Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

Algorithms for Online Implementations of Explicit MPC Solutions

D. Sui L. Feng M. Hovd

Department of Engineering Cybernetics, Norwegian University of Science and Technology, Norway, 7491 (e-mail: {sui.dan;feng.le;morten.hovd}@itk.ntnu.no)

Abstract: One of the key problems in Model Predictive Control (MPC) is the inherent on-line computational complexity, which restricts its application to slow dynamic systems. To address this issue, multi-parametric programming technique is introduced in MPC (explicit MPC), where the optimization effort is moved off-line. The optimal solution is given in an explicitly piecewise affine function defined over a polyhedral subdivision of the set of feasible states. Instead of solving an optimization problem, the on-line work is simplified to identify the region the current state belongs to and simply evaluate the piecewise affine function. Hence, identifying of the member of the solution partition that contains a given point (referred to as a point location problem) impacts on the time to implement the explicit controller in real-time, which is one component of the complexity of explicit MPC. In this paper, two simple algorithms for point location problems are proposed to efficiently implement of explicit MPC solutions, which aim at reducing the number of polyhedral sets that are candidates to contain the state at the next time instant.

Keywords: Explicit model predictive control; Point location problems; Reachability analysis.

1. INTRODUCTION

Consider the following discrete-time constrained linear system with bounded disturbances:

$$x(t+1) = Ax(t) + Bu(t) + w(t) \tag{1}$$

$$u(t) \in U, \quad x(t) \in X, \ \forall t \geq 0, \tag{2}$$

$$w(t) \in W, \qquad \forall t \geq 0, \tag{3}$$

where $x(\cdot), u(\cdot)$ and $w(\cdot)$ are the state, control and disturbance variables respectively and $X \subset \mathbb{R}^{n_x}, U \subset \mathbb{R}^{n_u}, W \subset \mathbb{R}^{n_w}$ are the corresponding constraints and disturbance set, each being a non-empty convex polytope containing the origin in its interior.

Model predictive control (MPC) is one strategy that deals with controller design for such system with physical constraints. Over the last few decades, MPC strategy has received much consideration since it is possible to handle constraints on input, state and output signals during the design procedure of the controller. However, its heavy on-line computational complexity is a key factor that limits MPC (especially robust MPC) to slow dynamic or small-scale systems. Recently, a lot of work that aims at reducing the on-line computational complexity has been reported. For example, Bemporad et al. (2002a,b) proposed an efficient approach, where the optimization effort of MPC for linear systems can be moved off-line. This approach is based on multi-parametric programming, through which the optimal solution is given in an explicitly piecewise affine function defined over a polyhedral subdivision of the set of feasible states, *i.e.*

$$u^*(t) = L_i x(t) + g_i, \ \text{ if } \ x(t) \in \mathscr{R}_i, \ \forall i \in \mathscr{I}, \tag{4}$$

where matrixes $L_i \in \mathbb{R}^{n_u \times n_x}$ and $g_i \in \mathbb{R}^{n_u}$ are associated with a convex polyhedral set $\mathscr{R}_i$ in $\mathbb{R}^{n_x}$, the set of feasible states $\mathscr{P}$ is a union of all these partitions, *i.e.* $\mathscr{P} = \cup_{i \in \mathscr{I}} \mathscr{R}_i$ and $int(\mathscr{R}_i) \cap int(\mathscr{R}_j) = \emptyset$ for all $i \neq j, i, j \in \mathscr{I}$. Once this affine function has been pre-calculated, the optimal solution can be computed for a particular parameter by determining the region that contains it.

The advantage of explicit MPC is that no time-consuming optimization is necessary and the control input can be computed within a short time period. Multi-parametric programming significantly decreases the cost of applying MPC to industrial systems and considerably simplifies the on-line implementation. However, multi-parametric programming also suffers some serious drawbacks, for example, the number of partitioning regions grows exponentially with the size of the control problem and may quickly reach a prohibitive number of elements, see Grieder (2004). In order to apply multi-parametric programming to cases where complex mathematical models or fast response systems are used, its computational complexity should be reduced.

As stated above, the on-line implementation of explicit MPC is simplified to identify where the current state $x(t)$ is. Once the region is found, the affine optimal control law associated with the region is evaluated and applied to the system. Identifying which region a given state belongs to is one of the key complexity issues of explicit MPC. Identification of the region that contains the current state is similar to a point location problem.

Problem 1 (Point location problem, see Spjøtvold et al. (2006)) Given a polyhedral cover $\mathscr{R} := \{\mathscr{R}_1, \mathscr{R}_2, \ldots, \mathscr{R}_p\}$ such that for every pair $(\mathscr{R}_i, \mathscr{R}_j) \in \mathscr{R} \times \mathscr{R}$, $i \neq j$, we have $int(\mathscr{R}_i) \cap int(\mathscr{R}_j) = \emptyset$. Given a point $x \in \mathscr{R}$, find $\mathscr{R}_i \in \mathscr{R}$ such that $x \in \mathscr{R}_i$.

⋆ This work is supported by Norwegian University of Science and Technology and project "Design of Advanced Controllers for Economic, Robust and Safe Manufacturing Performance".

Some key contributions, for example, Tøndel et al. (2003); Borrelli et al. (2001); Jones et al. (2006); Pannocchia et al. (2007); Spjøtvold et al. (2006) have been proposed to efficiently solve point location problems. In Tøndel et al. (2003), binary search tree is constructed over the polyhedral partitions. The auxiliary hyperplanes are used to subdivide the partition at each tree level. In Borrelli et al. (2001), it is shown that the region containing the point is associated to the piecewise affine optimal function $J^*(\cdot)$ with the largest value. Thereby, instead of identifying the region, each affine value $J^*(\cdot)$ corresponding to each region is calculated for the current point and compared to get the largest one. Recently, such a problem can be written as a weighted nearest neighbor search (Jones et al. (2006)) that can be solved in time linear in n_x, whereas this algorithm considers only the case when the cost function of optimization problems is linear. In Spjøtvold et al. (2006), authors proposed to utilize reachability analysis to solve a reduced point location problem instead of resolving the entire point location problem at every time instant.

In this paper, two simple algorithms for point location problems are proposed to efficiently implement of explicit MPC solutions. In the first algorithm, one suitable point contained in each associated region is first chosen as a reference point. The point location problem can be easily solved via comparison of the distances between each chosen point and the current state. In the second algorithm, modifications of reachability analysis approach is presented, which aims at reducing the number of polyhedral sets that are candidates to contain the state at the next time instant and avoiding the heavy pre-processing time. Both algorithms are introduced in detail in the next two sections.

Notation and Basic Definitions: For any matrix A, $\|A\|_\ell$ refers to the ℓ-norm of A, which has the following properties: $\|A+B\|_\ell \le \|A\|_\ell + \|B\|_\ell$; $\|AB\|_\ell \le \|A\|_\ell \|B\|_\ell$. $\|x\|_\ell$ refers to the ℓ-norm of vector $x \in \mathbb{R}^n$. $B_\ell(\varepsilon)$ is the ℓ norm-ball, *i.e.* $B_\ell(\varepsilon) = \{x \in \mathbb{R}^n : \|x\|_\ell \le \varepsilon\}$. Suppose $X \in \mathbb{R}^n$, then the interior of X is $int(X)$; $|X|$ is its cardinality. Suppose $X, Y \subset \mathbb{R}^n$, the Minkowski sum is $X \oplus Y = \{z \in \mathbb{R}^n : z = x+y, x \in X, y \in Y\}$.

Definition 1(One-step reachable set) The one-step reachable set $Reach(\Omega)$ of system (1)-(3) is the set of states to which system (1)-(3) under controller (4) evolves at the next time step from $x \in \Omega$, for all allowable disturbance $w \in W$.

To characterize it, let

$$Reach(\Omega) = \{x^+ : \exists x \in \Omega, x^+ = Ax + Bu^* + w,\ \forall w \in W\}. \quad (5)$$

2. COMPARISON OF DISTANCES

In this section a simple algorithm is proposed to solve a point location problem via comparison of the distances between points. In each region $\mathscr{R}_i$, some point is chosen as a reference point, which is shown as a dark circle in Fig. 1. The distance of the current state (shown as a dark triangle in Fig. 1) and each chosen point is computed. The values of distances may indicate a location search direction. In virtue of this useful information, the point location problem can be solved easily.

The off-line work is stated as follows: choose one suitable point in each region $\mathscr{R}_i$. Generally, the Chebychev center x_c^i of $\mathscr{R}_i$ is a good choice.

Given the current state $x(t)$, the first step of the on-line work is to compute the distance between the points $(x(t), x_c^i), \forall i \in \mathscr{I}$, *i.e.*

$$d_i(t) = \|x(t) - x_c^i\|_2,\ \forall i \in \mathscr{I}.$$

Define $\mathscr{D}(t) = \{d_i(t), \forall i \in \mathscr{I}\}$. Next step is to sort the elements of $\mathscr{D}(t)$ in ascending order and its corresponding index set is defined by $I(t) = \{j_1, j_2, \ldots\}$, where $j_1, j_2, \ldots \in \mathscr{I}$. To locate the point $x(t)$, first searching the region $\mathscr{R}_i$ from the index $i = j_1$ (the distance between $(x(t), x_c^{j_1})$ is smallest), check whether $x(t) \in \mathscr{R}_{j_1}$. If not, repeat it from the index $i = j_2$ and so on. In summary, the on-line algorithm to identify $x(t)$ is given as below:

Algorithm 1 At time t, the on-line computation involved for the point location problem is
(1) Compute $d_i(t), \forall i \in \mathscr{I}$, sort the set $\mathscr{D}(t)$ and obtain its index set $I(t)$;
(2) Set $k = 1$;
(3) Check whether $x(t) \in \mathscr{R}_{j_k}$ where $j_k \in I(t)$. If not, set $k := k+1$ and repeat Step (3).

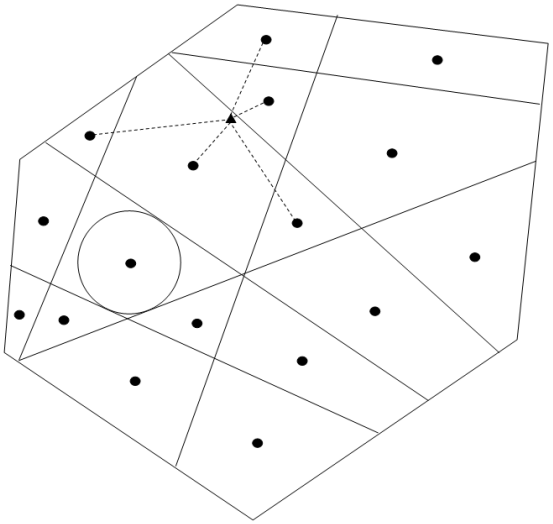

Fig. 1. Illustration of the point location problem via comparison distances. Chebychev centers are shown as dark circles; Point $x(t)$ is shown as a dark triangle; The distances between $x(t)$ and several Chebychev centers are shown by dash-line.

In general, the distance between x and the Chebychev center of the region where x stays is respectively small. It may be a useful clue for solving the point location problem. Compared with the linear search through the entire region $\mathscr{P}$ approach, Algorithm 1 seems more efficient. For example, in Fig. 1, the number of partitioned regions is 17. By using Algorithm 1, x will be successfully located at the second iteration ($k = 2$). However, for the linear search approach, in some worst case, it possibly takes 17 iterations to locate x.

The process starts the search from the shortest distance to the longest one, which actually provides a direction for the location search. It is reasonable to believe that few iterations are needed to locate x by using Algorithm 1 in some cases. The idea of Algorithm 1 is very simple and straightforward. Unlike the recent results for efficiently solving point location problems, its off-line work is much time-cheap. It can be

used for a general class of explicit MPC problems and for high dimensional models. However, it should be admitted that Algorithm 1 still possibly takes long time to locate x in some worst cases, for example, when the distance between x and the Chebychev center of its associated region is large.

3. REACHABILITY ANALYSIS

Instead of searching through the entire set $\mathscr{P}$ at each sample time, Spjøtvold et al. (2006) utilizes reachability analysis to solve a reduced point location problem at every time instant. In the algorithm, the reachable set $Reach(\mathscr{R}_i)$ needs to be computed. From (5), such a set can be characterized as

$$Reach(\mathscr{R}_i) = (A+BL_i)\mathscr{R}_i \oplus Bg_i \oplus W,\ \forall i \in \mathscr{I}. \tag{6}$$

Since equation (6) includes the Minkowski sum operation, the characterization of $Reach(\mathscr{R}_i)$ becomes very difficult and time-expensive, especially when n_x is large. Hence, this work is not applicable to complicated controller structures due to the prohibitive pre-processing time. In this section, an alternative algorithm is proposed, which also utilizes reachability analysis to solve a reduced point location problem. Unlike the work in Spjøtvold et al. (2006) to exactly compute $Reach(\mathscr{R}_i)$, a set of states that be reached at the next time instant from $x(t)$ is estimated via employing the system dynamics, the explicit control law and some reference points. The proposed work avoids the Minkowski sum operation and has less computational complexity, therefore, it needs less pre-processing time and is easily applicable.

The idea of the proposed algorithm is to estimate a set $x(t+1)$ reaches via utilizing some reference points. The members of the partition intersecting this set are guaranteed to contain $x(t+1)$, which are candidates to be searched in the next time step. Hence, instead of solving the entire point location problem at next step, a reduced point location problem associated with these regions is solved.

If it is known that $x(t) \in \mathscr{R}_i$, to estimate where $x(t+1)$ goes to, some reference point in $\mathscr{R}_i$ is needed in the proposed algorithm. Suppose in each region $\mathscr{R}_i$, some suitable point is chosen, defined by x_i (see dark circles in Fig. 2). Its nominal successive state is $\mathrm{x}_i^+ = A\mathrm{x}_i + Bu$, where $u = L_i\mathrm{x}_i + g_i$ (for example, see a white circle in Fig. 2). Since $x(t) \in \mathscr{R}_i$, $x(t+1) = Ax(t) + Bu^*(t) + w(t)$ where $u^*(t) = L_i x(t) + g_i$. Thus the difference between the points $x(t+1)$ and x_i^+ is, *i.e.*

$$x(t+1) - \mathrm{x}_i^+ = \Phi_i \delta(t) + w(t), \tag{7}$$

where $\Phi_i = A + BL_i$ and $\delta(t) = x(t) - \mathrm{x}_i$. If $\|\delta(t)\|_2 \le d_{max,i}$ and the disturbance $\|w(t)\|_2 \le \mathrm{w}, \forall t \ge 0$, then

$$\|x(t+1) - \mathrm{x}_i^+\|_2 \le \rho, \tag{8}$$

where $\rho = \|\Phi_i\|_2 d_{max,i} + \mathrm{w}$. Thereby, the distance between $(x(t+1), \mathrm{x}_i^+)$ is less than some upper bound ρ. Define the convex set $\Omega(\mathrm{x}_i) = \{x \in \mathbb{R}^{n_x} : \|x - \mathrm{x}_i^+\|_2 \le \rho\}$. It is easy to know that the state will reach the set $\Omega(\mathrm{x}_i)$ at the next time step. Define a new index set $N(\mathrm{x}_i) = \{i \in \mathscr{I} : \mathscr{R}_i \cap \Omega(\mathrm{x}_i) \neq \emptyset\}$. Then, if $x(t) \in \mathscr{R}_i$, referring to the given point x_i, the point location problem at the next time step reduces to a search through the set $N(\mathrm{x}_i)$, instead of searching the entire region $\mathscr{P}$.

For example, in Fig. 2, the number of partitioned regions is 27. For region $\mathscr{R}_1$ and point x_1, the index set $N(\mathrm{x}_1) = \{9, 10, 11, 12\}$. It means that the next state $x(t+1)$ will enter one of these four regions $\mathscr{R}_9, \mathscr{R}_{10}, \mathscr{R}_{11}, \mathscr{R}_{12}$ if $x(t) \in \mathscr{R}_1$. Hence, at time $t+1$, instead of searching a total of 27 regions, only 4 regions will just need to be searched.

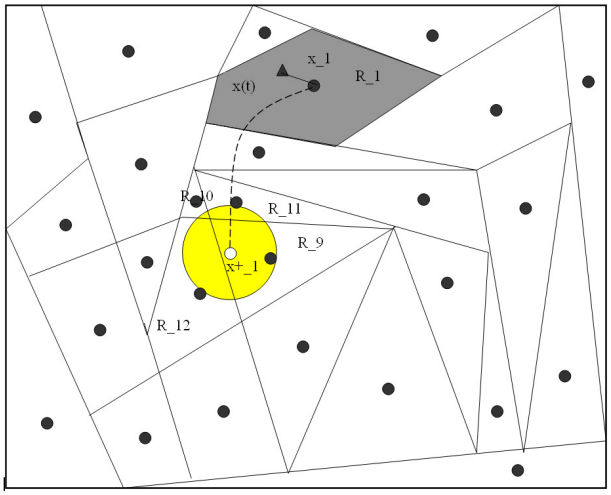

Fig. 2. Illustration of the point location problem via reachability analysis. Chosen points x_i are shown as dark circles; Point $x(t) \in \mathscr{R}_1$ is shown as a dark triangle; x_1^+ is shown as a white circle; The set $\Omega(\mathrm{x}_1)$ is shown as a shaded circle.

The key point of this algorithm is how to select appropriate points x_i such that sets $\Omega(\mathrm{x}_i)$ are not over-estimated. In general, x_i can be chosen as the Chebychev center of each region $\mathscr{R}_i$. Suppose each region $\mathscr{R}_i$ can be written as a set of linear inequalities, *i.e.* $\mathscr{R}_i = \{x \in \mathbb{R}^{n_x} : M_i x \le b_i\}$. The upper bound $d_{max,i}$ is obtained by solving the following optimization problem

$$\begin{aligned} &\max_x (x - \mathrm{x}_i)^T (x - \mathrm{x}_i) \\ &s.t.\ M_i x \le b_i. \end{aligned} \tag{9}$$

Let the optimal solution is x^*. Then $d_{max,i} = \|x^* - \mathrm{x}_i\|_2$. Such a choice of $d_{max,i}$ means that for all points in $\mathscr{R}_i$, the distance between it and x_i is less than $d_{max,i}$. However, based on this worst estimation of $d_{max,i}$, the size of $\Omega(\mathrm{x}_i)$ may be large, which directly implies that $N(\mathrm{x}_i)$ may have many elements. Therefore, instead of employing a single $d_{max,i}$, we use a sequence of values $\{d_{max,i}^j,\ j = 1, \ldots, n\}$, where $n \ge 1$ is an integer, $d_{max,i}^j < d_{max,i}^{j+1}$ and $d_{max,i}^n = d_{max,i}$ to compute a corresponding sequence of reachable sets $\{\Omega^j(\mathrm{x}_i),\ j = 1, \ldots, n\}$ and index sets $\{N^j(\mathrm{x}_i),\ j = 1, \ldots, n\}$. For convenience, the value $d_{max,i}^j$ can be chosen as $d_{max,i}^j = \frac{j}{n} d_{max,i}$.

In summary, the off-line computation required and the online algorithm are stated in Algorithm 2.1 and 2.2, respectively for point location problems via utilizing reachability analysis.

Algorithm 2.1 Off-line computation:
(1) Select a reference point x_i in each region $\mathscr{R}_i$;
(2) Compute the value $d_{max,i}$ for all $i \in \mathscr{I}$ via problem (9);
(3) Calculate the sequence of sets $\{\Omega^j(\mathrm{x}_i)\}$ and its corresponding index sets $\{N^j(\mathrm{x}_i)\}$.

Algorithm 2.2 At time t, given the current state $x(t)$ and known that $x(t) \in \mathscr{R}_i$, the on-line computation involved for the point location problem is
(1) Compute $\|\delta(t)\|_2$;

(2) Choose the index j^*, where $j^* = \min_j j \;\; s.t. \;\; \|\delta(t)\|_2 \leq d^j_{max,i}$;
(3) Compute $x(t+1) = \Phi_i x(t) + Bg_i + w(t)$;
(4) Recall the index set $N^{j^*}(\mathrm{x}_i)$ and solve the point location problem for $N^{j^*}(\mathrm{x}_i)$, that is, find $\mathscr{R}_k$, where $k \in N^{j^*}(\mathrm{x}_i)$ such that $x(t+1) \in \mathscr{R}_k$.

Remark 1. Note that the smaller the value of the index j^* is, the fewer elements $N^{j^*}(\mathrm{x}_i)$ has.

Due to inappropriately selecting the point x_i or the case that one point x_i is not enough for a large region $\mathscr{R}_i$, the value of the distance $\|\delta(t)\|_2$ may be large, resulting in a set $N(\mathrm{x}_i)$ having many elements. Alternative algorithm is proposed as follows to limit the number of elements of sets $N(\mathrm{x}_i)$.

Unlike Algorithm 2, in Algorithm 3, instead of selecting one point x_i for each region $\mathscr{R}_i$, several equidistantly placed points are placed into the region $\mathscr{R}_i$. The points are distributed in a way such that each axis is first divided into a total of m points, and only the points which are contained in $\mathscr{R}_i$ are taken. The points are defined by x_i^p (see dark circles in Fig. 3). The algorithm for generating these points named **grid** is available in Matlab (Multi-Parametric Toolbox, Kvasnica et al. (2005)).

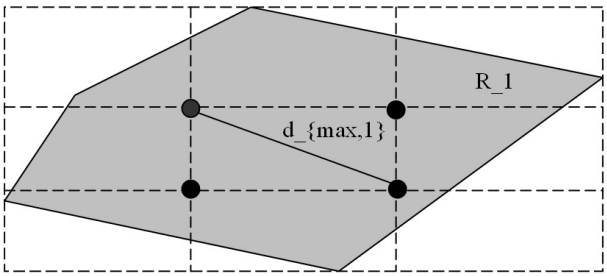

Fig. 3. Illustration of griding a polytope. Chosen points x_i^p are shown as dark circles.

If $x \in \mathscr{R}_i$, there always exists a point x_i^p such that $\|x - \mathrm{x}_i^p\|_2 \leq d_{max,i}$, where $d_{max,i}$ depends on the size of the grid, see Fig. 3. It is necessary to note that the positive integer m determines the size of the grid, in turn, determines the value of $d_{max,i}$. The off-line work can start from a small value of m to compute the set $N(\mathrm{x}_i^p)$. If the ratio of the number of elements of $N(\mathrm{x}_i^p)$, $|N(\mathrm{x}_i^p)|$, to the number of total partitioned regions, $|\mathscr{I}|$, is less than some prescribed value ε, then stop; else, increase the value of m. By doing so, no matter where $x(t)$ is, the number of regions which will be searched at the next step is bounded. It is less than $\varepsilon|\mathscr{I}|$.

In summary, the off-line computation required and the online algorithm are stated in Algorithm 3.1 and 3.2, respectively.

Algorithm 3.1 Off-line computation:
(1) Choose a positive integer $m \geq 1$. Set $j = m$;
(2) Place several equidistantly placed points x_i^p into $\mathscr{R}_i$, which are distributed such that each axis is divided into j points;
(3) Calculate the set $\Omega(\mathrm{x}_i^p)$ and its corresponding index set $N(\mathrm{x}_i^p)$ based on the value $d_{max,i}$;
(4) If $|N(\mathrm{x}_i^p)| \backslash |\mathscr{I}| \leq \varepsilon$, then stop; else, set $j = j+1$, go to Step (2).

Algorithm 3.2 At time t, given the current state $x(t)$ and known that $x(t) \in \mathscr{R}_i$, the on-line computation involved for the point location problem is
(1) Compute $\|\delta^p(t)\|_2$ for each p;
(2) Choose the index p^*, where $p^* = \min_p \|\delta^p(t)\|_2$;
(3) Compute $x(t+1) = \Phi_i x(t) + Bg_i + w(t)$;
(4) Recall the index set $N(\mathrm{x}_i^{p^*})$ and solve the point location problem for $N(\mathrm{x}_i^{p^*})$, that is, find $\mathscr{R}_k$, where $k \in N(\mathrm{x}_i^{p^*})$ such that $x(t+1) \in \mathscr{R}_k$.

Remark 2. The smaller the prescribed value ε is, the fewer the number of regions that will be searched at the next step is. However, it is necessary to note that the reduction comes at the cost of increased off-line work and storage space.

4. CONCLUSION

In this paper, two algorithms are presented in order to efficiently solve the point location problems. Algorithm 1 is very simple and in most cases it will have a good performance. Algorithm 2 and 3 aim at solving a reduced point location problem at every time instant via estimation of the reachable sets. For Algorithm 3, the more m is, the fewer the number of regions that will be searched is. However, this reduction actually comes at the cost of increased off-line burden and required storage space.

REFERENCES

A. Bemporad, F. Borrelli, and M. Morari. Model predicitve control based on linear programming-the explicit solution. *IEEE Transactions on Automatic Control*, 47, 2002a.

A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1):3–20, 2002b.

F. Borrelli, M. Baotić, A. Bemporad, and M. Morari. Efficient on-line computation of constrained optimal control. *Proceeding of 40th IEEE Conference of Decision and Control*, pages 1187–1192, 2001.

P. Grieder. *Efficient Computation of Feedback Controllers for Constrained Systems*. PhD thesis, Swiss Federal Institute of Technology Zurich, 2004.

C. N. Jones, P. Grieder, and S. V. Raković. A logarithmic-time solution to the point location problem for parametric linear programming. *Automatica*, 42:2215–2218, 2006.

M. Kvasnica, P. Grieder, and M. Baotić. Multi-parametric toolbox. *http://control.ee.ethz.ch/ mpt/*, 2005.

G. Pannocchia, J. B. Rawlings, and S. J. Wright. Fast, large-scale model predictive control by partial enumeration. *Automatica*, 43:852–860, 2007.

J. Spjøtvold, P. Raković, Tøndel, and T. A. Johansen. Utilizing reachability analysis in point location problems. *Proc. 45th IEEE conference on decision and control*, 2006.

P. Tøndel, T. A. Johansen, and A. Bemporad. Computation of piecewise affince control via binary search tree. *Automatica*, 39:945–950, 2003.