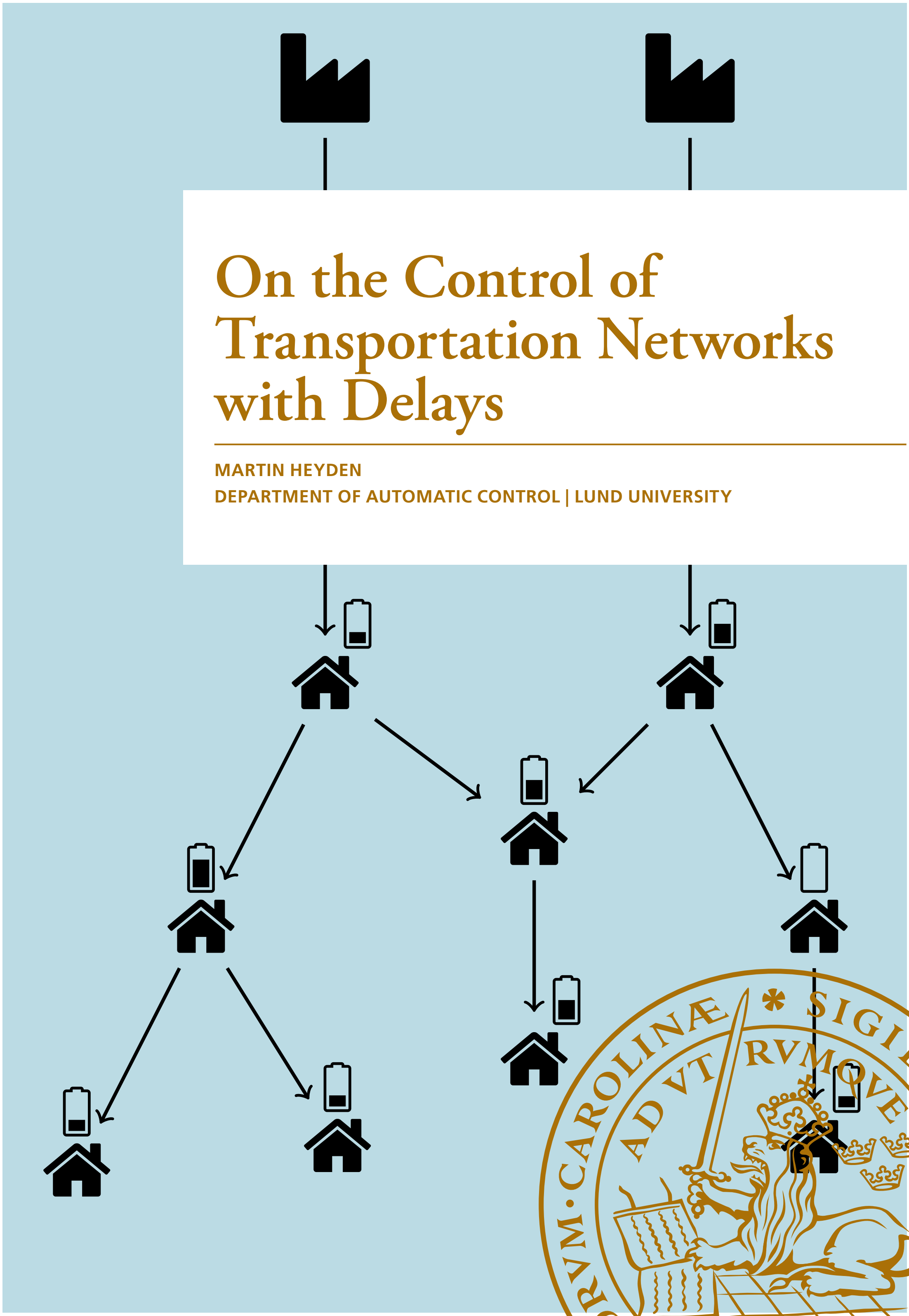
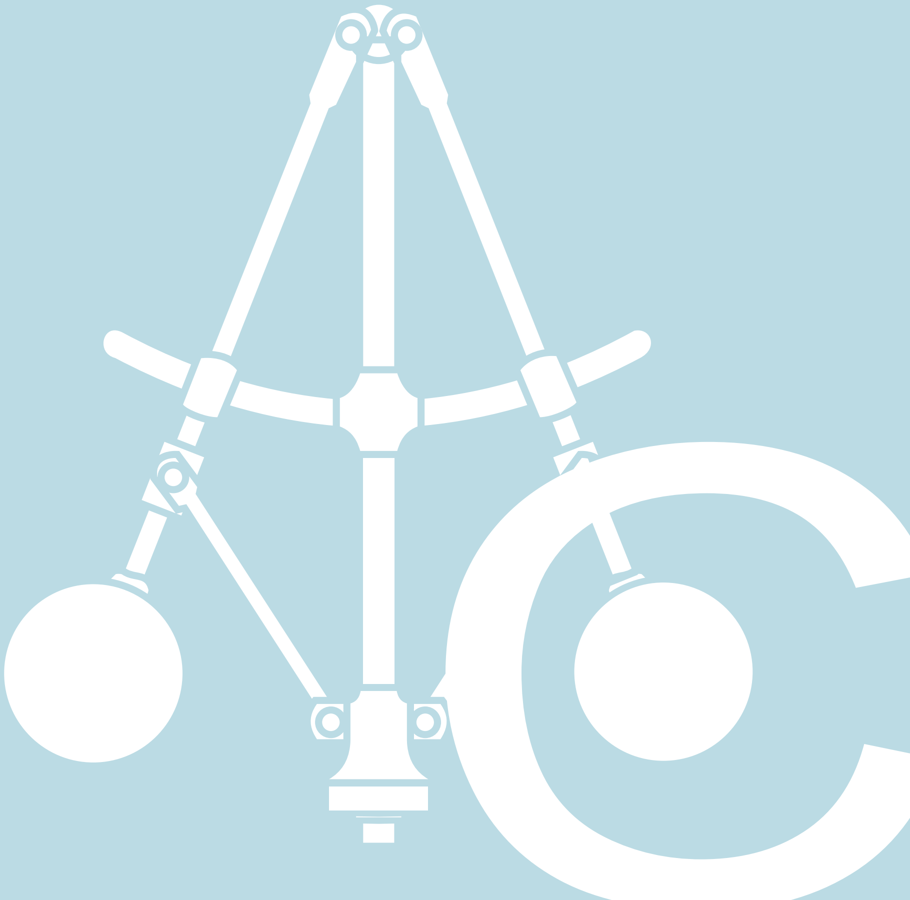


On the Control of Transportation Networks with Delays

MARTIN HEYDEN
DEPARTMENT OF AUTOMATIC CONTROL | LUND UNIVERSITY



On the Control of Transportation Networks with Delays

Martin Heyden

LUND UNIVERSITY

Department of Automatic Control

PhD Thesis TFRT-1133
ISBN 978-91-8039-117-7 (print)
ISBN 978-91-8039-118-4 (web)
ISSN 0280–5316

Department of Automatic Control
Lund University
Box 118
SE-221 00 LUND
Sweden

Abstract

In this thesis, a general model for transportation on directed tree graphs is studied. The nodes in the graph correspond to different storage locations, and the edges describe between which storage locations transportation is possible. The transportation is assumed to be subject to delay. Furthermore, nodes at the top of the network are allowed to produce more of the studied quantity. As an example, this setup can model an irrigation network, consisting of several pools that are connected via gates. The gates allow water to be transported from the upstream to the downstream pool. Each pool can be described by a node, and the edges describe which pools are connected by a gate. The production corresponds to taking water out from a reservoir and into a pool.

A common approach for control of large-scale networks is to stabilize the system around the optimal equilibrium point. However, as the operating conditions of the network change, the optimal equilibrium point will also change. In this thesis, the dynamic performance of the network is optimized, where the cost associated with deviations from the nominal levels is minimized. The transportation variations are not penalized, as it is assumed that this cost is negligible (for example, in the case of irrigation networks, gravity is responsible for the movement).

The optimal controller is shown to be highly structured, without imposing any structural constraints on the controller that normally limit performance. This structure allows for a simple and efficient implementation. The optimal transportation assignments can be calculated by a sweep through the graph, starting in the nodes without children, and iterating upwards. This implies that each gate in an irrigation network only needs to receive information from the gates downstream and send information to the gates upstream.

Even stronger results are derived for string graphs. Firstly, it is shown how to give optimal feed-forward for planned disturbances. These planned disturbances could for example be farmers taking water out of an irrigation network. This requires minor modifications to the aforementioned controller

structure, where the information about the planned disturbances can be communicated by a sweep through the graph. Secondly, it is shown how to allow for production in every node. This requires two sweeps, with one going in the upstream direction and one going in the downstream direction. These sweeps can be done in parallel, and thus the implementation time is unaffected. The resulting controller is applied to a more realistic simulation model for irrigation networks, where it outperforms a simple P controller in response to both step changes and disturbance rejection. For disturbance rejection of low-pass filtered disturbances, the performance is close to the theoretical maximum attained using a centralized controller with a perfect model.

The optimal control problem is also studied from a localized perspective, where each node tries to maximize its own utility. To coordinate, each node is presented with a price for having a certain level at each time point. It is shown how to calculate prices so that that the nodes' optimal levels align with the socially optimal levels. These prices can also be calculated by a sweep through the graph.

Acknowledgements

I am grateful to have had my supervisors Richard Pates and Anders Rantzer by my side throughout my Ph.D. studies. Thanks for always being positive, especially when I was not, and for giving invaluable feedback on how to present things.

Thanks to Mikael Johansson for a thorough review of my Licentiate thesis. Thank you, Carolina, for giving me a lot of great feedback during the last five years, and for your effort in making the department a great working place for everyone. Thanks to Johan Ruuskanen, Emil, and Anton Cervin for reducing the number of typos in this thesis.

The department of automatic control has been a great workplace and I would like to thank all present and former employees and guests for contributing to the good spirit. A special thanks to the aptly named A-team, consisting of Eva, Monika, Mika-san, Cecilia, and Ingrid for taking care of all administrative issues and for being great colleagues in general. Thanks to Anders Nilsson, Anders Blomdell, Pontus Andersson, and Alexander for keeping all computer systems and lab processes running smoothly, and to Leif for invaluable LaTeX help.

Thanks to everyone who has been joining disc golf, floorball, and board games, with special mentions to Albin, Gautham, Mattias, Martin M, Ylva, Jonas, Frida, Marcus G, and Marcus TA . Thanks to Nils, for your positiveness and for organizing various social events. Thanks to Emil for always being quick to laugh and for making teaching non-linear control a lot more fun.

Thanks to Calle, Torkel, Johan, PC, and Niklas for many adventures throughout the years. Thanks to Philip and Oscar for good virtual (and far to rarely, non-virtual) company. Thanks to Roger, for being an inspiring teacher, and for teaching me induction.

Thanks to my parents for always supporting me and for making me a person that does not give up easily. Finally, thanks to Christina for always being there for me.

Financial Support

This work was supported by the Swedish Foundation for Strategic Research through the project SSF RIT15-0091 SoPhy.
The author is a member of the LCCC Linnaeus Center and the ELLIIT Excellence Center at Lund University.

Contents

1

Introduction

How to best make use of available resources is an age-old problem. Classical examples include the allocation of fresh water throughout an irrigation network or the distribution of goods in a logistic network. Roughly speaking, such networks are generally operated based on steady-state considerations. That is, equilibrium flows of resources that optimize steady-state performance are identified, and control systems (if any are used at all) are used to stabilize operation around a given equilibrium. However this modus operandi really only tells part of the story. In reality, the operating points of these networks are continually changing as, for example, farmers vary the amounts of water they use to water their crops. To optimize performance it then becomes important to consider dynamic or transient effects. And these effects can be significant. For example, the efficiency of irrigation networks has been estimated to only be around 50%, with half of the losses coming from large-scale distribution losses, which occur before the water reaches the farms [Mareels et al., 2005].

In this thesis, the problem of optimally regulating the transportation of a resource throughout a network is considered. Particular emphasis is placed on scalable approaches that can handle (or even exploit) the inevitable delays resulting from transportation. Due to increasing globalization and digitalization, transportation systems often contain many transportation routes and storage locations. It is then necessary (or at least highly desirable) that the methods employed scale well as the systems grow in size[1]. This is typically not possible when using centralized approaches, where the calculation of each flow in the system at each point in time requires knowledge of the state of the entire system. Instead, one often searches for non-centralized controllers, where each point in the system only uses information from a subset of the network. The downside of this approach is

[1] Indeed these constraints are relevant even in small systems in which computational power and the communication bandwidth are limited. This could, for example, be the case for systems in rural areas.

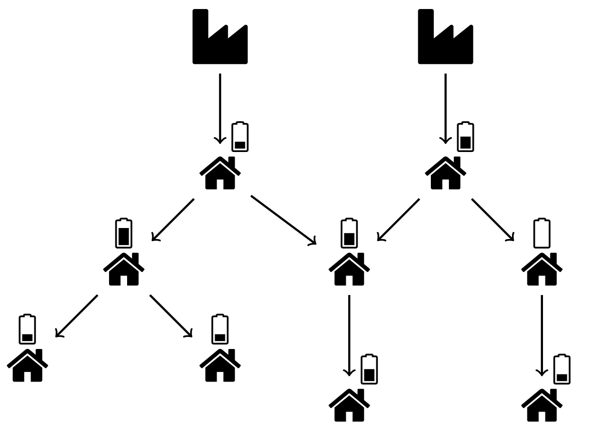


Figure 1.1 Graphical illustration of the problems considered in this thesis. The goal is to optimally distribute some quantity throughout the network subject to transportation delays.

that the performance is expected to be worse when less information is used. Examples of such controller structures are local controllers, where the calculation of each flow only uses information from the neighboring location, and decentralized controllers, where each flow is decided based only on the level in the source (or destination) node.

We consider a simple, but rather generic, model for transportation. Throughout, transportation networks are modeled using directed graphs, consisting of a set of edges and nodes. The edges allow for transportation between nodes in the graph, subject to a delay. See Figure 1.1 for a graphical illustration. This relatively simple setup covers the essence of most transportation problems. For example, it can describe the behavior of the water levels in an irrigation network on slow time scales [Evans et al., 2011] or the basic behavior of a logistics network [Subramanian et al., 2013].

The major contribution is to show how to achieve globally optimal performance (i.e. the level of performance that a controller with full state information can achieve) using a control scheme with a naturally distributed implementation. The is possible because the optimal controllers for the regulation problems considered have certain desirable structural properties. This structure allows the controller to be implemented using a sweep through the graph. The sweep starts at the nodes at the end of the graph (the leaves) and iterates upstream. Each node receives information from each of its downstream neighbors in the form of a single variable. It uses this to compute its control action, and then form a new variable that it passes upstream. This approach will be illustrated for a network with a string topology in Section 1.1. The implementation time of this control law scales with the depth of the network and represents a sort of middle ground between fully centralized and fully decentralized control. It is, therefore, best suited for applications where the underlying dynamics have fairly long time constants (such as irrigation or logistic networks). In these cases, it offers a simple and easily implementable alternative to conventional decentralized or distributed approaches while bringing the benefits of globally optimal performance. It is also shown that the parameterization of the control law can be calculated similarly, allowing the controller to be efficiently updated if the network changes.

The results discussed above holds for any directed graph without undirected cycles. They are also extended for string graphs to allow for optimal feed-forward for planned disturbances to the network. This is motivated by irrigation networks, consisting of connected canals from which farmers can take out water and use it to water their crops according to some pre-agreed schedule. However, planned disturbances could occur in many more transportation problems. The methods developed in this thesis are applied to a model for irrigation networks with third-order canal dynamics (as opposed to the first-order models used for the synthesis). This is achieved by low-pass filtering the inputs and finding a first-order approximation of the canal dynamics. The controller is compared to a simple P controller with feed-forward and an LQ controller synthesized using the third-order dynamics which gives the best possible performance. The structured controller derived in this thesis outperforms the simple P controller and almost achieves the theoretical optimum for rejection of low-pass filtered disturbances.

The approaches described above implicitly assume that all parts of the transportation network are cooperating. This thesis also studies the problem from a user perspective, where each node in the system wants to maximize their own utility, but has to pay a price for the amount of the quantity they receive. A method for calculating prices in a distributed way so that the nodes' optimal levels coincide with the social optimum, that is the levels that maximizes the sum of the utilities for all nodes, is derived. This allows for each user in the network to be satisfied with the resulting transportation assignments, even if they are not fully cooperating.

Outline

This introductory chapter will be concluded with an example problem on a string graph, for which the optimal controller, and its interesting structural properties, will be highlighted. Next follows Chapter 2 with relevant background and motivation for the thesis. This serves both as a summary and an extension of the motivation and background that is later found in the papers. The chapter starts with a discussion of the need for structured

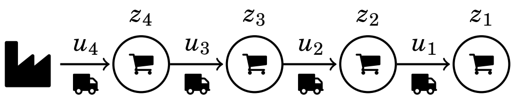


Figure 1.2 An illustration of the string graph example. There is a factory that produces a product, which is then transported to the different stores in the network. The goal is to optimize the inventory level of each store.

controllers and of previous results in the area. Then modeling approaches and control strategies for irrigation networks are introduced, followed by a introduction to modeling of logistics systems. Chapter 2 is then completed with a discussion of economical aspects relevant to the thesis. Chapter 3 contains a statement of contributions of the thesis together with a summary of the results, conclusions, and possible future work. Next follows the main part of this thesis, which is four papers written by the author.

1.1 String Graph Example

To give an introduction to the results achieved in this thesis, the optimal controller for a logistics system on string graph will be presented. The graph contains N nodes, where the inventory level of each node is given by $z_i[t]$. The goods can be transported from node $i+1$ to node i according to the choice of $u_i[t]$. Node N is the most upstream node in the network, and receives shipments from a factory which produces on demand. See Figure 1.2 for an illustration of the system.

If all transportation delays are assumed to be homogeneous, the system can be described by the following dynamics:

$$\begin{aligned} z_1[t+1] &= z_1[t] + u_1[t-1] \\ z_i[t+1] &= z_i[t] + u_i[t-1] - u_{i-1}[t], \quad 2 \le i \le N. \end{aligned} \tag{1.1}$$

All variables are relative to an equilibrium flow and a negative $u_i[t]$ corresponds to sending less goods than the equilibrium flow.

In this thesis, Linear Quadratic Control problems on the form

$$\begin{aligned} \underset{u_i, z_i}{\text{minimize}} \quad & \sum_{t=0}^{\infty} \left(r_N u_N[t]^2 + \sum_{i=1}^{N} q_i z_i[t]^2 \right) \\ \text{subject to} \quad & \text{Dynamics in (1.1)} \\ & z_i[0] \text{ given,} \end{aligned} \tag{1.2}$$

are considered. The objective is to minimize the cost due to the deviations from the equilibrium levels, $z_i[t]$, and the deviation from the equilibrium production, $u_N[t]$. There is no cost for the transportation assignments, $u_i[t]$, for $1 \le i \le N-1$. This corresponds to assuming that the transportation $u_i[t]$ is relative to an equilibrium flow, and varying the flow does not increase the cost. This could, for example, be the case when one sends more goods in a truck compared to the equilibrium level (note also for irrigation networks transportation is 'free', since it is performed by gravity).

For the problem in (1.2), the optimal outflow from node i (that is $u_{i-1}[t]$ $2 \le i \le N$), is given by[2]

$$u_{i-1}[t] = \frac{q_i}{q_i+\gamma_{i-1}}\Big(z_i[t]+u_i[t-1]\Big) - \frac{\gamma_{i-1}}{q_i+\gamma_{i-1}}\sum_{j=1}^{i-1}\Big(z_j[t]+u_j[t-1]\Big). \quad (1.3)$$

In the above, the paramter γ_i (which can be computed ahead of time) is given by the recursion

$$\gamma_1 = q_1, \quad \gamma_i = \frac{q_i\gamma_{i-1}}{q_i+\gamma_{i-1}}.$$

Furthermore, the optimal external production $u_N[t]$ is given by

$$u_N[t] = -\frac{X}{X+r_N}\sum_{j=1}^{N}\Big(z_j[t]+u_j[t-1]\Big),$$

where X is given by

$$X = -\frac{1}{2}\left(r_N-\gamma_N\right) + \sqrt{\gamma_N r_N + \frac{1}{4}\left(r_N-\gamma_N\right)^2}.$$

Why are these expressions for the inputs $u[t]$ interesting? For a node i to calculate its outflow $u_{i-1}[t]$, the node must know its own level $z_i[t]$, the incoming flow $u_i[t-1]$, and the sum

$$m_{i-1} = \sum_{j=1}^{i-1}\Big(z_j[t]+u_j[t-1]\Big).$$

This is because the optimal flow in (1.3) can be rewritten in terms of $m_{i-1}[t]$ as

$$u_{i-1}[t] = \frac{q_i}{q_i+\gamma_{i-1}}\Big(z_i[t]+u_i[t-1]\Big) - \frac{\gamma_{i-1}}{q_i+\gamma_{i-1}}m_{i-1}[t].$$

The sum m_i can be efficiently calculated by a sweep through the graph as follows. Starting with node one, in order to calculate

$$m_1[t] = z_1[t] + u_1[t-1],$$

[2] This follows from Theorem 1 and Theorem 2 in Paper I.

the node needs knowledge of its own level $z_1[t]$ and its incoming flow $u_1[t-1]$. It is natural that when node two decides its outflow $u_1[t-1]$ it also sends that information to node one. Thus $m_1[t]$ can be calculated using local information in node one. The value of $m_1[t]$ is sent upstream to node two, which needs it for the calculation of $u_1[t]$. Next, for node two, $m_2[t]$ can be calculated as

$$m_2[t] = m_1[t] + z_2[t] + u_2[t-1].$$

The first term $m_1[t]$ was already sent to node two, and the two last terms $z_2[t]$ and $u_2[t-1]$ are local information for node two, again assuming the incoming flow $u_2[t-1]$ was communicated at the previous time point. Thus the value of $m_2[t]$ can be calculated and then sent upstream to node three, which needs it for the calculation of $u_2[t]$. In general terms, $m_i[t-1]$ can be calculated recursively through the following formula

$$m_i[t] = m_{i-1}[t] + z_i[t] + u_i[t-1].$$

Thus the sum $m_i[t]$ can be calculated using a serial sweep of local communication through the graph. The aggregate $m_i[t]$ can also be used for the optimal production, which is equal to

$$u_N[t] = -\frac{X}{X + r_N} m_N[t].$$

So each node i only needs local information and the aggregate m_{i-1}, which can be calculated using only local communication. The resulting communication structure is illustrated in Figure 1.3. This structure is in contrast to most LQ-optimal controllers, where each node generally needs to know the full state of the system. The downside of this implementation method is that the implementation time will scale with the size of the network, and it is important that either the sample times are long or the communication delays are short.

Similarly, the calculation of controller parameters γ_i can also be implemented by a single sweep through the graph. This results in an efficient implementation, both in terms of communication, and calculation, which is suitable for systems that contain many nodes, or where communication is limited.

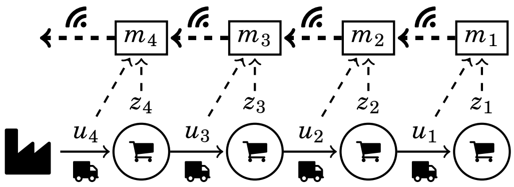


Figure 1.3 Illustration of the information flow for the string graph example. Dashed lines illustrate the communication. In order for a node i to calculate its aggregate m_i, the node needs m_{i-1} from the neighboring node, and z_i and u_i which is local information. Note that m_4 is used for the production and is thus sent from node four to the production site.

2

Background and Motivation

In this chapter, relevant background and motivation for the thesis are presented. This serves both as an extension and a summary of the motivation and background that is later given in the papers. Firstly, the issues that appear when trying to control large-scale systems, as well as previous results, are discussed. The focus is on the structural features of controllers which make them suitable for large-scale systems. Secondly, water irrigation networks are discussed. Two different modeling approaches are presented, the PDE-based St. Venant equations and modeling using system identification. Different control strategies are also discussed. Next, the similarities and differences between models used for logistics networks and the models studied in this thesis are discussed. Lastly, economical aspects relevant to this thesis are presented, including the difference, and connections between, the social optimum and a competitive equilibrium.

2.1 Structured Controllers

In this section, a background on structured controllers is given. The section begins with a discussion of the special considerations that are needed when designing controllers for large-scale systems. Next, previous results in the field of structured control are briefly presented, and their connections to the results in this thesis are highlighted.

Need for Structured Controllers

We start by motivating the usefulness of structured controllers for large-scale systems by discussing and illustrating different control strategies for a simple transportation network with a string graph topology, as illustrated in Figure 2.1. Goods can be moved between nodes connected by an edge, and produced in Node 4. The system considered is used as an example, and the discussion here holds for general interconnected systems. For each strategy, we will discuss some of the issues that appear, including the communication

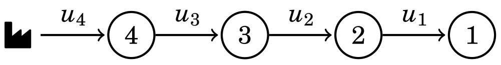


Figure 2.1 A schematic illustration of an example of the transportation systems studied in this thesis. Each input affects two nodes, except u_4. This is an example of an interconnected systems, and the discussion here holds for general interconnected systems.

requirements, and how well the controller can handle structural changes to the network.

If a controller were to be designed for this system using standard control methods, such as H_2 or H_∞ control, the controller would be static, but generally dense, i.e., the controller would have the following sparsity pattern:

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}.$$

To calculate any of the inputs u_i it is necessary to know the state of the entire system. For a small number of subsystems, this is not an issue, but as more subsystems are added, the resulting communication demands will be problematic to implement in practice. For example, both power networks and district heating networks could consist of thousands of subsystems, and if every subsystem were to communicate with each other it would lead to millions of communication channels. This would not be feasible to implement.

Alternatively, each subsystem could communicate with some coordinator. Then each subsystem would send its state to the coordinator, which then calculates all inputs and sends them to each node. This limits the number of communication channels to be proportional to the number of subsystems. However, there are still has some issues. Firstly, the amount of information received by the coordinator could be very large, leading to increased latency. Secondly, if there is a failure in the coordinator the entire system breaks down. For distributed controllers all over the system, it is likely that the network can handle the loss of a single component. Still, controllers on this form are often implemented using a central computer. Such approaches will be referred to as centralized. The resulting controller structure using a central computer is illustrated in Figure 2.2.

Another issue is if the system changes its operating conditions, for example when it starts to rain in an irrigation network, which can result in changed equilibrium flows and set-points. The dynamics around the new equilibrium flow can be different compared to the original equilibrium flow. As a consequence, it is important in many applications that the controller

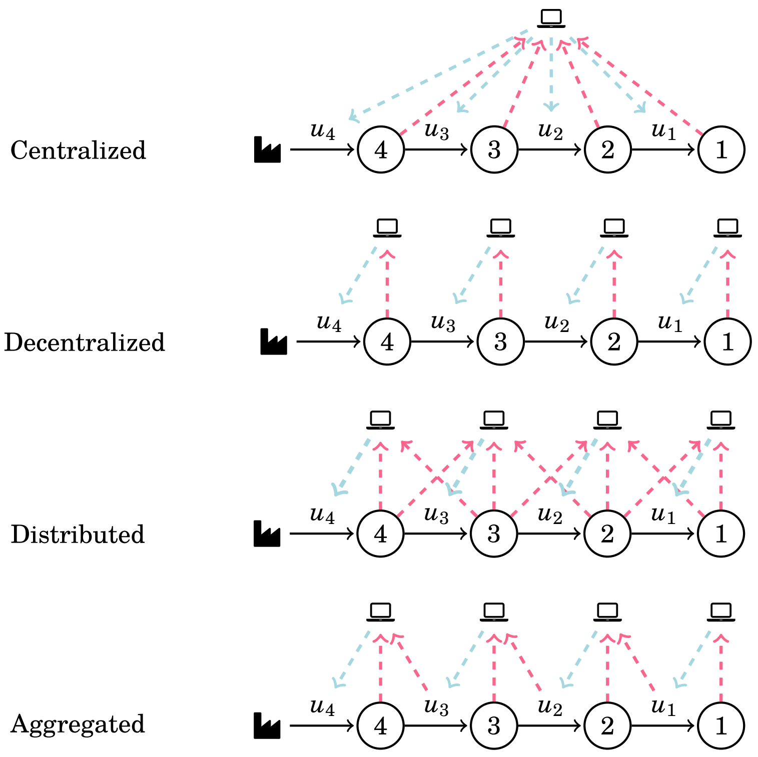


Figure 2.2 Illustration of different controller structures. The centralized controller uses a central computer, while the other controller structures use one computer for each node. The red arrows illustrate the measurements, and the blue arrows illustrate the communication of the computed inputs from the computers.

can handle these changes. Redesigning a centralized controller every time the network changes can in many cases be infeasible due to the large computational cost for doing so. One alternative is to design a controller that is robust to variations in operating conditions. However, performance will generally be better if the controller is designed for the current operating condition. The network can also change due to the addition or removal of components to the system, for example, in a power system where different producers are added or removed due to maintenance or shifting weather conditions. When new components are added to a system, the controller must be updated in order to handle the new components.

All of the issues discussed above are less impactful when using decentralized control. This is when each subsystem has its own controller, which is designed only for that subsystem. This control structure can be described by the sparsity pattern

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} * & 0 & 0 & 0 \\ 0 & * & 0 & 0 \\ 0 & 0 & * & 0 \\ 0 & 0 & 0 & * \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

and is illustrated in Figure 2.2. There can generally be an improved performance when dynamic controllers are used. This structure is very simple to implement, as each controller only needs a measurement from its subsystem, and thus no communication is needed.

When it comes to the synthesis of decentralized controllers there are two possibilities. The synthesis can either be carried out in a centralized fashion, with all controllers being designed in tandem, or in a decentralized way, where all controllers are tuned independently. A centralized synthesis will have the same problems as the centralized controllers for changes in the network. If the controllers are synthesized in a decentralized way, changes in the network's operating conditions can easily be handled. Then each controller can be re-tuned independently. Similarly, if new subsystems are added, only controllers for the new subsystems need to be designed, and all the old controllers can remain untouched. However, new difficulties arise, as it will in general be difficult to design the controllers so that performance of the entire system is optimized, and special care must be taken to verify stability in a distributed way. Centralized synthesis of distributed controllers will generally give better control performance than decentralized synthesis. Still, the performance will almost always be worse than when using a centralized controller, as less information is available for each decision.

A middle ground between the centralized and decentralized controllers is to design controllers that has information from some nodes, for example, from the neighbors. This would give the following sparsity pattern (which again may by dynamic):

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} * & * & 0 & 0 \\ * & * & * & 0 \\ 0 & * & * & * \\ 0 & 0 & * & * \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}.$$

The resulting controller structure is illustrated in Figure 2.2. This structure is still very efficient to implement. While it requires some communication, the amount of communication for each node is independent of the network size. However, the issue of ensuring stability and optimizing performance when each controller only considers a small part of the system remains.

Furthermore, the different controllers have a higher degree of interaction compared to the completely decentralized case. One way of overcoming this issue is to design all the controllers together. However, such design methods retain the problem of centralized controllers when it comes to change of operating conditions or the addition or removal of subsystems.

Yet another alternative, which is the structure of the controllers in this thesis, is to aggregate information through the graph. For example, using the following pattern:

$$\begin{bmatrix} * & * & 0 & 0 \\ 0 & * & * & 0 \\ 0 & 0 & * & * \\ 0 & 0 & 0 & * \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} * & 0 & 0 & 0 \\ 0 & * & 0 & 0 \\ 0 & 0 & * & 0 \\ 0 & 0 & 0 & * \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}. \tag{2.1}$$

An illustration is given in Figure 2.2. This controller structure can be implemented using local communication of states and inputs. For example, u_3 can be calculated as

$$u_3[t] = c_1 x_3[t] + c_2 u_4[t].$$

This requires a sweep through the graph to implement the optimal controller, which means that the execution time will scale with the size of the network. Thus it is only suitable for systems with low sample rates. However, it allows for an efficient way for information to propagate throughout the entire network. The structure in (2.1) only allows for information flow in one direction, and corresponds to decentralized control with feed-forward. However, it is also possible to aggregate information in both directions, as seen in this thesis. Controllers designed this way will likely benefit from all the controllers being designed together. However, for the controllers in this thesis this can be done in an efficient and distributed way.

Enforcing Structure

The issues discussed in the previous section have led to an effort in designing controllers with a structure that is suitable for large-scale system applications. In this section we highlight some of the previous work where the structure is enforced on the controller. In the next section we will give some examples for when the structure follows from the problem without imposing any constraints in the design. These problems are typically rather specialized, and methods based on enforced structure can be applied more generally. However, when the structure follows from the plant the controller is globally optimal. Furthermore, these cases can be used as an inspiration for new structures that can perhaps be enforced in more general cases, or as motivation for heuristic methods.

Early work on structured control includes the study of team games [Marschak, 1955; Radner, 1962]. In a team game, a set of cooperating decision-makers who have access to different information attempt to make decisions that are optimal for the entire team. Here the initial motivation was to study organizations. The idea was later extended to a control setting. For example, in [Ho and Chu, 1972] it was shown that for a partially nested information constraint, the optimal LQ controller is linear. In [Sandell and Athans, 1974] it was shown that the optimal controller for a two decision maker problem, where the measurement from one decision-maker is known for the other decision-maker one time unit later, is linear.

For infinite horizon LQ control of time-invariant systems with no structural constraints, it is well known that the optimal controller is linear and memoryless, and can be found by solving a Riccati equation. The examples above show that controllers with information constraints *can* be linear. However, this does not hold in general as highlighted in the important counterexample given in [Witsenhausen, 1968]. The presence of information constraints can also make the problems very hard to solve. For example, when considering the design of optimal decentralized controllers, the resulting computational problems can be NP-hard [Blondel and Tsitsiklis, 1997].

Heuristic Approaches. Given the difficulties in designing optimal structured controllers, it is common to not enforce optimality, but rather design controllers using some heuristic, which promotes the convergence of structured controllers. Multilevel or hierarchical control is one such strategy, which relies on splitting the control problem into different layers. The lowest layer typically consists of decentralized controllers designed using local models, and the higher layers are designed to coordinate the lower layers. This approach has for example been applied to power systems [Mansour and El Abiad, 1977; Schweppe, 1978] and manufacturing systems [Jones and McLean, 1986; Boukas et al., 2003].

Another method to design structured controllers is via decentralized or distributed PI controllers. However, there exists a class of systems where such a controller can never reject a constant disturbance and constant measurement noise [Andreasson et al., 2014]. It is also shown that letting the integral part depend on neighboring nodes extends the class of systems for which the steady-state error is zero. However, distributed PI control is able to stabilize district heating networks [De Persis et al., 2014]. Yet another alternative to enforcing structure on the plant is to instead have sparsity promoting terms when optimizing over possible controllers, as in [Fardad et al., 2011] and [Lin et al., 2013].

The problem can also be tackled heuristically by finding the equilibrium that maximizes the static performance. Then decentralized controllers can be designed to stabilize the optimal equilibrium. This method has for example been applied in internet congestion control [Kelly, 2000] and power systems [Kundur, 1994]. However, if the operating conditions change often, and thus the optimal equilibrium also changes often, then the transient of the system can be an important part of the performance of the system.

Optimal Structured Controllers. Even though the general problem of finding the optimal controller subject to structural constraint is very hard, there exist results in the literature for certain classes of problems.

Convexity has been a recurring theme when it comes to finding optimal structured controllers. If the resulting optimal control problem is convex, it can normally be solved using standard convex optimization methods. An important contribution was given in [Rotkowitz and Lall, 2006], where sufficient conditions for convexity were derived by defining the notation of quadratic invariance. It was later shown that it is also a necessary condition [Lessard and Lall, 2011]. For networks with delays, it was shown that the synthesis of local controllers is convex if the controllers can communicate with each other faster then the subsystems affect each other [Rotkowitz et al., 2010]. To just name a few examples, quadric invariance led the way to an optimal controller for a decentralized two-player problem [Lessard and Lall, 2012] and a characterization of the optimal distributed controllers subject to delays constraints [Matni, 2014].

The convex optimization approach often leads to centralized synthesis, which could be problematic if the network often changes. In [Langbort et al., 2004] distributed synthesis algorithms are derived for H_∞ control over arbitrary graphs.

More recent work include system level synthesis, where instead of optimizing over possible controllers, one optimizes over possible closed-loop systems responses [Wang et al., 2018; Anderson et al., 2019]. This can be seen as an extension of the quadratic invariance theory, as it is shown that the control problem that allows for a convex representation using quadratic invariance is a special case of the class of problems that can be solved using system level synthesis. Furthermore, for certain cases, and using parallel computation, the synthesis can be of order $\mathcal{O}(1)$.

The dominant information structure in the literature is that the information propagates a limited number of steps in the graph between every sample. Still, other approaches exist, for example in [Shah and Parrilo, 2013] where a class of optimal structured controllers with a similar structure to the one found in this thesis is derived. There the optimal H_2 controller for systems described by partially ordered sets (poset) is derived. By decoupling the problem, the solution is more computationally efficient compared to a solution based on quadratic invariance. The dynamics on the posets are restricted so that state i and input i can only affect a subsystem j if $i \preceq j$. The controller is restricted to respect the same structure, that is node i

can know the state of node j if $i \preceq j$. This structure is similar to the one in this thesis. However, for the results in this thesis, there is no benefit in estimating the unknown states, which there is in the mentioned work.

Decentralized Stability Verification. An issue with optimal structured controller is that they typically require centralized synthesis. This require full knowledge of every subsystem in the plant. However, decentralized synthesis requires decentralized stability verification. One way of conducting decentralized design using only local models is to use Passivity based design, see for example [Ortega et al., 2008] for an introduction. A Nyquist based scalable decentralized stability criterion for a network was given in [Lestas and Vinnicombe, 2006], where each subsystem only need knowledge of its own dynamics and those of its neighbors. In [Kao et al., 2009], an alternative graphical robust stability test for interconnected systems, where only an interconnection matrix and local dynamics are needed, was derived.

Achievable Performance. It is in general unknown how communication constraints affect the achievable performance of the control system. However there exist results for certain problem classes, and some of those results are presented here.

In [Bamieh et al., 2012], the ability of local feedback to maintain coherence in systems of different dimensions is studied. A more specialized case is studied in [Pates et al., 2017], where it is shown that for a platoon of vehicles an accordion-like motion will emerge for any controller using only local measurement. In [Langbort and Delvenne, 2010] it is shown that for a class of linear time-invariant discrete time systems a controller without communication is at least twice as bad as the optimal controller in the worst case. The trade-off between the model information used in the controller and the closed-loop performance was studied in [Farokhi et al., 2013].

Structure From the Plant

In some cases, the structure need not be enforced, but rather follows by itself from the unconstrained problem. The results in this thesis fall within this category. The downside is that the resulting controllers are only optimal for the specific problems. However, when possible, it allows for the usage of results from the general theory of control, such as robustness and performance guarantees. Furthermore, controllers synthesized in this way are often more transparent and easier to understand.

In [Madjidian and Mirkin, 2014] it is shown that a set of wind turbines can be optimally controlled using a combination of a distributed control and a rank one term depending on the average of all systems. Similar results can be achieved when there are a large number of systems, and each subsystem is only affected by the average of the other systems. By using a mean field approach, one can then control a large population, that does not even need to be cooperating. The operator only needs to send out a parameter based on the average of the entire system to the subsystems, for example by deciding the electricity price. Examples include control of charging for electrical vehicles [Parise et al., 2014], and demand management of electric loads [Grammatico et al., 2015]. A benefit of the mean-field approach is that it allows controlling agents, where each agent makes the choice that maximizes its own utility.

For infinite-dimensional spatially invariant systems it is shown that the optimal controller for quadratic objectives has an inherent degree of decentralization [Bamieh et al., 2002]. For H_∞ control it is harder to know whether there exists a structured optimal controller as the optimal controller is not unique. For systems with symmetric and Hurwitz state matrix, it is shown in [Bergeling et al., 2020] that there exists a structured H_∞-optimal controller that is suitable for distributed implementation.

2.2 Modeling and Control of Water Irrigation Networks

In this section an overview of approaches to modeling and control of irrigation networks is given. We start with a general description of what an irrigation network is and what the objectives are when controlling them. Next different modeling approaches will be presented. Lastly, the section is concluded with an overview of different controller structures for irrigation networks.

Water irrigation networks consist of a series of gates and water pools. Each gate allows water to be transported from one pool to the next, either by the use of gravity or by using a pump. Within each pool there can be one or more off-takes, which allows water to be taken from the channel and into a farm or a secondary channel. The off-takes are often only gravity-powered and require a certain water level to be used. Thus it is important that the water level in each pool is sufficiently high to allow for the off-take to be used. This often leads to the irrigation networks being run conservatively, which gives unnecessarily large water losses [Weyer, 2008].

As 70% of all fresh water is used for irrigation, improving the efficiency of irrigation networks will have a large effect on the amount of fresh water that is available. In Australia the water efficiency in the irrigation networks was estimated to be less than 50%, that is half of the supplied water was not used by the crops. Half of the losses were due to the large-scale distribution losses, which occur before the water reaches the farm [Mareels et al., 2005].

The control of an irrigation network is naturally split into two parts: a supervisory controller providing set-points for each canal, and channel controllers designed to achieve the given set-points [Cantoni et al., 2007]. This presentation focuses on channel controllers.

Control Objectives

It is important that the water level in each pool is sufficiently high to allow the off-takes to be used. On the other hand, higher water levels might lead to more seepage and spillage throughout the network and a higher outflow at the end of the network. Furthermore, if the water level changes too much, the canals might be worn down. The control objectives for an irrigation network can be summarized in terms of the following aspects [Weyer, 2008]:

- **Setpoint Regulation:** The water level must be kept above a certain level to allow for off-takes into farms and secondary channels. However, higher levels lead to more water wastage, so the water levels should not be higher than necessary.
- **Rejection of Load Disturbances:** The main disturbances affecting the system are the off-takes to farms and secondary channels. These flows could be either scheduled or unscheduled.
- **Flow Over Last Gate:** A big source of water loss is due to the outflow at the end of the channel. By regulating the flow over the last gate, these losses can be minimized.
- **Gate Movement:** Fast gate movements can introduce waves in the pools, which should be avoided. Furthermore, the power supply might be limited, for example, if the gates are powered by solar panels.

The water loss of the systems is mainly due to oversupply, which results in spillage in the pools and water-flow over the last gate [Cantoni et al., 2007]. Thus it is expected that the water loss can be reduced by closing the loop and allowing for the systems to be regulated more precisely and less conservatively.

Modeling Approaches

In this section different approaches to modeling the water levels in irrigation networks are introduced. The first is based on a set of partial differential equations called the Saint-Venant equations. While accurate, this model is hard to use in practice. We then outline two different system identification approaches. These can be useful both for the synthesis and evaluation of the controllers.

The Saint-Venant Equations. Assuming that each pool has a rectangular cross-section, the water level in the pools can accurately be modeled using the Saint-Venant Equations [Mareels et al., 2005]. The first equation,

$$\frac{\partial A}{\partial t} + \frac{\partial Q}{\partial x} = -q_{\text{off}}(x, t), \tag{2.2}$$

models the mass balance. The second equation,

$$\frac{\partial Q}{\partial x} + \left(\frac{gA}{B} - \frac{Q^2}{A^2}\right) + \frac{2Q}{A}\frac{\partial Q}{\partial x} + gA(S_f(Q, A, x, t) - \bar{S}(x)) = 0, \tag{2.3}$$

models the momentum balance. In the above, t is time, and $0 \leq x \leq L$ is the position along the pool of length L. $A(x, t)$ is the cross sectional area of the water and $B(x)$ the width of the pool. $Q(x, t)$ is the total flow through the cross section at position x and time t. The variable $q_{\text{off}}(x, t)$ is the off-take at position x, modeling both farm off-take, seepage, and evaporation. $\bar{S}_i(x)$ is the slope of the pool and g the gravitational acceleration. $S_f(Q, A, x, t)$ is a nonlinear friction model, for which there exists empirical formulae. Finally the boundary conditions are given by the inflow at the beginning of the pool and the outflow at the end of the pool, given by $Q(0, t)$ and $Q(L, t)$ respectively.

The model described by (2.2) and (2.3) is a nonlinear partial differential equation containing several parameters. To be able to use it, these parameters must be estimated. Simulating the model also requires some effort. First, an initial condition must be derived, typically from some steady state. The simulation then requires a numerical integration scheme, which will be computationally heavy, especially for large systems.

To use the Saint-Venant equations for controller synthesis typically requires a linearization around an equilibrium point, see for example [Litrico and Fromion, 2002]. A benefit of using this approach, compared to the system identification approach discussed next, is that it can require less data, especially if the physical parameters for the canals are already known. It can also allow for co-design, where the canals and the controllers are designed together.

System Identification Approaches. The off-takes are often close to the gates, and any waves earlier in the pool will in most cases reach the gate. Thus controlling the water level at the gates is normally sufficient. A model for the water level at the gate can be found using standard system identification methods, and in [Weyer, 2001] it was shown that a model on the form

$$\dddot{y}(t) + a_1\ddot{y}(t) + a_2\dot{y}(t) = \text{inflow} - \text{outflow}$$

gives a fairly accurate description of the water level at a gate. The model consists of an integrator and a second-order system. The integrator corresponds to maintaining the mass balance, and the second-order system models the wave dynamics. A model with this structure can easily be used to simulate the system and give insight into how waves can be avoided. However, if the controller is slow enough to avoid waves in the pool the model used for control design can be further simplified.

Mass Balance. If the wave dynamics are neglected or assumed to be small, the following simple mass balance model is sufficient to model the water level at the gate,

$$\alpha \dot{y}_i(t) = \text{inflow} - \text{outflow}.$$

In the above, α_i is the surface area of pool i. This simplified model is often used to model the water levels on slow time-scales [Cantoni et al., 2007]. For both the first and third-order model, the inflow and outflow are given by

$$\text{inflow} = \gamma_i h_i^{3/2}(t - \tau_i), \quad \text{outflow} = \gamma_{i+1} h_{i+1}^{3/2}(t),$$

where h is the water level relative to the gate position, which depends both on the water level y_i and the gate position, and γ_i is a constant depending on the gates geometry. Letting $u_i = \gamma_i h_i^{3/2}$ a standard linear system with a time delay is achieved,

$$\alpha \dot{y}_i(t) = u_i(t - \tau_i) - u_{i+1}(t).$$

This model has previously been used for controller design, see for example [Schuurmans et al., 1999; Litrico and Fromion, 2005].

Approaches to Control of Irrigation Networks

Control of irrigation networks can be split into three different approaches, decentralized, distributed, and centralized. In this section, the three approaches will be briefly presented.

Decentralized. There are two possibilities for decentralized control of irrigation networks. Each gate can either regulate the water level just before the gate, called local upstream control, or regulate the water level at the next downstream gate, called distant downstream control [Litrico et al., 2003]. The distant downstream configuration requires some communication, in that each gate must receive the measurement for the pool it is to control from the next gate downstream. Meanwhile, the local upstream configuration requires no communication and the pool can be controlled without introducing delays. This would at first sight seem like the best configuration. However, the local upstream configuration typically leads to more water wastages [Weyer, 2003]. For example, when the demand is decreasing, the local upstream control will discharge any unwanted water over the last gate. There can also be a problem when the demand is increasing, as the controller will try to get more water from downstream, and the water cannot flow upstream. Both configurations are depicted in Figure 2.3.

In either case, a common choice is to use a PI controller. The controllers can be designed both using system identification models [Weyer, 2008] or based on the St-Venant equations [Litrico et al., 2003]. It might also be necessary to augment the controllers with a low-pass filter to avoid exciting any waves in the canal.

There is a fundamental limitation when using the distant downstream control in a trade-off between, on the one hand, set-point regulation and disturbance rejection, and on the other hand, error-propagation in the network [Cantoni et al., 2007]. Intuitively, if the gain of the controllers is low, the disturbance rejection as well as the error propagation, is slow. On the other hand, if the gains are high, the disturbance rejection will be fast, but the error propagation will be increased.

Distributed. A simple way to improve the performance of the distant downstream control is to add a feed-forward compensator. The planned off-take in the pool to be controlled, as well as the flow out of the downstream gate, can be used for feed-forward. The resulting controller structure is illustrated in Figure 2.3. The addition of the feed-forward will decrease the limitations for the trade-off just discussed. This can be done for a PI controller by adding a feed-forward compensator [Weyer, 2008]. A more principled way of incorporating feed-forward is taken in [Li, 2014] and [Cantoni et al., 2007], where the optimal trade-off can be found using H-infinity loop-shaping. An issue with the feed-forward approach is that the flow at gate i at time t depends on the flow at gate $i-1$ at time t. Thus any implementation would either require a serial sweep, just as the controllers studied in this thesis, or use feed-forward on $u_{i-1}[t-1]$ for the calculation of $u_i[t]$.

More involved distributed approaches include [Lemos and Pinto, 2012], where LQ controllers are designed for each pool. The controllers use an iterative coordination scheme to improve the overall performance of the system. For every time sample, each gate calculates a candidate flow and sends it to both neighbors. Then each gate calculates a new candidate flow based on what the neighbors' candidate flows were. This continues until a stopping criterion has been met, for example, a maximum number of iterations. A similar approach is taken in [Negenborn et al., 2009a] using model predictive control. A downside with these coordination approaches is that while each node only communicates with its neighbors, it typically does so multiple times, with the performance of the controller increasing with the number of iterations. An alternative approach is taken in [Kearney et al., 2011]. There the *spare supply capacity* is used to try to improve the performance. The spare supply capacity for a pool is the set of all outflow loads for which there exists a set of flows upstream so that all the upstream constraints are satisfied. This needs to be calculated through a sweep starting at the most upstream pool, iterating downstream. A similar approach is taken in [Negenborn et al., 2009b], where a non-iterative distributed MPC

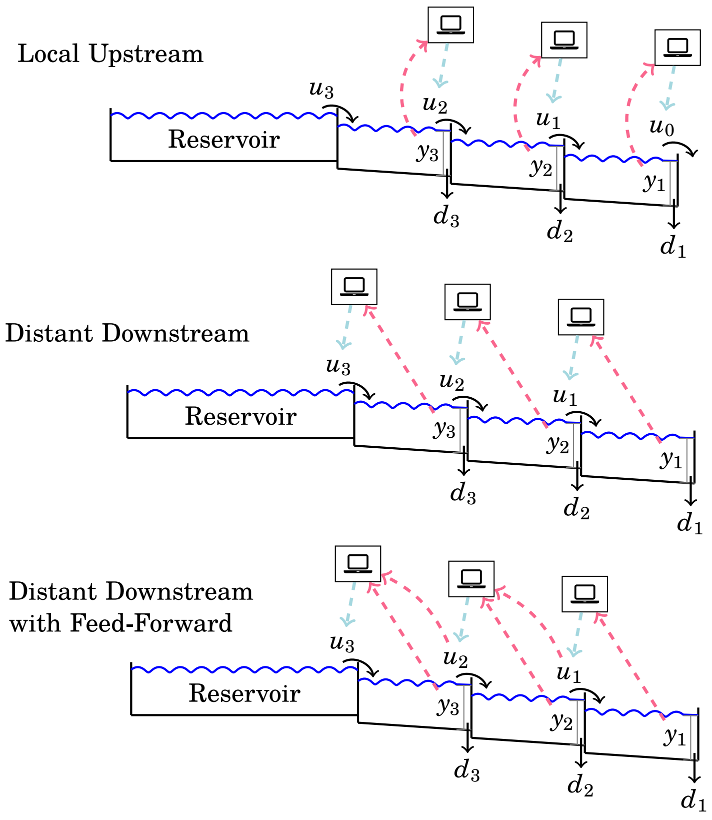


Figure 2.3 Illustration of different controller structures for irrigation networks. The top figure is local upstream, the middle figure is distant downstream and the bottom figure is distant downstream with feed-forward. Red dashed lines indicate communication and blue dashed lines show which input is being calculated by each controller. For the feed-forward case, both measurements, and the downstream flow u_i is used in the controller calculation.

controller is used. Again, the control at the most upstream canal is calculated first, and the rest follows in an iterative fashion. Interestingly, both these approaches iterate in the opposite direction to the iterations for the results in this thesis.

Centralized. Centralized LQ with feed-forward was used in [Weyer, 2008]. First and second-order models were used for the controller synthesis (depending on the length of each pool). Waves were avoided by penalizing a high-pass filtered version of the water levels. The performance when the predicted off-takes did not align with the true off-takes was investigated. For a difference of 60 minutes, the controller with feed-forward performed worse than the one without. This indicates that it is important that the farmers take out water close to when they ordered it if feed-forward is to be used. The performance was compared to a PI controller, and the LQ controller performed better.

Another centralized approach is to use MPC. In [Nasir et al., 2017] a Stochastic MPC controller is used to calculate the canal set-points. Here the disturbances are seen as stochastic disturbances, and the goal is to choose set points so that the risk that certain levels are achieved is minimized. Acting on the references is a distant downstream PI controller with feed-forward on the downstream pool.

2.3 Modeling of Logistics Systems

In this section, a brief overview of approaches to modeling logistics systems is given. The goal is to both highlight the similarities with the type of problems studied in this thesis, and show some of the differences, thus illustrating the extent to which our methods apply to these systems.

A classical problem in inventory control is the news-vendor problem. A salesman can each day decide how many newspapers to order. The salesman then tries to sell his newspapers on the street. Tomorrow no one will want to buy yesterday's paper, so the problem is static. In this problem, there is no transportation, and the most important aspect for solving the problem is forecasting the demand. This problem can be extended to a setting with multiple vendors and distribution centers and goods that retains their value over time. Still, having a forecast of the demand is necessary for achieving satisfactory performance. However, the efficiency of the transportation of the goods between the nodes is also important for the performance of the system. A two-tiered approach is natural, where the problem is split into first finding optimal set-points based on the demand forecasts and then optimizing the transportation assignments given the set-points.

A natural model for the inventory level for nodes in a graph is [Subramanian et al., 2013; Lin et al., 2004]

$$z_i[t+1] = z_i[t] + \sum_{j\in\mathcal{P}(i)} u_{ij}[t-\tau_{ij}] - \sum_{h\in\mathcal{C}(i)} u_{hi}[t], \tag{2.4}$$

where $\mathcal{P}(i)$ is the set of parents for node i (that is the nodes that send goods to node i) and $\mathcal{C}(i)$ is the set of children for node i (that is the nodes that receive goods from i). The variable $u_{ij}[t-\tau_{ij}]$ is the incoming transportation from node j and u_{hi} is the outgoing transportation towards node h, arriving τ_{hi} time units later. This model is studied in the thesis, and also extend to handle planned disturbances, corresponding to changes in the demand forecasts. The demand fluctuations can also be modeled by introducing a white noise disturbance at each node.

It is common to introduce the inventory position $p[t]$ and use it as a control variable, see for example [Hoberg et al., 2007]. The inventory position is the inventory level plus the goods in transit towards the node,

$$p_i[t] = z_i[t] + \sum_{j\in\mathcal{P}(i)} \sum_{d=1}^{\tau_{ij}} u_{ij}[t-d],$$

which has the following dynamics

$$p_i[t+1] = p_i[t] + \sum_{j\in\mathcal{P}(i)} u_{ij}[t] - \sum_{h\in\mathcal{C}(i)} u_{hi}[t].$$

Interestingly, this methodology appears naturally in the controllers in this thesis, for example in (1.3).

Backorders

The model in (2.4) can be extended by introducing states $b_i[t]$, modeling backorders, and inputs $o_{ij}[t]$, modeling orders. See for example [Subramanian et al., 2013]. The backorders are the orders that have not been met, and can be described by

$$b_i[t+1] = b_i[t] + \sum_{h\in\mathcal{C}(i)} o_{hi}[t] - \sum_{h\in\mathcal{C}(i)} u_{hi}[t].$$

The backorders and the orders are virtual states and inputs with no physical interpretation. However, the backorders can be an important part of the costs function for each node. From each node's perspective, the orders and the inflows can be seen as external influences and the node has to decide the outflows. Backorders arise when a node cannot, or chooses not to, meet an order. Backorders should be avoided as the company behind the order that is not met might look for a different seller in the future.

For the most part, the controllers considered in this thesis optimize the performance for the entire network. In this settings, backorders are less relevant between nodes in the system, as they are assumed to be fully cooperating. However the nodes in the systems could interact with external actors, and then backorder could be of interest. For example, in the setup in Paper II, the external disturbances could be replaced by external orders. Backorders could to some extent also be modeled by the theory derived for general convex cost functions in Paper I. Instead of introducing the backorder state, one could let negative levels z_i model a backorder, and make the cost function $f(z_i)$ for negative levels describe the cost of backorders. However, this alternative is limited in two ways. Firstly, it only allows for backorders when the inventory is completely emptied. Secondly, it only allows for backorders on orders that are from outside the network. The general convex cost function could also handle the case of state constraints (but not input constraints). How this compares to the computational efforts of MPC has not been explored.

Minimal Orders and Restock Policy

The model in (2.4) assumes that it is possible to transport goods at regularly spaced intervals. While this might be possible, it is preferable to avoid it, as the transportation costs would likely be unnecessarily high. An alternative is to design a restock policy for when to order more goods. By waiting to order until the order quantity is large, unnecessary transportation can be avoided. If one waits too long, on the other hand, backorders might occur as the inventory was not replenished fast enough.

Examples of restock policies include the (s, S) policy, where an order is placed to restock up to S when the inventory position reaches s [Zheng and Federgruen, 1991]. Another alternative is an (R, Q) policy where a batch of size Q is ordered when the inventory position is below R [Axsäter, 2005]. While these policies are simple to implement, the design can be more involved. These order policies are similar to event-based control, see for example [Aström, 2008]. Whether it is possible to extend the results in this thesis to an event based framework is an open question.

Bullwhip Effect

The bullwhip effect is an important concept in supply chains. This is when a change in demand downstream is amplified higher upstream, see for example [Lee et al., 1997]. A classic example of this phenomenon is when demand fluctuations at a retailer are amplified by a distributer, and then amplified even further by a producer. A way to reduce the bullwhip effect is to increase information sharing, and we note that a benefit of our method is that it should never show any bullwhip effect.

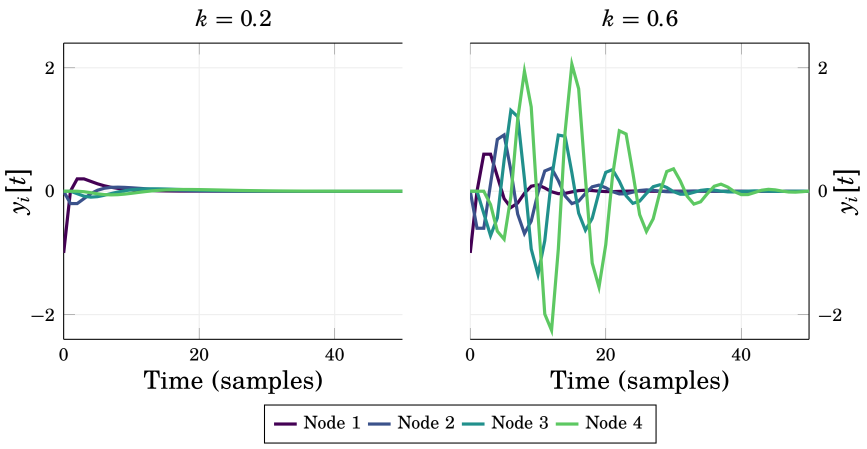


Figure 2.4 Illustration of the bullwhip effect for a simple logistics systems. Both figures have the same initial conditions. In the right figure the controller is poorly tunes, leading to a bullwhip-like effect.

The bullwhip effect is often due to poor forecasting. However, it can also occur due to inefficiencies in the transportation system when delays are present. We illustrate this with an example. Consider the example in the introduction with the dynamics in (1.1) and with a P controller in each node on the form

$$u_i[t] = ky_i[t].$$

The initial conditions are given as $y_1 = -1$ and $y_i = 0$ for $i \geq 2$, corresponding to a changed demand in the most downstream node. The system is simulated for two controllers, one with $k = 0.2$ and one with $k = 0.6$. The results can be seen in Figure 2.4. It can be seen that for $k = 0.6$ there is a bullwhip-like effect, where the level fluctuations are larger for nodes higher up the graph.

2.4 Economical Aspects

Classically, control methods are applied to systems where all components have the same owner, such as a factory. Then the goal is to maximize the performance of the entire system in a cooperative manner. However, control methods are also applied to systems where different components have different owners. Examples include power grids, where solar panels, wind turbines, and other power plants are owned by different persons or companies, and interact through a grid owned by another set of companies. In the case of water irrigation networks, the system interacts with many different farmers, and part of the system design is to decide how much water each farmer is allowed to take out from the system.

Despite this, the utilities of the subsystems are often not considered individually, and instead, the welfare of the collective is maximized (that is the sum over all utilities). However, considering the utilities for every subsystem can be important for several reasons. The owners of the different subsystems might care more about their own utility than the utility of the entire system. It can then not be expected for the different subsystems to make altruistic decisions. Instead, there must be some mechanism that makes the individual users' choices align with the social optimum. On the other hand, maximizing the total performance of the system gives the most reward to share. One way to share the benefits of the optimal performance is to design prices so that the users are compensated for making socially optimal choices (or penalized for not doing so). This can for example be achieved by having road tolls in transportation networks, or charging farmers for the water withdrawn from an irrigation network.

In this section, we will illustrate the difference, and the connection, between what is optimal for the individual users in the system and the maximum performance of the entire system. We motivate our treatment by starting with a brief introduction to two classical problems in economics. These two problems are the competitive market equilibrium and welfare maximization. We will also show how the problems studied in this thesis are a natural extension of the welfare maximization problem.

Competitive Market Equilibrium

A classical problem in economics is to find the equilibrium in a market. If every agent in a system considers multiple commodities at the same time, then an equilibrium is called a general equilibrium. We will limit ourselves to studying one commodity. Then any equilibrium is called a partial equilibrium.

We illustrate the concept with an example. A set of n agents are each given an initial endowment w_i of some commodity. The agents can buy from and sell to each other, and it is assumed that the price p is the same between all agents. Let x_i denote the amount of the commodity for each agent, and assume that each agent values that amount according to a utility function $U_i(x_i)$. Then the payoff of each agent is given by

$$U_i(x_i) - p(x_i - w_i). \tag{2.5}$$

For the market to be in equilibrium the total demand must be equal to the total supply, that is the nodes' choices of x_i must satisfy:

$$\sum_i x_i = \sum_i w_i. \tag{2.6}$$

The supply is fixed, but the demand depends on the price. So a price can only be an equilibrium price if the corresponding choices of x_i satisfy (2.6).

In general, the agents' choice of x_i could depend on how it affects the prices p. If there are many agents, each agent's effect on the prices is negligible, and it can be assumed that each agent makes its choice x_i as if it had no effect on the price. This is called a competitive market, and then each agent will choose the x that maximizes (2.5). Then, assuming the node utility $U_i(x_i)$ is monotonically increasing, the node demand for any given price p is increasing if

$$\frac{\mathrm{d}}{\mathrm{d}x_i}U_i(x_i) > p,$$

and decreasing if

$$\frac{\mathrm{d}}{\mathrm{d}x_i}U_i(x_i) < p.$$

Thus at any equilibrium it will hold for all nodes that

$$\frac{\mathrm{d}}{\mathrm{d}x_i}U_i(x_i) = p.$$

Welfare Maximization

For the welfare maximization problem, the goal is to find the amount of quantity for each agent so that the welfare of the entire system is maximized. For a general welfare function W this can be formulated as an optimization problem,

$$\begin{aligned} \text{maximize} \quad & W\big(U_1(x), \dots, U_n(x)\big) \\ \text{subject to} \quad & \sum_i x_i = \sum_i w_i. \end{aligned} \tag{2.7}$$

We assume $U_i(x)$ only depends on x_i and the welfare function W is given by the sum of the individual utilities.

Actually, every welfare maxima is a competitive equilibrium [Varian, 2003]. We illustrate this point via an example. Consider the welfare maximization problem

$$\begin{aligned} \text{maximize} \quad & \sum_i U_i(x_i) \\ \text{subject to} \quad & \sum_i x_i = \sum_i w_i \end{aligned} \tag{2.8}$$

where U_i is concave. The Lagrangian of the system is given by

$$\mathcal{L}(x, \lambda) = \sum_i U_i(x_i) + \lambda(\sum_i w_i - \sum_i x_i).$$

There exists x^* and λ^* so that x^* is the maximizer of (2.8) and

$$\frac{\partial \mathcal{L}}{\partial x_i}(x_i^*, \lambda^*) = \frac{\mathrm{d}}{\mathrm{d}x_i} U_i(x_i^*) - \lambda^* = 0$$
$$\frac{\partial \mathcal{L}}{\partial \lambda}(x_i^*, \lambda^*) = \sum_i w_i - \sum_i x_i^* = 0.$$

Then if we pick the price p to be $p = \lambda^*$, the first equation implies that x_i^* is the maximizer of the node utility (2.5), and the second implies that the supply is equal to the demand (that is (2.6) holds). We have thus constructed a competitive equilibrium based on the welfare maximizer.

Dynamical Welfare Maximization

Consider the problem in (2.7). A natural extension is to only allow transportation between agents that are connected. This gives the following problem

$$\begin{aligned} \underset{x_i,\ u_{ij}}{\text{maximize}} \quad & W\Big(U_1(x_1), \dots, U_n(x_n)\Big) \\ \text{subject to} \quad & x_i = w_i + \sum_j u_{ij} \\ & u_{ij} = -u_{ji}. \end{aligned}$$

In the above the sum is over nodes j that are connected to i and u_{ij} is the transportation from node j to node i, where a negative value implies that the transportation is from node i to node j instead. It must hold that $u_{ij} = -u_{ji}$ so that what node i receives from node j actually leaves node j. This modification does not change the optimal x as long as the corresponding graph is connected, as the goods can then be transported between all the nodes in the system.

However, if we also introduce transportation delays, the problem changes drastically. The problem is now dynamic, and we must consider the optimization over a time horizon T. The welfare maximization problem can be stated as

$$\begin{aligned} \text{maximize} \quad & \sum_{t=0}^{T} W\Big(U_1(x_1), \dots, U_n(x_n)\Big) \\ \text{subject to} \quad & x_i[t+1] = x_i[t] + \sum_j u_{ij}[t-1] - \sum_h u_{hi}[t] \\ & u_{ij}[t] \geq 0 \\ & x_i[0] = w_i. \end{aligned} \tag{2.9}$$

In the above the sum over j are the nodes that send goods to node i, and the sum over h are the nodes to which i sends goods. For welfare functions on the form

$$W\Big(U_1(x_1),\ldots,U_n(x_n)\Big) = \sum_i U_i(x_i),$$

(2.9) describes the types of problems studied in this thesis. Thus, the results given in this thesis give the solution to a class of dynamical welfare maximization problems, and the closed form solutions that are given could be used to provide additional insights and simplified analysis of the resulting equilibrium compared to using more standard numerical methods.

3
Contributions

The main contributions of this thesis are the four papers following this chapter. In the next section, a declaration of the contributions of each author is given, as well as a summary of the results in each paper. Next follows a short conclusion an outline of possible future work.

3.1 Summary of Papers

Paper I

Heyden, M., R. Pates, and A. Rantzer (2021). "Optimal transportation on directed tree graphs". *Under review for IEEE Transactions on Automatic Control*. Initial submission September 2020, resubmitted September 2021.

Authors' contributions: M. Heyden derived the results and wrote the manuscript with help from R. Pates. A. Rantzer helped revise the manuscript.

In this paper, optimal transportation on directed tree graphs subject to transportation delays is considered. It is shown that for a class of problems the resulting optimal controllers are highly structured, allowing for efficient implementation and synthesis. The optimal flow between two nodes only depends on the quantity in two sets, called the upstream and downstream sets. For rooted trees, the downstream set consists of the destination node and all its descendants, and the upstream set consists of the source node and all its descendants except the ones in the downstream set. As an example, consider the graph in Figure 3.1, for which the structure of the optimal controller is as in (3.1). All stars on each line have the same value (corresponding to the upstream set), and all rhomboids on each line have the same value (corresponding to the downstream set). Note that the optimal flows depend both on the node levels $z_i[t]$ and the incoming flows $u_{ij}[t-1]$ in the two sets. The level in the upstream and downstream set can be

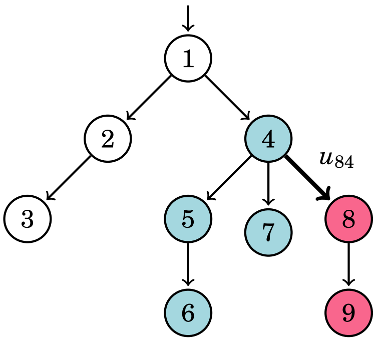


Figure 3.1 An example of a directed tree. For the highlighted edge $(8, 4)$ we have illustrated the upstream set U in light blue and downstream set D in pink. The incoming arrow into node one indicates that there is production in that node.

calculated by a sweep through the graph starting at the bottom of the graph and iterating upwards.

$$
\begin{bmatrix} u_{10}[t] \\ u_{21}[t] \\ u_{32}[t] \\ u_{41}[t] \\ u_{54}[t] \\ u_{65}[t] \\ u_{74}[t] \\ u_{84}[t] \\ u_{98}[t] \end{bmatrix}
=
\begin{bmatrix}
\ast & \ast & \ast & \ast & \ast & \ast & \ast & \ast & \ast \\
\star & \blacklozenge & \blacklozenge & \star & \star & \star & \star & \star & \star \\
0 & \star & \blacklozenge & 0 & 0 & 0 & 0 & 0 & 0 \\
\star & \star & \star & \blacklozenge & \blacklozenge & \blacklozenge & \blacklozenge & \blacklozenge & \blacklozenge \\
0 & 0 & 0 & \star & \blacklozenge & \blacklozenge & \star & \star & \star \\
0 & 0 & 0 & 0 & \star & \blacklozenge & 0 & 0 & 0 \\
0 & 0 & 0 & \star & \star & \star & \blacklozenge & \star & \star \\
0 & 0 & 0 & \star & \star & \star & \star & \blacklozenge & \blacklozenge \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \star & \blacklozenge
\end{bmatrix}
\begin{bmatrix} u_{10}[t-1] + z_1[t] \\ u_{21}[t-1] + z_2[t] \\ u_{32}[t-1] + z_3[t] \\ u_{41}[t-1] + z_4[t] \\ u_{54}[t-1] + z_5[t] \\ u_{65}[t-1] + z_6[t] \\ u_{74}[t-1] + z_7[t] \\ u_{84}[t-1] + z_8[t] \\ u_{98}[t-1] + z_9[t] \end{bmatrix}
\tag{3.1}
$$

Paper II

Heyden, M., R. Pates, and A. Rantzer (2021). "A structured optimal controller with feed-forward for transportation". *IEEE Control Systems Letters*.

Authors' contributions: M. Heyden derived the results and wrote the manuscript with help from R. Pates. A. Rantzer helped revise the manuscript.

In this paper the problem set up in Paper I is restricted to string graphs, but extended to allow for production in every node and optimal feed-forward for planned disturbances. To calculate the optimal production and internal flows, each node only needs to know two parameters $\delta_i[t]$ and $\mu_i[t]$. These two parameters can be implemented using two serial sweeps going in different directions, on the form

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

The sweep calculating the δs starts in the most downstream node (node 1), where δ_1 can be calculated using local information. Next δ_2 is calculated based on the value of δ_1. This continues up the graph until δ_N is calculated. The sweep calculating the μs starts in the most upstream node (node N) where μ_N is calculated based on local information. The sweep then continues, where μ_i is calculated based on μ_{i+1} until μ_1 is known. The necessary information about the disturbances can also be calculated by a sweep through the graph, going in the upstream direction.

The parameters for the optimal controller are also shown to be possible to calculate through a series of sweeps through the graph, only requiring local communication and scalar computations.

Paper III

Heyden, M., R. Pates, and A. Rantzer (2020). “Structured LQ-control of irrigation networks”. *Submitted to the European Control Conference (ECC) 2022.*

Authors’ contributions: M. Heyden derived the results, designed the controllers and simulations, and wrote the manuscript. R. Pates and A. Rantzer helped revise the manuscript.

In this paper, the results from Paper II are applied to third-order models for irrigation networks found in the literature. An illustration of the system can be seen in Figure 3.2. The objective is to keep the water levels close to the set-points while rejecting disturbances due to the farmers’ off-takes. The synthesis is carried out on a first-order approximation of the third-order model. By utilizing low-pass filtering of the input signal, and Kalman filtering of the output signal, the controller can be applied to the system with satisfactory performance. The performance was compared to a P controller with feed-forward and to an optimal LQ controller using the third-order dynamics and with full state knowledge. Our controller outperformed the P controller and had almost the optimal performance for disturbance rejection.

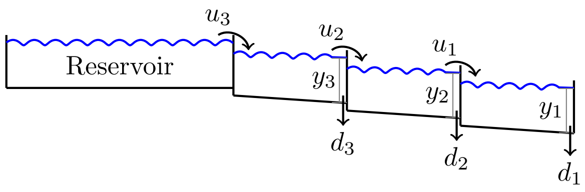


Figure 3.2 Schematic illustration of a water irrigation network. The goal is to choose the flows u_i so that the water levels y_i are close to the set-points, even when disturbances d_i are active.

Paper IV

Heyden, M., R. Pates, and A. Rantzer (2020). "Price based linear quadratic control under transportation delay". *IFAC-PapersOnLine* **53**:2, pp. 3192–3197.

Authors' contribution: M. Heyden derived the results and wrote the manuscript. R. Pates and A. Rantzer helped revise the manuscript.

In this paper, we consider both the global utility and the utility of the individual nodes in the system. It is assumed that each node values a level $z_i[t]$ according to the utility function $U_i(z_i[t])$ and that each node tries to maximize its utility. By introducing a set of prices $p_i[t]$ and making each node pay for the level they receive, each node's utility is on the form

$$\sum_{t=0}^{T} \Big(U_i(z_i[t]) - p_i[t]z_i[t] \Big).$$

Prices are derived using Lagrange multipliers so that the social optimum, that is the inventory levels that maximize the total utility in the system, is also optimal for every node. The calculation of these prices can be implemented using local communication. It is shown that this price scheme is budget neutral and beneficial for all nodes in the system.

Additional Publication

The author of the thesis has also contributed to the following publication, which is not part of the thesis as the results are covered by Paper I.

Heyden, M., R. Pates, and A. Rantzer (2018). "A structured linear quadratic controller for transportation problems". In: *2018 European Control Conference (ECC)*. IEEE, pp. 1654–1659.

3.2 Conclusions and Future Work

In this thesis, a class of structured controllers was derived for a general model for transportation. The controllers give globally optimal performance while still having a structure that allows for an efficient implementation. When production is only allowed in the top node, the implementation relies on a single sweep through the graph. When production is allowed in every node two parallel sweeps are needed. In either case, the sweeps will use only local communication and scalar computations. The controller was tested on a model for irrigation networks found in the literature where it showed promising results by outperforming a P controller with feed-forward.

The work presented in this thesis could be extended and built upon in several ways. The results for optimal feed-forward are expected to extend to tree graphs. Production in every node is also expected to be possible for tree graphs, however both the derivation, and the implementation of the optimal controller is expected to be more complex. The principle of optimizing the shifted levels as seen in the papers could be used to quickly derive extensions motivated by applications. For example, it is expected that a few saturations can be handled efficiently.

The results in Paper III are a first step towards applying the derived theory to irrigation networks. However, important future work includes evaluating and analyzing robustness to modeling errors, communication delays, and measurement noise, and ultimately, testing the controller on the physical process.

Derivation of prices as in Paper IV should be relatively straightforward for the settings in Paper I and Paper II as closed-form expressions of the optimal inputs are known, which can give the optimal levels, and the optimal Lagrange multipliers. The closed-form solutions could also be used to study how the performance of the system changes, for example, if transportation delays are reduced or increased.

It would be interesting to investigate how to extend the ideas in this thesis so that they can be applied to problems with more general dynamics and cost functions. This thesis gives examples for when a particular type of structured controller is globally optimal. However, there may be other examples or problem classes that admit a similar form of structured solution. It would also be interesting to explore the potential of this type of controller structure both heuristically and theoretically within the framework of structured optimal control.

Bibliography

Anderson, J., J. C. Doyle, S. H. Low, and N. Matni (2019). “System level synthesis”. *Annual Reviews in Control*.

Andreasson, M., D. V. Dimarogonas, H. Sandberg, and K. H. Johansson (2014). “Distributed PI-control with applications to power systems frequency control”. In: *2014 American Control Conference*. IEEE, pp. 3183–3188.

Aström, K. J. (2008). “Event based control”. In: *Analysis and design of nonlinear control systems*. Springer, pp. 127–147.

Axsäter, S. (2005). “A simple decision rule for decentralized two-echelon inventory control”. *International Journal of Production Economics* **93**, pp. 53–59.

Bamieh, B., M. R. Jovanovic, P. Mitra, and S. Patterson (2012). “Coherence in large-scale networks: dimension-dependent limitations of local feedback”. *IEEE Transactions on Automatic Control* **57**:9, pp. 2235–2249.

Bamieh, B., F. Paganini, and M. A. Dahleh (2002). “Distributed control of spatially invariant systems”. *IEEE Transactions on automatic control* **47**:7, pp. 1091–1107.

Bergeling, C., R. Pates, and A. Rantzer (2020). “H-infinity optimal control for systems with a bottleneck frequency”. *IEEE Transactions on Automatic Control* **66**:6, pp. 2732–2738.

Blondel, V. and J. N. Tsitsiklis (1997). “Np-hardness of some linear control design problems”. *SIAM journal on control and optimization* **35**:6, pp. 2118–2127.

Boukas, E. et al. (2003). “Hierarchical control of production and maintenance rates in manufacturing systems”. *Journal of quality in maintenance engineering* **9**:1, pp. 66–82.

Cantoni, M., E. Weyer, Y. Li, S. K. Ooi, I. Mareels, and M. Ryan (2007). “Control of large-scale irrigation networks”. *Proceedings of the IEEE* **95**:1, pp. 75–91.

De Persis, C., T. N. Jensen, R. Ortega, and R. Wisniewski (2014). “Output regulation of large-scale hydraulic networks”. *IEEE Transactions on Control Systems Technology* **22**:1, pp. 238–245.

Evans, R., L. Li, I. Mareels, N. Okello, M. Pham, W. Qiu, and S. K. Saleem (2011). “Real-time optimal control of river basin networks”. *IFAC Proceedings Volumes* **44**:1, pp. 11459–11464.

Fardad, M., F. Lin, and M. R. Jovanović (2011). “Sparsity-promoting optimal control for a class of distributed systems”. In: *Proceedings of the 2011 American Control Conference*. IEEE, pp. 2050–2055.

Farokhi, F., C. Langbort, and K. H. Johansson (2013). “Optimal structured static state-feedback control design with limited model information for fully-actuated systems”. *Automatica* **49**:2, pp. 326–337.

Grammatico, S., B. Gentile, F. Parise, and J. Lygeros (2015). “A mean field control approach for demand side management of large populations of thermostatically controlled loads”. In: *2015 European Control Conference (ECC)*. IEEE, pp. 3548–3553.

Ho, Y.-C. and K.-C. Chu (1972). “Team decision theory and information structures in optimal control problems–part I”. *IEEE Transactions on Automatic control* **17**:1, pp. 15–22.

Hoberg, K., J. R. Bradley, and U. W. Thonemann (2007). “Analyzing the effect of the inventory policy on order and inventory variability with linear control theory”. *European Journal of Operational Research* **176**:3, pp. 1620–1642.

Jones, A. T. and C. R. McLean (1986). “A proposed hierarchical control model for automated manufacturing systems”. *Journal of manufacturing systems* **5**:1, pp. 15–25.

Kao, C.-Y., U. Jönsson, and H. Fujioka (2009). “Characterization of robust stability of a class of interconnected systems”. *Automatica* **45**:1, pp. 217–224.

Kearney, M., M. Cantoni, and P. M. Dower (2011). “Non-iterative distributed MPC for large-scale irrigation channels”. In: *2011 Australian Control Conference*. IEEE, pp. 217–223.

Kelly, F. P. (2000). “Models for a self–managed internet”. *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* **358**:1773, pp. 2335–2348.

Kundur, P. (1994). *Power System Stability and Control*. McGraw-Hill Professional. ISBN: 007035958X.

Langbort, C., R. S. Chandra, and R. D’Andrea (2004). “Distributed control design for systems interconnected over an arbitrary graph”. *IEEE Transactions on Automatic Control* **49**:9, pp. 1502–1519.

Langbort, C. and J.-C. Delvenne (2010). “Distributed design methods for linear quadratic control and their limitations”. *IEEE Transactions on Automatic Control* **55**:9, pp. 2085–2093.

Lee, H. L., V. Padmanabhan, and S. Whang (1997). “The bullwhip effect in supply chains”. *Sloan management review* **38**, pp. 93–102.

Lemos, J. M. and L. F. Pinto (2012). “Distributed linear-quadratic control of serially chained systems: application to a water delivery canal”. *IEEE Control Systems Magazine* **32**:6, pp. 26–38.

Lessard, L. and S. Lall (2011). “Quadratic invariance is necessary and sufficient for convexity”. In: *Proceedings of the 2011 American Control Conference*. IEEE, pp. 5360–5362.

Lessard, L. and S. Lall (2012). “Optimal controller synthesis for the decentralized two-player problem with output feedback”. In: *2012 American Control Conference (ACC)*. IEEE, pp. 6314–6321.

Lestas, I. and G. Vinnicombe (2006). “Scalable decentralized robust stability certificates for networks of interconnected heterogeneous dynamical systems”. *IEEE transactions on automatic control* **51**:10, pp. 1613–1625.

Li, Y. (2014). “Offtake feedforward compensation for irrigation channels with distributed control”. *IEEE Transactions on Control Systems Technology* **22**:5, pp. 1991–1998.

Lin, F., M. Fardad, and M. R. Jovanović (2013). “Design of optimal sparse feedback gains via the alternating direction method of multipliers”. *IEEE Transactions on Automatic Control* **58**:9, pp. 2426–2431. DOI: 10.1109/TAC.2013.2257618.

Lin, P.-H., D. S.-H. Wong, S.-S. Jang, S.-S. Shieh, and J.-Z. Chu (2004). “Controller design and reduction of bullwhip for a model supply chain system using z-transform analysis”. *Journal of process control* **14**:5, pp. 487–499.

Litrico, X. and V. Fromion (2002). “Infinite dimensional modelling of open-channel hydraulic systems for control purposes”. In: *Proceedings of the 41st IEEE Conference on Decision and Control, 2002.* Vol. 2. IEEE, pp. 1681–1686.

Litrico, X. and V. Fromion (2005). “Design of structured multivariable controllers for irrigation canals”. In: *Proceedings of the 44th IEEE Conference on Decision and Control*. IEEE, pp. 1881–1886.

Litrico, X., V. Fromion, J.-P. Baume, and M. Rijo (2003). “Modelling and PI control of an irrigation canal”. In: *2003 European Control Conference (ECC)*. IEEE, pp. 850–855.

Madjidian, D. and L. Mirkin (2014). “Distributed control with low-rank coordination”. *IEEE Transactions on Control of Network Systems* **1**:1, pp. 53–63.

Mansour, M. O. and A. H. El Abiad (1977). “Hierarchical control of power systems”. *IFAC Proceedings Volumes* **10**:1, pp. 130–134.

Mareels, I., E. Weyer, S. K. Ooi, M. Cantoni, Y. Li, and G. Nair (2005). “Systems engineering for irrigation systems: successes and challenges”. *IFAC Proceedings Volumes* **38**:1, pp. 1–16.

Marschak, J. (1955). “Elements for a theory of teams”. *Management science* **1**:2, pp. 127–137.

Matni, N. (2014). “Distributed control subject to delays satisfying an $\mathcal{H}_\infty$ norm bound”. In: *53rd IEEE Conference on Decision and Control*. IEEE, pp. 4006–4013.

Nasir, H. A., M. Cantoni, and E. Weyer (2017). “An efficient implementation of stochastic MPC for open channel water-level planning”. In: *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE, pp. 511–516.

Negenborn, R. R., P.-J. van Overloop, T. Keviczky, and B. De Schutter (2009a). “Distributed model predictive control of irrigation canals”. *Networks & Heterogeneous Media* **4**:2, p. 359.

Negenborn, R. R., A. Sahiir, Z. Lukszcr, B. De Schutter, and M. Morari (2009b). “A non-iterative cascaded predictive control approach for control of irrigation canals”. In: *2009 IEEE International Conference on Systems, Man and Cybernetics*. IEEE, pp. 3552–3557.

Ortega, R., A. Van Der Schaft, F. Castanos, and A. Astolfi (2008). “Control by interconnection and standard passivity-based control of port-hamiltonian systems”. *IEEE Transactions on Automatic control* **53**:11, pp. 2527–2542.

Parise, F., M. Colombino, S. Grammatico, and J. Lygeros (2014). “Mean field constrained charging policy for large populations of plug-in electric vehicles”. In: *53rd IEEE Conference on Decision and Control*. IEEE, pp. 5101–5106.

Pates, R., C. Lidström, and A. Rantzer (2017). “Control using local distance measurements cannot prevent incoherence in platoons”. In: *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE, pp. 3461–3466.

Radner, R. (1962). “Team decision problems”. *The Annals of Mathematical Statistics* **33**:3, pp. 857–881.

Rotkowitz, M., R. Cogill, and S. Lall (2010). “Convexity of optimal control over networks with delays and arbitrary topology”. *International Journal of Systems, Control and Communications* **2**:1-3, pp. 30–54.

Rotkowitz, M. and S. Lall (2006). “A characterization of convex problems in decentralized control”. *IEEE Transactions on Automatic Control* **51**:2, pp. 274–286.

Sandell, N. and M. Athans (1974). “Solution of some nonclassical LQG stochastic decision problems”. *IEEE Transactions on Automatic Control* **19**:2, pp. 108–116.

Schuurmans, J., A. Hof, S. Dijkstra, O. Bosgra, and R. Brouwer (1999). “Simple water level controller for irrigation and drainage canals”. *Journal of irrigation and drainage engineering* **125**:4, pp. 189–195.

Schweppe, F. C. (1978). “Power systems ‘2000’: hierarchical control strategies”. *IEEE spectrum* **15**:7, pp. 42–47.

Shah, P. and P. A. Parrilo (2013). “H_2-optimal decentralized control over posets: a state-space solution for state-feedback”. *IEEE Transactions on Automatic Control* **58**:12, pp. 3084–3096.

Subramanian, K., J. B. Rawlings, C. T. Maravelias, J. Flores-Cerrillo, and L. Megan (2013). “Integration of control theory and scheduling methods for supply chain management”. *Computers & Chemical Engineering* **51**, pp. 4–20.

Varian, H. R. (2003). *Intermediate Microeconomics:A Modern Approach*. 6th ed. W. W. Norton & Company. ISBN: 0-393-97830-3.

Wang, Y.-S., N. Matni, and J. C. Doyle (2018). “Separable and localized system-level synthesis for large-scale systems”. *IEEE Transactions on Automatic Control* **63**:12, pp. 4234–4249.

Weyer, E. (2001). “System identification of an open water channel”. *Control engineering practice* **9**:12, pp. 1289–1299.

Weyer, E. (2003). “Controller configurations for irrigation channels derived from sequential minimum variance control”. In: *Proceedings of 2003 IEEE Conference on Control Applications, 2003. CCA 2003.* Vol. 2. IEEE, pp. 1209–1214.

Weyer, E. (2008). “Control of irrigation channels”. *IEEE Transactions on Control Systems Technology* **16**:4, pp. 664–675.

Witsenhausen, H. S. (1968). “A counterexample in stochastic optimum control”. *SIAM Journal on Control* **6**:1, pp. 131–147.

Zheng, Y.-S. and A. Federgruen (1991). “Finding optimal (s, S) policies is about as simple as evaluating a single policy”. *Operations research* **39**:4, pp. 654–665.

Paper I

Optimal Transportation on Directed Tree Graphs

Martin Heyden   Richard Pates   Anders Rantzer

Abstract

We consider the problem of optimal transportation and production of a quantity throughout a network, where the transportation is subject to delay. It is shown that for a simple network model the optimal policy is sparse and highly structured. The results hold for a broad class of convex cost functions, and in the quadratic case, we give closed-form expressions for the optimal flows and the optimal production. The optimal controller can be both synthesized and implemented using distributed communication, making it suitable for large-scale applications. The performance of the optimal controller is studied for networks of different sizes and topologies, and compared to a local controller designed using off-the-shelf methods. The optimal controller gives a significant increase in performance for non-zero initial conditions.

Submitted to IEEE Transactions on Automatic Control.

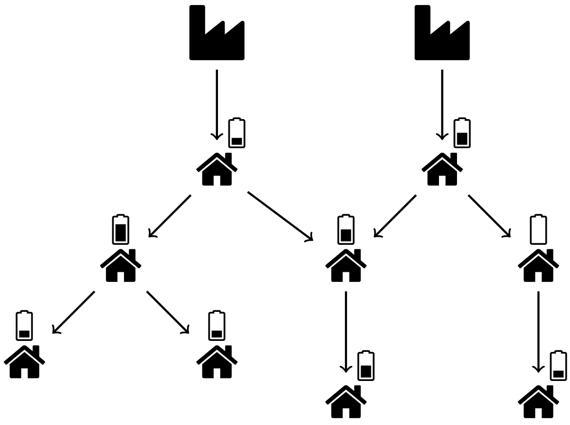


Figure 1. An illustration of the type of problems studied in this paper. At the top of the network is a production plant that produces a commodity. The commodity should be optimally distributed to all the users in the system.

1. Introduction

Whether it be to maximize throughput in a traffic network, minimize losses in an electrical power system, or improve fairness when managing Internet congestion, large-scale systems are typically operated with some notion of performance in mind. Many such networks are also constructed out of dynamical components, and a fundamental challenge is to operate them in a manner that maximizes performance. In many cases, including those listed above, this balance is struck by designing control schemes to stabilize the system about *an equilibrium point* that is chosen to optimize the network's given measure of performance (e.g. [Kundur, 1994; Kelly et al., 1998; Ukkusuri and Özbay, 2013]).

However, in most cases, it could be argued that these large-scale systems are in fact never in equilibrium for long, but instead shifting from operating point to operating point to balance the current demands of the users. There is of course still a value in shifting towards an equilibrium that optimizes performance, but it is also clear that the notion of *dynamic performance* is an important one. Take for example electrical power systems; the objective is really to minimize losses throughout operation, rather than the losses associated with particular operating points.

A major bottleneck in the application of control methods to improve dynamic performance in applications is a lack of scalability. This has prompted a great deal of research within the control community on optimal control under constraints on the controller structure (e.g. sparsity). Early work in this area includes that on team decision problems, where different cooperating decision-makers have access to different information (see for example [Radner, 1962]). Another common approach is to enforce the desired structure on the controller when conducting synthesis. An important contribution to this approach was given in [Rotkowitz and Lall, 2005], where the notation of Quadratic Invariance was used to give a condition under which the structured controller synthesis problem can be recast as a convex optimization problem. Indeed the role of convexity has been a recurring theme, consider for example the work on system level synthesis [Anderson et al., 2019; Wang et al., 2018] and network realizability [Rantzer, 2019].

Related approaches are based on large-scale optimization methods. For example, in [Mårtensson and Rantzer, 2009] a method for distributed synthesis of a distributed LQ controller is derived by constructing local estimates for the gradients of the global cost function. A slightly different approach is taken in [Lin et al., 2013], where structured controllers are synthesized by posing an optimization problem with a sparsity-promoting term.

In this work, we study an optimal control problem for a simple dynamic model for transportation that includes the effect of transportation delays. The objective is to optimally produce and distribute some quantity throughout a network, as illustrated in Figure 1. The notion of performance that we consider is derived from a natural *dynamic* extension of the classical welfare maximization problem in economics. Throughout we consider the case of a single commodity, however, the results presented could be extended to the case of multiple commodities.

While we will give economic motivations for the problem studied, the main contribution lies in showing that the optimal controller is highly structured. These structural features allow for an efficient and scalable implementation of the control, even for large networks. This is in contrast with standard methods for controller synthesis, such as Riccati based approaches, which in general scales poorly with size, both in terms of synthesis and implementation. Critically this structure is also achieved without imposing any a priori constraints on the controller structure. Instead, the controller inherits its structure directly from the plant, *without the need to enforce constraints*. This means that no performance is lost as a result of implementing these distributed control actions, in contrast to the methods discussed above.

Our results are thus similar in nature to others that show an unconstrained optimal controller is structured in such a way that it is suitable for large-scale implementation. Notable results along these lines include [Bamieh et al., 2002], where it is shown that for spatially invariant systems, the optimal controller will be spatially invariant and localized. Furthermore in [D'Andrea and Dullerud, 2003] it is shown that for such systems the optimal distributed controller can be found by solving linear matrix inequalities. In [Shah and Parrilo, 2013] an optimal controller for systems structured through partially ordered sets is found by solving a set of independent Riccati equations. Other examples for which a distributed controller is globally optimal include [Bergeling et al., 2020], where it is shown that for systems with symmetric and Hurwitz state matrix, an optimal H_∞ controller can be found using a simple calculation involving the matrices of the system's state-space representation. Furthermore, if the plant has a sparsity pattern, the optimal controller will be distributed. Another interesting structured controller is presented in [Madjidian and Mirkin, 2014], where the controller consists of a decentralized part and a rank-one coordination term.

1.1 Problem Formulation

In welfare maximization, the goal is to optimally distribute a number of commodities among a set of agents so that the total social welfare is maximized. There are many ways of defining the social welfare (see [Varian, 2003] for an introduction). In this work, we will consider the classical utilitarian welfare functions, where the total welfare is the sum of the utility for all agents. Furthermore, the individual welfare functions are individualistic, meaning that the utility of the agents only depends on the amount of the quantity each agents has at each point in time. To capture dynamic aspects, we introduce constraints that model the effect of transportation delays between the agents', as structured by a graph. The model we use in fact coincides exactly with the types of model that are used to study irrigation networks and supply chains, as discussed further below.

Assume that each agent i starts with an initial amount w_i of the commodity, and the utility for node i in having an amount z_i of the commodity is given by $U_i(z_i)$. Then the welfare maximization problem can be formulated as

$$\begin{aligned} \underset{z_i}{\text{maximize}} \quad & \sum_i U_i(z_i) \\ \text{subject to} \quad & \sum_i z_i = \sum_i w_i. \end{aligned} \tag{1}$$

As we can see, the objective is to maximize the collective utility, subject to the constraint that the new levels respect the initial endowments.

We will now present a natural dynamic extension of this standard welfare maximization problem, by only allowing for the commodity to be transported with a delay, between agents that are connected. This extension, for instance, captures the dynamic effects of shifting from one equilibrium configuration to another. This would be relevant when an agent changes their utility, leading to a new set of equilibrium flows that optimize (1). Our formulation allows the welfare changes associated with this transition to be quantified, and our main contribution is to design an optimal controller, which are inherently distributed, to facilitate this transition.

We start by introducing these dynamic constraints. To describe the topology of the network, we introduce a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consisting of a set of vertices $\mathcal{V} = \{1, \dots, N\}$ and directed edges $\mathcal{E}$, where $(i, j) \in \mathcal{E}$ if there exists a path from node j to node i. We also denote the parent set and children set of the jth node as $\mathcal{P}(j)$ and $\mathcal{C}(j)$ respectively. That is $k \in \mathcal{P}(j)$ if $(j, k) \in \mathcal{E}$ and $i \in \mathcal{C}(j)$ if $(i, j) \in \mathcal{E}$.

Let $z_i[t] \in \mathbb{R}$ be the amount of the quantity held by agent i at time t. We will consider the following dynamical system, which captures the essence of transportation subject to delay throughout a network. At its heart, this model is a conservation equation, and the idea is that this simple system can describe the basic features of transportation in a wide range of settings.

$$z_i[t+1] = \alpha\left(z_i[t] + \sum_{j \in \mathcal{P}(i)} u_{ij}[t-1]\right) - \sum_{h \in \mathcal{C}(i)} u_{hi}[t] \tag{2}$$

This equation describes how $z_i[t]$ evolves over time. Each input $u_{ij}[t] \in \mathbb{R}$ denotes the amount of the quantity being transported from node j to node i (associated with the edge $(i, j) \in \mathcal{E}$). All variables are relative to an equilibrium that accounts for the steady state flows in the network and the consumption of the individual agents. The constant $\alpha,\ 0 < \alpha \leq 1$, is the decay rate of the stored quantity. The first summation gives the amount of the quantity arriving at the ith node, and the second summation gives the amount leaving the ith node. The one-step delay in the first summation captures the delay in transportation. In Section 2.5 it will be shown how the results presented in this paper can also be applied to networks with non-homogeneous delays.

We will also allow for production at some nodes in the network. We denote the set of all such nodes as I and denote the production as $u_{i0},\ i \in I$. The zero node is not part of the network, that is $0 \notin \mathcal{V}$. However, in a slight abuse of notation, we let $0 \in \mathcal{P}(i)$ if $i \in I$. This allows external production to be described in a manner consistent with the dynamics in (2). We will assume that only nodes that are at the top of the network can be producers. This will be made precise in Section 2. Note that (2) is not a state-space model, however a realization with states $\{z_i[t], u_{ij}[t-1],\ j \in \mathcal{P}(i)\},\ \forall i$, can be introduced.

Models of the type in (2) have been used for control of water irrigation networks [Evans et al., 2011; Schuurmans et al., 1999; Litrico and Fromion, 2005]. On fast time scales, water levels in an irrigation network do not follow such simple dynamics. More accurate descriptions are based on the Saint Venant equations [Coron et al., 1999], or third-order system identification models [Weyer, 2001]. However, the simple model can still be used for controller design, assuming that there is an inner controller that ensures that the wave dynamics are not excited [Cantoni et al., 2007]. This inner controller could for example be a low pass filter. We plan to apply the controller presented in this work to a model for water irrigation networks in a future paper.

The proposed model in (2) also coincides with the dynamics used to model supply chains [Subramanian et al., 2013; Lin et al., 2004]. Additional dynamics, such as backorders, are often introduced for use in the cost function. This can however be taken into account in the calculations of the optimal equilibrium flows, or in local controllers in every node.

The dynamic extension of the welfare maximization problem (1) that we will study in this paper is:

$$\begin{aligned}
\underset{u,z}{\text{minimize}} \quad & \sum_{t=0}^{\infty}\left(\sum_{i\in\mathcal{V}} f_i(z_i[t]) + \sum_{i\in I} g_i(u_{i0}[t])\right) \\
\text{subject to} \quad & \text{Dynamics in (2)}, \\
& z_i[0] = z_i^0 \in \mathbb{R},\ u_{ij}[-1] = u_{ij}^0 \in \mathbb{R}.
\end{aligned} \tag{3}$$

In the above $f_i(z_i[t])$ is the cost (or negative utility) for node i to have access to $z_i[t]$ goods. The function $g_i(u_{i0})$ is the cost for node i to produce u_{i0}. The problem corresponds to minimizing the total cost[1] for the entire system, which is a welfare maximization problem for one commodity. Note that the major difference between the problems in (1) and (3) is the static constraint in (1) has been replaced by its dynamic analogue from (2). As the variables are relative to an equilibrium, the problem in (3) accounts for the change in costs due to the deviations from the nominal operation of the transportation network.

1.2 Preview of Results

The key feature of our results is that they show how the structure in the underlying graph can be exploited when solving the dynamic welfare problem described in the previous subsection. In particular, we will show that the optimal control on each edge (i, j) will be dependent on two sets $U(i, j)$ and $D(i, j)$, which we will call the upstream set and the downstream set. These sets are subsets of the nodes of the graph. They are different for each edge and reflect the local structural properties of the graph.

We will show that the problem in (3) admits a structured solution for general strictly convex cost functions. For the case of quadratic cost functions, a closed-form solution can be found. Our main contribution is to show

[1] Note that we choose to minimize the total cost instead of maximizing the total utility.

that the solution to the optimal control problem

$$\begin{aligned} \underset{u,z}{\text{minimize}} \quad & \sum_{t=0}^{\infty}\left(\sum_{i\in\mathcal{V}} q_i z_i[t]^2 + \sum_{i\in I} r_i u_{i0}[t]^2\right) \\ \text{subject to} \quad & \text{Dynamics in (2)} \\ & z_i[0] = z_i^0 \in \mathbb{R},\; u_{ij}[-1] = u_{ij}^0 \in \mathbb{R}, \end{aligned} \tag{4}$$

where $q_i > 0$ and $r_i > 0$, admits a highly structured solution (contrary to the standard linear quadratic regulator problem). We also explain how these results can be generalized to the case of general convex functions. The optimal controller for quadratic cost functions can be calculated according to the formula

$$u_{ij}[t] = \frac{\gamma_{U(i,j)}}{\gamma_{U(i,j)} + \gamma_{D(i,j)}} \alpha m_{U(i,j)}[t] - \frac{\gamma_{D(i,j)}}{\gamma_{U(i,j)} + \gamma_{D(i,j)}} \alpha m_{D(i,j)}[t]. \tag{5}$$

The constants $\gamma_{U(i,j)}$ and $\gamma_{D(i,j)}$ only depend on the problem data within their respective set and can be computed ahead of time. The variable $m_{U(i,j)}$ is the total quantity stored in the nodes in the upstream set, and $m_{D(i,j)}$ is the total quantity stored in the nodes in the downstream set. The definition of the upstream and downstream sets will be made precise in the next section. However to get the intuition, consider Figure 2. Taking u_{84} as an example, then the upstream set is indicated by blue and the downstream set by pink. Observe in particular that some nodes are neither in the upstream nor the downstream set. This implies the optimal control law is sparse (with sparsity defined by the elements in the up and downstream sets). The structure of the resulting controller for the graph in Figure 2 is illustrated in Figure 3. Furthermore, both the γ parameters and the aggregate levels m can be calculated by a sweep through the graph, using only local communication. This allows for efficient synthesis and implementation of the control law in (4). Similarly, the optimal production for (4) depends only on the total level in the graph at time t.

2. The Optimal Controller

In this section we will demonstrate that the solution to the optimal control problem in (4) does indeed have the structure hinted at in (5). This will be presented as Theorem 1 and Theorem 2. We will also show how the internal flows for the more general problem (3) can be found by solving a static convex optimization problem in Theorem 3.

However, we start by introducing the required graph-theoretic notions to define the upstream $U(i,j)$ and downstream $D(i,j)$ sets of a given edge (i,j)

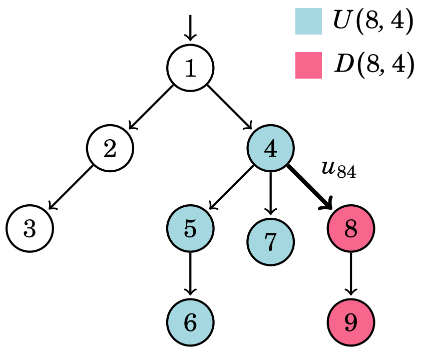


Figure 2. An example of a directed tree. Node Four has one parent node in node one, and three children nodes in nodes five, seven, and eight. For the highlighted edge $(8,4)$ we have illustrated the upstream set U in light blue and downstream set D in pink. The incoming arrow into node one indicates that there is production in that node.

$$\begin{bmatrix} u_{10}[t] \\ u_{21}[t] \\ u_{32}[t] \\ u_{41}[t] \\ u_{54}[t] \\ u_{65}[t] \\ u_{74}[t] \\ u_{84}[t] \\ u_{98}[t] \end{bmatrix} = \begin{bmatrix} * & * & * & * & * & * & * & * & * \\ \star & \blacklozenge & \blacklozenge & \star & \star & \star & \star & \star & \star \\ 0 & \star & \blacklozenge & 0 & 0 & 0 & 0 & 0 & 0 \\ \star & \star & \star & \blacklozenge & \blacklozenge & \blacklozenge & \blacklozenge & \blacklozenge & \blacklozenge \\ 0 & 0 & 0 & \star & \blacklozenge & \blacklozenge & \star & \star & \star \\ 0 & 0 & 0 & 0 & \star & \blacklozenge & 0 & 0 & 0 \\ 0 & 0 & 0 & \star & \star & \star & \blacklozenge & \star & \star \\ 0 & 0 & 0 & \star & \star & \star & \star & \blacklozenge & \blacklozenge \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \star & \blacklozenge \end{bmatrix} \begin{bmatrix} u_{10}[t-1] + z_1[t] \\ u_{21}[t-1] + z_2[t] \\ u_{32}[t-1] + z_3[t] \\ u_{41}[t-1] + z_4[t] \\ u_{54}[t-1] + z_5[t] \\ u_{65}[t-1] + z_6[t] \\ u_{74}[t-1] + z_7[t] \\ u_{84}[t-1] + z_8[t] \\ u_{98}[t-1] + z_9[t] \end{bmatrix}$$

Figure 3. Structure of optimal feedback law for the directed tree in Figure 2. $\star$ corresponds to the upstream set and $\blacklozenge$ to the downstream set. For the production u_{10} the $*$ indicates that information from all nodes in the graph is used. In every row, each symbol corresponds to the same value. This does not hold between different rows.

in a graph. We will begin with the definition of a rooted tree. For that case, the definitions simplify greatly, which will hopefully make the presentation easier to follow. We also believe that rooted trees, corresponding to a single producer, will occur naturally in applications. After stating the theorems in Section 2.2, we will discuss how the optimal controller can be implemented using local communication in Section 2.3. We will then give the necessary definitions to apply the theorems to arbitrary directed trees in Section 2.4. Finally, in Section 2.5 we will discuss the limitations due to the assumptions and how they can be relaxed.

2.1 Graph Structuring

In this section we give the necessary definitions required to describe the solution for rooted trees. We begin with the definition of a directed tree and a directed rooted tree.

Definition 1
A directed graph is a directed tree if the undirected version of the graph contains no cycles. □

Definition 2
A directed graph is a directed rooted tree if it contains a single vertex such that there exists a unique directed path from this vertex to every other vertex in the graph. Equivalently a directed tree is rooted if it contains a single node with no parents. □

For the rest of the paper it is implicit that all graphs are directed. The upstream and downstream sets have a simple definition for a rooted tree.

Definition 3
For a rooted tree, the upstream and the downstream sets for an edge (i, j) are defined as follows:

1. The upstream set $U(i, j)$ is the source node j and all its descendants when the edge (i, j) is removed.
2. The downstream set $D(i, j)$ is the destination node i and all its descendants. □

The definition is illustrated in Figure 2. When it is clear from context we sometimes drop the arguments (i, j) from the upstream and downstream sets. The corresponding definition for non-rooted trees is given in Definition 9, when the results for this case are presented.

The following definition of the depth of each node is needed in order to calculate the controller gains.

Definition 4
For a rooted tree we define the depth $d(i)$ of a node i as follows

- The root has depth 0.
- For any other node i with parent j it holds that $d(i) = d(j) + 1$. □

The definition of the depth for nodes in non-rooted trees will be given in Definition 5.

2.2 Theorem Statements

With the necessary definitions in place for the rooted trees, we are almost ready to present our results. However, we must first formalize the assumptions used in the theorem statements. The consequences of these assumptions will be discussed further in Section 2.5.

To ensure existence and uniqueness of the optimizer for (3), we make the following assumption.

Assumption 1
The functions f_i and g_i are strictly convex and satisfy $f_i(z_i) \geq 0$, $g_i(u_{i0}) \geq 0$, $f_i(0) = 0$, and $g_i(0) = 0$. Furthermore it holds that there exists an n, c_i, and d_i so that $g_i(u_{i0}) < c_i u_{i0}^n$ for all nodes $i \in \mathcal{I}$ and $f_i(z_i) < d_i z_i^n$ for all nodes i with no children. □

Note that the cost function in (4) satisfies Assumption 1.

We also make the assumption that only the root can be a producer, which simplifies the solution greatly.

Assumption 2
All nodes in the producer set I have depth zero, i.e $i \in I \Rightarrow d(i) = 0$. □

The assumption generalizes to non-rooted trees by using the definition of the depth for a non-rooted trees given in Definition 5.

We will now state the main results of this paper. First, the case of quadratic cost functions in problem (4) is considered. As already alluded to in the introduction, the optimal controller is highly structured in that it only needs the aggregate levels in the upstream and downstream sets to be implemented. This is demonstrated through Theorems 1 and 2. We then show in Theorem 3 how the optimal internal flows can be found for general convex cost functions. Here the controller retains the structure, in that it needs information from the same nodes as in the quadratic case. However, finding the optimal flow on a link will require solving a static convex optimization problem. The proofs will be given in Section 3.

To simplify the description of the aggregate levels we make the following definition for a set of nodes S. The variable $m_S[t]$ describes the total amount of the quantity in the set S, including the quantity currently in transit towards a node in the set.

$$m_S[t] = \sum_{i\in S}\left(z_i[t] + \sum_{j\in\mathcal{P}(i)} u_{ij}[t-1]\right) \tag{6}$$

Now we can give a formal statement of the optimal internal flows.

Theorem 1
Consider the problem in (4) under Assumption 2, and suppose that the underlying graph is a rooted tree as defined in Definition 2. For every edge $(i, j) \in \mathcal{E}$ with upstream set $U(i, j)$ and downstream set $D(i, j)$ as defined in Definition 3, let

$$\begin{aligned}\gamma_{U(i,j)} &= \left(\sum_{k\in U}\frac{1}{\alpha^{2(d(k)-d(j))}q_k}\right)^{-1}\\ \gamma_{D(i,j)} &= \left(\sum_{k\in D}\frac{1}{\alpha^{2(d(k)-d(j))}q_k}\right)^{-1},\end{aligned}$$

where $d(k)$ is the depth of node k as defined in Definition 4. Then the optimal value of $u_{ij}[t]$ is given by

$$u_{ij}[t] = \frac{\gamma_{U(i,j)}}{\gamma_{U(i,j)}+\gamma_{D(i,j)}}\alpha m_{U(i,j)}[t] - \frac{\gamma_{D(i,j)}}{\gamma_{U(i,j)}+\gamma_{D(i,j)}}\alpha m_{D(i,j)}[t]. \tag{7}$$

where $m_{U(i,j)}[t]$ and $m_{D(i,j)}[t]$ are the aggregate quantities of the upstream and downstream sets as defined in (6). □

Note that the optimal flow on link (i, j) only depends on the aggregate levels in the upstream and downstream sets. This gives a sparse and highly structured controller. Similarly, the optimal gains also only depend on the local cost functions in the upstream and downstream set, allowing for efficient synthesis. In the next subsection it will be shown how the aggregate levels and the γ parameters can be calculated recursively through the graph, allowing for an efficient implementation.

In Figure 3 the structure of the feedback law is illustrated for the graph in Figure 2. Also note that (7) can be implemented as a state feedback law, by letting $\{z_i[t], u_{ij}[t-1]\}$ be the state of the system.

Remark 1
The result is of course compatible with the results presented for a string graph in [Heyden et al., 2018]. Then the upstream set is just the source node, and the downstream set is all the descendants of the source node, as follows from Definition 3. □

The calculation of the optimal production is also structured and only requires the aggregate level for the entire graph.

THEOREM 2
Consider the problem in (4) under Assumption 2, and suppose that the underlying graph is a rooted tree as defined in Definition 2. Let

$$R = \left(\sum_{i \in I} \frac{1}{r_i} \right)^{-1},$$

and

$$\gamma_{\mathcal{V}} = \left(\sum_{i \in \mathcal{V}} \frac{1}{\alpha^{2d(i)} q_i} \right)^{-1},$$

where $d(i)$ is the depth of node i as defined in Definition 4. Furthermore, let

$$X = -\frac{1}{2}\left[(1-\alpha^2)R - \alpha^2\gamma_{\mathcal{V}}\right] + \sqrt{\alpha^2\gamma_{\mathcal{V}}R + \frac{1}{4}\left[(1-\alpha^2)R - \alpha^2\gamma_{\mathcal{V}}\right]^2}.$$

Then the optimal total production $U[t] = \sum_{i \in I} u_{i0}[t]$ is given by

$$U[t] = -\frac{X}{X+R}\alpha m_{\mathcal{V}}[t],$$

where $m_{\mathcal{V}}$ is defined as in (6). The optimal production for the individual producers $i \in I$ is given by

$$u_{i0}[t] = \frac{R}{r_i} U[t].$$

□

The constant X is the solution to a scalar Riccati equation defined by the aggregate costs $\gamma_{\mathcal{V}}$ and R.

REMARK 2
Note that for rooted trees only the root can be a producer, and thus $U[t] = u_{i0}$ where i is the root. We state the theorem in this general way so that it can easily be extended to hold for non-rooted trees. □

We will now state the results for general convex cost functions. The main difference is that more computations are required to find the optimal flows.

THEOREM 3
Consider the problem in (3) under Assumptions 1 and 2, and suppose that the underlying graph is a rooted tree as defined in Definition 2. Then for every edge $(i, j) \in \mathcal{E}$ with upstream set $U(i, j)$ and downstream set $D(i, j)$

as defined in Definition 3, the optimal value of $u_{ij}[t]$ is given by the optimal u for the optimization problem

$$\begin{aligned}
\underset{u,x_k}{\text{minimize}} \quad & \sum_{k\in U(i,j)} f_k(x_k) + \sum_{k\in D(i,j)} f_k(x_k) \\
\text{subject to} \quad & \sum_{k\in U(i,j)} \alpha^{d(j)-d(k)} x_k = \alpha m_{U(i,j)}[t] - u \\
& \sum_{k\in D(i,j)} \alpha^{d(j)-d(k)} x_k = \alpha m_{D(i,j)}[t] + u.
\end{aligned}$$

In the above $m_{U(i,j)}[t]$ and $m_{D(i,j)}[t]$ are defined by (6) and $d(k)$ is the depth of node k as defined in Definition 4. □

Again the optimal flow only requires the aggregate levels in the upstream and downstream sets. However, to find the flow on a link, a convex optimization problem with the aggregate levels of the upstream set and downstream set as input must be solved. The optimization problems gain more decision variables as the node depth decreases. However, these type of problems are well studied and there exists plenty of software that can solve such problem even for a quite large number of decision variables, for example CVX [Grant and Boyd, 2014].

Once the definitions for the upstream and downstream set has been extended to non-rooted trees in Definitions 5 and 9, Theorem 1 and Theorem 3 can be extended to also hold for non-rooted trees. We also give the general case of Theorem 2 in Proposition 2 in Section 3 as a mechanism for proving Theorem 2.

2.3 Controller Implementation

Theorem 1 and Theorem 3 limit the information each node needs to the upstream set and the downstream set of each of its outgoing links. We will now show that by exploiting that the aggregate levels can be calculated by a sweep through the graph, the controller can be implemented using only local communication. Only rooted trees are covered in this subsection. Non-rooted trees can be handled similarly, as discussed in the next subsection.

For all nodes without children, we define $m_i[t] = z_i[t] + u_{ij}[t-1]$, where j is the unique parent for node i. And for all nodes with children, we define the node aggregate level $m_i[t]$ to be

$$m_i[t] = z_i[t] + u_{ij}[t-1] + \sum_{k\in \mathcal{C}(i)} m_k[t].$$

Where j again is the unique parent of node i, except for the root, where $u_{ij} = u_{i0}$. The calculation of $m_i[t]$ only requires information from the neighboring

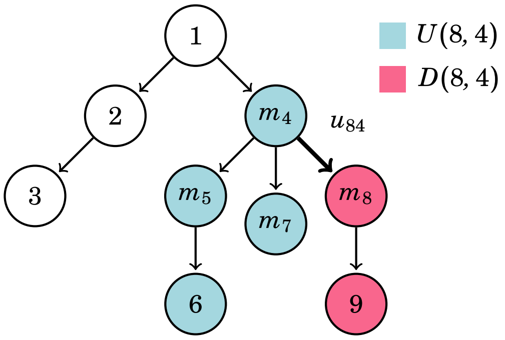


Figure 4. The example in Figure 2 revisited. The upstream and downstream sets for $(8,4)$ are highlighted in blue and pink. However, by utilizing the aggregation of inventory level in (8) node 4 can find all the information it needs from its neighbors. Node four's aggregate level is given by $m_4 = m_5 + m_7 + m_8$. Then $m_{U(8,4)} = m_4 - m_8$ and $m_{D(8,4)} = m_8$. This results in no direct communication with nodes six and nine for node four.

nodes, and the computation of $m_i[t]$ can thus be implemented using only local communication. The definition for $m_i[t]$ is different from (6) in that the index is a node and note a set.

For link (i,j) the upstream and downstream levels can by computed from the node aggregate levels of the source j and the destination i as

$$m_{U(i,j)}[t] = m_j[t] - m_i[t], \quad m_{D(i,j)}[t] = m_i[t]. \tag{8}$$

This allows for a localized scheme for calculating the quantities needed to calculate the optimal flows. This is illustrated in Figure 4, where it is shown how the upstream and downstream level for the input considered in Figure 2 can be calculated.

The aggregate costs γ_U and γ_D can be calculated in a similar way. Let

$$\gamma_i = \left(\frac{1}{q_i} + \sum_{j\in\mathcal{C}(i)} \frac{1}{\alpha^2\gamma_j}\right)^{-1}.$$

Then γ_i can be calculated by local communication, starting at the nodes which have no children. This allows γ_U and γ_D to be calculated as

$$\gamma_{U(i,j)} = \left(\frac{1}{\gamma_j} - \frac{1}{\alpha^2\gamma_i}\right)^{-1}, \quad \gamma_{D(i,j)} = \alpha^2\gamma_i.$$

Where again each node only needs to communicate with its neighbors.

The formulas proposed here give an efficient way to implement the controller that greatly reduces the communication requirements for each node.

This allows for a scalable implementation as the number of communication channels for each node remains low, even as the graph grows large. As the communication is in series, the time required for communication will grow as the depth of the graph increases. However, the depth is typically sub-linear in the number of nodes, with the exception being when the graph is a string.

The distributed iteration of the γ parameter also allows for the controller to be efficiently updated when the graph changes. If a node i is added to or removed from the network, then only those nodes that have i as an ancestor needs to update their feedback law. How many nodes this is will depend on the depth of node i and the graph topology.

2.4 Non-Rooted Trees

In this section the upstream and the downstream sets for non-rooted trees will be defined. We will also show that the control can still be implemented using local communication, similar to the previous subsection.

As there is no unique root in non-rooted trees we have to generalize the definition of the depth of a node.

DEFINITION 5
For a non-rooted tree we define the depth $d(i)$ of a node i as follows: Pick any node in the graph and define the depth for that node to be d_0. Then define the depth of all other nodes by the following rules.

- For a node i with parent j it holds that $d(i) = d(j) + 1$.
- For a node i with child k it holds that $d(i) = d(k) - 1$.

d_0 is then chosen so that $min_{i \in \mathcal{V}} d(i) = 0$. □

Note that now multiple nodes can satisfy $d(i) = 0$ and thus Assumption 2 allows for multiple producers.

We define the existence of an undirected path in the following natural way. This is used for the upcoming definitions.

DEFINITION 6
There exists an undirected path between node n_1 and node n_k if there exists a sequence of nodes $(n_1, n_2, \dots n_k)$ such that $(n_i, n_{i+1}) \in \mathcal{E}$ or $(n_{i+1}, n_i) \in \mathcal{E}$ for all $1 \leq i \leq k-1$. □

To describe the upstream and downstream sets, we split the nodes with the same depth into blocks. The definition is illustrated in Figure 5.

DEFINITION 7
Two nodes i and j are in the same block if they satisfy the following:

1. Node i and node j has the same depth, i.e. $d(i) = d(j)$.

2. There exists a undirected path between node i and node j when the incoming links to node i and node j are removed. □

For rooted trees each block contains only one node. Let the block of node i be denoted $B(i)$. For each block B we let $\mathcal{C}(B)$ denote the set of blocks which have a node with an incoming link from the block B. That is $c \in \mathcal{C}(B)$ if there exist nodes $(i \in c,\ j \in B)$ such that $(i, j) \in \mathcal{E}$. There can only be one such node i in each block, as otherwise there would be a cycle.

Each block P can be associated with a set of nodes $\mathcal{F}(P)$, which we call its family, in the following way.

Definition 8
For a block P, $i \in \mathcal{F}(P)$ if

- $i \in P$

or if

- There exists an undirected path from i to a node in P when all incoming links to the block P, that is $\forall(k, l),\ k \in P$, are removed. □

For a node $i \in P$ we also let $\mathcal{F}(i) = \mathcal{F}(P)$. The definition is illustrated in Figure 5. We note that the calculation of $m_i[t]$ in the previous subsection, it holds that $m_i[t] = m_{\mathcal{F}(i)}[t]$.

We are now ready to define the upstream and downstream sets for general directed trees. The upstream and downstream set for (i, j) together make up the family of the source node $\mathcal{F}(j)$. The definition is illustrated in Figure 5 and Figure 6.

Definition 9
For a non-rooted directed tree we define the upstream and the downstream set as follows.

A node k is in the upstream set $U(i, j)$ if the following holds:

1. $k \in \mathcal{F}(j)$
2. $k = j$ or there exists an undirected path between k and the source j when (i, j) is removed from $\mathcal{G}$.

A node k is in the downstream set $D(i, j)$ if the following holds:

1. $k \in \mathcal{F}(j)$
2. $k = i$ or there exists an undirected path from node k to the destination i when (i, j) is removed from $\mathcal{G}$. □

With the updated definition for the depth, and the upstream and downstream set, the theorem statements also hold for non-rooted trees.

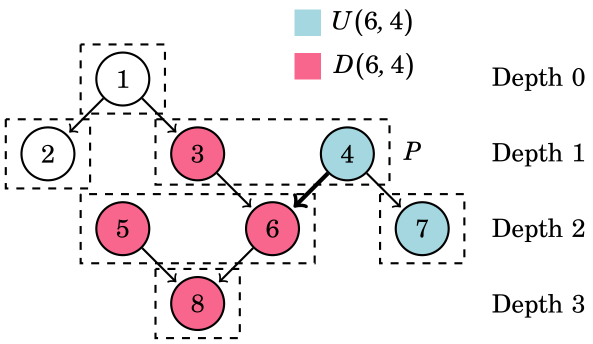


Figure 5. Illustration of the graph structuring. Each row of nodes in the picture of the graph corresponds to nodes that have the same depth. All blocks in the graph are marked by dashed squares. For the block P, containing nodes three and four, the set $\mathcal{F}(P)$ are all the colored nodes. For the link $(6,4)$, the upstream set is marked as blue and the downstream set as pink.

Proposition 1
If Definitions 3 and 4 are replaced by Definitions 5 and 9, then Theorems 1 and 3 hold whenever the underlying graph is a directed tree. □

We will now consider how the optimal controller for non-rooted trees can be implemented. To do this the following notation is introduced.

Definition 10
Consider an edge (i,j) where $j \in P$ and with upstream set $U(i,j)$ and downstream set $D(i,j)$. We let the nodes in $U(i,j)$ that are in P be denoted $U_P(i,j)$ and the set of children blocks of P who's nodes are in $U(i,j)$ as $U_C(i,j)$. Similarly define $D_P(i,j)$ as the set of nodes in P that are in $D(i,j)$ and the set of children blocks of P who's nodes are in $D(i,j)$ as $D_C(i,j)$. □

The definition is illustrated in Figure 6. Note that the nodes in each children block are either all in $U(i,j)$ or all in $D(i,j)$. Also note that both $U_C(i,j)$ and $D_P(i,j)$ can be empty. For a rooted tree, $U_P(i,j)$ is a set containing only the source node j and $D_P(i,j)$ is the empty set.

The idea behind the implementation is the same as for rooted trees in that we calculate an aggregate of the levels and the cost functions. The main difference is that now the aggregate level will be on block level instead of node level. We start by defining the aggregate level for blocks B without children. These blocks contain only one node $i \in B$, and we define for these blocks

$$m_B[t] = z_i[t] + \sum_{j \in \mathcal{P}(i)} u_{ij}[t-1].$$

Note that for non-rooted trees, each node might have multiple parents. By iterating through the graph, we define the aggregate level for block B with children blocks $\mathcal{C}(B)$, as

$$m_B[t] = \sum_{i \in B} \left(z_i[t] + \sum_{j \in \mathcal{P}(i)} u_{ij}[t-1] \right) + \sum_{c_i \in \mathcal{C}(B)} m_{c_i}[t]. \tag{9}$$

Each block needs information from its nodes and its children blocks. Once the block aggregate levels are known, the upstream and downstream level for every edge (i, j) can be calculated as

$$\begin{aligned} m_{D(i,j)}[t] &= \sum_{k \in D_P(i,j)} \left(z_k[t] + \sum_{l \in \mathcal{P}(i)} u_{kl}[t-1] \right) + \sum_{c_k \in D_C(i,j)} m_{c_k}[t] \\ m_{U(i,j)}[t] &= m_{B(j)}[t] - m_{D(i,j)}[t]. \end{aligned}$$

The calculation of the downstream aggregate requires information from the nodes in $D_P(i, j)$ and the blocks in $D_C(i, j)$. The upstream aggregate can reuse the calculation for $m_B(j)$ and $m_D(i, j)$ and does not require any additional communication.

We can also similarly define the aggregate cost for blocks B with no children nodes as $\gamma_B = q_i$ and for all other blocks as

$$\gamma_B = \left(\sum_{i \in b} \frac{1}{q_i} + \sum_{c_i \in \mathcal{C}(b)} \frac{1}{\alpha^2 \gamma_{c_i}} \right)^{-1}.$$

This allows for the upstream and downstream aggregate costs to be calculated in the following way

$$\begin{aligned} \gamma_{D(i,j)} &= \left(\sum_{k \in D_P(i,j)} \frac{1}{q_k} + \sum_{c_k \in D_C(i,j)} \frac{1}{\alpha^2 \gamma_{c_k}} \right)^{-1} \\ \gamma_{U(i,j)} &= \left(\frac{1}{\gamma_{B(j)}} - \frac{1}{\gamma_{D(i,j)}} \right)^{-1}. \end{aligned}$$

In the rooted tree case, each node only needed to communicate with its direct neighbors. For non-rooted trees each node needs to communicate with all the nodes in its block and all its children blocks. This is illustrated in Figure 6, where all the quantities needed to calculate the flow on the highlighted link are shown. This will in general require more communication channels, and also some abstraction in handling the block to node communication. One could, for example, assign one node as responsible for the calculations of m_B for each block and then share the aggregate with

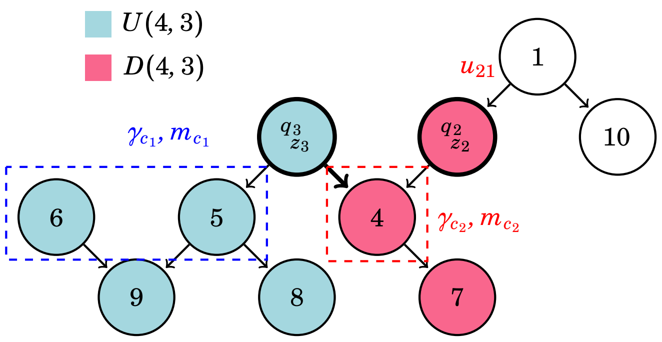


Figure 6. Illustration of the calculation for the flow on the highlighted link $(4, 3)$. The upstream set is highlighted in blue and the downstream set in pink. Furthermore $U_P = \{3\}$ and $D_P = \{2\}$ are highlighted as the two nodes with a thick border. The two children blocks have also been highlighted as dashed squares. It holds that $U_C = \{c_1\}$ and $D_C = \{c_2\}$, where $c_1 = \{5, 6\}$ and $c_2 = \{4\}$. Finally all the quantities needed to calculate the flow u_{21} have been written out. That is for the children block aggregate the levels m_{c_i} and the aggregate cost γ_{c_i}. And for the nodes in U_P and D_P, node level z_2 and z_3, node inflow u_{21}, and node costs q_2 and q_3.

the other nodes in the block. Still, each node needs at most to communicate with nodes that have a maximum of one depth difference. Thus the communication requirement will typically grow sub-linearly in the number of nodes in the graph, with the exception if the graph only has a depth of two.

2.5 Limitations and Generalizations

In this section, we will discuss the limitations imposed by the assumptions, and how they can be relaxed.

The assumptions on f and g in Assumption 1 are needed to ensure existence and uniqueness of the solution to (3), but still allow for quite general cost functions.

Assumption 2 is a rather natural assumption for rooted trees, in that the top node is the only producer. However, structured controllers can be achieved with Assumption 2 relaxed, see [Heyden et al., 2021] for the case of a string graph with production in every node. Similar results could be derived for the graph structures considered in this paper. However, due to space constraints, this was not considered. One implication for the assumption for non-rooted trees is that there can be nodes without parents that still can not be producers. This would in most cases be unnatural. However, the assumption can be relaxed in this case as well.

For the assumption that the graph is a directed tree, we note that a graph that is not a directed tree can always remove a transportation link while still allowing for any distribution of the quantity throughout the network. Directed tree graphs will thus appear naturally when one penalizes the number of transportation links when calculating the equilibrium flow. The downside is that the system is less robust to failures on links. However, a new equilibrium flow can be calculated if a failure occurs.

If the underlying graph is not a directed tree, then all cycles in the graph need to be removed in order to be able to apply the controller presented in this paper. For each cycle it is enough to remove one link to break the cycle, corresponding to fixing the flow to the equilibrium flow. This can always be done in a way so that the resulting graph after breaking all cycles is a directed tree. Synthesizing a controller on this reduced graph will of course not give the optimal performance. However, if it is necessary to have a scalable controller, the resulting performance might still be better compared to, for example, using a local controller. This method should only be applied when the equilibrium flow is already given, for example for an irrigation network that already contains cycles.

In the dynamics in (2), there is an implicit assumption that the transportation delays are homogeneous (there is a one-step delay associated with the transportation on each link). However, this can easily be overcome by introducing additional nodes with a cost function that makes the level always be zero. For non-homogeneous delays, the effect of a delay of τ_{ij} on a link (i, j) can be introduced by replacing the link with a series of $\tau_{ij} - 1$ fictitious nodes with the cost function

$$f_i(z_i) = \begin{cases} 0, & z_i = 0 \\ \infty, & z_i \neq 0. \end{cases}$$

Then the level in those nodes will always be zero, and the effect is the same as if there was a delay of τ_{ij}. Assumption 1 will still be satisfied as these new fictitious nodes will always have children. The special expression for the controller gains in the quadratic case can also be extended to this case. To do this, instead assign the fictitious nodes a quadratic cost $q_i z_i^2$. Theorems 1 and 2 can then be applied, and considering the limit $q_i \to \infty$ for the fictitious nodes gives the appropriate solution. Note that in this limit, all the controller gains in (7) will be well defined. All non fictitious nodes will have finite γ_U and γ_D. The fictitious nodes will have $\gamma_U \to \infty$ which results in $u_{ij}[t] = \alpha z_j[t]$ for all fictitious nodes j. This means that everything that is transported into a fictitious node is immediately transported out, as expected. Similarly, $\gamma_{\mathcal{V}}$ in Theorem 2 is well defined in the limit.

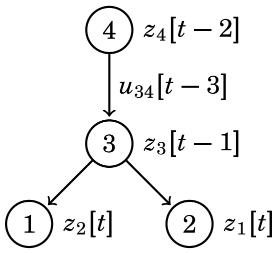


Figure 7. Illustration of a shifted level vector. The shift of each node is proportional to the layer of the node. The highlighted link flow $u_{34}[t-3]$ only directly affect the source $z_4[t-2]$ and the destination $z_3[t-1]$, which are both in the same shifted level.

3. Derivation of Optimal Controller

In this section, we prove the results presented in the previous section. Before getting into the details, we start by describing the main idea behind the proof.

Throughout this section, we will use the layer of a node instead of its depth, as that is notationally more convenient. We define the layer of a node by first finding the maximum layer,

$$l_{\max} = \max_{i \in \mathcal{V}} d(i).$$

The layer of a node is then defined as $l(i) = l_{\max} - d(i)$. As can be seen from the definition, the layers start from the bottom of the graph, while the depth starts from the top. We also let $l(B)$ denote the layer of the nodes in block B and T as the block in the maximum layer. Then it holds that $l(T) = l_{\max}$. There can only be one such block T as the graph is assumed to be connected.

3.1 Proof Idea

The proof relies on decomposing the problem into independent sub-problems. To do so a shifted level vector, defined for a block P as

$$\Big\{ z_i[t - l(i)], \quad i \in \mathcal{F}(P) \Big\},$$

will be studied. See Figure 7 for a graphical illustration. In Figure 7 it can be noted that the highlighted flow $u_{34}[t-3]$ will only directly affect the levels $z_3[t-1]$ and $z_4[t-2]$ which are both in the same shifted level. More generally, every flow $u_{ij}[t]$ will only directly affect the nodes within one shifted level vector.

We define the cost associated with a shifted level for a block P

$$F_P(z,t) := \sum_{i\in\mathcal{F}(P)} f_i(z_i[t-l(i)]).$$

This allows the node cost in (4) to be written in terms of the shifted levels in the following way,

$$\sum_{t=0}^{\infty}\sum_{i\in\mathcal{V}} f_i(z_i[t]) = \sum_{i\in\mathcal{V}} f_i(z_i[0]) + \sum_{t=1}^{l_{\max}} \sum_{P:l(P)=t-1} F_P(z,t) + \sum_{t=l_{\max}+1}^{\infty} F_T(z,t). \qquad (10)$$

This formula is illustrated for a three node string graph in Figure 8. Recall that T is the block consisting of the top layer, and $F_T(z,t)$ is then the cost for a shifted level containing all the nodes in the graph.

In Figure 8 we see that all but one of the levels within a shifted level can be chosen freely by choosing the outflow from each node appropriately. For example, it holds that $z_3[2]$ and $z_2[3]$ can be chosen freely by picking $u_{23}[1] = z_3[1] - z_3[2]$ and $u_{12}[2] = z_2[3] + u_{23}[1] - z_2[2]$. The last level $z_1[4]$ will then be decided by the total level of the previous shifted level. Thus the possible levels for the shifted vector $\{z_3[2],\ z_2[3],\ z_1[4]\}$ are only constrained by the sum of the previous shifted vector $z_3[1] + z_2[2] + z_1[3]$. Furthermore, the figure indicates that the sum of the shifted levels is independent of the internal flows u_{ij}. This implies that each shifted level can be optimized independently, as each flow decision only affects one shifted level. We will show that this holds for all directed trees by introducing a weighted sum of the shifted levels

$$S_P[t] = \sum_{i\in\mathcal{F}(P)} \alpha^{l(i)-l(P)} z_i[t-l(i)],$$

and showing that it satisfies a simple conservation law. The weighting reflects the fact that if we move some quantity c from a node to one of its children, then only αc will arrive, due to the decay.

3.2 Decomposition

We will now formalize the ideas discussed in the previous subsection. This will allow the problem in (3) to be decomposed into several independent sub-problems, as will be seen in Lemma 2.

The time-shifted levels are an important tool in the proof. However, the time shift is not suitable for controller implementation. Therefore it is necessary to establish a relationship between the shifted level and aggregate level, and we make the following definition.

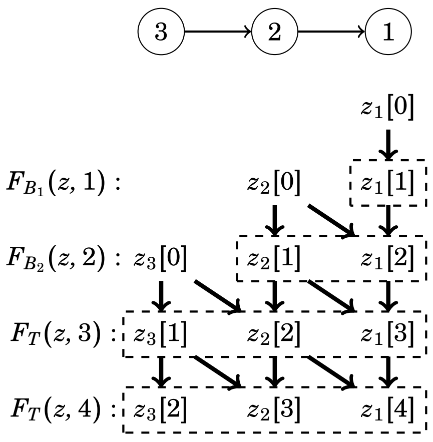


Figure 8. Illustration of the dynamics for the three node graph at the top of the picture. The graph has three blocks. The block B_1 contains node one, the block B_2 contains node two, and the top block T contains node three. Each node has been shifted proportional to its layer, making each horizontal slice a shifted level. The arrows indicate where the quantities in a node can go, i.e. either stay in the node or be transported, with a delay, to a child. Each term included in (10), except $f_i(z_i[0])$, has been highlighted by dashed rectangles.

Definition 11
Define the aggregate level $m_P[t]$ for block P as

$$m_P[t] = \sum_{i\in\mathcal{F}(P)} \left(z_i[t] + \sum_{j\in\mathcal{P}(i)} u_{ij}[t-1] \right). \qquad \square$$

This definition is different from (6) as it is defined on a block, and not on a set of nodes. However, it holds that $m_P[t] = m_{\mathcal{F}(P)}[t]$.

We will now show some important properties of the shifted level vectors and their relationship to the weighted sum. These properties are related to the dynamics and are independent of the optimization problems. Part (a) shows that the time update of a shifted level is given by the level at the previous time plus the inflow. For example in Figure 8, it can be seen that $S_2[3] = z_2[2] + z_1[3] = \alpha(z_2[1] + z_1[2] + u_{23}[0]) = \alpha(S_2[2] + u_{23}[0])$. Part (b) shows that a shifted level can be arbitrarily distributed as long as it satisfies the constraint given by (a), for example in Figure 8, $z_3[2]$ and $z_2[3]$ can be chosen freely by picking $u_{23}[1] = z_3[1] - z_3[2]$ and $u_{12}[2] = z_2[3] + u_{23}[1] - z_2[2]$. $z_1[4]$ will then be such that the constraint in (a) is satisfied. Part (c) shows that the internal flows has no effect on the shifted level sums in (10). In Figure 8 this can be seen, where the total level in the dashed rectangles is unaffected by the internal flows. (b) and (c) can then be used together to show that each shifted inventory level can be optimized independently. Finally, for part (d), it is shown how the current aggregate level can be used to predict a future shifted level. Again, from Figure 8 we can see that, for example, $\alpha z_3[1]+\alpha^2 z_2[2]+\alpha^3 z_1[3] = z_3[0]+z_2[0]+z_1[0]$, which gives that $\alpha m_T[0] = S_T[3]$.

Lemma 1
For any block P, define for $t \geq l(P)+1$

$$S_P[t] = \sum_{i\in\mathcal{F}(P)} \alpha^{l(i)-l(P)} z_i[t-l(i)].$$

Assume that $z[t]$ satisfies the dynamics in (2). Then

a) $S_P[t]$ satisfies the conservation law

$$S_P[t] = \alpha\left(S_P[t-1] + \sum_{\substack{i\in P\\ j\in\mathcal{P}(i)}} u_{ij}[t-l(i)-2]\right). \tag{11}$$

b) Given a shifted level for P and the inflow to P,

$$\begin{aligned} z_i[t-l(i)-1], &\quad i\in\mathcal{F}(P)\\ u_{ij}[t-l(i)-2], &\quad i\in P, \end{aligned} \tag{12}$$

then the only constraint on the next shifted level

$$z_i[t-l(i)], \quad i\in\mathcal{F}(P) \tag{13}$$

is that (11) is satisfied. In addition the flows

$$u_{ij}[t-l(j)-1],\ j\in\mathcal{F}(P)$$

are unique for every pair of values for (12) and (13).

c) Neither $S_P[\tau]$ for $\tau = l(P)+1$, nor $S_T[\tau]$ for $\tau > l_{\max}$, depend of the internal flows $u_{ij}[t]$, for $t\geq 0$.

d) It holds that

$$\alpha m_P[t-l(P)-1] = S_P[t]. \qquad \square$$

Proof Part (a), (c), and (d) are proved in the appendix. We prove (b) here. Recall that the dynamics are given by

$$z_i[t - l(i)] = \alpha\Big(z_i[t - l(i) - 1] + \sum_{h \in \mathcal{P}(i)} u_{ih}[t - l(i) - 2]\Big) - \sum_{j \in \mathcal{C}(i)} u_{ji}[t - l(i) - 1]. \quad (14)$$

A node level $z_i[t - l(i)], i \in P$ is determined when all of the node's inflows and outflows directly affecting the level, and the previous level $z_i[t-l(i)-1]$, are determined . It was assumed that the inflows to nodes in P,

$$u_{ij}[t - l(j) - 2],\ j \in P,$$

are given as well as the previous level

$$z_i[t - l(i) - 1],\ i \in \mathcal{F}(P).$$

Thus all that remains is the flows going between any pair of nodes in $\mathcal{F}(P)$. Consider the graph containing the nodes that have not yet fixed their level $z_i[t - l(i)]$, that is $i \in \mathcal{F}(P)$. And the edges that have not yet fixed their flow $u_{ij}[t - l(j) - 1]$, that is (i, j) $j \in \mathcal{F}(P)$. This graph is a directed tree and thus contains a node that is incident to only one edge (otherwise the graph would contain a cycle). The flow on this edge is the only term that is unspecified in (14) for the node and must be chosen according to the unique value so that the target node level in (13) is achieved.

Now the graph of edges with undecided flows and nodes with non fixed levels contain one edge and one node less. However, it is still a directed tree as all the remaining nodes stay connected and no cycles have been introduced. And there will still exist a node with only a single edge with an unspecified flow. The flow on this edge must again be uniquely chosen so that the wanted node level in (13) is achieved.

This can be continued until the graph of edges with undecided flows and nodes with non-fixed levels only contains two nodes and one edge. Then the flow on the last edge can be chosen so that at least one of the nodes gets the correct value. By (a), both nodes will have the correct level if and only if (11) is satisfied. This flow will be unique, as there is only one flow for each node that gives the correct level. □

Before we continue we will show that there exists a unique solution for the problems in (3) and (4). First, we show that there always exists an input trajectory that gives a finite cost. This is done by constructing a set of inputs that takes all the node levels to zero in finite time with finite control effort. Then all future cost will be zero, as $f_i(0) = g_i(0) = 0$ by Assumption 1. Let the production at time $t = 0$ be such that,

$$\sum_{i\in I} u_{i0}[0] = -\sum_{i\in\mathcal{V}}\Big(z_i[0] + \sum_{j\in\mathcal{P}(i)} u_{ij}[-1]\Big)$$

and $u_{i0}[t] = 0,\ t \geq 1$. Then $m_{\mathcal{V}}[t] = 0,\ \forall t \geq 1$. Now it only remains to find internal flows so that all levels are zero in finite time. This is possible by (b) and (d) in Lemma 1. Thus the sum over F_T will in (10) will be zero. All other sums F_P in (10) can be made finite by Lemma 1b as all $\mathcal{F}(P)$ will contain nodes with children which has finite cost by Assumption 1. The problems in (3) and (4) both have a strictly convex cost function, with a cost bounded from above by the previous argument, and a cost bounded from below by zero by Assumption 1. Thus the optimal values for $z_i[t]$ and $u_{i0}[t]$ for both (3) and (4) exist and are unique. The internal flows are not part of the cost function. However, it follows from (b) in Lemma 1, that they will also be unique.

We now use Lemma 1 to decompose the problem by showing that each shifted level, corresponding to a row in Figure 8, can be optimized independently.

Lemma 2
Assume that z is the minimizer for (3). Then for $1 \leq t \leq l_{max}$ and all blocks P s.t. $l(P) = t$, z is also the minimizer for

$$\begin{array}{ll} \underset{z_i[t-(i)],i\in\mathcal{F}(P)}{\text{minimize}} & F_P(z,t) \\ \text{subject to} & S_P[t] = \alpha m_P[0] \end{array}$$

And for $t > l_{max}$, z is the minimizer for

$$\begin{array}{ll} \underset{z_i[t-l(i)],i\in\mathcal{V}}{\text{minimize}} & F_T(z,t) \\ \text{subject to} & S_T[t] = \alpha m_T[t-l_{max}-1] \end{array}$$ □

Proof Recall that $F_P(z,t) = \sum_{i\in\mathcal{F}(P)} f_i(z_i[t-l(i)])$ and thus corresponds to a shifted level. By (b) in Lemma 1, the only constraint on a shifted level is that it satisfies (11). By (c) in Lemma 1 the shifted level corresponding to the constraint for each term in (10) is unaffected by the internal flows. Thus each term in (10) can be minimized independently, without affecting the optimal value of the other terms. Applying Lemma 1d to (11) completes the proof. □

3.3 Proof of the Theorems

Lemma 2 can be used to find the optimal levels z_i for the problems in (3) and (4) in a centralized way, relying on the method in the proof of

Lemma 1b to then find the optimal flows. However, to achieve the distributed implementation of the controller described in the previous section, one needs a more direct approach to finding the optimal link flows. To do this we need to understand how the flow $u_{ij}[t]$ affects the upstream and downstream sets of the corresponding link. If the flow on a link is zero, then both the upstream and the downstream set satisfy Lemma 1d. It is then natural that when the flow is not zero, what is taken from one set arrives in the other set. This is formalized in the following lemma.

Lemma 3
Assume that $z[t]$ satisfies the dynamics in (2). Then for every edge (i,j) where the source node j is in block P, and with upstream set $U(i,j)$ and downstream set $D(i,j)$, it holds that:

$$\begin{aligned} \sum_{k\in U(i,j)} \alpha^{l(k)-l(P)} z_k[t-l(k)] &= \alpha m_{U(i,j)}[t-l(P)-1] - u_{ij}[t-l(j)-1] \\ \sum_{k\in D(i,j)} \alpha^{l(k)-l(P)} z_k[t-l(k)] &= \alpha m_{D(i,j)}[t-l(P)-1] + u_{ij}[t-l(j)-1] \end{aligned} \tag{15}$$

□

For the proof, see the appendix. We are now ready to prove Theorem 3. The main idea is to show that the condition in Lemma 2 holds for all blocks inside $\mathcal{F}(P)$. This is similar to the principle of optimality in dynamic programming, where the optimal distribution for the entire $\mathcal{F}(P)$ must also be optimal for each subset of nodes in $\mathcal{F}(P)$. Applying Lemma 3 to each such block then gives the theorem.

Proof of Theorem 3 We will first show that a necessary condition for z_i to be a minimizer for (3) is to also be a minimizer for

$$\begin{aligned} \underset{z_k[t-(l(k)-1)],k\in\mathcal{F}(B)}{\text{minimize}} \quad & \sum_{k\in\mathcal{F}(B)} f_k(z_k[t-l(k)]) \\ \text{subject to} \quad & S_B[t] = \alpha m_B[t-l(B)-1], \end{aligned} \tag{16}$$

for all blocks B and all $t \geq l(B)+1$.

For any block B and time $t \geq l(B)+1$ we pick a block P in the following way: For $t \leq l_{\max}$ there exists a block P s.t. $\mathcal{F}(B) \in \mathcal{F}(P)$ and $t = l(P)+1$. For $t > l_{\max}$ take $P = T$, then $\mathcal{F}(B) \in \mathcal{F}(P)$. By Lemma 2 it then holds for that block P that the optimal z for (3) must also be a minimizer for

$$\begin{aligned} \underset{z_k[t-l(k)],k\in\mathcal{F}(P)}{\text{minimize}} \quad & \sum_{k\in\mathcal{F}(P)} f_k(z_k[t-l(k)]) \\ \text{subject to} \quad & S_P[t] = \alpha m_P[t-l(P)-1]. \end{aligned} \tag{17}$$

We have from the definition of $S_P[t]$ that

$$S_P[t] = \sum_{k\in\mathcal{F}(P)\setminus\mathcal{F}(B)} \alpha^{l(k)-l(P)} z_k[t-l(k)] + \alpha^{l(B)-l(P)} S_B[t],$$

and from Lemma 1d that $S_B[t] = \alpha m_B[t-l(B)-1]$. Thus z_i must be the minimizer of (16) to be the minimizer of (17). If not, the value of the objective function in (17) could be improved by using the minimizer for (16) for all $i \in \mathcal{F}(B)$ and using the same z_i for $i \in \mathcal{F}(P) \setminus \mathcal{F}(B)$. And it follows that for every block B that the optimal $z_i[t]$ for (3) must satisfy (16).

Next we show that the constraint in (16) holds if and only if the constraint in (15) holds. Adding the two equality-constraints in (15) gives the constraint in (16). Thus if (15) holds, then the constraint in (16) must also hold. Also if the constraint in (16) holds, then so must (15) for any $u_{ij}[t-l(j)-1]$. This can be seen by adding and subtracting u_{ij} to (16) and then splitting the equality into two. Thus for any outgoing link from B, that is $(i,j),\ j \in B$, z is a minimizer for (16) if and only if z and $u = u_{ij}[t-l(j)-1]$ are a minimizer for

$$\begin{aligned}
\underset{u,z_k}{\text{minimize}} \quad & \sum_{k\in U} f_k(z_k[t-l(i)]) + \sum_{k\in D} f_k(z_k[t-l(k)]) \\
\text{subject to} \quad & \sum_{k\in U} \alpha^{l(k)-l(B)} z_k[t-l(k)] = \alpha m_U[t-l(B)-1] - u \\
& \sum_{k\in D} \alpha^{l(k)-l(B)} z_k[t-l(k)] = \alpha m_D[t-l(B)-1] + u.
\end{aligned}$$

Let $x_i = z_i[t-l(i)]$, shift the time by $l(B)-1$ and use that $l(k)-l(B) = l(k)-l(j) = d(j)-d(k)$ and the theorem statement is achieved. Which then must be a necessary condition for optimality.

We previously showed that the optimal flows exists and are unique. Thus the necessary condition is also sufficient. □

The optimal internal flows for the general problem in Theorem 3 can be used to prove Theorem 1, but we must still find the optimal production for Theorem 2. We can use Lemma 2, i.e. the fact that the shifted level vectors are optimally distributed, to simplify the problem of finding the optimal production. This allows us to describe the cost that is affected by the production using only the total level $m_T[t]$. Once this has been done the problem of finding the optimal production can be rewritten as a problem with only one state and as many inputs as there are nodes with production.

PROPOSITION 2
Let $Q(m_T[t-l_{\max}-1])$ denote the optimal value for the optimization problem

$$\begin{aligned} \underset{z_i[t-l(i)],i\in\mathcal{V}}{\text{minimize}} \quad & \sum_{i\in\mathcal{V}} f_i(z_i[t-l(i)]) \\ \text{subject to} \quad & \sum_{i\in\mathcal{V}} \alpha^{l(i)-l_{\max}} z_i[t-l(i)] = \alpha m_T[t-l_{\max}-1] \end{aligned}$$

for $t \geq l_{\max}+1$. Then the optimal production $u_{i0}[t]$ for problem (3) is given by the minimizer for

$$\begin{aligned} \underset{u_{i0}}{\text{minimize}} \quad & \sum_{t=0}^{\infty} \left[Q(m_T[t]) + \sum_{i\in I} g_i(u_{i0}[t]) \right] \\ \text{subject to} \quad & m_T[t+1] = \alpha m_T[t] + \sum_{i\in I} u_{i0}[t]. \end{aligned}$$

□

Proof Recall that the cost can be written as in (10). From (d) in Lemma 1 we have that for $P : l(p) = t-1$, it holds that $S_P[t] = \alpha m_P[0]$, and thus the production only affect the terms $\sum_{t=l_{\max}+1}^{\infty} F_T(z,t)$, in (10). If the internal flows are chosen optimally, then by Lemma 2, the total cost that the inflows affect are then given by

$$\sum_{t=0}^{\infty} \left[Q(m_T[t]) + \sum_{i\in I} g_i(u_{i0}[t]) \right].$$

The fact that $m_T[t+1] = \alpha m_T[t] + \sum_{i\in I} u_{i0}[t]$ follows from (a) and (d) in Lemma 1, since

$$\begin{aligned} \alpha m_T[t-l_{\max}-1] &= S_T[t] \\ &= \alpha\Big(S_T[t-1] + \sum_{i\in I} u_{i0}[t-l_{\max}-2] \Big) \\ &= \alpha\Big(\alpha m_T[(t-1)-l_{\max}-1] + \sum_{i\in I} u_{i0}[t-l_{\max}-2] \Big). \end{aligned}$$

□

We are almost ready to prove Theorems 1 and 2. However, first we need the following lemma, which can be used to solve the problem in Theorem 3 and find the function Q in Proposition 2 when the cost functions are quadratic.

LEMMA 4
Consider the problem

$$\begin{aligned} &\text{minimize} && \sum_{i\in P} q_i m_i^2 \\ &\text{subject to} && \sum_{i\in P} \alpha^{l(i)-l(P)} m_i = m. \end{aligned} \tag{18}$$

Let

$$\gamma = \left(\sum_{i\in P} \frac{\alpha^{2(l(i)-l(P))}}{q_i} \right)^{-1}.$$

Then the minimizing m_i for (18) is given by

$$m_i = \frac{\alpha^{l(i)-l(P)}}{q_i} \gamma m$$

and the optimal value of the objective function is given by γm^2. □

For the proof, see the Appendix. Now all that remains is to prove Theorems 1 and 2. Lemma 4 allows the problem in Theorem 3 to be solved algebraically, giving a closed-form solution. The problem in Proposition 2 can be recast as a standard LQ problem with one state and one input, which can also be solved exactly.

Proof of Theorems 1 and 2 Using Lemma 4 and that the cost functions are quadratic, Theorem 3 gives that the optimal u_{ij} for (4) is given by the minimizer for

$$\gamma_{U(i,j)}(\alpha m_{U(i,j)}[t] - u)^2 + \gamma_{D(i,j)}(\alpha m_{D(i,j)}[t] + u)^2.$$

Differentiating with respect to u gives the optimal link flow as in Theorem 1.

By Lemma 4 we have that for quadratic cost functions, the function $Q(m_T[t])$ in Proposition 2 is given by $Q(m_T[t]) = \gamma_T \alpha^2 m_T^2[t]$. Let the total production at time t be denoted $U = \sum_{i\in I} u_{i0}[t]$. Then if the total production is split optimally along the producers, the cost is $\sum_{i\in I} g_i(u_{i0}[t]) = RU^2$, and each individual production is given by $u_i = RU/r_i$ (by application of Lemma 4 with $\alpha = 1$). This allows us to formulate the problem in Proposition 2 as a standard linear quadratic control problem:

$$\begin{aligned} &\underset{U}{\text{minimize}} && \sum_{t=0}^{\infty} \gamma_T \alpha^2 m_T[t]^2 + RU[t]^2 \\ &\text{subject to} && m_T[t+1] = \alpha m_T[t] + U[t] \end{aligned}$$

This problem can be solved using the Riccati equation, see for example [Bertsekas, 1995]. Let

$$A = \alpha, \quad B = 1, \quad Q = \alpha^2 \gamma_T.$$

Then the solution to the scalar Riccati equation

$$A^T X A - (A^T X B)(B^T X B + R)^{-1}(A^T X B)^T + Q = X,$$

is as given in the theorem statement. The optimal total production then follows from

$$U[t] = -(B^T X B + R)^{-1} B^T X A m_T[t]. \qquad \square$$

4. Simulation Examples

In this section, we will study the performance of the optimal controller in networks of different sizes and topologies. We aim to illustrate the closed-loop behavior and study how the control performance scales with the size of the graph. We restrict ourselves to quadratic cost functions and study two cases designed to illustrate dynamic performance in the face of changes in equilibrium point and random disturbances. For both cases, the simulations were run on two different types of graphs; a directed string graph (every node except the last has one child), and a binary tree graph (every node has two children, except for those at the maximum depth). All code used to generate the figures is available on GitHub[2] along with code for a numerical test of Theorem 1 and Theorem 2 .

First, we consider performance when the network is subject to non-zero initial conditions. This could for instance be of interest if the underlying equilibrium is changing (the initial conditions correspond to the difference between the old equilibrium point, and the new one). From the derivation of the optimal controller, we saw that this case can be split into two parts. The optimal distribution of the available goods, and the optimal production. The optimal production essentially corresponds to solving a first-order system, and we will focus on the optimal distribution here. Thus we will normalize the initial conditions to sum to zero. To ensure a wide range of initial conditions were considered, the initial node levels $z_i[0]$ and the initial transport levels $u_{ij}[-1]$ were drawn from a multivariable Gaussian distribution

$$\mathcal{N}\left(0, \frac{M}{M-1} \cdot \left[I - \frac{\mathbf{1}\mathbf{1}^T}{M}\right]\right),$$

[2] `https://github.com/Martin-Heyden/TAC-Directed-Trees`

where M is the sum of the total number of nodes and links. This choice guarantees that the initial values sum to zero, i.e.

$$\sum_{i\in\mathcal{V}}\left(z_i[0]+\sum_{j\in\mathcal{P}(i)}u_{ij}[-1]\right)=0,$$

and keeps the diagonal elements of the covariance matrix the same for all graph sizes. For simplicity, the cost function for each node was set to $q_i = 1$. The decay factor was $\alpha = 0.99$ and the simulations were run 1000 times for each graph size.

Secondly, we consider the case of persistent Gaussian disturbances acting on the node levels $z_i[t]$. This case could be motivated for example by having an unknown demand on the nodes of the underlying system. We again set $q_i = 1$ for all nodes, and $\alpha = 0.99$. The state disturbances were set to be normally distributed with zero mean and a standard deviation of 0.01. We considered the optimal controller both with and without production. For the case of production, we set the production cost to be $r = 10$. 1000 simulations were run for each graph size, and each simulation was run for 100 time steps.

To provide some sort of comparison, a local controller was also designed with off-the-shelf computational tools and tested on the same simulation cases. By local, we mean that the local controller was forced to satisfy certain structural constraints restricting which variables were available for feedback. In particular, the transportation on each link was decided based only on the levels in the source node, the children of the source node, and the goods in transit towards those nodes. For simplicity, the control law was further restricted to be static. The Matlab function `fmincon` was used to conduct the non-convex optimization corresponding to minimizing the $\mathcal{H}_2$ gain over all controllers satisfying the sparsity constraints. That is minimizing $||C(A+BK)^{-1}B||_2$ where A, B, C is a state space representation of the system from u_{ij} to z_k and K satisfy the sparsity constraints. The full details can be found in the code. Of course, there are no guarantees that this process will yield an optimal controller subject to the constraints (indeed, for larger graphs, not even a local minima was found). However, since the controllers designed using the tools from this paper are globally optimal (they give a structured solution to a standard unconstrained LQ problem), the performance of any local controller will necessarily be worse, and its purpose is more to provide perspective than represent a 'good way' to design decentralized controllers.

The simulation results for the first case (random initial conditions) are presented in Figures 9 and 10. The cost per node for different graph sizes can be seen in Figure 9. We can see for the string graph that the optimal controller gets slightly better performance as the number of nodes increases.

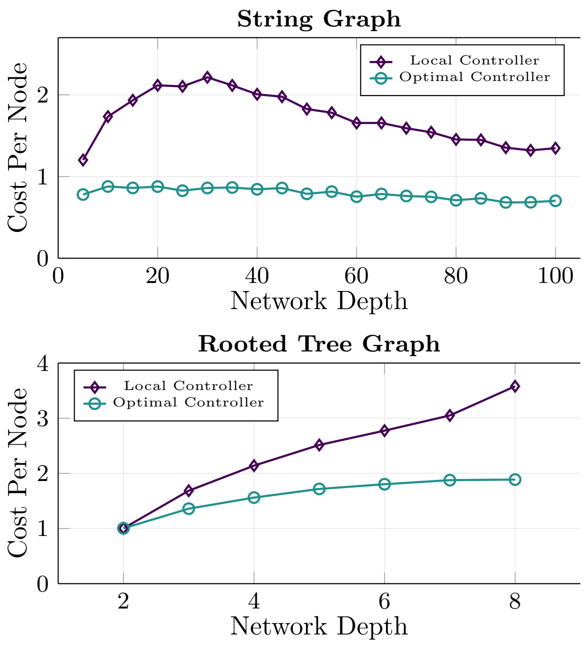


Figure 9. Performance comparison for the local and the optimal controller for a string graph and a rooted-tree graph with non-zero initial conditions. In both cases, the initial values have been chosen so that they sum to zero.

This is not too surprising, since it would otherwise, for example, suggest that two string graphs with forty nodes could be controlled better independently than one graph with eighty nodes (if we ignore the fact that the initial conditions need not sum to zero in each graph of forty nodes). For the local controller, the performance first decreases, and then increases. This might be due to two competing effects. On the one hand, it should be easier to control a longer string for the same reasons as for the optimal controller. However, as the graph gets shorter each local controller has access to a larger subset of the total information, which should increase performance. For the tree case, we again see that the performance of the local controller decreases as the depth of the graph increases. We could expect similar behavior as for the string case. However, it is not feasible to synthesize the local controller for deeper graphs due to the exponential increase in the number of nodes. We can also note that the string structure seems more efficient than the tree structure for this problem, in that it yields a lower cost per node. The state trajectories for a few nodes in a string graph of length 20 have been plotted in Figure 10. Note that the optimal controller

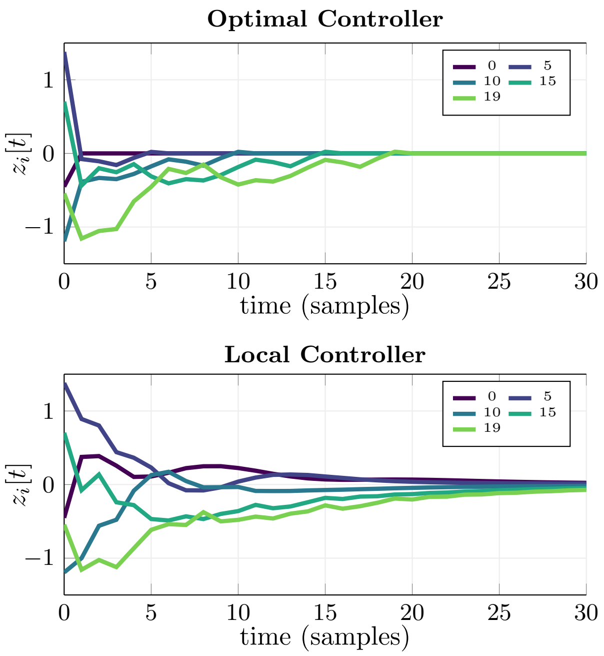


Figure 10. State trajectories of the optimal and the local controller for a string graph with 20 nodes and non-zero initial conditions. The legend indicates the depth of the plotted node.

brings the system to the optimal configuration at time 20. This is because the initial condition sums to zero, and then all shifted levels $S_T[t]$ will sum to zero, and thus all $z_i[t - l(i)]$ will equal zero for $t > l_{\max}$.

The simulation results for the second case (persistent Gaussian disturbances) for different graph sizes can be seen in Figure 11 which shows that the performance for the optimal controller without production and the local controller both improve as the graph grows. However, this feature all but disappears when production is allowed. This is likely due to the fact that the larger the graph is, the more likely it is that the total level is zero, meaning the need for production declines as the graphs become larger. Secondly, while the optimal controller still outperforms the local one, the difference is much smaller now. We can also note that the difference in performance between the string case and the tree case is negligible now.

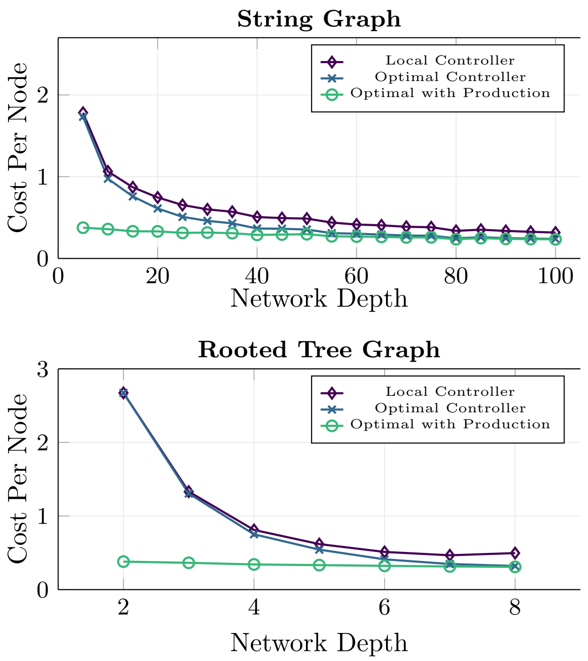


Figure 11. Performance comparison for the local and the optimal controller for a string graph and a rooted-tree graph. In both cases the system is subjected to Gaussian noise.

5. Conclusion

We have studied a class of optimal transportation problems over a network, where the transportation is subject to delay. It was shown that under simple modeling assumptions the optimal control policy has a sparse and structured solution, suitable for large-scale systems. In the case of quadratic cost functions, closed-form expressions for the optimal control were derived. The controller was compared to a local controller designed with off-the-shelf methods. The optimal controller showed a significant improvement in performance for the case of non-zero initial conditions, corresponding to the dynamical adjustment from one equilibrium point to another.

Acknowledgment

The authors gratefully acknowledge financial support from Swedish Foundation for Strategic Research through the project SSF RIT15-0091 SoPhy.

A. Appendix

Proof of Lemma 1 **(a)** For any block without children, which is just a node without children, the lemma follows from the dynamics. Now assume that for a block P the Lemma holds for all the children blocks $\{C \in \mathcal{C}(P)\}$. Then for all such blocks C it holds that

$$S_C[t] = \alpha\Big(S_C[t-1] + \sum_{\substack{i\in C\\ j\in\mathcal{P}(i)}} u_{ij}[t-l(i)-2]\Big). \tag{19}$$

The shifted level for P can be rewritten as

$$S_P[t] = \sum_{j\in P} z_j[t-l(j)] + \frac{1}{\alpha}\sum_{C\in\mathcal{C}(P)} S_C[t]. \tag{20}$$

Using the dynamics, the sum over the nodes in the parent block can be rewritten as

$$\sum_{j\in P} z_j[t-l(j)] = \sum_{j\in P}\Bigg[\alpha\Big(z_j[t-l(j)-1] + \sum_{k\in\mathcal{P}(j)} u_{jk}[t-l(j)-2]\Big) - \sum_{i\in\mathcal{C}(j)} u_{ij}[t-l(j)-1]\Bigg]. \tag{21}$$

Since $\mathcal{P}(i) = P$ for $i \in \mathcal{C}(P)$ and $l(i) = l(j) - 1$ for $j \in \mathcal{P}(i)$, it holds that

$$\sum_{\substack{i\in C\\ j\in\mathcal{P}(i)}} u_{ij}[t-l(i)-2]\Big) = \sum_{j\in P}\sum_{i\in\mathcal{C}(j)} u_{ij}[t-l(j)-1].$$

Inserting (19) and (21) into (20) proves the statement.

(c) Let $\tau = 1$. Then for all P s.t. $l(P) = \tau - 1$ the set $\mathcal{F}(P)$ is just a node, and for that node the lemma holds, as

$$S_P[1] = z_i[1] = \alpha\Big(z_i[0] + \sum_{j\in\mathcal{P}(i)} u_{ij}[-1]\Big).$$

Now assume the lemma holds for for all blocks B s.t. $l(B) = \tau - 2$, for some fixed $\tau \le l_{\max} + 1$. Then for any P such that $l(P) = \tau - 1$, using (20) and (11) gives

$$S_P[\tau] = \sum_{i\in P} z_i[0] + \sum_{\substack{i\in P\\ j\in\mathcal{P}(i)}} u_{ij}[-1] + \sum_{C\in\mathcal{C}(P)} S_C[\tau-1].$$

Thus (c) holds for $\tau \leq l_{\max}+1$ as $l(C)=\tau-2$.

Now we consider $S_T[\tau]$. The case of $\tau = l_{\max}+1$ follows from above. Let $\tau = l_{\max}+1+\bar{\tau},\ \bar{\tau} \geq 1$. Repeated application of (11) to $S_T[l_{\max}+1+\bar{\tau}]$ gives that

$$S_T[l_{\max}+1+\bar{\tau}] = \alpha^{\bar{\tau}} S_T[l_{\max}+1] - \sum_{t=0}^{\bar{\tau}-1}\Big(\alpha^{\bar{\tau}-t}\sum u_{i0}[t]\Big).$$

(d) The lemma holds for blocks with no children blocks, that is nodes with no children, as the dynamics gives

$$\begin{aligned} S_i[t] &= z_i[t-l(i)] \\ &= \alpha\Big(z_i[t-l(i)-1] + \sum_{j\in\mathcal{P}(i)} u_{ij}[t-l(i)-2]\Big) \\ &= \alpha m_i[t-l(i)-1]. \end{aligned}$$

Now assume that the lemma holds for all children blocks of some block P. $S_P[t]$ can be written as as

$$S_P[t] = \sum_{i\in P} z[t-l(i)] + \frac{1}{\alpha}\sum_{C\in\mathcal{C}(P)} S_C[t].$$

Applying the dynamics for the first term gives (21). For the second term (a) gives

$$\frac{1}{\alpha}S_C[t] = S_C[t-1] + \sum_{\substack{i\in C\\ j\in\mathcal{P}(i)}} u_{ij}[t-l(i)-2].$$

Using the induction assumption we have that $S_C[t-1] = \alpha m_C[t-l(C)-2] = \alpha m_C[t-l(P)-1]$. The internal flows cancel and thus P also satisfies the lemma. □

Proof of Lemma 3 The edge (i,j) splits the set $\mathcal{F}(P)$ into two parts, the upstream set $U(i,j)$ and the downstream set $D(i,j)$. The first step of the proof is to consider the dynamics on these sets when $u_{ij}[t-l(j)-1]=0$. This allows Lemma 1d to be applied to the upstream and the downstream sets. The second step is to consider the effect caused by a non zero flow on u_{ij}. This will be easy as this change only affects two nodes.

Consider the nodes in the upstream set that are also in P, that is U_P (recall Definition 10). When $u_{ij}=0$ Lemma 1d can be applied to U_P as if it were a block, since it would be a block if the edge (i,j) was removed. This gives

$$\sum_{k\in U(i,j)} \alpha^{l(k)-l(P)} z_k[t-l(k)] = \alpha m_U[t-l(P)-1], \tag{22}$$

where we have used that $U(i,j) = \mathcal{F}(U_P)$ and $m_{U_P} = m_U$.

For the downstream set there are two cases. We will show that for both cases it holds that

$$\sum_{k \in D(i,j)} \alpha^{l(k)-l(P)} z_k[t - l(k)] = \alpha m_D[t - l(P) - 1]. \tag{23}$$

If the set of nodes D_P is non empty, then Lemma 1d can be applied to D_P when $u_{ij} = 0$ (following the same logic as for the U_P),

$$\sum_{k \in \mathcal{F}(D_P)} \alpha^{l(k)-l(P)} z_k[t - l(k)] = \alpha m_{D_P}[t - l(D_P) - 1].$$

This is equivalent to (23) as $l(D_P) = l(P)$. If D_P is empty then the set of blocks D_C contains only one block with (i,j) as its only incoming link, as otherwise D_P would not be empty. With slight abuse of notation[3] Lemma 1d gives that

$$\sum_{k \in \mathcal{F}(D_C)} \alpha^{l(k)-l(D_C)} z_k[t - l(k)] = \alpha m_{D_C}[t - l(D_C) - 1].$$

However, $m_{D_C}[t - l(D_C) - 1] = \alpha m_{D_C}[t - l(D_C) - 2]$ when the flow on (i,j) is zero, as then there is no inflow to the block m_{D_C}. As $l(D_C) = l(P) - 1$ it thus holds that

$$\frac{1}{\alpha} \sum_{k \in \mathcal{F}(D_C)} \alpha^{l(k)-l(D_C)} z_k[t - l(k)] = \alpha m_{D_C}[t - l(P) - 1].$$

Finally, using that $1/\alpha \cdot \alpha^{-l(D_C)} = \alpha^{-l(P)}$ gives

$$\sum_{k \in \mathcal{F}(D_C)} \alpha^{l(k)-l(P)} z_k[t - (l(k) - 1)] = \alpha m_{D_C}[t - l(P) - 1].$$

which is equivalent to (23).

Now consider the effect of changing to a non zero value for $u_{ij}[t - l(i) - 1]$ while keeping all other flows constant on (22) and (23). All levels other than the source j and the destination i will be unaffected by this change. The source will decrease its level by u and the destination will increase its level by αu. However, the destination node is weighted by $1/\alpha$, which gives the lemma. $\square$

[3] Note that D_C is defined to be a set of blocks. However, here the set only contains one block, and we use it to describe that block.

Proof of Lemma 4 Decreasing m_i by ϵ allows m_j to be increased by

$$\epsilon\alpha^{l(i)-l(P)}/\alpha^{l(j)-l(P)}.$$

At optimality it must thus hold that

$$q_i m_i = q_j m_j \frac{\alpha^{l(i)-l(P)}}{\alpha^{l(j)-l(P)}}.$$

Which can be rewritten as

$$m_j = \frac{q_i}{q_j}\frac{\alpha^{l(j)}}{\alpha^{l(i)}} m_i.$$

The constraint can then be rewritten as

$$\alpha^{l(i)-l(P)} m_i + \sum_{\substack{j\in P\\ j\neq i}} \alpha^{l(j)-l(P)} \frac{q_i}{q_j}\frac{\alpha^{l(j)}}{\alpha^{l(i)}} m_i = m,$$

which gives that

$$\frac{m_i q_i}{\alpha^{l(i)-l(P)}}\left(\sum_{j\in P}\frac{\alpha^{2(l(j)-l(P))}}{q_j}\right) = m.$$

From which the expression for the optimal m_i follows. The optimal value is achieved by inserting the optimal value for each m_i into the cost function:

$$\sum_{i\in P} q_i m_i^2 = \gamma^2 m^2 \sum_{i\in P} q_i \left(\frac{\alpha^{l(i)-l(P)}}{q_i}\right)^2 = \gamma m^2$$ □

References

Anderson, J., J. C. Doyle, S. H. Low, and N. Matni (2019). “System level synthesis”. *Annual Reviews in Control*.

Bamieh, B., F. Paganini, and M. A. Dahleh (2002). “Distributed control of spatially invariant systems”. *IEEE Transactions on automatic control* **47**:7, pp. 1091–1107.

Bergeling, C., R. Pates, and A. Rantzer (2020). “H-infinity optimal control for systems with a bottleneck frequency”. *IEEE Transactions on Automatic Control* **66**:6, pp. 2732–2738.

Bertsekas, D. P. (1995). *Dynamic programming and optimal control*. Vol. 1. Athena scientific Belmont, MA.

Cantoni, M., E. Weyer, Y. Li, S. K. Ooi, I. Mareels, and M. Ryan (2007). “Control of large-scale irrigation networks”. *Proceedings of the IEEE* **95**:1, pp. 75–91.

Coron, J.-M., B. d’Andréa-Novel, and G. Bastia (1999). “A lyapunov approach to control irrigation canals modeled by saint-venant equations”. In: *1999 European control conference (ECC)*. IEEE, pp. 3178–3183.

D’Andrea, R. and G. E. Dullerud (2003). “Distributed control design for spatially interconnected systems”. *IEEE Transactions on automatic control* **48**:9, pp. 1478–1495.

Evans, R., L. Li, I. Mareels, N. Okello, M. Pham, W. Qiu, and S. K. Saleem (2011). “Real-time optimal control of river basin networks”. *IFAC Proceedings Volumes* **44**:1, pp. 11459–11464.

Grant, M. and S. Boyd (2014). *CVX: matlab software for disciplined convex programming, version 2.1*. http://cvxr.com/cvx.

Heyden, M., R. Pates, and A. Rantzer (2018). “A structured linear quadratic controller for transportation problems”. In: *2018 European Control Conference (ECC)*. IEEE, pp. 1654–1659.

Heyden, M., R. Pates, and A. Rantzer (2021). “A structured optimal controller with feed-forward for transportation”. *IEEE Control Systems Letters*.

Kelly, F., A. Maulloo, and D. Tan (1998). “Rate control for communication networks: shadow prices, proportional fairness and stability”. *Journal of the Operational Research society* **49**:3, pp. 237–252.

Kundur, P. (1994). *Power System Stability and Control*. McGraw-Hill Professional. ISBN: 007035958X.

Lin, F., M. Fardad, and M. R. Jovanović (2013). “Design of optimal sparse feedback gains via the alternating direction method of multipliers”. *IEEE Transactions on Automatic Control* **58**:9, pp. 2426–2431. DOI: 10.1109/TAC.2013.2257618.

Lin, P.-H., D. S.-H. Wong, S.-S. Jang, S.-S. Shieh, and J.-Z. Chu (2004). “Controller design and reduction of bullwhip for a model supply chain system using z-transform analysis”. *Journal of process control* **14**:5, pp. 487–499.

Litrico, X. and V. Fromion (2005). “Design of structured multivariable controllers for irrigation canals”. In: *Proceedings of the 44th IEEE Conference on Decision and Control*. IEEE, pp. 1881–1886.

Madjidian, D. and L. Mirkin (2014). “Distributed control with low-rank coordination”. *IEEE Transactions on Control of Network Systems* **1**:1, pp. 53–63.

Mårtensson, K. and A. Rantzer (2009). “Gradient methods for iterative distributed control synthesis”. In: *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*. IEEE, pp. 549–554.

Radner, R. (1962). “Team decision problems”. *The Annals of Mathematical Statistics* **33**:3, pp. 857–881.

Rantzer, A. (2019). “Realizability and internal model control on networks”. In: *2019 18th European Control Conference (ECC)*. IEEE, pp. 3475–3477.

Rotkowitz, M. and S. Lall (2005). “A characterization of convex problems in decentralized control”. *IEEE transactions on Automatic Control* **50**:12, pp. 1984–1996.

Schuurmans, J., A. Hof, S. Dijkstra, O. Bosgra, and R. Brouwer (1999). “Simple water level controller for irrigation and drainage canals”. *Journal of irrigation and drainage engineering* **125**:4, pp. 189–195.

Shah, P. and P. A. Parrilo (2013). “H_2-optimal decentralized control over posets: a state-space solution for state-feedback”. *IEEE Transactions on Automatic Control* **58**:12, pp. 3084–3096.

Subramanian, K., J. B. Rawlings, C. T. Maravelias, J. Flores-Cerrillo, and L. Megan (2013). “Integration of control theory and scheduling methods for supply chain management”. *Computers & Chemical Engineering* **51**, pp. 4–20.

Ukkusuri, S. V. and K. M. A. Özbay (2013). *Advances in Dynamic Network Modeling in Complex Transportation Systems*. Springer.

Varian, H. R. (2003). *Intermediate Microeconomics:A Modern Approach*. 6th ed. W. W. Norton & Company. ISBN: 0-393-97830-3.

Wang, Y., N. Matni, and J. C. Doyle (2018). “Separable and localized system-level synthesis for large-scale systems”. *IEEE Transactions on Automatic Control* **63**:12, pp. 4234–4249. ISSN: 0018-9286. DOI: 10.1109/TAC.2018.2819246.

Weyer, E. (2001). “System identification of an open water channel”. *Control engineering practice* **9**:12, pp. 1289–1299.

Paper II

A Structured Optimal Controller with Feed-Forward for Transportation

Martin Heyden Richard Pates Anders Rantzer

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.

An extended version of M. Heyden, R. Pates and A. Rantzer, "A Structured Optimal Controller With Feed-Forward for Transportation," in IEEE Control Systems Letters, vol. 6, pp. 1130-1135, 2022, doi: 10.1109/LCSYS.2021.3088666. 2021 ©IEEE, reprinted with permission.

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 [Kalman et al., 1960]. 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 [Rotkowitz and Lall, 2006; Lessard and Lall, 2011], System Level Synthesis [Wang et al., 2018], and the use of large-scale optimization techniques (e.g. [Lin et al., 2013]). A downside with these approaches is that they improve scalability 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 $v_1, v_2, \ldots, v_N$ and directed edges $(e_N, e_{N-1}), \ldots, (e_2, e_1)$. The dynamics are given by

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

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 τ_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 ith node. Finally $d_i[t] \in \mathbb{R}$ is the fixed production/consumption at the ith 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 [Cantoni et al., 2007] or a simple supply chain system [Subramanian et al., 2013]. A state-space representation for (1) can be obtained by setting $z_i[t]$, $u_i[t-\delta]$, $1 \le \delta \le \tau_i$ 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,

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

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 [Ahuja et al., 1995], and could be used to compliment such approaches by optimally adjusting a nominal schedule in real time using the feedback principle.

A similar problem has been studied in a previous paper [Heyden et al., 2018]. 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

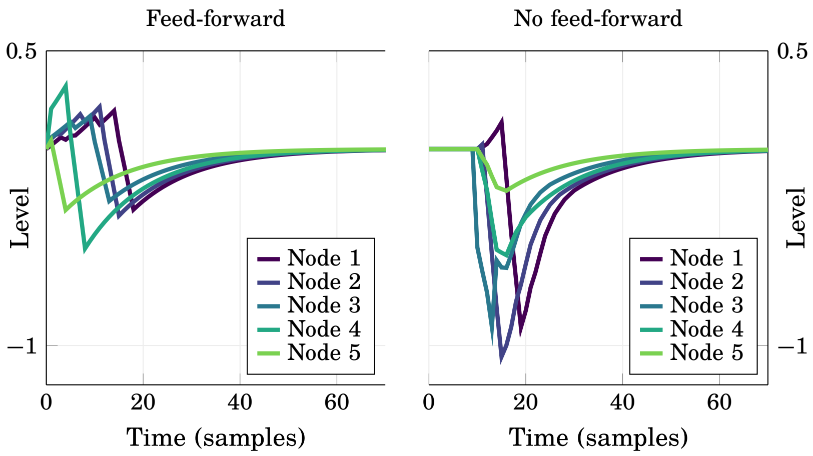


Figure 1. Example of the effect of feed forward. The graph has five nodes and transportation delay $\tau_1 = 3$, $\tau_2 = 2$, $\tau_3 = 5$ and $\tau_4 = 4$. There is a disturbance in node three from time 10 to 13 and in node 2 from time 12 to 15. We can see that the feed-forward manages to handle the 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.

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 ith node $\delta_i[t]$ and $\mu_i[t]$ can be computed recursively through relationships on the form

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

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 δ's, and once from the final node to the first to compute the μ'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 σ_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 σ_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

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 */
$\gamma_1 = q_1, \quad \rho_1 = r_1$                                  // initialize first node
send $\gamma_1$ and $\rho_1$ to upstream neighbor
for node i = 2:N do
    $\gamma_i = \frac{\gamma_{i-1} q_i}{\gamma_{i-1}+q_i}, \quad \rho_i = \frac{\rho_{i-1} r_i}{\rho_{i-1}+r_i}$
    send $\gamma_i$ and $\rho_i$ to upstream neighbor
end
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
/* Second Sweep Downstream direction */
$X_N(H+2) = -\frac{\gamma_N}{2} + \sqrt{\gamma_N \rho_N + \frac{\gamma_N^2}{4}}$
for node i = N:1 do
    $X_i(\tau_i) = \frac{\rho_i(X_{i+1}(1)+\gamma_i)}{X_{i+1}(1)+\gamma_i+\rho_i}$              // Not for node N
    $X_i(t-1) = \frac{\rho_i(X_i(t)+\gamma_i)}{X_i(t)+\gamma_i+\rho_i}$,   // $1 \le t-1 \le \tau_i - 1$ or for i =N, $1 \le t-1 \le H+1$
    $g_i(i) = \frac{X_i(i)}{X_i(i)+\gamma_i}, \quad 2 \le i \le \tau_i$
    $g_{i+1}(1) = \frac{X_{i+1}(1)}{X_{i+1}(1)+\gamma_i}$
    $b_i = g_{i+1}(1) \prod_{j=2}^{\tau_i} g_i(j)$
    send $X_i(\tau_i)$, $b_i$ to downstream neighbor.
    // for $1 \le l, m \le \tau_i$
    $P_i(1,m) = \frac{X_i(1)}{\rho_i}$
    $P_i(l,m) = (1 - \frac{X_i(l)}{\rho_i}) g_i(l) P_i(l-1,m) + \frac{X_i(l)}{\rho(i)}, \quad l \le m$
    $P_i(l,m) = (1 - \frac{X_i(l)}{\rho_i}) P_i(l-1,m) + \frac{X_i(l)}{\rho(i)}, \quad l > m$
end
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
/* Third Sweep, upstream direction */
$h_1 = P_1(\tau_1, \tau_1) g_2(1)$
send $h_1$ to upstream neighbor.
for node i= 2:N-1 do
    $h_i = (1 - P_i(\tau_i, 1)) b_i h_{i-1} + P_i(\tau_i, \tau_i) g_{i+1}(1)$
    send $h_i$ to upstream neighbor.
end
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
/* Some final local Calculations */
// For $1 \le \Delta \le \tau_i$, Empty Product, $\prod_{j=2}^{1}$ = 1
$\phi_i(\Delta) = \left(1 - P_i(\tau_i, \Delta) - (1 - P_i(\tau_i, 1)) h_{i-1} \prod_{j=2}^{\Delta+1} g_k(j)\right)$
$a_i = \frac{X_i(1)}{r_i} + \frac{\gamma_i}{q_i}(1 - \frac{X_i(1)}{\rho_i})$
$c_i = -\left(\frac{X_i(1)}{r_i} - \frac{\gamma_i X_i(1)}{q_i \rho_i}\right)(1 - h_{i-1}) + \frac{\gamma_i}{q_i} h_{i-1}$
```

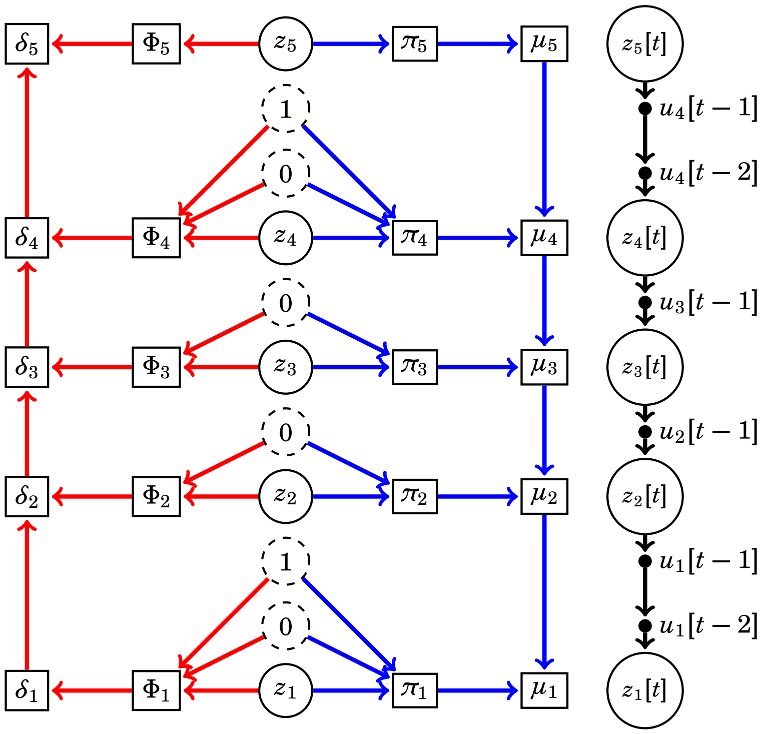


Figure 2. Illustration of the structured approach for calculating the optimal inputs using Algorithm 2, for a 5 node example with $\tau_1 = 2$, $\tau_2 = 1$, $\tau_3 = 1$, $\tau_4 = 2$. The graph at the right of the figure illustrates the underlying dynamics of the network as in (1). The left part of 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 Δ, 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 ith node can be determined using the variables at the ith 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      $\Phi_i[t] = \phi_i(1)z_i[t] + \sum_{\Delta=0}^{\tau_i-1} \phi_i(\Delta+1)\Big(u_i[t-(\tau_i-\Delta)] + D_i[t+\sigma_i+\Delta]\Big)$
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
       // Empty Product, $\prod_{j=2}^{1}$ = 1
7      $\pi_i[t] = z_i[t] + \sum_{\Delta=0}^{\tau_i-1}\Big(u_i[t-(\tau_i-\Delta)] + D_i[t+\sigma_i+\Delta]\Big)\prod_{j=2}^{\Delta+1} g_i(j)$
8      $\mu_i[t] = \pi_i[t] + b_i\mu_{i+1}[t]$
9      send $\mu_i[t]$ downstream
10 end
/* Calculate outputs */
11 $u_{i-1}[t] = (1-\frac{\gamma_i}{q_i})\big(z_i[t] + u_i[t-\tau_i] + D_i[t+\sigma_i]\big) - a_i\delta_{i-1}[t] + c_i\mu_i[t] + d_i[t] - D_i[t+\sigma_i]$
12 $v_i[t] = -\frac{X_i(1)}{r_i}\Big(\delta_{i-1}[t] + (1-h_{i-1})\mu_i[t]\Big)$
```

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[1], 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 (1) are inherited by the optimal control

[1] `https://github.com/Martin-Heyden/cdc-letters-2021`

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

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

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

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

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 extended version. The algorithm consists of three serial sweeps. The first sweep starts at node 1 and calculates γ_i and ρ_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.

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.

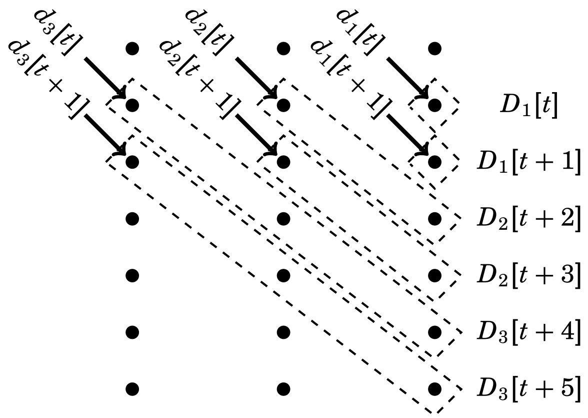


Figure 3. Illustration for the terms included in $D_i[t]$ for a three node graph with $\tau_1 = \tau_2 = 2$. The first row of dots corresponds to $z_3[t]$, $z_2[t]$, $z_1[t]$ and the second corresponds to $z_3[t+1]$, $z_2[t+1]$, $z_1[t+1]$, and so on. Due to lack of space only $d_i[t]$ and $d_i[t+1]$ 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.

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
      /* For $t+\sigma_i \leq s < t+\sigma_N+H$ */
6     if $d_i[s-\sigma_i]$ changed or $D_{i-1}[s]$ received then
7         send $D_i[s] = D_{i-1}[s] + d_i[s-\sigma_i]$ upstream
8     end
9 end
```

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 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.

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

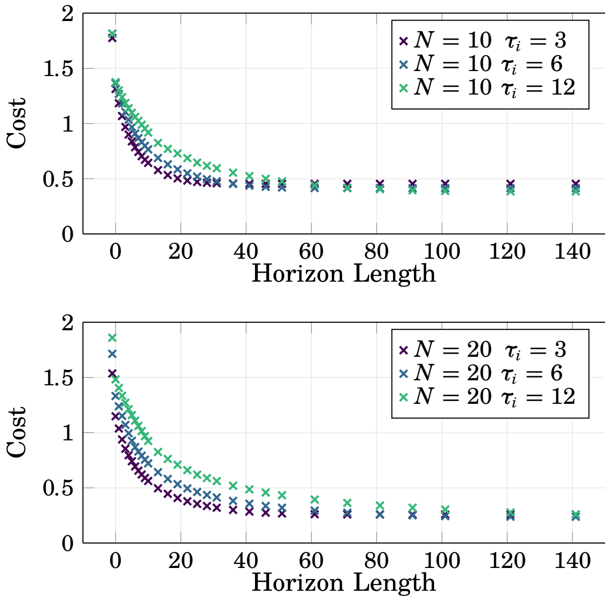


Figure 4. Simulations comparing the effect the planning horizon has on the performance. The data-points with highest cost for each configuration corresponds to not using the planned disturbances at all. While the the rest corresponds to using all planned disturbances announced up to H time units ahead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

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

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

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

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

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 γ_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

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

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 ρ_k is as given in Algorithm 1. So the calculations of γ_i and ρ_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

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

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

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

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

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

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 μ_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 δ as defined in Algorithm 2. This in turn requires P, h, and ϕ which were defined in Algorithm 1 and Φ 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 holds 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 γ_k be defined as in Algorithm 1. The optimization problem

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

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

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

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

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

It holds that

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

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

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

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

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

allows (8) to be rewritten as

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

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,

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

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

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

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

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

$$\underset{x}{\text{minimize}} \quad c_1(a+b+x)^2 + c_2(a+x)^2 + c_3x^2$$

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 exits 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

$$x = -\frac{1}{c_1+c_2+c_3}\left((c_1+c_2)a + c_1b\right)$$

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

$$c_1(a+b)^2 + c_2a^2 + 2[(c_1+c_2)a + c_1b)]x + (c_1+c_2+c_3)x^2.$$

Inserting the minimizer gives

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

The first term can be written as

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

Which gives that the objective function has the minimum value

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

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

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

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 γ_k, ρ_k, X_k, g_k, μ_k be defined as in Algorithms 1 and 2. Let for $1\le k\le N-1$ and $1\le\Delta\le\tau_k$, and for $k=N$ and $1\le\Delta\le H+2$

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

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

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

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 $1 \leq \Delta \leq \tau_k$

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

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

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

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

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

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

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

as

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

and

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

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 $1 \le \Delta \le \tau_i - 1$ if $k < N$ and $1 \le \Delta \le H + 1$ 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

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

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

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

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

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

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

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 $k \le N-1$

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

□

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 $1 \le i < \tau_k$. We will prove the lemma by showing that for

$1 \leq \Delta \leq \tau_k$

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

$$(1-P_k(\Delta,1))\delta_{k-1}[0] - \Big[h_{k-1}b_k(1-P(\Delta,1)) + P_k(\Delta,\Delta)g_{k+1}(1)\prod_{j=\Delta+1}^{\tau_k} g_k(j)\Big]\mu_{k+1}[0]$$

$$-\sum_{d=0}^{\tau_k-1} B_k[d]\Big[P_k(\Delta,\min(d+1,\Delta))\prod_{j=\Delta+1}^{d+1} g_k(j) + (1-P_k(\Delta,1))h_{k-1}\prod_{j=2}^{d+1} g_k(j)\Big] \tag{18}$$

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 Δ 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:

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

$$(1 - \frac{X_k(\Delta)}{\rho_k})\Big(m_{k-1}[0] + \sum_{i=1}^{k-1} \sum_{d=\sigma_i}^{\sigma_{i+1}-1} \bar{V}_i[d] + \sum_{d=\sigma_k}^{\sigma_k+\Delta-2} V_k[d]\Big)$$

$$-\frac{X_k(\Delta)}{\rho_k}\Big[\sum_{d=0}^{\tau_k-1} B_k[d]\prod_{j=\Delta+1}^{d+1} g_k(j) + \mu_{k+1}[0]g_{k+1}(1)\prod_{j=\Delta+1}^{\tau_k} g_k(j)\Big] \tag{19}$$

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

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

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

$$-\Big(1-\frac{X_k(\Delta)}{\rho_k}\Big)\Big(h_{k-1}b_k\big(1-P(\Delta-1,1)\big)+P_k(\Delta-1,\Delta-1)g_{k+1}(1)\prod_{j=\Delta}^{\tau_k} g_k[j]\Big)$$
$$-\frac{X_k(\Delta)}{\rho_k}g_{k+1}(1)\prod_{j=\Delta+1}^{\tau_k} g_k(j)$$
$$= -h_{k-1}b_k\big(1-P(\Delta,1)\big)-P_k(\Delta,\Delta)g_{k+1}(1)\prod_{j=\Delta+1}^{\tau_k} g_k(j),$$

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

$$-(1-\frac{X_k(\Delta)}{\rho_k})\Big[P_k(\Delta-1,\min(d+1,\Delta-1))\prod_{j=\Delta}^{d+1} g_k(j)+$$
$$(1-P_k(\Delta-1,1))h_{k-1}\prod_{j=2}^{d+1} g_k(j)\Big]-\frac{X_k(\Delta)}{\rho_k}\prod_{j=\Delta+1}^{d+1} g_k(j)$$
$$= -P_k(\Delta,\min(d+1,\Delta))\sum_{j=\Delta+1}^{d+1} g_k(j)-(1-P_k(\Delta,1))h_{k-1}\prod_{j=2}^{d+1} g_k(j).$$

Thus (18) holds for k and δ as well.

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

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

the RHS of (18) reduces to

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

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

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

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

$$\begin{aligned}&(1 - P_k(\tau_k, 1))\delta_{k-1}[0] + h_k\mu_{k+1}[0]\\ &\qquad - \sum_{i=d}^{\tau_k} B_k[d]\Big[P_k(\tau_k, d+1) + \big(1 - P_k(\tau_k, 1)h_{k-1}\big)\prod_{j=2}^{d+1} g_k(j)\Big]\end{aligned}$$

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

$$\begin{aligned}&(1 - P_k(\tau_k, 1))\delta_{k-1}[0] + h_k\mu_{k+1}[0]\\ &\qquad + \sum_{i=d}^{\tau_k} B_k[d]\Big[(1 - P_k(\tau_k, d+1)) - \big(1 - P_k(\tau_k, 1)h_{k-1}\big)\prod_{j=2}^{d+1} g_k(j)\Big].\end{aligned}$$

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

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

□

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

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

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

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

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

Ahuja, R. K., T. L. Magnanti, J. B. Orlin, and M. Reddy (1995). "Applications of network optimization". *Handbooks in Operations Research and Management Science* **7**, pp. 1–83.

Cantoni, M., E. Weyer, Y. Li, S. K. Ooi, I. Mareels, and M. Ryan (2007). "Control of large-scale irrigation networks". *Proceedings of the IEEE* **95**:1, pp. 75–91.

Heyden, M., R. Pates, and A. Rantzer (2018). "A structured linear quadratic controller for transportation problems". In: *2018 European Control Conference (ECC)*. IEEE, pp. 1654–1659.

Kalman, R. E. et al. (1960). "Contributions to the theory of optimal control". *Bol. soc. mat. mexicana* **5**:2, pp. 102–119.

Lessard, L. and S. Lall (2011). "Quadratic invariance is necessary and sufficient for convexity". In: *Proceedings of the 2011 American Control Conference*, pp. 5360–5362. DOI: 10.1109/ACC.2011.5990928.

Lin, F., M. Fardad, and M. R. Jovanović (2013). "Design of optimal sparse feedback gains via the alternating direction method of multipliers". *IEEE Transactions on Automatic Control* **58**:9, pp. 2426–2431. DOI: 10.1109/TAC.2013.2257618.

Rotkowitz, M. and S. Lall (2006). "A characterization of convex problems in decentralized control". *IEEE Transactions on Automatic Control* **51**:2, pp. 274–286. ISSN: 0018-9286. DOI: 10.1109/TAC.2005.860365.

Subramanian, K., J. B. Rawlings, C. T. Maravelias, J. Flores-Cerrillo, and L. Megan (2013). "Integration of control theory and scheduling methods for supply chain management". *Computers & Chemical Engineering* **51**, pp. 4–20.

Wang, Y., N. Matni, and J. C. Doyle (2018). "Separable and localized system-level synthesis for large-scale systems". *IEEE Transactions on Automatic Control* **63**:12, pp. 4234–4249. ISSN: 0018-9286. DOI: 10.1109/TAC.2018.2819246.

Paper III

A Structured Optimal Controller for Irrigation Networks

Martin Heyden Richard Pates Anders Rantzer

Abstract

In this paper, we apply an optimal LQ controller, which has an inherent structure that allows for a distributed implementation, to an irrigation network. The network consists of a water reservoir and connected water canals. The goal is to keep the levels close to the set-points when farmers take out water. The LQ controller is designed using a first-order approximation of the canal dynamics, while the simulation model used for evaluation uses third-order canal dynamics. The performance is compared to a P controller and an LQ controller designed using the third-order canal dynamics. The structured controller outperforms the P controller and is close to the theoretical optimum given by the third-order LQ controller for disturbance rejection.

Submitted to the European Control Conference (ECC) 2022.

1. Introduction

A large share of the available fresh water in the world is used for irrigation networks that supply water for food production. These networks are often only powered by gravity, and thus the water levels must be sufficiently high to enable transportation of the water. As a consequence, irrigation networks are often operated conservatively, as the farmers must be able to get water when they need it [Weyer, 2008]. The efficiency of irrigation networks was estimated to be around 50 %, with half of the losses coming from large-scale distribution losses, which occur before the water reaches the farms [Mareels et al., 2005]. Improving the performance of these networks could lead to large savings in water that could allow for higher food production.

In the research literature, there are two dominant paths for controlling irrigation networks, namely local PI control [Weyer, 2002; Litrico et al., 2003; Lozano et al., 2010] and centralized LQ or MPC control [Weyer, 2003; Neshastehriz et al., 2014]. Other approaches include distributed LQ [Lemos and Pinto, 2012] and distributed H_∞-control [Cantoni et al., 2007]. These distributed approaches typically use multiple iterations of communication for each sample time. An alternative is a non-iterative predictive controller [Negenborn et al., 2009]. Here the inputs are calculated sequentially by a communication sweep through the network. This implementation structure is similar to the one used in this paper.

In this work, the structured optimal LQ controller with a distributed implementation studied in our previous paper [Heyden et al., 2021] is applied to a model for irrigation networks. This controller combines the simple and efficient implementation of distributed methods with the performance of centralized controllers. This LQ controller is synthesized using a model with first-order pool dynamics but evaluated on a model with third order-pool dynamics found in the literature. This is not a design choice as the structured LQ controller can only be synthesized on first-order dynamics. However, such first-order models are easier to identify. Furthermore, the first-order pool dynamics describe the system well on slow time scales, and controllers frequently *designed* using them, see for example [Schuurmans et al., 1999; Litrico and Fromion, 2005]. However, it is important to not excite the wave dynamics. In this paper this is achieved by applying a low-pass filter at each gate. This means that the controller can be designed based on a first-order model in conjunction with knowledge of the dominant wave frequency.

Our contributions are twofold. Firstly, we show how to apply the structured controller in [Heyden et al., 2021] to irrigation models based on third-order canal dynamics. Secondly, we compare the performance to a simple P controller and an LQ controller with full state knowledge synthesized using the third-order canal dynamics. The P controller gives a baseline for easily achievable performance while the LQ controller gives the best possible performance. For disturbance rejection of low-pass filtered disturbances, the structured controller is very close to the best performance and outperforms the P controller. For a change in set-points, the structured controller is in-between the maximum performance and the performance of the P controller.

2. Problem Description

Irrigation networks consist of a set of canals (often called pools), gates, and off-takes. The canals are connected with gates that allow for the flow between the canals to be regulated. The off-takes, often located at the gates, allow water to be taken from the canal to a farmer. The gates and off-takes are typically only powered by gravity, and thus the levels at the gates and off-takes must be sufficiently high to allow the water to be transported. Many irrigation networks are located in rural areas, where both communication and computational capabilities are limited.

When controlling irrigation networks, there are typically several objectives that are considered [Weyer, 2008]. Firstly, to keep the canal levels close to the set-points to allow the off-takes to be used. Secondly, minimize gate movement to reduce wear and tear and minimize energy consumption. And finally, minimize the flow over the last gate to reduce water wastage. At the same time, the controller must handle the disturbances due to the off-takes.

To model a string of N pools, we assume we measure the levels $y_1, y_2, \dots, y_N$ relative to a nominal value at the end of each pool. Each pool i is affected by an inflow u_i, an outflow u_{i-1}, and a disturbance d_i. The flows u_i between two pools are also relative to a nominal flow. For a schematic of the system, see Figure 1. The disturbance d_i is the off-take to the farm(s) at gate i. We assume that these disturbances are planned, that is the controller knows, but cannot change, the value of $d_i[t]$. This means that the farmers must tell the irrigation network controller in advance that they will take out water. To be consistent with our previous work, we denote the most upstream canal as canal N. This canal has inflow from a reservoir with a capacity so large it can be assumed to be infinite for the purposes of regulation. Finally, we let the flow over the last gate be fixed. Fixing the flow over the last gate is possible if the level is close to the set-point and it is highly desirable to do so since the flow over the last gate is wasted [Cantoni et al., 2007].

Next, we will describe the simulation models used, including the dynamics for each pool in the system. The section is then concluded with a presentation of the performance criterion used.

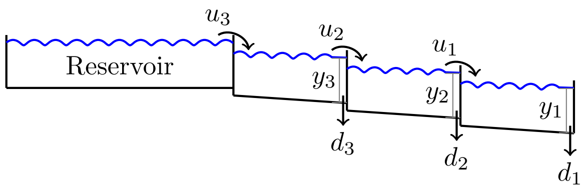


Figure 1. Graphical illustration of the problem considered. At the top of the network is a reservoir with infinite capacity. Each pool i has an inflow u_i, an outflow u_{i-1} (except pool 1) and a disturbance d_i which takes water out of the pool. The goal is to regulate the water level y_i at the gates.

2.1 Network Model

In this paper models for two different pools are used. For evaluation, we use third-order models found using system identification on pools 9 and 10 in the Haughton main river. See [Ooi et al., 2005] for the origin of the parameters, where it was also shown that the models are as accurate as a PDE approach using the St-Venant equations. We use the two pool models to construct networks containing multiple pools. The first network type is a non-homogeneous network which alternates between the first and the second pool model. The second network type is a homogeneous network using only the first pool model. This network is suitable to clearly see the effect of, for example, changing the size of the network.

Two modifications to the original pool models are made. Firstly, in [Ooi et al., 2005] the flow over a gate i is on the form $(y_i - p_i)^{3/2}$ where p is the position of the gate relative to the nominal water level. This non-linearity can be canceled out (see for example [Cantoni et al., 2007]) by letting $u_i = (y_i - p_i)^{3/2}$. Secondly, we expand the pool models with a disturbance corresponding to an off-take. The assumption is that the off-take takes water out of the pool in the same way as the outflow. The modified pool dynamics are on the form

$$\begin{aligned} y_i[t+1] &= b_{i,1}u_i[t-\tau_i] - b_{i,2}u_i[t-\tau_i-1] + b_{i,3}u_i[t-\tau_i-2] \\ &- c_{i,1}(u_{i-1}[t]-d_i[t]) + c_{i,2}(u_{i-1}[t-1]-d_i[t-1]) - c_{i,3}(u_{i-1}[t-2]-d_i[t-2]) \\ &+ y_i[t] + \alpha_{i,1}(y_i[t]-2y_i[t-1]+y_i[t-2]) + \alpha_{i,2}(y_i[t]-y_i[t-1]). \end{aligned} \tag{1}$$

The sample time is one minute and the parameters for the two pools can be found in Table 1.

2.2 Performance Evaluation

The performance of the system is measured as the deviation from the nominal values for the levels y_i and flows u_i, and how much the input

Pool	Order	$b_{i,1}$	$b_{i,2}$	$b_{i,3}$	$c_{i,1}$	$c_{i,2}$	$c_{i,3}$	$\alpha_{i,1}$	$\alpha_{i,2}$	τ_i
1	1	0.069			0.063					3
1	3	0.137	0.155	0.053	0.190	0.333	0.175	0.978	0.468	3
2	1	0.0213			0.0156					14
2	3	0.134	0.244	0.114	0.101	0.185	0.087	0.314	0.814	16

Table 1. The parameters for the first and third-order models. For the first-order model we let $b_i = b_{i,1}$ and $c_i = c_{i,1}$.

changes, that is $(u_i[t+1] - u_i[t])^2$. This is done by considering the cost

$$\sum_{t=0}^{\infty}\sum_{i=1}^{N}\Big(q_i y_i[t]^2 + r_i u_i[t]^2 + \rho_i(u_i[t+1] - u_i[t])^2\Big). \tag{2}$$

The reason for penalizing $(u_i[t+1]-u_i[t])^2$ is twofold. Firstly it penalizes the wear and tear of the actuator. Secondly, it reduces the energy consumption. The amount of energy available can be limited, for example when the only available energy comes from solar power. The structured controller can only be used when $\rho_i = 0$ for all inputs and $r_i = 0$ for all inputs except for $i = N$, which is the flow out from the reservoir into pool N. The effect of these limitations will be explored in the simulation section.

3. A Structured Optimal Controller for a First-Order System

As previously discussed, a controller for an irrigation network must handle the disturbances from the off-takes. If the network is in a rural area there might also be a limit on the available communication capabilities and computational power. Due to this, a promising candidate for control of irrigation networks is the structured optimal LQ controller with a distributed implementation studied in our previous paper [Heyden et al., 2021]. We will in this section present a slight variation of that structured controller, designed for a network model where the pools have the following first-order dynamics,

$$y_i[t+1] = y_i[t] + b_i u_i[t - \tau_i - \bar{\tau}] - c_i(u_{i-1}[t - \bar{\tau}] - d_i[t - \bar{\tau}]). \tag{3}$$

The dynamics in (3) is a first-order approximation of the third-order dynamics in (1) when all the inputs and planned disturbances are low-pass filtered. The low-pass filter, which is used to suppress the wave dynamics, is the source of the additional delay $\bar{\tau}$. The low-pass filter and the model in (3) will be discussed further in the next section.

Before that, we will present the optimal controller for the first-order pool dynamics in (3). That is we study the following Linear Quadratic control problem

$$\begin{aligned} \underset{y,u}{\text{minimize}} \quad & \text{cost in (2)} \\ \text{subject to} \quad & \text{dynamics in (3)} \\ & y[0] \text{ and } d_i[t] \text{ given.} \end{aligned} \tag{4}$$

The following Theorem shows that two algorithms can be used to calculate the necessary parameters for, and the implementation of, the optimal LQ controller for the problem in (4). Both algorithms are implemented through a serial sweep using local communication and scalar computations. This means that the optimal LQ controller can be implemented in a distributed way.

THEOREM 1
Assume that $r_i = 0$ for $i \neq N$, $\rho_i = 0$ for all i, and that $d_i[s] = 0$ for all $s > t+H$. Let $\sigma_i = \sum_{j=1}^{i-1} \tau_j$. Then the minimizing $u_i[t]$ for the problem in (4) is given by running Algorithm 2 with the parameters from Algorithm 1. □

Proof The result is a minor extension of the results in [Heyden et al., 2021]. For completeness, the proof is given in the appendix. □

Algorithm 1: Computation of control parameters.

```
Input: $q_i$, $r$, $b_i$, $c_i$
Output: $\gamma_i$, $\hat{b}_i$, $g$
1  send $\gamma_1 = q_1$ and $\hat{b}_1 = b_1$ to upstream neighbor
2  for gate $i$ = 2:N do // Sweep through the pools
3      $\hat{b}_i = b_i/c_i \cdot \hat{b}_{i-1}$
4      $q_i = c_i^2/\hat{b}_{i-1}^2 \cdot q_i$
5      $\gamma_i = \frac{\gamma_{i-1}q_i}{\gamma_{i-1}+q_i}$
6      send $\gamma_i$ and $\hat{b}_i$ to upstream neighbor
7  end
8  $r = r/\hat{b}_N^2$                                   // Gate N
9  $X = -\gamma_N/2 + \sqrt{\gamma_N r + \frac{\gamma_N^2}{4}}$          // Gate N
10 $g = \frac{X}{X+\gamma_N}$                              // Gate N
```

For both algorithms all measurements and calculations are made at the gates. Gate i is at the end of pool i, and is responsible for deciding u_{i-1}.

Algorithm 1 can be used to calculate the parameters needed to implement the feedback law. The algorithm consists of a sweep through the graph. On line 3-4 the parameters b and q_i are re-scaled, corresponding to

Algorithm 2: Implementation of control law.

```
Input: $y_i[t]$, new $d_i[s]$, output from Algorithm 1
Output: $q_i$,$\gamma_i$,$\hat{b}_i$

/* old $u_i[t-s]$ and $D_i[t+s]$ are kept in memory */
1  $y_i[t] = \frac{\hat{b}_{i-1}}{c_i} y_i[t]$                    // Done in parallel for $i \geq 2$
2  $d_1[s] = c_1 d_1[s], \quad d_i[s] = b_{i-1} d_i[s] \; i \geq 2$
   /* Update unchanged $D_i[t_0 + \sigma_i]$ */
3  for gate $i = N{:}2$ do // Done in parallel, $\mathcal{O}(1)$
4      Send $D_{i-1}[t + \sigma_{i-1} + \tau_{i-1}] = D_i[t + \sigma_i] - d_i[t]$ downstream.
5  end
   // Start sweep through graph
6  $m_1[t] = z_1[t] + \sum_{s=1}^{\tau_i+\bar{\tau}} u_1[t-s] \sum_{s=1}^{\bar{\tau}} d_1[t-s] + \sum_{s=0}^{\tau_i} D_1[t + \sigma_i + s]$
7  send $m_1[t]$ upstream
8  for gate $i = 2{:}N$ do
       /* For $t + \sigma_i \leq s < t + \sigma_N + H$ */
9      if $d_i[s - \sigma_i]$ changed or $D_{i-1}[s]$ received then
10         send $D_i[s] = D_{i-1}[s] + d_i[s - \sigma_i]$ upstream
11     end
12     $p_i[t] = y_i[t] + \sum_{s=\tau_i}^{\tau_i+\bar{\tau}} u_i[t-s] - \sum_{s=1}^{\bar{\tau}} u_{i-1}[t-s] + \sum_{s=0}^{\bar{\tau}} d_i[t-s]$
13     $m_i[t] = m_{i-1}[t] + p_i[t] + \sum_{s=1}^{\tau_i-1} u_i[t-s] + \sum_{s=1}^{\tau_i} D_i[t + \sigma_i + s]$
14     send $m_i[t]$ upstream
15 end
16 $u_{i-1}[t] = (1 - \gamma_i/q_i) p_i[t] - \gamma_i/q_i \cdot m_{i-1}[t], \quad 2 \leq i \leq N$
17 $u_N[t] = -\frac{X}{r}\left[m_N + \sum_{s=\tau_N+1}^{H} D_N[t + \sigma_N + s] \prod_{j=2}^{d+1} g\right]$
18 send $u_i$ downstream                                  // Done in parallel for all gates
19 $u_{i-1}[t] = 1/\hat{b}_{i-1} \cdot u_{i-1}$                     // Done in parallel for all gates
```

a transforming the dynamics in (3) to the form in [Heyden et al., 2021]. On line 5 the parameter γ_i is calculated recursively. Finally, when the sweep is completed, the parameters needed to calculate the optimal outflow form the reservoir are calculated on line 8-10, including another scaling on line 8.

Algorithm 2 is used for the on-line implementation of the optimal controller. The algorithm assumes that each gate stores its incoming and outgoing flow and the disturbance sums $D_i[s]$, defined as

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

Line 1-2 is a change of variables. On line 3-5 the $D_i[s]$ for which no new disturbances d_i are announced are updated. Only one D_i for each gate needs to be sent downstream, as the rest of the needed D_i were already known in the gate form the previous time point. This can be done in parallel for all gates.

Next a serial sweep starts at the most downstream pool (pool one) and goes through the graph in the upstream direction. The sweep accomplishes two things. Firstly, on lines 9-11 all D_i for which a new disturbance d_j was announced are updated. When the controller is initialized all non zero D_i need to be updated this way. Secondly, $m_i[t]$ and $p_i[t]$ which are used for the calculation of u_i are calculated on lines 12-14. The variable $p_i[t]$, which is the predicted level in pool i at time $t + \bar{\tau} + 1$ when the outflow $u_{i-1} = 0$, is calculated on line 12. The calculation of p only requires only local and neighboring information, where the incoming flow to pool i from gate i must be known. For the calculation of $m_i[t]$ on line 13, which is the total level in the first i pools, only local information, $p_i[t]$, and the previous $m_{i-1}[t]$ is needed. Finally $m_i[t]$ is sent upstream on line 14.

After the sweep is completed all the inputs can be calculated on lines 16-17, relying only on p_i and m_{i-1} (and D_N for u_N). The input $u_i[t]$ is then sent downstream to gate i on line 18, as it is needed for the calculation of p_i and m_i in future time-steps. Finally, all inputs are re-scaled on line 19. A sketch of the information flow for the implementation is found in Figure 2

4. Applying the Structured Controller to an Irrigation Network

In this section, we will go through the steps taken to apply the controller presented in Section 3 to the simulation models with third-order pool dynamics presented in Section 2. While the previous section had strong theoretical motivation, this section will be more practical. The steps taken here are certainly not the only way to apply the structured controller just presented to the irrigation network model with third order pool dynamics. However, they get the job done in a fairly transparent way.

The bode magnitude diagrams in Figure 3 indicate that the third-order models are well approximated by a first-order model for low frequencies. However, for higher frequencies, there is a peak that must be taken care of. This is done by applying low-pass filters to the inputs and the disturbances. Next, a first-order approximation of the system, including the low-pass filter, is found. Finally, a Kalman filter is used to handle the difference between the first and third-order models. It ensures that the control law acts on the best approximation of the state of the first-order model, based on measurements taken on the third-order model.

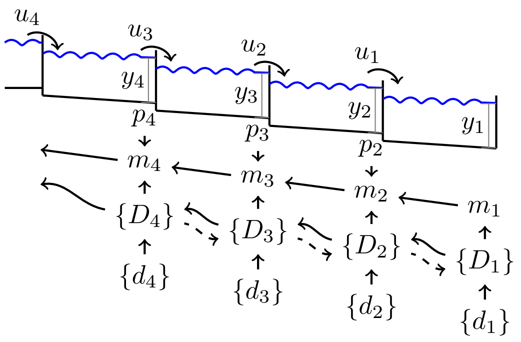


Figure 2. Illustration of the communication structure for the structured controller with 4 pools. The value for m_i is calculated by a sweep through the graph, requiring the downstream m_{i-1}, the local p_i and a local set of disturbances D_i (line 12-14 in Algorithm 2). The disturbances D can be calculated in two ways. If any new underlying d_j is announced, then the corresponding D_i must be calculated through a similar sweep to m, going through the graph upstream (line 9-11 in Algorithm 2), illustrated by solid arrows . However, if there are no new planned disturbance d_j, then the aggregate disturbances D_i can be updated from the upstream gate (line 3-4 in Algorithm 2), illustrated by dashed arrows. Note that for the outflow from the reservoir u_4, m_4 and D_4 will be used and hence they are sent to that gate as indicated by the arrows.

4.1 Low-pass Filter

The third-order system has a poorly damped node, which introduces two problems. Firstly, one wants to avoid introducing waves into the pools. And secondly, the structured LQ-controller must be designed using a first-order model, which can not describe the frequency peak.

One alternative to remedy both issues is to design an inner controller at the gate which takes a flow reference and then controls the flow. It should be designed so that the transfer function from the flow reference to the level in the pool would be close to first-order. This would require a detailed model of the pools on both sides of the gate.

We instead choose to add a low-pass filter to each input and each planned disturbance. Filtering the disturbance is natural since waves should be avoided both in the pools and in the off-takes to the farmers. However, additional consideration might need to be taken to make sure that the farmers get the amount of water that they ordered and that the delivery time is not delayed too much by the low pass filter. This could be accomplished by, for example, modifying the farmer's order before applying the low-pass filter.

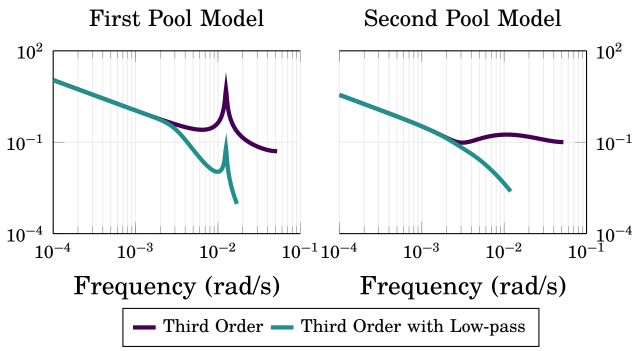


Figure 3. The bode magnitude plot for the transfer function from inflow u_i to level y_i for the two models in (1), with and without low-pass filter. It can be seen that for low frequencies the third-order pool model can be well described by a first-order system, but for higher frequencies, there is a resonance peak. The low-pass filter manages to suppress this peak.

The low pass filter at each gate must be designed based on its two neighboring pools so that no waves are induced in either pool. For simplicity, we use the same low pass filter for all gates, which then must suppress the wave dynamics in both pools. A Butterworth filter is used for the low-pass filter, as it has minimal effect on the pass-band. The Matlab command `butter` is used for the design and the final design is a third-order filter with a cut-off frequency $3 \cdot 10^{-3}$ rad/sec. The resulting bode magnitude plot before and after the low-pass filtering can be found in Figure 3. The third-order models are used both in the design and the evaluation of the low-pass filter. However, if detailed models were not available, it would still be possible to design the low-pass filter based only on knowledge of the dominant wave frequency and evaluate it using open-loop tests in the canals.

4.2 First-order approximation

First-order models on the form

$$y_i[t+1] = y_i[t] + b_i u_i[t-\tau_i] - c_i(u_{i-1}[t] - d_i[t]), \tag{5}$$

where $b_i > 0$ and $c_i > 0$, have been shown to describe the water level in a pool well on slow timescales [Cantoni et al., 2007]. Just as in the the third-order model, in the above $u_i = (y_i - p_i)^{3/2}$, and y_i denotes the water level in the ith pool. Parameters for a suitable first-order description of the same two pools from the Haughton main river were given in [Weyer, 2003]. However, upon closer examination there was a large difference between the first and third-order model in terms of their DC gains for the inflow into the second pool. To counteract this, we modified the first-order model, where

$b_{i,1}$ was increased by a factor of 1.5. In practice a more principled approach should be used to construct a suitable reduced order model, however it is reassuring that working in this ad-hoc manner still resulted in a good enough model for conducting synthesis.

To handle the addition of the low-pass filter we propose a simple update to (5) as already given in (3)

$$y_i[t+1] = y_i[t] + b_i u_i[t - \tau_i - \bar{\tau}] - c_i(u_{i-1}[t-\bar{\tau}] - d_i[t-\bar{\tau}]).$$

The additional delay $\bar{\tau}$, which is the same for all pools, can intuitively be motivated as an approximation of the effect of the low pass filter. We also let τ_i be different from the ones in [Ooi et al., 2005], as it was noted that this had a positive effect on the performance. The parameters b_i and c_i are unchanged.

The parameters τ_i and $\bar{\tau}$ are chosen as follows. First the optimal $\bar{\tau}$ is found by simulating the response for both pools to an outflow u_{i-1} corresponding to a constant positive input, followed by a constant zero input, followed by a negative input. That is

$$u[t] = \begin{cases} 1 & t < t_1 \\ 0 & t_1 \leq t < t_2 \\ -1 & t_2 \leq t < t_3 \\ 0 & t \geq t_3. \end{cases} \tag{6}$$

The idea is that this describes when a pool is emptied and then filled. A similar open-loop experiment could easily be conducted in an irrigation network. The value for $\bar{\tau}$ that minimizes the least square error (normalized for each pool) is chosen. This is an integer optimization problem, but the number of reasonable values are limited so we can expect to find the optimal value. The resulting value for $\bar{\tau}$ is 10. Next, the optimal τ_i for each pool is found by minimizing the least square error when the inflow u_i is as in (6). The resulting value for the first pool model is $\tau_i = 2$ and for the second pool model $\tau_i = 15$. The resulting system responses when the inflow is zero and the outflow is as in (6) are plotted in Figure 4, where it can be seen that the first-order system gives a good approximation of the low-pass filtered third-order system.

4.3 Kalman Filter

The first-order approximation describes the behavior on slow time scales of the third-order model with a low-pass filter. However, there are still some differences. For example, the step response for the first-order model starts slower but finishes faster. These differences can be handled by introducing a Kalman filter, so that in the short term the controller trusts the first-order

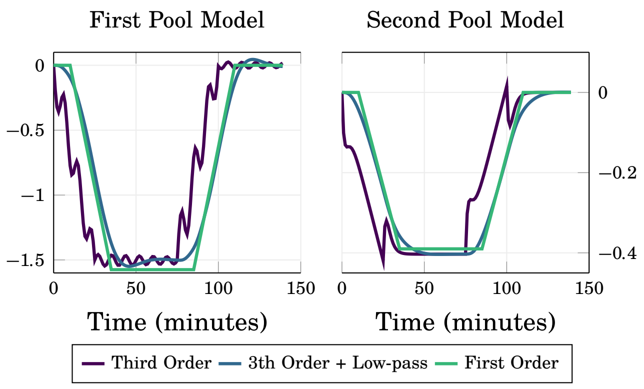


Figure 4. Time response from the outflow (u_{i-1}) for the third-order model in (1), third-order model with low pass, and first-order model with additional delay in (3). It can be seen that the low pass filter suppresses most oscillations, and that the first-order pool model captures the behavior of the third-order pool model with a low-pass filter well.

model, but in the long term it still utilizes the measurements from the third-order model.

For the Kalman filter design we consider the same dynamics used in the controller design, but with added (unknown) state disturbance $v_i[t]$ and measurement disturbance $w_i[t]$,

$$\begin{aligned} x_i[t+1] &= x_i[t] + b_i u_i[t-\tau_i-\bar{\tau}] - c_i(u_{i-1}[t-\bar{\tau}] - d_i[t-\bar{\tau}]) + w_i[t] \\ y_i[t] &= x_i[t] + v_i[t]. \end{aligned}$$

Changing the relationship between the modeled variance of w_i and v_i allows balancing how much the Kalman filter trusts the measurements compared to the first-order model.

The Kalman filter is updated using the following scalar dynamics which can be implemented locally at each gate,

$$\begin{aligned} \hat{y}_{t|t} &= \hat{y}_{t|t-1} + L(y_t - \hat{y}_{t|t-1}) \\ \hat{y}_{t+1|t} &= \hat{y}_{t|t} + b_i u_i[t-\tau_i-\bar{\tau}] - c_i(u_{i-1}[t-\bar{\tau}] - d_i[t-\bar{\tau}]). \end{aligned}$$

In the above, L is the solution to the scalar Riccati equation,

$$L = L - L^2/(L+R_2) + R_1,$$

where R_1 is the variance of w_i and R_2 is the variance of v_i. For the simulations we use $R_1 = 1$ and $R_2 = 100$. The a priori estimate $x_{t|t-1}$ is used in the calculation of the inputs at time t. This gives a minute of time for propagating information through the string graph.

5. Comparison Controllers

We design two additional controllers to use for comparisons with the structured controller. Firstly, we design a LQ controller using the third-order pool model in (1) to get the best possible performance in terms of the performance criterion in (2). Secondly, we design a simple P controller that will give a baseline in terms of easily achievable performance.

To get a fair comparison, the disturbance will be low pass filtered for these controllers as well. Furthermore, as the structured controller does not have integral action but instead relies on feed-forward to reject load disturbances, we let the standard LQ controller and P controller also use feed-forward and have no integral action.

5.1 Third-Order LQ

To get a baseline of the best possible performance we consider an LQ controller synthesized directly on the third-order dynamics. This controller is not meant to be implementable in practice so we let the controller have access to full state information.

Consider a state space representation for the transfer function in (1) on the form

$$\begin{aligned} x_i[t+1] &= A_i x_i[t] + B_i(1)u_i[t] + B_i(2)u_{i-1}[t] \\ y_i[t] &= C_i x_i[t]. \end{aligned}$$

Then using the dynamics $A = \mathrm{diag}(A_1, A_2, \dots, A_N)$, $C = \mathrm{diag}(C_1, C_2, \dots, C_N)$,

$$B = \begin{bmatrix} B_1(1) & 0 & & \\ B_2(2) & B_2(1) & 0 & \\ \ddots & \ddots & \ddots & \ddots \\ & 0 & B_N(2) & B_N(1) \end{bmatrix}, \quad v[t] = \begin{bmatrix} B_1(2)d_1[t] \\ B_2(2)d_2[t] \\ \vdots \\ B_N(2)d_N[t] \end{bmatrix},$$

the dynamics of the water levels $y_i[t]$ for a network with N pools can be described by

$$\begin{aligned} x[t+1] &= Ax[t] + Bu[t] + v[t] \\ y[t] &= Cx[t]. \end{aligned}$$

Let $Q = \mathrm{diag}(q_1 C_1^T C_1,\ q_2 C_2^T C_2,\ \dots,\ q_N C_N^T C_N)$, then the cost due to the pool levels can be expressed as $\sum_{i=1}^N q_i y_i[t]^2 = x[t]^T Q x[t]$. Now, let S be the solution to the Riccati equation

$$S = A^T SA - A^T SB(B^T SB + R)^{-1} B^T SA + Q,$$

and define

$$\begin{aligned} K &= -(B^T SB + R)^{-1} B^T SA \\ K_v &= -(B^T SB + R)^{-1} B^T \\ \Pi[t] &= (A + BK)^T \Pi[t+1] + Sv[t], \quad \Pi[H+1] = 0. \end{aligned} \tag{7}$$

Then the optimal input $u[t]$ is given by

$$u[t] = Kx[t] + K_v\Pi[t]. \tag{8}$$

A derivation of the optimal feed-forward for the known disturbance can be found in the extended version of this paper.

To allow for a penalty on the change in input $(u_i[t] - u_i[t-1])^2$ we introduced new states, corresponding to $u_i[t-1]$ and $(u_i[t] - u_i[t-1])^2$. Additional states could be introduced to further improve the performance of the LQ controller, such as penalizing a high pass filtered version of the output to reduce the oscillations in the system, see for example [Weyer, 2008]. As this LQ controller is only used to get the maximum performance, we consider only the aspects captured by the performance measure.

5.2 P-Controller

For the P-controller we consider a configuration where the controller at gate i is designed to control the water level at the end of pool $i-1$, which is the level just before the downstream gate $i-1$. This setup is often called distant downstream control [Weyer, 2002]. The low-pass filter that was used to filter the inputs for the structured controller is also used for the P-controller. We use feed-forward both on the outflow from the downstream gate $i-1$ and on the off-take at the downstream gate. Thus the controller is on the form

$$u_i[t] = -k_i y_i[t] + k_{ff}\frac{c_i}{b_i}(u_{i-1}[t-1] - d_i[t+\tau_i]).$$

The fraction c_i/b_i is used to account for the different coefficients in the inflow and outflow. The feed-forward on the downstream input $u_{i-1}[t-1]$ is delayed as otherwise $u_i[t]$ would depend on all $u_j[t]$ for $j < i$.

We use the following values for the controller parameters,

$$k_i = \frac{\pi}{2(\tau_i + \bar{\tau})b_i}\frac{1}{4}, \qquad k_{ff} = 1. \tag{9}$$

The choice of k_i was partially found by hand-tuning, but can also be theoretically motivated. For the design of the P-controller the outflow from the downstream gate can be modeled as a disturbance. Using the model in (3) gives the following continuous time dynamics

$$\dot{y}_i = b_i u_i(t - \tau_i - \bar{\tau}) + d_i(t) \Rightarrow G(s) = \frac{b_i}{s}e^{-(\tau_i+\bar{\tau})s}.$$

Ignoring the feed-forward, the controller is on the form $u_i(t) = -k_i y_i(t)$, which gives the loop transfer function

$$\frac{k_i b_i}{s}e^{-(\tau_i+\bar{\tau})s}.$$

Picking k_i as in (9), the time responses for all the pools will have the same shape, but with different time constants. Considering the gain margin

$$\frac{\pi}{2(\tau_i + \bar{\tau})b_i k_i}$$

and the phase margin

$$\frac{\pi}{2} - (\tau_i + \bar{\tau})b_i k_i$$

shows that the choice of k_i gives a gain margin of 4 and a phase margin of 67.5 degrees.

6. Simulations

In this section we use the two networks discussed in Section 2 to compare the performance of the three different controllers. In the first part we consider cost functions that satisfies the assumption for the structured controller, that is $r_i = 0,\ i \neq N$ and $\rho_i = 0$. We explore both the time response for the different controllers, and study how they scale with the size of the network. Next we explore the limitations for the structured controller by comparing how well one can balance the deviations in inputs and in the levels. All code used for the simulation is available on GitHub[1].

We use the cost function parameters $q_i = 1$, $r_N = 0.3$, $r_i = 0,\ i \neq N$ and $\rho_i = 0$. In Figure 5 the time responses for the three different controllers are depicted. Canal one, three, and five are modeled as the first pool and canal two and four are modeled as the second pool. The initial condition is $[5, 0, 0, 0, -5]$, corresponding to a change in set-point resulting in water needing to be moved through the graph. Then there is a disturbance in pool one between time 250 and 450, corresponding to a change in level of 1 unit/minute. It can be seen that the third-order LQ-controller is very aggressive for the step response and this step response would neither be wanted, nor implementable at the gates.

Next we consider how the performance of the different controllers scales with the size of the network. From now on, all pools have the dynamics in the first pool model. This allows us to clearly see the effect of the varied parameter. Also, to limit the effect of the design decision for the P-controller, we ran a set of different controllers with k_i as a factor of [0.25, 0.5, 1, 1.5, 2] of the nominal value, and picked the best performance for each configuration. The left graph in Figure 6 depicts how the change in the number of pools affect the cost when the disturbance is kept in pool $N-1$, which is the second pool counting from the reservoir. For the right graph,

[1] `https://github.com/Martin-Heyden/ECC-irrigation-network`

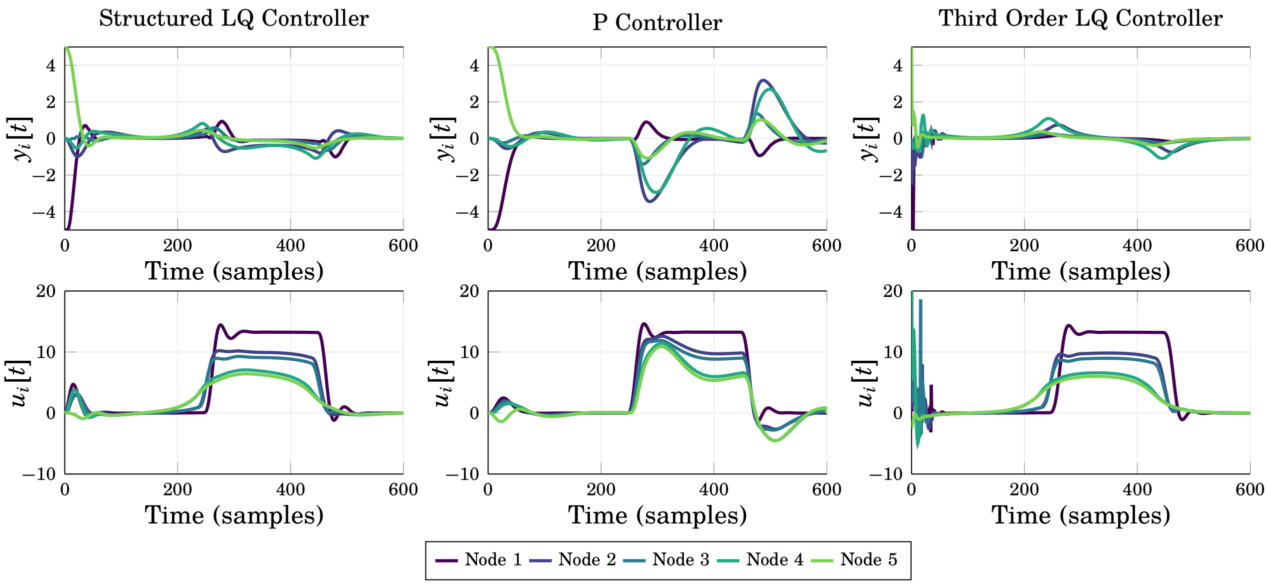


Figure 5. Illustration of the time response for the different controllers. The systems starts with initial condition $[-5, 0, 0, 0, 5]$ corresponding to a step change. Between time 250 and 450 there is a disturbance in pool one with a discharge rate that gives a change of one unit per minute to the level.

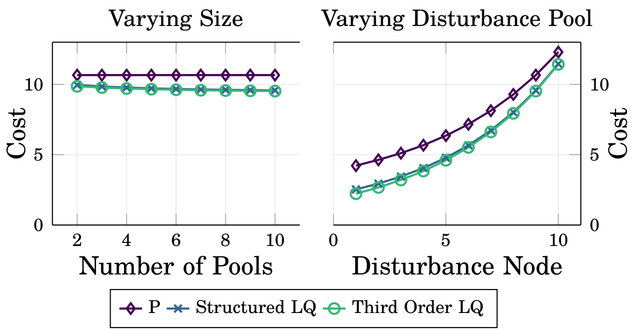


Figure 6. Comparison of the performance for the different controllers when there is a planned disturbance in the network. In the left figure the disturbance is always in pool $N-1$ and N is varying. In the right figure N is fixed to 10 and the pool with the disturbance is varied.

the number of pools in the network is fixed to 10, and the pool which the disturbance acts upon is varied. For both cases the disturbance is acting between time 200 and 400. We can see that the two LQ-controllers improves performance slightly when the graph size increases, while the P-controller does not utilize the additional pools. A bigger difference is seen when the disturbance pool is varied. Here it can be noted that there is an increase in performance for all the controllers when the disturbance pool is far away from the reservoir. For the P-controller this is partly due to the controller only using the pools upstream of the disturbance. In general, the reason that the performance is increased when the disturbance is further downstream could be that it is more efficient when the transportation from the reservoir and the other pools are all in the same direction. We also note that the performance of the two LQ-controllers are almost identical for both cases. This is likely due to the fact that the disturbances are low-pass filtered, and thus the low-pass filtering of the inputs for the structured controllers does not limit the performance much.

Indeed, in Figure 7 we consider the performance when there is a change in set-points, requiring water to be moved from the N'th pool (the pool after the reservoir) to the first pool (the most downstream one). Unsurprisingly, it can be seen that the third-order LQ controller outperforms the structured controller, as it can directly cancel out the waves. On the other hand, the time response in Figure 5 indicated that the third-order LQ controller needs to be made less aggressive, and the performance of the third-order LQ controller can most likely not be reached with a controller suitable for implementation. The difference between the structured controller and the P-controller is bigger here than for the disturbance rejection.

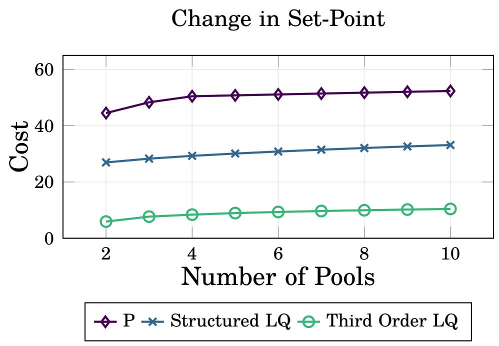


Figure 7. The performance for the different controller for non zero initial conditions, corresponding to a change in set-point. The initial conditions are $y_1 = -1$, $y_N = 1$ and $y_i = 0$ for $2 \leq i \leq N-1$.

Finally, we consider how well the trade-off between input deviations and state deviations can be handled by the structured controller. In Figure 8 we have plotted $\sum y_i[t]^2$ on the x-axis and $\sum u_i[t]$ and $\sum (u_i[t] - u_i[t-1])^2$ respectively on the y-axis for different design parameters. For the structured LQ controller r_N is varied and for the third-order LQ controller r_i and ρ_i respectively are varied. The simulations are carried out on a ten pool network with a disturbance in pool five between time 200 and 400. For the trade-off between the quadratic deviations in the input and in the states, the structured controller allows through the parameter r_N to hold up quiet well to the third-order LQ controller. However, it can be seen that the difference is larger for lower input deviations, which is to be expected. When it comes to minimizing the square of change in input, $(u_i[t] - u_i[t-1])^2$, the structured LQ controller have only a limited ability to influence the trade off though the parameter r_N. Consequently, the trade off becomes quickly worse than for the standard third-order LQ controller. However, if we consider the time response in Figure 5 the input variations looks quite timid, with it being almost constant during the disturbance. If one wanted to reduce the input changes further, one could consider adding an additional local controller to the low-pass filter that minimizes the input changes.

A. Appendix

A.1 Proof of Theorem 1

In this section Theorem 1 will be proven. We start with flows between two pools, that is $u_i,\ i < N$, and then find the optimal flow u_N from the reservoir. We remind ourselves of the following definitions, which will be used in the

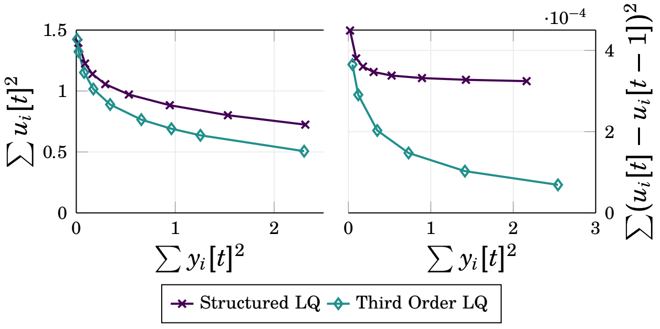


Figure 8. Comparison for how well the two different LQ controllers can handle the trade off between level deviations and input deviations. In the left figure each data point shows the square of input deviations and levels deviations for a choice of design parameters. In the right figure the square of the levels and change in input $(u_i[t] - u_i[t-1])^2$ is plotted for different design parameters. It can be seen that the structured controller can do a fairly good job of handling the trade off between levels and input deviation, while the trade off between levels deviations and change in input is worse.

proof:

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

The proof will rely on results presented in the extended version of [Heyden et al., 2021][2]. Note that in that paper the notation for u_N is v_N. Furthermore, we call the level in each node y instead of z in this paper.

Optimal Internal Flows. We will derive the optimal controller for the following dynamics

$$y_i[t+1] = y_i[t] + u_i[t - \tau_i - \bar{\tau}] - u_{i-1}[t - \bar{\tau}] + d_i[t - \bar{\tau}]. \tag{10}$$

When that is done, we will present a change of variables that transforms the dynamics to the model used for control synthesis in (3).

To get back to the dynamics studied in [Heyden et al., 2021] we apply the following change of variables. Let $\nu_i[t] = u_i[t - \bar{\tau}]$ for $i \leq N - 1$, $\nu_N[t] = u_N[t - \tau_N - \bar{\tau}]$, $\delta_i[t] = d_i[t - \bar{\tau}]$, $\Delta_i[t] = D_i[t - \bar{\tau}]$, and finally

$$\xi_k[t] = \sum_{i=1}^{k} \Big(y_i[t] + \sum_{s=1}^{\tau_i} \nu_i[t - s] \Big).$$

[2] Which is Paper II in this thesis.

In terms of these variables the dynamics in (10) are given by

$$\begin{aligned} y_1[t+1] &= y_1[t] + \nu_1[t-\tau_1] + \delta_1[t] \\ y_i[t+1] &= y_i[t] + \nu_i[t-\tau_i] - \nu_{i-1}[t] + \delta_i[t] \\ y_N[t+1] &= y_N[t] + \nu_N[t] - \nu_{N-1}[t] + \delta_N[t], \end{aligned}$$

which are the the dynamics studied in [Heyden et al., 2021], but with production only in the top node. Lemma 1-iii from [Heyden et al., 2021] holds for any production, and we can always change what time is defined as zero, which gives that for $k < N$ (it holds that $v_k[t] = 0$ and $\bar{V}_i[t] = D_i[t],\ i < N$)

$$\nu_{k-1}[t] = (1 - \frac{\gamma_k}{q_k})(y_k[t] + \nu_k[t-\tau_k]) + \delta_k[t]$$
$$- \frac{\gamma_k}{q_k}\Big(\xi_{k-1}[t] + \Delta_k[t+\sigma_k] + \sum_{i=1}^{k-1}\sum_{d=0}^{\tau_i-1}\Delta_i[t+\sigma_i+d]\Big).$$

For the outflow of node N, which has production, it holds that $\bar{V}_N = D_N[t] + v_N[t]$. And thus ν_{N-1} is given by

$$\nu_{N-1}[t] = (1 - \frac{\gamma_N}{q_N})y_N[t] + \delta_N[t] + \nu_N[t]$$
$$- \frac{\gamma_N}{q_N}\Big(\xi_{N-1}[t] + \nu_N[t] + \Delta_N[t+\sigma_N] + \sum_{i=1}^{N-1}\sum_{d=0}^{\tau_N-1}\Delta_N[t+\sigma_N+d]\Big).$$

Rewriting either expression in terms of the original variables and shifting the time variable by $\bar{\tau}$ gives for $k < N$

$$u_{k-1}[t] = (1 - \frac{\gamma_k}{q_k})(y_k[t+\bar{\tau}] + u_k[t-\tau_k]) + d_k[t]$$
$$-\frac{\gamma_k}{q_k}\left[\sum_{i=1}^{k-1}\Big(y_i[t+\bar{\tau}] + \sum_{s=1}^{\tau_i}u_i[t-s]\Big) + D_k[t+\sigma_k] + \sum_{i=1}^{k-1}\sum_{s=0}^{\tau_i-1}D_i[t+\sigma_i+s]\right]. \tag{11}$$

This expression can not be used for implementation, as $y_i[t+\bar{\tau}]$ is not known at time t. This problem is however easily solved by using the dynamics, which gives that

$$y_i[t+\bar{\tau}] = y_i[t] + \sum_{s=1}^{\bar{\tau}}\Big(-u_{i-1}[t-s] + d_i[t-s] + u_i[t-\tau_i-s]\Big).$$

Collecting all terms in (11) gives for u_i, $1 \le i < k-1$,

$$-\frac{\gamma_k}{q_k}\Big(\sum_{s=1}^{\tau_i}u_i[t-s] + \sum_{s=\tau_i+1}^{\tau_i+\bar{\tau}}u_i[t-s] - \sum_{s=1}^{\bar{\tau}}u_i[t-s]\Big) = -\frac{\gamma_k}{q_k}\sum_{s=\bar{\tau}+1}^{\tau_i+\bar{\tau}}u_i[t-s],$$

for u_{k-1}

$$(1-\frac{\gamma_k}{q_k})(-\sum_{s=1}^{\bar{\tau}} u_{k-1}[t-s]) - \frac{\gamma_k}{q_k}\Big(\sum_{s=1}^{\tau_{k-1}} u_{k-1}[t-s] + \sum_{s=\tau_{k-1}+1}^{\tau_{k-1}+\bar{\tau}} u_{k-1}[t-s]\Big)$$
$$= -\frac{\gamma_k}{q_k}\sum_{s=\bar{\tau}+1}^{\tau_{k-1}+\bar{\tau}} u_{k-1}[t-s] - \sum_{s=1}^{\bar{\tau}} u_{k-1}[t-s],$$

and finally for u_k

$$(1-\frac{\gamma_k}{q_k})\Big(u_k[t-\tau_k] + \sum_{s=\tau_k+1}^{\bar{\tau}+\tau_k} u_k[t-s]\Big) = (1-\frac{\gamma_k}{q_k})\sum_{s=\tau_k}^{\bar{\tau}+\tau_k} u_k[t-s].$$

This gives that

$$u_{k-1}[t] = (1-\frac{\gamma_k}{q_k})(y_k[t]+\sum_{s=\tau_k}^{\tau_k+\bar{\tau}} u_k[t-s]+\sum_{s=1}^{\bar{\tau}} d_k[t-s])+d_k[t]-\sum_{s=1}^{\bar{\tau}} u_{k-1}[t-s]$$
$$-\frac{\gamma_k}{q_k}(\sum_{i=1}^{k-1}\Big(y_i[t]+\sum_{s=\bar{\tau}+1}^{\tau_i+\bar{\tau}} u_i[t-s]+\sum_{s=1}^{\bar{\tau}} d_i[t-s]\Big)+\sum_{i=1}^{k-1}\sum_{s=0}^{\tau_i-1} D_i[t+\sigma_i+s]+D_k[t+\sigma_k]),$$

which is equal to

$$u_{k-1}[t] = (1-\frac{\gamma_k}{q_k})\Big[y_k[t] + \sum_{s=\tau_k}^{\tau_k+\bar{\tau}} u_k[t-s] - \sum_{s=1}^{\bar{\tau}} u_{k-1}[t-s] + \sum_{s=0}^{\bar{\tau}} d_k[t-s]\Big]$$
$$-\frac{\gamma_k}{q_k}\Big[\sum_{i=1}^{k-1}\Big(y_i[t] + \sum_{s=\bar{\tau}+1}^{\tau_i+\bar{\tau}} u_i[t-s] + \sum_{s=1}^{\bar{\tau}} d_i[t-s]\Big) + \sum_{s=1}^{\bar{\tau}} u_{k-1}[t-s]$$
$$+\sum_{i=1}^{k-1}\sum_{d=0}^{\tau_i-1} D_i[t+\sigma_i+d] + (D_k[t+\sigma_k] - d_k[t])\Big].$$

Now, let

$$p_k[t] = y_k[t] + \sum_{s=\tau_k}^{\tau_k+\bar{\tau}} u_k[t-s] - \sum_{s=1}^{\bar{\tau}} u_{k-1}[t-s] + \sum_{s=0}^{\bar{\tau}} d_k[t-s]$$

and

$$m_k[t] = \sum_{i=1}^{k} \left[y_i[t] + \sum_{s=\bar{\tau}+1}^{\tau_i+\bar{\tau}} u_i[t-s] + \sum_{s=1}^{\bar{\tau}} d_i[t-s] + \sum_{s=0}^{\tau_i-1} D_i[t+\sigma_i+s] \right] + \sum_{s=1}^{\bar{\tau}} u_k[t-s] + D_k(t+\sigma_{k+1}).$$

Since $D_k[t+\sigma_k] - d_k[t] = D_{k-1}[t+\sigma_k]$ it then holds that

$$u_{k-1}[t] = (1 - \frac{\gamma_k}{q_k}) p_k[t] - \frac{\gamma_k}{q_k} m_{k-1}[t].$$

$m_k[t]$ can be calculated recursively as follows:

$$m_k[t] = m_{k-1}[t] + p_k[t] + \sum_{s=1}^{\tau_k-1} u_k[t-s] + \sum_{s=1}^{\tau_k} D_k[t+\sigma_k+s]$$

where it is used that $d_k[t] + D_{k-1}[t+\sigma_{k-1}+\tau_k] = D_k[t+\sigma_k]$.

Optimal Production. The steps in Lemma 3 in [Heyden et al., 2021], can be carried out with $\rho_N = r$ to find the optimal ν_N (that is v_N in the lemma). Equation 16 in [Heyden et al., 2021] will then give that

$$\nu_N[t] = -\frac{X}{r} \left(\xi_{N-1}[t] + \sum_{i=1}^{N-1} \sum_{s=\sigma_i}^{\sigma_{i+1}-1} \Delta_i[t+s] + \mu_N[t] \right).$$

Where X is given by

$$X = -\frac{\gamma_N}{2} + \sqrt{\gamma_N r + \frac{\gamma_N^2}{4}},$$

and $\mu_N[t]$ is given by

$$\mu_N[t] = y_N[t] + \sum_{s=0}^{H} \Delta_N[t+\sigma_N+s] \prod_{j=2}^{s+1} g,$$

where $g = X/(X+\gamma_N)$ and the product over an empty set is defined to be 1. Note that in [Heyden et al., 2021] all $X_N(i)$ are the same, as otherwise $X_N(H+2)+\gamma_N$ would not be a solution to the Riccati equation, and that τ_N was defined as $H+1$ in [Heyden et al., 2021] for notational convenience.

Going back to the original variables and shifting the time variable by $\tau_n + \bar{\tau}$ gives

$$u_N[t] = -\frac{X}{r}\Big[\sum_{i=1}^{N-1}\Big(y_i[t+\tau_N+\bar{\tau}] + \sum_{s=1}^{\tau_i} u_i[t+\tau_N-s] + \sum_{s=\sigma_i}^{\sigma_{i+1}-1} D_i[t+s+\tau_N]\Big)$$
$$+ y_N[t+\tau_N+\bar{\tau}] + \sum_{s=0}^{H} D_N[t+\sigma_N+\tau_N+s]\prod_{j=2}^{s+1} g\Big]. \quad (12)$$

Using the dynamics to rewrite $y_i[t+\tau_N+\bar{\tau}]$ gives

$$y_i[t+\tau_N+\bar{\tau}] = y_i[t] + \sum_{s=\tau_i+1-\tau_N}^{\tau_i+\bar{\tau}} u_i[t-s] - \sum_{s=1-\tau_N}^{\bar{\tau}} u_{i-1}[t-s] + \sum_{s=1-\tau_N}^{\bar{\tau}} d_i[t-s].$$

One can note that all terms in the RHS will not be known at time t. However, the issue solves itself as follows. Collecting all terms containing u_i $i \leq N$ in (12) gives

$$\sum_{s=1-\tau_N}^{\tau_i-\tau_N} u_i[t-s] + \sum_{s=\tau_i+1-\tau_N}^{\tau_i+\bar{\tau}} u_i[t-s] - \sum_{s=1-\tau_N}^{\bar{\tau}} u_i[t-s] = \sum_{s=\bar{\tau}+1}^{\tau_i+\bar{\tau}} u_i[t-s].$$

And all terms including u_N gives

$$\sum_{s=-\tau_N+\tau_N+1}^{\tau_N+\bar{\tau}} u_N[t-s] = \sum_{s=1}^{\tau_N+\bar{\tau}} u_N[t-s].$$

And we note that both sums are now quantities known at time t.

All the disturbances $d_i[t]$ for $t \geq 0$ are given by

$$\sum_{i=1}^{N}\sum_{s=0}^{\tau_N-1} d_i[t+s] + \sum_{i=1}^{N-1}\sum_{s=\sigma_i}^{\sigma_{i+1}-1} D_i[t+s+\tau_N] + \sum_{s=0}^{H} D_N[t+\sigma_N+\tau_N+s]\prod_{j=2}^{d+1} g. \quad (13)$$

It holds that $D_N[t+\sigma_N+\tau_N+s] = 0$ for all $s+\tau_N > H$ by the assumption that $d_i[t] = 0$ for $t > H$. Thus the last term in (13) is given by

$$\sum_{s=0}^{H-\tau_N} D_N[t+\sigma_N+\tau_N+s]\prod_{j=2}^{s+1} g$$
$$= \sum_{s=\tau_N}^{H} D_N[t+\sigma_N+s]\prod_{j=2}^{s-\tau_N+1} g = \sum_{s=\tau_N}^{H} D_N[t+\sigma_N+s]\prod_{j=\tau_N+2}^{s+1} g.$$

Using the definition for $D_i[t]$, the first two terms in (13) are equal to

$$\sum_{i=1}^{N}\sum_{s=0}^{\tau_N-1} d_i[t+s] + \sum_{i=1}^{N-1}\sum_{s=\sigma_i+\tau_N}^{\sigma_{i+1}+\tau_N-1}\sum_{j=1}^{i} d_j[t+s-\sigma_j].$$

Collecting all d_k terms for a given k gives

$$\sum_{s=\sigma_k}^{\sigma_k+\tau_N-1} d_k[t+s-\sigma_k] + \sum_{i=k}^{N-1}\sum_{s=\sigma_i+\tau_N}^{\sigma_{i+1}+\tau_N-1} d_k[t+s-\sigma_k] = \sum_{s=\sigma_k}^{\sigma_{N+1}-1} d_k[t+s-\sigma_k].$$

And thus the first two terms in (13) are equal to,

$$\sum_{i=1}^{N}\sum_{s=\sigma_i}^{\sigma_{N+1}-1} d_i[t+s-\sigma_i] = \sum_{i=1}^{N}\sum_{s=\sigma_i}^{\sigma_{i+1}-1}\sum_{j=1}^{i} d_j[t+s-\sigma_j] = \sum_{i=1}^{N}\sum_{d=\sigma_i}^{\sigma_{i+1}-1} D_i[t+d].$$

Thus the total effect of the planned disturbances in the expression for $u_N[t]$ in (12) is

$$\sum_{s=1}^{\bar\tau} d_i[t-s] + \sum_{i=1}^{N}\sum_{s=\sigma_i}^{\sigma_{i+1}-1} D_i[t+s] + \sum_{s=\tau_N}^{H} D_N[t+\sigma_N+s]\prod_{j=\tau_N+2}^{s+1} g.$$

So $u_N[t]$ is given by

$$u_N[t] = -\frac{X}{r}\Big[\sum_{i=1}^{N}\Big(y_i[t] + \sum_{s=\bar\tau+1}^{\tau_i+\bar\tau} u_i[t-s] + \sum_{s=1}^{\bar\tau} d_i[t-s] + \sum_{s=\sigma_i}^{\sigma_{i+1}-1} D_i[t+s]\Big)$$
$$+ \sum_{s=1}^{\bar\tau} u_N[t-s] + \sum_{s=\tau_N}^{H} D_N[t+\sigma_N+s]\prod_{j=2}^{d+1} g\Big].$$

Which can be expressed as

$$u_N[t] = -\frac{X}{r}\left[m_N + \sum_{s=\tau_N+1}^{H} D_N[t+\sigma_N+s]\prod_{j=2}^{d+1} g\right].$$

Change of variables. The structured controller is synthesized for dynamics on the form in (10), while the plant model is on the form in (3). However, there exists a simple change of variables that allows us to transform between the two models. Consider the synthesis dynamics

$$y_i[t+1] = y_i[t] + b_i u_i[t-\tau_i-\bar\tau] - c_i(u_{i-1}[t-\bar\tau] - d_i[t-\bar\tau]).$$

Let

$$\hat b_1 = b_1, \quad \hat b_i = \frac{b_i}{c_i}\hat b_{i-1}, \quad z_1 = y_1, \quad z_i = \frac{\hat b_{i-1}}{c_i} y_i,$$

and
$$\hat{u}_i = \hat{b}_i u_i, \quad \hat{d}_1 = c_1 d_1, \quad \hat{d}_i = \hat{b}_{i-1} d_i \; i \geq 2.$$
Then the dynamics in (3) are transformed to
$$z_i[t+1] = z_i[t] + \hat{u}_i[t - \tau_i - \bar{\tau}] - \hat{u}_{i-1}[t - \bar{\tau}] + \hat{d}_i[t - \bar{\tau}]. \tag{14}$$
This follows trivially for node 1. For node i, we get by replacing u_{i-1} with $1/\hat{b}_{i-1} \cdot \hat{u}_{i-1}$
$$y_i[t+1] = y_i[t] + b_i u_i[t - \tau_i - \bar{\tau}] - c_i/\hat{b}_{i-1} \cdot u_{i-1}[t - \bar{\tau}] - c_i d_i[t - \bar{\tau}]).$$
Which can be rewritten as
$$\hat{b}_{i-1}/c_i y_i[t+1] = \hat{b}_{i-1}/c_i y_i[t] + \hat{b}_{i-1} b_i/c_i u_i[t - \tau_i - \bar{\tau}] - \hat{u}_{i-1}[t - \bar{\tau}] - \hat{b}_{i-1} d_i[t - \bar{\tau}]).$$
Applying the suggested change of variables gives the dynamics in (14). For the cost parameters it follows that
$$r u_N^2 = \frac{r}{\hat{b}_N^2} \hat{u}_N^2, \quad q_i y_i = \frac{q_i c_i^2}{\hat{b}_{i-1}^2} z_i, \quad i \geq 2.$$

This change of variables is implemented in Algorithm 1 on lines 4 and 8, and in Algorithm 2 on lines 1-2 and 19.

A.2 LQ with known disturbance

Here we give the derivation of a LQ controller with feed-forward. That is we consider the problem
$$\begin{aligned} \text{minimize} \quad & \sum_{t=0}^{\infty} x[t]^T Q x[t] + u[t]^T R u[t] \\ \text{subject to} \quad & x[t+1] = Ax[t] + Bu[t] + v[t] \\ & x[0] \text{ and } v[t] \text{ given.} \end{aligned}$$

This is a well studied problem when $v[t] = 0$, see for example [Bertsekas, 2012], and we only consider the extension due to the planned disturbance $v[t]$.

Assume that $v[t] = 0$ for all $t > N$ and R is positive definite. Let S be the solution to the algebraic Riccati equation
$$S = A^T S A - A^T S B (B^T S B + R)^{-1} B^T S A + Q.$$
Note that S will be symmetric. The cost to go from time $N+1$ is given by $x[N+1]^T S x[N+1]$, and the optimal $u[N]$ is given by the minimizer for

the cost to go from time $t = N$:

$$\begin{aligned} x[N]^T Qx[N] + u[N]^T Ru[N] + x[N+1]^T Sx[N+1] = \\ x[N]^T Qx[N] + u[N]^T Ru[N] + \\ (Ax[N] + Bu[N] + v[N])^T S(Ax[N] + Bu[N] + v[N]). \end{aligned} \tag{15}$$

Collecting all terms which has $u[N]$ in them gives

$$u[N]^T Ru[N] + 2(Ax[N] + v[N])^T SBu[N] + u[N]^T B^T SBu[N].$$

The problem is convex, and differentiating with respect to u gives that the optimal u is given by

$$\begin{aligned} 2(B^T SB + R)u[N] &= -2B^T S(Ax[N] + v[N]) \\ u[N] &= -(B^T SB + R)^{-1} B^T S(Ax[N] + v[N]). \end{aligned}$$

Let $\Pi[N] = Sv[N]$. It holds that $u[N] = Kx[N] + K_v \Pi[N]$, where K and K_v are as in (7). Inserting the expression for $u[N]$ into (15) and only considering terms that depend on $x[N]$ gives for the cost to go:

$$\begin{aligned} &x^T[N]Qx[N] + (Ax[N] + v[N])^T \Big[SB(B^T SB + R)^{-T} R(B^T SB + R)^{-1} B^T S \\ &\quad + S - 2SB(B^T SB + R)^{-1} B^T S \\ &\quad + SB(B^T SB + R)^{-T} B^T SB(B^T SB + R)^{-1} B^T S \Big] (Ax[N] + v[N]) = \\ &x^T[N]Qx[N] + (Ax[N] + v[N])^T \left[S - SB(B^T SB + R)^{-1} B^T S \right] (Ax[N] + v[N]). \end{aligned} \tag{16}$$

The terms containing only $x[N]$ simplifies to $x^T[N]Sx[N]$. For the terms containing $x[N]$ and $v[N]$ we get

$$\begin{aligned} 2v^T[N](S - SB(B^T SB + R)^{-1} B^T S)Ax[N] \\ = 2v^T[N]S(A + BK)x[N] \\ = 2\Pi[N]^T (A + BK)x[N]. \end{aligned}$$

Thus the cost to go at time $N - 1$ is given by

$$\begin{aligned} x[N-1]^T Qx[N-1] + u[N-1]^T Ru[N-1] + x[N]^T Sx[N] \\ + 2\Pi[N]^T (A + BK)x[N]. \end{aligned}$$

Now assume that the cost to go for some t, $t \leq N - 1$, is given by

$$\begin{aligned} &x[t]^T Qx[t] + u[t]^T Ru[t] + x[t+1]^T Sx[t+1] + 2\Pi[t+1]^T (A + BK)x[t+1] \\ &= x[t]^T Qx[t] + u[t]^T Ru[t] + (Ax[t] + Bu[t] + v[t])^T S(Ax[t] + Bu[t] + v[t]) \\ &\quad + 2\Pi[t+1]^T (A + BK)(Ax[t] + Bu[t] + v[t]). \end{aligned} \tag{17}$$

The assumption holds for $t = N - 1$ by the previous calculations. Differentiating w.r.t $u[t]$ gives

$$\begin{aligned}2(B^T SB{+}R)u[t] &= -2B^T S(Ax[N] + v[N]) - 2B^T(A + BK)^T \Pi[t+1]\\ \Rightarrow u[t] &= -(B^T SB + R)^{-1} B^T(SAx[t] + Sv[t] + (A + BK)^T \Pi[t+1])\\ &= Kx[t] + K_v\Big(Sv[t] + (A + BK)^T \Pi[t+1]\Big).\end{aligned}$$

Letting $\Pi[t] = (A + BK)^T \Pi[t+1] + Sv[t]$ gives that

$$u[t] = Kx[t] + K_v \Pi[t], \tag{18}$$

as long as the cost to go is given by (17).

Now we consider the cost to go in (17). Every term that was in the cost to go for $t = N$ in (16) will remain, that is the term $x[t]^T Sx[t]$ and $2v^T[t]S(A + BK)x[t]$. However new terms will be added due to the addition of $\Pi[t+1]$ to the expression for $u[t]$ and the new term

$$2\Pi[t+1](A + BK)x[t+1].$$

in (17) compared to (15). We ignore the term $2\Pi[t+1](A+BK)x[t+1]$ for now, and focus on the effect of $\Pi[t+1]$ in the expression for u. The resulting effect on (17) for terms that include $x[t]$ are given by (where the first term is due to $u[t]^T Ru[t]$, the second is due to $(Bu[t])^T SBu[t]$, and the third is due to $(Bu[t])^T SAx[t]$)

$$\begin{aligned}&2\Pi[t+1]^T(A + BK)K_v^T RKx[t]\\ &+2\Pi[t+1]^T(A + BK)K_v^T B^T SBKx[t]\\ &+2\Pi[t+1]^T(A + BK)K_v^T B^T SAx[t] =\\ &-2\Pi[t+1]^T(A + BK)K_v^T B^T SAx[t]\\ &+2\Pi[t+1]^T(A + BK)K_v^T B^T SAx[t] = 0.\end{aligned}$$

Where we have used that $(R + B^T SB)K = -B^T SA$. The effect of the new term $2\Pi[t+1](A + BK)x[t+1]$ in terms of $x[t]$ is given by

$$2\Pi[t+1]^T(A + BK)(A + BK)x[t].$$

So the total effect of the disturbances on the cost to go is given by

$$\begin{aligned}2v[t]^T S(A + BK)x[t] + 2\Pi[t+1]^T(A + BK)(A + BK)x[t]&\\ = 2\Pi[t]^T(A + BK)x[t],&\end{aligned}$$

and thus the cost to go is given on the assumed form in (17) for $t - 1$ as well. Thus (17) and (18) holds for $0 \le t \le N$.

References

Bertsekas, D. (2012). *Dynamic programming and optimal control*. Vol. 1. Athena scientific.

Cantoni, M., E. Weyer, Y. Li, S. K. Ooi, I. Mareels, and M. Ryan (2007). "Control of large-scale irrigation networks". *Proceedings of the IEEE* **95**:1, pp. 75–91.

Heyden, M., R. Pates, and A. Rantzer (2021). "A structured optimal controller with feed-forward for transportation". *IEEE Control Systems Letters*.

Lemos, J. M. and L. F. Pinto (2012). "Distributed linear-quadratic control of serially chained systems: application to a water delivery canal". *IEEE Control Systems Magazine* **32**:6, pp. 26–38.

Litrico, X. and V. Fromion (2005). "Design of structured multivariable controllers for irrigation canals". In: *Proceedings of the 44th IEEE Conference on Decision and Control*. IEEE, pp. 1881–1886.

Litrico, X., V. Fromion, J.-P. Baume, and M. Rijo (2003). "Modelling and PI control of an irrigation canal". In: *2003 European Control Conference (ECC)*. IEEE, pp. 850–855.

Lozano, D., C. Arranja, M. Rijo, and L. Mateos (2010). "Simulation of automatic control of an irrigation canal". *Agricultural water management* **97**:1, pp. 91–100.

Mareels, I., E. Weyer, S. K. Ooi, M. Cantoni, Y. Li, and G. Nair (2005). "Systems engineering for irrigation systems: successes and challenges". *IFAC Proceedings Volumes* **38**:1, pp. 1–16.

Negenborn, R. R., A. Sahiir, Z. Lukszcr, B. De Schutter, and M. Morari (2009). "A non-iterative cascaded predictive control approach for control of irrigation canals". In: *2009 IEEE International Conference on Systems, Man and Cybernetics*. IEEE, pp. 3552–3557.

Neshastehriz, A. R., M. Cantoni, and I. Shames (2014). "Water-level reference planning for automated irrigation channels via robust MPC". In: *2014 European Control Conference (ECC)*. IEEE, pp. 1331–1336.

Ooi, S. K., M. Krutzen, and E. Weyer (2005). "On physical and data driven modelling of irrigation channels". *Control Engineering Practice* **13**:4, pp. 461–471.

Schuurmans, J., A. Hof, S. Dijkstra, O. Bosgra, and R. Brouwer (1999). "Simple water level controller for irrigation and drainage canals". *Journal of irrigation and drainage engineering* **125**:4, pp. 189–195.

Weyer, E. (2002). "Decentralised pi control of an open water channel". *IFAC Proceedings Volumes* **35**:1, pp. 95–100.

Weyer, E. (2003). “LQ control of an irrigation channel”. In: *42nd IEEE International Conference on Decision and Control (IEEE Cat. No. 03CH37475)*. Vol. 1. IEEE, pp. 750–755.

Weyer, E. (2008). “Control of irrigation channels”. *IEEE Transactions on Control Systems Technology* **16**:4, pp. 664–675.

Paper IV

Price Based Linear Quadratic Control Under Transportation Delay

Martin Heyden   Richard Pates   Anders Rantzer

Abstract

We study a simple transportation problem on a string graph. The objective is to regulate the node levels of some decaying quantity to optimize dynamical performance. This can be achieved by controlling the flows, which are subject to delay, between neighbouring nodes. The problem is considered from two perspectives. In the first (the social perspective), all nodes cooperate to find the flows that maximize the aggregated utility of the entire transportation network. In the second (the user perspective), the nodes instead try to maximimize their own utility. Our main contribution is to give an implementation of the feedback law that gives the social optimum, that only depends on the local states and a set of prices defined by a distributed update rule. These prices align the social and user optimum in a budget neutral way, and give all nodes no worse cost than if they were on their own.

1. Introduction

In this work we study the optimal transportation of a decaying quantity. The dynamics studied could for example describe a transportation network, as illustrated in Section 2. The objective is to control the node levels by regulating the transportation between the nodes to optimize performance. The challenge is to do this in a manner that scales well with network size, whilst accounting for dynamical effects such as transportation delays.

To capture the essence of the problem, we consider a string network with N nodes in discrete time. The nodes are numbered as in Figure 1, where the most downstream node has index one, the second most downstream has index two, and so on. Furthermore, we index the links according to the node which they enter. We let the transportation delay be one time unit on every link, and define the dynamics to be

$$z_i[t+1] = \alpha\left(z_i[t] + u_i[t-1]\right) - u_{i-1}[t], \tag{1}$$

for $1 \le i \le N$. The variable $z_i[t]$ is the level of the quantity in node i (at time step t), and the control input $u_{i-1}[t]$ is the amount leaving node i (if written in state-space form, a choice of the system system state would be $\{z_i[t],\ u_i[t-1]\}$). At the boundaries we have $u_0[t] = 0$, $u_N[t] = 0$. The constant $0 < \alpha \le 1$ is the decay rate.

We assume that node i values its level $z_i[t]$ according to the quadratic function, $U_i(z_i[t]) = b_i z_i[t] - 1/2 q_i z_i[t]^2$, where $q_i > 0$, $b_i > 0$, and $b_{i+1} = \alpha b_i$. The last assumption arises naturally when considering transportation about an equilibrium, and will be motivated fully in the next section. We study the problem from two perspectives. First we consider the social optimum problem, where the objective is to maximize the sum of local utility functions for the nodes:

$$\begin{aligned} \underset{z,u}{\text{maximize}} \quad & J(z) = \sum_{i=1}^{N}\sum_{t=1}^{T}\left(b_i z_i[t] - \frac{1}{2} q_i z_i[t]^2\right) \\ \text{subject to} \quad & \text{Dynamics in (1)} \\ & z_i[0] \text{ given.} \end{aligned} \tag{2}$$

In the above, $z[t] \in \mathbb{R}^N$ is defined for $1 \le t \le T$ and $u[t] \in \mathbb{R}^{N-1}$ is defined for $0 \le t \le T-1$. This problem is an instance of a Linear Quadratic control problem with a linear term in the optimization criterion. The absence of an input penalty allows for a highly structured solution that is efficient to calculate. This was demonstrated for the infinite horizon case in [Heyden et al., 2018] (with $b_i = 0$) by finding the solution to an algebraic Riccati equation.

Our main contribution is to provide a distributed solution to eq. (2), using a pricing mechanism to implement the feedback law. The prices are used to

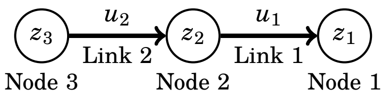


Figure 1. Illustration of the studied problem when $N = 3$. There are three nodes whose levels z are to be controlled. Each link has a corresponding input u that is the flow entering the link and arriving one time unit later. We have also indicated the indexing conventions for the nodes and links.

adjust the utility of the individual nodes, so that each has an optimal level when considering its own utility. Our scheme is budget neutral, is simple to implement even in very large networks, and leaves no node worse off than if it were on its own. These results are previewed at the end of this section, and presented in Section 3.

Our results add to the growing body of work on distributed control. Early effort in this regard include team game problems, where a set of agents work towards a common goal, but with different information, see for example [Radner, 1962]. Important work along these lines includes [Rotkowitz and Lall, 2006], where sufficient conditions for finding a distributed controller using convex optimization were given. More recent work includes System Level Synthesis, see for example [Anderson et al., 2019], that allows for scalable synthesis and implementation of distributed controllers using a novel controller architecture. In contrast, the structure in the controllers in this paper is inherited from the plant. This is similar in nature to the work on spatially invariant systems from [Bamieh et al., 2002], where the optimal control law was shown to be localized in space. The structure of the controllers derived in this work also have strong similarities to those from [Shah and Parrilo, 2013], where the optimal poset-causal controller is found.

The objectives of this paper are also well aligned with the theme of solving optimization problems using prices based on Lagrange multipliers. For pioneering work on the use of prices for coordination, see [Cohen, 1978], which was later used to control water supply networks in [Carpentier and Cohen, 1993]. The use of Lagrange multipliers for coordination is well studied, for example as shadow prices in the work on Internet congestion control from [Kelly et al., 1998], [Low et al., 2002], and in the distributed MPC schemes from [Giselsson et al., 2013]. Lagrange multipliers have also been suggested for controlling power grids [Jafarian et al., 2016], [Jokić et al., 2009]. Normally these problems are either static, or of high complexity. Requiring either solving for all the prices and states at the same time, or solving a Riccati equation. In our specific problem, the prices are not the Lagrange multipliers, but rather a linear combination of current node levels and goods in transit. These prices are much simpler to compute than the

Lagrange multipliers are.

Preview of Results

To give an incentive for the individual nodes to follow the social optimum, we will introduce prices and study the problem from a node perspective. Each node i will be presented with a price vector $p_i[t]$ that will affect the nodes utility proportional to their levels, so that its total utility $V_i(z_i, p_i)$ is given by

$$V_i(z_i, p_i) = p_i[0]z_i[0] + \sum_{t=1}^{T}\Big(b_i z_i[t] - \frac{1}{2}q_i z_i^2[t] - p_i[t]z_i[t]\Big). \tag{3}$$

Typically increasing $z_i[t]$ will lead to a trade off between the increased utility from the $b_i z_i[t] - 1/2 q_i z_i^2[t]$ term, and the decreased utility from the cost $p_i[t]z_i[t]$. The utility function in (3) will be further discussed in Section 2. Each node will consider the following problem

$$\begin{aligned} \underset{z_i}{\text{maximize}} \quad & V_i(z_i, p_i) \\ \text{subject to} \quad & p_i \text{ given.} \end{aligned} \tag{4}$$

We find the solution to social optimum problem by studying the Lagrangian of the problem. The main contribution lies in deriving a set of prices from the Lagrange multipliers that allows for a distributed implementation of the optimal feedback law and aligns social and user optimum. However, in contrast to 'typical' Lagrangian approaches, the prices are given by a simple, temporally decoupled, expression

$$p_i[t+1] = \begin{cases} b_i - \gamma_i \sum_{j=1}^{i} z_j[t] + u_j[t-1] & 0 \le t \le T-i \\ 0 & t > T-i. \end{cases} \tag{5}$$

In the above, γ is a constant that can be computed ahead of time by the following iteration:

$$\gamma_1 = q_1, \quad \gamma_i = \frac{\alpha^2 \gamma_{i-1} q_i}{\alpha^2 \gamma_{i-1} + q_i}, \; i \ge 2. \tag{6}$$

With p as in (5), the optimal inputs are given by

$$u_{i-1}[t] = \alpha(z_i[t] + u_i[t-1]) - \frac{1}{q_i}\big(p_i[t+1] - b_i\big). \tag{7}$$

The combined structure of (5) and (7) allows for a simple implementation of the optimal u using only local communication. The expression for the optimal prices in (5) indicates that the price should increase the more a node values its level from the term b_i, and decrease when more goods are available.

2. Motivation for the Problem

We will consider a simple model of a generic transportation network for a decaying quantity. This could for example be a district heating network, or an inventory control system for decaying goods. In this section we will show that when controlling such a system around an equilibrium, the dynamics in (1) arise. This could, for example, be of interest if the operating conditions changes and the system needs to be shifted from the old to the new equilibrium point.

Each node i in the network has a constant production (or consumption) w_i. Furthermore, the quantity can be transported along the links of the system. Finally, we make the simplifying assumption that the decay has a homogeneous rate $1-\alpha$ throughout the system. We can write the dynamics for the level ζ_i in each node i as

$$\zeta_i[t+1] = \alpha\big(\zeta_i[t] + v_i[t-1]\big) + w_i - v_{i-1}[t].$$

In the above v_{i-1} is the quantity leaving node i and v_i is the quantity arriving to node i. The quantity leaving the node goes immediately into transportation and will take one time unit to arrive. For the physical interpretation the flows $v[t]$ must be positive. This will generally be the case if there is producer at the top of the network.

Let the flows $v[t] = \bar{v}$ be constant. Then each node will have an equilibrium level $\bar{\zeta}_i$ where the inflow equals the outflow,

$$\bar{\zeta}_i = \frac{1}{1-\alpha}(w_i + \alpha\bar{v}_i - \bar{v}_{i-1}).$$

We assume that each node values its level according to a quadratic function $U_i(\zeta_i)$. Then the optimal equilibrium is the solution to

$$\begin{aligned} \underset{\bar{v}}{\text{maximize}} \quad & \sum_i U_i(\bar{\zeta}_i) \\ \text{subject to} \quad & \bar{\zeta}_i = \frac{1}{1-\alpha}(w_i + \alpha\bar{v}_i - \bar{v}_{i-1}). \end{aligned} \tag{8}$$

Now we study the system around this equilibrium. We introduce a new level vector $z \in \mathbb{R}^N$ describing the levels relative to the optimal levels, and a new input vector $u \in \mathbb{R}^{N-1}$ that describe the flows relative to the optimal flows,

$$\begin{aligned} u_i[t] &= v_i[t] - \bar{v}_i, \\ z_i[t] &= \zeta_i[t] - \bar{\zeta}_i. \end{aligned}$$

Then the dynamics for z are given by (1). The utility relative to the optimum can be written as

$$U_i(\bar{\zeta}_i + z_i[t]) - U_i(\bar{\zeta}_i) = \sum_{t=1}^{T}\left(b_i z_i[t] - \frac{1}{2}q_i z_i^2[t]\right),$$

where $q_i < 0$. Note that $u_i[t]$ can take negative values.

REMARK 1
Since $\bar{\zeta}_i$ solves (8), we must have that

$$b_{i+1} = \alpha b_i. \tag{9}$$

To see this, observe that if it were not the case, then the utility could be improved by making a small perturbation ϵ to $\bar{v}_i$ which would increase $\bar{\zeta}_i$ by $\alpha/(1-\alpha)\epsilon$ and decrease $\bar{\zeta}_{i+1}$ by $1/(1-\alpha)\epsilon$. □

The optimal control around the equilibrium can be found by solving (2). The lack of penalty on the flows can be motivated by that the cost of changing the transportation is small. For example, if the transportation is done via trucks, then there is typically a very low, or no additional cost, if a truck transports more goods. However, there is still a loss in moving the quantity in that it is not being utilized while in transportation.

How can the user problem in (3) be motivated? It is natural that the users in the transportation network pay, or are paid, for changes in equilibrium levels. If the new equilibrium level of a node is lower, then that node would expect to be paid to actively send away some of its quantity, since this reduction will reduce its own utility. Similarly, if a node is to receive a higher level, that node would be expected to pay for it. This is captured by the term $p_i[0]z_i[0]$. Since the new equilibrium cannot be reached immediately, the nodes should also pay for the time periods where they have a higher level, and be compensated while it is too low. This is captured by the terms $p_i[t]z_i[t]$. We note that close to the equilibrium, $p_i[t] > 0$, $t \geq 1$. We shall later see that that is the case for $t = 0$ as well.

3. Results

We start by giving the solution to (2), in Theorem 1 below. This result shows that the i-th entry of the optimal control input can computed based only on local measurements of the quantity z and the goods in transit u, and a local price p_i. Next we show in Proposition 1 how these prices can be used to align the user problem in (4) to the social optimum. The prices have additional appealing properties. Firstly, the node utilities are higher than if they had zero flows and no payments, and secondly, the sum of all payments equal zero. The proofs of the results presented here will be given in Section 5.

THEOREM 1
Define γ_i as in (6), and $p[t]$ by

$$p_i[t+1] = \begin{cases} b_i - \gamma_i \sum_{j=1}^{i} z_j[t] + u_j[t-1], & 0 \leq t \leq T-i \\ 0 & t > T-i. \end{cases} \tag{10}$$

Then the optimal u for (2) is given by

$$u[t] = \alpha\,[\mathbf{0}\;\; I] \begin{bmatrix} z_1[t] + u_1[t-1] \\ \vdots \\ z_{N-1}[t] + u_{N-1}[t-1] \\ z_N[t] \end{bmatrix} - \begin{bmatrix} 0 & \frac{1}{q_2} & & 0 \\ & & \ddots & \\ 0 & & & \frac{1}{q_N} \end{bmatrix} (p[t+1] - b). \qquad (11)$$

With $p[t] = [p_1[t], \ldots, p_N[t]]^T$ and $b = [b_1, \ldots, b_N]^T$, $\mathbf{0}$ a column vector of length $N-1$ and I an identity matrix of dimension $N-1$. □

If we write out the expressions for each input we get (7). From the theorem we see that there exists a simple method for calculating the optimal feedback law, using only local states and local prices. Furthermore, $p_i[t+1]$ can be calculated recursively through the graph,

$$p_i[t+1] = \gamma_i \left(-z_i[t] - u_i[t-1] + \frac{1}{\gamma_{i-1}} p_{i-1}[t+1] \right) + (1 - \frac{1}{\alpha\gamma_{i-1}}) b_i,$$

requiring only local communication. This expression is also interesting in that each node only needs to share a combination of its level and utility function. This gives some privacy compared to sharing both the level and the utility function.

Equation (7) has a very natural interpretation from the user optimal perspective, as it is the solution to

$$\begin{aligned} \underset{u_{i-1}[t-1]}{\text{minimize}} \quad & b_i z_i[t] - \frac{1}{2} q_i z_i^2[t] - p_i[t] z_i[t] \\ \text{subject to} \quad & z_i[t] = \alpha(z_i[t-1] + u_i[t-2]) - u_{i-1}[t], \end{aligned}$$

which corresponds to the node optimizing its utility for the next time point.

Remark 2
At first sight it may seem like (11) is non causal as the input at time t depends on prices at time $t+1$. However, from (10) we can see that prices at time $t+1$ depends on state at time t, and the expression is indeed causal. As the prices are associated with the states when the input has taken affect, it is natural that the prices are one time-index ahead of the inputs. □

Remark 3
It might be surprising that some of the prices are zero and thus the corresponding nodes will have optimal levels, $z_i = -b_2/q_2$, as t gets closer to T. This is due to the boundary effects of the system, where the level of a node can be increased without decreasing the value of others. This is achieved by by exploiting that the goods sent at time $T-1$ will not reach its destination within the optimization horizon . □

Proposition 1
In addition to the definitions in Theorem 1, let $m = \min(T - i, N)$, and

$$p_i[0] = \sum_{t=1}^{T-(i-1)} \left(\alpha^{t-(t_0+\tau)} b_i\right) - \alpha\left(\sum_{j=i}^{m} \gamma_j \sum_{k=1}^{j} z_k[0] + u_k[-1] + \sum_{j=N+1}^{T} \gamma_N \alpha^{2(j-N)} \sum_{k=1}^{N} z_k[0] + u_k[-1]\right)$$

Then:

(i) The optimal z for (2) and (4) are equal.

(ii) The node utility satisfies

$$V(z_i, p_i) \geq \sum_{t=1}^{T} \left(b_i \alpha^t z_i[0] - q_i(\alpha^t z_i[0])^2\right).$$

(iii) The sum of payments are equal to zero,

$$\sum_{i=1}^{N} \left(p_i[0] z_i[0] - \sum_{t=1}^{T} p_i[t] z_i[t]\right) = 0.$$

□

The proposition shows not only that is possible to align the user and social optimum, but also that each node is never worse off than if they had no in- or outflow. This is what would happen if the node was not a part of the transportation network. This is an important property, as otherwise the nodes would be reluctant to be part of the network. Furthermore the payment scheme is budget neutral (the payments sum to zero). This is significant as if the scheme had a budget deficit, it would be very hard to find someone to supply additional money to drive the system, while receiving nothing in return.

4. Analysis of the Lagrangian

In this section we perform the necessary analysis of the Lagrangian for (2) needed to prove Theorem 1 and Proposition 1. An important part is to construct an alternative user utility based on the Lagrange multipliers, and showing that it is equal to the original one in (4).

4.1 Lagrangian

The Lagrangian of (2) is given by

$$L(z, u, \lambda) = J(z) + \sum_{t=0}^{T-1} \Big[\lambda_1[t+1]\Big\{\alpha\big(z_1[t] + u_1[t-1]\big) - z_1[t+1]\Big\}$$
$$+ \sum_{i=1}^{N-1} \lambda_i[t+1]\Big\{\big(\alpha(z_i[t] + u_i[t-1]) - u_{i-1}[t]\big) - z_i[t+1]\Big\}$$
$$+ \lambda_N[t+1]\Big\{\big(\alpha z_N[t] - u_{N-1}[t]\big) - z_N[t+1]\Big\}\Big].$$

The Lagrange dual variable has dimensions $\lambda[t] \in \mathbb{R}^N$ for $1 \leq t \leq T$. The dual variables $\lambda_i[t]$ has a natural economic interpretation as the marginal change in social utility when $z_i[t]$ changes.

4.2 Alternative User Optimal Problem

Based on the Lagrangian we define an alternative user utility function, and show that it is equal to the original in (3). In this formulation the node utility will include a cost based on the level change,

$$\hat{V}_i(z_i, \lambda_i) = \sum_{t=1}^{T} b_i z_i[t] - \frac{1}{2} q_i z_i^2[t] - \lambda_i[t] \underbrace{\big(z_i[t] - \alpha z_i[t-1]\big)}_{\text{change in level}}. \qquad (12)$$

Note that all the terms in $\hat{V}_i$ are in the Lagrangian L. By letting

$$p_i[t] = \begin{cases} \alpha\lambda_i[1] & t = 0 \\ \lambda_i[t] - \alpha\lambda_i[t+1] & 1 \leq t \leq T-1 \\ \lambda_i[T] & t = T, \end{cases} \qquad (13)$$

the node utility can be rewritten as

$$\hat{V}_i(z_i, \lambda_i)$$
$$= \sum_{t=1}^{T-1} b_i z_i[t] - \frac{1}{2} q_i z_i^2[t] - (\lambda_i[t] - \alpha\lambda_i[t+1]) z_i[t] + \alpha\lambda_i[1] z_i[0] - \lambda_i[T] z_i[T]$$
$$= \sum_{t=1}^{T} \Big(b_i z_i[t] - \frac{1}{2} q_i z_i^2[t] - p_i[t] z_i[t]\Big) + p_i[0] z_i[0] = V_i(z_i, p_i). \qquad (14)$$

Thus the two different user optimal problems are equal, and we can analyze either one of them. We will use the Lagrangian version for analysis, while the p version will be used for implementation.

4.3 Optimality Conditions

The optimization problem in (2) is concave as it is the maximization of a concave cost function under affine constraints. Thus necessary and sufficient optimality conditions are given by the KKT conditions (see [Boyd and Vandenberghe, 2004])

$$\nabla_\lambda L = 0, \quad \nabla_u L = 0, \quad \nabla_z L = 0.$$

$\nabla_\lambda L = 0$ is equal to the dynamics constraint being satisfied.

For a standard LQ problem with a penalty on the input, $\nabla_u L = 0$ gives u as a function of λ. See [Cannon et al., 2008] for a slightly more general MPC case. Here we instead get the following

$$\frac{\partial L}{\partial u_i[t]} = -\lambda_{i+1}[t+1] + \alpha\lambda_i[t+2] = 0, \quad 0 \le t \le T-1 \tag{15a}$$

$$\frac{\partial L}{\partial u_i[T-1]} = -\lambda_{i+1}[T] = 0. \tag{15b}$$

Note that it is due to the lack of penalty on u that $\nabla_u L$ is independent of u.

Next we study $\nabla_z L$. Normally this allows us to solve for λ given z, going backwards in time. Calculating the gradients gives

$$\frac{\partial L}{\partial z_i[t]} = b_i - q_i z_i[t] + \alpha\lambda_i[t+1] - \lambda_i[t] \quad 1 \le t \le T-1 \tag{16a}$$

$$\frac{\partial L}{\partial z_i[T]} = b_i - q_i z_i[T] - \lambda_i[T]. \tag{16b}$$

Combining the two optimality conditions, we get the following lemma.

LEMMA 1

The optimal inventory level z satisfies

$$z_i[t] = \frac{\alpha q_{i-1}}{q_i} z_i[t+1] \tag{17}$$

for $i \ge 2$ and $t \le T-1$. □

Proof Using (16a) and (15a) gives for $t \le T-2$

$$\begin{aligned} z_i[t] &= \frac{\alpha\lambda_i[t+1] - \lambda_i[t] + b_i}{q_i} \\ &= \frac{\alpha q_{i-1}}{q_i} \frac{\alpha\lambda_{i-1}[t+2] - \lambda_{i-1}[t+1] + b_{i-1}}{q_{i-1}} + \frac{b_i - \alpha b_{i-1}}{q_i} \\ &= \frac{\alpha q_{i-1}}{q_i} z_{i-1}[t+1] \end{aligned}$$

where we have used that $\alpha b_{i-1} = b_i$. The case for $t = T-1$ follows similarly. □

5. Proof of Theorem 1 and Proposition 1

We are now ready to prove Theorem 1 and Proposition 1.

Proof of Theorem 1: For every input u_{i-1}, $i \leq N$, we have from the dynamics that

$$z_i[t+1] = \alpha(z_i[t] + u_i[t]) - u_{i-1}[t] \Rightarrow u_{i-1}[t] = \alpha(z_i[t] + u_i[t]) - z_i[t+1].$$

Using (16a–b) we get for $t \leq T-2$, that the optimal u_{i-1} must satisfy

$$u_{i-1}[t] = \alpha(z_i[t] + u_i[t]) + \frac{\alpha\lambda_i[t+2] - \lambda_i[t+1] + b_i}{q_i}$$

and for $t = T-1$,

$$u_{i-1}[T-1] = z_{i+1}[T-1] + u_i[T-1] + \frac{-\lambda_i[T] + b_i}{q_i}.$$

Thus with the relation between p and λ as defined in (13) we have that the optimal u is given by (11). The expressions for p in (10) follows from Proposition 2 and Lemma 5 (see the appendix).

Proof of Proposition 1: As the nodes choices of levels has no effect on the prices, the optimal level from the nodes perspective must satisfy

$$0 = \frac{\partial \hat{V}_i}{\partial z_i[t]} = \frac{\partial L}{\partial z_i[t]}.$$

This must also hold for the social optimum, thus proving (i).

Furthermore we see that choosing the social optimum inventory levels are better than choosing $z_i[t] = \alpha^t z_i[0]$, as it is not a minimizer of V_i. Thus proving (ii).

The sum of all the payments are

$$-\sum_{i=1}^{N}\sum_{t=1}^{T} \lambda_i[t]\Big(z_i[t] - \alpha z_i[t-1]\Big). \tag{18}$$

Using that

$$z_i[t] - \alpha z_i[t-1] = -u_{i-1}[t-1] + \alpha u_i[t-2],$$

The sum in (18) can be rewritten as

$$\sum_{i=1}^{N-1}\left(\sum_{t=0}^{T-2}\Big(-\lambda_{i+1}[t+1] + \alpha\lambda_i[t+2]\Big)u_i[t] - \lambda_{i+1}[T]u_i[T-1]\right).$$

This is equal to zero, since $\lambda_{i+1}[t+1] = \alpha\lambda_i[t+2]$ and $\lambda_i[T] = 0$ for $i \geq 2$. Thus proving (iii).

6. Conclusions

We have considered an optimal control problem for a simple transportation network from the social and user perspective. By solving the social problem using a Lagrange multiplier approach, we gave an implementation of the feedback law in terms of local prices and local states that allows for a distributed implementation. Furthermore, these prices aligned the two problem in a budget neutral way so that the nodes are never worse off than if they had been on their own.

A. Appendix

In the appendix we will derive the optimal Lagrange multipliers $\lambda[t_0+\tau]$ in terms of $z[t_0-1]$ and $u[t_0-2]$. We will show that each λ can be found as a sum of the the corresponding node levels in Lemma 2. These node levels can in turn be found by studying a time shifted aggregate level as shown in Lemma 3. This shifted aggregate can then be written in terms of a non shifted aggregate at t_0-1 in Lemma 4.

Lemma 2
The optimal Lagrange multipliers are given by

$$\lambda_i[t_0] = \sum_{t=t_0}^{T} \alpha^{t-t_0}\big(b_i - q_i z_i[t]\big).$$ □

Proof We have from (16) that $\lambda_i[T] = b_i - q_i z_i[T]$ and $\lambda_i[t] = b_i - q_i z_i[t] + \alpha\lambda_i[t+1]$. From that the lemma follows trivially. □

Next we show how each node level can be written in terms of a time shifted level vector.

Lemma 3
The optimal inventory levels satisfy

$$z_i[t_0+k] = \begin{cases} \dfrac{\gamma_{i+k}}{\alpha^k q_i} \displaystyle\sum_{j=1}^{i+k} \dfrac{z_j[t_0+k+(i-j)]}{\alpha^{k+i-j}} & i+k \le N \\ \dfrac{\gamma_N}{\alpha^{N-i} q_i} \displaystyle\sum_{j=1}^{N} \dfrac{z_j[t_0+k+(i-j)]}{\alpha^{N-j}} & i+k > N. \end{cases} \tag{19}$$ □

Proof We start by showing the lemma for $k = 0$. Using (17) gives

$$\begin{aligned}
z_2[t] &= \frac{\alpha q_1}{q_2} z_1[t+1] \Rightarrow \\
(1 + \frac{\alpha^2 q_1}{q_2}) z_2[t] &= \frac{\alpha^2 q_1}{q_2} \left(\frac{z_1[t+1]}{\alpha} + z_2[t] \right) \Rightarrow \\
z_2[t] &= \frac{\alpha^2 \gamma_1}{q_2 + \alpha^2 \gamma_1} \left(\frac{z_1[t+1]}{\alpha} + z_2[t] \right).
\end{aligned}$$

Now assume that (19) holds for $i-1$ and $k = 0$. Then using (17) again gives

$$\begin{aligned}
z_i[t] &= \frac{\alpha q_{i-1}}{q_i} z_{(i-1)}[t+1] \\
&= \frac{\alpha q_{i-1}}{q_i} \frac{\gamma_{i-2}}{q_{i-1} + \gamma_{i-2}} \left(\sum_{j=1}^{i-1} \frac{z_j[t + ((i-1) - j)]}{\alpha^{(i-1)-j}} \right)
\end{aligned}$$

Which gives that

$$\left(1 + \frac{\alpha^2 \gamma_{i-1}}{q_i} \right) z_i = \frac{\alpha^2 \gamma_{i-1}}{q_i} \left(\sum_{j=1}^{i} \frac{z_j[t + (i-j)]}{\alpha^{i-j}} \right)$$

From which it follows that the lemma holds for $k = 0$. Now assume that the lemma holds for $k - 1$. Then if $i + k \leq N$

$$\begin{aligned}
z_i[t_0 + k] &= \frac{q_{i+1}}{\alpha q_i} z_{i+1}[t_0 + k - 1] \\
&= \frac{q_{i+1}}{\alpha q_i} \frac{\gamma_{(i+1)+(k-1)}}{\alpha^{k-1} q_{i+1}} \sum_{j=1}^{i+1+k-1} \frac{z_j[t_0 + (k-1) + ((i+1) - j)]}{\alpha^{(k-1)+(i+1)-j}} \\
&= \frac{\gamma_{i+k}}{\alpha^k q_i} \sum_{j=1}^{i+k} \frac{z_j[t_0 + k + (i-j)]}{\alpha^{k+i-j}}
\end{aligned}$$

For $i + k > N$ define $\hat{k}$ and $\hat{t}_0$ so that

$$\begin{aligned}
i + \hat{k} &= N \\
\hat{t}_0 + \hat{k} &= t_0 + k.
\end{aligned} \tag{20}$$

Then using that $z_i[t_0 + k] = z_i[\hat{t}_0 + \hat{k}]$ gives the second part. □

Finally, we will show that the time shifted level vector can be written in terms of $z[t_0 - 1]$.

LEMMA 4
The optimal z for (2) satisfies for $i+k \le N$:

$$\sum_{j=1}^{i+k} \frac{z_j[t_0+k+(i-j)]}{\alpha^{k+i-j}} = \alpha \sum_{j=1}^{i+k} \left(z_j[t_0-1]+u_j[t_0-2]\right)$$

and for $i+k > N$:

$$\sum_{j=1}^{N} \frac{z_j[t_0+k+(i-j)]}{\alpha^{N-j}} = \alpha^{k+i-N+1} \sum_{j=1}^{N} \Big(z_j[t_0-1]+u_j[t_0-2]\Big) \qquad \square$$

Proof We start with the first equality. Using that

$$z_j[t+n] = \alpha^{n+1}(z_j[t-1]+u_j[t-2]) \\ - \sum_{\tau=0}^{n-1} \alpha^{(n-1)-\tau} u_{j-1}[t+\tau] + \sum_{\tau=0}^{n-2} \alpha^{n-1-\tau} u_j[t+\tau],$$

we have for $i+k \le N$:

$$\sum_{j=1}^{i} \frac{z_j[t_0+k+(i-j)]}{\alpha^{k+i-j}} = \alpha \sum_{j=1}^{n} \Big(z_i[t_0-1]+u_i[t_0-2]\Big) \\ + \alpha^{-1-\tau}\Big(\sum_{j=1}^{i} \sum_{\tau=0}^{k+(i-j)-2} u_j[t_0+\tau] - \sum_{j=1}^{i} \sum_{\tau=0}^{k+(i-j)-1} u_{j-1}[t_0+\tau]\Big)$$

Since $u_0 = 0$ and $k+(i-j)-2 < 0$ for $j > i-2$ the last row equals to zero:

$$\sum_{j=1}^{i-2} \sum_{\tau=0}^{k+(i-j)-2} u_j[t+\tau] - \sum_{j=2}^{i-1} \sum_{\tau=0}^{k+(i-j)-1} u_{j-1}[t+\tau] = 0$$

For the second equality we use (20) again,

$$\sum_{j=1}^{N} \frac{z_j[t_0+k+(i-j)]}{\alpha^{N-j}} = \sum_{j=1}^{N} \frac{z_j[\hat{t}_0+\hat{k}+(i-j)]}{\alpha^{\hat{k}+i-j}} \\ = \alpha \sum_{j=1}^{N} \Big(z_j[\hat{t}_0-1]+u_j[\hat{t}_0-2]\Big) \\ = \alpha^{k-(N-i)+1} \sum_{j=1}^{N} \Big(z_j[t_0-1]+u_j[t_0-2]\Big) \qquad \square$$

Where we have used that $\hat{t}_0 - t_0 = k - \hat{k} = k-(N-i)$ and that the system is closed.

We also need the following lemma, which shows that there exist a boundary effect in the optimal controller that makes some of the states locally optimal.

Lemma 5
The optimal inventory levels satisfy

$$z_i[t] = \frac{b_i}{q_i} \quad \forall t \geq T - (i-2),\ i \geq 2. \qquad \square$$

Proof We start by showing the lemma for $i = 2$. As $u_1[T-1]$ only affects $z_2[T]$, the optimal value corresponds to maximizing the local utility, so that $z_2[T] = b_2/q_2$. Thus

$$u_1[T-1] = -\frac{b_2}{q_2} + \alpha(z_2[T-1] + u_2[T-2])$$

and $z_2[T] = b_2/q_2$, independent of all other $u_i[t]$.

Now assume that the lemma holds for all $i \leq n$. Then $u_i[t]$ only needs to consider z_{i+1} for all $t \geq T - i$. Thus the optimal $u_n[t]$ satisfies

$$u_n[t] = -\frac{b_{n+1}}{q_{n+1}} + \alpha(z_{n+1}[t] + u_{n+1}[t-1])$$

$\forall t \geq T - n$ and

$$z_{n+1}[t] = \frac{b_{i+1}}{q_{i+1}} \quad \forall t \geq T - (n-1).$$

Thus the Lemma holds for all n. $\square$

We are now ready to state the following proposition, which gives expressions for the optimal λ.

Proposition 2
Let $m = \min(T - t_0 - (i-1), N)$ and

$$\Xi_i(t_0, \tau) = \alpha^{1-\tau}\Bigg(\sum_{j=i+\tau}^{m} \gamma_j \sum_{k=1}^{j} z_k[t_0 - 1] + u_k[t_0 - 2] + \sum_{j=N+1}^{T-t_0+1} \gamma_N \alpha^{2(j-N)} \sum_{k=1}^{N} z_k[t_0 - 1] + u_k[t_0 - 2] \Bigg)$$

Then the optimal λ's are given by

$$\lambda_i[t_0 + \tau] = \sum_{t=t_0+\tau}^{T-(i-1)} \alpha^{t-(t_0+\tau)} b_i - \Xi_i(t_0, \tau) \qquad \square$$

Proof From Lemmas 2 and 5 we have that

$$\lambda_i[t_0+\tau] = \alpha^{-\tau} \sum_{k=\tau}^{T-t_0-(i-1)} \alpha^k \Big(b_i - q_i z_i[t_0+k] \Big)$$

Combining Lemmas 3 and 4 gives that

$$\alpha^k q_i z_i[t_0+k] = \begin{cases} \alpha\gamma_{i+k} \sum_{j=1}^{i+k} z_j[t_0-1] + u_j[t_0-2] & i+k \leq N \\ \alpha\gamma_N \alpha^{2(k+i-N)} \sum_{j=1}^{N} z_j[t_0-1] + u_j[t_0-2] & i+k > N \end{cases}$$

Which gives that

$$\begin{aligned} \alpha^{-\tau} \sum_{k=\tau}^{T-t_0-(i-1)} \alpha^k q_i z_i[t_0+k] &= \alpha^{1-\tau} \Big(\sum_{j=i+\tau}^{m} \gamma_j \sum_{k=1}^{j} z_k[t_0-1] + u_k[t_0-2] \\ &+ \sum_{j=N+1}^{T-t_0+1} \gamma_N \alpha^{2(j-N)} \sum_{k=1}^{N} z_k[t_0-1] + u_k[t_0-2] \Big) \end{aligned}$$

□

References

Anderson, J., J. C. Doyle, S. H. Low, and N. Matni (2019). “System level synthesis”. *Annual Reviews in Control.*

Bamieh, B., F. Paganini, and M. A. Dahleh (2002). “Distributed control of spatially invariant systems”. *IEEE Transactions on automatic control* **47**:7, pp. 1091–1107.

Boyd, S. and L. Vandenberghe (2004). *Convex optimization*. Cambridge university press.

Cannon, M., W. Liao, and B. Kouvaritakis (2008). “Efficient MPC optimization using Pontryagin’s minimum principle”. *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal* **18**:8, pp. 831–844.

Carpentier, P. and G. Cohen (1993). “Applied mathematics in water supply network management”. *Automatica* **29**:5, pp. 1215–1250.

Cohen, G. (1978). “Optimization by decomposition and coordination: a unified approach”. *IEEE Transactions on Automatic Control* **23**:2, pp. 222–232.

Giselsson, P., M. D. Doan, T. Keviczky, B. De Schutter, and A. Rantzer (2013). “Accelerated gradient methods and dual decomposition in distributed model predictive control”. *Automatica* **49**:3, pp. 829–833.

Heyden, M., R. Pates, and A. Rantzer (2018). “A structured linear quadratic controller for transportation problems”. In: *2018 European Control Conference (ECC)*. IEEE, pp. 1654–1659.

Jafarian, M., J. Scherpen, and M. Aiello (2016). “A price-based approach for voltage regulation and power loss minimization in power distribution networks”. In: *2016 IEEE 55th Conference on Decision and Control (CDC)*, pp. 680–685.

Jokić, A., M. Lazar, and P. P. van den Bosch (2009). “Real-time control of power systems using nodal prices”. *International Journal of Electrical Power & Energy Systems* **31**:9, pp. 522–530.

Kelly, F. P., A. K. Maulloo, and D. K. Tan (1998). “Rate control for communication networks: shadow prices, proportional fairness and stability”. *Journal of the Operational Research society* **49**:3, pp. 237–252.

Low, S. H., F. Paganini, and J. C. Doyle (2002). “Internet congestion control”. *IEEE Control Systems Magazine* **22**:1, pp. 28–43.

Radner, R. (1962). “Team decision problems”. *The Annals of Mathematical Statistics* **33**:3, pp. 857–881.

Rotkowitz, M. and S. Lall (2006). “A characterization of convex problems in decentralized control”. *IEEE Transactions on Automatic Control* **51**:2, pp. 274–286.

Shah, P. and P. A. Parrilo (2013). “H_2-optimal decentralized control over posets: a state-space solution for state-feedback”. *IEEE Transactions on Automatic Control* **58**:12, pp. 3084–3096.