Nested Benders Decomposition and Dynamic Programming for Reservoir Optimization

T.W. Archibald, C.S. Buchanan
[†] K.I.M. McKinnon, L.C. Thomas

October 1996

MS 96-001

[†] Department of Mathematics and Statistics
Department of Business Studies
Report also appears as Dept of Business Studies Working paper 96/2
Supported by EPSRC research grant GRH43700

Department of Mathematics and Statistics
University of Edinburgh, The King's Buildings, Edinburgh EH9 3JZ
Tel. (33) 131 650 5042 E-Mail : ken@maths.ed.ac.uk

Nested Benders Decomposition and Dynamic Programming for Reservoir Optimization

T.W. Archibald, C.S. Buchanan
† K.I.M. McKinnon, L.C. Thomas
Dept of Business Studies, † Dept of Mathematics and Statistics,
University of Edinburgh EH9 3JZ, UK

October 1996

Abstract

This paper presents a computational comparison of nested Benders decomposition and dynamic programming (DP) for stochastic optimization problems arising from the optimization of hydro-electric generation from hydraulically linked reservoirs. The following assumptions are made: the amount of electricity produced by a generator is a concave piecewise linear function of the volume taken from the feeder reservoir and is independent of the volume in the feeder reservoir; the reward for generation is a concave piecewise linear function of total generation; the final value is a separable concave piecewise linear function of residual volume. These assumptions allow the problem to be formulated as a linear programming problem. The examples considered have between 3 and 17 reservoirs, two weather states, three runoff patterns and five periods. The examples are solved exactly by the simplex method and nested Benders decomposition and solved approximately by discrete DP. A full version of DP is used for the 3 and 4 reservoir cases, and a decomposition method is used for all examples. The full DP results are within 1% of the optimal and the DP decomposition results are within 3.2% of the optimal. Timings are given for serial and parallel versions of the algorithms. An analysis is given of how the different methods scale with the number of periods, reservoirs, weather states and runoff patterns, and also how applicable they are to more general problems.

Key words: Dynamic programming, Benders decomposition, stochastic optimization, hydro-electric, reservoir operation.

1 Introduction

This paper compares the performance of 4 computational techniques (the revised simplex method, nested Benders decomposition, full dynamic programming and a dynamic programming decomposition) applied to stochastic optimization problems which model reservoir optimization. A core set of examples which can be solved by all of these methods are presented and the methods are compared in terms of the accuracy of their solutions; the type of problems for which they are efficient; and their serial and parallel solution times. Surveys of LP and DP models of reservoir operation can be found in [9, 12]. These papers do not give comparisons of the computational effectiveness of LP and DP approaches on the same class of problems, which is what we do here, though [3] presents such a comparison for LP and stochastic quasigradient methods.

Recently stochastic optimization problems have been solved using nested Benders decomposition, importance sampling and parallel processing. Dantzig and Glynn [2] show that these techniques could be combined to solve a class of multistage problems, while the ideas of aggregating cuts in Benders decomposition using scenario sampling is explained in [4] for a number of stochastic problems. A method of scheduling hydro-electric generation using Benders decomposition with fast starts is explained in [5]. Since we wish to compare the accuracy of state space approximations in DP and DP decomposition models, we need to obtain the exact answer to the reservoir problem by LP methods. For this reason we do not use scenario sampling in this paper, though the LP approaches use both parallel processing and Benders decomposition. The LP and DP methods outlined here can be combined with scenario sampling as described in [8] for LP and [6] for DP.

The reservoir models considered in this paper include an underlying weather state that allows the persistence of drought or flood conditions to be modelled by a Markov chain. While this feature is not uncommon in DP models [10], previously it has not been explicitly dealt with in LP models.

The problem of maximizing the expected income from the use of water from a network of reservoirs when future weather conditions and demands are uncertain can be formulated as a stochastic optimization problem. The value of the water may arise from use in agriculture, water supply or hydro-electric generation. In this paper we shall use examples where the value is due to hydro-electric generation, but the techniques used here can be applied to the other situations. There are two approaches to modelling the uncertainty in runoff. In this paper we assume that the runoff for a period is known before the release decisions for that period are made. The alternative, which is followed by [7, 10], assumes the action is taken before the runoff for the current period is known. After the runoff is known an adjustment has then to be made to allow for the actual runoff. In practice the weather state will change continuously within the period chosen for the optimization, so neither approach is accurate. The alternative approach is more conservative in its decisions than the one we adopt. As the length of the periods is reduced, both methods become arbitrarily accurate.

The following notation will be used.

t – index for time periods.
i – index of runoff vectors.
I^t – set of the indices of runoff vectors at period t.
s – index of underlying weather states.
S^t – set of the indices of underlying weather states at period t.
s, i – index of weather state which includes the underlying weather state and the run-off vector.
$\boldsymbol{h}$ – vector of volumes stored in reservoirs (water state).
$\underline{\boldsymbol{h}}, \overline{\boldsymbol{h}}$ – lower and upper bounds on vector $\boldsymbol{h}$.
$\boldsymbol{x}$ – vector of flows (or releases) between reservoirs. Some of these flows pass through generators, some are spills or controlled releases and some represent flooding. These are the actions in the problem.
$\underline{\boldsymbol{x}}, \overline{\boldsymbol{x}}$ – lower and upper bounds on vector $\boldsymbol{x}$.

M	–	node-arc incidence matrix. $M_{r,f} = 1$ if flow f enters reservoir r, $M_{r,f} = -1$ if flow f leaves reservoir r, and $M_{r,f} = 0$ otherwise.
$\boldsymbol{q}_i^t$	–	vector of runoffs into the reservoirs in period t when runoff index is i.
$r^t(\boldsymbol{h}, \boldsymbol{x})$	–	income from generation in period t when water state is $\boldsymbol{h}$ and flow vector is $\boldsymbol{x}$. (In the examples used in this paper the income from generation depends only on the period so r is indexed by t alone. However the methods and analysis given also apply to the case where the income is also a function of weather state, s, i.)
$p_{i\|s}^t$	–	probability of runoff with index i occurring in period t when in underlying state s.
$P_{\tilde{s}\|s,i}^t$	–	probability of a transition from weather state s, i in period t to underlying weather state $\tilde{s}$ in period $t+1$.
$v_s^t(\boldsymbol{h})$	–	expected future value at the start of period t when in underlying weather state s and water state $\boldsymbol{h}$ (before the runoff is known).
$V_{s,i}^t(\boldsymbol{h})$	–	expected future value in period t when in weather state s, i and water state $\boldsymbol{h}$ (after the runoff is known).
$\gamma_{s,i}^t(\boldsymbol{h})$	–	expected future value at the end of period t when in weather state s, i and water state $\boldsymbol{h}$ (after the action has been taken but before it is known what underlying weather state occurs at the start of period $t+1$).
w	–	scene in the scenario tree. Each scene has an associated weather state s_w, i_w.
W^t	–	set of all scenes in the scenario tree at period t.
$w \to \tilde{w}$	–	transition from scene w to scene $\tilde{w}$.
C^t	–	set of possible scene transitions between periods t and $t+1$. $C^t = \{w \to \tilde{w} \| w \in W^t\}$.
π_w	–	probability of scene w occurring.

The optimization problem can be expressed by the dynamic programming (DP) recurrences (1), (2) and (3). The state of the system at a time immediately prior to making the release decision is specified by the triple $(s, i, \boldsymbol{h})$, where s is the underlying weather state which contains all the information from the past and any forecasts which may be available about the future weather, i is the index of the runoff vector for the current period and $\boldsymbol{h}$ is the vector of reservoir volumes. The DP recurrences below define the functions $v_s^t(\boldsymbol{h})$, $V_{s,i}^t(\boldsymbol{h})$ and $\gamma_{s,i}^t(\boldsymbol{h})$ recursively.

$$\begin{aligned}
v_s^t(\boldsymbol{h}) &= \sum_{i \in I^t} p_{i|s}^t V_{s,i}^t(\boldsymbol{h}); && (1)\\
V_{s,i}^t(\tilde{\boldsymbol{h}}) &= \max_{\boldsymbol{x},\boldsymbol{h}} (r^t(\tilde{\boldsymbol{h}}, \boldsymbol{x}) + \gamma_{s,i}^t(\boldsymbol{h})) && (2)\\
&\quad \text{subject to } \boldsymbol{h} - M\boldsymbol{x} = \tilde{\boldsymbol{h}} + \boldsymbol{q}_i^t, \\
&\quad \underline{\boldsymbol{x}} \le \boldsymbol{x} \le \overline{\boldsymbol{x}}, \underline{\boldsymbol{h}} \le \boldsymbol{h} \le \overline{\boldsymbol{h}}; \\
\gamma_{s,i}^t(\boldsymbol{h}) &= \sum_{\tilde{s} \in S^{t+1}} P_{\tilde{s}|s,i}^t v_{\tilde{s}}^{t+1}(\boldsymbol{h}). && (3)
\end{aligned}$$

Figure 1 shows the state transitions for the weather states used in the examples in this study. Here $S^t = \{\text{Dry,Wet}\}$ and $I^t = \{\text{Hi,Med,Lo}\}$.

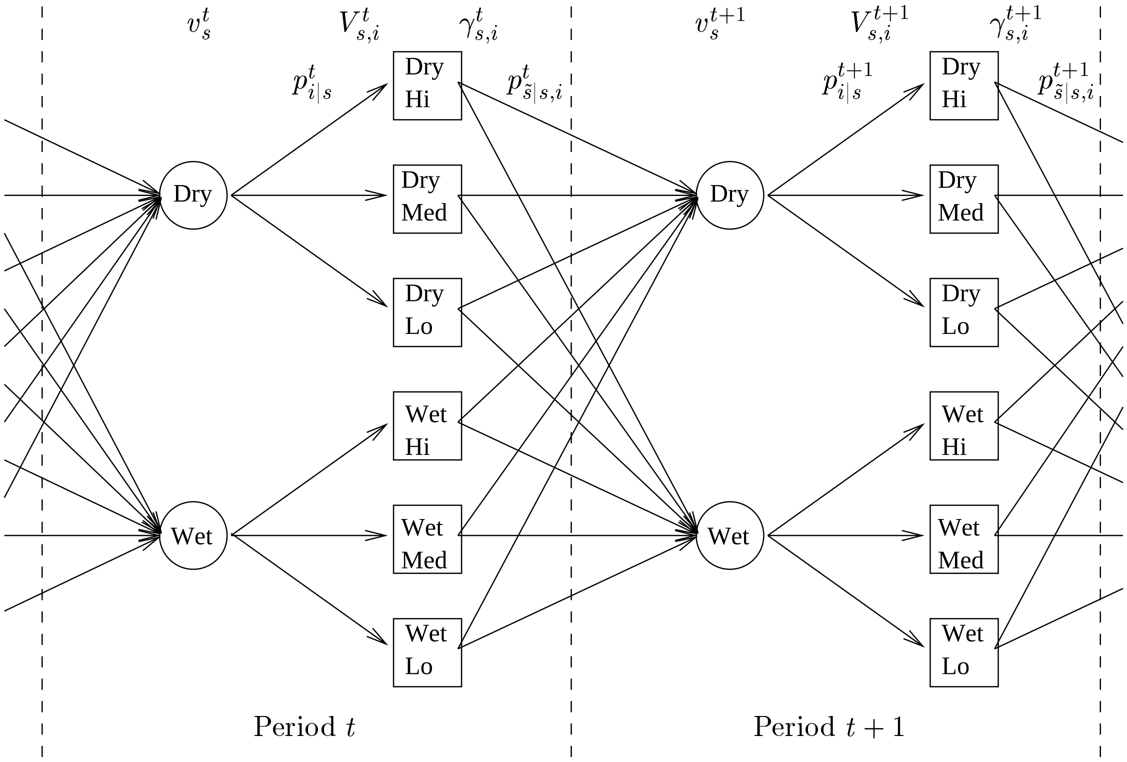

Figure 1: State transitions

A solution to the problem as formulated in (1) to (3) requires the value functions to be calculated for every period and for all weather and water states. In practice however we are normally interested in the solution of this problem for a single initial weather and water state. To specify an optimal solution to this problem we need to know the optimal actions only for the water states which can be reached using optimal actions. For problems with many reservoirs this region is usually a very small proportion of the whole water state space. Two of the methods studied in this paper, nested Benders decomposition and dynamic programming decomposition, are able to exploit this to greatly reduce computation.

The possible sequences of weather patterns starting from an initial known weather state can be represented as in figure 2 by a tree of scenarios. Each of the nodes on this tree is referred to as a scene. Each scene has an associated underlying weather state and runoff index. Several scenes at the same period can have the same weather state. The path from the node representing the initial state to any other node represents the sequence of weather states passed through. In an optimal solution there will be a vector of volumes and releases for each node in the tree, and these volumes and releases may be different for the same current weather state depending on the path followed to reach that state. To formulate this problem as a single optimization problem we therefore need to have different flow and reservoir volume variables for every scene in the scenario tree. For initial scene w_1 with weather state s_1, i_1, initial water state $\boldsymbol{h}_1$ and terminal value functions $\gamma^T_{s,i}$ the problem is the following:

$$\begin{aligned} V^1_{s_1,i_1}(\boldsymbol{h}_1) \quad = \quad & \max \sum_{t=1}^{T} \sum_{w \in W^t} \pi_w r^t(\boldsymbol{h}_w, \boldsymbol{x}_w) + \sum_{w \in W^T} \pi_w \gamma^T_{s_w,i_w}(\boldsymbol{h}_w + M\boldsymbol{x}_w + \boldsymbol{q}^t_{i_w}) \\ & \text{subject to } \boldsymbol{h}_{\tilde{w}} = \boldsymbol{h}_w + M\boldsymbol{x}_w + \boldsymbol{q}^t_{i_w}, \forall w \to \tilde{w} \in C^t, \qquad (4) \\ & \underline{\boldsymbol{x}} \leq \boldsymbol{x}_w \leq \overline{\boldsymbol{x}}, \underline{\boldsymbol{h}} \leq \boldsymbol{h}_w \leq \overline{\boldsymbol{h}}, \forall w \in \bigcup_{t=1}^{T} W^t. \end{aligned}$$

π_w, the probability of scene w occurring, satisfies the recurrence:

$$\begin{aligned} \pi_{\tilde{w}} &= \pi_w P^t_{s_{\tilde{w}}|s_w,i_w} p^{t+1}_{i_{\tilde{w}}|s_{\tilde{w}}}, \forall w \to \tilde{w} \in C^t, \\ \pi_{w_1} &= 1. \end{aligned}$$

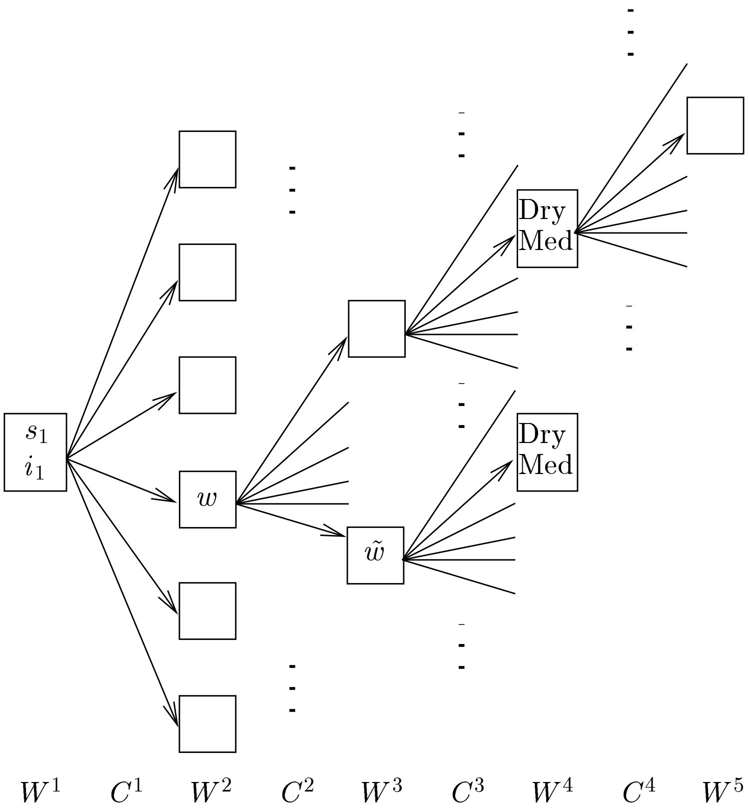

Figure 2: Scenario tree

2 Restrictions on problems considered

Note that $\tilde{\boldsymbol{h}}$ is a parameter within the optimization problem (2) and makes a linear contribution to the right hand side of a linear equation. Hence provided $\gamma_{s,i}^t(\boldsymbol{h})$ are concave functions of $\boldsymbol{h}$ and $r^t(\tilde{\boldsymbol{h}}, \boldsymbol{x})$ concave functions of $\tilde{\boldsymbol{h}}$ and $\boldsymbol{x}$ jointly, the functions $V_{s,i}^t(\tilde{\boldsymbol{h}})$ will be concave functions of $\tilde{\boldsymbol{h}}$. Also the $P_{\tilde{s}|s,i}$ are non-negative so provided that the functions $v_{\tilde{s}}^{t+1}(\boldsymbol{h})$ are concave, $\gamma_{s,i}^t(\boldsymbol{h})$ will be concave. Also since the $p_{i|s}$ are non-negative it follows that if the functions $V_{s,i}^t(\boldsymbol{h})$ are concave, then the functions $v_s^t(\boldsymbol{h})$ are concave. All these results also hold if concave is replaced by concave and piecewise linear.

Since NBD and RS are applicable only to convex problems, we have to make some restrictions on the form of the examples that we consider. In general for the hydro-electric optimization problem the immediate reward function is not concave. The amount generated by a hydro-generator (and hence the reward for the generation) depends on both the flow through the generator and the reservoir level, which itself depends on the volume in the reservoir. The dominant terms in the immediate reward function behave like $\tilde{h}_r x_g$, where x_g is the flow through generator g and $\tilde{h}_r$ is the volume in the reservoir r which feeds generator g. Such a product of variables is not concave. In many cases however the range of water levels in a reservoir is small compared to the minimum distance between the reservoir surface and the generator it feeds, so it is a good approximation to assume the head is constant. We shall make this assumption here.

We also assume that a generator's output of electricity is proportional to the flow through it, i.e. of the form $\phi_g x_g$ where the constant ϕ_g is the electricity generated per unit volume. G denotes the set of indexes of flows through generators. We assume further that the income from the sale of electricity in a period is a piecewise linear concave function of the total amount generated in that period. For period t this function has L^t linear sections and the marginal revenue per unit of generation in section l is θ_l^t. The form of the function giving reward in terms of total generation is shown in figure 3. We also assume that each flow x_f which represents flooding incurs a penalty $\rho_f x_f$, for some non-negative constant ρ_f. F denotes the set of indexes of flows which represent flooding. With these assumptions r^t is a piecewise linear concave function of $\boldsymbol{x}$ only and, provided it is being maximized, it can be modelled by the following linear constraints.

$$\begin{aligned} r^t(\boldsymbol{x}) &= \sum_{l=1}^{L^t} \theta_l^t z_l - \sum_{f \in F} \rho_f x_f, \qquad (5) \\ & \sum_{l=1}^{L^t} z_l = \sum_{g \in G} \phi_g x_g, \\ & 0 \le z_l \le \overline{z}_l^t, l = 1, \ldots, L^t. \end{aligned}$$

Under our assumptions $\theta_l^t > \theta_{l+1}^t$, so the maximization ensures that if $z_l > 0$ then $z_j = \overline{z}_j^t$ for all $0 \le j < l$. Finally we assume that the terminal value functions $\gamma_{s,i}^T$ are concave and piecewise linear. Problem (4) is then a linear programming problem.

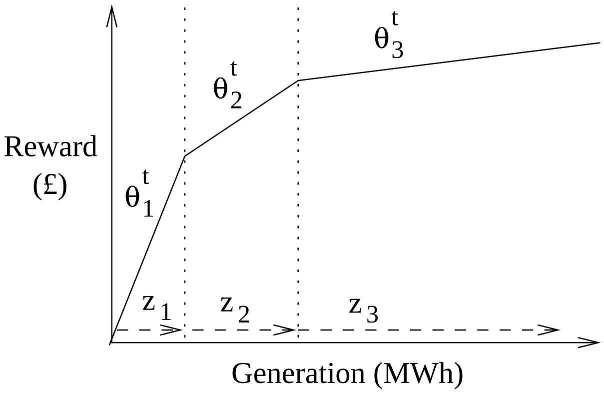

Figure 3: Electric generation reward

3 Solution methods

In this paper we compare four solution methods for the reservoir optimization problem. These are the revised simplex method (RS) for the problem formulated as a single LP, nested Benders decomposition (NBD), full dynamic programming (FDP) with a uniform discretization of the state and action spaces, and a DP decomposition technique (DPD) in which the state and action space discretizations are a better match to the structure of the problem.

3.1 Revised simplex: RS

Since with the above assumptions problem (4) is a LP problem, it can be solved as a single problem by the revised simplex method. The number of scenes in the scenario tree grows exponentially with the number of periods. A problem with k weather states, m runoffs per weather state and n periods has $(k^n m^n - 1)/(km - 1)$ scenes. For this study we have chosen $k = 2$, $m = 3$ and $n = 5$, which gives a tree with 1555 scenes. For a problem with 17 reservoirs this gives a single LP with 54425 rows and 179349 columns, which is a large problem and is near the limit of what is reasonable to solve as a single LP. In this study we have used the simplex implementation in OSL v1.2. The solution process is started from the all logical basis.

3.2 Nested Benders decomposition: NBD

It follows from the above assumptions on the r^t and $\gamma^T_{s,i}$ that each of the future value functions $\gamma^t_{s,i}$ is also concave and piecewise linear. Subproblem (2) is therefore an LP problem. The problem can therefore be solved by nested Benders decomposition (NBD) with the subproblems being solved by the simplex method.

The central step in NBD is the step (2) with the function $\gamma^t_{s,i}$ replaced with some concave piecewise linear upper bound $\overline{\gamma}$, which gives the problem

$$NBD^t_{s,i}(\tilde{\boldsymbol{h}}, \overline{\gamma}) : \quad \overline{V}(\tilde{\boldsymbol{h}}) = \max_{\boldsymbol{x},\boldsymbol{h}} (r^t(\boldsymbol{x}) + \overline{\gamma}(\boldsymbol{h})) \tag{6}$$
$$\text{subject to } \boldsymbol{h} - M\boldsymbol{x} = \tilde{\boldsymbol{h}} + \boldsymbol{q}^t_i,$$
$$\underline{\boldsymbol{x}} \le \boldsymbol{x} \le \overline{\boldsymbol{x}}, \underline{\boldsymbol{h}} \le \boldsymbol{h} \le \overline{\boldsymbol{h}}.$$

Since $\gamma_{s,i}^t(\boldsymbol{h}) \le \overline{\gamma}(\boldsymbol{h})$ for all $\boldsymbol{h}$, it follows that $V_{s,i}^t(\tilde{\boldsymbol{h}}) \le \overline{V}(\tilde{\boldsymbol{h}})$ for all $\tilde{\boldsymbol{h}}$. Because $\overline{\gamma}$ is concave, a local linearization around the solution for one given value of $\tilde{\boldsymbol{h}}$ yields an affine function $L(\tilde{\boldsymbol{h}})$ which is an upper bound on $\overline{V}(\tilde{\boldsymbol{h}})$ for all $\tilde{\boldsymbol{h}}$, i.e. $\overline{V}(\tilde{\boldsymbol{h}}) \le L(\tilde{\boldsymbol{h}})$ for all $\tilde{\boldsymbol{h}}$. It follows that $V_{s,i}^t(\tilde{\boldsymbol{h}}) \le \overline{V}(\tilde{\boldsymbol{h}}) \le L(\tilde{\boldsymbol{h}})$ for all $\tilde{\boldsymbol{h}}$, so $L(\tilde{\boldsymbol{h}})$ is an upper bounding function for $V_{s,i}^t(\tilde{\boldsymbol{h}})$. $L(\tilde{\boldsymbol{h}})$ can be used to tighten the upper bound function $\overline{\gamma}$ from the previous period.

There are different ways to form the functions $\overline{\gamma}$. One approach is to express $\overline{\gamma}_{s,i}^{t-1}$ as a sum of individual upper bounding functions $\overline{V}_{\tilde{s},\tilde{i}}^t$ on $V_{\tilde{s},\tilde{i}}^t$ using (3) and (1) to give

$$\overline{\gamma}_{s,i}^{t-1}(\boldsymbol{h}) = \sum_{\tilde{s}\in S^t} P_{\tilde{s}|s,i}^{t-1} \sum_{\tilde{i}\in I^t} p_{\tilde{i}|\tilde{s}}^t \overline{V}_{\tilde{s},\tilde{i}}^t(\boldsymbol{h}).$$

With this form for $\overline{\gamma}_{s,i}^{t-1}$, the new cut is used to tighten one of the $\overline{V}_{\tilde{s},\tilde{i}}^t$ bound functions. A second approach combines a family of cuts into an aggregate cut using

$$L_{s,i}^{t-1}(\boldsymbol{h}) = \sum_{\tilde{s}\in S^t} P_{\tilde{s}|s,i}^{t-1} \sum_{\tilde{i}\in I^t} p_{\tilde{i}|\tilde{s}}^t L_{\tilde{s},\tilde{i}}^t(\boldsymbol{h}).$$

$L_{s,i}^{t-1}$ is then a cut which can be used to tighten the bounds function $\overline{\gamma}_{s,i}^{t-1}$ directly. A third approach merges cuts to form upper bound functions $\overline{v}_s^t$ on v_s^t and then expresses $\overline{\gamma}_{s,i}^{t-1}$ as the sum of the bounds $\overline{v}_s^t$. In this study we use the second approach, called the aggregate cut method. The only cuts which are combined are those that are generated for the same value of $\tilde{\boldsymbol{h}}$. This produces smaller subproblems than the alternative multi-cut method, but in general the bounds are accurate over a smaller region of the state space. We choose the aggregate cut method because the individual LP subproblems are small relative to the number of cuts, so introducing multi-cuts increases subproblem solution time significantly.

The value functions for some weather state s, i and any bounding functions for them are valid for all scenes ω which have that weather state. However the areas where the bounds have to be accurate are likely to be different in different scenes. Sharing cuts between different scenes with the same weather state has the advantage that cuts generated for one scene are used by other scenes with the same state, so improving the accuracy of the initial solutions. The disadvantage is that many of the cuts accumulated for a scene will be inaccurate in the region of interest for that scene, so the subproblems will be unnecessarily large. In this study we build up separate bounding functions for each scene, which is a good strategy if the problem is to be solved to high accuracy.

A piecewise linear function which has to be maximized can be represented by the set lying below all of the hyperplanes which define its sections. In the aggregate cut case the PWL function $\overline{\gamma}$ can be represented as the minimum of all the hyperplanes (cuts) defining it. At some stage in the algorithm assume the cuts associated with scene w are indexed by $u \in U_\omega$ for some finite index set U_ω, and that cut u has gradient $\boldsymbol{\gamma_u}$ and intersection k_u. As new cuts are generated their indices are added to the set U_ω. Using this representation for the aggregate cut method, problem (6) can be restated in the form

$$\overline{V}_\omega(\tilde{\boldsymbol{h}}) \quad = \quad \max_{\boldsymbol{x},\boldsymbol{h},\boldsymbol{z},\overline{\gamma}} \left(\sum_{l=1}^{L^t} \theta_l^t z_l - \sum_{f\in F} \rho_f x_f + \overline{\gamma} \right), \qquad (7)$$

$$\begin{aligned}
&\text{subject to } \boldsymbol{h} - M\boldsymbol{x} = \tilde{\boldsymbol{h}} + \boldsymbol{q}^t_{i_\omega},\\
&\underline{\boldsymbol{x}} \leq \boldsymbol{x} \leq \overline{\boldsymbol{x}}, \boldsymbol{h} \leq \tilde{\boldsymbol{h}} \leq \overline{\boldsymbol{h}},\\
&\sum_{l=1}^{L^t} z_l = \sum_{g \in G} \phi_g x_g,\\
&0 \leq z_l \leq \overline{z}^t_l, l = 1, ...L^t,\\
&\overline{\gamma} \leq (\boldsymbol{\gamma}_u)^T \boldsymbol{h} + k_u, u \in U_\omega.
\end{aligned}$$

In dynamic programming the value functions are evaluated in their entirety in a systematic manner starting from the final period and working back to the initial period. When the calculation is done for period t it can be assumed that all the value functions at period $t+1$ are fully evaluated. By contrast in NBD an attempt is made to evaluate the value surface only in the regions which might be part of the optimal solutions given the starting state. When a subproblem is solved, giving solution $\boldsymbol{x}^*, \boldsymbol{h}^*$, one of the options is to generate a cut and pass that back to the previous period as described above. Alternatively $\boldsymbol{h}^*$, which has been calculated using bounds on γ rather than the true value, can be passed to all the subproblems at the following period which are connected in the scenario tree to the solved problem. Eventually these problems will pass back cuts which may improve the upper bound γ in a neighbourhood of $\boldsymbol{h}^*$.

Different versions of NBD have different rules for deciding when to pass a cut backwards to the previous period and when to pass an inventory, i.e. a vector of reservoir volumes, forward to the future. All methods start with a given state in period 1 (unlike DP which starts a final cost for all states in period T). In this study arbitrarily high and constant upper bounds functions $\overline{\gamma}$ for all scenes w are used initially. The results presented in this paper were obtained using the fast forward fast backward strategy (FFFB), which we found is the most effective for large problems, agreeing with [11]. The FFFB strategy alternates between forward phases, in which problems are solved in order of increasing period with their results being passed to the following period until the final period T is reached, and then a backward phase starting at period T and ending at period 1, in which the problems are re-solved and cuts are generated and passed back to their parents at the previous period.

When solving a problem by NBD a large number of the subproblems can be solved independently. The use of parallel computing architectures can take advantage of this. Subproblems are distributed amongst several processors, and cuts and inventories are passed between processors as they are generated. The efficiency of any parallel implementation is heavily influenced by both the speed of the communications between processors and the balance of work to processors.

As we are implementing our NBD algorithm on a local area network of Sun workstations, the workload is distributed amongst processors so as to minimize the need for communications between processors. This has been achieved by dividing the scenario tree into self-contained subtrees of nearly equal size, and placing one subtree on each processor. The divisions have been made as close as possible to the first problem. This allows the subtrees to be solved concurrently. The FFFB strategy ensures that the amount of work in each subtree is similar and so ensures there is good load balance. The only communications between processors is when the head node of a subtree has to pass a cut up or receive an inventory.

Within any one period the subproblems are similar. Initially they differ only in their right hand sides. This changes as cuts are added to $\overline{\gamma}$. This similarity can be exploited by using the basis which solved one subproblem in one scene w in the scenario tree to give a hot start to the neighbouring scene in the same period the first occasion it is solved. On subsequent visits to that scene its previous optimal basis is used for the hot start. This leads to a very significant reduction in the number of simplex iterations taken. The simplex implementation in OSL v1.2 has been used to solve the subproblems.

3.3 Full dynamic programming: FDP

The convexity assumption, which was necessary to allow the problem to be solved by nested Benders decomposition and the simplex method, is not required by dynamic programming methods. (There are techniques which can exploit convexity to reduce the action space optimization when the problem is convex, but these will not be used in this study as this would not then give useful information about solution times when dynamic programming is applied to the more general non-convex case.) Calculating v_s^t from the $V_{s,i}^t$ as in (1) rather than directly from the v_s^{t+1} leads to a reduction in computation time and memory usage, especially when the number of runoffs is high.

In the versions of dynamic programming used in this paper, the water state space and the action space are discretized. The only actions considered are those which result in water state transitions between the defined discrete levels, so we do not have to consider interpolation between discretized states. These actions are always feasible in the continuous problem, and from this it follows that the objective function values calculated are lower bounds on the values for the continuous problem. For a given transition between the water states we need to find the actions which achieve this in a way that maximizes the immediate reward. This is a simpler task if the output of any reservoir directly enters at most one other reservoir and there are no cycles in the network. We refer to this as the *single output assumption.* Given runoffs and changes in reservoir volumes, it is then easy to calculate the flows between reservoirs by working down from the top reservoirs, i.e. those that do not receive any discharges from other reservoirs. Knowing the flow between reservoirs, the optimal immediate reward is obtained by using as much of this as possible to generate and only allowing spill or controlled release when the generator is operating at its maximum capacity. This topology for the hydrological network rules out situations like pump storage, but the DP method can be extended to allow more general hydrological networks at the expense of additional computations.

In the full version of dynamic programming each component of $\boldsymbol{h}$, i.e. the volume in one reservoir, is restricted to one of k equally spaced values, which include the upper and lower bounds on the reservoir's volume. The reservoir state space therefore consists of k^R discrete values, and a similar number of actions for each state, as all transitions to accessible states at the following stage have to be considered. For a 4 reservoir network with 11 discrete values per reservoir, the work per stage involves about 11^8 state/action evaluations. Each evaluation involves several operations as it requires the calculation of the flows between reservoirs and then the immediate reward. This is a large problem and is the largest case we consider by full DP.

The full DP is highly suited to parallel processing as the maximizations in (2) are non-trivial, independent tasks. We have developed a parallel implementation of full DP that performs the maximizations for different runoffs on separate processors. For a problem with m runoffs, we would then hope to see the solution time reduced by a factor of m. As the number of runoffs in the examples considered in this paper has been kept low to limit the number of scenes in the LP model, the number of processors that can be used is restricted. If our aim had been to develop an efficient parallel algorithm to solve problems with such a small number of runoffs, then we would have partitioned the water state space also, so that more processors could be employed.

3.4 Dynamic programming decomposition: DPD

As discussed in section 1, the optimal solution to the problem for a given initial starting state visits only a very small proportion of the state and action space. Nested Benders decomposition gives one automatic technique for homing in on this region and excluding the rest by the use of suitable bounds. The second DP approach we consider in this paper also attempts to restrict the region of the state space which has to be evaluated. In the full DP approach a uniform grid of values is chosen over each reservoir volume independently. In the second approach states are chosen for groups of reservoirs in an attempt to populate the areas of the state space most likely to be visited in an optimal solution. Taking capacity to mean the difference between the upper and lower bounds on a reservoir's volume, we note that in optimal operation the ratio of the volume in a reservoir above its lower bound to the capacity of the reservoir is often similar for all reservoirs. We say that a set of states in which all these proportions are equal satisfies the *equally full heuristic*. If this were exactly true in optimal solutions then we could reduce the water state space to a single variable that defines the amount of water in any reservoir as a proportion of the total capacity of that reservoir, all reservoirs having the same proportion. Because this approximation is seldom accurate, we use a decomposition method. This is described fully in [1]. A summary is given below.

In the DP decomposition method a DP problem is solved for each reservoir. The problem for a given reservoir (called the *focus reservoir*) models the state space of the focus reservoir in detail and splits the remaining reservoirs into two groups depending on whether or not their outflows could eventually reach the focus reservoir. The state spaces of each of these groups are then approximated by the equally full heuristic. This gives a DP with a state space of either 2 or 3 dimensions.

Each problem gives a target release for its focus reservoir as a function of the volumes in the focus reservoir and the total volumes in the two groups. Since the target releases are determined independently, they do not necessarily constitute a feasible set of actions for the full problem. We therefore need a rule to identify a feasible set of actions from the target releases. Provided the hydrological network satisfies the single output assumption the following simple method, which we will call the *top-down correction method*, will construct a feasible operating policy for the full system. First note that for the top reservoirs the target releases are always feasible, so these reservoirs' actual releases are set equal to their target releases. Now select any reservoir for which the actual releases from all the reservoirs upstream of it have been set, and apply the target reservoir at this reservoir. If this is feasible given the actual flow coming in to that reservoir, then set the actual release from the reservoir equal to its target release. If the target release is not feasible because there is not enough water flowing into the reservoir, set the actual release to leave the reservoir's volume at its lower bound at the beginning of the next period. If the target release is not feasible because there is too much water flowing into the reservoir, set the actual release to leave the reservoir's volume at its upper bound at the beginning of the next period. Repeat this step until all the actual releases have been set.

A simulator is needed to evaluate the feasible solution produced by the top-down correction method. The value of the actions produced by the method is calculated for each possible path through the scenario tree and their expected value is calculated. This can then be compared with the results from the other methods. Because the top-down correction method yields feasible solutions, the expected value calculated will be a lower bound on the true optimum value.

In our parallel implementation of the DP decomposition method, each problem is solved simultaneously and within each problem the maximizations for different runoffs are performed simultaneously. For a problem with R reservoirs and m runoffs, there are Rm tasks that can be performed in parallel. If this is not sufficient to keep all available processors occupied, then the water state space in each problem could also be partitioned.

4 Results

Because we have chosen examples in this study which can be formulated exactly as linear programming problems, the solutions given by RS and NBD are exact up to numerical rounding. The FDP results however are not exact because the state and action spaces have been represented by a finite set of discrete values. The DPD method does not give an objective function value that can be compared with the results of the other methods. The objective function value for each subproblem is a lower bound on the objective function value for the continuous problem, because it is determined using releases that are feasible for that problem. However these releases are likely to be very different from the releases generated by combining the results of all the subproblems. The expected value for this method has been calculated using a simulator which evaluates the reward along every path in the scenario tree and using the top-down correction method to produce a feasible set of releases in each scene. Both these comparisons are given below in table 1. Note that the FDP and DPD results are less than the corresponding RS and NBD results because FDP and DPD with top-down correction give releases which, for the continuous problem, are feasible but not necessarily optimal. Given the fact that such a small number of discrete values are being used to represent high dimensional state spaces, it is remarkable that FDP is within 1% and DPD is within 3.2% of the true value.

The CPU times in seconds on a 70MHz SUN Sparc 5 for the solution of each example by the four methods is given in table 2. NBD is the fastest of the four methods and so should be used if the assumptions that led to the problem being formulated as a LP problem (see section 2) are valid. DPD is the next fastest method. Both NBD and DPD are practical methods for large problems. NBD solves P17R faster than RS solves P3R, while DPD solves P17R faster than FDP solves P4R.

Table 3 shows the speed-ups obtained from the parallel implementations of NBD, FDP and DPD. (RS does not parallelize on a network of workstations.) NBD parallelizes extremely well, usually showing a superlinear speed-up. This is because the parallel implementation takes less CPU time and the amount of paging it performs is significantly less. These effects can be seen in table 4 which shows the elapsed time, the CPU time and the number of passes up and down the scenario tree for the serial and parallel implementations of NBD. The reason that the serial and parallel implementation do not have the same total CPU time is that the subproblems often have multiple optimal solutions so the method is not deterministic. The parallel and serial algorithms select different optimal bases in the first few subproblem solutions and this causes the tree to be searched in different ways with different cuts being generated. In three of the examples in this study the parallel solution required fewer passes. The difference between elapsed time and CPU time in the serial case is due to paging. The reduction in paging in the parallel case is due to the fact that the working set is divided between several processors, so a higher proportion of it remains in main memory.

Table 4 also reveals that the large difference in the solution times for P4R and P8R by NBD is largely due to the difference in the number of passes required, not the difference in the number of reservoirs.

Example	Expected Reward	RS & NBD	FDP	DPD
P3R	19796383	0.00%	0.94%	1.25%
P4R	19620562	0.00%	0.52%	0.74%
P8R	18721904	0.00%	—	2.21%
P17R	18691236	0.00%	—	3.13%

Table 1: Expected rewards and percentage error

Example	RS	NBD	FDP	DPD
P3R	7214.63	94.98	173.24	15.67
P4R	11505.23	106.97	15214.62	524.13
P8R	38709.65	764.93	—	1986.30
P17R	181509.60	2254.46	—	11177.98

Table 2: CPU time (sec) on a 70MHz SUN Sparc 5

Example	NBD	FDP	DPD
P3R	6.08× on 6	2.33× on 3	2.15× on 3
P4R	6.01× on 6	2.54× on 3	2.68× on 4
P8R	5.94× on 6	—	3.29× on 8
P17R	7.47× on 6	—	5.55× on 8

Table 3: Speed-ups on a network of 70MHz SUN Sparc 5

	Serial solution			Parallel solution		
Example	Elp	CPU	Passes	Elp	CPU	Passes
P3R	109.05	94.98	5	17.95	89.14	5
P4R	108.24	106.97	6	18.01	90.39	5
P8R	851.32	764.93	35	143.29	700.93	33
P17R	3286.40	2254.46	66	439.88	2042.15	62

Table 4: Details of serial and parallel solutions

5 Conclusions

The different methods compared in this study have properties which make them suitable for different types of reservoir problem. The number of scenes in the scenario tree used by the revised simplex method (RS) and nested Benders decomposition (NBD) is approximately $(km)^{T-1}$, where k is the number of underlying weather states, m is the number of runoffs per weather state, and T is the number of time periods. This increases exponentially with the number of periods, which limits the number of periods which can be solved by these methods to 5 or 6. The work also increases as the $T-1$ power of the total number of weather states. Scenario sampling can be used with NBD to overcome the problem of the exponential growth in the number of scenarios, but this technique will only find an approximate solution overcomes. The work in dynamic programming on the other hand increases linearly with the number of periods, linearly with the number of runoffs and at worst as the square of the number of underlying weather states. This makes the DP approach suitable for long time horizons with detailed modelling of the runoffs and underlying weather states.

The RS and NBD approaches are also restricted to linear or convex problems, which rules them out for problems where the variation in head is significant. This is however often a second order effect.

When using DP to solve non-convex problems, it is necessary to discretize the state space. If this is done in a uniform manner the number of discrete values grows exponentially with the number of reservoirs. This effect, known as the curse of dimensionality, limits the size of problem which can be tackled to 3 or 4 reservoirs. This paper has presented results from a dynamic programming decomposition method which avoids this curse of dimensionality by decomposing large problems into a number of low dimensional problems. For the examples studied this gave objective function values within 3.2% of the optimum, which is very satisfactory for problems with up to 17 reservoirs.

Because the different methods have different strengths, hybrid methods which combine their strengths are of great interest. For example to solve a long term reservoir problem, the Benders approach could be used to model the first few periods accurately and the DP approach could be used for periods after that. The DPD would be solved first to determine the values of the different water states at the end of the interface period, and the NBD would then be solved with these as its terminal values. The terminal values in the test problems used in this paper were determined in this way. DPD was applied to solve each problem using arbitrary terminal values. Separable piecewise linear approximations to the resulting value functions were then constructed and used as the terminal values in the test problems. Hence the test problems can be thought of as 10 period models involving two 5-period cycles. This approach was adopted to reduce the effect of using arbitrary terminal values.

The computational results for the serial implementations show that NBD is much more efficient for this type of problem than RS. The comparison using elapsed time instead of CPU time is even more in favour of NBD: the working set is much smaller in NBD which leads to much less paging in large problems. In addition NBD parallelizes well on a network of workstations whereas is is not possible to parallelize the RS method on workstations.

6 Acknowledgements

The authors are grateful to the Engineering and Physical Sciences Research Council for their support of this work under grant GRH43700.

References

[1] T. W. Archibald, K. I. M. McKinnon, and L. C. Thomas. An aggregate stochastic dynamic programming model of multiple reservoir systems. *Working paper 95/12, Department of Business Studies, University of Edinburgh.*, 1995.

[2] G. B. Dantzig and P. W. Glynn. Parallel processors for planning under uncertainty. *Annals of Operations Research*, 22:1–21, 1990.

[3] J. Dupacova, A. Gaivoronski, Z. Kos, and T. Szantai. Stochastic programming in water management: A case study and a comparison of solution techniques. *European Journal of Operational Research*, 52:28–44, 1991.

[4] G. Infanger. *Planning under uncertainty.* Boyd & Fraser, 1994.

[5] J. Jacobs, G. Freeman, J. Grygier, D. Morton, G. Schultz, K. Staschus, and J. Stedinger. Socrates — a system for scheduling hydroelectric generation under uncertainty. *Annals of Operations Research*, 59:99–133, 1995.

[6] J. Kelman, J. R. Stedinger, L. A. Cooper, E. Hsu, and S. Yuan. Sampling stochastic dynamic programming applied to reservoir operation. *Water Resources Research*, 26:447–454, 1990.

[7] B. F. Lamond and A. Boukhtouta. Optimizing future hydro-power production using markov decision processes. Technical Report 95-44, Faculte des Sciences de l'Administration, Universite Laval, Quebec, 1995.

[8] M. V. F. Pereira and L. M. V. G. Pinto. Multi-stage stochastic optimization applied to energy planning. *Mathematical Programming*, 52:359–375, 1991.

[9] K. Reznicek and T. Cheng. Stochastic modelling of reservoir operations. *European Journal of Operational Research*, 50:235–248, 1991.

[10] J. R. Stedinger, B. F. Sule, and D. P. Loucks. Stochastic dynamic programming models for reservoir operation optimization. *Water Resources Research*, 20:1499–1505, 1984.

[11] R. J. Wittrock. Advances in a nested decomposition algorithm for solving staircase linear programs. Technical Report SOL 83-2, Systems Optimization Laboratory, Department of Operations Research, Stanford University, 1983.

[12] S. Yakowitz. Dynamic programming applications in water resources. *Water Resources Research*, 18:673–696, 1982.

7 Appendix: Problem data

Figure 4 shows the hydrological network for each of the four test problems and tables 5 to 10 give the rest of the data for these test problems. The runoff into a reservoir during a period is equal to the product of the runoff capture for that reservoir and the total rainfall for that period divided by 9.861. The penalties for flooding are set to zero in all four problems.

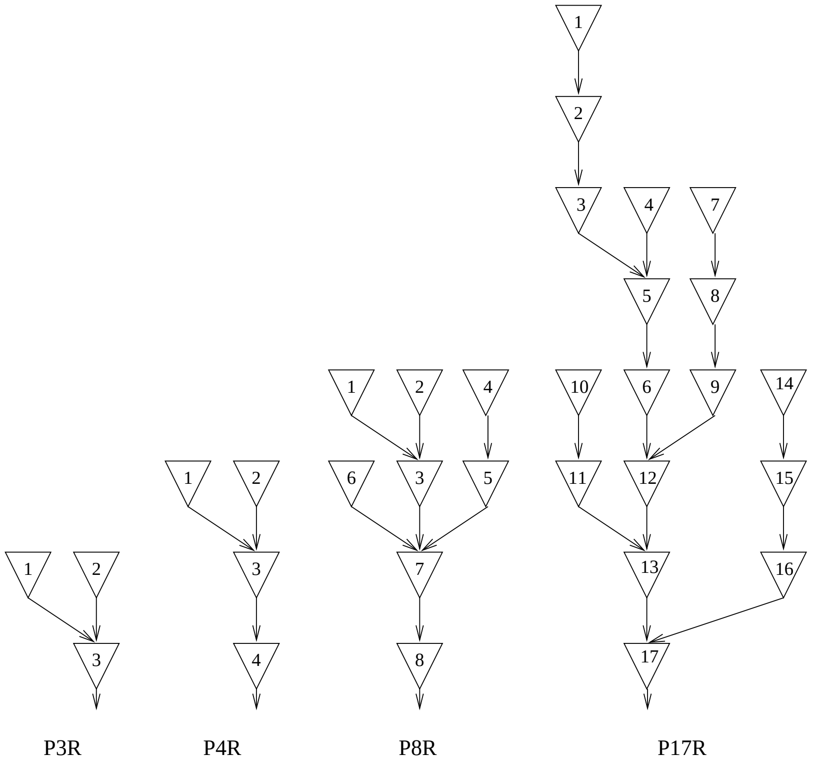

Figure 4: Hydrological networks

Period	High Price (£/MWh)	Demand at High Price (MWh)	Low Price (£/MWh)	Total Rainfall (Mm^3) Lo	Med	Hi
1	0.85	1432779.5	0.51	4240.23	19327.56	39739.83
2	0.91	1432779.5	0.55	4079.50	16166.12	32518.62
3	1.35	1432779.5	0.81	1660.59	5549.77	10811.60
4	1.00	1432779.5	0.60	2269.02	15730.27	33942.55
5	0.73	1432779.5	0.44	4313.20	20272.24	41863.89

Table 5: Period specific data common to all 4 problems

i	s	$p_{i\|s}$	$P_{Dry\|i,s}$	$P_{Wet\|i,s}$
Lo	Dry	0.50	0.95	0.05
Med	Dry	0.30	0.75	0.25
Hi	Dry	0.20	0.55	0.45
Lo	Wet	0.10	0.60	0.40
Med	Wet	0.30	0.30	0.70
Hi	Wet	0.60	0.10	0.90

Table 6: Transition probabilities common to all problems and periods

	Res	Turbine	Head	Terminal Value (£/Mm^3)						
	Cap	Cap	(MWh	0-30%		30-70%		70-100%		Runoff
Res	(Mm^3)	(Mm^3)	/Mm^3)	Dry	Wet	Dry	Wet	Dry	Wet	Capture
1	31030	4532.25	183.40	203.86	156.76	143.35	103.26	102.6	45.6	2.775
2	31049	3541.80	175.12	185.42	138.78	123.53	77.80	82.5	34.4	2.737
3	29363	11973.38	177.94	65.40	31.12	47.68	17.70	32.3	6.0	4.349

Table 7: Reservoir specific data for P3R

	Res	Turbine	Head	Terminal Value (£/Mm^3)						
	Cap	Cap	(MWh	0-30%		30-70%		70-100%		Runoff
Res	(Mm^3)	(Mm^3)	/Mm^3)	Dry	Wet	Dry	Wet	Dry	Wet	Capture
1	31030	4532.25	183.40	203.51	157.33	140.22	102.67	100.87	42.62	2.775
2	31049	3541.80	175.12	188.61	139.95	119.00	77.68	73.78	30.74	2.737
3	12866	8301.90	44.70	57.21	29.29	51.48	18.98	44.77	12.77	1.238
4	16497	10222.23	172.12	42.23	14.23	32.97	7.56	21.20	3.01	3.111

Table 8: Reservoir specific data for P4R

	Res	Turbine	Head	Terminal Value (£/Mm^3)						
	Cap	Cap	(MWh	0-30%		30-70%		70-100%		Runoff
Res	(Mm^3)	(Mm^3)	/Mm^3)	Dry	Wet	Dry	Wet	Dry	Wet	Capture
1	20097	3919.93	120.79	210.36	169.14	160.65	118.78	116.09	52.39	1.777
2	504	384.10	66.36	134.05	107.78	95.75	76.99	69.19	62.11	0.044
3	10429	3949.62	84.11	100.48	65.03	71.77	37.88	51.86	20.80	0.954
4	13015	920.10	125.90	225.12	160.77	138.83	83.14	94.41	39.67	1.139
5	18034	4413.92	114.27	117.05	87.93	104.91	66.57	93.38	33.28	1.598
6	4694	3141.01	36.33	86.64	59.45	64.25	43.61	46.26	33.60	0.413
7	12866	7882.02	67.40	62.87	21.67	31.44	9.61	24.08	6.23	1.855
8	11803	8727.84	170.16	19.00	5.45	15.26	3.89	10.99	3.16	2.081

Table 9: Reservoir specific data for P8R

Res	Res Cap (Mm^3)	Turbine Cap (Mm^3)	Head (MWh /Mm^3)	Terminal Value (£/Mm^3) 0-30% Dry	0-30% Wet	30-70% Dry	30-70% Wet	70-100% Dry	70-100% Wet	Runoff Capture
1	675	480.40	24.87	116.00	109.72	116.00	83.60	116.00	69.53	0.059
2	17217	1709.60	87.90	219.43	195.00	161.12	105.50	111.07	54.00	1.521
3	2205	7464.30	41.70	94.46	71.77	94.46	61.50	94.46	58.66	0.197
4	504	384.10	66.36	93.39	93.40	93.39	85.16	93.39	69.54	0.044
5	5260	2839.60	60.70	110.35	68.99	81.52	46.89	81.52	33.04	0.479
6	5169	2988.00	53.50	59.08	55.37	59.08	30.41	59.08	21.04	0.475
7	13015	920.10	125.90	219.33	165.38	140.99	95.21	109.82	45.69	1.139
8	12454	5013.50	62.23	148.92	103.69	106.25	76.93	99.72	39.99	1.100
9	5580	2698.70	71.30	62.31	61.96	62.31	29.68	62.31	22.34	0.498
10	2566	2439.10	20.00	82.92	85.74	68.99	52.23	68.99	45.49	0.225
11	2128	2572.40	25.40	49.65	40.08	49.65	25.78	49.65	25.78	0.188
12	12866	8301.90	44.70	57.48	39.45	49.17	25.95	39.14	5.39	1.238
13	6934	7055.20	22.70	29.91	17.50	23.91	6.31	23.91	2.39	0.617
14	3038	351.90	3.90	51.56	60.41	51.56	43.23	51.56	33.12	0.266
15	3041	1132.20	11.96	49.25	49.50	49.25	34.69	49.25	30.37	0.269
16	5724	1647.50	45.00	40.31	31.52	36.59	31.52	36.59	17.79	0.511
17	11632	11831.30	118.00	40.28	21.91	22.71	6.14	11.15	1.05	1.035

Table 10: Reservoir specific data for P17R