Distributed Control Design for Large-Scale Interconnected Systems

Cédric Langbort\textsuperscript{a} \quad Raffaello D’Andrea\textsuperscript{b}

Ramu S. Chandra\textsuperscript{c}

\textsuperscript{a}. Center for the Mathematics of Information, California Institute of Technology, 1200 E. California Blvd, MS 136-93, Pasadena CA 91125. Email: clangbort@ist.caltech.edu

\textsuperscript{b}. Mechanical & Aerospace Engineering, Cornell University, Ithaca NY, 14853. Email: rd28@cornell.edu

\textsuperscript{c}. General Electric Research. Email: chandrar@research.ge.com
Introduction

In 1974, in his guest editorial for a special issue of the IEEE Transactions on Automatic Control [1], Richard Bellman emphasized the need for “theories of large systems”. “This is particularly so”, he added, “since society is composed of large systems”.

Loosely speaking (we will give a more precise definition later), a large-scale (interconnected) system is one that is composed of numerous subunits which are dynamically coupled and/or exchanging information with each other. Examples abound in our “interconnected age”, where the dependence of our lives on very large integrated microchips, power grids and communication networks (to name just a few) has both made Bellman’s statement more obvious and his call for adapted control tools even more pressing than thirty years ago. As this special issue illustrates, the field has indeed made much progress to keep up with these technological developments and has in fact reached a point where it is impossible to give an exhaustive survey of the literature.

Our goal, in this paper, is to present a set of tools which, although based on quite recent developments in robust control theory, allow to perform control de-
sign of complex networks in the spirit of Bellman’s ideas. In particular, our framework hinges on the concept of convex relaxations and provides a control design methodology that blends nicely with computation and can thus handle the numerous degrees of freedom, structural constraints, and uncertainties inherent to large-scale systems.

To emphasize our self-proclaimed affiliation with Bellman’s program, we found it fitting to structure our paper around some of its quotes and use them as section headers. In order to keep the presentation accessible to a wide audience, we are only presenting the main ideas of the theory. Most of the mathematical details and proofs can be found in [2]-[3]-[4].

“A precise way of handling imprecise systems”

The first step of any control design procedure is modeling. As already pointed out in Bellman’s paper, models of large-scale interconnected systems are especially subject to uncertainties. This is not only due to an accumulation of subsystem-level uncertainties but also to the very nature of the control-oriented modeling process. Indeed, for the dimension of the full system’s state space to be reason-
able and its description to be practical (in the sense that the corresponding data does not take so much space in a computer’s memory as to render any calculation hopeless), we typically cannot use the most accurate models of the subsystems and must rely on reduced order models. The result is uncertainty on the actual dynamics of the system, which must be taken into account for control design, if one wants to ensure appropriate behavior on the true system.

Besides, even if we could model every subsystem perfectly, our description of the full system would still involve some uncertainty, as a result of our poor knowledge of the coupling mechanisms at play between these units. For example, in the case of cooperative vehicles transmitting their position to teammates over a wireless network, we know that the information received by each vehicle will be degraded (because of quantization and possible delay-inducing congestion in the network) but the exact characteristics of this degradation depend on many hard-to evaluate and/or time-varying factors. It is thus safer, for the purposes of control design, to treat these degradations as uncertainties (by, say, assuming that the true time-varying transmission delay belongs to an interval) and, again, ensure that the obtained control law is robust to them.
Finally, there is a last type of impreciseness which is both specific and essential to large-scale interconnected systems. Roughly speaking, it lies in the fact that different units may know different things about the system, with different degrees of precision and that, even if they have information regarding the same elements, their beliefs may be incompatible. This segmentation of information can also be modeled as uncertainty. For example, if a car on an automated highway only knows the state of the road at its current position but not at the location where it is going, and if it believes that the truck in front of it has a functional braking system, even though the truck has detected a problem with the brakes, it will have to either acquire this new information or, if impossible, be cautious when deciding on its next move and hence, use a robust control law.

The framework we propose to handle these three types of uncertainties is that of robust control theory and Linear Fractional Representations (LFR) [5]. It is interesting to note that those methods do not use randomness as a model of imprecision, which is also what Bellman advocated in his text.
Modeling

Following the discussion of the previous section, we decide to model large-scale systems as follows. Consider a network or undirected graph with a dynamical system sitting at each of its $N$ vertices. The $i^{th}$ such subsystem is represented by a multi-input/multi-output (MIMO) mapping $G_i$ satisfying

\[
\begin{pmatrix}
  z_i \\
  w_i
\end{pmatrix} = G_i \begin{pmatrix}
  d_i \\
  v_i
\end{pmatrix},
\]

where the vector-valued signals $d_i$ and $z_i$ designate a local disturbance and performance output, respectively. Signals $v_i$ and $w_i$ correspond to the aggregate coupling signals used by the subsystem, namely

\[
v_i = \begin{pmatrix}
  v_{i1} \\
  \vdots \\
  v_{iN}
\end{pmatrix} \quad \text{and} \quad w_i = \begin{pmatrix}
  w_{i1} \\
  \vdots \\
  w_{iN}
\end{pmatrix},
\]

where $v_{ij}$ (respectively $w_{ij}$) is the input received by (respectively, output sent to) subsystem $j$, with the convention that those vector-valued signals are of dimension zero if there is no link edge joining subsystems $i$ and $j$. We will designate the set of all disturbances and performance outputs by $\{d_i\}$ and $\{z_i\}$, respectively.

More generally, when considering a variable $M_i$ indexed over a set $\mathcal{I}$, the notation $\{M_i\}$ will designate the set of all the values of $M_i$, when index $i$ varies in $\mathcal{I}$, while
\{M_i\}_{i \in J} will be the set of all values of \(M_i\) when \(i\) varies in a subset \(J\) of \(I\).

We refer to Figure 1 for a depiction of such large-scale systems. Later in this paper, we will focus on linear time-invariant (LTI) subsystems for which the map \(G_i\) admits a state space representation of the form

\[
\dot{x}_i = A_i x_i + B_i v_i + B^d_i d_i \\
 w_i = C_i x_i + D_i v_i + D^d_i d_i \\
z_i = C^z_i x_i + D^z_i v_i + D^{zd}_i d_i.
\]

As mentioned earlier, uncertainties can enter the description of the system in several ways. For example, the coupling relation between subsystems is typically poorly known, which can be captured by writing

\[ v_{ij} = \Delta_{ij} w_{ji} \quad (1) \]

for all pairs of vertices \((i, j)\) joined by an edge. In (1), \(\Delta_{ij}\) is only known to belong to a set of input-output operators characterized, for example, by some upper-bound on their norm. By adding self-loops to the original graph (that is, edges corresponding to pairs of vertices of the form \((i, i)\)), one can also handle uncertainties at the subsystem level in this fashion. Regrouping all the uncertainties together on the one hand and all the subsystems’ input/output maps on the
Figure 1: A model of large-scale system with $N = 5$ subsystems. Each subsystem has a disturbance input $d_i$, a performance output $z_i$ and shares coupling signals $v_{ij}$ and $w_{ij}$ with its neighbors.
other hand, we obtain two block-diagonal operators $\Delta$ and $G$ which can be used to represent the whole system as the feedback interconnection.

Such linear fractional representations (so-called because the resulting input/output map from $\{d_i\}$ to $\{z_i\}$ is a rational function of $\Delta$) have been proposed by several authors as models of large-scale systems (see, for example, [6]-[7]) and are at the heart of the design method we are about to present. The main elements of this method were first introduced in [8].

**Stability and Performance**

Once we have a model, the next natural question that arises for interconnections depicted in Figure 1 is that of stability. How can one ensure that, in the absence of disturbances, the state of every subsystem will go to zero asymptotically in time? Then, if this is the case, how can we ensure satisfactory disturbance rejection in the sense that

$$\sum_i \int_0^\infty \|z_i(t)\|^2 dt < \gamma \sum_i \int_0^\infty \|d_i(t)\|^2 dt$$

for some constant $\gamma$?

In the context of LFR, one typically answers such questions by resorting to
robust stability tools, such as the celebrated (Scaled) Small Gain Theorem [10] or Integral Quadratic Constraints methods [11]. However, when particularizing them to large-scale systems, these stability tests should be implementable with no centralized knowledge of the subsystems’ characteristics. In other words, we want to be able to infer stability of the full system from the properties of individual subsystems, with no single one precisely knowing what its neighbors are. One basic idea to achieve this goal can be exposed mathematically as follows.

Assume that for each subsystem $i$, there exists a positive function $V_i$ of state $x_i$, which vanishes at the origin, and functions $P_{ij}$ of the interconnection variables $v_{ij}$, $w_{ij}$ such that

$$\frac{d}{dt} V_i(x_i(t)) < \sum_j P_{ij}(v_{ij}(t), w_{ij}(t)) \text{ for all } t$$

(2a)

$$v_{ij} = \Delta_{ij} w_{ji} \text{ for all } i, j \Rightarrow \sum_i \sum_j P_{ij}(v_{ij}(t), w_{ij}(t)) = 0,$$

(2b)

then the system is stable, with $\sum_i V_i$ as a Lyapunov function. In essence, conditions (2) are a dissipativity result, in the sense of Jan Willems [9]. Each $V_i$ can be thought of as an energy stored in each system while, for any input $v_{ij}$ and output $w_{ij}$, $P_{ij}(v_{ij}, w_{ij})$ corresponds to a (generalized) power received by subsystem $i$ from subsystem $j$. Then, condition (2a) means that part of the power received by
Subsystem $i$ from its environment is degraded while condition (2b) means that the interconnection conserves power. It is then clear that, if both hold, energy is dissipated overall, driving the system to the origin and thus proving stability. When dealing with linear time-invariant subsystems as defined before and quadratic supply rates and storage functions, conditions (2) can be checked by solving a set of linear matrix inequalities (LMIs), as (2a) is equivalent to

$$\begin{pmatrix}
A_i & B_i & B_i^d \\
I & 0 & 0 \\
C_i & D_i & D_i^d \\
0 & I & 0 \\
C_i^z & D_i^z & D_i^{zd} \\
0 & 0 & I
\end{pmatrix}^t
\begin{pmatrix}
0 & X_i & 0 & 0 & 0 & 0 \\
X_i & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & Z_{i11} & Z_{i12} & 0 & 0 \\
0 & 0 & (Z_{i12}^t) & Z_{i22} & 0 & 0 \\
0 & 0 & 0 & 0 & I & 0 \\
0 & 0 & 0 & 0 & 0 & -I
\end{pmatrix}
\begin{pmatrix}
A_i & B_i & B_i^d \\
I & 0 & 0 \\
C_i & D_i & D_i^d \\
0 & I & 0 \\
C_i^z & D_i^z & D_i^{zd} \\
0 & 0 & I
\end{pmatrix} < 0$$

for all $i$, where

$$Z_{i11} := -\text{diag}_{1 \leq j \leq N} X_{ij}$$

$$Z_{i22} := \text{diag}_{1 \leq j \leq N} (X_{ji})$$

$$Z_{i12} := \text{diag}_{1 \leq j \leq N} \left(-\text{diag}_{i \leq j \leq N} Y_{ij}, \text{diag}_{i < j \leq N} Y_{ji}^t\right).$$

LMIs are particular instances of semi-definite programs (SDPs). The goal is to minimize a linear function of the matrix variables, subject to the constraint that
some other linear function of the variables is a positive-definite matrix. In (3),
the variables are \( \{X_i\} \) and \( \{X_{ij}\} \), \( \{Y_{ij}\} \) and the constraint is in fact written as
a negative-definiteness condition, using the symbol ‘≺’. This formulation is of
course equivalent since a matrix \( M \) is positive definite (that is \( M ≻ 0 \)) if and only
if \( -M ≺ 0 \). Also, (3) could be turned into an optimization problem since \( M ≺ 0 \)
if and only if the smallest \( t ≥ 0 \) such that \( M - tI ≺ 0 \) is 0.

What makes SDPs particularly attractive, and explains why so much effort
has been devoted recently to characterizing classes of problems in controls and
engineering-at-large that could be re-written in this form (see, for example, [12]),
is the fact that there exist efficient algorithms to approximately solve them in
polynomial time on a computer.

“Approximate policies will play an important role”

The next natural step following system analysis is controller synthesis to guaran-
tee stability and disturbance rejection. For large-scale systems, this process first
requires to choose an architecture that is compatible with the spatially distributed
nature of the problem. Many choices are possible but, for illustration purposes,
we will consider the three types of control laws depicted in Figure 2:

(a) a **centralized controller**, where a single controller receives the measurements of all subsystems and determines all the control actions at once,

(b) a *(fully)* **decentralized controller**, where a unit is assigned to each of the plant’s subsystems, providing it with control action based solely on local measurements, and

(c) a **distributed controller**, where these controlling units can share information with some of their neighbors over a network.

The main difference between these architectures does not lie so much in the physical interconnections and/or routing procedures that their implementation requires, as in the various *information patterns* available to the controllers that is, the specification of what is known and measured by each controlling unit. For example, a centralized controller could in fact be implemented just like the distributed controller of Figure 2.(c), assuming that the links between controller’s subsystems transmit all the received measurements in a flooding-like manner. However, dis-
Figure 2: Three possible control architectures. (a). A centralized controller, (b). a (fully) decentralized controller, and (c). a distributed controller.
tributed controllers are not centralized because each of the controller’s subsystem only has a model of the corresponding plant’s subsystem and must thus compute its inputs using incomplete information.

Therefore, our choice of a particular control law should not only depend on the perceived amount of information that needs to be exchanged to implement it (the number of edges in the interconnection graph of the controller, according to which the decentralized architecture would be most desirable) but also on the possibility of practically synthesizing it according to its information pattern, as the size of the whole system increases. From this viewpoint, a centralized controller disqualifies itself naturally as its design amounts to solving a MIMO control problem with a large number of inputs, outputs and states; a situation where current optimization-based methods are known to perform poorly if structure is not taken into account at the numerical level.

The decentralized and distributed information patterns, on the other hand, can be tackled for very large systems using the dissipativity conditions exposed in the previous section. If each of the plant’s subsystem can be rendered dissipative through interconnection with a controlling subsystem (designed knowing solely
this particular subsystem) and the supply rates satisfy condition (2b), then the resulting closed-loop will be stable and reject perturbations appropriately.

Controllers can be computed by solving a synthesis problem for each subsystem, with coupling only occurring through the closed-loop supply rates. When subsystems are LTI and storage and supply rates are quadratic, decentralized and distributed control design amount to solving a set of bilinear matrix inequalities (BMIs). These BMIs result from the application of conditions (3) to a yet unknown subsystem which depends linearly on the controller’s parameters. As is well-known in the field of robust control, solving arbitrary BMIs is an NP-hard problem.

It turns out that the BMIs corresponding to distributed (dynamic output feedback) controllers is in fact equivalent to a set of LMIs with additional variables, while those stemming from decentralized control design do not have this property. In loose terms, the reason for this difference is that, in the distributed case, we have enough degrees of freedom in the closed-loop supply rate matrices and the signals exchanged between subsystems to remove the non-convexity of the problem.
In summary, what we have seen is that some information patterns give rise to tractable control design problems while others do not. In fact, strictly speaking, we have only shown the (in)tractability of our method for certain patterns. However, various other results, such as [14]-[13], suggest that decentralized control is indeed intractable and, as a result, that distributed architecture should be preferred.

“Computers will play an important part”

The last ingredient in Bellman’s program was computation. The exact way in which computers were supposed to enter the picture was unclear but it is already remarkable that, thirty years ago, he was hinting in the direction that closed-form analytic solutions (or even the concept of optimality) were probably not the most appropriate ones for controlling large systems, and that control laws would somehow have to be designed with (as opposed to merely implemented on) computers. Interestingly, this vision does not seem to have been universally accepted even today, as was recalled recently in [15].
We have already commented on some of the computation-oriented aspects of modern robust control methods in general and of our tools in particular. The control law is obtained as the output of a tractable optimization problem that can be solved in polynomial time. The desire of turning our design problem into such a computationally solvable one is what prompted us to use a distributed architecture for the controller. However, the story does not end here.

Since the models describing large-scale systems are themselves of large dimension, a control design algorithm that terminates in polynomial time in this parameter may still be too slow to be of practical use. One way to speed up the computations consists in running them in parallel, by assigning a processor to each of the subsystems, and having it run all the calculations pertaining to that subsystem. Here is a possible way to perform such a parallelization, based on decomposition ideas from convex optimization that exploit the natural structure of the problem. As we explained before, a satisfactory controller with distributed architecture can be constructed by solving a set of coupled LMIs, which can be
re-written in the following form

Find symmetric matrices $X_i \succ 0$ and $X_{ij}$, $i, j = 1...N$

such that $L_i(X_i, \{X_{ij}\}_{j \in N_i}) \prec 0$ for all $i$ \hspace{1cm} (4)

$X_{ij} = -X_{ji}$.

In LMI (4), each map $L_i$ takes its values in the space of symmetric matrices and, as before, $M \prec 0$ means that matrix $M$ is negative definite. In the case of analysis, variables $X_i$ simply correspond to candidate storage functions while each $X_{ij}$ corresponds to a candidate supply rate for edge $(i, j)$. For the synthesis LMIs, the correspondence is not as transparent. In both cases, however, there is one LMI for each subsystem involving both private ($X_i$, which is known solely to subsystem $i$) and shared ($\{X_{ij}\}$) variables, and two LMIs are coupled if and only if the corresponding subsystems can exchange information, that is if and only if $j$ belongs to the neighborhood $N_i$ of $i$. In other words, the semi-definite program has inherited the structure of the plant.

To exploit this structure for parallel computing, we first convert (4) to an opt-
mization problem as follows

\[
\min_{\{t_i, \{X_i\}, \{X_{ij}\}\}} \sum_i t_i
\]

subject to \( L_i(\mathcal{X}_i, \{\mathcal{X}_{ij}\}_{j \in \mathcal{N}_i}) \leq t_i I \), for all \( i \)

\[
\mathcal{X}_i > 0 ; \ t_i \geq -1 , \text{ for all } i
\]

\[
\mathcal{X}_{ji} = -\mathcal{X}_{ij} , \text{ for all } i, j.
\]

Problem (4) is feasible if and only if all the optimal \( t'_i \)s for problem (5) are negative.

Since this new problem is convex in all its variables, we can perform the optimization one variable at a time, which means that its optimum is given by

\[
\min_{\mathcal{X}_{ij} = -\mathcal{X}_{ji}} \sum_i \phi_i(\{\mathcal{X}_{ij}\}_{j \in \mathcal{N}_i})
\]

where each function \( \phi_i(\{\mathcal{X}_{ij}\}_{j \in \mathcal{N}_i}) \) is defined as the optimal value of the following problem

\[
\min_{\mathcal{X}_i, t_i} t_i
\]

subject to \( L_i(\mathcal{X}_i, \{\mathcal{X}_{ij}\}_{j \in \mathcal{N}_i}) \leq t_i I \)

\[
\mathcal{X}_i > 0 ; \ t_i \geq -1.
\]

Hence, all we have to do to solve our original LMI is minimize the function \( \Phi = \sum_{i=1}^{N} \phi_i \) and check the value of each \( \phi_i \) at optimum.
We cannot perform this optimization using the familiar gradient descent method because function $\Phi$ is not differentiable. Nevertheless, we can build on its convexity and use subgradients instead (see sidebar). Explaining all the details and properties of subgradient-based optimization algorithms would take us too far afield and we refer the interested reader to [12]-[16]. For our purposes, we only need to point out the following facts.

a. For an appropriate choice of positive scalars $\alpha_k$, the update rule

$$
\mathcal{X}^{(k+1)}_{ij} = \mathcal{X}^k_{ij} + \alpha_k g_k \text{ for all } i, j
$$

$$
\Phi^{k+1}_{\text{best}} = \min \left( \Phi^k_{\text{best}}, \Phi(\{\mathcal{X}^{(k+1)}_{ij}\}) \right)
$$

converges to the minimum of $\Phi$, where $g_k$ is a subgradient of $\Phi$ at the point $\{\mathcal{X}^k_{ij}\}$.

b. Subgradient $g_k$ can be computed by combining subgradients of functions $\phi_i$.

c. Subgradients of functions $\phi_i$ are naturally obtained as part of the output of any primal-dual solver (like the popular SeDuMi) solving problem (6).

This suggests the following simple distributed algorithm to minimize $\Phi$ and, hence, to solve LMI (4):
1. Given a value for \( \{X_{ij}^0\} \) and \( \Phi_{best}^0 \).

2. Compute \( \phi_i(\{X_{ij}\}_{j \in \mathcal{N}_i}) \) for all \( i \) and a corresponding sub-gradient (fact c).

3. All subsystems transmit their subgradient to their neighbors and use the received information to compute the relevant components of \( g_k \) (fact b).

4. All \( X_{ij}'s \) are updated using those subgradients (fact a).

5. Iterate until no progress is made.

This algorithm converges to the minimum of \( \Phi \) while requiring only limited communication between processors. In practice, however, the subgradients of the various functions \( \phi_i \)'s will not be available at the same time since computing them involves solving SDP (6), with a different map \( L_i \), and hence different convergence time, for each subsystem. As explained in [12]-[17], the basic steps outlined above can be modified to accommodate such an asynchronous framework. A typical result is that convergence to the minimum of \( \Phi \) is still guaranteed as long as all subgradients are used infinitely often. Overall, the proposed algorithm is thus
very well-suited for practical distributed implementation and has been shown to perform well in cases for systems too large to be handled in a centralized fashion [16].

**Extensions and Connections**

The main analysis and synthesis ideas presented above can be adapted to take advantage of additional structure in large-scale systems, yielding both less conservative and even more computationally tractable results.

Consider for example the large-scale system of Figure 3, where all subsystems are identical and connected only to their nearest neighbors on a ring. Such models occur naturally in the context of semi-discretized partial differential equations or regular arrays with periodic boundary conditions. In that case, the large-scale system is spatially invariant and we can use this symmetry to our advantage.

If we restrict ourselves to checking stability with identical storage functions
Figure 3: A spatially invariant large scale systems (a) and the corresponding closed-loop (b). All subunits are identical and interconnected to their neighbors in an identical manner. Control design is simplified by exploiting this symmetry. The obtained controller is distributed, with the same interconnection structure as the plant.
and supply rates for all systems, that is

\[ X_i = X \text{ and } X_{i(i+1)} = - \begin{bmatrix} 0 & I \\ I & 0 \end{bmatrix} X_{i(i-1)} \begin{bmatrix} 0 & I \\ I & 0 \end{bmatrix} \text{ for all } i, \]

LMIs (3) collapse to a single one, irrespective of the number of units in the whole system. The same reduction also occurs for distributed controller synthesis and allows to design a controller with the same spatial structure as the plant with a relatively small computational burden.

In fact, an equivalent LMI can also be derived using multidimensional system ideas and algebraic transformations as described in [2]. This latter approach has the additional benefit of addressing spatially invariant systems over general groups, and always results in a single, computationally attractive, analysis and synthesis LMI, even in the limit of an infinite number of subsystems. In [4], it was shown how better closed-loop performance can be guaranteed by using additional properties of the group and aggregating subsystems according to central subgroups. The result is a hierarchical controller, with one unit assigned to every element of the quotient group.

Another nice feature of those spatially invariant design tools is that they can
also be used to tackle some finite extent system with boundary conditions, as the ones pictured in Figure 4. In this figure, all blocks represent identical sub-systems except for the end ones which stand for boundary conditions. If some relation, that we call spatial reversibility, exists between these boundary conditions and the state space representation of the subsystem, it can be shown that the system of Figure 4 behaves as if embedded in an infinite, spatially invariant one. This property is analogous to the so-called method of images in partial differential equations theory, which states that solutions to Laplace’s or the heat equation on certain bounded domains can be obtained as solutions of the same equation on an infinite region, once mirror-image singularities are introduced. For example, consider the heat equation over a finite bar with insulated ends. If we model the bar as the interval \([0, 1]\) and assume that it is subject to a heat distribution \(Q(x)\), then the temperature profile over \([0, 1]\) is the same as for an infinite bar with heat distribution

\[
\tilde{Q}(x) = \begin{cases} 
Q(x) & \text{if } x \in (k, k + 1], k \text{ even} \\
-Q(x) & \text{if } x \in (k, k + 1], k \text{ odd}.
\end{cases}
\]

For spatially reversible finite extent systems, a controller can be designed by applying the method of [2] to the extended spatially invariant system and trun-
Figure 4: Large-scale system with boundary conditions. (a). If the sub-unit has some particular properties, this system behaves as if embedded in the periodic system of Figure 3 and the same, simpler, synthesis methods can be used, (b). The same reduction can be used for 2-D system as well.
cating the obtained control law. Again, the main benefit of this approach is the simplicity of the obtained LMI. Details on this approach can be found in [18].

An Example

As an illustration of our tools, we present an application to the control of a power grid with failing transmission lines. The main objectives are to ensure stability of the grid in the face of such failures, and to give some guarantees on the level of performance degradation if, in addition, some of the generators experience disturbances. In other words, we are trying to avoid a blackout. Although our presentation has so far only focused on continuous-time dynamics, we chose to treat this problem in the discrete-time framework, as a way to demonstrate a possible extension of the approach. Details can be found in [20].

Following [19], we consider the network of Figure 5, consisting of \( N \) interconnected load-driving generators. The dynamics of the \( i^{th} \) generator is given by

\[
x_i(k+1) = R_i x_i(k) + L_i I_i(k) + B_i u_i(k)
\]

where, at each time \( k \geq 0 \), vector \( I_i(k) \) (respectively \( V_i(k) \)) contains the real
and imaginary part of the current (respectively voltage) deviations from a chosen operating point, $u_i(k)$ is the control torque applied to regulate the generator, and vector $x_i(k)$ is the state of the generator, corresponding to the deviations from the rotor’s angular velocity’s and angle’s reference profile.

Each generator is connected to a load with admittance matrix $Y_{ii}$ and to other generators through a transmission line of varying admittance $\Delta_{ij}(k)Y_{ij}, j = 1...N$. We assume that the functions $\Delta_{ij}$, which model failure, are taking their values in $\{0, 1\}$ but that other characteristics are unknown.

Using Kirchhoff’s laws, we derive that

$$I_i(k) = Y_{ii}V_i(k) + \sum_{j \neq i} \Delta_{ij}(k)Y_{ij}(V_i(k) - V_j(k))$$

and, in turn, that

$$x_i(k+1) = (R_i + L_iY_{ii}K_i)x_i + L_i\sum_{j \neq i} \Delta_{ij}(k)Y_{ij}(K_ix_i(k) - K_jx_j(k)) + B_i^u u_i \quad (8)$$

for all $i = 1...N$.

Introducing a disturbance $d_i$, on the control torque, a measured output $y_i = x_i$ and a performance output $z_i^t = \frac{1}{10} \begin{bmatrix} x_i^t & u_i^t \end{bmatrix}$, for each generator, equations (8) can be captured by an LFR as described above, with each subsystem being an LTI
Figure 5: (a). A power network with $N = 3$ generators; (b). The corresponding model including multiple self-loops.
discrete-time system with

\[ A_i := (R_i + L_i Y_{ii} K_i); \quad B_i := \begin{bmatrix} L_i & \ldots & L_i \end{bmatrix} \; 4(N-1) \quad ; \quad B_i^d := B_i^u \]

and coupling signals satisfying

\[ w_{ij} = -Y_{ij} K_i x_i \text{ for } i \neq j ; \quad w_{ii} = \begin{bmatrix} Y_{1i} \\ \vdots \\ Y_{(i-1)i} \\ Y_{(i+1)i} \\ \vdots \\ Y_{Ni} \end{bmatrix} K_i x_i. \]

The interconnection graph has an edge for every pair of connected generators, along with multiple self-loops adjacent to every vertex (one loop for every neighbor of the corresponding generator).

We applied our synthesis methods to model (8), for the case of \( N = 10 \) generators interconnected over the complete graph and with the following data

\[ R_i = \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & 0 \end{bmatrix}, \quad L_i = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \]

\[ B_i^u = \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \quad K_i = \begin{bmatrix} 0 & 0 \\ 0 & -1 \end{bmatrix}. \]
Our LMIs produced a stabilizing distributed controller with a guaranteed upper-bound on closed-loop performance of \( \gamma = 3.6 \). Other methods, like [21], assume a stochastic model for the failures and give necessary and sufficient SDP conditions for the existence of a stabilizing controller with some specified bound on performance. However, they require adding an LMI for each of the possible failure modes which, in our case, would involve \( 2^{45} \) different cases, assuming that failures are independent! Our approach, while admittedly more conservative both in the representation of the failures and on the guaranteed upper-bound, results in practically tractable computations.

**Conclusions and New vistas**

With its ability to incorporate inherent uncertainties and its computationally attractive synthesis methods, modern robust control theory has gone a long way in answering some of Bellman’s questions regarding control of large-scale systems.
However, there is one aspect of the problem which has remained mostly untouched and, to again use Bellman’s words, constitutes “a good field [...] because it has not been worked on before”. While we now know how to design controllers for a large family of networks and respect their underlying, pre-existing topology, we are still missing the tools for an integrated theory of networks, where topology and control systems can be designed jointly. Could this be a challenge for the next thirty years?

A Side Bar: Subgradients

Subgradients generalize the concept of gradient for non-differentiable, convex functions.

Let \( f : U \to \mathbb{R} \) be a real-valued convex function defined over an open convex subset of \( \mathbb{R}^n \). A vector \( v \in \mathbb{R}^n \) is called a subgradient of \( f \) at \( x \) if, for any \( y \in U \)

\[
f(y) - f(x) \geq v.(y - x),
\]

where ‘.’ denotes the usual scalar product. The set of all subgradients of \( f \) at \( x \) is called the subdifferential at \( x \).
In words, the subdifferential at $x$ corresponds to the set of hyperplanes which, passing through $(x, f(x))$, leave the graph of $f$ entirely in one of the two half-spaces they define. For example, if $f$ is the ‘absolute value’ function $x \mapsto |x|$, the subdifferential at 0 is the interval $[-1, 1]$, while the only subgradient at $x > 0$ (respectively $x < 0$) is 1 (respectively $-1$).

It can be shown that, for convex functions, the subdifferential is always a non-empty, convex, compact set at every point. Also $f$ has a local minimum at a point $x$ if and only if the zero vector belongs to the subdifferential at $x$.

References


