Available online at www.sciencedirect.com

ScienceDirect

Energy Procedia 122 (2017) 415–420

Energy Procedia

www.elsevier.com/locate/procedia

CISBAT 2017 International Conference – Future Buildings & Districts – Energy Efficiency from Nano to Urban Scale, CISBAT 2017 6-8 September 2017, Lausanne, Switzerland

Smart Cities (Urban Simulation, Big Data)

Balancing comfort and energy consumption of a heat pump using batch reinforcement learning with fitted Q-iteration

José Vázquez-Canteli[a], Jérôme Kämpf[b], Zoltán Nagy[a]*

[a] *Intelligent Environments Laboratory, Department of Civil, Architectural and Environmental Engineering, The University of Texas at Austin, Austin, TX, USA*
[b] *ENERGY Institute, Haute Ecole d'Ingénierie et d'Architecture de Fribourg (HEIA-FR), Fribourg, Switzerland*

Abstract

In this study, a heat pump satisfies the heating and cooling needs of a building, and two water tanks store heat and cold respectively. Reinforcement learning (RL) is a model-free control approach that can learn from the behaviour of the occupants, weather conditions, and the thermal behaviour of the building in order to make near-optimal decisions. In this work we use of a specific RL technique called batch Q-learning, and integrate it into the urban building energy simulator CitySim. The goal of the controller is to reduce the energy consumption while maintaining adequate comfort temperatures.

Keywords: *artificial intelligence, energy management, thermal comfort, building simulation, energy storage*

* Corresponding author. Tel.: +1-512-897-0841;
E-mail address: nagy@utexas.edu

Nomenclature

S	states	Q	action-value	γ	discount factor
A	actions	V	state-value	π	policy
R	rewards	α	learning rate	ANN	Artificial Neural Network
S'	States at the following time-step	P	transition probability	MDP	Markov Decision Process
T	Boltzmann exploration constant	RL	Reinforcement Learning		

1. Introduction

Space heating is a large portion of the energy that buildings consume. In the residential sector, buildings account for approximately 30% of the global energy consumption, which grew at an average annual rate of 1.8% a year between 1971 and 2010 [1]. Some advanced control approaches such as Model-Predictive Control (MPC) are often too costly to implement in small residential settings because they require identifying the dynamic model of the system to be controlled. On the other hand, model-free control approaches allow a simpler implementation without the need for a thermal model of the building. Reinforcement learning is a type of model-free controller that is able to adapt to the changing environmental conditions, including not only weather conditions but also human preferences and behavior.

Reinforcement learning has been proved to be an effective algorithm for optimal energy control in low exergy buildings [2]. In this paper, we propose a batch reinforcement learning (BRL) controller with fitted Q-iteration to minimize the energy consumption of a heat pump that supplies heat and cooling to two water tanks. These tanks store and supply the energy to a building. The BRL controller learns from the outdoor temperatures, indoor temperature of the building, and the temperature of the water tanks in order to estimate the best times to provide additional heating or cooling energy. In order to test the suitability of the proposed controller, we implement it in CitySim, a building energy simulator developed at EPFL that computes an estimation of the energy demand for heating, cooling, and lighting of every building [3]. This approach allows evaluating the performance of the controller under different weather scenarios and in buildings of different characteristics.

1.1. Reinforcement learning

Reinforcement learning can be formalized using a Markov Decision Process (MDP). An MDP contains four elements: a set of states S, a set of actions A, a reward function R: S x A and transition probabilities between the states P: S x A x S ϵ [0,1]. The policy π then maps states to actions as π: S→ A, and the value function $V^{\pi}(s)$ of a state s is the expected return for the agent when starting in the state s and following the policy π, i.e.

$$V^{\pi}(s) = r\big(s, \pi(s)\big) + \gamma \sum P(s, \pi(s), s')\, V^{\pi}(s') \quad (1)$$

where r is the reward received for taking the action $a = \pi(s_k)$, and $\gamma \in [0,1]$ is a discount factor for future rewards. An agent that uses $\gamma = 1$ will give greater importance to seeking long term rewards, whereas an agent using $\gamma = 0$ will assign a greater value to states that lead to high immediate rewards. Reinforcement learning is particularly useful when the model dynamics (P and R) are not known, and have to be determined or estimated through interaction of the agent with the environment as depicted in Fig. 1 (a). Two approaches can be used to determine the values V^{π} of every state. In the model-based approach, the rewards and transition probabilities of the model are first learned, and then used to find the values by solving the system of equations represented by eq. (1). In the model-free approach, the agent learns the values associated to every (s, a) pair without explicitly calculating the transition probabilities or the expected rewards [4].

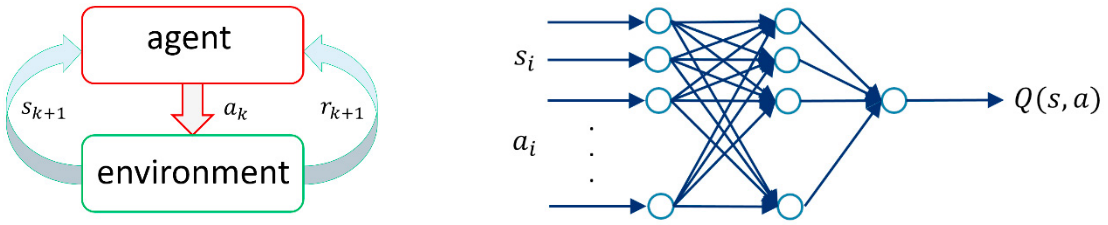

Figure 1. (a) Agent and environment interaction; (b) Artificial Neural Network for fitted Q-iteration

In this paper we focus on the use of Q-Learning, which is the most widely used model-free reinforcement learning technique due to its simplicity [5]. In simple tasks with small finite state sets, all transitions can be represented with a table storing state–action values, or q-values. Each entry in the table represents a state–action (s,a) tuple, and q-values are updated as

$$Q_{k+1}(s,a) = Q_k(s,a) + \alpha[r(s,a) + \gamma \max Q(s',a') - Q(s,a)] \tag{2}$$

$\alpha \in (0,1)$ is the learning rate, which explicitly defines to what degree new knowledge overrides old knowledge. For $\alpha = 0$, no learning happens, while for $\alpha = 1$, all prior knowledge is lost at every iteration. With this tabular approach, only one q-value is updated at a time and it requires states and actions to be discrete. When the state-action space is larger, and has continuous values, the Q-table is substituted by an Artificial Neural Network (ANN) [6] that maps states and actions directly to their q-values as Fig. 1 (b) illustrates. Instead of updating the q-values, the weights of the network are updated. This is the approach taken in this paper.

2. Methodology

We consider a heat pump and a water tank for supply and storage of heating and cooling to the building. Every time step, the tank provides an amount of heat to the building that is sufficient to maintain its interior air temperature at 21 C. Therefore, the heat pump supplies the tank with the heat it needs to vary its temperature from its current value to a target temperature, plus the amount of energy that the building will extract from the tank at that time step. With a traditional death-band rule based controller, and assuming winter conditions, the heat pump would provide the tank with heat only after its temperature fell below $Tmin$. The amount of heat provided would increase the temperature of the tank to $Tmax$, while satisfying the heating needs of the building, which would extract heat from the tank in the same period.

2.1. Definition of the states

Our approach is to replace the values $Tmin$ and $Tmax$ by a single parameter $Ttarget$, which is the action of the reinforcement learning controller, and it is proportional to the heat stored in the tank. Since the objective of the controller is to minimize the energy consumption of the heat pump, the penalty (or negative reward) is the amount of electrical energy consumed by the heat pump every hour. In order to account for any possible thermal discomfort, another penalty is included to account for any indoor temperature that happens to fall below 21 C. In the reinforcement learning problems, the actions and states should be ideally chosen such that the expectancy of receiving a reward under a given state and taking a given action is constant. In this system, the variables that contribute the most to the electrical consumption of the heat pump are: heat demand of the building, $Ttarget$ of the heat tank, current temperature of the heat tank, and ambient outdoor temperature. Of these variables, the heat demand of the building is highly dependent on the ambient outdoor temperature, the indoor temperature, and the internal and solar heat gains, which are correlated to the hour of the day. Fig. 2 shows the relationship between the penalty that the learning agent receives (proportional to the electricity consumption) and the different variables that are relevant in the determination of such penalty. Those variables are, therefore, used as states and actions. We also defined an additional state s_5 which is the outdoor temperature 22 hours ago, and which is used as an estimator of the forecasted temperature 2 hours ahead.

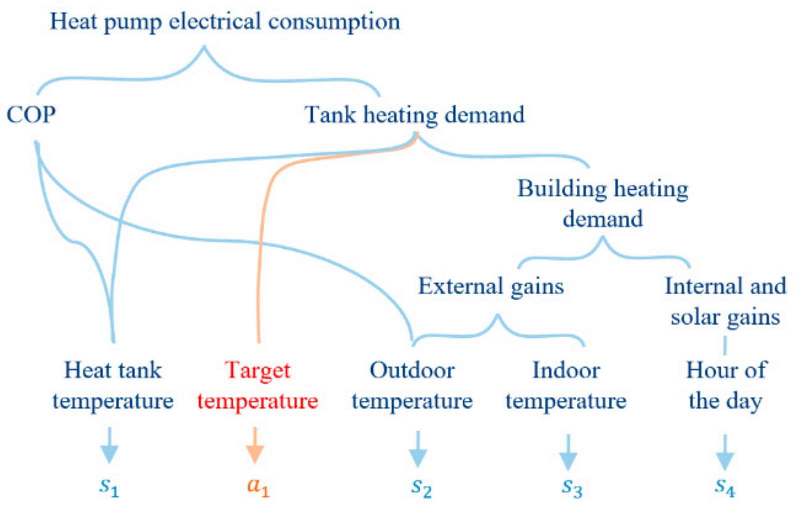

Figure 2. State-action space definition.

2.2. Action selection

The ϵ-greedy policy and Boltzmann action-selection are two of the most common approaches to select the actions in the state-action space. Boltzmann action-selection selects the actions with a probability that is related to its q-value following eq. (3). Actions with higher q-values are more likely to be selected than actions with lower q-values. T is the Boltzmann temperature constant. High values of T (i.e. T > 3) make the probability of selecting any action very homogeneous regardless of their q-values. Low values of T (i.e. T < 0.1) lead the action-selection process towards a greedy policy, in which actions with the highest q-values are selected most of the time. It is convenient to start the learning process with high values of T in order to increase exploration, and then reduce T in order to exploit the acquired knowledge to maximize the rewards obtained. The probability of selecting a given action a under a state s using Boltzmann action-selection equation is represented in eq. (3).

$$\Pr(a|s) = \frac{e^{\frac{Q(a,s)}{T}}}{\sum e^{\frac{Q(s,e)}{T}}} \quad (3)$$

In our simulations we gradually reduced T from 3.0 to 0.02 in order to make the controller evolve from an almost random search policy towards a greedy policy in which best actions are exploited.

2.3. Update of q-values

Instead of updating the q-values for each state-action tuple in a tabular approach, we used an ANN to map the different combinations of actions and states to their respective q-values. In this case, q-values are updating using eq. (2) and using $\alpha = 1$. The resulting q-values are then used as targets to fit the regression model using the ANN. The experience is gathered in batches D_k that contain the tuples (s_i, a_i, r_i, s_{i+1}). Then a batch is completed, the q-values are updated using all the available batches and the ANN re-trained. Table 1 contains the sequence of the algorithm we have implemented of batch reinforcement learning with fitted Q-iteration for a non-episodic task.

Table 1. Batch reinforcement learning with fitted Q-iteration

Algorithm 1: Batch reinforcement learning with fitted Q-iteration for non-episodic task

```
1:  Initialization of all states $s$ and actions $a$
2:  $Q \longleftarrow Q_0$   // q-values are initialized to small random values and the ANN is trained
3:  trainANN(S, A, Q)   // Train the ANN to initialize the weights
4:  $D \longleftarrow 0$   // Reset batch of states, actions and rewards
5:  Repeat for $k_0 = 0$ until k < number of batches
6:      Repeat for $i_0 = 0$ until i < experiences per batch
7:          scaleStates()
8:          $a_i \longleftarrow actionSelection(s_i, Q_k)$
9:          unscaleStates()
10:         $s_{i+1} \longleftarrow getStates(s_i, a_i)$
11:         $r_i \longleftarrow getRewards(s_i, a_i, s_{i+1})$
12:         $D_k \longleftarrow (s_i, a_i, r_i, s_{i+1})$
13:         i++
14:     end
15:     scaleStates()
16:     $Q_{k+1}(s,a) = r(s,a) + \gamma \max Q_k(s',a')$   // Update q-values using data from all previous batches D
17:     trainANN($S, A, Q_{k+1}$)   // Train network with states, actions and q-values from all the batches D
18:     unscaleStates()
19:     k++
21: end
```

3. Results and discussion

In order to test the controller, we simulated our building under winter conditions in Austin, TX. using a typical day January the 1st. The controller was trained and tested for that typical day in order to test its learning capabilities. Fig. 3 illustrates the behavior of the temperature of the heat tank with respect to the outdoor air temperature T_{out}. One of the main difficulties this controller faces is achieving a proper balance between immediate and future rewards. Since there is a heat storage tank, the increase of its target temperature will automatically lead to a penalty associated with the increased electricity consumption. However, if the target temperature is increased during a period of high outdoor temperature (for winter conditions), the heat pump will store energy when its COP is high. This behavior would increase the expected rewards in the long term even if the immediate rewards are negative. This trade-off between immediate and future rewards is managed with the discount factor γ, which is present in eq. 2.

Fig. 3 (a) and (b) show how increasing the discount factor reduces the dependency of the q-values on the immediate rewards r_k, while it assigns greater importance to achieving long-term rewards. This is particularly important in this system in which there is a delay between the action and the perceptible benefit from taking such action. In Fig. 3 (b), q-values are more correlated to the outdoor temperature T_{out}, and therefore learn better from the variation of the heat pump COP and the electrical consumption.

One limitation of this controller is that the action (target temperature of the tank) is not directly related to the electricity consumption of the heat pump, which also depends on the building heating demand. Although we have included as states some variables that are useful to estimate the heating needs of the building, further work will explore the use of the heat flow from the heat pump as the action of the controller.

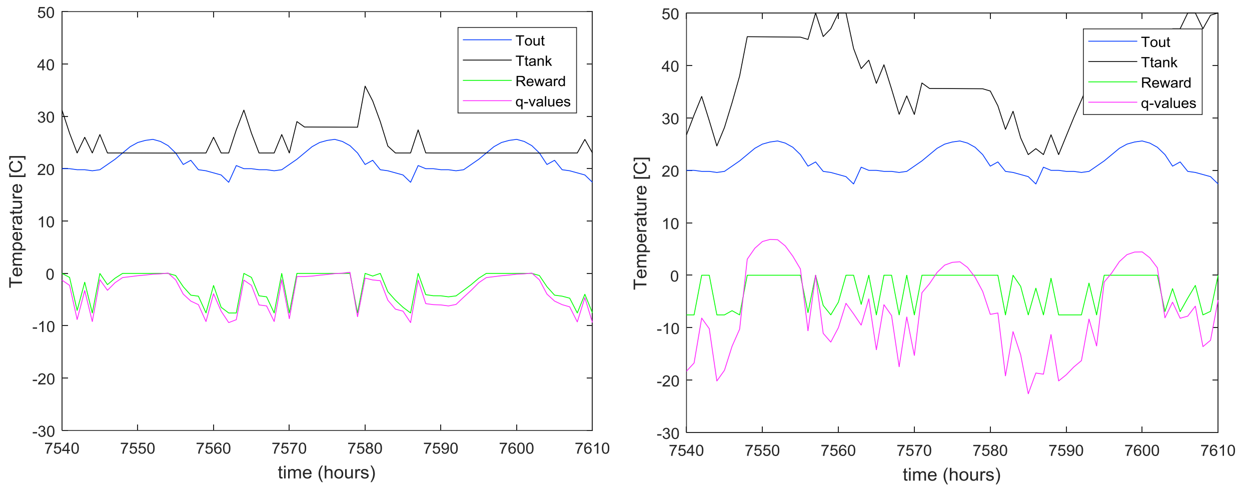

Figure 3. (a) simulation results for $\gamma = 0.25$; (b) simulation results for $\gamma = 0.75$

4. Conclusion

Model-based control systems can be costly to implement, which is a major drawback that prevents certain types of controllers such as MPC from being widely implemented in a residential setting. Reinforcement learning, by contrast, is a model-free control that learns from the environment where it is implemented and adapts its behaviour accordingly. When sufficient amount of data is available for training, it can reach near optimal solutions. In order to increase the speed of learning, we have used an ANN to map the state-action space to the Q-function. This allows the controller to work with continuous states and actions, and also to speed up the learning process. We have also observed that the definition of the discount factor plays a critical role on the results. The reason is that the storage tank created a delay between the current actions and the long-term rewards. Increasing the discount factor improves the results by assigning a greater importance to future rewards.
Additionally, the integration of this controller in a building energy simulator will allow us to test it under various weather conditions and for different buildings in order to test its suitability for widely implementation in a residential built-environment.

References

[1] International Energy Agency Agency. Transition to Sustainable Buildings. 2013.
[2] Yang, L., Nagy, Z., Goffin, P., Schlueter, A., Reinforcement learning for optimal control of low exergy buildings. Applied Energy, 2015
[3] Robinson, D., Computer modelling for sustainable urban design, Earthscan: London, p. 121, 2011.
[4] Barto, A.C., Sutton R.S., Reinforcement Learning: An Introduction, 1998.
[5] Watkins, CJCH., Dayan, P., Technical Note: Q-Learning. Machine Learning 1992; 8:279–92.
[6] Busconiu, L., Babuska, R., De Shutter, B., Ernst, D., Reinforcement Learning and Dynamic Programming Using Function Approximators, Automatic and Control Engineering Series, 2010.