Efficient Robust Model Predictive Control using Chordality

Shervin Parvini Ahmadi, Anders Hansson and Sina Khoshfetrat Pakazad

Ahmadi, S. P., Hansson, A., Pakazad, S. K., (2019), Efficient Robust Model Predictive Control using Chordality, *2019 18TH EUROPEAN CONTROL CONFERENCE (ECC)*, 4270-4275.
https://doi.org/10.23919/ECC.2019.8796011

Efficient Robust Model Predictive Control using Chordality

Shervin Parvini Ahmadi[1], Anders Hansson[2] and Sina Khoshfetrat Pakazad[3]

***Abstract*— In this paper we show that chordal structure can be used to devise efficient optimization methods for robust model predictive control problems. To this end, first the problem is converted to an equivalent robust quadratic programming formulation. We then illustrate how the chordal structure can be used to distribute the computations in a primal-dual interior-point method among computational agents, which in turn allows us to accelerate the algorithm by efficient parallel computations. We investigate performance of the framework in Julia using numerical examples.**

I. Introduction

Model Predictive Control (MPC) is a class of controllers which predicts the future response of a plant using an explicit process model. It is an important control strategy which has been widely used in industry in recent years, [26]. Its root goes back to [6]. The main reason for its application in industry is its capability in handling constraints on control signals and states. In the early years, however, its usage was restricted to traditional process industry since the computational power was limited in terms of speed. A significant amount of research has been conducted since then and as the computational power has grown, the applicability has been extended to faster and more time critical processes. As was mentioned in [15], "one avenue has been what is called explicit MPC, [2], where the optimization problem is solved parametrically off-line. Another avenue has been to exploit the inherent structure of the optimization problems stemming from MPC, [11], [31], [28], [3], [32], [27], [13], [14], [29], [17], [1], [7], [4], [30], [16], [8], [9], [19], [23]. Typically this has been to use Riccati recursions to efficiently compute search directions for Interior Point (IP) methods or actives set methods to solve the optimization problem." It is argued in [15] that these exploited structures can be summarized as *chordal structure*. Therefore, the same structure exploiting software can be used to accelerate all computations for MPC. This holds irrespective of what MPC formulation is considered, and what type of optimization algorithm is used. In this paper we will in detail discuss robust MPC, which was not discussed in the above mentioned reference. We assume that the reader is familiar with the receding horizon strategy of MPC and we will only discuss the associated constrained finite-time optimal control problem. We will from now on refer to the associated problem as the MPC problem.

[1]Shervin Parvini Ahmadi is with Division of Automatic Control, Linkoping University, Sweden, e-mail: shervin.parvini.ahmadi@liu.se

[2]Anders Hansson is with Division of Automatic Control, Linkoping University, Sweden, e-mail: anders.g.hansson@liu.se

[3]Sina Khoshfetrat Pakazad is with C3 IoT, Redwood city, California, USA, e-mail: sina.pakazad@c3iot.com

The remaining part of the paper is organized as follows. We will in Section II state the robust MPC problem which is formulated using a scenario tree. In Section III we will see that a robust MPC problem is a special case of general Robust Quadratic Programs (RQPs). In Section IV we will discuss how chordal sparsity arises and how it can be utilized in convex optimization problems. In Section V we will discuss some technicalities regarding the implementation in Julia. We will test performance of the framework in Section VI. Finally, we will give some conclusions and discuss generalizations of our results in Section VII.

Notation

We denote with $\mathbf{R}$ the set of real numbers, with $\mathbf{R}^n$ the set of n-dimensional real-valued vectors and with $\mathbf{R}^{m\times n}$ the set of real-valued matrices with m rows and n columns. We denote by $\mathbf{N}$ the set of natural numbers and by $\mathbf{N}_n$ the subset $\{1,2,\ldots,n\}$ of $\mathbf{N}$. For a vector $x\in\mathbf{R}^n$ the matrix $X=\mathbf{diag}(x)$ is a diagonal matrix with the components of x on the diagonal. For two matrices A and B the matrix $A\oplus B$ is a block-diagonal matrix with A as the 1,1-block and B as the 2,2-block. For a symmetric matrix A the notation $A(\succeq)\succ 0$ is equivalent to A being positive (semi)-definite.

II. Robust MPC

There are many ways to define robust (linear) MPC problems. However, they all fall into the category

$$\min_{u(p)}\max_{p\in\mathcal{P}}\frac{1}{2}\sum_{k=0}^{N-1}\begin{bmatrix}x_k(p)\\u_k(p)\end{bmatrix}^T Q\begin{bmatrix}x_k(p)\\u_k(p)\end{bmatrix}+\frac{1}{2}x_N(p)^T S x_N(p)$$
$$\text{s.t. } x_{k+1}(p)=A(p)x_k(p)+B(p)u_k(p)+v_k(p),\ x_0=\bar{x}$$
$$Cx_k(p)+Du_k(p)\leq e_k$$

where $\mathcal{P}$ is some set. Here $A(p)\in\mathbf{R}^{n\times n}$ and $B(p)\in\mathbf{R}^{n\times m}$. We also assume that there are q inequality constraints for each k and that the dimensions of the other matrices and vectors are compatible with this. Above e_k is not a basis vector. One usually makes the assumption that p depends on k and that u_k only depends on values of p prior to k, the so-called non-anticipativity constraint. Since point-wise maximum over convex functions preserves convexity, it follows that the above problem also is convex. It should, however, be stressed that it is in general not tractable unless further assumptions are made on $\mathcal{P}$, such as e.g. finiteness. It is possible to also let C, D, Q, S, and e_k depend on p without destroying convexity.

We will consider a special important case that is obtained by letting the dynamics evolve as

$$x_{k+1}(\bar{p}_k)=A(p_k)x_k(\bar{p}_{k-1})+B(p_k)u_k(\bar{p}_{k-1})+v_k(p_k)$$

where $x_0 = \bar{x}$ and $\bar{p}_k = (p_0, p_1 \ldots, p_k)$, with $p_k \in \mathcal{P}_k$, where $\mathcal{P}_k$ are finite sets with cardinality M_k. We realize that the number of equality constraints grows exponentially with k in case the cardinality is independent of k. In order to get tractable problems one often let $M_k = 1$ for $k > N_r$ for some integer N_r. Then the problem can be written

$$\begin{aligned}\min_u \max_{\bar{p}_{N-1} \in \bar{\mathcal{P}}_{N-1}} & \frac{1}{2} \sum_{k=0}^{N-1} \begin{bmatrix} x_k(\bar{p}_{k-1}) \\ u_k(\bar{p}_{k-1}) \end{bmatrix}^T Q \begin{bmatrix} x_k(\bar{p}_{k-1}) \\ u_k(\bar{p}_{k-1}) \end{bmatrix} \\ & + \frac{1}{2} x_N(\bar{p}_{N-1})^T S x_N(\bar{p}_{N-1}) \\ \text{s.t. } & x_{k+1}(\bar{p}_k) = A(p_k)x_k(\bar{p}_{k-1}) + B(p_k)u_k(\bar{p}_{k-1}) + v_k(p_k) \\ & Cx_k(\bar{p}_{k-1}) + Du_k(\bar{p}_{k-1}) \le e_k\end{aligned}$$

where $x_0 = \bar{x}$, $u = (u_0, u_1(\bar{p}_0), \ldots, u_{N-1}(\bar{p}_{N-1}))$, and where $\bar{\mathcal{P}}_k = \mathcal{P}_0 \times \mathcal{P}_1 \times \cdots \times \mathcal{P}_k$.

We will now reformulate the problem into an equivalent problem with more variables and constraints. We let all states and control signals depend on $p = \bar{p}_{N_r} \in \mathcal{P} = \bar{\mathcal{P}}_{N_r}$ with cardinality $M = M_0 \times M_1 \times \cdots \times M_{N_r}$, i.e. we introduce M independent scenarios which we constrain using so-called non-anticipativity constraints:

$$\begin{aligned}\min_u \max_{p \in \mathcal{P}} & \frac{1}{2} \sum_{k=0}^{N-1} \begin{bmatrix} \bar{x}_k(p) \\ \bar{u}_k(p) \end{bmatrix}^T Q \begin{bmatrix} \bar{x}_k(p) \\ \bar{u}_k(p) \end{bmatrix} + \frac{1}{2} x_N(p)^T S x_N(p) \\ \text{s.t. } & \bar{x}_{k+1}(p) = A(p_k)\bar{x}_k(p) + B(p_k)\bar{u}_k(p) + v_k(p_k) \\ & Cx_k(p) + Du_k(p) \le e_k\end{aligned}$$

where $x_0(p) = \bar{x}$,

$$\begin{aligned}&\bar{u}_k(p_0, \ldots, p_k, p^1_{k+1}, \ldots, p^1_{N_r}) = \\ &\bar{u}_k(p_0, \ldots, p_k, p^2_{k+1}, \ldots, p^2_{N_r})\end{aligned}$$

for all $p^1_{k+1}, \ldots, p^1_{N_r}$; $p^2_{k+1}, \ldots, p^2_{N_r}$, and where

$$u = (\bar{u}_0(p), \ldots, \bar{u}_{N-1}(p))$$

We further define an enumeration of all scenarios using an index $j \in \{1, 2, \ldots, M\}$ which make it possible to define the equivalent problem

$$\min_u \max_{1 \le j \le M} \frac{1}{2} \sum_{k=0}^{N-1} \begin{bmatrix} x^j_k \\ u^j_k \end{bmatrix}^T Q \begin{bmatrix} x^j_k \\ u^j_k \end{bmatrix} + \frac{1}{2}(x^j_N)^T S x^j_N \tag{1}$$

$$\text{s.t. } x^j_{k+1} = A^j_k x^j_k + B^j_k u^j_k + v^j_k, \quad x^j_0 = \bar{x} \tag{2}$$

$$Cx^j_k + Du^j_k \le e_k \tag{3}$$

$$\bar{C}u = 0 \tag{4}$$

where $u = (u^1, u^2, \ldots, u^M)$ with $u^j = (u^j_0, u^j_1, \ldots, u^j_{N-1})$, and where

$$\bar{C} = \begin{bmatrix} C_{1,2} & -C_{1,2} & & & \\ & C_{2,3} & -C_{2,3} & & \\ & & \ddots & \ddots & \\ & & & C_{M-1,M} & -C_{M-1,M} \end{bmatrix}$$

where

$$C_{j,j+1} = \begin{bmatrix} I & 0 \end{bmatrix}$$

and where I is an identity matrix of dimension m times the number of time instances that scenarios j and $j+1$ have a control signal in common. Notice that several of the matrices A^j_k, B^j_k and v^j_k are also constrained, but that we do not have to write that out, since they are not optimization variables. Several other authors have investigated how the structure stemming from scenario trees can be exploited in a stochastic setting, e.g. [12], [22], [20], [10].

III. Robust QP

The robust MPC problem is a special case of a so-called Robust Quadratic Program (RQP). Consider the RQP

$$\min_{\tau,t,z} \tau \tag{5}$$

$$\text{s.t. } \frac{1}{2}\left(z^j_0\right)^T \mathcal{Q}^j_0 z^j_0 + t^j_1 \le \tau, \quad j \in \mathbf{N}_M \tag{6}$$

$$\frac{1}{2}\left(z^j_k\right)^T \mathcal{Q}^j_k z^j_k + t^j_{k+1} \le t^j_k, \quad j \in \mathbf{N}_M,\ k \in \mathbf{N}_{N-1} \tag{7}$$

$$\frac{1}{2}\left(z^j_N\right)^T \mathcal{Q}^j_N z^j_N \le t^j_N, \quad j \in \mathbf{N}_M, \tag{8}$$

$$\mathcal{A}z = b \tag{9}$$

$$\mathcal{C}z \le d \tag{10}$$

where $\mathcal{Q}^j_k \succeq 0$, i.e. positive semidefinite, where $\mathcal{A}$ has full row rank, and where the matrices and vectors are of compatible dimensions. Here $z = (z^1, \ldots, z^M)$ with $z^j = (z^j_0, \ldots, z^j_N)$, and the inequality in (10) is component-wise inequality.

The problem in (1–4) is equivalent with the problem in (5–10). To see this we let $Q^j_k = Q$ and $z^j_k = (x^j_k, u^j_k)$ for $k = 0, \ldots, N-1$, and $Q^j_N = S$ and $z^j_N = x^j_N$. We also let

$$\begin{aligned}b^j &= (\bar{x}, v^j_0, v^j_1, \ldots, v^j_{N-1}) \\ e^j &= (e_0, e_1, \ldots, e_{N-1})\end{aligned}$$

and

$$\mathcal{A}^j = \begin{bmatrix} I & & & & & & \\ -A^j_0 & -B^j_0 & I & & & & \\ & & -A^j_1 & -B^j_1 & I & & \\ & & & & \ddots & & \\ & & & & & -A^j_{N-1} & -B^j_{N-1} & I \end{bmatrix}$$

$$\mathcal{D}^j = \begin{bmatrix} C & D \end{bmatrix} \oplus \begin{bmatrix} C & D \end{bmatrix} \oplus \ldots \oplus \begin{bmatrix} C & D \end{bmatrix}$$

Finally we let $\mathcal{A} = \oplus_{j=1}^M \mathcal{A}^j \oplus \tilde{C}$, $\mathcal{D} = \oplus_{j=1}^M \mathcal{D}^j$, $b = (b^1, \ldots, b^M)$ and $e = (e^1, \ldots, e^M)$. Here $\tilde{C}$ is a matrix obtained from $\bar{C}$ by combining its columns with zero columns such that the non-anticipatively constraint holds.

IV. Chordal Sparsity and Convex Optimization

Consider the following convex optimization problem

$$\min_x \quad F_1(x) + \cdots + F_N(x), \tag{11}$$

where $F_i : \mathbf{R}^n \to \mathbf{R}$ for all $i = 1, \ldots, N$. We assume that each function F_i is only dependent on a small subset of elements of x. Let us denote the ordered set of these indexes by $J_i \subseteq \mathbf{N}_n$. We can then rewrite the problem in (11), as

$$\min_x \quad \bar{F}_1(E_{J_1}x) + \cdots + \bar{F}_N(E_{J_N}x), \tag{12}$$

where E_{J_i} is a 0–1 matrix that is obtained from an identity matrix of order n by deleting the rows indexed by $\mathbf{N}_n \setminus J_i$. The functions $\bar{F}_i : \mathbf{R}^{|J_i|} \to \mathbf{R}$ are lower dimensional descriptions of F_is such that $F_i(x) = \bar{F}_i(E_{J_i}x)$ for all $x \in \mathbf{R}^n$ and $i \in \mathbf{N}_N$. For details on how this structure can be exploited using message passing the reader is referred to [18].

A brief summary is that we may define a so-called sparsity graph for the above optimization problem with n nodes and edges between two nodes j and k if x_j and x_k appear in the same term $\bar{F}_i$. We assume that this graph is chordal, i.e. every cycle of length four our more has a chord.[1] The maximal complete subgraphs of a graph are called its cliques. If the original graph is chordal then there exists a tree of the cliques called the clique tree which is such that it enjoys the clique intersection property. This property is that all elements in the intersection of two cliques C_i and C_j should be elements of the cliques on the path between the cliques C_i and C_j. It is then possible to use the clique tree as a computational tree where we non-uniquely assign terms of the objective function to each clique in such a way that all the variables of the term in the function are elements of the clique. After this we may solve the optimization problem distributively over the clique tree by starting with leafs and for each leaf solve a parametric optimization problem, where we optimize with the respect to the variables of the leaf problem which are not variables of the parent of the leaf in the clique tree. The optimization should be done parametrically with respect to all the variables that are shared with the parent. After this the optimal objective function value of the leaf can be expressed as a function of the variables that are shared with the parent. This function is sent to the parent and added to its objective function term. The leaf has been pruned away, and then the optimization can continue with the parent assuming all its children has also carried out their local optimizations. Eventually we reach the root of the tree, where the remaining variables are optimized. Then we can finally go down the tree and recover all optimal variables. This is based on the fact that we have stored the parametric optimal solutions in the nodes of the clique tree.

Notice that each function $\bar{F}_i$ in (12) can contain equality and inequality constraints using indicator functions, therefore the RQP in (5-10) is a special case of the problem in (12).

We now study a robust MPC case when $N_r = 1$, $M_0 = M_1 = 2$ and $N = 4$ in more detail. Then $M = 4$. The corresponding sparsity graph for the equivalent RQP problem as in (5-10) is shown in Figure 1. The reason for using dashed line is to make the edges more obvious, otherwise there is no difference between dashed and solid edges. Moreover we do not show all the edges related to the nodes x_0^j and u_0^j since this would clutter the graph. Actually all of the eight variables x_0^j and u_0^j have edges connecting them. We realize that the sparisty graph is not chordal. A chordal embedding is obtained by adding edges such that $C_0 = \{x_0^1, u_0^1, x_0^2, u_0^2, x_0^3, u_0^3, x_0^4, u_0^4, t_1^1, t_1^2, t_1^3, t_1^4, \tau\}$, $C_1^1 = \{x_1^1, u_1^1, x_2^1, x_1^2, u_1^2, x_2^2, t_2^1, t_2^2\}$ and $C_1^3 = \{x_1^3, u_1^3, x_2^3, x_1^4, u_1^4, x_2^4, t_2^3, t_2^4\}$ are complete graphs. A clique tree for the chordal embedding is illustrated in Figure 2, where $C_{k+1}^j = \{x_k^j, u_k^j, x_{k+1}^j, t_{k+1}^j, t_{k+2}^j\}$ with $k \in \mathbf{N}_{N-2}$. To assign functions to nodes in the clique tree, let us first define the following sets for $j \in \mathbf{N}_M$

$$
\begin{aligned}
\mathscr{C}_0^j &= \{(x_0^j, u_0^j, \tau, t_1^j) : \frac{1}{2}\begin{bmatrix} x_0^j \\ u_0^j \end{bmatrix}^T Q \begin{bmatrix} x_0^j \\ u_0^j \end{bmatrix} + t_1^j \le \tau\} \\
\mathscr{C}_k^j &= \{(x_k^j, u_k^j, t_k^j, t_{k+1}^j) : \\
&\qquad \frac{1}{2}\begin{bmatrix} x_k^j \\ u_k^j \end{bmatrix}^T Q \begin{bmatrix} x_k^j \\ u_k^j \end{bmatrix} + t_{k+1}^j \le t_k^j\}, k \in \mathbf{N}_{N-1} \\
\mathscr{C}_N^j &= \{(x_N^j, t_N^j) : \frac{1}{2}(x_N^j)^T S x_N^j \le t_N^j\} \\
\mathscr{D}_k^j &= \{(x_k^j, u_k^j) : Cx_k^j + Du_k^j \le e\}, \qquad k \in \mathbf{N}_{N-1} \\
\mathscr{E}_k^j &= \{(x_k^j, u_k^j, x_{k+1}^j) : \\
&\qquad x_{k+1}^j = A_k^j x_k^j + B_k^j u_k^j + v_k^j\}, \quad k \in \mathbf{N}_{N-1} \\
\mathscr{F}^j &= \{x_0^j : x_0^j = \bar{x}\}
\end{aligned}
$$

and the sets

$$
\begin{aligned}
\mathscr{G}^j &= \{(u_0^j, u_0^{j+1}) : u_0^j = u_0^{j+1}\} \quad j \in \mathbf{N}_{M-1} \\
\mathscr{H}^j &= \{(u_1^j, u_1^{j+1}) : u_1^j = u_1^{j+1}\} \quad j \in \{1, 3\}
\end{aligned}
$$

The assignments of functions then become for C_0

$$
\begin{aligned}
&\tau + \sum_{j=1}^{M} \{\mathscr{I}_{\mathscr{C}_0^j}(x_0^j, u_0^j, \tau, t_1^j) + \mathscr{I}_{\mathscr{D}_0^j}(x_0^j, u_0^j) + \\
&\mathscr{I}_{\mathscr{E}_0^j}(x_0^j, u_0^j, x_1^j) + \mathscr{I}_{F^j}(x_0^j)\} + \sum_{j=1}^{M-1} \mathscr{I}_{G^j}(u_0^j, u_0^{j+1})
\end{aligned}
$$

for C_1^1

$$
\begin{aligned}
&\sum_{j=1}^{\frac{M}{2}} \{\mathscr{I}_{\mathscr{C}_1^j}(x_1^j, u_1^j, t_1^j, t_2^j) + \mathscr{I}_{\mathscr{D}_1^j}(x_1^j, u_1^j) + \\
&\mathscr{I}_{\mathscr{E}_1^j}(x_1^j, u_1^j, x_2^j)\} + \mathscr{I}_{\mathscr{H}^j}(u_1^1, u_1^2)
\end{aligned}
$$

for C_1^3

$$
\begin{aligned}
&\sum_{j=\frac{M}{2}+1}^{M} \{\mathscr{I}_{\mathscr{C}_1^j}(x_1^j, u_1^j, t_1^j, t_2^j) + \mathscr{I}_{\mathscr{D}_1^j}(x_1^j, u_1^j) + \\
&\mathscr{I}_{\mathscr{E}_1^j}(x_1^j, u_1^j, x_2^j)\} + \mathscr{I}_{\mathscr{H}^j}(u_1^3, u_1^4)
\end{aligned}
$$

for C_{k+1}^j, where $j \in \mathbf{N}_M,\ k \in \mathbf{N}_{N-3}$

$$
\begin{aligned}
&\mathscr{I}_{\mathscr{C}_{k+1}^j}(x_{k+1}^j, u_{k+1}^j, t_{k+1}^j, t_{k+2}^j) + \mathscr{I}_{\mathscr{D}_{k+1}^j}(x_{k+1}^j, u_{k+1}^j) \\
&+ \mathscr{I}_{\mathscr{E}_{k+1}^j}(x_{k+1}^j, u_{k+1}^j, x_{k+2}^j)
\end{aligned}
$$

and for C_N^j, where $j \in \mathbf{N}_M$

$$
\begin{aligned}
&\mathscr{I}_{\mathscr{C}_{N-1}^j}(x_{N-1}^j, u_{N-1}^j, t_{N-1}^j, t_N^j) + \mathscr{I}_{\mathscr{D}_{N-1}^j}(x_{N-1}^j, u_{N-1}^j) \\
&+ \mathscr{I}_{\mathscr{E}_{N-1}^j}(x_{N-1}^j, u_{N-1}^j, x_N^j) + \mathscr{I}_{\mathscr{C}_N^j}(x_N^j, u_N^j)
\end{aligned}
$$

[1] In case the graph is not chordal we make a chordal embedding, i.e. we add edges to the graph until it becomes chordal. This corresponds to saying that some of the $\bar{F}_i$ depend on variables that they do not depend on.

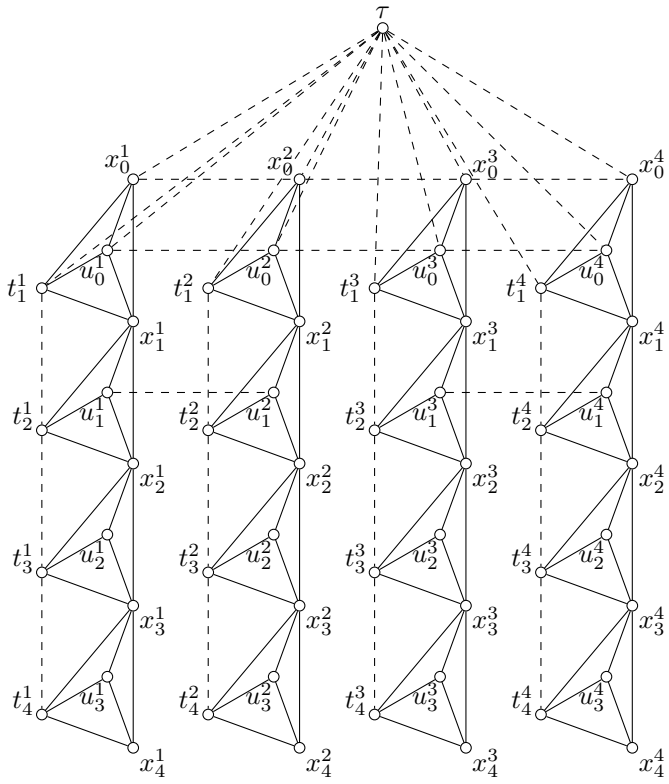

Fig. 1: Sparsity graph for the problem in (5-10) for $N = 4$ and $M = 4$.

where $\mathscr{I}_{\mathscr{X}}$ is the indicator function for the set $\mathscr{X}$. Notice that $A_0^1 = A_0^2$, $A_0^3 = A_0^4$, $B_0^1 = B_0^2$, $B_0^3 = B_0^4$, $v_0^1 = v_0^2$ and $v_0^3 = v_0^4$ due to non-anticipativity constraints. Now, solving the problem in (5–10) using a primal-dual interior-point method can be done distributedly using the clique tree, as described in [18]. It is worth mentioning that it is possible to introduce even more parallelism for computations on each branch of the tree as described in [15].

V. Implementation

The distributed primal-dual interior-point algorithm proposed in [18] is implemented in Julia, [5] in such a way that we can take advantage of parallel computations using multiple processors. This is an extension of the work presented in [25] and [24]. The problem formulation considered in [25] is a coupled QP with affine equality and affine inequality constraints, whereas in this paper we consider coupled convex problems with affine equality and convex inequality constraints. It should be stressed that the proposed algorithm in [18] requires exact gradients and hessians of the objective function and the inequality constraints. Therefore, functions have to be provided that evaluate these quantities. The general idea in the implementation is as follows. A tree is implemented as a data structure using linked lists. Hence, a node in the tree has information regarding the assigned subproblem, information regarding the processor IDs in which its parent or child nodes are defined and finally a list of references to children and parent nodes.

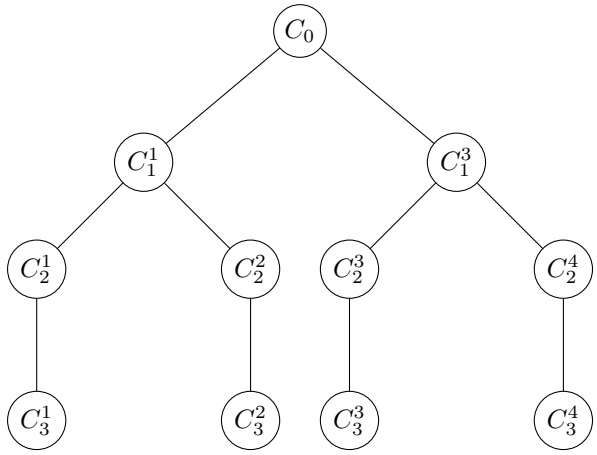

Fig. 2: Clique trees for the problem in (5-10) for $N = 4$ and $M = 4$.

The algorithm consists of different phases which are conducted at each iteration. There are, in particular, search direction, step size and termination criteria calculation phases. These calculations rely on performing message passing upwards and downwards through the tree. Therefore, a recursive function is defined for each phases of the algorithm such that starting from the root node, the whole tree is traversed once and a particular function is called while visiting a node. Among various approaches for tree traversal, the so-called pre-order traversal method is employed. In this method, first the root node is visited, then a pre-order traversal of the most left subtree is done recursively which is followd by a recursive pre-order traversal of the second most left subtree and so on. For instance, a pre-order traversal for the tree in Figure (2) is $[C_0, C_1^1, C_2^1, C_3^1, C_2^2, C_3^2, C_1^3, C_2^3, C_3^3, C_2^4, C_3^4]$. Notice that when a node is visited, return values of the functions can be interpreted as a message from child node to parent node or vice versa. As it is explained in [25], the way that we benefit from parallelism is that we first store different subtrees of the main tree in different processors. Then for any phase of the algorithm, when a node is visited and a particular function is called, first it is checked if there are children nodes which are defined in other processors, and if so, then for all of those children nodes, the same function is called and without waiting for the return values, i.e. messages, the same function is called on the children nodes which are defined in the same processor, if there are any. Finally, the messages from children nodes defined in other processors are fetched as soon as their computations are finished.

VI. Numerical Experiments

For the problem in (1–4), we generate randomly 20 linear systems with $n = 5$, $m = 3$ and $q = 1$. We consider neither unstable systems nor systems whose Controllability Gramian is larger than 100. We also randomly generate Q and S such that they are positive semidefinite. Furthermore, the initial starting point $\bar{x}$ is chosen so that it is feasible with respect to inequality constraints, as it is required for the algorithm

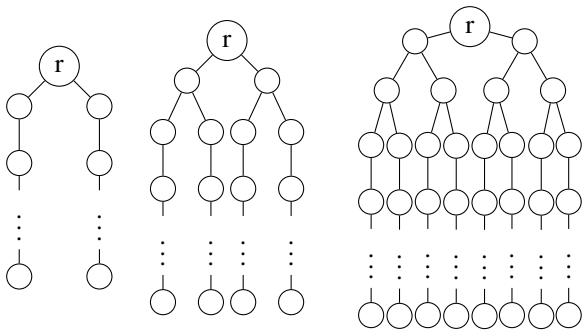

Fig. 3: Clique trees for the problem in (1–4) for 2, 4 and 8 scenarios (M) from left to right.

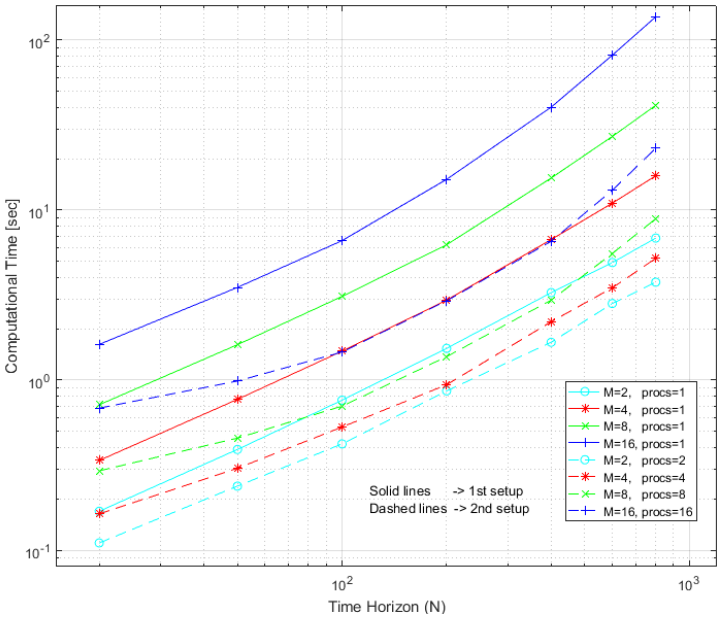

Fig. 4: The average computational time spent at each iteration of the algorithm for the problems with 2, 4, 8 and 16 scenarios over 20 randomly generated systems using the two setups.

in [18]. We consider problems with 2, 4, 8 and 16 scenarios (M) and 7 different time horizons (N) between 20 and 800. The clique trees for the problems with 2, 4 and 8 scenarios is illustrated in Figure (3). We evaluate the algorithm in [18] in two separate setups. In the first setup, we store the tree in a single processor and carry out the computations sequentially. Hence we do not take advantage of parallel computations. In the second setup, we store different subtrees of the tree in separate processor and carry out the computations such that we can benefit from parallel computations. There is not a unique way for selection of subtrees. However, in order to fully benefit from parallel computations, we select the subtrees using the following rule. Let us say that C is the ordered set of nodes when a pre-order traversal method is performed. Then the first subtree will be a subset of C which contains the first element which is the root node, until the first element which does not have a child node. The second subtree will be a subset of C starting from the element which is right after the first element which does not have a child node, until the second element of C which does not have a child node. We repeat this procedure until we split the tree in M subtrees and then we store each subtree in a separate processor. For example using this rule, the first, second, third and fourth subtrees of the tree in Figure (2) are $\{C_0, C_1^1, C_2^1, C_3^1\}$, $\{C_2^2, C_3^2\}$, $\{C_1^3, C_2^3, C_3^3\}$ and $\{C_2^4, C_3^4\}$, respectively.

We run the algorithm in Julia 0.6.2 which is installed on an Intel Xeon X5675 @3.07 GHz processor with 24 cores and a Linux operating system. The α and β parameters for step size calculations are set to 0.01 and 0.8. The initial iterates for dual variables are chosen to be $\lambda(0) = v(0) = \mathbf{1}$. There are three parameters for checking the termination of the algorithm at each iteration. Two of them are primal and dual residual norms. It is observed that for all of the examples, these parameters are within the tolerance of 10^{-6}, after a few iterations. The third parameter is the so-called surrogate duality gap. It is observed that the reduction rate of the surrogate duality gap might be very low for some examples which in turn lead to too many iterations if we want to have a small tolerance. This is specially the case for big-size problems. Therefore, in order to deal with this issue, we set the stopping criteria for surrogate duality gap to 0.1 and we terminate the algorithm if there are more than 150 iterations. It turns out that, out of 1120 runs of the algorithm, 204 of them required more than 150 iterations in order to meet the criteria for surrogate duality gap. We believe that the problems which require many iterations for convergence are ill-conditioned problems.

We illustrate the average computational time spent at each iteration of the algorithm for the problems with 2, 4, 8 and 16 scenarios over 20 randomly generated systems using the two above-mentioned setups in Figure (4). As can be seen, the computations are accelerated for all the problems by using multiple processors. Nevertheless, it is more advantageous to use multiple processors for big-size problems, especially the ones which have small decrease rate of surrogate duality gap. Notice that the speed-up obtained is not equal to the number of processors used. There are several reasons for this. One is communication overhead. Another is difference in workload due to imperfect balancing. The third is that while we run the algorithm on multiple processors, there is also another type of parallelization taking place which is imposed by the programming language. In fact, the linear algebra operations that we use are carried out using multiple processors to increase efficiency. This, in turn, affects the computational time of the algorithm. See [21] for more details on parallelization for linear algebra.

VII. Conclusions

We have in this paper shown how it is possible to make use of the inherent chordal structure of a robust MPC problem in order to exploit IP methods that make use of any chordal structure to distribute its computations over several compu-

tational agents that can work in parallel. We have evaluated the efficiency of the framework in Julia, using numerical experiments. We argue that this level of abstraction, i.e. chordality, is more appropriate than a more detailed level of abstraction where one tries to see Riccati recursion structure. The reason for this is that chordality is a more general concept. It also appears when the dynamic equations are obtained from spatial discretization of partial differential equations. Hence we believe that this structure can be utilized using the same formalism as we have presented above. How to carry out these extensions is left for future work. We will also investigate how to deal with ill-conditioned problems and look into alternative primal-dual methods which might cope better with ill-conditioned problems.

ACKNOWLEDGMENT

This research has been supported by ELLIIT and by the Wallenberg Artificial Intelligence, Autonomous Systems and Software Program (WASP) funded by Knut and Alice Wallenberg Foundation, which is gratefully acknowledged. The authors are also grateful for discussion with Dimitris Kouzoupis.

REFERENCES

[1] M. Åkerblad and A. Hansson. Efficient solution of second order cone program for model predictive control. *International Journal of Control*, 77(1):55–77, January 2004.

[2] A. Alessio and A. Bemporad. *A Survey on Explicit Model Predictive Control*, pages 345–369. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[3] E. Arnold and H. Puta. An SQP-type solution method for constrained discrete-time optimal control problems. In R. Bulirsch and D. Kraft, editors, *Computational Optimal Control*, volume 115 of *International Series of Numerical Mathematics*, pages 127–136. Birkhäuser Verlag, Basel, 1994.

[4] D. Axehill, L. Vandenberghe, and A. Hansson. Convex relaxations for mixed integer predictive control. *Automatica*, 46:1540–1545, 2010.

[5] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah. Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98, 2017.

[6] C. R. Cutler and B. L. Ramaker. Dynamic matrix control—a computer control algorithm. In *Proceedings of the AIChE National Meeting*, Huston, Texas, 1979.

[7] M. Diehl, H. J. Ferreau, and N. Haverbeke. *Efficient Numerical Methods for Nonlinear MPC and Moving Horizon Estimation*, pages 391–417. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[8] A. Domahidi, A. U. Zgraggen, M. N. Zeilinger, M. Morari, and C. N. Jones. Efficient interior point methods for multistage problems arising in receding horizon control. In *51st IEEE Conference on Decision and Control*, pages 668–674, Maui, USA, 2012.

[9] G. Frison. *Algorithms and Methods for Fast Model Predictive Control*. Phd thesis, Technical University of Denmark, 2015.

[10] G. Frison, D. Kouzoupis, M. Diehl, and J. B. Jorgensen. A high-erformance Riccati based solver for tree-structured quadratic programs. In *Proceedings of the 20th IFAC World Congress*, pages 14964–14970, 2017.

[11] T. Glad and H. Jonson. A method for state and control constrained linear quadratic control problems. In *Proceedings of the 9th IFAC World Congress*, Budapest, Hungary, 1984.

[12] J. Gondzio and A. Grothey. Paralell interior-point solver for structured quadratic programs: Applications to financial planning problems. *Ann. Oper. Res.*, 152:319–339, 2007.

[13] V. Gopal and L. T. Biegler. Large scale inequality constrained optimization and control. *IEEE Control Systems Magazine*, 18(6):59–68, 1998.

[14] A. Hansson. A primal-dual interior-point method for robust optimal control of linear discrete-time systems. *IEEE Transactions on Automatic Control*, 45(9):1639–1655, 2000.

[15] A. Hansson and S. K. Pakazad. Exploiting chordality in optimization algorithms for model predictive control. *arXiv:1711.10254*, 2017.

[16] J. L. Jerez, E. C. Kerrigan, and G. A. Constantinides. A sparse condensed QP formulation for control of LTH systems. *Automatica*, 48:999–1002, 2012.

[17] Jorgensen. *Moving Horizon Estimation and Control*. PhD thesis, Technical University of Denmark, 2004.

[18] S. Khoshfetrat Pakazad, A. Hansson, M. S. Andersen, and I. Nielsen. Distributed primal–dual interior-point methods for solving tree-structured coupled convex problems using message-passing. *Optimization Methods and Software*, pages 1–35, 2016.

[19] E. Klintberg. *Structure Exploiting Optimization Methods for Model Predictive Control*. Phd thesis, Chalmers University of Technology, 2017.

[20] C. Leidereiter, A. Potschka, and H. G. Bock. Dual decomposition of QPs in scenario tree NMPC. In *Proceedings of the 2015 European Control Conference*, pages 1608–1613, 2015.

[21] J.-Y. L'Excellent. *Multifrontal methods: parallelism, memory usage and numerical aspects*. PhD thesis, Ecole normale supérieure de lyon-ENS LYON, 2012.

[22] R. Marti, S. Lucia, D. Sarabia, R. Paulen, S. Engell, and C. de Prada. An efficient distributed algorithm for multi-stage robust nonlinear predictive control. In *Proceedings of the 2015 European Control Conference*, pages 2664–2669, 2015.

[23] I. Nielsen. *Structure-Exploiting Numerical Algorithms for Optimal Control*. Phd thesis, Linköping University, 2017.

[24] S. Parvini Ahmadi. Distprimdualintpoint. `https://github.com/ShervinParvini/DistPrimDualIntPoint`, 2018.

[25] S. Parvini Ahmadi and A. Hansson. Parallel exploitation for tree-structured coupled quadratic programming in julia. In *22nd International Conference on System Theory, Control and Computing*, page Accepted for presentation, Sinaia, Romania, Oct. 2018.

[26] S. J. Qin and T. A. Badgwell. A survey of industrial model predictive control technology. *Control Engineering Practice*, 11:722–764, 2003.

[27] C. V. Rao, S. J. Wright, and J. B. Rawlings. Application of interior-point methods to model predictive control. Preprint ANL/MCS-P664-0597, Mathematics and Computer Science Division, Argonne National Laboratory, May 1997.

[28] M. C. Steinbach. A structured interior point SQP method for nonlinear optimal control problems. In R. Bulirsch and D. Kraft, editors, *Computational Optimal Control*, volume 115 of *International Series of Numerical Mathematics*, pages 213–222. Birkhäuser Verlag, Basel, 1994.

[29] L. Vandenberghe, S. Boyd, and M. Nouralishahi. Robust linear programming and optimal control. Internal report, Department of Electrical Engineering, University of California, Los Angeles, 2001.

[30] Y. Wang and S. Boyd. Fast model predictive control using online optimization. *IEEE Transactions on Control Systems Technology*, 18:267–278, 2010.

[31] S. J. Wright. Interior-point methods for optimal control of discrete-time systems. *J. Optim. Theory Appls.*, 77:161–187, 1993.

[32] S. J. Wright. Applying new optimization algorithms to model predictive control. *Chemical Process Control-V*, 1996.