Proceedings of the 2007 IEEE Symposium on Approximate
Dynamic Programming and Reinforcement Learning (ADPRL 2007)

Opposition-Based Reinforcement Learning in the Management of Water Resources

M. Mahootchi, H. R. Tizhoosh, K. Ponnambalam
Systems Design Engineering, University of waterloo, 200 University Avenue West, Waterloo, Ontario, N2L 3G1, Canada, mmahootc@engmail.uwaterloo.ca, tizhoosh@uwaterloo.ca, ponnu@uwaterloo.ca

***Abstract*— Opposition-Based Learning (OBL) is a new scheme in machine intelligence. In this paper, an OBL version Q-learning which exploits opposite quantities to accelerate the learning is used for management of single reservoir operations. In this method, an agent takes an action, receives reward, and updates its knowledge in terms of action-value functions. Furthermore, the transition function which is the balance equation in the optimization model determines the next state and updates the action-value function pertinent to opposite action. Two type of opposite actions will be defined. It will be demonstrated that using OBL can significantly improve the efficiency of the operating policy within limited iterations. It is also shown that this technique is more robust than Q-Learning.**

***Index Terms*— water reservoirs, Q-learning, opposite action, reinforcement learning.**

I. Introduction

FINDING efficient operating policies in multi-reservoir applications has been a challenging research area in the past decades. Many attempts using traditional methods including linear and non-linear optimization techniques have been performed to overcome the curse of dimensionality in real-world applications. However, most of these efforts have included different varieties of simplifications and approximations, which usually make the operating policies inefficient in practice. Using optimization techniques along with simulation, such as Reinforcement Learning (RL) techniques, could be a suitable alternatives for this purpose. RL is a powerful and well-known technique in machine learning research to cope well with many optimization and simulation problems. It is also called Simulation-Based Dynamic Programming [1] in which a decision maker (agent) optimizes an objective function through interacting with deterministic or stochastic environments. These interactions might cause some instant reward or punishment which are accumulated during the training process and called action-value functions. These values are the basis for the agent to take proper actions in different situations (states). Based on what has been proven by Watkins [2], these values converge to steady states if each action state pair is visited for infinite number of times - practically multiple times; however, this may take too much time in real-world applications. Therefore, the question may come up how to achieve an optimal solution with fewer interactions. Opposition-Based Learning (OBL) scheme, which is firstly introduced by Tizhoosh [3], could be a suitable answer to the mentioned question. Tizhoosh has shown that using this scheme in some soft computing methods such as Genetic Algorithms (GA), Neural Networks (NN), and Reinforcement Learning (RL) can generally speed up the training process. However, this is completely problem dependent. He also used this scheme with Reinforcement Learning (RL) in finding a path to a fixed goal in discrete $grid\ worlds$ of different sizes [3]. In this specific example, an agent takes an action in the current state and updates the respective action-value function in addition to those functions, which are related to opposite actions or states. The criterion for granting reward or punishment to an agent is the distance to a fixed goal inside the grid. Moreover, the environment under the study in this case study is totally deterministic. In this paper, we investigate the effect of opposition-Based Learning (OBL) scheme using Q-learning method for the reservoir management. To show that this scheme is efficient, it is applied on a single reservoir problem which is completely stochastic in terms of inflow to reservoir. Therefore, we can easily find the optimal or near-optimal policies and performances by regular Q-Learning and simulation in a reasonable time. The corresponding results would consequently be extended for multi-reservoir applications in the future research. The paper is organized as follows: In the next section, a simple model of a single reservoir will be explained. Section 3 will provide a general review of Stochastic Dynamic Programming (SDP). In section 4, Q-Learning will be briefly reviewed. Some basic concepts of OBL and a version of opposition-based algorithm using Q-Learning in the reservoir management will be described in section 5 and 6. Finally, in sections 7 and 8, some experimental results and conclusions will be provided.

1-4244-0706-0/07/$20.00 ©2007 IEEE

II. SINGLE RESERVOIR MODEL

Generally, the goal in the reservoir management is to find a policy mapping defined states to optimal water releases such that the maximum benefit or the minimum cost is achieved in long or short periods of time of reservoir operations. For a single reservoir, this model can be illustrated as described in following subsections [4], [5].

objective function - The objective function can be considered as follows:

$$Z \;=\; \max \sum_{t=1}^{T} f^t(s^t, a^t, d^t), \tag{1}$$

Generally in a single reservoir application, one of the following objective functions is used:

$$Z1 \;=\; \max \sum_{t=1}^{T} (a^t \times C^t), \tag{2}$$

$$Z2 \;=\; \min \sum_{t=1}^{T} ((a^t - d^t)^2), \tag{3}$$

where T is the number of periods, s^t is the storage volume in period t, a^t is the amount of release in period t, d^t is the given demand in period t, and C^t is the unit price of water released in period t.

Balance Equation - This equality constraint indicates the conservation of mass with respect to in- and output of the reservoir:

$$s^{t+1} = s^t + I^t - a^t \quad \forall\, t \;=\; 1 \;\cdots\; T, \tag{4}$$

where I^t is the amount of inflow to the reservoir in period t. We can also consider evaporation or seepage in the balance equation.

Minimum and maximum storage - The following constraint provides a flood control while considering some aspect of recreation or maintaining a minimum level for powerplant operations:

$$s^t_{MIN} \le s^t \le s^t_{MAX} \quad \forall\, t \;=\; 1 \;\cdots\; T, \tag{5}$$

where s^t_{MAX} and s^t_{MIN} are the maximum and minimum storage levels in period t, respectively.

Minimum and maximum releases - The purpose of the following constraint is to provide a suitable water quality for existence of wildlife and fish, and preventing flood in downstream:

$$a^t_{MIN} \le a^t \le a^t_{MAX} \quad \forall\;\; t \;=\; 1 \;\cdots\; T, \tag{6}$$

where a^t_{MAX} and a^t_{MIN} are the maximum and minimum release in period t, respectively.

III. STOCHASTIC DYNAMIC PROGRAMMING

In Dynamic Programming (DP) with stochastic situation called SDP, two different ways have been developed for finding the optimal policy and value functions: policy iteration and value iteration [1]. In the first one, using an arbitrary policy, a set of equations is established and solved for finding value functions corresponding to this policy in steady state. This step is called policy evaluation. Using these value functions, the second step called policy improvement is performed to create a new policy. It has been proved that if these steps are repeated frequently enough, the policy and corresponding value functions will converge to the optimal and steady state points. The value iteration version of the SDP starts from arbitrary value functions for all possible states in the last period and continues by updating these values with a recursive function iteratively. Every new value function is basically the best expected value of the objective or the recursive function with one or more actions as the best actions among all admissible actions from current period to the end of period. In other words, in every iteration of this method, both policy improvement and policy evaluation are performed. This has been also proved to converge to steady state points. There are two ways to specify the admissible actions for performing each of mentioned methods of SDP in reservoir management: pessimistic and optimistic. Assumed that inflow is the only stochastic parameter in the reservoir model, admissible actions for every state (storage level in each period) based on these two schemes are determined according to steps explained in Table I.

It is obvious that in pessimistic version of SDP, which is the conventional SDP in literature, an action will be admissible if the next storage level computed from balance equation (4) meets the minimum storage level for all discrete values of inflow. Of course, actions which cause a violation of minimum storage level for some inflows can be significantly penalized. Therefore, these actions will be most likely discarded from the optimal policy at the end of the optimization process. In the optimistic scheme, if there exists only one discrete value of inflow in which the next storage computed from the

TABLE I
FINDING ADMISSIBLE ACTIONS

1) find all possible actions with respect to maximum and minimum release, $\{R^t_{min} = a^t_1,\; a^t_2,\; \ldots\; a^t_K = R^t_{max}\}$
2) discretize inflow to reservoir to N values, $\{I^t_1,\; I^t_2,\; \ldots\; I^t_N\}$ with probability $\{P(I^t_1),\; P(I^t_2),\; \ldots\; P(I^t_N)\}$
3) discretize storage level of reservoir to N values with respect to maximum and minimum storage level in each period $\{s^t_{min} = s^t_1, s^t_2, \ldots s^t_M\}$
4) find the admissible actions as follows: for all actions $k = 1 : K$, for all states $i = 1 : M$, and for all discrete inflows $n = 1 : N$
   - Pessimistic: $A^t_i = \{(a^t_k),\; \forall\; n \;|s^t_i + I^t_n - a^t_k \le s^{t+1}_{min} \quad n = 1, \ldots N\}$
   - Optimistic: $A^t_i = \{(a^t_k,\; \exists\; n \;|s^t_i + I^t_n - a^t_k \le s^{t+1}_{min} \quad n = 1, \ldots N\}$

TABLE II

FINDING VALUE FUNCTIONS V_i^t FOR ALL ADMISSIBLE ACTIONS IN OPTIMISTIC SCHEME FOR SDP FOR EACH STATE

for all admissible actions a_k^t in state i and time t, k=1,... K_i^t
** initialize immediate and accumulated reward IR=0, AR=0
for all discrete values of inflow in time t, I_n^t,
with probability $P(I_n^t)$ $n = 1, \dots N$ with probabilities
* Find the next storage based on balance equation:
$s^{t+1} = s_i^t + I_n^t - a_k^t$
* Find the actual storage:
$s_{(actual)} = \max(s_{min}^{t+1},\ s^{t+1})$
* Find the actual release from reservoir:
$a_{(actual)} = s_i^t + I_n^t - s_{(actual)}$
* Calculate the immediate reward:
$IR = IR + f(a_{(actual)}) \times P_{(I_n^t)}$
* Calculate the accumulated reward
$AR = AR + V_j^t \times P_{(I_n^t)}$
j= the closest discrete value of storage to $s_{(actual)}$
** $Q^t_{(i,a_k^t)} = IR + AR$
** take another action
calculate the value function pertinent to state i and period t
$V_i^t = \max_k Q_{(i,a_k)}$

balance equation (4) satisfies the minimum storage, the corresponding action will be admissible. Therefore, some modifications are needed for finding the value functions because one action in this scheme might lead to violation of the minimum storage level for some discrete values of inflow in the balance equation. As illustrated in Table II, for each action, the actual release and related reward should be computed and multiplied with the probability of the respective inflow used in balance equation. The summation of all these values makes the immediate reward for corresponding action. Moreover, the next storage should be set to the minimum storage level if an action violates the minimum storage. In this situation, the transition probability from current storage to minimum storage is the summation of all probabilities of inflows which cause a violation of the minimum storage level in the balance equation. SDP can be applied in a stochastic optimization problem subject to existing the transition probabilities.In absence of these probabilities, Q-Learning as a model-free technique would be an option to tackle the problem.

IV. Q-LEARNING

Q-Learning has been derived from the formulation of the Stochastic Dynamic programming (SDP) [1], [6]. This method uses the Robbins-Monro algorithm to take an average of action-value function based on the real observations in simulation [7]. In other words, after each observation, the corresponding action-value function is updated:

$$Q^t_{(k+1)}(i,a) = Q^t_{(k)}(i,a) + \tfrac{1}{NOV(i,a)} \times [R + \gamma \max_{b \in A(j,\acute{t})} Q^{\acute{t}}_k(j,b) - Q^t_{(k)}(i,a)], \quad (7)$$

where R is total immediate reward, $NOV(i,a)$ is the number of visits for action a in state i. The term $\frac{1}{NOV(i,a)}$ can be substituted with α and called learning rate. This parameter is a very important component in the learning process by which the convergence of all action-value functions to the steady state points is controlled. Two types of learning rates are actually used in the learning process: constant and variable. In the first one, which is suitable for dynamic systems, the learning rate is constant through the whole process of learning; however, this might cause some instability or significant oscillation in action-value functions during the learning process. In many applications, this value is considered very small; therefore, much more time for leaning is needed. In the second type, the learning rate changes with time. After each experience, this value is updated for corresponding action-value function; however, it may depends on the initial events and consider very small weights for others which are in distance to these observations. To decrease the effect of this drawback, some researchers considered $\frac{A}{NOV(i,a)}$ in stead of $\frac{1}{NOV(i,a)}$ in which A can be a small value [1]. Another important issue in Q-Learning method is the policy of taking action in each iteration. This policy could be a combination of exploration and exploitation. An agent usually would like to explore the environment in initial experience and proceed with more exploitation as the learning process is continued. There are four common policies in taking actions: greedy, ϵ -greedy, Softmax, and random policy [8]. Taking action in greedy policy is only exploitation of achieved knowledge. In ϵ-greedy policy, an agent prefers takes greedy actions most of the time (exploitation), and it takes a random action in a certain amount of time (exploration for $\epsilon\%$ of the time). In the Softmax policy, which has been derived from Gibbs distribution, an agent chooses action a with the probability [8]:

$$P(a) = \frac{e^{Q^t(i,a)/\tau}}{\sum_{b \in |A(i)|} e^{Q^t(i,b)/\tau}}. \quad (8)$$

τ is a positive number called temperature. For high temperatures, the probability for choosing all actions are the same; however, small temperatures cause higher probability for actions with high action-value estimation. Admissible actions in Q-Learning are analogously determined based on what was explained for SDP in Table I. An interval in which all inflows to reservoir are placed on can be considered. This interval would be extracted from a specific distribution or existing historical data. The minimum and maximum of the interval are used in balance equation to find admissible actions for each state in pessimistic or optimistic strategy, respectively [9].

V. Opposition-Based Learning (OBL)

Opposition-Based Learning (OBL) scheme has been used in different ways in machine learning algorithms. For example, to apply this scheme to Neural Networks(NN) [3], [10], two different networks are constructed: one with random weights and another with their opposites. Given these two networks, the training processes are performed simultaneously with training data for one complete epoch, and total error pertinent to each of these networks are computed. The best one in terms of error function is selected for the next epoch. Tizhoosh proposed two different versions to find opposite weights in Neural Networks [3]. Ventresca and Tizhoosh [10] also used opposite transfer function to improve the time of learning for the backpropagation algorithm in feed-forward multi-layer perceptron networks. OBL has also been tested for Genetic Algorithms (GA) in which anti-chromosomes are determined for some chromosomes with the lowest fitness values in the population by total change of all bits of the chromosomes from zero to one or vice versa. In the Reinforcement Learning (RL) method, an opposite action or state can be defined with two different types: type I, which is based on boundaries of action and state, and type II in which the opposite action or state are calculated based on respective action-value functions [3]. Let s be the current state and a the action taken by the agent. Assuming that minimum and maximum state and action can be numerically determined as s_{min}, s_{max}, a_{min}, and a_{max}, respectively. Based on the type I definition, the opposite action, $\breve{a}$ and the opposite states $\breve{s}$ are determined as follows [3]:

$$\breve{a} = a_{max} + a_{min} - a, \tag{9}$$

$$\breve{s} = s_{max} + s_{min} - s. \tag{10}$$

In the reservoir management, an opposite action/state in each iteration will be determined with respect to the boundary conditions of storage levels as follows:

$$\breve{a} = R^t_{max} + R^t_{min} - a, \tag{11}$$

$$\breve{s} = s^t_{max} + s^t_{min} - s. \tag{12}$$

In type II, action-value functions have an important role to specify opposite actions and states. To compute opposite action, the following formula may be used:

$$\breve{a} \in \{\hat{a} \mid Q^t_{(i,\hat{a})} \approx \max_{(b \in A_i)} Q^t_{(s_i,b)} + \min_{(b \in A_i)} Q^t_{(s_i,b)} - Q^t_{(s_i,a)}\}. \tag{13}$$

Tizhoosh [11] has suggested a similarity matrix, $\eta_{i,j}$, which can be considered for finding opposite states:

$$\eta_{(i,j)} = 1 - \frac{\sum_{(b \in A_i)} |Q^t_{(s_i,b)} - Q^t_{(s_j,b)}|}{\sum_{(b \in A_i)} \max((Q^t_{(s_i,b)}),\ (Q^t_{(s_j,)}))}, \tag{14}$$

where $Q_{(s_i,a)}$ is the action value for state i and action a, $\eta_{(i,j)}$ is a value showing the similarity between state i and state j, A_i is the set of all admissible actions in state i. To perform the training process after each interaction, reinforcement agent, whether using type I or type II opposition, could update at least four action-value functions corresponding to each action and state pair. If the agent knows the transition functions, it can use the stochastic parameters to calculate the next state for each pair [9]. Otherwise, the agent has to use the knowledge, which is acquired from previously taken actions, to establish an intuition about the reward and the next state for opposite action and opposite state [11].

VI. Opposition-Based Q-Learning in Reservoir Management

In the reservoir management, it is assumed that the reinforcement agent knows the transition function which is the balance equation, it also knows the reward function f^t_a. Therefore, it can calculate the next state and the immediate reward for opposite action and opposite state. Moreover, because the majority of action-value functions have zero value at the very beginning of learning process or may have been previously observed multiple times, opposite action and state determined in type II are not accurate enough. Therefore, some kind of function approximation such as a feed-forward multi-layer perceptron networks can be used to increase the accuracy of choosing opposite action or state in type II with respect to current knowledge extracted from previous observations. Furthermore, running a feed-forward multi-layer perceptron is time-consuming and not reasonable to be trained frequently during the learning process. Therefore, the network could be trained only every several episodes. The distance between two consecutive trainings is considered as a model parameter and should be specified at the beginning of the learning. As a result, there are two action-value functions: one is obtained from the direct interactions of the agent with the environment, the second is only used to find the type II opposite actions and states. A version of Q-Learning using OBL based on types I and II is illustrated in Table III.

VII. Experimental Results

All information in a single-reservoir case study has been derived from Fletcher [12]. One cycle in this problem is one complete year with 12 months. Minimum and maximum storage and release in different months of year are given in Table IV.

Inflow to reservoir is normally distributed and its monthly averages are given in Table IV. The coefficient of variance ($\frac{\sigma_{It}}{\mu_{It}}$) in each month of a year is a pre-determined value used to obtain the variance of inflow.

Moreover, the evaporation from reservoir is neglected. The objective function in Fletcher's model is to maximize the power generation. The benefit of release per/unit is approximated in a single value for each month in Table V. In addition to above objective function, the total least-squared error is considered as a new objective, which should be minimized during the learning process. The demand for power generation in different months of a year are given in the Table IV. Each state and decision variable was discretized into 55 and 7 equal intervals, respectively. Therefore, there are 8 discrete values for storage in each month, in which the state of the system in each iteration should belong to one of them. Total number of actions that can be taken by the agent is 56. Each episode in Q-learning or opposition-based Q-Learning is equivalent to 30 years . the number of repetitions for each episode is considered as a parameter. Furthermore, the results of opposition-based Q-Learning in this study focus on opposite actions not opposite states. A multi-layer perceptron (MLP) with one hidden layer and twenty nodes as a function approximation is used to find the type II opposite actions. As mentioned in Table III, the training of the networks is only performed after every c episodes during the learning process. This parameters is determined at the beginning of the learning. According to the steps described in the table, before the first training of the network, the Opposite action is computed based on type I using equation 9 , and it is then calculated based on type II using equation 13. The decisions of Q-learning are compared to decisions derived from opposition-based Q-Learning. Moreover, SDP as a concrete method in a single reservoir application is used for verifying the performance of both learning methods. As it is clear, the solution of SDP is much closer to global solution; therefore, it is expected that the Q-Learning and opposition-based method achieve the same results. Furthermore, to show the performance of each method, mean, variance, and coefficient of variance $(\frac{\sigma}{\mu})$ of the benefit or least-squared error in a year are computed based on a simulation of the reservoir from 50 to 2000 years using the policy achieved from the learning step. To compare the results of Q-Learning and opposition-based Q-Learning, we have run 20 complete experiments including the learning and simulation steps and computed the average and variance of annual average, annual variance, and annual coefficient of variance for each specific set of parameters.

Performance - In the Tables VI, VII, and VIII, the average of annual average gain and annual variance of gain in 20 different experiments with different sets of parameter for regular Q-Learning, opposition-based Q-Learning type I, and opposition-based Q-Learning type II are shown. A code is assigned to each set of parameters. Moreover, in Table VIII the number of

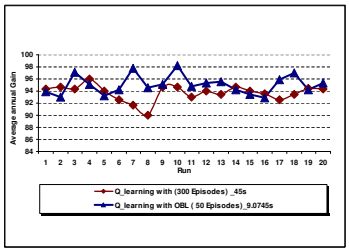

Fig. 1. Comparing the average of annual gain in Q-Learning and opposition-based Q-Learning type I.

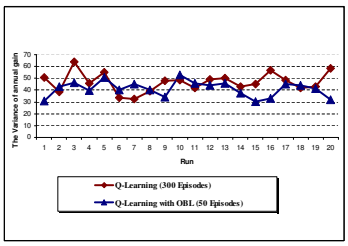

Fig. 2. Comparing the variance of annual gain in Q-Learning and opposition-based Q-Learning type I.

times that a neural network is trained is also given. As it is clear there is a significant difference in average gain for opposite-based Q-Learning both type I or II in the number of small episodes. For instance in 50 episodes, the average of average gains for 20 runs is 79.60 in Q-learning compared to 95.04271 and 91.04361 in opposition-based Q-Learning type I and Type II, respectively. It is worth mentioning that in this situation the average of variance and coefficient of variance in opposition versions especially in type I are almost the same (e.g., for 50 episodes, the variances are 40.0 and 41.11 for Q-learning and opposition-based Q-Learning type I, respectively). It means that the average of gain dramatically improves without increasing variance. To show that opposition-based Q-Learning has better results than regular Q-Learning at the beginning of the learning, we compare the performance of these two methods in terms of average gain, variance and coefficient of variance in Figures 1-3 for 20 runs with 50 and 300 episodes for opposition-based Q-Learning type I and regular Q-learning, respectively. As it is clear in these figures, for most of runs, the opposition-based Q-learning is more efficient than regular Q-learning. Figures 4 and 5 also compare these two types of Q-learning in 300 episodes and 1000 episodes for opposition and regular Q-learning, respectively. As shown in Figure 4, the average gain in opposition version is higher than the average in Q-Learning for most runs. However, the variance of opposition version become worse for almost all runs

TABLE III

THE PROCESS OF Q-LEARNING IN OBL BASED ON TYPE I AND II

1) initialize action-value functions and opposite action-value functions, $Q^t_{i,a} = 0$, and $\breve{Q}^t_{i,a} = 0$
2) initialize the number of episodes between two different runs of the training of the neural networks, c,
3) descritize storage level and release with respect to physical conditions
4) find the admissible actions based on optimistic or pessimistic scheme, $A^t(i)$, for all discrete values of storage and periods
5) determine the number of years in each episode, $noyears$, learning rate, α, the maximum number of episodes $noepisode$, the number of hidden layer and nodes, hd and hn, for the networks
6) set the number of training the networks, $r = 0$ and the number of episode, $h = 1$
7) start with h^{th} episode and taking a random value for the current storage (current state)
8) set $year = 1$ at the beginning of each episode
9) set the period t
10) take an admissible action based on the selected policy and receive reward
11) update the corresponding action-value function:
$Q^t_{(k+1)}(i,a) = Q^t_{(k)}(i,a) + \alpha \times [R+ \gamma \max_{b \in A(j,t)} Q^t_k(j,b) - Q^t_{(k)}(i,a)]$
12) checking the following conditions:
** if $h < c$
* compute opposite action, $\breve{a}$, based on type I using equation 9
* update the action-value function:
$Q^t_{(k+1)}(i,\breve{a}) = Q^t_{(k)}(i,\breve{a}) + \alpha \times [R+ \gamma \max_{b \in A(j,t)} Q^t_k(j,b) - Q^t_{(k)}(i,\breve{a})]$
** if $h = c \times r$,
* $r = r + 1$
* set opposite action-value functions:
$\breve{Q}^t(i,a) = Q^t(i,a)$ for all i and a
* consider action-state pairs which are visited enough as the training data
* run the feed-forward multi-layer perceptron networks
* approximate action-value functions for action-state pairs in the test data using trained networks, $\breve{Q}^t(i,a)$
* compute the opposite action using equation 13 with opposition action-value functions $\breve{Q}^t_{(k+1)}(i,a)$
* update action-value functions and their opposites for the action and the opposite action, $\breve{Q}^t(i,a)$,$\breve{Q}^t(i,\breve{a})$, and $Q^t(i,\breve{a})$
** if $h \geq c \times r$
* compute the opposite action using equation 13 with opposition action-value functions, $\breve{Q}^t_{(k+1)}(i,a)$
* update opposite action-value functions and their opposites corresponding to the action and the opposite action, $\breve{Q}^t(i,a)$, $\breve{Q}^t(i,\breve{a})$, and $Q^t(i,\breve{a})$
13) go to step 10 if $t \neq T$; otherwise, go to the next step
14) set $year = year + 1$, if $year \leq noyears$, go to 9; otherwise, go to the next step
15) set the number of episodes $h = h + 1$, if $h \leq noepisode$ go to 7; otherwise, go to the next step
16) find the best decisions for all states

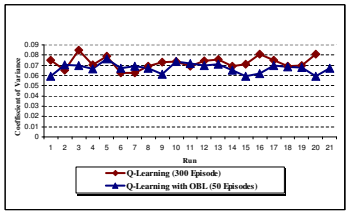

Fig. 3. Comparing the coefficient of variance of annual gain in Q-Learning and opposition-based Q-Learning type I.

TABLE IV

MAXIMUM AND MINIMUM STORAGES AND RELEASES, AVERAGE INFLOW, AND DEMAND

Value m^3	Month											
	1	2	3	4	5	6	7	8	9	10	11	12
Max. Storage	8	8	8	8	8	8	8	8	8	8	8	8
Min. Storage	1	1	1	1	1	1	1	1	1	1	1	1
Max. release	4	4	6	6	7.5	12	8.5	8.5	6	5	4	4
Min. release	0	0	0	0	0	0	0	0	0	0	0	0
Average inflow	3.4	3.7	5	5	7	6.5	6	5.5	4.3	4.2	4	3.7
Demand	3	4	5	6	5	7	4	7	5	5	5	5

TABLE V

THE BENEFIT OF RELEASE PER UNIT FOR EACH MONTH OF A YEAR

benefit ($)	Month											
	1	2	3	4	5	6	7	8	9	10	11	12
Release	1.4	1.1	1.0	1.0	1.2	1.8	2.5	2.2	2.0	1.8	2.2	1.8

TABLE VI

PERFORMANCE OF Q-LEARNING (N_E=NUMBER OF EPISODES)

code	N_E	Ave. Gain	Var. of Gain	Coeff. of var.
1	50	80.99	40.00	0.079
6	100	86.47	46.70	0.073
12	200	90.1	48.48	0.084
19	300	93.7	46.63	0.081
26	400	96.59	51.18	0.074
31	500	97.12	53.31	0.075
36	700	98.13	55.23	0.075
41	1000	98.98	55.78	0.075
47	2000	99.12	57.29	0.076

TABLE VII

PERFORMANCE OF OPPOSITION-BASED Q-LEARNING TYPE I

code	N_E	Ave. Gain	Var. of Gain	Coeff. of var.
2	50	95.04	41.11	0.067
7	100	97.12	44.62	0.069
13	200	98.47	53.88	0.075
20	300	98.94	62.21	0.079
27	400	99.44	62.59	0.079
32	500	99.54	64.78	0.081
37	700	99.68	66.21	0.082
42	1000	99.70	66.59	0.082
48	2000	99.54	65.34	0.081

TABLE VIII

PERFORMANCE OF OPPOSITION-BASED Q-LEARNING TYPE II

code	No. of episodes	No. of NN	Average of Gain	Variance of Gain	Coefficient of variance
3	50	5	91.04	59.12	0.079
4	50	2	93.04	51.62	0.077
5	50	1	94.26	40.52	0.067
8	100	10	96.37	62.09	0.084
9	100	5	96.65	63.87	0.081
10	100	2	97.33	54.33	0.069
11	100	1	96.68	46.23	0.07
14	200	20	99.58	63.18	0.08
15	200	10	99.49	63.39	0.08
16	200	4	99.45	61.44	0.079
17	200	2	99.32	61.09	0.079
18	200	1	98.94	53.87	0.075
21	300	30	100.15	64.9	0.08
22	300	6	99.89	68.68	0.083
23	300	3	99.67	64.44	0.080
24	300	2	99.41	60.36	0.078
25	300	1	99.19	58.95	0.077
28	400	2	99.71	62.33	0.079
29	400	1	99.49	60.61	0.078
33	500	5	99.74	61.26	0.078
34	500	2	99.7	60.60	0.083
35	500	1	99.42	66.20	0.78
38	700	3	99.78	67.54	0.082
39	700	1	99.67	64.42	0.080
43	1000	10	99.78	69.93	0.084
44	1000	5	99.80	67.46	0.082
45	1000	2	99.41	65.77	0.081
46	1000	1	99.12	57.29	0.076
49	2000	1	99.54	65.58	0.081

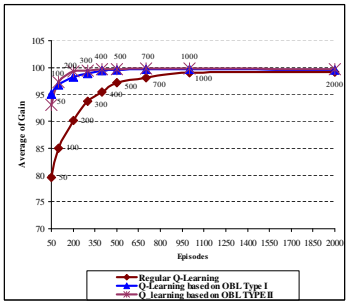

Fig. 4. Comparing the average of annual gain in Q-Learning with opposition-based Q-Learning.

(Figure 5). This means that using opposition version might be not useful during the entire learning process.

As the number of episodes increases, the average of average gain converges to steady state points for three methods with different set of parameters; however, the efficiency of opposition-based learning decreases in terms of variance and coefficient of variance (Figure 6 and 7). As illustrated in these figures, the average of average gain remains unchanged after 500 episodes for opposition versions of the learning; however, their variances preserve a constant value which is higher than the variance of the regular Q-Learning. This means that for a certain number of episodes, Q-Learning would

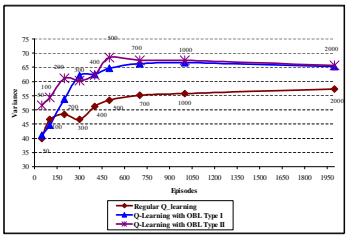

Fig. 5. Comparing the variance of annual gain in Q-Learning and opposition-based Q-Learning

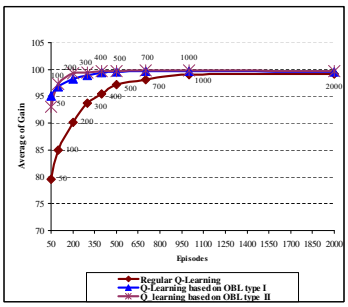

Fig. 6. The average of average of annual gain related to 20 runs for different episodes in learning methods

produce the same results as its opposition version in terms of average gain with smaller variance.

Robustness - Since the learning process is performed using simulation, the operating policy at the end of each experiment might be different. Since the policies in this application at the end of different simulation runs are, subject to sufficient iterations, almost the same, we can assume that there is only one optimal policy in this case study. Therefore, a learning method with more stable policies at the end of all experiments represents more robustness. In order to investigate the quality of Q-learning and its opposition versions in terms of robustness, the following criteria called mean and variance of distance,

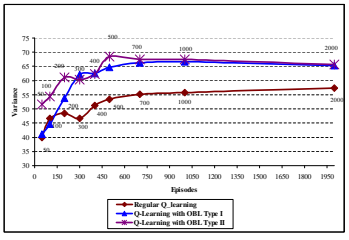

Fig. 7. The average of variance of annual gain related to 20 runs for different episodes in learning methods

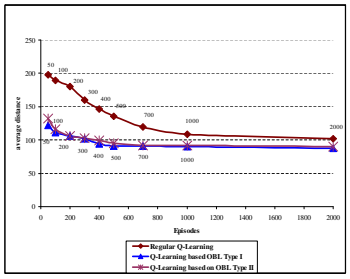

Fig. 8. The mean of distance between two policies in learning methods

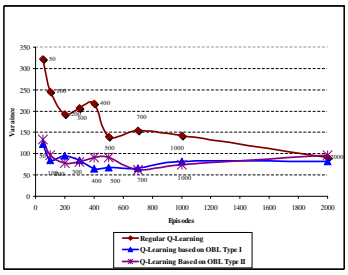

Fig. 9. The variance of distance between two policies in learning methods

$\bar{L}$ and σ_L^2:

$$L_{(k,\acute{k})} = \sum_{t=1}^{T}\sum_{i=1}^{M}(|\pi^t_{(i,k)} - \pi^t_{(i,\acute{k})}|), \quad for\ k = 1 : K = \frac{K_1!}{2!\times(K_1-2)!}, and\ \acute{k} = k+1 : K, \tag{15}$$

$$\bar{L} = \frac{\sum_{k=1,\acute{k}=k+1}^{K} L_{(k,\acute{k})}}{K}, \tag{16}$$

$$var(L) = \sigma_L^2 = \frac{\sum_{k=1,\acute{k}=k+1}^{K} (L_{(k,\acute{k})} - \bar{L})^2}{K-1}, \tag{17}$$

where K_1 is the number of experiment ($k1 = 20$) for each method, M is the number of discrete levels for the storage or the state ($M = 8$), $\pi^t_{(i,k)}$ and $\pi^t_{(i,\acute{k})}$ are two different action policies which map the state i^{th} and the period t^{th} to the optimal action a, and K is a value showing the total pairwise combinations of these action policies ($K = 190$). Therefore, in our case study, there are 190 different values for $L_{(k,\acute{k})}$ introducing the rate of difference between two different action policies. These values used to find $\bar{L}$ or σ_L^2 as two criteria showing the robustness of each method. Figures 8 and 9 illustrate the trend of changes in these values, $\bar{L}$ and σ_L^2, in Q-Learning and its opposition versions for different episodes. As it is obvious in these figures, opposition Q-Learning is more robust for small number of episodes. However, as the number of episodes increases, the mean and variance of distance in both methods decreases and converges to the steady state points. If there are multiple solutions for the problem at hand, we can substitute the $\pi^t_{(i,k)}$ with $P_{\pi^t_{(i,k)}}$ as a policy performance in equation 15.

VIII. CONCLUSIONS

It has been shown in this paper that the opposition-based Q-learning is more robust than regular Q-learning in terms of the investigated criteria. In other words, this version shortens the exploration phase and establishes a low variance gain more efficiently compared to Q-Learning. This can be particularly observed at the beginning of the learning. Furthermore, the opposite version in both types also gives an efficient policy leading to higher objective function in the simulation. This is a promising effect that could be employed in large scale applications when it is not possible to experience all possible action-state pairs for large number of learning steps. Of course, it is necessary to investigate and verify the achievements in this paper for other applications with different kind of objective functions.

REFERENCES

[1] A. Gosavi, *Simulation-based optimization: parametric optimization techniques and reinforcement learning*, Norwel, Massachusetts, U.S.A., 2003.

[2] C. J. G. H. Watkins, *learning from delayed rewards*, Dept. of psychology, University of Cambridge, England, 1989.

[3] H. R. Tizhoosh, *Opposition-based learning: a new scheme for machine intelligence*, Proceedings of the international conference on computational intelligence for modeling, control and automation, Vienna, Austria, vol. I, pp. 695-701, 2005.

[4] Daniel P. Loucks, Jery R. Stedinger, A. Douglas Haith, *Water resource systems planning and analysis*, Prentice Hall, New Jersey, U.S.A, 1981.

[5] Larry W. Mays, Tung Yeou-koung, *Hydrosystems engineering and management*, McGraw-Hill, U.S.A, 1992.

[6] S. Haykin, *Neural networks: a comprehensive foundation*, 2rd ed Prentice Hall, U.S.A, 1999.

[7] H. Robbins and S. Monro, *A stochastic approximation method*, Annals of mathematical statistics, Vol. 22, No. 3, pp. 400-407, 1951.

[8] R. S. Sutton, A. G. Barto, *Reinforcement learning: an introduction*, MIT press, Cambridge, Massachusetts, 1998.

[9] M. Mahootchi, H. R. Tizhoosh, K. Ponnambalam, *Reservoir operation optimization by reinforcement learning* , to be published in proceedings of international conference on stormwater and urban water systems modeling: contemporary modeling of urban water systems, Canada, Toronto, 2006.

[10] M. Ventresca, H. R. Tizhoosh, *Improving the convergence of backpropagation by opposite transfer functions*, to be published in International Joint Conference on Neural Networks, Vancouver, BC, Canada, 2006.

[11] H. Tizhoosh, *Opposition-based reinforcement learning*, to be published in the journal of advanced computational intelligence and intelligent informatics, 2005.

[12] S. G. Fletcher, *A new formulation for the stochastic control of systems with bounded state variables: an application to reservoir systems* , Ph.D. thesis, Dept. of systems design engineering, University of Waterloo, Canada, 1995.