Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

Machine Learning Assisted Solutions of Mixed Integer MPC on Embedded Platforms

Yannik Löhr * **Martin Klaučo** ** **Miroslav Fikar** **
Martin Mönnigmann *

* *Automatic Control and Systems Theory, Ruhr-Universität Bochum, Bochum, Germany*
(yannik.loehr, martin.moennigmann)@rub.de.
** *Institute of Information Engineering, Automation and Mathematics, Slovak University of Technology in Bratislava*
(martin.klauco,miroslav.fikar)@stuba.sk

Abstract: Many control applications, especially in the field of energy systems, require a simultaneous decision for continuous and binary values of control inputs. In optimal control methods like model predictive control (MPC), this leads to the problem of solving expensive mixed-integer programs online. As this solution in practice has to be calculated with low cost embedded hardware with low energy demand, it is necessary to reduce the computational demand in advance. We present an approach to replacing the mixed-integer program by a simpler quadratic program by means of learning techniques. To be more specific, we design a neural network and a support vector machine to classify the optimal control policies for the binary inputs offline and evaluate this decision in the online step as the basis for the solution of the quadratic program. As a result, we achieve a controller suitable for implementation on embedded hardware. We demonstrate its applicability to a domestic heating system. The results indicate a very high quality of the approximation of the primary optimal controller that solves mixed-integer programs online.

Keywords: nonlinear predictive control, energy control, data-based control, neural networks, classification, heat flows, control applications

1. INTRODUCTION

Energy systems often require to optimize components with binary or discrete input signals, as it is the case with simple thermostats (Drgoňa et al., 2015) or complex turbines (Ferrari-Trecate et al., 2004). In the context of MPC, this in general means that mixed-integer programs need to be solved to find optimal solutions, see Morari and Barič (2006); Richards and How (2005). In the last decades, there have been lots of innovations to speed up the solution of mixed-integer programming problems in control applications, as described by Bixby (2012), Zheng et al. (2014) or Achterberg et al. (2016), leading to the availability of several highly optimized codes. Nevertheless, it is still a complex task to solve this type of problem within a reasonable amount of time, especially when it comes to implementation on limited, embedded hardware usually available in small scale energy systems.

Recently, the approximation of optimal control algorithms for implementation on embedded hardware with limited resources became attractive due to innovations in supervised and unsupervised machine learning methods. Hertneck et al. (2018) investigated the approximation of a robust model predictive controller by arbitrary supervised learning techniques. Klaučo et al. (2019) improved active set methods used for quadratic programs, which arise in MPC, by providing near-to-optimal warm starts, by means of approaches like k-NN and SVM classifiers.

The related work of Lucia and Karg (2018), Drgoňa et al. (2018) or Löhr et al. (2019) distinguishes itself to be more practical, as neural nets are trained to resemble the MPC for resonant power converters or building HVAC systems.

In this paper, we consider a class of MPC problems, where the design model has continuous linear dynamics but has both continuous and binary control inputs. Formulations of such MPC problems lead to mixed-integer linear (MILP) or quadratic programming (MIQP) problems, depending on the choice of the objective function. In general, these types of MILPs or MIQPs are challenging, even on standard hardware with state-of-the-art solvers like Gurobi or CPLEX. We present an approach where a machine learning tool estimates the value of the binary variables. By fixing the binary variables, we reduce the complexity of the MI problem to a standardized LP/QP

⋆ This work was supported by the Alexander von Humboldt Foundation (AvH-1065182-SVK) and by the German Federal Ministry for Economic Affairs and Energy under grant 03ET1274B. M. Klaučo and Miroslav Fikar gratefully acknowledge the contribution of the Scientific Grant Agency of the Slovak Republic under the grants 1/0585/19 and the contribution of the Slovak Research and Development Agency under the project APVV 15-0007.

problem. We acknowledge that the presented control algorithm may lead to suboptimal solutions compared to the full-fidelity MILP/MIQP controller. Still, it is embeddable on hardware with limited computation and memory resources. We demonstrate the applicability of the proposed algorithm for a domestic heating system.

This work considers two machine learning approaches used to approximate the binary variables − first, a simple neural network, and second a support vector machine tool. The same set of training data is used to train both classifiers. Neural networks proved useful in our previous publication (Löhr et al., 2019), but it is also supported by recent work by Bertsimas and Stellato (2019), which has also been used in connection with improving the computational burden of the mixed-integer optimization.

In Section 2, we introduce the considered MPC problem, before we present an algorithm to reduce the complexity of the MPC problem which is required to be solved online in Section 3. The control performance for different machine learning approaches is investigated for a domestic heating system in Section 4.

2. PROBLEM STATEMENT

The main objective of the paper is to present an embeddedable approach to a class of mixed-integer MPC problems. The standard formulation of the MPC with continuous and binary control inputs reads

$$\min_{u_k,\delta_k} \ \ell_{\mathrm{N}}(x_N) + \sum_{k=0}^{N-1} \ell(x_k, u_k, \delta_k) \tag{1a}$$

$$\text{s.t. } x_{k+1} = Ax_k + B_{\mathrm{c}}u_k + B_\delta\delta_k + Ed_k \tag{1b}$$

$$u_k \in \mathcal{U}_{\mathrm{c}}, x_k \in \mathcal{X}, \delta_k \in \mathcal{U}_{\mathrm{b}} \tag{1c}$$

$$x_0 = x(t) \tag{1d}$$

where the states are denoted as $x \in \mathbb{R}^{n_{\mathrm{x}}}$, continuous control inputs as $u \in \mathbb{R}^{n_{\mathrm{u}}}$, binary control inputs as $\delta \in \{0,1\}^{n_\delta}$, and disturbances as $d \in \mathbb{R}^{n_{\mathrm{d}}}$. The objective function (1a) consist of a terminal penalty and stage cost, spanned over the prediction horizon of N, where

$$\ell_{\mathrm{N}}(x_N) = x_N^\intercal Q_{\mathrm{N}} x_N, \tag{2}$$

$$\ell(x_k, u_k, \delta_k) = x_k^\intercal Q_{\mathrm{x}} x_k + u_k^\intercal Q_{\mathrm{u}} u_k + \delta_k^\intercal Q_\delta \delta_k, \tag{3}$$

with positive semi definite Q_{N}, Q_{x} and positive definite Q_{u} and Q_δ. Note that the algorithm presented later in the paper is not limited to a quadratic cost function. Furthermore, constraints (1b) and (1c) are enforced for $k \in \{0,\dots,N-1\}$. Control inputs and states are constrained to $\mathcal{U}_{\mathrm{c}}$, $\mathcal{U}_{\mathrm{b}}$ and $\mathcal{X}$, which are polyhedral sets.

We denote the initial conditions for the MPC (1) as

$$\theta = [x_0^\intercal \ d^\intercal]^\intercal, \quad \theta \in \mathbb{R}^{n_\theta}, \tag{4}$$

and the optimal solution to (1) at time step t by $U^\star(t) = \left[u_0^\star, u_1^\star, \dots, u_{N-1}^\star\right]^\intercal \in \mathbb{R}^{n_u N}$ and $\Delta^\star(t) = \left[\delta_0^\star, \delta_1^\star, \dots, \delta_{N-1}^\star\right]^\intercal \in \mathbb{R}^{n_\delta N}$ for continuous and binary inputs, respectively. As common in MPC, only the first element of the solution, namely the pair $\{u_0^\star, \delta_0^\star\}$, is applied to the system in a receding horizon fashion (Mayne et al., 2000).

3. LEARNING BINARY INPUTS IN MPC

In general, it is a challenging task to implement controllers based on the solution of a MIQP on embedded hardware. We present an efficient approach that is based on separating the MIQP resulting from (1) into a QP, where the binary inputs are fixed with an estimation from a supervised learning procedure. After introducing this reformulation, we give details on the data preparation and training, before we compactly state the full implementation procedure.

3.1 Reformulation of the optimal control problem

MIQP problems can be solved with branch-and-bound methods, where the integer variables are relaxed, and subsequently, a sequence of continuous-variable QP is solved. Given the size of the relaxed QP problem, it can be solved in the milli-to-microsecond range even on embedded hardware (Domahidi and Jerez, 2014–2019). The proposed solution to the optimal control problem presented in Section 2 stems from the principle of formulating a relaxed version of the MIQP (1), and solving it for fixed values of the binary variables.

Let $\delta_k^{\mathrm{approx}} \in \{0,1\}^{n_\delta}$ be the approximation of the binary elements of the optimal solution to (1) at each prediction step k, and assume that its value is available by evaluation of a suitable machine learning procedure. Then, if we fixed the binary variables in (1) to $\delta_k^{\mathrm{approx}}$, we would obtain a continuous variable QP. However, since the total number of binary variables in (1) is Nn_δ, it can still be computationally challenging to get approximate values of the $\delta_k^{\mathrm{approx}}$ for the entire prediction horizon. Therefore, we further relax the optimization problem. We fix δ_0 to the approximate value and treat the remaining δ_{k} as optimization variables subject to the constraints $\delta_k \in [0,1]$. Specifically, we reformulate (1) as a quadratic MPC problem

$$\min_{u_k,\delta_k} \ \ell_{\mathrm{N}}(x_N) + \sum_{k=0}^{N-1} \ell(x_k, u_k, \delta_k) \tag{5a}$$

$$\text{s.t. } x_{k+1} = Ax_k + B_{\mathrm{c}}u_k + B_\delta\delta_k + Ed_k, \tag{5b}$$

$$u_k \in \mathcal{U}_{\mathrm{c}}, x_k \in \mathcal{X}, \tag{5c}$$

$$\delta_0 = \delta_0^{\mathrm{approx}}, \tag{5d}$$

$$\delta_k \in [0,1], \tag{5e}$$

$$x_0 = x(t), \tag{5f}$$

where constraints (5b)-(5c) are enforced for $k \in \{0,\dots,N-1\}$, while (5e) holds for $k \in \{1,\dots,N-1\}$.

Naturally, the solution to the "super-relaxed" QP (5) will be a suboptimal solution to the MIQP given in (1). The potential suboptimality of the solution of the super-relaxed QP is remedied by the receding horizon policy implementation, where such a controller is resolved in a closed-loop at each sampling instant. Numerical results will show that if we provide a good approximate value of $\delta_0^{\mathrm{approx}}$, we obtain almost identical control performance compared to the performance of the closed loop with the MIQP controller, with the added benefit of an embeddable control algorithm, as summarized in Section 3.3.

3.2 Machine Learning-based Classification Task

The objective of the machine learning classification is to minimize a classification error

$$\min_{\mathcal{C}} \sum_{i=1}^{n_\mathrm{p}} \left(\delta_i^\star - \mathcal{C}(\theta_i)\right)^2, \tag{6}$$

where n_p is the number of points for which the error is evaluated, $\mathcal{C}$ denotes a classifier, and θ_i is the initial condition for which $\delta_i^\star$ was obtained by solving (1). Formally, we define

$$\theta_i \in \mathcal{I}, \tag{7}$$

where the set $\mathcal{I}$ is a collection of all feasible initial conditions for MIQP in (1). Similarly, we define

$$\delta_i^\star \in \mathcal{T} \tag{8}$$

as a set of optimal solutions to the MIQP (1), where $\delta_i^\star$ stands for the first element of a particular optimal solution corresponding to θ_i from (7). Conversely, set $\{\theta_i, \delta_i^\star\}$ is referred to as training tuple, while sets $\mathcal{I}$, and $\mathcal{T}$ form a training data set.

The goodness of the fit (6) heavily depends not only on the cardinality of sets $\mathcal{I}$, and $\mathcal{T}$, which is n_p, but also on the data structure. In our work, we consider an equidistant split of $\mathcal{I}$ alongside each axis. Such an approach proved to be valid also in other scientific works (Klaučo et al., 2019).

The subsequent task is to find a classifier $\mathcal{C}(\theta)$ that assigns binary values for a new vector of inputs to fit best to the information stored in the training data. The construction of $\mathcal{C}(\theta)$ can be performed off-line with the two steps of (i) generating the training data and (ii) designing the classifier, where the first step has previously been introduced. There exist several ML-methods that can be used for the design of the classifier. In particular, we will investigate neural networks (NN) and support vector machines (SVM) to that effect.

Neural Nets Classifier The first classifier used to obtain the $\delta_0^{\mathrm{approx}}$, is a neural network. The NN is an aggregated function consisting of weighted interconnected activation functions arranged in several layers, that maps input parameters $\theta \in \mathbb{R}^{n_\theta} \rightarrow \delta^{\mathrm{approx}} \in \mathbb{R}^{n_\delta}$. To obtain the weights of individual activation functions, we must solve an optimization problem, given as

$$\min_{\alpha} \sum_{i=1}^{n_\mathrm{p}} \Big(\delta_i^\star - \varphi_{\mathrm{NN}}(\alpha, \theta_i)\Big), \quad \text{s.t. } \delta_i^\star \in \mathcal{T}, \theta_i \in \mathcal{I}, \tag{9}$$

where the variable α stands for the weights, n_p is the number of samples in the training data, and $\varphi_{\mathrm{NN}}(\cdot)$ stands for the neural network. The function that represents the neural network is visualized in the Fig. 1, where green and red dots represent a sigmoid activation function in the form of

$$\varphi(\alpha, z) = \frac{2}{1 + \mathrm{e}^{\alpha z}} - 1, \tag{10}$$

where the α is the weight and z is an aggregated input to a particular sigmoid. The blue dots represent the linear output layer, given as $\delta^{\mathrm{approx}} = \alpha z$. Note, that the α denotes an arbitrary value of the weighing parameters, and its particular value may differ between individual nodes. In Fig. 1, the number in brackets denotes the corresponding element in the input or output vector, respectively. For general knowledge about the neural nets and the training procedure, we direct the reader, for example, to the work by Bishop (1995); Hornik (1991). In our work, we use the training algorithms in the *Deep Learning Toolbox* in MATLAB, particularly the command `fitnet`.

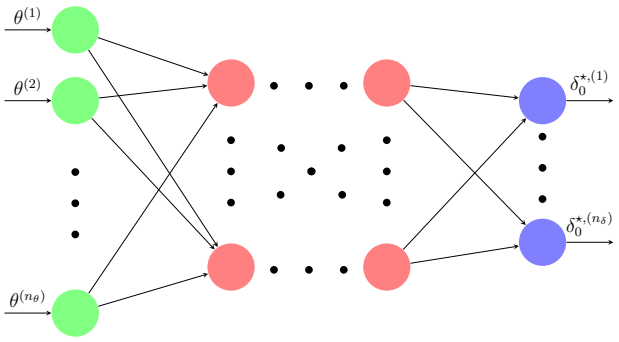

Fig. 1. Illustration of the neural network for the classification task.

SVM Classifier The support vector machine belongs to a family of classification trees (Breiman et al., 1984). The main advantage of the SVM classifier is that it assigns scalar output w_{SVM} value to either -1 or 1, by evaluating an affine function, namely:

$$w_{\mathrm{SVM}} = 1 \ \text{ if } a^\intercal\theta - b \geq 1, \tag{11}$$

$$w_{\mathrm{SVM}} = -1 \ \text{ if } a^\intercal\theta - b \leq -1, \tag{12}$$

where the coefficients a, and b can be found by constructing a simple feasibility problem, which is a linear optimization problem (Drucker et al., 1996). A simple post-processing of the result from (11) is required to get the value of each j-th element in $\delta^{\mathrm{approx},(j)}$. Concretely, we set

$$\delta^{\mathrm{approx},(j)} = 1 \text{ if } w_{\mathrm{SVM}} = 1, \tag{13a}$$

$$\delta^{\mathrm{approx},(j)} = 0 \text{ if } w_{\mathrm{SVM}} = -1. \tag{13b}$$

To obtain the SVM classifiers in this work, we use the command `fitcsvm` which is part of the *Statistics and Machine Learning Toolbox*. The main advantage of the SVM is that the output has already a binary nature, which is not true for the neural network. However, the main disadvantage is that for each element in the vector $\delta_0^{\mathrm{approx}}$, a single SVM model is required. This results in n_δ SVM models, for data needs to be stored as a part of the control algorithm.

3.3 Implementation of the ML-enhanced Algorithm

The proposed reformulation approach is separated into the off-line performed preparation phase and the on-line phase. The preparation phase includes the generation of the training data set $\left\{(\theta_1, \delta_1), \ldots, (\theta_{n_\mathrm{p}}, \delta_{n_\mathrm{p}})\right\}$ and the identification of a suitable, machine-learning based approximation of $\delta_0^\star$, as described in Section 3.2. The on-line phase combines the evaluation of the classifier $\mathcal{C}(\theta)$ and the solution to (5). The on-line phase requires the following steps:

(1) measure state $x(t)$ and disturbance $d(t)$ to obtain $\theta = [x(t), d(t)]$;
(2) approximate the optimal binary feedback law as $\delta^{\mathrm{approx}}(t) = \mathcal{C}(\theta)$;
(3) solve (5) with $\delta_0 = \delta^{\mathrm{approx}}(t)$ to obtain $\{U^\star, \Delta^\star\}$;
(4) implement the pair $\{u_0^\star, \delta_0^\star\}^\intercal$ to the system (1b) and repeat from Step 1 at the subsequent time instant.

4. RESULTS

4.1 Example Heating System

We investigate the efficiency of our approach for the domestic heating system shown in Fig. 2, which was derived from Löhr and Mönnigmann (2018). The system comprises a stratified thermal energy storage and two heat generators, specifically an electrical heat pump and a heating rod. The storage is split into two parts, which store heat on temperature levels suitable for space heating (SH) and domestic hot water (DHW) heating. The stored energies are denoted as $E_{\mathrm{sh}}(t)$ and $E_{\mathrm{dhw}}(t)$, respectively. The heat generated by the heat pump is separated to these two levels, such that the inputs read $q_{\mathrm{sh}}(t) = q_1 u_1(t)$ and $q_{\mathrm{dhw}}(t) = q_2 u_2(t)$. The input $q_{\mathrm{dis}}(t) = q_3 u_3(t)$ describes the heat that is drawn from the storage to supply space heating. The heating rod can be operated at three discrete stages, which can be represented by $q_{\mathrm{hr}}(t) = q_4\delta_1(t) + q_5\delta_2(t)$. It is the task of the system to provide heat according to SH and DHW demand, $q_{\mathrm{l1}}(t)$ and $q_{\mathrm{l2}}(t)$, respectively. These disturbance variables are modeled as time series of the form

$$\mathcal{E}(t) = [e(t), e(t+1), \dots], \tag{14}$$

of which at least the next N elements are assumed to be available. A discrete-time state space representation of the

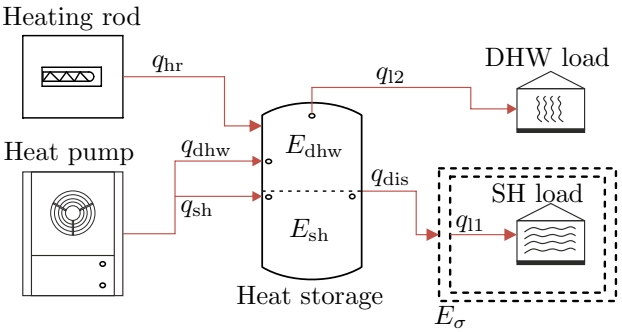

Fig. 2. Sketch of domestic heating system

form (1b) of the system with states $x = [E_{\mathrm{sh}}, E_{\mathrm{dhw}}, E_\sigma]^\intercal \in \mathbb{R}^{n_{\mathrm{x}}}$, continuous inputs $u = [u_1, u_2, u_3]^\intercal \in \mathbb{R}^{n_{\mathrm{u}}}$, binary inputs $\delta = [\delta_1, \delta_2]^\intercal \in \mathbb{R}^{n_\delta}$, and disturbances $d = [q_{\mathrm{l1}}, q_{\mathrm{l2}}]^\intercal \in \mathbb{R}^{n_{\mathrm{d}}}$ was presented in Löhr et al. (2019), where the system matrices read

$$A = \begin{bmatrix} \alpha_1 & \nu & 0 \\ 0 & \alpha_2 - \nu & 0 \\ 0 & 0 & \alpha_3 \end{bmatrix}, B_{\mathrm{u}} = \begin{bmatrix} \beta_1 q_1 & 0 & -\beta_3 q_3 \\ 0 & \beta_2 q_2 & 0 \\ 0 & 0 & \beta_3 q_3 \end{bmatrix}$$

$$B_\delta = \begin{bmatrix} 0 & 0 \\ \beta_4 q_4 & \beta_5 q_5 \\ 0 & 0 \end{bmatrix}, E = \begin{bmatrix} 0 & -\gamma_1 \\ 0 & -\gamma_2 \\ -\gamma_3 & 0 \end{bmatrix} \text{ and } C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

where $\alpha_i,\ i = 1, 2, 3$, ν, $\beta_i,\ i = 1, \dots, 5$ and $\gamma_i,\ i = 1, 2, 3$ are efficiencies.

The continuous inputs are subject to the upper bounds

$$0 \leq u_i(t) \leq 1, i = 1, 2, 3, \tag{15}$$

with the additional constraint

$$0 \leq u_1(t) + u_2(t) \leq 1 \tag{16}$$

on the heat generated by the heat pump. The heating rod is restricted to operation at discrete stages, such that it holds

$$\delta_i(t) \in \{0, 1\}, i\ = 1, 2. \tag{17}$$

We restrict the states by bounds on the heat storage

$$E_i^{\min} \leq E_i(t) \leq E_i^{\max},\ i = \{\mathrm{sh}, \mathrm{dhw}\} \tag{18}$$

where $E_i^{\min}$ and $E_i^{\max}$ are minimum and maximum heat capacity per storage section, and further define

$$-\sigma \leq E_\sigma(t) \leq \sigma, \tag{19}$$

for state $E_\sigma(t)$, where σ is a measure for the maximal tolerated discomfort.

4.2 Control Experiments

We compare the control performance achieved by the approximation approach from Section 3 to the optimal solution, in which the original hybrid MPC problem (1) is solved at each time instant. We consider a MPC problem in reference tracking formulation for the system introduced in 4.1, where the task is to meet the thermal demand and track specific levels of stored energy, i.e., $r(t) = [E_{\mathrm{sh}}^{\mathrm{set}}(t), E_{\mathrm{dhw}}^{\mathrm{set}}(t), 0]^\intercal$. The comparison is performed for a 5 day simulation with sampling time $T_{\mathrm{s}} = 900\,\mathrm{s}$ and control horizon $N = 48$. Further details on the values of the disturbances can be found in Löhr et al. (2019). We use YALMIP (Löfberg (2004)) to setup MPC problems (1) and (5) for the domestic heating system and we use Gurobi to solve them.

In the following, we analyze various methods for the classification task required to approximate the binary input. First, we trained a neural network with the *fitnet* command from the *Deep Learning Toolbox*. We denote this approach by *NN+QP*. The network consists of one input layer with 8 nodes, two hidden layers with 8 and 4 nodes, respectively, and one output layer with 2 nodes. As node transfer functions, we assigned the *hyperbolic tangent sigmoid* function to the hidden layers and a linear function to the output layer. The input training data set was created like presented in Section 3.2 with 500 uniformly distributed points per element of θ. The corresponding target training data was obtained by choosing 5 combinations and solving (1) for the resulting 2500 values. We add the complete open-loop trajectory for every obtained solution to the training data set, i.e. we use all intermediate pairs $\{\theta_i, \delta_i^\star\}$ from $k = 0$ to $k = N - 1$, hence the total number of samples is $n_{\mathrm{p}} = 120000$. In the training procedure, *Bayesian Regularization back-propagation* with a maximum number of 3500 training epochs was chosen as a loss function. The performance criterion was a sum of squares error, and the goal tolerance was set to 10^{-8}. The neural net classifier was constructed in 171 s [1]. The construction of the support vector machine classifiers took longer since we had to construct 2 of them. The combined time was approximately 360 s. For the constructions of the SVMs, we chose to optimize the hyperparameters, which improved the resulting predictions. Results obtained with this combination of methods are labeled *SVM+QP*.

In the following, we compare simulation results for both classification methods (*NN+QP* and *SVM+QP*) with the optimal solution denoted by *MIQP*. Figures 3 – 4 indicate the very good match between the approximation approach based on a neural network. The output time profiles of the hybrid MPC solution and both analyzed combinations

[1] Both machine learning approaches were created on a PC with Core i5, 8 GB of RAM, and MATLAB R2019a.

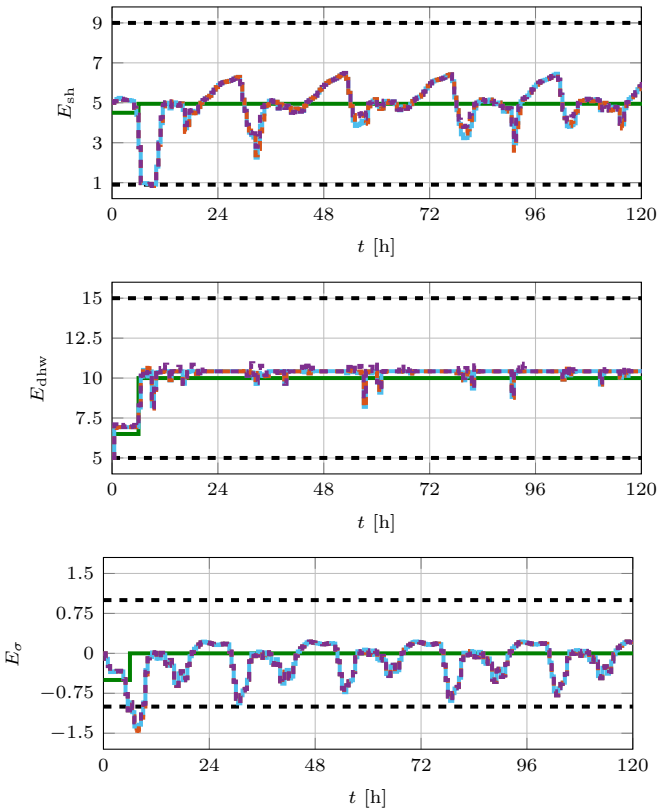

Fig. 3. Comparison of closed-loop profiles with hybrid MPC (blue line) and ML-enhanced control algorithm (dashed red line for *NN+QP*, dashed purple line for *SVM+QP*). The green line represents the reference; the black dashed lines represent the boundaries.

shown in Figure 3 are even almost identical, while there are differences in the results for inputs u_2, δ_1 and δ_2. This difference essentially originates from classification results of u_4 and u_5 which differ from the optimal solution, i.e., $\delta^{\mathrm{approx}}(t) \neq \delta^{\star}(t)$. In that case, two properties of our proposed approach are responsible for the almost equal trajectories. First, due to the receding horizon principle, the classification is repeated fast enough if not too many wrong classifications follow consecutively, which we will analyze in the next subsection. More important is the online solution of the relaxed QP (5). This solution determines the optimal open-loop trajectory with the limitation that the predicted values of $\delta(t)$ are not realizable. As the objective function does not change between *MIQP* and *NN+QP* simulations, $u(t)$ is operated to compensate for wrong classifications, which explains the differences in Fig. 4(b).

4.3 Analysis of Machine Learning Tools

We provide a quantitative performance comparison of the proposed approach for the ML-classification methods in Table 1.

The almost identical root mean square error value numerically shows the very good approximation capabilities already observed in Section 4.2. For both ML-methods used for classification, the performance decreases only slightly, while the online step was accelerated by a factor of about 3. We analyze the mean objective function values to

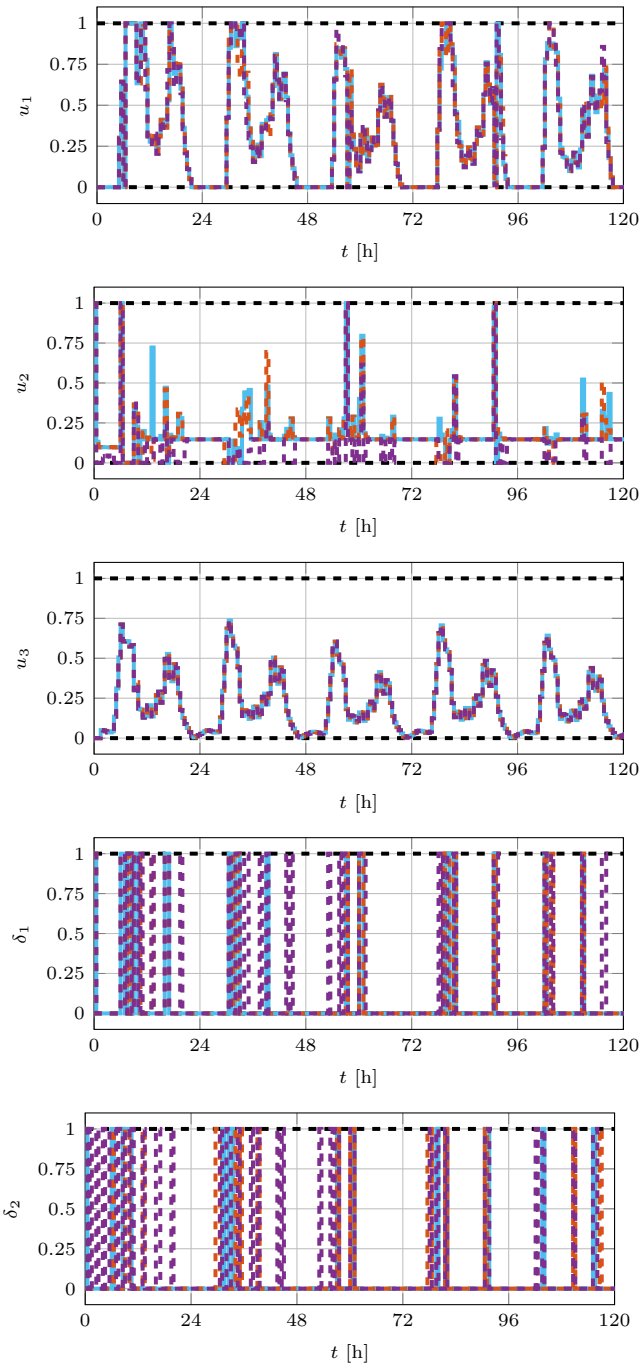

Fig. 4. Comparison of control input time profiles with hybrid MPC (blue line) and ML-enhanced control algorithm (dashed red line for *NN+QP*, dashed purple line for *SVM+QP*). Black dashed lines represent boundaries.

indicate the suboptimality of the approach. The increase is related to the fraction of misplaced binaries, i.e., to the number of simulation time steps where the classification output is not equal to the optimal solution. Evidence on the sensitivity to wrong classifications can be drawn from the two reference cases *R25* and *R0*, where 25% of binaries are placed wrong or all binaries are fixed to zero, i.e., $\delta^{\mathrm{approx}} = \mathbf{0}$, respectively.

Note again, that our proposed algorithm is suitable for hardware implementation, as only exactly one QP needs

Table 1. Quantitative comparison of mean on-line computation time, relative amount of misplaced binaries, root mean square error (RMSE), integral absolute error (IAE), integral squared error (ISE) and mean value of the objective function.

	Mean comp. time [ms]	Misplaced binaries [%]	RMSE [–]	IAE [–]	ISE [–]	Mean obj. value [–]
MIQP	67	–	0.569	103.52	94.83	21.6
NN+QP	23	4.8%	0.573	104.01	95.74	22.2
SVM+QP	20	11.6%	0.575	104.85	93.95	24.3
R25	33	25.0%	0.771	147.01	170.60	65.4
R0	28	13.0%	0.761	127.83	172.13	36.1

to be solved online. In contrast, the number of relaxations, i.e., the number of QP's that need to be solved to find the solution of the primary MIQP-MPC (1) scales with N and n_δ. Consequently, the computational demand for the MIQP-MPC easily exceeds the capability of embedded platforms for larger horizons or a higher number of binary inputs.

5. CONCLUSION AND OUTLOOK

The present paper proposes an embeddable reformulation of a mixed-integer MPC problem that uses machine-learning methods. A suitable ML-method approximates the binary part of the control law, while the continuous part is obtained by the solution of a relaxed QP problem. The mandatory classification task was executed based on a uniformly distributed training data set consisting of states, inputs, and disturbances, for which the MIQP-MPC problem was solved to get the corresponding optimal values for the binary inputs. A simulation study reveals that a lean NN with two hidden layers outperforms the SVM in terms of correct classifications. Numerical results show that, due to the two-stage structure of our approach, both ML-methods yield a comparable control performance. Moreover, the approach is suitable for embedded implementation and calculated the on-line step three times faster than the optimal MIQP solution.

While the considered example system is robust with respect to the observed number of wrong classifications, it will be constructive to improve the classification quality and extend the simulation study to further system classes. After successful further simulation studies, the approach can be implemented on embedded hardware and applied to real systems.

REFERENCES

Achterberg, T., Bixby, R.E., Gu, Z., Rothberg, E., and Weninger, D. (2016). Presolve reductions in mixed integer programming. Technical report, Zuse Institute Berlin, Berlin.

Bertsimas, D. and Stellato, B. (2019). Online mixed-integer optimization in milliseconds. Available from `https://arxiv.org/pdf/1907.02206`.

Bishop, C.M. (1995). *Neural Networks for Pattern Recognition*. Oxford University Press, Inc., New York, NY, USA.

Bixby, R.E. (2012). A brief history of linear and mixed-integer programming computation. *Documenta Mathematica*, 107–121.

Breiman, L., Friedman, J., Stone, C.J., and Olshen, R. (1984). *Classification and Regression Trees*. Chapman and Hall/CRC.

Domahidi, A. and Jerez, J. (2014–2019). Forces professional. Embotech AG. URL `https://embotech.com/FORCES-Pro`.

Drgoňa, J., Klaučo, M., and Kvasnica, M. (2015). MPC-based reference governors for thermostatically controlled residential buildings. In *2015 54th IEEE Conference on Decision and Control (CDC)*, 1334–1339. doi:10.1109/CDC.2015.7402396.

Drgoňa, J., Picard, D., Kvasnica, M., and Helsen, L. (2018). Approximate model predictive building control via machine learning. *Applied Energy*, 218, 199–216. doi:10.1016/j.apenergy.2018.02.156.

Drucker, H., Burges, C.J.C., Kaufman, L., Smola, A., and Vapnik, V. (1996). Support vector regression machines. In *Proceedings of the 9th International Conference on Neural Information Processing Systems*, NIPS'96, 155–161. MIT Press, Cambridge, MA, USA.

Ferrari-Trecate, G., Gallestey, E., Letizia, P., Spedicato, M., Morari, M., and Antoine, M. (2004). Modeling and control of co-generation power plants: A hybrid system approach. *IEEE Transactions on Control Systems Technology*, 12(5), 694–705.

Hertneck, M., Köhler, J., Trimpe, S., and Allgöwer, F. (2018). Learning an approximate model predictive controller with guarantees. *IEEE Control Systems Letters*, 2(3), 543–548.

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2), 251 – 257. doi:https://doi.org/10.1016/0893-6080(91)90009-T.

Klaučo, M., Kalúz, M., and Kvasnica, M. (2019). Machine learning-based warm starting of active set methods in embedded model predictive control. *Engineering Applications of Artificial Intelligence*, 77, 1–8. doi:10.1016/j.engappai.2018.09.014.

Löfberg, J. (2004). YALMIP : A Toolbox for Modeling and Optimization in MATLAB. In *Proc. of the CACSD Conference*. Taipei, Taiwan. Available from `http://users.isy.liu.se/johanl/yalmip/`.

Löhr, Y. and Mönnigmann, M. (2018). Optimal operation of electrical heating system with hybrid model predictive control. *IFAC-PapersOnLine*, 51(28), 274 – 279. doi:https://doi.org/10.1016/j.ifacol.2018.11.714. 10th IFAC Symposium on Control of Power and Energy Systems CPES 2018.

Löhr, Y., Mönnigmann, M., Klaučo, M., and Kalúz, M. (2019). Mimicking predictive control with neural networks in domestic heating systems. In *2019 22nd International Conference on Process Control (PC19)*, 19–24. doi:10.1109/PC.2019.8815030.

Lucia, S. and Karg, B. (2018). A deep learning-based approach to robust nonlinear model predictive control. *IFAC-PapersOnLine*, 51(20), 511 – 516. doi:https://doi.org/10.1016/j.ifacol.2018.11.038. 6th IFAC Conference on Nonlinear Model Predictive Control NMPC 2018.

Mayne, D.Q., Rawlings, J.B., Rao, C.V., and Scokaert, P.O.M. (2000). Constrained model predictive control: Stability and optimality. *Automatica*, 36(6), 789 – 814. doi:10.1016/S0005-1098(99)00214-9.

Morari, M. and Barič, M. (2006). Recent developments in the control of constrained hybrid systems. *Computers & Chemical Engineering*, 30(10), 1619 – 1631. Papers form Chemical Process Control VII.

Richards, A. and How, J. (2005). Mixed-integer programming for control. In *Proceedings of the 2005, American Control Conference, 2005.*, 2676–2683 vol. 4. doi:10.1109/ACC.2005.1470372.

Zheng, X., Sun, X., and Li, D. (2014). Improving the performance of MIQP solvers for quadratic programs with cardinality and minimum threshold constraints: A semidefinite program approach. *INFORMS Journal on Computing*, 26(4), 690–703.