Distributed Reinforcement Learning for Building Energy Optimization

Arshad Javeed

LUND
UNIVERSITY

Department of Automatic Control

MSc Thesis
TFRT-6242
ISSN 0280-5316

Department of Automatic Control
Lund University
Box 118
SE-221 00 LUND
Sweden

Printed in Sweden by Tryckeriet i E-huset
Lund 2024

Abstract

Heating, ventilation, and air-conditioning (HVAC) systems are ubiquitous and are one of the major components responsible for energy consumption in a typical building system. In light of the imminent energy crisis, there is an increasing demand to revisit the building systems and save as much energy as possible. In this regard, the scope of the thesis is to explore opportunities for data-driven optimization of HVAC systems. Traditionally, optimization in HVAC systems has relied on offline optimization requiring domain expertise to schedule a set of optimal controls, but such approaches often require extensive domain expertise. Another drawback is that as the system changes over time, these controls go out of tune and need to be adapted. Thus, data-driven optimization approaches (such as reinforcement learning) appear more appealing due to their ability to adapt online and model and solve complex problems.

A primary objective of the thesis is to carry out exploratory data analysis experiments to quantify the savings potential and expose the optimization space for RL. The main objective is to explore distributed reinforcement learning via multi-agent RL strategies (MARL) and compare and contrast the pros and cons of MARL with single-agent RL. The work benchmarks two of the popular contemporary MARL strategies, centralized training and decentralized execution, and value-mixing approaches, along with proposing two novel MARL enhancements in HVAC systems: a linear value-mixing strategy (inspired by Q-function mixing, QMIX) and turn-based games, that attempt to alleviate some of the problems of multi-agent credit assignment and non-stationarity surrounding MARL.

The experimental results include the learning performance of various RL strategies and the performance benchmarks against the closed-loop controller under realistic conditions. The experimental results reveal that the RL strategies perform significantly better than the closed-loop controller (with a few exceptions), achieving power savings of up to 15% on yearly simulations with live weather profiles. The results also highlight the tradeoffs between optimality and sampling efficiency, further corroborating the prejudice about MARL, where the single-agent RL performs better in terms of optimality, while the MARL approach displays faster learning.

Acknowledgements

Firstly, I would like to thank my supervisors, Bo Bernhardsson and Johan Åkesson for all their valuable comments and guidance throughout, and for helping us formulate pragmatic experiments to deliver meaningful and impacting results.

I would also like to thank Clas Jacobson for sharing his expertise on optimal control and the theoretical aspects surrounding reinforcement learning.

I would also like to thank Magdalena Atlevi for the initial setup and for helping us get acquainted with all the tools and software required for running our experiments and resolving our problems.

Thanks to Viktor Linders for answering general queries and sharing your wisdom on math and numerical optimization, and the great Fika conversations.

Finally thanks to my fellow colleagues, Antoni Carrillo Sala and Ewoud Donck for getting through this together and being there for collaboration and exchanging ideas.

Contents

1

Introduction

Building systems have been identified to play a vital role in addressing the current energy crisis and mitigating environmental impact. It is estimated that building systems are responsible for 40% of global energy consumption and account for nearly 30 % of carbon emissions [Shaikh et al., 2014; Energy Efficiency & Renewable Energy, n.d.; Abergel T., 2018]. Thus, there is a growing need to optimize the operational efficiency of these systems. This optimization would directly translate into a reduced carbon footprint and impact on overall sustainability. For business organizations, improved operational efficiency would directly translate to energy savings, and a data-driven optimization approach would lead to reduced operational and maintenance costs.

Typically, energy systems comprise low-level rule-based and PID controllers (supervisory control loops) whose objective are to meet the cooling load requirements by tracking a set of setpoints. The underlying problem here is the lack of intelligence or "adaptability", which leads to problems at different levels. Firstly, the task of setting the controller setpoints is typically crafted using domain-specific knowledge, and modeling all the dynamic factors affecting the performance of the system can be challenging. Then, there is the aspect of tuning the supervisory control elements. As in the case of any real-world system, the system dynamics are bound to vary over time due to wear and tear or other external factors, which means the controller ought to be re-tuned to adapt to the changing dynamics. Over time, these problems can have a deteriorating impact on overall performance and energy consumption. Today, adjusting the setpoints and the controllers follows a reactive approach, i.e. the performance is evaluated periodically, and domain-specific and engineering knowledge is required to tweak the parameters. Instead, the thesis discusses a proactive approach, where an autonomous entity learns to anticipate operating conditions and outputs an optimal set of parameters. Thus, data-driven optimization can be employed to implicitly model the behavior of the system to intelligently choose the setpoints for the building.

The project will focus on a chiller plant responsible for meeting the cooling load requirements of a data center. The objective here is to optimize the operational efficiency of the HVAC system by leveraging the available degrees of freedom. The

execution involves choosing the optimal setpoints for supervisory control loops to impact the overall power consumption. The primary job of these supervisory control loops is to efficiently track the setpoints. Thus, improving operational efficiency boils down to the efficient staging and setpoint scheduling of the different equipment involved.

1.1 Literature Review

HVAC systems is comprised of regulatory control elements - rule-based controllers and PID controllers. The conventional approach involves analyzing the cooling load requirements of the building, under varying external conditions and other influencing factors to craft a setpoint schedule for the supervisory control elements. There are two challenges with this approach. Firstly, it is nearly impossible to account for all the factors impacting power consumption, such as occupancy, heat dissipation, equipment performance discrepancies, etc [Nagy et al., 2023]. Secondly, the schedule fails to adapt to the specific system and may require engineering efforts to retune the system regularly.

Considering the some of these challenges, optimization approachs is have been explored for the task. As real-world control systems are often tagged with hard constraints, a model predictive control (MPC) strategy has been a popular choice for diverse optimization problems in HVAC systems. [Ma et al., 2012; Maasoumy et al., 2011] have demonstrated the use of MPC in cooling systems, with a primary focus of designing simplified but decently accurate equipment models to be used by MPC. [Wei et al., 2014] present MPC as a co-scheduling strategy, where the MPC controller assists the primary control strategy. To account for model uncertainties, [Oldewurtel et al., 2010] present a stochastic MPC. Similarly, there are other model-based demonstrations for HVAC control [Ghahramani et al., 2014; Meimand and Jazizadeh, 2022]. However, as noted before, a decently accurate model is imperative in the case of MPC, which in turn leads to modeling several unknown factors affecting the dynamics. Further, the current MPC-based approaches fail to scale to higher order models and cost functions [Liang et al., 2015; Killian and Kozek, 2018; O'Dwyer et al., 2017].

In this regard, the model-free optimization techniques are more appealing, as they require little to no supervision, and reinforcement learning in particular are prominent due to its completely unsupervised nature. Model-free approaches would inherently learn a model of the system given the observations, this property enables model-free RL to solve complex problems. Reinforcement learning has been widely used for HVAC optimization [Sierla et al., 2022], two approaches have been predominantly pursued: i) the reinforcement learning agent having complete responsibility of granular control signals, ii) the agent controlling the setpoints for the supervisory control. The former approach could be challenging to implement in the real/world and putting a black model directly in charge makes the approach

less interpretable. Currently, these methods have been successfully applied to binary actuator signals (predominantly). The latter approach appears more feasible and has been studied extensively. Currently, there are a plethora of methodologies that apply reinforcement learning (as model-free methods) to HVAC systems. The reason behind the popularity of model-free approaches is the complex nature of building systems. Although model-based approaches such as model predictive control (MPC), have desirable properties, it is nearly impossible to accurately model the whole system. Heat transfer interactions between building components, the impact of occupancy rate, and the layout of the building plan all affect the model and, in turn, impact the efficiency of model-based approaches. [Zhang et al., 2019] demonstrate that reinforcement learning in the case of HVAC systems does not require an overly accurate dynamics model (unlike MPC), and shows the end-to-end RL design process. [Wei et al., 2017] is one of the first approaches employing deep reinforcement learning for HVAC control. [Gao et al., 2019] employ a hybrid approach, combining MPC and RL to trade energy and comfort in an HVAC system. [Luo et al., 2022] focus on controlling commercial buildings using deep RL, specifically presenting the real-world challenges associated with RL approaches when deploying the algorithm in a real-world scenario.

Despite the optimism of deep reinforcement learning in complex problems such as HVAC control, one of the key challenges with reinforcement learning-based approaches is scalability, specifically concerning the action space of the algorithm (even in the case of deep RL approaches). The explosion in action space implies a larger exploration landscape for the agents, or in other words, a longer time to converge. [Gao et al., 2019] argue that employing continuous action space rather than discretizing or grinding the action space reduces the cardinality of the action space. [Wei et al., 2017] address the curse of dimensionality by training separate critic networks for separate agents, hinting towards multi-agent RL. However, this can lead to suboptimal solutions if the agents are strongly coupled. Similarly, there are other demonstrations of deep RL for HVAC control [Gao et al., 2020; Wei et al., 2017; Zhang et al., 2019; Wang et al., 2024a].

The problem of scalability in RL can be tackled in two ways: i) a distributed approach (analogous to distributed machine learning approaches) exploring data parallelization, where multiple (identical) agents explore and learn the environment in parallel and the learnings are then aggregated, ii) Multi-agent RL (MARL), where the problem is broken down into multiple subproblems, and the agents work towards a common goal. [Czech, 2021] contrast the two approaches. The original problem of exploration due to action space exploration still persists in the data parallelism approach, as each of the agents will still have to explore the entire action space. The latter approach can lead to significant benefits if there exists decoupling between the agents, which can be exploited to reformulate the problem (action spaces). [Busoniu et al., 2008; Stefano V. Albrecht, 2024] comprehensively review the MARL strategies and contrast it to single-agent RL.

[Hanumaiah and Genc, 2021] have applied reinforcement learning to tune indi-

vidual setpoints in a multi-zone building and, furthermore, have demonstrated that running season-specific agents in a cooperative multi-agent reinforcement learning yields better results. [Charbonnier et al., 2022] show despite the success of MARL in HVAC systems, the performance of the multi-agent systems degrades as the number of agents in the mix is increased. The authors alleviate the problem by performing centralized training on the available historic HVAC data and then proceed with decentralized execution. "Non-stationarity", "adaptability", and "credit assignment" are some of the contemporary challenges of MARL in general and remain an active area for research.

1.2 Scope of Thesis

The primary objective of the thesis will be to try to identify if there is any saving potential to be exploited and then explore the feasibility of employing reinforcement learning for optimization in HVAC systems. This is achieved by formulating a set of exploratory data analysis experiments to explore the optimization terrain and manifest the saving potential. Although the main objective of the thesis is "multi-agent reinforcement learning" (MARL), tackling MARL in a complex problem such as an HVAC optimization can be daunting. Thus, the problem is first addressed in the context of single-agent RL focusing on efficient RL problem formulation. MARL is then gradually introduced by turning the original problem into a cooperative setting, with multiple agents working towards a common goal.

As a first step, the existing MARL algorithms are tested and benchmarked against the single-agent RL, contrasting the pros and cons. Specifically, the centralized training and decentralized execution (CTDE) and value-decomposition (summixing and QMIX) approaches are evaluated. Motivated by some of the challenges in MARL and the limitations of these approaches in the context of the given problem, two novel approaches are proposed to alleviate the problems of non-stationarity and multi-agent credit assignment: the proposed "LINMIX" strategy exploits the reward structure of the problem assuming the fact that the overall power consumption is conservative and additive, giving rise to a linear mixing strategy as opposed to a non-linear mixer (QMIX). To alleviate the problem of non-stationarity, a turn-based game is proposed, where the agents play in turns to facilitate other agents to better model and distinguish the consequences of their own actions and the actions of other agents.

Finally, as part of evaluation and benchmarking, all of the RL implementations are benchmarked against the closed-loop controller under multiple testbeds (emulating real-world conditions that may be potentially encountered) to quantify the savings.

1.3 Problem Formulation

The section presents a mathematical formulation of the problem. A typical RL optimization problem involves identifying the relevant inputs, outputs, and control signals, categorizing them into states and actions (decision variables), and specifying the reward function. It also includes identifying the constraints, and the need to model additional uncertainties associated with the system variables. The proposed framework closely follows the optimal control framework for sequential decision processes. [Powell, 2022a; Powell, 2022b] lays out the merits of such a framework, and breaks down the problem into 6 fundamental steps. The idea here is to pursue a "model-first" approach, i.e. model all the state and decision variables before diving into the reinforcement learning/optimization aspects of the problem. The subsequent sections present the problem formulation with a level of abstraction to focus on the high-level objectives of the problem.

The HVAC system comprises a chiller plant responsible for maintaining the load requirements of the IT room. The model of the system can be executed using the open-loop controller for the purpose of optimization. The model takes in a set of measurements, temperature, relative humidity, reference cooling load, and control inputs, and executes supervisory control loops to maintain the setpoints. Thus at a high level, we are interested in influencing the supervisory control elements via the setpoints and staging individual equpiments (such as chillers, cooling towers and pumps) to impact the operation (Figure 1.1).

Narrative

The problem involves fulfilling the load requirements of a data center, where a chiller plant is responsible for pumping the coolant to the room. The data center load requirements can be assumed to have daily or weekly fluctuations as per the traffic. The external factors, weather (temperature and humidity) will have an impact on maintaining the load requirement, but should also be highly correlated within a short time span. The agent (or the responsible intelligence) can measure the external temperature and humidity and the current state of the IT room, i.e. the current status of the equipments and the outputs. The control variables include equipment staging (ability to turn ON/OFF individual equipments - chillers, cooling towers and pumps) and the setpoints for the chiller plant. The ultimate goal is to minimize the total energy consumption while maintaining the load requirements of the data center.

States

The system states should encapsulate all the information that might be deemed relevant to the RL optimization problem. At a high level, we may have several dynamic states of the system and exogenous information. The dynamic states of the system may comprise of all the internal states of the system, such as the number of individual equipments running and their status, while the exogenous information is the

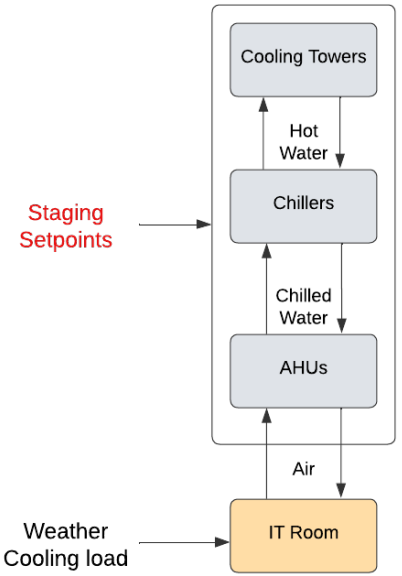


Figure 1.1 Abstract model of the chiller plant

external variables impacting the system (such as weather). A naive approach would be to include all the state/measurement variables available and have the algorithm figure out relevant information on its own, but that could significantly slow down the learning process.

In the given problem, turning ON/OFF a chiller or a cooling tower depends on the current status of the equipment staging (marked by †) and we can expect the decisions to also be influenced by the current operating performance of the equipments, such as the chiller PLR (part load ratio) or CR (cycling ratio) of the chillers (marked by ‡). For instance, it might be more efficient to turn ON both chillers, if, for example, the chiller is not operating at its expected efficiency. As the problem involves reference tracking, the reference cooling load is also included as the state variable as it is an obvious requirement. The state variables (s_t) are marked "STATE" in Table 1.1. It is also assumed that all the states listed in Table 1.1 are known and measurable with sufficient accuracy.

Decision Variables

The decision variables (or actions) are essentially the control inputs to the system. The decision variables a_t are marked "DECISION" in Table 1.1. It is worth noting that the actions comprise both discrete and continuous actions. The discrete actions correspond to the staging control, while the continuous actions represent the setpoints for the supervisory control loops. The initial objective is to have a single

policy tuning all the control variables. Then the decision variables could be decoupled between multiple policies, e.g. staging and setpoint controls. In the subsequent sections, a data-driven approach is pursued to provide insights into choosing the relevant set of variables for optimization.

Variable	Type	Description	Role	Constraints
T_{db}	real	outdoor dry bulb temperature	EXOGENOUS	$[12, 35]$ C
H	real	outdoor relativity humidity	EXOGENOUS	$[0, 1.0]$
T_{air}	real	Supply air temp. setpoint in IT room	CONSTANT	287.15
T_{ref}	real	Return air temp. setpoint in IT room	CONSTANT	299.15
$\ddagger\eta_{\text{PLR, ch1}}$	real	part load ratio of compressor 1	STATE	
$\ddagger\eta_{\text{COP, ch1}}$	real	operating efficiency of chiller 1	STATE	$\sim [0, 10.0]$
$\ddagger\eta_{\text{PLR, ch2}}$	real	part load ratio of compressor 2	STATE	
$\ddagger\eta_{\text{COP, ch2}}$	real	operating efficiency of chiller 2	STATE	$\sim [0, 10.0]$
$\dagger n_{\text{chs}}$	discrete	number of chillers running	STATE	$\{0, 1, 2\}$
$\dagger n_{\text{cts}}$	discrete	number of cooling towers running	STATE	$\{0, 1, 2\}$
$\dagger n_{\text{pps}}$	discrete	number of pumps running	STATE	$\{0, 1, 2, 3\}$
l_{ref}	real	current load to track	EXOGENEOUS	$[0, 2 \times 10^6]$
n_{chs}	discrete	number of chillers to run	DECISION	$\{1, 2\}$
n_{cts}	discrete	number of cooling towers to run	DECISION	$\{0, 1, 2\}$
n_{pps}	discrete	number of pumps to run	DECISION	$\{0, 1, 2, 3\}$
T_{chwst}	real	leaving water temperature for chiller 1	DECISION	$[279.53, 288.09]$
T_{chwsdt}	real	leaving water temperature between differential	DECISION	$[0, 5.0]$
dp	real	chilled water pump Δp for chiller plant	DECISION	$[118402, 300000]$
T_{ewt}	real	cooling water temperature at condenser inlet	DECISION	$[299.81, 302.59]$
T_{room}	real	IT room temperature	REWARD	
P_{total}	real	total power consumption	REWARD	
P_{ch}	real	chiller power consumption	REWARD	
P_{chw}	real	pump power consumption	REWARD	
P_{pp}	real	pump power consumption	REWARD	
P_{ct}	real	cooling tower power consumption	REWARD	
P_{fan}	real	fan power consumption	REWARD	

Table 1.1 System Variables

* potential low-level variables. †augmented state variable. ‡performance variable

Exogenous Information

The exogenous information includes uncertainty associated with measurements and disturbances. A primary source of uncertainty could be surrounding the temperature measurements (T_{db}) and humidity (H). Anticipating the trends in these measurements can be a valuable insight to the model, $\hat{T}_{\text{db},t} = f(T_{\text{db},t-1}, T_{\text{db},t-2}, ...)$ (similarly for RH), where f can be a forecasting model, enabling the agent to anticipate future trends and plan ahead. However, it may be worth noting that explicit modeling efforts might be redundant in the case of reinforcement learning optimization leveraging neural network policy functions, as the network should eventually learn a useful non-linear combination of variables and learn to anticipate trends.

Transition Function

The transition function $P(s_t, a_t, s_{t+1}) \in [0,1]$ defines the transition of the system from the current state s_t as a consequence of the action a_t, observing the new state and exogenous information s_{t+1}, w_{t_1}. In our case, the transition function would be simulated by the building function mock-up units (FMU) models that are fit on prior building data recorded, thus would serve as a vital tool for defining a simulation environment. The simulation model was made available by Carrier, the model is assumed to reasonably accurate within the operating conditions listed in Table 1.1.

Objective Function

The primary objective is to maintain the cooling load of the IT room, and the secondary objective is to minimize operating power consumption during the process. Thus a preliminary reward candidate can be as per Equation 1.1,

$$R(s_t, a_t) = -|T_{\text{ref},t} - T_{\text{room},t}| - \alpha \hat{P}_{total,t} \tag{1.1}$$

Where $\hat{P}_t$ would be the estimated power consumption over the simulation interval, the hyperparameter α can act as a weighing factor to assign relative weights to the different objectives involved. As both quantities are strictly positive, maximizing the negative of the quantities would be equivalent to maximizing the reward in the context of reinforcement learning convention.

Learning Policies

Having defined the reward function, the objective is to learn an optimal policy that would minimize the cumulative rewards (Equation 1.2),

$$\arg\max_{\pi} \mathbf{E}_{S_0} \left[\Sigma_{t=0}^{\tau} \gamma^t R(s_t, \pi(s_t))) | S_0 \right] \tag{1.2}$$

Where the expectation is with respect to the initial state (temperature, humidity, and reference cooling load). The optimal policy can be found using several approaches, and reinforcement learning or deep reinforcement learning (and multi-agent RL) is one of the ways.

2

Background

2.1 Dynamic Programming

The problem at hand of choosing the optimal staging sequence and setpoints for the chiller system can be viewed as a sequential decision-making problem, as immediate actions taken can have long-term effects. Sequential decision-making problems are often modeled as Markov decision processes (MDPs) that build on the following assumptions:

1. A system is defined with a set of states $\mathscr{S} \in \mathbb{R}^{n_s}$, where n_s is the cardinality of the state-space.

2. Given a state s_t, the system can execute an action (a_t) $\mathscr{A} \in \mathbb{R}^{n_a}$, n_a is the cardinality of the action space.

3. Having executed the action, the system then transits to a new state s_{t+1} as per the transition function $T : \mathbb{R}^{n_s} \times \mathbb{R}^{n_a} \times \mathbb{R}^{n_s} \to [0,1], T = P(s_{t+1}|s_t, a_t)$. One of the crucial assumptions in an MDP is that the transition to the next state is completely determined by the current state and the current action executed, i.e. s_t, a_t. During the process, the system receives also a reward $R(s_t, a_t, s_{t+1})$.

Having formulated the problem in terms of the MDP framework. Dynamic programming (DP) is a classical approach to solving sequential decision-making problems. Building upon the MDP framework, dynamic programming assumes that the system receives a reward while transiting to the next state, $R : \mathbb{R}^{n_s} \times \mathbb{R}^{n_s} \to \mathbb{R}, R(s_t, a_t)$. The objective then is to simply minimize the cumulative rewards (Equation 2.1), which would then yield an optimal control sequence, $a_0, a_1, a_2, \ldots a_T$. A discount factor $\gamma < 1$ is often applied for discounted MDP problems.

$$\max_{a_0, a_1, a_2, \ldots a_T} \Sigma_{t=0}^{T} \gamma^t \mathbb{E}_s[R(s_t, a_t, s_{t+1})] \tag{2.1}$$

A naive way to solve Equation 2.1 would be to employ a brute force approach, which would test all the combinations of actions over the horizon T and then pick the optimal sequence that would yield the minimal cost. However, this would be inefficient in practice. Dynamic programming solves the problem (2.1) using Bellman's principle of optimality (Equation 2.2), where V represents a value-function given a state s_t and V^* implies the optimal value, given the current state, i.e. the highest possible cumulative reward if acted optimally starting from the current state s_t. Conceptually, Bellman's principle of optimality states that optimality from point A-C implies optimality from point A-B and then optimality from B-C (assuming B to be an intermediate point between A and C).

$$
\begin{aligned}
V^* &= \max_{a_t} R(s_t, a_t, s_{t+1}) + \max_{a_{t+1}, \ldots a_T} \Sigma_{\tau=t+1}^{T} \gamma^t R(s_\tau, a_\tau) \\
V^* &= \max_{a_t} R(s_t, a_t, s_{t+1}) + \gamma V^*(s_{t+1})
\end{aligned} \tag{2.2}
$$

In case of a stochastic problem, Equation 2.2 can be rewritten by introducing the system transition function $P(s_{t+1}|s_t, a_t)$ as follows,

$$
V^* = \max_{a_t} \Sigma_{s_{t+1}} P(s_{t+1}|s_t, a_t) \left[R(s_t, a_t, s_{t+1}) + \gamma V^* \right] \tag{2.3}
$$

In other words, we would then be maximizing the expected reward, with the expectation over all possible states s_{t+1} given s_t.

Dynamic programming exploits the recursive relationship established by the Bellman equation (Equation 2.3). Two prominent ways of solving a DP problem are forward and backward DP. Forward DP starts with the initial state s_0 and executes planning until the final state s_T is reached, at each step the algorithm would choose an optimal action as per the recursive relationship and update the value functions at each state s_t and repeat the process until the value estimates convergence, i.e. when the value function does not change between two successive iterations. Backward DP is a similar approach but the problem is solved backward in time, this may be useful depending on the context of the problem where there is a natural reverse ordering of states, e.g. a shortest path planning problem.

Solving a DP problem involves two fundamental steps, policy evaluation, and policy improvement steps, formally known as the policy improvement algorithm (formally known as policy iteration algorithm, Algorithm 1, [Sutton and Barto, 2018]). The policy evaluation step involves determining the value function (V_{k+1}) under the current policy π using the current value function (V_k). The process is repeated until approximate convergence, i.e. $|V_k - V_{k+1}| \to 0$. The policy improvement step involves improving the policy by looking up the optimal action using cost function approximation. The two steps are alternated until the policy improvement yields no improvement, i.e. until the policy converges.

Algorithm 1 Policy Iteration

Initialization
$\Delta > 0$
$V : \mathbb{R}^{n_s} \times \mathbb{R}^{n_a} \rightarrow \mathbb{R}$
$\pi : \mathbb{R}^{n_s} \rightarrow \mathbb{R}^{n_a}$

Policy Evaluation
while $\delta > \Delta$ **do**
  for $s \in \mathscr{S}$ **do**
    $v \leftarrow V_k(s)$
    $V_{k+1}(s) \leftarrow \Sigma_{s_{t+1}} P(s_{t+1}|s_t, a_t)\,[\, R(s_t, a_t, s_{t+1}) + \gamma V_k(s_{t+1})]$
    $\delta = \max(\delta, |v - V_{k+1}(s)|)$
  end for
end while

Policy Improvement
for $s \in \mathscr{S}$ **do**
  $\pi_{k+1}(s) \leftarrow \arg\max_{a_t} \Sigma_{s_{t+1}} P(s_{t+1}|s_t, a_t)\,[\, R(s_t, a_t, s_{t+1}) + \gamma V_k(s_{t+1})]$
end for

Having solved the DP problem, the optimal policy given any state would be, $a_t : \arg\max_{a_t} \Sigma_{s_{t+1}} P(s_{t+1}|s_t, a_t)\,[\, R(s_t, a_t, s_{t+1}) + \gamma V^*]$.

2.2 Motivation for Reinforcement Learning

Although the dynamic programming approach is theoretically sound and eventually bound to converge, there are a few challenges that make it infeasible to employ forward or backward DP in a real-world task (like that of the problem of building energy optimization problem). Taking a closer look at Equation 2.3 reveals the following subtleties:

1. The transition function $P(s_{t+1}|s_t, a_t)$ is often unknown or hard to model, which makes the expectation over the states $\mathbb{E}_s$ intractable.

2. The value function of a state s_t involves another expectation over the action space and state space, $V(s_t) = \Sigma_{a_t} P(a_t|s_t) \Sigma_{s_{t+1}} P(s_{t+1}|s_t, a_t)[\, R(s_t, a_t, s_{t+1}) + V^*(s_{t+1})]$, which makes approximating the value function of the next state s_{t+1} challenging. The problem can be further exacerbated if the state/action spaces are continuous.

3. The forward and backward DP involves iterating the value function until convergence. This may not be feasible if the state-space is continuous or comprises of too many states.

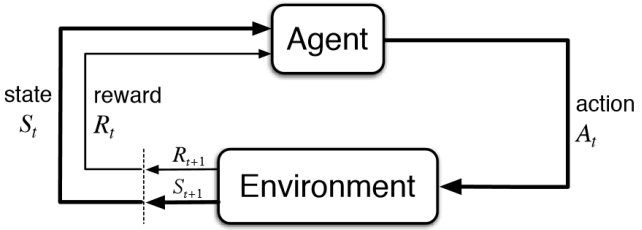


Figure 2.1 Reinforcement learning

In this regard, reinforcement learning can be seen as a way to solve the Bellman equation while addressing these challenges. A reinforcement learning agent learns to approximate the Bellman equation and/or learns an optimal policy by trial and error, underpinned by the concept of feedback/reward.

2.3 Reinforcement Learning

The basic idea behind reinforcement learning is to solve the Bellman equation in an unsupervised way. Looking at Equation 2.3, the only supervision is manifested in two different aspects. Firstly, the term $P(s_{t+1}|s_t, a_t)$ represents the state transition probabilities, without which the estimation of the expected value of the current state would be inaccurate. Secondly, the value function of the next state $V^*(s_{t+1})$ is part of Bellman's principle of optimality, without the optimal value estimate of the next state, it is not feasible to derive the policy at the current state. Thus, reinforcement learning tackles these issues by learning the value function and/or policy function solely by interaction with the environment (or the process). Figure 2.1 demonstrates the idea behind reinforcement learning. In each step, the agent (policy) executes an action and observes the new state (s_{t+1}) and the reward (r_{t+1}). The process can be repeated to collect a set of trajectories, $\mathscr{T} = \{s_0, a_0, r_1, s_1, \ldots s_T, a_T, r_{T+1}, s_{T+1}\}$ which are then used to optimize the policy behavior to maximize the expected rewards. The problem of unknown dynamics is circumvented by replacing the expectation $\mathbb{E}_s$ with empirical averages, and a simple way to approximate the cost of the next state is to use empirical Monte-Carlo (MC) estimates based on the observed trajectories. Temporal-difference learning is an alternative approach to MC reward estimation (discussed subsequently).

The two fundamental elements of the PI algorithm (Algorithm 1), policy evaluation and policy improvement serve as a backbone for all RL algorithms [Kaelbling et al., 1996; Shakya et al., 2023], and different strategies are applied to circumvent the challenges of classical DP. The following properties of reinforcement are appealing and address the challenges introduced in Section 2.2:

1. Model-free reinforcement learning approaches do not require a description of

the model. Such RL approaches circumvent the need for the model in Equation 2.3 by employing a temporal difference learning approach (inspired by Monte Carlo methods) $V(s_t) \leftarrow (1-\alpha)V(s_t) + \alpha[R(s_t, a_t, s_{t+1}) + \gamma V(s_{t+1})]$, where α is the learning rate and γ is the discount factor. This enables learning a value function solely by the experiences, without the need for an explicit/accurate model description.

2. The TD learning also circumvents the expectation over the states $\mathbb{E}_s$. The assumption is that as the agent learns to prefer optimal actions, $V(s_t)$ would be influenced by the optimal next state and eventually converge.

3. As noted earlier, learning an optimal value function involves visiting all the states and estimating the value functions from the respective states. This may not be feasible in the case of a continuous state space, RL addresses this problem by learning a "critic" that learns to extrapolate the value function from known interactions to unexplored states. This is formerly referred to as value function approximation.

4. Finally, the problem of expectation over actions $\mathbb{E}_a$ is solved by policy function approximation. Here the idea is to directly optimize a policy $\pi_\theta : \mathbb{R}^{n_s} \rightarrow \mathbb{R}^{n_a}$ by updating the policy parameters θ that would lead to predicting optimal actions as a function of state. Again, the idea here is to extrapolate the learnings between states.

Typically, RL involves elements of the policy iteration algorithm (Algorithm 1). Specifically, the iterative policy evaluation and policy improvement steps and the optimal policy is then derived from the optimal value function via the $\pi \sim \arg\max_{a_t} \Sigma_{s_{t+1}} P(s_{t+1}|s_t, a_t)\,[\,R(s_t, a_t, s_{t+1}) + \gamma V_k(s_{t+1})]$. However, these steps are not feasible in case of practical applications involving large and/or continuous state/action spaces. Large or continuous state spaces make it infeasible to run the iterative policy evaluation step. Large/continuous action spaces make it furthermore impractical to run the policy improvement step, as the operator $\pi \sim \arg\max_{a_t}$ becomes intractable.

These bottlenecks can be solved by parametrizing the policy and the value function as follows,

$$\hat{V}_\phi(s_t) = \big[\, R(s_t, \pi_\theta(s_t), s_{t+1}) + \gamma \hat{V}_\phi(s_{t+1})\big] \tag{2.4}$$

Where $\hat{V}$ represents the approximate value function predicted by the critic (parameterized by ϕ). Now the goal is to optimize the parameters θ, ϕ so that Equation 2.4 emulates the optimal Bellman equation 2.3. In the case of deep RL, both the policy and the value functions are parametrized by deep neural networks, leveraging the ability of deep neural networks as non-linear function approximations. In this

case the parameters θ, ϕ in Equation 2.4 correspond to the weights of the neural network which are then optimized using a gradient-based update rule.

This type of parametrization in deep RL is referred to as an actor-critic architecture, where the actor is simply the policy network, responsible for choosing an action given a state, and the critic is the value function responsible for discerning the consequence of the given state/state-action pair. Although there are several variants and implementations of deep-reinforcement learning algorithms [Wang et al., 2024b; Arulkumaran et al., 2017], the core idea is the same, the actor is optimized via the policy gradient theorem (described subsequently), and the critic is optimized to reduce the Bellman error.

The objective function for the critic is as per Equation 2.5, which simply represents the Bellman error or the TD target. $V_\phi(s_t)$ is the value prediction of the critic network given the current state and $V_{\phi_{\text{target}}}$ is a target critic, to circumvent the moving target problem. Often, $V_{\phi_{\text{target}}}$ is simply a delayed version of the same critic. And the expectation $\mathbb{E}_\tau$ is with respect to the observed trajectories (τ), i.e all the observed state-action-reward-state-action tuples (Equation 2.5).

$$L_\phi = \min_\phi \mathbb{E}_\tau \left[V_\phi(s_t) - \left(r_t + \gamma V_{\phi_{\text{target}}}(s_{t+1}) \right) \right]^2 \tag{2.5}$$

The objective function for the policy network on the other hand is more intricate and is derived using the so called policy gradient theorem. The high-level objective for the critic is to simply maximize the expected advantages ($\hat{A}$, a measure of value of the next state),

$$L_\theta = \max_\theta \mathbb{E}_{\tau \sim \pi_\theta} \left[\Sigma_{t=0}^{T-1} \hat{A}_{\pi_\theta}(s_t, a_t) \right] \tag{2.6}$$

The advantages $\hat{A}_{\pi_\theta}(s_t, a_t)$ can simply be the MC estimate (REINFORCE algorithm) or the observed reward minus the base value function (vanilla policy gradient). The optimization process involves taking a step along the negative of the gradient, i.e. $\theta = \theta - \alpha \nabla_\theta L_\theta$, which implies that a crucial step is to compute the gradient of the objective function with respect to the policy parameters. However, this is not as straightforward due to the outer expectations involved. It would have been more convenient if the gradient was inside the expectation. We can attempt to achieve this by expanding the outer expectation and by using the log trick.

We can start by computing the gradient of the expected returns (2.6) with respect to the policy parameters θ,

$$-\nabla_\theta L_\theta = \boxed{-\nabla_\theta E_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T-1}\hat{A}_\theta\left(s_t,a_t\right)\right]} \tag{2.7}$$

$$= -\nabla_\theta \int_\tau P(\tau \mid \theta)\sum_{t=0}^{T-1}\hat{A}_{\pi_\theta}\left(s_t,a_t\right)d\tau$$

$$= -\nabla_\theta \int_\tau P(s_{t+1}|s_t,a_t)\pi_\theta(a_t|s_t)\sum_{t=0}^{T-1}\hat{A}_{\pi_\theta}\left(s_t,a_t\right)d\tau$$

$$= -\int_\tau P(s_{t+1}|s_t,a_t)\nabla_\theta\pi_\theta(a_t|s_t)\sum_{t=0}^{T-1}\hat{A}_{\pi_\theta}\left(s_t,a_t\right)d\tau \tag{2.8}$$

Equation 2.8 is intractable to approximate because it it no longer an expectation over the observed trajectories as the term $\pi_\theta(a_t|s_t)$ is lost. We could multiply and divide by the term by $\pi_\theta(a_t|s_t)$ to recover the trajectory probability,

$$-\nabla_\theta L_\theta = -\int_\tau P(s_{t+1}|s_t,a_t)\frac{\pi_\theta(a_t|s_t)}{\pi_\theta(a_t|s_t)}\nabla_\theta\pi_\theta(a_t|s_t)\sum_{t=0}^{T-1}\hat{A}_{\pi_\theta}\left(s_t,a_t\right)d\tau$$

Using the fact that $P(\tau \mid \theta) = \pi_\theta\left(a_t \mid s_t\right)P\left(s_{t+1} \mid s_t,a_t\right)$ and $\nabla_\theta \log\pi(\tau \mid \theta) = \frac{1}{\pi_\theta(a_t|s_t)}\nabla_\theta\pi_\theta(a_t|s_t)$, we have,

$$\Rightarrow \nabla_\theta L_\theta = -\int_\tau P(\tau \mid \theta)\nabla_\theta \log\pi_\theta\left(a_t \mid s_t\right)\sum_{t=0}^{T-1}\hat{A}_{\pi_\theta}\left(s_t,a_t\right)d\tau$$

$$= \boxed{-E_\tau\left[\sum_{t=0}^{T-1}\nabla_\theta \log\pi_\theta\left(a_t|s_t\right)\hat{A}_{\pi_\theta}\left(s_t,a_t\right)\right]} \tag{2.9}$$

Comparing Equations 2.7 and 2.9, we have accomplished the task, thus the expectation $\mathbb{E}_\tau$ can be replaced by empirical averages, something that is widely used in batch optimization and machine learning. The above policy gradient theorem is central to all actor-critic algorithms: A2C, TRPO, PPO, DDPG, TD3 ([Mnih et al., 2016; Schulman et al., 2017a; Schulman et al., 2017b; Lillicrap et al., 2019; Fujimoto et al., 2018]).

Since this work employs the proximal policy optimization (PPO) algorithm primarily, it is worth highlighting some of the implementation aspects of PPO. One of the properties of PPO is that it is an on-policy algorithm, i.e. the policy being optimized is also the one used for exploration and capturing the data

(trajectories). The nuance in Equation 2.7 is that the expectation expands as $\Sigma_{s_0}\rho(s_0)\Sigma_{a_t\sim\pi_\theta}\pi_\theta(a_t|s_t)\hat{A}_{\theta_{\text{old}}}(s_t,a_t)$, where $\hat{A}_{\theta_{\text{old}}}$ is the evaluation under the old policy and the critic while the action probabilities are computed by the current policy $\pi_\theta(a_t|s_t)$, and ρ is the initial state distribution. PPO (and its predecessor TRPO) overcome the discrepancy by introducing importance sampling, $\Sigma_{a_t\sim\pi_\theta}\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}\hat{A}_{\theta_{\text{old}}}(s_t,a_t)$ which makes it feasible to use the estimate of the old policy under the new policy in Equation 2.7. PPO takes this a step further by clipping the probability ratio to ensure stability during learning as in Equation 2.3

$$L_\theta = E_\tau\left[\min\left(r_\tau(\theta)A_\tau, \text{clip}(r_\tau(\theta), 1-\varepsilon, 1+\varepsilon)A_\tau\right)\right] \tag{2.10}$$
$$\textit{where, } r_\tau(\theta) = \frac{\pi_\theta(a \mid s)}{\pi_{\theta_{old}}(a \mid s)}$$

Where $\varepsilon > 0$ is a clipping parameter. Essentially, the sampling ratio r_τ is clipped between $1-\varepsilon$ and $1+\varepsilon$ so that the policy does not diverge too much between updates, ensuring stability.

Phasic policy gradient (PPG, [Cobbe et al., 2020]) is a successor of PPO, which aims to make better use of shared network structures between the policy and value net, without compromising the performance, while both the policy and value net can benefit from learning common features. The implementation however has not evolved yet, thus this work primarily uses PPO.

2.4 Multi-agent Reinforcement Learning (MARL)

Multi-agent RL builds upon the idea of single-agent RL by reformulating the problem and having multiple agents working together or against each other. This is in stark contrast to the data parallelism approach, where all the agents would solve the same problem but benefit from data parallelism (wisdom of crowd). MARL, on the other hand, breaks down the original problem into multiple/simpler problems for the agents to collectively solve. One motivation for doing so might be due to the nature of the problem, e.g. a team of RL agents playing a game of soccer, or multiple agents performing decentralized control in a power grid. Decentralized learning can be an advantage in cases where the agents cannot fully observe the states and the actions of the system. For instance, in the case of control of a large power grid, it would be impossible for every station to have full access to the measurements of the whole system. The decentralized execution forces the agents to work with limited or local observations and anticipate the behavior of the other agents.

Another motivation for employing MARL would be to exploit the decoupling between the different systems to improve the sampling efficiency in RL. Let $\mathscr{A} = \{\mathscr{A}_1, \mathscr{A}_2, \ldots \mathscr{A}_{n_a}\}$ be the joint action space comprising of individual action spaces of the agents. Learning an optimal policy in RL involves the agent sufficiently exploring the action space, so a single agent would have to explore the entire

joint action space $|\mathscr{A}_1| \times |\mathscr{A}_2| \times \ldots |\mathscr{A}_{n_a}|$, where $|\cdot|$ denotes the cardinality of the action space. But if the subsystems can be viewed as independent or are decoupled to a sufficient degree, then one can have the agents simultaneously explore their individual action spaces for faster exploration, $|\mathscr{A}_1| + |\mathscr{A}_2| + \ldots |\mathscr{A}_{n_a}|$, which would in turn translate to better sampling efficiency in RL.

Thus the goal in MARL is now to learn a set of policies $\pi = \{\pi_{A_1}, \pi_{A_2}, \ldots, \pi_{A_n}\}$ that would jointly constitute the optimal policy. Some of the key challenges in MARL are the problems of credit assignment and non-stationarity. The problem of credit assignment arises due to the fact that the critic should accurately discern the individual proportion of the reward, given a joint action, i.e. which agent's action was it that led to a higher/lower observed reward? The problem of non-stationarity is due to the fact that each of the agents learns individually and observes the effect of other agents via the environment. This implies that each agent has to not only the dynamics of the environment (in the case of model-free RL) but also learn to anticipate the behavior of other agents.

These challenges are addressed in MARL by relying on communication mechanisms for modeling agent-agent interaction. Specifically, this translates to the level of visibility between the agents during the learning process. For instance, it may be assumed that all of the agents can see each other's actions during the training phase, in other words, the agents share their executed action with all the other agents. Another mechanism could be much more conservative, where the agents do not have the liberty to share their actions, but only their observed rewards, so that the agents can collectively maximize the rewards earned by all the agents. The former approach is termed as centralized training and decentralized execution (CTDE), where the training happens assuming complement visibility, while the execution is still decentralized. The latter approach serves as the motivation for value-decomposition or value-mixing approaches in MARL, where the agents are optimized to maximize a "mixed" cumulative reward.

As agents operate on local and global information, the MARL problem formulation is often based upon partially observable Markov decision processes (POMDPs). A POMDP builds upon the typical Markov decision process, except that the states are only partially observable,

- Let s_t be the global state, shared among all the agents. And $o_{i,t}$ denote the local observation for the agent A_i observed as per the observation function $\mathscr{O}(o_{i,t+1}|s_{t+1}, a_t)$. Then the complete observable state for agent A_i would be $z_{i,t} = \{s_t, o_{i,t}\}$.

- Let $a_{A_i} \sim \pi_{A_i}(z_{i,t})$ denote the action executed by agent A_i according to its policy (acting on its local observation). Thus, the joint action is simply the set of individual agent actions, $\{a_{A_1}, a_{A_2}, \ldots, a_{A_n}\}$.

- Each of the agents now observe individual rewards $R_{A_1}, R_{A_2}, \ldots, R_{A_n}$.

- Let $<>$ denote the concatenation operation. Thus, $< o_{1,t}, o_{2,t}, \ldots, o_{n,t} >$ would denote that joint observation and $< a_{A_1}, a_{A_2}, \ldots, a_{A_n} >$ denote the joint action.

The transition function is then given by $P(\tilde{s}_{t+1}, o_{t+1} | \tilde{s}_t, a_t)$, here $\tilde{s}$ denotes the complete (unobservable) system state, and o is the joint observation set. The goal for each is to learn an optimal policy $\pi^*_{A_i}$ that maximizes its own cumulative rewards, but at the same time enables other agents to maximize their rewards (as the reward received by a particular agent depends on the joint action executed). For each of the agents to maximize their individual rewards might be quite straightforward, which can be viewed as each agent running an independent single-agent RL task. However, it might lead to suboptimal policies [Tan, 1997], as the joint optimal may not be simply the combination of individual optimal policies. This approach in fact is formally referred to as the "independent learning" approach in MARL and serves as a benchmark. The two broad classes of MARL algorithms, centralized training, and decentralized execution (CTDE) and value-decomposition approaches are designed to alleviate these problems in MARL. The algorithms differ in terms of the level of coordination and communication between the agents.

The centralized training and decentralized execution approach ([Kraemer and Banerjee, 2016; Oliehoek et al., 2008]) assumes complete visibility, i.e. the agents communicate their executed actions and rewards during the training, while the execution happens on the local observations. This means that the critic in the CTDE approach is tasked to learn $Q(\{s_t, < o_{1,t}, o_{2,t}, \ldots, o_{n,t} >\}, < a_{A_1}, a_{A_2}, \ldots, a_{A_n} >)$, while the agents themselves act according to $a_{A_i} \sim \pi_{A_i}(z_{i,t})$ and receive respective rewards. The policies are optimized using the policy gradient theorem utilizing the central critic. The CTDE trades off the level of decentralization for better learning. However, in purely decentralized applications access to the complete state and actions might be restrictive. Also, the critic is exposed to more information (the joint state and action) which might result in slower learning if the agents are decoupled.

The value-decomposition/mixing approaches ([Rashid et al., 2018; Sunehag et al., 2017]) on the other hand, operate under more restrictive assumptions. Each agent, instead of sharing its observation and executed action, now only shares its reward (the critic estimate). To enable cooperative learning, a central mixer mixes the individual critic estimates to derive a global loss. This implies that each of the agents themselves has localized critics trained on local information, $Q_\theta(z_{i,t}, a_{A_i})$, while the central mixes these estimates, $Q_{\text{tot}} = f_{\theta_{\text{mix}}}(Q_{A_1}, Q_{A_2}, \ldots, Q_{A_n})$. The parameters $\theta, \theta_{\text{mix}}$ are then jointly optimized based on the TD error, $L_{\theta,\theta_{\text{mix}}} = \left[Q_{\text{tot}} - \left(r_t + \gamma Q_{\text{tot, target}}\right)\right]^2$. Thus, mixing approaches tradeoff learning for decentralization. [Gronauer and Diepold, 2022; Stefano V. Albrecht, 2024] provide a more detailed overview of the algorithms and other learning strategies in MARL.

3

Exploratory Data Analysis

This section takes a data-driven approach to reveal the characteristics of the system and other insights that will be useful in formulating reinforcement learning experiments. In any MIMO system, it can be overwhelming to operate on all the state and control variables involved (Table 1.1) and reinforcement learning is no exception. Besides, operating on the entire set of inputs might be redundant, as often there are variables that have a marginal impact on the desired output. Thus, as a primary objective, a set of experiments is carried out to quantify the impact of input variables on the output variables (power consumption). The results from these experiments are then used to build an intuition and to formulate a reduced landscape for RL optimization.

The decision variables listed in Table 1.1 are of two types, discrete (ON/OFF) and continuous variables (setpoints). This implies that there are potentially two primary control strategies to exploit: **staging control**, where the individual equipments (chillers, cooling towers, pumps, fans) can be turned ON or OFF, and **setpoint tuning**, where the setpoints for running equipments can be tweaked. Both control strategies can potentially save energy depending on the operating conditions and external factors. The following experiments help quantify the saving potential and the impact of the controls on the overall power consumption.

3.1 Power Distribution

Before moving on to correlation analysis between the inputs and output, it is worth analyzing the power distribution of the system. Figure 3.1 shows the power contributions of individual equipment during typical operating conditions. All the setpoints were set to nominal values with all equipments turned ON. It can be observed that the chillers, cooling water pumps, and chilled water pumps are the major contributors, with the chillers and the cooling towers responsible for nearly 75% of total power consumption. Thus, the chillers and cooling towers would be a primary focus for optimization. It is worth noting that the chiller operation is affected by both the staging and setpoints, while the cooling towers can only be turned ON or OFF.

Thus, to fully exploit the saving potential, it is imperative to explore both control strategies.

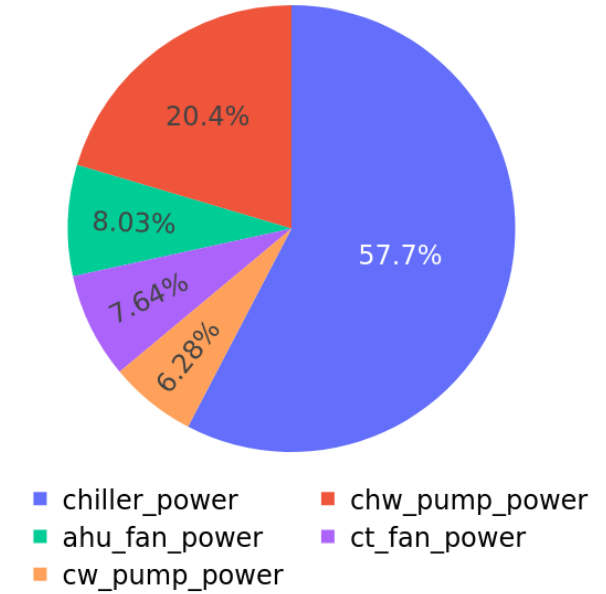


Figure 3.1 Power distribution between equipments: chillers and cooling towers account for nearly 75 % of total power consumption

P_{ch}: `chiller_power`, P_{ct}: `chw_pump_power` + `ct_fan_power`, P_{chw}: `ahu_fan_power` + `chw_pump_power`

3.2 Sensitivity Analysis

The idea here is to quantify the impact of the input variables (DECISIONS) on the output variables (performance metrics), which would then serve as a motivation for selecting a subset of control variables for RL optimization. Given the transition function $s_{t+1}, R_t = f(s_t, a_t, w_t)$, we are interested in the impact of the decision variables s_t on the output variables. It is worth noting that the input and output variables here are vectors (i.e. MIMO system) and that the transition function is non-linear, constraining the method of analysis, the typical grey box identification and least-squares analysis would produce inaccurate results in case of a system with non-linear dynamics. Thus, sensitivity analysis is carried out as a systematic and prudent way to measure the influence of each of the input variables.

Sensitivity analysis includes computing the partials $\delta_i = |\frac{\partial f(s_t,a_t)}{\partial a_i}|$, where the index i denotes the ith action. While this is feasible to compute in the case of continuous actions (such as the real setpoints), it is not possible to approximate the effect of staging by computing the partial derivatives. Thus, to quantify the effect of discrete variables, the analysis resorts to combinatorial experiments by considering all possible combinations.

The partials $\frac{\partial f(s_t,a_t)}{\partial a_i}$ are approximated by sweeping each of the setpoints from its minimum to its maximum value (see Table 1.1) while keeping all the other controls constant and with all the equipments turned ON, and repeating the process for different load conditions. The magnitude of the partials with respect to the actions (δa_i) should give us an approximate estimate of the strength of the correlation of the particular action. Thus, we would like to pick actions with a higher correlation. Figure 3.2a plots the results for different load conditions.

For the staging experiments, the setpoints are fixed to their nominal values (middle value in their respective ranges), and different combinations of chillers and cooling towers are tested. Since pumps are not major contributors to the overall power consumption (Figure 3.1), the pump staging is ignored. Figure 3.2b plots the results. If a staging combination has a significant impact, we would expect a substantial change in the total power upon switching the control. Although the strength of the correlation can be hard to determine, these results can be interpreted to build intuition for the characteristics of the optimal staging.

The following observations can be made by inspecting the results,

1. Figure 3.2b reveals that there is a significant potential to save power via equipment staging (turning equipment ON/OFF) at higher loads than at lower loads.

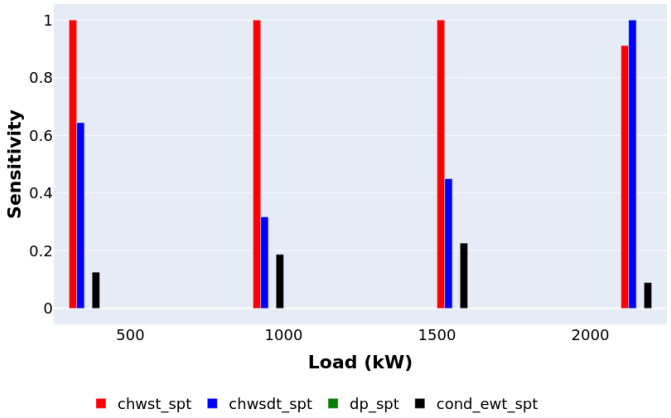
2. For moderate and higher loads, chiller staging appears to have a significant impact. For instance, at loads 1000 kW and 2000 kW, running two chillers appears to be more optimal than just one chiller ON.

3. At higher loads (greater than 1500 kW), it is essential to have both chillers ON to meet the load requirements successfully.

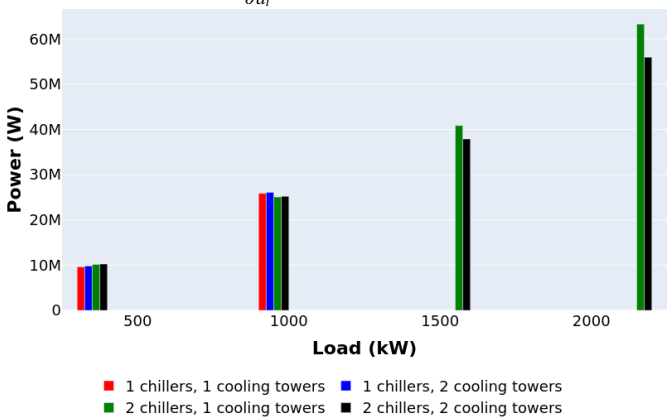
4. In the case of setpoints (Figure 3.2a), the chiller water temperatures ($T_{\text{chwst}}, T_{\text{chwsdt}}$) appear to have the most impact across the operating load conditions. While the evaporating water temperature (T_{ewt}) appears to have a lesser but significant impact.

5. The differential pressure setpoint (dp), on the other hand, appears to have little to no impact on the total power consumption.

6. T_{chwst} appears to have a significant impact on the power consumption across the operating load conditions, while the differential water temperature (T_{chwst}) appears to have a relatively higher impact at lower and higher loads.



(a) Sensitivity ($|\frac{\partial f(s_t,a_t)}{\partial a_i}|$) of setpoints (scored relatively)



(b) Impact of equipment staging

Figure 3.2 Sensitivity analysis - shows the impact of setpoints and equipment staging on total power under varying load conditions. Variables are described in Table 1.1. Outdoor temperature (T_{db}): 22 °C, humidity (H): 0.8

3.3 Efficiency Curves

This section conducts more rigorous experiments to determine the impact of setpoints on overall efficiency. Efficiency here includes the total power consumption and the coefficient of power (COP) in the case of the chillers. A preliminary experiment includes sweeping one setpoint at a time (while the other real setpoints stay fixed to a nominal value in their respective range) and performing equipment staging at the same time. Further experiments include performing a sweep of the setpoints that are deemed significant for a chosen set of cooling loads and external

weather conditions to reveal the characteristics of the optimal. While finding the true optimal setpoints and staging sequence solely through data analysis can be a challenging task due to the number of factors and controls involved, the intention of these experiments is to help identify a potential solution that could serve as a benchmark for RL experiments and help interpret the results.

The results from the previous section implied that the staging control variables could have a considerable impact depending on the cooling load and that the setpoints had a significant impact across the loads. Thus, as a first step, the control variables are swept from their minimum to maximum value, and at the same time, the stagging sequence is also varied, from having none of the equipments ON to having all of the equipments ON. Figure 3.3 shows the results, where the curves represent the total power as a consequence of sweeping the particular control variable (X-axis) and the error bands capture the impact of equipment staging, one curve per operating cooling load. Similarly, Figure 3.4 plots the chiller equipment efficiency as a consequence of the setpoints and staging. The chiller efficiency is the chiller part load ratio, the ratio of the actual cooling load to the chiller's cooling capacity. Ideally, we would like to utilize the full potential of the chiller to maximize efficiency. It is evident that there exists an optimal staging as well as setpoint control. The optimality in setpoint control is implied by the peaking of the curves, while optimality in staging is further implied by the variation (error bars in the Figure) at this peak efficiency or power consumption.

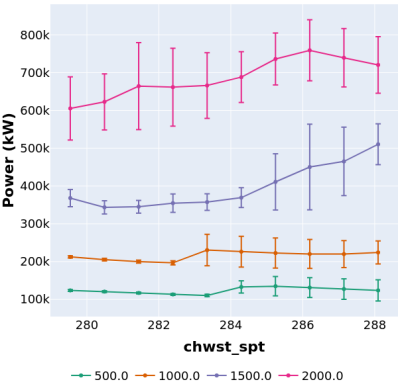
The following observations can be made by inspecting the results in Figures 3.3 - 3.4:

1. The chiller setpoints $T_{\text{chwst}}, T_{\text{chwsdt}}$ appear to have the most influence on the total power, while the evaporating water temperature (T_{ewt}) appears to have a smaller impact on the power consumption. The rest of the variables, the differential pressure-temperature (dp) for instance, has no observable impact.

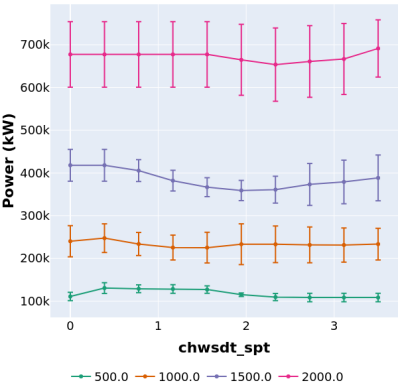
2. In terms of chiller efficiency curves (Figure 3.4), we again observe that $T_{\text{chwst}}, T_{\text{chwsdt}}$ have a dominant role on the overall operating efficiency of the chillers, which may, in turn, translate to lower power consumption.

3. In both sets of curves, we observe that equipment staging can have a significant impact, more so in case of higher loads than in the case of lower loads.

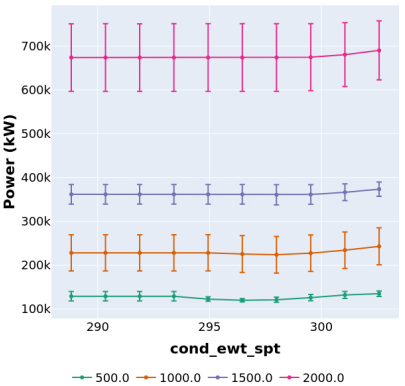
The key takeaway from these experiments is that there is an optimal region of operation defined by staging and setpoint control and that there exists a reduced set of variables to work with that can potentially save energy. Although these experiments evaluated the impact of the control variables, each control variable was varied one at a time, so the true optimal may not necessarily correspond to individual optimal setpoints observed. Finding an optimal combination of controls given the external conditions and the reference load is an optimization problem in itself,



(a) Effect of chiller setpoint (T_{chwst})



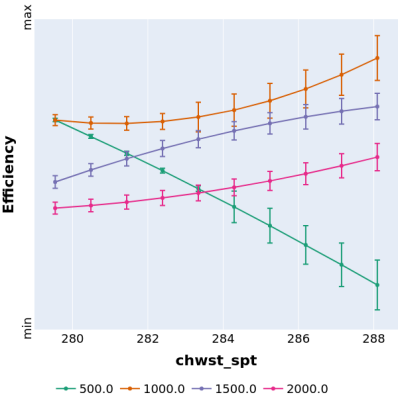
(b) Effect of chiller differential setpoint (T_{chwsdt})



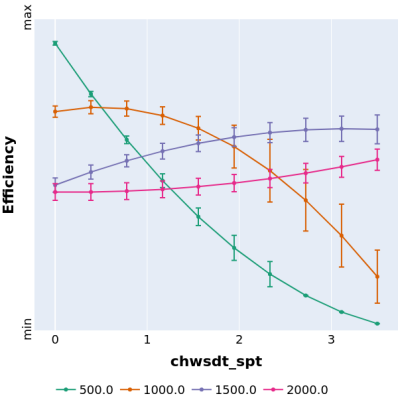
(c) Effect of condenser evaporating temperature (T_{ewt})

Figure 3.3 Power curves manifest the optimality of power consumption in terms of control setpoint (X-axis) and equipment staging (error bars). Outdoor temperature (T_{db}): 22 °C, humidity (H): 0.8. For each experiment (subplot), all the other setpoints are fixed to their middle values.

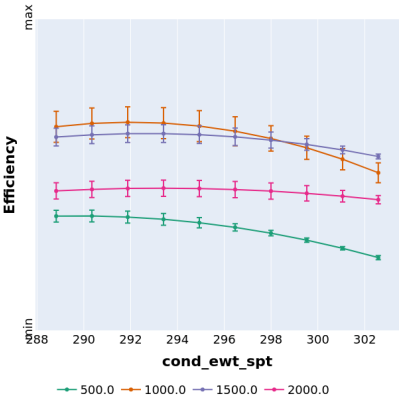
and it may not be feasible to arrive at a true optimal solely by such data analysis and experimentation. Figure 3.5 further explores joint optimal as a function of `chwst_spt, chwsdt_spt` for 4 sets of cooling loads. It is clearly evident that there are optimal regions of operations for these setpoints, and conversely mistuning these setpoints can incur a staggering penalty of anywhere between 100 kW to 200 kW. Thus, the objective of RL optimization is to learn such joint terrains and an optimal control policy.



(a) Effect of chiller setpoint (T_{chwst})



(b) Effect of chiller differential setpoint (T_{chwsdt})



(c) Effect of condenser evaporating temperature (T_{ewt})

Figure 3.4 Power curves manifest the optimality of power consumption in terms of control setpoint (X-axis) and equipment staging (error bars). Outdoor temperature (T_{db}): 22 °C, humidity (H): 0.8. For each experiment (subplot), all the other setpoints are fixed to their middle values.

3.4 Test Cases - Optimal Regions

Insights from these experiments can also be translated into a set of test cases that can serve as a guideline for the verification of reinforcement learning algorithms and discern the performance of the RL algorithms at later stages. Another way these insights may be useful maybe to impart learning via imitation learning, where an expert policy imparts the optimal/suboptimal controls onto the agent as part of pre-training, but this approach is beyond the scope of the thesis.

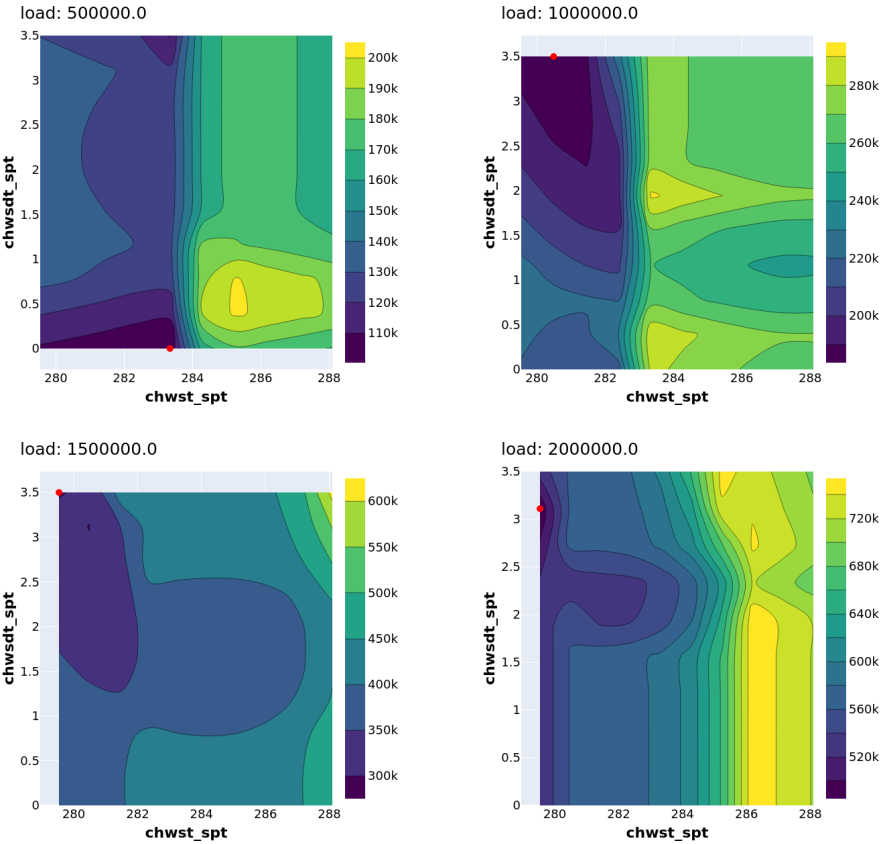


Figure 3.5 Chiller efficiency curves indicating equipment efficiency in terms of control setpoint (X-axis) and equipment staging (error bars). Higher values are considered good. Outdoor temperature (T_{db}): 22 °C, humidity (H): 0.8. For each experiment (subplot), all the other setpoints are fixed to their middle values.
`chwst_spt` - T_{chwst}, `chwsdt_spt` - T_{chwsdt}, `chwst_spt` - T_{ewt}, `dp_spt` - dp.

We observe that the optimal T_{chwst} shifts from a higher value to a lower value as the cooling load is increased (Figure 3.3). The optimal T_{chwsdt}, on the other hand, shifts from a lower value at lower loads to a higher value at higher loads (Figure 3.3). Furthermore, the chiller efficiency also peaks at these optimals (Figure 3.4), which further implies the optimality of these setpoints. Table 3.1 summarizes the optimal setpoints for fixed load and external conditions. The optimal setpoints were recorded as per the experiments in Figure 3.5, with all the equipments turned ON. These setpoints may serve as a guideline for benchmarking RL as part of RL experiments carried out at later stages.

A further possibility may be to interpolate the optimal for the intrim loads and repeat the process for other outdoor temperatures and humidity. But as mentioned before, that would in turn allude to an optimization problem, which is better off handled by reinforcement learning rather than carrying out manual experiments.

External Conditions			**Setpoints**		
Load	T_db	RH	chwst	chwsdt_spt	cond_ewt_spt
500 kW	294.15	0.8	283.34	0	299.81
1 MW	294.15	0.8	280.48	3.50	299.81
1.5 MW	294.15	0.8	279.53	3.50	299.81
2.0 MW	294.15	0.8	279.53	3.11	299.81

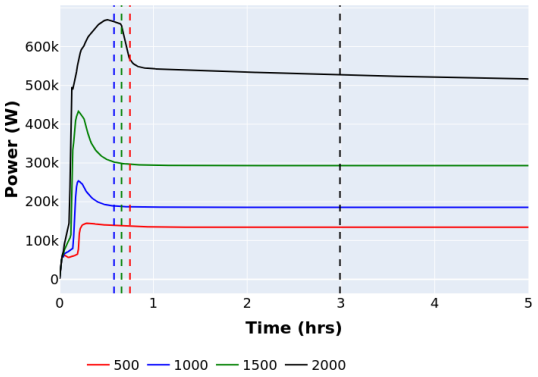
Table 3.1 Test Cases - Optimal Setpoints

3.5 Step Response

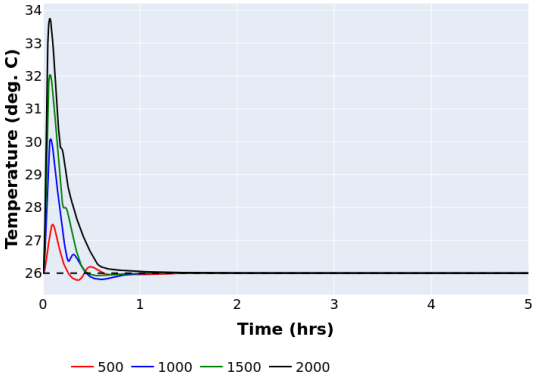
It is also worth analyzing the characteristics of the step response of the system. Characteristics such as the rise time, overshoot, and settling time will be crucial for choosing a time constant for RL experiments. The choice depends on two factors, the settling time of the total power consumption, and secondly, the time it takes for the controller to meet the temperature reference, i.e. to drive $|T_{\text{ref}} - T_{\text{room},t}|$ to zero.

Figure 3.6a plots the power output for a step load reference of varying magnitudes. The dashed lines mark the settling time for the total power to settle down 2% of its final value. It can be observed that for low and moderate loads, the total power consumption settles down within 1 hr, while the settling time is pushed to 3 hrs for a load of 2000 kW. Figure 3.6b plots the IT room temperature with the dashed line indicating the reference room temperature (of 26 °C) for fixed outdoor temperature and humidity. Across the loads, the controller appears to meet the reference within a time duration of 1 hr.

Given that the objective function (Equation 1.1) includes both total power and temperature error, it is important to consider the settling time for both responses. A time constant of > 3 hrs appears to be a reasonable choice to approximate the steady-state behavior under all the load conditions. Nevertheless, a smaller time constant can be chosen for more dynamic simulations.



(a) Total power. Dashed lines mark the settling time.



(b) IT room temperature. Dashed line marks the reference.

Figure 3.6 Response characteristics for different operating loads. Outdoor temperature (T_{db}): 22 °C, humidity (H): 0.8

4

Reinforcement Learning Methodology

Based on the observations and insights gained from carrying out the exploratory data analysis, the following section breaks down the original problem into tangible problems that can be solved by employing reinforcement learning. The intention here is to introduce complexity gradually and fine-tune the problem formulation by incorporating the learnings along the way. Tackling multi-agent RL (MARL) right away can be a daunting task due to a number of reasons, non-stationarity, multi-agent credit assignment, and agent-agent interaction are some of the major challenges in MARL. Thus, the optimization problem is first addressed in the context of single-agent reinforcement learning and then gradually introduce MARL into the mix, comparing and contrasting the problems and benefits of MARL.

4.1 Setting the Stage for RL

The problem of optimizing controls of the chiller plant to minimize the operating power consumption can be modeled as a sequential decision process (MDP), as immediate controls can have long-term effects, and reinforcement learning can be viewed as a means to solve the dynamic programming (DP) problem. The goal is to learn an optimal policy $\pi : \arg\max_{\pi} \Sigma_{s_{t+1}} P(s_{t+1}|s_t, \pi(s_t))\,[\,R(s_t, a_t) + \gamma V^*(s_{t+1})]$ (as introduced in Section 2). Reinforcement learning relies on learning a policy by interaction, the agent has to trade-off between exploration and exploitation to learn the best strategy based on the feedback received (reward). The available functional mock-up unit (FMU) emulating the dynamics of a real plant is utilized for performing offline learning while adhering to action constraints.

Although reinforcement learning can be seen as an optimization problem, unlike other conventional optimization problems like linear quadratic regulator (LQR) or model predictive control (MPC), reinforcement learning is accompanied by several challenges [Dulac-Arnold et al., 2019] that are not associated with the typical optimization problems. Some of the significant challenges surrounding RL are, i)

scalability of states and action spaces, ii) specificity of reward functions, and iii) observation and actuator constraints. These issues can have a tremendous impact on the solution of reinforcement learning. Thus it is imperative to have the right motivation with these aspects of the RL problem formulation.

The problem of having large state spaces is not as severe compared to large action spaces, this is because the deep RL policies (neural networks) are typically good at focusing on the relevant aspects of the inputs and building useful non-linear representations, but the key here is to capture the relevant inputs in the first place. The cardinality of action spaces, on the other hand, can be a significant bottleneck. In order for the agent to learn an optimal policy, the agent has to sufficiently explore the action space and then figure out the optimal sequence. Suppose cardinality is n_a actions each with a domain $\mathscr{A}_i \in \mathbb{R}$, then exploration involves exploring the joint space of size $|\mathscr{A}_1| \times |\mathscr{A}_2| \times \ldots |\mathscr{A}_{n_a}|$ given a particular state (several exploration strategies exist to effectively explore the action terrain [Ladosz et al., 2022]). Thus the cardinality and the domain of the action spaces play a significant role and large action spaces result in slower convergence. While learning an optimal solution does require exploring the whole action space, it can often be overwhelming and infeasible in the case of continuous action spaces, and the idea to extrapolate the learnings based on the explorations.

Reward function formulation is unarguably one of the critical aspects of reinforcement learning. An underspecified reward function (failing to capture the required objectives) may lead to a suboptimal solution and an overspecified reward function on the other hand can be detrimental, with the agent not learning anything at all. The sole objective in RL is for the agent to simply maximize the rewards using whatever strategies are at the agent's disposal, in other words, the reward function defines the policy learned by the agent. The problem of reward shaping can be further exemplified if the agent is working towards multiple objectives. For instance (as in this case), maintaining the cooling load requirement, minimizing the power consumption, and meeting actuator constraints. Such conflicting objectives can often muddle the objectives and the feedback signal. Thus extensive care has to be taken in crafting the reward function in RL [Tamar et al., 2015; Roijers et al., 2013]. Laying more emphasis on the gains incentivizes, the agent often learns faster, for instance, a non-linear reward function can be used to accentuate gains and suppress the pitfalls (more subtleties surrounding reward reshaping will be discussed in the subsequent sections).

Physical systems are often accompanied by constraints, whether it be constraints surrounding the observation of states or the actuator constraints - hard and rate-limiting constraints (see Table 1.1), alluding to safe reinforcement learning [Gu et al., 2023]. While there have been theoretical frameworks that explicitly model the constraints by formulating a constrained MDP [Gattami et al., 2021], there have not been practical implementations that are robust. Within the framework of classical RL, a formal way to incorporate constraints is to perform clipping and introduce an action penalty, where the actions are clipped to their limits (min/max value)

and the action penalty introduces a proportional negative reward for frequent action switching. The proposed work follows a similar approach and ensures that the hard constraints are met at all costs and that the rate-limiting constraints are met to a sufficient degree.

The single-agent approach involves a single agent exploring the joint action domain to learn an optimal policy, while the multi-agent approach would break down the problem-setting to decompose the action space so that individual agents would explore their respective action spaces while still working towards a common goal. The motivation for multi-agent reinforcement learning (MARL) is to exploit the decoupling between the action spaces that may lead to faster exploration (in turn faster learning), $|\mathscr{A}_1| + |\mathscr{A}_2| + \ldots |\mathscr{A}_{n_a}|$ instead of $|\mathscr{A}_1| \times |\mathscr{A}_2| \times \ldots |\mathscr{A}_{n_a}|$, i.e. each agent would explore its own action space instead of a single agent explore the joint action space. However, MARL introduces additional problems that were non-existent in RL [Busoniu et al., 2008; Wong et al., 2022]: i) non-stationarity, ii) multi-agent credit assignment, and iii) partial observablity. In fact, if the subsystems are highly coupled, these challenges can exacerbate the learning process and lead to a suboptimal solution. [Stefano V. Albrecht, 2024] is a comprehensive review of the challenges in MARL and contrasts MARL with single-agent RL.

The problem of non-stationarity arises due to the fact that the term $\Sigma_{s_{t+1}} P(s_{t+1}|s_t, \pi(s_t))$ in the Bellman equation (Equation 2.4) turns into $\Sigma_{s_{t+1}} P(s_{t+1}|s_t, < \pi_{A_i}, \pi_{-A_i} > (s_t))$, where π_{A_i} is the policy for agent A_i and $\pi_{-A_i} = \{\pi_{A_j} \mid A_j \neq A_i, \forall A_i \in \{agents\}\}$, a set of all the policies except agent i, and the notation $<>$ indicates concatenation. The implication is that the agent not only has to learn the dynamics and the consequences of its own actions but also learn to distinguish between the dynamics and effects of other agents. The problem is further exacerbated by the fact that each of the agents in the mix is learning themselves (moving target), adding to the non-stationarity of reinforcement learning. Thus in the case of strongly coupled systems, decoupling can make things worse for RL.

Multi-agent credit assignment is another vital aspect of MARL. In a single-agent RL setting, the MDP assumption assigns a reward $R(s_t, a_t, s_{t+1})$ as a consequence of action a_t given a state s_t at time t to the agent. As we now have multiple agents, the immediate reward function would turn into $R(s_t, < a_1, \ldots a_n >_t, s_{t+1})$. As the agents rely on the reward signal to learn an optimal policy, the underlying problem now concerns the distribution of the global reward, in other words, which of the agents' actions contributed significantly towards the observed reward, and which of the agents had a negative effect? The subsequent sections will formulate multiple variants of MARL to counter this problem and propose a few novel methods to alleviate the issues.

The last problem of partial observability in MARL is due to the decentralized nature of the problem and turns the MDP into POMDP [Åström, Karl Johan, 1965]. The agents now have a local observation ($o_{i,t}$) and can optionally share a global state (s_t), i.e. $z_{i,t} = \{s_t, o_{i,t}\}$, which would act at as the state vector for reinforcement for the particular agent. If there are no restrictions with respect to observability, all the

agents can simply share a global state (as in the context of this work).

4.2 RL Problem Formulation

A typical problem formulation in reinforcement learning involves identifying the states, actions, and reward functions. However, because of the issues discussed before, it is important to identify and rightly motivate these choices and simply not include all the available/redundant information since this would slow down the learning process. Some of the choices presented are based on hindsight but are nevertheless sufficiently motivated.

The task for the reinforcement learning agent is to learn an optimal control strategy to run the chiller plant. The objective is to minimize the operational power consumption while meeting the cooling load requirements. The chiller plant comprises of discrete and continuous states and actions (refer Table 1.1). The discrete controls correspond to the equipment staging/status, $n_{\text{chs}}, n_{\text{cts}}, n_{\text{pps}}$ corresponding to the number chillers, cooling towers and pumps to run. Although each of discrete variables can be treated as a binary variable (one per equipment), the intention of having discrete variables was to circumvent the combinatorial overhead. For instance, given that the plant consists of 2 chillers, 2 cooling towers, and 3 pumps that can turned ON/OFF, the joint domain in case of binary representation would then be $|\mathscr{A}_{\text{stag}}| = 2^2 \times 2^2 \times 2^3 = 128$, while the discrete representation would result in $|\mathscr{A}_{\text{stag}}| = 2 \times 2 \times 3 = 12$, a drastic reduction in the action space exploration. As part of the exploratory data analysis, the setpoints $T_{\text{chwst}}, T_{\text{chwsdt}}, T_{\text{ewt}}$ (corresponding to chiller leaving water temperatures and evaporating water temperature) were identified to be the most influential on the output power consumption. Although these setpoints can take on continuous values in their respective ranges (refer Table 1.1), an attempt was made to discretize them (as per Equation 4.1) by choosing the number of intervals for the action n_a, with the a_-, a^+ denoting the minimum and maximum values for the respective action. This is because considering the respective ranges, it would be redundant to explore the entire action space, and having too granular actions can lead to frequent action switching.

$$a_{\text{discrete}} = \frac{a}{\Delta a}\ ,\ \Delta a = \frac{a^+ - a_-}{n_a} \tag{4.1}$$

Another motivation is that discrete action spaces often result in better convergence [Charbonnier et al., 2022]. But there is also a downside with the discrete representation, the discrete action space now has a larger domain $|\mathscr{A}_{\text{spts}}| = \mathbb{R}^{n_{\text{chwst}}} \times \mathbb{R}^{n_{\text{chwsdt}}} \times \mathbb{R}^{n_{\text{ewt}}}$ (instead of $|\mathscr{A}_{\text{spts}}| \in \mathbb{R}^3$). However, the benefits of discrete exploration outweigh the overhead associated with continuous action exploration and is thus preferred. With regards to honoring the hard actuator constraints, all the setpoints are clipped to stay between their minimum and maximum values. The

rate-limiting constraints are incorporated as a soft constraint or a penalty as part of the reward function.

The state space comprises all the relevant information that might potentially influence the action of the agent. The obvious factors here are the external weather conditions T_{db}, H (temperature and humidity), and the current cooling load to track l_{ref}. These states were retained in continuous form for better generalizability, as discretizing a state loses the ordering information, inhibiting the model from extrapolating its learnings. For consistent representation of states, all the continuous state variables were normalized to zero mean and unit standard deviation (Equation 4.2), where the means and standard deviations are computed based on the respective rages (as listed in Table 1.1).

The current staging status of the equipment was also included as part of the state as we expect the decisions (staging or choice of setpoints) to be influenced by the current staging status.

$$s_{\text{transformed}} = \frac{s - \mu_s}{\sigma_s} \tag{4.2}$$

Reward function formulation is another crucial aspect of reinforcement learning. As mentioned before, an over-specified or under-specified reward function can derail the learning process. The objective here is to minimize the operating power consumption while maintaining the cooling load requirement. In this regard, a naive reward function would be to simply maximize the negative of the temperature error and the negative of the power consumption. As in Equation 4.3, since both the terms are positive, maximizing the negative quantities would lead to minimizing the original quantities.

As part of data analysis, it was also observed that the primary reward objective of maintaining the load requirements was easier to meet, i.e. there existed several combinations of equipment staging that could meet the load requirement, but the desirable solution was to sufficiently satisfy the secondary objective as well. Thus the trivial approach of having $\alpha_1 > \alpha_2$ resulted in the agent not paying much attention to the power objective, as the reward would always be dominated by the load requirement component. As there were plenty of staging combinations that could meet the requirement, the solution often converges to a suboptimal one (in terms of power consumption). Therefore, α_2 was made significantly higher than α_1.

$$R(s_t, a_t, s_{t+1}) = -\alpha_1 |T_{\text{ref},t} - T_{\text{room},t}| - \alpha_2 \hat{P}_{total,t} \tag{4.3}$$

Another flaw with the naive reward function (Equation 4.3) is that the reward function with respect to the power savings is linear, which may not be the most efficient reward formulation. Instead, we can use a non-linear mapping such as $\log(\hat{P}_{\text{total},t})$ to accentuate the power savings and suppress the power expenditure. Figure 4.1 compares the negative linear function and negative log of normalized power, the log function provides a higher incentive (with respect to relative power

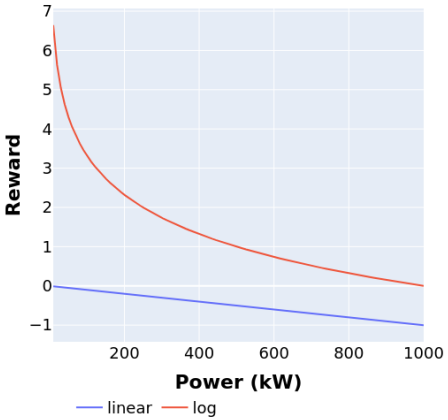


Figure 4.1 Reward functions: linear vs non-linear (log function) reward scaling

consumption) than the linear function which should have a positive impact on the learning process. Finally, the rewards are translated to optimistic rather than pessimistic, i.e. instead of maximizing the negative of the reward, the range was mapped from $[-\infty, 0]$ to the range $[0, 1]$ via clipping and min-max scaling. Thus, distinguishing the positive rewards from the termination penalty (under constraint violation). It is speculative that if $\Sigma_{t=0}^{\tau} R(s_t, a_t, s_{t+1}) > R_{\text{termination}}$, the agent would simply learn to terminate the episode by performing an invalid action, which may be part of the encounters during the initial learning phase. This was another motivation for having optimistic rewards and a negative termination reward (penalty). An alternative way would be to make $R_{\text{termination}}$ sufficiently large ($R_{\text{termination}} << 0$), although theoretically sound, it would result in large a critic variance.

Another aspect of reward function formulation is the frequency of switching actions. When dealing with a real process such as a chiller plant, it is important to ensure that the controls are not abused by frequent switching for marginal or no improvements. For instance, it would be unrealistic to turn chillers/cooling towers/pumps ON at a time instant and then have them turned OFF immediately thereafter. This characteristic of the control is incorporated into the reward function as a rate-limiting constraint, where a penalty is added if the action does not match the previous action executed, $-\mathbf{1}\{a_t \neq a_{t-1}\}$, where $\mathbf{1}$ if the indicator variable equal to 1 if the previous action differs from the present action.

Variable	Role	Range	Representation
T_{db}	STATE	$[12, 35]$ C	Continuous
H	STATE	$[0, 1.0]$	Continuous
$\dagger n_{\text{chs}}$	STATE	$\{0, 1, 2\}$	Discrete, one-hot encoding
$\dagger n_{\text{cts}}$	STATE	$\{0, 1, 2\}$	Discrete, one-hot encoding
$\dagger n_{\text{pps}}$	STATE	$\{0, 1, 2, 3\}$	Discrete, one-hot encoding
l_{ref}	STATE	$[0.5, 2]$ MW	Continuous
n_{chs}	DECISION	$\{0, 1, 2\}$	Discrete
n_{cts}	DECISION	$\{0, 1, 2\}$	Discrete
n_{pps}	DECISION	$\{0, 1, 2, 3\}$	Discrete
T_{chwst}	DECISION	$[279.538, 288.094]$	Discrete, $n^{\text{chwst}} = 10$
T_{chwsdt}	DECISION	$[0, 5.0]$	Discrete, $n^{\text{chwsdt}} = 5$
T_{ewt}	DECISION	$[299.817, 302.594]$	Discrete, $n^{\text{ewt}} = 10$
T_{room}	REWARD		
P_{total}	REWARD		

Table 4.1 System representation for RL experiments

$$R(s_t, a_t, s_{t+1}) = \begin{cases} \alpha_1 \left(1 - \frac{|T_{\text{ref},t} - T_{\text{room},t}| - T_-}{T^+ - T_-}\right) \\ -\alpha_2 \log(\max(P_{\text{total},t}, \alpha_P)) & \text{if } a_t \in \mathscr{A}_{\text{valid}} \\ -\alpha_3 \mathbf{1}\{a_t \neq a_{t-1}\} \\ \\ R_{\text{termination}}, & \text{otherwise} \end{cases} \tag{4.4}$$

The final reward function was formulated as per Equation 4.4 and addresses all the critical issues in the naive reward function (Equation 4.3). The α's in the equation act as weighing factors for the different objectives in the reward function and $R_{\text{termination}} < 0$ is the penalty for executing an invalid action which immediately ends the episode, inhibiting the agent from gathering positive reward (Table 4.2 lists hyperparameters used as part of the experiments).

Table 4.1 summarizes the system states and decision variables involved. The continuous variables are represented as is, except for normalization as per Equation 4.2, while the discrete variables are encoded using one-hot encoding. And the discrete action space is formed by discretizing the action space evenly as per the parameter n.

4.3 Multi-agent RL Enhancements

Multi-agent reinforcement learning exploits system decoupling to break down with multiple agents working towards a common goal (cooperative learning). Given the decision variables in Table 4.1, a logical decomposition can be to have two agents

Hyperparameter	Value
α_1	1.5
α_2	1.0
α_3	0.025
α_p	0.01
$R_{\text{termination}}$	-10.0

Table 4.2 Reward function hyperparameters

in the problem setting, one performing the staging control and the other performing setpoint control. This is partly motivated by the exploratory data analysis experiments (Figure 3.3), which manifested the existence of an optimal for both staging sequence and setpoint control, where it is clearly evident that the power curves peak at an optimal setpoint, and at the particular optimum, staging can have a further impact (vertical bars). Having two agents simultaneously explore the space can lead to significant improvement in sampling efficiency. While it is possible to further decompose the action space by having more agents exploring individual or subsets of controls, it is speculative that this would exacerbate non-stationarity and lead to poor/sub-optimal performance. Thus, the exploration would involve $|\mathscr{A}_{\text{stag}}| + |\mathscr{A}_{\text{spts}}|$ instead of $|\mathscr{A}_{\text{stag}}| \times |\mathscr{A}_{\text{spts}}|$, comprising of the staging and setpoint actions (Table 4.1).

However, as discussed before, MARL suffers from two severe performance bottlenecks, the problem of non-stationarity and the problem of multi-agent credit assignment. In this case, the non-stationarity implies that the staging agent should learn the optimal staging sequence while also anticipating and adapting to the learning behavior of the setpoint agent (similarly for the setpoint agent). While it is obvious that the total power consumption is a function of the actions of both agents, the challenge here is to distinguish the individual contributions so that agents receive accurate feedback which in turn leads to better learning.

Although several MARL strategies exist that counter these problems, the following section presents two novel strategies in the context of HVAC control to alleviate the fundamental problems discussed by exploiting the problem structure. The problem of non-stationarity is addressed by formulating a turn-based game for HVAC control to facilitate agents learning the consequence of their own actions and environment dynamics as well as adapting to the actions of the other agents. The problem of credit multi-agent assignment is addressed by formulating a custom parametric reward mixing strategy by exploiting the structure of the total reward for the problem. The subsequent sections present the mathematical arguments underpinning the proposed approaches and their implications. Experimental benchmarks include the learning performance of popular MARL strategies CTDE (central training and decentralized execution) and value decomposition (sum mixing and Q-mixing) methods. The experimental results reveal that the combination of the two proposed approaches does improve the learning properties in the case of MARL.

MARL Problem Formulation

Let $\mathscr{A}_{\text{stag}}, \mathscr{A}_{\text{spts}}$ be the domain of the actions for the two agents, comprising of staging control and setpoint control. As there are no restrictions on the observations, both the agents share the same global state $\mathscr{S}$ which encapsulates all the states listed in Table 4.1. Let R be the global reward and $R_{\text{stag}}, R_{\text{spts}}$ be the individual rewards for the agents. The goal is then to learn a joint policy $\pi^* = \{\pi^*_{\text{stag}}, \pi^*_{\text{spts}}\}$ that would maximize the cumulative rewards.

It is obvious that the global reward is inversly proportional to the total power consumption of the chiller plant (as per the reward formulation in Equation 4.4). The challenge now lies in decomposing the global reward fairly between the two agents, i.e. splitting the reward proportionately that more accurately describes the consequence of the agents' actions. This is challenging as the objectives of the two agents are tangled and interdependent. Although the setpoint agent is primarily concerned with tweaking the setpoints of the chillers, it can also have an impact on the other equipment, as suboptimal chiller setpoints may result in more work for the cooling towers and the pumps, in other words, the agents' rewards overlap to some degree. Thus one of the objectives of MARL is to figure out the optimal reward contributions so that the individual agent rewards have a positive effect on the global reward. The centralized learning and decentralized execution (CTDE) circumvents this by learning a joint value function, where the agents manifest their rewards and actions, and a centralized critic would then optimize the TD loss. In the value decomposition approach on the other hand, the agents have local critics and the cost is aggregated by a mixer, which would combine the localized critic estimates as in Equation 4.5, where Q_{tot} represents the total value and θ_{mix} represents parameterization for the mixing function.

$$Q_{\text{tot}} \sim f_{\theta_{\text{mix}}}(Q_{\text{stag}}, Q_{\text{spts}}) \tag{4.5}$$

The following sections present two strategies, LINMIX and turn-based games. LINMIX attempts to learn a better parameterization of the mixing function, while turn-based game attempts to alleviate non-stationarity encountered by agent-agent interactions.

LINMIX - Parametric Linear Mixing

LINMIX is a custom linear parametric mixing strategy inspired by value decomposition network (VDN) [Sunehag et al., 2017] and QMIX [Rashid et al., 2018]. It should be noted that QMIX is a superset of the proposed approach and can represent a much richer class of non-linear mixing functions. The proposed approach on the other exploits the reward structure to compose linear parametrization for $f_{\theta_{\text{mix}}}$, instead of running a non-linear mixing strategy when it is evident that the global reward is linear in the given problem. The proposed strategy does have some subtle differences compared to QMIX (which will be discussed subsequently).

Value decomposition networks assume the rewards to be additive as in Equation 4.6. The TD error is then based on the aggregated value function Q_{tot} which influences the policy loss. This implies that the agents get feedback from other agents via Q_{tot}. VDN (Equation 4.6) is a simple strategy that forces each agent to maximize the individual returns and, at the same time steers the agents so as to maximize the global reward. However, as pointed out before, reward decoupling in the case of staging and setpoint agents is not as quite straightforward. It is possible to simply reward both the agents based on the total power consumption, i.e. $R_{\text{stag}}, R_{\text{spts}} \sim f(P_{\text{total}})$, but it would not accurately capture the individual contributions of the agents.

$$Q_{\text{tot}} = \Sigma_{A_i} Q_{A_i}(s, a_{A_i}) \ , \ \forall A_i \in \text{agents} \tag{4.6}$$

QMIX is a strategy that proposes to alleviate this problem by fitting a mixing network (neural network), where $f_{\theta_{\text{mix}}} \sim f_{\theta_{\text{mix}}}(Q_1, \ldots, Q_{A_n}, s_t)$ (in Equation 4.5), $f_{\theta_{\text{mix}}}$ performs a non-linear mixing on the individual agent Qs. In a cooperative setting, the value function must satisfy the monotonicity property (Equation 4.7). The monotonicity property implies that a gain in local agent rewards has a positive effect on the global reward, i.e. the agent rewards are not competitive. And QMIX ensures monotonicity by restricting the weights of the neural network to be positive. While the non-linear function is certainly capable of representing a broad class of value functions, in the context of the given problem it might be redundant, adding to learning overhead. This is due to the fact that the total power consumption is additive due to the contributions of different equipments involved, which can further be decomposed as additive contributions of staging and setpoint control (Equation 4.8). Thus $f_{\theta_{\text{mix}}}$ can be viewed as a linear function in agent Qs.

$$\frac{\partial Q_{\text{tot}}}{\partial Q_{A_i}} \geq 0 \ , \ \forall A_i \in \text{agents} \tag{4.7}$$

$$\begin{aligned} P_{\text{total}} &= P_{\text{chillers}} + P_{\text{cooling towers}} + P_{\text{pumps}} \\ &= (P_{\text{chillers, stag}} + P_{\text{chillers, spts}}) \\ &+ (P_{\text{cooling towers, stag}} + P_{\text{cooling towers, spts}}) \\ &+ (P_{\text{pumps, stag}} + P_{\text{pumps, spts}}) \end{aligned} \tag{4.8}$$

Proposition 1. *Multi-agent VDN converges to single-agent reinforcement learning solution if the agent rewards are monotonic and independent, i.e. iff* $r_t = \Sigma_{A_i} r_{t,A_i} \ , \ \forall A_i \in agents$.

Proof for Proposition 1. The following proof derives the result using the cumulative policy returns under VDN. An alternate proof based on the recursive relation and independence assumptions also exists[Tang et al., 2023].

- Let $r_{A_1}, r_{A_i}, \ldots, r_{A_n}$ be the individual agent rewards.

- Let $g_{A_i} = \Sigma_{t=0} \gamma^t R_{A_i}(s_t, < \pi_{A_i}, \pi_{-A_i} > (s_t), s_{t+1} >)$ be the cumulative returns for agent A_i. Here $< \pi_{A_i}, \pi_{-A_i} >$ represents the joint policy, policy of agent A_i and the rest of the agents.
- Let $\pi = \{\pi_{A_1}, \pi_{A_2}, \ldots, \pi_{A_n}\}$ be the set of policies to be learnt.

Multi-agent reinforcement learning under VDNs would involve optimizing the policies to maximize the sum of agent rewards,

$$\max_{\pi=\{\pi_{A_1},\pi_{A_2},\ldots,\pi_{A_n}\}} \mathbb{E}_{s_0}\left[g_{A_1} + g_{A_2} + \ldots g_{A_n}\right]$$
$$\max_{\pi=\{\pi_{A_1},\pi_{A_2},\ldots,\pi_{A_n}\}} \mathbb{E}_{s_0}\left[\Sigma_{A_i}\Sigma_{t=0}\gamma^t R_{A_i}(s_t, < \pi_{A_i}, \pi_{-A_i} > (s_t), s_{t+1} >)\right]$$
$$\Sigma_{A_i} \max_{\pi_{A_i}} \mathbb{E}_{s_0}\left[\Sigma_{t=0}\gamma^t R_{A_i}(s_t, < \pi_{A_i}, \pi_{-A_i} > (s_t), s_{t+1} >)\right]$$

Since the agent rewards are independent, each agent maximizing its reward should converge, while other agents ($\tilde{\pi}_{-A_i}$) should have no impact on agent π_{A_i} and reward R_{-A_i},

$$\Sigma_{A_i}\mathbb{E}_{s_0}\left[\Sigma_{t=0}\gamma^t R^*_{A_i}(s_t, < \pi^*_{A_i}, \tilde{\pi}_{-A_i} > (s_t), s_{t+1} >)\right]$$

We can then use the fact that $\Sigma_{A_i} R^*_{A_i} = R^*$ and $\bigcup_{A_i} < \pi^*_{A_i}, \tilde{\pi}_{-A_i} >= \pi^*$,

$$\mathbb{E}_{s_0}\left[\Sigma_{t=0}\gamma^t R^*(s_t, \pi^*(s_t), s_{t+1} >)\right]$$
$$\Leftrightarrow \max_{\pi} \mathbb{E}_{s_0}\left[\Sigma_{t=0}\gamma^t R(s_t, \pi(s_t), s_{t+1} >)\right]$$

Thus, we have converted the MARL objective into a single-agent RL objective function, implying that under the assumptions made, the MARL solution should converge to the single-agent RL solution. Of course, the advantage would then be the sampling efficiency.

The proof for proposition 1 hinges upon the independence assumption of the agent rewards. However, as discussed before, $R_{\text{stag}}, R_{\text{spts}}$ are not completely independent. But the idea here is to extend Proposition 1 by introducing state-dependent non-linear multiplies that would decompose the rewards, by enforcing a constraint that the reward coefficients sum up to 1. This assumption is valid as the total power consumption is conservative, i.e. the total power consumption can be perfectly decomposed into contributions from individual equipments in an additive manner. Equation 4.9 formulates the linear mixing strategy (LINMIX). The coefficients ω_{A_i} are the outputs of a shallow neural network as a function of the current state. The monotonicity property can be easily satisfied by choosing the softmax activation

function (Equation 4.10). It is worth noting that although the overall function is linear in terms of Q_{A_i}, the coefficients ω_{A_i} themselves can be non-linear functions of the current state.

$$\begin{aligned} Q_{\text{tot}} &= \Sigma_{A_i} \omega_{A_i}(s_t, a_t) Q_{A_i}(s, a_{A_i}) \\ s.t.\ \Sigma_{A_i} \omega_{A_i}(s_t) &= 1\ ,\ \forall A_i \in \text{agents} \end{aligned} \tag{4.9}$$

$$\frac{\partial Q_{\text{tot}}}{\partial Q_{A_i}} = \omega_{A_i} = \frac{e^{z_{A_i}}}{\Sigma_{A_j} e^{z_{A_j}}} \geq 0 \tag{4.10}$$

Thus, if the $\omega_{A_i}(s_t, a_t)Q_{A_i}(s, a_{A_i})$'s are independent, then Proposition 1 should hold and we should converge to the same solution as the single-agent RL.

Turn-based Games

A turn-based game involves the agents playing in turns during the learning process. In each turn, one of the agents would have the liberty to freely choose an action (as per its current policy) while the other agents would be restricted to their previous action. Recalling the non-stationarity with MARL, each agent has to learn the consequences of its own actions, and learn to distinguish the dynamics from the behavior of other agents. The core problem is that all of these feedback signals are squeezed into $\Sigma_{s_{t+1}} P(s_{t+1}|s_t, < \pi_{A_i}, \pi_{-A_i} > (s_t))$ which the TD learning algorithm has to then learn to approximate. Thus by formulation of a turn-based game, the agents see the transition as $\Sigma_{s_{t+1}} P(s_{t+1}|s_t, < \pi_{A_i}, \bar{\pi}_{-A_i} > (s_t))$ ($\bar{\pi}$ indicates that the agent retaining the previous action), reducing the variability in the TD target (making it analogous to single-agent RL task). It is also worth noting that the argument of turn-based games can be taken a step further, where the agent policies may be updated in turns, and in each cycle, a subset of agents act according to their previous policy. The following section draws the implications of turn-based games on the fundamental building blocks of RL, value iteration, and policy improvement. The latter approach of updating agent policies in turns, however, could not be tested due to time constraints.

Reinforcement learning is essentially an unsupervised way to solve Bellman's equation (Equation 2.3). Temporal difference learning enables this by having the agent learn by interaction, circumventing $P(s_{t+1}|s_t, a_t)$ in the Bellman equation. The equation is approximated as $V(s_t) \leftarrow \alpha V(s_t) + (1-\alpha)\max_{a_t}[R(s_t, a_t) + \gamma V(s_{t+1})]$ ([Sutton, 1988]), where α is the learning rate. As the agent learns to play optimally by repeated interaction, the TD update should eventually converge and we would then derive the optimal policy using the value function.

The policy improvement step involves looking up the greedy action that maximizes the value function at the current state (Equation 4.11), where $< a_{A_i}, a_{-A_i} >$

represents the joint action of the agents. But in a turn-based game, the agents act in turns, meaning while one of the agents takes an action, the rest of the agents keep their actions fixed. Thus the lookup for the policy improvement step involves $\max_{<a_{A_i}>}$ instead of $\max_{<a_{A_i},a_{-A_i}>}$ (Equation 4.12). This implies that at each TD update step, the critic has to approximate $P(s_{t+1}|s_t, < a_{A_i}, \bar{a}_{-A_i} >)$ instead of $P(s_{t+1}|s_t, < a_{A_i}, a_{-A_i} >)$, thus reducing the number of factors influencing the dynamics. As a consequence of agents restricting their actions, agent $-A_i$ gets to observe the effect of agent A_i, while agent A_i gets to observe the effect of its own actions.

$$\pi : \max_{<a_{A_i},a_{-A_i}>} \Sigma_{s_{t+1}} P(s_{t+1}|s_t, < a_{A_i}, a_{-A_i} >) \left[R(s_t, < a_{A_i}, a_{-A_i} >, s_{t+1}) + \gamma V(s_{t+1})\right] \tag{4.11}$$

$$\pi : \max_{<a_{A_i}>} \Sigma_{s_{t+1}} P(s_{t+1}|s_t, < a_{A_i}, \bar{a}_{-A_i} >) \left[R(s_t, < a_{A_i}, \bar{a}_{-A_i} >, s_{t+1}) + \gamma V(s_{t+1})\right] \tag{4.12}$$

Value iteration on the other hand benefits from reduced computation (Equations 4.13 - 4.15), where the summation $\Sigma_{<a_{A_i},a_{-A_i}>}$ can be simplified to $\Sigma_{<a_{A_i}>}$. As the learning progresses, turn-based execution should yield higher rewards when at least one of the agents starts behaving optimally (Equation 4.16), thus, steering the overall learning.

$$\begin{aligned} V(s_t) &= \Sigma_{<a_{A_i},a_{-A_i}>} \pi(< a_{A_i}, a_{-A_i} > |s_t) \Sigma_{s_{t+1}} P(s_{t+1}|s_t, < a_{A_i}, a_{-A_i} >) \\ &\quad \left[R(s_t, < a_{A_i}, a_{-A_i} >, s_{t+1}) + \gamma V(s_{t+1})\right] \qquad (4.13) \\ &= \Sigma_{<a_{A_i},a_{-A_i}>} \pi(< a_{A_i} > |s_t) \pi(< a_{-A_i} > |s_t) \Sigma_{s_{t+1}} P(s_{t+1}|s_t, < a_{A_i}, a_{-A_i} >) \\ &\quad \left[R(s_t, < a_{A_i}, a_{-A_i} >, s_{t+1}) + \gamma V(s_{t+1})\right] \qquad (4.14) \\ &= \Sigma_{<a_{A_i}>} \pi(< a_{A_i} > |s_t) \pi(< \bar{a}_{-A_i} > |s_t) \Sigma_{s_{t+1}} P(s_{t+1}|s_t, < a_{A_i}, \bar{a}_{-A_i} >) \\ &\quad \left[R(s_t, < a_{A_i}, \bar{a}_{-A_i} >, s_{t+1}) + \gamma V(s_{t+1})\right] \qquad (4.15) \end{aligned}$$

$$\Sigma_t^{t+1} R(s_t, < a_{A_i}, \bar{a}^*_{-A_i} >, s_{t+1}) \geq \Sigma_t^{t+1} R(s_t, < a_{A_i}, a_{-A_i} >, s_{t+1}) \tag{4.16}$$

5

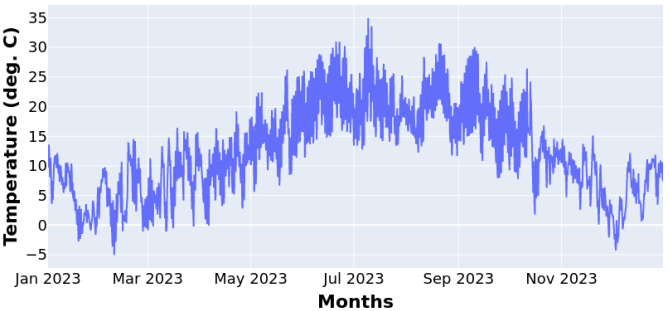
Experiments

Apart from the specification and representation of state/action spaces and the reward function (Table 4.1), it is imperative to design fair learning experiments to be able to efficiently learn a useful policy in RL. The learning experiments for RL should be representative of all the potential real-world conditions that may be encountered, but at the same time, the environment should not be too overwhelming which would result in the agent not learning at all. Fairness here implies fairness with respect to the episodic rewards. Designing fair episodes can be a crucial aspect of reinforcement learning, as PPO (or any on-policy algorithm) has a tendency to get biased toward the training data. Thus, it is imperative to avoid "sorting" the data (weather and load patterns), which might otherwise lead to the infamous "catastrophic forgetting" problem, which is also persistent in reinforcement learning (as the policies and value functions themselves are neural networks) [Zhang et al., 2023]. In our case, the episodic rewards are greatly influenced by two main factors, the reference cooling load to track and external weather conditions (temperature and humidity), and having inconsistent episodes would often result in the easier episodes dominating and the agent being biased. Both the tracking load and weather data can have a wide range and distinctive properties during everyday operations (refer Table 1.1 for respective ranges). However, it is nearly impossible to represent the full range during the training phase, and doing so might be redundant and inefficient, this is because the variables $T_{\text{db}}, H, l_{\text{ref}}$ are continuous variables (Table 4.1) and it is incumbent to learn a relationship rather than exposing the agent to every possible combination of value in the range. This was also one of the motivating factors for the continuous state representation of these variables. Thus, the training experiments are designed by exploring the properties of these variables and incorporating the respective trends spanning the domain.

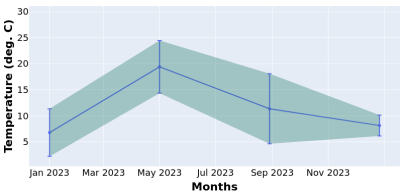
Although the cooling load for the data center has a continuous range of 500 kW to 2 MW, the load requirement often follows a discrete and stable pattern in the defined simulation setup. For instance, the requirement may be 75 % of the capacity during busy hours and a nominal 50 % rest of the time, with minor fluctuations over the hours. Thus, for the learning experiment the load profile is assumed to be piecewise constant with a step size of 300 kW, from 500 kW to 2 MW. Real-world

weather patterns on the other hand are more intricate in nature. Apart from considering the range, weather patterns often have seasonality and trends that need to be accounted for. It is also important to have realistic combinations of outdoor temperature and humidity, as the two are often correlated and intertwined. Figure 5.1a shows the weather profile for the city of Frankfurt for the year 2023. Aggregating the data over a period of four months reveals the seasonal patterns more clearly (Figures 5.1b - 5.1c). These trends are modeled as normal distributions by computing the means and the variances in the respective intervals. The weather conditions for the training phase are then set by randomly choosing a season and then sampling the corresponding distribution. This way, the complexity of the state space is reduced dramatically while still exposing the agent to the different properties and characteristics in the data, without inducing bias. Further, the episodes are made sufficiently long, and sufficient randomization is introduced to sample all possible combinations in every episode.

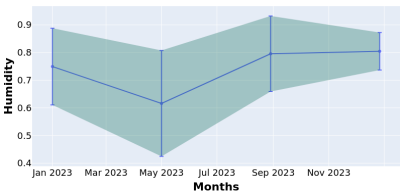
The RL agents were trained on Frankfurt weather conditions (Figure 5.1), but were also evaluated on other weather profiles (Florida, Figure A.1) in order to observe the ability of the agents to generalize. Florida weather was also included to have fair ground of comparison, this is because the closed-loop controller was optimized on Florida data.



(a) Temperature



(b) Temperature trends



(c) Humidity trends

Figure 5.1 Frankfurt weather

The overall framework is depicted in Figure 5.2. The process follows a typical

RL step, given a state, the agent chooses an action (staging and setpoint controls), executes the action, transits to a new state, and observes the reward. After having collected enough examples, a gradient step is executed to update the policy and the critic networks, and the whole process is repeated. The RL step involves the agent executing the action for a sufficiently long time, allowing the system to settle. This steady-state assumption simplifies the dynamics during training (although the evaluation is more dynamic, discussed later). This assumption motivates a choice for a lower discount factor (γ) for training, as once a steady state is reached, the power consumption is bound to remain constant as long as the agent holds its action. The implication is that the critic is myopic and should better be able to approximate the steady-state behavior. Other hyperparameters for training are listed in Table 5.1. The choice of an RL simulation step of 4 hrs is motivated by response analysis, as detailed in Section 3.4.

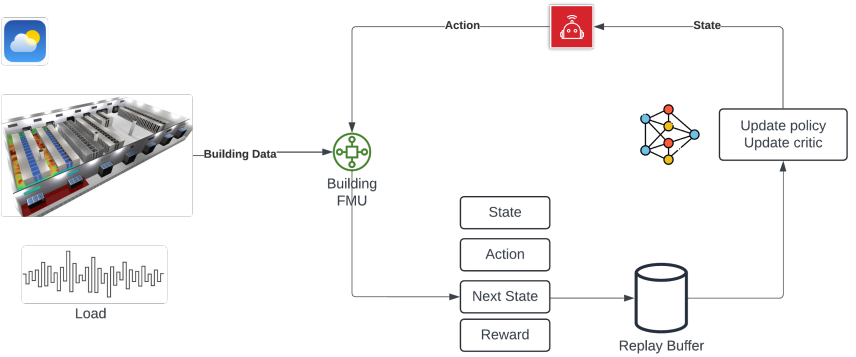
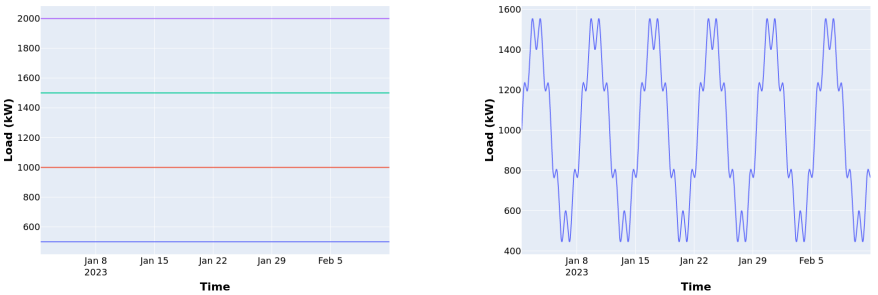


Figure 5.2 Reinforcement Learning Framework for HVAC Control

Hyperparameter	Value
Simulation step time	4 hrs
Discount factor	0.1
Load update frequency	3 RL steps
Weather update frequency	6 RL steps
Episode length	90 RL steps
Step size for load variation	300 KW

Table 5.1 Training Hyperparameters

For evaluation, on the other hand, real-world weather conditions and load patterns are used as is and the agent is exposed to live data. The weather profile is simply the raw hourly data (Figure 5.1a) and the load profile consists of possible real-world patterns, constant loads, and load patterns with daily and weekly



(a) Constant loads **(b)** Daily and weekly load trends

Figure 5.3 Test-bench: load profiles

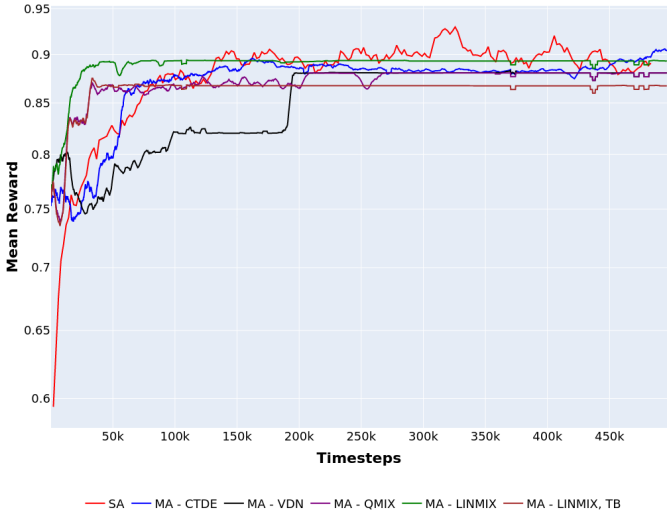
trends. Figure 5.3a shows the constant load profile where the load remains constant throughout the year and Figure 5.3b is a load pattern with both daily and weekly trends. The trends were generated by imposing sinusoids of different time periods, a slower sinusoid of period 24×7 hours to emulate weekly variations, and a faster sinusoid of period 24 hours to represent daily fluctuations. To further analyze the behavior of the reinforcement learning agent, the profiles are dissected by zooming in on specific weeks of the year to monitor the controls and discern whether the reinforcement learning controls make sense.

5.1 Results

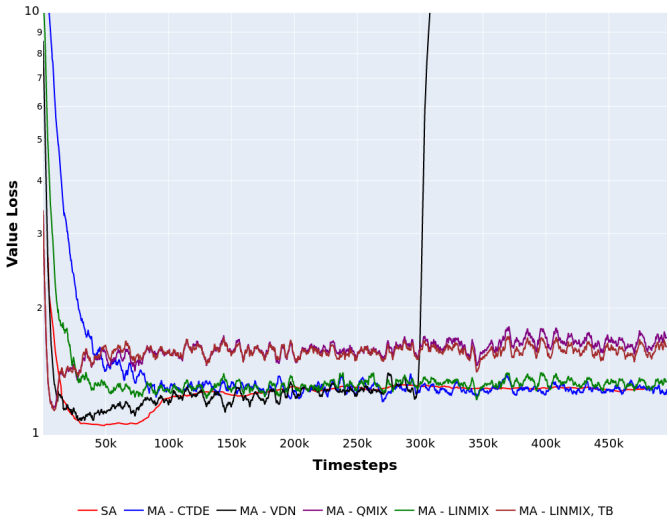
The metrics for evaluation primarily include the total power consumption of RL control benchmarked against the closed-loop controller under different operating conditions, at the same time ensuring that the IT room temperature setpoint is met. The closed-loop controller runs supervisory sequence controllers and PI controllers for reference tracking and often serves as a benchmark. The closed-loop controller is made available by Carrier. Apart from the performance metrics, the results include the learning curves for RL that depict the learning progress of the agent and the convergence. The results are presented for all the approaches discussed thus far: single-agent RL and all of the existing MARL approaches - CTDE and value decomposition (sum and Q-mixing), and two of the proposed MARL enhancements: LINMIX and turn-based games.

Learning Curves

Figure 5.4 compares the learning progress (reward curve and critic error) of all the approaches RL tested. The reward curves (Figure 5.4a) are simply the evaluation of the algorithms as the learning progresses, and the value loss (Figure 5.4b) represents the critic loss (TD error) during training. Ideally, we would like the reinforcement



(a) Normalized rewards



(b) Value loss (TD error)

Figure 5.4 Learning curves captured by fixing all stochastic parameters between runs **SA**: single-agent RL. **MA - CTDE**: multi-agent centralized learning and decentralized execution. **MA - VDN**: multi-agent sum-mixing. **MA - QMIX**: multi-agent QMIX. **MA - LINMIX**: multi-agent linear mixing (proposed). **MA - TB**: multi-agent QMIX and turn-based game (proposed)

learning agent to learn fast and at the same time achieve the highest possible reward. Convergence may be characterized as the point at which the rewards stabilize and the critic loss has reduced to its lowest value (ideally to zero). Right off the bat, the reward curves (Figure 5.4a) reveal an interesting property that relates back to the single-agent vs multi-agent dilemma discussed initially: the tradeoff between faster learning and optimality in MARL. The MARL algorithms learn faster than the single-agent RL but have a tendency to converge to a suboptimal solution (with the exception of the CTDE approach). Faster convergence can be attributed to the fact that the agents explore their individual action spaces, leading to faster exploration, which in turn translates to the agents learning the consequences of the actions much faster, or in other words, faster learning. The convergence to a suboptimal solution is also a consequence of the very same property, as the agents learn the consequences of their own actions with minimal information sharing with others, there is a tendency to not explore the whole optimization space jointly and get stuck in a local minima.

This problem is not as prominent in the case of the CTDE approach, where a single critic models the joint action executed by all the agents, and as expected, the CTDE approach performs quite similarly to the single-agent RL approach.

In terms of value-decomposition approaches, MARL can benefit from decomposed action spaces by having to model smaller action spaces, but this also introduces an overhead in terms of the learning and coordination between the agents. For instance, in Figure 5.4b, the CTDE critic converges to the same value as the single-agent critic, while the QMIX-based approaches incur a higher bias, and the sum-mixing approach has a hard time converging. In terms of the performance of two of the proposed approaches, (LINMIX and turn-based games), we observe that the LINMIX approach learns faster than QMIX and achieves a higher expected reward (Figure 5.4a). A plausible reason could be the better parameterization of the mixing function in the context of the given problem, whereas the QMIX algorithm has to explore a wider class of non-linear function parametrization (which might be redundant in the context of the given problem). This is also evident in terms of the observed value loss (Figure 5.4b), where the proposed LINMIX approach settles down to a lower value error than the QMIX approach, implying a better parametrization of the mixing function. The proposed turn-based approach performed worse in terms of the expected rewards and achieved a critic loss on par with the QMIX approach. But it does appear to stabilize the training to result in a smoother convergence than QMIX (Figure 5.4a).

Although the total reward is linear in nature, the VDN approach (which simply attempts to maximize the sum of the agent rewards) appears to be the slowest learner (Figure 5.4a). It also displays a spurious behavior where the critic diverges abruptly and never manages to recover back (Figure 5.4b). The credit assignment problem could be a plausible reason, although the total reward is linear in nature, it may not be simply the sum of individual agent rewards.

Performance Evaluation

As mentioned before, the evaluation benchmarks the RL controllers against the closed-loop controller on two different weather profiles (Frankfurt and Florida) under identical load conditions. The Florida data is included in the metrics to have a fair ground for comparison, as the closed-loop controller is primarily optimized under Florida weather conditions. As all the RL agents are trained on Frankfurt weather data, this can be seen as an opportunity to discern the ability of the agents to generalize, as Florida temperatures are higher compared to Frankfurt (see Figures 5.1 & A.1). At the same time, this also implies that the savings reported on Florida data may be conservative, and there may have been higher savings if perhaps the RL models were exposed to similar conditions during the training phase.

Figures B.1a - B.1e in the appendix show the performance evaluation of the agents under constant and sinusoidal loads. The performance metrics include the total power consumption benchmarked against the closed-loop and the IT room temperature error. Firstly, it is clearly evident that all the RL agents perform considerably better than the closed-loop controller, albeit some of the MARL algorithms incur a higher power consumption under certain conditions. The saving potential appears to be much higher in colder weather conditions (Frankfurt) compared to warmer temperatures (Florida, Figures C.1a - C.1e). Secondly, the IT room temperature error $|T_{\text{ref},t} - T_{\text{room},t}|$ remains nominal (close to zero) in all the cases, implying the temperature requirements being met. Although all of the agents perform better than the closed-loop controller, the single-agent RL approach performs the best in the case of both constant and sinusoidal loads. This is in line with the intuition that MARL algorithms can only perform as best as single-agent RL but may display sampling efficiencies. It is also worth noting that QMIX in particular displays a spurious behavior in the case of sinusoidal loads (Figures B.1e & C.1e), despite the steady learning curve and convergence (Figure 5.4a).

It is further worth analyzing if all the RL agents have a consensus in terms of control, i.e. whether the leanings between the agents roughly resemble each other. This appears to be indeed the case. Figures B.2a and C.1e plot the staging controls executed by the RL agents under sinusoidal load and Figures B.2b and C.2b plots the setpoint control. Although the controls are not completely with agreement with each other, there a noticible similarity, indicating a consensus at a higher level. This corroborates the learnings of the agents and our beliefs of optimality in terms of controls.

On Frankfurt weather data, comparing the mean power savings (Figures B.1f - B.1g), we observe that the proposed LINMIX approach performs the best compared to the rest of the MARL algorithms (with a better sampling efficiency, as discussed in the previous section), including the CTDE approach. With the turn-based approach, the results are ambivalent, in some cases outperforming the QMIX approach (sinusoidal load and constant load at 1000 kW), while inferior in other cases. As noted before, the single-agent approach achieves the most savings con-

sistently. With respect to the evaluation under Florida weather conditions (Figures C.1f - C.1g), we observe that the MARL agents have a hard time generalizing. The single-agent RL approach outperforms the closed-loop controller achieving positive savings, but the savings potential drops as compared to Frankfurt weather. Part of this can be attributed to the dissimilar weather conditions between Frankfurt and Florida.

The saving potential tends to decrease with increasing loads (Figure B.1f), which is in line with the experiments carried out as part of the exploratory data analysis. Although all the agents (including the single-agent RL) display peculiarity at a constant load of 2000 kW, where the savings are negative, implying that RL agents fail to outperform the closed-loop controller or at least at best. A plausible reason could be the exclusion of loads higher than 2000 kW as part of training, impeding the policy to extrapolate the learnings around the limiting values of the state.

Further dissecting and analyzing the controls, Figure B.3 contrasts the RL controls with the closed-loop controller. The wetbulb temperature fluctuates around 13 °C B.3a) and the operating load varies around 800 kW (Figure B.3b). Under these chosen conditions, the savings appear to be remarkable, nearly 25% (Figure B.3c). The closed-loop controller prefers to have one chiller ON, with the chiller differential setpoint ($T_{\text{chwsdt\_spt}}$) set to zero, while the RL controller run both chillers with a nominal differential setpoint of 2.5 °C. Furthermore, the RL controllers run a lower condenser evaporating temperature ($T_{\text{cond\_ewt\_spt}}$), which might be more efficient given that the outdoor temperature is quite low.

Figure C.3 compares the controls on a warm day in Florida with the wetbulb temperature of around 23 °C. The savings appear to marginal in this case, around 7 % (Figure B.3c). The RL agents again prefer a nomial chiller differential setpoint of 2.5 °C, but now prefer a higher condenser evaporating temperature. Figures B.4 - B.5 and C.4 - C.5 compare the results under similar external conditions.

6

Summary

The main focus of the work carried out was to manifest the saving potential in contemporary methods used for HVAC building control, and thus identify the opportunity and scope for a data-driven optimization technique such as reinforcement learning. The problem of optimizing power consumption in HVAC systems was modeled as a Markov decision process due to the sequential nature of the problem, with immediate controls having both short and long-term consequences. Reinforcement learning was chosen to solve the dynamic programming problem due to the complexities involved and the limitations surrounding the classical DP approaches. The extensive exploratory data analysis carried out also served as a motivation for identifying a reduced optimization landscape for reinforcement learning in formulating tangible problems.

The reinforcement learning problem formulation was extensively motivated, particularly highlighting the pros and cons of the compromises made surrounding the representation of the system states and actions. A naive reward function to optimize the total power consumption of the system was proposed but was soon evolved to also address practical concerns like action switching penalties and temperature violation errors. Further alleviated the learning process in RL by formulating a non-linear reward function to lay more emphasis on power savings than the expenditure. The training experiments were formulated to incorporate potential real-world encounters, such as yearly weather and load patterns. At the same time, with the caution that the learning was not "episodic", which would induce bias and unfairness, which would in turn result in some of the episodes dominating the learning process.

The optimization problem was first tackled using single-agent RL before diving into multi-agent RL. In the case of MARL, two fundamentally different strategies were compared, the popular centralized training and decentralized execution approach, which trades off the level of decentralization for learning efficiency, and value-decomposition which prioritizes decentralized learning. The pros and cons of the approaches were discussed, along with highlighting some of the key challenges in MARL. This motivated the proposal of two novel MARL enhancements in the context of HVAC optimization. The proposed LINMIX strategy attempted to exploit the reward structure given the fact that the overall power consumption is a

physical quantity and should thus always be conserved. Thus, the LINMIX strategy attempted to perform a linear mixing of the agents' critics. To alleviate the problem of non-stationarity, turn-based games were proposed, where the two agents would play in turns so as to reduce the variability in the dynamics and enable agents to learn to distinguish the consequences of their own action vs the behavior of other agents.

The evaluation included benchmarking both the single-agent RL and MARL approaches against the closed-loop controller on real-world weather patterns and load profiles. The experimental results revealed that RL optimization certainly had a scope for saving power. All of the RL approaches outperformed the closed-loop controller, albeit with different potential margins. The observed results also corroborated the single-agent vs multi-agent dilemma of optimality vs learning efficiency, the single-agent RL achieved the best overall results while MARL approaches learned faster but had a tendency to converge to a suboptimal solution. Given the results observed, the single-agent RL appears to be a winner in terms of performance. Given that the scale of the problem was fairly small (comprising of 6 actions and only two agents), MARL may not be the best approach in this situation. Perhaps, there would have been more significant differences in case of complex problems involving larger action spaces with the scope of having multiple agents. For a problem on a smaller scale, the MARL challenges impede the learning process, outweighing the benefits.

6.1 Future Work

Translating algorithmic learning to the real world is often challenging. The primary concern, especially in the case of RL, is safety. Although the proposed work included both hard and soft constraints (as indicated in Table 1.1) to honor the constraints with respect to the physical controls. There may be more elaborate rules that apply and may be imperative to adhere to. In such cases, in addition to introducing an action penalty, it is necessary to have a "post-processing" step that filters out the invalid actions given the current state, so that the agent does not abuse the control by probing the system at will. Another, perhaps more robust approach would be to pursue an RL approach modeled on CMDPs (constrained MPDs), however, the implementations of such approaches have not matured. Of course, it is also worth exploring more robust optimization techniques such as MPC or robust MPC, but MPC methods often require a very accurate or decently accurate model of the system and the online computation associated with MPC may also be a bottleneck.

Real-world processes often tend to deviate in real-time operation, which in turn implies that the control applied must be adaptive. In such cases, a model-based RL approach can be applied, where an approximate model of the system can be updated periodically (as and when sufficient data is accumulated) to improve its accuracy. The purpose of the model can be two fold, firstly, the RL policy can be

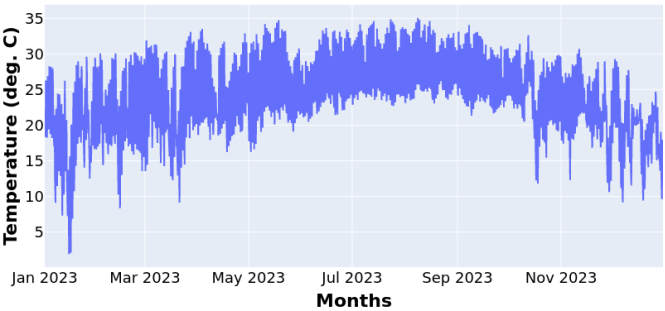
updated given the new model (incremental training), and secondly, the model can serve towards online planning, techniques such as multi-step lookahead and rollout optimization may be performed with value function approximation using the base RL policy. Literature suggests that such techniques have better capabilities of adapting to changing dynamics and other model perturbations, thus it is a viable option to explore.

Lastly, regarding RL sampling efficiency, as alluded to previously, one of the avenues that is worth exploring is imitation learning, where the knowledge from a mediocre controller can be transferred to accelerate the RL learning processes. For instance, the existing closed-loop controller can be used to impart optimal or suboptimal behavior under known external conditions and the RL agent can then be trained to generalize the learnings to other conditions.

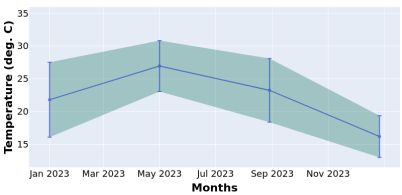
Furthermore, another dimension to the optimization problem is to exploit tuning the underlying PID and sequence controllers, which would further enhance the performance. For instance, the RL optimization algorithm can be used to optimize the PID parameters, based on which the agent would then act upon. Again, the agent can be initialized with the existing/suboptimal control parameters to accelerate the training.

A

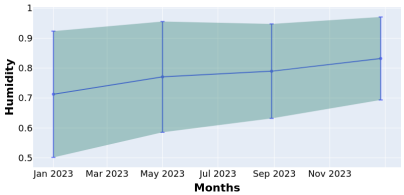
Weather Profiles



(a) Temperature



(b) Temperature trends



(c) Humidity trends

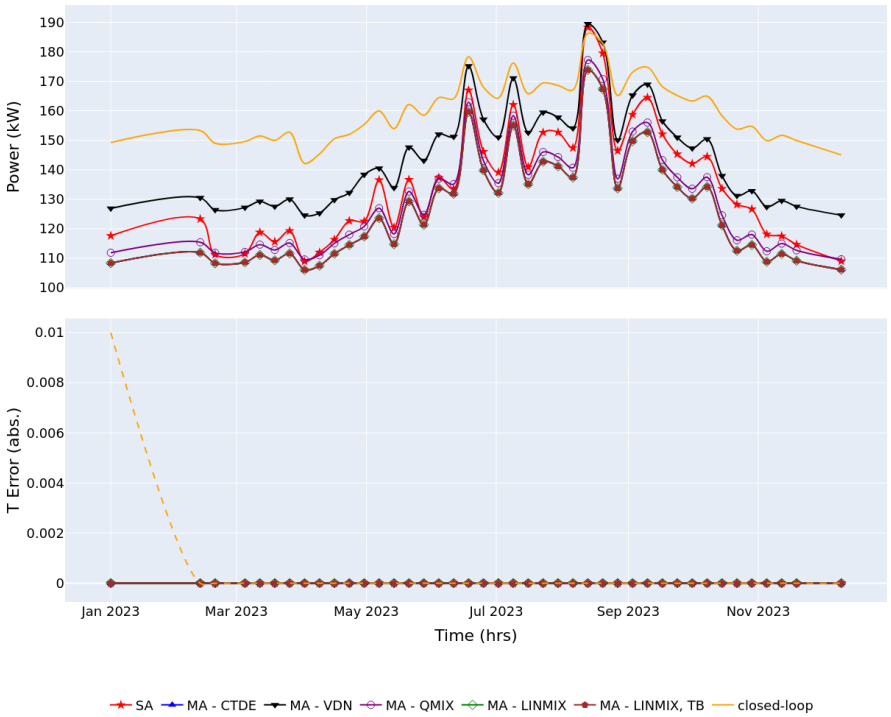
Figure A.1 Florida weather

B

Results - Frankfurt

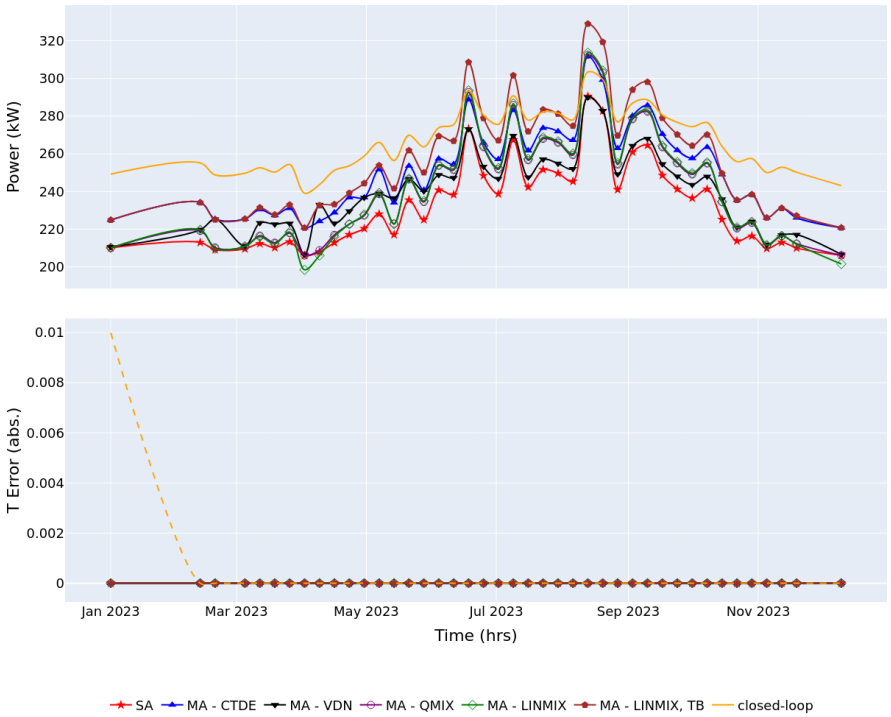
(a) Load 500 kW

Figure B.1 Performance metrics under constant load



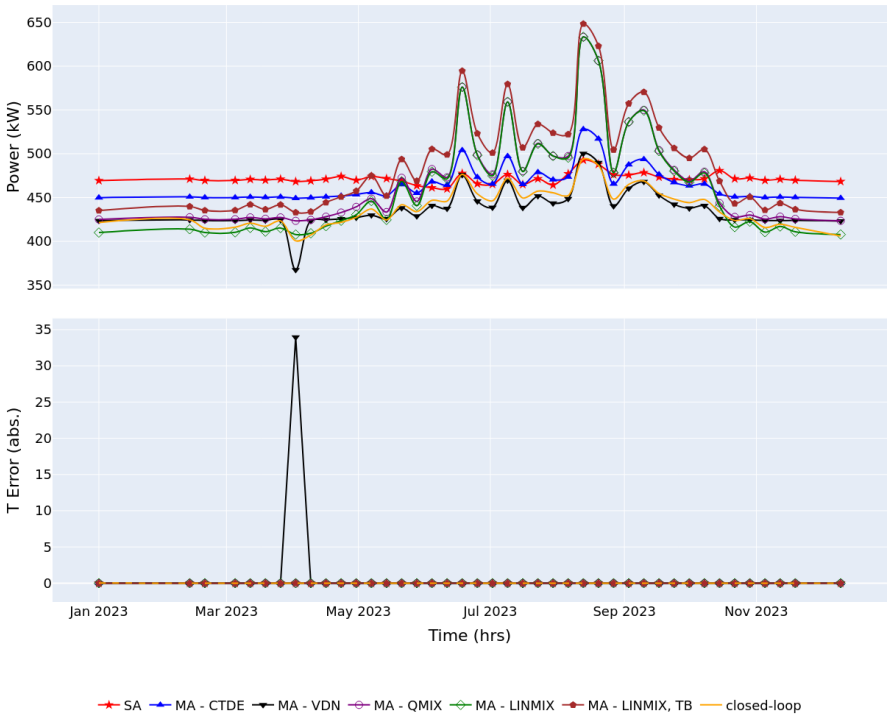
(b) Load 1500 kW

Figure B.1 Performance metrics under constant load



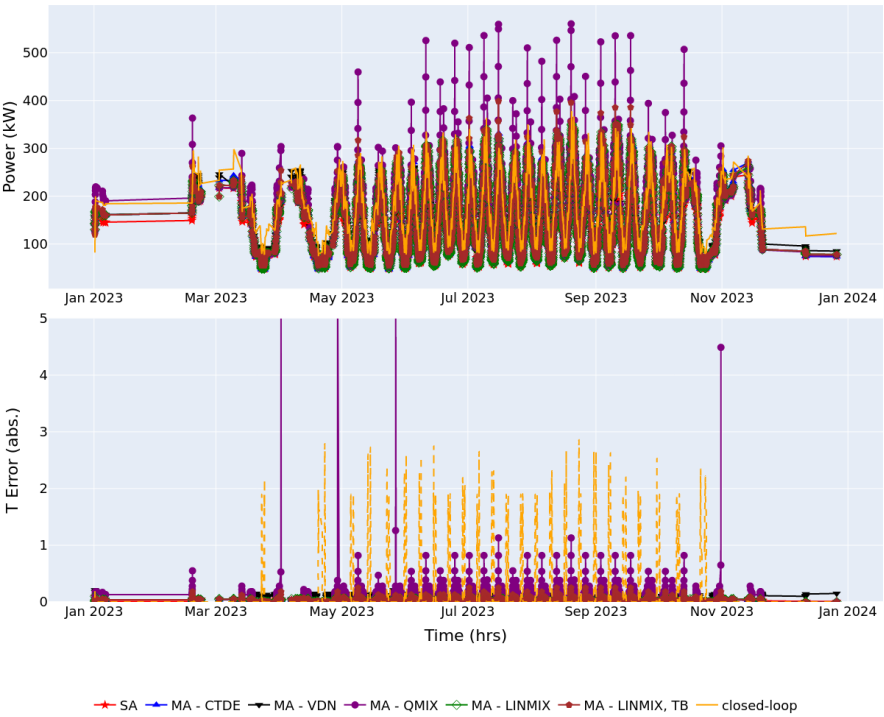
(c) Load 1500 kW

Figure B.1 Performance metrics under constant load



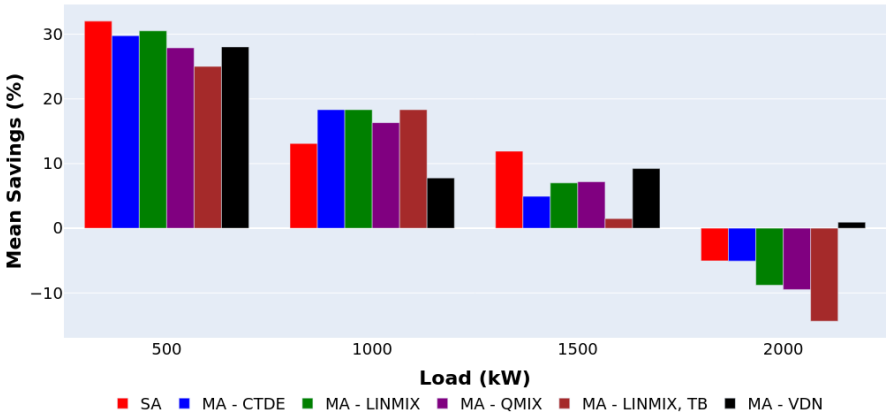
(d) Load 2000 kW

Figure B.1 Performance metrics under constant load

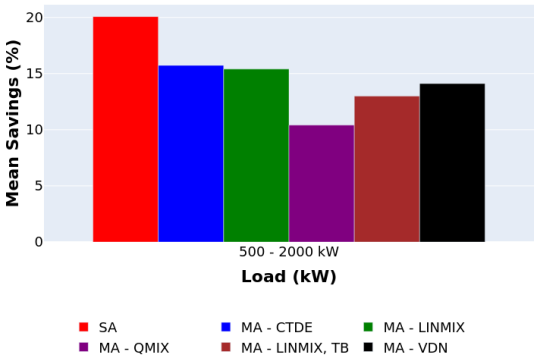


(e) Sinusoidal load, range 500 - 2000 kW

Figure B.1 Performance metrics under sinusoidal load
SA: single-agent RL. **MA - CTDE**: multi-agent centralized learning and decentralized execution. **MA - VDN**: multi-agent sum-mixing. **MA - QMIX**: multi-agent QMIX. **MA - LINMIX**: multi-agent linear mixing (proposed). **MA - TB**: multi-agent QMIX and turn-based game (proposed). **CL**: closed-loop controller

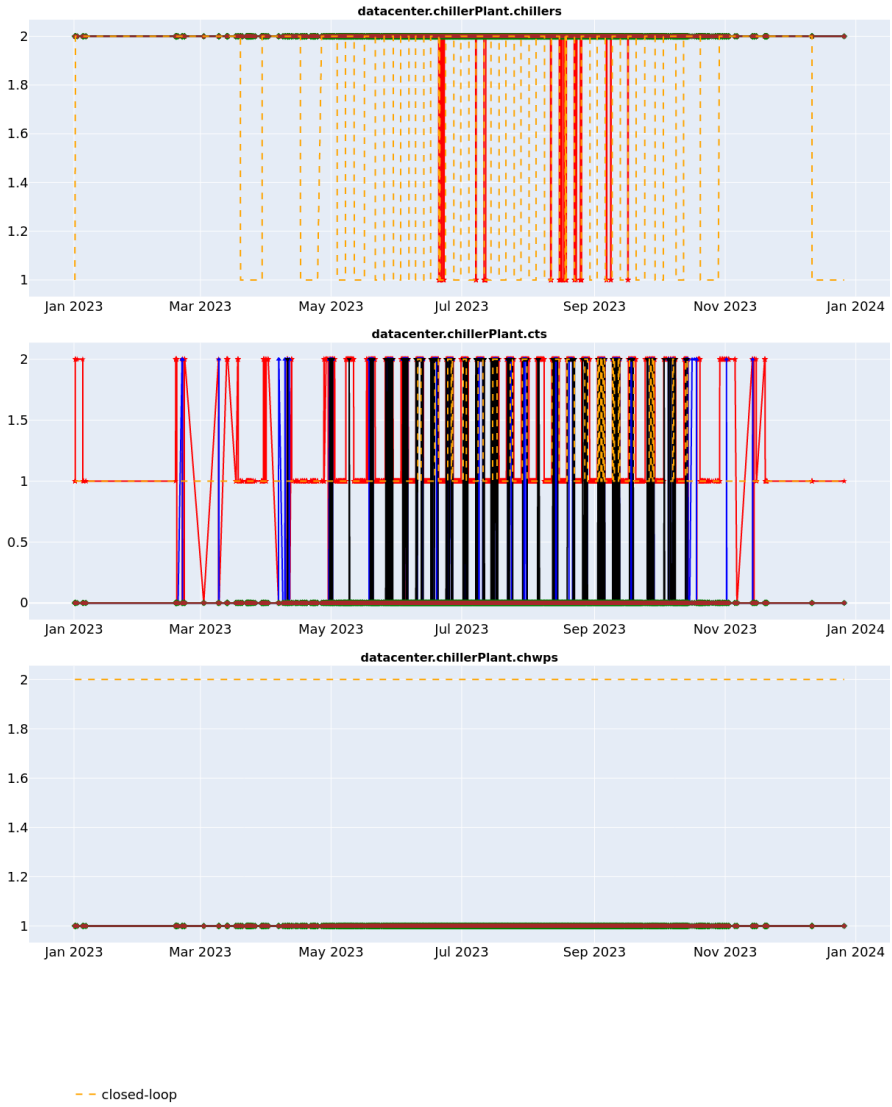


(f) Constant load



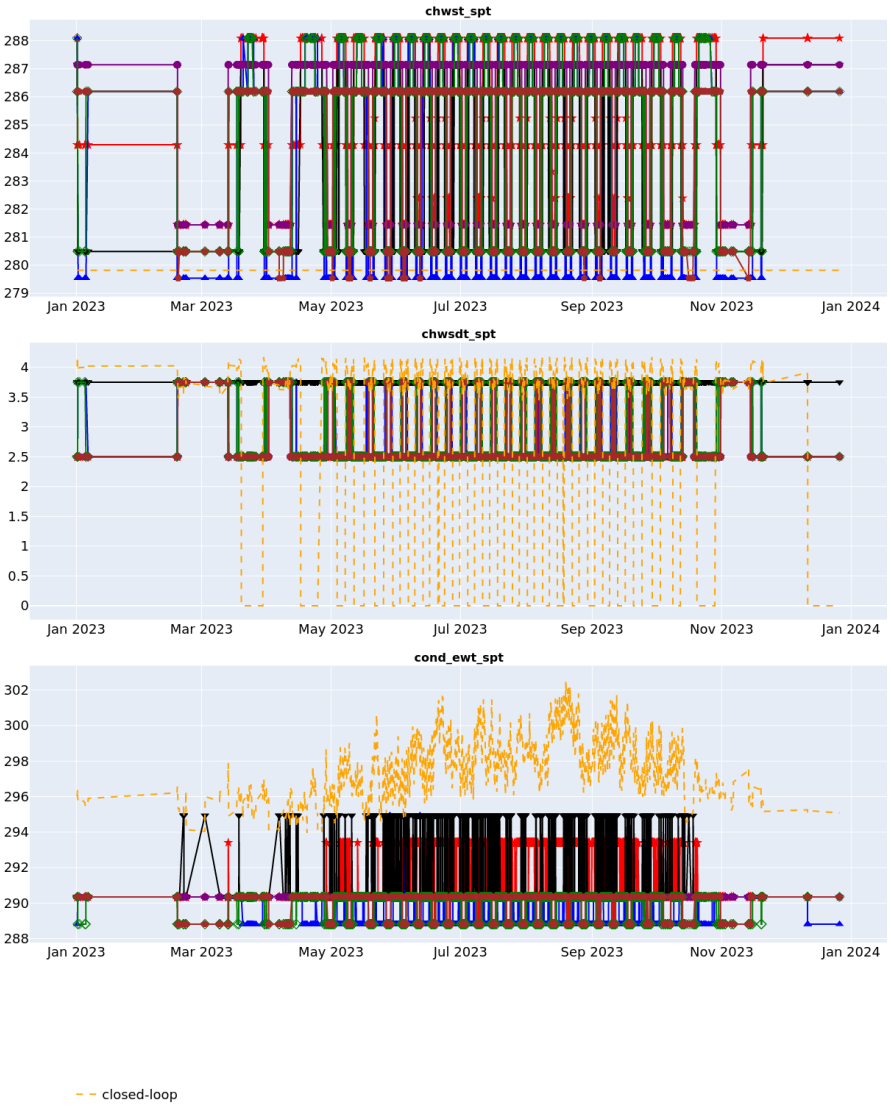
(g) Sinusoidal load

Figure B.1 Average yearly savings (relative to closed-loop controller). All RL agents appear to a positive saving, with the constant load of 2000 kW being an outlier.
SA: single-agent RL. **MA - CTDE**: multi-agent centralized learning and decentralized execution. **MA - VDN**: multi-agent sum-mixing. **MA - QMIX**: multi-agent QMIX. **MA - LINMIX**: multi-agent linear mixing (proposed). **MA - TB**: multi-agent QMIX and turn-based game (proposed). **CL**: closed-loop controller



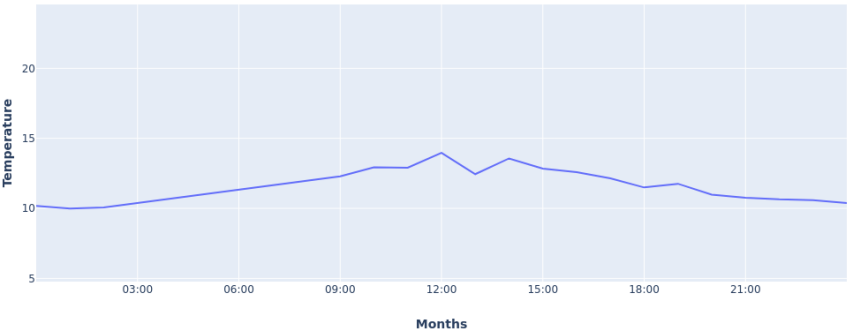
(a) Staging Control

Figure B.2 Control consensus, all RL agents seem to prefer a roughly similar control strategy.

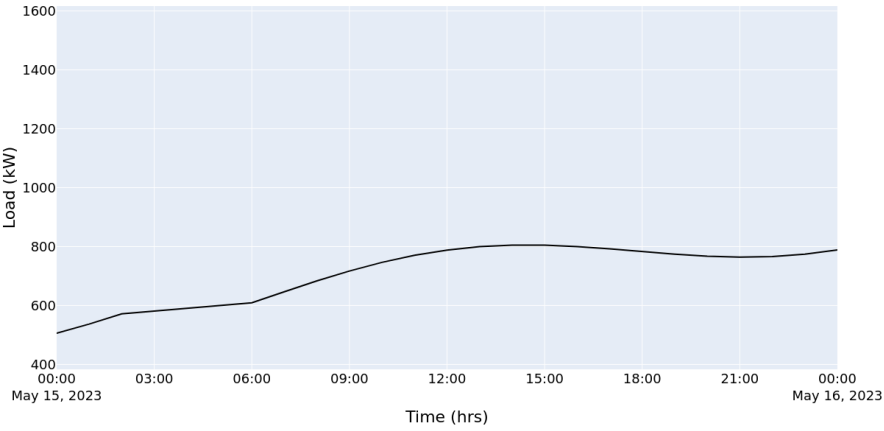


(b) Setpoint Control

Figure B.2 Control consensus: all RL agents seem to prefer a roughly similar control strategy, although there are a few discrepancies.
SA: single-agent RL. **MA - CTDE**: multi-agent centralized learning and decentralized execution. **MA - VDN**: multi-agent sum-mixing. **MA - QMIX**: multi-agent QMIX. **MA - LINMIX**: multi-agent linear mixing (proposed). **MA - TB**: multi-agent QMIX and turn-based game (proposed). **CL**: closed-loop controller

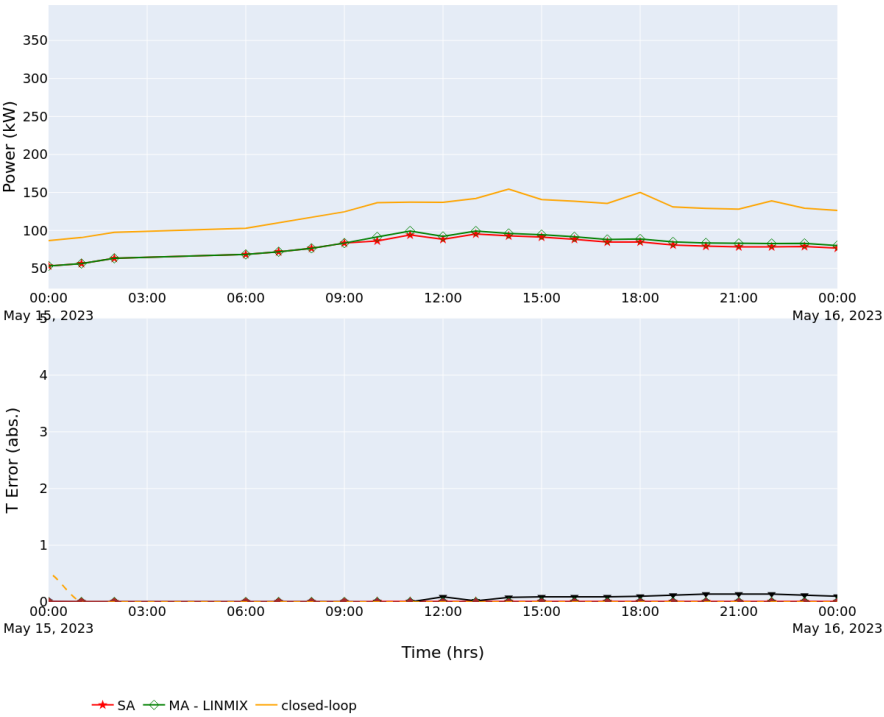


(a) Wetbulb temperature



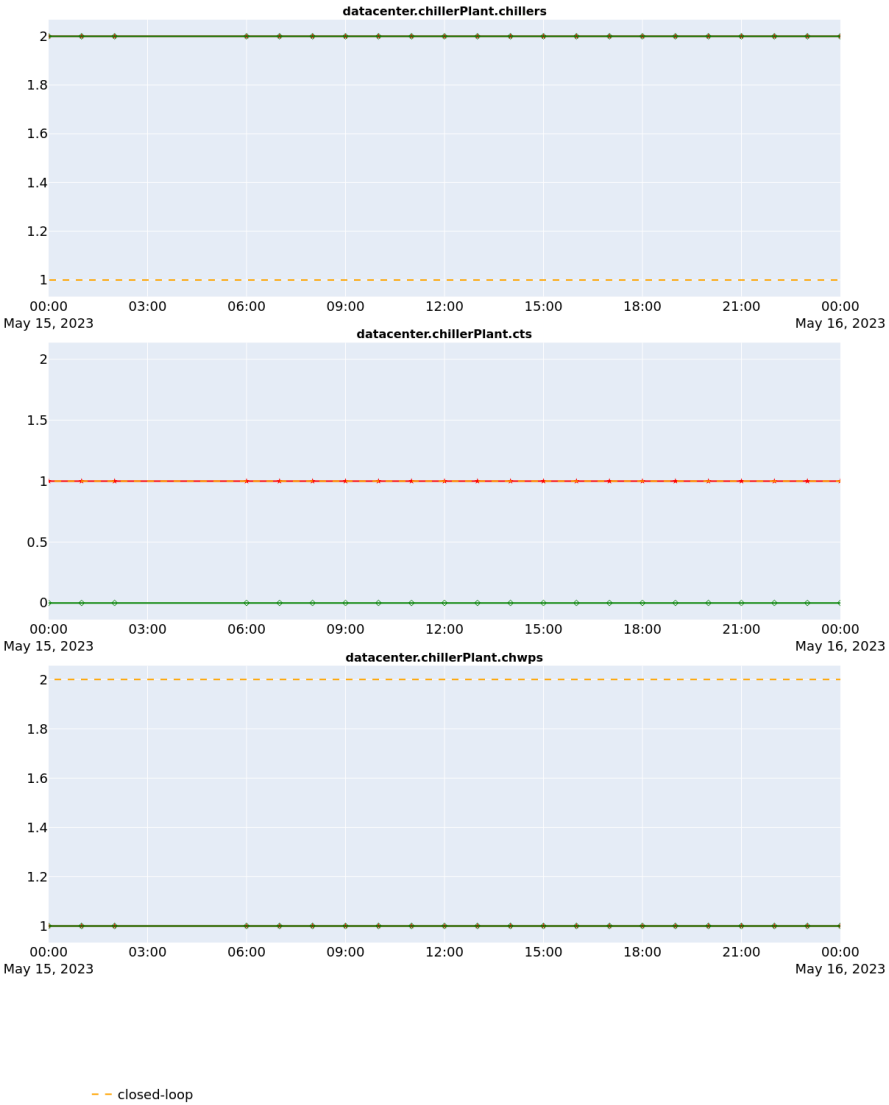
(b) Load

Figure B.3 Operating conditions for a day in spring



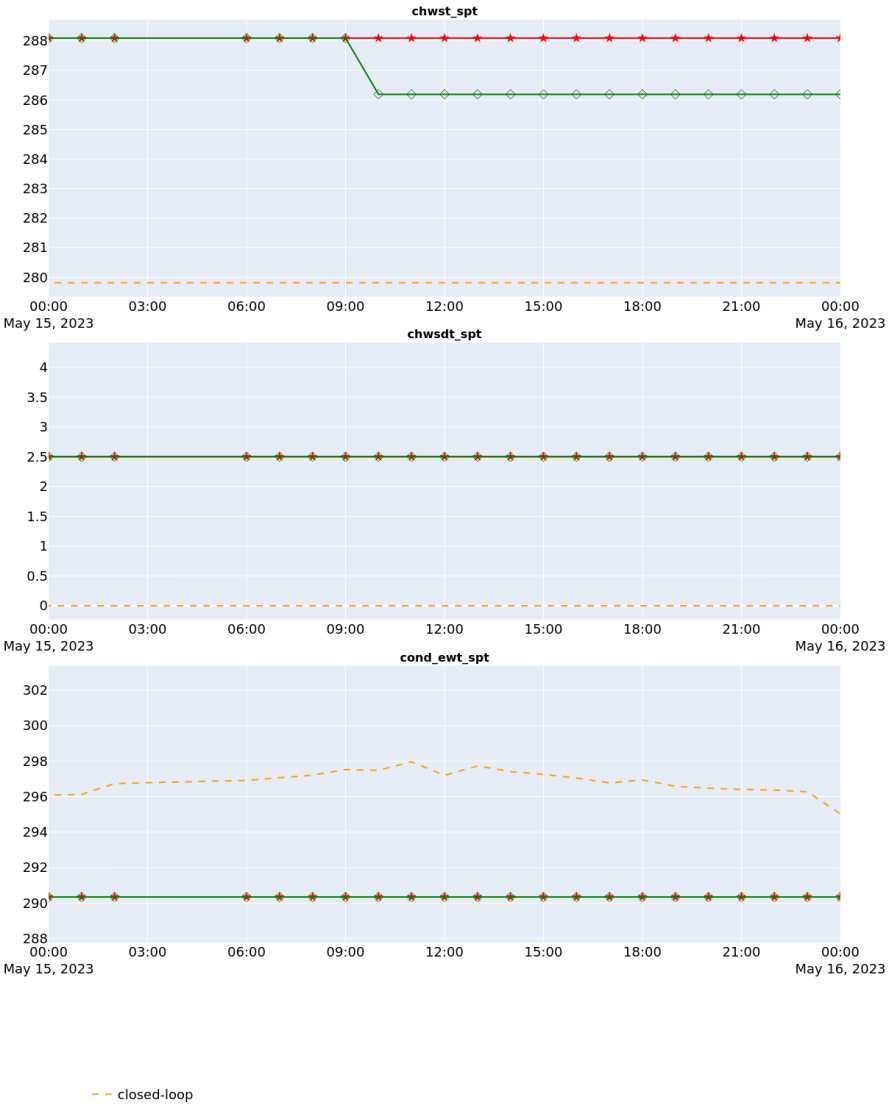
(c) Metrics

Figure B.3 Operating conditions for a day in spring
SA: single-agent RL. **MA - LINMIX**: multi-agent linear mixing (proposed). **CL**: closed-loop controller



(d) Staging Control

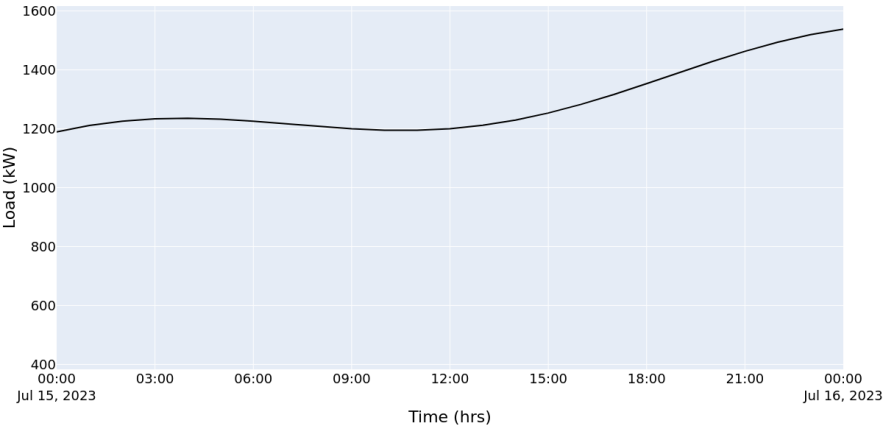
Figure B.3 Operating conditions for a day in spring
SA: single-agent RL. **MA - LINMIX**: multi-agent linear mixing (proposed). **CL**: closed-loop controller



(e) Setpoint Control

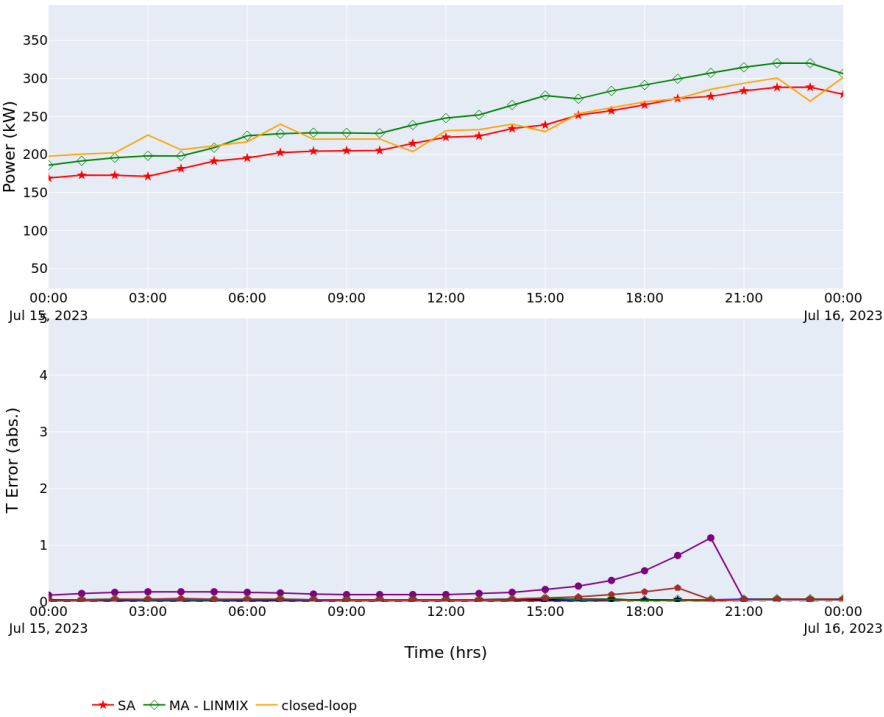
Figure B.3 Operating conditions for a day in spring
SA: single-agent RL. **MA - LINMIX**: multi-agent linear mixing (proposed). **CL**: closed-loop controller

(a) Wetbulb temperature



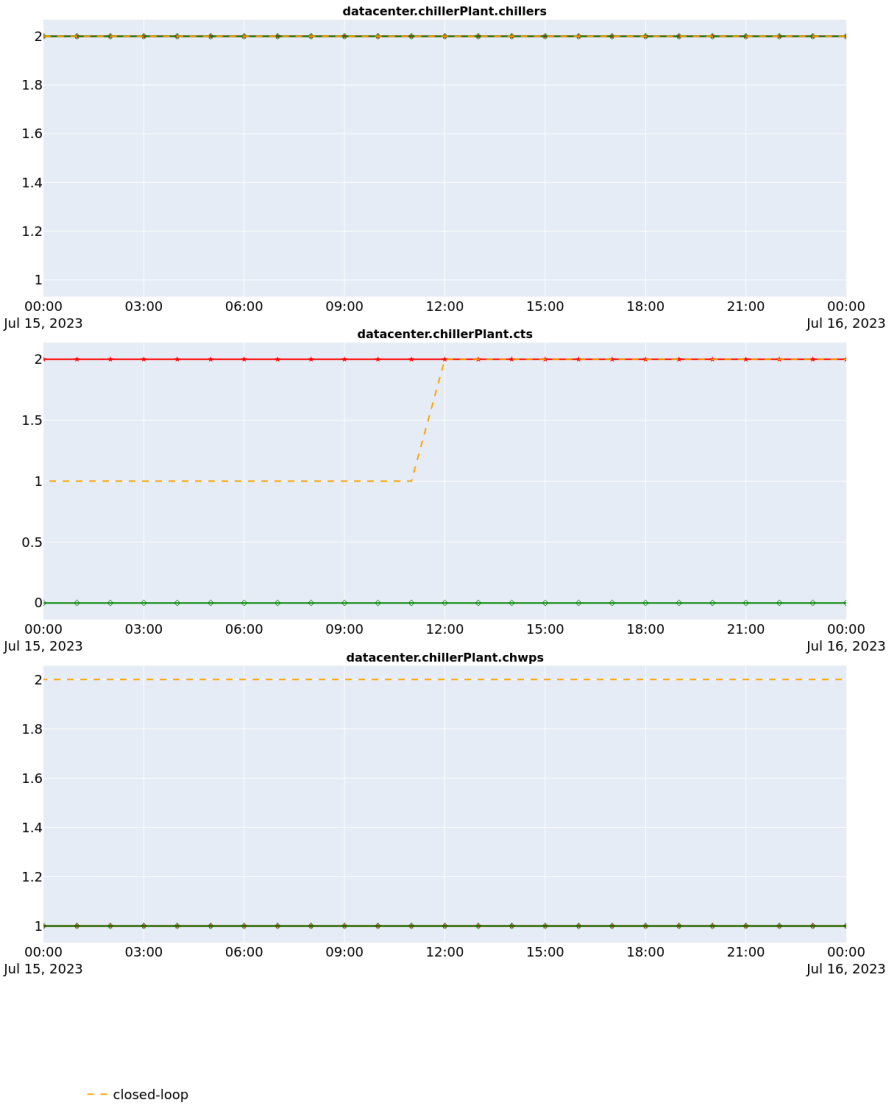
(b) Load

Figure B.4 Operating conditions for a day in summer



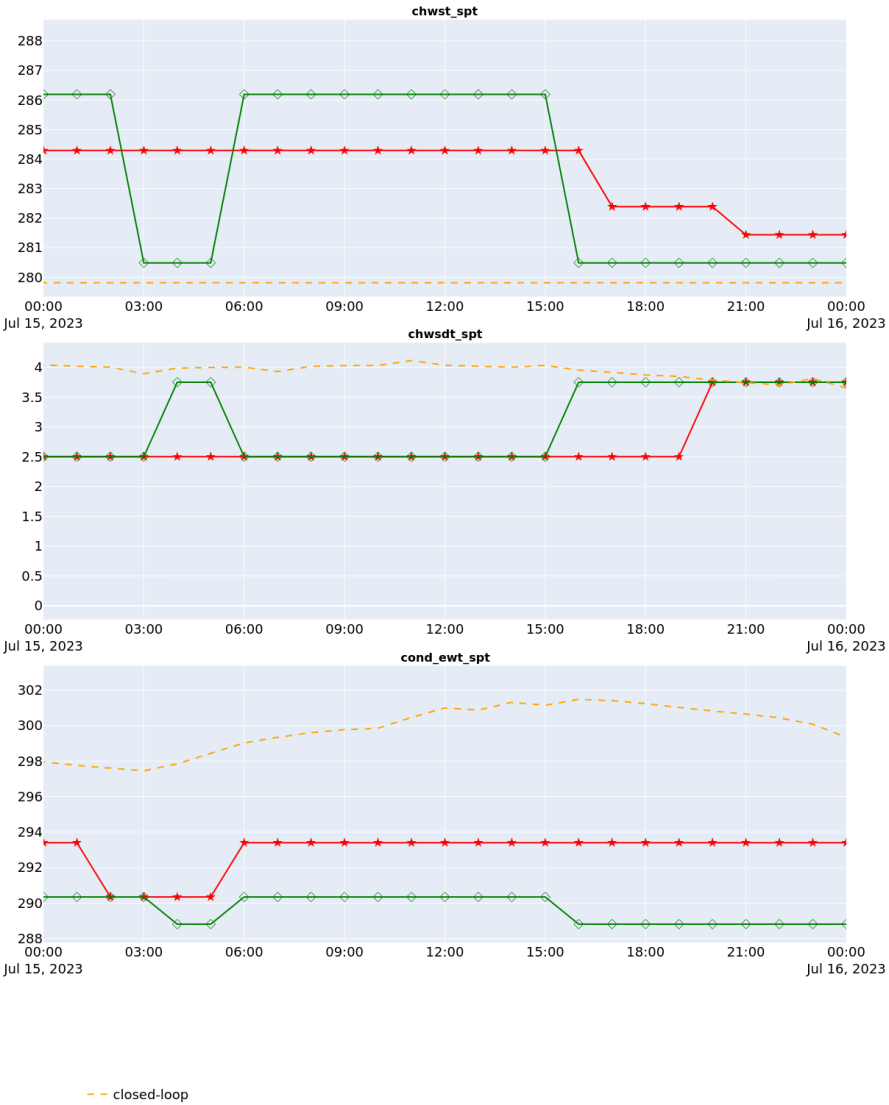
(c) Metrics

Figure B.4 Operating conditions for a day in summer
SA: single-agent RL. **MA - LINMIX**: multi-agent linear mixing (proposed). **CL**: closed-loop controller



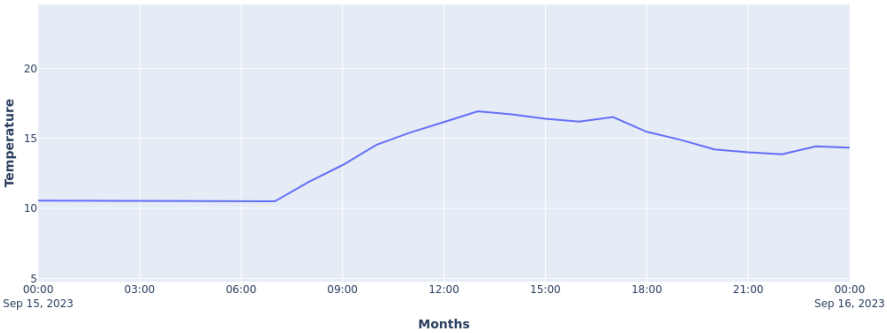
(d) Staging Control

Figure B.4 Operating conditions for a day in summer
SA: single-agent RL. **MA - LINMIX**: multi-agent linear mixing (proposed). **CL**: closed-loop controller

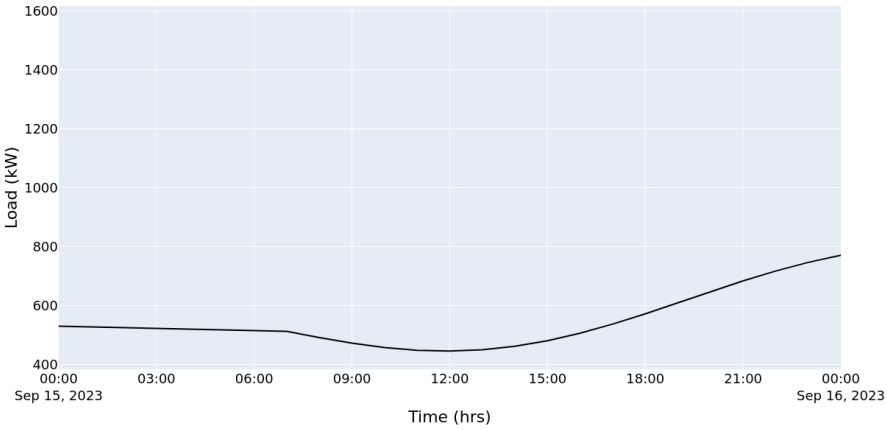


(e) Setpoint Control

Figure B.4 Operating conditions for a day in summer
SA: single-agent RL. **MA - LINMIX**: multi-agent linear mixing (proposed). **CL**: closed-loop controller

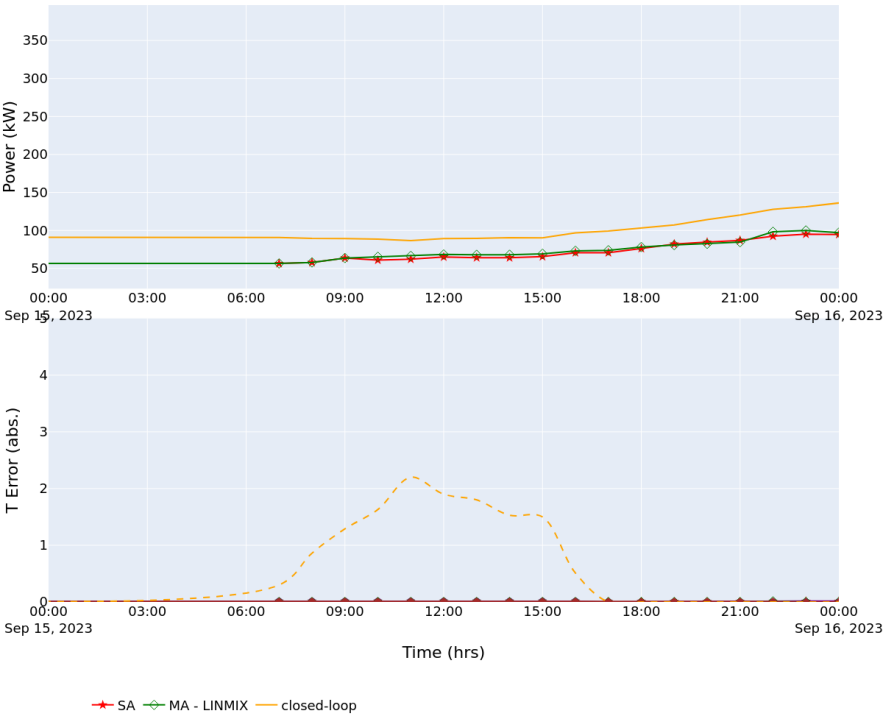


(a) Wetbulb temperature



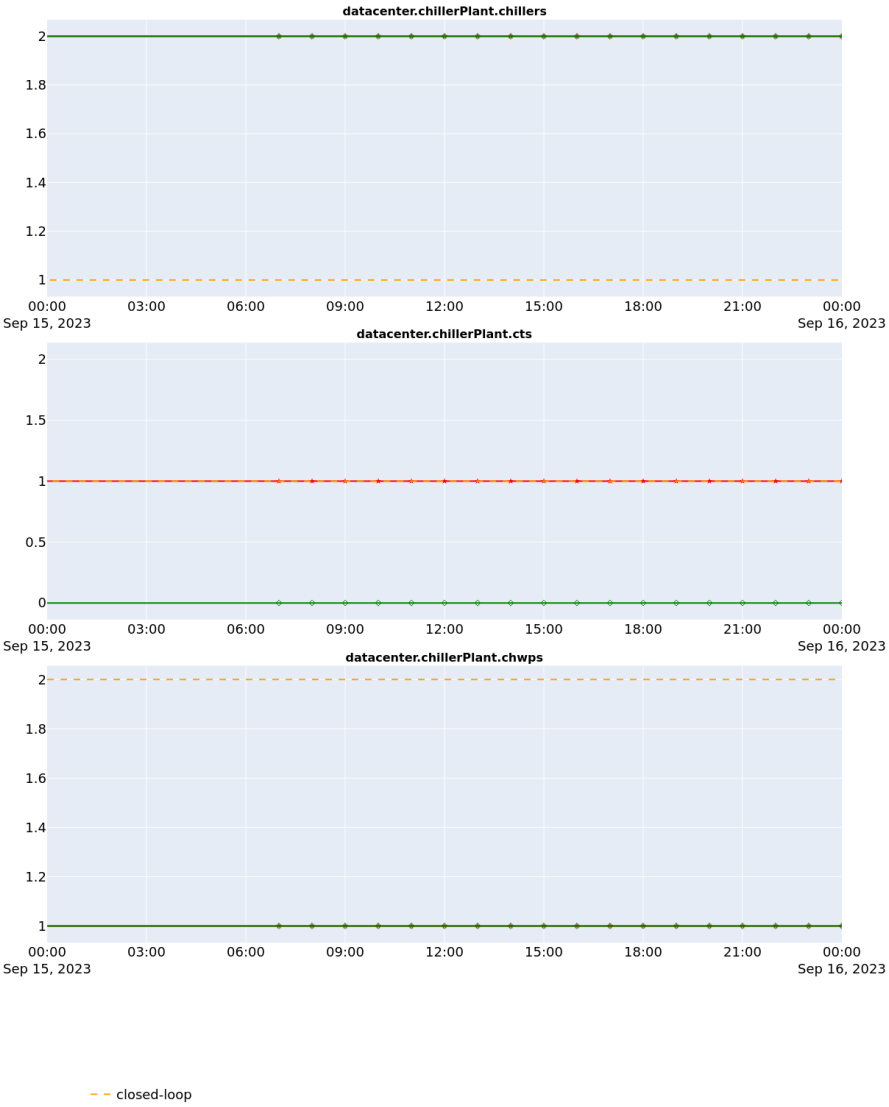
(b) Load

Figure B.5 Operating conditions for a day in autumn



(c) Metrics

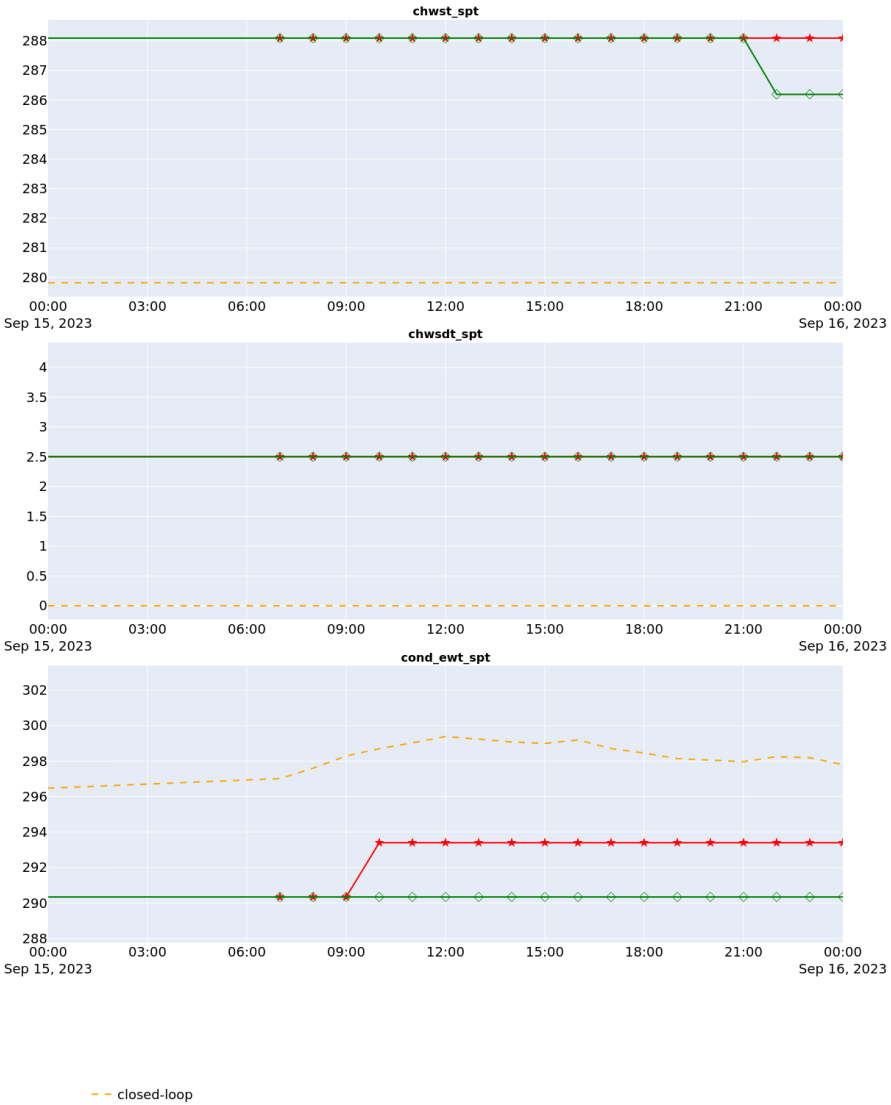
Figure B.5 Operating conditions for a day in autumn
SA: single-agent RL. **MA - LINMIX**: multi-agent linear mixing (proposed). **CL**: closed-loop controller



(d) Staging Control

(e) Operating conditions for a day in autumn

SA: single-agent RL. **MA - LINMIX**: multi-agent linear mixing (proposed). **CL**: closed-loop controller

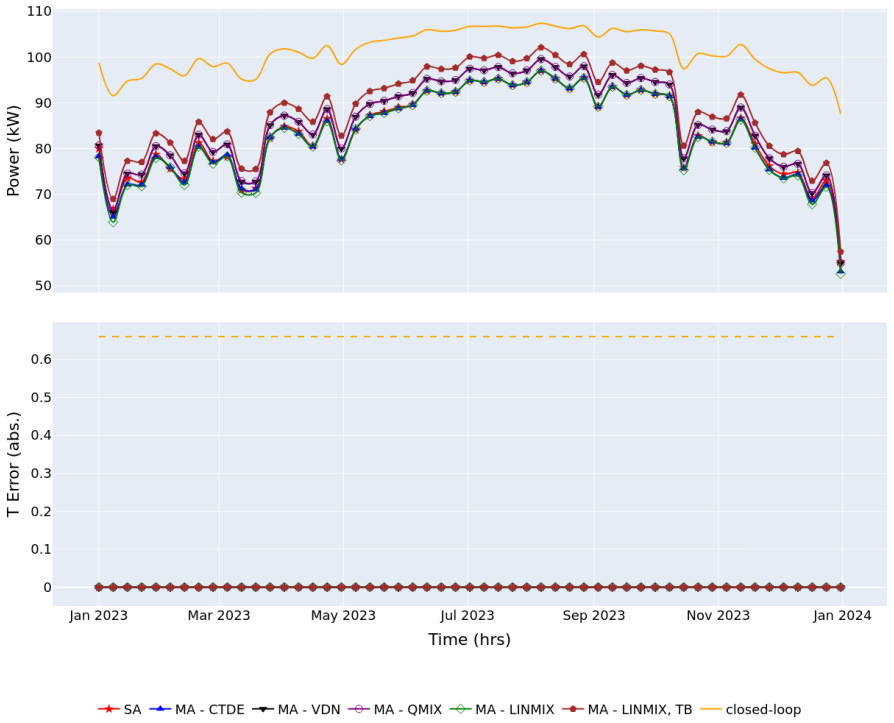


(f) Setpoint Control

Figure B.5 Operating conditions for a day in autumn
SA: single-agent RL. **MA - LINMIX**: multi-agent linear mixing (proposed). **CL**: closed-loop controller

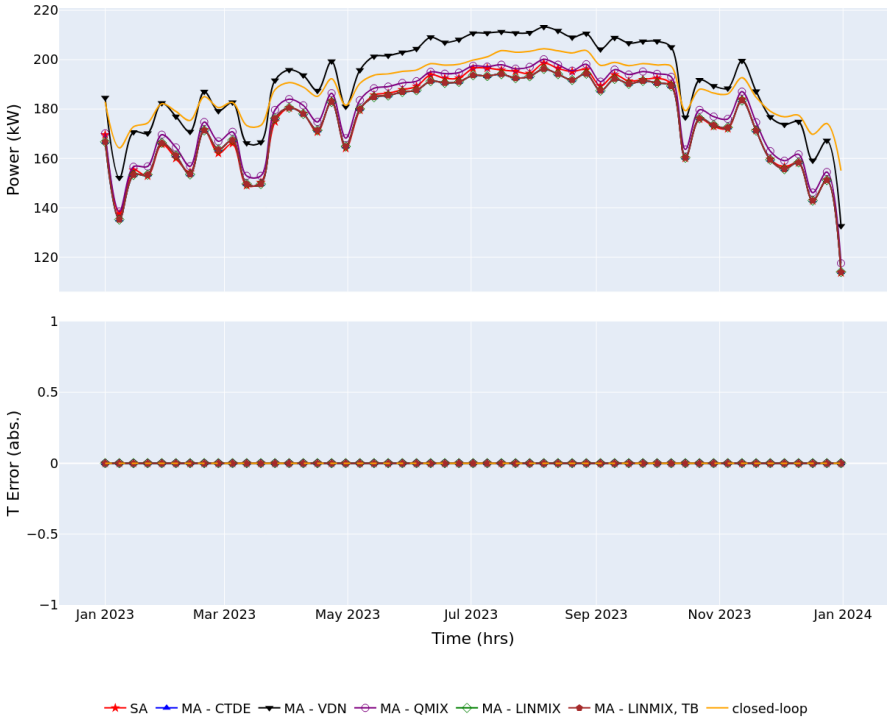
C

Results - Florida



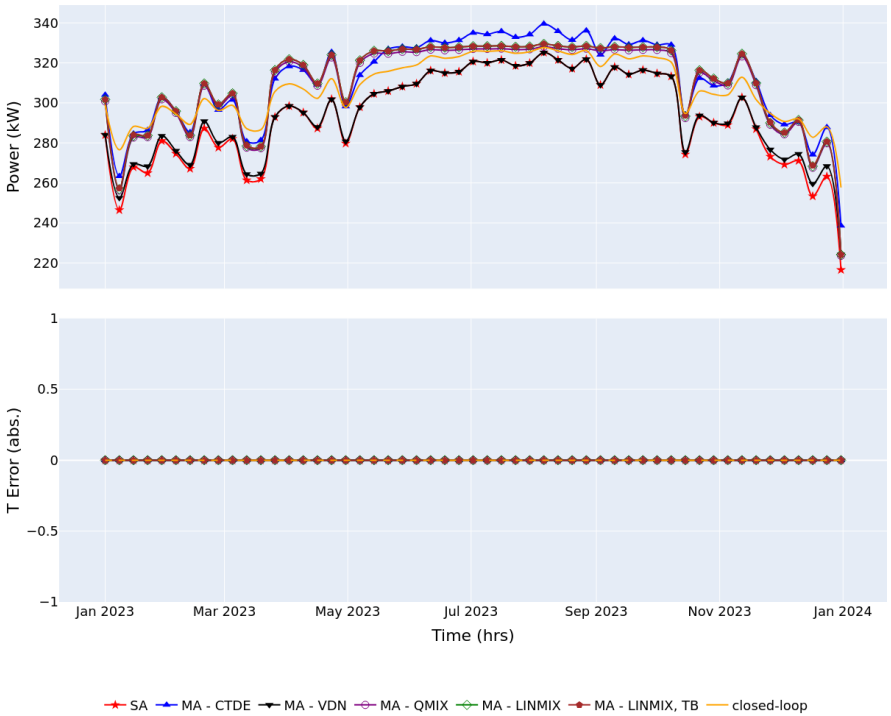
(a) Load 500 kW

Figure C.1 Performance metrics under constant load



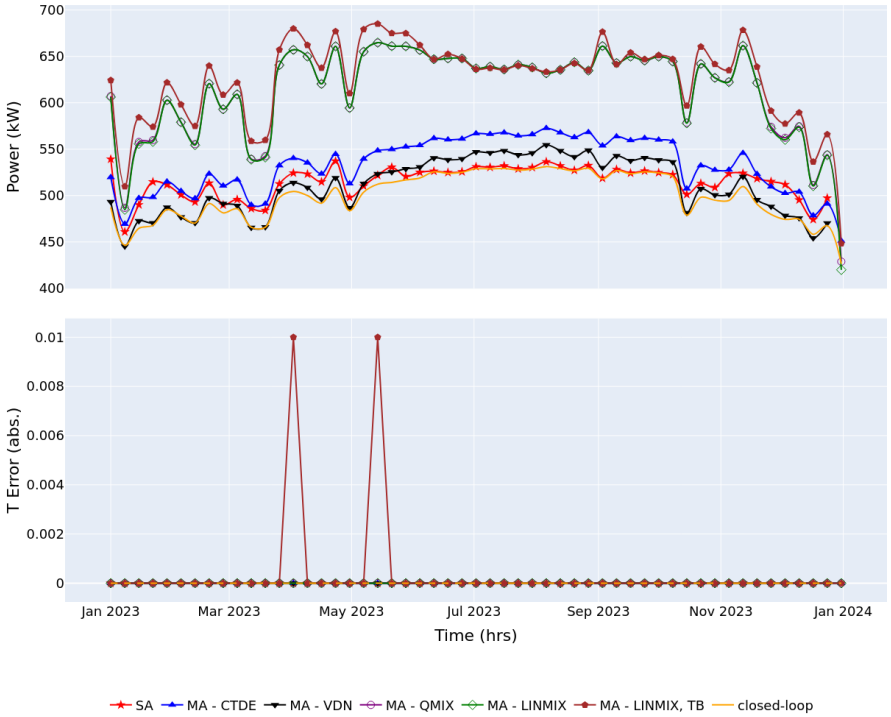
(b) Load 1500 kW

Figure C.1 Performance metrics under constant load



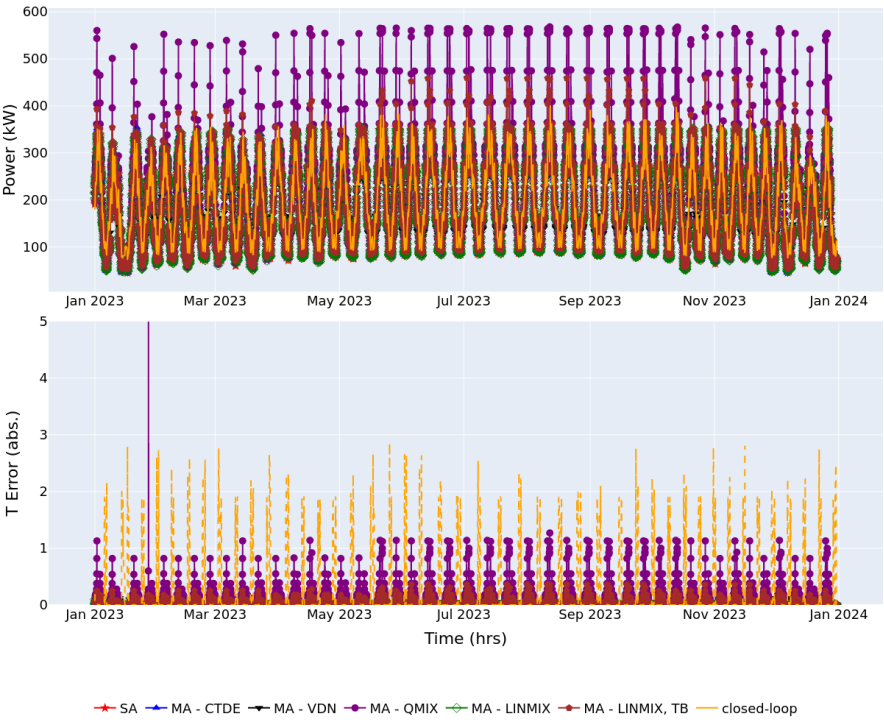
(c) Load 1500 kW

Figure C.1 Performance metrics under constant load



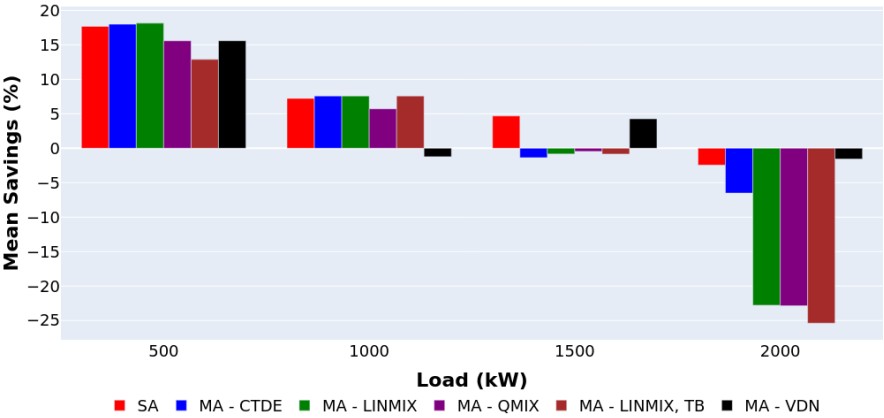
(d) Load 2000 kW

Figure C.1 Performance metrics under constant load

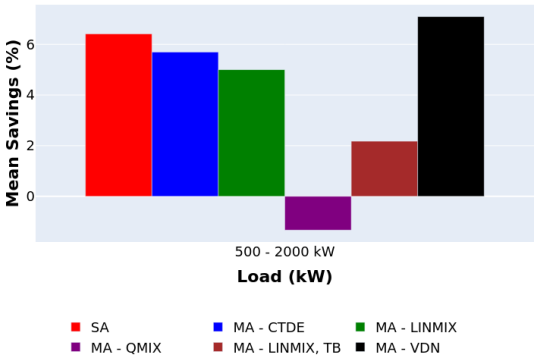


(e) Sinusoidal load, 500 - 2000 kW

Figure C.1 Performance metrics under sinusoidal load
SA: single-agent RL. **MA - CTDE**: multi-agent centralized learning and decentralized execution. **MA - VDN**: multi-agent sum-mixing. **MA - QMIX**: multi-agent QMIX. **MA - LINMIX**: multi-agent linear mixing (proposed). **MA - TB**: multi-agent QMIX and turn-based game (proposed). **CL**: closed-loop controller

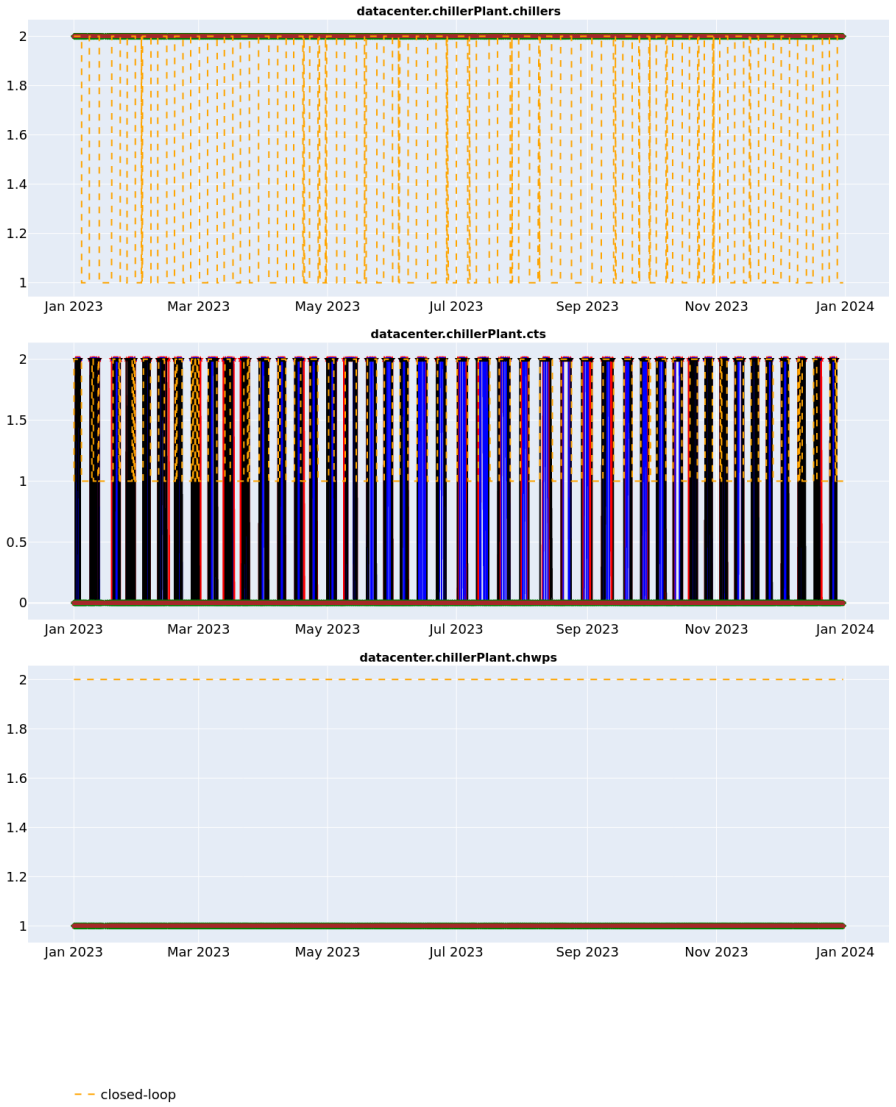


(f) Constant load



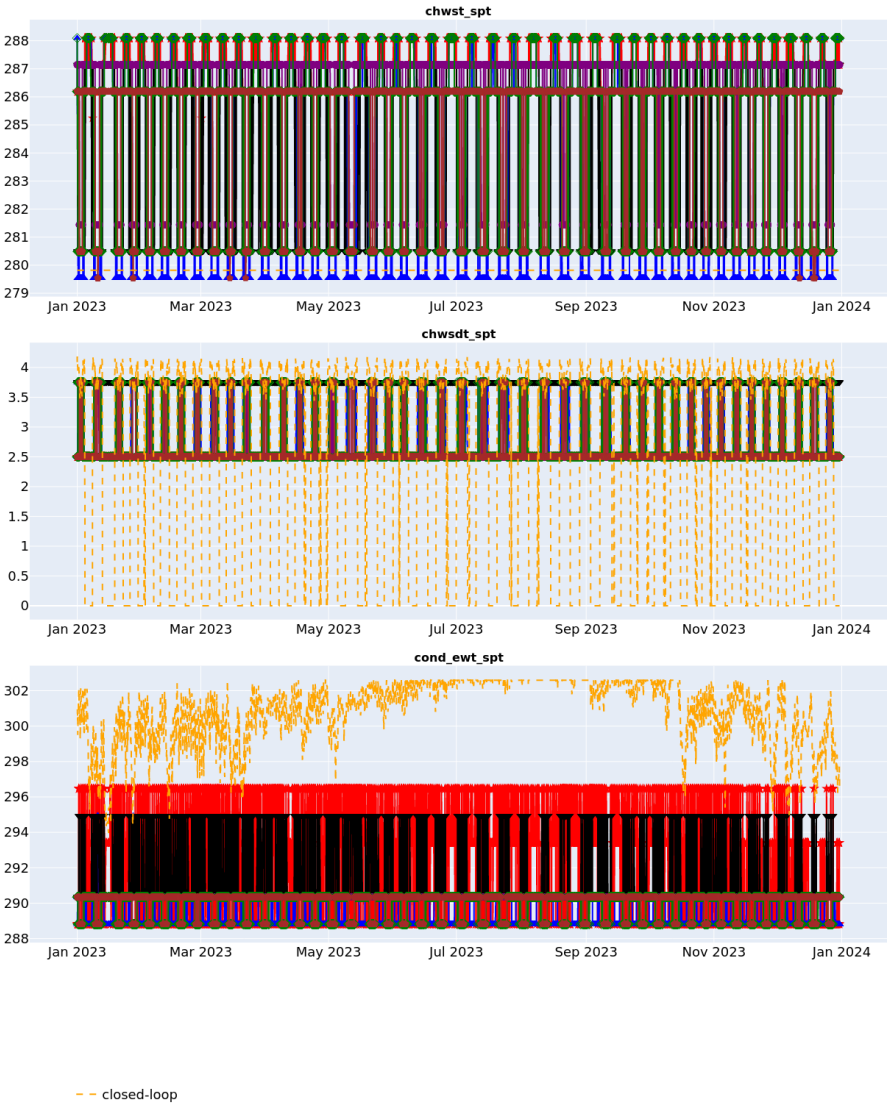
(g) Sinusoidal load

Figure C.1 Average yearly savings (relative to closed-loop controller). All RL agents appear to a positive saving, with the constant load of 2000 kW being an outlier.
SA: single-agent RL. **MA - CTDE**: multi-agent centralized learning and decentralized execution. **MA - VDN**: multi-agent sum-mixing. **MA - QMIX**: multi-agent QMIX. **MA - LINMIX**: multi-agent linear mixing (proposed). **MA - TB**: multi-agent QMIX and turn-based game (proposed). **CL**: closed-loop controller



(a) Staging Control

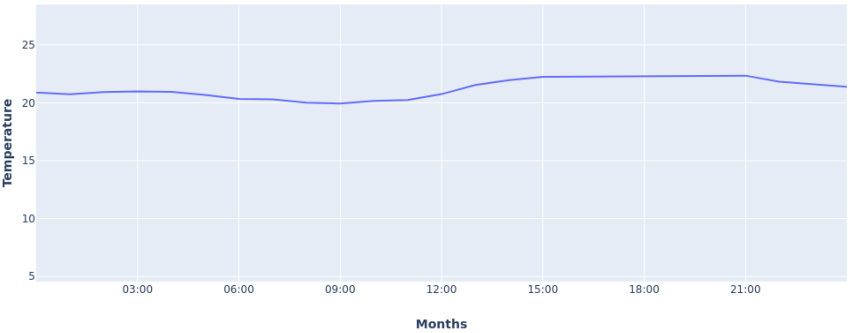
Figure C.2 Control consensus, all RL agents seem to prefer a roughly similar control strategy.



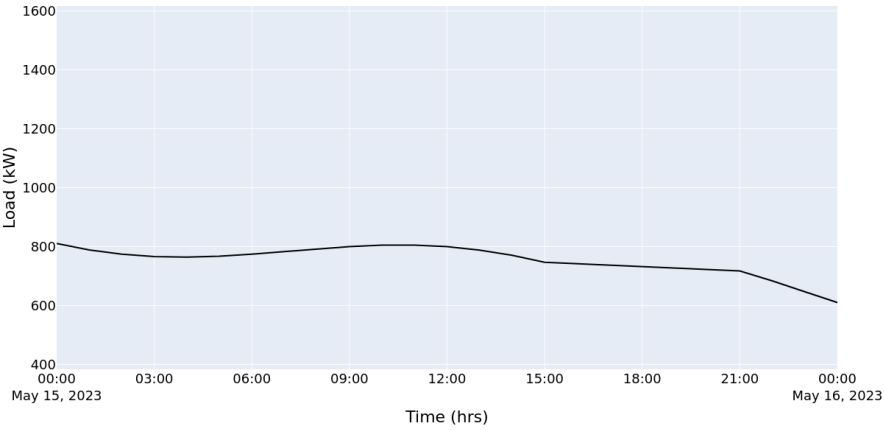
(b) Setpoint Control

Figure C.2 Control consensus: all RL agents seem to prefer a roughly similar control strategy, although there are a few discrepancies.

SA: single-agent RL. **MA - CTDE**: multi-agent centralized learning and decentralized execution. **MA - VDN**: multi-agent sum-mixing. **MA - QMIX**: multi-agent QMIX. **MA - LINMIX**: multi-agent linear mixing (proposed). **MA - TB**: multi-agent QMIX and turn-based game (proposed). **CL**: closed-loop controller

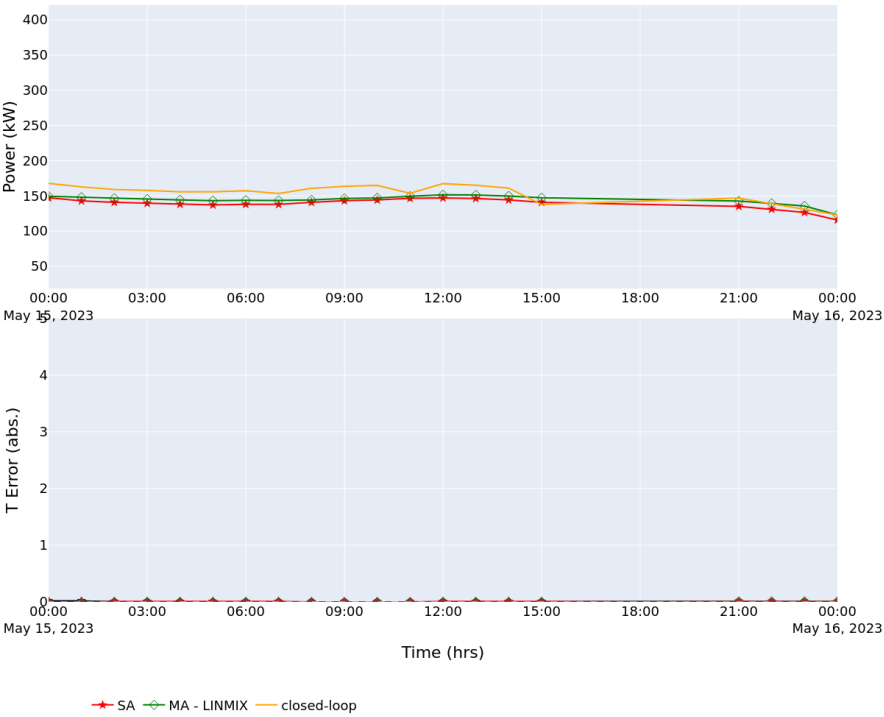


(a) Wetbulb temperature



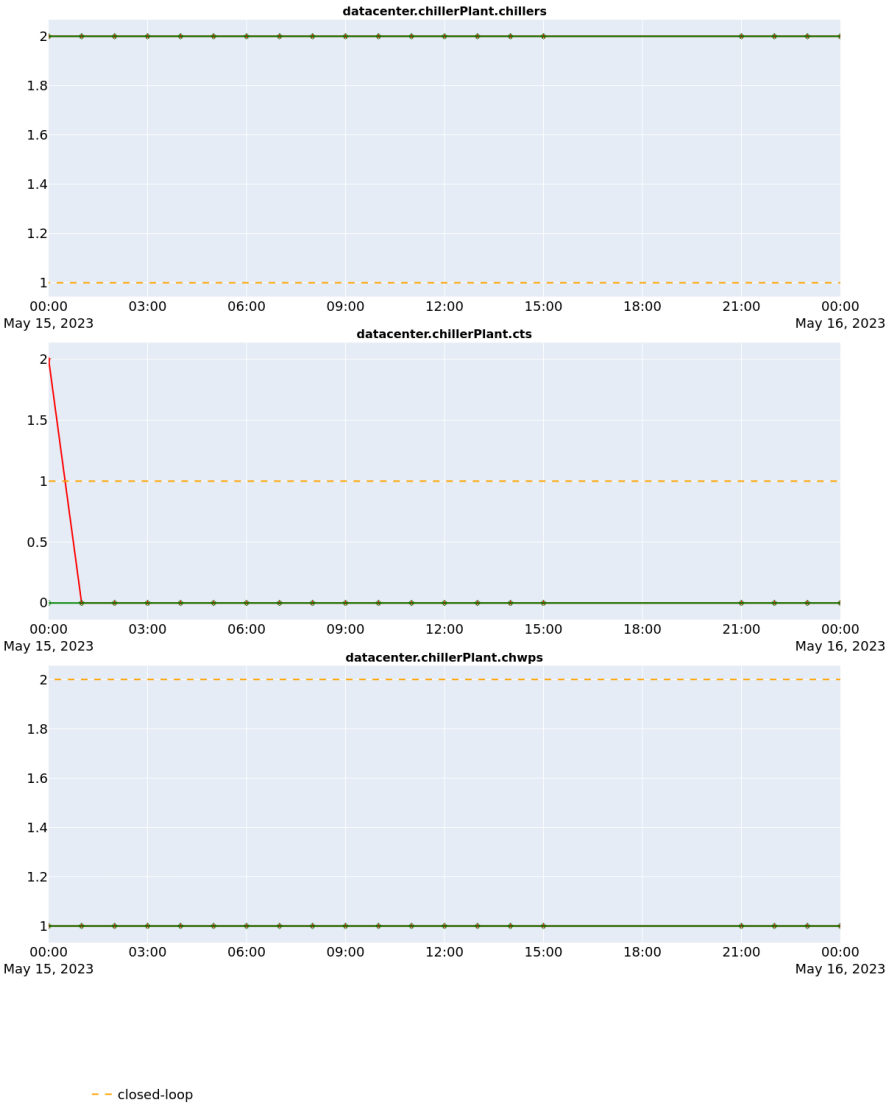
(b) Load

Figure C.3 Operating conditions for a day in spring



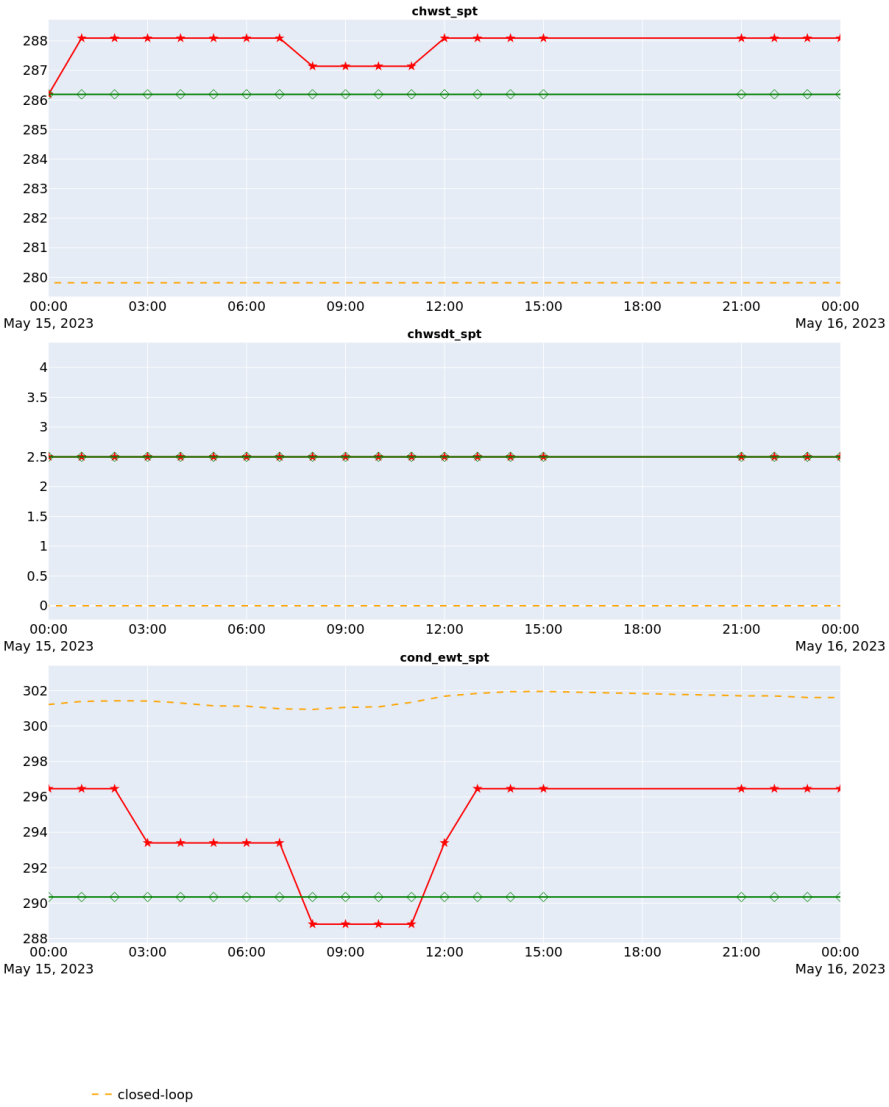
(c) Metrics

Figure C.3 Operating conditions for a day in spring
SA: single-agent RL. **MA - LINMIX**: multi-agent linear mixing (proposed). **CL**: closed-loop controller



(d) Staging Control

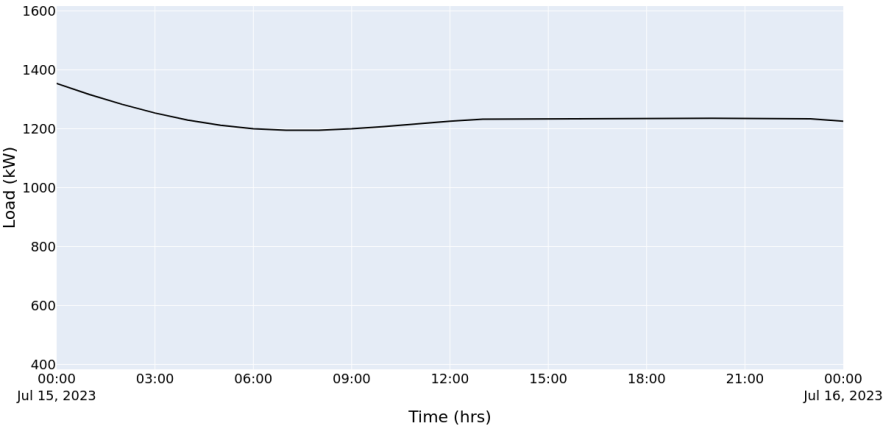
Figure C.3 Operating conditions for a day in spring
SA: single-agent RL. **MA - LINMIX**: multi-agent linear mixing (proposed). **CL**: closed-loop controller



(e) Setpoint Control

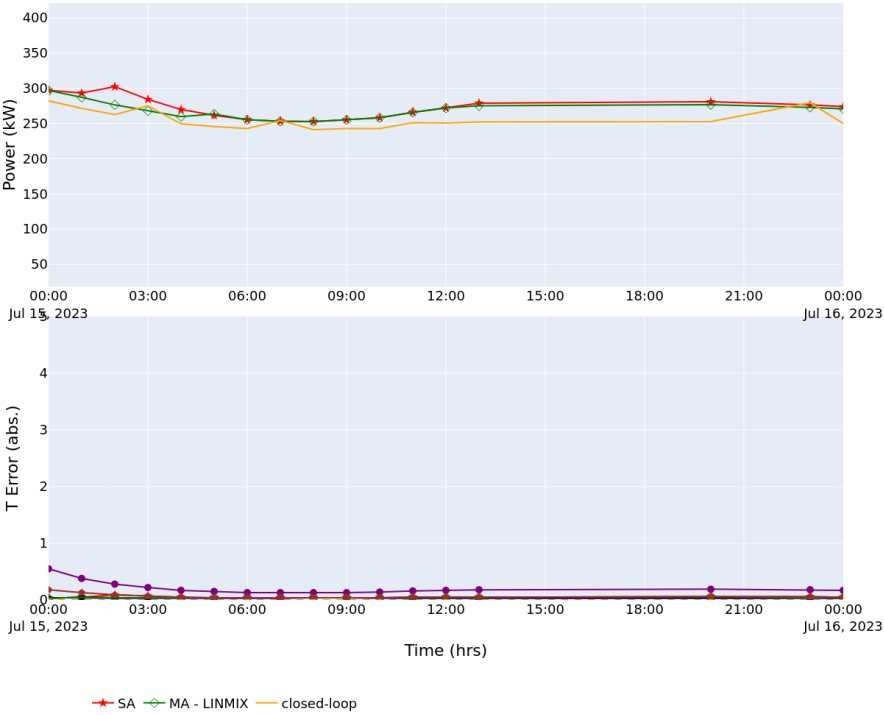
Figure C.3 Operating conditions for a day in spring
SA: single-agent RL. **MA - LINMIX**: multi-agent linear mixing (proposed). **CL**: closed-loop controller

(a) Wetbulb temperature



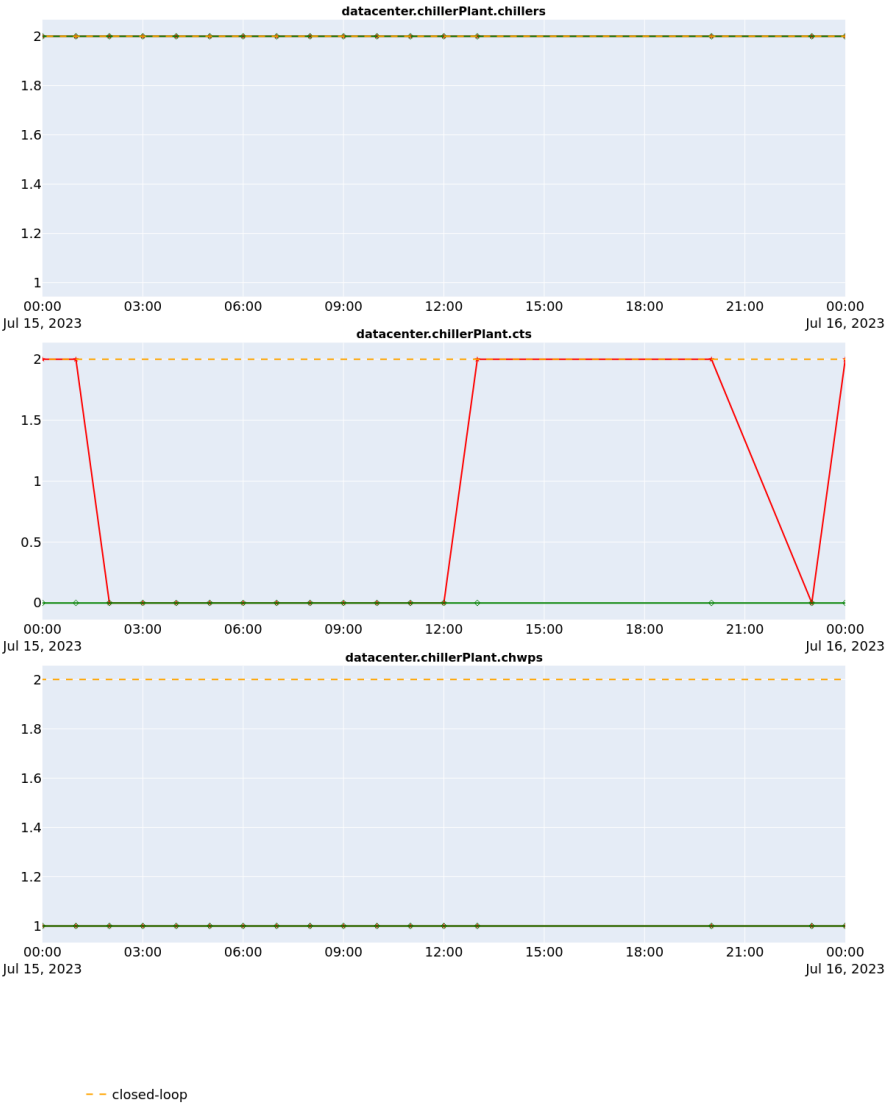
(b) Load

Figure C.4 Operating conditions for a day in summer



(c) Metrics

Figure C.4 Operating conditions for a day in summer
SA: single-agent RL. **MA - LINMIX**: multi-agent linear mixing (proposed). **CL**: closed-loop controller

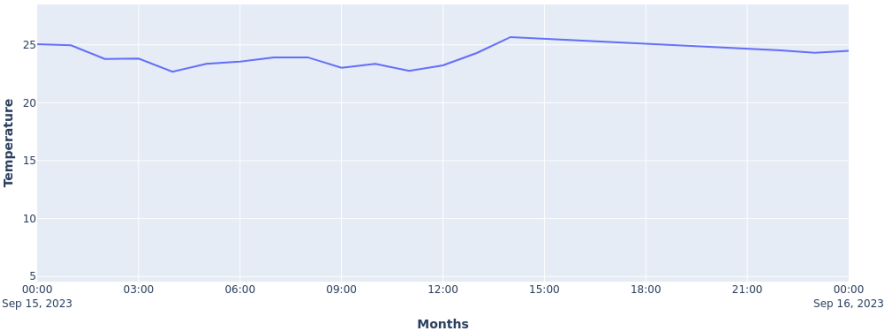


(d) Staging Control

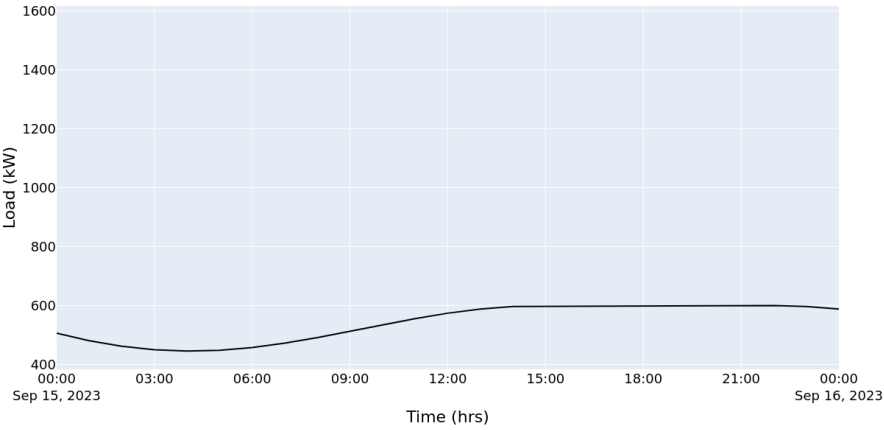
Figure C.4 Operating conditions for a day in summer
SA: single-agent RL. **MA - LINMIX**: multi-agent linear mixing (proposed). **CL**: closed-loop controller

(e) Setpoint Control

Figure C.4 Operating conditions for a day in summer
SA: single-agent RL. **MA - LINMIX**: multi-agent linear mixing (proposed). **CL**: closed-loop controller

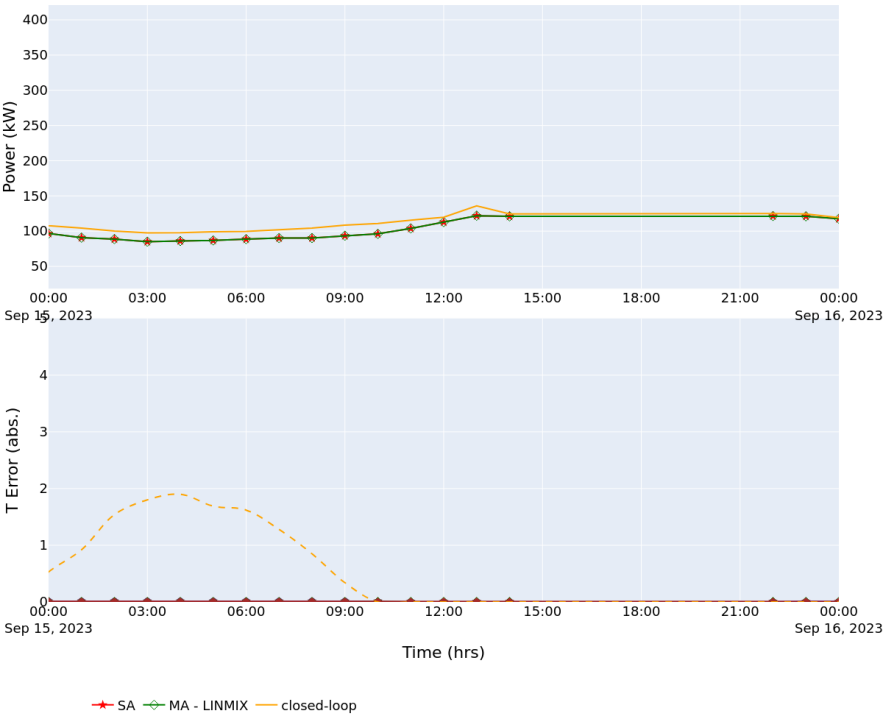


(a) Wetbulb temperature



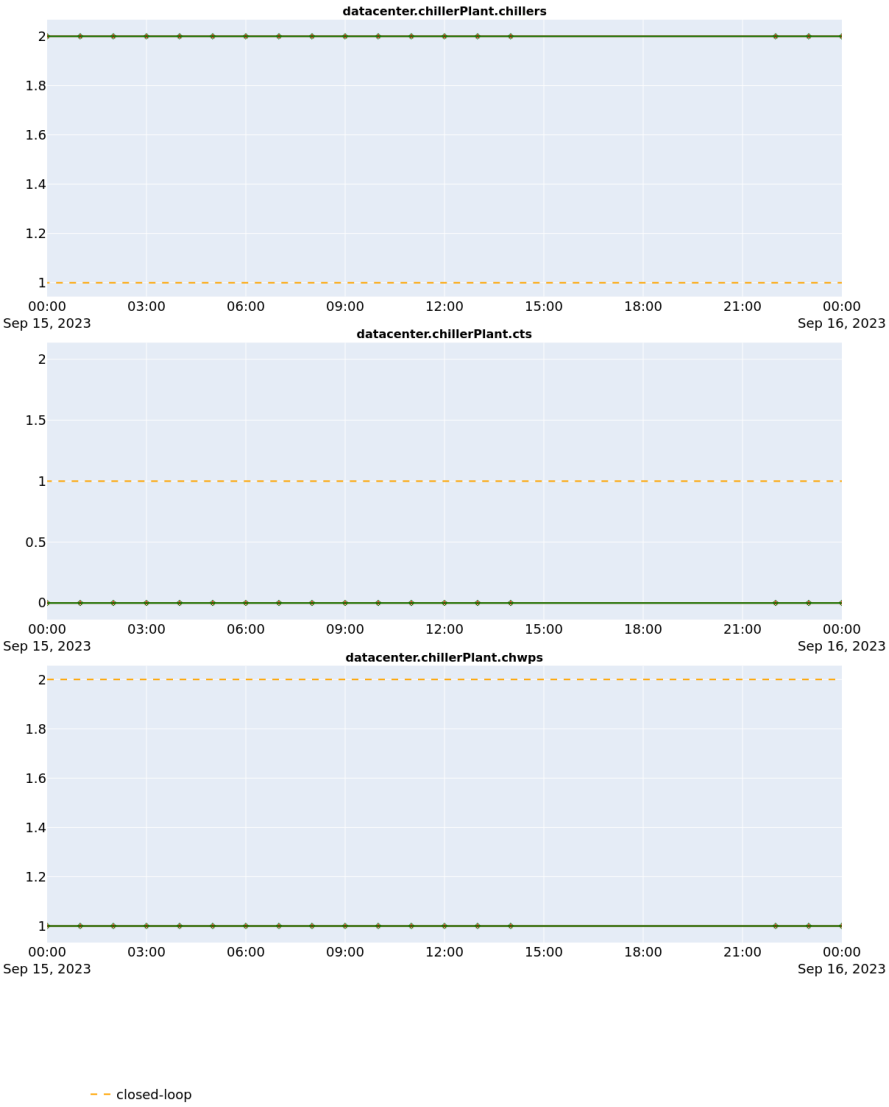
(b) Load

Figure C.5 Operating conditions for a day in autumn



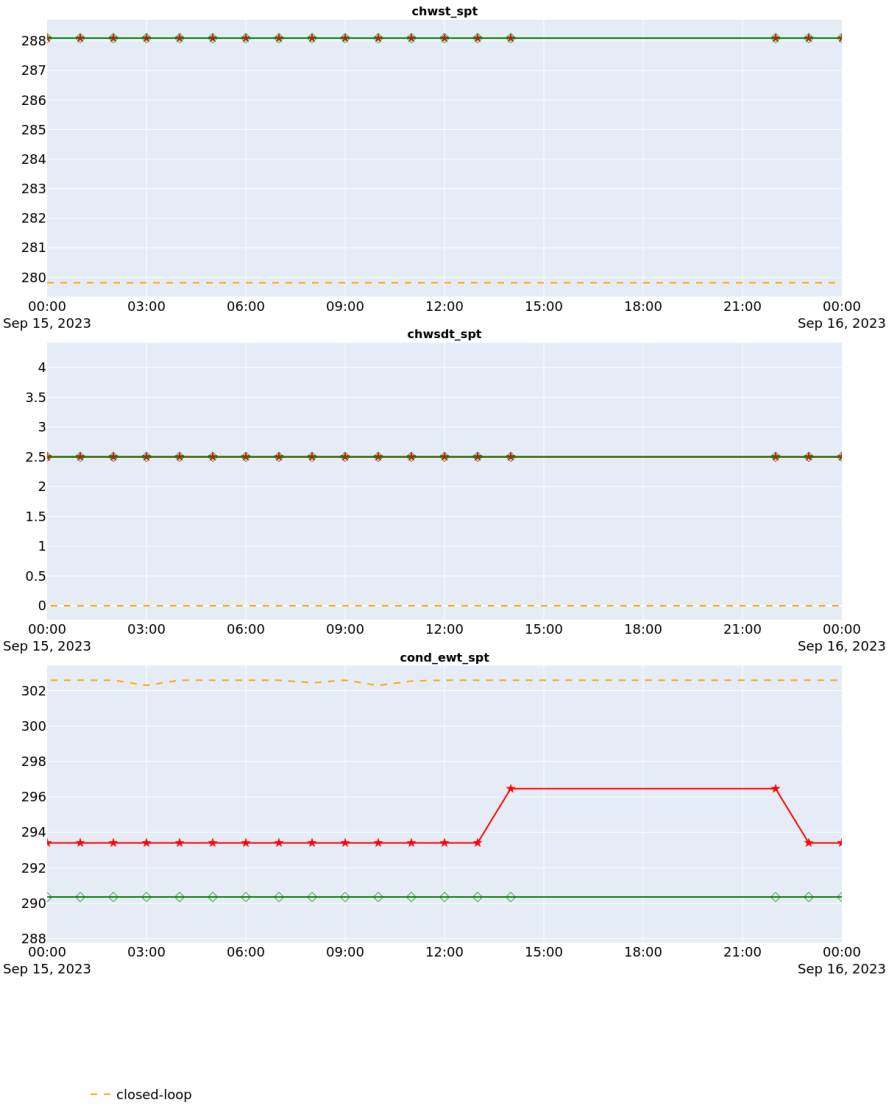
(c) Metrics

Figure C.5 Operating conditions for a day in autumn
SA: single-agent RL. **MA - LINMIX**: multi-agent linear mixing (proposed). **CL**: closed-loop controller



(d) Staging Control

Figure C.5 Operating conditions for a day in autumn
SA: single-agent RL. **MA - LINMIX**: multi-agent linear mixing (proposed). **CL**: closed-loop controller



(e) Setpoint Control

Figure C.5 Operating conditions for a day in autumn
SA: single-agent RL. **MA - LINMIX**: multi-agent linear mixing (proposed). **CL**: closed-loop controller

D

Tools & Software

Software	Version	Description
ModelonImpact	1.8.1	A modelling and simulation environment used to interact with Carrier's building models.
PyFMI	2.11.0	A python package used to interact with functional mockup units (FMUs).
Gymnasium	0.26.1	An API standard for implementing compatible RL environments. A custom environment was implemented as a wrapper for the FMU.
Stablebaselines3	2.3.0	A robust Python implementation of contemporary reinforcement learning algorithms.
raylib	1.8.0	A robust and scalable Python implementation of single-agent and multi-agent RL algorithms.
MARLlib	1.0.3	An extension of raylib supporting multi-agent RL algorithms. The library was forked to implement compatibility to support agents with dissimilar action spaces

Bibliography

Abergel T., B. D. (2018). “Global status report - world green building council”.

Arulkumaran, K., M. P. Deisenroth, M. Brundage, and A. A. Bharath (2017). “Deep reinforcement learning: a brief survey”. *IEEE Signal Processing Magazine* **34**:6, pp. 26–38. ISSN: 1053-5888. DOI: 10.1109/msp.2017.2743240.

Åström, Karl Johan (1965). “Optimal Control of Markov Processes with Incomplete State Information I”. eng. Journal of Mathematical Analysis and Applications **10**, 174–205. ISSN: 0022-247X. DOI: {10.1016/0022-247X(65)90154-X}.

Busoniu, L., R. Babuska, and B. De Schutter (2008). “A comprehensive survey of multiagent reinforcement learning”. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* **38**:2, pp. 156–172. DOI: 10.1109/TSMCC.2007.913919.

Charbonnier, F., T. Morstyn, and M. D. McCulloch (2022). “Scalable multi-agent reinforcement learning for distributed control of residential energy flexibility”. *Applied Energy* **314**, p. 118825. ISSN: 0306-2619. DOI: https://doi.org/10.1016/j.apenergy.2022.118825.

Cobbe, K., J. Hilton, O. Klimov, and J. Schulman (2020). *Phasic policy gradient*. arXiv: 2009.04416 [cs.LG].

Czech, J. (2021). “Distributed methods for reinforcement learning survey”. In: Belousov, B. et al. (Eds.). *Reinforcement Learning Algorithms: Analysis and Applications*. Springer International Publishing, Cham, pp. 151–161. ISBN: 978-3-030-41188-6. DOI: 10.1007/978-3-030-41188-6_13.

Dulac-Arnold, G., D. Mankowitz, and T. Hester (2019). *Challenges of real-world reinforcement learning*. arXiv: 1904.12901 [cs.LG].

Energy Efficiency & Renewable Energy, O. of (n.d.). “Buildings energy data book” ().

Fujimoto, S., H. van Hoof, and D. Meger (2018). *Addressing function approximation error in actor-critic methods*. arXiv: 1802.09477 [cs.AI].

Gao, G., J. Li, and Y. Wen (2019). *Energy-efficient thermal comfort control in smart buildings via deep reinforcement learning*. arXiv: 1901.04693 [cs.SY].

Gao, G., J. Li, and Y. Wen (2020). "Deepcomfort: energy-efficient thermal comfort control in buildings via reinforcement learning". *IEEE Internet of Things Journal* **7**:9, pp. 8472–8484. DOI: 10.1109/JIOT.2020.2992117.

Gattami, A., Q. Bai, and V. Aggarwal (2021). "Reinforcement learning for constrained markov decision processes". In: Banerjee, A. et al. (Eds.). *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*. Vol. 130. Proceedings of Machine Learning Research. PMLR, pp. 2656–2664.

Ghahramani, A., F. Jazizadeh, and B. Becerik-Gerber (2014). "A knowledge based approach for selecting energy-aware and comfort-driven hvac temperature set points". *Energy and Buildings* **85**, pp. 536–548. ISSN: 0378-7788. DOI: https://doi.org/10.1016/j.enbuild.2014.09.055.

Gronauer, S. and K. Diepold (2022). "Multi-agent deep reinforcement learning: a survey". *Artificial Intelligence Review* **55**:2, pp. 895–943. ISSN: 1573-7462. DOI: 10.1007/s10462-021-09996-w.

Gu, S., L. Yang, Y. Du, G. Chen, F. Walter, J. Wang, Y. Yang, and A. Knoll (2023). *A review of safe reinforcement learning: methods, theory and applications*. arXiv: 2205.10330 [cs.AI].

Hanumaiah, V. and S. Genc (2021). *Distributed multi-agent deep reinforcement learning framework for whole-building hvac control*. arXiv: 2110 . 13450 [cs.LG].

Kaelbling, L. P., M. L. Littman, and A. W. Moore (1996). *Reinforcement learning: a survey*. arXiv: cs/9605103 [cs.AI].

Killian, M. and M. Kozek (2018). "Implementation of cooperative fuzzy model predictive control for an energy-efficient office building". *Energy and Buildings* **158**, pp. 1404–1416. ISSN: 0378-7788. DOI: https://doi.org/10.1016/j.enbuild.2017.11.021.

Kraemer, L. and B. Banerjee (2016). "Multi-agent reinforcement learning as a rehearsal for decentralized planning". *Neurocomputing* **190**, pp. 82–94. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2016.01.031.

Ladosz, P., L. Weng, M. Kim, and H. Oh (2022). "Exploration in deep reinforcement learning: a survey". *Information Fusion* **85**, pp. 1–22. ISSN: 1566-2535. DOI: 10.1016/j.inffus.2022.03.003.

Liang, W., R. Quinte, X. Jia, and J.-Q. Sun (2015). "Mpc control for improving energy efficiency of a building air handler for multi-zone vavs". *Building and Environment* **92**, pp. 256–268. ISSN: 0360-1323. DOI: https://doi.org/10.1016/j.buildenv.2015.04.033.

Lillicrap, T. P., J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra (2019). *Continuous control with deep reinforcement learning*. arXiv: 1509.02971 [cs.LG].

Luo, J., C. Paduraru, O. Voicu, Y. Chervonyi, S. Munns, J. Li, C. Qian, P. Dutta, J. Q. Davis, N. Wu, X. Yang, C.-M. Chang, T. Li, R. Rose, M. Fan, H. Nakhost, T. Liu, B. Kirkman, F. Altamura, L. Cline, P. Tonker, J. Gouker, D. Uden, W. B. Bryan, J. Law, D. Fatiha, N. Satra, J. Rothenberg, M. Waraich, M. Carlin, S. Tallapaka, S. Witherspoon, D. Parish, P. Dolan, C. Zhao, and D. J. Mankowitz (2022). *Controlling commercial cooling systems using reinforcement learning*. arXiv: 2211.07357 [cs.LG].

Ma, Y., F. Borrelli, B. Hencey, B. Coffey, S. Bengea, and P. Haves (2012). "Model predictive control for the operation of building cooling systems". *IEEE Transactions on Control Systems Technology* **20**:3, pp. 796–803. DOI: 10.1109/TCST.2011.2124461.

Model-Based Hierarchical Optimal Control Design for HVAC Systems (2011). Vol. ASME 2011 Dynamic Systems and Control Conference and Bath/ASME Symposium on Fluid Power and Motion Control, Volume 1. Dynamic Systems and Control Conference, pp. 271–278. DOI: 10.1115/DSCC2011-6078. eprint: https://asmedigitalcollection.asme.org/DSCC/proceedings-pdf/DSCC2011/54754/271/2766723/271_1.pdf.

Meimand, M. and F. Jazizadeh (2022). "Human-in-the-loop model predictive operation for energy efficient hvac systems". In: pp. 178–187. DOI: 10.1061/9780784483954.019.

Mnih, V., A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu (2016). *Asynchronous methods for deep reinforcement learning*. arXiv: 1602.01783 [cs.LG].

Nagy, Z., B. Gunay, C. Miller, J. Hahn, M. M. Ouf, S. Lee, B. W. Hobson, T. Abuimara, K. Bandurski, M. André, C.-L. Lorenz, S. Crosby, B. Dong, Z. Jiang, Y. Peng, M. Favero, J. Y. Park, K. Nweye, P. Nojedehi, H. Stopps, L. Sarran, C. Brackley, K. Bassett, K. Govertsen, N. Koczorek, O. Abele, E. Casavant, M. Kane, Z. O'Neill, T. Yang, J. Day, B. Huchuk, R. T. Hellwig, and M. Vellei (2023). "Ten questions concerning occupant-centric control and operations". *Building and Environment* **242**, p. 110518. ISSN: 0360-1323. DOI: https://doi.org/10.1016/j.buildenv.2023.110518.

O'Dwyer, E., L. De Tommasi, K. Kouramas, M. Cychowski, and G. Lightbody (2017). "Prioritised objectives for model predictive control of building heating systems". *Control Engineering Practice* **63**, pp. 57–68. ISSN: 0967-0661. DOI: https://doi.org/10.1016/j.conengprac.2017.03.018.

Oldewurtel, F., A. Parisio, C. Jones, M. Morari, D. Gyalistras, M. Gwerder, V. Stauch, B. Lehmann, and K. Wirth (2010). "Energy efficient building climate control using stochastic model predictive control and weather predictions". In: Ieee Service Center, 445 Hoes Lane, Po Box 1331, Piscataway, Nj 08855-1331 Usa, pp. 5100–5105. DOI: https://doi.org/10.1109/ACC.2010.5530680.

Oliehoek, F. A., M. T. J. Spaan, and N. Vlassis (2008). "Optimal and approximate q-value functions for decentralized pomdps". *J. Artif. Int. Res.* **32**:1, pp. 289–353. ISSN: 1076-9757.

Powell, W. (2022a). *Reinforcement Learning and Stochastic Optimization*. John Wiley & Sons, Ltd.

Powell, W. (2022b). *Sequential Decision Analytics and Modeling (with Python)*. John Wiley & Sons, Ltd.

Rashid, T., M. Samvelyan, C. S. de Witt, G. Farquhar, J. Foerster, and S. Whiteson (2018). *Qmix: monotonic value function factorisation for deep multi-agent reinforcement learning*. arXiv: 1803.11485 [cs.LG].

Roijers, D. M., P. Vamplew, S. Whiteson, and R. Dazeley (2013). "A survey of multi-objective sequential decision-making". *Journal of Artificial Intelligence Research* **48**, pp. 67–113. ISSN: 1076-9757. DOI: 10.1613/jair.3987.

Schulman, J., S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel (2017a). *Trust region policy optimization*. arXiv: 1502.05477 [cs.LG].

Schulman, J., F. Wolski, P. Dhariwal, A. Radford, and O. Klimov (2017b). *Proximal policy optimization algorithms*. arXiv: 1707.06347 [cs.LG].

Shaikh, P. H., N. B. M. Nor, P. Nallagownden, I. Elamvazuthi, and T. Ibrahim (2014). "A review on optimized control systems for building energy and comfort management of smart sustainable buildings". *Renewable and Sustainable Energy Reviews* **34**, pp. 409–429. ISSN: 1364-0321. DOI: https://doi.org/10.1016/j.rser.2014.03.027.

Shakya, A. K., G. Pillai, and S. Chakrabarty (2023). "Reinforcement learning algorithms: a brief survey". *Expert Systems with Applications* **231**, p. 120495. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2023.120495.

Sierla, S., H. Ihasalo, and V. Vyatkin (2022). "A review of reinforcement learning applications to control of heating, ventilation and air conditioning systems". *Energies* **15**:10. ISSN: 1996-1073. DOI: 10.3390/en15103526.

Stefano V. Albrecht Filippos Christianos, L. S. (2024). *Multi-Agent Reinforcement Learning: Foundations and Modern Approaches*. MIT Press.

Sunehag, P., G. Lever, A. Gruslys, W. M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls, and T. Graepel (2017). *Value-decomposition networks for cooperative multi-agent learning*. arXiv: 1706.05296 [cs.AI].

Sutton, R. S. (1988). "Learning to predict by the methods of temporal differences". *Mach. Learn.* **3**:1, pp. 9–44. ISSN: 0885-6125. DOI: 10.1023/A:1022633531479.

Sutton, R. S. and A. G. Barto (2018). *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA. ISBN: 0262039249.

Tamar, A., Y. Glassner, and S. Mannor (2015). "Optimizing the cvar via sampling". *Proceedings of the AAAI Conference on Artificial Intelligence* **29**:1. DOI: 10.1609/aaai.v29i1.9561.

Tan, M. (1997). "Multi-agent reinforcement learning: independent versus cooperative agents". In: *International Conference on Machine Learning*.

Tang, S., M. Makar, M. W. Sjoding, F. Doshi-Velez, and J. Wiens (2023). *Leveraging factored action spaces for efficient offline reinforcement learning in healthcare*. arXiv: 2305.01738 [cs.LG].

Wang, H., X. Chen, N. Vital, E. Duffy, and A. Razi (2024a). "Energy optimization for hvac systems in multi-vav open offices: a deep reinforcement learning approach". *Applied Energy* **356**, p. 122354. ISSN: 0306-2619. DOI: https://doi.org/10.1016/j.apenergy.2023.122354.

Wang, X., S. Wang, X. Liang, D. Zhao, J. Huang, X. Xu, B. Dai, and Q. Miao (2024b). "Deep reinforcement learning: a survey". *IEEE Transactions on Neural Networks and Learning Systems* **35**:4, pp. 5064–5078. DOI: 10.1109/TNNLS.2022.3207346.

Wei, T., Y. Wang, and Q. Zhu (2017). "Deep reinforcement learning for building hvac control". In: *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1–6. DOI: 10.1145/3061639.3062224.

Wei, T., Q. Zhu, and M. Maasoumy (2014). "Co-scheduling of hvac control, ev charging and battery usage for building energy efficiency". In: *Proceedings of the 2014 IEEE/ACM International Conference on Computer-Aided Design*. ICCAD '14. IEEE Press, San Jose, California, pp. 191–196. ISBN: 9781479962778.

Wong, A., T. Bäck, A. V. Kononova, and A. Plaat (2022). *Deep multiagent reinforcement learning: challenges and directions*. arXiv: 2106.15691 [cs.LG].

Zhang, T., X. Wang, B. Liang, and B. Yuan (2023). "Catastrophic interference in reinforcement learning: a solution based on context division and knowledge distillation". *IEEE Transactions on Neural Networks and Learning Systems* **34**:12, pp. 9925–9939. ISSN: 2162-2388. DOI: 10.1109/tnnls.2022.3162241.

Zhang, Z., A. Chong, Y. Pan, C. Zhang, and K. P. Lam (2019). "Whole building energy model for hvac optimal control: a practical framework based on deep reinforcement learning". *Energy and Buildings* **199**, pp. 472–490. ISSN: 0378-7788. DOI: https://doi.org/10.1016/j.enbuild.2019.07.029.

Lund University
Department of Automatic Control
Box 118
SE-221 00 Lund Sweden

Document name
MASTER'S THESIS

Date of issue
June 2024

Document Number
TFRT-6242

Author(s)
Arshad Javeed

Supervisor
Johan Åkesson, Carrier, Sweden
Bo Bernhardsson, Dept. of Automatic Control, Lund University
Pontus Giselsson, Dept. of Automatic Control, Lund University, Sweden (examiner)

Title and subtitle
Distributed Reinforcement Learning for Building Energy Optimization

Abstract
Heating, ventilation, and air-conditioning (HVAC) systems are ubiquitous and are one of the major components responsible for energy consumption in a typical building system. In light of the imminent energy crisis, there is an increasing demand to revisit the building systems and save as much energy as possible. In this regard, the scope of the thesis is to explore opportunities for data-driven optimization of HVAC systems. Traditionally, optimization in HVAC systems has relied on offline optimization requiring domain expertise to schedule a set of optimal controls, but such approaches often require extensive domain expertise. Another drawback is that as the system changes over time, these controls go out of tune and need to be adapted. Thus, data-driven optimization approaches (such as reinforcement learning) appear more appealing due to their ability to adapt online and model and solve complex problems.

A primary objective of the thesis is to carry out exploratory data analysis experiments to quantify the savings potential and expose the optimization space for RL. The main objective is to explore distributed reinforcement learning via multi-agent RL strategies (MARL) and compare and contrast the pros and cons of MARL with single-agent RL. The work benchmarks two of the popular contemporary MARL strategies, centralized training and decentralized execution, and value-mixing approaches, along with proposing two novel MARL enhancements in HVAC systems: a linear value-mixing strategy (inspired by Q-function mixing, QMIX) and turnbased games, that attempt to alleviate some of the problems of multi-agent credit assignment and non-stationarity surrounding MARL.

The experimental results include the learning performance of various RL strategies and the performance benchmarks against the closed-loop controller under realistic conditions. The experimental results reveal that the RL strategies perform significantly better than the closed-loop controller (with a few exceptions), achieving power savings of up to 15% on yearly simulations with live weather profiles. The results also highlight the tradeoffs between optimality and sampling efficiency, further corroborating the prejudice about MARL, where the single-agent RL performs better in terms of optimality, while the MARL approach displays faster learning.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280-5316

ISBN

Language
English

Number of pages
1-108

Recipient's notes

Security classification

http://www.control.lth.se/publications/