Preprints of the 19th World Congress
The International Federation of Automatic Control
Cape Town, South Africa. August 24-29, 2014

Diffusion Based Stopping Criterion for Distributed Optimization

Taylan Ayken * **Jun-ichi Imura** *,**

* *Dept. of Mechanical and Environmental Informatics, Tokyo Institute of Technology, Tokyo, Japan (e-mail: {tayken, imura} at cyb.mei.titech.ac.jp).*
** *JST CREST, Tokyo, Japan*

Abstract: As the size of system models to be optimized gets larger, distributed optimization, where each local optimization problem is solved by an individual computer in parallel to derive a global optimal solution more quickly and robustly than centralized methods, is becoming one of the important topics. However most distributed optimization techniques need a supervisor that checks the status of all the optimization algorithms running in a distributed way and messages them to stop. In this paper, we propose a diffusion based stopping criterion for distributed optimization algorithms. We then compare both supervised and diffusion based criteria by numerical simulation to show that the diffusion based criterion does not add any overhead.

Keywords: Large Scale Optimization Problems, Decentralized Control, Distributed Optimization, Stopping Criterion, Dual Decomposition, Distributed MPC.

1. INTRODUCTION

There is a global trend towards controlling large-scale network systems in a more sophisticated way as computational power increases. One of the prospective methods to solve this kind of control problem is Model Predictive Control (MPC), which is well known as a kind of real-time optimal control. However as the system size grows, it takes longer to optimize this problem in a centralized manner. Also a centralized system is not as robust as a distributed system. If the central computer optimizing the system has a trouble, the whole system shuts down. This can be solved by having redundant optimization system setup. The other problem is if one of the sub-systems has a problem, this changes the whole network graph. This means for a centralized system to be robust against sub-system failures, it should have multiple network configuration graphs and should be able to switch to a suitable one according to the configuration. This is infeasible for very large systems such as nation wide power grids.

In order to solve this problem Distributed MPC is one of the effective methods as we can divide the system into smaller sub-systems and solve these smaller sub-systems in a distributed way to find the optimum solution, see e.g., Nedic and Ozdaglar (2009), Johansson et al. (2008), Nedic et al. (2010), Wei et al. (2010), Giselsson and Rantzer (2010), Ma et al. (2011), Ono and Williams (2010), Zhu and Martinez (2012) for different methods. Then this solution can be used directly as a control input or can be used as a reference input to lower level controllers that already exist in these systems. Dual Decomposition method, Boyd et al. (2008) and Boyd et al. (2011) , is one of the most efficient methods for distributing the general cost function to local cost functions for which the optimization problem can be relatively easily solved. Also this method is robust to sub-system failures because the sub-systems, which can be defined as a part of the network and the computer used for running the optimization algorithm, are calculating only a local optimal solution. Additionally in a distributed system, a sub-system failure can be quickly recognized by neighboring sub-systems as they will not receive any messages from it, thus realizing whether the sub-system is off-line or not. However, the conventional distributed algorithm requires a central supervisor that checks if all variables of sub-systems converged to the optimum or not and sends them a stopping message if they all have converged. This means that there is a supervisor that requires global information and it should be connected to all sub-systems by communication lines, which contradicts to the essence of distribution. Finally this supervisor means that there is a single point of failure in our system which compromises the robustness feature; meaning that if something happens to this supervisor, the whole system stops functioning.

This paper proposes a new type of distributed stopping criterion called here diffusion based stopping criterion for distributed optimization, which enables us to determine *in a distributed way* whether or not the optimization error derived by all optimization algorithms running on sub-systems converges within a certain threshold by using local information only.

The remainder of this paper is organized as follows. Section 2 gives a more detailed description of the system, describes the problem and looks at the distribution method. Then in Section 3, we propose a diffusion based stopping criterion. In Section 4, we apply our algorithm to a dispatch control problem, and show that our algorithm is effective from the viewpoints of computation time and robustness. Section 5 concludes the paper.

2. DISTRIBUTED OPTIMIZATION

2.1 Network formulation

A network system can be represented with the following equations:

$$x(k+1) = Ax(k) + Bu(k) \tag{1}$$
$$Cx(k) + Du(k) \leq E \tag{2}$$
$$\eta(k) = Fx(k) + Gu(k) + Hd(k) = 0 \tag{3}$$

$A = diag(A_i,\ i = 1,\ldots,M)$, $B = diag(B_i,\ i = 1,\ldots,M)$,
$C = diag(C_i,\ i = 1,\ldots,M)$, $D = diag(D_i,\ i = 1,\ldots,M)$,
$E = [E_1^T \cdots E_M^T]^T$, $F = [F_1 \cdots F_M]$,
$G = [G_1 \cdots G_M]$, $H = [H_1 \cdots H_r]$,

$$x(k) = [x_1^T(k) \cdots x_M^T(k)]^T, \quad x_i(k) \in R^{n_i},$$
$$u(k) = [u_1^T(k) \cdots u_M^T(k)]^T, \quad u_i(k) \in R^{m_i},$$
$$d(k) = [d_1(k) \cdots d_r(k)]^T, \quad d_i(k) \in R$$

where $x(k)$ and $u(k)$ are state and control input vectors, respectively, $d(k)$ is the exogenous signal input and k is the discrete time.

We here consider M sub-systems that interact with each other throughout equality constraints (3).

Remark 1. It is possible to consider A, B, C and D matrices that are not block-diagonal. This problem can be solved by dual decomposition, i.e. by using two variables for the shared variable and adding an equality constraint to them by using matrices F and G. Also, although the exogenous signal input, $d(k)$, is only included in (3), it can be included in (1) & (2). The method proposed here can be easily extended to these cases by adding additional variables and changing (1)-(3). Thus for simplicity of notation, we focus on the above case.

Consider a cost function that we will use for optimization:

$$J_0 = \sum_{k=0}^{N-1} x^T(k)Qx(k) + u^T(k)Ru(k) \tag{4}$$

where $x(k)$ and $u(k)$ are state and control vectors; $Q = diag(Q_i)$ and $R = diag(R_i)$ are positive (semi-)definite matrices for defining the cost and N denotes the prediction horizon length.

Problem 2. Given $x(0)$, $d(k)$, $k = 0,\ldots,N-1$, Q and R; find $u(k)$ minimizing J_0 while satisfying (1)-(3) for $k = 0,\ldots,N-1$.

Consider a network system in Fig. 1 where the nodes denoted by the circles and the square express sub-systems and a user (demand), respectively. Suppose that sub-system at node 1, say, sub-system 1 is a source, which is a type of sub-system that supplies the commodity flowing in the single commodity network flow problem such as a water well or a generator, and it has the following equations:

$$x_1(k+1) = A_1x_1(k) + B_1u_1(k)$$
$$0 \leq x_{11}(k) + x_{12}(k) \leq 1000$$

where $x_1(k) = [x_{11}(k)\ x_{12}(k)]^T$ is the state vector expressing the supplied water or power to each line.

In a similar way, suppose that the sub-system at node 2, say, sub-system 2 is a storage sub-system, which stores the commodity flowing in the single commodity network flow problem

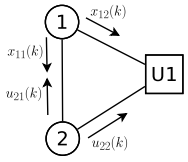

Fig. 1. An example graph for explaining system matrices.

such as a water tower or a battery, and it has the following equations:

$$x_2(k+1) = x_2(k) - u_{21}(k) - u_{22}(k),$$
$$-1000 \leq u_{21}(k) + u_{22}(k) \leq 1000,$$
$$0 \leq x_2(k) \leq 8000$$

where $x_2(k)$ is the state variable and $u_2(k) = [u_{21}(k)\ u_{22}(k)]^T$ is the control vector.

Also, suppose that the user at node $U1$ consists of an exogenous signal d_1, which is the demand in the single commodity network flow problem. It has to be satisfied by both the output of sub-system 1 and sub-system 2, so we have the following equality constraint:

$$x_{11}(k) + u_{21}(k) = 0$$
$$x_{12}(k) + u_{22}(k) + d_1(k) = 0$$

In this way, this example can be expressed by (1)-(3).

2.2 Distributed optimization based on dual decomposition

The system described in Problem 2 can be solved by using a centralized optimization approach. In fact, for a small scale system, it may be better to use a centralized optimization algorithm because it is usually easier to be implemented. However, as the number M of sub-systems grows, the calculations will take longer to execute.

One possible way to distribute the global cost function (4) into local cost functions, while satisfying the equality constraint, is to use the dual decomposition method, Boyd et al. (2008). We start by decomposing the cost function and equality constraint. Because of the properties of Q, R, F, G and H matrices, we can write (4) and (3) as, respectively,

$$J_0 = \sum_{k=0}^{N-1} \sum_{i=1}^{M} x_i^T(k)Q_ix_i(k) + u_i^T(k)R_iu_i(k) \tag{5}$$

$$\eta(k) = \sum_{i=1}^{M} \hat{F}_iX_i(k) + \sum_{j=1}^{r} H_jd_j(k) = 0 \tag{6}$$

where $x_i(k)$ and $u_i(k)$ are state and control vectors for the sub-system i; Q_i and R_i are cost matrices for the sub-system i; and $\hat{F}_i = [F_i\ G_i]$ and $X_i(k) = [x_i^T(k)\ u_i^T(k)]^T$.

So consider to separate the global cost function (5) into local cost functions for individual sub-systems by using the Lagrangian of the global cost function (5):

$$L_0 = J_0 + \sum_{k=0}^{N-1} \lambda^T(k)\eta(k) \tag{7}$$

where the vector $\lambda(k)$ denotes the dual variables (also called "price") at time k.

If we insert (5) and (6) into (7) and change the order of summation, we get

$$L_0 = \sum_{i=1}^{M} \left[\sum_{k=0}^{N-1} x_i^T(k) Q_i x_i(k) + u_i^T(k) R_i u_i(k) + \lambda^T(k) \hat{F}_i X_i(k) \right] + \sum_{k=0}^{N-1} \sum_{j=1}^{r} \lambda^T(k) H_j d_j(k) \quad (8)$$

We can then define the part inside the parentheses as the local cost function. So for node i, the local cost function is

$$L_{i,0} = \sum_{k=0}^{N-1} x_i^T(k) Q_i x_i(k) + u_i^T(k) R_i u_i(k) + \lambda_i^T(k) X_i(k) \quad (9)$$

where $\lambda_i(k)^T = \lambda^T(k)\hat{F}_i$. Thus (7) is equal to

$$L_0 = \sum_{i=1}^{M} L_{i,0} + \sum_{k=0}^{N-1} \lambda^T(k) K(k) \quad (10)$$

where $K(k) = \sum_{j=1}^{r} H_j d_j(k)$.

Because of the properties of A, B, C, D and E matrices, we can write (1) and (2) as, respectively,

$$x_i(k+1) = A_i x_i(k) + B_i u_i(k) \quad (11)$$

$$C_i x_i(k) + D_i u_i(k) \leq E_i \quad (12)$$

Suppose that $\lambda_i(k)$ $k = 0, \ldots, N-1$ is given. We can write Problem 2 as:

Problem 3. Given $x_i(0)$, $\lambda_i(k)$, $k = 0, \ldots, N-1$, Q_i and R_i; find $u_i(k)$ minimizing $L_{i,0}$ while satisfying (11) and (12) for $k = 0, \ldots, N-1$.

This problem can be converted to quadratic programming problem and solved accordingly. If the solution $u_i^*(k)$ that minimizes (9) for given $\lambda_i^*(k)$ $\forall i$ also satisfies (3), then it is a solution that minimizes (7) which means it is a solution that minimizes (4), i.e. the global cost, while satisfying (3).

We now have to find a way to update $\lambda_i(k)$, $k = 0, \ldots, N-1$, for each sub-system. Define $\bar{\lambda} = [\lambda^T(0) \cdots \lambda^T(N-1)]^T$ and $\bar{\eta} = [\eta^T(0) \cdots \eta^T(N-1)]^T$. Then we define a gradient update rule for the dual variables $\bar{\lambda}$ as

$$\begin{aligned} \hat{\lambda}(\tau+1) &= \hat{\lambda}(\tau) - \alpha_\tau \bar{\eta} \\ \bar{\lambda} &= \hat{\lambda}(\tau_{end}) \end{aligned} \quad (13)$$

where α_τ denotes the step size at iteration $\tau \in \{0, 1, \ldots\}$ and one of the important factors that determine the convergence speed of the algorithm. We choose to use a fixed step size algorithm for simplicity. Also τ_{end} denotes the final iteration when a certain convergence criterion has been met.

We can use this to write the gradient update rule for $\bar{\lambda}_i$. We know that $\lambda_i(k) = \hat{F}_i^T \lambda(k)$ thus if we multiply both sides of (13) with $diag(\hat{F}_i)^T$ we get

$$\begin{aligned} \hat{\lambda}_i(\tau+1) &= \hat{\lambda}_i(\tau) - \alpha_\tau \bar{\eta}_i \\ \bar{\lambda}_i &= \hat{\lambda}_i(\tau_{end}) \end{aligned} \quad (14)$$

where $\bar{\eta}_i = [\eta_i^T(0) \cdots \eta_i^T(N-1)]$ and $\eta_i(k)$ can be defined as

$$\eta_i(k) = \hat{F}_i^T \hat{F}_i X_i(k) + \sum_{j \in \mathcal{N}_i} \hat{F}_i^T \hat{F}_j X_j(k) + \sum_{j=1}^{r} \hat{F}_i^T H_j d_j(k) \quad (15)$$

As a stopping criterion, we choose to satisfy the equality constraint $\bar{\eta}_i = 0$ with a minimal relaxation parameter denoted as ε, so our convergence criterion for sub-system i becomes:

$$\|\bar{\eta}_i\|_\infty < \varepsilon \quad (16)$$

where $\| \ldots \|_\infty$ is the maximum norm.

This relaxation parameter can be used for adjusting between the precision of solution and the convergence speed. When this convergence criterion is satisfied at some sub-system, it sends a message to a supervisor stating that the solution of the optimization algorithm running on it converged into some allowable solution boundary. The supervisor keeps track of all sub-systems' status and decides when to stop according to all sub-systems' status. This is the most widely used criterion in distributed optimization.

3. DIFFUSION BASED STOPPING CRITERION

3.1 Communication graph and stopping criterion matrix

As we stated before, the above supervisor based stopping criterion has several problems. It needs global information, causes an increase in costs as it requires additional communication lines and affects the robustness of the system as if something happens, causes the whole system to stop functioning. So we develop a diffusion based stopping criterion, which eliminates these problems. We name this stopping criterion diffusion based as the convergence message of a sub-system is not sent directly to the whole network, but "diffuses" through the the network via the stopping criterion matrix.

Assumption 4. With this stopping criterion, sub-systems only communicate with their immediate neighbors.

Each sub-system can learn the status of other sub-systems in the system by receiving messages from only its immediate neighbors. The information each sub-system receives can be considered as local as it can know if the whole system has converged or not but it cannot know if a particular sub-system has converged or not, provided that the sub-system is not its immediate neighbor.

For this criterion to work, we create a communication graph G^* by using (3). If both sub-systems i and j have 1's on the same row of F_i, F_j, G_i or G_j, they are neighbours in G^*. An example of the communication graph G^* can be seen in Fig. 2. We can create this graph as the first row of both F_1 and G_4 contains 1 so they are neighbors. In a similar way, the second row of both F_2 and G_5 contains 1 so they are neighbors, the third row of both F_3 and G_5 contains 1 so they are neighbors, the fourth row of both G_4 and G_5 contains 1 so they are neighbors, the fifth row of both F_1 and F_2 contains 1 so they are neighbors and the eighth row of F_3, G_4 and G_5 contains 1 so they are neighbors. The resulting neighbor sets are:

$$\begin{aligned} \mathcal{N}_1 &= \{2,4\}, \\ \mathcal{N}_2 &= \{1,5\}, \\ \mathcal{N}_3 &= \{4,5\}, \\ \mathcal{N}_4 &= \{1,3,5\}, \\ \mathcal{N}_5 &= \{2,3,4\} \end{aligned} \quad (17)$$

After this, each sub-system has to store a stopping criterion matrix, denoted $S_i \in \{0,1\}^{\varnothing G^* \times (|\mathcal{N}_i|+1)}$ for sub-system i, where

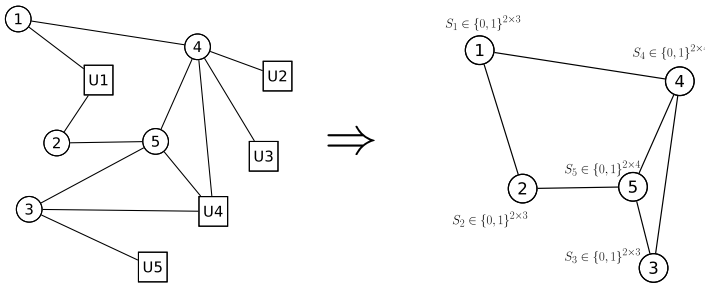

Fig. 2. An example graph G and its communication graph G^*

$\varnothing G^*$ is the diameter of the communication graph G^*, the maximum of the distance between any sub-systems (the distance between two sub-systems is given by the minimum number of edges between the sub-systems), $\mathscr{N}_i$ is the set of neighboring sub-systems of sub-system i and $|\mathscr{N}_i|$ is the number of neighbors of sub-system i. This matrix stores information about the convergence status of optimization algorithms running on each sub-system. If an element of S_i is 1, that means the algorithm of the corresponding sub-system has converged, i.e., (16) was satisfied; if it is 0 it means the opposite is true. The elements of the first row of S_i express the converge status of the algorithms running on sub-systems i and $j \in \mathscr{N}_i$, direct neighbors of sub-system i.

More rigorously, let $S_i(a,b)$ denote the (a,b)th element of S_i. This implies that the element $S_i(1,j)$, $j \leq |\mathscr{N}_i|$ keeps track of the algorithm running on the direct neighbor sub-system labeled as j and the element $S_i(1,|\mathscr{N}_i|+1)$ keeps track of the algorithm running on the sub-system i itself. The elements of the next rows of S_i keeps track of the status of the previous rows of S_i and S_j $\forall j \in \mathscr{N}_i$. The element $S_i(a+1,j)$, $j \leq |\mathscr{N}_i|$ keeps track of the status of the ath row of S_j and the element $S_i(a+1,|\mathscr{N}_i|+1)$ keeps track of the status of the ath row of S_i. This way by checking the elements of this matrix, we can determine the convergence status of all the algorithms, e.g., if some S_i has elements of all 1, then all S_j have elements of all 1 (see Theorem 6 for the details), that means the algorithms running on all sub-systems of the grid has converged. The details of this matrix S_i will be given at the example later.

We can say that this matrix contains only local information in the sense that, although it contains information about the convergence status of all the algorithms, there is no way of finding out the status of a particular sub-system if that sub-system is not a direct neighbor.

3.2 Algorithm

To sum it up, we can merge the dual decomposition based distributed optimization and the proposed diffusion based stopping criterion with the following algorithm.

Definition 5. Define the following messages from sub-system i with the stopping criteria matrix S_i:

(1) *convergence event message*: an arbitrary message stating that convergence criterion in (16) is satisfied for sub-system i.
(2) *convergence event break message*: an arbitrary message stating that the convergence criterion in (16) is not satisfied for sub-system i.
(3) *mth row event message*: an arbitrary message stating that every element of the mth row of S_i is 1.
(4) *mth row event break message*: an arbitrary message stating that every element of the mth row of S_i is not 1.

Then we can write down the algorithm for distributed optimization with the diffusion based stopping criterion for sub-system i with the stopping criterion matrix S_i as:

Algorithm 1. (Proposed distributed algorithm).

Let $x_i(k,\tau)$ and $u_i(k,\tau)$ represent the state and control input vectors at time step k, iteration τ, respectively. We define $\bar{u}_i(\tau) = [u_i^T(0,\tau)\cdots u_i^T(N-1,\tau)]^T$, $\bar{d}_i = [d_i^T(0)\cdots d_i^T(N-1)]^T$ and $\bar{X}_i(\tau) = [X_i^T(0,\tau)\cdots X_i^T(N-1,\tau)]^T$. Suppose that $x_i(0,1)$ is given.

(1) $\tau \leftarrow 1$, $\bar{\lambda}_i(0) \leftarrow 0$, $S_i \leftarrow 0^{\varnothing G^* \times (|\mathscr{N}_i|+1)}$
(2) $\bar{u}_i(\tau) \leftarrow arg\,min_{u_i} L_{i,0}$ given $\bar{\lambda}_i(\tau)$
(3) Send $\bar{X}_i(\tau)$ to sub-system j $\forall j \in \mathscr{N}_i$, receive $\bar{X}_j(\tau)$ from all sub-systems $j \in \mathscr{N}_i$
(4) $\bar{\eta}_i(\tau) \leftarrow diag(\hat{F}_i)^T diag(\hat{F}_i)\bar{X}_i(\tau)$
$+ \sum_{j\in\mathscr{N}_i} diag(\hat{F}_i)^T diag(\hat{F}_j)\bar{X}_j(\tau) + \sum_{j=1}^{r} diag(\hat{F}_i)^T diag(H_j)\bar{d}_j$
(5) $\hat{\lambda}_i(\tau) \leftarrow \hat{\lambda}_i(\tau-1) + \alpha_\tau \bar{\eta}_i(\tau)$
(6) If $\|\bar{\eta}_i(\tau)\|_\infty < \varepsilon$:
$S_i(1,j+1) \leftarrow 1$, send *convergence event message* to all sub-systems $j \in \mathscr{N}_i$.
Else:
$S_i(1,j+1) \leftarrow 0$, send *convergence event break message* to all sub-systems $j \in \mathscr{N}_i$, $\tau \leftarrow \tau+1$, and go to step 2.
(7) If sub-system i receives *convergence event message* from sub-system $j \in \mathscr{N}_i$:
$S_i(1,j) \leftarrow 1$.
Else if sub-system i receives *convergence event break message* from sub-system $j \in \mathscr{N}_i$:
$S_i(1,j) \leftarrow 0$, $\tau \leftarrow \tau+1$, and go to step 2.
(8) For $m \in \{1,\ldots,\varnothing G^*-1\}$:
(a) If every element of mth row of S_i is 1:
$S_i(m+1,j+1) \leftarrow 1$ and send *mth row event message* to all sub-systems $j \in \mathscr{N}_i$.
Else:
$S_i(m+1,j+1) \leftarrow 0$ and send *mth row event break message* to all sub-systems $j \in \mathscr{N}_i$, $\tau \leftarrow \tau+1$, and go to step 2.
(b) If sub-system i receives *mth row event message* from a sub-system $j \in \mathscr{N}_i$:
$S_i(m+1,j) \leftarrow 1$.
Else if receives *mth row event break message* from a sub-system $j \in \mathscr{N}_i$:
$S_i(m+1,j) \leftarrow 0$, $\tau \leftarrow \tau+1$, and go to step 2.
(9) If every element of the $\varnothing G^*$th row of S_i is 1:
$\tau_{end} = \tau$ and $u_i(0,\tau_{end})$ are outputted as a solution.
Else:
$\tau \leftarrow \tau+1$, and go to step 2.

With this algorithm, sub-system i sends a total of $\varnothing G^*+1$ messages at each iteration: Once for $\bar{X}_i(\tau)$, once for *convergence event message* or *convergence event break message* and then $\varnothing G^*-1$ *mth row event message* or *mth row event break message*.

Convergence of dual decomposition is guaranteed if a suitable step size, α_τ is selected, which means the gradient rule will not oscillate and will converge to a result in finite time. However this is not enough to guarantee convergence. But because of

$$S_2 = \begin{bmatrix}0&0&0\\0&0&0\end{bmatrix} \xrightarrow{\text{Step 1}} \begin{bmatrix}1&0&0\\0&0&0\end{bmatrix} \xrightarrow{\text{Step 2}} \begin{bmatrix}1&0&1\\0&0&0\end{bmatrix} \xrightarrow{\text{Step 3}} \begin{bmatrix}1&1&1\\0&0&1\end{bmatrix} \xrightarrow{\text{Step 4}} \begin{bmatrix}1&1&1\\0&0&1\end{bmatrix} \xrightarrow{\text{Step 5}} \begin{bmatrix}1&1&1\\1&1&1\end{bmatrix}$$
$$S_5 = \begin{bmatrix}0&0&0&0\\0&0&0&0\end{bmatrix} \qquad \begin{bmatrix}0&0&0&0\\0&0&0&0\end{bmatrix} \qquad \begin{bmatrix}1&0&0&0\\0&0&0&0\end{bmatrix} \qquad \begin{bmatrix}1&0&0&1\\1&0&0&0\end{bmatrix} \qquad \begin{bmatrix}1&1&0&1\\1&0&0&0\end{bmatrix} \qquad \begin{bmatrix}1&1&1&1\\1&1&1&1\end{bmatrix}$$

Fig. 3. Example of behavior of stopping criterion matrices of sub-systems 2 and 5

the following theorem, we can guarantee that our algorithm converges on a solution in finite time.

Theorem 6. In the above algorithm, an arbitrarily chosen S_i has elements of all 1s iff the algorithm of all sub-systems have converged to a solution at iteration τ.

Proof. Suppose that there is some i such that S_i has elements of all 1s and there is sub-system j in which the solution of the corresponding algorithm did not converge at iteration τ. This means that all sub-systems in N_j have a 0 in their first row. This, in turn, makes the last element of their second row 0. This also makes one of the elements of the second rows of neighbors of N_j 0. This effect cascades in the whole graph and makes at least one of the elements of the last row of S_i 0. This contradicts with the assumption of all elements of S_i are 1. The converse is obvious. This completes the proof.

3.3 Example

As an example, consider the graph G^* in Fig. 2, where the resulting $\varnothing G^*$ is 2. That means we have $S_1 = 0^{2\times 3}$ for sub-system 1, $S_2 = 0^{2\times 3}$ for sub-system 2, $S_3 = 0^{2\times 3}$ for sub-system 3, $S_4 = 0^{2\times 4}$ for sub-system 4 and $S_5 = 0^{2\times 4}$ for sub-system 5 initially. Let's follow stopping criterion matrices S_2 and S_5 of sub-systems 2 and 5, respectively as shown in Fig. 3.

We suppose that the elements $S_2(1,1)$, $S_2(1,2)$ and $S_2(1,3)$ of S_2 express the status on convergence of the algorithm of sub-system 1, sub-system 5 and sub-system 2, respectively. The elements $S_2(2,1)$, $S_2(2,2)$ and $S_2(2,3)$ of S_2 express the 1st row status of stopping criterion matrices S_1, S_5 and S_2, respectively and they are set to 1 by *1st row event message* or reset to 0 by *1st row event break message*. Also we suppose that the elements $S_5(1,1)$, $S_5(1,2)$, $S_5(1,3)$ and $S_5(1,4)$ of S_5 express the status on convergence of the algorithm of sub-system 2, sub-system 3, sub-system 4 and sub-system 5, respectively. The elements $S_5(2,1)$, $S_5(2,2)$, $S_5(2,3)$ and $S_5(2,4)$ of S_5 express the first row status of stopping criterion matrices S_2, S_3, S_4 and S_5, respectively.

Initially suppose that S_2 and S_5 have all zero. Then let us explain how stopping criterion matrices S_2 and S_5 behave under the following fictional story:

Step 1 (The algorithm running on sub-system 1 converges) Sub-system 1 sends out *convergence event message*, which causes $S_2(1,1)$ to become 1 but this has no effect on S_5 as sub-system 1 it is not in $\mathcal{N}_5$.

Step 2 (The algorithm running on sub-system 2 converges) Sub-system 2 sends out *convergence event message*, which causes $S_2(1,3)$ to become 1 and the $S_5(1,1)$ to become 1.

Step 3 (The algorithm running on sub-system 5 converges) Sub-system 5 sends out *convergence event message*, which causes $S_2(1,2)$ to become 1 and $S_5(1,4)$ to become 1. At this time, $S_2(2,3)$ becomes 1 as all elements in the first row of S_2 are 1, it sends out *1st row event message*, which causes $S_5(2,1)$ to become 1.

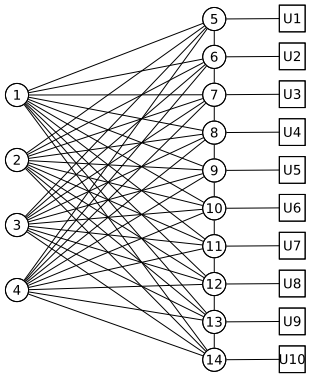

Fig. 4. Physical system used for simulation

Step 4 (The algorithm running on sub-system 3 converges) Sub-system 3 sends out *convergence event message*, which does not change S_2 as sub-system 3 is not in $\mathcal{N}_2$ but $S_5(1,2)$ becomes 1.

Step 5 (The algorithm running on sub-system 4 converges) Sub-system 4 sends out *convergence event message*, which does not change S_2 as sub-system 4 is not in $\mathcal{N}_2$ but $S_5(1,3)$ becomes 1. At this time, $S_5(2,4)$ becomes 1 as all elements in the first row of S_5 are 1, it sends out *1st row event message*, which causes $S_2(2,2)$ to become 1. At the same iteration, these sub-systems will receive *1st row event message*s from sub-systems 1, 2 and 4 as the algorithms running on all of their neighbors have converged, thus causing the remaining elements of their second row to become 1.

Step 7 Now every element in the last row of S_2 and S_5 are 1, the algorithms running on sub-systems 2 and 5 will stop optimizing as the whole graph has converged. The same events will happen to all the remaining sub-systems and they will stop their optimization process at iteration τ.

4. NUMERICAL SIMULATIONS

4.1 Problem setting of numerical simulation

We apply the proposed algorithm to the network system in Fig. 4.

We will run two algorithms: the distributed one with supervised stopping criterion and the distributed one with diffusion based stopping criterion.

Suppose that sub-systems $1-4$ are source nodes; sub-systems $5-14$ are storage node and users $U1-U10$ are demand. The dynamics of (1)-(3) are given in a similar way to the case of the example of Fig. 2. We consider Problem 2 for $x(0) = 0$, $N = 10$ and randomly created positive (semi-)definite Q and R matrices. For the design parameters of the algorithm, we set $\alpha_\tau = 0.05$ and $\varepsilon = 3$ as these values gave the best results for both precision and calculation time. In this situation, we consider to derive the optimal control input sequence under the model predictive control strategy, i.e., the receding horizon strategy.

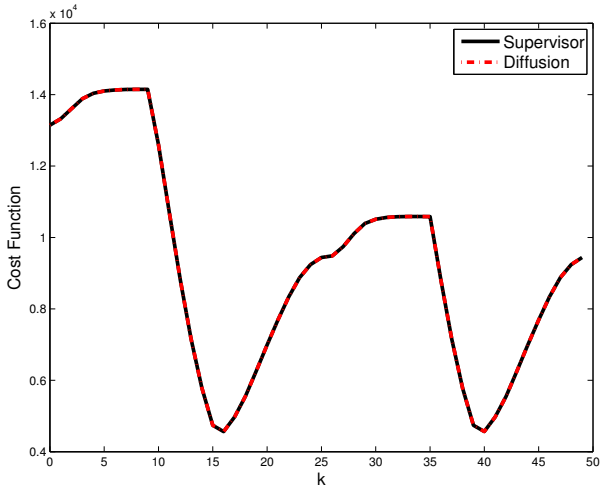

Fig. 5. Cost function comparison of two algorithms.

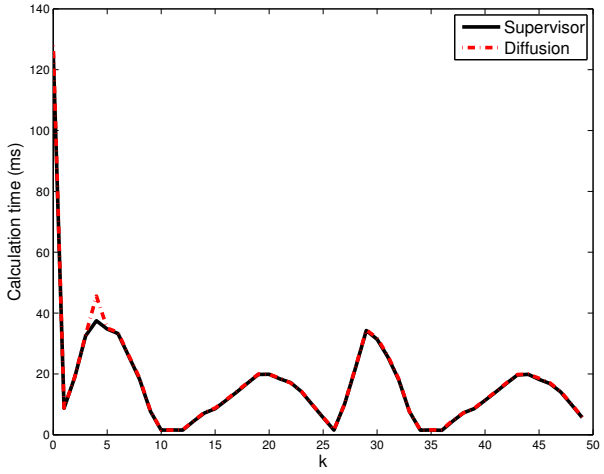

Fig. 6. Calculation time comparison of two algorithms.

4.2 Comparison on optimal cost value and calculation time

The resulting optimal cost value and calculation time at each time step in this case are shown in Figs. 5 and 6, respectively. As one can see in Fig. 5, there is no difference between supervised stopping criterion and diffusion based stopping criterion because the optimization parts are the same.

As for calculation time in Fig. 6, the calculation time shown for both algorithms is the time at which each stopping criterion determines the optimization algorithm has converged. As one can see, the use of diffusion based stopping criterion does not add any significant overhead. The difference is probably related to small numerical errors caused by computation time measurement functions in MATLAB or possible computer load change.

Table 1 shows the sum of maximum calculation times for the whole simulation excluding the time step 0 which is the initialization part where all dual variable vectors are zero initially. We believe the difference between supervisor and diffusion criteria is from numeric errors caused by computation time measurement functions. In this table, 2 node refers to the system in Fig. 1, 5 node refers to the system in Fig. 2 and 14 node refers to the system in Fig. 4. As one can see, number of neighbors of sub-systems is a factor for the optimization time but graph

Table 1. Total calculation times of different size systems

	2 node	5 node	14 node
Supervisor	241.79 ms.	880.05 ms.	672.33 ms.
Diffusion	234.76 ms.	883.26 ms.	704.78 ms.

complexity is also a factor. We believe that the reason 5 node case takes longer to optimize is $U1$ and $U4$ are connected to multiple sub-systems which cannot be reduced to a constraint that can be represented by (2).

5. CONCLUSION

This paper has proposed a new type of stopping criterion, called here diffusion based stopping criterion, for distributed optimization, where we can determine in a distributed way whether or not the optimization algorithm converges. This approach increases the robustness of the distribution optimization and decreases the communication cost as it does not require extra communication lines between a supervisor and sub-systems.We have showed that there is no time difference between supervised and diffusion based stopping criteria by running simulations of the dispatch control problem for both algorithms and comparing the results.

REFERENCES

Boyd, S., Xiao, L., Mutapcic, A., and J., M. (2008). Notes on decomposition methods. Available at `http://see.stanford.edu/materials/lsocoee364b/08-decomposition_notes.pdf`.

Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1), 1–122.

Giselsson, P. and Rantzer, A. (2010). Distributed model predictive control with suboptimality and stability guarantees. In *Proceedings of the 49th IEEE Conference on Decision and Control*, 7272–7277.

Johansson, B., Keviczky, T., Johansson, M., and Johansson, K. (2008). Subgradient methods and consensus algorithms for solving convex optimization problems. In *Proceedings of the 47th IEEE Conference on Decision and Control*, 4185–4190.

Ma, Y., Anderson, G., and Borrelli, F. (2011). A distributed predictive control approach to building temperature regulation. In *American Control Conference (ACC)*, 2089–2094.

Nedic, A. and Ozdaglar, A. (2009). Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54(1), 48–61.

Nedic, A., Ozdaglar, A., and Parrilo, P. (2010). Constrained consensus and optimization in multi-agent networks. *IEEE Transactions on Automatic Control*, 55(4), 922–938.

Ono, M. and Williams, B. (2010). Decentralized chance-constrained finite-horizon optimal control for multi-agent systems. In *Proceedings of the 49th IEEE Conference on Decision and Control*, 138–145.

Wei, E., Ozdaglar, A., and Jadbabaie, A. (2010). A distributed newton method for network utility maximization. In *Proceedings of the 49th IEEE Conference on Decision and Control*, 1816–1821.

Zhu, M. and Martinez, S. (2012). On distributed convex optimization under inequality and equality constraints. *IEEE Transactions on Automatic Control*, 57(1), 151–164.