Available online at www.sciencedirect.com
ScienceDirect
automatica
Automatica 42 (2006) 2215–2218
www.elsevier.com/locate/automatica

Technical communique

A logarithmic-time solution to the point location problem for parametric linear programming☆

C.N. Jones[a,*], P. Grieder[b], S.V. Raković[c]

[a]Control Group, Department of Engineering, University of Cambridge, Trumpington Street, Cambridge CB2 1PZ, UK
[b]Automatic Control Laboratory, Swiss Federal Institute of Technology, Physikstrasse 3, ETL K13.2, CH-8092 Zurich, Switzerland
[c]Imperial College London, Exhibition Road, London SW7 2BT, UK

Received 20 September 2005; received in revised form 10 April 2006; accepted 13 July 2006
Available online 22 September 2006

Abstract

The optimiser of a (multi) parametric linear program (pLP) is a piecewise affine function defined over a polyhedral subdivision of the set of feasible states. 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. This is the so-called *point location problem*. In this paper, we show that this problem can be written as an additively weighted nearest neighbour search that can be solved in time linear in the dimension of the state space and logarithmic in the number of regions.

It is well-known that linear model predictive control (MPC) problems based on linear control objectives (e.g., 1- or ∞-norm) can be posed as pLPs, and on-line calculation of the control law involves the solution to the point location problem. Several orders of magnitude sampling speed improvement are demonstrated over traditional MPC and closed-form MPC schemes using the proposed scheme.

Keywords: Predictive control; Parametric programming; Controller complexity

1. Introduction

In this note, we consider the following parametric linear program (pLP):

$$V^\star(x) = \min_y \{c^{\mathrm{T}} y \mid (x, y) \in \mathscr{P}\}, \tag{1}$$

where $x \in \mathbb{R}^n$ is the parameter, $y \in \mathbb{R}^m$ is the optimiser, $y^\star(x)$ is the set of optimisers that achieves the minimum in (1) and $\mathscr{P}$ is a polytope.[1]

In recent years, it has become well-known that the optimiser $y^\star(\cdot)$ of (1) is a piecewise affine function (PWA) defined over a polyhedral partition of the feasible parameters (Borrelli, 2003).[2] The problem studied in this note is known as the *point location problem*: given the pre-computed PWA function $y^\star(\cdot)$ and a particular parameter x, compute efficiently the vector $y^\star(x)$.

This problem is motivated from the model predictive control (MPC) literature. It is standard practice to implement an MPC controller by solving an on-line optimal control problem that, when the system is linear, the constraints are polyhedral and the cost is linear (e.g. 1-, ∞-norm or polytopic), amounts to computing a single linear program at each sampling instant. By pre-computing the PWA function $y^\star(\cdot)$ off-line, the on-line calculation of the control input then becomes one of solving the point location problem and therefore an efficiency improvement translates directly into improved sampling rates and/or reduced hardware costs.

☆ This paper was presented at the IFAC 2005 meeting. This paper was recommended for publication in revised form by Associate Editor Jay H. Lee under the direction of editor A.L. Tits.

* Corresponding author.
E-mail address: cnj22@cam.ac.uk (C.N. Jones).

[1] See Definitions section for definition of polytope.

[2] Several methods of computing this affine function can be found in the literature (e.g., Bemporad, Morari, Dua, & Pistikopoulos, 2002; Borrelli, 2003; Tøndel, Johansen, & Bemporad, 2003a).

0005-1098/$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.automatica.2006.07.010

The complexity of solving the point location problem is clearly dependent on the number of affine regions in the solution $y^\star(\cdot)$. This number of regions is known to grow very quickly and possibly exponentially, with horizon length and parameter dimension (Bemporad et al., 2002). The complexity of the solution therefore implies that for large problems an efficient method for solving the point location problem is needed.

The key contributions to this end have been made in Tøndel, Johansen, and Bemporad (2003b) and Borelli, Baotić, Bemporad, and Morari (2001). In Tøndel et al. (2003b), the authors propose to construct a binary search tree over the polyhedral partition. Therein, auxiliary hyper-planes are used to subdivide the partition at each tree level. Note that these auxiliary hyperplanes may subdivide existing regions. The necessary on-line identification time is logarithmic in the number of subdivided regions, which may be significantly larger than the original number of regions. Although the scheme works very well for smaller partitions, it is not applicable to large controller structures due to the prohibitive pre-processing time or online memory requirements. However, the scheme in Tøndel et al. (2003b) is applicable to *any* type of piecewise-affine controller, whereas the algorithm proposed in this paper considers only the case in which controllers are obtained via an MPC problem with a *linear* cost.

In Borelli et al. (2001) the authors exploit the convexity properties of the PWA value function $V^\star(\cdot)$ to solve the point location problem efficiently. Instead of checking if the point is contained in a polyhedral region, each affine piece of the value function is evaluated for the current state. Since the value function is PWA and convex, the region containing the point is associated to the affine function that yields the largest value. Although this scheme is efficient, it is still linear in the number of regions.

In this note, we combine the concept of region identification via the value-function (Borelli et al., 2001) with the construction of search trees (Tøndel et al., 2003b), by using the link between *parametric* linear programming and Voronoi diagrams recently established in Raković, Grieder, and Jones (2004). We demonstrate that the PWA cost function can be interpreted as a weighted power diagram, which is a type of Voronoi diagram, and exploit results in Arya, Mount, Netanyahu, Silverman, and Wu (1998) to solve the point location problem for Voronoi diagrams in logarithmic time at the cost of very simple pre-processing operations on the controller partition.

Definitions and notation

A *polyhedron* is the intersection of a finite number of halfspaces and a *polytope* is a bounded polyhedron. If P is a polyhedron and $H = \{x \mid a^{\mathrm{T}}x \leqslant b\}$ is a halfspace such that $P \subseteq H$, then $P \cap \{x \mid a^{\mathrm{T}}x = b\}$ is a *face* of P. Given any integer q let $\mathbb{N}_q \triangleq \{1, 2, \ldots, q\}$.

2. Problem formulation

We begin with a discussion of the structure of the solution to (1). First, we need to introduce the notion of a *complex* of polytopes:

Definition 1 (*Grünbaum, 2000*). A finite family $\mathscr{C}$ of polyhedra in $\mathbb{R}^n$ is a *complex* if

- every face of a member of $\mathscr{C}$ is itself a member of $\mathscr{C}$,
- the intersection of any two members of $\mathscr{C}$ is a face of each of them.

If a polyhedron $\mathscr{Q} \in \mathscr{C}$ and is of dimension n, then we call $\mathscr{Q}$ a *cell* of the complex.

A basic result on the nature of the solution to a PLP is given next:

Theorem 1 (*Borrelli, 2003*). *Let $\mathscr{P} \subset \mathbb{R}^{n+m}$ be a polyhedron and*

$$\pi(\mathscr{P}) \triangleq \{x \in \mathbb{R}^n \mid \exists y \in \mathbb{R}^m \text{ such that } (x, y) \in \mathscr{P}\}.$$

For each x in $\pi(\mathscr{P})$, let $V^\star(x)$ be as defined in (1).

Then $V^\star : \mathbb{R}^n \to \mathbb{R}$ is a convex, PWA defined over a complex $\mathscr{C}$, such that the union of the cells of $\mathscr{C}$ is the set $\pi(\mathscr{P})$. Furthermore, there exists a continuous, PWA[3] *$y^\star(\cdot) : \mathbb{R}^n \to \mathbb{R}^m$ such that $c^{\mathrm{T}}y^\star(x) = V^\star(x)$ for every $x \in \pi(\mathscr{P})$.*

Thus by Theorem 1, the optimal cost of (1) is a convex, PWA of the parameter x, taking $\mathbb{R}^n$ to $\mathbb{R}$ and is defined over a complex $\mathscr{C} = \{\mathscr{R}_1, \ldots, \mathscr{R}_R\}$:

$$V^\star(x) = F_r^{\mathrm{T}}x + f_r \quad \text{if } x \in \mathscr{R}_r, \quad r \in \mathbb{N}_R, \tag{2}$$

where each cell $\mathscr{R}_r$ is a polyhedron. Furthermore, the optimiser of pLP (1) is a PWA of x taking $\mathbb{R}^n$ to $\mathbb{R}^m$:

$$y^\star(x) = T_r x + t_r \quad \text{if } x \in \mathscr{R}_r, \quad r \in \mathbb{N}_R.$$

2.1. Point location problem

Problem 1. *Given a parameter x and complex $\mathscr{C} = \{\mathscr{R}_1, \ldots, \mathscr{R}_R\}$, determine any integer*[4] *$\mathrm{i}(x) \in \mathbb{N}_R$ such that polyhedron $\mathscr{R}_{\mathrm{i}(x)}$ contains x.*

The function $\mathrm{i}(\cdot)$ then defines the optimiser $y^\star(\cdot)$ as

$$y^\star(x) = T_{\mathrm{i}(x)}x + t_{\mathrm{i}(x)}.$$

As $V^\star(\cdot)$ is convex, the $\mathrm{i}(\cdot)$ can be written as (Borelli et al., 2001):

$$\mathrm{i}(x) = \arg \max_{r \in \mathbb{N}_R} \{F_r^{\mathrm{T}}x + f_r\}. \tag{3}$$

As was proposed in Borelli et al. (2001), $\mathrm{i}(x)$ can be computed from (3) by simply evaluating the cost $F_r^{\mathrm{T}}x + f_r$ for each $r \in \mathbb{N}_R$ and then taking the largest. This procedure requires $2nR$ flops and has a storage requirement of $(n+1)R$.

This procedure implicitly assumes that the problem is non-degenerate. We make this assumption of non-degeneracy here

[3] Note that in general, the optimiser of (1) is set-valued.

[4] The parameter may be on the boundary of several regions.

for brevity and refer the reader to Jones (2005) for a simple extension that allows this approach to be used for degenerate problems.

In the following sections we will show that with a negligible pre-processing step, (3) can be computed in *logarithmic* time, which is a significant improvement over the *linear* time result of Borelli et al. (2001).

3. Point location and nearest neighbours

In this section we show that for pLPs, the point location problem can be written as an *additively weighted nearest neighbour search*, or a search over R points in $\mathbb{R}^n$ to determine which is closest to the parameter x.

Consider the finite set of points called *sites* $\mathscr{S} \triangleq \{s_1, ..., s_R\}$ and the weights $\mathscr{W} \triangleq \{w_1, \ldots, w_R\}$, where $(s_i, w_i) \in \mathbb{R}^n \times \mathbb{R}, \forall i \in \mathbb{N}_R$. Given a point x in $\mathbb{R}^n$, the weighted nearest neighbour problem is the determination of the point $s_r \in \mathscr{S}$ that is closest to x in the sense defined below, for all $s_j \in \mathscr{S}$, $j \in \mathbb{N}_R$. Associated with each site is a set of points $\mathscr{L}_r \subset \mathbb{R}^n$ such that for each $x \in \mathscr{L}_r$, x is closer to s_r than to any other site

$$\mathscr{L}_r \triangleq \{x \,|\, \|s_r - x\|_2^2 + w_r \leqslant \|s_j - x\|_2^2 + w_j, \forall j \in \mathbb{N}_R\}. \tag{4}$$

Note that the sets $\mathscr{L}_r$ form a complex $\mathscr{C}_V \triangleq \{\mathscr{L}_1, \ldots, \mathscr{L}_R\}$ called a *power diagram*, which is a type of *Voronoi diagram* (Aurenhammer, 1991).

We now prove the main result:

Theorem 2. *If $\mathscr{C}$ is a solution complex of pLP* (1)*, then $\mathscr{C}$ is the intersection of a power diagram with $\pi(\mathscr{P})$.*

Proof. It suffices to show that for any solution complex of pLP (1), $\mathscr{C} \triangleq \{\mathscr{R}_1, \ldots, \mathscr{R}_R\}$, it is possible to define a set of sites and weights such that their power diagram is equivalent to $\mathscr{C}$.

It follows from Theorem 1 and (2)–(3) that x is contained in cell $\mathscr{R}_r$ if and only if

$$F_r^{\mathrm{T}} x + f_r \geqslant F_j^{\mathrm{T}} x + f_j \quad \forall j \in \mathbb{N}_R,$$

Define the R sites and weights as

$$s_r \triangleq \frac{F_r}{2},$$

$$w_r \triangleq -f_r - \left\| \frac{F_r}{2} \right\|_2^2 = -f_r - \|s_r\|_2^2. \tag{5}$$

For all $r \in \mathbb{N}_R$ and a given x it follows that:

$$\|s_r - x\|_2^2 + w_r = -F_r^{\mathrm{T}} x - f_r + \|x\|_2^2.$$

Recalling the definition of $\mathscr{L}_r$ in (4) we obtain the following $\forall j \in \mathbb{N}_R$:

$$\begin{aligned} \mathscr{L}_r &\triangleq \left\{ x \in \mathbb{R}^n \,\middle|\, \begin{array}{l} \|s_r - x\|_2^2 + w_r \\ \quad \leqslant \|s_j - x\|_2^2 + w_j, \end{array} \right\} \\ &= \left\{ x \in \mathbb{R}^n \,\middle|\, \begin{array}{l} -F_r^{\mathrm{T}} x - f_r + \|x\|_2^2 \\ \quad \leqslant -F_j^{\mathrm{T}} x - f_j + \|x\|_2^2, \end{array} \right\} \\ &= \{x \in \mathbb{R}^n \mid F_r^{\mathrm{T}} x + f_r \geqslant F_j^{\mathrm{T}} x + f_j, \} \end{aligned} \tag{6}$$

Therefore, the relationship $x \in R_r \Rightarrow x \in L_r$ has been shown. The reverse implication is not true, since the union of all $\mathscr{L}_r$ covers $\mathbb{R}^n$. However, we see from (6) and Theorem 1 that $\mathscr{L}_r \cap \pi(\mathscr{P}) = \mathscr{R}_r$. Thus the equivalence of the solution complex $\mathscr{C}$ and the intersection of the feasible region of the pLP and the power diagram of the set of sites and weights (5) is established. □

A very important consequence of Theorem 2 is that the point location problem (3) can be solved by determining which site s_r is closest to the parameter x:

$$\begin{aligned} \mathrm{i}(x) &= \left\{ r \in \mathbb{N}_R \,\middle|\, \begin{array}{ll} \|s_r - x\|_2^2 + w_r & \\ \quad \leqslant \|s_j - x\|_2^2 + w_j & \forall j \in \mathbb{N}_R \end{array} \right\} \\ &= \min_{r \in \mathbb{N}_R} \left\| \begin{pmatrix} s_r \\ \sqrt{w_r} \end{pmatrix} - \begin{pmatrix} x \\ 0 \end{pmatrix} \right\|_2^2. \end{aligned}$$

Since this problem has been well studied in the computational geometry literature we propose to adapt an efficient algorithm introduced in Arya et al. (1998) that solves the nearest neighbour problem in *logarithmic time* and thereby solves the point location problem in logarithmic time.

As shown in Arya et al. (1998), it is possible to pre-process the R sites and weights in $O(nR \log R)$ time and $O(nR)$ space, such that the nearest neighbour can be identified in $O(c_{n,\varepsilon} \log R)$ time, where $c_{n,\varepsilon}$ is a factor depending only on state-space dimension n and accuracy ε.

Remark 1. If the cost in (1) is quadratic, then the resulting solution may or may not be a power diagram, although examples are known in which it is not. Therefore, the result presented in this paper does not currently extend in general to parametric quadratic programs.

Remark 2. As the optimiser $y^\star(\cdot)$ can be chosen to be continuous (Borrelli, 2003) the error ε in determining the region translates into a maximum error in the optimiser that is proportional to ε. Therefore, if pLP (1) solves an optimal control problem, the error in the control input can be made arbitrarily small with an appropriate selection of ε.

Remark 3. A description of the complex is not required for the proposed method, but only the cost function $V^\star(\cdot)$, which is returned by all current pLP implementations.

4. Example

In this section we compare the complexity of the approach presented in this paper with that discussed in Borelli et al. (2001) for very large systems. Although the scheme in Tøndel et al. (2003b) may lead to more significant runtime improvements than Borelli et al. (2001) and indeed than the proposed scheme for small examples, the necessary pre-processing time is prohibitive for large partitions. For example, the method (Tøndel et al., 2003b) requires over 150, 000, 000 LPs when applied to a complex that consists of only 12, 290 regions, and we therefore refrain from performing a comparison to that scheme.

The currently available multi-parametric solvers (Kvasnica, Grieder, Baotić, & Morari, 2003; Jones, 2005) produce reliable

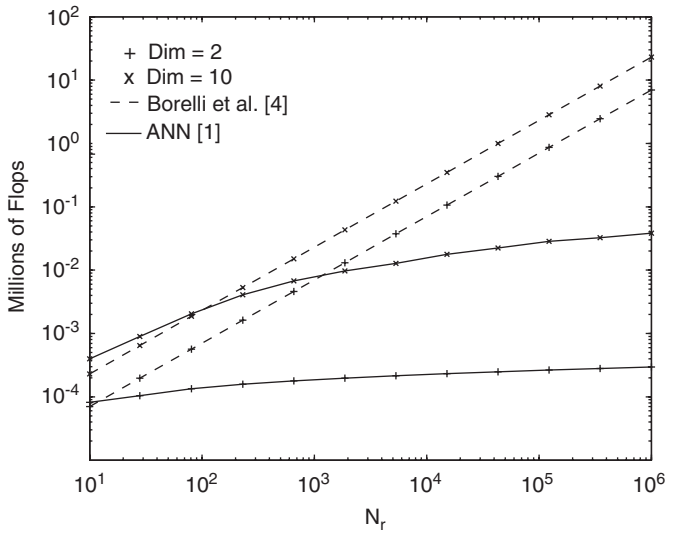

Fig. 1. Comparison of ANN (solid lines) to (Borelli et al., 2001) (dashed lines).

results for partitions of up to approximately 30,000 regions. However, it is of interest to solve much larger control problems and methods are currently being developed that will provide their solutions. Therefore, in order to give a speed comparison we have randomly generated vectors F_r and f_r in the form of (3). The code developed in Arya et al. (1998), which is available at Mount and Arya (1988), was then used to execute 1000 random queries and the worst-case is plotted in Fig. 1. For all of the queries the error parameter ε was set to zero and therefore the solution returned is the exact solution. It should be noted that the preprocessing time for one million regions and 20 dimensions is merely 22.2 s.

Fig. 1 shows the number of floating point operations (flops) as a function of the number of regions for the two approaches and the dimension of the state-space. Note that both axes are logarithmic.

A 3.0 GHz Pentium 4 computer can execute approximately 800×10^6 flops/s. It follows that for a 10-dimensional system whose solution has one million regions, the control action can be computed at a rate of 20 kHz using the proposed method, whereas that given in Borelli et al. (2001) could run at only 35 Hz.

It is clear from Fig. 1 that the calculation speed of the proposed method is very good for systems with a large number of regions. Furthermore, note that problems where ANN does worse than Borelli et al. (2001) are virtually impossible to generate, i.e. a partition in dimension $n=10$ with less than $R=100$ regions is very difficult to contrive. Hence, it can be expected that for all systems of interest, the proposed scheme will result in a significant increase in speed. Since explicit feedback MPC is generally being applied to systems with very fast dynamics, any speedup in the set-membership test is useful in practice, i.e. the scheme proposed here is expected to significantly increase sampling rates.

Remark 4. The examples in this paper have been prepared with the MPT toolbox (Kvasnica et al., 2003) and the ANN library (Mount & Arya, 1988).

References

Arya, S., Mount, D. M., Netanyahu, N. S., Silverman, R., & Wu, A. Y. (1998). An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM*, *45*(6), 891–923.

Aurenhammer, F. (1991). Voronoi diagrams—a survey of a fundamental geometric data structure. *ACM Computing Surveys*, *23*(3),

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

Borrelli, F. (2003). *Constrained optimal control of linear and hybrid systems*, Lecture notes in control and information sciences (Vol. 290). Berlin: Springer.

Borelli, F., Baotić, M., Bemporad, A., & Morari, M. (2001). Efficient on-line computation of constrained optimal control. *Proceedings of the 40th IEEE conference on decision and control* (pp. 1187–1192). Orlando, Florida.

Grünbaum, B. (2000). *Convex polytopes*. 2nd ed., Berlin: Springer.

Jones, C. N. (2005). *Polyhedral tools for control*. Ph.D. thesis, University of Cambridge, 2005.

Kvasnica, M., Grieder, P., Baotić, M., & Morari, M. (2003). Multi parametric toolbox (MPT). In *Hybrid systems*: *Computation and control*, Lecture Notes in Computer Science. Berlin: Springer.

Mount, D., & Arya, S. (1998). ANN: Library for approximate nearest neighbor searching.

Raković, S. V., Grieder, P., & Jones, C. (2004). *Computation of Voronoi diagrams and delaunay triangulation via parametric linear programming*. Technical Report AUT04-03, Automatic Control Lab, ETHZ, Switzerland.

Tøndel, P., Johansen, T. A., & Bemporad, A. (2003a). An algorithm for multi-arametric quadratic programming and explicit MPC solutions. *Automatica*, *39*(3), 489–497.

Tøndel, P., Johansen, T. A., & Bemporad, A. (2003b). Computation of piecewise affine control via binary search tree. *Automatica*, *39*(5), 945–950.