Comparison of optimizers for model predictive thermal control of buildings

Andersen, Torben

Published in:
Energy and AI

DOI:
10.1016/j.egyai.2023.100332

2024

Document Version:
Publisher's PDF, also known as Version of record

Energy and AI 15 (2024) 100332

Contents lists available at ScienceDirect

Energy and AI

journal homepage: www.elsevier.com/locate/egyai

Comparison of optimizers for model predictive thermal control of buildings

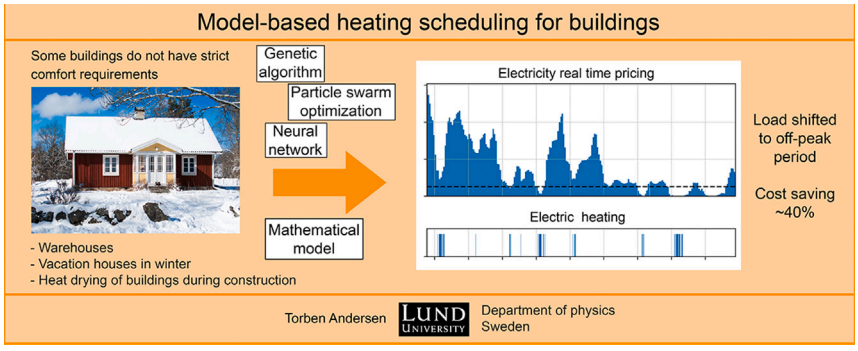
Torben Andersen

Department of physics, Lund University, Professorsgatan 1, Lund, 226 63, Sweden

HIGHLIGHTS

- Electric HVAC scheduling for cost reduction and load shift to off-peak hours.
- Model predictive control with genetic algorithm, particle swarm optimization or neural network.
- Warehouses, vacation homes and heat drying of buildings under construction.
- Confirmed by simulation and real-time, full-scale experiment.

GRAPHICAL ABSTRACT



ARTICLE INFO

Keywords:
Home energy management system
Demand response
Genetic algorithm
Particle swarm optimization
Neural network
Model predictive control

ABSTRACT

Considering recent developments in the energy sector, further reduction of electricity cost and flattening of the electric power demand curve are needed. We have focused on an autonomous electric heater control system that can easily be implemented in existing buildings without strict comfort requirements. Examples are winter heating of warehouses and vacation homes, and heat drying of buildings under construction. We have set up a system that typically reduces electricity cost by about 40% on the basis of automatic weather and real time pricing forecasts. The system uses the building as an energy reservoir over periods with high electricity cost. Using a model predictive control system, we compare use of a genetic algorithm, a particle swarm optimization, and a neural network for heater control, all working in a closed loop to reduce the influence of modeling errors. We have simulated the performance of the systems using realistic data and found that all three optimizers give about the same performance, varying only a few percent in efficiency. However, the computational and memory requirements of the neural network are much lower than for the other optimizers, so it is preferable for use with inexpensive microcontrollers. We carried out a full-scale experiment at a residential house and found agreement with simulation results.

1. Introduction

In an electric power grid, the produced power must match the consumed power. In addition to adjusting the power injected into the grid, in the future, sophisticated options will be available for controlling power consumption on the demand side in combination with large-scale power storage facilities [1–3]. However, at the moment, adjusting real time electricity prices is the dominating demand response control tool [4], giving customers a strong incentive to shift power consumption to times with low electricity cost.

Due to the present shortage of electric power and the need to flatten the power demand curve, we have focused on short-term solutions that, in existing buildings, can readily be implemented with simple hardware, such as small and inexpensive microcontrollers. We study the situation, where electric heating can store energy in the form of heat in a building at times, when electricity is in surplus and cost is low. For conventional residential buildings, the saving potential with this approach is limited to 5–10% [5], because overheating a building will

E-mail address: torben.andersen@astro.lu.se.

https://doi.org/10.1016/j.egyai.2023.100332
Received 5 June 2023; Received in revised form 28 November 2023; Accepted 20 December 2023
Available online 22 December 2023

be uncomfortable for its users. However, there is a group of buildings for which a temporary overheating can be accepted:

- Industrial warehouses and workshops
- Second (vacation) homes.
- Air drying of buildings under construction

In Sweden alone, there are almost 600 000 vacation houses and well over half of these are assumed to be heated electrically during the winter to keep them dry, and to avoid freezing of water systems. The power consumption for these houses may be about 250 GWh for each of the three winter months. A saving of, say, 40%, with an average electricity cost estimate of about 4 Swedish Crowns (SEK) per kWh, would lead to a total saving of the order of 1.2 billion SEK for consumers, and also contribute to a much needed shift of the consumption profile for the grid. We do not have data for the saving potential for industrial warehouses and workshops, but it seems reasonable to assume that the saving potential for those is of the same order of magnitude.

Such a system will be most attractive for buildings that are well-insulated. The electric heating appliances may be simple resistive heating elements, heat pumps, or air blast heaters, all of which take electric power from the grid as their main energy source, and are easy to control with existing hardware and technology. Hence, for electric Heating, Ventilation, and Air Conditioning (HVAC), we have studied an autonomous Energy Management System (EMS), which is based on weather forecasts and electricity real time pricing 24 h ahead.

Recently, two types of electricity optimization approaches have drawn interest for residential dwellings:

- *Controllers based on a neural networks with reinforcement learning*. This approach uses a neural network to control HVACs. It is often not straightforward to prepare training datasets for supervised training of a neural network, because of lack of a "ground truth" for a heating schedule based upon the forecasts. Hence, several teams have turned to reinforced learning, where the system trains itself during operation [5–7]. Using an "agent" to ad-hoc train the neural network, the system at the same time controls the HVAC and fulfills the temperature requirements. It is an advantage that the system does not need prior knowledge of the building and heating system. It learns over time how to control the heaters and how the occupants behave. Disadvantages are that it may take some time before the system will have tuned itself, and it may not handle abrupt changes in operating conditions or electricity cost well. Also, it is sensitive to external heat sinks or heat sources, for instance if an unknown heater is turned on by house residents.
- *Model predictive control (MPC)*. Such a system controls the HVAC on the basis of optimizations. It determines future system performance by running a mathematical model of the real system [8]. Such a model can be precise but requires some prior knowledge of model parameters.

We here focus on the second alternative. A control system of this type is easy to retrofit, using existing hardware and technology. Some work in the field has apparently been done by private actors and power companies to assist consumers, but little has been published on the subject, because the information typically is proprietary. Hence, the objectives of this article are to present and compare three different schemes for model predictive control of the buildings described, and, in particular, to demonstrate that a very shallow neural network can fulfill the optimization requirements.

We then study and compare the following three optimization schemes for heating control, all using the building itself as a heat reservoir during periods with low electricity cost:

- Genetic algorithm
- Particle swarm optimization
- Neural network

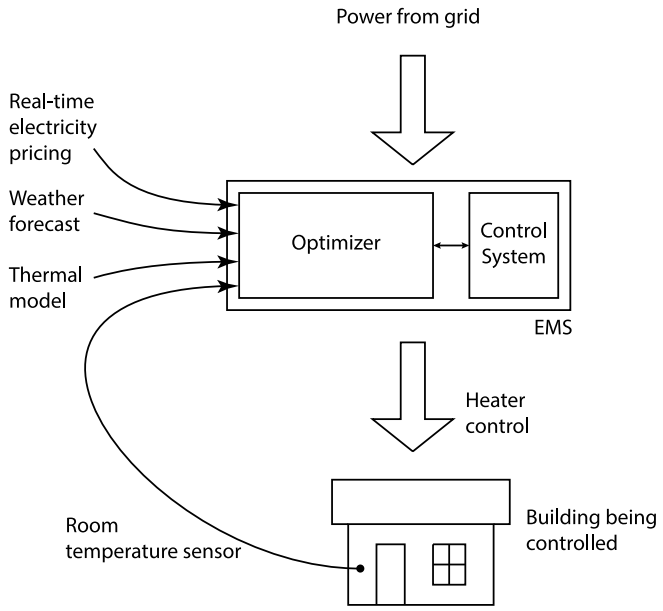


Fig. 1. Principle of the proposed energy management system.

The proposed system relies on retrieval of future weather and price data through communication with external Application Programming Interfaces (APIs). The system runs an optimization at regular intervals, typically every hour.

Our EMS can be readily established on the consumer side in existing buildings with only little additional hard- or software, leading to significant electricity cost savings. We here focus on heating systems, since air conditioning is generally not needed in residential buildings in Sweden, but the principles apply equally well to cooling. We document the correct function of the system and the simulation model by a full-scale experiment on a residential building using the genetic algorithm for optimization.

2. Methodology

The principle of the proposed EMS is shown in Fig. 1. A control system retrieves electricity real-time pricing and weather forecasts 24 h ahead and determines the best heating strategy. Real-time pricing for the next 24 h can typically be retrieved automatically from an API of the local electric power company, and weather forecasts for 24 h (or more) from a meteorological institute. The weather forecast is updated every hour, and the real time pricing every 24 h. The optimization is done every hour and the system operates in a closed loop mode, because the present building temperature is fed back to the control system to correct for any building temperature errors.

The task is, every hour, to estimate the "best" heater values for each of the coming 24 h. The heater values are real numbers that are constrained to an interval from zero to the max capacity of the heater. For any choice of heater values, we apply an objective function to rate the usefulness of the choice. Evaluation of the objective function involves two parts:

- A simulation with a thermal model to predict a time history of the building temperature.
- Evaluation of the result by finding the value of the objective function.

As is common practice for MPC-systems, only the very first predicted heater value will be used at any particular moment, even though all heater values for the next 24 h are predicted as a part of the optimization.

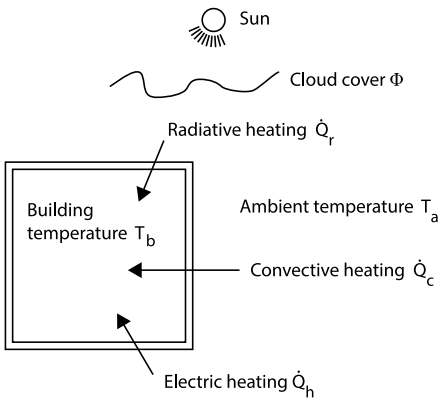


Fig. 2. Thermal model of building.

3. Methods

With an optimizer, the task is then to determine optimal heater values for each of the next 24 h on the basis of the objective function. In the following, we first present a thermal model and an objective function, which underlie all three optimization schemes mentioned above. Next, we present the individual optimizers.

3.1. Thermal model

The building can be seen as a single-input-single-output system with heating as input and temperature as output. We use the thermal model shown in Fig. 2. A more accurate model could include several heat reservoirs, i.e. have several time constants (real poles in the left half plane). However, because the thermal model is used in a closed loop (see Fig. 1), it is here permissible to approximate the thermal performance of the building with that of a single heat reservoir. Minor modeling errors will anyway be corrected by closed-loop control. The same is true for the influence of wind, which is not included in the model.

Referring to Fig. 2, the corresponding differential equation simply becomes

$$\frac{dT_b}{dt} = \frac{1}{C}\left(h_{wall}\left(T_a - T_b\right) + h_{rad}\frac{(8-\Phi)}{8}\gamma + \dot{Q}_h\right), \tag{1}$$

where h_{wall} is a proportionality factor defining the effect of convection and conduction through the building walls, h_{rad} is a proportionality factor defining heating from solar influx through windows, Φ is the dimensionless meteorological cloud cover value scaled from 0 to 8, γ is a daylight factor, C is the heat capacity of the inner building, t is time, and the other symbols are defined in Fig. 2. On Laplace form, the transfer function then is that of a regular time constant:

$$\frac{\mathcal{L}\{T_b\}}{\mathcal{L}\{\dot{Q}_{ext}\}} = \frac{1/h_{wall}}{1+\tau s}, \tag{2}$$

with

$$\dot{Q}_{ext} = h_{wall}T_a + h_{rad}\frac{(8-\Phi)}{8}\gamma + \dot{Q}_h \tag{3}$$

and where s is the Laplace operator, $\mathcal{L}\{T_b\}$ and $\mathcal{L}\{\dot{Q}_{ext}\}$ are the Laplace transforms of T_b and $\dot{Q}_{ext}$, and the time constant is $\tau = C/h_{wall}$.

As already mentioned, selection of the values for the various parameters is not critical. The parameter τ is the most important parameter for the cost optimization. A first estimate of τ can be taken from step responses and typically has a value of 1–2 days. Hourly values of γ depend on the time of the year and could be taken as 1 from one hour after Sunrise until 1 h before Sunset (depending on window locations), and then otherwise 0. The value of h_{wall} can be estimated rather well from the assumption that the heating system is capable of keeping an indoor comfort temperature with an external temperature down to, say, −25 °C.

3.2. Objective function

The objective function provides a figure of merit for a certain choice of heating values over a given period. Whenever real time pricing is available for the electricity, we use a period of 24 h, but during the beginning of a day, 24-h real-time pricing may not yet be available, and we then work with a somewhat shorter look-ahead.

The objective of our heat regulator is to keep the building temperature close to a specific value, however allowing the temperature occasionally to rise higher to provide energy storage. With a simulated building temperature time history arranged as a time series of length n, the *temperature* objective function, P_t, is computed as

$$P_t = \frac{1}{n}\sum_{i=1}^{n} p_i, \text{ where} \tag{4}$$

$$p_i = \begin{cases} \left(T_s - T_{b,i}\right)^{\lambda} & \text{if } T_{b,i} \le T_s \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

Here $T_{b,i}$ is the building temperature with time index i, and T_s the setpoint defining the desired building temperature. For our work, we have set the exponent γ to the value 3, which provides a significant penalty, when building temperatures are low. This is important for many applications to avoid that the building temperature gets below the freezing point, thereby damaging installations.

The *cost* objective function is simply the total cost of electricity over 24 h:

$$P_c = \sum_{i=1}^{n} c_i \dot{Q}_{h,i}\Delta t, \tag{6}$$

where c_i is the cost per electric energy (i.e. per kWh) and $\dot{Q}_{h,i}$ is the electric heating power, both with time index i, and Δt the time series sampling interval (1 h). We then set the total objective function, P, to

$$P = \alpha P_t + \beta P_c \tag{7}$$

where α and β are user selected weights defining the relative importance of temperature control and cost. For all work described in this article, we used the values $\alpha = 1\,\text{SEK}/^{\circ}\text{C}^3$ and $\beta = 1$. For a good choice of heating values, our objective function is small, so the optimizer then attempts to minimize the objective function.

3.3. Genetic algorithm

We use a genetic algorithm as shown in Fig. 3[9,10]. We first select a random population of sets of 24 heater values for 24 h. An individual of the population is then a vector of 24 real numbers. Using the thermal model described above, for each individual, we perform a simulation in the time domain of the thermal performance of the building, taking the forecast for ambient temperature and insolation through windows into account. The simulation gives an estimate of the building temperature over the next 24 h, corresponding to the individual selected. Then the objective function is used to generate a figure of merit for the suitability of a population individual at hand. This is repeated for all individuals of the population.

Again referring to Fig. 3, we next select the most promising individuals for parenting and we perform a crossover operation to generate children. We use a probability of 50% for an offspring to inherit genes from each of the parents. In addition to the gene-sharing, we add a mutation, in which each gene has a certain probability of mutating. Not shown in Fig. 3 is the use of an elitism ratio specifying that a small percentage of the parents are transferred directly to the children (parthenogenesis). The purpose is to prevent loss of genes that already were found to be good. Finally, the best children are used as new parents to form yet another offspring generation and the process continues until a stop criterion is fulfilled.

We have made numerous tests with different choices of the meta-parameters of the genetic algorithm for our application. It was found

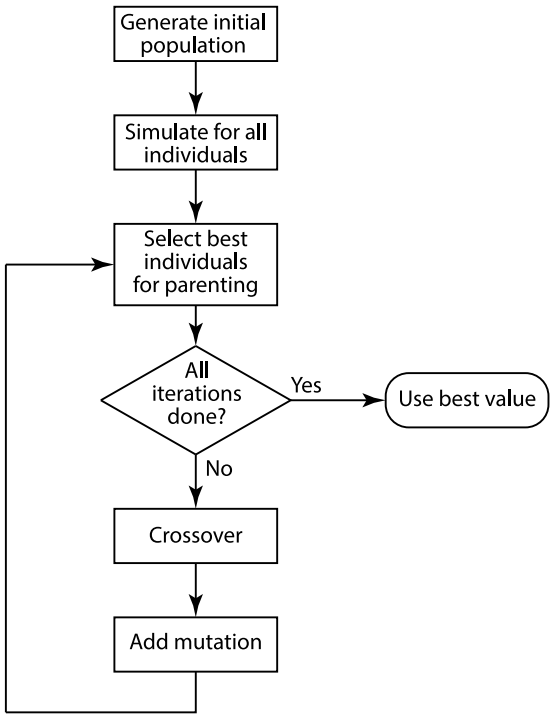


Fig. 3. Principle of genetic algorithm used for optimization.

Table 1
Genetic algorithm metaparameters.

Parameter	Value
Number of iterations	1000
Population size	5
Probability of mutation	10%
Crossover probability	50%
Parenting percentage	30%
Elitism ratio	1%

beneficial in our case to use a small parenting population and a large number of generations. The values in Table 1 gave reliable convergence.

Use of a genetic algorithm has shown to give robust and sensible results for our application. We will present some results in Sections 5.1 and 5.2. It is not guaranteed that the solution is the best possible, in fact it probably never is. The genetic algorithm [11] is coded in Python 3 and is executed once every hour.

3.4. Particle swarm optimization

Particle swarm optimization [12,13] is a metaheuristic method that so far has found only limited use for energy control [14,15]. Again, the task is, every hour, to estimate the "best" heater values for each of the coming 24 h, when future electricity cost and a weather forecast are available. The purpose is then to estimate an n-dimensional heating vector, where n = 24. The principle of swarm optimization is to launch P "birds" (called "particles") in n-dimensional space with random start locations and with random velocities, both within boundary values. At each subsequent time step, the particles change their velocity vector, based jointly on the previous velocity, the direction to the best location found by the entire swarm, and the best location seen by the particle itself. Randomness is added to avoid local minima. A figure of merit for each location is found by running the thermal model using the heating vector as input and then using the objective function already described. In contrast to what is the case for the genetic algorithm, the individual particles live during the entire optimization process.

Table 2
Particle swarm optimization metaparameters.

Parameter	Value
Swarm size	500
Max number of iterations	60
Cognitive coefficient	0.5
Social coefficient	0.5

The algorithm is simple. Assume that a particle with index i at time t has a location defined by the n-dimensional vector $\mathbf{x}_t^i$ and a velocity vector $\mathbf{v}_t^i$. The new velocity at time step $t + \Delta t$ then is

$$\mathbf{v}_{t+\Delta t}^i = \Omega \mathbf{v}_t^i + \varphi_c \xi(\mathbf{r}_\mathbf{t}^\mathbf{i} - \mathbf{x}_t^i) + \varphi_s \xi(\mathbf{s}_t - \mathbf{x}_t^i) \tag{8}$$

where $\mathbf{r}_t^i$ is the best location that the particle has ever seen itself, and $\mathbf{s}_t$ the best that the entire flock of particles has seen at time t. The user-selected weighting factor φ_c is the *cognitive coefficient*, and φ_s is the *social coefficient*. The symbol ξ represents a random real number drawn from the interval [0,1] at every instance to add randomness to the search process. The factor Ω is selected by the user and must be less than or equal to 1 to preserve convergence. As suggested in [16], we use the value 1 at the first iteration and then decrease it linearly to 0.4 at the last iteration. The new position at time step $t + \Delta t$ is

$$\mathbf{x}_{t+\Delta t}^i = \mathbf{x}_t^i + \mathbf{v}_{t+\Delta t}^i \tag{9}$$

In addition, we have introduced a limit to the maximum velocity any particle may have as described in [17]. Also, whenever a particle reaches a boundary, we set its velocity to zero [16]. Both features are known to improve convergence, and that was also the case for our application.

We selected the metaparameters on the basis of suggestions in the literature and by experimenting. The values are shown in Table 2. We used an existing library [18] that we modified to take into account the additions mentioned above.

3.5. Neural network

In addition to the genetic algorithm, a neural network [19] may also provide an estimate of optimal heating values. Our main incentive for studying the neural network solution has been to reduce computation time and computer memory footprint as compared to use of the other optimization methods. The input to the neural network is the real time pricing, external temperature forecast and the sky coverage forecast for the next 24 h, and the output is a dataset with hourly heating values. It may be desirable to adjust the setpoint, T_s, during operation, and the neural network must work in closed-loop mode to avoid drift from estimation errors. Hence, in addition to the weather and pricing forecasts, we also include the temperature setpoint and the initial building temperature as inputs to the neural network. As for the other optimizers, the neural network is run every hour.

For supervised learning, a number of training cases with known inputs and targets are needed. As input values for our training cases, we used 24-h sequences of real time pricing, outdoor temperature and sky coverage forecasts randomly sampled from records taken over about three months by the end of 2022 at a location in southern Sweden. We selected setpoints and initial building temperatures randomly in the interval $[6, 12]$ °C. As targets for the training, we used "optimal" heating values determined by simulation using the genetic algorithm and the particle swarm optimization method, alternately. All inputs were normalized to have a zero mean and a standard deviation of 1. The outputs were normalized to the interval [0,1].

Our neural network must be suitable for microcontrollers, for instance running Tensorflow Lite for microcontrollers. Hence, the size of the neural network must be as small as possible. The neural network finally selected for our study is shown in Fig. 4. It is indeed shallow. The network has two input branches, one for temperature setpoint and

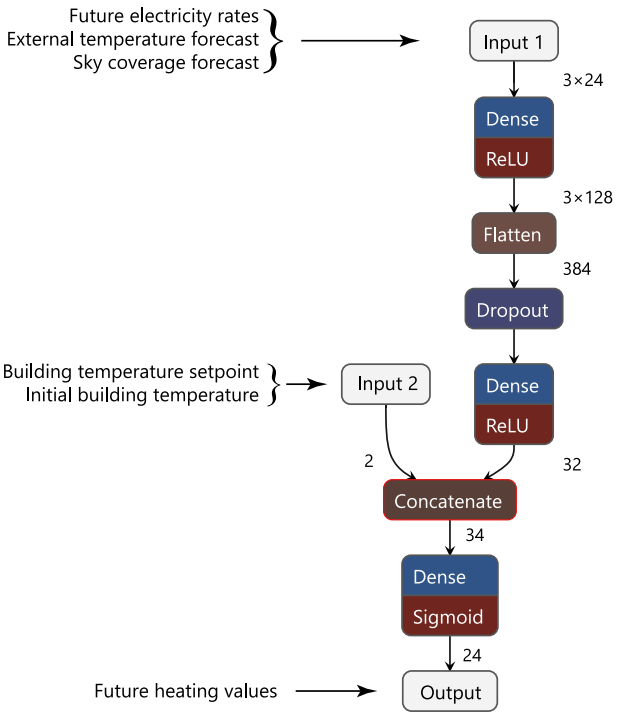


Fig. 4. Neural network for estimation of 14 hourly heating values.

Table 3
Neural network metaparameters.

Parameter	Value
Number of training datasets	6552
Number of validation datasets	1640
Batch size	256
Number of epochs	1000
Number of trainable parameters	16 360
Optimizer	Adam
Loss function	Mean squared error

temperature feedback, and one for hourly values. In the normal case, when the forecasts have a length of 24 h, we only need one output value (the first heating value over a 24 h period). However, at specific times of the day, depending on the power company, there may not be 24 h of forecasts available, so we chose to let the neural network estimate all heating values for the first 14 h. By including input values from the past, it is then possible to use the same neural network also when the forecast does not include all 24 h values.

We also considered using a Long Short Term Memory (LSTM) network but it was abandoned. The strength of an LSTM lies in its capability to carry over previous time series information (such as trends), which is not an issue for our application. Weather and electricity pricing both vary rather erratically due to external factors not included in the time series.

For the neural network we used Python 3 and Keras with Tensorflow. Metaparameters are shown in Table 3.

4. Implementation

The proposed system was evaluated both by simulation and experimentally. Using a simulation model, we compared the optimization approaches, and with the experimental setup, we verified system performance using the genetic algorithm.

Table 4
Simulation parameters.

Parameter	Symbol	Value	Unit
Thermal time constant	τ	24	h
Wall heat transfer coefficient	h_{wall}	17.8	W/°C
Heat capacity	C	1.54×10^6	J/°C
Solar influx constant	h_{rad}	250	W
ODE solver integration interval		5	min
Heater wattage		800[a]	W
Setpoint		9	°C
Initial condition		9	°C

[a] Value for primary room being controlled. Total heater wattage was 1800 W.

4.1. Simulation model

We performed simulations to test and compare the regulator principles, and we also compared their performance to that of a regular thermostat. We simulated heating of a residential building using true real-time data for two days in December 2022 in southern Sweden. The hourly pricing was taken from the NordPool power market, including value added tax. Instead of a temperature forecast, we used actual, recorded outdoor temperature measured over the two days at the residential dwelling that we also used later for experimental verification. The insolation was taken from weather forecasts for the same period.

Fig. 5 shows the simulation model used. The "regulator" in the figure then includes one of the optimizers described in Section 3, or a regular thermostat with a specified setpoint and deadband for comparison. The simulation was coded in Python 3 running under Windows 11 on an Intel i7-11850H CPU. We used the system parameters shown in Table 4. Most of the values are identical to those of the system used for the experimental verification to be described later. We controlled one part of the house with an 800 W heater. Another part had a 1000 W heater working as a slave unit.

Solving the ordinary differential equation (1) analytically is straightforward, but for coding reasons it was easier to solve the equation numerically to find a temperature time history. The only dynamics involved in the thermal model is a large time constant. We used an explicit Runge–Kutta fourth order solver, which is readily available in the Scipy library.

4.2. Experimental setup

To verify that the controller principle works well in practice, we installed a real-time system in an existing residential building. The system continuously retrieved hourly updated weather forecasts from the API of the Swedish Meteorological and Hydrological Institute, and daily electricity real time pricing from the NordPool API. The algorithms of our control systems gave the required average heating for every hour. The heaters were controlled ON/OFF using a contactor by pulse width modulation with a time step of 1 h. For instance, a heater value of, say, 33%, then corresponds to the heater being switched on for 20 min and off for 40 min. We also recorded external and internal building temperatures. The parameters were the same as those given in Table 4. All software for the experiment was written in Python 3. The controller ran on an i7-4790S CPU at 3.20 GHz using Windows 10, and the average CPU load from the controller was low. The system has been tested for a few months and no major problems were seen with the controllers described above.

5. Results

5.1. Simulation results

To illustrate performance under different conditions, we chose a time series having a large electricity price dip during the first night and a more flat price profile during the second night. The hourly real

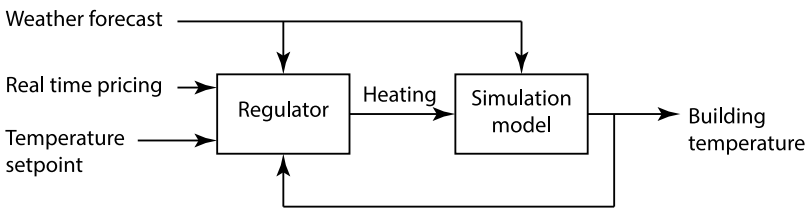


Fig. 5. Simulation model architecture.

Table 5
Electricity usage and cost for the simulation period of 48 h.

Method	Consumption kWh	Cost SEK (Swedish Crowns)
Genetic algorithm	16.1	67.5
Genetic algorithm, τ underestimated by 50%	16.1	70.9
Genetic algorithm, τ overestimated by 100%	15.8	65.4
Particle swarm optimization	16.2	66.4
Neural network	16.1	66.2
Thermostat	21.4	92.6

Table 6
Optimizer comparison when running under Windows 11 on an i7-11850H/2.50 GHz processor.

Method	Peak memory allocation kB	Typical CPU time s
Genetic algorithm	407	32
Particle swarm optimization	911	105
Neural network	24[a]	0.064[b]

[a] Size of network file for Tensorflow Lite for microcontrollers.
[b] Without GPU.

time pricing is shown in the uppermost plot of Fig. 6, and the heating plans found by the genetic algorithm, the particle swarm optimization, and the neural network in the three plots below. The necessary heating using a thermostat can be seen in the fifth plot of the figure, and the estimated insolation through windows in the plot below. For the days chosen, the Sun was shining only during two short periods. Finally, the building temperatures obtained from the simulation are presented in the lower plot together with the outdoor temperature, which was used by the optimizers. The first night had a substantial kWh price dip, so all three optimizers increased the building temperature during that period to store energy. On the second day, the kWh price was rather flat with little saving potential, and the optimizers all spread out heating over time.

The total power consumption and electricity cost for the period is shown in Table 5. The cost savings obtainable by heat scheduling as compared to the use of simple thermostat depends on the variation of the real time pricing over time. In most cases studied, based on real weather and cost data from the end of 2022 in southern Sweden, we found cost savings over 40%, but in this example, the saving potential was smaller due to the partially flat kWh price profile, so the cost saving was only about 28%.

We also made simulation runs for which we deliberately underestimated the building time constant by 50%, or overestimated it by 100% in the objective function with respect to that of the simulation model (24 h). The results are also shown in Table 5. The consequences of using a wrong time constant for the optimizer are marginal, as long as the time constant is considerably longer than the sampling time

Table 6 shows the peak runtime memory for the control program with different optimizers. Obviously, the values depend on the specific coding of the algorithms. Hence, the values are very approximate, but they are still a good basis for evaluating the complexity of the different optimizers. For the neural network, the memory allocation given is the actual size of the neural network file, when exporting it to a microcontroller running Tensorflow lite. This cannot be compared directly to the other memory values but does give an order of magnitude. Tensorflow Lite for microcontrollers itself takes only about 16 kB. All execution times given in this table were taken from runs on an Intel i7-11850H running Windows 11 at 2.50 GHz, here without use of GPU.

5.2. Experimental verification

We ran a full-scale experiment over a couple of months at a house in Southern Sweden using a genetic algorithm. For illustration, we here again show a period having both significant dips with a low electricity price and a more flat region without major dips. The results are shown in Fig. 7. The thermal time constant, τ, of the model used for optimization was set to 24 h. The genetic algorithm works well when it comes to exploiting the low electricity cost at locations A, C and D. In the B region, the price curve is flatter, so the algorithm spreads out heating more over time to prevent the building temperature from dropping too much. The setpoint was here also 9 °C.

For the period covered by this experiment, the total power consumption was 13.4 kWh and the energy cost 27.0 SEK. For comparison, we ran a simulation for the system with a thermostat using the same weather data and real time prices and found the corresponding values to be 14.8 kWh and 66.6 SEK. The cost saving using the genetic algorithm was then 59% for this period

We finally compared the experimental result to that of a simulation with a genetic algorithm using our thermal model. From the experimental data, the simulation took the electricity rates and the weather data as input, and we then ran a complete simulation with the thermal model and the genetic algorithm. The resulting house temperature from the simulation is shown in the blue curve of Fig. 6. By comparing the simulation temperature curve to the curve from the experiment, it seems that the value of the thermal time constant (24 h) used in the model may be a little underestimated. However, the control system still gives reasonable results and closed loop operation also removes drift.

6. Discussion

With a simulation model, we have compared optimizers for “smart” controllers that can easily be used for existing buildings with no strict comfort requirements. We have compared a genetic algorithm, a particle swarm optimization, a neural network and a regular thermostat controller.

With our EMS, the simulation example in Section 5.1 using a genetic algorithm showed a cost saving of 28% compared to using a thermostat regulator, whereas the experiment period presented in Section 5.2 had a saving of 59%. Since practically useful datasets for “benchmarking” of EMS’s do not exist, it is difficult to state exactly how useful a given controller is. The cost benefit of any EMS depends very much on the variability of the real time pricing of electricity. However, during a continuous run with our control system for about two months by the end of 2022, we saw in most cases cost savings over 40% as compared to use of a regular thermostat.

As can be seen from Table 5 and Fig. 6, the genetic algorithm and the particle swarm optimization give rather similar results and are about equal in performance. However, as shown in Table 6, the

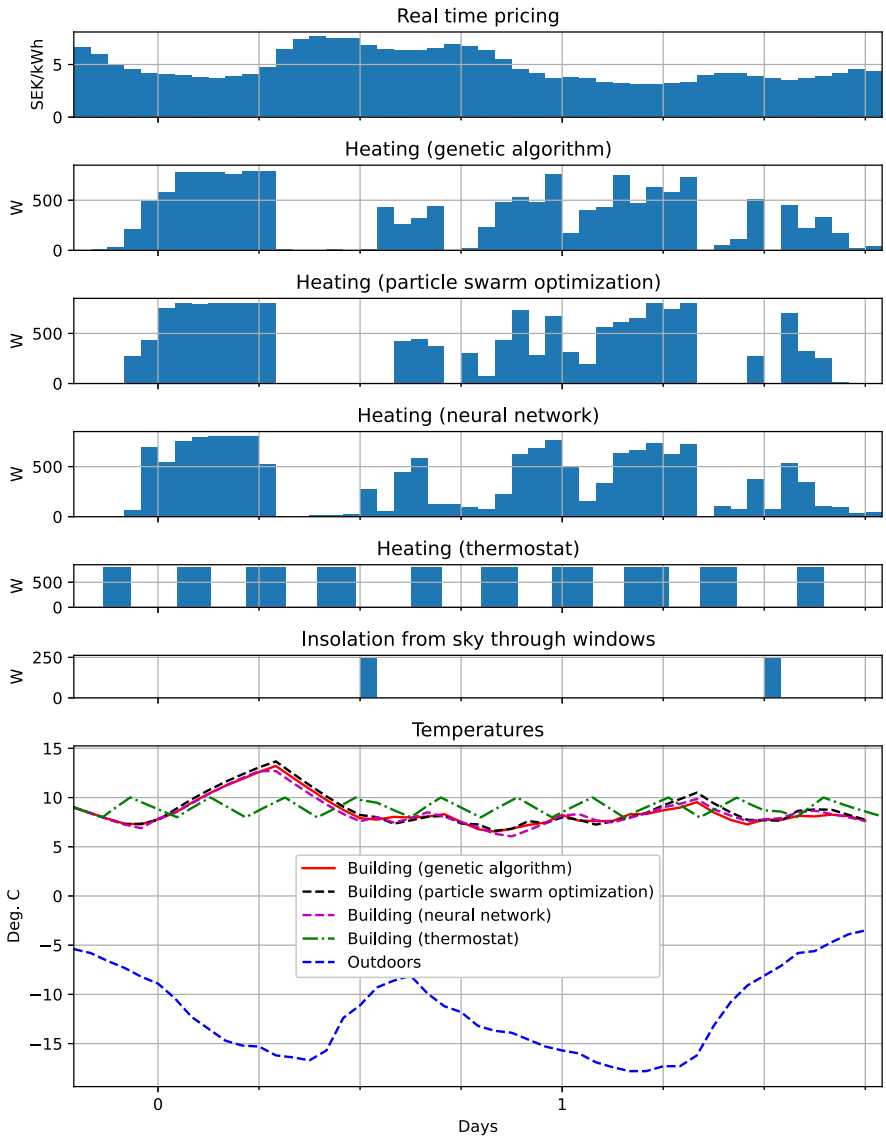


Fig. 6. Simulation results. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

memory and the computation time requirements for the particle swarm optimization are larger, so for our application, the genetic algorithm is preferable. A more general comparison of the two optimizers can be found in [20].

The neural network is very shallow. It has been trained with supervised learning on simulation data from a blend of data from the genetic algorithm and the particle swarm optimization. Obviously, it cannot be expected to perform better than those two optimizers. However, the neural network is very small and has only little execution time, so it is well suited for low-cost microcontrollers, for instance running Tensorflow Lite for microcontrollers.

As mentioned in Section 1, neural networks based on reinforcement learning are also possible [5–7], but they are more complex, and it is still unclear, whether they are preferable over genetic algorithms or particle swarm optimizations for our applications, and whether they can handle a variety of different existing buildings well. This also highlights the need for generally accepted practical benchmarking datasets for our type of application that can be used for comparison of different energy management systems.

We are here using a model-based approach. It is a disadvantage of the method, that estimates of model parameters are needed beforehand. However, the few parameters are rather easy to estimate, and the influence of wrong parameter choices will be reduced, because our system operates in closed loop. As already described, the global heat transfer coefficient for all walls together, h_{wall}, is straightforward to estimate. Selection of the correct value for the time constant plays a role for cost optimization but as shown in Table 5, a correct choice of the time constant is not critical.

The electricity pricing and the weather forecasts are available every hour, so we chose to use the same resolution for our optimizers. It would be possible to work with a higher resolution for the optimization but considering that the time constant of the buildings typically are 1–2 days, it was felt that a higher resolution would not increase the efficiency of the optimizer significantly. Obviously, the thermal model works with a much smaller integration interval (generally a few minutes).

In addition to what is already described, we have also set up and tested a Kálmán filter and an adaptive controller to continuously adjust the value of the time constant of the model, but we did not find it useful. Use of the adaptive controller will lead to some cost savings, but it is a disadvantage that it will not work well, if there are external heat sources or sinks not accounted for, because they will "confuse" the adaptive controller. An adaptive controller is therefore mainly of interest for applications, where there are no heat sources and sinks that

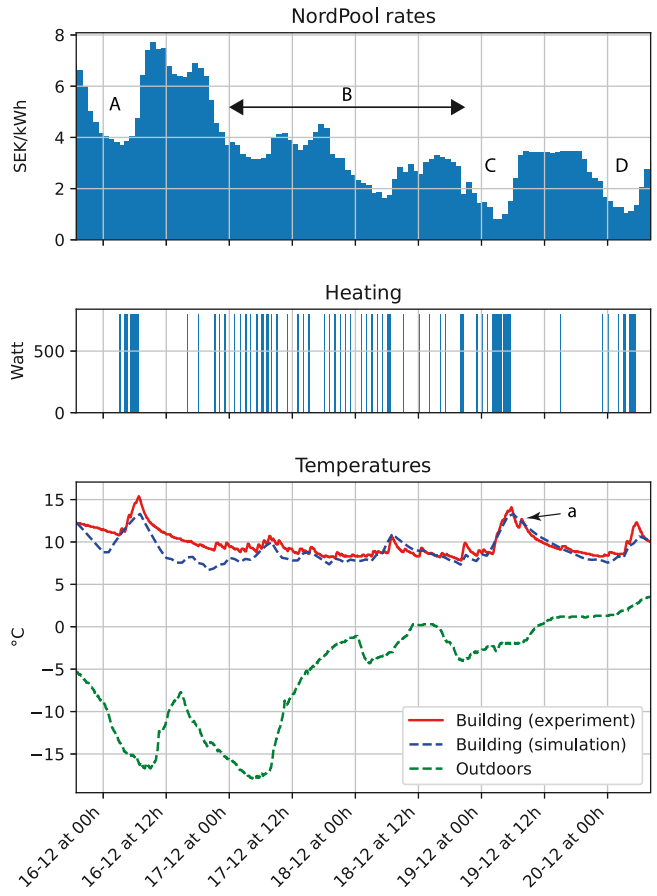


Fig. 7. Results from residential house experimental verification. The small peak at "a" is due to a contactor malfunction and can be disregarded.

are unknown to our control system, or when it is difficult to estimate the thermal time constant. However, that is normally not the case.

It could be a drawback of the simulation model presented in Section 4.1, that the same mathematical model is used both for the algorithms and for the simulation. Discrepancies in building modeling could be masked by use of the same model for both purposes. However, the experimental validation of Section 5.2 largely confirmed the simulation results.

The proposed system uses the building itself for energy storage by increasing temperature during periods with low electricity price. This is beneficial both from the point of view of cost and power grid loading. However, a higher building temperature leads to a larger heat loss through the walls and therefore increases total power consumption. Thus, our proposed system is most attractive for well-insulated buildings.

We have primarily focused on the applications outlined in Section 1, but it would be rather easy to change the objective function of the optimizers to accommodate specific needs for the occupants to have acceptable temperatures to match their daily routines. However, the benefit of the system would then diminish.

7. Conclusion

In summary, we have studied and tested an EMS with either of three different optimizers for temperature control of buildings with little or no comfort requirements. Simulation and a full-scale experiment show that the proposed system works well with savings of up to 59%, although our experience has shown that energy cost savings generally were about 40% during the winter months for a house in southern Sweden. The proposed system is useful for a significant electric power consumer sector: Warehouses, vacation homes and air-drying of buildings under construction. Use of the system is simple and it is easy to implement with microcontrollers in existing buildings. And it will provide a demand response that contributes to leveling of electric power consumption over time, and thereby to reducing price volatility and pollution.

CRediT authorship contribution statement

Torben Andersen: Conceptualization, Software, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

The author is grateful to Dr. Mette Owner-Petersen, Lund University, Sweden, for many valuable discussions and for proofreading.

References

[1] Zhou B, Li W, Chan KW, Cao Y, Kuang Y, Liu X, Wang X. Smart home energy management systems: Concept, configurations, and scheduling strategies. Renew Sustain Energy Rev 2016;61:30–40. http://dx.doi.org/10.1016/j.rser.2016.03.047, URL https://www.sciencedirect.com/science/article/pii/S1364032116002823.

[2] Hussain HM, Javaid N, Iqbal S, Hasan Q, Khursheed K, Alhussein M. An efficient demand side management system with a new optimized home energy management controller in smart grid. Energies 2018;11:190. http://dx.doi.org/10.3390/en11010190.

[3] Fatima I, Javaid N, Shafiq S, Asif S, Rahim MH, Aslam S. Comparative study of meta-heuristic approaches towards utilization of home energy management: Comparative meta-heuristic approaches. In: 2018 international conference on computing, mathematics and engineering technologies (iCoMET). 2018, p. 1–6. http://dx.doi.org/10.1109/ICOMET.2018.8346436.

[4] Albadi M, El-Saadany E. A summary of demand response in electricity markets. Electr Power Syst Res 2008;78(11):1989–96. http://dx.doi.org/10.1016/j.epsr.2008.04.002, URL https://www.sciencedirect.com/science/article/pii/S0378779608001272.

[5] Barrett E, Linder S. Autonomous HVAC control, a reinforcement learning approach. In: Bifet A, May M, Zadrozny B, Gavalda R, Pedreschi D, Bonchi F, Cardoso J, Spiliopoulou M, editors. Machine learning and knowledge discovery in databases. Cham: Springer International Publishing; 2015, p. 3–19.

[6] Dutreilh X, Kirgizov S, Melekhova O, Malenfant J, Rivierre N, Truck I. Using reinforcement learning for autonomic resource allocation in clouds: Towards a fully automated workflow. In: ICAS 2011 the seventh international conference on autonomic and autonomous systems. 2011, p. 67–74.

[7] Lissa P, Deane C, Schukat M, Seri F, Keane M, Barrett E. Deep reinforcement learning for home energy management system control. Energy AI 2021;3:100043. http://dx.doi.org/10.1016/j.egyai.2020.100043, URL https://www.sciencedirect.com/science/article/pii/S2666546820300434.

[8] Xi Y, Li D. Predictive control: Fundamentals and developments. 2019, http://dx.doi.org/10.1002/9781119119593.

[9] Wirsansky E. Genetic algorithms with Python. 1st ed.. Packt Publishing Ltd.; 2020.

[10] Reza M, Polaki S, Sinha Roy D. A genetic algorithm for optimal power scheduling for residential energy management. In: IEEE 15th international conference on environment and electrical engineering, EEEIC 2015 - Conference proceedings. 2015, http://dx.doi.org/10.1109/EEEIC.2015.7165494.

[11] Solgi R. Github. [link]. URL https://github.com/rmsolgi/geneticalgorithm/.

[12] Kennedy J, Eberhart R. Particle swarm optimization. In: Proceedings of ICNN'95 - International conference on neural networks, Vol. 4. 1995, p. 1942–8. http://dx.doi.org/10.1109/ICNN.1995.488968.

[13] Marini F, Walczak B. Particle swarm optimization (PSO). A tutorial. Chemometr Intell Lab Syst 2015;149:153–65. http://dx.doi.org/10.1016/j.chemolab.2015.08.020, URL https://www.sciencedirect.com/science/article/pii/S0169743915002117.

[14] Abid S, Zafar A, Khalid R, Javaid S, Qasim U, Khan ZA, Javaid N. Managing energy in smart homes using binary particle swarm optimization. In: Barolli L, Terzo O, editors. Complex, intelligent, and software intensive systems. Cham: Springer International Publishing; 2018, p. 189–96.

[15] Lee KP, Chng CW, Tong DL, Tseu KL. Optimizing energy consumption on smart home task scheduling using particle swarm optimization. Procedia Comput Sci 2023;220:195–201. http://dx.doi.org/10.1016/j.procs.2023.03.027, The 14th International Conference on Ambient Systems, Networks and Technologies Networks (ANT) and The 6th International Conference on Emerging Data and Industry 4.0 (EDI40). URL https://www.sciencedirect.com/science/article/pii/S1877050923005628.

[16] Carlisle A, Dozier G. An off-the-shelf PSO. In: Proceedings of the workshop on particle swarm optimization. 2001.

[17] Eberhart R, Shi Y. Comparing inertia weights and constriction factors in particle swarm optimization. In: Proceedings of the 2000 congress on evolutionary computation. CEC00 (Cat. no.00TH8512), Vol. 1. 2000, p. 84–8. http://dx.doi.org/10.1109/CEC.2000.870279.

[18] Github. [link]. URL https://github.com/tisimst/pyswarm.

[19] Chollet F. Deep learning with Python. Manning Publications Co.; 2017.

[20] Eberhart RC, Shi Y. Comparison between genetic algorithms and particle swarm optimization. In: Porto VW, Saravanan N, Waagen D, Eiben AE, editors. Evolutionary programming VII. Berlin, Heidelberg: Springer Berlin Heidelberg; 1998, p. 611–6.