A Policy Gradient with Parameter-based Exploration Approach for Zone-heating

Kevin Van Vaerenbergh*, Yann-Michaël De Hauwere*, Bruno Depraetere†, Kristof Van Moffaert*, Ann Nowé*
*Artificial Intelligence Lab, Vrije Universiteit Brussel, Pleinlaan 2, B-1050 Brussels, Belgium
Email: {kevvaere,ydehauwe,kvmoffae,anowe}@vub.ac.be †Flanders' Make, Campus Arenberg, Celestijnenlaan 300 - bus 4027, B-3001 Heverlee, Belgium
Email:bruno.depraetere@flandersmake.be

***Abstract*—Heating a home is an energy consuming task. Most thermostats are programmed to turn on the heating at a particular time in order to reach and maintain a predefined target temperature. A lot of energy is often wasted since most of these thermostats do not take energy consumption into account but are only concerned with reaching the target temperature. In this paper we present a learning approach based on policy gradient with parameter estimations to balance user comfort with energy consumption. Our results show that our approach is capable of offering good trade-off solutions between these objectives.**

I. INTRODUCTION

Most households have some form of HVAC (Heating, Ventilation and Air Conditioning) system installed to control the inside temperature. Whether it is through air conditioning in warmer regions or central heating in colder regions or sometimes both. These systems are typically installed and configured by a technician and many people never change its settings. However, people may change their habits over time due to changes in their commute, children growing up or any other changes in schedule. So what might have been reasonable at the time of installation, may not be suitable anymore after some time. Furthermore, initially misconfigured systems can be energy-inefficient. Finding a good trade-off between user comfort and energy efficiency is not an easy task, especially not when attempting to balance these manually.

Several approaches for dealing with multi-objective problems exist. The most intuïtive way is to use a *scalarisation function* [1] to transform the multi-objective problem to a standard single-objective problem, which can be solved by traditional techniques. This approach is called a *single-policy* mechanism as the algorithm will only converge to a single solution for a particular scalarisation function. In order to find multiple trade-off solutions, one would typically use several scalarisation functions with different weighting parameters and combine the results.

In this paper, we discuss the use of Reinforcement Learning (RL), more specifically Policy Gradient with Parameter-based Exploration (PGPE) with a scalarisation function to guide a Model Predictive Controller (MPC) for a zone heating scenario in which user comfort and energy consumption are to be leveraged and optimised. Reinforcement learning [2] is a machine learning technique that involves an agent operating in a (possibly unknown) environment and receiving a scalar feedback signal for its behaviour. By sampling actions and observing the feedback signal the agent adjusts its estimate of the quality of an action.

The remainder of this paper is organised as follows. In Section II we introduce the necessary background information for our approach. We discuss our approach and experimental setup in Section III together with other work that has been performed in this domain. Next, we present the results of our approach in Section IV. Finally, we conclude this paper with a summary and possible avenues for future research.

II. BACKGROUND ON LEARNING APPROACH

A. Reinforcement Learning (RL)

RL problems [2] are a class of machine learning problems where an agent must learn to interact with an unknown environment, using a "trial and error" approach. At a given time-step t, the agent may execute one of a set of *actions* $a \in \mathcal{A}$, possibly causing the environment to change its *state* $s \in \mathcal{S}$, and generate a (scalar) *reward* $r \in \mathbb{R}$. Both state and action spaces can be multidimensional, continuous or discrete. An agent's behaviour is represented by its *policy*, mapping states to actions. The aim of a RL algorithm is to optimise the policy, maximising the reward accumulated by the agent. Simply stated, RL consists in learning from a teacher (the environment) who cannot tell us *what* to do next (the optimal policy), but only *how good* we are doing so far (the reward signal). It therefore offers a suitable tool for controlling systems with limited feedback information: the target state at the sensor location(s) can be incorporated in the reward signal, favouring the desired behaviour.

Two main families of RL approaches can be distinguished. In *value based* methods, such as Q-learning, the maximum expected future reward $Q^*(s, a)$ obtained by taking action a in state s is estimated: the policy consists of selecting the action a which maximises Q^* in the current state s. This can be done by storing and updating estimates in tabular form (which is often used in discrete state spaces), or by using function approximators if $\mathcal{A}$ and $\mathcal{S}$ are too big, or continuous. In *direct policy search*, the space of policies is searched directly in an attempt to maximise the reward: for example, in Policy Gradients (PG) methods [3], the policy is

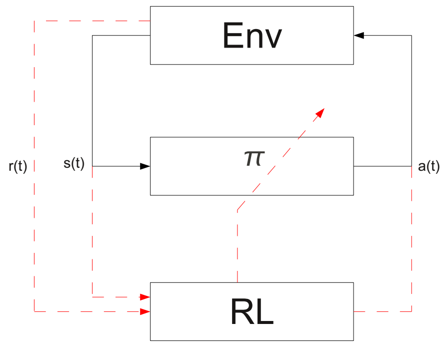

Fig. 1: Block representation of RL. Env: environment being controlled. Π: policy (controller), generating the action (control signal) $a(t)$ given the current state of the system $s(t)$; RL: learning algorithm, adapting the policy based on state, action, and reward signals. Continuous lines indicate online interactions, while discontinuous lines refer to the learning process, which can be online or offline, depending on the particular algorithm.

represented as a parametric probability distribution over the action space, whose parameters are updated following a Monte Carlo estimate of the expected reward.

In both value based and policy based methods, learning is done using a sequence of *epochs*, each consisting of one or multiple interactions with the environment.

In this paper, we apply an existing variant of the Policy Gradient (PG) method, called PGPE[4]. In this approach, the parameters of a controller are adapted based on the return collected during the whole epoch, regardless of the trajectory in the state space. In the remainder of this section we briefly describe PGPE, referring the reader to [4], [5] for further details.

B. PG with parameter exploration (PGPE)

In PG methods, the policy is represented as a parametric probability distribution over the action space, conditioned on the current state of the environment. Epochs are subdivided into discrete time-steps: at every step, an action is randomly drawn from the distribution, conditioned by the current state, and executed in the environment, which updates its state accordingly. After an epoch has been completed, the parameters of the policy are updated, following a Monte Carlo estimate of the expected cumulative (discounted) reward.

A major disadvantage of PG methods is that drawing a random action at every time-step may result in noisy control signals ([6]), as well as noisy gradient estimates. Moreover, the policy is required to be differentiable w.r.t. its parameters.

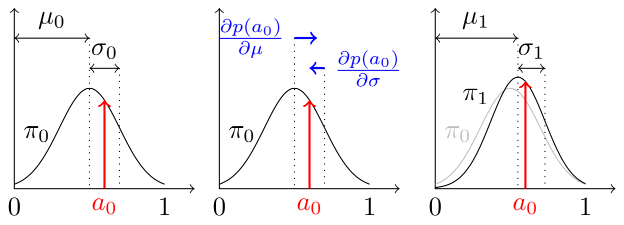

Fig. 2: A simple example illustrating the effect of one step of PGPE, with no state information and single stage epochs ($T = 1$). A single policy parameter $\mathcal{A} = [0, 1]$ is sampled from a Gaussian prior π, with $\boldsymbol{\theta} = (\mu, \sigma)$. *Left:* the first epoch is executed, drawing a parameter value $a_0 \sim \pi_0(a)$, and observing a return R_0. *Center:* as $R_0 > b$, following the gradient (1) increases $\pi(a_0)$. *Right:* updated prior π_1, ready for the next epoch.

To overcome these issues, PGPE was introduced [7], [4]. In this method, the random sampling and policy evaluation steps are, in a sense, "inverted": the policy is a parametric function, not necessarily differentiable, therefore it can be an arbitrary parametric controller; the parameter value to be used is sampled at the beginning of each epoch from a Gaussian distribution, whose parameters are in turn updated at the end of the epoch, again following a Monte Carlo estimate of the gradient of the expected return. In other words, rather than searching the parametric policy space directly, PGPE performs a search in a "meta-parameter" space whose points correspond to probability distributions over the parametric policy space.

To simplify notation, we consider a parametric policy f_a with a scalar parameter a. Be $\boldsymbol{\alpha} = (\mu, \sigma)$ the meta-parameter defining the Gaussian distribution $p_{\boldsymbol{\alpha}}(a)$ over parameter values. The index we intend to maximise is the expected value of the return R given a, $J = E\{R|a\}$. The gradient of this expected return J with respect to the meta-parameter $\boldsymbol{\alpha}$ can be estimated as follows (see [4] for details):

$$\nabla_{\boldsymbol{\alpha}} J \approx \frac{1}{N} \sum_{n=1}^{N} \nabla_{\boldsymbol{\alpha}} \log p_{\boldsymbol{\alpha}}(a^n)(R^n - b), \tag{1}$$

where $\boldsymbol{\theta}^n$ is the parameter used at the n-th of the N epochs considered (typically $N = 1$), and b is a *baseline* return, which, in the simplest case, is the average return observed so far. Fig. 2 illustrates the effect of a single step of PGPE.

C. Multi-Objective Optimisation

In a multi-objective optimisation (MOO) ([8]) problem we want to optimise a vector function with each element representing an objective. When wanting to maximise the problem, we maximise the function $\mathbf{F}(\mathbf{x}) = \max\{\mathbf{f^1}(\mathbf{x}), \mathbf{f^2}(\mathbf{x}), ..., \mathbf{f^m}(\mathbf{x})\}$ where m represents the number op objectives and f^i is the value of the i-th objective. A Pareto dominant solution x_1, dominates another solution x_2 iff there exists one objective i, for which $f^i(x_2) < f^i(x_1)$ and $f^j(x_2) \leq f^j(x_1)$ is valid for all other objectives. The collection of non-dominated, optimal solutions is referred to a the *Pareto front*. When optimising a solution in a multi-objective environment, conflicts can arise while optimising the objectives simultaneously. In such case a trade-off between the different objectives must be learned.

Multi-Objective Reinforcement Learning

Different from single-objective learning, multi-objective learning consists of an objective space with more then one dimension. Therefore the return of such a problem provides a vector of rewards:

$$\mathbf{R}(s_i, a_i) = (R_1(s_i, a_i), \dots R_m(s_i, a_i)) \tag{2}$$

where m represents the number of objectives. A solution for a multi-objective problem is a policy π, evaluated using the expected return $\mathbf{J}^\pi$:

$$\mathbf{J}^\pi \equiv \left[E\left[\sum_{t=0}^{\infty} \gamma^t R_1(s_t, \pi(s_t)) \right], \dots, E\left[\sum_{t=0}^{\infty} \gamma^t R_m(s_t, \pi(s_t)) \right] \right] \tag{3}$$

In general, it is a vector with an expected return for each objective.

In a multi-objective environment, a solution π_1 can only dominate another policy π_2 when the reward is greater for at least one objective and not strictly less for every other objective in π_1. Two solutions are incomparable when they each improve the solution in one objective where the other solution does not do better in the same objective.

Scalarised MORL

Most work in multi-objective reinforcement learning uses *scalarisation* functions [9] to reduce the dimensionality of the multi-objective environment. A linear scalarisation function is often used to reduce the objective space, where the main focus can be set by appointing a higher weight value to the objective that is of greater importance. This gives the user some control over the solution. Applying a scalarisation function s on a m-dimensional reward vector $\mathbf{V}$ using a weight vector w, a value V_w is calculated using the following:

Definition A scalarisation function s is a function that projects a vector $\mathbf{V}$ to a scalar

$$V_{\mathbf{w}} = s(\mathbf{V}, \mathbf{w}) \tag{4}$$

where $\mathbf{w}$ is a weight vector parametrising s.

III. Approach

A. Goals and objectives

The main focus of this research lies on adaptability of HVAC systems to user demand patterns and user preferences, i.e. tailoring of the performance of these systems to the specific circumstances imposed on them by their everyday use. By taking into account patterns in user behaviour and expectations, system usage can be optimised, both in the service provided by the system to the end user, and the resources needed to keep the system running. In this paper we focus specifically on thermostats for zone heating in households. As already mentioned, these systems are typically installed by technicians who adopt a kind of default setting which will suit most people. In general, several parameters have to be configured: the desired target temperature, the time when the heating should begin, the time when the heating can be turned off. Using incorrect values for these parameters will result in either energy loss when the system is heating when no user is present, or when a predefined target temperature was programmed that is too high, or when the heating is turned on too late, all will result in user discomfort. Our goal is to optimise these parameters based on the actual user behaviour and his preferences. This means that we allow our learning approach to decrease/increase the time when the system should start heating and decrease/increase the target temperatures. At the end of this section, we illustrate in more detail how this is achieved with PGPE and how these two objectives are leveraged to obtain a policy which aims at satisfying the preferences of the user and decreasing the energy consumption. A secondary goal is to minimise the cost of learning. This means that we will try to disturb the user of the system as little as possible by trying to stay in the predefined comfort zone of the user.

B. Related work

Zone heating is not a recent domain of research and has already been investigated for decades to attempt to improve basic thermostats in which a typical on/off schedule is programmed. Two main tracks of research can be identified.

First, a lot of research has been performed on training controllers for households such that a target temperature can be reached by a certain time. Since a thermostat does not have access to a full thermal model of the house, which includes insulation, orientation and windows, ... , reaching these targets is a difficult task. Several approaches exist based on genetic algorithms [10], fuzzy PD-control [11], model predictive control [12] and reinforcement learning [13]. More recently, many companies have jumped on this zone heating track by releasing so-called smart thermostats, that optimise the moment in time when the heating system should have reached a particular temperature. The definition of smart, however, is ambiguous. In many of these systems, the concept of smart indicates the ability to control the heating system from a remote location and manually ensure that energy can be conserved. Other systems will use presence sensors and some learn to achieve the desired temperature by the time the user arrives. This usually comes at a cost in terms of energy efficiency and sometimes user comfort if no user walks in front of the presence sensors for some time. Our research is located in this second track of research, where we will optimise both target time and target temperature while actively try to optimise energy consumption.

C. Approach

Contrary to [13], reinforcement learning can also be applied as global learner that defines a control policy that is used by an underlying controller. For this simulation we used a model predictive controller (MPC) as this underlying controller. The defined control policy is a set of target temperatures that must be reached at a certain point in time, and these target temperatures are passed onto the MPC controller. Based on these values, it determines the necessary control actions to ensure the control policy is applied correctly. .

MPC is a model-based control strategy [14]. It relies on a dynamic thermal model of the room under control, including the heater elements and any thermal interactions with the outside (losses for example). By using this model, the future behaviour for any set of control signals can be predicted. This is then used to solve a numerical optimisation, in which the set of control signals is looked for that optimises a criterion of the predicted values (cost function), such as the difference between the target temperature and the realised one, while also taking into account any system limitations (equality or inequality constraints). As a result, the main advantage of MPC is that it takes future time-steps into account when optimising the current control value. This then allows to for example anticipate on a future requested temperature increase, and to already heat up ahead of time, thus reducing discomfort. This is an important advantage not available to many alternative control methods such as PID control, on/off control and deadband control [15], [16], [17], and will become more and more relevant due to more energy efficient houses being built with smaller heater powers, making it needed to anticipate further ahead. Another advantage is that MPC offers a very flexible framework. We can easily add more terms to the cost function to for example include penalties for when the temperature is too high, or add thermal comfort constraints that ensure the temperature is never more than $5 \circ C$ below the reference temperature. Another extension that could be applied would be to solve the problem in a stochastic sense, for example optimising the control given a 50% probability of temperature reference 1 being applicable and a 50% probability of reference 2.

Even though MPC relies on a dynamic thermal model, it is generally not sensitive to the quality of the model. This can be explained by the fact that, after optimisation, only the first value of the calculated control sequence is applied, the system's output is recorded, and the optimisation problem is resolved, now looking ahead one step further. Because this problem is recalculated at every step, and because it is always initialised to match the currently observed system output, this approach becomes capable of reacting to observed differences between model and plant, thus introducing a type of feedback, and thus reducing the sensitivity to model-plant mismatch [14]. Better models are desirable, as these will lead to more accurate predictions and thus better results, but for zone-heating this is not a real issue, since the overall behaviour can be modelled relying on linear models, which can easily be identified experimentally during normal machine operation.

As global controller we use the policy gradient with parameter-based exploration algorithm to learn when the thermostat should start heating the house and what its target temperature should be, such that the user is satisfied with both the comfort level of the environment as well as the power consumption required to achieve this comfort level. The RL algorithm uses a policy that is adjusted during the experiments. The initial policy is a reference trajectory generated based on different temperature/time-step pairs. Every pair is a temperature that the controller has to reach at a certain time-step. Using these pairs, we generate a trajectory which has a target temperature for every time-step (60 seconds). To facilitate the learning process, the whole trajectory is represented by 9 different temperature/time-step pairs. These 9 pairs represent the important temperatures or time shifts in the trajectory. These points represent the policy that our PGPE algorithm will adjust to achieve the best trajectory given a certain trade-off between personal comfort and energy efficiency.

To generate a trajectory, 9 target temperatures and 9 time-steps must be chosen. Choosing a value for these parameters is done by sampling a value from a Gaussian distribution over each parameter. At the start of every experiment we have to define initial value μ and the variance σ, for each parameter. When the user does an override, the search space is reduced by removing the part of the search space that does not respect the comfort of the user. i.e.: it the user thinks it is to hot, an override will tell the algorithm that future action, higher then the current chosen one will result in an override. Therefor these actions are dismissed from the future search space.

D. Experimental setup

We tested our approach on a Matlab simulation of a small office with one person working on his computer. The goal is to heat up the office respecting the trade-off chosen between energy efficiency and user comfort. The person simulated is a standard user who will manually adjust the current target temperature when it is $2°C$ to warm or to cold. This manual override will result in an immediate change to the wanted temperature. When an override occurs, the reward for that trajectory will be severely punished.

At the start of each day, the PGPE algorithms samples a value from the distribution representing the 9 target temperatures that must be reached. A full reference trajectory is generated for every time-step (60 seconds) with a target temperature. We use a MPCl controller to achieve those temperatures. When a day is simulated, we calculate a reward based on the energy consumption and the difference between the wanted temperature and the target temperature.

The Gaussian distribution over the target temperatures results in an acceptable adjustment of maximum $2°C$ colder or warmer. The distribution over the time-step varies between 20 minutes less or more.

The experiments done for this paper use a stateless PGPE algorithm without symmetric sampling, a learning rate of 0.8 and an initial σ of 0.8.

IV. Results

We experimented with two different initial trajectories; a cold and a warm trajectory. This means that the initial trajectory is too cold or too warm to be in the comfort zone of the simulated person.

In both cases, we compare the learned trajectory after 200 epochs with different weights while also comparing the energy profit with a standard manual thermostat setting. This standard manual setting refers to a thermostat that starts heating when the person is at home (manual enabling) or when the thermostat is heating a fixed amount of minutes in advance. Each run consists of sampling a trajectory, applying it for the whole day and gathering a reward based on the energy consumed in a day and the different between the wanted temperature and the room temperature. For the energy consumed we sum the wattage over a day. The comfort is calculated only when the person is in the room.

In Fig. 3 we see the evolution from the initial trajectory, which in this case is to cold, to the learned trajectories for the different trade-offs between comfort and energy efficiency. We compare three different weight settings; comfort 0.9 and energy 0.1, comfort 0.5 and energy 0.5 and comfort 0.1 and energy 0.9. Each weight selection converges to a different reference trajectory based on the focus of the reward function. When focussing on energy consumption, the converged trajectory finds the lowest temperature that is still in the comfort zone of the user. In this environment, the comfort zone is $1°C$ more or less then the wanted temperature. If we focus on comfort we need to converge to a reference trajectory that perfectly follows the wanted temperature when the user is present. Logically when choosing for a divide focus on comfort and energy efficiency, we want the learned trajectory to find a good trade-off between reaching a perfect comfort and decreasing energy consumption were possible.

Fig. 4 compares the different trajectories found by the reinforcement learning algorithm when starting from a reference trajectory that is to warm for the user. We compare using the same weight settings as the previous experiment and can easily conclude that even when starting from a warm trajectory, we still find a good trajectory that fit the different user comfort and energy efficiency trade-offs.

In table I we compare the average energy consumption for different settings. The fixed begin will always start heating $1h$ in advance to reach the target temperature before the user is supposed to be home. The manual setting will start heating the room when the user is at home. These two settings are compared with the learned trajectories for the three differente weight settings. When using the fixed begin thermostat setting

TABLE I: Table with a comparison of the consumed energy (in Watt) for different focusses (energy, comfort and $50/50$) and standard settings (Start heating $1h$ in advance and reference temperature set when user is home).

Ref. Traj.	Energy
Fixed begin	466370W
Manual set	451090W
Focus energy	430780W
Focus comfort	449710W
$50/50$	432530W

we achieve a perfect comfort when we start heating $1h$ in advance but consume a lot of energy. Manually setting the thermostat at the wanted temperature when the user is home results in a lower energy consumption then the fixed one but still higher then the average of the learned trajectories. There is also a difference between the learned trajectories. Obviously the focus has a big impact on the energy consumption. Up to $30000W$ can be saved when the user is focussing on energy efficiency.

These experiments show us that we can converge to a good reference trajectory after 200 days of simulating which in real life means that we have to learn for more then half a year. We cannot ask the user to provide feedback during the whole learning stage this is why the number of overrides need to reach zero as fast as possible. In Figures. 5 and 6 we notice that the number of overrides decreases very fast for each learning process. Only during the first few days (30) we have some overrides but this decreases very fast to reach zero. When we focus on energy efficiency we see that a sporadic override still occurs later because we want to minimise energy consumption and therefor flirting with the limits of the comfort zone.

When learning a reference trajectory fitted for the needs of the user, we do not want to bother the user to much as such that he does not get annoyed by the system. Therefor we added a little domain knowledge that uses the user's override and discards all the samples which will lead to the same override. When we look at the different experiments, we noticed that after 2 months the algorithm is stable enough to learn while not interrupting the user and staying in his comfort zone. This algorithm will continue learning until it reaches the best possible reference trajectory or until the user's behaviour changes.

V. Discussion

In most cases, a programmable thermostat will be able to reach a given target temperature that is set by the end user. To enjoy the comfort of a pre-heated room at a given temperature, the user must program the thermostat to start heating at a fixed amount of time before he is presumed to be at home. This amount will vary given the exterior temperature and the inertia of the home. We introduce a learning aspect to the thermostat such that the user can easily choose between a comfort and energy consumption. This trade-off will influence the learning process of our algorithm and will result in a reference trajectory that can be used by different controllers.

When heating a room we rely on user feedback to indicate the level of comfort the user is willing to accept. In this simulated environment, we set the comfort level at $+1$ and $-1°C$ of his wanted temperature. This can vary between users, causing a larger or smaller gain in energy consumption when focussing on energy efficiency.

VI. Summary

In this paper, we introduced a learning aspect to the zone-heating case to reach a reference trajectory for heating a room given a trade-off between user comfort and energy efficiency. We compared different trade-offs between energy efficiency and comfort using a certain weight value in a weighted sum over the two reward (one for the comfort and one for the energy used). We noticed that when focusing on energy, the learned trajectory will reach the comfort limit to save energy avoiding the manual override of the user. Logically, when focusing on

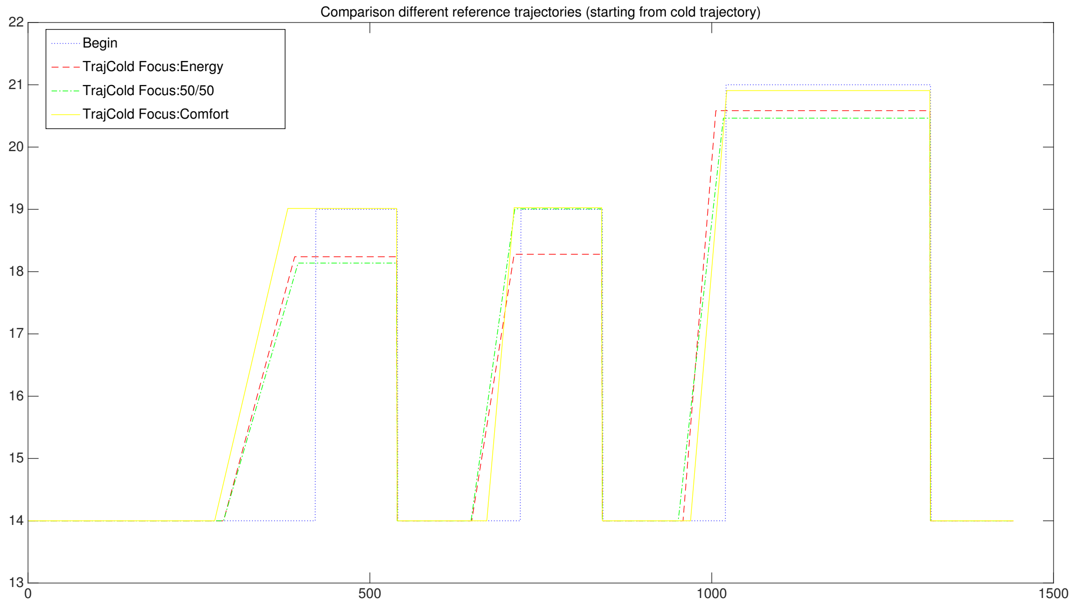

Fig. 3: Comparison of learned trajectories w.r.t. different weight settings.

comfort, a trajectory is learned that fully satisfies the users comfort, disregarding the energy usage.

To speed up the learning process we added domain knowledge as such that we easily prune the search space and quickly minimise the users overrides. After a few runs, the number of overrides decrease to zero and the algorithm will continue learning without disturbing the user.

VII. Future work

For future work we want to expand our learning to take into account a series of days that vary in outside temperature and user comfort. When simulating a week with 7 different days w.r.t. the outside temperature and when the user demands a different comfort for weekdays and weekends, we cannot focus on one reference trajectory. Therefor we will extend our approach to use several trajectories, each with the probability of a certain day of the week.

Currently we are only adjusting the temperature after an override, while if the user does an override before we reach a certain target temperature we can limit the search space of the targets time to earlier points in time.

Acknowledgements

This research is supported by IWT-SBO project PERPETUAL. (grant nr. 110041).

References

[1] K. Miettinen and M. M. Mkel, "On scalarizing functions in multi-objective optimization," *OR Spectrum*, vol. 24, pp. 193–213, 2002, 10.1007/s00291-001-0092-9.

[2] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*, ser. Adaptive Computation and Machine Learning. Mit Press, 1998.

[3] J. Peters and S. Schaal, "Policy gradient methods for robotics," in *Proceedings of the IEEE Intl. Conf. on Intelligent Robotics Systems (IROS)*, 2006.

[4] F. Sehnke, C. Osendorfer, T. Rückstieß, A. Graves, J. Peters, and J. Schmidhuber, "Parameter-exploring policy gradients," *Neural Networks*, vol. 23, no. 4, pp. 551–559, 2010.

[5] M. Gagliolo, K. Van Vaerenbergh, A. Rodriguez, A. Nowé, S. Goossens, G. Pinte, and W. Symens, "Policy search reinforcement learning for automatic wet clutch engagement," in *15th International Conference on System Theory, Control and Computing — ICSTCC 2011*. IEEE, 2011, pp. 250–255.

[6] R. Munos and M. Littman, "Policy gradient in continuous time," *Journal of Machine Learning Research*, vol. 7, pp. 771–791, 2006.

[7] F. Sehnke, C. Osendorfer, T. Rückstiess, A. Graves, J. Peters, and J. Schmidhuber, "Policy gradients with Parameter-Based exploration for control," in *ICANN '08: Proceedings of the 18th international conference on Artificial Neural Networks, Part I*. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 387–396.

[8] D. M. Roijers, P. Vamplew, S. Whiteson, and R. Dazeley, "A survey of multi-objective sequential decision-making." *J. Artif. Intell. Res. (JAIR)*, vol. 48, pp. 67–113, 2013.

[9] P. Vamplew, J. Yearwood, R. Dazeley, and A. Berry, "On the limitations of scalarisation for multi-objective reinforcement learning of pareto fronts," in *Proceedings of the 21st Australasian Joint Conference on Artificial Intelligence: Advances in Artificial Intelligence*, ser. AI '08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 372–378.

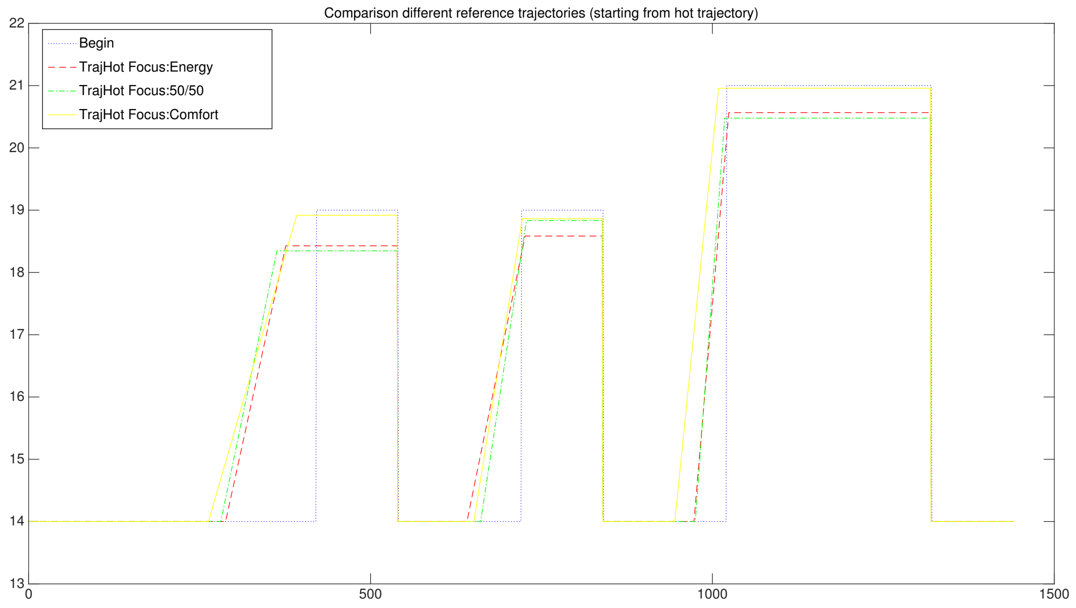

Fig. 4: Comparison of learned trajectories w.r.t. different weight settings.

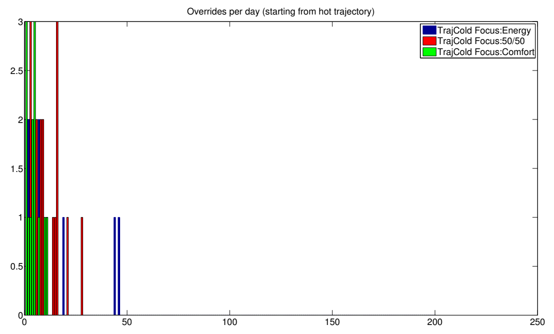

Fig. 5: Overrides per day starting from cold trajectory.

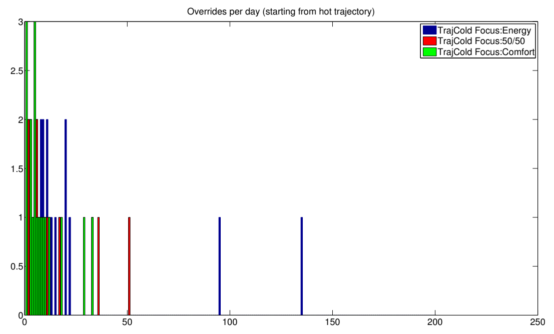

Fig. 6: Overrides per day starting from hot trajectory.

[10] S. Dickinson and A. Bradshaw, "Genetic algorithm optimisation and scheduling for building heating systems," in *Genetic Algorithms in Engineering Systems: Innovations and Applications, 1995. GALESIA. First International Conference on (Conf. Publ. No. 414)*, Sep 1995, pp. 106–111.

[11] P. Salgado, J. Cunha, and C. Couto, "A computer-based fuzzy temperature controller for environmental chambers," in *Industrial Electronics, 1997. ISIE '97., Proceedings of the IEEE International Symposium on*, Jul 1997, pp. 1151–1156 vol.3.

[12] A. Afram and F. Janabi-Sharifi, "Theory and applications of {HVAC} control systems ? a review of model predictive control (mpc)," *Building and Environment*, vol. 72, no. 0, pp. 343 – 355, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0360132313003363

[13] K. Dalamagkidis, D. Kolokotsa, K. Kalaitzakis, and G. Stavrakakis, "Reinforcement learning for energy conservation and comfort in buildings," *Building and Environment*, vol. 42, no. 7, pp. 2686 – 2698, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0360132306001880

[14] C. E. Garcia, D. M. Prett, and M. Morari, "Model predictive control: Theory and practice?a survey," *Automatica*, vol. 25, no. 3, pp. 335 – 348, 1989.

[15] G. Geng and G. Geary, "On performance and tuning of pid controllers in hvac systems," in *Control Applications, 1993., Second IEEE Conference on*, Sep 1993, pp. 819–824 vol.2.

[16] M. Anderson, M. Buehner, P. Young, D. Hittle, C. Anderson, J. Tu, and D. Hodgson, "Mimo robust control for hvac systems," *Control Systems Technology, IEEE Transactions on*, vol. 16, no. 3, pp. 475–483, May

2008.

[17] J. Bai and X. Zhang, “A new adaptive {PI} controller and its application in {HVAC} systems,” *Energy Conversion and Management*, vol. 48, no. 4, pp. 1043 – 1054, 2007.