A SELECTIVE IMPROVEMENT TECHNIQUE FOR FASTENING NEURO-DYNAMIC PROGRAMMING IN WATER RESOURCE NETWORK MANAGEMENT

A. Castelletti[a], D. de Rigo[a], A. E. Rizzoli[b], R. Soncini-Sessa[a], E. Weber[a]

[a] *Dipartimento di Elettronica e Informazione, Politecnico di Milano, Italy*
[b] *IDSIA, Manno, Switzerland*

Abstract: An approach to the integrated water resources management based on Neuro-Dynamic Programming (NDP) with an improved technique for fastening its Artificial Neural Network (ANN) training phase will be presented. When dealing with networks of water resources, Stochastic Dynamic Programming provides an effective solution methodology but suffers from the so-called "curse of dimensionality", that rapidly leads to the problem intractability. NDP can sensibly mitigate this drawback by approximating the solution with ANNs. However in the real world applications NDP shows to be considerably slowed just by this ANN training phase. To overcome this limit a new training architecture (SIEVE: Selective Improvement by Evolutionary Variance Extinction) has been developed. In this paper this new approach is theoretically introduced and some preliminary results obtained on a real world case study are presented.

Keywords: Integrated Water Resources Management; Stochastic Dynamic Programming; Neuro-dynamic Programming; Evolutionary algorithm

1. INTRODUCTION

The problem of designing optimal management (and planning) procedures for networks of water resources (reservoirs and distribution networks) has attracted analysts since the pioneering work of Rippl [1883], towards the end of the XIX century. Its main difficult is the presence together of nonlinear dynamics, high coupling among the states of these systems (and thus among the anthropical controls influencing them), risk and uncertainty, and finally decision making with conflicting objectives. It is easily arguable that the introduction of multiple managing objectives - they may be conflicting each other, expressing the intrinsically conflicting nature of most of the decisional problems - involves the need for exploring the solution sensitivity with respect to several aggregations of the objectives (searching the widest consensus). Technically such exploration implies an expensive repetition of an optimization algorithm that can only achieve a solution for a single (aggregated) objective. It follows that the efficiency of this algorithm constitutes the very core of the problem, as also demonstrated by the wide number of pertaining publications available in the literature (see for instance Yakowitz [1982], Yeh [1985], Tejada-Guibert *et al.* [1993], Lamond and Boukhtouta [1997], and Soncini-Sessa *et al.*[2001]). Thanks to its flexibility in handling nonlinearity and high coupling among states, Dynamic Programming (DP) [Bellman and Dreyfus, 1959] has been frequently used in the area of water management.

Nevertheless, its extraordinary flexibility implies a critical drawback: a dimensionality increase of the problem, i.e. an addition of reservoirs, generates an exponential increase in the time required to finding a solution. This problem, called by Bellman the "curse of dimensionality", highly limits the application of DP to real world water systems, consisting of more than two or three reservoirs: given its structural nature, it poses serious questions on the opportunity of using DP in the IWRM.

Many authors approached the problem by weakening at least one between the high state coupling and the high non-linearity, thus narrowing the application fields. Only a few tried to include both in the problem formulation, thus preserving the generality of the solution. An interesting attempt in this sense, first explored by Bellman itself [1963], is to reduce the freedom degrees of the DP solution, by resorting to a fixed class of functional approximations. In 1996, Bertsekas and Tsitsiklis exposed a methodology, named *Neuro-Dynamic Programming* (NDP), using Artificial Neural Networks (ANNs) as functional approximator of the DP solution. The ANN valuable global approximation properties allow the exploration of the search-space discretisation grid with a lower resolution, thus reducing the time required by the solution of one-step of the Bellman equation (see eq. 6), without degrading the accuracy. However this important time reduction is partially lost in the training of the ANN as the dimensionality of the problem increases. In order to overcome this problem we developed an *ad hoc* algorithm (SIEVE: Selective Improvement by Evolutionary Variance Extinction) to increase the ANN training performances.

2. THE PROBLEM

Let us consider now a water system composed by a set of interconnected reservoirs and wide-area water distribution systems. The problem solution is achieved when the control policy were obtained: it returns $\boldsymbol{u}_t$ – the vector of the volumes to be released from each reservoir and the distribution decisions – once current storage values $\boldsymbol{s}_t$ are known, so closing the feedback loop (Maas [1962]) showed in Fig. 1. In order to represent the problem of managing a regulated lake, typically a feedforward compensation has been added to the feedback control scheme: by adding it, the policy also depends upon the vector $\boldsymbol{I}_t$, which represents the meteorological information and the catchment state. Both these systems are subjected to stochastic disturbances $\boldsymbol{\varepsilon}_t$.

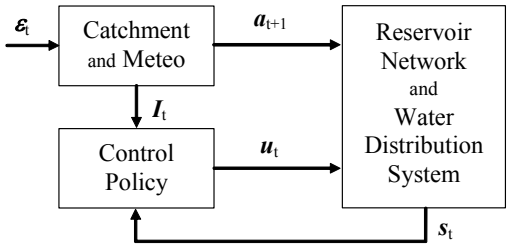

Fig. 1. Closed loop control scheme with feed-forward compensation.

The control policy directly descends from the solution of the following optimal control problem. The mass conservation equation for a generic reservoir is:

$$s_{t+1} = s_t + a_{t+1} - r_{t+1}(s_t, u_t, a_{t+1}) \tag{1}$$

where r_{t+1} is the actual release in $[t, t+1)$, while a generic element of the distribution system is usually represented without state. The model of the whole dynamic system, composed of the meteorological system, the catchment and the reservoir, can thus be represented in the compact vectorial equation:

$$\boldsymbol{x}_{t+1} = \boldsymbol{f}_t(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{\varepsilon}_{t+1}) \tag{2}$$

where $\boldsymbol{x}_t = [\, \boldsymbol{s}_t \;\; \boldsymbol{I}_t \,]$ is the state vector. Because of climate periodicity, the function $\boldsymbol{f}_t(\cdot,\cdot,\cdot)$ has a periodicity T equal to one year. During the system evolution, the state transition from $\boldsymbol{x}_t$ to $\boldsymbol{x}_{t+1}$ can produce an instantaneous cost:

$$g_t = \sum_{j=1}^{k} w^j g_t^j \tag{3}$$

where w^j is the weight of the j-th objective. Also the step costs are periodic with period T. By considering different w^j sets it is possible to taking account of several objective aggregations, so exploring the actual effects of conflicts on to the derived policies. We define the *policy* $p = \{\boldsymbol{m}_0, \boldsymbol{m}_1, \ldots\}$ as an infinite sequence of periodic functions $\boldsymbol{u}_t = \boldsymbol{m}_t(\boldsymbol{x}_t)$ of period T.

The optimal control problem is to find the policy that minimizes a function of the costs in the future, over an infinite horizon. Using the expected value operator on the disturbances $\boldsymbol{\varepsilon}$, given an initial state $\boldsymbol{x}_0$ and a discount rate α for the future costs, the cost function of a given policy p is defined as:

$$J(\boldsymbol{x}_0, p) = \lim_{h\to\infty} \sum_{t=0}^{h} \underset{\boldsymbol{\varepsilon}_1 \ldots \boldsymbol{\varepsilon}_{h+1}}{E} [\alpha^t g_t(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{\varepsilon}_{t+1})]$$

The optimal control problem is therefore solved when we find the policy p^o minimizing

$$J(\boldsymbol{x}_0) = \min_{p} J(\boldsymbol{x}_0, p) \tag{4a}$$

subject to:

$$\boldsymbol{x}_{t+1} = \boldsymbol{f}_t(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{\varepsilon}_{t+1}) \tag{4b}$$

$$\boldsymbol{\varepsilon}_t \sim \boldsymbol{\Phi}_t(\boldsymbol{\varepsilon}_{t+1} \mid \boldsymbol{x}_t, \boldsymbol{u}_t) \tag{4c}$$

$$\boldsymbol{x}_t \in S_t \quad \boldsymbol{u}_t \in U_t(\boldsymbol{x}_t) \quad \boldsymbol{\varepsilon}_t \in D_t \tag{4d}$$

$$\boldsymbol{u}_t = \boldsymbol{m}_t(\boldsymbol{x}_t) \tag{4e}$$

where S_t, U_t, D_t are the discretised domains of the state, control and disturbance.

3. SOLUTION BASED ON STOCHASTIC DYNAMIC PROGRAMMING

The optimal control problem solution (4a-4e) by SDP is based on the evaluation of the optimal cost-to-go, which is defined as the cumulative expected cost resulting from optimal actions, i.e. the cost it would incur (from time $t+1$ onwards) if the system were initially in state $\boldsymbol{x}_{t+1}$ and the system's future trajectory were obtained applying optimal control decisions in every state transition. We name this cost $H_{t+1}^o(\boldsymbol{x}_{t+1})$. If the optimal cost-to-go were known for every value of $\boldsymbol{x}_{t+1}$, the optimal decision $\boldsymbol{m}_t^o(\boldsymbol{x}_t)$ at time t would be easily found minimizing the expected value of the present cost and the discounted optimal cost-to-go from time $t+1$:

$$\boldsymbol{m}_t^o(\boldsymbol{x}_t) = \arg\min_{\boldsymbol{u}_t} \underset{\boldsymbol{\varepsilon}_{t+1}}{\mathrm{E}} [g_t(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{\varepsilon}_{t+1}) + \alpha H_{t+1}^o(\boldsymbol{x}_{t+1})] \tag{5}$$

The optimal cost-to-go associated with the present state is therefore given by the recursive equation:

$$H_t^o(\boldsymbol{x}_t) = \min_{\boldsymbol{u}_t} \mathrm{E}_{\boldsymbol{\varepsilon}_{t+1}}[g_t(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{\varepsilon}_{t+1}) + \alpha H_{t+1}^o(\boldsymbol{x}_{t+1})] \qquad (6)$$

which is known as *Bellman equation* and its solution as *Bellman function*. Under the previous hypotheses, it can be shown that the Bellman function, periodic of period T, can be obtained using the Successive Approximations Algorithm (SAA) [Bertsekas, 1995]. It evolves backwards in time from T to 1, solving the equation (6) verifying the constraints (4b-4e).

To determine the right hand side of equation (6), the algorithm, for each value of $\boldsymbol{x}_t$, must explore all the possible values of $\boldsymbol{u}_t$ and of $\boldsymbol{\varepsilon}_t$. Since this algorithm operates on a discrete search space, we have always implicitly assumed that the domains of $\boldsymbol{u}$, $\boldsymbol{x}$ and $\boldsymbol{\varepsilon}$ were discrete. Actually, it is up to the system analyst to find a satisfactory discretisation of the continuous domains of these variables. The discretisation choice is essential since it reflects on the algorithm complexity, which is combinatorial in the number of states, controls and in their discretisations. Assuming to have n states, each one discretised into N classes, the computational cost of SDP is proportional to:

$$N^n \times T \qquad (7)$$

where T is the number of time steps. In other words, if we increase the discretisation resolution, thus improving the adherence of our model to the real world, or if we consider more controls and states, to describe more complex water systems, it may happen that the time required computing a policy becomes excessively long.

Many methods have been devised in order to overcome this limitation. Some modelers (Turgeon [1981], Saad *et al.* [1994], Archibald *et al.* [1997]) simplify the state coupling by smart aggregations of the reservoir topologies, and thus they can be applied only for particular ones. Georgakakos and Marks [1987] and Georgakakos [1989], proposed an approach based on Pontriagyn's Maximum principle which does not suffer from the dimensionality problem, but requires quadratic cost functions: it is a serious limitation according to IWRM paradigm. In the following, we introduce a new approach based on neuro-dynamic programming [Bertsekas and Tsitsiklis, 1996] which has the advantage of retaining the ability of SDP to deal with highly non-linear problems, while reducing the algorithm complexity thanks to the approximation of the Bellman functions via ANNs.

4. SOLUTION BASED ON NEURO-DYNAMIC PROGRAMMING

The functional fixed class approach proposes to overcome the "curse of dimensionality" by using an approximation $\tilde{H}$ of the Bellman Function $H_\cdot^o(\cdot)$ to represent the behaviour of the original function interpolating from a limited subset $\bar{S}_t$ of points extracted from the discretisation grid S_t, so that $\boldsymbol{x}_t \in \bar{S}_t \subset S_t$. Since computing a point of $H_\cdot^o(\cdot)$ is computationally very expensive, in terms of both CPU time and memory space, reducing the number of computed points will be extremely beneficial.

Among various function approximation schemes we are mainly interested in multilayer feedforward networks, as they have been shown to be universal approximators (Hornik [1989], Kreinovich [1991]) and linear increasing with increases the dimensionality. We can approximate a highly nonlinear map $H(\boldsymbol{x})$, such as the Bellman function, where $\boldsymbol{x}$ is a vector, with a feedforward network $\tilde{H}(\boldsymbol{x}, \boldsymbol{\vartheta})$, where $\boldsymbol{\vartheta}$ is the ANN parameters vector. The improvement is not so remarkable when we deal with CPU time, since ANNs must be trained.

4.1 Training the Bellman Function Approximations

In a feedforward ANN neurons are organized in layers: the input layer is directly connected with the inputs, the output layer takes the outputs of the hidden layer (one, or more) and produces the network output. The Bellman function approximators will always have n inputs, where n is the number of state variables, and a single output (the cost-to-go value).

The training is performed by using classical first (Back Propagation) or second order (e.g. Levenberg-Marquardt) descent methods. In the Back Propagation algorithm the main problem is the descent of the weight gradient and research has focused on the development of gradient descent algorithms, which would converge quickly. Most of the time required training a network is spent in these computations, where the trade-off is between accuracy and computational complexity, since most accurate algorithms require the inversion of the Jacobian and the Hessian of the weight matrices of considerable dimensions.

4.2 The NDP Algorithm

Once an approximation architecture $\tilde{H}(\boldsymbol{x}, \boldsymbol{\vartheta})$ of $H(\boldsymbol{x})$ has been found, the sub-optimal policy $\tilde{\boldsymbol{m}}_t(\boldsymbol{x}_t)$ is given by [Bertsekas and Tsitsiklis, 1996]:

$$\tilde{\boldsymbol{m}}_t(\boldsymbol{x}_t) = \arg\min_{\boldsymbol{u}_t} \mathrm{E}_{\boldsymbol{\varepsilon}_{t+1}}[g_t(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{\varepsilon}_{t+1}) + \alpha \tilde{\mathrm{H}}_{t+1}(\boldsymbol{x}_{t+1}, \boldsymbol{\vartheta}_{t+1})] \quad \forall \boldsymbol{x}_t \in \bar{S}_t \subset S_t \qquad (8)$$

Comparing equation (8) with (5), it appears that $\tilde{\mathrm{H}}_{t+1}(\boldsymbol{x}_{t+1})$ must be trained using $\mathrm{H}_{t+1}^{\mathrm{o}}(\boldsymbol{x}_{t+1})$ as target and the vector $\boldsymbol{x}_{t+1}$ as the pattern. The original Bellman function is not available, but we can exploit the recursive nature of the Bellman equation to generate the Bellman functions needed to train their approximations thanks to the *approximate DP formula*:

$$\hat{H}_t(\boldsymbol{x}_t) = \min_{\boldsymbol{u}_t} \mathrm{E}_{\boldsymbol{\varepsilon}_{t+1}}[g_t(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{\varepsilon}_{t+1}) + \alpha \tilde{\mathrm{H}}_{t+1}(\boldsymbol{x}_{t+1}, \boldsymbol{\vartheta}_{t+1})] \qquad (9)$$

The left-hand side of (9) is an approximate cost-to-go function, which can be used to train $\tilde{H}_t(\boldsymbol{x}_t, \boldsymbol{\vartheta}_t)$, which, in turn, will be used in (9) to obtain $\hat{H}_{t-1}(\boldsymbol{x}_{t-1})$. It can be proven formally that if the approximator is "good enough", then $\tilde{H}_t$ will be a close approximation of the optimal cost-to-go function H_t^o.

The algorithm is therefore a simple rewriting of the original SAA:

Initialisation (Step 0)

a. The current algorithm iteration index j is set to 0. Initialise $H_0^{<0>}(\cdot) = 0$ for each state value.

b. Train an ANN $\tilde{H}_T(x_T, \vartheta_T)$ using the discretisation grid of x_{t+1} as the pattern and the identically null function $H_0^{<0>}(\cdot)$ as the target.

Main loop (Step 1)

c. Compute backwards in time, for t from T-1 down to 0, T functions $\hat{H}_t^{<j>}(\cdot)$ using equation (9).

- At each step t, after knowing $\hat{H}_t^{<j>}(\cdot)$, compute its approximation $\tilde{H}_t^{<j>}(\cdot, \vartheta_t)$ training the ANN.
- When t = 0, check whether an appropriate convergence criterion, measuring the distance between two Bellman functions at successive iterations, has been satisfied. If not, increment the iteration index j and go back to the beginning of Step 1 after having set $H_0^{<j+1>}(\cdot) = H_T^{<j>}(\cdot)$.

End of the Main loop.

5. PRELIMINARY RESULTS

Currently, the algorithm we presented in this paper has been applied only to some test cases to verify its correct functioning. First positive results has been presented in De Rigo *et al*. [2001].

New simplified control optimizations cases are accomplished using discretisation grid with very low resolution, in a IWRM multiobjective problem (in the Piave catchment, Italy) having three reservoirs. The outcomes show that SDP with discretisation classes equal to 10, 6 and 7 points (420 points, 41.27 hours needed to complete the computation) achieves performances almost identical to NDP with discretisation classes of 6, 3, 3 points (54 points, 9.13 hours). SDP with the same discretisation classes (6, 3, 3 points) requires 5.32 hours of computation, but gives worse performances for each management objective: the worsening is from 5% to 100%, depending to the objective. Thus NDP is almost 450% faster than equivalent SDP, even for so low-resolution discretisation grid. A coarse grid offers an under-estimation of the real NDP potentialities, because the Bellman function can be described (approximating it to any desired accuracy) with a finite-dimension vector of ANN parameters, and its dimension does not depend on discretisation grid density, provided that the set of $\hat{H}_\cdot(\cdot)$ punctual evaluations on the grid will contain enough information in order to permit an unbiased interpolation. When this happens, the ANN interpolation becomes "good enough" so if we look for comparable SDP performances, we will have to sample $H_\cdot(\cdot)$ with an hash table (with its keys being the discretisation grid) really huge. Looking at the NDP from this point of view, we can see that it is, after all, almost obviously joined with the sampling theory, and with the degree of information redundancy. Another question (and an interesting further study) is how to obtain the highest level of information, using the lowest sampling cardinality of the discretised state space (e.g. using adaptive sampling).

Notice that ANN training can require a relevant part of NDP computation time. The available Piave tests (SDP and NDP with 54 points) showed that the ANN training spends over 40% of the CPU time.

6. AN IMPROVED *NDP* SOLUTION: *ANN* TRAINING USING *SIEVE*

Using NDP instead of SDP, we can achieve a sensible reduction of computer time and memory requirements, so enabling IWRM model accuracy improvements by extending the system state in order to take advantage of some other significant information that we ignored before. This is an important goal, but we have to be careful: ANN approximation depend on choosing right ANN parameters: the trade-off is between accuracy and computational cost. The complexity of the ANN training involves the cost of a single gradient descent step (but its efficiency depends mainly on the Hessian approximation, stability and well conditioning, which generally increase their cost by improving the numerical properties of the curvature matrix), the number of steps needed for a required accuracy, the number of neurons and - if we use multiple hidden layers - their distribution in each layer. These factors are strongly related with the question of avoiding local minima searching for the ANN parameter vector that satisfies the approximation accuracy request.

For example, adopting a great number of neurons, it is most probable to find local minima corresponding to small ANN output error (with respect to the target training set), so the absolute minimum may not be needed. This of course follows from the added possibility to work around one or many wrong initialized neurons by adjusting the best ones. However the ANN training cost per step, using second-order methods, involves an Hessian matrix inversion: we cannot so easily add neurons and thus parameters when the ANN training time becomes a significant part of the total NDP time cost. And we cannot risk overfitting and waviness errors in a close approximation of the Bellman function (the *min* operator of (9) may lead to potentially catastrophic result of wiggling unexpected oscillations). So we have to avoid overfitting, to reduce the ANN training time requirement and to explore the ANN parameter space with multiple parameter initialization in order to find the absolute minimum or at least a sub-optimal local minimum, without spending computational resources for bad parameter initializations.

We choice multiple initializations instead of some other perturbation technique (like simulated annealing), thinking to the high nonlinearity that is typical for the DP application in IWRM problems. A consequence of this strong nonlinearity is the great variety of Bellman function representing the optimal cost-to-go for this kind of problems: that is relevant if we have to tune the perturbation technique guessing its tuning constants.

We will present the SIEVE (Selective Improvement by Evolutionary Variance Extinction) technique: an evolutionary algorithm with geometrical selection (sieving)

and pseudo-genetic generation by adaptive decreasing-variance perturbations.

6.1 The SIEVE core

The core of the SIEVE architecture is to use iteratively a selection (sieving upon an inverse geometrical series) of the best parameter vectors, reducing exponentially the number of parameter vectors surviving at the next iteration. We compensate this reduction with a geometrical extension of the computational resources dedicated to train each parameter vector (making so that each iteration uses the same amount of computations as the others), until the absolutely best vector passes the last sieve.

After each sieving selection phase, the survived vectors are put before a generative phase in which some other vectors are generated from them by adding perturbing noise. The noise variance decreases increasing the iteration number, in order to preserve the best training result achieved from the last parameter vectors, therefore leaving the possibility to significantly perturb some vector (exploration of new areas of the parameter space). All the sieved parameter vectors and the new generated from them are then trained, and so one for each iteration.

The first (iteration 0) set of parameter vectors is not quite a random generated set, but has to contain also the best parameter vectors resulting in each ANN training of the "nearest" Bellman functions (e.g. if the SAA algorithm is being to train $\tilde{H}_t^{<j>}(\cdot,\vartheta_t)$ from $\hat{H}_t^{<j>}(\cdot)$ – it was obtained using equation (9) – the nearest $\tilde{H}_{\cdot}^{<\cdot>}(\cdot,\vartheta_t)$ are $\tilde{H}_{t+1}^{<j>}(\cdot,\vartheta_t)$ and $\tilde{H}_t^{<j-1>}(\cdot,\vartheta_t)$, that are yet computed and available at this time). So the parameter vectors of the nearest Bellman functions – together with the remaining random generated vectors of the first iteration – have the possibility to survive until the last iteration, improving itself by training. On the other hand, they have the possibility to exponentially generate other parameters similar to them, searching the most satisfactory during the generative phases preceding the training ones. This approach attempt to maximize the probability of finding new good parameter vectors being able to describe a Bellman discontinuity with respect to the neighbor Bellman functions (using random initialization vectors), and also to maximize the probability to train the neighbor parameter vectors adapting them to a Bellman function not so far from the nearest.

6.2 The SIEVE generalized architecture

The first formulation of the SIEVE training architecture provided for a sieve selection by a factor ¼ (Fig. 2), a generative phase that doubled the number of the sieved parameter, and finally a doubling of the computation time for each parameter survived with respect to the computation time used in the previous iteration. During the iteration sequence the variance of the noise added in the generative phase was being even more little, until it went to extinction.

It is easy to generalize this architecture, by formalizing 3 factors: the selection factor s, the generative factor *g* and the geometrical ratio *k* characterizing the training

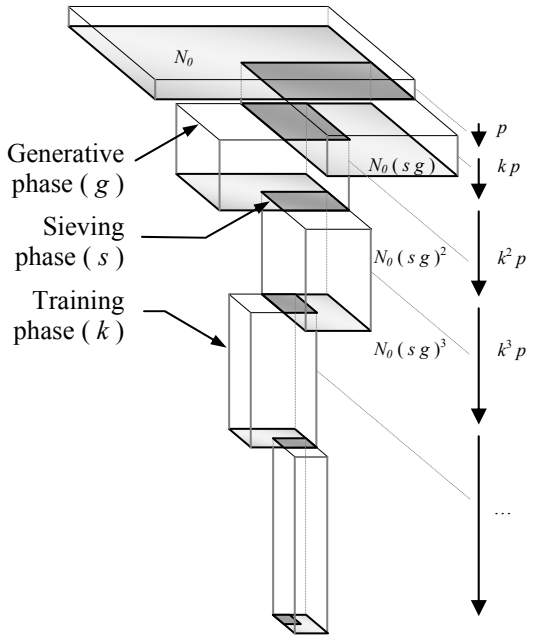

Fig. 2**. The SIEVE generalized architecture.**

increment (in the first formulation s=0.25, g=2, k=2). If the iteration 0 uses p flops to train each parameter vector, and n is the number of iterations, then the computational cost of the SIEVE in flops is given by:

$$C=\sum_{i=0}^{n}\underbrace{N_0 p\,(sgk)^i}_{C_i}=\begin{cases}\dfrac{p(1-sgk)^{n+1}}{(sg)^n(1-sgk)} & \text{if } sgk\neq 1\\[2ex] \dfrac{p(n+1)}{(sg)^n} & \text{otherwise}\end{cases} \tag{10}$$

under the hypothesis of training in the last iteration only the best vector ($N_n=1$), so that the number of trained vectors during the i-th iteration will be $N_i=1/(sg)^{n-i}$, $i\in[0,\cdots,n]$ (obviously all the real values for N_i, and thus in (10) for C and C_i must be properly rounded).

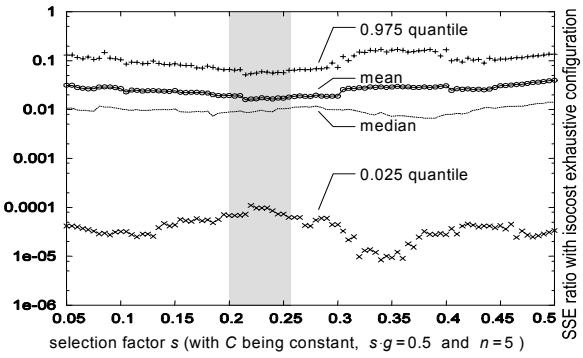

Fig. 3. SSE ratio sensitivity analysis.

First available test (Fig. 3) shows that the original configuration (s=0.25, g=2, k=2) accomplishes a relevant reduction of the functional norm – discretised using the Sum of Square Errors (SSE) – with respect to an exhaus-

tive configuration (s=1, g=1, k=1) having the same computational cost and the same training depth. Sensitivity analysis near the original configuration (leaving k=2, the same cost C and thus $s \cdot g$=0.5) shows the robustness of reducing the selection factor s by increasing the generative factor g. This robustness has been achieved with a noise variance reduction at each iteration by a geometrical factor of 4. The best mean performances seem to be near the interval $s \in [0.2, 0.25]$ in which the SSE range appears to be least. When we are writing this paper, numerical test are in progress searching for the best configuration of *s, g, k* parameters, and of n, it determinate the initial number of parameter vectors, due *s, g, k* were fixed.

7. CONCLUSIONS

An approach to the integrated water resources management based on neuro-dynamic programming has been presented. Neuro-dynamic programming allows to reduce the amount of memory needed to store the Bellman functions during the solution of an optimal control problem. It also reduces the computation time when the state space, used as training pattern, is sampled with a coarser grid, while the ANN, which approximates the Bellman function, still manages to maintain a good approximation performance. The ANN training phase on NDP requires a relevant fraction of the computational time: we propose the SIEVE (Selective Improvement by Evolutionary Variance Extinction) technique in order to achieve better performances.

The first results are promising, but there is space for more research, especially on the efficient sampling of the discretised state space, trying to obtain the most efficient approximation of the Bellman function.

8. REFERENCES

Archibald, T.W., McKinnon, K.I.M., and Thomas, L.C., An aggregate stochastic dynamic programming model of multireservoir systems, *Water Resour. Res.*, 33, pp. 333-340, 1997.

Bellman, R.E., and Dreyfus, S.E., Functional approximations and dynamic programming, *Mathematical Tables and Other Aids to Computation*, 13, pp. 247–251, 1959.

Bellman, R.E., Kabala, R., Kotkin, B., Polynomial approximation – a new computational technique in dynamic programming*, Mathematical Tables and Other Aids to Computation*, 17, pp. 155–161, 1963.

Bertsekas, D.P., *Dynamic Programming and Optimal Control*, Athena Scientific, Belmont, MA, 1995.

Bertsekas, D.P., and J.N. Tsitsiklis, *Neuro-Dynamic Programming*, Athena Scientific, Belmont, MA, 1996.

De Rigo, D., Rizzoli, A.E., Soncini-Sessa, R., Weber, E., Zenesi, P., Neuro-Dynamic Programming for the Efficient Management of Reservoir Networks, presented at*: MODSIM 2001*, Canberra, Australia, 2001.

Georgakakos, A.P., and Marks, D.H., A new method for real-time operation of reservoir systems, *Water Resour. Res.*, 23(7), pp. 1376–1390, 1987.

Georgakakos, A.P., Extended Linear Quadratic Gaussian Control for the real-time operation of reservoir systems, in *Dynamic Programming for Optimal Water Resources Systems Analysis*, A. Esogbue, ed., Prentice Hall Publishing Company, NJ, pp. 329–360, 1989.

Hornik, K., Multilayer feedforward networks are universal approximators, *Neural Networks*, 2, pp. 359–366, 1989.

Kreinovich, V., Arbitrary nonlinearity is sufficient to represent all functions by neural networks: a theorem, *Neural Networks*, vol.4, pp. 381–383, 1991.

Lamond, B.F., Boukhtouta, A., Optimizing future hydropower production using Markov Decision Processes, Working Paper 95-44, CRAEDO, Université Laval, Quebec, 1997.

Maas, A., M.M. Hufschmidt, R. Dorfam, H.A. Thomas, S.A. Marglin and G.M. Fair, *Design of Water Resource Systems*, Harvard Univ. Press, 1962.

Piccardi, C. and R. Soncini-Sessa, Stochastic dynamic programming for reservoir optimal control: dense discretization and inflow correlation assumption made possible by parallel computing. *Water Resour. Res.*, 27(2), pp. 729–741, 1991.

Rippl, W., The capacity of storage reservoirs for water supply. *Minutes Proc. Inst. Civ. Eng.,* 71, pp. 270-278, 1883.

Saad, M., Turgeon, A., Bigras, P., Duquette, R., Learning Disaggregation Technique for the Operation of Long-Term Hydroelectric Power Systems, *Water Resour. Res.,* 30(11), pp. 3195-3203, 1994.

Soncini-Sessa, R., Castelletti, A., Rizzoli, A.E., Weber, E., New ideas in the design of water reservoir management policies, invited paper at: IFAC Workshop "Modelling and Control in Environmental Issues", Yokohama, Japan, 2001.

Tejada-Guibert, J.A., Johnson, S.A., Stedinger, J.R., Comparison of two approaches for implementing multi-reservoir operating policies using dynamic programming, *Water Resour. Res.,* 29, pp. 369-380, 1993.

Turgeon, A., A decomposition method for the long-term scheduling of reservoirs in series, *Water Resour. Res.,* 17, pp. 1565-1570, 1981.

Yakowitz, S., Dynamic programming applications in water resources, *Water Resour. Res.,* 18, pp. 673-696, 1982.

Yeh, W., Reservoir management and operations models: a state of the art review, *Water Resour. Res.,* 21, pp. 1797-1818, 1985.