Tree-based fitted Q-iteration for multi-objective Markov decision processes in water resource management

F. Pianosi, A. Castelletti and M. Restelli

ABSTRACT

Multi-objective Markov decision processes (MOMDPs) provide an effective modeling framework for decision-making problems involving water systems. The traditional approach is to define many single-objective problems (resulting from different combinations of the objectives), each solvable by standard optimization. This paper presents an approach based on reinforcement learning (RL) that can learn the operating policies for all combinations of objectives in a single training process. The key idea is to enlarge the approximation of the action-value function, which is performed by single-objective RL over the state-action space, to the space of the objectives' weights. The batch-mode nature of the algorithm allows for enriching the training dataset without further interaction with the controlled system. The approach is demonstrated on a numerical test case study and evaluated on a real-world application, the Hoa Binh reservoir, Vietnam. Experimental results on the test case show that the proposed approach (multi-objective fitted Q-iteration; MOFQI) becomes computationally preferable over the repeated application of its single-objective version (fitted Q-iteration; FQI) when evaluating more than five weight combinations. In the Hoa Binh case study, the operating policies computed with MOFQI and FQI have comparable efficiency, while MOFQI provides a continuous approximation of the Pareto frontier with no additional computing costs.

Key words | multi-objective optimization, optimal control, reinforcement learning, reservoir operation, tree-based models

F. Pianosi (corresponding author)
A. Castelletti
M. Restelli
Dipartimento di Elettronica e Informazione,
Politecnico di Milano Piazza L. da Vinci,
32, I-20133 Milano,
Italy
E-mail: *pianosi@elet.polimi.it*

INTRODUCTION

Many decision-making problems involving the management of water resource systems have two distinctive features: (i) they involve a sequence of decisions made at discrete time instants over a system affected by stochastic inputs, and (ii) each decision has an immediate effect, but also influences the long-term dynamics of a wide range of multiple environmental, social and economic issues (see e.g. Hipel (1992)). In other and more rigorous terms, these problems concern the sequential multi-objective optimization of discrete-time dynamic systems affected by stochastic disturbances. For these problems, multi-objective Markov decision processes (MOMDPs) provide a powerful theoretical and operational framework for analysis and resolution (White 1982, 1988).

The conventional approach to solving MOMDPs is to convert the problems themselves into a single-objective optimization, combining all the objective functions into a single functional form that can be handled by any standard single-objective optimization method. A well-known option is the linear combination of the objectives, known as the *weighted-sum method*: many single-objective problems associated with different combination coefficients (weights) are solved and a subset of the theoretical Pareto-optimal solutions (operating policies) to the multi-objective problem is obtained. Yet, with the growth in the number of objectives, the repetitions of single-objective problems scale exponentially, thus making the approach computationally intensive, if not prohibitive.

Multi-objective reinforcement learning (MORL) has recently emerged as a potentially interesting alternative to the above two approaches to efficiently design

doi: 10.2166/hydro.2013.169

multi-objective operating policies (Gábor *et al.* 1998; Mannor & Shimkin 2004; Natarajan & Tadepalli 2005; Barrett & Narayanan 2008; Vamplew *et al.* 2009; Lizotte *et al.* 2010). For a recent survey of MORL, we refer the reader to Vamplew *et al.* (2010).

In this paper, we present a novel MORL algorithm, which is a multi-objective extension of the single-objective fitted Q-iteration (FQI) algorithm (Ernst *et al.* 2005; Castelletti *et al.* 2010). The key idea of multi-objective FQI (MOFQI) is to enlarge the continuous approximation of the action-value function, which FQI performs over the state-control space, and also the weight space by including new variables, the weights, within the arguments of the action-value function. As a result, MOFQI is able to approximate, with a single learning process, all the optimal policies associated with any convex linear combination of the objectives. MOFQI inherits the benefits of the FQI algorithm, including the possibility of using both parametric and non-parametric regression algorithms. In this paper, we use *extremely randomized trees* (Extra Trees) to approximate the action-value function because of their capability both to reduce the bias with respect to traditional linear approximation schemes and to keep the variance low with respect to other non-parametric techniques. In particular, we propose a modified version of the original Extra Trees (Geurts *et al.* 2006), where an improved pre-pruning criterion is introduced to deal with functions that need different degrees of generalization over the regression domain, as happens when applying MOFQI.

The properties of MOFQI are first evaluated by application to a numerical test case study of a two-objective reservoir system. Pareto-optimal operating policies designed by tree-based MOFQI are compared with those generated by several runs of tree-based FQI for different linear combinations of the objectives, and the nearly optimal solution provided by stochastic dynamic programming (SDP). The potential of the proposed MOFQI approach is subsequently explored by application to a real-world case study, the operation of Hoa Binh reservoir in Northern Vietnam.

The remainder of the paper is organized as follows: in the next section, the formulation of Markov decision processes (MDPs) and MOMDPs is presented with reference to optimal reservoir operation. Then, we describe the FQI and MOFQI algorithms and the Extra Trees models, both in the original and the modified version introduced for the first time in this paper. The subsequent sections present the comparative analysis of iterated FQI and MOFQI applied to the numerical test case study and the real-world case study. The paper closes with concluding remarks and directions for future research.

MOMDPS

MDPs have been used since the early 1950s for the planning and operation of reservoir systems because natural reservoir inflows can be modeled using Markov stochastic processes and the mass conservation equations describing the state (reservoir storage) transitions are very similar to those found in inventory theory, where MDPs have been originally developed (e.g. Wang & Adams 1986; Lamond & Boukhtouta 2002; Turgeon 2005).

MDPs

A discrete-time continuous MDP is described as a tuple $\langle X, U, P, R, \gamma, \mu \rangle$, where $X \subset \mathbb{R}^n$ is the continuous state space, $U \subset \mathbb{R}^m$ is the continuous action space, $P(y|x, u)$ is the transition model that defines the transition density between state x and y under action u, $R(x, u, y)$ is a reward function that specifies the immediate reward when state y is reached by taking action u in state x, $\gamma \in [0,1)$ is a discount factor, and μ is the initial-state distribution from which the initial state is drawn. For a reservoir system composed of several reservoirs, the m action variables are the release decisions at each reservoir outlet (e.g. taken on a daily basis), the n state variables are the storages in each reservoir, possibly augmented with any other state variable required by the catchment models, the transition density $P(y|x, u)$ is the probability of the next storage (or augmented state) y, given the current state x and action u resulting from the probability density function of the residual of the catchment model, and $R(x, u, y)$ is the immediate reward (e.g. daily hydropower production) associated with the operation objective. An operating policy is a mapping from states (e.g. storage) to controls (release volumes), $\pi: X \rightarrow U$, so that $u = \pi(x)$.

The *value* $V^\pi(x)$ of state x under the operating policy π is the expected return when starting in x and following π thereafter:

$$V^\pi(x) = \int_X (R(x, \pi(x), y) + \gamma V^\pi(y)) P(\mathrm{d}y|x, \pi(x))$$

Given the initial-state distribution μ, the expected return of operating policy π is defined as:

$$J^\pi_\mu = \int_X V^\pi(x)\mu(\mathrm{d}x) \quad (1)$$

To simplify the notation, in the following, subscript μ will be omitted where possible.

Solving an MDP means finding an operating policy that maximizes the value function V in each state. The optimal value function is the solution of the Bellman optimality equation (Bellman 1957). For control purposes, it is better to consider action values, i.e. the value of taking action u in state x and following a policy π thereafter. The optimal *action-value function* is the solution of the following reformulation of the Bellman equation:

$$Q^*(x, u) = \int_X \left(R(x, u, y) + \gamma \max_{u' \in U} Q^*(y, u') \right) P(\mathrm{d}y|x, u)$$

Given the optimal action-value function, the associated optimal operating policy is the one that takes in each state the action with the highest value (greedy policy), $\pi^*(x) = \arg\max_{u \in U} Q^*(x, u)$.

MOMDPs

MOMDPs are an extension of the MDP model, where several reward functions are defined, one for each objective. MOMDPs are a particularly suitable formalism to model multi-purpose reservoir systems that are generally operated to balance multiple, often conflicting objectives (Castelletti *et al.* 2008).

Formally, an MOMDP is described as a tuple $\langle X, U, P, \mathbf{R}, \gamma, \mu \rangle$, where $\mathbf{R} = [R_1, \ldots, R_q]$ is a q-dimensional vector of reward functions (for instance, daily hydropower production, water supply for irrigation or civil use, etc.). In MOMPDs, any policy π is associated with q value functions $\mathbf{V}^\pi = [V^\pi_1, \ldots, V^\pi_q]$, where V^π_i is defined as:

$$V^\pi_i(x) = \int_X (R_i(x, \pi(x), y) + \gamma_i V^\pi_i(y)) P(\mathrm{d}y|x, \pi(x))$$

Given the initial-state distribution μ and the vector of value functions $\mathbf{V}^\pi$ for policy π, it is possible (as done in Equation (1)) to compute the vector of expected returns $\mathbf{J}^\pi = [J^\pi_1, \ldots, J^\pi_q]$.

Solving an MOMDP means finding the set of Pareto-optimal policies Π^*, which maps onto the so-called Pareto frontier $\mathcal{J}^* = \{J^{\pi^*} | \pi^* \in \Pi^*\}$.

The traditional approach to MOMDPs is to transform them into multiple single-objective problems by combining the different rewards with some *scalarizing function* $\psi: \mathbb{R}^q \to \mathbb{R}$ (Perny & Weng 2010). The most straightforward choice for ψ is a convex combination of the objectives (weighted-sum method) using weights $\boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_q] \in \Lambda^{q-1}$, where Λ^{q-1} is the unit $(q-1)$-dimensional simplex (so that $\sum_{i=1}^q \lambda_i = 1$ and $\lambda_i \geq 0$ for all i). To simplify notation, in the following, we drop the superscript $q-1$ from the simplex symbol when there is no risk of confusion. Each vector of weights $\boldsymbol{\lambda}$ defines a single-objective MDP with the following reward function:

$$R_{\boldsymbol{\lambda}}(x, u, y) = \boldsymbol{\lambda}' \cdot \mathbf{R} = \sum_{i=1}^q \lambda_i R_i(x, u, y)$$

By linearity of the mathematical expectation and the weighted sum, the expected return of policy π with weight vector $\boldsymbol{\lambda}$ is: $J^\pi_{\boldsymbol{\lambda}} = \boldsymbol{\lambda}' \cdot \mathbf{J}^\pi$. Since all optimal policies of such single-objective MDPs are provably Pareto-optimal solutions of the original MOMDP (Chatterjee *et al.* 2006), the Pareto frontier can be estimated by computing the set of expected-return vectors obtained for all possible values of $\boldsymbol{\lambda}$.

An approximation of the set of Pareto-optimal policies, and the associated Pareto frontier, is obtained by considering a finite number of sample weight combinations $\hat{\Lambda} \subset \Lambda$. The more weight combinations that are evaluated, the more accurate the frontier approximation,

but also the longer the computational time required. The advantage is that each single-objective MDP can be solved by standard single-objective methods like dynamic programming and linear programming, or reinforcement learning (RL) techniques. A major limitation is that if the Pareto frontier has concave regions, Pareto-optimal solutions lying in such regions cannot be found by the weighted-sum method.

TREE-BASED FQI

RL algorithms are receiving growing attention in reservoir operation design (e.g. Lee & Labadie 2007; Castelletti *et al.* 2010) for their ability in alleviating the dual dynamic programming's curse of dimensionality (Bellman 1957) and modeling (Tsitsiklis & Roy 1996). RL allows the computation of $Q^*(x,u)$ by directly interacting with the environment (online learning) by a trial-and-error process or, more interestingly for natural systems applications, on the basis of experience samples previously collected from the system, which enables the solution to the optimization problem to be learned offline (batch mode).

Fitted Q-iteration

The batch-mode RL approach we adopt in this paper aims at determining the best operating policy given a set of N tuples $D = \{\langle x_i, u_i, y_i, r_i\rangle\}_{1\le i\le N}$, where $r_i = R(x_i, u_i, y_i)$, previously collected according to a given sampling strategy.

In particular, good results have been recently achieved by the FQI algorithm (Ernst *et al.* 2005), a model-free approach derived from the *fitted value iteration* approach (Gordon 1999). The underlying idea of FQI is to reformulate the RL problem as a sequence of non-linear regression problems.

Given the dataset D, in the first iteration of the algorithm, for each tuple $\langle x_i, u_i, y_i, r_i\rangle$, the corresponding training pair is set as having (x_i, u_i) as input and r_i as output (i.e. $(x_i, u_i) \to r_i$), and the goal is to use a regression algorithm to estimate a function that approximates the expected immediate reward $Q_1(x, u) = E_{y\sim P(\cdot|x,u)}[R(x, u, y)]$. The second iteration, based on the approximation $\hat{Q}_1$ of the Q_1-function, extends the optimization horizon one step further, by estimating function $\hat{Q}_2$ through regression on the following training dataset:

Algorithm 1 FQI algorithm

input: $\mathrm{D} = \{\langle x_i, u_i, y_i, r_i\rangle\}_{1\le i\le N}$, a regression algorithm $\mathcal{R}$, the number of iterations L

initialize: $\hat{Q}_0(x,u) = 0, \forall x \in X, \forall u \in U$

for $k = 1$ to L **do**

$$T_k = \left\{ \left[(x_i, u_i) \to r_i + \gamma \max_{u'\in U} \hat{Q}_{k-1}(y_i, u') \right] \right\}_{1\le i\le N}$$

$$\hat{Q}_k = \mathcal{R}(T_k)$$

end for

return $\hat{\pi}^*_L(x) = \arg\max_{u\in U} \hat{Q}_L(x,u)$

$$T_2 = \left\{ \left[(x_i, u_i) \to r_i + \gamma \max_{u\in U} \hat{Q}_1(y_i, u) \right] \right\}_{1\le i\le N}$$

By proceeding in the same way, at the kth iteration, the approximation $\hat{Q}_{k-1}$ is used to compute the optimal action-value function at horizon k. The procedure iterates until the action-value function converges or a maximum number of iterations is reached (see Ernst *et al.* (2005) for a discussion about the stopping condition and the convergence properties of the algorithm). The FQI algorithm is summarized in Algorithm 1.

Several studies have reported very good results with a wide range of approximation techniques: kernel-based regressors (Ormoneit & Sen 2002), tree-based regressors (Ernst *et al.* 2005), neural networks (Riedmiller 2005), CMAC (Timmer & Riedmiller 2007) and advantage weighted regression (Neumann & Peters 2009). All these works show that batch-mode RL algorithms effectively exploit the information contained in the collected samples, so that very good performance can be achieved, even using small datasets. The size of the dataset is a key factor for many real-world applications, since collecting a large amount of data from a real system may be considerably expensive. Finally note that, while in stochastic approximation algorithms (e.g. Tsitsiklis & Roy 1996) only parametric function approximators can be used, the batch nature of FQI allows for also using non-parametric

regression. In this work, we used extremely randomized trees (Extra Trees) (Geurts *et al.* 2006), a non-parametric, tree-based ensemble method for supervised problems, that offers several advantages, such as robustness against irrelevant features, good trade-off between bias and variance, and high computational efficiency. As shown in Ernst *et al.* (2005), Castelletti *et al.* (2010), Bonarini *et al.* (2008) and Tognetti *et al.* (2009), Extra Trees performs provably well when used in the FQI algorithm, even with relatively small datasets.

Extremely randomized trees

Tree-based methods are non-parametric supervised learning methods based on the idea of decision-tree models, which are tree-like structures representing a cascade of rules leading to numerical values (Breiman *et al.* 1984). These structures, composed of decision nodes, branches and leaves, are obtained by partitioning, according to a certain splitting criterion, the set of the input variables into a series of subsets, until either the numerical values belonging to each subset vary just slightly or only few elements remain. When the splitting process is over, the branches represent the hierarchical structure of the subset partitions, while the leaves are the finest subsets associated with the terminal branches. Each leaf is finally associated with a numerical value.

In this study, we explore Extra Trees (Geurts *et al.* 2006), which randomizes (totally or partially) both the input variable and the cut-point selection when splitting a node, and creates an ensemble of trees to compensate for the randomization, via averaging of the constituent tree outcomes. The Extra Trees building algorithm grows an ensemble of M trees. Nodes are split using the following rule: K alternative cut directions (input variables) are randomly selected and, for each one, a random cut-point is chosen; a score is then associated with each cut direction, and the one maximizing the score is adopted to split the node. The algorithm stops partitioning a node if its cardinality is smaller than $n_{\min}$, and the node is therefore a leaf. Each leaf is assigned a value obtained as the average of the outputs associated with the inputs falling in that leaf. The estimates produced by the M trees are finally aggregated with an arithmetic average. The rationale behind the approach is that the combined use of randomization and ensemble averaging provides more effective variance reduction than other randomization methods, while minimizing the bias of the final estimate. Extra Trees is thus characterized by three parameters (i.e. K, $n_{\min}$ and M), whose values can be fixed on the basis of empirical evaluations.

TREE-BASED MOFQI

In this paper, we propose to extend the FQI algorithm to multi-objective problems, thus producing the MOFQI. The idea is to enlarge the state space X with the unit $(q-1)$-dimensional simplex Λ^{q-1}, in order to consider different weight combinations of the q objectives. The dataset used in MOFQI thus takes the form:

$$D_{\mathrm{MO}} = \{\langle x_i, \boldsymbol{\lambda}_i, u_i, y_i, \boldsymbol{\lambda}_i, \boldsymbol{\lambda}_i' \cdot \mathbf{R}(x_i, u_i, y_i)\rangle\}_{1 \le i \le N_{\mathrm{MO}}} \tag{2}$$

Note that the weight vector $\boldsymbol{\lambda}_i$ is regarded as a further state variable whose transition density always returns the same value $\boldsymbol{\lambda}_i$. The result of regression on dataset (2) is an optimal action-value function parameterized by $\boldsymbol{\lambda}$: $\hat{Q}^*(x, \boldsymbol{\lambda}, u)$. In this way, it is possible to generalize information even over the weight space and, after a single training process, MOFQI learns a continuous approximation of the optimal policy over the weight space:

$$\hat{\pi}^*_{\boldsymbol{\lambda}}(x) = \arg\max_{u \in U} \hat{Q}(x, \boldsymbol{\lambda}, u)$$

from which the approximate Pareto frontier can be derived.

The state space wherein MOFQI operates has a higher dimension than the corresponding single-objective algorithm. As a consequence, to obtain similar performances, MOFQI, in general, requires more tuples than the ones used by FQI for a given weight vector (i.e. $N_{\mathrm{MO}} > N$), but, exploiting the generalization over the weight space, fewer tuples than the ones needed by reiterated application of FQI. Furthermore, the cost of generating the experience samples $< x_i,\ u_i,\ y_i >$ (by either data collection or model simulation) is the same for MOFQI and FQI. In fact, the dataset D_{MO} can be generated from the dataset D of FQI

without collecting new experience samples from the system, but simply reusing the same samples many times with different, randomly generated weight values. The number of tuples in the MOFQI dataset D_{MO} will thus be given by the number of tuples in D multiplied by a factor k equal to the number of weight samples, $N_{\mathrm{MO}} = k \times N$.

Improved Extra Trees

In Extra Trees, the degree of generalization is controlled by the $n_{\min}$ parameter, which determines the minimum number of samples to split a node. As a result, the size of the leaves (which determines the degree of generalization) will be approximately the same all over the regression domain, while, in general, different regions may require different levels of regularization. This is particularly the case with the action-value function that must be estimated by MOFQI, where different regions of the weight space correspond to different combinations of the objectives, which may vary more or less rapidly with the state and action values. Since using a fixed generalization degree for all the possible weight-state-action combinations is unlikely to yield good results, in this paper, we propose a pre-pruning criterion that adaptively determines whether or not to split a node based on the input/output data under consideration, increasing the leaf size when the output variability decreases and vice versa.

The criterion works as follows. Given a dataset $D = \{\langle i^l, o^l \rangle\}_{1 \le l \le N}$, a cut that partitions D into two distinct subsets, D_l and D_r, with N_l and N_r samples respectively, is evaluated on the basis of the probability that the samples of the two subsets have been generated by two distributions with different means. This score actually requires the computation of a two-tailed Student's t-test in which the null hypothesis is that the two distributions have the same mean; the t-statistic to test whether the two distribution means are different can be calculated as follows:

$$t_{D_l,D_r} = \frac{|\hat{\mu}(o|D_l) - \hat{\mu}(o|D_r)|}{\sqrt{\dfrac{\hat{\sigma}^2(o|D_l)}{N_l} + \dfrac{\hat{\sigma}^2(o|D_r)}{N_r}}}$$

where $\hat{\mu}(o|D_l)$ and $\hat{\mu}(o|D_r)$ are the sample means of the output values of the two subsets, and $\hat{\sigma}^2(o|D_l)$ and $\hat{\sigma}^2(o|D_r)$ are the sample variances. The distribution of the test statistic is approximated as being an ordinary Student's t-distribution with the degrees of freedom ν calculated using:

$$\nu_{D_l,D_r} = \frac{\left(\dfrac{\hat{\sigma}^2(o|D_l)}{N_l} + \dfrac{\hat{\sigma}^2(o|D_r)}{N_r}\right)^2}{\left(\dfrac{\hat{\sigma}^2(o|D_l)}{N_l}\right)^2 \dfrac{1}{N_l - 1} + \left(\dfrac{\hat{\sigma}^2(o|D_r)}{N_r}\right)^2 \dfrac{1}{N_r - 1}}$$

Finally, the score function is defined as the two-tailed p-value:

$$\mathrm{Score}(D_l, D_r) = 2(1 - F(t_{D_l,D_r}|\nu_{D_l,D_r}))$$

where $F(x|\nu)$ is the Student's t cumulative distribution function in x of a t-distribution with ν degrees of freedom. The larger the score, the higher the probability that the means of the outputs of the two partitions are different; among the K random cuts, the one with the best score is chosen.

The split process is stopped when no cut scores above a predefined threshold τ. The idea is that it is not worth choosing a cut direction which produces two subsets that have a probability larger than $1 - \tau$ of sharing the same mean. In other words, no split occurs when the null hypothesis (i.e. the two partitions have the same mean) is accepted for all the cuts at $(1 - \tau)\%$ confidence level. This stop criterion generates trees with unequal size leaves: larger leaves in regions of the input space where the function to be approximated is nearly constant, and smaller leaves where the output values rapidly change.

TEST CASE STUDY

To evaluate the proposed MOFQI approach, we consider a numerical case study of a two-objective reservoir. The scalar state variable x represents the water volume stored in the reservoir, the action u is the release decision, and the state-transition model that provides the future state y is the mass balance equation:

$$y = x + \varepsilon - \max(\underline{u}, \min(\bar{u}, u)) \tag{3a}$$

where ε is the reservoir inflow, generated by a white noise process with normal distribution $\varepsilon \sim N(40,100)$, and $\underline{u}$ and $\bar{u}$ are the minimum and maximum feasible release associated with the storage x according to the relations:

$$\bar{u} = x \quad \text{and} \quad \underline{u} = \max(x - 100, 0) \tag{3b}$$

The reservoir operation must balance two conflicting objectives: to control flooding along the reservoir shores and to supply water for irrigation. The reward associated with the flood objective is the negative of the cost due to the excess level with respect to a flooding threshold $\bar{h}: R_1(x,u,y) = -\max(h - \bar{h}, 0)$, where h is the reservoir level, given by the storage x divided by the reservoir surface S (in the following experiments, $S = 1$ and $\bar{h} = 50$). The reward function for the irrigation objective is the negative of the deficit in the water supply with respect to the water demand $\bar{\rho}: R_2(x,u,y) = -\max(\bar{\rho} - \rho, 0)$, where $\rho = \max(\underline{u}, \min(\bar{u}, u))$ is the actual release from the reservoir and the water demand $\bar{\rho}$ is 50. Given the non-economic nature of the above performance indicators and since the MOMDP can be solved optimizing over a finite-time horizon, we set the discount factor γ to 1 for all the objectives.

Experimental setup for SDP

As a reference solution, we consider the nearly optimal solutions computed by SDP at 11 different values of the weight vector, namely $\boldsymbol{\lambda} = [\lambda_1, 1 - \lambda_1]$ with $\lambda_1 \in \hat{\Lambda} = \{0.0, 0.1, 0.2, \ldots, 1.0\}$. For each $\boldsymbol{\lambda}$, a single-objective problem is defined and solved by SDP. Although this solution is still an approximation due to the discretization of x, u and y, the dense discretization grid used allows for achieving near-optimal performance.

Experimental setup for FQI and MOFQI

The simple reservoir model considered in this case study allows us to feed both FQI and MOFQI algorithms with experience samples drawn uniformly random from the state-action space; for each sample, the next state and reward values are obtained from the generative model (3). The only difference between the two datasets is that in the multi-objective case, the state space has one more dimension that represents the value of weight λ_1. As a result, the dataset takes the shape presented in Equation (2).

In both FQI and MOFQI, the optimal action-value function was approximated by Extra Trees with M=100 trees and a number K of alternative cut directions equal to the number of state and actions variables (2 for FQI and 3 for MOFQI). The threshold τ for computing the score function described in the section on improved Extra Trees was tuned by trial and error: the results shown in the following were obtained by setting $\tau = 0.98$; however, other values in the range [0.8,1) provided similar performances. The number of algorithm iterations L is set to 10, which is the same number of iterations performed by SDP.

Performance evaluation methodology

The operating policies designed by SDP and RL algorithms are evaluated using Monte Carlo simulations. Each simulation consists of 100 steps and is repeated 10 times (using independent realizations of inflow trajectories) under 10 different initial states chosen uniformly random over the state space X. As a consequence, the performance of each policy π is evaluated on 100 scenarios of 100 steps. The result is a performance vector $\bar{\mathbf{J}}^{\pi}$ whose components are the average immediate rewards associated with each objective under π.

To compare the different policies, we use a performance metric L similar to the one presented in Hansen & Jaszkiewicz (1998). It is based on pair comparison of the aggregate objective value $J^{\pi}_{\boldsymbol{\lambda}} = \boldsymbol{\lambda}' \cdot \mathbf{J}^{\pi}$ provided by the policy π under examination and the optimal value $J^{\pi^*}_{\boldsymbol{\lambda}}$, at all possible $\boldsymbol{\lambda}$. Differently from Hansen & Jaszkiewicz (1998), in our metric, the differences $J^{\pi^*}_{\boldsymbol{\lambda}} - J^{\pi}_{\boldsymbol{\lambda}}$ are normalized by the range of variation of the optimal function values, i.e.,

$$\Delta J^*_{\boldsymbol{\lambda}} = \sum_{i=1}^{q} \boldsymbol{\lambda}_i \left(\max_{\boldsymbol{\lambda}' \in \Lambda} J_i^{\pi^*_{\boldsymbol{\lambda}'}} - \min_{\boldsymbol{\lambda}' \in \Lambda} J_i^{\pi^*_{\boldsymbol{\lambda}'}} \right)$$

In the proposed test case study, the solutions by SDP are used as the reference Pareto-optimal policies π^*, and the loss

L takes the form:

$$L = \frac{1}{N_\lambda} \sum_{i=1}^{N_\lambda} \frac{J_{\lambda_i}^{\pi^{SDP}} - J_{\lambda_i}^{\pi}}{\Delta J_\lambda^{SDP}} \quad (4)$$

where N_λ is the number of evaluated weight vectors (11). According to this (non-negative) measure, the smaller the loss value, the better the approximation of the Pareto frontier.

Experimental results

Figure 1 shows the Pareto frontiers obtained by Monte Carlo simulations (as explained in the previous section) of the non-dominated solutions produced by the different algorithms. The approximation of the Pareto frontier produced by FQI using 10,000 tuples is similar (if not slightly worse) to the one of MOFQI when using the same 10,000 samples to build a dataset with 20,000 tuples. Nonetheless, from the computational perspective, it is worth recalling that FQI has to solve as many single-objective learning problems (using 10,000 tuples for each of them) as the number of weight samples to be evaluated.

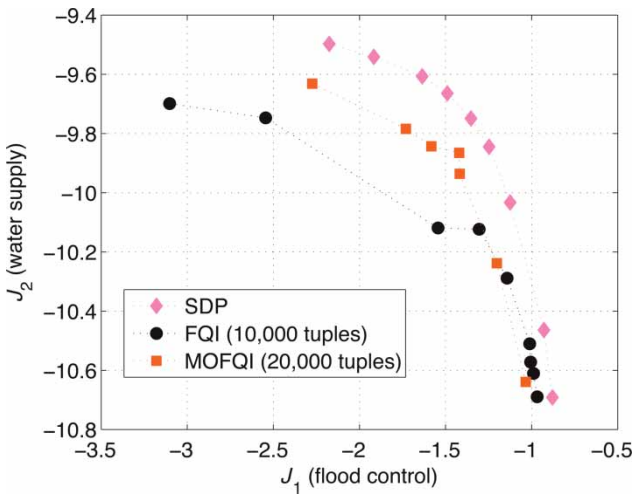

Figure 1 | Test case study: comparison between the approximated Pareto frontiers obtained with SDP, FQI and MOFQI for 11 weight values.

As MOFQI has been proposed as an alternative approach to the repeated application of single-objective FQI following the weighting method, we will discuss in what conditions MOFQI outperforms repeated FQI. Table 1 shows the loss L (as defined in Equation (4)) for FQI and MOFQI with respect to SDP, as the number of experience samples varies between 1000 and 10,000 and the multiplication factor k varies from 1 to 5. In the table, we have reported the average loss over 10 runs and the related standard deviation. As expected, the loss reduces for both the approaches as the number of experience samples increases. Also, the number of tuples required by MOFQI is larger than the one needed by FQI in order for the accuracy to be the same. More interestingly, Table 1 shows that, starting from the same set of experience samples, MOFQI performs as well as FQI (or even better) when its training dataset is about five times larger than the one of FQI. From the computational perspective, while the training time of FQI grows linearly with the number of weight combinations to be evaluated, MOFQI is independent of such a number. So, we can conclude that, to have a dense approximation of the Pareto frontier, MOFQI is

Table 1 | Test case study: average loss L over 10 runs with related standard deviation for FQI and MOFQI (with different multiplication factors k) with respect to SDP when different sample sizes are considered

		MOFQI				
Samples	**FQI**	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$
1,000	0.332 ± 0.008	0.603 ± 0.034	0.539 ± 0.027	0.482 ± 0.021	0.406 ± 0.017	0.329 ± 0.017
2,000	0.255 ± 0.004	0.508 ± 0.024	0.480 ± 0.025	0.446 ± 0.018	0.383 ± 0.021	0.287 ± 0.009
3,000	0.249 ± 0.007	0.506 ± 0.023	0.427 ± 0.015	0.365 ± 0.021	0.280 ± 0.015	0.251 ± 0.010
4,000	0.231 ± 0.005	0.453 ± 0.015	0.368 ± 0.008	0.302 ± 0.016	0.256 ± 0.013	0.232 ± 0.010
5,000	0.217 ± 0.007	0.361 ± 0.010	0.339 ± 0.024	0.256 ± 0.010	0.243 ± 0.009	0.221 ± 0.015
10,000	0.191 ± 0.010	0.302 ± 0.029	0.219 ± 0.012	0.195 ± 0.009	0.191 ± 0.008	0.187 ± 0.009

computationally preferable to repeated FQI. In the proposed test case, this is true as long as more than five weight combinations are required.

HOA BINH RESERVOIR CASE STUDY

The second case study considered in this paper is a real-world system, Hoa Binh reservoir in Northern Vietnam (Figure 2). The reservoir has a surface area of about 198 km^2 and an active storage of 6×10^9 m^3. The main objectives of reservoir operation are hydropower production (the plant has a capacity of 1,920 MW and produces more than 7,000 GWh per year) and flood mitigation in the downstream city of Hanoi. The reservoir and the downstream river network are modeled by a combination of conceptual and data-driven models with a daily resolution time step. A detailed description of the system and associated model can be found in Castelletti *et al.* (2011); here, we will provide a brief description of the problem formulation as an MOMDP.

The problem can be modeled with two state variables, the reservoir storage and the corresponding day of the year. Since the system can be described as cyclostationary with period T=365 days, we can obtain a stationary MDP by enlarging the state space with the time variable (Castelletti *et al.* 2010). The action variable u is the release decision for the next 24 hours, and the future state is the reservoir storage y on the day after, estimated by a mass balance equation such as (3a), and t+1. The minimum and maximum feasible release $\underline{u}$ and $\bar{u}$ are computed based on the storage and inflow values, and taking into account the rating curves of the bottom and intermediate gates and the spillways. The reward associated with the hydropower objective (R_1) is the value (ranging from 0 to 1) of the daily hydropower production, the one of flood control (R_2) is the cost of floods changed in sign, i.e. it ranges from -1 (maximum cost) to 0 (minimum cost), and it is estimated by a non-linear function of the water level in Hanoi.

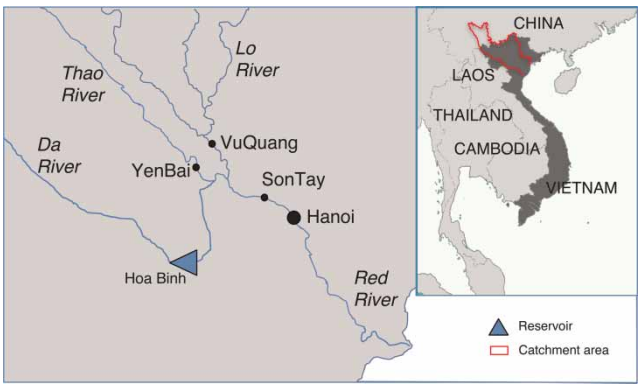

Figure 2 | The Hoa Binh reservoir water system in Northern Vietnam.

Experimental setup for FQI and MOFQI

Time series of measured flows over the period 1957–1978, together with the simulation model, were used to generate the dataset for FQI and MOFQI. For each day in the time series, 10 storage and action values were randomly sampled: the storage is drawn uniformly over the range 3.7151×10^9 to 1.0415×10^{10} m^3, while the action is randomly chosen from a finite set of 20 values of daily average release, ranging from 0 to 13,000 m^3/s. The total number of experience samples is 73,650. First, FQI is repeatedly applied under six different values of the weight λ_1. Then, MOFQI is applied once, using the enriched dataset of Equation (2). Such a dataset is obtained by associating each original experience sample with k random sampled weight values. In this experiment, $k-2$ weight values were randomly chosen, while the remaining two were set to $\lambda_1 = 1$ (hydropower only) and $\lambda_1 = 0$ (flood control only). In the following, results relevant to the case $k = 3$ and $k = 7$ will be compared.

In all the optimization experiments, Extra Trees was used to approximate the state-action-value function, with $M = 100$ trees and a number of alternative cut directions K equal to 3 for FQI and 4 for MOFQI. After some trial and error, the threshold τ for computing the score function described in the section on improved Extra Trees was set to 0.9 (again, threshold values higher than 0.8 all provided comparable results). The number of algorithm iterations L was decided based on the analysis of the system functioning. From the definition of the hydropower objective, it follows that the optimal reservoir operation should allocate the hydropower production in the period of maximum energy value, i.e. from April to June. The storage to sustain such production must have been created in the previous flood season

(August–September of the previous year), that is, around 200 days before. The optimal operation horizon for the flood objective, instead, is much shorter since the time required to empty the reservoir in anticipation of a big flood is around 10 days. Since the number of algorithm iterations should correspond to the maximum length of the operation horizon for the considered objectives, L was set to 200.

Experimental results

All the operating policies were simulated under the time series of observed reservoir inflows and Lo and Thao discharges over the time horizon 1957–1978 (calibration dataset) and the horizon 1995–2004 (validation dataset). The performances over the validation dataset can be compared with the historical operation (average reward values: $J_1 = 0.292$, $J_2 = -0.129$), as the reservoir construction was finished in 1989 and its filling was completed in 1994.

Table 2 reports the objective values over the calibration dataset, Table 3 those of the validation dataset. Figure 3 shows the policy performances over the validation dataset. First, it can be noticed that the performances of MOFQI are obviously higher with a larger multiplication factor k. More interestingly, the comparison of FQI and MOFQI (with $k = 7$) for λ_1 between 0 and 0.5 (i.e. flood control is more relevant than hydropower production) shows that while FQI outperforms MOFQI over the calibration dataset, it is outperformed over the validation dataset. The result is consistent with the findings of Caruana (1997) for the multi-task learning algorithm: learning the solutions of multiple tasks simultaneously, as in MOFQI, improves generalization abilities, while learning on a single task, as in reiterate FQI, is more likely to overfit the data.

For $\lambda_1 > 0.5$, it can be noticed that MOFQI performs rather poorly with respect to FQI. The reason is that, as stated above, the maximization of the hydropower objective requires a longer time horizon (around 200 days) than the flood control objective (about 30 days). Therefore, as the number L of algorithm iterations is increased to properly approximate the 'component' of the action-value function related to the hydropower maximization, the approximation errors in the flood control 'component' accumulate and override the overall quality of the function approximation. To reduce such approximation errors, the number of tuples must be increased, i.e. a much larger multiplication factor k must be used for MOFQI.

CONCLUSIONS

In this paper, we presented an extension of batch-mode RL to solve MOMDP problems. The MOFQI algorithm we proposed relies on the continuous approximation of the action-value function over the weight space and the subsequent approximation of the Pareto frontier by simply sampling this function for a desired number of weight values.

Experience gained from experiments on a numerical test case of a multi-purpose water reservoir shows that MOFQI provides an accurate approximation of the Pareto frontier as computed with several repetitions of SDP for different values of the weight vector. In addition, MOFQI becomes

Table 2 | Hoa Binh case study: objective values (hyd = hydropower; flo = flooding) over the calibration dataset 1957–1978

	FQI			MOFQI ($k = 3$)			MOFQI ($k = 7$)		
λ_1	hyd	flo	J	hyd	flo	J	hyd	flo	J
0.0	0.244	−0.043	−0.043	0.222	−0.043	−0.043	0.206	−0.041	−0.041
0.1	0.269	−0.046	−0.015	0.225	−0.062	−0.033	0.207	−0.050	−0.024
0.5	0.284	−0.056	0.114	0.232	−0.068	0.082	0.247	−0.057	0.095
0.7	0.300	−0.057	0.194	0.236	−0.075	0.143	0.260	−0.064	0.163
0.9	0.326	−0.076	0.286	0.262	−0.088	0.228	0.283	−0.077	0.247
1.0	0.328	−0.100	0.328	0.326	−0.100	0.326	0.328	−0.093	0.328

Table 3 | Hoa Binh case study: objective values (hyd = hydropower; flo = flooding) over the validation dataset 1995–2004

	FQI			MOFQI ($k = 3$)			MOFQI ($k = 7$)		
λ_1	hyd	flo	J	hyd	flo	J	hyd	flo	J
0.0	0.244	−0.106	−0.106	0.215	−0.076	−0.076	0.210	−0.076	−0.076
0.1	0.278	−0.110	−0.071	0.223	−0.091	−0.060	0.221	−0.083	−0.053
0.5	0.279	−0.119	0.080	0.234	−0.102	0.066	0.244	−0.083	0.081
0.7	0.299	−0.124	0.173	0.250	−0.181	0.121	0.272	−0.174	0.138
0.9	0.335	−0.184	0.284	0.268	−0.194	0.223	0.278	−0.178	0.233
1.0	0.338	−0.203	0.339	0.336	−0.201	0.337	0.335	−0.195	0.336

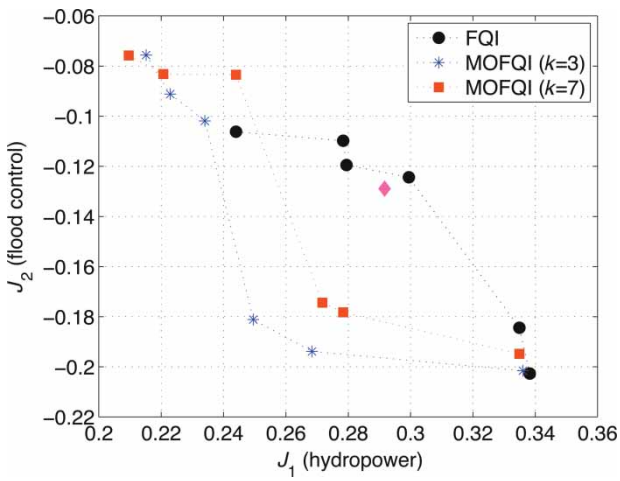

Figure 3 | Hoa Binh case study: comparison between the approximated Pareto frontiers obtained with FQI and MOFQI with different multiplication factors (validation dataset). The diamond is the historical operation performance.

computationally more efficient than the reiterated application of its single-objective twin FQI, when more than five points are used to approximate the Pareto frontier. This result is obtained on a bi-objective control problem. Supposedly, the advantage over FQI grows as the number of objectives increases, thus making the approach particularly suitable to many real-world problems including two, three or more objectives (the so-called many-objective problems), such as the operation of multi-purpose water reservoir networks, and, generally, any other water resource management problem involving various environmental, economic and social issues.

The Hoa Binh reservoir case study was used to evaluate the benefits from the applicability of MOFQI on a real-world sized problem. Performance of the operating policies designed by MOFQI and FQI were compared for different weight combinations via simulation using inflow data for a historical period not included in the policy design. The approximation of the Pareto frontier produced by FQI using 73,650 tuples dominates the one by MOFQI when using the same 73,650 samples to build a dataset with $3 \times 73{,}650$ tuples. By enlarging the dataset to $7 \times 73{,}650$ tuples, MOFQI provides comparable results, in terms of aggregated objective value, to those by FQI, especially at the extreme ends of the weight range, while providing slightly poorer performances for more balanced weight combinations, probably due to the mismatch in time scales of the two considered objectives. However, the advantage of MOFQI is that, with no additional computation cost, the approximation of the Pareto front by MOFQI can be made much more dense, in principle nearly continuous, thus allowing for a more accurate exploration of trade-offs and, correspondingly, for better informed decision making.

The current implementation of MOFQI has two main limitations, which will be the subject of future study. First, the computational cost associated with the use of extremely randomized trees to approximate the action-value function becomes particularly high as the number of tuples increases, becoming unfeasible with datasets with more than 1×10^6 tuples. A parallel implementation of the tree building algorithm would reduce by about $1/M$ the time required by MOFQI to compute the Pareto front. Alternatively, more efficient regression schemes (e.g. artificial neural networks or kernel regression) could be explored. Second, a more extensive evaluation of the advantage of MOFQI against its

single-objective twin for a higher number of objectives is required to generalize the analysis of the algorithm's computational requirements, also compared with the evolutionary multi-objectives method.

REFERENCES

Barrett, L. & Narayanan, S. 2008 Learning all optimal policies with multiple criteria. In: *Proceedings of 25th International Conference on Machine Learning (ICML 2008)*, Helsinki, Finland, 5–9 July, pp. 41–47.

Bellman, R. E. 1957 *Dynamic Programming*. Princeton University Press, Princeton, NJ, USA.

Bonarini, A., Caccia, C., Lazaric, A. & Restelli, M. 2008 Batch reinforcement learning for controlling a mobile wheeled pendulum robot. In: *Artificial Intelligence in Theory and Practice II: IFIP 20th World Computer Congress, TC 12: IFIP AI 2008 Stream*, 7–10 September 2008, Milano, Italy, Springer Verlag, Vol. 276, pp. 151–160.

Breiman, L., Friedman, J., Olshen, R. & Stone, C. 1984 *Classification and Regression Trees*. Wadsworth, Belmont.

Caruana, R. 1997 Multitask learning. *Machine Learning* **28** (1), 41–75.

Castelletti, A., Galelli, S., Restelli, M. & Soncini-Sessa, R. 2010 Tree-based reinforcement learning for optimal water reservoir operation. *Water Resources Research* **46** (W09507).

Castelletti, A., Pianosi, F., Quach, X. & Soncini-Sessa, R. 2011 Assessing water resources management and development in Northern Vietnam. *Hydrology and Earth System Sciences Discussions* **8** (4), 7177–7206.

Castelletti, A., Pianosi, F. & Soncini-Sessa, R. 2008 Reservoir control under economic, social and environmental constraints. *Automatica* **44** (6), 1595–1607.

Chatterjee, K., Majumdar, R. & Henzinger, T. 2006 Markov decision processes with multiple objectives. In: *Proceedings of the 23rd International Symposium on Theoretical Aspects of Computer Science (STACS 2006)*, Marseilles, France, 23–25 February, Lecture Notes in Computer Science 3884, Springer, pp. 325–336.

Ernst, D., Geurts, P. & Wehenkel, L. 2005 Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research* **6** (1), 503–556.

Gábor, Z., Kalmár, Z. & Szepesvári, C. 1998 Multi-criteria reinforcement learning. In: *Proceedings of the Fifteenth International Conference on Machine Learning (ICML 1998)*, Madison, Wisconsin, USA, 24–27 July, pp. 197–205.

Geurts, P., Ernst, D. & Wehenkel, L. 2006 Extremely randomized trees. *Machine Learning* **63** (1), 3–42.

Gordon, G. 1999 Approximate Solutions to Markov Decision Processes. Ph.D. thesis, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA.

Hansen, M. P. & Jaszkiewicz, A. 1998 Evaluating the quality of approximations to the non-dominated set. Technical Report IMM-REP-1998-7, Technical University of Denmark.

Hipel, K. W. 1992 Multiple objective decision making in water resources. *Journal of the American Water Resources Association* **28** (1), 3–12.

Lamond, B. & Boukhtouta, A. 2002 *Water Reservoir Applications of Markov Decision Processes*. (E. A. Feinberg & A. Shwartz, eds), Springer, USA, pp. 537–558.

Lee, J.-H. & Labadie, J. W. 2007 Stochastic optimization of multireservoir systems via reinforcement learning. *Water Resources Research* **43** (11), 1–16.

Lizotte, D. J., Bowling, M. & Murphy, S. A. 2010 Efficient reinforcement learning with multiple reward functions for randomized controlled trial analysis. In: *Proceedings of 27th International Conference on Machine Learning (ICML 2010)*, Haifa, Israel, 21–24 June, pp. 695–702.

Mannor, S. & Shimkin, N. 2004 A geometric approach to multi-criterion reinforcement learning. *Journal of Machine Learning Research* **5**, 325–360.

Natarajan, S. & Tadepalli, P. 2005 Dynamic preferences in multi-criteria reinforcement learning. In: *Proceedings of the 22nd International Conference on Machine Learning (ICML 2005)*, Bonn, Germany, 7–11 August, pp. 601–608.

Neumann, G. & Peters, J. 2009 Fitted Q-iteration by advantage weighted regression. In: *Advances in Neural Information Processing Systems 22 (NIPS 2008)*, Vancouver, British Columbia, Canada, 8–11 December, MIT Press, Cambridge, MA, pp. 1177–1184.

Ormoneit, D. & Sen, Ś. 2002 Kernel-based reinforcement learning. *Machine Learning* **49** (2), 161–178.

Perny, P. & Weng, P. 2010 On finding compromise solutions in multiobjective Markov decision processes. In: *Proceedings of the 19th European Conference on Artificial Intelligence (ECAI 2010)*, Lisbon, Portugal, 16–20 August, pp. 969–970.

Riedmiller, M. 2005 Neural fitted Q iteration – first experiences with a data efficient neural reinforcement learning method. In: *Proceedings of the 16th European Conference on Machine Learning (ECML 2005), Porto, Portugal, 3–7 October*, pp. 317–328.

Timmer, S. & Riedmiller, M. 2007 Fitted Q iteration with CMACs. In: *IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning (ADPRL)*, Honolulu, HI, pp. 1–8.

Tognetti, S., Restelli, M., Savaresi, S. M. & Spelta, C. 2009 Batch reinforcement learning – an application to a controllable semi-active suspension system. In: *Proceedings of the 6th International Conference on Informatics in Control, Automation and Robotics, Intelligent Control Systems and Optimization (ICINCO 2009)*, Milan, Italy, 2–5 July, INSTICC Press, pp. 228–233.

Tsitsiklis, J. & Roy, B. V. 1996 Feature-based methods for large scale dynamic programming. *Machine Learning* **22**, 59–94.

Turgeon, A. 2005 Solving a stochastic reservoir management problem with multilag autocorrelated inflows. *Water Resources Research* **41** (12), W12414. ISSN 0043-1397.

Vamplew, P., Dazeley, R., Barker, E. & Kelarev, A. 2009 Constructing stochastic mixture policies for episodic multiobjective reinforcement learning tasks. In: *Proceedings of the 22nd Australasian Joint Conference on Advances in Artificial Intelligence (AI 2009)*, Melbourne, Australia, 1–4 December, Springer, pp. 340–349.

Vamplew, P., Dazeley, R., Berry, A., Issabekov, R. & Dekker, E. 2010 Empirical evaluation methods for multiobjective reinforcement learning algorithms. *Machine Learning* **84**, 1–30.

Wang, D. & Adams, B. J. 1986 Optimization of real-time reservoir operations with Markov decision processes. *Water Resources Research* **22** (3), 345–352.

White, D. 1982 Multi-objective infinite-horizon discounted Markov decision processes. *Journal of Mathematical Analysis and Applications* **89** (2), 639–647.

White, D. 1988 Further real applications of Markov decision processes. *Interfaces* **18** (5), 55–61.

First received 31 October 2011; accepted in revised form 5 July 2012. Available online 2 January 2013