Chapter 16
Partitioning Approaches for Large-Scale Water Transport Networks

Carlos Ocampo-Martínez and Vicenç Puig

16.1 Introduction

Large-scale systems (LSS) present control theory with new challenges due to the large size of the plant and of its model [13, 22]. The goal to be achieved with control methods for this kind of systems is to obtain a reasonable solution with a reasonable effort in modelling, designing and implementing the controller.

As discussed in previous chapters, MPC has been proved to be suitably applied for the control of LSS as drinking-water networks [3], sewer networks [14], open-flow channel networks [18] or electrical networks [15]. Nevertheless, the main hurdle for MPC control (as any other control technique), when applied to LSS in a centralized way, is the non-scalability. The reason is that a huge control model is needed, being difficult to maintain/update and which needs to be rebuilt on every change of the system configuration, e.g., when some part of the system should be stopped because of maintenance actions or malfunctions. Subsequently, a model change would require re-tuning the centralized controller. It is obvious that the cost of setting up and maintaining the monolithic solution of the control problem is prohibitive. A way of circumventing these issues might be by looking into *decentralized* MPC (DMPC) or *distributed* MPC techniques, where networked local MPC controllers are in charge of controlling part of the entire system. The main difference between distributed and decentralized MPC is that the former *uses* negotiations and re-computations of local control actions within the sampling period to increase the level of cooperation, whereas the latter does not (at the benefit of computation time, but at the cost of optimality).

The industrial success of the traditional centralized MPC (CMPC) drives now a new interest in this old area of distributed control, and distributed MPC has become one of the hottest topics in process control in the early twenty-first century, worldwide. Thus, two research projects (HDMPC [10] and WIDE [24]) are currently being

C. Ocampo-Martínez (✉)
Institut de Robòtica i Informàtica Industrial, CSIC-UPC, Barcelona, Spain
e-mail: cocampo@iri.upc.edu

V. Puig
Research Center Supervision, Safety and Automatic Control (CS2AC-UPC), Terrassa, Spain

© Springer International Publishing AG 2017
V. Puig et al. (eds.), *Real-Time Monitoring and Operational Control of Drinking-Water Systems*, Advances in Industrial Control,
DOI 10.1007/978-3-319-50751-4_16

carried out in Europe, both focused on the development of decentralized and distributed MPC techniques. Few works have been recently published in this area; see, e.g., [6, 11, 16, 19, 20, 23], among others.

However, in order to apply decentralized or distributed MPC approaches to LSS, there is a prior problem to be solved: the system decomposition into subsystems. The importance of this issue has already been noticed in classic control books addressing the decentralized control of LSS as [13, 22]. The decomposition of the system into subsystems could be carried out during the modelling of the process by identifying subsystems as parts of the system on the basis of physical insight, intuition or experience. But, when a large-scale complex system with many states, inputs and outputs is considered, it may be difficult, even impossible, to obtain partitions by physical reasoning. A more appealing alternative is to develop systematic methods, which can be used to decompose a given system by extracting information from its structure and representing it as a graph. Then, this structural information can be analysed by using methods coming from graph theory. Consequently, the problem of system decomposition into subsystems leads to the problem of graph partitioning, i.e., the decomposition of graph into subgraphs.

Graph partitioning is an important problem with extensive application in scientific computing [12], optimization, very large-scale integration (VLSI) design [8], task partitioning for parallel processing, control of cascading failures, among others. However, the development of graph partitioning algorithms that allow the decomposition of LSS into subsystems for being used in decentralized or distributed MPC is still very incipient and available methods are quite limited. In [22], a hierarchical LBT decomposition that leads to a input-reachable hierarchy for some particular systems is presented. A more general approach is based on the ε-decomposition method, which is based on decomposing the system in weakly coupled subsystems (see also [22]). The algorithm proceeds sequentially disconnecting the edges of the system graph that are smaller than a prescribed threshold ε and identifying the disconnected subgraph of the resulting graph. The obtained subsystems correspond to the subsystems with mutual coupling smaller or equal than ε. However, the tuning of this parameter is not a trivial issue and only a trial and error approach is currently available.

16.2 Problem Statement

A graph can be defined as an abstract representation of a set of objects from a certain collection, where some pairs of objects are connected by links. The interconnected elements are typically called *vertices* while the connection links are called *edges*. These latter elements may be *directed* (asymmetric) or *undirected* (symmetric) according to their connection features, what makes that the whole graph is directed or undirected as well. It is also possible to distinguish graphs whether or not their vertices and edges are weighted (weighted/unweighted graphs).

Consider a dynamical system represented in general form by the state-space equations

$$\mathbf{x}(k+1) = \mathbf{g}(\mathbf{x}(k), \mathbf{u}(k), \mathbf{d}(k)), \tag{16.1a}$$

$$\mathbf{y} = \mathbf{h}(\mathbf{x}(k), \mathbf{u}(k), \mathbf{d}(k)), \tag{16.1b}$$

where $\mathbf{x}(k) \in \mathbb{R}^{n_x}$ and $\mathbf{x}(k+1) \in \mathbb{R}^{n_x}$ are, respectively, the current and successor system states in discrete time, $\mathbf{u} \in \mathbb{R}^{n_u}$ is the system input, $\mathbf{y} \in \mathbb{R}^{n_y}$ is the system output and $\mathbf{d}\imath\mathbb{R}^{n_d}$ is a bounded process disturbance. Moreover, $\mathbf{g} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_d} \rightarrow \mathbb{R}^{n_x}$ is the states mapping function and $\mathbf{h} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_d} \rightarrow \mathbb{R}^{n_y}$ corresponds with the output mapping function. Suppose now that it is desired to decompose (16.1) into subsystems. With this aim, the graph representation of the system model (16.1) is determined (by using the system topology) and incidence matrix $\mathbf{B}_{ij}$ is then stated, which describes the connections (edges) between the graph vertices (system inputs, outputs and states). Without loss of generality, $\mathbf{B}_{ij}$ and the directionality of the edges are derived from the relation between system equations (rows of $\mathbf{B}_{ij}$) and system variables (columns of I_M), as proposed by [22, 25, 26]. There are alternative matrix representations for a (directed) graph such as the *adjacency matrix* and the *Laplacian matrix* (see [2]), which are related to the matrix representation used in this paper. Once $\mathbf{B}_{ij}$ has been obtained from the system directed graph (digraph), the problem of the decomposition into subsystems can be formulated in terms of partitioning the corresponding graph into subgraphs. Since such partitioning is oriented to the application of a decentralized control strategy (in particular, DMPC), the resultant subgraphs should have the following features (see [13, 22]):

- nearly the same number of vertices;
- few connections between the subgraphs.

These features guarantee that the obtained subgraphs have a similar size which balances computations between subsystem controllers and allows minimizing communications between them. Hence, the problem of graph partitioning can be more formally established as follows:

Problem 16.1 (*Standard Graph Partitioning*) Given a graph $G(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ denotes the set of vertices, $\mathcal{E}$ is the set of edges and $M \in \mathbb{Z}_{\geq 1}$, find M subsets $\mathcal{V}_1$, $\mathcal{V}_2, \ldots, \mathcal{V}_M$ of $\mathcal{V}$ such that

1. $\bigcup_{i=1}^{M} \mathcal{V}_i = \mathcal{V}$,
2. $\mathcal{V}_i \cap \mathcal{V}_j = \emptyset$, for $i \in \{1, 2, \ldots, M\}$, $j \in \{1, 2, \ldots, M\}$, $i \neq j$,
3. $|\mathcal{V}_1| \approx |\mathcal{V}_2| \approx \cdots \approx |\mathcal{V}_M|$,
4. the *cut size*, i.e., the number of edges with endpoints in different subsets $\mathcal{V}_i$, is minimized.

Remark 16.1 Defining the *vertex-based weight* of a subset $\mathcal{V}_i$ as

$$\Omega_i \triangleq \sum_{j=1}^{|\mathcal{V}_i|} \omega_i^j, \tag{16.2}$$

where ω_i^j corresponds to the weight of the j-th vertex of the subset $\mathcal{V}_i$, the following condition should be added to Problem 16.1 in the case of *weighted graph partitioning*:

- $\Omega_i \approx \Omega/M$, with $i \in \{1, 2, \ldots, M\}$, where

$$\Omega \triangleq \sum_{i=1}^{M} \Omega_i. \tag{16.3}$$

Remark 16.2 Conditions 3 and 4 of Problem 16.1 are of high interest from the decentralized control point of view since they are related to the degree of interconnection between resultant subsystems and their size balance, respectively.

Graph partitioning is considered as a $\mathcal{NP}$-complete problem [22]. However, it can be solved in polynomial time for $|\mathcal{V}_i| = 2$ (Kernighan-Lin algorithm) [4, 7]. Since this condition is quite restrictive for large-scale graphs, alternatives for graph partitioning based on fundamental heuristics are properly accepted. Two main classes of successful heuristics have evolved over the years, trying to achieve the proper trade-off between partitioning speed and quality. They are the *minimum degree-based* ordering algorithms (MDB) and the *graph partitioning-based* ordering algorithms (GPB) [9].

16.3 Proposed Approaches

16.3.1 Using Graph Theory

This approach consists in proposing a partitioning algorithm, as much automatized as possible, through which a partition of a dynamical system can be found, which allows its decomposition in subsystems. This algorithm requires to represent the dynamical system as a graph, which can be obtained from the system structure [22].

Main Algorithm

The partitioning algorithm proposed in this chapter follows some ideas developed in [9] for graph partitioning purposes. However, some refining steps have been added as well as some of the original procedures have been drastically changed in order to find partitions oriented to split dynamical networked systems. Hence, the different parts/routines of the main proposed algorithm are presented and explained in sections below. The current version of the algorithm is thought to be used offline, i.e., the partitioning of the system is not carried out online. A further improvement could be to adapt the proposed algorithm such that the partitioning could be done online when some structural change of the network occurs. In this way, the potential benefit of using a DMPC approach described in the Introduction could be fully exploited.

Start-up: This procedure requires the definition of the graph, i.e., *the incidence matrix*[1] $\mathbf{B}_{ij}$, which describes the connections between the graph vertices, their directionality and, in some cases, the weight of each edge.

Preliminary partitioning: This procedure performs a preliminary automatic partitioning of the graph as follows. The vertex $v_j \in \mathcal{V}$, for $j \in \{1, 2, \dots, |\mathcal{V}|\}$, with maximum weight ω is found and defined as the centre of the first subgraph G_1. Then, all vertices connected to this vertex of maximum weight are assigned to G_1. At this point, the set of non-selected vertices is defined as

$$\mathcal{V}_r \triangleq \{v_j \in \mathcal{V} : v_j \notin \mathcal{V}_1\}.$$

This procedure is now repeated for all vertices $v_j \in \mathcal{V}_r$ (now for $j = \{1, 2, \dots, |\mathcal{V}_r|\}$) until $\mathcal{V}_r$ is empty, after the corresponding updating. This routine highlights the subgraphs of higher connectivity. The resultant subgraphs with just one vertex are merged to the closest subgraph. Once a set of subgraphs $G_i(\mathcal{V}_i, \mathcal{V}_i)$, for $i = 1, 2, \dots, M$, is obtained, it is possible to determine some useful indexes for the entire graph and each one of the resultant subgraphs. These indexes are as follows:

- $\varphi_i \triangleq |\mathcal{V}_i|$ (from now on called *subgraph internal weight* of G_i);
- ε_i, denoted as the *cut size*[2] of the subgraph G_i (from now on called *subgraph external weight* of G_i);
- $\varphi^{\max} \triangleq \max\limits_i \varphi_i$, for $i = 1, 2, \dots, M$;
- $\bar{\varphi} \triangleq \frac{1}{M} \sum\limits_{i=1}^{M} \varphi_i$ (arithmetic mean).

Notice that at this stage, the number M of subgraphs is obtained in an automatic way so it is not imposed.

Remark 16.3 Notice that introducing the set $\tilde{\mathcal{E}}_a \subset \mathcal{E}$, defined as the set of edges with endpoints in other subgraphs different to G_a, the representation of subgraphs G_i such that

[1]The *incidence matrix* of a directed graph $G(\mathcal{V}, \mathcal{E})$, denoted as $\mathbf{B}_{ij}$, is defined such that

$$\mathbf{B}_{ij} = \begin{cases} -1 & \text{if the edge } z_j \text{ leaves vertex } v_i, \\ 1 & \text{if the edge } z_j \text{ enters vertex } v_i, \\ 0 & \text{otherwise.} \end{cases}$$

This matrix has dimensions $\varphi \times \eta_e$, where φ corresponds with the total number of vertices and η_e denotes de total number of edges [2]. Additionally, the weight of the j-th vertex, denoted as ω^j, for $j = 1, 2, \dots, \varphi$, where $\varphi \triangleq |\mathcal{V}|$, is computed. The weight ω^j represents the number of edges connected to this vertex. Moreover, ω^j is also known as the *vertex degree* [5].

[2]See Problem 16.1.

$$\bigcup_{i=1}^{M} G_i = G,$$

can be slightly modified to $G_i(\mathcal{V}_i, \mathcal{E}_i, \tilde{\mathcal{E}}_i)$ for completeness purposes. Also notice that $\varepsilon_i \triangleq |\tilde{\mathcal{E}}_i|$.

Uncoarsening—Internal balance: This procedure aims at the reduction of the number of subgraphs, trying to achieve similar internal weights for all of them. This process starts determining the set

$$\mathcal{L} = \{G_i, i = 1, 2, \ldots, m : \varphi_i \leq \bar{\varphi}\}, \tag{16.4}$$

with $m \in \mathbb{Z}_+$ and $m < M$. For each $G_i \in \mathcal{L}$, the set of neighbour[3] subgraphs, denoted as $\mathcal{N}_i$, is determined and expressed as

$$\mathcal{N}_i = \{G_j,\ j = 1, 2, \ldots, h_i : G_j \text{ is neighbour of } G_i\}, \tag{16.5}$$

with $h_i = |\mathcal{L}_i|$. If the condition

$$\varphi_i + \varphi_j \leq \bar{\varphi}, \quad i \in \{1, 2, \ldots, m\},\ j \in \{1, 2, \ldots, h_i\} \tag{16.6}$$

holds for $G_i \in \mathcal{L}$ and $G_j \in \mathcal{N}_i$, then these two subgraphs are merged. If there are two or more subgraphs $G_j \in \mathcal{N}_i$ such that (16.6) holds, the subgraph $G_j \in \mathcal{N}_i$ with minimum internal weight is selected. Once two subgraphs are merged, $\bar{\varphi}$ is updated.
This procedure is iterated until no additional merging was possible. It is considered that the internal balance has been achieved when either

- $\bar{\varphi} \leq \varphi_i \leq \varphi^{\max}$, for $i = 1, 2, \ldots, M$, or
- G_i with $\varphi_i \leq \bar{\varphi}$ cannot be merged with any of its neighbours since the φ associated with the resultant subgraph might be greater than $\varphi^{\max}$.

Refining—External balance: This procedure aims at the reduction of the cut size of the resultant subgraphs. To achieve this goal, define ω_i^j as the degree of the j-th vertex of the i-th subgraph, with $j \in \{1, 2, \ldots, \varphi_i\}$ and $i \in \{1, 2, \ldots, M\}$. From this definition, two indexes can be stated as follows:

- the *vertex internal degree*, denoted as $\hat{\omega}_i^j$, which represents the number of connections of the vertex $v_j \in \mathcal{V}_i$, for $j \in \{1, 2, \ldots, \varphi_i\}$, $i \in \{1, 2, \ldots, M\}$, with other vertices $v_p \in \mathcal{V}_i$, $p \in \{1, 2, \ldots, \varphi_i\}$, $p \neq j$;
- the *vertex external degree*, denoted as $\breve{\omega}_i^j$, which represents the number of connections of the vertex $v_j \in \mathcal{V}_i$, for $j \in \{1, 2, \ldots, \varphi_i\}$, $i \in \{1, 2, \ldots, M\}$, with other vertices $v_p \in \mathcal{V}_q$, $p \in \{1, 2, \ldots, \varphi_q\}$, $q \in \{1, 2, \ldots, M\}$, $q \neq i$.

[3]Two subgraphs are called *neighbours* if they are contiguous and share edges (see, e.g., [1] among many others).

Algorithm 16.1 Graph partitioning algorithm

```
1: $\mathbf{B}_{ij} \leftarrow$ System topology
2: $G(\mathcal{V}, \mathcal{E}) \leftarrow \mathbf{B}_{ij}$
3: for $j = 1$ to $\varphi$ do
4:   Compute $\omega^j$
5: end for
6: $\mathcal{V}_r \leftarrow \mathcal{V}, i = 1$
7: repeat
8:   Find $v \in \mathcal{V}_r$ with maximum $\omega$
9:   $\mathcal{V}_i \leftarrow v$ and all its neighbour vertices
10:  $\mathcal{V}_r \triangleq \mathcal{V} - \left\{ \bigcup_{h=1}^{i} \mathcal{V}_h \right\}$
11:  $i = i + 1$
12: until $\mathcal{V}_r = \emptyset$
13: for $i = 1$ to $M$ do {Compute some indexes}
14:  $\varphi_i \triangleq |\mathcal{V}_i|$                  {internal weight}
15:  $\varepsilon_i \triangleq |\tilde{\mathcal{E}}_i|$                  {external weight}
16: end for
17: $\varphi^{\max} \triangleq \max_i \varphi_i$
18: $\bar{\varphi} \triangleq \frac{1}{M} \sum_{i=1}^{M} \varphi_i$                 {arithmetic mean}
19: Compute $\mathcal{L}$                       {see (16.4)}
20: $b_{\text{int}} = \text{false}$                    {Internal balance}
21: while $b_{\text{int}} = \text{false}$ do
22:  for $i = 1$ to $m$ do
23:    Compute $\mathcal{N}_i$                  {see (16.5)}
24:    for $j = 1$ to $h$ do
25:      if $\varphi_i + \varphi_j \leq \bar{\varphi}$ then {see (16.6)}
26:        $G_* = G_i \cup G_j$
27:        $G_{\text{new}} \leftarrow G_*$ with minimum $\varphi_*$
28:        Update $\bar{\varphi}$
29:      end if
30:    end for
31:  end for
32:  Update $\varphi_i$
33:  $b_{\text{ext}} = \text{false}$                   {External balance}
34:  while $b_{\text{ext}} = \text{false}$ do
35:    for $i = 1$ to $M$ do
36:      for $j = 1$ to $\varphi_i$ do
37:        Compute $\hat{\omega}_i^j$ and $\breve{\omega}_i^j$
38:        if $\hat{\omega}_i^j < \breve{\omega}_i^j$ then
39:          Move $v^j$ from $G_i$ to its neighbour
40:        end if
41:        Update $\varphi_i, \bar{\varphi}, \varphi^{\max}$
42:      end for
43:    end for
44:    Update all indexes
45:    Check external balance (nodes)
46:  end while
47:  Check internal balance (subgraphs)
48: end while
49: return $\mathcal{P}$                        {see (16.7)}
```

Hence, for a given vertex $v_j \in \mathcal{V}_i$, if $\hat{\omega}_i^j < \breve{\omega}_i^j$, then vertex v_j is moved from subgraph $G_i(\mathcal{V}_i, \mathcal{E}_i, \tilde{\mathcal{E}}_i)$ to the subgraph in which most of its edges have their endpoint (like in the AVL tree algorithm [5]). All indexes should be updated for the M subgraphs and the next vertex is analysed. This procedure will last until each subgraph vertex fulfills $\hat{\omega}_i^j \geq \breve{\omega}_i^j$.

The Complete Algorithm: Algorithm 16.1 collects all the procedures/routines mentioned and explained before. Hence, applying this algorithm to the graph associated wit a given dynamical system, the expected result consists of a set of subgraphs which determines a particular system decomposition. This set $\mathcal{P}$ is then defined as

$$\mathcal{P} = \left\{ G_i,\ i = 1, 2, \ldots, M\ :\ \bigcup_{i=1}^{M} G_i = G \right\}. \tag{16.7}$$

Auxiliary Routines

Despite Algorithm 16.1 yields an automatic partitioning of a given graph, it does not imply that the resultant set P follows the pre-established requirements stated in Problem 16.1. In this sense, complementary routines can be useful for improving the partitioning process according to the considered application. Additional auxiliary routines could be added such that the generated partitioning takes into account the control performance that would be achieved when used in decentralized or distributed MPC control.

Prefiltering: In general, the resultant solution given by the Algorithm 16.1 is nearly appropriate in terms of $\hat{\omega}$ and $\breve{\omega}$, but it highly depends on the topology and complexity of the graph. For this reason, in order to obtain a better graph partitioning, sometimes it can be useful to make a *Prefiltering* routine, where all the vertexes with $\omega = 1$ are virtually merged to this vertex that shares its unique edge. This procedure creates *supranodes*, which should be properly recognized at the moment of determining the partitioning of the dynamical system from the decomposition of its associated graph. Moreover, doing the manual merging of those vertices reduces the work done by subsequent routines.

Post-filtering: On the other hand, suppose that after partitioning a given graph $G(\mathcal{V}, \mathcal{E})$ by using Algorithm 16.1, all the M resultant subgraphs fulfil

$$\bar{\varphi} \leq \varphi_i \leq \varphi^{\max}, \quad \text{for} \quad i \in \{1, 2, \ldots, M\}. \tag{16.8}$$

However, the following situation could occur. Suppose a subgraph G_a with $\varphi_a \ll \bar{\varphi}$, which is placed next to a subgraph G_b and fulfills (16.8). The merging of subgraphs G_a and G_b, expressed as $G_c \triangleq G_a \cup G_b$, is not allowed since $\varphi_c \geq \varphi_{\max}$. The *post-filtering* routine implements an approximation and a parameterization, i.e., by adding a small tolerance δ, the existence of the resultant subgraph G_c is now allowed since $\varphi_c \leq \varphi^{\max} + \delta$. This relaxation allows to have less subgraphs but with higher complexity and internal weight.

Anti-oscillation: This procedure leads to solve a possible issue when the *refining (external balance)* routine is run. When a vertex is moved from one subgraph to another according to its internal and external degrees, there exists the possibility of doing this movement during an infinite time if there is no specification of routine ending. Therefore, the refining routine is then run within a for loop and the parameter ρ is set as the maximum number of iterations that this procedure is executed. Afterwards, since the resulting set of subgraphs is stored at each iteration $t' \in \mathbb{Z}_+$, $t' = \{1, 2, \dots, \rho\}$, the configuration of M subgraphs with minor ε_i, for $i = 1, 2, \dots, M$, can be chosen.

Some Practical Issues
Given that the partitioning algorithm proposed in this chapter is mainly thought for performing decentralized control of LSS, several features could be taken into account to achieve a convenient system partitioning and less complex controller designs. For instance, an additional routine that would restrict the connection of subgraphs with unidirectional edges would be very useful since a pure hierarchical control scheme can be straightforwardly implemented, decreasing the inherent loss of performance of a decentralized control scheme.

16.3.2 Using Masks

The application of DMPC to WTN depends crucially on how the network is decomposed into subsystems. Identifying subsystems is not an easy task in a large-scale network as it involves to find automatically *sufficiently small* sections of the networked plant that are not *too coupled* among them. The partitioning algorithm, proposed in this chapter, aims to obtain this decomposition automatically by identifying clusters of elements that are strongly connected with each other but weakly interconnected with the other clusters, in order to represent the whole network as a set of loosely coupled subsystems [21]. The current version of the algorithm is thought to be used offline, that is the partitioning of the system is static and is not carried out online. A further improvement could be to adapt the proposed algorithm such that the partitioning could be done online when, for instance, some structural change of the network appears.

As a starting point, the partitioning algorithm requires the following information of the WTN:

1. The interconnection structure characterized by the matrix

$$\mathbf{I_c} = \left[\, \mathbf{A}_{sp}\ \mathbf{B}_{sp} \,\right], \tag{16.9a}$$

where

$$\mathbf{A}_{sp} = \begin{bmatrix} \mathbf{A} & 0 \\ 0 & 0 \end{bmatrix}, \quad \mathbf{B}_{sp} = \begin{bmatrix} \mathbf{B} \\ \mathbf{E} \end{bmatrix}, \tag{16.9b}$$

where $\mathbf{A}$ and $\mathbf{B}$ are the system matrices in (12.8), the subscript sp identifies the matrices used for system decomposition and $\mathbf{E} \triangleq [\mathbf{E}_u \quad \mathbf{E}_d]$ is the matrix related to the equality constraints (12.9b). In order to take into account input bounds, new normalized inputs are introduced $\bar{\mathbf{u}} \triangleq \mathbf{u}/\mathbf{u}^{\max}$ so that $\bar{\mathbf{u}} \in [0, 1]$. Thus, new matrices $\bar{\mathbf{B}}$ and $\bar{\mathbf{E}}$ are introduced in (16.9b) to take into account the rescaling. From matrix $\mathbf{I_c}$, the adjacency matrix Ψ of the network graph can be obtained by replacing the nonzero elements by ones, leaving the null elements unchanged.

2. A threshold value ε is used for determining whether a term, which takes into account the actuator capacity (maximum allowable flow) and its usage frequency, has a negligible *effect* on the entire plant. In this way, the less important actuators are filtered out, in order to reduce the coupling degree of the system and identify independent subnetworks.

The partitioning algorithm proceeds by decomposing the matrix $\mathbf{I_c}$ into a set of submatrices, named as *partitions* and denoted by $\mathcal{P}_\varepsilon = \{\mathbf{I_c}_1, \ldots, \mathbf{I_c}_M\}$. Then, $\mathcal{P}_\varepsilon$ correspond to a set of subgraphs (subsystems) obtaining by deleting the edges corresponding to elements of $\mathbf{I_c}$ with magnitude no larger than ε. That is, the idea behind the partitioning approach is to neglect less important elements (i.e., links) in matrix $\mathbf{I_c}$ such that the resulting $\tilde{\mathbf{I}}_c$ is less coupled. Ideally, $\tilde{\mathbf{I}}_c$ should lead to a permutation matrix $\mathbf{P}$ such that $\mathbf{P}'\tilde{\mathbf{M}}\mathbf{P}$ is block diagonal. This procedure is repeated iteratively by reducing ε until an enough number of partitions is obtained. Algorithm 16.2 summarizes the steps of the proposed partitioning algorithm.

Partitions can be tuned by means of parameter ε of the proposed approach, which makes the user able to attempt matching the desired number and size of subsystems.

Typically, in the first iteration, Algorithm 16.2 neglects a high number of elements of $\mathbf{I_c}$, highly reducing the matrix connectivity degree and obtaining a subsystem decomposition. Then, once the sets of states/inputs relative to each partition are computed, the task of finding a suitable $\mathbf{P}$ that block-diagonalizes the matrix $\mathbf{P}'\tilde{\mathbf{M}}\mathbf{P}$ is a matter of linear algebra implementation. Every subsystem is composed by sets of state and input variables that are linked, meaning that are in the same block in the $\mathbf{P}'\tilde{\mathbf{M}}\mathbf{P}$ diagonal. Let $\mathcal{X}^i$ and $\mathcal{U}^i$ be, respectively, the sets of state and input variables assigned to subsystem i, while $|\mathcal{X}^i|$ and $|\mathcal{U}^i|$ determine the number of variables for each set. A subsystem is created if both numbers are different than zero. All state and input variables that are not assigned to any of the currently created subsystems, i.e., that do not belong to $\mathcal{X}^i$ or $\mathcal{U}^i$, respectively, are available for the next iteration. Otherwise, variables already assigned to a subsystem in the current or in a previous iteration are *masked*[4] to prevent their reassignment to other subsystem.

Then, a new iteration of the algorithm starts by decreasing ε (e.g., halving ε). Algorithm 16.2 iterates until all state variables are assigned to a subsystem. Note that the algorithm may terminate even if some inputs are not assigned to any subsystem, which is due to automatic threshold-based neglecting process. Such issue can be managed by manually including unassigned inputs to proper subsystem following engineering insight.

[4]Consider a variable to be masked when it does not belong to any set since it has already been classified in a previous iteration.

The importance of the mask arises from the structure of the algorithm. In fact, if not excluded, all previously assigned states and inputs would be part of the next iteration partition, introducing couplings and hence increasing the size of the resulting submodels. The aforementioned inclusion easily follows from the decreasing of ε among sequential iterations.

Algorithm 16.2 Automatic partitioning algorithm

```
1: Initialise masks to a neutral value
2: Initialise the sets of unassigned variables $\mathcal{X}$ and $\mathcal{U}$ with all state and input variables,
   respectively
3: Determine the number of unassigned states: $N_x = |\mathcal{X}|$;
4: Init $\varepsilon$
5: while $N_x > 1$ do
6:   Apply masks to $\mathbf{A}_{sp}$ and $\mathbf{B}_{sp}$
7:   $\mathbf{I_c} = [\mathbf{A}_{sp}\ \ \mathbf{B}_{sp}\bar{\mathbf{u}}]$
8:   For all elements of $\mathbf{I_c}$
9:   if $\mathbf{I_c}_{i,j} < \varepsilon$ then
10:     $\tilde{\mathbf{I}}_{ci,j} = 0$;
11:   else
12:     $\tilde{\mathbf{I}}_{ci,j} = 1$;
13:   end if
14:   Find $\mathbf{P}$ such that $\mathbf{P}'\tilde{\mathbf{M}}\mathbf{P}$ is block diagonal
15:   Identify parts satisfying $N_{x^i} = |\mathcal{X}^i| > 0$ and $N_{u^i} = L(\mathcal{U}^i) > 0$ and add to previous ones
16:   Update $N_x$
17:   Update masks with updated states and inputs
18:   Update $\varepsilon$
19: end while
```

Few remarks on the above algorithm:

1. At any iteration of Algorithm 16.2, the numerical value of ε is a crucial tuning knob of the approach. A guideline is that the larger is the decreasing step, the larger is the size of the obtained subsystems. Ways for automatically determining the step size are a subject of current research.
2. Matrix $\mathbf{E}$ in (16.9b) defines a constraint among actuators that can be easily taken into account if all the actuators belong to the same subsystem. Otherwise, since each controller manipulates every partition independently from the others, negotiations between controllers would be required to guarantee the fulfilment of node constraints.
3. The use of masks to prevent state reassignment avoids that submodels have overlapping states and inputs: if a state variable is used in a model by a controller, no other controller can use it. The main benefit of this choice is the very low level of coupling between partitions, but the price to pay is a potential decrease of closed-loop performance.
4. The current structure of the algorithm is unsuitable to handle state overlaps because it relies on links between elements that present different degree of coupling. Hence, once the stronger couplings are eliminated (using masking), the

weaker ones gain relative importance. State overlaps may be introduced a posteriori based on engineering insight, in order to increase the adherence with respect to the original centralized model. Handling overlapping in an automatic way is also a current research topic.

5. In some cases, even relatively small connections, i.e., capable of carrying a minor amount of water, are very important for demand satisfaction. A way of accounting for such an issue is to perform a simulation using, for instance, a CMPC controller and compute the average percentage of use for each actuator. Thus, this information could be used to weight $\bar{\mathbf{u}}$ component-wise. The main drawback of this approach is the need of (and dependence on) simulation.
6. Note that the proposed algorithm can be customized by setting different importance levels of states vs. inputs, by weighting the related components in $\mathbf{I_c}$ from its statement at (16.9a).
7. The structure of the proposed algorithm suggests that termination is achieved if the ε value is decreased at each iteration. However, at the current status of the development, the algorithm cannot guarantee any property for the resulting partitioning but the assignment of all system state variables to a subsystem.

The decomposition process of matrix $\mathbf{I_c}$ reported here is similar to the one proposed by the ε-decomposition method in [21]. The underlying idea in both cases is to disconnect those actuators corresponding to interconnections with strength smaller than the prescribed ε, identifying the disconnected subsystems. According to [21], there are s different ε-decompositions $\mathcal{P}_\varepsilon$ that can be obtained for different values of ε satisfying

$$\max_{i \neq j} |m_{ij}| = \varepsilon_1 < \varepsilon_2 < \cdots < \varepsilon_K = 0,$$

with $K \leq dim(\mathbf{I_c})$. Moreover, such decompositions are nested, that is the partitions obtained satisfy: $\mathcal{P}_{\varepsilon_1} \subset \mathcal{P}_{\varepsilon_2} \cdots \mathcal{P}_{\varepsilon_K}$ with $\mathcal{P}_{\varepsilon_1}$ being the finest and $\mathcal{P}_{\varepsilon_k}$ the coarsest. The main novelty of the algorithm presented in this chapter is the matrix normalization taking into account actuator physical/operative limits and the iterative threshold updating that allows one to take into account weaker coupling without being influenced by the stronger ones.

16.4 Simulations and Results

16.4.1 Results Using Masks-Based Approach

Using the partitioning algorithm presented in this section, the aggregate model of the Barcelona WTN is decomposed in three subsystems, as depicted in Fig. 16.1 in different colours. The resultant decomposition follows the scheme shown in Fig. 16.2, where μ_i denotes the i-th vector of shared variables among the subsystems S_j, for $j = 1, \ldots, M$. The subsystems are defined by the following elements:

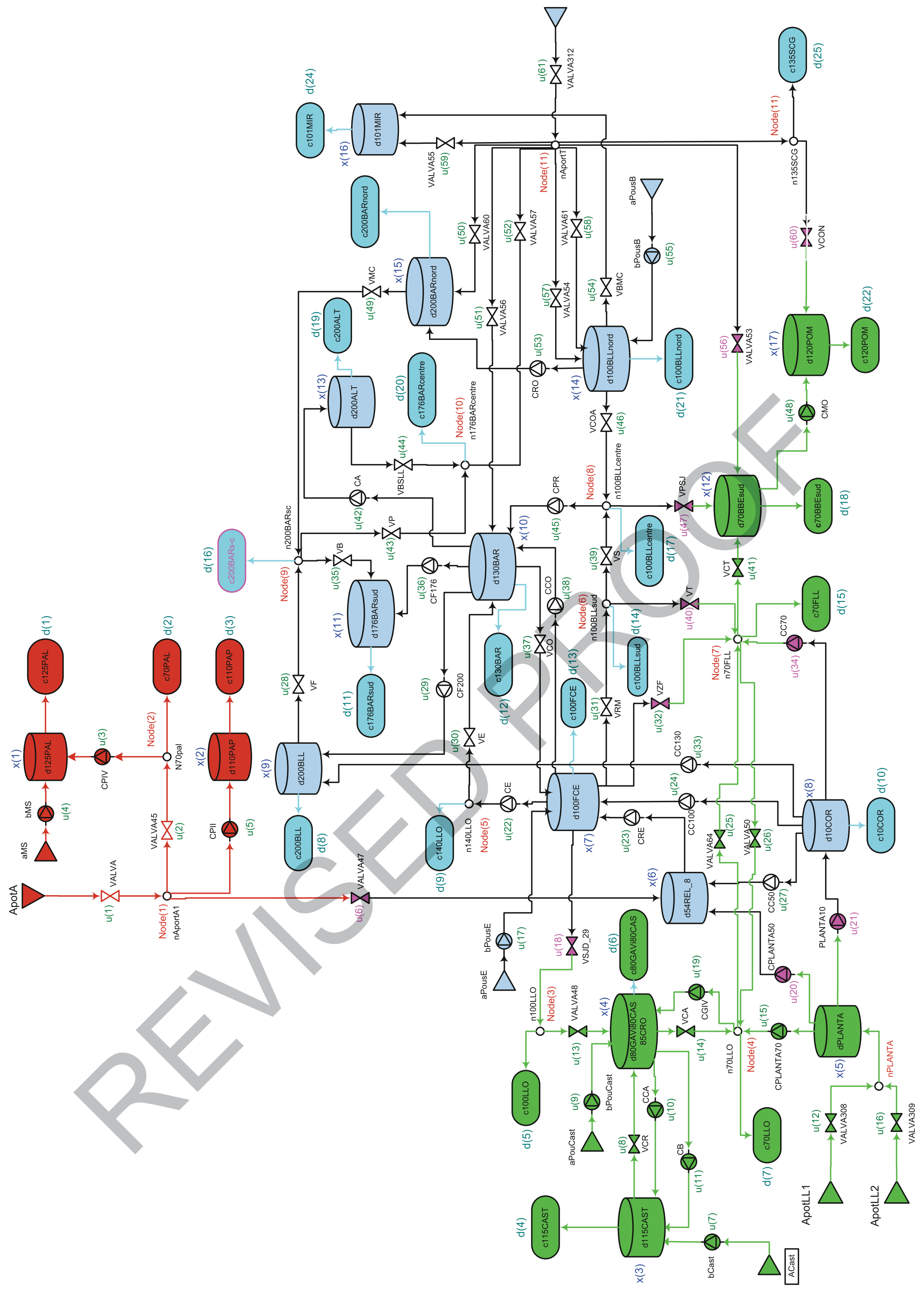

Fig. 16.1 Partition of the Barcelona WTN, aggregate model

- *Subsystem 1:* Composed by tanks x_i, $i \in \{1, 2\}$, inputs u_j, $j \in \{1 : 5\}$, demands d_l, $l \in \{1, 2, 3\}$ and nodes n_q, $q \in \{1, 2\}$. It is represented in Fig. 16.1 with red colour and corresponds to Subsystem S_1 in Fig. 16.2.

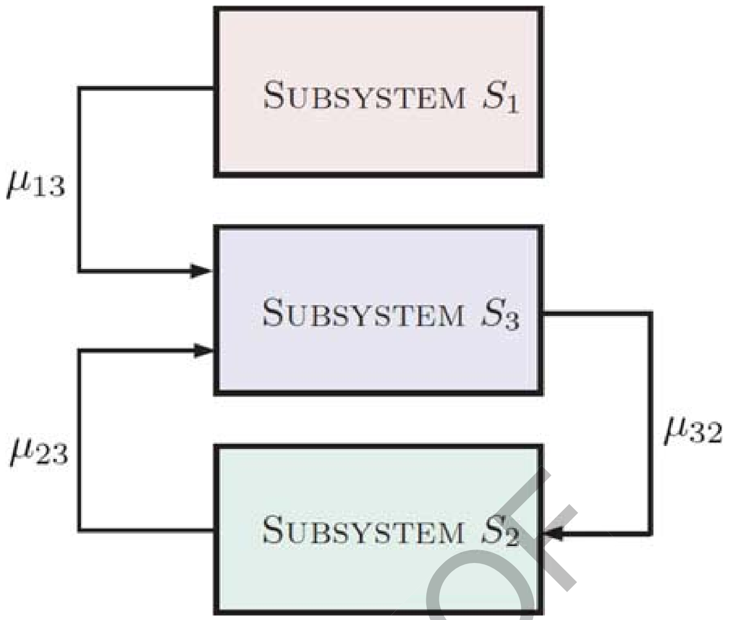

Fig. 16.2 Conceptual scheme of the partitioned Barcelona WTN

Table 16.1 Dimension comparison between the subsystems and the whole network

Elements	Subsystem 1	Subsystem 2	Subsystem 3	Whole model
Tanks	2	5	10	17
Actuators	5	22	34	61
Demands	3	7	15	25
Nodes	2	3	6	11

- *Subsystem 2:* Composed by tanks x_i, $i \in \{3, 4, 5, 12, 17\}$, inputs u_j, $j \in \{7 : 16, 18, 19, 25, 26, 32, 34, 40, 41, 47, 48, 56, 60\}$, demands d_l, $l \in \{4 : 7, 15, 18, 22\}$ and nodes n_q, $q \in \{3, 4, 7\}$. It is represented in Fig. 16.1 with green colour and corresponds to Subsystem S_2 in Fig. 16.2.
- *Subsystem 3:* Composed by tanks x_i, $i \in \{6 : 11, 13 : 16\}$, the inputs u_j, $j \in \{6, 17, 20 : 24, 27 : 31, 33, 35 : 39, 42 : 46, 49 : 55, 57, 58, 59, 61\}$, demands d_l, $l \in \{8 : 14, 16, 17, 19, 20, 21, 23, 24, 25\}$ and nodes n_q, $q \in \{5, 6, 8 : 11\}$. It is represented in Fig. 16.1 with blue colour and corresponds to Subsystem S_3 in Fig. 16.2.

Table 16.1 collects the resultant dimensions for each subsystem and the corresponding comparison with the dimensions of the vectors of variables for the entire aggregate network.

16.4.2 Results using Graph-Theory-Based Approach

This section presents the results of the application of Algorithm 14 for the partitioning of the Barcelona WTN into compositional subsystems [17]. Algorithm 16.1

and auxiliary routines presented in Sect. 16.3.1 have been designed for any system. However, some particular features should be introduced depending on the considered case study and control law in order to obtain a suitable decomposition. More precisely, the graph of the Barcelona WTN has been derived from its mathematical model expressed in the way introduced in Chap. 12, i.e.,

$$\mathbf{x}(k+1) = \mathbf{A}\,\mathbf{x}(k) + \mathbf{B_u}\,\mathbf{u}(k) + \mathbf{B_d}\,\mathbf{d}(k), \tag{16.10a}$$

$$0 = \mathbf{E_u}\,\mathbf{u}(k) + \mathbf{E_d}\,\mathbf{d}(k), \tag{16.10b}$$

under the following considerations:

- every tank, sector of consume, water source and node is considered as a vertex of the graph;
- every pump, valve and link with a sector of consume is considered as a graph edge.

In order to evaluate the partitioning results obtained from the application of Algorithm 16.1 and auxiliary routines to the Barcelona WTN, the following indexes are taken into account additionally to those previously introduced:

- $\varepsilon \triangleq \sum_{i=1}^{M} \varepsilon_i$,
- $\bar{\varepsilon} \triangleq \frac{\varepsilon}{M}$ (arithmetic mean),
- $\sigma_\varphi^2 \triangleq \frac{1}{M} \sum_{i=1}^{M} (\varphi_i - \bar{\varphi})^2$,
- $\sigma_\varepsilon^2 \triangleq \frac{1}{M} \sum_{i=1}^{M} (\varepsilon_i - \bar{\varepsilon})^2$.

Remark 16.4 Notice that although ε is not directly related to the number of shared edges between subgraphs obtained by using Algorithm 16.2, this index gives an indirect idea about their level of interconnection. Recall that the objective of the partitioning algorithm is the minimization of indexes σ_φ^2, ε and ε_i (for $i = 1, 2, \ldots, M$) to obtain a graph decomposition as less interconnected as possible and with similar number of vertices for each subgraph (internal weight).

Table 16.2 summarizes the partitioning results obtained applying Algorithm 16.1 (A1) combined with the auxiliary routine/filters presented in Sect. 16.3.1 performing the following combinations:

Table 16.2 Results for different partitioning approaches

Routine combination	M	$\bar{\varphi}$	$\bar{\varepsilon}$	σ_φ^2	σ_ε^2	ε
1	17	10.59	3.76	53.88	25.32	64
2	13	6.30	4.15	21.39	27.80	54
3	10	8.20	5.10	31.73	32.76	52
4	6	13.67	6.33	**14.88**	25.22	**38**

1. No auxiliary routines are considered.
2. A1 and prefiltering (pre-F) routine only.
3. A1 in addition to pre-F and post-filtering (post-F) routines.
4. A1 in addition to pre-F, post-F and anti-oscillation (AO) routines.

This distinction has been done in order to understand how the proposed routines affect the partitioning results.

Using only the Algorithm 16.1, the resultant partitioning $\mathcal{P}$ is comprised by 17 subgraphs. Many of them are small and cannot be merged since their neighbour subgraphs have internal weights with values quite close to $\bar{\varphi}$ (see Sect. 16.3.1). Moreover, there are several vertices with $\omega = 1$, which correspond to network water sources and demands, leading to unnecessarily difficult algorithm computations due to sizes of the resultant subgraphs (in terms of internal weight). By employing the pre-F routine, the previous problems are fixed and Algorithm 16.1 produces 13 subgraphs (see Table 16.2). Additionally, if the refining routine embedded within Algorithm 16.1 is complemented with the post-F routine, setting $\delta = 2$, a partitioning with ten subgraphs is reached.[5] Finally, if the AO routine is also considered, setting the refining limit to $\rho = 250$, a partitioning with six subgraphs is now reached. According to Table 16.2, this last partitioning (Combination 4) satisfies the minimization of the average of the internal weights for all resultant subgraphs as well as the interconnection degree between subgraph measured through ε. It is important to highlight that the proposed partitioning approach automatically determines the final number of partitions M (six for this case) when the conditions 3 and 4 of Problem 16.1 are fulfilled (see Remark 16.2). The tuning parameters δ and ρ also influence on the obtained value of M.

Notice that each subgraph of the final decomposition corresponds to a subsystem of the Barcelona WTN with the number of elements presented in Table 16.3. Figure 16.3 shows, in different colours, the obtained subsystems of Barcelona WTN.

Moreover, Fig. 16.4 schematically shows the disposition of the resultant subsystems S_i, for $i \in \{1, \ldots, 6\}$, and the sets μ_{ij} of shared links between the network subsystems corresponding to the control inputs u (manipulated flows), whose directionality is defined from S_i to S_j for $j \in \{1, \ldots, 6\}, i \neq j$. Table 16.4 collects the number of control inputs of each set μ_{ij}.

Table 16.3 Dimension comparison of the WTN subsystems

Subsystem	Tanks	Actuators	Demands	Nodes
1	13	36	20	5
2	11	11	11	0
3	13	22	20	3
4	9	16	12	2
5	6	10	8	2
6	15	26	17	3
Total	67	121	88	15

[5]Notice that increasing the parameter δ implies that $\sigma^2_{\varepsilon_i}$ becomes bigger.

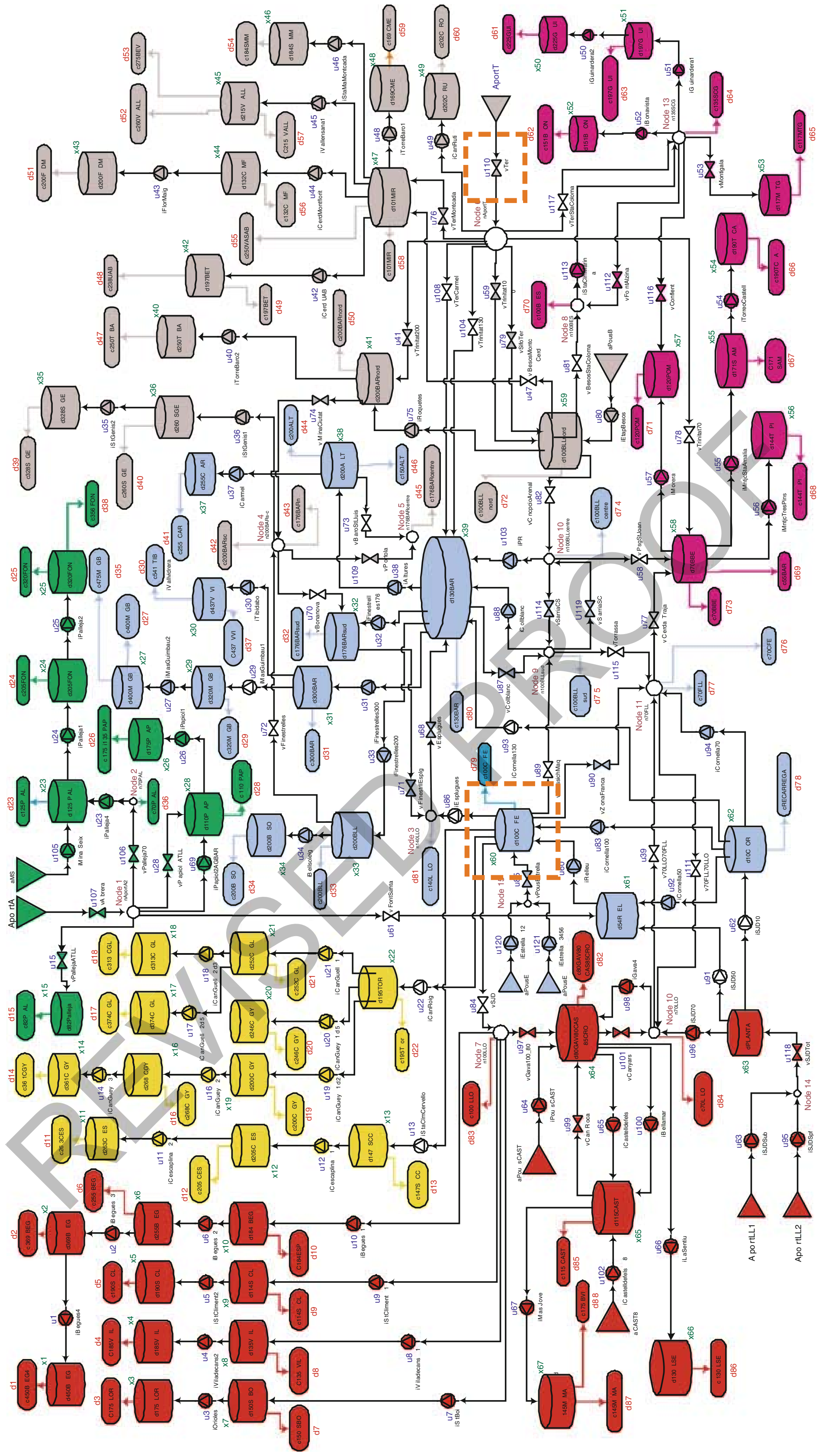

Fig. 16.3 Definitive partition of the Barcelona WTN. The *key* elements are properly featured

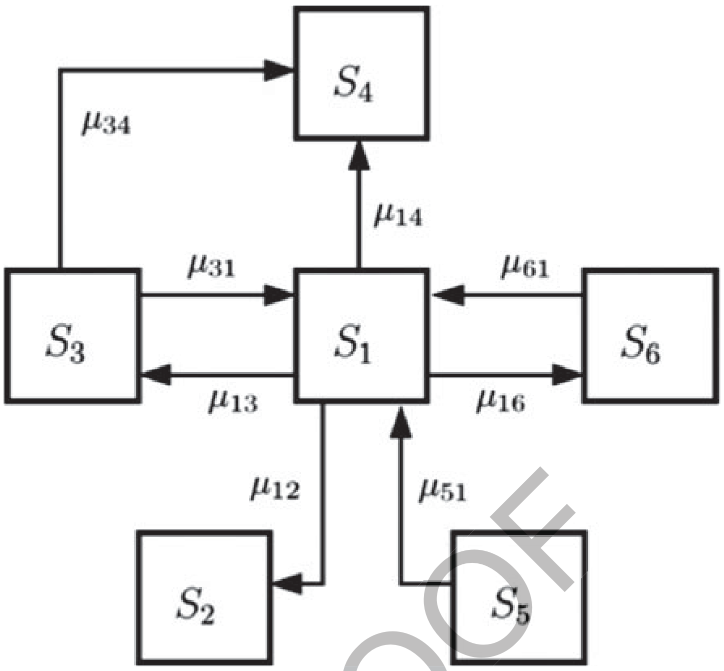

Fig. 16.4 Network subsystems S_i and their sets of shared connections μ_{ij}

Table 16.4 Dimensions of shared links μ_{ij}

Set	μ_{12}	μ_{13}	μ_{14}	μ_{16}	μ_{31}	μ_{34}	μ_{51}	μ_{61}
Number of u's	2	2	2	2	4	3	1	3

16.5 Conclusions

This chapter has proposed two approaches for the automatic partitioning of a WTN into subsystems intended to be applied along with a non-centralized model predictive control strategy. The algorithm transforms the dynamical model of the given system into a graph representation. Once the equivalent graph has been obtained, the problem of graph partitioning is then solved. The resultant partitions are composed of a set of non-overlapping subgraphs such that their sizes, in terms of number of vertices, are similar and the number of edges connecting them is minimal. To achieve this goal, the algorithm applied a set of procedures based on identifying the highly connected subgraphs with balanced number of internal and external connections. Some additional prefiltering and post-filtering routines are also needed to be included to reduce the number of obtained subsystems. The performance of the proposed decomposition approach has been assessed in a real case study based on the Barcelona WTN. A study of the effect of auxiliary routines on the basic partitioning algorithm has also been included showing the benefits of their use.

References

1. Addario-Berry L, Dalal K, Reed B (2008) Degree-constrained subgraphs. Discrete Appl Math 156(7):1168–1174
2. Bondy JA, Murty USR (2008) Graph theory. Graduate series in mathematics, vol 244. Springer
3. Brdys M, Ulanicki B (1994) Operational control of water systems: structures, algorithms and applications. Prentice Hall International, UK
4. Bui TN, Moon BR (1996) Genetic algorithm and graph partitioning. IEEE Trans Comput 45(7):841–855
5. Cormen TH (2001) Introduction to algorithms. The MIT Press
6. Dunbar WB (2007) Distributed receding horizon control of dynamically coupled nonlinear systems. IEEE Trans Autom Control 52(7):1249–1263
7. Dutt S (1993) New faster kernighan-lin-type graph-partitioning algorithms. In: Proceedings of the IEEE/ACM international conference on computer-aided design. IEEE Computer Society Press, pp 370–377
8. Fjallstrom P (1998) Algorithms for graph partitioning: a survey. In: Linkoping electronic articles in computer and information science, vol 3, issue no 10
9. Gupta A (1997) Fast and effective algorithms for graph partitioning and sparse-matrix ordering. IBM J Res Dev 41(1):171–183
10. HD-MPC (2008) Hierarchical and distributed model predictive control of large-scale complex systems. Home page
11. Keviczky T, Borrelli F, Balas GJ (2006) Decentralized receding horizon control for large scale dynamically decoupled systems. Automatica 42(12):2105–2115
12. Li F, Zhang W, Zhang Q (2007) Graphs partitioning strategy for the topology design of industrial network. IET Commun 1(6):1104–1110
13. Lunze J (1992) Feedback control of large-scale systems. Prentice Hall, Great Britain
14. Marinaki M, Papageorgiou M (2005) Optimal real-time control of sewer networks. Springer, Secaucus, NJ (USA)
15. Negenborn RR (2008) Multi-agent model predictive control with applications to power networks. PhD thesis, Delft University of Technology, Delft, The Netherlands
16. Negenborn RR, De Schutter B, Hellendoorn J (2008) Multi-agent model predictive control for transportation networks: serial vs. parallel schemes. Eng Appl Artif Intell 21(3):353–366
17. Ocampo-Martinez C, Bovo S, Puig V (2011) Partitioning approach oriented to the decentralised predictive control of large-scale systems. vol 21, 775–786
18. Van Overloop PJ (2006) Model predictive control on open water systems. Delft University Press, Delft, The Netherlands
19. Rawlings JB, Stewart BT (2008) Coordinating multiple optimization-based controllers: new opportunities and challanges. J Process Control 18(9):839–845
20. Scattolini R (2009) Architectures for distributed and hierarchical model predictive control: a review. J Process Control 19(5):723–731
21. Sezer ME, Siljak DD (1986) Nested ε-decomposition and clustering of complex systems. Automatica 22(3):321–331
22. Šiljak DD (1991) Decentralized control of complex systems. Academic Press
23. Venkat AN, Hiskens IA, Rawlings JB, Wright SJ (2008) Distributed MPC strategies with application to power system automatic generation control. IEEE Trans Control Syst Technol 16(6):1192–1206
24. WIDE (2008) Decentralized and wireless control of large-scale systems. http://ist-wide.dii.unisi.it/
25. Zecevic AI, Šiljak DD (1994) Balanced decompositions of sparse systems for multilevel parallel processing. IEEE Trans Circuits Syst I: Fund Theory Appl 41(3):220–233
26. Zecevic A, Siljak DD (2010) Control of complex systems: Structural constraints and uncertainty. Springer Science & Business Media