser draw-down during a year with low snow-pack.

Figure 5.8 compares the primal and dual objective values and true cost for the (a) infeasible and (b) feasible test problems. In (a), the primal and dual objectives continue to grow, even after the true cost, the norm of the state-constraint violations and the duality gap have "settled down" (see dashed lines in Figure 5.9(a) and (b)). This growth in the objectives is due to continued increase in the multipliers v_i^1 acting on these violated state constraints in the dryer scenarios.

Figure 5.9(a) and (b) also compares the state-constraint violation norm and the duality gap to those in the feasible test problem. Interestingly, the duality gap in the infeasible problem appears to be better (smaller) than the gap for the feasible problem. This is an artifact of the way the gap is calculated as a ratio of the difference between the primal and dual objectives divided by the primal objective. The difference continues to grow, but more slowly than the primal objective. Thus the

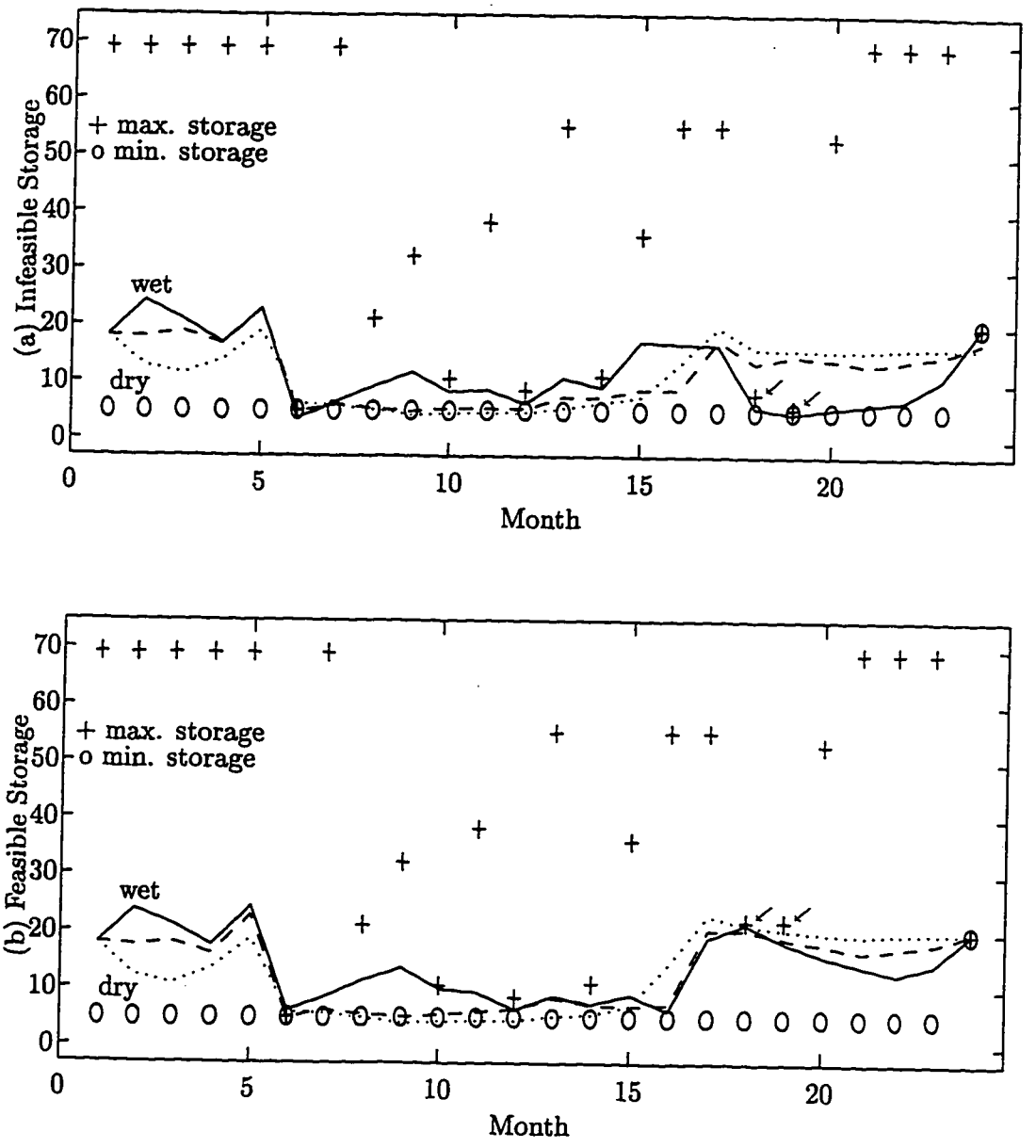

Figure 5.7: End-of-month storage at a reservoir on the Yuba River for the (a) infeasible and (b) feasible test problems. In (a), observe the violated maximum storage constraints in the dry and medium scenarios at the 18th and 19th months as well as the violated terminal storage targets in those scenarios.

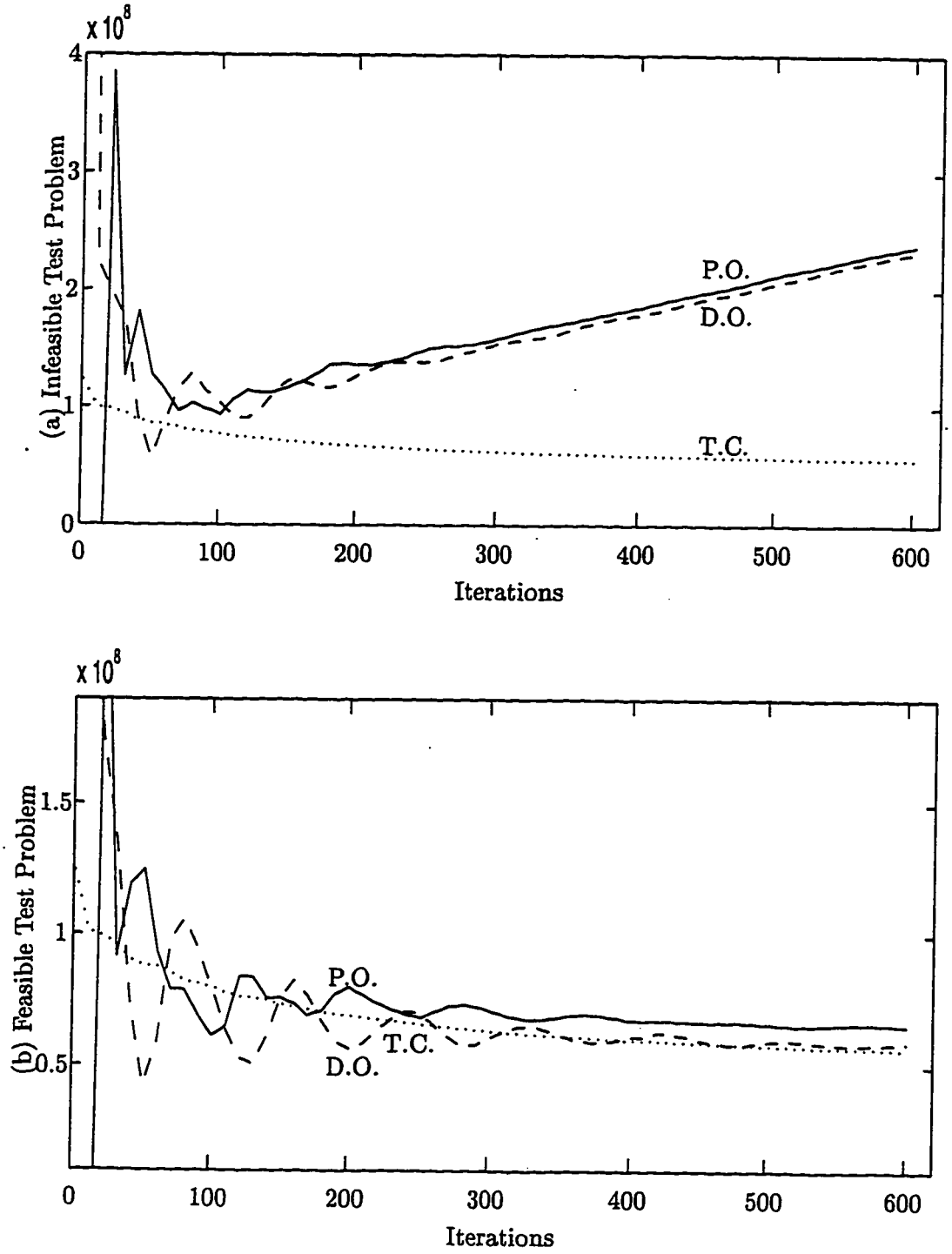

Figure 5.8: Primal and dual objective values and the true cost at each iteration for the (a) infeasible and (b) feasible test problems.

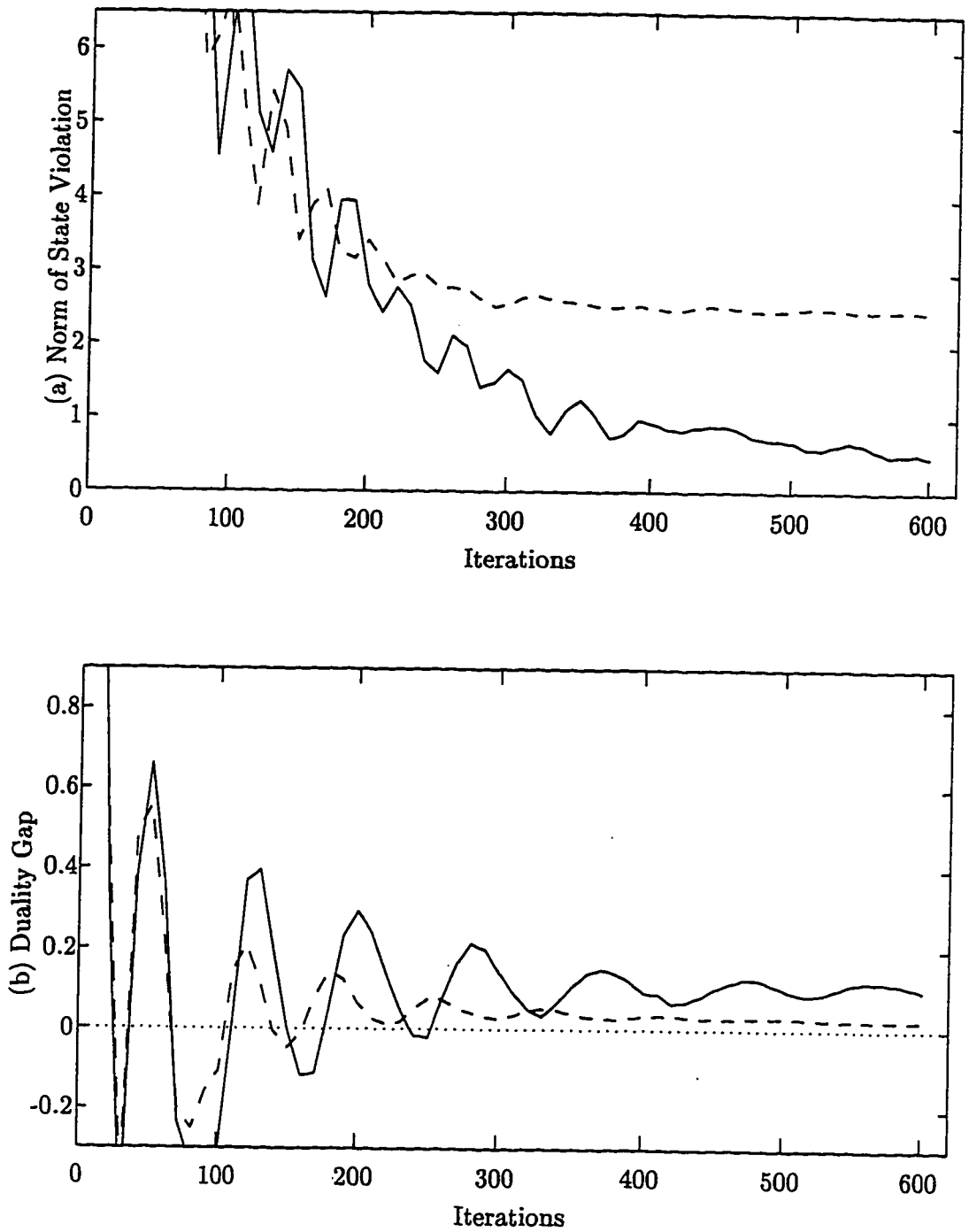

Figure 5.9: A comparison of (a) state-constraint violation and (b) duality gap for the infeasible (dashed line) and the feasible (solid line) test problems.

gap appears to decrease. This unbounded growth of the objectives provides a warning flag for problem infeasibility.

On close inspection of Figure 5.7(b), one can see less-significant storage-constraint violations in months 6 and 9. These are due to "small" infeasibilities that stem from low maximum storage in month 6 and from small (or negative) inflows (including evaporation) in the dry scenarios of the summer months (7,8,9). It is comforting that these small constraint violations (infeasibilities) do not have such a profound effect as the larger infeasibilities investigated above.

Chapter 6

CONCLUSIONS

In this paper we have presented a new algorithm for the convex multistage stochastic programming problem (MSP); one that is appropriate for the large scale *nonlinear* (convex) problem (*e.g.* hydro scheduling) and is easily parallelizable. Theorems providing optimality conditions for the MSP and theorems guaranteeing linear convergence of the algorithm were also presented as was a specialization of the algorithm for the hydro scheduling problem. A test problem and results were discussed and algorithm performance was investigated.

The Algorithm

Recent work in algorithm development for the stochastic hydro scheduling problem has focused on Benders decomposition type methods. These have proven successful for the linear or piecewise linear problem. But the costs involved in the hydro scheduling problem are known to be nonlinear. Thus, the (*dynamic splitting*) algorithm was developed with the nonlinear (convex) problem in mind.

The algorithm is based on the application of Spingarn's (1983, 1985) operator splitting method to the saddle point problem associated with the multistage stochastic programming problem. Spingarn's method imposes a decomposability which results in two main subproblems to be solved at each iteration. The crucial structure of the subproblems is revealed only in the splitting of the saddle point problem formulation.

The *dynamic subproblem* is reformulated as an unconstrained linear-quadratic dynamic programming problem. It is solved via a linear feedback loop (extended to the scenario tree setting). In the hydro setting, this problem is separable by

watershed.

The *separable subproblem* results in separate sub-subproblems for the augmented, dual and primal variables for each information/decision state $i \in \mathcal{I}$ in the scenario tree. In the hydro setting, the primal variable problems are further separable by spilled and turbined release. In the specialization to the hydro problem, a dual problem formulation for the turbined release problem is solved utilizing properties of the Fenchel conjugate function. The other sub-subproblems are solved easily in closed form.

This separability of the primal variable problem points out a particularly attractive feature of the algorithm. The algorithm creates separability in competing objectives (*e.g.* competing water users). If, for example, terms were added to the objective for the optimal spillage of water for fish survival (or irrigation or recreation), separate subproblems would result for each user. Of course these other objectives would need to be properly weighted against the hydropower cost objective.

At each iteration, the solution to the dynamic subproblem is feasible in the problem dynamics. Penalties (multipliers) in the objective push the subproblem toward state-variable feasibility. In contrast, the solution to the separable subproblem is always feasible in the control constraints. On convergence of the algorithm, the subproblem solutions will, in principle, be equal and so the solution will be feasible in all respects.

The duality gap results of Section 5.6 provide a measure of performance and a criterion for stopping. They also provide a warning flag for problem infeasibility.

As Discussed in Section 5.5.1, parameter choices are important to the tailoring of the algorithm for a specific problem. As this can take experimentation, the algorithm might be most useful in a situation where it would get repeated use. Hydropower scheduling is a good example. PG&E runs their model every two weeks and the results are discussed in biweekly planning sessions.

Further Work

Further work on the algorithm should include improvements in the handling of infeasible state constraints, especially the terminal storage targets. When these targets are removed or reduced, algorithm performance is improved.

Further work on the choosing of parameters would make future application easier and more attractive. Theoretical work on the relationship between the parameter λ (also μ and ν) and the linear convergence rate constant $1/\sqrt{1+(1/\alpha)^2}$ (introduced in Theorem 4.11) might prove fruitful in this respect.

BIBLIOGRAPHY

[1] L. Alercon and D. Marks. A stochastic dynamic programming model for the operation of the high aswan dam. *Tech. Rep. 246, Ralph M. Parsons Lab., Dept. of Civ. Eng., MIT*, 1979.

[2] T. W. Archibald, C. S. Buchanon, K. I. M. McKinnon, and L. C. Thomas. Nested benders decomposition and dynamic programming for reservoir optimization. *Working paper 96/2, Dept. of Business Studies, Univ. of Edinburgh*, 1996.

[3] A. J. Askew. Optimum reservoir operating policies and the importance of a reliability constraint. *Water Resour. Res.*, 10(1):51–56, 1974a.

[4] A. J. Askew. Chance-constrained dynamic programming and the optimization of water resource systems. *Water Resour. Res.*, 10(6):1089–1106, 1974b.

[5] R. Bellman. *Dynamic Programming.* Princeton Univ. Press, Princeton N.J., 1957.

[6] J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numer. Math.*, 4:238–252, 1962.

[7] Dimitri P. Bertsekas. *Constrained optimization and Lagrange multiplier methods.* Academic Press inc., New York, 1982.

[8] Dimitri P. Bertsekas. *Dynamic Programming.* Prentice-Hall, inc., Englewood Cliffs, N.J., 1987.

[9] Dimitri P. Bertsekas and S.E. Shreve. *Stochastic optimal control: the discrete time case.* Academic Press inc., New York, 1978.

[10] John R. Birge. Solution methods for stochastic dynamic linear programs. Technical report, Report 80/29, Systems Optimization Lab.,Dept of Operations Research, Stanford University, CA, 1980.

[11] John R. Birge. Decomposition and partitioning methods for multistage stochastic linear programs. *Operations Research*, 33(5):989–1007, 1985.

[12] R. L. Bras, R. B. Buchanan, and K. C. Curry. Real time adaptive closed loop control of reservoirs with the high aswan dam as a case study. *Water Resour. Res.*, 19(1):33–52, 1983.

[13] Chara and Pant. Optimal discharge policy of multi-reservoir linked system using successive variations. *Int. J. Systems Sci.*, 15(1):31–52, 1984.

[14] George H.-G. Chen and R. Tyrrell Rockafellar. Convergence rates in forward-backward splitting. *Siam J. Optimization*, 7:421–444, 1997.

[15] J. Douglas and H. H. Rachford. On the numerical solution of heat conduction problems in two and three space variables. *Transactions of the American Math. Society*, 82:421–439, 1956.

[16] J. Dupacova. Water resources system modeling using stochastic programming techniques. In P. Kall and A. Precopa, editors, *Recent Results in Stochastic Programming.* Springer-Verlag, New York, 1980.

[17] Jonathan Eckstein. *Splitting methods for monotone operators with applications to parallel optimization.* PhD thesis, Massachusetts Institute of Technology, 1989.

[18] Joel H. Ferziger. *Numerical methods for engineering and application.* John Wiley & sons, inc., New York, 1981.

[19] Efi Foufoula-Georgiou and Peter K. Kitandis. Gradient dynamic programming for stochastic optimal control of multidimensional water resources systems. *Water Resour. Res.*, 24(8):1345–1358, 1988.

[20] S. Gal. Optimal of stochastic multireservoir waster supply system. *Water Resour. Res.*, 15(4):737–749, 1979.

[21] S. Gal. Parameter iteration dynamic programming. *Management Science*, July, 1989.

[22] B. G. Gorenstin, N. M. Campodonico, J. P. Costa, and M. V. F. Pereira. Stochastic optimization of a hydro-thermal system including network constraints. *IEEE Transactions on Power Apparatus and Systems*, 7(2):791–797, 1992.

[23] R. A. Howard. *Dynamic Programming and Markov Processes.* MIT Press, Cambridge, Mass, 1960.

[24] W.-C. Huang, R. Harboe, and J. J. Bogardi. Testing stochastic dynamic programming models on observed or forecasted inflows. *J. of Water Resources Planning and Management*, 117(1):28–36, 1991.

[25] Yoshiro Ikura and George Gross. Efficient large-scale hydro system scheduling with forced spill conditions. *IEEE Transactions on Power Apparatus and Systems*, PAS-103(12):3502–3520, 1984.

[26] J. J. Jacobs, G. Freeman, J. Crygier, D. Morton, G. L. Schultz, K. Staschus, and J. R. Stedinger. Socrates — a system for scheduling hydroelectric generation under uncertainty. *Annals of Operations Research*, 59:99–133, 1995.

[27] S. A. Johnson, J. R. Stedinger, and C. A. Shoemaker. Computational improvements in dynamic programming. *Forefronts*, 4(7):3–7, 1988.

[28] S. A. Johnson, J. R. Stedinger, C. A. Shoemaker, Y. Li, and J.A. Tejada-Guibert. A numerical solution of continuous-state dynamic programs using linear and spline interpolation. *Operations Research*, 41(3):484–500, 1993.

[29] Jerson Kelman, Jery R. Stedinger, Lisa A. Cooper, Eric Hsu, and Sun-Quan Yuan. Sampling stochastic programming applied to reservoir operation. *Water Resour. Res.*, 26(3):447–454, 1990.

[30] P. L. Lions and B. Mercier. Splitting algorithms for the sum of two nonlinear operators. *SIAM J. Numer Anal.*, 16(6):964–979, 1979.

[31] J. D. C. Little. The use of storage water in a hydroelectric system. *Oper. Res*, 3:187–197, 1955.

[32] D. R. Maidment and V. T. Chow. Stochastic state variable dynamic programming for reservoir systems analysis. *Water Resour. Res.*, 17(6):1578–1584, 1981.

[33] A. S. Manne. Product mix alternatives: flood control electric power and irrigation. *Int. Econ. Rev*, 8(1):30–52, 1962.

[34] P. B. D. Masse. *Les Reserves et la Gerulation de l'avenir dans la vie economique.* Hermann and Cie, Paris, 1946 in French.

[35] G. J. Minty. Monotone (nonlinear) operators on hilbert space. *Duke Math. J.*, 29:243–247, 1962.

[36] N. R. Mizyed, J. C. Loftis, and D. G. Fontane. Operation of large multireservoir systems using optimal-control theory. *J. of Water Resour. Planning and Mgmt.*, 118(4):371–387, 1992.

[37] M. Papageorgiou. Optimal multireservoir network control by the discrete maximum principle. *Water Resour. Res.*, 21(12):1824–1830, 1985.

[38] Gregory B. Passty. Ergodic convergence to a zero of the sum of monotone operators in hilbert space. *J. of Math. Anal. and Appl.*, 72:383–390, 1979.

[39] M. V. F. Pereira and L. M. V. G. Pinto. Stochastic optimization of a multireservoir hydroelectric power system. *Water Resour. Res.*, 21(6):779–792, 1985.

[40] M. V. F. Pereira and L. M. V. G. Pinto. Stochastic dual dynamic programming. *Mathematical Programming*, 52:359–375, 1991.

[41] K. Reznicek and T. Cheng. Stochastic modeling of reservoir operations. *Euro. J. of Oper. Res.*, 50:235–248, 1991.

[42] R. Tyrrell Rockafellar. Characterization of subdifferentials of convex functions. *Pacific Journal of Mathematics*, 17(3):497–510, 1966.

[43] R. Tyrrell Rockafellar. *Convex Analysis.* Princeton University Press, Princeton N.J., 1970a.

[44] R. Tyrrell Rockafellar. Monotone operators associated with saddle-functions and minimax problem. In F.E. Browder, editor, *Proceedings of Symposia in Pure Mathematics*, volume 18(1), pages 241–250. American Mathematical Society, 1970b.

[45] R. Tyrrell Rockafellar. Monotone operators and the proximal point algorithm. *SIAM J. Control And Optimization*, 14(5):877–898, 1976a.

[46] R. Tyrrell Rockafellar. Augmented lagrangians and applications of the proximal point algorithm. *Math. Oper. Res.*, 1:97–116, 1976b.

[47] R. Tyrrell Rockafellar. *Network flows and monotropic optimization.* Wiley, New York, 1984.

[48] R. Tyrrell Rockafellar. Extended linear-quadratic programming. *SIAM J. Optimization, News and Views*, 1:3–6, 1992.

[49] R. Tyrrell Rockafellar. Modeling structure and decomposition in multistage stochastic programming. West Coast Optimization Meeting; Seattle, 1993a.

[50] R. Tyrrell Rockafellar. Lagrange multipliers and optimality. *SIAM Review*, 35:183–236, 1993b.

[51] R. Tyrrell Rockafellar and R. J.-B. Wets. A lagrangian finite generation technique for solving linear-quadratic problems in stochastic programming. *Math. Programming Study*, 28:63–93, 1986.

[52] R. Tyrrell Rockafellar and R. J.-B. Wets. Scenarios and policy aggregation in optimization under uncertainty. *Math. of Oper. Res.*, 16(1):119–147, 1991.

[53] Gary L. Schultz. Systems Engineering/Operations Research at The Pacific Gas and Electric Co., San Francisco, CA., Private communication, 1995, 1996.

[54] Pamela Shaw. Columbia Basin Research, Seattle, WA., Private communication, 1997.

[55] S.A. Soliman and G.S. Christesen. Application of functional analysis to optimization of a variable head multireservoir power system for long-term regulation. *Water Resour. Res.*, 22(6):852–858, 1986.

[56] Jonathan E. Spingarn. Partial inverse of a monotone operator. *Appl. Math. Optim.*, 10:247–265, 1983.

[57] Jonathan E. Spingarn. Applications of the method of partial inverses to convex programming: decomposition. *Mathematical Programming*, 32:199–223, 1985.

[58] Jery R. Stedinger, Bola F. Sule, and Daniel P. Louks. Stochastic dynamic programming models for reservoir operation optimization. *Water Resour. Res.*, 20(11):1499–1505, 1984.

[59] S. Y. Su and R. A. Deininger. Modeling the regulation of lake superior under uncertainty of future water supplies. *Water Resour. Res.*, 10(1):11–25, 1974.

[60] Andre Turgeon. Optimal operation of multireservoir power systems with stochastic inflows. *Water Resour. Res.*, 16(2):275–283, 1980.

[61] Andre Turgeon. A decomposition method for the long-term scheduling of reservoirs in series. *Water Resour. Res.*, 17(6):1565–1570, 1981a.

[62] Andre Turgeon. Optimal short-term hydro scheduling from the principle of progressive optimality. *Water Resour. Res.*, 17(3):481–486, 1981b.

[63] Juan B. Valdes, Jenny Montbrun-Di Filipo, Keneth M. Strzpek, and Pedro J. Restreppo. Aggregation-disaggregation approach to multireservoir operation. *J. of Water Resour. Plan. and Mgmt*, 118(4):423–444, 1992.

[64] R. J.-B. Wets. Large scale linear programming techniques. In Y. Ermoliev and R. Wets, editors, *Numerical Results for Stochastic Optimization*, chapter 3. Springer, Berlin, 1988.

[65] Sidney Yakowitz. Dynamic programming applications in water resources. *Water Resour. Res.*, 18(4):673–696, 1982.

[66] William W-G. Yeh. Reservoir management and operations models: a state of the art review. *Water Resour. Res.*, 21(12):1797–1818, 1985.

[67] G. K. Young. Finding reservoir operating rules. *J. Hydraul. Div. Am. Soc. Civ Eng.*, 93(HY6):297–321, 1976.

VITA

Name: David H. Salinger

Date of Birth: 29 August, 1961

Education:

The University of Washington, Seattle

Ph.D. in Applied Mathematics, 1997.

The Pennsylvania State University, State College

M.S. in Mathematics, 1987.

The State University of New York at Oneonta

B.A. in Geology and Mathematics (dual), 1984.