

# A Structured Optimal Controller with Feed-Forward for Transportation

Martin Heyden, Richard Pates and Anders Rantzer\*

January 31, 2022

## Abstract

We study an optimal control problem for a simple transportation model on a path graph. We give a closed form solution for the optimal controller, which can also account for planned disturbances using feed-forward. The optimal controller is highly structured, which allows the controller to be implemented using only local communication, conducted through two sweeps through the graph.

## 1 Introduction

In this paper we study a simple Linear Quadratic control transportation problem on a network. Such problems have well known solutions based on the Riccati equation [?]. This gives a static feedback law

$$u = Kx,$$

where  $u$  is the input to the system,  $x$  is the state of the system and  $K$  is a matrix with real entries. This matrix is in general dense. This is undesirable in large-scale problems, since it implies that measurements from the entire network are required to compute the optimal inputs at every node. Furthermore a centralized coordinator with knowledge of the entire system is required to determine the matrix  $K$ , and a complete redesign will be required in response to any changes in the network.

These factors have led to the development of a range of general purpose methods for structured control system design. Some notable themes include the notion of Quadratic Invariance [1, 2], System Level Synthesis [3], and the use of large-scale optimization techniques (e.g. [4]). A downside with these approaches is that they improve scalability

---

\*This work was supported by the Swedish Foundation for Strategic Research through the project SSF RIT15-0091 SoPhy.

The authors are members of the LCCC Linnaeus Center and the ELLIIT Excellence Center at Lund University.

The authors are with the Department of Automatic Control, Lund University, Box 118, SE-221 00 Lund, Sweden.

© 2021 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

at the expense of performance. That is they search over families of controllers that exclude the dense optimal controller for (2). While in comparison with the alternative this may be an acceptable trade-off, it implicitly assumes that just because the feedback law is dense, it cannot be efficiently implemented.

The main result of this paper is to show that the simple structure in our problem allows the optimal control law to be computed and implemented in a simple and scalable manner. The resulting control actions are the same as those from a Riccati approach, and could in principle be calculated that way. However, there are extra structural features in the control law that are obscured by the resulting dense feedback matrix representation, and it is not obvious how to exploit these to give a scalable implementation from the gain matrix obtained from the Riccati equation.

## 1.1 Problem Formulation

We consider the problem of transportation and production of goods on a directed path graph with vertices  $v_1, v_2, \dots, v_N$  and directed edges  $(e_N, e_{N-1}), \dots, (e_2, e_1)$ . The dynamics are given by

$$z_i[t+1] = z_i[t] - u_{i-1}[t] + u_i[t - \tau_i] + v_i[t] + d_i[t]. \quad (1)$$

All the variables are considered to be defined relative to some equilibrium. In the above  $z_i[t] \in \mathbb{R}$  is the quantity in node  $i$  at time  $t$ . The system can be controlled using the variables  $u_i[t] \in \mathbb{R}$  and  $v_i[t] \in \mathbb{R}$ . The variable  $u_i[t]$  denotes the amount of the quantity that is transported from node  $i+1$  to node  $i$  (again relative to some equilibrium flows), and the transportation takes  $\tau_i$  time units. For the last node  $N$  it is assumed that  $u_N[t] = 0$  for all  $t$ . The variable  $v_i[t]$  denotes the flexible production or consumption of the quantity at the  $i$ th node. Finally  $d_i[t] \in \mathbb{R}$  is the fixed production/consumption at the  $i$ th node. This will be treated like a forecast, or *planned disturbance*, that is known to the designer, but cannot be changed. This model could for instance describe a water irrigation network [5] or a simple supply chain system [6]. A state-space representation for (1) can be obtained by setting  $z_i[t], u_i[t - \delta]$ ,  $1 \leq \delta \leq \tau_i$  to equal the system state.

The goal is to optimally operate this network around some equilibrium point. The performance is measured by the cost of deviating from the equilibrium levels  $q_i z_i^2$  and the cost of the variable production  $r_i v_i^2$ , where  $q$  and  $r$  are strictly positive constants. We thus consider the following linear quadratic control problem on a graph with  $N$  nodes,

$$\begin{aligned} & \underset{z, u, v}{\text{minimize}} && \sum_{t=0}^{\infty} \sum_{i=1}^N (q_i z_i[t]^2 + r_i v_i[t]^2) \\ & \text{subject to} && \text{dynamics in (1)} \\ & && z[0], d_i[t]. \end{aligned} \quad (2)$$

Note that there is no penalty on the internal flows  $u_i$ . This can for example be motivated by the transportation costs already being covered by the costs of the nominal flows (or in the case of water irrigation networks that gravity does the moving). This problem is in effect a dynamic extension of the types of scheduling problems considered in transportation networks [7], and could be used to compliment such approaches by optimally adjusting a nominal schedule in real time using the feedback principle.

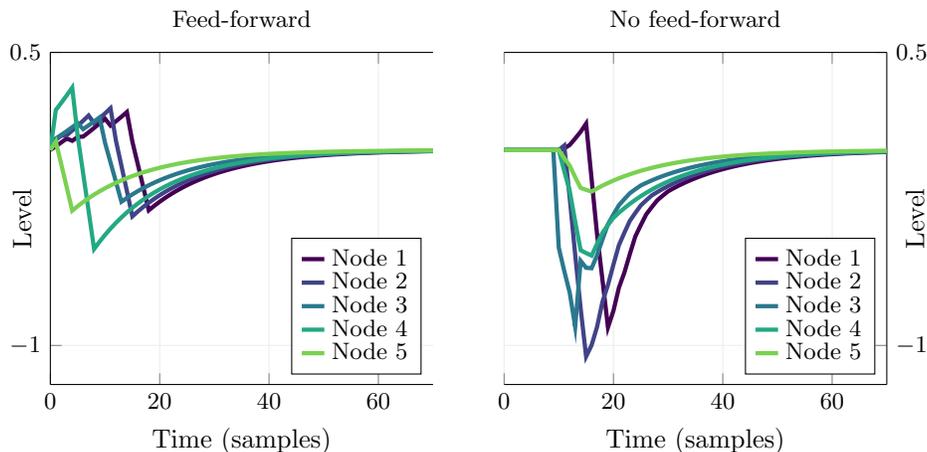


Figure 1: Example of the effect of feed forward. The graph has five nodes and transportation delay  $\tau_1 = 3$ ,  $\tau_2 = 2$ ,  $\tau_3 = 5$  and  $\tau_4 = 4$ . There is a disturbance in node three from time 10 to 13 and in node 2 from time 12 to 15. We can see that the feed-forward manages to handle the disturbances better by spreading out their effect throughout the graph. To quantify the difference one can consider the cost in (2), which is 3.11 with feed-forward and 11.35 without feed-forward.

A similar problem has been studied in a previous paper [8]. However we give several important extensions, in that we allow for non-homogeneous delays, production in every node and optimal feed-forward for planned disturbances. Allowing for non homogeneous delays is important as that will be the case for almost all applications. Taking planned disturbances into account allows for increased performance whenever such disturbances can be forecast. Finally, allowing for some variation in the consumption  $v_i$  for each node will also generally increase performance whenever such variation is possible. The effect the feed-forward of planned disturbances can have on the controller performance is illustrated in Figure 1, where we see that the controller with feed-forward anticipates the action of the disturbances, allowing the effect to be better spread through the graph and the node levels to be more tightly regulated. This results in a significant improvement in performance.

## 1.2 Result Preview

The key structural feature that we identify in the optimal control law for (2) is that optimal inputs can be computed recursively by two sweeps through the graph (even though the control law that would be obtained from the Riccati equation would be dense). More specifically, two intermediate variables local to the  $i$ th node  $\delta_i[t]$  and  $\mu_i[t]$  can be computed recursively through relationships on the form

$$\begin{aligned}\delta_i[t] &= f(\text{local\_variables}, \delta_{i-1}), \\ \mu_i[t] &= g(\text{local\_variables}, \mu_{i+1}),\end{aligned}$$

from which the optimal inputs  $u_i[t]$  and  $v_i[t]$  can be calculated based only on local variables. Conceptually this step is rather similar to solving a sparse system of equations with the structure of a directed path graph using back substitution.

The details are given in [Algorithm 2](#), and this process is illustrated in [Figure 2](#). This allows the optimal inputs to be computed by sweeping once through the graph from the first node to the final node to compute the  $\delta$ 's, and once from the final node to the first to compute the  $\mu$ 's. Both sweeps can be conducted in parallel. This represents a sort of middle ground between centralised control and decentralised control, in which global optimality is preserved whilst only requiring distributed communication. The price for this is that the sweep through the whole graph must be completed before the inputs can be applied, but for systems with reasonably long sample times (which is likely true in transportation or irrigation systems) this seems a modest price to pay. Interestingly the controller parameters can be computed in a similar distributed manner, allowing the controller to also be synthesised in a simple and scalable manner. This is shown in [Algorithm 1](#).

## 2 Results

In this section we present two algorithms that together allow for the solution of (2). The first of these algorithms computes the parameters of a highly structured control law for solving (2), whereas the second shows that the control law has a simple distributed implementation. These features will be discussed in [Section 3](#). In this section we will demonstrate that under suitable assumptions on the planned disturbances  $d_i[t]$ , [Algorithms 1](#) and [2](#) give the optimal solution to (2). This constitutes the main theoretical contribution of the paper.

In the absence of the planned disturbances (i.e. with  $d_i[t] \equiv 0$ ), (2) is an infinite horizon LQ problem in standard form. It is of course highly desirable in applications to be able to include information about upcoming disturbances in the synthesis of the control law. However if we are given an infinite horizon of disturbances, (2) is no longer tractable. For the theoretical perspective, it turns out that the suitable assumption on the horizon length is as follows:

**Assumption 1.** *Let the aggregate delay  $\sigma_k$  in (1) be*

$$\sigma_k = \sum_{i=1}^{k-1} \tau_i.$$

*Given a horizon length  $H \geq 0$ , assume that  $d_i[t] = 0$  for all  $t > H + (\sigma_N - \sigma_i)$  and for all  $1 \leq i \leq N$ .*

Observe that if  $d_i[t] = 0$  for all  $t > H$  then [Assumption 1](#) holds. Thus the assumption captures the natural notion of having a finite horizon  $H$  of information about the disturbances  $d_i[t]$  available when constructing the control input. In [Section 4](#) we will investigate how the length of the horizon affects the performance of the controller.

We will now state the main results of this paper. The following theorem shows that (2) can be solved by running two simple algorithms; one for calculating all the

necessary parameters, and one for computing the optimal inputs. Both algorithms can be implemented using only local communication as discussed in [Section 3](#). A graphical illustration of the implementation of [Algorithm 2](#), which is the algorithm used for the on-line implementation, can be found in [Figure 2](#).

**Theorem 1.** *Let*

$$D_i[t] = \sum_{j=1}^i d_j[t - \sigma_j].$$

*Assume that  $H$  and  $d_j[t]$  satisfy [Assumption 1](#). Then the optimal inputs  $u_i[t]$  and  $v_i[t]$  for the problem in (2) are given by running [Algorithm 2](#) with the parameters calculated by [Algorithm 1](#).*

*Proof.* See the appendix. A sketch of the proof can be found in [Section 5](#). □

**Remark 1.** *In most cases the choice of  $H$  can be made without considering its effect on the controller implementation, and can instead be chosen based only on the nodes' ability to forecast their disturbances. In applications it would also be natural to incorporate new information on upcoming disturbances in a receding horizon fashion. This will be further discussed in [Section 3.2](#).* ◇

**Remark 2.** *There is an asymmetry in [Assumption 1](#) in that  $(\sigma_N - \sigma_i)$  grows as  $i$  decreases from  $N$  to 1. This means that this assumption allows nodes further down the graph to have longer horizons of planned disturbances. Of course there is no reason to believe that these nodes are better at predicting their disturbances. It is just that the derived theory can handle those disturbances in a straightforward manner since the optimal controller lumps the disturbances into time shifted sums, with a time shift proportional to  $\sigma_i$ .* ◇

### 3 Implementation

In this section we will discuss the structure in [Algorithms 1](#) and [2](#), and explain how they can be used to implement an optimal feedback control law for solving (2) in a distributed manner. In both cases the order in which the computations occur is highly structured. This is illustrated for [Algorithm 2](#), which is the algorithm that must be run to compute the control inputs, in [Figure 2](#). Matlab code for using these algorithms to calculate the optimal control inputs is available at [github](#)<sup>1</sup>, as well as code to verify that [Theorem 1](#) holds numerically. We will also discuss how to calculate  $D_i$  and incorporate updates to the planned disturbances in a receding horizon style in [Algorithm 3](#).

#### 3.1 Algorithms 1 and 2, and the Optimal Control Law

The problem in (2) is at its heart an LQ problem, and the optimal controller is given by a static feedback law. The corresponding feedback matrix is generally dense, and that is the case for (2) as well. However certain special structural features of the process

<sup>1</sup><https://github.com/Martin-Heyden/cdc-letters-2021>

---

**Algorithm 1:** Computation of control parameters.

---

**Input:**  $q_i, r_i, \tau_i, H$   
**Output:**  $\gamma_i, g_i(j), P_i(\tau_i, 1), h_i, \phi_i(\Delta), a_i, c_i$

---

```

/* First Sweep, upstream direction */
1  $\gamma_1 = q_1, \quad \rho_1 = r_1$  // initialize first node
2 send  $\gamma_1$  and  $\rho_1$  to upstream neighbor
3 for node  $i = 2:N$  do
4    $\gamma_i = \frac{\gamma_{i-1}q_i}{\gamma_{i-1}+q_i}, \quad \rho_i = \frac{\rho_{i-1}r_i}{\rho_{i-1}+r_i}$ 
5   send  $\gamma_i$  and  $\rho_i$  to upstream neighbor
6 end
.....
/* Second Sweep Downstream direction */
7  $X_N(H+2) = -\frac{\gamma_N}{2} + \sqrt{\gamma_N\rho_N + \frac{\gamma_N^2}{4}}$ 
8 for node  $i = N:1$  do
9    $X_i(\tau_i) = \frac{\rho_i(X_{i+1}(1)+\gamma_i)}{X_{i+1}(1)+\gamma_i+\rho_i}$  // Not for node N
10   $X_i(t-1) = \frac{\rho_i(X_i(t)+\gamma_i)}{X_i(t)+\gamma_i+\rho_i}, \quad // 1 \leq t-1 \leq \tau_i-1$  or for  $i=N, 1 \leq t-1 \leq H+1$ 
11   $g_i(i) = \frac{X_i(i)}{X_i(i)+\gamma_i}, \quad 2 \leq i \leq \tau_i$ 
12   $g_{i+1}(1) = \frac{X_{i+1}(1)}{X_{i+1}(1)+\gamma_i}$ 
13   $b_i = g_{i+1}(1) \prod_{j=2}^{\tau_i} g_i(j)$ 
14  send  $X_i(\tau_i), b_i$  to downstream neighbor.
    // for  $1 \leq l, m \leq \tau_i$ 
15   $P_i(1, m) = \frac{X_i(1)}{\rho_i}$ 
16   $P_i(l, m) = (1 - \frac{X_i(l)}{\rho_i})g_i(l)P_i(l-1, m) + \frac{X_i(l)}{\rho_i}, \quad l \leq m$ 
17   $P_i(l, m) = (1 - \frac{X_i(l)}{\rho_i})P_i(l-1, m) + \frac{X_i(l)}{\rho_i}, \quad l > m$ 
18 end
.....
/* Third Sweep, upstream direction */
19  $h_1 = P_1(\tau_1, \tau_1)g_2(1)$ 
20 send  $h_1$  to upstream neighbor.
21 for node  $i = 2:N-1$  do
22    $h_i = (1 - P_i(\tau_i, 1))b_i h_{i-1} + P_i(\tau_i, \tau_i)g_{i+1}(1)$ 
23   send  $h_i$  to upstream neighbor.
24 end
.....
/* Some final local Calculations */
// For  $1 \leq \Delta \leq \tau_i$ , Empty Product,  $\prod_{j=2}^1 = 1$ 
25  $\phi_i(\Delta) = \left(1 - P_i(\tau_i, \Delta) - (1 - P_i(\tau_i, 1))h_{i-1} \prod_{j=2}^{\Delta+1} g_k(j)\right)$ 
26  $a_i = \frac{X_i(1)}{r_i} + \frac{\gamma_i}{q_i} \left(1 - \frac{X_i(1)}{\rho_i}\right)$ 
27  $c_i = -\left(\frac{X_i(1)}{r_i} - \frac{\gamma_i X_i(1)}{q_i \rho_i}\right) (1 - h_{i-1}) + \frac{\gamma_i}{q_i} h_{i-1}$ 

```

---

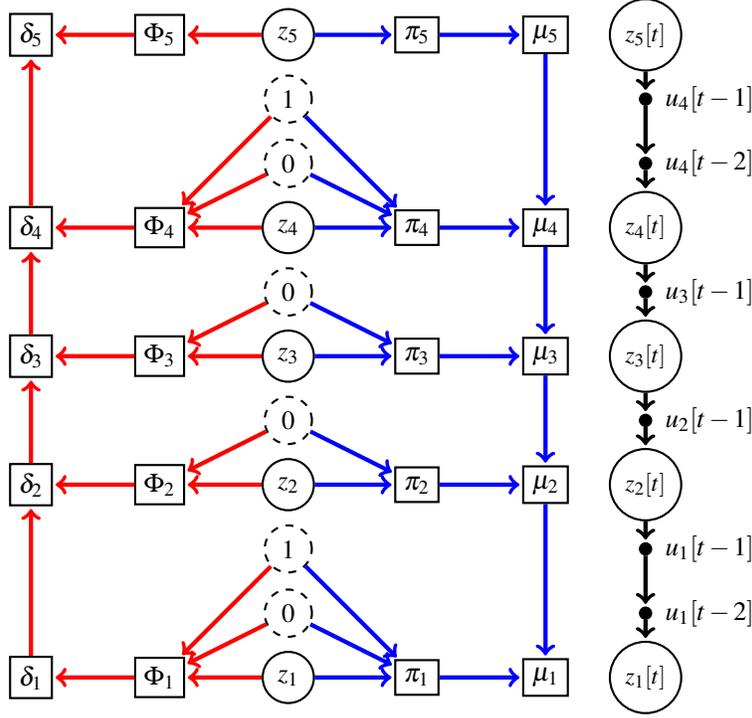


Figure 2: Illustration of the structured approach for calculating the optimal inputs using Algorithm 2, for a 5 node example with  $\tau_1 = 2$ ,  $\tau_2 = 1$ ,  $\tau_3 = 1$ ,  $\tau_4 = 2$ . The graph at the right of the figure illustrates the underlying dynamics of the network as in (1). The left part of the figure illustrates the structure of the computations required to compute the optimal control according to Algorithm 2. The solid circles corresponds to node states and dashed circles the quantities in transit. The number in each dashed circle denotes the value of  $\Delta$ , which then maps to  $u_k[t - (\tau_k - \Delta)]$ . The rectangles indicates the different intermediate calculations needed to determine the variables required to compute the optimal inputs (lines 2–3 and 7–8 in Algorithm 2). These are horizontally aligned with the location in the network where they could be locally performed. The arrows indicate information flow. Each intermediate can be calculated using only the quantities from the incoming arrows. An upstream sweep is performed (the red arrows) in order to calculate the variables  $\delta_i[t]$ . The local intermediate  $\Phi_i[t]$  variables are calculated (line 2), and then aggregated into the  $\delta_i[t]$ 's (line 3), which are sequentially passed up the graph. For the downstream sweep the local  $\pi_i[t]$  variables are calculated (line 7) and aggregated into the  $\mu_i[t]$ 's (line 8). Both sweeps can be conducted in parallel, and once they have completed, the optimal inputs for the  $i$ th node can be determined using the variables at the  $i$ th location according to lines 11 and 12 in Algorithm 2.

---

**Algorithm 2:** Distributed Controller Implementation.

---

**Input:**  $z_i[t], u_i[t - (\tau_i - \Delta)], d_i[t], D_i[t + \sigma_i + \Delta]$   
**Output:**  $u_i[t], v_i[t]$

---

```

// Let  $\tau_N = H + 1$ .
/* Upstream sweep - Done in parallel with downstream sweep */
1 for node  $i = 1:N$  do
2    $\Phi_i[t] = \phi_i(1)z_i[t] +$ 
       $\sum_{\Delta=0}^{\tau_i-1} \phi_i(\Delta + 1) \left( u_i[t - (\tau_i - \Delta)] + D_i[t + \sigma_i + \Delta] \right)$ 
3    $\delta_i[t] = \Phi_i[t] + (1 - P_i(\tau_i, 1))\delta_{i-1}[t]$ 
4   send  $\delta_i[t]$  upstream
5 end
.....
/* Downstream sweep - Done in parallel with upstream sweep */
6 for node  $i = N:1$  do
7   // Empty Product,  $\prod_{j=2}^1 = 1$ 
8    $\pi_i[t] = z_i[t] + \sum_{\Delta=0}^{\tau_i-1} \left( u_i[t - (\tau_i - \Delta)] + D_i[t + \sigma_i + \Delta] \right) \prod_{j=2}^{\Delta+1} g_i(j)$ 
9    $\mu_i[t] = \pi_i[t] + b_i\mu_{i+1}[t]$ 
10  send  $\mu_i[t]$  downstream
11 end
.....
/* Calculate outputs */
12  $u_{i-1}[t] = (1 - \frac{\gamma_i}{q_i}) \left( z_i[t] + u_i[t - \tau_i] + D_i[t + \sigma_i] \right)$ 
       $- a_i\delta_{i-1}[t] + c_i\mu_i[t] + d_i[t] - D_i[t + \sigma_i]$ 
13  $v_i[t] = -\frac{X_i(1)}{r_i} \left( \delta_{i-1}[t] + (1 - h_{i-1})\mu_i[t] \right)$ 

```

---

(1) are inherited by the optimal control law. It is these features we exploit to give a scalable implementation in [Algorithms 1](#) and [2](#), which we will now discuss.

In terms of the algorithm variables, the optimal node production  $v_i[t]$  for (2) is given by

$$v_i[t] = -\frac{X_i(1)}{r_i} \left( \delta_{i-1}[t] + (1 - h_{i-1})\mu_i[t] \right), \quad (3)$$

and the optimal internal flows  $u_i[t]$  are given by

$$u_{i-1}[t] = (1 - \frac{\gamma_i}{q_i}) \left( z_i[t] + u_i[t - \tau_i] + D_i[t + \sigma_i] \right) - a_i\delta_{i-1}[t] + c_i\mu_i[t] + d_i[t] - D_i[t + \sigma_i]. \quad (4)$$

The parameters in these control laws (the symbols without a time index, which includes  $X_i(1)$ ) are calculated in a simple and structured manner by [Algorithm 1](#). Of course having an efficient method for computing the control law is less critical than having an efficient real time implementation of the control law (which is performed by [Algorithm 2](#)), since the control law can be computed ahead of time. However the fact that this step is also highly structured indicates that the approach is scalable, since

it allows for the the control law to be simply and efficiently updated in response to changes to the dynamics in (1) (perhaps resulting from the introduction of more nodes).

**Algorithm 1** computes all the parameters needed to give a closed form solution for the problem in (2). The origin of the parameters in **Algorithm 1** are discussed briefly in the proof idea in **Section 5** and full details are found in the proof in the appendix. The algorithm consists of three serial sweeps. The first sweep starts at node 1 and calculates  $\gamma_i$  and  $\rho_i$ . The second sweep starts at node  $N$ , and calculates  $X_i(t)$ . The calculation of  $X_i(t)$  has both local steps (line 10) and steps that requires communication (line 9). Also during the second sweep, the parameters  $g$ ,  $b$  and  $P$  are calculated locally. The third sweep starts at node 1 again, and calculates the parameter  $h$ , which is needed to calculate the optimal production. Finally, after the third sweep, the parameters  $\phi_i(\Delta)$ ,  $a_i$  and  $c_i$  are calculated in each node independently.

The real time implementation of the optimal control law also has a simple distributed implementation. This is the role of **Algorithm 2**, and the structure of the implementation is illustrated in **Figure 2**. The algorithm proceeds through two sweeps through the graph. These sweeps are independent of one another, and can be conducted in parallel. In the upstream sweep (from node 1 to node  $N$ ), a set of local variables ( $\Phi_i[t]$  and  $\delta_i[t]$ ) are computed according to lines 2–3. This is done sequentially, since the computation of  $\delta_i[t]$  depends on  $\delta_{i-1}[t]$ .  $\delta_{i-1}[t]$  then gives all the information node  $i$  needs from downstream nodes. Similarly the downstream sweep sequentially computes the  $\pi_i[t]$  and  $\mu_i[t]$  variables. Here  $\mu_i[t]$  gives all the information needed from nodes upstream of node  $i$ . Once these two sweeps are completed, the optimal inputs can be calculated locally using lines 11 and 12.

### 3.2 Receding Horizon and Calculation of $D_i[t]$

We will now discuss how to implement the controller in a receding horizon style to account for updates and new information about the planned disturbances  $d_i[t]$ . In terms of both the optimal control problem in (2) and the controller implementation in **Algorithm 2**, the planned disturbances are treated as fixed quantities, that are known up to some horizon length  $H$  into the future (and equal to zero thereafter, c.f. **Assumption 1**). The idea is that  $d_i[t]$  determines the anticipated consumption of the quantity at node  $i$  and time  $t$ . Having this information available ahead of time allows the optimal control law to anticipate the predicated usage, and optimally 'schedule' the transportation of the quantity through the network. As we will see in the examples this can lead to a significant improvement in performance. However, in practice we would want to update the values of the  $d_i[t]$ 's as time passes, and more up to date information becomes available.

A natural way to do this is to use a receding horizon approach. In this setting we assume that at each point in time, we essentially have a fresh problem, with a new set of planned disturbances. **Algorithm 2** can then be used to compute the first optimal input for this problem, after which the problem resets, and we get a new horizon of planned disturbances. This ensures we always make the best action available to us with a given horizon of information about the disturbances. The question is then, how to efficiently update the part of the control law that depends on the planned disturbances.

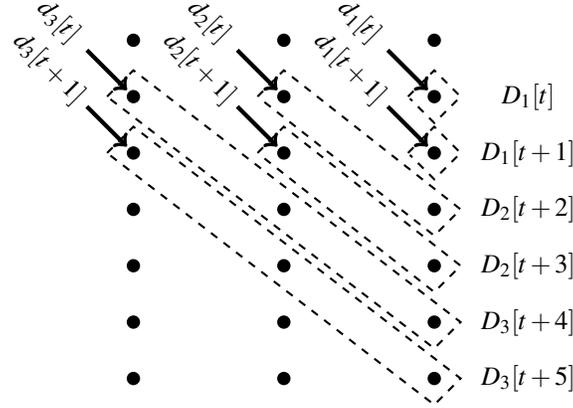


Figure 3: Illustration for the terms included in  $D_i[t]$  for a three node graph with  $\tau_1 = \tau_2 = 2$ . The first row of dots corresponds to  $z_3[t], z_2[t], z_1[t]$  and the second corresponds to  $z_3[t+1], z_2[t+1], z_1[t+1]$ , and so on. Due to lack of space only  $d_i[t]$  and  $d_i[t+1]$  has been drawn. However, the pattern follows through the graph. From the figure we can see that  $D_1[t+3] = D_2[t+3] - d_2[t]$ , corresponding to the first part of [Algorithm 3](#).

The changes required to accommodate this are rather minor. The planned disturbances do not affect the control parameters or the distributed structure of the implementation of the control law. To see this, observe from [Algorithms 1](#) and [2](#) that all of the information about the planned disturbances is handled through the variables  $D_i[t]$  defined in [Theorem 1](#). An illustration of the relationship between individual disturbance  $d_i$  and shifted disturbance vectors  $D_i[t]$  can be found in [Figure 3](#). Thus the structure of the implementation of the control law remains the same, and only  $D_i[t]$  needs to be updated as new information become available. This can be done efficiently by a sweep starting at the bottom of the graph. After all, although in the receding horizon framework we assume we have a ‘new’ set of planned disturbances at each point in time, these will share a large amount of information with the planned disturbances from the previous time step. This is the role of [Algorithm 3](#), which we now explain.

All the  $D_i[t]$ ’s where none of the underlying  $d_j[t]$  were changed can easily be updated. For  $1 \leq \Delta \leq \tau_i - 1$  the information in  $D_i[t + \sigma_i + \Delta]$  will be useful in node  $i$  at the next time step as

$$D_i[t + 1 + \sigma_i + \Delta - 1] = D_i[t + \sigma_i + \Delta].$$

When  $\Delta = 0$  the information can be used at the downstream neighbor as  $D_i$  satisfies

$$D_{i-1}[t + 1 + \sigma_i - 1] = D_i[t + \sigma_i] - d_i[t].$$

These updates can be done in all nodes simultaneously and the time it takes is thus independent of the size of the graph.

However, the shifted sums  $D_i$  has to be initialized at time zero, and also updated when new disturbances  $d_i$  are planned for times  $t > 0$ .  $D_i[t]$  only requires information

---

**Algorithm 3:** Calculation of  $D$ 

---

```
Input: changed  $d_i[s]$ 
Output: updated  $D_i$ 
/* Update Disturbances in parallel,  $\mathcal{O}(1)$ . Necessary even if there
are no new disturbances */
1 for node  $i = N:1$  do
2   | Send  $D_{i-1}[t + 1 + \sigma_i - 1] = D_i[t + \sigma_i] - d_i[t]$  downstream.
3   | Discard  $D_i[t + \sigma_i]$ 
4 end
.....
/* Update Disturbances due to new planned disturbances */
5 for node  $i = 1:N$  do
6   | /* For  $t + \sigma_i \leq s < t + \sigma_N + H$  */
7   | if  $d_i[s - \sigma_i]$  changed or  $D_{i-1}[s]$  received then
8   |   | send  $D_i[s] = D_{i-1}[s] + d_i[s - \sigma_i]$  upstream
9   | end
10 end
```

---

from downstream, and can thus be calculated by a sweep starting at node one and going upstream. Starting at the first node, any of the  $d_1[s]$ ,  $t \leq s \leq t + \sigma_N + H$  that have changed are sent to node two. Then for every node  $i$ , the aggregate  $D_i[s]$ ,  $t + \sigma_i \leq s \leq t + \sigma_N + H$  is sent upstream if it has changed. This will be the case if node  $i$  received  $D_{i-1}[s]$ ,  $t + \sigma_i \leq s \leq t + \sigma_N + H$  from its downstream neighbor, or if  $d_i[s]$ ,  $t \leq s \leq \sigma_N - \sigma_i + H$  has changed.

The steps for updating the disturbances are summarized in [Algorithm 3](#). While the algorithm is essentially a sweep through the graph in the upstream direction, it might be best to not implement it in the upstream sweep of [Algorithm 2](#) as then the downstream sweep would have to be done after the upstream sweep, due to its need for the shifted disturbance vectors. On the other hand, the calculation does not rely on measurement from the system, and can thus be carried out either before or after [Algorithm 2](#).

We are now ready to discuss how the choice of  $H$  affect the implementation of the controller. Firstly, a larger  $H$  will lead to a very slight increase in the synthesis time due to more iterations of  $X_N(t)$  being required. Secondly increasing  $H$  will increase the memory requirement in node  $N$ , in that it requires to store  $D_N[t + \Delta]$  for  $0 \leq \Delta \leq H$ . Finally the requirement for the communication bandwidth when updating  $D_i[t]$  will depend on the number of new disturbances  $d_j[t]$ , but is upper bounded by  $H$  if  $d_i[t] = 0$  for  $t > H$  and by  $H + \sigma_N$  if  $d_i[t] = 0$  for  $t > H + \sigma_N - \sigma_i$ . Thus if the bandwidth is limited, and a lot of new disturbances are expected to be planned, one might need to limit the size of  $H$ . Otherwise it can be freely chosen based on the nodes abilities to forecast disturbances.

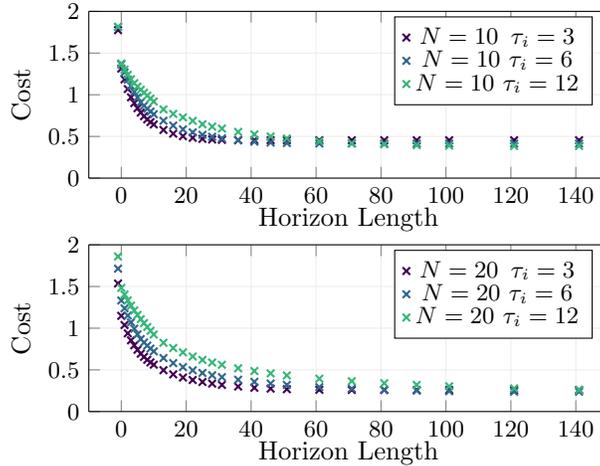


Figure 4: Simulations comparing the effect the planning horizon has on the performance. The data-points with highest cost for each configuration corresponds to not using the planned disturbances at all. While the the rest corresponds to using all planned disturbances announced up to  $H$  time units ahead in every node.

## 4 Simulations

In this section we explore the effect the feed-forward horizon has on the controller performance through simulations. In Figure 4 the performance for different horizon lengths is shown. Two random nodes are affected by disturbances of total size between minus one and zero and during a time interval of length between 1 and 5. The node level cost is given by  $q_i = 1$ . The production cost is given by  $r_i = 10N$ , where  $N$  is the number of nodes. This is an attempt to keep the production cost similar for different values of  $N$ . There are 50 simulations done for each case with random disturbances as previously described. For all the cases when  $N$  is the same, all the disturbances are the same for all the different horizons and delay values. The horizon lengths are the same for all nodes, i.e it is assumed that  $d_i[t + d] = 0$  for  $d > H$ .

We can see that a large part of the performance increase form having feed-forward can be achieved for short disturbance horizons. We can also see that for larger delays, and for more nodes, a longer horizon is needed to get the same effect. As a rule of thumb, at least for this example, it seems like a horizon longer than  $2/3$  of the total delay gives almost no effect, and even a horizon of  $1/3$  of the total delay gives most of the performance increase.

## 5 Proof Idea

In this section we will describe the main idea behind the technique used to derive the results, which is to study a time shifted sum of the node-levels  $z_i$ , and a time shifted sum of the production  $v_i$ . This will allow the problem to be solved in terms of these shifted

sums, essentially reducing it to a problem with scalar variables. Outside the disturbance horizon the problem can be solved by a Riccati equation in one variable. While inside the disturbance horizon the problem is solved using dynamic programming, where each step has scalar variables.

Now for the definitions of the shifted sums, let the sum of a shifted level  $S_k$  and sum of a shifted production vector  $V_k$  be defined as

$$S_k[t] = \sum_{i=1}^k z_i[t - \sigma_i], \quad V_k[t] = \sum_{i=1}^k v_i[t - \sigma_i].$$

Also Let  $\bar{V}_k[t] = V_k[t] + D_k[t]$  to shorten some expressions.

We illustrate the main idea by considering these shifted sums with a short example. Consider a path graph with  $N > 2$ ,  $\tau_1 = 1$ , and  $\tau_2 > 1$ . Then  $S_2[3] = z_1[3] + z_2[2]$ . It can be checked that

$$S_2[3] = z_1[0] + z_2[0] + \bar{V}_1[0] + \bar{V}_2[1] + \bar{V}_2[2] + u_1[-1] + u_2[-\tau_2].$$

Note that the internal transportation  $u_1[0]$  and  $u_1[1]$  have canceled, and the sum is thus independent of the internal transportation  $u$  (except those with negative time index, which correspond to initial conditions). Also note that any values for  $z_2[2]$  and  $z_1[3]$  can be achieved as long as  $z_1[3] + z_2[2] = S_2[2]$ . This follows from that  $z_2[2]$  can take any value by choosing the appropriate value for  $u_1[1]$ . Thus the cost of  $q_2 z_2[2]^2 + q_1 z_1[3]^2$  only depends on the value of  $S_2[3]$ , which is independent of all internal transportation  $u_i[t]$ ,  $t \geq 0$ . This means that all inputs except  $u_1[1]$  can ignore its effect on the terms in  $S_2[3]$ . Furthermore,  $S_2[3]$ , and thus the corresponding cost, only depends on the sums  $V_2[1]$  and  $V_2[2]$  and not the individual productions  $v_1[1]$ ,  $v_1[2]$ ,  $v_2[0]$  and  $v_2[1]$ .

This idea can be generalized. The cost function can be rewritten in terms of shifted levels, where each shifted level sum is independent of the internal transportation. Each shifted level can thus be minimized independently with respect to the internal transportation  $u$ . Just as in the example, the only constraint is that the shifted sum has the correct value. The optimal cost for a shifted vector  $S_k[t]$  is given by the solution to

$$\begin{aligned} & \underset{z_i}{\text{minimize}} && \sum_{i=1}^k q_i z_i[t + \sigma_k - \sigma_i]^2 \\ & \text{subject to} && S_k[t + \sigma_k] = c, \end{aligned} \tag{5}$$

where  $c$  depends on the initial conditions,  $V_i$ , and  $D_i$ . The problem has the solution  $z_i[t + \sigma_k - \sigma_i] = \gamma_k / q_i \cdot c$  and cost  $\gamma_k c^2$ , where  $\gamma_k$  is as given in [Algorithm 1](#). Once the optimal level  $z_k[1]$  is calculated, the optimal value for  $u_{k-1}[0]$  can be found from the dynamics, which gives

$$\begin{aligned} u_{k-1}[0] = & \left(1 - \frac{\gamma_k}{q_k}\right) (z_k[0] + u_k[-\tau_k]) + v_k[0] + d_k[0] \\ & - \frac{\gamma_k}{q_k} \bar{V}_k[\sigma_k] - \frac{\gamma_k}{q_k} (m_{k-1}[0] + \sum_{i=1}^{k-1} \sum_{d=0}^{\tau_i-1} \bar{V}_i[\sigma_i + d]). \end{aligned}$$

Where  $m_k[t] = \sum_{i=1}^k (z_i[t] + \sum_{\delta=1}^{\tau_i} u_i[t - \delta])$ . After inserting the optimal values for  $v_k$  and  $V_i$  the expression in (4) is achieved. Note that all the terms with coefficient 1 corresponds to what would be in the node  $k$  at time  $t = 1$  if  $u_{k-1}[0] = 0$ , and all terms with coefficient  $-\gamma_k/q_k$  gives the total quantity in  $S_k[1]$ .

Furthermore, each shifted level sum only depends on the shifted production sums  $V_k[t]$ , and not the individual productions  $v_i[t]$ ,  $i \leq k$ . The optimal way to produce a specific amount  $V_k[t]$  with a shifted production vector can be found by solving a problem similar to (5), with the optimal  $v_i$  given by  $v_i[t - \sigma_i] = \rho_k/r_i \cdot V_k[t]$  for  $i \leq k$ , and the cost given by  $\rho_k V_k[t]^2$ , where  $\rho_k$  is as given in Algorithm 1. So the calculations of  $\gamma_i$  and  $\rho_i$  in the first sweep in Algorithm 1 thus corresponds to solving the optimal distribution for a shifted level vector  $S_k$  and the optimal production for a shifted production vector  $V_k$ . This is covered in Lemma 1.

Assuming all  $u_i$  are picked so that the shifted levels are optimized, the total level cost is given by

$$\sum_{t=0}^{\infty} \sum_{i=1}^N q_i z_i[t]^2 = \sum_{i=0}^N q_i z_i[0]^2 + \sum_{i=1}^{N-1} \sum_{t=\sigma_i+1}^{\sigma_{i+1}} \gamma_i S_i[t]^2 + \sum_{t=\sigma_N+1}^{\infty} \gamma_N S_N[t]^2. \quad (6)$$

And assuming all  $v_i$  are picked so that each shifted production vector is optimized, then the total production cost is given by

$$\sum_{t=0}^{\infty} \sum_{i=1}^N r_i v_i[t]^2 = \sum_{i=1}^{N-1} \sum_{t=\sigma_i}^{\sigma_{i+1}-1} \rho_i V_i[t]^2 + \sum_{t=\sigma_N}^{\infty} \rho_N V_N[t]^2. \quad (7)$$

This allows the problem in (2) to be solved in terms of  $V_i$  and  $S_i$ , reducing it to a problem in scalar variables. The scalar problem can be solved analytically, giving a closed form solution.

This is done by first solving for all  $V_N[t]$  outside the disturbance horizon, that is for  $t > \sigma_N + H$ . Using that outside the disturbance horizon the dynamics for the shifted levels  $S_N[t]$  are  $S_N[t+1] = S_N[t] + V_N[t]$  gives that those  $V_N[t]$  are given by the solution to

$$\begin{aligned} & \underset{V_N[t]}{\text{minimize}} && \sum_{t=\sigma_N+H+1}^{\infty} \gamma_N S_N[t]^2 + \rho_N V_N[t]^2 \\ & \text{subject to} && S_N[t+1] = S_N[t] + V_N[t]. \end{aligned}$$

This problem can be solved through a Riccati equation in one variable, giving expressions for  $V_N[t]$ ,  $t > \sigma_N + H$  in terms of  $S_N[t]$ . And more importantly, a cost to go in terms of  $S_N[\sigma_N + H + 1]$ , that is  $X_N[H + 2]$  in Algorithm 1.

Each shifted sum in (6) can be expressed in terms of initial conditions, shifted production vectors, and shifted disturbance vectors,

$$S_k[\sigma_k + \Delta] = m_{k-1}[0] + z_k[0] + \sum_{i=1}^{k-1} \sum_{d=\sigma_i}^{\sigma_{i+1}-1} \bar{V}_i[d] + \sum_{d=\sigma_k}^{\sigma_k+\Delta-1} \bar{V}_k[d].$$

Using the cost to go from the Riccati equation as the terminal cost allows the rest of the  $V_i$ 's to be found analytically using dynamic programming. When solving this

problem the cost to go  $X_i$  in [Algorithm 1](#) is used. The parameter  $g$  also appears naturally in the solution to each dynamic programming step, and the upstream aggregate  $\mu_i$  in [Algorithm 2](#) is used to simplify the expressions. This is covered in detail in [Lemma 3](#).

The resulting solution gives  $V_i[t]$  in terms of initial conditions and the previous  $V_k$ 's in (7). However,  $V_1[0]$  is known, which gives  $V_1[1]$  and so on. When rewriting  $V_i[0]$  in terms of only initial conditions the expressions can be simplified by using  $\delta$  as defined in [Algorithm 2](#). This in turn requires  $P$ ,  $h$ , and  $\phi$  which were defined in [Algorithm 1](#) and  $\Phi$  which was defined in [Algorithm 2](#). For the details see [Lemma 4](#).

## 6 Conclusions and Future Work

In this paper we studied an optimal control problem on a simple transportation model. We showed that the optimal controller is highly structured, allowing for a distributed implementation consisting of two sweeps through the graph. The optimal controller can also handle planned disturbances in an efficient way.

We believe that the results presented here can be extended to more general graph structures. More specifically for any graph with the structure of a directed tree both the proof technique and the results could be extended. We plan to explore this in a future publication.

## A Appendix

The proof follows the structure of the proof idea. Before we start we restate the definition of  $m_k$  which was mentioned in the proof idea.

$$m_k[t] = \sum_{i=1}^k \left( z_i[t] + \sum_{d=1}^{\tau_i} u_i[t-d] \right)$$

Also, we let the product over an empty set be equal to one, e.g.,  $\prod_{i=2}^1 g_i = 1$ .

The proof will derive the optimal inputs at time  $t = 0$ . As the problem has an infinite horizon, one can freely shift the time, and the results will thus hold for all  $t \geq 0$ . We begin by showing that each shifted level can be optimally distributed and find the corresponding internal flows.

**Lemma 1.** *The following holds*

(i) *Every shifted level  $S_k$  satisfies*

$$S_k[t + \sigma_k + 1] = z_k[t] + u_k[t - \tau_k] + m_{k-1}[t] + \sum_{i=1}^{k-1} \sum_{d=0}^{\tau_i-1} \bar{V}_i[t + \sigma_i + d] + \bar{V}_k[t + \sigma_k]$$

(ii) Let  $\gamma_k$  be defined as in [Algorithm 1](#). The optimization problem

$$\begin{aligned} & \underset{z_i}{\text{minimize}} && \sum_{i=1}^k q_i z_i [t + \sigma_k - \sigma_i]^2 \\ & \text{subject to} && S_k[t + \sigma_k] = m, \end{aligned}$$

has the solution  $z_i = \gamma_k / q_i m$  and the optimum value is given by  $\gamma_k m^2$ .

(iii) When  $u$  is chosen optimally, the cost for (2) is given by

$$\sum_{t=0}^{\infty} \sum_{i=1}^N q_i z_i [t]^2 = \sum_{i=0}^N q_i z_i [0]^2 + \sum_{i=1}^{N-1} \sum_{t=\sigma_i+1}^{\sigma_{i+1}} \gamma_i S_i [t]^2 + \sum_{t=\sigma_N+1}^{\infty} \gamma_N S_N [t]^2.$$

Also, the optimal  $u_k[0]$  is given by

$$\begin{aligned} u_{k-1}[0] = & (1 - \frac{\gamma_k}{q_k})(z_k[0] + u_k[-\tau_k]) + v_k[0] + d_k[0] \\ & - \frac{\gamma_k}{q_k} \bar{V}_k[\sigma_k] - \frac{\gamma_k}{q_k} (m_{k-1}[0] + \sum_{i=1}^{k-1} \sum_{d=0}^{\tau_i-1} \bar{V}_i[\sigma_i + d]). \end{aligned}$$

*Proof.* For  $k = 1$  (i) reduces to the dynamics. Now assume that (i) holds for  $k - 1$ . It follows from the definition of  $S_k$  that

$$S_k[t + \sigma_k + 1] = z_k[t + 1] + S_{k-1}[t + \sigma_k + 1]. \quad (8)$$

It holds that

$$S_k[t + 1] = S_k[t] + \bar{V}_k[t] + u_k[t - \sigma_k - \tau_k], \quad (9)$$

since  $u_i[t - \sigma_i - \tau_i]$  will cancel out for  $i < k$ . This allows  $S_{k-1}[t + \sigma_k + 1]$  to be rewritten as

$$S_{k-1}[t + \sigma_k + 1] = S_{k-1}[t + \sigma_{k-1} + 1] + \sum_{\Delta=\sigma_{k-1}+1}^{\sigma_k} \bar{V}_{k-1}[t + \Delta] + \sum_{\Delta=0}^{\tau_{k-1}-1} u_{k-1}[t - \Delta]. \quad (10)$$

Using the induction assumption that (i) holds for  $k - 1$ , (10) and the dynamics,

$$z_k[t + 1] = z_k[t] - u_{k-1}[t] + u_k[t - \tau_k] + v_k[t] + d_k[t], \quad (11)$$

allows (8) to be rewritten as

$$\begin{aligned} S_k[t + \sigma_k + 1] = & z_k[t] - u_{k-1}[t] + u_k[t - \tau_k] + v_k[t] + d_k[t] \\ & + z_{k-1}[t] + u_{k-1}[t - \tau_{k-1}] + m_{k-2}[t] + \sum_{i=1}^{k-2} \sum_{d=0}^{\tau_i-1} \bar{V}_i[t + \sigma_i + d] \\ & + \bar{V}_{k-1}[t + \sigma_{k-1}] + \sum_{\Delta=\sigma_{k-1}+1}^{\sigma_k} \bar{V}_{k-1}[t + \Delta] + \sum_{\Delta=0}^{\tau_{k-1}-1} u_{k-1}[t - \Delta]. \end{aligned}$$

In the above it holds that

$$z_{k-1}[t] + u_{k-1}[t - \tau_{k-1}] - u_{k-1}[t] + \sum_{\Delta=0}^{\tau_{k-1}-1} u_{k-1}[t - \Delta] + m_{k-2}[t] = m_{k-1}[t]$$

and

$$\begin{aligned} v_k[t] + d_k[t] + \sum_{i=1}^{k-2} \sum_{d=0}^{\tau_i-1} \bar{V}_i[t + \sigma_i + d] + \bar{V}_{k-1}[t + \sigma_{k-1}] + \sum_{\Delta=\sigma_{k-1}+1}^{\sigma_k} \bar{V}_{k-1}[t + \Delta] \\ = \sum_{i=1}^{k-1} \sum_{d=0}^{\tau_i-1} \bar{V}_i[t + \sigma_i + d] + \bar{V}_k[t + \sigma_k]. \end{aligned}$$

And thus (i) holds for  $k$  as well.

For (ii) the proposed solution satisfies the constraint as

$$\sum_{i=1}^k \frac{1}{q_i} = \frac{1}{\gamma_k}.$$

If the proposed solution was not optimal then it would be possible to improve it by increasing  $z_i$  by epsilon and decreasing  $z_j$  by epsilon for  $i, j \leq K$  as the problem is convex. However

$$\frac{\partial}{\partial z_i} q_i z_i [t + \sigma_k - \sigma_i]^2 = 2\gamma_k m$$

for  $z_i[t + \sigma_k - \sigma_i] = \gamma_k / q_i m$  and all  $i$ , and thus the proposed solution is optimal.

For (iii) note that the sum  $\sum_{t=0}^{\infty} \sum_{i=1}^N q_i z_i [t]^2$  can be written in terms of shifted level vectors as follows,

$$\begin{aligned} \sum_{t=0}^{\infty} \sum_{i=1}^N q_i z_i [t]^2 = \\ \sum_{i=0}^N q_i z_i [0]^2 + \sum_{i=1}^{N-1} \sum_{t=\sigma_{i+1}}^{\sigma_{i+1}} \sum_{j=1}^i q_j z_j [t - \sigma_j]^2 + \sum_{t=\sigma_N+1}^{\infty} \sum_{j=1}^N q_j z_j [t - \sigma_j]^2. \quad (12) \end{aligned}$$

The inner sums corresponds to the objective in (ii). From (i) it follows that  $S_k[t]$ ,  $t \leq \sigma_{i+1}$  is independent of  $u_j[t]$ ,  $\forall t \geq 0, \forall j$  and that  $S_N[t]$  is independent of  $u_j[t]$ ,  $\forall t, j$ . Thus each shifted level sum in (12) is independent of the internal flows. Now consider arbitrary, but fixed productions  $V$  and disturbances  $D$ . Then by (i) the sum of all shifted levels are fixed. If there exists  $u$  so that each sum over shifted levels in (12) is the optimal solution to the problem in (ii), then those inputs must be optimal for the given  $V$  and  $D$ . By choosing  $u_{j-1}[t - \sigma_j - 1]$  so that  $z_j[t - \sigma_j]$  is optimal for (ii) for  $2 \leq j \leq i$  gives that all  $z_j[t - \sigma_j]$  are optimally for  $2 \leq j \leq i$ . However, since the constraint will always be satisfied,  $z_1[t]$  will be optimal as well. Using (i), the optimal  $z_k[1]$  from (ii) is given by

$$z_k[1] = \frac{\gamma_k}{q_k} \left( m_{k-1}[0] + z_k[0] + u_k[-\tau_k] + \sum_{i=1}^{k-1} \sum_{\Delta=0}^{\tau_i-1} \bar{V}_i[\sigma_i + \Delta] + \bar{V}_k[\sigma_k] \right).$$

Inserting the dynamics in (11) into the LHS and solving for  $u_{k-1}[0]$  gives the expression in (iii).  $\square$

Now we will give the solution to the optimization problem which will arise in the dynamic programming problem that will need to be solved in the next lemma.

**Lemma 2.** *Let  $X_i$  and  $g_i(j)$  be defined as in Algorithm 1. Then*

(i) *Let  $j \geq 1$ . The optimization problem*

$$\underset{x}{\text{minimize}} \quad X_i(j+1)(a+b+x)^2 + \gamma_i(a+x)^2 + \rho_i x^2$$

*has minimizer*

$$x = -\frac{X_i(j)}{\rho_i}(a + g_i(j+1)b),$$

*with optimum value  $X_i(j) \cdot (a + g_i(j+1)b)^2 + f(b)$ .*

(ii) *The optimization problem*

$$\underset{x}{\text{minimize}} \quad X_{i+1}(1)(a+b+x)^2 + \gamma_i(a+x)^2 + \rho_i x^2$$

*has minimizer*

$$x = -\frac{X_i(\tau_i)}{\rho_i}(a + g_{i+1}(1)b),$$

*with optimum value  $X_i(\tau_i) \cdot (a + g_{i+1}(1)b)^2 + f(b)$ .*

*Proof.* We will show that the optimization problem

$$\underset{x}{\text{minimize}} \quad c_1(a+b+x)^2 + c_2(a+x)^2 + c_3 x^2$$

has the solution

$$x = -\frac{c_1 + c_2}{c_1 + c_2 + c_3} \left( a + \frac{c_1}{c_1 + c_2} b \right)$$

and the minimal value is on the form

$$\frac{c_3(c_1 + c_2)}{c_1 + c_2 + c_3} \left( a + \frac{c_1}{c_1 + c_2} b \right)^2 + f(b),$$

where  $f(b)$  is independent of  $a$ . The lemma then follows by applying the above and using the definition for  $X_i$  and  $g_i(j)$ .

There exists a unique solution as the problem is strictly convex. Differentiating the objective function with respect to  $x$  gives that the optimal  $x$  is given by

$$x = -\frac{1}{c_1 + c_2 + c_3} \left( (c_1 + c_2)a + c_1 b \right)$$

from which the proposed  $x$  follows. The objective function can be rewritten as

$$c_1(a+b)^2 + c_2 a^2 + 2[(c_1 + c_2)a + c_1 b]x + (c_1 + c_2 + c_3)x^2.$$

Inserting the minimizer gives

$$\frac{1}{c_1 + c_2 + c_3} \left( (c_1 + c_2 + c_3)(c_1(a+b)^2 + c_2a^2) - [(c_1 + c_2)a + c_1b]^2 \right).$$

The first term can be written as

$$\begin{aligned} & (c_1 + c_2 + c_3)(c_1(a+b)^2 + c_2a^2) \\ &= (c_1 + c_2 + c_3)[(c_1 + c_2)a^2 + 2c_1ab + c_1b^2] \\ &= (c_1 + c_2)^2a^2 + 2(c_1 + c_2)c_1ab + c_1^2b^2 + \\ & \quad c_3(c_1 + c_2)a^2 + 2c_1c_3ab + (c_2 + c_3)c_1b^2. \end{aligned}$$

Which gives that the objective function has the minimum value

$$\frac{1}{c_1 + c_2 + c_3} \left[ c_3(c_1 + c_2)a^2 + 2c_1c_3ab + (c_2 + c_3)c_1b^2 \right].$$

The last term is independent of  $a$ . The dependence on  $a$  is thus given by

$$\frac{c_3(c_1 + c_2)}{c_1 + c_2 + c_3} \left( a^2 + \frac{2c_1}{c_1 + c_2}ab \right) = \frac{c_3(c_1 + c_2)}{c_1 + c_2 + c_3} \left( a + \frac{c_1}{c_1 + c_2}b \right)^2 + f(b)$$

□

Armed with the results from the previous lemma, we will now apply dynamic programming to (2). We will show that the problem can be solved in terms of the shifted levels  $S_k$ , shifted productions  $V_k$  and shifted disturbances  $D_k$ . Outside of the horizon the problem can be solved using the Riccati equation. Using the cost to go given by the Riccati equation as initialization we can apply dynamic programming using the results from the previous lemma.

**Lemma 3.** *Let  $\gamma_k$ ,  $\rho_k$ ,  $X_k$ ,  $g_k$ ,  $\mu_k$  be defined as in Algorithms 1 and 2. Let for  $1 \leq k \leq N-1$  and  $1 \leq \Delta \leq \tau_k$ , and for  $k = N$  and  $1 \leq \Delta \leq H+2$*

$$\begin{aligned} \xi_k[\Delta-1] &= m_{k-1}[0] + z_k[0] + \sum_{i=1}^{k-1} \sum_{d=\sigma_i}^{\sigma_{i+1}-1} \bar{V}_i[d] \\ &+ \sum_{d=0}^{\tau_k-1} \left( u_k[-(\tau_k-d)] + D_k[\sigma_k+d] \right) \prod_{j=\Delta+1}^{d+1} g_k(j) \\ &+ \sum_{d=\sigma_k}^{\sigma_k+\Delta-2} V_k[d] + \mu_{k+1}[0]g_{k+1}(1) \prod_{j=\Delta+1}^{\tau_k} g_k(j). \quad (13) \end{aligned}$$

Then the optimal  $V_k$  for (2) is given by

$$V_k[\sigma_k + (\Delta-1)] = -\frac{X_k(\Delta)}{\rho_k} \xi_k[\Delta-1].$$

The optimal individual productions are given by

$$v_k[\Delta-1] = \frac{\rho_k}{r_k} V_k[\sigma_k + (\Delta-1)].$$

*Proof.* By Lemma 1-(i) and (9) each shifted inventory level  $S_k[\sigma_k + \Delta]$  with  $1 \leq \delta \leq \tau_k$  satisfies

$$S_k[\sigma_k + \Delta] = m_{k-1}[0] + z_k[0] + \sum_{i=1}^{k-1} \sum_{d=\sigma_i}^{\sigma_{i+1}-1} \bar{V}_i[d] + \sum_{d=\tau_k-(\Delta-1)}^{\tau_k} u_k[-d] + \sum_{d=\sigma_k}^{\sigma_k+\Delta-1} \bar{V}_k[d]. \quad (14)$$

By (iii) in Lemma 1 the cost can be rewritten as

$$\sum_{t=0}^{\infty} \sum_{i=1}^N q_i z_i[t]^2 = \sum_{i=0}^N q_i z_i[0]^2 + \sum_{i=1}^{N-1} \sum_{t=\sigma_{i+1}}^{\sigma_{i+1}-1} \gamma_i S_i[t]^2 + \sum_{t=\sigma_N+1}^{\infty} \gamma_N S_N[t]^2.$$

And similarly, by Lemma 1-(ii), the optimal cost for a shifted production  $V_i[t]$  is given by  $\rho_i V_i[t]^2$  and individual productions are given by  $v_i[t] = \rho_i / r_i \cdot V_i[t + \sigma_i]$ . This gives the total production cost in terms of  $V_i$  as

$$\sum_{t=0}^{\infty} \sum_{i=1}^N r_i v_i[t]^2 = \sum_{i=1}^{N-1} \sum_{t=\sigma_i}^{\sigma_{i+1}-1} \rho_i V_i[t]^2 + \sum_{t=\sigma_N}^{\infty} \rho_N V_N[t]^2.$$

We can thus solve the problem in terms of  $S_i$  and  $V_i$ , and then recover the optimal  $v_i$ . To that end define the cost to go for  $1 \leq k \leq N-1$  and  $1 \leq \Delta \leq \tau_k$

$$\begin{aligned} \Gamma_k[\Delta] = & \sum_{t=\sigma_k+\Delta}^{\sigma_{k+1}} \left( \gamma_k S_k[t]^2 + \rho_k V_k[t-1]^2 \right) + \sum_{i=k+1}^{N-1} \sum_{t=\sigma_{i+1}}^{\sigma_{i+1}-1} \left( \gamma_i S_i[t]^2 + \rho_i V_i[t-1]^2 \right) \\ & + \sum_{t=\sigma_N+1}^{\infty} \left( \gamma_N S_N[t]^2 + \rho_N V_N[t-1]^2 \right). \end{aligned}$$

And for  $k = N$  and  $\Delta \geq 1$

$$\Gamma_N[\Delta] = \sum_{t=\sigma_N+\Delta}^{\infty} \left( \gamma_N S_N[t]^2 + \rho_N V_N[t-1]^2 \right).$$

We will show for  $1 \leq k \leq N-1$  and  $1 \leq \Delta \leq \tau_i$ , and for  $k = N$  and  $1 \leq \Delta \leq H+2$ , that

$$\Gamma_k[\Delta] = X_k(\Delta) \xi_k[\Delta-1]^2 + f(b), \quad (15)$$

where  $f(b)$  is independent of  $V_k[t]$ .  $f(b)$  can thus be ignored in the optimization of  $V_k[t]$ .

Using Lemma 1-(i) combined with (9) and that all  $D_N[t] = 0$  for  $t > H + \sigma_N$  it follows that the optimal  $V_N[t]$  for  $t > \sigma_N + H$  is given by the solution to the problem

$$\begin{aligned} & \underset{V_N[t]}{\text{minimize}} && \sum_{t=\sigma_N+H+1}^{\infty} \gamma_N S_N[t]^2 + \rho_N V_N[t]^2 \\ & \text{subject to} && S_N[t+1] = S_N[t] + V_N[t] \\ & && S_N[\sigma_N + H + 1] = m_N[0] + \sum_{i=1}^{N-1} \sum_{\delta=\sigma_i}^{\sigma_{i+1}-1} \bar{V}_i[\delta] + \sum_{\delta=\sigma_N}^{\sigma_N+H} \bar{V}_N[\delta]. \end{aligned}$$

This is a standard LQR problem and the solution can be found by solving the following Riccati equation

$$X = X - X^2/(\rho_N + X) + \gamma_N \Rightarrow X = \frac{\gamma_N}{2} + \sqrt{\gamma_N \rho_N + \frac{\gamma_N^2}{4}}.$$

Now let  $X_N(H+2) = X - \gamma_N$ . Then  $\Gamma_N[\sigma_N + H + 2]$  is given by

$$\Gamma_N[\sigma_N + H + 2] = S_N[\sigma_N + H + 1]^2 X_N(H + 2).$$

Note that the cost for  $S_k[\sigma_N + H + 1]$  is not part of  $\Gamma_N[\sigma_N + H + 2]$ , but it is part of the cost to go given by the solution  $X$  to the Riccati equation. Furthermore, the optimal  $V_N[t]$  for  $t = \sigma_N + H + 1$  is given by

$$V_N[t] = -\frac{X}{X + \rho_N} S_N[t] = -\frac{X_N(H+1) + \gamma_N}{X_N(H+1) + \gamma_N + \rho_N} S_N[t] = -\frac{X_N(H)}{\rho_N} S_N[t].$$

For  $\Delta = H + 2$  and  $k = N$  the expression for  $\xi_k[\Delta - 1]$  reduces to

$$\xi_N[H + 1] = m_N[0] + \sum_{i=1}^{N-1} \sum_{d=\sigma_i}^{\sigma_{i+1}-1} \bar{V}_i[d] + \sum_{d=\sigma_N}^{\sigma_N+H} \bar{V}_N[d],$$

as  $\mu_{N+1} = 0$ ,  $u_N = 0$  and  $D_N[t] = 0$  for  $t > \sigma_N + H$ . By (14)  $\xi_N[H + 1] = S_N[\sigma_N + H + 1]$  and thus the lemma and (15) holds for  $k = N$  and  $\Delta = H + 2$ .

Assume that (15) holds for  $k + 1$  and  $\Delta = 1$ . Then the optimal  $V_k[\sigma_{k+1} - 1]$  is given by the minimizer for

$$\Gamma_k[\tau_k] = \Gamma_{k+1}[1] + \gamma_k S_k[\sigma_{k+1}]^2 + \rho_k V_k[\sigma_{k+1} - 1]^2.$$

Using the assumption for the cost to go in (15) gives that  $\Gamma_{k+1}[1] = X_{k+1}(1) \xi_{k+1}[0]^2$  and thus the optimal  $V_k[\sigma_{k+1} - 1]$  is given by the optimal value for the problem

$$\underset{V_k[\sigma_{k+1}-1]}{\text{minimize}} \quad X_{k+1}(1) \xi_{k+1}[0]^2 + \gamma_k S_k[\sigma_{k+1}]^2 + \rho_k V_k[\sigma_{k+1} - 1]^2.$$

For  $\Delta = 1$  (13) reduces to

$$\xi_k[0] = m_{k-1}[0] + \sum_{i=1}^{k-1} \sum_{d=\sigma_i}^{\sigma_{i+1}-1} \bar{V}_i[d] + \mu_k[0], \quad (16)$$

as

$$\mu_k[0] = \pi_k[0] + \mu_{k+1} g_{k+1}(1) \prod_{j=2}^{\tau_k} g_k(j)$$

and

$$\pi_k[0] = z_k[0] + \sum_{d=0}^{\tau_k-1} \left( u_k[-(\tau_k - d)] + D_k[\sigma_k + d] \right) \prod_{j=2}^{d+1} g_k(j).$$

We also note that by (14), as  $\sigma_{k+1} = \sigma_k + \tau_k$ ,

$$S_k[\sigma_{k+1}] = m_k[0] + \sum_{i=1}^k \sum_{d=\sigma_i}^{\sigma_{i+1}-1} \bar{V}_i[d].$$

Applying Lemma 2-(ii) with

$$\begin{aligned} a &= S_k[\sigma_{k+1}] - V_k[\sigma_{k+1} - 1] \\ b &= \xi_{k+1}[0] - S_k[\sigma_{k+1}] = \mu_{k+1}[0] \\ x &= V_k[\sigma_{k+1} - 1], \end{aligned}$$

gives that the lemma and (15) hold for  $k$  and  $\Delta = \tau_k$  as

$$\xi_k[\tau_k - 1] = m_k[0] + \sum_{i=1}^{k-1} \sum_{d=\sigma_i}^{\sigma_{i+1}-1} \bar{V}_i[d] + \sum_{d=\sigma_k}^{\sigma_k + \tau_k - 2} \bar{V}_k[d] + \mu_{k+1}[0]g_{k+1}(1).$$

Assume that (15) holds for some  $k$  and  $\Delta + 1$ , where  $1 \leq \Delta \leq \tau_i - 1$  if  $k < N$  and  $1 \leq \Delta \leq H + 1$  if  $k = N$ . Then  $V_k[\sigma_k + \Delta - 1]$  can be found as the minimizer for

$$\underset{V_k[\sigma_k + \Delta - 1]}{\text{minimize}} \quad X_k(\Delta + 1)\xi_k[\Delta]^2 + \gamma_k S_k[\sigma_k + \Delta]^2 + \rho_k V_k[\sigma_k + \Delta - 1]^2.$$

Using that two of the terms in (14) can be rewritten as

$$\sum_{d=\tau_k - (\Delta - 1)}^{\tau_k} u_k[-d] + \sum_{d=\sigma_k}^{\sigma_k + \Delta - 1} \bar{V}_k[d] = \sum_{d=0}^{\Delta - 1} \left( u_k[-(\tau_k - d)] + D_k[\sigma_k + d] \right) + \sum_{d=\sigma_k}^{\sigma_k + \Delta - 1} V_k[d]$$

and with  $x = V_k[\sigma_k + \Delta - 1]$ ,  $a = S_k[\sigma_k + \Delta] - V_k[\sigma_k + \Delta - 1]$ , which equals

$$\begin{aligned} a &= m_{k-1}[0] + z_k[0] + \sum_{i=1}^{k-1} \sum_{d=\sigma_i}^{\sigma_{i+1}-1} \bar{V}_i[d] \\ &\quad + \sum_{d=0}^{\Delta - 1} \left( u_k[-(\tau_k - d)] + D_k[+\sigma_k + d] \right) + \sum_{d=\sigma_k}^{\sigma_k + \Delta - 2} V_k[d], \end{aligned}$$

and  $b = \xi_k[\Delta] - S_k[\sigma_k + \Delta]$ , which gives

$$b = \sum_{d=\Delta}^{\tau_k - 1} \left( u_k[-(\tau_k - d)] + D_k[\sigma_k + d] \right) \prod_{j=\Delta+2}^{d+1} g_k(j) + \mu_{k+1}[0]g_{k+1}(1) \prod_{j=\Delta+2}^{\tau_k} g_k(j).$$

By applying Lemma 2-(i) it follows that (15) and the lemma holds for  $k$  and  $\Delta$  as well. Thus the lemma holds for all  $1 \leq \Delta \leq \tau_k$  for  $1 \leq k \leq N - 1$  and  $1 \leq \Delta \leq H + 2$  for  $k = N$ .  $\square$

All that remains now is to find expressions for  $V_k[\sigma_k]$  in terms of the initial conditions. The following lemma allows us to do so, using the expressions for  $V_k$  derived in the previous lemma.

**Lemma 4.** Let  $h_k, P_k(i, j), \phi_k(\Delta), \pi_k[0], \mu_k[0], \Phi_k[0]$ , and  $\delta_k[0]$  be defined as in Algorithms 1 and 2. Then for  $k \leq N-1$

$$m_k[0] + \sum_{i=1}^k \sum_{d=\sigma_i}^{\sigma_{i+1}-1} \bar{V}_i[d] = \delta_k[0] - h_k \mu_{k+1}[0] \quad (17)$$

*Proof.* Let  $B_k[0] = z_k[0] + u_k[-\tau_k] + D_k[\sigma_k]$  and  $B_k[i] = u_k[-(\tau_k - i)] + D_k[\sigma_k + i]$  for  $1 \leq i < \tau_k$ . We will prove the lemma by showing that for  $1 \leq \Delta \leq \tau_k$

$$\begin{aligned} m_{k-1}[0] + \sum_{i=1}^{k-1} \sum_{d=\sigma_i}^{\sigma_{i+1}-1} \bar{V}_i[d] + \sum_{d=\sigma_k}^{\sigma_k+\Delta-1} V_k[d] = \\ (1 - P_k(\Delta, 1)) \delta_{k-1}[0] - \left[ h_{k-1} b_k (1 - P(\Delta, 1)) + P_k(\Delta, \Delta) g_{k+1}(1) \prod_{j=\Delta+1}^{\tau_k} g_k(j) \right] \mu_{k+1}[0] \\ - \sum_{d=0}^{\tau_k-1} B_k[d] \left[ P_k(\Delta, \min(d+1, \Delta)) \prod_{j=\Delta+1}^{d+1} g_k(j) + (1 - P_k(\Delta, 1)) h_{k-1} \prod_{j=2}^{d+1} g_k(j) \right] \quad (18) \end{aligned}$$

More specifically we will show that

1. (18) holds for  $k = 1$  and  $\Delta = 1$ .
2. If (18) holds for some  $k$  and  $\Delta - 1$  then it holds for  $\Delta$  as well.
3. If (17) holds for  $k - 1$  then (18) holds for  $k$  and  $\Delta = 1$ .
4. If (18) holds for  $k$  and  $\Delta = \tau_k$  then (17) holds for  $k$ .

For  $k = 1$  and  $\Delta = 1$  the LHS of (18) is just  $V_1[t]$ . The RHS of (18) equals  $-X_1(1)/\rho_1 \cdot \mu_k[0]$  as  $P_1(1, m) = X_1(1)/\rho_1$ ,  $\delta_0 = 0$ , and  $h_0 = 0$ . And by Lemma 3 the RHS is also equal to  $V_1[t]$  since by (16)

$$\xi_1[0] = \mu_k[0].$$

Thus (18) holds for  $k = 1$  and  $\Delta = 1$ .

Applying Lemma 3 on  $V_k[\sigma_k + \Delta]$  for the LHS of (18) gives:

$$\begin{aligned} m_{k-1}[0] + \sum_{i=1}^{k-1} \sum_{d=\sigma_i}^{\sigma_{i+1}-1} \bar{V}_i[d] + \sum_{d=\sigma_k}^{\sigma_k+\Delta-1} V_k[d] = \\ \left(1 - \frac{X_k(\Delta)}{\rho_k}\right) \left( m_{k-1}[0] + \sum_{i=1}^{k-1} \sum_{d=\sigma_i}^{\sigma_{i+1}-1} \bar{V}_i[d] + \sum_{d=\sigma_k}^{\sigma_k+\Delta-2} V_k[d] \right) \\ - \frac{X_k(\Delta)}{\rho_k} \left[ \sum_{d=0}^{\tau_k-1} B_k[d] \prod_{j=\Delta+1}^{d+1} g_k(j) + \mu_{k+1}[0] g_{k+1}(1) \prod_{j=\Delta+1}^{\tau_k} g_k(j) \right] \quad (19) \end{aligned}$$

Now assume that (18) holds for  $k$  and  $\Delta - 1$ , we then show that (18) holds for  $k$  and  $\Delta$ . Using (19) gives for the coefficients for the different terms of the LHS for (18) as follows. For  $\delta_{k-1}[0]$  we get

$$\left(1 - \frac{X_k(\Delta)}{\rho_k}\right)(1 - P_k(\Delta - 1, 1)) = 1 - P_k(\Delta, 1).$$

For the terms in front of  $\mu_{k+1}[0]$  we get

$$\begin{aligned} & - \left(1 - \frac{X_k(\Delta)}{\rho_k}\right) \left( h_{k-1} b_k (1 - P(\Delta - 1, 1)) + P_k(\Delta - 1, \Delta - 1) g_{k+1}(1) \prod_{j=\Delta}^{\tau_k} g_k[j] \right) \\ & \quad - \frac{X_k(\Delta)}{\rho_k} g_{k+1}(1) \prod_{j=\Delta+1}^{\tau_k} g_k(j) \\ & = -h_{k-1} b_k (1 - P(\Delta, 1)) - P_k(\Delta, \Delta) g_{k+1}(1) \prod_{j=\Delta+1}^{\tau_k} g_k(j), \end{aligned}$$

and for the coefficient for  $B[d]$ ,

$$\begin{aligned} & - \left(1 - \frac{X_k(\Delta)}{\rho_k}\right) \left[ P_k(\Delta - 1, \min(d + 1, \Delta - 1)) \prod_{j=\Delta}^{d+1} g_k(j) + \right. \\ & \quad \left. (1 - P_k(\Delta - 1, 1)) h_{k-1} \prod_{j=2}^{d+1} g_k(j) \right] - \frac{X_k(\Delta)}{\rho_k} \prod_{j=\Delta+1}^{d+1} g_k(j) \\ & = -P_k(\Delta, \min(d + 1, \Delta)) \sum_{j=\Delta+1}^{d+1} g_k(j) - (1 - P_k(\Delta, 1)) h_{k-1} \prod_{j=2}^{d+1} g_k(j). \end{aligned}$$

Thus (18) holds for  $k$  and  $\delta$  as well.

Assume that (17) holds for  $k - 1$ . Then we can show that (18) holds for  $k$  and  $\Delta = 1$ . Using that

$$\pi_k[0] = \sum_{d=0}^{\tau_k-1} \left( B[d] \prod_{j=2}^{d+1} g_k(j) \right), \quad (20)$$

the RHS of (18) reduces to

$$\left[ 1 - P_k(1, 1) \right] \delta_{k-1}[0] - \left[ h_{k-1} (1 - P_k(1, 1)) + P_k(1, 1) \right] (\pi_k[0] + b_k \mu_{k+1}[0]).$$

Using (19) with  $\Delta = 1$ , the definition for  $P_k(1, 1)$ , and inserting (17) gives that the LHS of (18) is equal to

$$(1 - P_k(1, 1)) \left[ \delta_{k-1}[0] - h_{k-1} \mu_k[0] \right] - P_k(1, 1) \left[ \sum_{d=0}^{\tau_k-1} B[d] \prod_{j=\Delta+1}^{d+1} g_k(j) + \mu_{k+1}[0] b_k \right].$$

Using (20) and the definition for  $\mu_k[0] = \pi_k[0] + b_k \mu_{k+1}[0]$  shows that the RHS and LHS are equal. And thus (18) hold for  $k$  and  $\Delta = 1$  if (17) holds for  $k - 1$ .

Finally, we will show that if (18) holds for  $k$  and  $\Delta = \tau_k$  then (17) holds for  $k$ . Using the definition for  $h_k$  the RHS of (18) reduces to

$$(1 - P_k(\tau_k, 1))\delta_{k-1}[0] + h_k\mu_{k+1}[0] - \sum_{i=d}^{\tau_k} B_k[d] \left[ P_k(\tau_k, d+1) + (1 - P_k(\tau_k, 1)h_{k-1}) \prod_{j=2}^{d+1} g_k(j) \right]$$

For  $\Delta = \tau_k$  the LHS of (17) is equal to the LHS of (18) plus

$$z_k[0] + \sum_{d=1}^{\tau_k} u_k[-d] + \sum_{d=\sigma_k}^{\sigma_{k+1}-1} D_k[d] = \sum_{d=0}^{\tau_k-1} B_k[d].$$

Thus it holds that the LHS of (17) is equal to

$$(1 - P_k(\tau_k, 1))\delta_{k-1}[0] + h_k\mu_{k+1}[0] + \sum_{i=d}^{\tau_k} B_k[d] \left[ (1 - P_k(\tau_k, d+1)) - (1 - P_k(\tau_k, 1)h_{k-1}) \prod_{j=2}^{d+1} g_k(j) \right].$$

Using the definition for  $\phi_i(\Delta)$  in Algorithm 1 and  $\Phi_i$  and  $\delta_k$  in Algorithm 2 shows that (18) gives (17) for  $\Delta = \tau_k$ , as

$$\sum_{i=d}^{\tau_k} B_k[d] \left[ (1 - P_k(\tau_k, d+1)) - (1 - P_k(\tau_k, 1)h_{k-1}) \prod_{j=2}^{d+1} g_k(j) \right] = \Phi_k[0].$$

□

We are now finally ready to prove the theorem, which follows from the previous lemmas.

*Proof of Theorem 1:* Lemma 3 with  $\Delta = 1$  and Lemma 4 gives that

$$\begin{aligned} V_k[\sigma_k] &= -\frac{X_k(1)}{\rho_k} \left[ m_{k-1}[0] + \sum_{i=1}^{k-1} \sum_{d=\sigma_i}^{\sigma_{i+1}-1} \bar{V}_i[d] + \mu_k[0] \right] \\ &= -\frac{X_k(1)}{\rho_k} \left[ \delta_{k-1}[0] + (1 - h_k)\mu_k[0] \right] \end{aligned} \quad (21)$$

from which the optimal  $v_k[0] = \rho_k/r_k \cdot V_k[\sigma_k]$  as in Algorithm 2 follows. Using Lemma 1-(iii), Lemma 4, and that  $v_k[0] = \rho_k/r_k \cdot V_k[\sigma_k]$  gives that

$$\begin{aligned} u_{k-1}[0] &= (1 - \frac{\gamma_k}{q_k})(z_k[t] + u_k[-\tau_k] + D_k[0]) + d_k[0] - D_k[0] \\ &\quad + V_k[\sigma_k] \left( \frac{\rho_k}{r_k} - \frac{\gamma_k}{q_k} \right) - \frac{\gamma_k}{q_k} (\delta_{k-1}[0] - h_{k-1}\mu_k[0]). \end{aligned}$$

Inserting (21) gives that the optimal  $u$  is as in Algorithm 2.

The results will hold for  $t \neq 0$  as the problem has an infinite horizon and one can always change the variables so that current time is time zero. □

## References

- [1] M. Rotkowitz and S. Lall, “A characterization of convex problems in decentralized control,” *IEEE Transactions on Automatic Control*, vol. 51, no. 2, pp. 274–286, 2006.
- [2] L. Lessard and S. Lall, “Quadratic invariance is necessary and sufficient for convexity,” in *Proceedings of the 2011 American Control Conference*, 2011, pp. 5360–5362.
- [3] Y. Wang, N. Matni, and J. C. Doyle, “Separable and localized system-level synthesis for large-scale systems,” *IEEE Transactions on Automatic Control*, vol. 63, no. 12, pp. 4234–4249, 2018.
- [4] F. Lin, M. Fardad, and M. R. Jovanović, “Design of optimal sparse feedback gains via the alternating direction method of multipliers,” *IEEE Transactions on Automatic Control*, vol. 58, no. 9, pp. 2426–2431, 2013.
- [5] M. Cantoni, E. Weyer, Y. Li, S. K. Ooi, I. Mareels, and M. Ryan, “Control of large-scale irrigation networks,” *Proceedings of the IEEE*, vol. 95, no. 1, pp. 75–91, 2007.
- [6] K. Subramanian, J. B. Rawlings, C. T. Maravelias, J. Flores-Cerrillo, and L. Megan, “Integration of control theory and scheduling methods for supply chain management,” *Computers & Chemical Engineering*, vol. 51, pp. 4–20, 2013.
- [7] R. K. Ahuja, T. L. Magnanti, J. B. Orlin, and M. Reddy, “Applications of network optimization,” *Handbooks in Operations Research and Management Science*, vol. 7, pp. 1–83, 1995.
- [8] M. Heyden, R. Pates, and A. Rantzer, “A structured linear quadratic controller for transportation problems,” in *2018 European Control Conference (ECC)*. IEEE, 2018, pp. 1654–1659.