Paper ID: ACC00-IEEE1027

Multi-objective Prioritisation and Reconfiguration for the Control of Constrained Hybrid Systems

E.C. Kerrigan*, A. Bemporad†, D. Mignone†, M. Morari† and J.M. Maciejowski*

*Control Group, Department of Engineering
University of Cambridge, Trumpington Street
Cambridge CB2 1PZ, United Kingdom
http://www-control.eng.cam.ac.uk
eck21,jmm@eng.cam.ac.uk

†Institut für Automatik
ETH - Swiss Federal Institute of Technology
ETHZ - ETL, CH 8092 Zürich, Switzerland
http://www.aut.ee.ethz.ch
bemporad,mignone,morari@aut.ee.ethz.ch

Abstract

In many applications, the control objectives and constraints can be assigned a hierarchy of levels of priority. Often a disturbance or a fault occurs, resulting in some constraints or objectives being violated. Inadequate handling of this situation might result in component or even system-wide failures. This paper presents several methods for handling a large class of multi-objective formulations and prioritisations for model predictive control of hybrid systems, using the new mixed logic dynamical (MLD) framework. A new method, which does not require logic variables for prioritising soft constraints, is also presented.

Keywords: hierarchical control, feasibility, mixed logic dynamical (MLD) systems, predictive control, fault handling

1 Introduction

In many applications, the control objectives and/or constraints can be described according to a hierarchy and assigned different levels of priority. Sometimes a disturbance or a fault occurs, resulting in some constraints being violated. Inadequate handling of this situation might result in component or even system-wide failure. A systematic method for describing control objectives and for handling the constraint violations is a crucial part in the design of a complete control system.

Model predictive control (MPC) is arguably the most popular advanced control technique in the process industry, due to the intuitive control problem formulation and ease with which safety constraints, performance constraints and economic objectives can be incorporated in the controller design and synthesis. Some commercial MPC products already incorporate various degrees of multi-objective formulation and constraint handling [1]. Multiple objectives are usually implemented by solving a sequence of optimisation problems; the optimal steady-state input and output targets are calculated, followed by the dynamic optimisation. Constraint handling in the event of infeasibility is achieved using soft constraints or by dropping low priority constraints until the problem becomes feasible. The problem with the soft constraint approach is that constraint prioritisation is not guaranteed. The dropping of low priority constraints can be far from optimal, since the constraint violations aren't minimised.

In practical applications one can distinguish between *hard constraints* such as physical or safety limitations, which cannot be relaxed, and *soft constraints* such as performance constraints on the plant states, which can be relaxed if necessary. Often in practice plants are driven towards their set-point by defining zones or funnels within which the controlled variables should lie [1]. These zones and funnels can be specified using hard or soft constraints. Hard constraints can give rise to infeasible problems, while well-designed soft constrained problems can be made to be feasible for all time.

One way of introducing soft constraints is via the use of exact penalty functions [2]. In [3] a soft constraint method is presented which minimises the duration and then the size of the violation. However, priorities are not taken into account.

In [4] an algorithm is presented which minimises the prioritised constraint violations via the solution of a sequence of QPs or LPs. It is shown in [5] how the same sequence of LPs can be reduced to solving a single LP, thereby reducing the computational load. These methods typically solve for the so-called *lexicographic minimum* of the violated constraints; constraint violations at lower priority levels cannot be decreased without increasing higher prioritised constraint violations. However, this method does not necessarily result in the maximum number of satisfied lower-priority constraints.

A method for prioritising soft constraints using propositional logic was first described in [6]. These ideas were extended in [7] using the new *mixed logic dynamical* (MLD) framework. This method is not only applicable to LTI systems, but also to hybrid systems that can be cast into MLD form.

Section 2 of this paper starts by introducing MLD systems. Section 3 extends the constraint prioritisation method pre-

sented in [7, Sect. 5.1] to handle a larger class of multi-objective formulations and prioritisations. A method, which does not require logic variables for prioritising soft constraints, is also presented. The main contribution of this paper is a general framework for multi-objective and reconfigurable control which is presented in Section 4. A brief, illustrative example is given in Section 5. Finally, some conclusions are drawn in Section 6.

2 Mixed Logic Dynamical Systems

A system is said to be *hybrid* if it consists of *continuous* components interacting with *logical/discrete* components. A very large class of practical applications fall into this class, which is why a lot of research effort has recently been dedicated to obtaining methods for the modelling and control of hybrid systems.

The MLD modelling framework, introduced in [7], allows one to represent systems which can be described by interdependent physical laws, logical rules and operating constraints. It allows a large class of systems to be described such as constrained linear systems, finite state machines, systems with discrete states and/or inputs and nonlinear systems which can be approximated by piecewise affine functions. For further details on propositional logic and MLD systems, the reader is referred to [7].

The general MLD form is given by:

$$x(k+1) = Ax(k) + B_1u(k) + B_2\delta(k) + B_3z(k) \tag{1a}$$
$$y(k) = Cx(k) + D_1u(k) + D_2\delta(k) + D_3z(k) \tag{1b}$$
$$E_2\delta(k) + E_3z(k) \leq E_1u(k) + E_4x(k) + E_5 \tag{1c}$$

where $x \in \mathbb{R}^{n_c} \times \{0,1\}^{n_l}$ are the continuous and binary states, $u \in \mathbb{R}^{m_c} \times \{0,1\}^{m_l}$ are the inputs, $y \in \mathbb{R}^{p_c} \times \{0,1\}^{p_l}$ the outputs, $\delta \in \{0,1\}^{r_l}$ and $z \in \mathbb{R}^{r_c}$ represent binary and continuous auxiliary variables. The latter are introduced when propositional logic statements are transformed into linear inequalities. All the constraints on the state, input, δ and z are contained in (1c). The description in (1) only appears to be linear; the variables δ are constrained to be binary. MLD systems can be controlled by formulating an MPC problem and solving it using a mixed-integer quadratic program (MIQP), for which efficient solvers have recently become available [8].

3 Soft Constraint Prioritisation

This section presents a method which can be applied to MLD (and hence also LTI) systems to describe the prioritisation of soft constraints in MPC. The resulting problem can then be solved using a single MIQP.

3.1 Initial Formulation

Constrained MPC usually results in the formulation of a quadratic programming problem. Consider the following optimisation problem:

$$\min_{\theta\in\Theta} \theta' H\theta + f'\theta \tag{2}$$

where the set $\Theta = \{\theta : A\theta \leq b\}$ is a polytope (i.e. a bounded polyhedron); H, f, A, b and θ are defined by the MPC problem. One way of softening the constraints is by modifying (2) and solving the following MIQP:

$$\min_{\theta,\varepsilon,\delta} \theta' H\theta + f'\theta + \varepsilon' S\varepsilon + \rho M_p'\delta \tag{3a}$$

subject to

$$A\theta \leq b + C\varepsilon \tag{3b}$$
$$0 \leq \varepsilon \leq M_\varepsilon \tag{3c}$$

where $\varepsilon \in \mathbb{R}^s$ is a vector of slack variables, representing the constraint violations; C is a matrix of appropriate dimension whose rows are 0 or e_k' according to whether the k'th constraint is hard or soft; $S \succ 0$ is a weighting matrix which decides the trade-off between cost and constraint violation; the upper bounds on the slack variables M_ε are necessary in order that the scalar ρ, which is defined as

$$\rho > \max_{\theta,\varepsilon,\delta} \theta' H\theta + f'\theta + \varepsilon' S\varepsilon \tag{4}$$

subject to (3b) and (3c), be finite. The vector $\delta \in \{0,1\}^r$ is defined by introducing logic variables $\delta_i, i = 1 \ldots r$, and associating them with constraint violations at r levels of priority using the propositional logic statements:

$$\begin{aligned} [\delta_1 = 0] &\rightarrow [\varepsilon_{1_1} + \ldots + \varepsilon_{1_{c_1}} = 0] \\ [\delta_2 = 0] &\rightarrow [\varepsilon_{2_1} + \ldots + \varepsilon_{2_{c_2}} = 0] \\ \vdots &\quad\quad \vdots \\ [\delta_r = 0] &\rightarrow [\varepsilon_{r_1} + \ldots + \varepsilon_{r_{c_r}} = 0] \end{aligned} \tag{5}$$

where ε_{i_j} represents the constraint violation of the j'th constraint on the i'th priority level. If $[\delta_i = 0]$ is true, then the i'th set of constraints will be satisfied. These logic statements are converted to linear inequalities [7] and added to the constraints of the MIQP[1].

The key to this approach is that the vector M_p is chosen such that the highest priority is assigned to δ_1 and the lowest to δ_r. A good choice is

$$M_p \triangleq \begin{bmatrix} 2^{r-1} & 2^{r-2} & \ldots & 2^0 \end{bmatrix}' . \tag{6}$$

[1] Note that (5) requires fewer auxiliary variables and inequalities than the definition given in [7, Sect. 5.1], since the equalities on ε_{i_j} in the latter are also captured by the equalities in (5). Furthermore, the above statements with $\rightarrow$ require fewer inequalities than those with $\leftrightarrow$.

This choice of M_p ensures that the satisfaction of higher prioritised constraints always results in a greater reduction in cost than the satisfaction of any combination of lower prioritised constraints, e.g. if it is possible to choose $\delta = [0\ 1\ 0]'$ in the solution of the MIQP, this will result in a lower optimal cost than choosing $\delta = [1\ 0\ 0]'$.

Since ρ forms an upper bound on the performance and violation cost, it can be seen that the objective function in (3) can always be decreased by increasing the number of satisfied priority levels. The performance and violation cost is minimised after $M_p'\delta$ has been minimised.

3.2 An Extension to the Basic Formulation

If a constraint on a given priority level cannot be satisfied, (5) and (6) will not guarantee the satisfaction of all the other constraints on the same priority level. In order to ensure that some of the constraints on the violated priority levels be satisfied, if possible, the term $s_1'\varepsilon$ with $s_1 > 0$ large enough needs to be added to (3a). This results in additionally penalising the weighted l_1-norm of the violations [2, 3]. It is not yet clear how to design the weights in s_1 in the mixed-objective MIQP to guarantee the satisfaction of the maximum number of constraints on a given violated priority level.

A systematic way of guaranteeing the satisfaction of the maximum number of constraints is discussed below by generalising the definition of δ and M_p. Firstly, associate each of the c_i slack variables ε_{i_j} on priority level i with a logic variable

$$[\delta_{i_j} = 0] \rightarrow [\varepsilon_{i_j} = 0], \quad i = 1 \dots r, j = 1 \dots c_i \,. \tag{7}$$

The vector δ is then defined as

$$\delta \triangleq [\delta_{1_1} \ \dots \ \delta_{i_{(c_i-1)}} \ \delta_{i_{c_i}} \ \delta_{(i+1)_1} \ \dots \ \delta_{r_{c_r}}]' \,. \tag{8}$$

The propositional logic statements (7) can now be implemented by converting them to linear inequalities. This is achieved by replacing (3c) with

$$0 \leq \varepsilon \leq \mathrm{diag}(M_\varepsilon)\delta \,. \tag{9}$$

Secondly, modify the components of M_p such that the same integer weight m_i is assigned to each of the c_i constraints on priority level i:

$$M_p \triangleq [m_1\mathbf{1}_{c_1}' \ \dots \ m_i\mathbf{1}_{c_i}' \ \dots \ m_r\mathbf{1}_{c_r}']' \tag{10}$$

where $\mathbf{1}_{c_i}$ is the unit vector of size c_i and $m_r = 1$. In order to ensure that the satisfaction of higher prioritised constraints always results in a larger reduction of the cost than the satisfaction of any combination of lower prioritised constraints, the weights for a priority level need to be larger than the sum of all the elements of M_p associated with the lower priorities. At the same time one would like to keep the weights as small as possible to avoid running into numerical problems. An m_i which satisfies this is given by

$$m_i = 1 + \sum_{j=i+1}^{r} c_j m_j \,. \tag{11}$$

For example, if there are $s = 4$ constraints with the second and third on the same priority level, i.e. $r = 3$, $c_1 = 1$, $c_2 = 2$ and $c_3 = 1$, then $M_p = [6\ 2\ 2\ 1]'$ would be suitable. If there are the same number of priority levels as soft constraints, e.g. $r = s = 4$, then $M_p = [8\ 4\ 2\ 1]'$, which is the same as in (6).

Note that soft constraints and constraint priority can be included in the MLD structure (1) by incorporating the slack vector ε in the $z(k)$-vector and the logic variables δ in the $\delta(k)$-vector. It should also be clear that any logic variables in the MLD system can be included in δ and M_p and hence prioritised, e.g. the open/closed status of a valve.

3.3 Getting Closer to Pareto-optimality

The methods used in [6, Sect. 3.2] and [7, Sect. 5.1] do not guarantee the satisfaction of the maximum number of constraints and therefore cannot simultaneously be applied to both time and output prioritisation; the formulation results in the relaxation of all constraints on the same and lower priority level once a certain priority level cannot be satisfied. This is not desirable and a sequence of MIQPs needs to be solved in order to satisfy the lower priority constraints and incorporate both time and output prioritisation. The method presented in Sections 3.1 and 3.2 provides a solution to this problem by maximising the number of satisfied constraints subject to the given prioritisation with a *single* MIQP.

Often one desires a solution which is *output-prioritised minimum time-optimal*, i.e. the duration of violation for the higher-prioritised outputs are minimised before proceeding to minimise the duration of violation of the lower-prioritised outputs. To minimise the *duration* of constraint violations, one would like constraint violations to occur only in the first part of the prediction horizon if it is not possible to satisfy the constraints over the whole horizon. This translates into future constraints having a higher priority than earlier ones. The ideas in [6, 7] can be used to implement this for each output. The *time* priorities for output i are defined by the following $P-1$ propositional logic statements:

$$[\delta_i(k) = 0] \rightarrow [\delta_i(k+1) = 0], \quad k = 1 \dots P-1 \tag{12}$$

where P is the length of the prediction horizon used in MPC. These logic statements can be translated into the linear inequalities

$$\delta_i(k+1) - \delta_i(k) \leq 0, \quad k = 1 \dots P-1 \,. \tag{13}$$

Each logic variable is also associated with a constraint as in Section 3.2. The *output* priorities are defined by assigning the same weight m_i to each of the P associated logic variables

on output i, i.e. $c_i = P, \forall i$. In solving the MIQP, the duration of violations κ_i for output i is minimised; constraints from time $k = \kappa_i + 1$ to $k = P$ are satisfied and constraints from $k = 1$ up to time $k = \kappa_i$ are relaxed. Because of the interaction between the choice of M_p and (12), the same MIQP also achieves this for the other outputs, subject to the given output prioritisation, as desired. Finally, the constraint violations and performance cost are also minimised.

By simply maximising the number of satisfied constraints the solution might not be *Pareto-optimal* in terms of duration *and* size of the violations. The choice of S determines how close the solution is to being Pareto-optimal [3]. Possible solutions are to assign arbitrarily large values to S, perhaps reflecting the priorities or solving $\min_{\theta,\varepsilon,\delta}(\varepsilon' S\varepsilon + \rho M_p'\delta)$ whenever (2) is infeasible. The latter (output-prioritised minimum time) approach is Pareto-optimal if $\varepsilon' S\varepsilon$ is used as the cost function for the size of violations[2]. Various combinations of S, s_1 and sequences of optimisation problems can be chosen to satisfy different Pareto-optimal definitions.

Solving for the lexicographic minimum [4, 5] does not result in an output-prioritised minimum time-optimal solution or the satisfaction of the maximum number of constraints. A suggestion for determining the lexicographic minimum of ε, while guaranteeing an output-prioritised minimum time-optimal solution, is to first solve $\min_{\theta,\varepsilon,\delta} M_p'\delta$ to determine which constraints cannot be satisfied, fixing those constraints which can be satisfied and then solving for the lexicographic minimum of ε.

3.4 Prioritisation without Requiring Logic Variables

Sometimes it is not required that low-priority constraint violations be minimised in an optimal fashion. A new, alternative method which does not require logic variables to prioritise soft constraints is to solve

$$\min_{\theta,\varepsilon,\eta} \theta' H\theta + f'\theta + s_1'\varepsilon + \varepsilon' S_1\varepsilon + s_2'\eta + \eta' S_2\eta \tag{14a}$$

subject to

$$A\theta \leq b + C\varepsilon \tag{14b}$$

$$\eta_{i+1} \geq \gamma_i \eta_i \tag{14c}$$

$$\eta \geq \varepsilon \geq 0 \tag{14d}$$

where $S_1 \succ 0$, $S_2 \succ 0$ and each element of s_1 and s_2 is positive; $\gamma_i > 0$ is a sufficiently small scalar. The slack vector η defines the priorities; constraints are relaxed in a prioritised fashion with constraint i taking a higher priority than constraint $i+1$. The actual constraint violations are represented by ε.

It is not yet clear in this scheme how large to choose the various slack vector penalty weights in order to achieve Pareto-optimality or to guarantee the satisfaction of the maximum number of constraints. This method also suffers from the relaxation of all lower-prioritised constraints once a certain constraint cannot be met. Additional slack vectors similar to η could be added to the soft constraints to define additional priorities. By introducing integer variables and an appropriate M_p, further control over the satisfaction of constraints can be achieved.

[2] This method is also numerically more robust since S and $\rho > \max_{\theta,\varepsilon,\delta} \varepsilon' S\varepsilon$ need not be as large.

4 A Framework for Multi-objective and Reconfigurable Control

Often it is desired that a controller bring the system into a desired operating region before moving the input and output variables to their optimal set-points. This kind of objective can be described by logic statements. Better performance might be achieved if these *state-dependent objectives* can be integrated into the controller; in [6, Sect. 3.1] it was shown how the inclusion of logic in the controller synthesis can improve the performance of a semi-batch reactor. Another situation which might occur is that the objectives and/or priorities might change when a fault occurs. Usually these events are handled by an external supervisor only when the fault occurs. Future knowledge of the effect of disturbances could help prevent failures. A reconfigurable, fault-tolerant controller can increase plant availability and minimise the economic cost of a failure. This section describes a general framework whereby logic- and state-dependent objectives can be combined with the prioritisation methods presented in Section 3 to design a reconfigurable controller.

Assume that an observer and fault detection scheme for MLD systems such as in [9] is implemented together with an MPC controller. The inputs to this combined system are the current and future set-point $r(t)$, current and past plant inputs $u(t)$ and outputs $y(t)$ and a signal from the operator $\psi(t) \in \{0,1\}^{q_l}$ which is used to define the current and future operating modes of the controller. The outputs from this system are the current control inputs to the plant $u(t)$ and estimates of the past, current and future states $\hat{x}(t)$ and past and current faults $\hat{\phi}(t) \in \{0,1\}^f$. The last two outputs could be used by the system operator or software to activate alarms or to set the operating mode $\psi(t)$.

The operating modes of the controller can be written in terms of propositional logic statements

$$\begin{aligned} condition_1 &\rightarrow objectives_1 \\ condition_2 &\rightarrow objectives_2 \\ \vdots \quad &\quad \vdots \end{aligned} \tag{15}$$

where *condition* can be any literal made up of logic statements regarding the current and future states, inputs and operating mode of the plant. The objectives for the given operating mode are defined by the literal *objectives*.

The priorities of soft constraints for each *condition* are defined by different combinations of (7) in *objectives*. Constraints can be added or relaxed by modifying *objectives* appropriately. The variables to be minimised in each *condition*, and their corresponding weights, are defined by adding

$$v = \sum_k \delta_k w_k g_k(x,u,r,z,\delta) \tag{16}$$

to the corresponding *objectives* where $g_k(x,u,r,z,\delta)$ is the linear function to be minimised, w_k is the corresponding weight. The logic variable $\delta_k = 1$ if and only if some given user-specified criteria regarding the operating condition and state of the system have been met; δ_k has to satisfy the inequality $\sum_k \delta_k \leq 1$. The auxiliary variable v is included in the cost function of the MIQP; δ_k determines whether or not $g_k(x,u,r,z,\delta)$ is included in the cost function.

5 Illustrative Example

Consider the 2-input ($m = 2$), 2-state ($n = 2$) discrete-time LTI system $x(k+1) = \Phi x(k) + \Gamma u(k)$, subject to hard constraints on the input $|u(k)| \leq d, k = 0 \ldots P-1$ and soft constraints on the states $|x(k)| \leq h + \varepsilon(k), k = 1 \ldots P$. Assume that it is more important to satisfy constraints on x_1 than on x_2. Also, one would like constraint violations to occur only in the first part of the prediction horizon. The aim of the controller is to regulate the states to the origin. The prediction horizon length is $P = 3$.

Since both upper and lower bounds on the states cannot be violated simultaneously, logic variables need be associated only with $nP = 6$ slack variables:

$$[\delta_i(k) = 0] \rightarrow [\varepsilon_i(k) = 0], \quad i = 1,2,\ k = 1,2,3\ .$$

If the logic vector $\delta = [\delta_1(3)\ \delta_1(2)\ \delta_1(1)\ \delta_2(3)\ \delta_2(2)\ \delta_2(1)]'$ is constructed with time and output priorities taken into account, then $M_p = [32\ 16\ 8\ 4\ 2\ 1]'$ will guarantee the maximum number of satisfied constraints, while taking priorities into account. If $M_p = [1\ 1\ 1\ 1\ 1\ 1]'$, then the maximum number of satisfied constraints is guaranteed, without taking priorities into account. Alternatively, if time priorities are defined as in (12):

$$[\delta_i(k) = 0] \rightarrow [\delta_i(k+1) = 0], \quad i = 1,2,\ k = 1,2$$

then $M_p = [4\ 4\ 4\ 1\ 1\ 1]'$ guarantees an output-prioritised minimum time-optimal solution.

Additionally, it is desired that the penalty weight on $x_2(k)$ be increased from $\frac{1}{2}$ to 1 when $|x_1(k)| \leq h/2$. By defining

$$v(k) = \delta_x(k)x_2(k) + (1-\delta_x(k))\frac{1}{2}x_2(k), \quad k = 1,2,3$$

as in (16) and introducing the propositional logic statement

$$[\delta_x(k) = 1] \leftrightarrow [|x_1(k)| \leq h/2], \quad k = 1,2,3$$

then $\sum_{k=0}^{P}(\|x_1(k)\|^2 + \|v(k)\|^2 + \|u(k)\|^2 + \|\varepsilon(k)\|^2) + \rho M_p'\delta$ can be thought of as a cost function with state-dependent weights on $x_2(k)$. After setting up the linear inequalities from the propositional logic statements, the MIQP problem is solved.

6 Conclusions

This paper presented systematic and flexible methods, which use propositional logic and the MLD modelling formalism, for prioritising soft constraints in MPC and guaranteeing the satisfaction of the maximum number of constraints. The method presented here can also be used to prioritise other conditions in a hybrid system, such as the status of valves or switches. The optimality, advantages and disadvantages of the prioritisation methods were briefly discussed. A method which does not require logic variables for prioritising soft constraints in an exact penalty function was also presented.

Finally, a framework was introduced which allows a large class of multi-objective and reconfiguration problems to be described, such as logic- and state-dependent objectives, constraints and priorities. These objectives and priorities are taken into account during the calculation of the optimal control profile.

References

[1] S.J. Qin and T.A. Badgwell. An overview of industrial model predictive control technology. In *Proceedings of Chemical Process Control - CPC V*, Tahoe City, CA, 1996. AIChE.

[2] N.M.C. de Oliveira and L.T. Biegler. Constraint handling and stability properties of model-predictive control. *AIChE Journal*, 40(7):1138–1155, July 1994.

[3] P.O.M. Scokaert and J.B. Rawlings. Feasibility issues in model predictive control. *AIChE Journal*, 45(8):1649–1659, August 1999.

[4] J. Vada, O. Slupphaug, and B.A. Foss. Infeasibility handling in linear MPC subject to prioritized constraints. In *Proceedings of the IFAC'99 World Congress*, Beijing, China, July 1999.

[5] J. Vada, O. Slupphaug, and T.A. Johansen. Efficient infeasibility handling in linear MPC subject to prioritized constraints. In *Proceedings of the European Control Conference*, Karlsruhe, Germany, August 1999.

[6] M.L. Tyler and M. Morari. Propositional logic in control and monitoring problems. *Automatica*, 35:565–582, 1999.

[7] A. Bemporad and M. Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35:407–427, 1999.

[8] A. Bemporad, D. Mignone, and M. Morari. An efficient branch and bound algorithm for state estimation and control of hybrid systems. In *Proceedings of the European Control Conference*, Karlsruhe, Germany, August 1999.

[9] A. Bemporad, D. Mignone, and M. Morari. Moving horizon estimation for hybrid systems and fault detection. In *Proceedings of the American Control Conference*, pages 2471–2475, San Diego, June 1999.