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

Learning Approximate Semi-Explicit Hybrid MPC with an Application to Microgrids ⋆,⋆⋆

Daniele Masti * **Tomas Pippia** ** **Alberto Bemporad** * **Bart De Schutter** **

* *IMT School for Advanced Studies, Lucca, Italy (e-mail: daniele.masti@imtlucca.it, alberto.bemporad@imtlucca.it).*
** *Delft University of Technology, Delft, The Netherlands (e-mail: t.m.pippia@tudelft.nl, b.deschutter@tudelft.nl)*

Abstract: We present a semi-explicit formulation of model predictive controllers for hybrid systems with feasibility guarantees. The key idea is to use a machine-learning approach to learn a compact predictor of the integer/binary components of optimal solutions of the multiparametric mixed-integer linear optimization problem associated with the controller, so that, on-line, only a linear programming problem must be solved. In this scheme, feasibility is ensured by a simple rule-based engine that corrects the binary configuration only when necessary. The performance of the approach is assessed on a well known benchmark for which explicit controllers based on domain-specific knowledge are already available. Simulation results show how our proposed method considerably lowers computation time without deteriorating closed-loop performance.

Keywords: Model Predictive Control, Machine Learning, Mixed-Integer Optimization, Modeling and Simulation of Power Systems

1. INTRODUCTION

In recent years Model Predictive Control (MPC) has become one of the leading optimal control techniques in industry (Mayne, 2014) due to its innate ability to handle constraints and due to its flexibility in terms of applicable cost functions (Diehl et al., 2010). This flexibility comes however at a price: MPC requires solving a constrained optimization problem on-line at each control step. Although this issue has become less and less problematic in recent years thanks to the advancements in both computational capabilities of the hardware and solver technology, nonlinear MPC formulations on embedded hardware have been anyway mostly relegated to the control of either laboratory or very slowly changing systems. This is unfortunate as for some systems, such as the ones involving both logical and dynamical elements, MPC is often one of the few applicable general-purpose control techniques.

When MPC is used to control such "hybrid" dynamical systems (Bemporad and Morari, 1999), however, the optimization problem that must be solved at each control step becomes of mixed-integer nature and thus combinatorial (Morrison et al., 2016). In order to solve this kind of problems, most solvers rely on the so-called Branch and Bound (B&B) approach to efficiently explore the set of combinations of non-continuous variables. While such schemes are usually very efficient, B&B strategies cannot completely avoid the worst-case scenario in which all the combinations of binary variables must be tested. For this reason, solving this kind of problems is usually deemed infeasible in embedded applications as the hardware requirements needed to guarantee that a solution will eventually be found within strict real-time constraints may be excessive. On top of this, we also note that the industry-grade state-of-the-art commercial B&B solvers only provide libraries for desktop applications.

Explicit (hybrid) MPC was proposed as a way to avoid solving the optimization problem on-line (Borrelli et al., 2005; Alessio and Bemporad, 2006). In practice, however, this approach is limited to simple use cases due to the possible explosion of both computation time and memory requirements, which can rapidly grow as the number of binary variables and constraints increase, up to the point where using an explicit controller may well become more expensive than using an implicit one (Cimini and Bemporad, 2017).

A different and more practical approach to soften the computational requirements of a predictive controller is to relax the requirement for exact solutions to the original control problem. Indeed many authors have explored this topic and proposed solutions such as:

- approximating/learning the explicit control law (Karg and Lucia, 2018; Maddalena et al., 2019);
- using especially crafted on-line optimization schemes that do not aim to fully solve the problem at each step to get acceptable closed-loop control performance (Gros et al., 2020; Diehl et al., 2005; Axehill et al., 2014);
- exploiting the domain knowledge of the designer to formulate a rule-based decision engine to either speed up the solver (Di Cairano et al., 2012) or to directly find a sub-optimal but feasible configuration (Pippia et al., 2019).

Some authors have instead explored the idea of heuristically provide a good initial guess to warm start to the mixed integer solver (Ingimundarson et al., 2007; Bemporad and Naik, 2018) and then fully solve the problem to optimality. Recent works extended this concept by proposing machine-learning (ML) ap-

⋆ This work has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska–Curie grant agreement No 675318 (INCITE).
⋆⋆D. Masti and T. Pippia have contributed equally to this work.

proaches (Bertsimas and Stellato, 2018; Masti and Bemporad, 2019) that can learn from a collection of previously solved examples to provide warm starts also in situations where other heuristic approaches would usually fail. The principal limitation of those approaches is that they either cannot guarantee that a feasible solution will be always found within affordable time bounds or that they require deep understanding of the control problem, i.e., domain knowledge. However, if we aim at finding a feasible solution in broad sense, then there are many applications in which this can be done. This idea has led to interesting results such as the ones presented in (Jun et al., 2019). Consider for example the electric dispatch problem in a microgrid connected to the utility grid: it is trivial that a suboptimal but possible solution is simply to shut down every generator and buy all the required power from the grid.

In this context, we explore the possibility of lightening the computational requirements of hybrid MPC controllers applied to microgrid operations by merging the two aforementioned approaches: we use ML techniques to learn a map from the realizations of the parameters of the optimization problem to the corresponding optimal binary solutions and, in case the predictor fails to provide a feasible solution, resort to simple rules arising from the problem structure to correct otherwise infeasible guesses. The resulting binary configuration is then used to reduce the original mixed-integer linear programming (MILP) problem to a linear program (LP). This in turn allows the user to discard the mixed-integer solver altogether and possibly even ensure that a solution will be found within polynomial time (Nesterov and Nemirovskii, 1994). The validity of this approach is assessed on a grid-connected microgrid power dispatch problem benchmark. The reason why we focus on this application is because, to the best of our knowledge, it is the only field in which domain knowledge based explicit controllers are available as benchmark.

We note that our approach is not restricted to the considered test case. Indeed, the choice of the benchmark is due to the fact that, to the best of the authors' knowledge, it is one of the few explicit rule-based controllers we can use to compare our method against.

The paper is organized as follows: in Section 2 we recall the structure of the problem and a well known model usually employed to parametrize the associated predictive controller. Section 3 is devoted to present the employed learning architecture. In Section 4 we provide experimental results and a comparison with domain knowledge based techniques. Lastly, in Section 5 we draw some conclusions and present suggestions for future work.

2. BENCHMARK DESCRIPTION

In this section we briefly recall the problem of power dispatching in a microgrid environment, i.e., an electrical power system in which storage elements (e.g., batteries and ultracapacitors, also called storage units), local generators, a bidirectional connection to the main grid, and uncontrollable loads are present at the same time.

Here the main goal of the energy management system is to solve the so called dispatch problem, i.e., provide the required power $P_{\text{load}}(k)$ to the uncontrollable loads at each time step k while minimizing the economic costs associated to the source of the energy. Many contributions in the literature focused on how to effectively model this kind of environment. In this work we will in particular consider a variation [1] of the model presented in Parisio et al. (2014) and we will also assume that both the values of P_{load}, P_{res} (the power output of the renewable sources) and the upper bounds on the power output of the dispatchable units are known and constant between two sampling times. A scheme of the considered microgrid is shown in Figure 1.

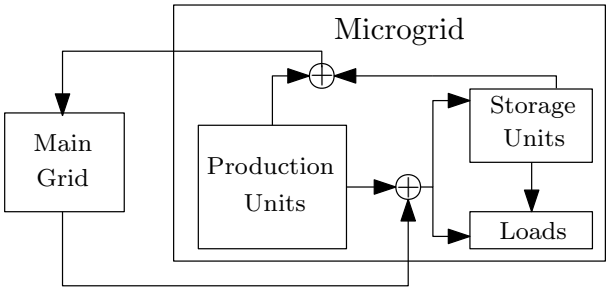

Fig. 1. Microgrid scheme considered in this work. Arrows represent power flows. The microgrid is in non-islanded mode, i.e., it is connected to the main grid.

2.1 Dynamical model of the energy storage systems

In this work we restrict our analysis to microgrids where the energy storage system can be modeled as a battery obeying the following hybrid dynamical law:

$$x_{\text{b}}(k+1) = \begin{cases} x_{\text{b}}(k) + \frac{T_{\text{s}}}{\eta_{\text{d}}} P_{\text{b}}(k) & \text{if } P_{\text{b}}(k) < 0 \\ x_{\text{b}}(k) + T_{\text{s}} \eta_{\text{c}} P_{\text{b}}(k) & \text{if } P_{\text{b}}(k) \geq 0 \end{cases} \quad (1)$$

where the state $x_{\text{b}}(k)$ indicates the level of energy stored in the energy storage system at time step k, η_{c} and η_{d} are the charging and discharging efficiencies, respectively, $P_{\text{b}}(k)$ is the power exchanged with the energy storage system at time step k, and T_{s} is the sampling interval of the discrete-time system.

As this kind of models are not immediately easily exploitable for control applications, we follow the Mixed Logical Dynamical approach (MLD) (Bemporad and Morari, 1999) and introduce a Boolean (binary) variable $\delta_{\text{b}}(k)$ to indicate whether the energy storage system is in the charging (i.e $\delta_{\text{b}}(k) = 1 \iff P_{\text{b}}(k) \geq 0$) or discharging mode at time step k. This allows us to rewrite model (1) more compactly as

$$x_{\text{b}}(k+1) = x_{\text{b}}(k) + T_{\text{s}} \left(\eta_{\text{c}} - \frac{1}{\eta_{\text{d}}} \right) z_{\text{b}}(k) + \frac{T_{\text{s}}}{\eta_{\text{d}}} P_{\text{b}}(k), \quad (2)$$

where $z_{\text{b}}(k) = \delta_{\text{b}}(k) P_{\text{b}}(k)$. Both the logic relationship between $\delta_{\text{b}}(k)$ and $P_{\text{b}}(k)$ and the definition of $z_{\text{b}}(k)$ can be recast using a set of mixed-integer linear inequalities (Bemporad and Morari, 1999).

2.2 Generator units

We consider two kinds of generation units in the grid: renewable sources (i.e., zero cost uncontrollable power sources with known but time-varying power output) and dispatchable generators. Differently from renewable energy sources, dispatchable generators can be controlled in terms of output power. In other words, the power they produce can be manipulated within some bounds and can therefore be considered as a control variable.

[1] We disregard the constant loss term in the energy storage system dynamics for simplicity.

We denote by $\boldsymbol{P}_{\rm dis}(k)$ the vector representing the power produced by dispatchable generators, i.e.,

$$\boldsymbol{P}_{\rm dis}(k) \;=\; \left[P_1^{\rm dis}(k), \ldots, P_{N_{\rm gen}}^{\rm dis}(k)\right]^\top,$$

where $P_i^{\rm dis}(k)$ indicates the power produced by dispatchable unit i, $i = 1, \ldots, N_{\rm gen}$ at time step k and $N_{\rm gen}$ is the total number of dispatchable units.

We use a binary variable $\delta_i^{\rm on}(k)$ to indicate whether the dispatchable generator i is active at time step k ($\delta_i^{\rm on}(k) = 1$) or not ($\delta_i^{\rm on}(k) = 0$).

2.3 Energy prices

All the energy flows in the systems have an associated cost. There are thus three different prices: $c_{\rm buy}(k)$ is the purchase price, $c_{\rm sale}(k)$ is the sale price, and $c_{\rm prod}(k)$ is the price for producing electricity with the dispatchable units. Note that the prices $c_{\rm buy}(k)$ and $c_{\rm sale}(k)$ are quantities related to the main grid. We also assume that prices are known in advance, e.g., by means of a predictor.

2.4 Main grid

The microgrid that we consider is connected to the main grid. This connection can be modeled as a binary variable $\delta_{\rm grid}(k)$ indicating whether energy is being bought from the main grid ($\delta_{\rm grid}(k) = 1$) or sold to it ($\delta_{\rm grid}(k) = 0$) at time step k. We denote by $P_{\rm grid}$ the power exchanged with the main grid, obtaining

$$\begin{cases} \delta_{\rm grid}(k) = 0 \iff P_{\rm grid}(k) < 0 \text{ (exporting case)} \\ \delta_{\rm grid}(k) = 1 \iff P_{\rm grid}(k) \geq 0 \text{ (importing case).} \end{cases} \tag{3}$$

In order to model the economic cost and revenue when exchanging energy with the main grid, we can resort to an auxiliary variable $C_{\rm grid}$ defined as

$$\begin{cases} C_{\rm grid}(k) = c_{\rm sale}(k) P_{\rm grid}(k) \iff P_{\rm grid}(k) < 0, \\ C_{\rm grid}(k) = c_{\rm buy}(k) P_{\rm grid}(k) \iff P_{\rm grid}(k) \geq 0. \end{cases} \tag{4}$$

Once again, by using the MLD framework, we can link together $\delta_{\rm grid}$ and $C_{\rm grid}$ by resorting to a set of mixed-integer linear constraints.

3. OPTIMIZATION PROBLEM

3.1 Control problem formulation

The goal of the controller is to handle the optimization of the energy management system of the microgrid, choosing how much energy to produce or trade with the main grid while guaranteeing the satisfaction of both load needs and physical constraints of the grid. It is then clear that a constrained optimization-based control technique, and in particular MPC, is a natural choice for this task. We note that the use of a hybrid MPC formulation for this task is indeed well known in the literature and promising results have been obtained by multiple authors (Parisio et al., 2016; Cominesi et al., 2018; Velarde et al., 2017). The formulation we adopt in this work will be the one described in (Pippia et al., 2019). The control problem formulation is the following:

$$\min_{\substack{\boldsymbol{P}_{\rm dis}(k),\, P_{\rm grid}(k),\\ P_{\rm b}(k),\, \boldsymbol{\delta}(k),\, \boldsymbol{z}(k)}} J(\boldsymbol{P}_{\rm dis}(k), C_{\rm grid}(k), c_{\rm prod}(k)) \tag{5}$$

subject to:

$$E_1 \boldsymbol{\delta}(k) + E_2 \boldsymbol{z}(k) \leq E_3 \boldsymbol{u}(k) + E_4 \tag{6a}$$

$$P_{\rm b}(k) = \sum_{i=1}^{N_{\rm gen}} P_i^{\rm dis}(k) + \\ + P_{\rm res}(k) + P_{\rm grid}(k) - P_{\rm load}(k) \tag{6b}$$

$$\underline{P}_{\rm b} \leq P_{\rm b}(k) \leq \overline{P}_{\rm b} \tag{6c}$$

$$\underline{P}_{\rm grid} \leq P_{\rm grid}(k) \leq \overline{P}_{\rm grid} \tag{6d}$$

$$\delta_i^{\rm on}(k) \underline{P}_i^{\rm dis} \leq P_i^{\rm dis}(k) \leq \delta_i^{\rm on}(k) \overline{P}_i^{\rm dis} \tag{6e}$$

$$\underline{x}_{\rm b} \leq x_{\rm b}(k) \leq \overline{x}_{\rm b} \tag{6f}$$

$$\boldsymbol{u}(k) = \left[\boldsymbol{P}_{\rm dis}^\top(k),\; C_{\rm grid}(k),\; P_{\rm b}(k)\right]^\top$$

$$\text{for } i = 1, \ldots, N_{\rm gen}$$

$$\text{for } k = 0, \ldots, N_{\rm p} - 1$$

where $N_{\rm p}$ is the prediction horizon, and constraint (6a) arises from the MLD formulation of the model presented in the previous sections (interested readers may also refer to Parisio et al. (2014) for a detailed description). Constraint (6b) represents power balance and ensures that at each time step all the generated power is either consumed, stored, or sold to the main grid. The other constraints are related to the limits that all the power contributions must respect. In particular, in constraint (6f), $\underline{x}_{\rm b}$ and $\overline{x}_{\rm b}$ represent the lower and upper bound of the state of charge of the battery. Constraints (6c)–(6e) model the physical bounds on, respectively, the power exchanged with the battery, the power exchanged with the main grid, the power produced by each production unit i, i.e., $P_i^{\rm dis}(k)$, and the level of charge of the battery.

The cost function chosen is a sum of economic costs and arises from the local energy production through dispatchable units and from the exchange of energy with the main grid:

$$J(\boldsymbol{P}_{\rm dis}(k), C_{\rm grid}(k), c_{\rm prod}(k)) = \\ \sum_{j=0}^{N_{\rm p}-1} \left(C_{\rm grid}(k+j) + c_{\rm prod}(k+j) \sum_{i=1}^{N_{\rm gen}} P_i^{\rm dis}(k+j) \right) \tag{7}$$

3.2 Towards a data-driven explicit controller

It is well known in the literature that solving an MILP problem is a $\mathcal{NP}$-hard problem in general. Nevertheless, by treating $x(0)$, $c_{\rm buy}$, $c_{\rm sale}$, $c_{\rm prod}$, $P_{\rm load}$, $P_{\rm res}$ as *parameters* of the problem, and by observing that the *structure* of the problem remains the same at each time step, we can build a machinery that can lead to an explicit MPC controller. This idea becomes even more appealing as we do not actually need to build such a predictor for all the decision variables as, once the binary components are set, the real-valued ones can be found separately. As practical matter, this would translate into relaxing the original MILP based controller into a partially explicit controller that must solve only an LP at each time step. Figure 2 shows the resulting optimization scheme.

In particular, for what concerns problem (5)–(6f), a possible approach to achieve this is then the following:

(1) extract a representative dataset of N examples of parametric realizations,

$$Z_N = \{(x_0(k), c_{\rm buy}^i(k), c_{\rm prod}^i(k), \\ c_{\rm sale}^i(k), P_{\rm load}^i(k), P_{\rm res}^i(k))\} \tag{8}$$

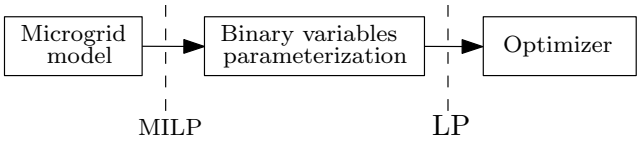

Fig. 2. Proposed solution scheme.

with $Z_N \in \mathbb{R}^{(1+5N_\mathrm{p})\times N}$, $\forall i = k, \ldots, k+N_\mathrm{p}-1$, $\forall k = 1, \ldots, N$.

(2) solve off-line the corresponding optimization problems
(3) extract the set of optimal binary tuples associated to the components of the grid at each step in the prediction horizon:

$$O_N^j = \{(\delta_1^{\mathrm{on}(i)}(k), \ldots, \delta_{N_\mathrm{gen}}^{\mathrm{on}(i)}(k), \delta_\mathrm{grid}^i(k), \delta_\mathrm{b}^i(k))\}$$

where each $O_N^j \in \{0,1\}^{(N_\mathrm{gen}+2)\times N}$, $\forall i = k+j, \forall j = 0, \ldots, N_\mathrm{p}-1, \forall k = 1, \ldots, N$
(4) use machine learning/function approximation methods to build a map from parameter values to binary tuples.

Multiple contributions have already explored the use of ML techniques for predicting the optimal active set of optimization problems. However, most of such approaches were meant to achieve the best possible predicting performance with little regard of any secondary use the learned classifier may need to serve, e.g., providing facilities to assess the correctness and robustness of the given prediction. For this reason, in this work, we focus on the idea of using more interpretable techniques. This immediately draws a comparison with other kind of interpretable heuristic rule based approaches, such as one presented in (Pippia et al., 2019), whose decision engine will be used as comparison term for the rest of the paper.

As we are dealing with real-time applications, we also need a learning architecture with a small computational footprint and possibly running also in bounded time for throughput predictability. As the quantities we are trying to predict are binary in nature, a natural choice is to resort to a decision-tree classifier (Breiman et al., 1984) with a limited a-priori number of nodes for which both the decision path of each prediction is clearly inspectable (Marchese Robinson et al., 2017) and a vast literature about establishing the importance of each provided input feature exists. One of the principal limitations of decision trees is however their instability. For this reason we also consider the use of random-forest classifiers (Hastie et al., 2009), which try to solve this issue by bagging more decision trees together, at the cost of both a more problematic interpretability and higher computational requirements.

Each predictor is trained to predict the tuple of binary decision variables corresponding to the action of a specific time step within the prediction window. In practice, this means that we will have N_p predictors, each one trained on a different dataset tuple $\left(Z_N, O_N^j\right)$. The reason for this choice is twofold: on the one hand, very small trees simply lack the approximation power required to efficiently predict hundreds of outputs at the same time and, on the other hand, it would greatly facilitate the user in establishing which input feature influences which output. In both cases, the loss function used to grow the classifiers is the entropy criterion (Hastie et al., 2009).

3.3 Prediction override for avoiding infeasibility

The proposed approach is still not yet able to ensure the feasibility of the prediction with respects to the constraints (6). A possible approach to avoid infeasibility in this case is to inspect the behavior of the proposed predictor, categorize the cases of bad behavior (i.e., infeasible predictions), and implement a fail-safe override mechanism that ensures feasibility in such specific occasions. While this is in general as hard as designing a whole explicit controller, in many systems (including the one we analyze) trivial feasible configurations are indeed simple to recover. Moreover, we note that even in case the prediction leads to an infeasible configuration, it will probably still be close to the real optimal one. This means that the task the user is asked will not be to design a complete substitute optimal controller as a whole, but rather to simply provide a limited set of feasibility corrections, without the need of caring about optimality.

Based on the previous discussion, we consider the three following possible sources of infeasibility:

(1) $\sum_{i=1}^{N_\mathrm{gen}} \delta_i^{\mathrm{on}}(k)\overline{P}_i^{\mathrm{dis}} < P_\mathrm{load}(k) - P_\mathrm{res}(k)$ AND $\delta_\mathrm{grid}(k) = 0$, i.e., the local production units alone are not able to satisfy the loads but the grid is set to export mode. In this case, we override $\delta_\mathrm{grid}(k)$ and set it to $\delta_\mathrm{grid}(k) = 1$, i.e., we set the grid to import mode;
(2) $P_\mathrm{res}(k) - P_\mathrm{load}(k) > \delta_\mathrm{b}(k)|\underline{P}_\mathrm{b}(k)|$ AND $\delta_\mathrm{grid}(k) = 1$, i.e., there is a surplus of generation, higher than the power that the battery can absorb, but the grid is set to import mode. In this case, $\delta_\mathrm{grid}(k)$ is set to 0, i.e., to export mode;
(3) $0 < P_\mathrm{load}(k) - P_\mathrm{res}(k) < \sum_{i=1}^{N_\mathrm{gen}} \delta_i^{\mathrm{on}}(k)\underline{P}_i^{\mathrm{dis}}$ AND $\delta_\mathrm{grid}(k) = 1$, i.e., the loads are higher than renewable power and the minimum power that can be produced with the dispatchable units is higher than the necessary extra energy to satisfy the loads, but the grid is set to import mode. In other words, in this specific case, there is a small excess of energy coming from the dispatchable units that has to be exported to the main grid. Therefore, we override the rules setting the main grid to export case, i.e., $\delta_\mathrm{grid}(k) = 0$.

4. SIMULATIONS

4.1 Setup

Simulations were carried out solving problem (5)–(6) subject to the aforementioned parameterization of the binary variables through ML algorithms. We focus in particular on a Random Forest method (RF7) and a Decision Tree (DT7) with maximum depth of 7 levels. The level of depth chosen is a trade-off between complexity and approximation power. As benchmarks, we consider both the full MILP original problem and the rule-based (RB) approach presented in (Pippia et al., 2019), which, as explained earlier, requires a considerable amount of prior domain knowledge.

The classifiers were trained using $\approx$ 16000 samples obtained by solving the real MILP optimization problem with Gurobi (Gurobi Optimization Inc., 2016) and using real data for the renewable energy sources and the loads from year 2018 taken from the ENTSO-E Transparency Platform (Hirth et al., 2018), while the prices profiles have been designed similarly to (Pippia et al., 2019). The amount of dispatchable units is set to $N_\mathrm{gen} =$

3, the sampling time is $T_s = 30$ min, and the prediction horizon of the MPC algorithm is $N_p = 48$, corresponding to 24h. This in turn results in $(1 + 1 + N_{gen}) \cdot 48 = 240$ binary variables in the optimization problem. Moreover, each simulation considers a simulation time of one day. We note that all the real value components of the dataset were normalized using the empirical mean and standard deviation of the training set.

In order to assess the performance of the proposed methods, we performed 150 simulations using renewable sources and loads data from year 2017. Hence, the total number of optimization problems solved for each method is $150 \cdot \frac{N_p}{T_s} = 7200$.

The training procedure of each classifier was carried out using negligible computational resources on a machine equipped with an Intel core I7-8565U and 16 GB of RAM. The implementation was carried out using the Scikit-learn (Buitinck et al., 2013).

4.2 Results

We compare three different measures for all the methods:

(1) the average open-loop and closed-loop costs;
(2) computation time;
(3) the amount of infeasible configurations for each method.

Regarding the performance in terms of costs, Table 1 shows the average open-loop and closed-loop costs, in €, associated to the binary configurations produced by the predictors. The open-loop cost is the value of the cost (7) obtained after a single optimization of problem (5), while the closed-loop cost is the cost computed at the end of a simulation, when all the inputs applied to the system are known. Both the ML and the RB methods achieve a similar value of the open-loop cost, with RB being slightly worse than the proposed approach. For what concerns the closed-loop cost, the three parametrization methods achieve very similar performance to the MILP one, with a difference of at most 1.3%. Note that the RB approach achieves a smaller value than the MILP one. This is simply due to the fact that the RB approach leaves a smaller charge in the battery at the end of the simulation, thus leading to a smaller value of the closed-loop cost, as explained in (Pippia et al., 2019).

Table 2 compares the on-line computation time of all the methods. Moreover, for the ML and RB methods, we also show the percentage of decrease with respect to the MILP case and the standard deviation. For all the parameterized methods, we can notice a tremendous decrease in computation time of at least 96%. This was expected due to the fact that the parameterized methods solve only one linear programming problem instead of a mixed-integer one. Furthermore, in Figure 3 we show the elapsed run time of each single simulation, with the y-axis in log-scale. We can notice from the figure that, while the MILP approach has a certain variability in total simulation time, for the other methods run-time is quite constant.

As already noted, the predictors might sometimes lead to infeasible configurations for what concerns binary variables. To explore how often this happens, in Table 3 we show the number of infeasible binary variable configurations for all the parametrized methods, i.e., how many times on average the override explained in Section 3.3 must be applied. Note that the RB method does not yield infeasible configurations, as it was designed using domain knowledge to avoid this issue.

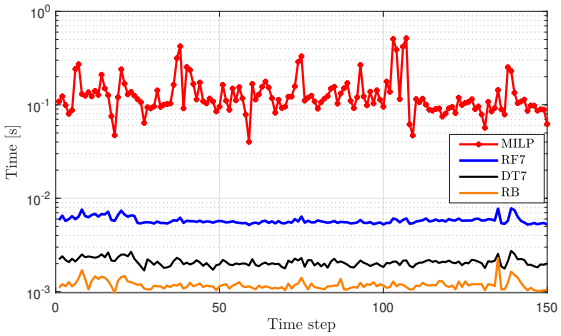

Fig. 3. Elapsed run times of each single simulation. The y-axis is in log-scale.

While both architectures are quite robust to this issue, it is also apparent that in this case RF7 outperforms DT7.

	OL	CL
MILP	4202.4	4514.5
RB	4341.1 (3.3%)	4479.3 (−0.8%)
RF7	4213.4 (0.3%)	4574.5 (1.3%)
DT7	4213.4 (0.3%)	4761.9 (0.9%)

Table 1. Average open-loop (OL) and closed-loop (CL) costs of each simulation performed. The percentage shows the increase in the cost w.r.t. the MILP case. The costs are in €.

	CPU time	% Decrease
MILP	6.47(3.72)	-
RB	0.06(0.01)	99%
RF7	0.27(0.02)	96%
DT7	0.10(0.01)	98%

Table 2. Average computation time of each simulation performed, in seconds. The standard deviation σ is shown between brackets. The percentage shows the decrease w.r.t. the MILP case.

	RB	RF7	DT7
% infeasible	0%	0.71%	6.85%

Table 3. Amount of infeasible binary variable configurations for each parametrized method.

4.3 Discussion

From the simulation results, it can be seen how the ML methods presented in this article are able to achieve in general a similar cost and computation time w.r.t. the RB method. Moreover, compared to the MILP method, the ML methods guarantee a much faster on-line run time while having a slightly worse performance in terms of costs. However, as explained in the previous sections, the need of solving a complex MILP problem is removed, which in turn implies that there is no need to include expensive and dedicated hardware, as well as complex MILP solvers, in the controller implementation. Furthermore, the limited increase in the cost and the huge decrease in computation time, together with the fact that there is only a very small amount of domain knowledge needed to implement the controller, justifies the adoption of our approach, even when compared to the RB method. Lastly, when comparing the two ML methods in particular, i.e., RF7 and DT7, the DT7 method shows a higher infeasibility rate but it also shows a lower closed-loop cost and lower computation times w.r.t. the RF7 method. Given that the differences between the two methods are quite small, we can safely claim that the usage of either of the two methods, in this particular application, is equivalent.

5. CONCLUSIONS

In this paper we have explored the use of machine learning to get approximate semi-explicit formulations of a hybrid MPC controller, with feasibility guarantees given by the use of a simple and compact rule-based engine. Compared to a pure rule-based formulation, the proposed approach requires much less domain knowledge from the user. The effectiveness of the approach has been tested on a well-known energy management system benchmark and compared to a pure rule-based solution already available in the literature, showing reduced computation time with similar performance with respect to the original problem. In future work we will consider the integration of the mathematical structure of the problem in the learning procedure in order to achieve even better performance.

REFERENCES

Alessio, A. and Bemporad, A. (2006). Feasible mode enumeration and cost comparison for explicit quadratic model predictive control of hybrid systems. In *2nd IFAC Conf. on Analysis and Design of Hybrid Systems*, 302–308.

Axehill, D., Besselmann, T., Raimondo, D.M., and Morari, M. (2014). A parametric branch and bound approach to suboptimal explicit hybrid MPC. *Automatica*, 50(1), 240–246.

Bemporad, A. and Morari, M. (1999). Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3), 407–427.

Bemporad, A. and Naik, V.V. (2018). A numerically robust mixed-integer quadratic programming solver for embedded hybrid model predictive control. In *6th IFAC Conf. on Nonlinear Model Predictive Control*, 412–417.

Bertsimas, D. and Stellato, B. (2018). The voice of optimization. `https://arxiv.org/abs/1812.09991`.

Borrelli, F., Baotić, M., Bemporad, A., and Morari, M. (2005). Dynamic programming for constrained optimal control of discrete-time linear hybrid systems. *Automatica*, 41(10), 1709–1721.

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

Buitinck, L. et al. (2013). API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 108–122.

Cimini, G. and Bemporad, A. (2017). Exact complexity certification of active-set methods for quadratic programming. *IEEE Trans. Automatic Control*, 62(12), 6094–6109.

Cominesi, S.R., Farina, M., Giulioni, L., Picasso, B., and Scattolini, R. (2018). A two-layer stochastic model predictive control scheme for microgrids. *IEEE Trans. on Control Systems Technology*, 26(1), 1–13.

Di Cairano, S., Tseng, H.E., Bernardini, D., and Bemporad, A. (2012). Vehicle yaw stability control by coordinated active front steering and differential braking in the tire sideslip angles domain. *IEEE Trans. on Control Systems Technology*, 21(4), 1236–1248.

Diehl, M., Amrit, R., and Rawlings, J.B. (2010). A Lyapunov function for economic optimizing model predictive control. *IEEE Trans. Automatic Control*, 56(3), 703–707.

Diehl, M., Bock, H.G., and Schlöder, J.P. (2005). A real-time iteration scheme for nonlinear optimization in optimal feedback control. *SIAM Journal on control and optimization*, 43(5), 1714–1736.

Gros, S., Zanon, M., Quirynen, R., Bemporad, A., and Diehl, M. (2020). From linear to nonlinear MPC: bridging the gap via the real-time iteration. *International Journal of Control*, 93(1), 62–80.

Gurobi Optimization Inc. (2016). Gurobi optimizer reference manual. `https://www.gurobi.com`.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media.

Hirth, L., Mühlenpfordt, J., and Bulkeley, M. (2018). The ENTSO-E Transparency Platform – A review of Europe's most ambitious electricity data platform.

Ingimundarson, A., Ocampo-Martinez, C., and Bemporad, A. (2007). Model predictive control of hybrid systems based on mode-switching constraints. In *Proc. of 46th IEEE Conf. on Decision and Control*, 5265–5269. New Orleans, LA.

Jun, S., Lee, S., and Chun, H. (2019). Learning dispatching rules using random forest in flexible job shop scheduling problems. *International Journal of Production Research*, 57(10), 3290–3310.

Karg, B. and Lucia, S. (2018). Deep learning-based embedded mixed-integer model predictive control. In *Proc. of European Control Conf.*, 2075–2080.

Maddalena, E.T., da S. Moraes, C.G., Waltrich, G., and Jones, C.N. (2019). A neural network architecture to learn explicit MPC controllers from data. `https://arxiv.org/abs/1911.10789`.

Marchese Robinson, R.L., Palczewska, A., Palczewski, J., and Kidley, N. (2017). Comparison of the predictive performance and interpretability of random forest and linear models on benchmark data sets. *Journal of Chemical Information and Modeling*, 57(8), 1773–1792.

Masti, D. and Bemporad, A. (2019). Learning binary warm starts for multiparametric mixed-integer quadratic programming. In *2019 18th European Control Conference (ECC)*, 1494–1499.

Mayne, D. (2014). Model predictive control: Recent developments and future promise. *Automatica*, 50(12), 2967–2986.

Morrison, D.R., Jacobson, S.H., Sauppe, J.J., and Sewell, E.C. (2016). Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning. *Discrete Optimization*, 19, 79–102.

Nesterov, Y. and Nemirovskii, A. (1994). *Interior-Point Polynomial Algorithms in Convex Programming*, volume 13. SIAM.

Parisio, A., Rikos, E., and Glielmo, L. (2014). A model predictive control approach to microgrid operation optimization. *IEEE Trans. on Control Systems Technology*, 22(5), 1813–1827.

Parisio, A., Rikos, E., and Glielmo, L. (2016). Stochastic model predictive control for economic/environmental operation management of microgrids: An experimental case study. *Journal of Process Control*, 43, 24–37.

Pippia, T., Sijs, J., and De Schutter, B. (2019). A single-level rule-based model predictive control approach for energy management of grid-connected microgrids. *IEEE Trans. on Control Systems Technology*, 1–13.

Velarde, P., Valverde, L., Maestre, J.M., Ocampo-Martinez, C., and Bordons, C. (2017). On the comparison of stochastic model predictive control strategies applied to a hydrogen-based microgrid. *Journal of Power Sources*, 343, 161–173.