

SYSTEMS AND METHODS FOR OPTIMIZATION OF DATA CENTER COOLING

TECHINICAL FIELD

5 The present disclosure relates to the optimization of control of data center cooling. In particular, the present disclosure relates to the use of reinforcement learning in the optimization of control of data center cooling.

BACKGROUND

10

With the emergence and proliferation of services and applications such as cloud computing and social networks, data centers (DCs) play an increasingly important role. DCs consume a huge amount of energy and a large portion of this energy is used in cooling. Improving cooling efficiency while satisfying cooling requirements is
15 critical for service stability and saving energy resources. In this research area, canonical solutions are expert based and optimization usually takes place at the design stage. At the design stage of the data center, the experts on thermal management will deduce some operational settings, such as the airflow rate and supply temperature according to design load of the DC.

20

Figure 1 shows an example of a cooling control optimization approach in a data center. The data center 100 has a rack 110 which is cooled by a cooling system 120. The cooling system 120 comprises a coil 122 and a fan 124. The coil 122 is cooled by a supply of chilled water 132 which is chilled to 7 degrees. After cooling the coil
25 122, the water is returned as return water 134. A proportional-integral-derivative (PID) controller 136 controls a valve to regulate flow of the supply of chilled water 132. The fan 124 directs a cold airflow 142 which is cooled by the coil 122 to the rack 110. A return hot airflow 144 carries heat from the rack 110 to the coil 122. A second PID controller 146 controls the airflow at a fixed supply temperature.

30

As described above, the two PIDs 136 and 146 control the cooling of the data center. However, the airflow and chilled water flow are computed at the design stage of the data center cooling system and do not take into account factors such as cooling load.

Thus if the operating level of the data center deviates from the original design the cooling control may be non-optimal.

Cooling energy optimization involves the control of a sophisticated cooling system, which consists of multiple components, such as cooling tower, chiller, and ventilation system, etc. A common practice of DC cooling system control is to adjust the set-points, i.e., the target values of different control variables. For example, by setting the temperature control variable at the outlet of an air conditioner to the desired value, the air conditioner can adjust its internal state to meet the set-point by consuming a certain amount of energy. An optimal selection of these set-points can be challenging, as the process relies on the knowledge of the cooling system, from thermal dynamics to mechanics.

SUMMARY OF THE INVENTION

According to a first aspect of the present disclosure, a reinforcement learning method for optimizing control parameters for a data center cooling system is provided. The method comprises: initializing an actor neural network configured to generate an indication of a subsequent control action in response to an actor neural network input vector comprising an indication of a current state of the data center cooling system; initializing a critic neural network configured to generate a cost indication in response to a critic neural network input vector comprising an indication of the current state of the data center cooling system and a subsequent control action; receiving trace data indicating control actions, previous states of the data center cooling system, and corresponding temperature data and energy usage data for the data center cooling system; training the critic neural network by determining a set of parameter weights for the critic neural network which minimizes error between an output of the critic neural network and reward data corresponding to the temperature data and energy usage data for the data center cooling system; training the actor neural network by determining a set of parameter weights for the actor neural network which minimize the cost indication generated by the critic neural network; and using the trained actor neural network to generate a set of control parameters for the data center cooling system.

In an embodiment, critic neural network comprises a plurality of layers and a second last layer of the plurality of layers is configured to output predicted temperature data and predicted energy usage data, and a last layer of the plurality of layers is configured to evaluate a cost function of the predicted temperature data and predicted energy usage data to generate the cost indication, wherein determining a set of parameter weights for the critic neural network comprises determining a set of parameter weights that minimize the error between the predicted temperature data and predicted energy usage data and the corresponding temperature data and energy usage data for the data center cooling system.

In an embodiment, the cost function of the predicted temperature data and the predicted energy usage data is selected to minimize the predicted energy usage subject to satisfying a temperature constraint.

In an embodiment, the cost function comprises a penalty factor which controls a trade-off between predicted energy usage and the temperature constraint.

In an embodiment, the actor neural network input vector comprises indications of control settings for past time slots and wherein training the actor neural network further comprises calculating a cooling cost function which is dependent on the difference of the outputs of the actor neural network for consecutive time slots.

In an embodiment, the method further comprises dividing the trace data into training data and validation data, and using the validation data to calculate a validation error for sets of parameter weights for the critic neural network and the actor neural network, and selecting a set of parameter weights for the critic neural network and the actor neural network using the validation error.

In an embodiment, using the validation data to calculate a validation error comprises calculating an underestimation error and selecting a set of parameter weights for the critic neural network and the actor neural network comprises using the underestimation error.

an embodiment, the control actions comprise a plurality of temperature set points for the data center cooling system.

5 In an embodiment, training the critic neural network and / or training the actor neural network comprises using gradient decent.

According to a second aspect of the present disclosure a cooling control optimization system for a data center is provided. The cooling control optimization system comprises a processor and a data storage device. The data storage device stores
10 computer executable instructions operable by the processor to: initialize an actor neural network configured to generate an indication of a subsequent control action in response to an actor neural network input vector comprising an indication of a current state of a data center cooling system of the data center; initialize a critic
15 neural network configured to generate a cost indication in response to a critic neural network input vector comprising an indication of the current state of the data center cooling system and a subsequent control action; receive trace data indicating control actions, previous states of the data center cooling system, and corresponding temperature data and energy usage data for the data center cooling system; train the
20 critic neural network by determining a set of parameter weights for the critic neural network which minimizes error between an output of the critic neural network and reward data corresponding to the temperature data and energy usage data for the data center cooling system; train the actor neural network by determining a set of parameter weights for the actor neural network which minimize the cost indication generated by the critic neural network; and use the trained actor neural network to
25 generate a set of control parameters for the data center cooling system.

According to a yet further aspect, there is provided a non-transitory computer-readable medium. The computer-readable medium has stored thereon program instructions for causing at least one processor to perform operations of a method
30 disclosed above.

BRIEF DESCRIPTION OF THE DRAWINGS

In the following, embodiments of the present invention will be described as non-limiting examples with reference to the accompanying drawings in which:

5 Figure 1 shows an example of a cooling control optimization approach in a data center;

Figure 2 illustrates a typical reinforcement learning framework;

10 Figure 3a shows a deep reinforcement learning framework;

Figure 3b shows a deep reinforcement learning framework of embodiments of the present invention;

15 Figure 4 is a block diagram showing functional modules of a cooling control optimization system according to an embodiment of the present invention;

Figure 5 illustrates a system architecture of a data center cooling system model;

20 Figure 6 illustrates the main components of two data center cooling systems;

Figure 7 is a flowchart showing a method of optimizing cooling control settings for a data center according to an embodiment of the present invention;

25 Figure 8 shows the architecture of a critic neural network and an actor neural network in an embodiment of the present invention;

Figure 9 is a block diagram showing a technical architecture of a cooling control optimization system according to an embodiment of the present invention;

30 Figure 10 shows a comparison of simulation results from methods of optimizing cooling control;

Figures 11a to 11c show a comparison of example power usage effectiveness (PUE) and temperature traces obtained from a simulation;

Figure 12 shows the testing of different settings for a penalty factor in embodiments of the present invention;

5 Figure 13 shows a 3-D model of a target system on which embodiments of the present invention were tested;

Figure 14 shows the results for testing different settings for the target system shown in Figure 13;

10

Figure 15 shows estimated temperature curves with and without a de-underestimation validation method used in embodiments of the present invention;

Figure 16 shows energy saving for different values of a penalty parameter in
15 embodiments of the present invention; and

Figures 17a and 17b show the effects of changing the value of a penalty parameter on airflow rate and temperature in embodiments of the present invention.

20 DETAILED DESCRIPTION

The present disclosure relates to the application of reinforcement learning to the optimization of data center cooling control.

25 Figure 2 illustrates a typical reinforcement learning framework. Reinforcement learning (RL) deals with agents that learn to take better actions directly from experience of interacting with the environment. In Figure 2, an agent 210 receives data indicative of a state 204 which is observable data that will be used to make a control decision relating to an action to be taken 212 on a data center and its cooling
30 system 202. The agent also evaluates a reward 206 which is the evaluation of taking an action at a certain state. RL uses trace data containing (state, action and reward) to train the agent 210 to optimize the control settings.

Figure 3a shows a deep reinforcement learning framework. Deep reinforcement learning (DRL) uses deep neural networks to implement RL. In the DRL implementation shown in Figure 3a, an actor network 302 is trained to select an action a_t from an observed state s_t . The actor network 302 is trained using a critic network 304. The critic network 304 is trained to evaluate a cost or reward r_t of performing action a_t from an observed state s_t .

Figure 3b shows a deep reinforcement learning framework of embodiments of the present invention. Similar to the framework shown in Figure 3a, the framework also includes an actor network and a critic network. However the design of the framework is different. In the framework of embodiments of the present invention, the design of the critic-network 308 and the actor-network 306, at time t , utilize the action data a_{t+1} and reward data r^1_{t+1}, r^2_{t+1} at time slot $t+1$ to evaluate an action to be taken at the next time slot $t+1$. Such design can be used to solve the control delay problem, as there is a time delay to execute an action in practice. Thus, the framework inherits a prediction procedure as in the state action evaluation; the state is set to a recent history, while the reward is set to the future cost. In this case a standalone prediction network is not required. Additionally, the framework shown in Figure 3b includes a dual reward architecture r^1_{t+1}, r^2_{t+1} in the critic-network 308, a cost function is utilized to combine the two reward items r^1_{t+1}, r^2_{t+1} . The cost function c_{t+1} is specially designed to make sure the energy cost will be minimized while the cooling effect is good.

Figure 4 is a block diagram showing functional modules of a cooling control optimization system according to an embodiment of the present invention. As shown in Figure 4, the cooling optimization system 310 comprises an input module 324a, an output module 324b, and a neural network module 324c. The input module 324a is used to format the input data into the form which can be directly used in the training task. The output module 324b outputs control settings determined from the trained neural networks to a database for use in controlling the cooling of the data center. The neural network module 324c comprises the storage for the actor network 306 and the critic network 308. The neural network module 324c further comprises a training component 324d which trains the neural networks using the input trace data and a validation component 324e which is used to select optimum parameter

weights for the neural networks. A technical architecture of a cooling control optimization system is described below with reference to Figure 9.

To study the cooling control optimization problem, in this section we utilize a simulation model to present a cooling control optimization problem formulation. The simulation model is based on the widely adopted building energy simulation platform EnergyPlus. Although the model is largely simplified, it does capture the major cooling dynamics and is thus adequate for studying the cooling control optimization.

Figure 5 illustrates a system architecture a data center cooling system model. As illustrated in Figure 5, the model 500 comprises a first server zone 510 and a second server zone 520 and their associated cooling systems. The two zones are different in size, location, and their corresponding cooling systems. The first zone has a DX cooling system 512 and the second zone has a chiller cooling system 522.

In the following, we describe the cooling system with a focus on identifying the state space (parameters that characterize the system), the action space (control variables), and the reward counters (optimization objectives), while omitting the details of the facility structures and operation processes that are unrelated to problem setup (yet might be critical parts of the overall system).

The DC 500 has two server zones 510 520 placed side by side, z1 and z2, with each zone being a standalone server room. The two zones are different in size ($15.24 \times 15.24 \text{m}^2$ and $15.24 \times 17.00 \text{m}^2$ for z1 and z2, respectively) but similar in other construction aspects. The heat in each zone is generated by IT equipment (ITE) and other sources (such as illuminations), with ITE as the dominant heat source. The load of the ITE is defined as αL , where L is the designed load density (per square meter) and α is a load factor that varies at different time slots. In our simulation, we use a public trace collected from Wikimedia to set α to be the same for z1 and z2 while using different load density 4kw and 2kw for z1 and z2 respectively. The heat generated by the illumination is assumed to be a constant (per square meter) as the lights inside the DC are on all the time.

In the context of the RL framework, we use a tuple of workload level and the ambient temperature to represent the state, since both of them affect the cooling load. We use the tuple of PUE and the IT equipment outlet temperature of each zone to compute the reward. In the context of the DC cooling, PUE needs to be minimized and the outlet temperature needs to be kept within a specific range.

In the target simulation model, z1 and z2 are equipped with different cooling systems: direct expansion (DX) cooling system 512 for z1 and chilled water (chiller) cooling system 522 for z2. Both cooling systems are supplied with cool water from a cooling tower, but they use the cool water in different ways. In the DX system 512, the cool water passes through coils and cool down the airflow passing over the coils. In the chiller system 522, the cool water is used first to refrigerate another water stream (chilled water), which in turn cools down the airflow supplied to the DC.

Figure 6 illustrates the main components of two data center cooling systems. In the DX cooling system 512, the intake ambient air flow is first cooled by two types of evaporative coolers: directive (DEC) and indirect (IEC), and then passes over the DX cooling coils and is further fed to the first zone 510. The underlying control algorithm in EnergyPlus (referred as the DefaultE+ control algorithm) uses the following five set points to control the cooling system: DEC outlet temperature (airflow) T_{dec} , IEC outlet temperature (airflow) T_{iec} , chilled water loop outlet temperature (water flow) T_{cw} , DX cooling coil outlet temperature (airflow) T_{dx} , and chiller cooling air loop outlet temperature (airflow) T_{ch} .

We formulate the cooling control optimization problem as follows. We are given a time-varying tuple of the ambient air temperature T_{amb} and the load factor H_{ite} . The problem is to determine the values of five control set-points to minimize the objective function as stated in Eq. (1) below. The objective function aims to strike a balance between minimizing the PUE and preventing overheating in the server zone. In particular, the objective function consists of two parts: the first part is PUE (denoted as ϵ_{PUE}), which is to be minimized; the second part accounts for the penalty of the overheating (for both z1 and z2). The penalty function has the form of $\lambda \cdot \ln(1 + \exp(T_{zi} - \varphi))$, for , with λ , T_{zi} , and φ denoting the penalty pricing factor, average IT

equipment outlet temperature of zone i , and overheating threshold, respectively. The penalty term takes the standard form of the soft plus activation function which has been implemented in most deep learning frameworks thus easy to be implemented. During the training, we minimize this cost function (reverse of the reward), since
 5 commonly the training optimization algorithms are designed for minimization.

Figure 7 is a flowchart showing a method of optimizing cooling control settings for a data center according to an embodiment of the present invention. The method shown in Figure 7 may be carried out on the cooling control optimization system 310
 10 described above with reference to Figure 4.

In step 702, the input module 324a of the cooling control optimization system 310 receives an offline data trace which comprises historical data series of the state, action and reward data. The data trace comprises entries for state ($T_{amb}(t)$, $H_{ite}(t)$),
 15 action ($T_{dec}(t)$, $T_{iec}(t)$, $T_{cw}(t)$, $T_{dx}(t)$ and $T_{ch}(t)$), and reward data $y(t)$.

In step 704, the input module 324a of the cooling control optimization system 310 organizes the trace data received in step 702 and divides the data into training data and validation data. The data is organized into the form of X_Q , X_μ , and y_r , where X_Q is
 20 the input of the Q-neural network Q , X_μ is the input of the actor network μ , and y_r is the energy (PUE) and temperature readings that can be used to compute the loss data y according to the following equation:

$$y = \epsilon_{PUE} + \lambda \cdot \ln(1 + \exp(T_{out}^{Zone} - T_{thresh})) \quad (\text{equation 1})$$

25 Where y is the cost function (or reward function) when the data center has a thermal zone, ϵ_{PUE} is the measured PUE (which is equal to the measured cooling energy cost), T_{out}^{Zone} is the average temperature in the thermal zone, and T_{thresh} is a pre-defined threshold value of the zone temperature, and λ is a penalty factor which may be manually tuned.

30 Different to canonical RL approach in which the future reward data are also included in the evaluation of the action (discounted return), here the future reward information

is not used, as the workload and weather trends determine the system transition. Note that as there will be an affecting time for an action to take effect, we shrink the state observation back for one time slot, such that the action we computed based on the current state is going to take effect in the next time slot. All these data are prepared as time series of N time steps, which are further divided into training data and validation data. The training data is used for training of the neural networks and the validation data is used to validate the neural networks to find the best parameter settings.

In step 706, the training component 324d of the neural network module 324c initializes the critic network 308 and the actor network 306. Before the training starts, we first initialize two neural networks. The critic network 308 (parameterized by θ) approximates the Q-value of a state-action pair: it takes the current state and the next action to take (combined into a vector ϕ) as the input, and outputs a scalar value which represents the cost of an action a taken at a state s . In this disclosure, we also consider recurrent decision making, which means that a short recent history of states and actions are incorporated in the input of the Q network 308, i.e., concatenating into a vector as the input X_Q to the Q network 308. Recurrent decision making can be helpful when the data is noisy, as will be shown below. We also propose a special design of the Q network 308 that can ease the training, in which the second last layer of the Q network 308 is designed to output the predicted PUE and temperature data, with which the cost is computed according to equation (1) in the last layer. With such design, we can easily check the predicted PUE and temperature information from the Q network 308, which can be helpful to show the quality of the Q network 308 directly. The actor network 306 (parameterized by μ) is a policy network: it takes the recent state-action history and current state (X_μ) and outputs the new action a_t to take.

In steps 708 and 710 training of the neural networks is carried out. A standard neural network training procedure with multi-training-epochs. Within each batch (in random order) of training data is used to update the weights of the neural networks using gradient descent.

In step 708, the training component 324d of the neural network module 324c updates the critic network 308. The critic network 308 is updated by minimizing the mean square error between the output of the second last layer of Q and the raw reward data y_r

5

In step 710, the training component 324d of the neural network module 324c updates the actor network 306 by minimizing the output of Q when taking action a_t current state according to the output of μ .

10 In some embodiments, a special training cost item (to be minimized) is imposed to the consecutive output of the actor network (x_j^μ and x_{j+1}^μ) to reduce control action abrupt fluctuation. This cost item is calculated by the following equation:

$$\nu_{CC} = \sum_{j \in I_i \& j+1 \in I_i} (x_j^\mu - x_{j+1}^\mu)^2.$$

15

This cost item is added to the cost function in the training process. In training the actor network, we add the cooling cost item will force the difference of the outputs of the actor network among consecutive time slots to be small. This can be used to solve the unstable problem of the actor network, to prevent abrupt action changes. It is important because in practice one cannot apply abruptly changing control action, as the action will take time to effect.

20

In step 712, the validation component 324e of the neural network module 324c selects the best parameter weight settings for the critic network 308 and the actor network 306. To avoid over-fitting, we also compute the validation error to keep track of the best weight parameter settings for the two neural networks respectively. One important note is that for the actor network 306, the validation error can be small in the beginning due to that at that time the critic network 308 is not well learned. For safety, we use a periodical re-initialization of the validation value of the actor network 306 to solve this problem.

25

30

In step 714, the output module 324b of the cooling control optimization system 300 uses the optimized actor network 306 to determine control settings for the data center cooling control system. The control settings may be stored in a database, and used to set the parameters of the cooling system for the data center.

5

Algorithm 1 shown below shows the initialization, training and validation steps of the method 700 shown in Figure 7.

Algorithm 1 CCA: Cooling Control Algorithm with Offline Trace

```

1: Data preparation: Collect an offline trace with historical data series of the state, action and reward data. Organize the
   data into the form of  $X_Q$ ,  $X_\mu$ , and  $y_r$ , where  $X_Q$  is the input of the Q-neural network  $Q$ ,  $X_\mu$  is the input of the policy
   network  $\mu$ , and  $y_r$  is the energy(PUE) and temperature readings that can be used to compute the loss data  $y$  according
   to eqn (1). Divide the data into training data (used for training the neural network) and validation data (used for validating
   the neural network to find the best parameter settings). Denote  $I_V$  as the index of sample data in the validation set.
2: Initialization: Create neural networks  $Q(X_Q|\theta^Q)$  and  $\mu(X_\mu|\theta^\mu)$ , and randomly set the values of the weight parameters  $\theta^Q$ 
   and  $\theta^\mu$ . Set initial validation error  $E_{val}^Q = 1e100$ ,  $E_{val}^\mu = 1e100$  and  $Q_{best} = Null$ ,  $\mu_{best} = Null$ .
3: for  $Epoch = 1, \dots, MaxEpoch$  do
4:   Randomly divide the training data into batches of size  $M$  and denote the index set of the data in the  $i^{th}$  batch as  $I_i$ .
5:   for  $i = 1, \dots, N/M$  do
6:     Update  $\theta^Q$  by minimizing:
7:      $\sum_{j \in I_i} (y_r(j) - Q_{-2}(X_Q(j)|\theta^Q))^2 / M$ , where  $Q_{-2}$  means the outputs of the second last layer of  $Q$ .
8:     Update  $\theta^\mu$  by minimizing:
9:      $\sum_{j \in I_i} Q([X_\mu(j), \mu(X_\mu(j)|\theta^\mu)]|\theta^Q) / M$ .
10:   end for
11:   DUE validation:  $E_{val}^Q \leftarrow \sum_{j \in I_V} \max(y(j) - Q(X_Q(j)|\theta^Q), 0)$ .
12:   if  $E_{val}^Q < E_{val}^Q$  then
13:      $E_{val}^Q \leftarrow E_{val}^Q$ 
14:      $Q_{best} = \theta^Q$ 
15:   end if
16:    $E_{val}^\mu \leftarrow \sum_{j \in I_V} Q([X_\mu(j), \mu(X_\mu(j)|\theta^\mu)]|\theta^Q)$ .
17:   if  $E_{val}^\mu < E_{val}^\mu$  then
18:      $E_{val}^\mu \leftarrow E_{val}^\mu$ 
19:      $\mu_{best} = \theta^\mu$ 
20:   end if
21: end for
22: Return: The optimal  $Q$  and  $\mu$  neural network with weight parameter settings  $Q_{best}$  and  $\mu_{best}$  respectively.

```

10

Figure 8 shows the architecture of a critic neural network and an actor neural network in an embodiment of the present invention.

The critic network 308 (denoted as Q network) takes an input vector 802 X_Q and outputs a negative reward y which is a scalar value. The critic network 308, comprises three hidden layer 804 806 and 808. Two of the hidden layers 804 and 806 have *tanh* activation. The third hidden layer 808 has a linear output. To reduce the learning difficulty, the second last layer 808 of the critic network 308 outputs the predicted energy and temperature data, concatenated as y_r , which is then used to compute y according to equation (1) by a soft-plus layer 810. In training, the critic

20

network 308 is trained by minimizing the error between the predicted y_r and the real data.

The actor network 306 (denoted as μ) takes state vectors 812 X_μ as input. The actor network 306 has two hidden layers 814 and 816. The first hidden layer 814 has linear activation and the second hidden layer 816 has a tanh activation function. The final layer 818 of the actor network 306 outputs the next control action a . The actor network 306 network is optimized to reduce the loss function computed by the critic network 308. We found that a variety of different neural network architectures can achieve similar results with necessary hyper-parameter tuning. Experimental results on comparing different network architectures are discussed below. Note that to fit for the tanh activation used in the neural networks, we normalize all data entries into the range $(-1, 1)$ and de-normalize the output of the neural networks when they are needed to compute the real energy and temperature values.

Figure 9 is a block diagram showing a technical architecture of a cooling control optimization system according to an embodiment of the present invention. The technical architecture 300 of the cooling control optimization system 310 is for performing steps of exemplary methods described above. Typically, the methods are implemented by a number of computers each having a data-processing unit. The block diagram as shown in Figure 9 illustrates a technical architecture 300 of a computer which is suitable for implementing one or more embodiments herein.

The technical architecture 300 includes a processor 322 (which may be referred to as a central processor unit or CPU) that is in communication with memory devices including secondary storage 324 (such as disk drives), read only memory (ROM) 326, random access memory (RAM) 328. The processor 322 may be implemented as one or more CPU chips. The technical architecture 300 may further comprise input/output (I/O) devices 330, and network connectivity devices 332.

The secondary storage 324 is typically comprised of one or more disk drives or tape drives and is used for non-volatile storage of data and as an over-flow data storage device if RAM 328 is not large enough to hold all working data. Secondary storage 324 may be used to store programs which are loaded into RAM 328 when such

programs are selected for execution. In this embodiment, the secondary storage 324 has an input module 324a, an output module 324b, and a neural network module 324c comprising non-transitory instructions operative by the processor 322 to perform various operations of the method of the present disclosure. As depicted in
5 Figure 9, the modules 324a-324c are distinct modules which perform respective functions implemented by the cooling control optimization system 310. It will be appreciated that the boundaries between these modules are exemplary only, and that alternative embodiments may merge modules or impose an alternative decomposition of functionality of modules. For example, the modules discussed
10 herein may be decomposed into sub-modules to be executed as multiple computer processes, and, optionally, on multiple computers. Moreover, alternative embodiments may combine multiple instances of a particular module or sub-module. It will also be appreciated that, while a software implementation of the modules 324a-324c is described herein, these may alternatively be implemented as one or
15 more hardware modules (such as field-programmable gate array(s) or application-specific integrated circuit(s)) comprising circuitry which implements equivalent functionality to that implemented in software. The ROM 326 is used to store instructions and perhaps data which are read during program execution. The secondary storage 324, the RAM 328, and/or the ROM 326 may be referred to in
20 some contexts as computer readable storage media and/or non-transitory computer readable media.

The I/O devices may include printers, video monitors, liquid crystal displays (LCDs), plasma displays, touch screen displays, keyboards, keypads, switches, dials, mice,
25 track balls, voice recognizers, card readers, paper tape readers, or other well-known input devices.

The network connectivity devices 332 may take the form of modems, modem banks, Ethernet cards, universal serial bus (USB) interface cards, serial interfaces, token
30 ring cards, fiber distributed data interface (FDDI) cards, wireless local area network (WLAN) cards, radio transceiver cards that promote radio communications using protocols such as code division multiple access (CDMA), global system for mobile communications (GSM), long-term evolution (LTE), worldwide interoperability for microwave access (WiMAX), near field communications (NFC), radio frequency

identity (RFID), and/or other air interface protocol radio transceiver cards, and other well-known network devices. These network connectivity devices 332 may enable the processor 322 to communicate with the Internet or one or more intranets. With such a network connection, it is contemplated that the processor 322 might receive information from the network or might output information to the network in the course of performing the method operations described herein. Such information, which is often represented as a sequence of instructions to be executed using processor 322, may be received from and outputted to the network, for example, in the form of a computer data signal embodied in a carrier wave.

The processor 322 executes instructions, codes, computer programs, scripts which it accesses from hard disk, floppy disk, optical disk (these various disk based systems may all be considered secondary storage 324), flash drive, ROM 326, RAM 328, or the network connectivity devices 332. While only one processor 322 is shown, multiple processors may be present. Thus, while instructions may be discussed as executed by a processor, the instructions may be executed simultaneously, serially, or otherwise executed by one or multiple processors.

It is understood that by programming and/or loading executable instructions onto the technical architecture 300, at least one of the CPU 322, the RAM 328, and the ROM 326 are changed, transforming the technical architecture 300 in part into a specific purpose machine or apparatus having the novel functionality taught by the present disclosure. It is fundamental to the electrical engineering and software engineering arts that functionality that can be implemented by loading executable software into a computer can be converted to a hardware implementation by well-known design rules.

Although the technical architecture 300 is described with reference to a computer, it should be appreciated that the technical architecture may be formed by two or more computers in communication with each other that collaborate to perform a task. For example, but not by way of limitation, an application may be partitioned in such a way as to permit concurrent and/or parallel processing of the instructions of the application. Alternatively, the data processed by the application may be partitioned in such a way as to permit concurrent and/or parallel processing of different portions

of a data set by the two or more computers. In an embodiment, virtualization software may be employed by the technical architecture 300 to provide the functionality of a number of servers that is not directly bound to the number of computers in the technical architecture 300. In an embodiment, the functionality disclosed above may be provided by executing the application and/or applications in a cloud computing environment. Cloud computing may comprise providing computing services via a network connection using dynamically scalable computing resources. A cloud computing environment may be established by an enterprise and/or may be hired on an as-needed basis from a third party provider.

Below, we present numerical evaluation results of the proposed cooling control algorithm (CCA) based on simulation. Simulation on the EnergyPlus platform is carried out to collect the training data and evaluate the proposed algorithms. Two baseline algorithms are compared with our proposed solution: one is the default control algorithm DefaultE+ from EnergyPlus, which computes the set-points according a target zone temperature with the underlying model; another is a general two stage (TS) control optimization algorithm adapted from T. Chow, G. Zhang, Z. Lin, and C. Song, "Global optimization of absorption chiller system by genetic algorithm and neural network," Energy and buildings, vol. 34, no. 1, pp. 103–109, 2002, which is trained with the same data for the proposed approach.

We use EnergyPlus to collect the training data and assess different control algorithms for the following reasons. First, it is impossible to directly test control algorithms on a real DC due to the potential risk and the long running time. Second, EnergyPlus, whose development is an initiative of the U.S. Department of Energy Building Technologies Office, is a widely recognized and reliable simulation platform to model building cooling energy consumption. Third, EnergyPlus provides the flexibility that allows simulation with user-defined algorithms, control actions, and schedules.

The simulation is configured as follows. We adopt the original DC model provided by the EnergyPlus platform to make this simulation-based study tractable, as shown in Figure 5. We choose Singapore as the location and select the corresponding weather file to revise the simulation configuration file accordingly. We use a CPU

loading trace collected from the monitoring system of Wikimedia as the workload trace for the DC model in the simulation. The whole simulation period is one year, and simulation data are collected every 6 minutes.

5 We use a random control algorithm to generate a one-year simulation trace to train the proposed algorithm. The control variables are randomly selected from valid ranges (obtained from a trace generated by the DefaultE+ algorithm) for the simulation model and then smoothed to ensure that the actions fluctuate smoothly. For the whole one-year simulation period, we select the last 45% as the test period.

10

For the proposed CCA, the hidden layer sizes of Q network are set to 50, 50 and 3, and 50, 50 for the μ network. The optimization algorithm to update the weight parameters is Adadelata. The maximum training epoch MaxEpoch is set to 200. The training batch size is set to 128. The penalty factor λ in equation (1) is set to 0.01; as
15 λ directly controls the trade-off between energy cost and cooling effects, we need to manually tune this parameter. The temperature threshold T_{thresh} in equation (1) is set to 29. Setting of T_{thresh} is dependent on the target temperature one want to achieve.

For the two-stage optimization algorithm, we adopt the approach from T. Chow et al
20 with the following changes otherwise it will be incomparable to our approach. In the first stage, we train the same evaluation network like CCA to replace the original neural network designed for modeling chiller efficiency. In the second stage, an iterative differential evolution (DE) optimization algorithm is used to find the optimal solution for each test state. Ideally, the TS algorithm can perform no worse than the
25 CCA algorithm if the optimization algorithm itself is optimized for this problem. Here as we focus on the design of CCA, for TS approach, a general optimization algorithm is used.

For the proposed CCA and the TS optimization algorithms, the optimal control
30 settings generated by these two algorithms are tested by simulation on the EnergyPlus platform. That is, for each state at the testing phase, we use the settings provided by the CCA or the TS algorithm and then record the resulting state changes and rewards for performance evaluation.

In this section, we present the simulation results of the average PUE and maximum outlet temperature (during the test period), obtained by using DefaultE+, TS, and CCA algorithms according to embodiments of the present invention. Based on these results we further compare and evaluate the underlying control algorithms.

5

Table I shows the first and second-order statistics of the PUE and maximum outlet temperature in 10 independent runs with different τ settings for TS and CCA.

TABLE I
COMPARISON OF THE PROPOSED CCA WITH THE BASELINE ALGORITHMS:
DEFAULTE+ AND TS. THE RESULTS ARE BASED ON 10 TIMES OF
INDEPENDENT TESTS.

	DefaultE+	TS $\tau = 1$	CCA $\tau = 1$	TS $\tau = 3$	CCA $\tau = 3$
PUE	1.376 $\pm$ 0.000	1.371 $\pm$ 0.000	1.333$\pm$0.003	1.346 $\pm$ 0.023	1.307 $\pm$ 0.017
$\max(T_{z_1})$	28.617 $\pm$ 0.000	26.242 $\pm$ 0.000	28.910$\pm$0.361	30.995 $\pm$ 3.121	30.111 $\pm$ 2.759
$\max(T_{z_2})$	28.655 $\pm$ 0.000	26.288 $\pm$ 0.000	29.124$\pm$0.353	30.291 $\pm$ 3.307	33.250 $\pm$ 0.005

10 Figure 10 shows a comparison of simulation results from methods of optimizing cooling control. Figure 10 shows the distribution of PUE and maximum outlet temperature T_{z1} and T_{z2} over the test period achieved by different control algorithms, respectively CCA indicates the methods according to the present invention. The results are based on 10 independent tests. The line indicated by an arrow in each
15 box is the median, and the triangle is the mean value.

From Figure 10 and table I, we can observe the following:

The proposed CCA algorithm with $\tau = 1$ achieves the best control results, by
20 reducing the PUE from 1.37 to 1.33, which is 11% cooling power saving, while maintaining the temperature of both zones under or nearby the pre-defined threshold 29. This shows that the actor network can indeed attain optimal or close-to-optimal control settings.

25 The TS algorithm with the general optimization algorithm shows unstable performance. To improve its performance, a specialized optimization procedure will be necessary. Being compared to the TS approach, the proposed CCA is an end-to-

end solution. CCA can directly output the control setting with the pre-trained policy network, which can be carefully tuned and tested in an offline manner; while for the case of TS, an online optimization algorithm has to be used, which poses higher computation cost and accuracy problem in real time.

5

The recurrent version of CCA with $\tau = 3$ has unstable results. This is reasonable as recurrent decision making can be beneficial if the data are noisy; however, as a simulated case is studied here, the data generated are free of noise. In Section VI we show how recurrent decision making can be useful in a case with real data collected from a physical DC.

10

Figures 11a to 11c show a comparison of example power usage effectiveness (PUE) and temperature traces obtained from a simulation. Figure 11a shows PUE for DefaultE+, TS and CCA. Figure 11b shows T_{z1} ; and Figure 11c shows T_{z2} .

15

From Figures 11a to 11c, it can be seen that the PUE curve of CCA is lower than that of DefaultE+ and TS while achieving higher but still satisfying temperature curves in both zones. Note that at the beginning of the test period, the temperature of the DC zone has a fast drop for TS, due to transient from the DefaultE+ algorithm to the learning algorithm (the DefaultE+ algorithm provides the settings before the test period).

20

In the following section, we study different designs of the neural network and compare their performances. We compare our network design with three other different implementations: 1) TargetNet: with the target network of DDPG, which is used to avoid fast changing of the Q network to stabilize the training; 2) ReluNet: a four-relu-layer based Q network (1024-512-256-3) to test whether relu activation can work or not; 3) LstmNet: with a LSTM layer to process the recent history trace. As we use the recent states and actions in the last τ steps as the input to the Q network, we can use an LSTM layer to process it first as LSTM is a recurrent neural network which is suitable for dealing with sequential data. The LSTM layer outputs its hidden units which are then fed into a normal Q network as described in Section IV-D. We test these different designs with τ set to 1; for LstmNet, we show the results with τ set to 3. Test results of these different designs are shown in Table II.

30

TABLE II
COMPARING DIFFERENT NEURAL NETWORK IMPLEMENTATIONS:
TARGETNET, RELUNET, LSTMNET WITH THE ORIGINAL CCA DESIGN.
ALL TEMPERATURES ARE IN °C.

	CCA	TargetNet	ReluNet	LstmNet
PUE	1.333e+00±3.371e-03	1.338e+00±1.301e-03	1.326e+00±2.128e-07	1.343e+00±2.305e-02
$max(T_{z_1})$	2.891e+01±3.614e-01	2.850e+01±1.281e-01	2.949e+01±7.958e-14	2.874e+01±2.186e+00
$max(T_{z_2})$	2.912e+01±3.525e-01	2.867e+01±1.208e-01	2.951e+01±1.709e-09	2.864e+01±2.640e+00

Comparing to the original CCA, Table II shows that these different architectures achieve very similar results. With TargetNet, the results are almost the same to the original design, which is reasonable as we use a long offline trace to train the Q network in each training episode, which can lead to a very stable learning process. With relu layers, we achieve similar results but slightly higher temperature reading with lower PUE; with LSTM layer added, we achieve a slightly lower temperature but higher PUE. Given sufficient hyperparameter tuning, these designs may also achieve satisfying performance. In this simulation case, as adding larger size relu layers and LSTM layers can slow down the training speed, we adopt the simpler design illustrated in Figure 8.

Despite network architecture studied above, the hyperparameter setting is also a critical issue in applying DRL algorithm to practical applications. In this subsection, we discuss how we set the hyper-parameters used in our algorithm, such that it may shed some light on similar applications. A key hyper-parameter is the learning rate. In our approach, we choose to use Adadelta as the training optimization algorithm, which does not oblige us to set the learning rate manually. In our case, Adadelta works great, yet it may not work on other problems. When the learning rate needs to be set manually, we recommend trying from a small value like 1e-5. Another key hyper-parameter is the initialization range of the weight of the neural networks. We use zero mean and 0.01 as the standard deviation to generate the random weights. Smaller initialization range tends to result in a more stable training process.

A critical hyper-parameter that related to the loss function in CCA is the penalty factor λ in equation (1). In our experiments, we found that it needs to be manually tuned to see a satisfying trade-off between energy and temperature.

Figure 12 shows the testing of different settings for a penalty factor in embodiments of the present invention. Figure 12 shows the effect of changing λ from 0.0 to 0.04, and how PUE and the maximum outlet temperature of each zone change accordingly. We notice that a proper setting of λ will be distinct from problem to problem. For example, in the next section, we will show the result of a different test case, in which the best λ setting is different from this simulation case.

To further investigate the proposed algorithm, we test the proposed CCA on a data trace collected from the National Super Computing Centre (NSCC) of Singapore and show its performance on optimizing the energy cost while satisfying the cooling requirements (rack intake temperature).

Figure 13 shows a 3-D model of a target system on which embodiments of the present invention were tested. There are 26 racks in the target system 1300. Three precision cooling units (PCUs) 1310 supply cold air for these racks. The PCUs 1310 supply cold air at about 20 degrees. Cold air enters the cold aisle 1320 and then goes through the racks and at last returns back to the PCUs 1310. There are other cooling facilities installed: for racks 1-20, an additional warm water-cooling system is used to cool the CPU/GPU and memory chips; for racks 21-26, an additional rear-door cooling system is used. The warm water-cooling system and the rear door cooling system will not be studied here thus we omit further details.

We try to optimize the total supply flow rate of the target PCUs shown in Figure 13, aiming to minimize their power consumption while maintaining the average intake temperature of the racks (measurement point at the height of the 36U of each rack). To apply the proposed algorithm, we collect the related data for the optimization goal, as shown in Table III.

TABLE III
THE DATA ENTRIES AND THEIR USAGE IN OPTIMIZATION OF THE AIRFLOW
RATE OF THE TARGET PCUS.

State	P1: total power consumption of racks 1-26; P2: total power consumption of the other racks; P3: total heat load recorded in warm water cooling system; F1: water flow rate; T1: water flow supply temperature; F2: air flow rate of other PCUs; T2: average supply temperature of other PCUs; F3: rear door cooling distribution units chilled water flow rate; and T3: the average supply temperature of the target PCUs.
Action	F4: the airflow supply rate of the target PCUs
Reward	P4: power consumption of the target PCUs; and T4: average rack intake temperature.

Note that several measurements of the warm water-cooling system and rear-door cooling system are also used. Our experiments show that including these reading
5 can increase the approximation accuracy of the Q-network. We collected these data entries for every 3 minutes from March 1 to 15 of 2017. For these data, we use the first 85% as the training data and the last 15% as the test data. With the above data, we utilize the proposed algorithm to train the Q and μ network. In this case, as the power consumption can be directly computed by the fan law from the airflow rate, we
10 will only rely on the Q to approximate the inlet temperature which we use as the thermal indicator.

Figure 14 shows the results for testing different settings for the target system shown in Figure 13. The vertical axis on Figure 14 shows the normalized mean absolute
15 error (MAE) of the Q network. The results show that the MAE of the Q network is less than 0.1 degrees and the results show that with $\tau=4$, the best Q learning quality can be achieved with the lowest MAE around 0.094 for the temperature prediction. This proves that the network can successfully capture the system dynamics.

20 Figure 14 also shows that when dealing with real data with noise, recurrent decision making is better as we found that when setting $\tau = 4$ we can get the best temperature estimation results. Even though MAE lower than 0.1 degree appears to be a good result, it can underestimate the temperature as the regression process will try to fit

more on most of the data falling in the middle of the distribution. This may lead to over-optimistic energy saving estimation. As we cannot directly apply the CCA algorithm to a real DC, it is important that we can generate some convincing theoretical results first without underestimation. To solve this problem, we change the validation strategy in Algorithm 1 and apply a special de-underestimation (DUE) validation method, which works in the following manner. In line 11 of CCA, when computing the validation error, we replace the original square error into a function which only considers the underestimation error, as shown below:

$$E'_{val} \leftarrow \sum_{j \in I_V} \max(y(j) - Q(X_Q(j)|\theta^Q), 0).$$

Figure 15 shows estimated temperature curves with and without a de-underestimation validation method used in embodiments of the present invention. We can observe that with DUE, although the prediction error is increased, the cases when the network underestimates the temperature are largely resolved.

Second, we study how much energy saving we can achieve when using different penalty parameter λ (used in the objective function of equation (1)).

Figure 16 shows energy saving for different values of a penalty parameter in embodiments of the present invention. Figure 16 shows testing different λ settings for the NSCC case. Energy saving decreases when λ increases, at the same time the maximum temperature decreases. From Figure 16 we can see that with DUE, we can achieve energy saving 15% easily if we set the temperature threshold larger than 26.6 degrees. If we want a lower maximum temperature such as 26.4, we can still save about 10%. When we set the target maximum temperature lower than 26.2, we will have to use more cooling power than the actual setting adopted.

Figures 17a and 17b show the effects of changing the value of a penalty parameter on airflow rate and temperature in embodiments of the present invention. Figure 16a shows control setting (airflow rate) predicted by the proposed algorithm on the NSCC data trace when λ changed from 18 to 26: and Figure 17b shows the correspondingly predicted rack temperature. Darker color curves are with smaller λ settings, and the black curve (with "+" marker) is the real data trace collected in the test period.

DCs power the modern society as the infrastructure for information storage, processing, and dissemination. At the same time, DCs consume a formidable amount of electricity, among which a significant portion is used in cooling. To develop an optimal control policy for the complex cooling system for a DC is a complex task. We propose and verify an end-to-end DRL approach CCA for the control optimization of the cooling system of a DC. Compared to the existed TS optimization method, the proposed CCA can directly optimize a policy network based on the observed historical data, while the policy can output the optimized control settings for any given state. Adapted from DDPG and the actor-critic framework, our algorithm is a batch off-policy algorithm, with which we tested various algorithm settings to examine performance thoroughly.

We evaluate the proposed algorithm on the simulation platform EnergyPlus and a trace collected from a real DC. The simulation results show that our method can maintain the DC temperature within the predefined threshold while achieving lower PUE and save about 11% cooling energy compared to a baseline approach with manually designed control settings. Our results on the real trace show that we can achieve high evaluation accuracy and the predicted control setting can reduce the cooling cost around 15% while maintaining the rack intake temperature under a predefined threshold. The results prove that our algorithm can successfully learn the system dynamics from the monitoring data and can contribute to improving the cooling efficiency.

Whilst the foregoing description has described exemplary embodiments, it will be understood by those skilled in the art that many variations of the embodiments can be made within the scope and spirit of the present invention.

CLAIMS

1. A reinforcement learning method for optimizing control parameters for a data center cooling system, the method comprising:

initializing an actor neural network configured to generate an indication of a subsequent control action in response to an actor neural network input vector comprising an indication of a current state of the data center cooling system;

initializing a critic neural network configured to generate a cost indication in response to a critic neural network input vector comprising an indication of the current state of the data center cooling system and a subsequent control action;

receiving trace data indicating control actions, previous states of the data center cooling system, and corresponding temperature data and energy usage data for the data center cooling system;

training the critic neural network by determining a set of parameter weights for the critic neural network which minimizes error between an output of the critic neural network and reward data corresponding to the temperature data and energy usage data for the data center cooling system;

training the actor neural network by determining a set of parameter weights for the actor neural network which minimize the cost indication generated by the critic neural network; and

using the trained actor neural network to generate a set of control parameters for the data center cooling system.

2. A method according to claim 1, wherein the critic neural network comprises a plurality of layers and a second last layer of the plurality of layers is configured to output predicted temperature data and predicted energy usage data, and a last layer of the plurality of layers is configured to evaluate a cost function of the predicted temperature data and predicted energy usage data to generate the cost indication, wherein determining a set of parameter weights for the critic neural network comprises determining a set of parameter weights that minimize the error between the predicted temperature data and predicted energy usage data and the corresponding temperature data and energy usage data for the data center cooling system.

3. A method according to claim 2, wherein the cost function of the predicted temperature data and the predicted energy usage data is selected to minimize the predicted energy usage subject to satisfying a temperature constraint.
4. A method according to claim 3, wherein the cost function comprises a penalty factor which controls a trade-off between predicted energy usage and the temperature constraint.
5. A method according to any preceding claim, wherein the actor neural network input vector comprises indications of control settings for past time slots and wherein training the actor neural network further comprises calculating a cooling cost function which is dependent on the difference of the outputs of the actor neural network for consecutive time slots.
6. A method according to any preceding claim, further comprising dividing the trace data into training data and validation data, and using the validation data to calculate a validation error for sets of parameter weights for the critic neural network and the actor neural network, and selecting a set of parameter weights for the critic neural network and the actor neural network using the validation error.
7. A method according to claim 6, wherein using the validation data to calculate a validation error comprises calculating an underestimation error and wherein selecting a set of parameter weights for the critic neural network and the actor neural network comprises using the underestimation error.
8. A method according to any preceding claim wherein the control actions comprise a plurality of temperature set points for the data center cooling system.
9. A method according to any preceding claim wherein training the critic neural network and / or training the actor neural network comprises using gradient descent.
10. A computer readable medium carrying processor executable instructions which when executed on a processor cause the processor to carry out a method according to any one of claims 1 to 9.

11. A cooling control optimization system for a data center comprising a processor and a data storage device, the data storage device storing computer executable instructions operable by the processor to:

initialize an actor neural network configured to generate an indication of a subsequent control action in response to an actor neural network input vector comprising an indication of a current state of a data center cooling system of the data center;

initialize a critic neural network configured to generate a cost indication in response to a critic neural network input vector comprising an indication of the current state of the data center cooling system and a subsequent control action;

receive trace data indicating control actions, previous states of the data center cooling system, and corresponding temperature data and energy usage data for the data center cooling system;

train the critic neural network by determining a set of parameter weights for the critic neural network which minimizes error between an output of the critic neural network and reward data corresponding to the temperature data and energy usage data for the data center cooling system;

train the actor neural network by determining a set of parameter weights for the actor neural network which minimize the cost indication generated by the critic neural network; and

use the trained actor neural network to generate a set of control parameters for the data center cooling system.

12. A cooling control optimization system according to claim 11, wherein the critic neural network comprises a plurality of layers and a second last layer of the plurality of layers is configured to output predicted temperature data and predicted energy usage data, and a last layer of the plurality of layers is configured to evaluate a cost function of the predicted temperature data and predicted energy usage data to generate the cost indication, wherein determining a set of parameter weights for the critic neural network comprises determining a set of parameter weights that minimize the error between the predicted temperature data and predicted energy usage data and the corresponding temperature data and energy usage data for the data center cooling system.

13. A cooling control optimization system according to claim 12, wherein the cost function of the predicted temperature data and the predicted energy usage data is selected to minimize the predicted energy usage subject to satisfying a temperature constraint.

14. A cooling control optimization system according to claim 13, wherein the cost function comprises a penalty factor which controls a trade-off between predicted energy usage and the temperature constraint.

15. A cooling control optimization system according to any one of claims 11 to 14, wherein the actor neural network input vector comprises indications of control settings for past time slots and wherein data storage device stores computer executable instructions operable by the processor to: train the actor neural network by calculating a cooling cost function which is dependent on the difference of the outputs of the actor neural network for consecutive time slots.

16. A cooling control optimization system according to any one of claims 11 to 15, wherein the data storage device stores computer executable instructions operable by the processor to: divide the trace data into training data and validation data, and using the validation data to calculate a validation error for sets of parameter weights for the critic neural network and the actor neural network, and select a set of parameter weights for the critic neural network and the actor neural network using the validation error.

17. A cooling control optimization system according to claim 16, wherein using the validation data to calculate a validation error comprises calculating an underestimation error and wherein selecting a set of parameter weights for the critic neural network and the actor neural network comprises using the underestimation error.

18. A cooling control optimization system according to any one of claims 11 to 17, wherein the control actions comprise a plurality of temperature set points for the data center cooling system.

19. A cooling control optimization system according to any one of claims 11 to 18, wherein the data storage device stores computer executable instructions operable by the processor to: train the critic neural network and / or training the actor neural network using gradient decent.