A Primal-Dual Active-Set Method for Distributed Model Predictive Control

Sarah Koehler[1], Claus Danielson[2], Francesco Borrelli[1]

***Abstract*— A primal-dual active-set method is used for solving Model Predictive Control for large-scale systems with linear dynamics, additive disturbance and box constraints. We present a novel decentralized implementation of the primal-dual active-set method. The proposed algorithm is compared to dual decomposition methods. Theoretical and experimental results show the effectiveness of the proposed approach for large scale systems with high communication delays. The application to buildings control systems is thoroughly investigated.**

I. Introduction

Active-set methods are a class of constrained optimization algorithms. Active-set algorithms work by iteratively finding the subset of inequality constraints for which equality holds at the optimum. This set is called the optimal active-set. In traditional active-set methods, only one constraint is added or removed from the active-set during each iteration. As a result, traditional active-set methods can require a large number of iterations to find the optimal active-set. In this paper we examine a primal-dual active-set method for strictly convex quadratic programs that was initially proposed by Hintermüller [1] and expanded upon by Curtis [2]. In this algorithm, multiple constraints are added to or removed from the active-set during each iteration of the algorithm. As a result, the algorithm often converges in very few iterations. It has been shown that the algorithm has superlinear convergence [3].

In this paper, we apply this primal-dual active-set method to Model Predictive Control (MPC) problems. In particular, we consider a class of MPC problems that have additive disturbance, box constraints, and linear dynamics. The primal-dual active-set algorithm is suitable for this class of MPC problem.

This algorithm is particularly well suited for decentralized implementation in systems with high communication costs. Deploying model predictive controllers on large-scale distributed systems is challenging since it requires solving large-scale optimization problems in real-time on embedded hardware. One popular solution to this problem is to distribute the computations performed by the optimization algorithm amongst distributed platforms. Methods include matrix splitting [4], dual decomposition [5], and a distributed inexact Newton method [6]. The disadvantage of these solutions is high communication costs. These algorithms require the communication of a large number of variables during each iteration. Additionally, it may require many iterations to converge, requiring substantial communication to find the solution. It is well known in the parallel computing community that communication between processors is dramatically slower than computation on a single chip [7].

One example of a system with prohibitively high communication costs is heating, ventilation, and air-conditioning (HVAC) of modern commercial buildings. Commercial buildings contain many thermal zones, each having dedicated computing resources. However communicating between these computing resources can be expensive. Using standard building protocols such as BACnet and ARCnet, we have found communication delays ranging 0.03 to 0.4 seconds per floating point number. A proposed solution for distributed optimization of building control using MPC is dual decomposition [8]. However, the effect of these communication delays have been previously ignored in studies of dual decomposition.

In this paper we present a novel decentralized implementation of the primal-dual active-set method. This method is naturally decomposable for decentralized optimization for decentralized systems having dynamics coupled through the control inputs. This class of problems includes buildings, network flow [9], or multi agent problems [10]. We compare the computational and communication complexities of the primal-dual active-set algorithm with dual decomposition.

Our theoretical analysis shows that dual decomposition has lower computational costs but higher communication costs per iteration. In addition, since dual decomposition has sub-linear convergence it requires more iterations, and therefore more communication to converge. On the other hand, the primal-dual active-set algorithm requires less communication per iteration and fewer iterations to converge. Therefore, it is suited for distributed control with high communication latency. We confirm our theoretical findings with numerical tests which compare these algorithms. We find that the smaller number of iterations and number of variables communicated per iteration significantly reduces the computation time. As a result, the primal-dual active-set method outperforms dual decomposition in the proposed systems.

The paper is organized as follows. First, we introduce the primal-dual active-set method proposed by Hintermüller [1]. We show how this algorithm can be applied to Model Predictive Control problems. Next we describe our decentralized implementation of the primal-dual active-set method. The application of thermal building control is discussed. Finally we present theoretical and experimental results comparing the two distributed algorithms.

[1]S. Koehler and F. Borrelli are with the Department of Mechanical Engineering, University of California, Berkeley, CA 94720-1740, USA {skoehler, fborrelli}@berkeley.edu

[2] C. Danielson conducted this work while at the University of California, Berkeley. He is currently at Mitsubishi Electric Research Lab.

II. Primal-Dual Active-Set Method

In this section we describe the primal-dual active-set algorithm presented in [1] and [2]. This algorithm can be reformulated as a semi-smooth Newton method with local quadratic convergence.

A. Active-Set Algorithm

Consider the following strictly convex quadratic program,

$$\min_{z} \quad \tfrac{1}{2} z^T H z \tag{1a}$$
$$s.t. \quad Cz = d \tag{1b}$$
$$\underline{z} <= z <= \overline{z} \tag{1c}$$

where $z \in \mathbb{R}^p$, $H = H^T \succ 0$, $C \in \mathbb{R}^{q \times p}$ is full row-rank, $d \in \mathbb{R}^q$, $\underline{z} < \overline{z} \in \bar{\mathbb{R}}^p$, and $\bar{\mathbb{R}} = \mathbb{R} \cup \{\pm\infty\}$ is the extended real-line. We note that we are omitting a linear term in the cost (1a) which is used in [1] and [2].

Let $\nu \in \mathbb{R}^q$ be the dual variables corresponding to the equality constraints (1b) and let $\bar{\lambda} \in \mathbb{R}^p$ and $\underline{\lambda} \in \mathbb{R}^p$ be the corresponding dual variables for the upper and lower bound constraints (1c), respectively. The Karush-Kuhn-Tucker (KKT) necessary and sufficient conditions for optimality for the quadratic program (1) are

$$Hz + C^T\nu + \bar{\lambda} - \underline{\lambda} = 0 \tag{2a}$$
$$Cz - d = 0 \tag{2b}$$
$$\min(\overline{z} - z, \bar{\lambda}) = 0 \tag{2c}$$
$$\min(z - \underline{z}, \underline{\lambda}) = 0. \tag{2d}$$

Let $\mathcal{J} = \{1, \ldots, p\}$ be the index set for the elements of the vector $z \in \mathbb{R}^p$. For a primal variable $z \in [\underline{z}, \overline{z}]$, we define the following disjoint subsets of $\mathcal{J}$

$$\overline{\mathcal{A}} = \{j \in \mathcal{J} : z_j = \overline{z}_j\} \tag{3a}$$
$$\underline{\mathcal{A}} = \left\{j \in \mathcal{J} : z_j = \underline{z}_j\right\}. \tag{3b}$$

The set $\overline{\mathcal{A}}$ is the set of components of z that are active at their upper bound $\overline{z}$ and the set $\underline{\mathcal{A}}$ is the set of elements of z that are active at their lower bound $\underline{z}$. We define $\mathcal{A} = \underline{\mathcal{A}} \cup \overline{\mathcal{A}}$. The set $\mathcal{I} = \mathcal{J} \setminus \mathcal{A}$ is the set of elements of z that are not active. We will denote by $z_{\mathcal{A}}$ the elements z_j of z for $j \in \mathcal{A}$. Similarly, $z_{\mathcal{I}}$ is the elements z_j of z for $j \in \mathcal{I}$.

The primal-dual active-set algorithm uses the active-sets $\underline{\mathcal{A}}$ and $\overline{\mathcal{A}}$ to update the primal z and dual variables $\nu, \underline{\lambda}, \bar{\lambda}$. The primal and dual variables are then used to update the active-sets.

The algorithm is initialized with an active-set partition $\overline{\mathcal{A}}$, $\underline{\mathcal{A}}$, and $\mathcal{I}$. The active elements, $z_{\mathcal{A}}$, are set equal to their active bound and $z_{\mathcal{I}}$ is left as a free variable. The active-set partition must satisfy feasibility (2b), i.e. there must exist some $z_{\mathcal{I}}$ such that the set $\{z_{\mathcal{I}} | Cz = C_{\mathcal{A}} z_{\mathcal{A}} + C_{\mathcal{I}} z_{\mathcal{I}} = d\}$ is not empty.

The first step of the algorithm is to find primal variables $z_{\mathcal{I}}$ and equality dual variables ν that satisfy the stationarity (2a) and feasibility (2b) conditions

$$\begin{bmatrix} H_{\mathcal{I},\mathcal{I}} & C_{\mathcal{I}}^T \\ C_{\mathcal{I}} & 0 \end{bmatrix} \begin{bmatrix} z_{\mathcal{I}} \\ \nu \end{bmatrix} = \begin{bmatrix} -H_{\mathcal{I},\mathcal{A}} z_{\mathcal{A}} \\ -C_{\mathcal{A}} z_{\mathcal{A}} + d \end{bmatrix} \tag{4}$$

where $\bar{\lambda}_{\mathcal{I}} = \underline{\lambda}_{\mathcal{I}} = 0$. Solving (4) is called subspace minimization.

Next, the algorithm updates the inequality dual variables $\underline{\lambda}$ and $\bar{\lambda}$ to satisfy the stationarity (2a) condition

$$\bar{\lambda}_j = \begin{cases} -H_j z - (C^T\nu)_j & \text{if } j \in \overline{\mathcal{A}} \\ 0 & \text{if } j \notin \overline{\mathcal{A}} \end{cases} \tag{5a}$$

$$\underline{\lambda}_j = \begin{cases} H_j z + (C^T\nu)_j & \text{if } j \in \underline{\mathcal{A}} \\ 0 & \text{if } j \notin \underline{\mathcal{A}} \end{cases} \tag{5b}$$

where H_j is the j-th row of H.

The active-sets are then updated

$$\overline{\mathcal{A}}^+ \leftarrow \left\{j | (z_j > \overline{z}_j \text{ and } j \in \mathcal{I}) \text{ OR } (\bar{\lambda}_j > 0 \text{ and } j \in \overline{\mathcal{A}})\right\} \tag{6a}$$
$$\underline{\mathcal{A}}^+ \leftarrow \left\{j | (z_j < \underline{z}_j \text{ and } j \in \mathcal{I}) \text{ OR } (\underline{\lambda}_j > 0 \text{ and } j \in \underline{\mathcal{A}})\right\} \tag{6b}$$
$$\mathcal{I}^+ \leftarrow \left\{j | j \notin \overline{\mathcal{A}}^+ \cup \underline{\mathcal{A}}^+\right\}, \tag{6c}$$

where the plus superscript indicates the active-set at the next iteration. The algorithm then iterates using (4)-(6).

If (6) does not change the active set, then the complementarity conditions (2c) and (2d) are satisfied, and the algorithm can terminate because the algorithm has found primal z and dual ν, $\underline{\lambda}$, $\bar{\lambda}$ variables that satisfy the KKT conditions (2). In this case z is the optimal solution to the quadratic program (1). Thus, optimality can be verified by checking if the active-sets $\underline{\mathcal{A}}$ and $\overline{\mathcal{A}}$ have changed during the last iteration; the algorithm is summarized in Algorithm 1.

Algorithm 1: Primal-Dual Active-Set Method

Find feasible initial active-set partition $\overline{\mathcal{A}}_0$, $\underline{\mathcal{A}}_0$, $\mathcal{I}_0$.
while $\underline{\mathcal{A}}^+ \neq \underline{\mathcal{A}}$ *or* $\overline{\mathcal{A}}^+ \neq \overline{\mathcal{A}}$ **do**
  Solve subspace minimization problem (4)
  Update duals using (5)
  Update active-sets $\overline{\mathcal{A}}$ and $\underline{\mathcal{A}}$ using (6)
end

In [2], extensions are made to this algorithm to deal with potential cycling of the active-set updates. We do not consider this case in our paper.

B. Application to Model Predictive Control

In this section we apply the primal-dual active-set algorithm to the following linear model predictive control problem

$$\min_{x_k, u_k} \quad x_N^T P x_N + \sum_{k=0}^{N-1} (x_k^T Q x_k + u_k^T R u_k) \tag{7a}$$
$$s.t. \quad x_{k+1} = Ax_k + Bu_k + d_k \tag{7b}$$
$$\underline{x} \leq x_k \leq \overline{x}, \ k = 1, \ldots, N-1 \tag{7c}$$
$$\underline{u} \leq u_k \leq \overline{u}, \ k = 0, \ldots, N-1 \tag{7d}$$
$$x_0 = x(t) \tag{7e}$$

where $x_k \in \mathbb{R}^n$ is the predicted state under the control action $u_k \in \mathbb{R}^m$ and predicted disturbance $d_k \in \mathbb{R}^n$ over the horizon N, and $Q, R, P \succ 0$. We will use the shorthand notation $X = [x_1^T, \ldots, x_N^T]^T$ and $U = [u_0^T, \ldots, u_{N-1}^T]^T$.

The model predictive control problem (7) can be posed as a quadratic program in the standard form (1). The decision variables z are the state x_k and input u_k trajectories,

$$z = \begin{bmatrix} x_1^T & \dots & x_N^T & u_0^T & \dots & u_{N-1}^T \end{bmatrix}^T. \tag{8}$$

The dynamics (II-B) in the model predictive control problem (7) produce the equality constraints (2b) in the quadratic program (1). The equality constraint matrix C and vector d are given by

$$C = \begin{bmatrix} I & 0 & \dots & 0 & -B & 0 & \dots & 0 \\ -A & I & \dots & 0 & 0 & -B & & 0 \\ & & \ddots & \vdots & \vdots & & \ddots & \vdots \\ 0 & \dots & & I & 0 & 0 & \dots & -B \end{bmatrix} \tag{9a}$$

$$d = \begin{bmatrix} (Ax_0 + d_0)^T & d_1^T & \dots & d_{N-1}^T \end{bmatrix}^T. \tag{9b}$$

The cost function (7a) in the model predictive control problem (7) produces the quadratic cost in the quadratic program (1). The Hessian matrix H is given by

$$H = \text{diag}(2Q \ \dots \ 2Q \ 2P \ 2R \ \dots \ 2R). \tag{10}$$

Finally the lower and upper bounds $\underline{z}$ and $\overline{z}$ on the decision variables $z = [X^T, U^T]^T$ are given by

$$\underline{z} = \begin{bmatrix} \underline{x}^T & \dots & \underline{x}^T & \underline{u}^T & \dots & \underline{u}^T \end{bmatrix}^T \tag{11a}$$

$$\overline{z} = \begin{bmatrix} \overline{x}^T & \dots & \overline{x}^T & \overline{u}^T & \dots & \overline{u}^T \end{bmatrix}^T. \tag{11b}$$

Note that if a particular state or input does not have a lower or upper bound, we can use $\pm\infty$ as the bound on the state.

At the beginning of each iteration, Algorithm 1 has active-set partition $\overline{\mathcal{A}}$, $\underline{\mathcal{A}}$, and $\mathcal{I}$ that satisfies $\{X_\mathcal{I}, U_\mathcal{I} : \text{(II-B)}\} \neq \varnothing$ where $X_\mathcal{I}$ and $U_\mathcal{I}$ are the states $[x_k]_i$ and control inputs $[u_k]_i$ that are not saturated, $i \in \mathcal{I}$, respectively.

Algorithm 1 performs the subspace minimization (4) to update the state and input trajectory z to satisfy the dynamics equality constraints and stationarity condition (2a). The algorithm iterates until an optimal state and input trajectory z is found that satisfies bound constraints (7c), (7d).

III. A Decentralized Implementation on Star Communication Networks

In this section we present a decentralized implementation of the primal-dual active-set algorithm for distributed systems with star communication networks.

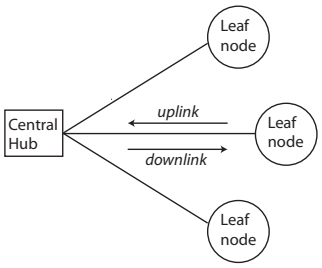

Fig. 1: Star Network Communication Graph

A. Decentralized Primal-Dual Active-Set Algorithm

We begin by defining our decentralized problem. We consider problem (7), with block diagonal A composed of r sub-blocks A_i:

$$A = \begin{bmatrix} A_1 & 0 & \dots \\ 0 & \ddots & \vdots \\ \vdots & \dots & A_r \end{bmatrix} \tag{12}$$

The B matrix is general and provides the coupling in the dynamics. Additionally, we consider the cost matrices P, Q, and R of (7a) are block diagonal

$$P = \begin{bmatrix} P_1 & & \\ & \ddots & \\ & & P_r \end{bmatrix}, Q = \begin{bmatrix} Q_1 & & \\ & \ddots & \\ & & Q_r \end{bmatrix}, R = \begin{bmatrix} R_1 & & \\ & \ddots & \\ & & R_r \end{bmatrix}. \tag{13}$$

The box constraints in problem (7) are naturally decoupled. We assume control action is computed in a decentralized fashion and the processors communicate through a star network (see Figure 1).

The splitting of the algorithm is as follows. The i-th leaf node is responsible for the states corresponding to sub-block A_i and the corresponding elements of $\underline{\lambda}$ and $\overline{\lambda}$. The elements of u may also be split up similarly (e.g. if $m = n$), but depend on the physical system.

The decentralized primal-dual active-set method is summarized in Algorithm 2. The active-sets $\overline{\mathcal{A}}$, $\underline{\mathcal{A}}$, and $\mathcal{I}$ are initialized at the leaf nodes; an example distributed initialization procedure is included at the beginning of Algorithm 2. Each leaf node sends its active-sets up to the central hub, where the full $\overline{\mathcal{A}}$, $\underline{\mathcal{A}}$, and $\mathcal{I}$ sets are aggregated. The central hub computes the subspace minimization problem (4). Then, the computed $z_\mathcal{I}$ and ν are sent down to the leaf nodes: each leaf node receives its relevant elements of x_k and u_k that were also elements of $z_\mathcal{I}$. The i-th leaf node then computes its components of dual variables $\underline{\lambda}$ and $\overline{\lambda}$. The active-sets are then updated in parallel by the leaf nodes. This process continues iteratively until the central hub can confirm that the full active-sets $\overline{\mathcal{A}}$, $\underline{\mathcal{A}}$, and $\mathcal{I}$ have not changed over the latest iteration.

Algorithm 2: Distributed Primal-Dual Active-Set

Set $u_k = 0 \ \forall k$. Compute
$x_{k+1} = Ax_k + Bu_k + d_k \ \forall k = 0, \dots, N-1$.
$\overline{\mathcal{A}}_0 = \{j = (k-1)n + i \mid [x_k]_i > \overline{x}\}$
$\underline{\mathcal{A}}_0 = \{j = (k-1)n + i \mid [x_k]_i < \underline{x}\}$
$\mathcal{I}_0 = \{j = (k-1)n + i \mid i \notin \mathcal{J}_0 \cup \mathcal{K}_0\}$
while $\underline{\mathcal{A}}^+ \neq \underline{\mathcal{A}}$ or $\overline{\mathcal{A}}^+ \neq \overline{\mathcal{A}}$ **do**
  Uplink: Send current active-set to central hub.
  Central Hub: Solve subspace min problem (4)
  Downlink: Send $z_\mathcal{I}$ and ν to leaf nodes.
  Leaf Node i: Update the leaf's components of dual variables $\underline{\lambda}$, $\overline{\lambda}$ using (5). Update $\overline{\mathcal{A}}, \underline{\mathcal{A}}$ by (6).
end while

B. Dual Decomposition

In this section we present a fixed step-size dual decomposition algorithm. In Section V-A we will compare this algorithm with our decentralized primal-dual active-set algorithm.

Dual decomposition uses a projected subgradient method to compute a solution [5], [11]. For (7), the projected subgradient method simplifies to a gradient projection method, where $[\cdot]_+$ is a projection onto the nonnegative real numbers.

The algorithm initializes with feasible dual variables $\underline{\lambda}$, $\bar{\lambda}$, and ν. Then, it updates the primal variables x_k and u_k based on the dual variables $\underline{\lambda}$, $\bar{\lambda}$, and ν. Next, the dual variables $\underline{\lambda}$, $\bar{\lambda}$, and ν are updated based on the primal variables x_k and u_k and a constant step-size α. The algorithm terminates when the dual variable updates are sufficiently small, e.g. $\|\lambda^{s+1} - \lambda^s\| < \epsilon$.

The steps of dual decomposition with a fixed step-size α for the MPC problem (7) are described in Algorithm 3 where the s-th primal updates are computed by

$$x_k^s = \frac{1}{2}Q^{-1}\big(\underline{\lambda}_k^s - \bar{\lambda}_k^s + A^T\nu_{k+1}^s - \nu_k^s\big)\ \forall k < N \tag{14a}$$

$$x_N^s = \frac{1}{2}P^{-1}\big(\underline{\lambda}_k^s - \bar{\lambda}_k^s - \nu_k^s\big) \tag{14b}$$

$$u_k^s = \frac{1}{2}R^{-1}\big(\underline{\lambda}_k^s - \bar{\lambda}_k^s + B^T\nu_{k+1}\big) \tag{14c}$$

and the $(s+1)$-th dual updates are computed by

$$\underline{\lambda}_k^{s+1} = \big[\underline{\lambda}_k^s + \alpha(-x_k + \underline{x})\big]_+ \tag{15a}$$

$$\underline{\lambda}_{k+N}^{s+1} = \big[\underline{\lambda}_{k+N}^s + \alpha(-u_k + \underline{u})\big]_+ \tag{15b}$$

$$\bar{\lambda}_k^{s+1} = \big[\bar{\lambda}_k^s + \alpha(x_k - \overline{x})\big]_+ \tag{15c}$$

$$\bar{\lambda}_{k+N}^{s+1} = \big[\bar{\lambda}_{k+N}^s + \alpha(u_k - \overline{u})\big]_+ \tag{15d}$$

$$\nu_k^{s+1} = \nu_k^s + \alpha(x_k - Ax_{k-1} - Bu_{k-1} - d_{k-1}). \tag{15e}$$

The splitting of the dual decomposition algorithm is the same as the primal-dual active-set method, except the dual variable ν is computed on the leaf nodes. On the uplink, the dual variables $\underline{\lambda}$, $\bar{\lambda}$, and ν are sent from the leaf nodes to the hub. The primal updates of (14a)–(14c) are done at the central processor. On the downlink, the primal variables x_k and u_k are sent down to the leaf nodes. The i-th component of the dual updates (15a)–(15e) are done at the i-th leaf node.

Algorithm 3: Dual Decomposition Algorithm

Initialize dual variables, $\underline{\lambda}_k^s, \bar{\lambda}_k^s, \nu_k^s$ for iteration s = 0.
while $\|\nu_k^{s+1} - \nu_k^s\| \not\leq \epsilon\ \forall k \in \{1, \ldots, N\}$, OR
$\|\underline{\lambda}_k^{s+1} - \underline{\lambda}_k^s\| \not\leq \epsilon\ \forall k \in \{1, \ldots, 2N\}$, OR
$\|\bar{\lambda}_k^{s+1} - \bar{\lambda}_k i^s\| \not\leq \epsilon\ \forall k \in \{1, \ldots, 2N\}$ **do**
  Uplink: Send dual variables to the central hub.
  Central Hub: Compute primal variables as in (14).
  Downlink: Send primal variables to the leaf nodes.
  Leaf Node i: Update leaf's components of the dual variables $\underline{\lambda}$, $\bar{\lambda}$, ν by (15).
end while

IV. Application: Thermal Building Control

Building HVAC control systems regulate fans, dampers, and opening of heating/cooling coils to control air temperature in the occupied spaces in the building. The central Air Handling Unit (AHU) provides cooling to the whole building, and the Variable Air Volume (VAV) boxes provide heating to each zone (see Figure 2). The control objective of building controllers is to regulate temperature in each thermal zone while minimizing energy consumption. Model predictive control for buildings has been shown to outperform the industry standard production logic in buildings. For more details on model predictive control in buildings, see [12], [13], [14], [15], and the references therein.

Heating, ventilation, and air-conditioning of buildings is an example of a decentralized control problem. In modern buildings, each thermal zone has its own sensors, actuators, and computational resources. In addition there are centralized sensors, actuators, and computational resources that couple the closed-loop dynamics of the thermal zones. Decentralized control schemes are attractive since they can take advantage of the distributed computational resources in each of the thermal zones and at the central hub. The processors at the central hub and in the thermal zones are connected by a star communication network as shown in Figure 1. A leaf node corresponds to a thermal zone and the central hub corresponds to the AHU.

We assume building dynamics with decoupled states and coupled inputs [16], [17]:

$$x_i(k+1) = A_i x_i(k) + B_i[u_m(k), u_i(k)]^T + d_i(k) \tag{16}$$

where $x_i(k)$ is the temperature of zone i at time k, $u_m(k) \leq 0$ is the central cooling power input, $u_i(k) \geq 0$ $(i \neq m)$ is the local heating power input, and $d_i(k)$ is the thermal load on the zone. The centrally produced cooling $u_m(k)$ couples the dynamics of the thermal zones. The building has n thermal zones and $m = n + 1$ control inputs. In practice, dynamic coupling in (16) is either negligible or is modeled by adding it into the disturbance $d_i(k)$.

The constraints on the system are comfort bounds on the temperature $[\underline{x}, \overline{x}]$ and control input saturation limits $[\underline{u}, \overline{u}]$. The MPC cost function is a weighted sum of the squared inputs which approximates the power consumption. Thus, the building control problem can be formulated as a standard MPC problem of the form (7). Splitting of the problem is as described in Section III-A, with the extra u_m stored at the central node since it physically corresponds to the central cooling

The standard building communication protocol is BACnet (Building Automation and Control networks) [18]. We consider the BACnet local area network (LAN) option of Attached Resource Computer Network (ARCnet). ARCnet's communication in a building is a star network, with one node acting as a central hub for communication, and the rest acting as leaf nodes [19] (c.f. Figure 1). Studies done to estimate delay time Δt_{delay} for the *uplink* messages on an ARCNET network find delay times of about 0.0317 seconds

per double precision floating point to 0.39155 seconds per double precision floating point [19].

In [8], the authors propose a model predictive control algorithm solved in a decentralized manner using dual decomposition. The results provided in the following section illustrate that this algorithm is not feasible under real-time communication constraints.

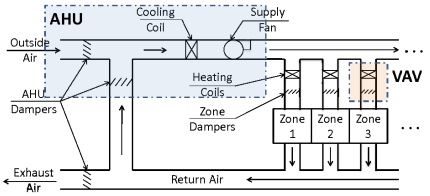

Fig. 2: Diagram of a Forced Air HVAC System

V. Analysis and Results

In this section we analyze the computational complexity of the primal-dual active-set method and compare with dual decomposition.

A. Theoretical Analysis

For a decentralized algorithm on a star communication network the execution time is given by

$$T = n_{iters}(\Delta t_{hub} + \Delta t_{leaf} + (m_{up} + m_{down})\Delta t_{delay}) \quad (17)$$

where n_{iters} is the total number of iterations needed for the algorithm to converge, Δt_{hub} is the amount of time required to perform the hub computations, Δt_{leaf} is the amount of time required to perform the leaf computations, Δt_{delay} is the amount of time required to communicate between the hub and leaves, and m_{up} and m_{down} are the number of floating precision variables passed between the central hub and leaf nodes.

A decentralized implementation on distributed platforms must balance the tradeoff between distribution of computation and minimizing communication. The best distributed algorithm minimizes the execution time T. In this section we present a theoretical analysis showing the tradeoffs between dual decomposition and the primal-dual active-set method.

First, we compare the number variables communicated m_{up} and m_{down} for each algorithm. The primal-dual active-set algorithm sends z and ν from the hub to the leaves and active-sets $\overline{\mathcal{A}}$, $\underline{\mathcal{A}}$, and $\mathcal{I}$ from the leaves to the hub during each iterations. Therefore $m_{up} = N(n+m)$ and $m_{down} = N(2n+m)$ for the primal-dual active-set algorithm. The dual decomposition algorithm sends primal variables x, u from the hub to the leaves and dual variables $\underline{\lambda}, \overline{\lambda}, \nu$ from the leaves to the hub during each iteration. Therefore $m_{up} = 2(n+m)N + nN$ and $m_{down} = N(n+m)$.

Next, we compare the number of iterations n_{iters} needed for convergence for each algorithm. For a gradient projection method implementation of dual decomposition, the rate of convergence is linear [20]:

$$\|z^{s+1} - z^*\| \leq \alpha \|z^s - z^*\|. \quad (18)$$

On the other hand, the primal-dual active-set is equivalent to a semi-smooth Newton method; for (7), this method has quadratic local convergence. Moreover, if the KKT matrix (4) meets certain special conditions, the algorithm converges in a finite number of steps [1]. Therefore, theoretically the primal-dual active-set algorithm requires many fewer iterations than the dual decomposition algorithm.

Finally, we compare the order of magnitude of the computation time required for each algorithm at the hub and at the leaves. The primal-dual algorithm requires at worst case, $O((N(n+m))^3)$ time to solve the subspace minimization in the hub processor and $O(N(n+m))$ time to update the dual variables at each leaf node. The dual decomposition algorithm requires $O((Nn(n+m)))$ time to update the primal variables in the central hub and $O(N(n+m))$ to update the dual variables in each leaf node.

The results of our analysis are summarized in Table I where "DD" for stands for dual decomposition and "PDAS" for primal-dual active-set. The primal-dual active-set performs worse than dual decomposition on the computation at the central hub, but does marginally better for the size of the messages $m_{up} + m_{down}$ sent at each iteration. Thus, the splitting of the computation scales more favorably for the dual decomposition. In general, dual decomposition performs better when the communication delay time Δt_{delay} is negligible and the computational power of the hub is limited. This is the opposite of the situation found in the HVAC control problem. The primal-dual active-set method performs best because $(m_{up} + m_{down})\Delta t_{delay} \gg \Delta t_{hub} + \Delta t_{leaf}$.

Algorithm	DD	PDAS
Hub	$O(Nn(n+m))$	$O((N(n+m))^3)$
Leaf Node	$O(N(n+m))$	$O(N(n+m))$
m_{up}	$N(3n+m)$	$N(n+m)$
m_{down}	$N(n+m)$	$N(2n+m)$

TABLE I: Theoretical Trade-offs

B. Numerical Analysis

In this section we present experimental results to compare the execution time T for the primal-dual active-set and dual decomposition algorithms.

For our comparison we look at two test MPC problems of the form (7). In the first test, the B matrix corresponds to building HVAC dynamics in (16). In the second test problem we constructed a random B matrix generated my `sprandsym.m` with density of 0.6. We use the best-case communication delay time of $\Delta t_{delay} = 0.03$ seconds per double floating point to estimate communication delay. The problem parameters are summarized in Table II.

A	`rand(n,1)`
$d_i(k)$	$[-2, 2]$
$[\underline{x}, \overline{x}]$	$[-5, 5]$
$[\underline{u}, \overline{u}]$	$[-3, 3]$
P, Q	$0.1I_n$
R	$10I_n$
N	5

TABLE II: Experimental Problem Parameters

Our numerical experiments were conducted in Matlab on a MacBook Pro with a 2.2 GHz Intel Core i7. Each result is averaged from a set of 10 trials. The precision required for dual decomposition convergence is $\epsilon = 1\times10^{-8}$, which is the default tolerance for the interior point algorithm in `quadprog.m`. The results are presented in Tables III, and IV where $T_{comp} = n_{iters}(\Delta t_{hub} + \Delta t_{leaf})$, $T_{delay} = n_{iters}(m_{up} + m_{down})\Delta t_{delay})$, and all time measurements are presented with the unit of seconds.

DD					
n	n_{iters}	T_{comp}	T_{delay}	Δt_{hub}	Δt_{leaf}
5	429.1	0.2220	1673.5	9.4929e-05	4.2354e-04
10	1257	0.7591	9616.1	1.0988e-04	5.3053e-04
50	5818.8	5.9688	219080	1.5394e-04	8.7108e-04
100	8766.6	11.1055	658810	1.6629e-04	0.0011
PDAS					
n	n_{iters}	T_{comp}	T_{delay}	Δt_{hub}	Δt_{leaf}
5	1	8e-04	3.15	6.1502e-04	1.768e-04
10	1.2	.0020	7.38	0.0014	1.5918e-04
50	2.6	0.1370	78.39	0.0526	3.5511e-04
100	2.43	0.7296	146.08	0.3004	4.0724e-04

TABLE III: Solver breakdown for HVAC problem

The data for Δt_{hub} and Δt_{leaf} are presented in Table III to support the theoretical analysis presented in the previous subsection. Even though Δt_{hub} is significantly smaller for dual decompostion, the net computation time T_{comp} is much larger than that of the primal-dual active-set method simply because n_{iters} is much larger.

n	DD			PDAS		
	n_{iters}	T_{comp}	T_{delay}	n_{iters}	T_{comp}	T_{delay}
5	358.7	0.787	807.075	1.1	0.00123	2.475
10	1518.2	7.6366	6831.9	1.6	0.00267	7.20
50	339.6	10.590	7641	2.2	0.1072	49.50
100	295.7	22.123	13307	2.2	.6161	99

TABLE IV: Solver breakdown for Random problem

In practice, the number of iterations n_{iters} is many orders of magnitude less for the primal dual active-set than for dual decomposition. From these results, it is quite clear that the impact of communication on solver time is not negligible – even in the best case scenario for communication delay Δt_{delay}. The worst case delay for the dual decomposition algorithm is on the order of days for the HVAC problem and the order of hours for the random problem. This result would be amplified for the worst case communication latency of $\Delta t_{delay} = 0.4$ seconds per point.

VI. Conclusions and Future Work

In our work, we have developed a novel distributed model predictive control algorithm that utilizes much less communication than most other proposed distributed algorithms. The conclusion of this research is that communication has a large impact on distributed algorithms. This idea is not new to the parallel computing community, and the work done here shows its need in practical distributed control.

VII. Acknowledgements

We would like to acknowledge Yudong Ma for his contribution of experimental building communication delay data. This material is based upon work supported by the National Science Foundation under Grant No. DGE 1106400.

References

[1] M. Hintermüller, K. Ito, and K. Kunisch, "The primal-dual active set strategy as a semismooth newton method," *SIAM Journal on Optimization*, vol. 13, 2002.

[2] F. Curtis, Z. Han, and D. Robinson, "A globally convergent primal-dual active-set framework for large-scale convex quadratic optimization," *Lehigh University, COR@L Technical Report 12T-13*, 2012.

[3] M. Hintermüller, "Semismooth newton methods and applications," 2010. Online at `http://www.math.uni-hamburg.de/home/hinze/Psfiles/Hintermueller_OWNotes.pdf`.

[4] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation*, vol. 290. Englewood Cliffs, NJ: Springer-Verlag, 1989.

[5] S. Boyd, L. Xiao, and A. Mutapcic, "Notes on decomposition methods," 2003. Online at `http://web.stanford.edu/class/ee392o/decomposition.pdf`.

[6] E. Wie, A. Ozdaglar, and A. Jadbabaie, "A distributed newton method for network utility maximization," *IEEE Transactions on Automatic Control*, vol. 58, September 2013.

[7] K. Asanovic, B. Catanzaro, J. Gebis, P. Husbands, D. Patterson, W. Plishker, J. Shalf, S.W.Williams, and K. Yelick, "The landscape of parallel computing research: A view from berkeley," tech. rep., Electrical Engineering and Computer Sciences, University of California at Berkeley, December 2006. Technical Report No. UCB/EECS-2006-183.

[8] Y. Ma, G. Anderson, and F. Borrelli, "A distributed predictive control approach to building temperature regulation," in *American Control Conference*, 2011.

[9] A. Venkat, I. Hiskens, J. Rawlings, and S. Wright, "Distributed mpc strategies with application to power system automatic generation control," *Control Systems Technology, IEEE Transactions on*, vol. 16, pp. 1192–1206, Nov 2008.

[10] R. R. Negenborn, B. D. Schutter, and J. Hellendoorn, "Multi-agent model predictive control: A survey," *CoRR*, vol. abs/0908.1076, 2009.

[11] A. Rantzer, "Dynamic dual decomposition for distributed control," in *American Control Conference, 2009*, pp. 884 –888, Jun. 2009.

[12] Y. Ma, F. Borrelli, B. Hencey, B. Coffey, S. Bengea, and P. Haves, "Model predictive control for the operation of building cooling systems," *Control Systems Technology, IEEE Transactions on*, vol. 20, pp. 796 –803, may 2012.

[13] F. Oldewurtel, A. Parisio, C. Jones, M. Morari, D. Gyalistras, M. Gwerder, V. Stauch, B. Lehmann, and K. Wirth, "Energy efficient building climate control using stochastic model predictive control and weather predictions," in *American Control Conference (ACC), 2010*, pp. 5100 –5105, 30 2010-july 2 2010.

[14] Y. Ma, A. Kelman, A. Daly, and F. Borrelli, "Predictive control for energy efficient buildings with thermal storage: Modeling, stimulation, and experiments," *Control Systems, IEEE*, vol. 32, pp. 44 –64, feb. 2012.

[15] S. C. Bengea, A. D. Kelman, F. Borrelli, R. Taylor, and S. Narayanan, "Model predictive control for mid-size commercial building hvac: Implementation, results and energy savings," in *The Second International Conference on Building Energy and Environment*, 2012.

[16] S. Koehler and F. Borrelli, "Building temperature distributed control via explicit mpc and "trim and respond" methods," in *European Control Conference*, July 2013.

[17] A. Kelman, Y. Ma, and F. Borrelli, "Analysis of local optima in predictive control for energy efficient buildings," *Journal of Building Performance Simulation*, Apr 2012.

[18] S. Bushby, "A new age in building control systems," *Construction Business Review*, vol. 4, pp. 62–65, March/April 1994.

[19] S. Wong, S. Hong, and S. Bushby, "Nistir 7038a simulation analysis of bacnet local area networks," *National Institute of Standards and Technology*, October 2003.

[20] D. Luenberger and Y. Ye, *Linear and Nonlinear Programming*. New York, NY: Springer, 3 ed., 2008.