Learning binary warm starts for multiparametric mixed-integer quadratic programming

Daniele Masti and Alberto Bemporad

Abstract— **In this paper we propose a lightweight neural network architecture that is able to learn the binary components of the optimal solution of a class of multiparametric mixed-integer quadratic programming (MIQP) problems, such as those that arise from hybrid model predictive control formulations.**

The predictor provides a binary warm-start to a specifically designed branch and bound (B&B) algorithm to quickly discover an integer-feasible solution of the given MIQP, with the aim of reducing the overall solution time required to find the global optimal solution on line.

I. Introduction

In recent years model predictive control (MPC) has become one of the leading advanced control techniques in industry [1]–[3] due to its innate ability to easily handle constraints on inputs and outputs. Contrarily to other control methodologies, MPC requires solving a constrained optimization problem on line at each sample step. For this reason, most used MPC schemes are formulated using convex, mostly quadratic, formulations of the control problem, due to the large availability of good solvers that can successfully and reliably run even on limited hardware (such as microcontrollers) [4].

When MPC is used to control hybrid dynamical systems [5], the optimization problem to solve in real-time becomes more challenging in embedded applications, due to the presence of binary variables that make the problem of mixed-integer nature, and therefore combinatorial [6]. Most mixed-integer programming (MIP) solvers rely on a branch and bound (B&B) scheme to efficiently explore the set of combinations of binary variables, usually represented as a binary tree. Explicit MPC was proposed as a way to avoid on-line mixed-integer programming [7], [8]. However, the approach is limited in practice to simple hybrid MPC problems, due to the possible explosion of memory and CPU requirements as a function of the number of binary variables and constraints. For a comparison of explicit MPC with (implicit) MPC based on on-line optimization in the case of linear MPC and quadratic programming the reader is referred to [9], where it is highlighted that in many cases the memory and computation requirements of implicit MPC are less demanding than for explicit MPC, even in the worst case.

A possible strategy to lighten computational requirements is to resort to approximate solutions of the optimization problem [10]. This is also justified theoretically, as closed-loop stability of a hybrid MPC system is guaranteed as soon as the optimal cost decreases from one sample step to the next [5]. An approximate explicit MPC solution can be obtained in two ways: (i) by approximating/learning the explicit control law [11], [12], or (ii) by using on-line optimization algorithms that do not require to fully solve the problem to optimality to get closed-loop control performance of acceptable quality.

The former approach has the limitation of possibly requiring the computation of many optimal solutions offline, one per sample of the vector of parameters (such as states and reference signals), to construct the approximation. In both cases, however, approximating the control law and renouncing to exact optima can bring unacceptable effects in the closed-loop performance. For this reason, many research efforts have instead focused on how to speed up the solution of the MIP problem to optimality, for example by providing a good initial guess to warm start the solver. A classical example is to provide the shifted solution found at the previous sample step and organize the B&B algorithm to take that into account [13], or to limit the number of possible switches of binary variables with respect to that solution [14]. Promising results have been also recently shown in [15] in using binary warm starts in MIP. The main limitation of such warm starting is that it does not perform well in case of sudden set-point changes or disturbance steps.

In this paper, we propose a B&B solver for mixed-integer quadratic programming (MIQP) problems that exploits binary warm starts. The key idea is to synthesize a lightweight function using machine learning techniques that acts as a suboptimal multiparametric solution of the binary components of the optimizer as a function of the parameters affecting the MIQP problem, and to use online the estimate provided by the learned function as an initial guess for the binary variables. When learning such a function, the emphasis is set on aiming at providing an integer-feasible initial guess, so that the MIQP solver is greatly advantaged when both an exact solution is sought and in the case the solver is stopped prematurely.

The paper is organized as follows. In Section II we introduce the class of multiparametric MIQP problems we deal with. In Section III we describe the structure of the proposed binary warm-start predictor and develop a branch-and-bound strategy in Section IV that is able to take advantage of the specific structure of the prediction. Finally, in Section V we report the numerical results obtained with the proposed methodology.

The authors are with the IMT School for Advanced Studies Lucca, Piazza San Francesco 19, 55100 Lucca {daniele.masti,alberto.bemporad}@imtlucca.it

II. Multiparametric mixed integer optimization problem

In this paper we consider multiparametric mixed integer quadratic optimization problems of the following form

$$\begin{aligned} \min_{z,\delta} \quad & \frac{1}{2}\begin{bmatrix} z \\ \delta \end{bmatrix}' Q \begin{bmatrix} z \\ \delta \end{bmatrix} + x'F' \begin{bmatrix} z \\ \delta \end{bmatrix} \\ \text{s.t.} \quad & G \begin{bmatrix} z \\ \delta \end{bmatrix} \leq W + Sx \\ & \delta \in \{0,1\}^q,\ z \in \mathbb{R}^m,\ x \in \mathbb{R}^n \end{aligned} \tag{1}$$

where $Q = Q' \succeq 0$, $Q \in \mathbb{R}^{(m+q)\times(m+q)}$, $F \in \mathbb{R}^{(m+q)\times n}$, $G \in \mathbb{R}^{p\times(m+q)}$, $W \in \mathbb{R}^p$, $S \in \mathbb{R}^{p\times n}$. For example in control applications a problem of the form (1) arises when formulating MPC problems based on mixed-logical dynamical (MLD) models of hybrid systems [5]. In this case, vector x contains all the parameters upon which the result of the optimization problem depends, such as current state and current (and possibly future) reference and measured disturbance signals. Problem (1) must be solved at each sample time t, $t = 0, 1, \ldots$, for the given value $x(t)$.

Problem (1) is typically solved by branch and bound (B&B) methods [16] together with different algorithms for solving quadratic programming (QP) relaxations, in which some of the variables δ_i are relaxed in the range $[0, 1]$ or fixed at 0 or 1. Possible QP solvers include active-set (AS) methods [17], [18], such as AS methods based on nonnegative least squares (NNLS) [19], [20] and dual active-set methods [21]. In particular, some AS methods benefit from warm-starting the active set, an information that is available during B&B. Other methods for solving QP relaxations within B&B include interior-point methods [22], accelerated gradient projection methods [23], and the alternating direction methods of multipliers [24].

The computational efficiency of an MIQP solver does not only depend on the way QP relaxations are solved, but also on the way B&B is performed. In particular, both the *branching* strategy and the availability of good *bounds* on the optimal solution influence the performance of the MIQP solver, which ultimately depends on the number and sizes of QP relaxations solved. In particular, during B&B a node of the search tree is fathomed when the associated QP relaxation leads to a cost that is not lower than the best currently available cost of an *integer feasible solution* (i.e., a solution that satisfies the constraints in (1). Therefore, knowing a *feasible configuration* of the binary variable δ, that is a value of $\delta \in \{0, 1\}^q$ such that there exists a $z \in \mathbb{R}^m$ satisfying the constraints in (1), is useful to get the corresponding bound $V_{\text{best}} = \frac{1}{2}\left[\begin{smallmatrix} z \\ \delta \end{smallmatrix}\right]' Q \left[\begin{smallmatrix} z \\ \delta \end{smallmatrix}\right] + x'F' \left[\begin{smallmatrix} z \\ \delta \end{smallmatrix}\right]$.

III. Binary predictor via compact neural networks

An alternative to using online MIQP solvers is to resort to explicit MPC ideas [7], [8] to find an optimizer function $z^*(x)$, $\delta^*(x)$ offline. This approach is usually limited to problems with few binary variables and constraints.

Here we take a semi-explicit approach that consists of determining an explicit function $\delta : \mathbb{R}^n \to \{0, 1\}^q$ such that $\delta(x)$ is a feasible configuration, for each given $x \in \mathbb{R}^n$. Each component function δ_i can be seen as a binary classifier that we will determine using machine learning (ML) techniques.

Consider the MIQP problem (1) and, for each parameter vector $x \in \mathbb{R}^n$, let $\delta^*(x)$ be the binary components of a corresponding optimizer. From a functional approximation point of view, the prediction task can be formulated as the problem of finding an approximator function $f : \mathbb{R}^n \to \{0, 1\}^q$ such that $f(x)$ is "as similar as possible to $\delta^*(x)$", $\forall x$. As we want to warm start a B&B solver, we are not strictly aiming at predicting the closest solution (according to some binary distance) to the optimal one, but also at avoiding that a possible classification error leads to a guess $f(x)$ that is not a feasible configuration. Indeed, from a B&B perspective, a feasible configuration is more useful than an unfeasible one even if the latter is closer to the optimal one, as it may provide a good bound. This poses some limits on the choice of the employable machine learning techniques, as most of them are linked to a rigid form of the loss function used during the learning phase.

The recent contribution [25], discovered by the authors while writing this paper, has been taken a similar perspective for semi-explicit QP problems, by attempting at finding the optimal active set as an explicit function of x. The method we will present next is tailored instead to MIQP problems, by trying to guess the value of binary variables. Our approach can be immediately extended to predicting active sets in multiparametric QPs too. In this case, the function to learn is the optimal active set function $f : \mathbb{R}^n \to \{0, 1\}^p$, defined by

$$f_i(x) = \begin{cases} 0 & \text{if} \quad G_i z^*(x) < W_i + S_i x \\ 1 & \text{if} \quad G_i z^*(x) = W_i + S_i x \end{cases} \tag{2}$$

where $z^*(x)$ is a optimal solution of the QP problem (1) ($q = 0$). In [26] it is shown that z^* and f are, respectively, a piecewise affine (PWA) and a piecewise constant function of x.

Proposed learning architecture

For embedded applications, we need a learning architecture with a compact memory footprint, lightweight in terms of CPU usage, and possibly running also in constant time for throughput predictability. Among various options, as noted in [27], a good choice is using a very compact feedforward neural network (ANN) with a small number of layers composed of neurons featuring ReLU (Rectified Linear Units, [28]) activation functions

$$f_{\text{ReLU}}(x) = \max\{0, x\} \tag{3}$$

In fact, with the choice in (3) the memory footprint is extremely compact, due to recent advancements in terms of group sparsity regularization techniques [29], and the simple structure of the ReLU map (3) results in a very small computational burden on the CPU. Moreover, the number of floating point operations (flops) involved in a evaluating an

ANN with ReLU activation functions is independent of the number of samples used in the training phase, in contrast for example to K nearest neighbors (KNN) classifiers [30], and are constant, contrarily for example to decision trees. In addition, ReLU functions are piecewise linear functions, and therefore lead to ANNs that have a PWA shape. Finally, they can be trained with arbitrary smooth loss functions. To circumvent the fact that they are not differentiable in zero, it is common practice to assume $\frac{d\max\{0,x\}}{dx}\Big|_{x=0} = 0$.

The network architecture used in this work is rather classical. We employ a stack of densely interconnect ReLU layers followed by an output layer of q sigmoidal neurons, one for each binary variable we want to predict. Let us call $y : \mathbb{R}^n \to \mathbb{R}^q$ the output of the network. No bias is used in the hidden layers. As the sigmoid function can take any value in the interval $[0, 1]$, an extra step is necessary for quantizing the learned function $y(x)$ to a binary value.

Here for binary quantization we simply employ the following threshold function $q_t : \mathbb{R}^q \to \mathbb{R}^q$

$$q_t(y) = \frac{1}{2}(1 + \text{sign}(y - 0.5)) \tag{4}$$

The overall architecture is depicted in Figure 1. Note that function (4) is not used during the training phase.

A. Choice of loss function

As mentioned above, we would like to predict δ as a function of x by balancing feasibility and optimality. We therefore pose the training process as a multi-objective optimization problem, with one loss function for each one of the two objectives.

For the objective of recovering optimal binary combinations, we resort to the standard binary cross-entropy loss function $f_{\text{err}} : \mathbb{R}^{2q} \to \mathbb{R}$

$$f_{\text{err}}(y, \delta) = -\sum_{i=1}^{p}(1 - \delta_i)\log(1 - y_i) + \delta_i \log(y_i) \tag{5}$$

commonly used for training most ANN classifiers.

Regarding the second objectiv (feasibility of δ), we simply embed the inequalities appearing in (1) that only involve binary components in the loss functions, for example using barrier functions that penalize the violation of the corresponding inequality constraints. An example of such purely binary inequality constraints are those deriving from the translation of Boolean constraints into integer inequalities [31]. Let us denote by I, $I \subseteq \{1, \dots, p\}$, the (possibly empty) set of purely binary linear inequality constraints appearing in (1). Then, we choose the loss associated due to the feasibility objective as

$$f_{\text{feas}}(y) = \sum_{i \in I} \max\{0, G_i \begin{bmatrix} 0 \\ q_s(y) \end{bmatrix} - W_i - S_i x\}^2 \tag{6}$$

where $q_s : \mathbb{R}^q \to \mathbb{R}^q$ is a sigmoidal smooth approximation of (4),

$$[q_s(y)]_i = \frac{1}{1 + e^{-\kappa\left(y_i - \frac{1}{2}\right)}} \tag{7}$$

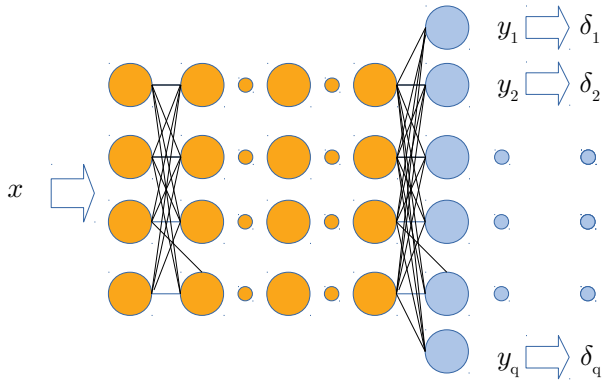

Fig. 1. ANN architecture: ReLU neurons (orange), sigmoidal neurons (light blue).

and the subscript i denotes the ith row (component) of a matrix (vector). In the following we choose $\kappa = 100$ in (7). Given N training samples $x^1, \dots, x^N$ and the corresponding optimal binary solutions $\delta^1, \dots, \delta^N$, $\delta^i = \delta^*(x^i)$, and by letting $w \in \mathbb{R}^{n_w}$ the vector of parameters defining the ANN $y_w : \mathbb{R}^n \to \{0, 1\}^q$ we want to train, we define the *training loss function*

$$\ell(w) = \frac{1}{N}\sum_{i=1}^{N} f_{\text{err}}(y_w(x_i), \delta^i) + \gamma f_{\text{feas}}(y_w(x^i)) \tag{8}$$

where γ is a tuning weight used for balancing the two components of the training loss. The predictor $\delta : \mathbb{R}^n \to \{0, 1\}^q$, resulting from minimizing (8) with respect to w given the N training samples, is $\delta(x) = q_t(y_w(x))$.

IV. ANN-B&B ALGORITHM

We now focus on how to exploit the predicted $\delta(x(t))$ on line to solve the MIQP (1) for each given $x(t)$.

A possibility is to use a standard depth-first B&B algorithm and, independently of the heuristic used to decide the variable δ_i to branch on during the search, explore first the sub-problem in which the value taken by δ_i is set to the predicted value $\delta_i(x)$. A possible drawback of this method is that, due to feasibility of $\delta(x)$, most likely all QP relaxations will be solved initially down to the leaf node in which all binary variables are fixed at $\delta(x)$, unless infeasibility is detected earlier.

We therefore take another approach of reversely traversing the solution tree. We start by solving a QP problem in which all the binary variables are set to their predicted value $\delta(x)$. If the QP is feasible we have obtained a valid upper bound V_0 on the optimal cost. Otherwise, we start solving a sequence of MIQPs, each one obtained by "unlocking" one more binary variable from the predicted value $\delta_i(x)$ to $\{0, 1\}$, as summarized in Algorithm 1.

A. A weighted unlocking strategy

The strategy for choosing the next variable to unlock is crucial in Algorithm 1. Here we propose two criteria to select the order in which we unlock binary variables:

Algorithm 1 Successive unlocking from binary warm start

1. Solve a QP problem as in (1) with δ locked at $\delta(x)$;
2. If the QP is feasible go to Step 5;
3. If no binary variable is locked go to Step 6;
4. Unlock a locked binary variable and solve a reduced MIQP using a standard B&B algorithm;
   4.1. If the reduced MIQP is feasible go to Step 5;
   4.2. Go to Step 4;
5. A feasible solution has been found; Go to Step 7
6. MIQP problem is infeasible;
7. End.

1) Prediction quality: Based on results on a new validation dataset, we sort the binary variables by their average prediction error. Then, in Algorithm 1 the variables with larger errors are unlocked first.

2) Prediction sensitivity: A prediction that is highly sensitive to variations of the input x is more prone to be infeasible. For all locked variables we can therefore compute the sensitivity

$$\left\| \frac{\partial y(x)}{\partial x} \right\|_\infty$$

The two criteria are combined as follows. Let $r_{\rm fit} : \{1,\dots,q\} \to \{1,\dots,q\}$ be the ranking associated with prediction quality (the smaller $r_{\rm fit}(i)$ the better the quality of the predictor $\delta_i(\cdot)$), and similarly $r_{\rm sens} : \{1,\dots,q\} \to \{1,\dots,q\}$ the one associated with prediction sensitivity. Then the variable i with the lowest score

$$f_{\rm score}(i) = \quad r_{\rm fit}(i) r_{\rm sens}(i)^{\hat\gamma} \tag{9}$$

is chosen, where $\hat\gamma \geq 0$ is a tuning parameter.

B. B&B algorithm

Once an integer feasible solution has been found, a standard B&B algorithm is solved from the root note to determine the global optimum of problem (1). The B&B phase takes advantage of (i) the upper-bound $V_{\rm best}$ given by the integer feasible solution found by Algorithm 1, (ii) QP relaxations that have been already tested by Algorithm 1 are not solved, (iii) the subtree originating from the integer feasible solution determined by Algorithm 1 is not explored. In order to exploit (ii) and (iii) we keep a list of explored binary combinations, employing standard hash-maps (a Bloom filter [32] could be used in alternative).

V. Numerical experiments

We test the proposed semi-explicit MIQP solution strategy on simple hybrid MPC problems, with a comparison with a standard B&B approach. We use the QP solver of Mosek [33] for solving QP relaxations and report results in terms of the total number of iterations counted when solving the entire MIQP, or of the best (possibly suboptimal) cost found when we limit the number of iterations to a fixed amount N.

A. Benchmark problems

We consider two MPC problems that arise from corresponding hybrid single-input single-output systems. The first hybrid prediction model is described by

$$\begin{cases} x_{k+1} = & \delta_k(A_1 x_k + B_1 x_k) + \\ & (1-\delta_k)(A_2 x_k + B_2 x_k) \\ \delta_k = & \begin{cases} 1 & \text{if} \quad x_{1k} \geq 0 \\ 0 & \text{otherwise} \end{cases} \\ y_k = & Cx_k \end{cases} \tag{10}$$

where x_{1k} denotes the first component of the state x_k, $x_k \in \mathbb{R}^2$, $x_0 = x(t)$ is the current state at time t, and A_1, A_2, B_1, B_2, C are defined as

$$A_1 = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \qquad A_2 = \begin{bmatrix} 2 & -0.9 \\ 1.3 & 0 \end{bmatrix}$$
$$B_1 = \begin{bmatrix} -0.2 \\ 0 \end{bmatrix} \qquad B_2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$
$$C = \begin{bmatrix} 0 & 1 \end{bmatrix}$$

Similarly, for the second problem we consider the prediction model

$$\begin{cases} x_{k+1} = & Ax_k + \delta_k B_1 x_k + \\ & + (1-\delta_k) B_2 x_k \\ y_k = & \delta_k C_1 x_k + (1-\delta_k) C_2 x_k \\ \delta_k = & \begin{cases} 1 & \text{if} \quad x_{1k} + x_{2k} \geq 0 \\ 0 & \text{otherwise} \end{cases} \end{cases} \tag{11}$$

where

$$A = \begin{bmatrix} 2 & -0.9 \\ 1.3 & 0 \end{bmatrix}$$
$$B_1 = \begin{bmatrix} -2 \\ 0 \end{bmatrix} \qquad B_2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$
$$C_1 = \begin{bmatrix} 0 & 0.3 \end{bmatrix} \qquad C_2 = \begin{bmatrix} -0.3 & -0.2 \end{bmatrix}$$

For system (10) we consider the following MPC problem

$$\begin{array}{ll} \min_{u_0,\dots,u_6} & \sum_{k=0}^{6} \left[(y_{k+1} - r_{k+1})^T Q (y_{k+1} - r_{k+1}) \right. \\ & \left. + (u_k - u_{k-1})^T R (u_k - u_{k-1}) \right. \\ \text{s.t.} & -1 \leq u_k \leq 1, \ k = 0,\dots,6 \\ & -3 \leq x_k \leq 3, \ k = 1,\dots,7 \\ & \text{dynamics (10)} \\ & x_0 = x(t), u_{-1} = u(t-1) \end{array} \tag{12}$$

while for system (11) we consider

$$\begin{array}{ll} \min_{u} & \sum_{k=0}^{6} \left[(y_{k+1} - r_{k+1})^T Q (y_{k+1} - r_{k+1}) \right. \\ & \left. + (u_k - u_{k-1})^T R (u_k - u_{k-1}) \right] \\ \text{s.t.} & -1 \leq u_k \leq 1, \ k = 0,\dots,6 \\ & -3 \leq x_{1k} \leq 3, \ k = 1,\dots,7 \\ & \text{dynamics (11)} \\ & x_0 = x(t), u_{-1} = u(t-1) \end{array} \tag{13}$$

In both cases the parameter vector x in (1) has 10 components, due to 2 states (x_0), one previous input (u_{-1}) and 7 reference signals ($r_1,\dots,r_7$).

The corresponding multiparametric MIQPs (1) are generated using YALMIP [34] and consist of, respectively, $q = 14$ and $q = 16$ binary optimization variables, out of a total of $\approx$50 variables.

B. Training set and offline learning

In both problems we draw parameters x from a zero mean unit variance Gaussian distribution and only retain a dataset of 11000 samples for which the corresponding MIQP is feasible. We use 9000 of such samples for training, 1000 for validation/tuning of the proposed B&B strategy, and 1000 for testing.

The ANN is implemented in Keras [35] using Tensorflow [36]. The network consists of 4 layers of 25 ReLU followed by an output layer of q sigmoidal neurons, one for each variable δ_i to predict.

The network has been trained using the ADAM algorithm [37], [38] using at most 100 epochs and using an early stopping strategy, with $\gamma = 0.01$ in (8) for both Problem 1 and Problem 2. The training phase takes less than a minute on a laptop equipped with an Intel Core i5 6200u (2.3GHz) processor, with a negligible quantity of RAM required.

The MIQP solver warm-started by the predictor δ (ANN-B&B) and a pure B&B solver were implemented in MATLAB R2018b and based on a classical depth-first/max-fractional part strategy, which typically performs well on the class of MIQP problems considered here.

C. Computational cost of the predictor

The proposed ANN employs approximately 2500 weights, represented in single precision. The corresponding memory footprint is comparable with the one required to store the MIQP matrices and could be further reduced by using either group sparsity regularization or low precision encoding techniques [39].

Evaluating the prediction $y(x)$ and its Jacobian (via back-propagation) for Problem 1 requires ≈90000 multiplications, ≈86000 sums, and ≈40 divisions, corresponding to executing a matrix multiplication and evaluating the activation functions for each layer of the ANN. Although employing the predictor involves some computation effort, this is less than 1/100 effort than solving the MIQP problem.

D. ANN-B&B results: reaching the global optimum

In this section we analyze the influence of the warm-start $\delta(x)$ on the iterations needed by the proposed ANN-B&B method to find the global optimum with respect to a standard depth-first B&B algorithm with branching selection done according to the binary variable with maximum fractional part. Numerical results are reported in Table I.

	Problem 1	Problem 2
average QP iterations (ANN-B&B)	1072	2936
average QP iterations (standard B&B)	1675	3867
average decrease of QP iterations	41.19%	34.21%
median decrease of QP iterations	43.62%	30.55%
cases of worse behavior	10/1000	17/1000

TABLE I
DECREASE OF QP ITERATIONS DURING B&B

It is clear that employing the warm start $\delta(x)$ provided by the predictor indeed reduces the total number of QP iterations required to solve the MIQP problem to optimality. Only in a very small percentage of cases (10 out of 1000 and 17 out of 1000 for Problem 1 and 2, respectively) warm-starting with $\delta(x)$ increases the number of QP iterations. In such cases, the predictor is wrong on multiple binary values and, at the same time, the standard B&B algorithm is able to quickly find the solution in a small number of iterations.

E. ANN-B&B results: finding an integer-feasible solution

We now investigate the number of unlockings required to find an integer-feasible solution. This is particularly relevant in resource-constrained control applications, where the solver may be stopped prematurely because of task preemption. Results are reported in Table II.

	Problem 1	Problem 2
first guess	942	943
1 unlocking	11	0
2 or more unlockings	47	57

TABLE II
NUMBER OF VARIABLE UNLOCKINGS REQUIRED BY ALGORITHM 1 TO FIND A FEASIBLE SOLUTION.

One can expect improvements of the results if more flexible functions are employed to determine the δ approximator.

F. ANN-B&B results: quality of initial guess

We analyze the quality of the first integer-feasible solution found by executing the ANN-B&B algorithm starting from the initial guess provided by $\delta(x)$.

	Problem 1	Problem 2
global optimum	725	310
suboptimal within 1% from optimum	79	130
between 1% and 5% from optimum	113	305
over 5% from optimum	83	255

TABLE III
SUBOPTIMALITY OF THE SOLUTION PROVIDED BY ALGORITHM 1.

The results reported in Table III are consistent with those reported in Table II.

G. ANN-B&B results: solution quality after limited iterations

We analyze the quality of the solution obtained when the number of QP iterations used during B&B is bounded by a fixed number $N = 600$, that is the smallest value for which both ANN-B&B and pure B&B always manage to find a feasible solution. We report the results for Problem 2 in Table IV

VI. CONCLUSIONS

In this paper we have proposed a neural network architecture aimed at predicting the value of the optimal binary solution of a multiparametric MIQP problem. The approach has been coupled with a B&B scheme specifically tailored to

average cost decrease	11.93%
median cost decrease	6.44%

TABLE IV

PROBLEM 2: PERCENTAGE DECREASE OF THE VALUE OF THE BEST SOLUTION FOUND BY ANN-B&B WITH RESPECT TO PURE B&B, WHEN BOTH ARE LIMITED TO $N = 600$ QP ITERATIONS.

exploit the binary warm start provided by the predictor. As a result, we can reduce the computational power required to find the global optimum, as well as improve the quality of the solution obtained within a limited number of computations. TRhe approach presented in this paper can be immediately extended to learn optimal active sets in multiparametric QP problems and to other type of MIP problems, such as mixed-integer linear programs.

Future work will focus on coupling the proposed technique with more advanced warm-starting strategies for MIQP, like the one reported in [20], and on predicting also real variables.

REFERENCES

[1] T. Samad, "A survey on industry impact and challenges thereof," *IEEE Control Systems Magazine*, vol. 37, no. 1, pp. 17–18, 2017.

[2] D. Mayne, "Model predictive control: Recent developments and future promise," *Automatica*, vol. 50, no. 12, pp. 2967–2986, 2014.

[3] A. Bemporad, "Model-based predictive control design: New trends and tools," in *45th IEEE Conf. on Decision and Control*, (San Diego, CA), pp. 6678–6683, 2006.

[4] D. Kouzoupis, G. Frison, A. Zanelli, and M. Diehl, "Recent advances in quadratic programming algorithms for nonlinear model predictive control," *Vietnam Journal of Mathematics*, 2018.

[5] A. Bemporad and M. Morari, "Control of systems integrating logic, dynamics, and constraints," *Automatica*, vol. 35, no. 3, pp. 407–427, 1999.

[6] G. Nemhauser and L. Wolsey, *Integer and combinatorial optimization*. Wiley-Interscience series in discrete mathematics and optimization, Wiley, 1988.

[7] F. Borrelli, M. Baotić, A. Bemporad, and M. Morari, "Dynamic programming for constrained optimal control of discrete-time linear hybrid systems," *Automatica*, vol. 41, pp. 1709–1721, Oct. 2005.

[8] A. Alessio and A. Bemporad, "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*, pp. 302–308, 2006.

[9] G. Cimini and A. Bemporad, "Exact complexity certification of active-set methods for quadratic programming," *IEEE Trans. Automatic Control*, vol. 62, no. 12, pp. 6094–6109, 2017.

[10] T. Marcucci, R. Deits, M. Gabiccini, A. Bicchi, and R. Tedrake, "Approximate hybrid model predictive control for multi-contact push recovery in complex environments," in *IEEE-RAS Int. Conf. on Humanoid Robotics (Humanoids)*, pp. 31–38, 2017.

[11] B. Karg and S. Lucia, "Deep learning-based embedded mixedinteger model predictive control," in *European Control Conf.*, pp. 2075–2080, 2018.

[12] D. Axehill, T. Besselmann, D. M. Raimondo, and M. Morari, "A parametric branch and bound approach to suboptimal explicit hybrid MPC," *Automatica*, vol. 50, no. 1, pp. 240–246, 2014.

[13] A. Bemporad, D. Mignone, and M. Morari, "An efficient branch and bound algorithm for state estimation and control of hybrid systems," in *European Control Conf.*, (Karlsruhe, Germany), Aug. 1999.

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

[15] A. Bemporad and V. V. Naik, "A numerically robust mixed-integer quadratic programming solver for embedded hybrid model predictive control," in *6th IFAC Conf. on Nonlinear Model Predictive Control*, 2018.

[16] C. A. Floudas, *Nonlinear and Mixed-Integer Optimization*. Oxford University Press, 1995.

[17] J. Nocedal and S. Wright, *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering, Springer New York, 2006.

[18] H. Ferreau, H. Bock, and M. Diehl, "An online active set strategy to overcome the limitations of explicit MPC," *Int. Journal of Robust and Nonlinear Control*, vol. 18, no. 8, pp. 816–830, 2008.

[19] A. Bemporad, "Solving mixed-integer quadratic programs via nonnegative least squares," in *5th IFAC Conf. on Nonlinear Model Predictive Control*, (Sevilla, Spain), pp. 73–79, 2015.

[20] A. Bemporad and V. Naik, "A numerically robust mixed-integer quadratic programming solver for embedded hybrid model predictive control," in *6th IFAC Conf. on Nonlinear Model Predictive Control*, (Madison, WI), pp. 502–507, 2018.

[21] D. Axehill and A. Hansson, "A mixed integer dual quadratic programming algorithm tailored for MPC," in *Proc. 45th IEEE Conf. on Decision and Control*, (San Diego, CA, USA), pp. 5693–5698, 2006.

[22] D. Frick, A. Domahidi, and M. Morari, "Embedded optimization for mixed logical dynamical systems," *Computers & Chemical Engineering*, vol. 72, pp. 21–33, 2015.

[23] V. Naik and A. Bemporad, "Embedded mixed-integer quadratic optimization using accelerated dual gradient projection," in *Proc. 20th IFAC World Congress*, (Toulouse, France), pp. 10723–10728, 2017.

[24] B. Stellato, V. Naik, A. Bemporad, P. Goulart, and S. Boyd, "Embedded mixed-integer quadratic optimization using the OSQP solver," in *European Control Conf.*, 2018.

[25] M. Klaučo, M. Kalúz, and M. Kvasnica, "Machine learning-based warm starting of active set methods in embedded model predictive control," *Engineering Applications of Artificial Intelligence*, vol. 77, pp. 1 – 8, 2019.

[26] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, no. 1, pp. 3 – 20, 2002.

[27] B. Karg and S. Lucia, "Efficient representation and approximation of model predictive control laws via deep learning," *arXiv preprint arXiv:1806.10644*, 2018.

[28] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Int. Conf. on Machine Learning (ICML)*, pp. 807–814, Omnipress, 2010.

[29] S. Scardapane, D. Comminiello, A. Hussain, and A. Uncini, "Group sparse regularization for deep neural networks," *Neurocomputing*, vol. 241, pp. 81 – 89, 2017.

[30] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. Springer Series in Statistics, New York, NY, USA: Springer New York Inc., 2001.

[31] F. Torrisi and A. Bemporad, "HYSDEL — A tool for generating computational hybrid models," *IEEE Trans. Contr. Systems Technology*, vol. 12, pp. 235–249, Mar. 2004.

[32] F. Putze, P. Sanders, and J. Singler, "Cache-, hash-, and space-efficient Bloom filters," *J. Exp. Algorithmics*, vol. 14, pp. 4:4.4–4:4.18, 2010.

[33] MOSEK ApS, *The MOSEK optimization toolbox for MATLAB manual. Version 8.1.*, 2017.

[34] J. Löfberg, "YALMIP: A toolbox for modeling and optimization in MATLAB," in *In CACSD Conf.*, (Taipei, Taiwan), 2004.

[35] F. Chollet *et al.*, "Keras." `https://keras.io`, 2015.

[36] M. Abadi, A. Agarwal, and P. B. et al, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[37] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[38] S. K. Sashank J. Reddi, Satyen Kale, "On the convergence of Adam and beyond," *Int. Conf. on Learning Representations*, 2018.

[39] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.