Algorithms of approximate dynamic programming for hydro scheduling

Iram Parvez [1], and *Jianjian* Shen[2]*

[1]Institute of Hydropower and Hydro informatics, Dalian University of Technology, Dalian 116024, China
[2]Key Laboratory of Ocean Energy Utilization and Energy Conservation of Ministry of Education, Dalian 116024, China

Abstract. In hydro scheduling, unit commitment is a complex sub-problem. This paper proposes a new approximate dynamic programming technique to solve unit commitment. A new method called Least Square Policy Iteration (LSPI) algorithm is introduced which is efficient and faster in convergence. This algorithm takes the properties of widely used algorithm least square temporal difference (LSTD), enhance it further and make it useful for optimization problems. First value function is to find a fixed policy by using least square temporal difference Q (LSTDQ) algorithm which is similar to LSTD, then LSPI is introduced for making the policy iteration algorithm by using the results of LSTDQ. It combines the data efficiency of LSTDQ and policy-search efficiency of policy iteration.

1 Introduction

Electric power system deals with generating a suitable schedule for unit commitment (UC) problem over a day/week. The aim of this scheduling is to reduce the cost and increase the profit. The problem become complex when it considers different constraints like spinning reserve, power balance, minimum up/down time, network security constraints etc. [1]. Unit commitment involves integer decision variables which also make it more complicated [2]. There are several methods which solve unit commitment problem like dynamic programming, lagrange relaxation, mixed integer linear programming, genetic algorithms, fuzzy logics etc. Among them dynamic programming lagrange relaxation and mixed integer linear programming are commonly used. For multi period decision problems dynamic programming is good. Dynamic programming deals with the variables directly and give optimum global decision but the problem with dynamic programming occurs when it solves all the configurations of units at each step of UC problem and this problem is called "curse of dimensionality". The initial phase for the development of approximate dynamic programming is "curse of dimensionality". This field is emerging under names like

* Corresponding author: shenjj@dlut.edu.cn

heuristic dynamic programming, adaptive dynamic programming, reinforcement learning and neuro dynamic programming. The field of control theory was the first which had started the computational method for dynamic programming practical. In 1980 field of computer science proposed the name of reinforcement learning [3]. Reinforcement learning mainly emphasized on discretizing action and spaces, it also focuses on continuous problems. Some models related to dynamic management had evolved using the adaptive dynamic programming in 1990's. Most of the work done on approximate dynamic programming is in 20's by [4].

A variety of approximate dynamic programming models/algorithms are introduced in recent years. Approximate dynamic programming can solve large number of markov decision problems. The basic of approximate dynamic programming is that instead of calculating all the optimal states, this method approximate the feasible states and in this way it solves the problem of "curse of dimensionality". The paper proposes the ADP technique to solve unit commitment problem by least square policy iteration algorithm. After the enlargement of LSPI algorithm, LSTDQ is introduced first which is used in policy evaluation step of policy iteration [5-8]. LSTDQ is good in finding the value function for a fixed policy. It uses data efficiently and converge faster than other methods, but it creates a problem [9] as it oscillates in MDP between two bad policies having 4 states just in number. LSPI is good in eliminating the parameters which require careful tuning.

2 Methodology

2.1 Approximate Policy Iteration

The policy iteration algorithm is used to make policies by using value functions. This algorithm has two parts one policy evaluation and second policy improvement. In large and continuous spaces, it is difficult to find the policy evaluation, so value function is approximated. This algorithm is presented in figure 1.

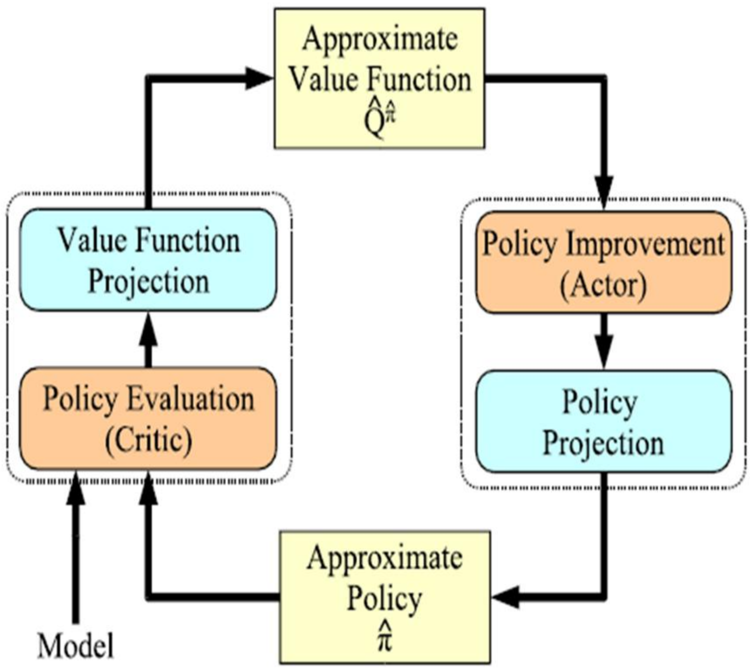

Approximate Value Function $\hat{Q}^{\hat{\pi}}$
Value Function Projection
Policy Evaluation (Critic)
Policy Improvement (Actor)
Policy Projection
Approximate Policy $\hat{\pi}$
Model

Figure 1 Approximate policy iteration [8]

2.2 Background

There are five components of Markov decision process

1. State: The state space represents the set of all possible states S_t (s_1 , s_2 ,s_3 ....s_n) which give the information at time t.
2. Actions or Decisions: $A(a_1$,a_2,a_3......$a_n)$ set of actions.
3. The transition model: P (s, a, $s^{'}$) is also called as the system model. The probability P make a transition to state $s^{'}$ while doing an action a.
4. Cost Function: When transition occurs cost function is obtained.
5. Discount Factor: For rewards in future γ [0,1] is used as a discount factor.

2.3 Least square temporal difference Q

The LSTDQ algorithm solves the linear system by using inversion of $A^{\sim}$. It uses the Basis function of state and action as shown in figure 2. It also controls the distribution of samples. It faces the singularity problem while using inversion $A^{\sim}$, this problem is minimized by using matrix of identity and it doesn't affect the convergence property of algorithm.

2.4 Least square policy iteration algorithm

The basic principle of least square policy iteration is it uses the policy iteration and LSTDQ together. It is off-policy algorithm and for generating the policies it uses the same data in every iteration. Figure 3 shows the least square policy iteration algorithm.
In Least square policy iteration algorithm, linear architecture is used for the approximation of state-action function.

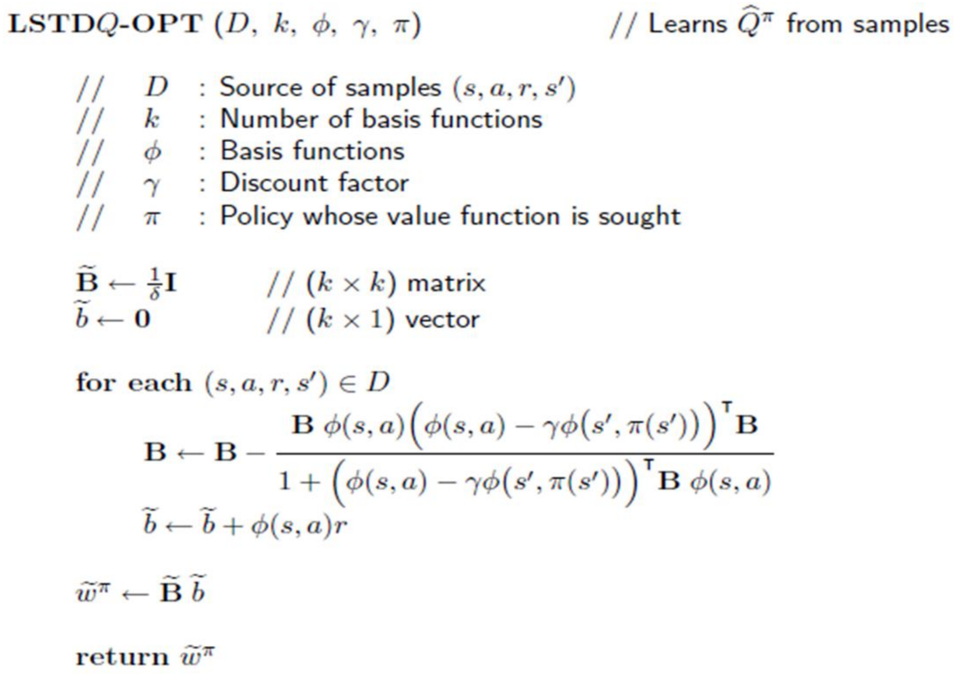
.

$$\widehat{Q}(s,a;\omega) = \sum_{i=1}^{k} \phi_i\,(s,a)\omega_i = \phi(s,a)^T\omega \qquad (1)$$

The maximization of approximate values over all actions in A gives the greedy policy π at any given state s. ω is the weight vector and φ is the basis function, argmax shows the points at which function value is maximized.

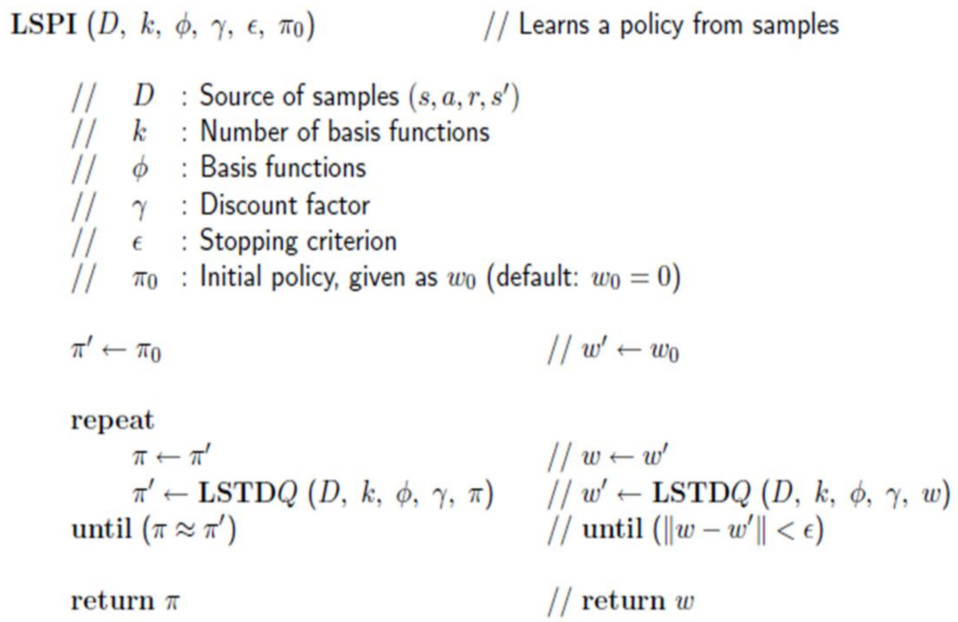
$$\pi(s) = argmax_{a\epsilon A}\widehat{Q}(s,a) = argmax_{a\epsilon A}\varphi(s,a)^T\omega \qquad (2)$$

Basis function φ and a set of parameters are used to represent policy π. Policy π along all sets of samples is fed in LSTDQ. LSTDQ is used to determine the policy π and give output ω^{π} for each s' of each sample set ($s,\ a, s'$). In every iteration samples of source D can be used for the call to LSTDQ. In each iteration of LSPI, the initial set of samples is reused and between iterations D is updated. Approximation architecture has parameters ω which is used to represent the policies and value functions. According to this, least square policy iteration is used as iteration in parameters ω. The generic bond is applied on policy iteration algorithm. LSPI also remove the error in architecture of actor-critic from the actor part.



```
LSTDQ-OPT ($D,\ k,\ \phi,\ \gamma,\ \pi$)                    // Learns $\widehat{Q}^{\pi}$ from samples

    // $D$ : Source of samples $(s,a,r,s')$
    // $k$ : Number of basis functions
    // $\phi$ : Basis functions
    // $\gamma$ : Discount factor
    // $\pi$ : Policy whose value function is sought

    $\widetilde{\mathbf{B}} \leftarrow \frac{1}{\delta}\mathbf{I}$        // $(k \times k)$ matrix
    $\widetilde{b} \leftarrow \mathbf{0}$         // $(k \times 1)$ vector

    for each $(s,a,r,s') \in D$
        $\mathbf{B} \leftarrow \mathbf{B} - \frac{\mathbf{B}\,\phi(s,a)\big(\phi(s,a) - \gamma\phi(s',\pi(s'))\big)^{\mathsf{T}}\mathbf{B}}{1 + \big(\phi(s,a) - \gamma\phi(s',\pi(s'))\big)^{\mathsf{T}}\mathbf{B}\,\phi(s,a)}$
        $\widetilde{b} \leftarrow \widetilde{b} + \phi(s,a)r$

    $\widetilde{w}^{\pi} \leftarrow \widetilde{\mathbf{B}}\,\widetilde{b}$

    return $\widetilde{w}^{\pi}$
```

Figure 2 Least square temporal difference algorithm[8]



```
LSPI ($D$, $k$, $\phi$, $\gamma$, $\epsilon$, $\pi_0$)               // Learns a policy from samples

    //  $D$   : Source of samples $(s, a, r, s')$
    //  $k$   : Number of basis functions
    //  $\phi$   : Basis functions
    //  $\gamma$   : Discount factor
    //  $\epsilon$   : Stopping criterion
    //  $\pi_0$  : Initial policy, given as $w_0$ (default: $w_0 = 0$)

    $\pi' \leftarrow \pi_0$                               // $w' \leftarrow w_0$

    repeat
        $\pi \leftarrow \pi'$                             // $w \leftarrow w'$
        $\pi' \leftarrow$ LSTDQ ($D$, $k$, $\phi$, $\gamma$, $\pi$)     // $w' \leftarrow$ LSTDQ ($D$, $k$, $\phi$, $\gamma$, $w$)
    until ($\pi \approx \pi'$)                         // until ($\|w - w'\| < \epsilon$)

    return $\pi$                                // return $w$
```

Figure 3 Least square policy iteration algorithm[8]

3 Discussion

There has been rich literature on solving unit commitment problem while considering different constraints. The algorithms and models are different even if they lie within the same category. [10-12] used augmented lagrange relaxation, sequential lagrange and MILP, lagrange relaxation decomposition and genetic algorithm, augmented lagrange relaxation, decomposition techniques (block coordinate descent and auxiliary problem principle), dynamic programming stochastic lagrange relaxation methods. The execution time and number of time stages varies linearly in lagrange relaxation. Dynamic programming with heuristic techniques is very efficient during imprecise hourly loads. Least square policy iteration is a new model free algorithm and is mainly used for the minimization of curse of dimensionality. It is good in eliminating the error, it has no parameters for tuning. Unlike other algorithms LSPI collect sample at one time and reused it at every iteration.

4 Conclusion

In this paper, least square policy iteration is presented as a new algorithm for solving curse of dimensionality in unit commitment problem. Least-squares policy iteration (LSPI) learns fixed policy approximation of state-action value function. First LSPI calls for least square temporal difference Q (LSTDQ) then LSPI uses the results of LSTDQ for making policy iteration algorithm. It is recommended that LSPI can be applied on UC problem as it converges relatively small number of trials with randomly selected actions. However, LSPI is a model free algorithm and there is no risk of overshooting as LSPI doesn't take gradient steps. Also, the function approximations are better used by LSPI with faster converge.

Numerous studies recommended the LSPI but further studies should be carried out on implementation of this algorithm on real world unit commitment problem. However, the extension of this work is suggested for the future to apply this algorithm for a specific case study and validate the algorithms. Also, the comparison with literature should be considered in future in order to get meaningful outcomes applicable for that particular application.

5 Funding

This research was supported by The National Natural Science Foundation of China (No. 51579029) and Fundamental Research Funds for the Central Universities (DUT19JC43). The authors would like to acknowledge the financial support from China scholarship council (CSC) for higher studies of Iram Parvez at Dalian University of Technology, China.

References

1. Séguin, S., P. Côté, and C. Audet, *Self-scheduling short-term unit commitment and loading problem.* IEEE Transactions on Power Systems, 2015. 31(1): p. 133-142.
2. Zheng, Q.P., J. Wang, and A.L. Liu, *Stochastic optimization for unit commitment—A review.* IEEE Transactions on Power Systems, 2014. 30(4): p. 1913-1924.
3. Sutton, R.S., *et al. Fast gradient-descent methods for temporal-difference learning with linear function approximation*. in *Proceedings of the 26th Annual International Conference on Machine Learning*. 2009. ACM.
4. Powell, W.B., *Approximate Dynamic Programming: Solving the curses of dimensionality*. Vol. 703. 2007: John Wiley & Sons.
5. Buşoniu, L., *et al. Online least-squares policy iteration for reinforcement learning control*. in *Proceedings of the 2010 American Control Conference*. 2010. IEEE.
6. Lazaric, A., M. Ghavamzadeh, and R. Munos, *Finite-sample analysis of least-squares policy iteration.* Journal of Machine Learning Research, 2012. 13(Oct): p. 3041-3074.
7. Bertsekas, D.P. and J.N. Tsitsiklis, *Neuro-dynamic programming*. Vol. 5. 1996: Athena Scientific Belmont, MA.

   Lagoudakis, M.G. and R. Parr, *Least-squares policy iteration.* Journal of machine learning research, 2003. 4(Dec): p. 1107-1149.
8. Koller, D. and R. Parr. *Policy iteration for factored MDPs*. in *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*. 2000. Morgan Kaufmann Publishers Inc.
9. Wang, S., *et al.*, *Short-term generation scheduling with transmission and environmental constraints using an augmented Lagrangian relaxation.* IEEE Transactions on Power Systems, 1995. 10(3): p. 1294-1301.
10. Frangioni, A., C. Gentile, and F. Lacalandra, *Sequential Lagrangian-MILP approaches for unit commitment problems.* International Journal of Electrical Power & Energy Systems, 2011. 33(3): p. 585-593.
11. Virmani, S., *et al.*, *Implementation of a Lagrangian relaxation based unit commitment problem.* IEEE Transactions on Power Systems, 1989. 4(4): p. 1373-1380.