POLITECNICO DI MILANO

FACOLTÀ DI INGEGNERIA DELL'INFORMAZIONE

Corso di Laurea in

Ingegneria dell'Automazione

OPTIMIZATION ISSUES FOR DEMAND-SIDE MANAGEMENT IN IRRIGATION NETWORKS

Relatore: Prof. MARCELLO FARINA
Correlatori: Prof. TANSU ALPCAN
Prof. MICHEAL CANTONI

Tesi di laurea di:
LUCA CASTELLI DEZZA
Matr. 780862

Anno Accademico 2012 - 2013

Acknowledgements

The following work has been possible thanks to Prof. Marcello Farina, from Politecnico di Milano, and Prof. Michael Cantoni, from the University of Melbourne. Their cooperation has created the possibility of an experience abroad that allowed me to see different approaches to control engineering and to experience different cultures.

A first thanks to Prof. Marcello Farina. His clear suggestions and serene advices have been fundamental during the permanence in Melbourne and in Milan.

Thanks to Prof. Michael Cantoni and Prof. Tansu Alpcan, from University of Melbourne. Their willingness and decisive support have been central for the development of this Thesis.

A special thanks to Anthony Oakes, co-founder and manager of Rubicon Water Pty Ltd, for the help and the collaboration.

A last thanks to the RHD office of the Department of Electrical and Electronic Engineering in the University of Melbourne for the warm welcome and assistance during the whole permanence in Australia

Contents

List of Figures

List of Tables

Abstract

In this Thesis, we will discuss an high-level optimizer solving the problem of finding optimal solutions for scheduling issues on a gravity-fed large-scale irrigation network. Our aim will be the scheduling of the requests from the farmers fulfilling given constraints on the water-level and minimizing the delivery delays in water supply. In this regard, we will develop different optimization problems in order to achieve our objective.

In particular, first of all, we will discuss a suitable model of an irrigation network's channel. Starting from the continuous-time model of a pool, we will develop a state-space discrete-time model of a string of pools in the centralized fashion. Secondly, to guarantee the water-level to fulfill given upper and lower bounds among the entire prediction horizon, we will define a generic optimization problem formulation whose dynamical constraints are included. Since our aim is to minimize the delivery delays, we will choose the initial time of the water supply as the decision variable. This will lead to a non-linear optimization problem. We will, then, turn into a $\{0, 1\}$ integer linear formulation via change of variables. Afterwards, we will reformulate it by relaxing the integer constraints. We will, hence, obtain two relaxed problem formulations that do not guarantee the preservation of the original shape of the requested profile. Consequently, we will devise different possible objective functions for the problems. In particular, regarding the relaxed formulations, one linear and two quadratic solutions will be discussed and tested. Finally, we will apply two distributed algorithms on the decomposed problem. On one hand, we will implement and test the Round Robin algorithm. On the other hand, a Primal Decomposition method will be applied. In the end, we will compare the results of both the distributed solutions and we will remark the drawbacks and the advantages of both.

Sommario

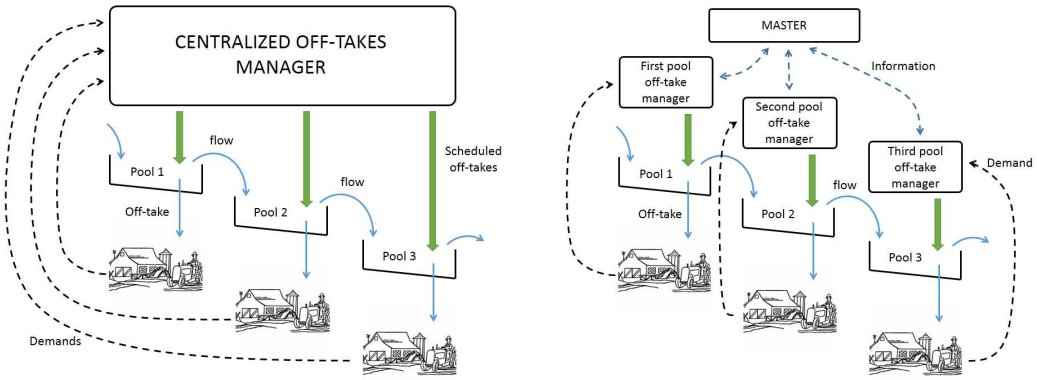
La maggior parte delle risorse idriche, attualmente impiegate nelle attività agricole, proviene dalla rete fluviale. A causa del continuo aumento della popolazione globale, si stima che, in pochi anni, questa risorsa potrebbe arrivare gradualmente a non coprire l'intero fabbisogno idrico necessario a questa attività produttiva. Per questi motivi, una gestione attenta e lungimirante di questa fondamentale risorsa naturale risulta essere essenziale, specialmente nelle aree geografiche caratterizzate da scarse precipitazioni, come alcune regioni dell'Australia, dove sono attualmente presenti reti dedicate alla distribuzione dell'acqua per l'irrigazione dei campi agricoli. In generale, i problemi di una rete di irrigazione di larga scala sono legati a una inefficiente gestione della risorsa che ha come conseguenze lo spreco o la mancanza dell'acqua durante certi periodi dell'anno. Una delle principali cause è il controllo manuale delle chiuse dove ancora non è presente una struttura automatizzata.

Una rete di irrigazione è costituita da serbatoi (laghi artificiali dove l'acqua viene raccolta durante la stagione umida) e da canali che forniscono l'acqua alle aziende agricole. Lungo questi canali, sono inserite delle chiuse che li dividono in sezioni e ne regolano il flusso in modo da mantenerne il livello d'acqua a un dato riferimento.

Per evitare le perdite e per garantire una sostenibilità a lungo termine, devono essere implementati sistemi di controllo automatici che garantiscano lo sfruttamento ottimale della risorsa idrica e minimizzino gli sprechi.

Fino ad ora, sono state realizzate alcune soluzioni centralizzate per la gestione delle richieste (si veda Figura 1.(a)). Possibili sviluppi con schemi distribuiti sono attualmente in fase di studio (si veda Figura 1.(b)).

In questa Tesi, il nostro scopo è quello di formulare un opportuno problema di ottimizzazione di alto livello che risolva il problema della pianificazione delle



(a) *Soluzione a due livelli centralizzata* (b) *Soluzione a due livelli distribuita*

Figure 1: Due schemi per la gestione e pianificazione delle richieste effettuate dagli utenti di una rete di irrigazione

richieste effettuate dagli utenti di una rete di irrigazione di larga scala. In particolare, verranno formulati alcuni problemi di ottimizzazione allo scopo di minimizzare i ritardi nel fornire servizi idrici agli utenti, rispettando vincoli sui massimi e minimi livelli dell'acqua nelle singole piscine.

Per prima cosa costruiremo un modello di un canale della rete di irrigazione. Partendo da un modello a tempo continuo di una singola piscina, arriveremo alla definizione di un modello centralizzato a tempo discreto in spazio di stato. In questo modello è stato inserito un sistema di controllo di basso livello decentralizzato.

Successivamente, formuleremo un problema di ottimizzazione che permetterà di esprimere in modo esplicito i vincoli massimi e minimi di livello dell'acqua che devono essere rispettati lungo tutto l'orizzonte temporale indicato. Svilupperemo quindi quattro problemi di ottimizzazione: un problema non lineare, uno lineare intero e due ottenuti dal rilassamento dei vincoli di interezza. Quindi, saranno discusse alcune possibili funzioni obiettivo. Verranno infine mostrati vantaggi e svantaggi dei problemi di ottimizzazione descritti, anche con l'ausilio di simulazioni e confronti.

Per ultimo, si procederà con la scomposizione del problema di ottimizzazione in sotto-problemi di scala ridotta, con l'ausilio di tecniche da letteratura. Questa fase sarà resa possibile grazie a un preventivo riordinamento degli elementi dei vettori e delle matrici coinvolti nei vincoli. Quindi, verranno discussi due diversi algoritmi distribuiti. In particolare, saranno presentati l'algoritmo Round Robin e

la scomposizione primale. Quest'ultima sarà valutata la soluzione più interessante sia in termini di ottimalità della soluzione ottenuta, sia dal punto di vista di possibili sviluppi futuri.

Chapter 1

Introduction

In this chapter, we will introduce some fundamental control problems related to the management of large-scale irrigation networks and discuss the motivations of our Thesis. Our goal is to provide the reader with basic knowledges regarding the system and the solutions that are already implemented. Hence, first of all, a general configuration of an irrigation network is described. Furthermore, the problems related to such a large scale system are described. Also, we will outline the original contribution of our work and, finally, the structure of the Thesis will be shown.

1.1 Context and Motivations

Around 70% of the water diverted from rivers and groundwater worldwide goes to irrigation and this underpins 40% of global food production. Since the world population is always increasing, the amount of water will shortly not be able to cope the water demand [8]. For these reasons, the water preservation is a very important issue, especially in dry areas. A forward-looking and careful management of this essential resource should be the most important matter, especially in that areas. Thereby, it is crucial for the agriculture to implement efficient irrigation management procedures [15].

Australia is a very dry continent and it is estimated that, not late, the water demand for agriculture will not be entirely satisfied. In particular, the North of Victoria is one of the most productive regions of Australia regarding the agriculture. It has been estimated that more than the 90% of the Australian fruit

and vegetable is produced in that area. The majority of irrigation in that region is developed through a civil infrastructure of reservoirs and open channels that supplies fresh water to farms as sketched in Figure 1.1.

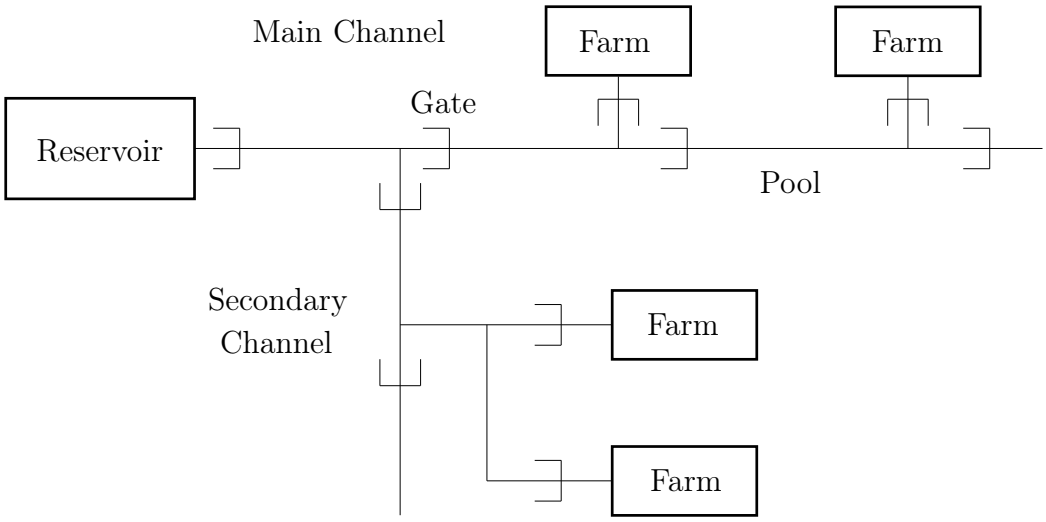


Figure 1.1: A sketch of an irrigation network, source [6]

1.1.1 General Structure of an Irrigation Network

During the winter, water is collected in the reservoirs, e.g., artificial lakes, and it is distributed to the farms during the summer through a large-scale irrigation network. The latter is composed by artificial open channels, i.e., main and secondary channels. The channels are divided into sections, called pools, by over-shot gates (e.g. FlumeGate^{TM}, see Figure 1.2.(a)). The water flow along the irrigation channels is regulated by the gates.

The water-level of each pool is monitored. The measure is transmitted from a radio antenna, that is installed close to the downstream gates, to the regulator of the upstream gate. The gate, in turn, is controlled in order to maintain the variations of the water-level to a set-point reference by varying the water flow. The implemented controller, by now, is simply PI regulator that guarantees the satisfaction of given performances, e.g. stability, robustness, etc.

The regulation of the water-level is an important and non-trivial issue, particularly since the water distribution is powered only by gravity. If the water-level decreases too much (e.g. with respect to a given threshold), the performances of the regulation and management systems are hampered because the water flow

(a) *Automated over-shot gate in a main channel*

(b) *Automated over-shot gate at off-take point*

Figure 1.2: Automated over-shot gates

travelling across a gate is proportional to the water-level, in view of the fact that water, that is supplied to the farms, is distributed through over-shot gates (see Figure 1.2.(b)).

The opening of gates on behalf of farmers causes a variation of the water-level. In terms of quality of service, it is also important to avoid fluctuations from set-points, which yields to fluctuations in flow at the off-take points, in view of the fact that the irrigation is gravity-fed. In particular, some lower bounds on the water-level are fixed in order to guarantee the gates to supply a certain water flow, in any case.

1.1.2 Problems of the Irrigation Network

Analysing the structure of the demand-supply mechanism, we can outline some problems that, presently, lead to inefficient water supply. The water distribution efficiency in irrigation networks can be defined as the ratio between the volume of water actually exploited for irrigation and the amount of water requested from the available resources. In many cases, the efficiency is estimated to be less than 50% [10].

Large scale distribution losses are mainly caused by the water oversupply [6]. This occurs, since present water management systems are designed to guarantee, in an overly-conservative way, suitable flow in the entire network, in any possible situations. However, the amount of water that is not requested by the users of the network, is lost at the end of the channel, usually in the sea. On the other hand, also under-supply conditions may cause serious consequences from the ecological standpoint, concerning the aquatic ecosystem. Problems of this type have occurred in the last years, during the dry season.

Performance losses can also occur since the openings of the gates at farms, in most cases, are presently manually actuated, which brings about possible delays and errors.

In conclusion, more efficient and automatic water managing systems are required, to reduce or avoid losses, to tackle under-supply problems, and to tame fundamental long-term environmental and sustainability issues.

1.1.3 Case Study

In this Thesis the case study consists of a part of the "Goulburn-Murray Irrigation District" that includes about 7000 km of irrigation channels (see Figure 1.4).

The water is collected in an artificial lake called "Nagambie Lake" obstructed by the "Goulburn Weir" (see Figure 1.3). The irrigation network in this area is

Figure 1.3: Pictures of the irrigation network of "Goulburn-Murray Irrigation District"; from left to right: the "Nagambie Lake", the "Goulburn Weir", gates of the main channel, a gate of a secondary channel, a farm

managed by a privately held Australian company called Rubicon Water Pty Ltd. Rubicon Water leads the irrigation industry all over the world and provides their customers all the necessary components, both hardware and software.

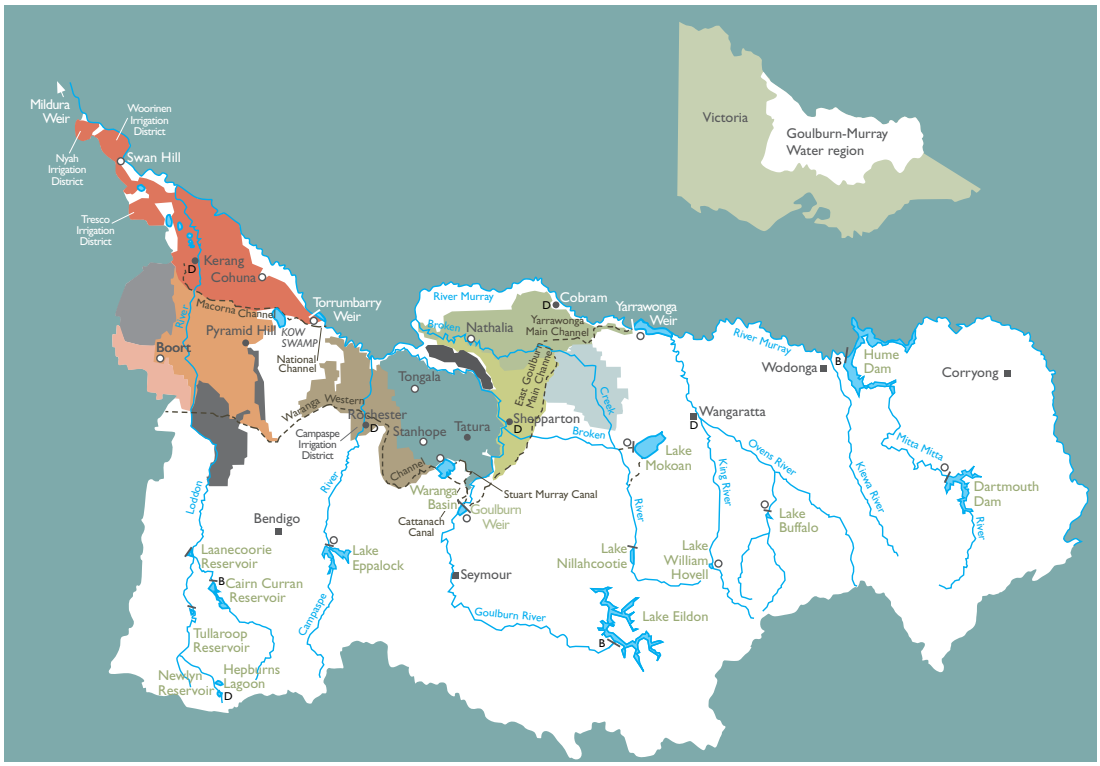


Figure 1.4: The Goulburne-Murray Irrigation District: the EMG Channel is represented by the dashed line starting from the Goulburne Weir going to the right-side

1.2 Statement of the Problem

In this section, we want to highlight the recent results towards the solution to the problem of enhancing the efficiency in water distribution. The discussed solutions have been developed by the Rubicon Water research group in collaboration with the University of Melbourne. Their cooperation has led to many important achievements in flow measurement and control. Today, they jointly own the intellectual property of many patents and they have been recognised by governmental and academic awards.

1.2.1 Management of the Requested Off-Takes

As discussed in the previous section, the losses in terms of water can be caused by a bad timing in irrigation, as a consequence of manual water scheduling on the supply canals. Moreover, as discussed, this undesirable situation reduces the performances in terms of efficiency of the overall irrigation network. In the last years, Rubicon Water, according to the Australian Government, is turning the manual gates at farms' connections into fully automated systems. In the meanwhile, it is developing the software to control the opening and the closure of the gates. On one hand, Rubicon's Confluent^{TM} software enables the managers of water to plan and manage the requests from the users. On the other hand, the farmers are provided a web application named Rubicon's FarmConnect® (see Figure 1.5). Through the web interface, the farmers can request the off-takes in

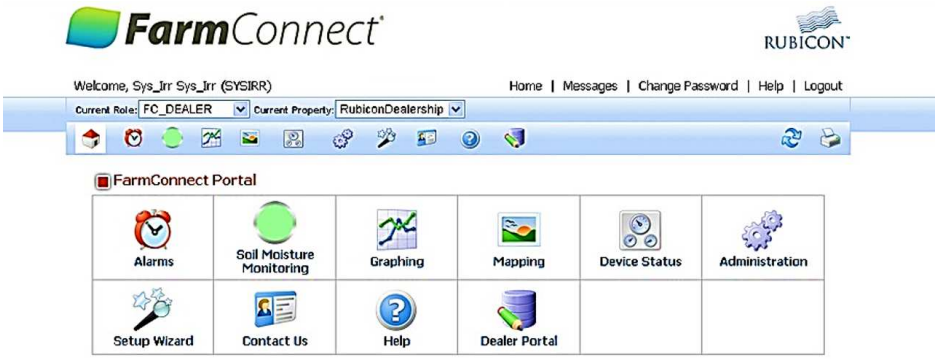


Figure 1.5: FarmConnect® web interface (http://www.rubiconwater.com)

terms of flow of water, supply time and starting time. All the requests are daily

collected by Confluent software. In order to schedule the demands, following a FIFO (first in first out) logic, the software checks that static constraints, in terms of maximum predicted flow rate, are satisfied. If not, the off-take is delayed at the first available time. These leads to non-optimal solutions in terms of delivery delays in water supply. Moreover, this solution does not guarantee the fulfilment of the lower and upper bound constraints on the water-level in the pools. In this Thesis, an optimal scheduling managing system will be developed.

1.2.2 Optimization Issue

In collaboration with Rubicon Water, a research group of the University of Melbourne has been active, in the last years, in studying control solutions for irrigation channel regulation [18].

Among the possible control solutions currently proposed, a two-layer one, discussed below, is particularly promising. This two-layer control architecture is endowed with a lower-layer decentralized/distributed control scheme and an higher-level off-take planner (see Figure 1.6).

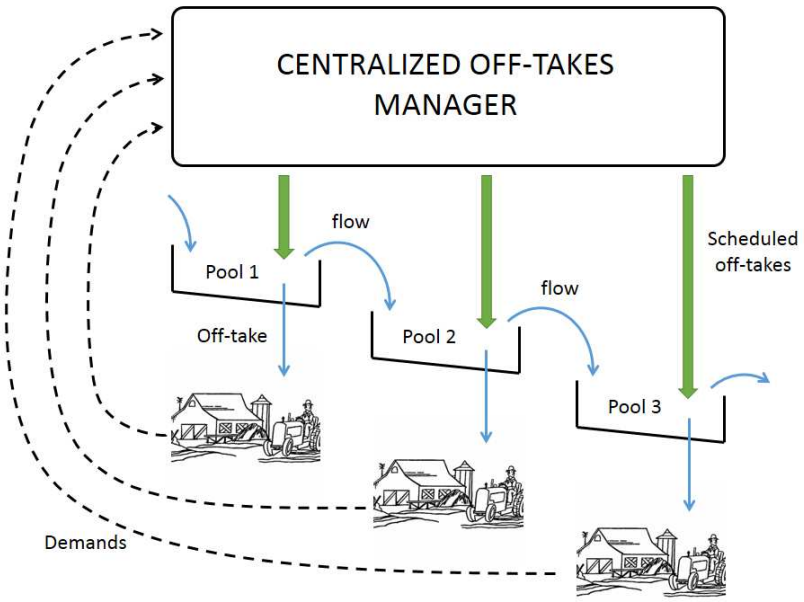


Figure 1.6: Structure of a centralized optimization problem: all the demands are collected by a centralized solver that sends back the off-takes scheduling

Concerning the first, it is designed to control water-levels at given set-point and consists in a decentralized/distributed scheme where, for each pool, a local

PI regulator is committed to the control of the local water-level. In view of its simplicity, the application of this scheme can be critical when constraints are likely to be violated.

To prevent critical situation, and to guarantee optimal water supply to the users (both concerning the minimization of supply delays and in terms of maximization of supply efficiency), higher-level optimization-based off-take scheduler is proposed. This Thesis will focus on the latter point.

1.3 Original Contributions of this Work

In [1] and [9], two optimization programs for off-take scheduling are already discussed and tested in simulation. Our work will develop this solution in order to obtain scalable and less computationally demanding algorithms. The scope of this Thesis is the creation and the testing of different formulations solving the scheduling problem. This result will be achieved through a linear relaxation of the constraints related to the original integer problem. Furthermore, we will implement and test decomposition methods on the main centralized optimization problem, in order to obtain distributed solutions (see Figure 1.7).

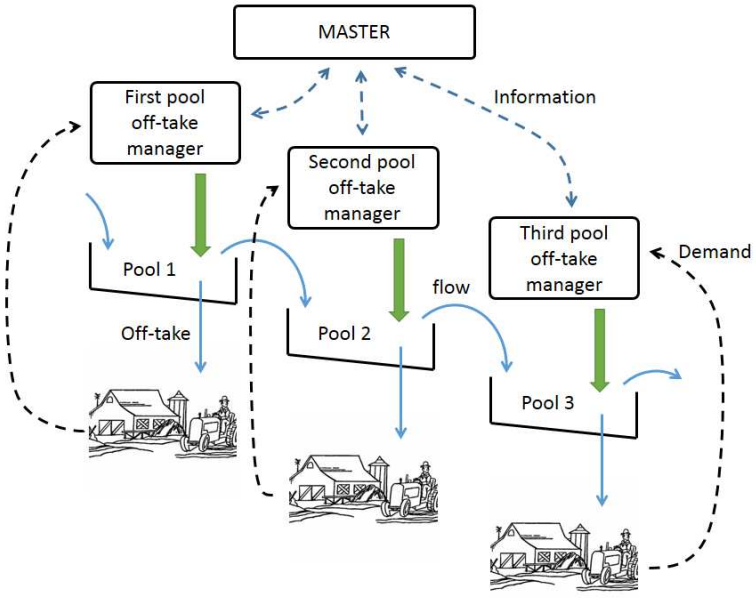


Figure 1.7: Structure of a decomposed optimization problem: the demands are sent to local computational units, solving their 'local' problems, according to information received from a master problem

These important and innovative solutions create the basis for negotiation mechanisms between the farmers and the water managers. These results will be implemented on the real system in the future.

1.4 Structure of the Thesis

In Chapter 2, we will develop the model of a channel. Starting from the continuous-time model of a pool, we will work out the centralized discrete-time model of a channel. A suitable decentralized control system, that is adopted in order to guarantee given performances in terms of robustness and closed-loop bandwidth, will be shown.

In Chapter 3, we will formalize the constraints of a generic optimization program and we will devise different centralized solutions for the scheduling optimization problem: 1) a non-linear formulation (NLP); 2) a $\{0, 1\}$ integer linear program (ILP); 3) an optimization program (RP1) obtained by relaxing the integer linear constraints; 4) a completely new linear problem formulation (RP2). Afterwards, we will devise some suitable objective functions for all the problems. For NLP and ILP, two different linear objective functions will be suggested. Concerning the relaxed programs, we will implement different solutions. We will, thus, show some simulation tests for comparing all the obtained results, both concerning their computational demand, and regarding the optimality properties of their solutions.

In Chapter 4, we will apply two distributed algorithms to the decomposed relaxed linear problem. We will solve the distributed problems applying the Round Robin algorithm and the primal decomposition method. The simulation tests of both the algorithms will be shown in Sections 4.2.2 and 4.3.5. We will, then compare the solutions and we will highlight the advantages and drawbacks of both.

Finally, in Chapter 5, we will draw some conclusions. Possible future developments of this work are discussed.

Chapter 2

A Discrete-Time Model of an Irrigation Network

In this chapter, we will discuss some preliminary concepts, whose importance will be cleared in the following of the Thesis. First of all, we will devise the model of a channel: in this model the low-level controllers, that guarantee the robustness and a good tracking level of the reference by the water-level, are included. In particular, a decentralized system structure is highlighted. Secondly, an high-level control strategy in the form of an optimization structure is described.

2.1 Continuous-Time Model of a Gravity-Fed Irrigation Network

In this section, we will focus on a part of an irrigation network, but all the considerations we make can be extended to larger-scale systems. As discussed, (in the case of irrigation network we are dealing with) the water is distributed via open channels under the power of gravity. Each channel is divided in sections by two gates (i.e. upstream and downstream gates). The stretch of a channel, lying between two gates, is called *pool*. Each channel is identified by some parameters that we will illustrate later on. In order to manage the water in an appropriate way, the gates are controlled via a decentralized control system. In Figure 2.1, the section of two pools is represented.

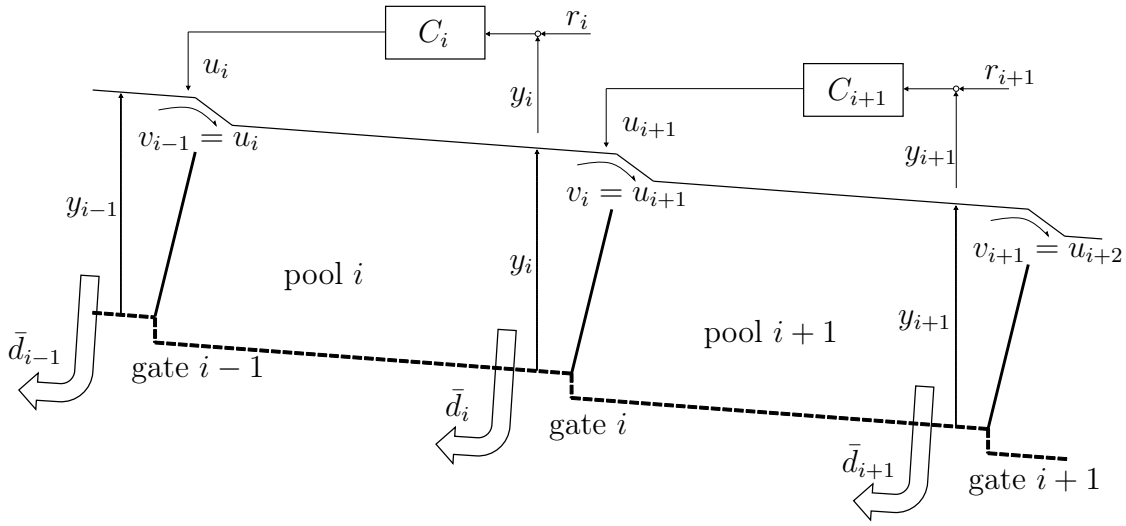


Figure 2.1: Sketch of an open-water channel with gates

2.1.1 Continuous-Time Model of a Single Pool

We will now devise a suitable model for our analysis. Due to the mass-conservation principle, we have to guarantee that the variation of amount of water in a pool y_i, must be equal to the difference between the input and output flows, called u_i and v_i, respectively. We denote by y_i the water-level in pool i. Note that the variable u_i acts at the $i-1$-th gate, while y_i is measured next to the gate i (see Figure 2.1). Thereby, we call $t_{d,i}$ the time that the water takes to travel from gate $i-1$ to gate i. The open-loop model of the i-th pool, described in [6], is:

$$\dot{y}_i(t) = c_{in,i}u_i(t - t_{d,i}) - c_{out,i}v_i(t) - c_{out,i}\bar{d}_i(t) \tag{2.1}$$

where:

- y_i is the water-level of the i-th pool at the bottom of the pool.
- u_i is the input water flow at gate $i-1$.
- v_i is the output water flow requested by the downstream pool. Moreover, it is the water flow entering in the $i+1$-th pool. Hence, the variable v_i is called the *coupling variable* of the system: in fact, it holds that $v_i = u_{i+1}$. In other words, v_i represents the information being sent by the downstream pool $i+1$ to the upstream one i in terms of requested water flow. The importance of such a variable will be discussed later on.

- $\bar{d}_i$ is the water off-take affecting the pool i. It is supposed to effect closely to the downstream gate i.

- $c_{in,i}$ and $c_{out,i}$ are parameters depending on the geometry (width, height, etc.) of the upstream and downstream gates, respectively, and on the surface of the pool [6].

- $t_{d,i}$, as revealed in advance, is the internal time delay that the water takes to travel from gate i to gate $i+1$.

In model (2.1), we are neglecting the dominant wave phenomenon. This is a very good approximation provided that each local control-loop bandwidth lies at a frequency below it. Without this assumption, the model would be much more complex and less prone to the optimization methods studied in this Thesis. The controller structure C_i will be discussed in the following.

Structure of the Decentralized Control System

In a decentralized control system, each pool is controlled separately, without taking into account of the dynamics of the other pools. Let the water level y_i be the controlled and measured variable for the i-th pool. The measure of y_i is detected by embedded sensors at the end of the pool, close to the downstream gate. The control variable is the water flow u_i coming from the upstream pool $i-1$ through the gate $i-1$. For the sake of simplicity, in our case, we will consider the actuator to have an instantaneous effect on the control variable: it is considered as a simple gain already included in the model. This approximation is made, assuming that the dynamics of the pool ($t_{d,i} \sim$ minutes) is much slower than the actuators' one (time constants in the order of seconds).

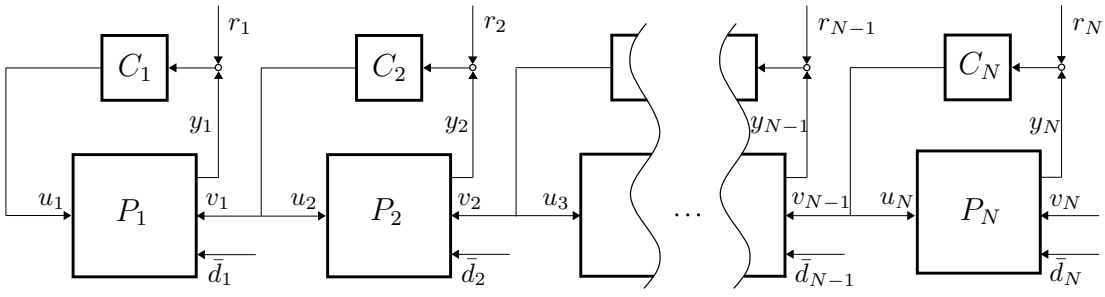


Figure 2.2: Scheme of the decentralized control system

To control the water-level in each pool in a decentralized fashion, PI regulators are used. The controllers are designed in order to guarantee that the controlled variable y_i follows the given water reference r_i, rejecting disturbances associated with changes in the load $\bar{d}_i$ in the i-th pool. The transfer function of the regulator is:

$$C_i(s) = \frac{\kappa_i(1 + s\phi_i)}{s(1 + s\rho_i)} \tag{2.2}$$

where the constants κ_i, ϕ_i and ρ_i are tuned to ensure closed-loop stability and robustness and to limit the closed-loop bandwidth. This regulator is already implemented in the systems of the irrigation network we are working on. As discussed in [17], this decentralized control system does not guarantee satisfactory disturbance rejection properties in transient conditions. This property is denoted string-instability and is defined in [17]. It is possible to reject the string-instability by using a distributed control scheme, rather than a decentralized one, i.e., by adding a feed-forward compensator for each pool. Nevertheless, the analysis of such a distributed control system (PI regulator plus feed-forward compensator) is not relevant for our Thesis. Indeed, the solutions, that we will achieve using the decentralized control system, can be easily modified in order to suit also the distributed control system. This can be done, just changing the predictor (see Section 3.2) which is based on the model we want to use.

2.1.2 Continuous-Time Model of a Channel

In the previous section, a suitable model of a controlled single pool is described. In this section, we are going to show the continuous-time model of an irrigation channel, where the decentralized control system is embedded.

Combining N pools of a channel, we obtain the following continuous-time model:

$$\begin{cases} \dot{y}_i(t) = c_{in,i}u_i(t - t_{d,i}) - c_{out,i}v_i(t) - c_{out,i}\bar{d}_i(t) \\ u_i(t) = v_{i-1}(t) \end{cases} \qquad \text{for } \; i = 1, \ldots, N \tag{2.3}$$

with the boundary condition $v_N = 0$ (we assume that there is no flow out of the last pool). Combining the equations of the single pool with the relative decentralized controller, we obtain the decentralized model of the channel. The

water-levels y_i, for $i = 1, \ldots, N$, are the measured variables. The control variables u_i, for $i = 1, \ldots, N$, are the outputs of the controllers. In particular, as anticipated in Section 2.1.1, u_i is also the water flow exiting from the pool $i-1$, and named v_{i-1}. We consider, hence, u_i and y_i as the outputs of the single controlled pool i, while the inputs are r_i, $\bar{d}_i$ and v_i.

An example of a controlled channel composed by four pools, endowed with the decentralized control system, is shown in Figure 2.3.

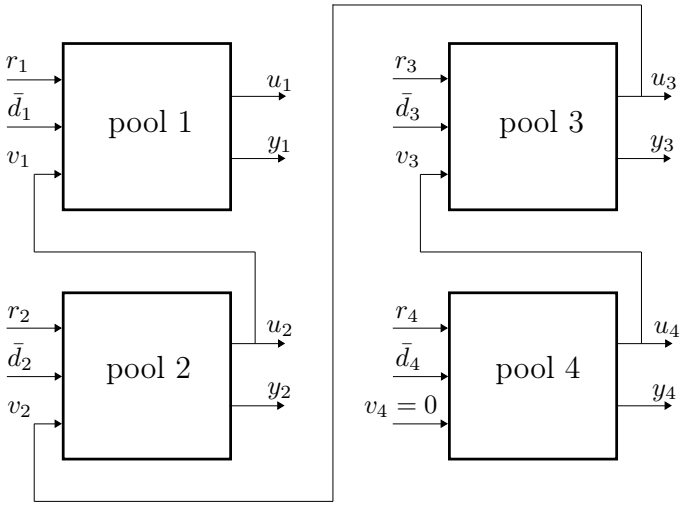


Figure 2.3: Sketch of a decentralized structure of a channel with $N = 4$ pools

The importance of the coupling variable v_i becomes clear from Figure 2.3. Focusing on pool N, and assuming that we start from steady state conditions, supposing $\bar{d}_i = 0$ for $i = 1, \ldots, N-1$, we can see that a change of the relative load $\bar{d}_N$ effects directly the water-level y_N, supposing the error $e_N = r_N - y_N$ at steady-state to be equal to zero. In turn, an increase/decrease of y_N, and, consequently, of $e_N = r_N - y_N \neq 0$, produces a variation of the control variable u_N (assumed to be equal to zero at steady-state due to the PI regulator). Since u_N must be equal to v_{N-1}, due to the boundary condition $v_i = u_{i+1}$, a change of v_{N-1} imposes a variation of y_{N-1}, that, in turn, makes u_{N-1} vary. Thus, iteratively, a change of the load $\bar{d}_N$ affects all the water-levels in the upstream pools from $N-1$ to 1.

Generally speaking, a variation in the load $\bar{d}_i$ affects the pools from $i-1$ to 1, involving all the upstream control systems.

This is a particular structure that will be highlighted in the following, and will be exploited for model decomposition purposes.

2.2 Discrete-Time Model of a Gravity-Fed Irrigation Network

In the previous section, a continuous-time model of a pool is devised and the overall model of a channel is described. In this section, our scope is, starting from the continuous-time model of the single controlled pool (2.1), to discretize it and to build a centralized structure in order to obtain a suitable model for our Thesis.

2.2.1 Discrete-Time Model of a Single Pool

To develop the discrete-time model, first of all, we are going to split (2.1) in two equations: a storage equation and a transport equation. The latter represents the delay, while the former contains the integrator term, i.e.:

$$\begin{cases} \dot{y}_i(t) = c_{in,i}u'_i(t) - c_{out,i}v_i(t) - c_{out,i}\bar{d}_i(t) \\ u'_i(t) = u_i(t - t_{d,i}) \end{cases} \tag{2.4}$$

The first equation in (2.4), can be easily discretized analytically, i.e.:

$$y_i(k+1) = y_i(k) + T_y c_{in,i}x^D_{i,1}(k) - T_y c_{out,i}v_i(k) - T_y c_{out,i}\bar{d}_i(k) \tag{2.5}$$

where T_y is the sampling time of the discrete-time model and $x^D_{i,1}(k)$ is obtained by discretizing the second equation of (2.4). Indeed, in order to discretize the delayed variable $u'_i(t) = u_i(t - t_{d,i})$, we need $n_d = \frac{t_{d,i}}{T_y}$ discrete states representing the delayed values of $u_i(t)$, i.e.:

$$\begin{cases} x^D_{i,1}(k+1) = x^D_{i,2}(k) \\ x^D_{i,2}(k+1) = x^D_{i,3}(k) \\ \quad\vdots \\ x^D_{i,n_d-1}(k+1) = x^D_{i,n_d}(k) \\ x^D_{i,n_d}(k+1) = u_i(k) \end{cases} \tag{2.6}$$

The latter is the discrete-time model of the i-th pool.

We will now show the model of the controller. As we have discussed in Section 2.1.1, each pool is controlled via a PI regulator whose transfer function is given

in (2.2). In particular, we can write the control variable $u_i(t)$ as function of the water-level error $r_i(t) - y_i(t)$. In the Laplace domain:

$$U_i(s) = C_i(s)\big(R_i(s) - Y_i(s)\big) \tag{2.7}$$

We can easily transform equation (2.7) in a time domain state-space expression. In general, we obtain:

$$\begin{cases} x_i^K(k+1) = A_i^K x_i^K(k) + B_i^K r_i(k) - B_i^K y_i(k) \\ u_i(k) = C_i^K x_i^K(k) + D_i^K r_i(k) - D_i^K y_i(k) \end{cases} \tag{2.8}$$

where $x_i^K(k)$ is the state variable vector of the i-th controller and (A_i^K, B_i^K, C_i^K, D_i^K) represents the state-space model of the regulator.

Recalling that $x_{i,n_d}^D(k+1) = u_i(k)$, we can write the overall discrete-time model for the controlled pool i as follows:

$$\begin{cases} y_i(k+1) = y_i(k) + T_y c_{in,i} x_{i,1}^D(k) - T_y c_{out,i} v_i(k) - T_y c_{out,i} \bar{d}_i(k) \\ x_{i,1}^D(k+1) = x_{i,2}^D(k) \\ x_{i,2}^D(k+1) = x_{i,3}^D(k) \\ \quad \vdots \\ x_{i,n_d-1}^D(k+1) = x_{i,n_d}^D(k) \\ x_{i,n_d}^D(k+1) = C_i^K x_i^K(k) + D_i^K r_i(k) - D_i^K y_i(k) \\ x_i^K(k+1) = A_i^K x_i^K(k) + B_i^K r_i(k) - B_i^K y_i(k) \end{cases} \tag{2.9}$$

Note that the control variable $u_i(k)$ is, as a matter of fact, a state variable of the controlled system model.

We can now express (2.9) in a matrix form, i.e.:

$$\begin{bmatrix} y_i(k+1) \\ x_{i,1}^D(k+1) \\ x_{i,2}^D(k+1) \\ \vdots \\ x_{i,n_d-1}^D(k+1) \\ x_{i,n_d}^D(k+1) \\ x_i^K(k+1) \end{bmatrix} = \begin{bmatrix} 1 & T_y c_{in,i} & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & 0 & & 0 & 0 \\ 0 & 0 & 0 & 1 & & 0 & 0 \\ \vdots & & & & \ddots & & \vdots \\ 0 & 0 & 0 & 0 & & 1 & 0 \\ -D_i^K & 0 & 0 & 0 & & 0 & C_i^K \\ -B_i^K & 0 & 0 & 0 & \cdots & 0 & A_i^K \end{bmatrix} \begin{bmatrix} y_i(k) \\ x_{i,1}^D(k) \\ x_{i,2}^D(k) \\ x_{i,3}^D(k) \\ \vdots \\ x_{i,n_d}^D(k) \\ x_i^K(k) \end{bmatrix} + \\ + \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ D_i^K \\ B_i^K \end{bmatrix} r_i(k) + \begin{bmatrix} -T_y c_{out,i} \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \end{bmatrix} \bar{d}_i(k) + \begin{bmatrix} -T_y c_{out,i} \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \end{bmatrix} v_i(k) \tag{2.10}$$

The output of this model is the measured water-level $y_i(k)$, so that the output equation results to be the following::

$$y_i(k) = \begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 & 0 \end{bmatrix} \begin{bmatrix} y_i(k) \\ x_{i,1}^D(k) \\ x_{i,2}^D(k) \\ x_{i,3}^D(k) \\ \vdots \\ x_{i,n_d}^D(k) \\ x_i^K(k) \end{bmatrix} \tag{2.11}$$

Renaming $x_i(k) = \begin{bmatrix} y_i(k) & x_{i,1}^D(k) & x_{i,2}^D(k) & x_{i,3}^D(k) & \cdots & x_{i,n_d}^D(k) & x_i^K(k)^T \end{bmatrix}^T$, we obtain, from (2.10) and (2.11), the following expression:

$$\begin{cases} x_i(k+1) = A_i x_i(k) + B_{r,i} r_i(k) + B_{d,i} \bar{d}_i(k) + B_{v,i} v_i(k) \\ y_i(k) = C_i x_i(k) \end{cases} \tag{2.12}$$

that is the discrete-time model of the controlled pool i (pool plus controller).

2.2.2 Discrete-Time Centralized Model of the Channel

Our scope, in this section, is to create the centralized model of the channel that we are going to apply in our Thesis. Let $x(k)$, $r(k)$ and $\bar{d}(k)$ be the state, the reference and the load vectors of the centralized model, respectively, i.e.:

$$x(k) = \begin{bmatrix} x_1(k) \\ x_2(k) \\ \vdots \\ x_N(k) \end{bmatrix}, \quad r(k) = \begin{bmatrix} r_1(k) \\ r_2(k) \\ \vdots \\ r_N(k) \end{bmatrix}, \quad \bar{d}(k) = \begin{bmatrix} \bar{d}_1(k) \\ \bar{d}_2(k) \\ \vdots \\ \bar{d}_N(k) \end{bmatrix} \tag{2.13}$$

Focusing on the coupling variables $v_i(k)$, we know that $u_i(k) = v_{i-1}(k)$ for all i and $v_N(k) = 0$ for all k. Hence, we can include the variables v_i in the model, stacking in the appropriate way the matrices we found in the previous section. In particular, the state-space model of the centralized system, will be:

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \\ x_3(k+1) \\ \vdots \\ x_N(k+1) \end{bmatrix} = \begin{bmatrix} A_1 & \Phi_1 & 0 & \cdots & 0 \\ 0 & A_2 & \Phi_2 & & 0 \\ 0 & 0 & A_3 & \ddots & 0 \\ \vdots & & & \ddots & \\ 0 & 0 & 0 & & A_N \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \\ \vdots \\ x_N(k) \end{bmatrix} + \begin{bmatrix} B_{r,1} & 0 & \cdots & 0 \\ 0 & B_{r,2} & & 0 \\ \vdots & & \ddots & \\ 0 & 0 & & B_{r,N} \end{bmatrix} r(k) + \begin{bmatrix} B_{d,1} & 0 & \cdots & 0 \\ 0 & B_{d,2} & & 0 \\ \vdots & & \ddots & \\ 0 & 0 & & B_{d,N} \end{bmatrix} \bar{d}(k) \tag{2.14}$$

where Φ_i is a matrix stacking the vector $B_{v,i}$ in the appropriate position. We can, then, stack all the sub-matrices C_i, in order to obtain the matrix of the output transformation C, in the following way:

$$C = \begin{bmatrix} C_1 & 0 & \cdots & 0 \\ 0 & C_2 & & 0 \\ \vdots & & \ddots & \\ 0 & 0 & & C_N \end{bmatrix} \tag{2.15}$$

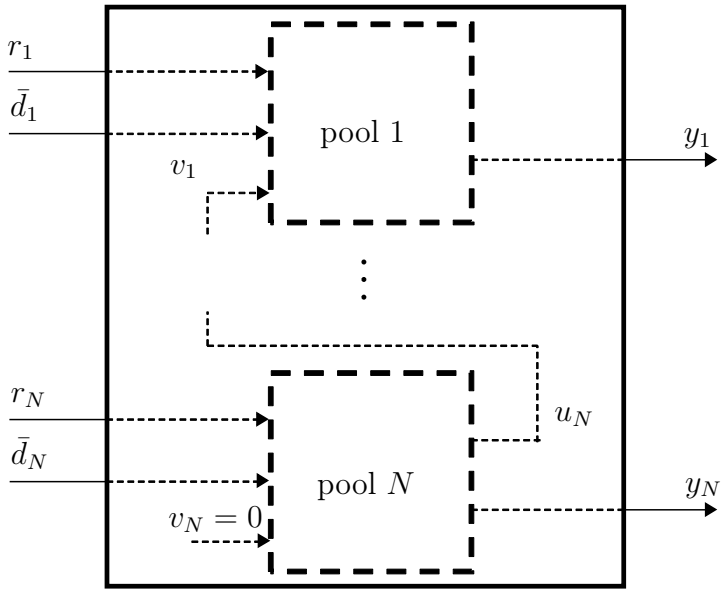


Figure 2.4: Centralized structure of the channel

Finally, the state-space model of the centralized system is:

$$\begin{cases} x(k+1) = Ax(k) + B_r r(k) + B_d \bar{d}(k) \\ y(k) = Cx(k) \end{cases} \tag{2.16}$$

After all this steps, the block-structure of the system has a triangular shape. This follows from the cascade interconnection structure depicted in Figure 2.4. In this figure, it is clear that the inputs of the decentralized model are the discretized references r_i and the loads $\bar{d}_i$ and the outputs are the water-levels y_i for all i. This feature will be used in the decomposition of the optimization problem that will be introduced in the following.

Initial Conditions

Once obtained the model (2.16), for simulation purposes, we may need to set the initial condition $x(0)$ of the states consistently with the model and in such a way that the system is initialized in a steady-state condition. We have to proceed through the following procedure, for each pool:

1. first of all, we have to set the water-level $y_i(0)$ equal to the relative reference at time zero $r_i(0)$

2. secondly, we impose all the states, related to the delay, $x^D_{i,j}(k)$ for $j = 1, \ldots, n_d$ to be equal to zero: $x^D_{i,j}(0) = 0$ for $j = 1, \ldots, n_d$

3. finally, concerning all the other states (the ones related to the controller) $x^K_i(k)$, we have to guarantee that there is no transients in absence of load, even if the reference is non-zero:

$$\bar{x}^K_i\big|_{k=0} = A^K_i \bar{x}^K_i\big|_{k=0} + B^K_i(\bar{r}_i - \bar{y}_i)\big|_{k=0} = (I - A^K_i)^{-1}(B^K_i(\bar{r}_i - \bar{y}_i))\big|_{k=0} \tag{2.17}$$

Since we imposed $\bar{r}_i(0) = \bar{y}_i(0)$, then:

$$\bar{x}^K_i(0) = 0 \tag{2.18}$$

2.3 The Optimization Issue

As discussed in the previous section, we are dealing with a centralized discrete-time model. The inputs are the water-level references r_i and the loads $\bar{d}_i$. As it has been detailed in Chapter 1, we will deal with the off-takes $\bar{d}_i$ scheduling problem, supposing the water-level references r_i to be set. The latter can be fixed and supposed decided a priori (based on past experience and knowledge on the plant) or can be given as outputs of a further optimization problem. In our case, we will assume that the water-references are not necessarily fixed at a certain value, but, however, known for the period of time we are applying the scheduling optimization problem.

The optimization problem we are dealing with, is an off-line one. Indeed, given a prediction horizon among which the off-takes must be scheduled, given the water-references in the prediction horizon and the demand of the farmers in terms of water at time k, the optimizer solves, centrally or in distributed way, a suitable optimization problem. After that, the optimal results, i.e., the water-supply scheduling over the prediction horizon, are transmitted to the systems that are devoted to water supply. In other words, a scheduling obtained at time k is applied for the entire prediction horizon, denoted by n_y, until time $k + n_y$.

2.3.1 Centralized Optimization Problem

The centralized optimization program receives, as input, the known water-level references r_i of each pool and the requested off-takes $\bar{d}_i$ from the users (farmers). The scheduled off-takes d_i are the outputs of the optimization. The structure of the centralized optimization problem is shown in Figure 2.5. The different solutions, proposed to achieve our goal, i.e. the scheduling of the off-takes, are described in Chapter 3.

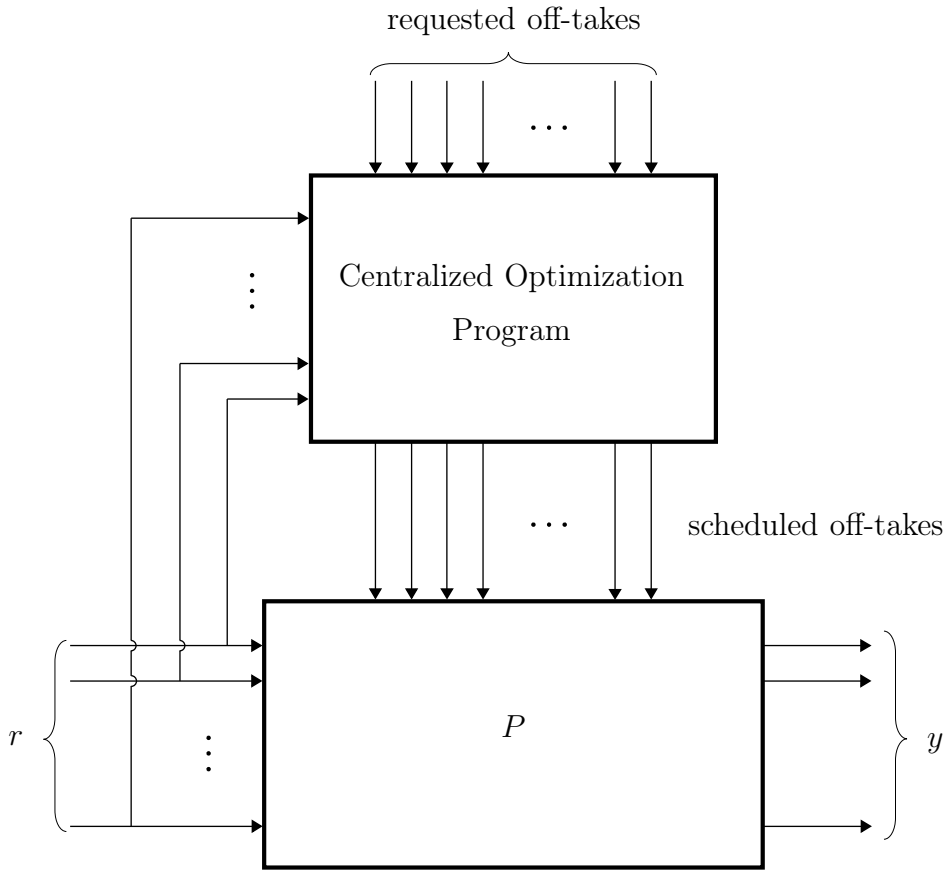


Figure 2.5: Centralized optimization scheme

2.3.2 Distributed Formulations of the Centralized Optimization Problem

Distributed solutions are very important especially in large-scale networks, where the computation of a centralized solution may be too computationally demanding. To obtain distributed solutions amounts to decomposing a given centralized problem into a number of small-scale ones, to be solved in a distributed, possibly recursive, fashion. In a decomposed structure, different and interconnected computing units solve their specific problem based on different sub-sets of decisional variables and use local information to achieve individual and global optimality. These local algorithms, indeed, achieve a global objective in a collective way.

In our Thesis we will show and deal with two possible strategies:

1. **Hierarchical strategies**

   This type of strategies includes two-layer optimization scheme (see Figure 2.6). The higher layer consists in a coordinator (i.e. the master problem

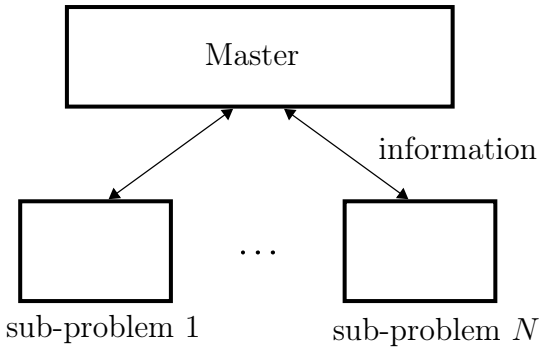


Figure 2.6: Hierarchical decomposition

   solver) which provides (with a star-like connection topology) data to the computing units at the lower layer, which, in turn, are devoted to solving small-scale optimization problems. In turn, the information received by the master is related to local optimal solution achieved by each sub-problem. In a price-coordination framework, the master solves the "central problem" and then sends back updated informations, in the form of limits in resource allocation or prices. The advantages of this technique are relative to a computational simplicity: as expected, the computationally demanding centralized optimization problem is decomposed into several sub-problems

that are much more easy and fast to be solved by the distributed lower-layer units. Moreover, the master solves a simple problem for the computational perspective, even on a large-scale framework. This kind of decomposable structure can be obtained with the *primal decomposition* that will be described in Chapter 4

2. **Completely distributed strategies**

In the other case, sketched in Figure 2.7, a one-level structure is implemented. It does not require a central coordinator, and, therefore, information is sent, in a neighbour-to-neighbour fashion, by the local computing units to the neighbouring ones. Such a decomposition method is suitable for our scheduling problem in irrigation networks in view of the triangular structure of the centralized model of a channel (see Section 2.2.2). As for the variables to be exchanged among sub-systems, it is worth mentioning that the information exchanged between the sub-problems is related to the coupling variables v_i introduced in Section 2.1.2. A suitable algorithm, called *Round Robin*, is discussed in Chapter 4.

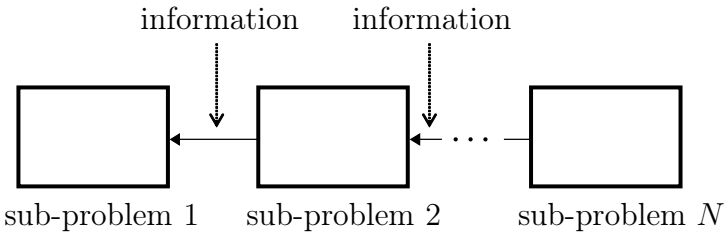


Figure 2.7: One-level decomposition

2.4 Case Study

We studied the case of the last two pools of the East Goulburn Main (EGM) channel, Victoria, Australia. It is composed by eleven pools and distributes water from the Nagambie Lake through the Goulburn-Murray Irrigation District (see Figure 1.4). The channels' data have been collected in the last years through identification methods. In Table 2.1 are shown the parameters $c_{in,i}$, $c_{out,i}$, τ_i, $c_{in,i}$ of the last two pools of EMG and the relative controllers' parameters k_i, ϕ_i, ρ_i.

pool				controller		
name	$c_{in,i}$	$c_{out,i}$	τ_i	k_i	ϕ_i	ρ_i
Campbells	0.055	0.036	5 min	0.74	71.83	8.52
Shifferlies	0.017	0.026	6 min	1.19	141.27	16.75

Table 2.1: Data of the last two pools of the EGM, source [9]

Chapter 3

Dynamical Centralized Problem

In this chapter our purpose is to describe the mathematical formulation of the problem expressed in the centralized fashion and to propose some suitable solutions to the problem. In particular, first of all, a non-linear problem is formulated where the decision variables are the delays τ_i in the water supply to the user i. Secondly, the problem is reformulated as an Integer $\{0, 1\}$ Linear Program via a change of variables. Thirdly a new optimization problem, obtained by a relaxation of the constraints, is described. Finally, a further problem formulation, strictly related to the previous one, will be described.

3.1 Mathematical Formulation

Starting from the model described in the previous chapter, we want to formulate a centralized optimization problem. Our scope is to illustrate the basic concepts for the mathematical model and to build a general formulation employed ahead.

First of all we introduce some basic notation about the problem we are going to deal with, similarly to [4]. A *standard form* optimization problem can be expressed in the following way:

$$\begin{array}{lll} \underset{z}{minimize} & f_0(z) & \\ \text{subject to} & f_i(z) \leq 0, & i = 1, \ldots, m \\ & h_i(z) = 0, & i = 1, \ldots, p \end{array} \tag{3.1}$$

In (3.1) the problem is to select the value of variable z (called the *optimization*

variable or *decision variable*) that minimizes the *objective function* $f_0(z)$ ($f_0 : \mathbb{R}^n \mapsto \mathbb{R}$) among all z that satisfy the conditions $f_i(z) \leq 0$, $i = 1, \ldots, m$, and $h_i(z) = 0$, $i = 1, \ldots, p$ called *inequality* and *equality constraints*, respectively. The functions $f_i : \mathbb{R}^n \mapsto \mathbb{R}$ are called the *inequality constraint functions* while $h_i : \mathbb{R}^n \mapsto \mathbb{R}$ are the *equality constraint functions*.

We can name the set of points for which both the objective and all constraint functions are defined

$$\mathcal{D} = \bigcap_{i=0}^{m} \operatorname{dom} f_i \cap \bigcap_{i=0}^{p} \operatorname{dom} h_i \tag{3.2}$$

as the *domain* of the optimization problem (3.1).

A point $z \in \mathcal{D}$ is said to be *feasible* if the conditions $f_i(z) \leq 0$, $i = 1, \ldots, m$, and $h_i(z) = 0$, $i = 1, \ldots, p$ are satisfied. Furthermore we say that the problem (3.1) is feasible if we can find at least a feasible point z, otherwise it is said to be *infeasible*. In particular, we call $z^\star$ the *optimal solution* of the problem (3.1), such that:

$$p^\star = f_0(z^\star) = \inf\{f_0(z) \mid f_i(z) \leq 0,\ i = 1, \ldots, m,\ \ h_i(z) = 0,\ i = 1, \ldots, p\} \tag{3.3}$$

The feasible point $z^\star$ is the solution to the optimization problem (3.1) and $p^\star$ is called *optimal value*.

The first step, we will deal with, is the formulation of the constraints of the optimization problem starting from the discrete-time dynamical system and the physical boundaries discussed in the previous chapter. Then, we have to formalize an objective (cost) function, the minimization of which implies the optimization of given performances (in terms of quality of service, i.e., satisfaction from the users) that we will introduce later on.

3.2 Prediction of the Water-Level and Constraints

First of all we focus on the dynamics of the system described by the state-space representation (A, B_r, B_d, C):

$$\begin{bmatrix} x(k+1) \\ y(k) \end{bmatrix} = \begin{bmatrix} A & B_r & B_d \\ C & 0 & 0 \end{bmatrix} \begin{bmatrix} x(k) \\ r(k) \\ \bar{d}(k) \end{bmatrix} \tag{3.4}$$

The distributed feedback controller, already embedded in model (3.4) is designed in order to make the controlled system output $y(k) \in \mathbb{R}^p$ follow a set-point reference $r(k) \in \mathbb{R}^p$. The system is affected by disturbances associated with changes in the load $\bar{d}(k) \in \mathbb{R}^N$ and $x(k) \in \mathbb{R}^n$ is the state of the dynamical system. Moreover the set-point reference $r(k)$ is assumed to be given for all $k \geq 0$. Note that $s(k) = \begin{bmatrix} s_1(k) & s_2(k) & \dots & s_m(k) \end{bmatrix}^T$ for a generic signal $s(k) \in \mathbb{R}^m$. Also, the notation $s^{(a:b)}$ stands for the vector $\begin{bmatrix} s(a)^T & s(a+1)^T & \dots & s(b)^T \end{bmatrix}^T$ for a discrete-time signal $s(k) \in \mathbb{R}^m$.

Given T_y, the sampling time associated with the discretized model of the dynamical system, we can compute the prediction of the output over a finite horizon of n_y slots of duration T_y. In particular, we want to compute the prediction of y from $k+1$ to $k+n_y$ by writing the dynamic equation of the discrete-time model (3.4) recursively:

$$\begin{aligned} \hat{y}^{(k+1:k+n_y)} =& \Gamma x^{(k)} + \Omega r^{(k:k+n_y-1)} + \\ &+ \Psi\left(\tilde{d}^{(k:k+n_y-1)} + d^{(k:k+n_y-1)}\right) \end{aligned} \tag{3.5}$$

where

$$\Gamma = \begin{bmatrix} CA \\ CA^2 \\ \vdots \\ CA^{n_y} \end{bmatrix}, \; \Omega = \begin{bmatrix} CB_r & 0 & \cdots & 0 \\ CAB_r & CB_r & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ CA^{(n_y-1)}B_r & CA^{(n_y-2)}B_r & \cdots & CB_r \end{bmatrix}, \tag{3.6}$$

$$\Psi = \begin{bmatrix} CB_d & 0 & \cdots & 0 \\ CAB_d & CB_d & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ CA^{(n_y-1)}B_d & CA^{(n_y-2)}B_d & \cdots & CB_d \end{bmatrix}, \tag{3.7}$$

Concerning the signal $\bar{d}$, it is composed by two terms:

- $\tilde{d}$, representing the scheduled and known load
- d, representing the signal component of the load to be scheduled

We can note that the output of the dynamical system depends on the current state $x(k)$, the history of the set-point $r^{(k:k+n_y-1)}$ and of the applied load $\tilde{d}^{(k:k+n_y-1)}$. As discussed, we suppose the set-point and the already scheduled load vectors known over the entire horizon as well as the current state. This means that the output $y^{(k+1:k+n_y)}$ is a linear transformation of the load to be applied $d^{(k:k+n_y-1)}$.

We have some constraint in terms of upper and lower bounds on the output that must be fulfilled over the entire prediction horizon. The water-level prediction must lie between those two given values because of physical reasons:

- the water is distributed via open water channels under the power of gravity. This means that a certain amount of water is requested to be in the channel in order not to loose performances in water distribution. Moreover a certain amount of water is necessary in order to preserve the aquatic ecosystem.
- the upper bound has to be imposed because of the finite hight of the channels' banks. This constraint is very important in order to avoid overflowing.

Formally, these constraints correspond to the following inequality:

$$\underline{y} \leq \hat{y}(k+1:k+n_y) \leq \bar{y} \tag{3.8}$$

supposing the values of the bounds fixed for the reasons explained above and $\underline{y} = [\underline{y}_1\, \underline{y}_2 \dots \underline{y}_p]^T$, $\bar{y} = [\bar{y}_1\, \bar{y}_2 \dots \bar{y}_p]^T$.

3.3 Optimization Problem Formulation

As introduced in the previous section, the constraints of our optimization problem are related to the prediction of the water-level. We can then formulate our optimization problem.

In general, as it will become clearer in the following, we describe the load to be scheduled as a suitable function of the generic free variable $z \in \mathcal{Z}$, used to minimize a given performance function $f(z)$. The generic optimization problem so obtained is:

$$
\begin{aligned}
\underset{z}{minimize} \quad & f(z) && \text{(3.9a)}\\
\text{subject to} \quad & \hat{y}^{(k+1:k+n_y)} = \Gamma x^{(k)} + \Omega r^{(k:k+n_y-1)} + \\
& \qquad + \Psi\big(\tilde{d}^{(k:k+n_y-1)} + d^{(k:k+n_y-1)}(z)\big) && \text{(3.9b)}\\
& \underline{y} \leq \hat{y}^{(k+1:k+n_y)} \leq \bar{y} && \text{(3.9c)}\\
& z \in \mathcal{Z} && \text{(3.9d)}
\end{aligned}
$$

We will now focus on the objective function (3.9a) and the constraint (3.9b). Specifically, we have to formulate the requested load-profiles in terms of a decisional variable to be minimized in order to achieve some stated performances.

In the following sections, four different solutions are proposed.

3.3.1 Non-Linear Program NLP Formulation

The first approach to this generic formulation is a direct consequence of the statement of the problem, as described in [1]. As discussed before, we aim to minimize the delays in water supply to the users. Furthermore we introduce the vector $l^{(k:k+n_y-1)}$ representing the load that is requested by the users at time k. Obviously $d^{(k:k+n_y-1)} = l^{(k:k+n_y-1)}$ would be the optimal solution of the problem: this would mean that all the users are supplied with water at the time they demand it.

Let τ_i denote the delivery delay in water supply to the i-th request. Particularly τ_i represents the interval between the time $k+k_i^-$ the load-profile $l_i^{(k+k_i^-:k+k_i^+)}$ is requested to start and the time it is eventually scheduled to start. The overall objective is, hence, to minimize the sum of all the delivery delays in order to

satisfy the users. The problem is, therefore, formulated as a non-linear one.

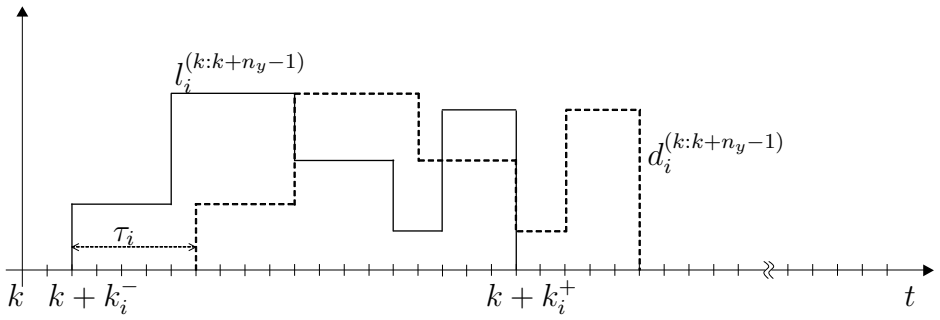


Figure 3.1: Temporal representation of a requested load-profile (solid line) and the delayed version (dashed line) over the horizon

As discussed, the delivery delays τ_i are decision variables: we can indeed express the scheduled load $d_i(k)$ as a delayed version of the requested profile $l_i(k)$. Particularly we can write $d_i(k) = l_i(k - \tau_i)$.

We now introduce the backward shift operator represented by J which is the lower shift matrix:

$$J = \begin{bmatrix} 0 & \cdots\cdots & 0 \\ 1 & 0 & \vdots \\ & \ddots & \ddots \\ 0 & & 1 \quad 0 \end{bmatrix}^{n_y \times n_y} \tag{3.10}$$

It is immediately clear that:

$$d_i^{(k:k+n_y-1)} = J^{\tau_i} l_i^{(k:k+n_y-1)} \tag{3.11}$$

We can generalize this result summing up all the requests. Let Π_i be a mapping matrix stacking the variables in a appropriate way, so that:

$$d^{(k:k+n_y-1)} = \begin{bmatrix} d_1^{(k:k+n_y-1)} \\ \vdots \\ d_N^{(k:k+n_y-1)} \end{bmatrix} = \sum_{i=1}^{N} \Pi_i d_i^{(k:k+n_y-1)} = \sum_{i=1}^{N} \Pi_i J^{\tau_i} l_i^{(k:k+n_y-1)} \tag{3.12}$$

Combined with the predictive model (3.5), this formulation provides a prediction of the state as a function of the delays. Note that:

- since the delay is a discrete variable ($\tau_i \in \mathbb{N}_0$), like the predictive model, it is a multiple of the sampling time T_y

- defined T_{max} as the longest transient of the dynamical system to be subject to the load schedule, it is necessary that:

$$\tau_i \leq n_y - T_{max} - k_i^+ \tag{3.13}$$

  for the load and the response transient to be entirely contained inside the predictive horizon.

As we said above the performance we want to achieve, in terms of quality of service, is the minimization of a measure of the overall delivery delays. Thus we consider the following sum-separable cost function:

$$\sum_{i=1}^{N} h_i(\tau_i) \tag{3.14}$$

where $h_i : \mathbb{N}_0 \mapsto \mathbb{R}$ for all $i \in \{1, \ldots, N\}$ is any function that penalizes the corresponding delivery delay. For example, we can simply consider an increasing linear function: $h_i(\tau_i) = a_i\tau_i$, where $a_i > 0$ for all i. We will deal with this solution ahead. In this formulation we will adapt the generic cost function (3.14).

The load scheduling task can now be formulated as the following mixed-integer non-linear program:

$$\begin{aligned}
\underset{\tau_i}{minimize} \quad & \sum_{i=1}^{N} h_i(\tau_i) && \text{(3.15a)}\\
\text{subjet to} \quad & \hat{y}^{(k+1:k+n_y)} = \Gamma x^{(k)} + \Omega r^{(k:k+n_y-1)} + \\
& \qquad + \Psi \tilde{d}^{(k:k+n_y-1)} + \Psi \sum_{i=1}^{N} \Pi_i J^{\tau_i} l_i^{(k:k+n_y-1)} && \text{(3.15b)}\\
& \underline{y} \leq \hat{y}^{(k+1:k+n_y)} \leq \bar{y} && \text{(3.15c)}\\
& \tau_i \leq n_y - T_{max} - k_i^+ && \text{(3.15d)}\\
& \tau_i \in \mathbb{N}_0 \ \text{ for } \ i = \{1, \ldots, N\} && \text{(3.15e)}
\end{aligned}$$

Solving the problem (3.15) we obtain a vector of N deliver delays. This

solution must belong to the set S^y defined by the constraint (3.15d). Note that the set S^y is finite because the number of possible values for the delays is also finite. This observation implies that the number of elements in the set S^y depends on the upper bound given by (3.15d) and the number of loads N to be scheduled. Since the upper bound for τ_i depends on the prediction horizon, the choice of n_y could directly influence the feasibility of the problem, if it makes the set S^y too small.

Since the number of solution is finite, a way of solving this problem is to compute the prediction (3.15b) for all the elements in the set S^y, check if (3.15c) is violated or not, and in the case the solution fulfills the constraint (3.15c), compute the associated cost (3.15a). An "optimal" solution would belong to the subset of solutions with the smallest value of cost function. Obviously this approach is time consuming and the computational complexity is an issue, especially because the magnitude of S^y can be too large.

3.3.2 A $\{0,1\}$ Integer Linear Program ILP Formulation

We will now reformulate the problem stated in the previous chapter as a $\{0,1\}$ linear program, via a suitable change of variables. To do this, we will focus on the relationship between the requested load $l_i^{(k:k+n_y-1)}$ and the scheduled one $d_i^{(k:k+n_y-1)}$. Specifically, for the i-th load, we can choose a set of n_i admissible delivery delays and we call them $\{\omega_i^1, \dots \omega_i^{n_i}\}$. This means that the event "the requested load-profile is delayed by ω_i^t" is associated to a delivery delay of $\omega_i^t \geq 0$. This can be represented by a binary variable that assumes the value '1' if the events occurs and '0' otherwise. Let $z_i \in \{0,1\}^{n_i}$ be a vector of binary decisional variables. We can build the matrix $M_i \in \mathbb{R}^{n_y \times n_i}$ representing all the possible delayed version of the requested profile $l_i^{(k:k+n_y-1)}$, i.e.:

$$M_i = \begin{bmatrix} J^{\omega_i^1} l_i^{(k:k+n_y-1)} & J^{\omega_i^2} l_i^{(k:k+n_y-1)} & \dots & J^{\omega_i^{n_i}} l_i^{(k:k+n_y-1)} \end{bmatrix} \tag{3.16}$$

We can see that the prediction constraint (3.15b) (in particular the expression of the requested load) can be written as a linear function in the new decisional variable z_i, where:

$$J^{\tau_i} l_i^{(k:k+n_y-1)} = M_i z_i \tag{3.17}$$

It is very important that the vector of variables z_i has only one element equal to '1' because we can choose only one profile among all the possible ones. Thus the constraint $\sum_{j=1}^{n_i} z_{i,j} = 1$ must be added. The cost function becomes:

$$\underset{z_i}{minimize} \sum_{i=1}^{N} C_i^T z_i \tag{3.18}$$

where $C_i^T = \begin{bmatrix} h_i(\omega_i^1) & h_i(\omega_i^2) & \cdots & h_i(\omega_i^{n_i}) \end{bmatrix}$. Here $h_i(\omega_i^t)$ is the cost associated to the water supply delayed of ω_i^t. Thanks to this, we can state the new integer-linear program as follows:

$$\begin{aligned}
\underset{z_i}{minimize} \quad & \sum_{i=1}^{N} C_i^T z_i && \text{(3.19a)}\\
\text{subject to} \quad & \hat{y}^{(k+1:k+n_y)} = \Gamma x^{(k)} + \Omega r^{(k:k+n_y-1)} + \\
& \qquad + \Psi \tilde{d}^{(k:k+n_y-1)} + \Psi \sum_{i=1}^{N} \Pi_i M_i z_i && \text{(3.19b)}\\
& \underline{y} \leq \hat{y}^{(k+1:k+n_y)} \leq \bar{y} && \text{(3.19c)}\\
& \sum_{j=1}^{n_i} z_{i,j} = 1 && \text{(3.19d)}\\
& z_i \in \{0,1\}^{n_i} \ \text{ for } \ i = \{1,\ldots,N\} && \text{(3.19e)}
\end{aligned}$$

The problem (3.19) is computationally easier to solve with respect to the non-linear one (3.15) although the two are equivalent to each other. Indeed, let $\tilde{S}_y$ be the set of n_i admissible delays of all the requested profiles. If the condition $S_y = \tilde{S}_y$ holds the problem (3.19) is equivalent to (3.15).

However, for large-scale irrigation networks and in the case of large prediction horizons, the integer program formulated in this section is still too computationally demanding. Therefore, our objective in the next sections, is to simplify it introducing some approximations and new assumptions. In this way, the problem will be reformulated as a more standard linear program.

3.3.3 A Relaxed Program RP1 Formulation

In the previous section an integer-linear program is presented. This solution is obtained by assuming a feasible set of possible delays and building the matrix M_i having in its column all the feasible delayed version of the requested profile. In this case the profile (i.e. the shape) of the requested load $l_i^{(k:k+n_y-1)}$ is preserved due to the fact that the decision variable z_i is binary. In this section, we start relaxing the constraint (3.19e), and replacing the integer variable z_i with the real one ξ_i:

$$z_i \in \{0,1\}^{n_i} \rightarrow 0 \leq \xi_{i,j} \leq 1 \quad \text{for} \quad j = \{1,\dots,n_i\} \tag{3.20}$$

such that the optimization program becomes:

$$\underset{\xi_i}{minimize} \quad \sum_{i=1}^{N} f_i(\xi_i) \tag{3.21a}$$

$$\text{subject to} \quad \hat{y}^{(k+1:k+n_y)} = \Gamma x^{(k)} + \Omega r^{(k:k+n_y-1)} + \\ + \Psi \tilde{d}^{(k:k+n_y-1)} + \Psi \sum_{i=1}^{N} \Pi_i M_i \xi_i \tag{3.21b}$$

$$\underline{y} \leq \hat{y}^{(k+1:k+n_y)} \leq \bar{y} \tag{3.21c}$$

$$\sum_{j=1}^{n_i} \xi_{i,j} = 1 \tag{3.21d}$$

$$0 \leq \xi_{i,j} \leq 1 \quad \text{for} \quad j = \{1,\dots,n_i\}, i = \{1,\dots,N\} \tag{3.21e}$$

In this way we do not take into account of the shape of the profile in the constraints. Specifically, while z_i plays the role of selector of the entire profile, the new variable ξ_i chooses the amount of "load" taken by each possible delayed profile.

Let us see a graphic example. Given a requested profile, and three admissible delayed version ($n_i = 3$), a feasible solution to the optimization problem (3.21) could be $\xi_i = \begin{bmatrix} 0.2 & 0 & 0.8 \end{bmatrix}^T$

We can notice from Figure 3.2 that the shape of the scheduled profile is different from the original requested one. We have to build a new objective function in order to penalize not only the delays in water supply, but also the

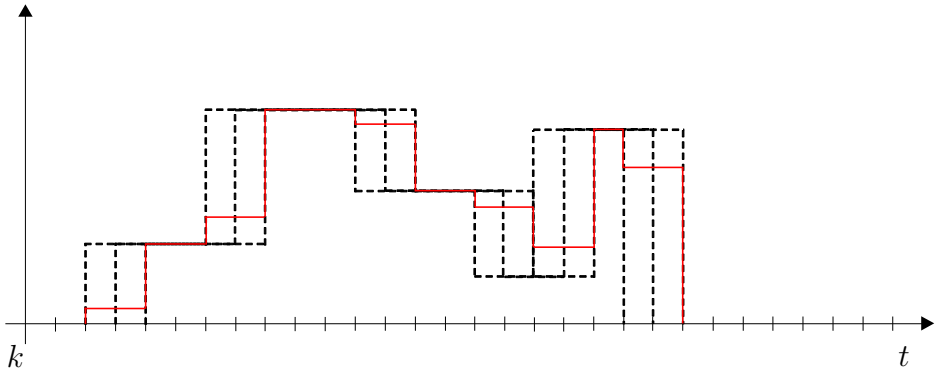


Figure 3.2: Graphical possible solution to the problem (3.21); the dashed lines represent the three admissible delayed version of the requested profile, the red one is the feasible solution

shape of the scheduled profiles.

Section 3.4.3 will be devoted to this issue.

3.3.4 A Second Relaxed Program RP2 Formulation

The solution shown in Section 3.3.3 is obtained developing the integer-linear problem described in [1]. In this section we express another formulation. Assuming that the i-th farmer demands for water in terms of a constant flow-rate of amplitude A_i for a time interval T_i, to be preferably scheduled at time k, we can easily re-write the problem depending on a new decision variable $\zeta_i^{(k:k+n_y-1)}$. While ξ_i chooses the amount of "load" taken by each admissible delayed profile, $\zeta_i^{(k:k+n_y-1)}$ (for the sake of simplicity we will use the reduced notation ζ_i) represents the amount of water to be supplied at every sampling time. The problem becomes:

$$\underset{\zeta_i}{minimize} \quad \sum_{i=1}^{N} g_i(\zeta_i) \tag{3.22a}$$

$$\text{subject to} \quad \hat{y}^{(k+1:k+n_y)} = \Gamma x^{(k)} + \Omega r^{(k:k+n_y-1)} + \\ + \Psi \tilde{d}^{(k:k+n_y-1)} + \Psi \sum_{i=1}^{N} \Pi_i \zeta_i \tag{3.22b}$$

$$\underline{y} \leq \hat{y}^{(k+1:k+n_y)} \leq \bar{y} \tag{3.22c}$$

$$\sum_{j=1}^{n_y - T_{max}} \zeta_{i,j} = A_i T_i \tag{3.22d}$$

$$0 \leq \zeta_{i,j} \leq A_i \quad \text{for} \quad j = \{1, \ldots, n_y - T_{max}\}, \quad i = \{1, \ldots, N\} \tag{3.22e}$$

Note that:

- in (3.22b) the predicted output is still a linear combination of the free variable ζ_i,
- the constraint (3.22d) guarantees that the amount of provided water is equal to the requested one,
- (3.22e) is necessary in order to fix A_i as the maximum supplied flow-rate,
- obviously, as discussed in Section 3.3.1, the longest transient T_{max} of the dynamical system must be entirely contained in the predictive horizon.

Hence, the condition $\zeta_{i,j} = 0$ for $j = \{n_y - T_{max}, \ldots, n_y\}$ must hold. This is imposed by the constraint (3.22d).

With this new approach the construction of the matrices M_i (containing all the admissible delayed version of the requested profile) is not required. Anyway, the problems of this strategy will devised later on.

3.4 Choices of the Objective Functions

As discussed in the previous sections, we want to minimize any index representing a given performance in order to maximize the quality of service. Thus, our matter is the construction of suitable objective functions leading to this attainment, for all the problems discussed before.

3.4.1 Objective Function for Non-Linear Program

We have already suggested a formulation for the cost function in the non-linear case in section 3.3.1. In this case, the only performance, we have to minimize, is the delivery delay in water-supply. Due to the structure of the problem, the functions to be minimized can be simply expressed as:

$$h_i(\tau_i) = a_i \tau_i \tag{3.23}$$

where $a_i > 0$ for all i.

Thus, the objective function will be the sum of (3.23) for $i = 1, \ldots, N$, i.e.:

$$\underset{\tau_i}{minimize} \quad \sum_{i=1}^{N} a_i \tau_i \tag{3.24}$$

In this way, we are minimizing the overall delivery delays among all the pools. By the way, this strategy will be not contemplated in the simulation tests because of the computational complexity of such a program. Moreover, in Section 3.3.2, we demonstrated that the non-linear formulation is equivalent to the integer linear one.

3.4.2 Objective Function for Integer Linear Program

We can apply the same rationale adopted for the integer-linear program. As shown in Section 3.3.2, the objective function can be represented by the sum of the scalar product between a vector C_i (whose elements are functions h_i of the admissible delays ω_i^t) and the vector of decisional variable z_i. In particular, we will use a vector C_i defined as $C_i = \begin{bmatrix} 1 & 2 & \cdots & n_i \end{bmatrix}^T$. The objective function is:

$$\underset{z_i}{minimize} \quad \sum_{i=1}^{N} C_i^T z_i \tag{3.25}$$

It penalizes the choice of a delayed admissible profile linearly with the supply delay.

3.4.3 Objective Functions for Relaxed Programs

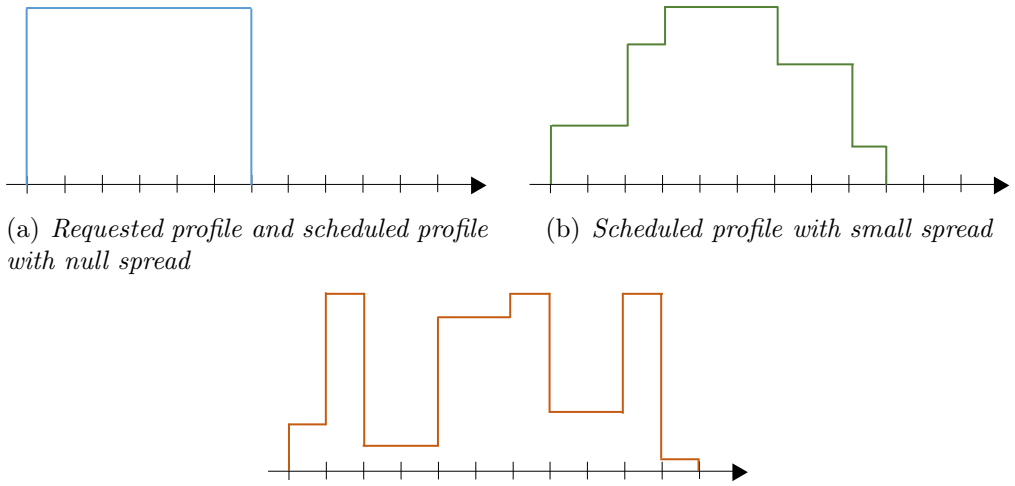
Concerning the relaxed formulations in Sections 3.3.3 and 3.3.4 the shape of the scheduled profile is not guaranteed to be fixed, as explained in Section 3.3.3. Consequently, in addition, the duration of the scheduled off-take is not guaranteed to be equal to the duration of the profile requested by the user. Therefore, in this section we propose to use an additional penalizing term to the cost function, properly tailored for the problem. Specifically,

1. as in the previous strategies, we have to minimize the delivery delays in terms of the time in which the amount of requested water is supplied. This first term can be expressed as:

$$\underset{\xi_i}{minimize} \quad \sum_{i=1}^{N} \alpha_i C_i^T \xi_i \tag{3.26}$$

   where C_i is the cost vector defined for the integer linear program.

2. we need to introduce a new term $P(\xi)$ in the objective function minimizing the variation of the load shape with respect to the one requested by the user (a qualitative idea is given in Figure 3.3). In order to maximize the quality of service, we should supply water with a duration of the scheduled profile as similarly as possible to the requested one. Indeed, the little is the variation, the minimal would be the difference between the interval



(a) *Requested profile and scheduled profile with null spread*

(b) *Scheduled profile with small spread*

(c) *Scheduled profile with large spread*

Figure 3.3: Example of spread in different scheduled profiles

$[k + k^-, k + k^+]$ and the lapse of time among which the water is actually supplied.

The entire objective function, hence, will be:

$$\underset{\xi_i}{minimize} \quad \sum_{i=1}^{N} \alpha_i C_i^T \xi_i + \beta_i P(\xi) \tag{3.27}$$

where α_i and β_i are parameters to be chosen to give more importance to one term rather than the other. In the next section, we will discuss two suitable examples for $P(\xi)$ addressing the problem.

First solution: Variance

As discussed, our scope is to find a function $P(\xi)$ minimizing the variation of the shape of the load profile with respect to the request.

Recall now that the variable $\xi_{i,j}$ can be interpreted as the weight (i.e., $\xi_{i,j} \in [0, 1]$) associated to the j-th profile, and that, in view of constraint (3.21d), i.e.:

$$\sum_{j=1}^{n_i} \xi_{i,j} = 1, \tag{3.28}$$

it can be interpreted as the probability of profile j. We now define with μ_i (in the interval $[1, n_i]$) the weighted mean value of index j, i.e., $\mu_i = \sum_{j=1}^{n_i} \xi_{i,j} j$. The latter can be viewed as the "average" profile (recall, however, that rational values are possible for μ_i).

Now define

$$\mathrm{var}_i(\xi_i) = \sum_{j=1}^{n_i} \xi_{i,j}(j - \mu_i)^2 \tag{3.29}$$

the "variance" of index j. Remark that the latter is indeed equal to zero, if $\xi_{i,j} = 1$, for some value of j, corresponding to the case where the shape of the requested profile is preserved. On the other hand, it increases as the number of non-zero values of $\xi_{i,j}$ grows (or if the values of j for which $\xi_{i,j} \neq 0$ span the whole set $\{1, n_i\}$), which corresponds to the case where the shape of the requested profile is modified, and the duration of the load signal increases. For the above reasons, we set

$$P_i(\xi_i) = var_i(\xi_i) \tag{3.30}$$

We obtain that, in view of equation (3.29):

$$\mathrm{var}_i(\xi_i) = \sum_{j=1}^{n_i} \xi_{i,j} j^2 + \sum_{j=1}^{n_i} \xi_{i,j} \Big(\sum_{j=1}^{n_i} \xi_{i,j} j \Big)^2 - 2 \sum_{j=1}^{n_i} \xi_{i,j} j \sum_{j=1}^{n_i} \xi_{i,j} j \tag{3.31}$$

Applying (3.21d), the variance becomes:

$$\mathrm{var}_i(\xi_i) = \sum_{j=1}^{n_i} \xi_{i,j} j^2 - \Big(\sum_{j=1}^{n_i} \xi_{i,j} j \Big)^2 \tag{3.32}$$

Remark: to better understand the rationale underlying the above idea, we note that it has a clear mechanical interpretation. Let $\xi_{i,j}$ be the j-th point mass of a system composed of n_i point masses. If we imagine the index j to be the x-axis coordinate of the j-th mass $\xi_{i,j}$ with respect to the origin $j = 0$, we can calculate the centre of mass COG as:

$$COG_i = \frac{1}{\sum_{j=1}^{n_i} \xi_{i,j}} \sum_{j=1}^{n_i} \xi_{i,j} j \tag{3.33}$$

From (3.21d), we know that $\sum_{j=1}^{n_i} \xi_{i,j} = 1$ for all i. We can now calculate the "moment of inertia" of the system with respect to the COG:

$$\mathcal{J}_i = \sum_{j=1}^{n_i} \xi_{i,j} \Big(j - \sum_{j=1}^{n_i} \xi_{i,j} j \Big)^2 \tag{3.34}$$

As we can see, (3.34) is identical to expression (3.29) of the variance. Developing (3.34), we obtain (3.32) that has been calculated from the definition of variance.

Developing (3.32), we obtain the following quadratic equation as function of the free variable ξ_i:

$$\mathrm{var}(\xi_i) = F_i^T \xi_i - \xi_i^T H_i \xi_i \tag{3.35}$$

where:

$$F_i = \begin{bmatrix} 1 & 4 & 9 & \cdots & n_i^2 \end{bmatrix}, \quad H_i = \begin{bmatrix} 1 & 2 & 3 & \cdots & n_i \\ 2 & 4 & 6 & & \vdots \\ 3 & 6 & 9 & & \\ \vdots & & & \ddots & n_i^2 - n_i \\ n_i & \cdots & & n_i^2 - n_i & n_i^2 \end{bmatrix} \tag{3.36}$$

The new objective function, hence, will be:

$$\underset{\xi_i}{minimize} \quad \sum_{i=1}^{N} \Big(\underbrace{\alpha_i C_i^T \xi_i}_{\text{(a)}} + \underbrace{\beta_i \big(F_i^T \xi_i - \xi_i^T H_i \xi_i \big)}_{\text{(b)}} \Big) \tag{3.37}$$

This is a quadratic function in the free variable ξ_i, minimizing both the delays (3.37a) and the spread (3.37b). However, we can see that the quadratic term is indefinite. Therefore, the related optimization problem is rather non-standard, in view of the fact that it is non-convex. This statement is very important to note and its consequences will be dealt with later on. We are going to show the results obtained with such an objective function in Section 3.5.4.

Second solution: a Customized Objective Function

We can build an alternative customized quadratic term in order to penalize the dispersion of ξ_i around a central point.

We develop a generic symmetric quadratic term, i.e:

$$f(\xi_i) = \xi_i^T \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n_i} \\ a_{2,1} & a_{2,2} & & \vdots \\ \vdots & & \ddots & \\ a_{n_i,1} & \cdots & & a_{n_i,n_i} \end{bmatrix} \xi_i \tag{3.38a}$$

$$= a_{1,1}\xi_{i,1}^2 + 2a_{1,2}\xi_{i,1}\xi_{i,2} + a_{2,2}\xi_{i,2}^2 + \cdots + a_{n_i,n_i}\xi_{i,n_i}^2 \tag{3.38b}$$

Thereby, in order to foster the solution ξ_i having only one element equal to 1 and all the remaining ones equal to 0, we give no weight to the addends in the form $a_{j,j}\xi_{i,j}^2$: thus, we impose the elements on the principal diagonal $a_{j,j} = 0$. Concerning the other addends $2a_{j,j+\bar{j}}\xi_{i,j}\xi_{i,j+\bar{j}}$, the weight must be proportional to the difference $\bar{j}$ between the indexes: we can simply thrust upon $a_{j,j+\bar{j}}$ to be equal to $\bar{j}$. Thus, the new matrix $\mathcal{H}_i$ will be:

$$\mathcal{H}_i = \begin{bmatrix} 0 & 1 & 2 & \cdots & n_i \\ 1 & 0 & 1 & & \vdots \\ 2 & 1 & 0 & & \\ \vdots & & & \ddots & 1 \\ n_i & \cdots & & 1 & 0 \end{bmatrix} \tag{3.39}$$

Also $\mathcal{H}_i$ is an indefinite quadratic form, and the same considerations of H can be applied.

3.5 Simulation Tests on the Centralized Problem

In this section the results obtained with the different strategies discussed in the previous sections, are shown. As we explained in Chapter 2, our case of study is the system composed by the two last pools of the East Goulburn Main channel.

3.5.1 Data

In the following simulations, the sampling time T_y is equal to 10 minutes and the prediction horizon is $n_y = 480$ periods of duration T_y. We suppose that the off-takes l are demanded (and preferably scheduled) at time $k = 0$. According to [9], we have:

	pool 1			pool 2		
$\underline{y}_i$	9.4			9.5		
$\bar{y}_i$	9.7			9.7		
$r_i^{(k:k+n_y-1)}$	9.5	if	$0 \leq t \leq 3600$	9.56	if	$0 \leq t \leq 1200$
	9.6	if	$t > 3600$	9.62	if	$t > 1200$
$d_i^{(k:k+n_y-1)}$	0	if	$0 \leq t \leq 100$	0	if	$0 \leq t \leq 100$
	10	if	$100 < t \leq 700$	10	if	$100 < t \leq 700$
	15	if	$700 < t \leq 1600$	15	if	$700 < t \leq 1900$
	5	if	$1600 < t \leq 1900$	5	if	$1900 < t \leq 3400$
	0	if	$t > 1900$	0	if	$t > 3400$

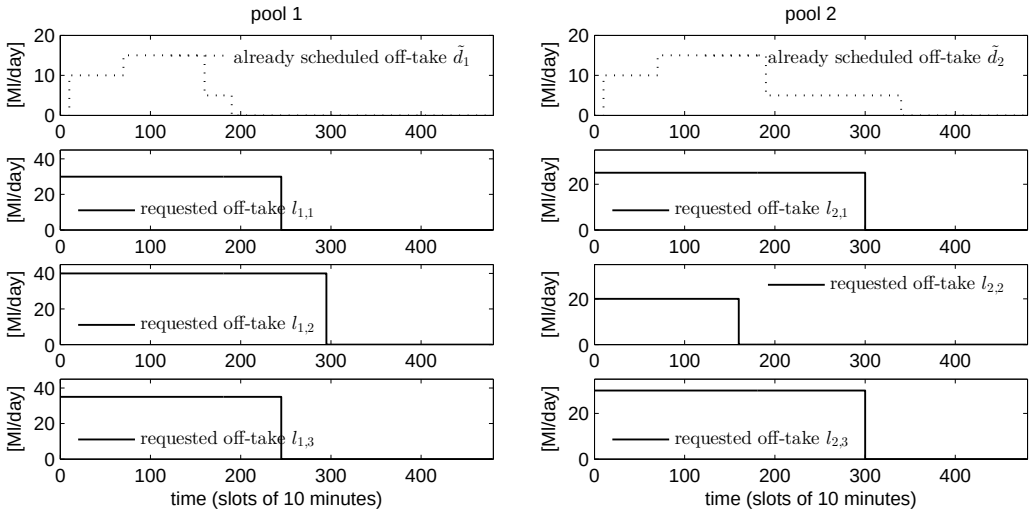
Table 3.1: Parameters for simulation tests, source [9]

Each off-take is defined in terms of the requested flow and duration. We will use the following values:

	pool 1			pool 2		
off-take	1	2	3	1	2	3
Flow $[Ml/\text{day}]$	30	40	35	25	20	30
Duration [10min]	245	295	245	300	160	300

Table 3.2: Parameters of off-takes for simulation tests, source [9]

According to the parameters in Tables 3.1 and 3.2, the requested off-takes are shown in Figure 3.4.



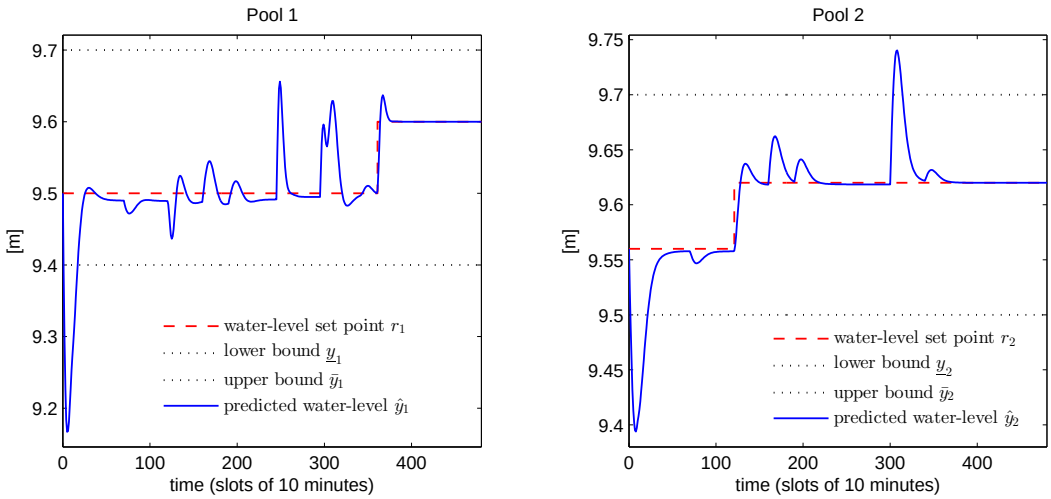
(a) *Requested (not-scheduled) off-takes in pool* 1

(b) *Requested (not-scheduled) off-takes in pool* 2

Figure 3.4: Requested off-takes

Directly applying the requested profiles of Figure 3.4, i.e. $d = l$, on the predictor, the predictions of the water-level in both pools are shown in Figure 3.5.

It now clearly appears the importance of scheduling the farmers' demands in order to guarantee the fulfillment of the constraints. Indeed, if the water is supplied as requested (supposing at $k = 0$), the lower bounds of both the pools



(a) *Predicted water-level in pool 1*

(b) *Predicted water-level in pool 2*

Figure 3.5: Predicted water-level response to not-scheduled profiles

and the upper bound of the pool 2 are not fulfilled (i.e. at time $k = 1$ and $k = 303$). This situation leads to the problems described in Section 3.2.

In the next section we will show the feasible results from the optimization problems discussed in the previous sections. The objective functions, devised in Section 3.4, will be adopted.

All the results are obtained by using a laptop with a *new generation CPU Intel CORE i7 3630QM (2.4GHz)* and a *8Gb RAM*.

3.5.2 Simulation Test on ILP

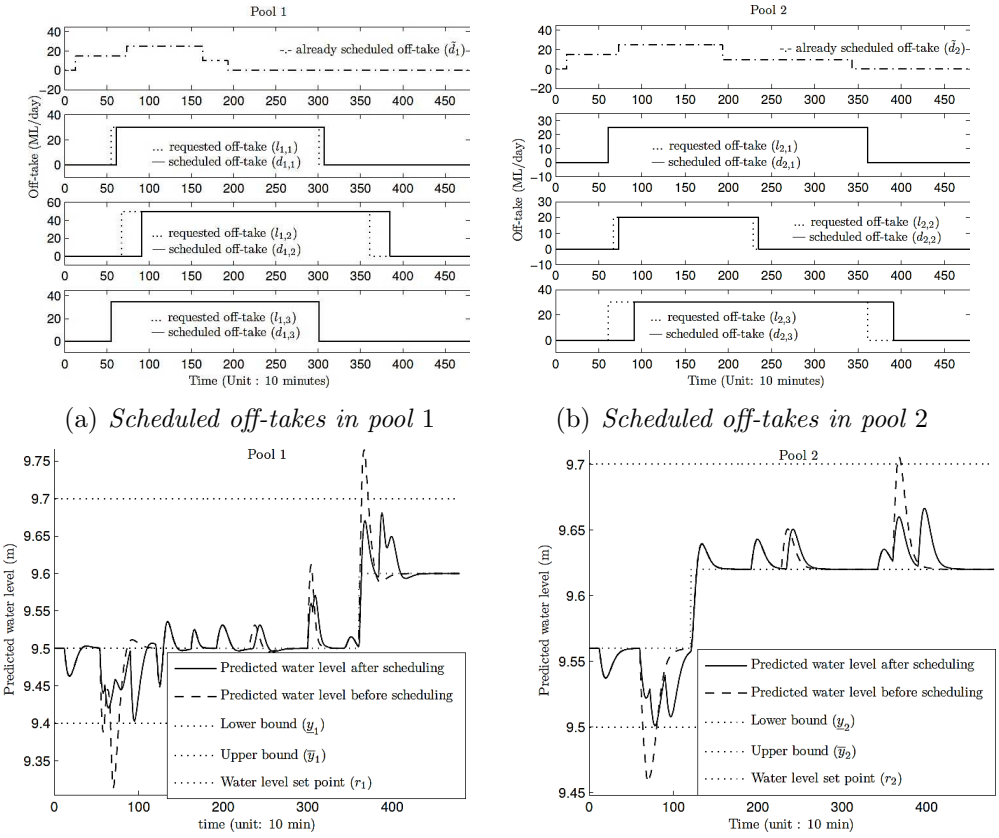
In order to solve the ILP, we need to define the set $\tilde{S}_y$ corresponding to the n_i admissible delayed profiles. This value must be selected in order to guarantee that the constraint (3.13) is fulfilled. From Table 3.2, we know that the maximum duration of the off-takes is $k^+_{max} = 300$ and we consider the longest dynamical transient in the pools T_{max} to be equal to 20 (see Figure 3.4). The maximum admissible delivery delay, thus, will be:

$$\tau_{i,max} = n_i = 480 - 300 - 20 = 160 \quad \text{for} \quad i = 1, \ldots, N \tag{3.40}$$

Solving the integer linear program (using the function *bintprog.m* of the *Optimization Toolbox* of MATLAB) for two pools and three off-takes from each pool (see Table 2.1) with our laptop is intractable, since the computational cost related to this strategy is very high. For this reason, in Figure 3.6 the results, as shown in [1], are illustrated. Note that, in this case, the off-takes are not necessarily requested at $k = 0$.

We can conclude that:

- both the upper and lower bounds of each pool are fulfilled (Figures 3.6.(c) and 3.6.(d))
- the shape of the requested off-takes are not modified.



(a) *Scheduled off-takes in pool* 1

(b) *Scheduled off-takes in pool* 2

(c) *Predicted water-level in pool 1*

(d) *Predicted water-level in pool 2*

Figure 3.6: Scheduled off-takes with integer-linear program ILP, source [1]

3.5.3 Simulation Test on RP1 with the Linear Objective Function LRP1

In this section, we show the results obtained solving RP1 with the linear objective function developed in Section 3.4, i.e.:

$$\underset{\xi_i}{minimize} \quad \sum_{i=1}^{N} \alpha_i C_i^T \xi_i \tag{3.41}$$

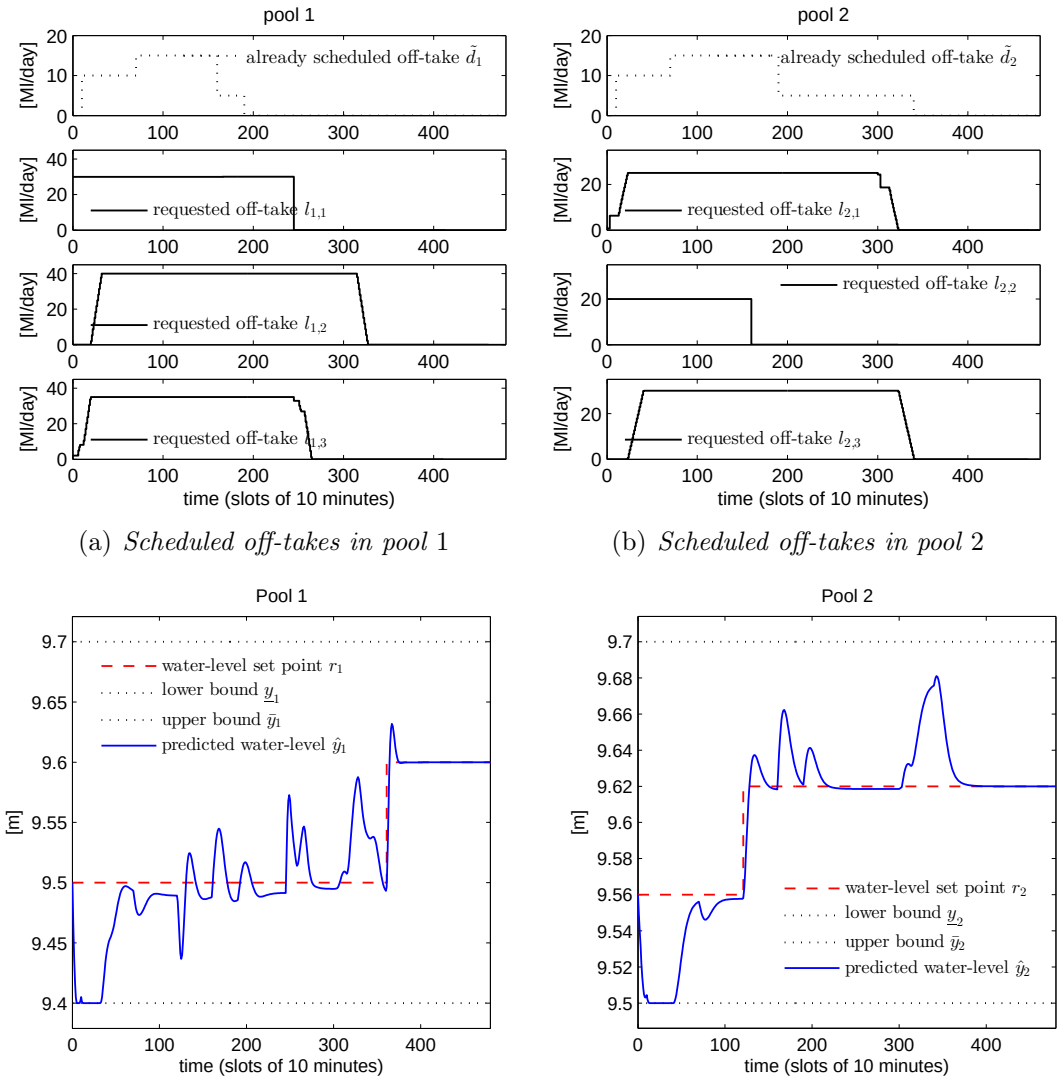
We denote this problem LRP1.

As discussed in Section 3.4.3, LRP1 minimizes the delivery delays without taking into account of the shape of the profiles. Using the function *linprog.m* of the *Optimization Toolbox* of MATLAB, the results are illustrated in Figure 3.7.

Even if the scheduled profile shape is not preserved, we can calculate the overall delay, as the sum of the time instants in which the off-takes start to be delivered $\bar{t}_i$ for $i = 1, \ldots, N$, i.e.:

$$\sum_{i=1}^{N} \bar{t}_i = 43 \ \text{ seconds} \tag{3.42}$$

The estimated computational time is about 42 seconds.



(a) *Scheduled off-takes in pool* 1

(b) *Scheduled off-takes in pool* 2

(c) *Predicted water-level in pool 1*

(d) *Predicted water-level in pool 2*

Figure 3.7: Scheduled off-takes with linear relaxed program LRP1

3.5.4 Simulation Test on RP1 with the Quadratic Objective Function QRP1

In this section, we show the solutions obtained by RP1 with the quadratic objective function coming from the variance, i.e.:

$$\underset{\xi_i}{minimize} \quad \sum_{i=1}^{N}\Big(\alpha_i C_i^T \xi_i + \beta_i\big(F_i^T \xi_i - \xi_i^T H_i \xi_i\big)\Big) \tag{3.43}$$

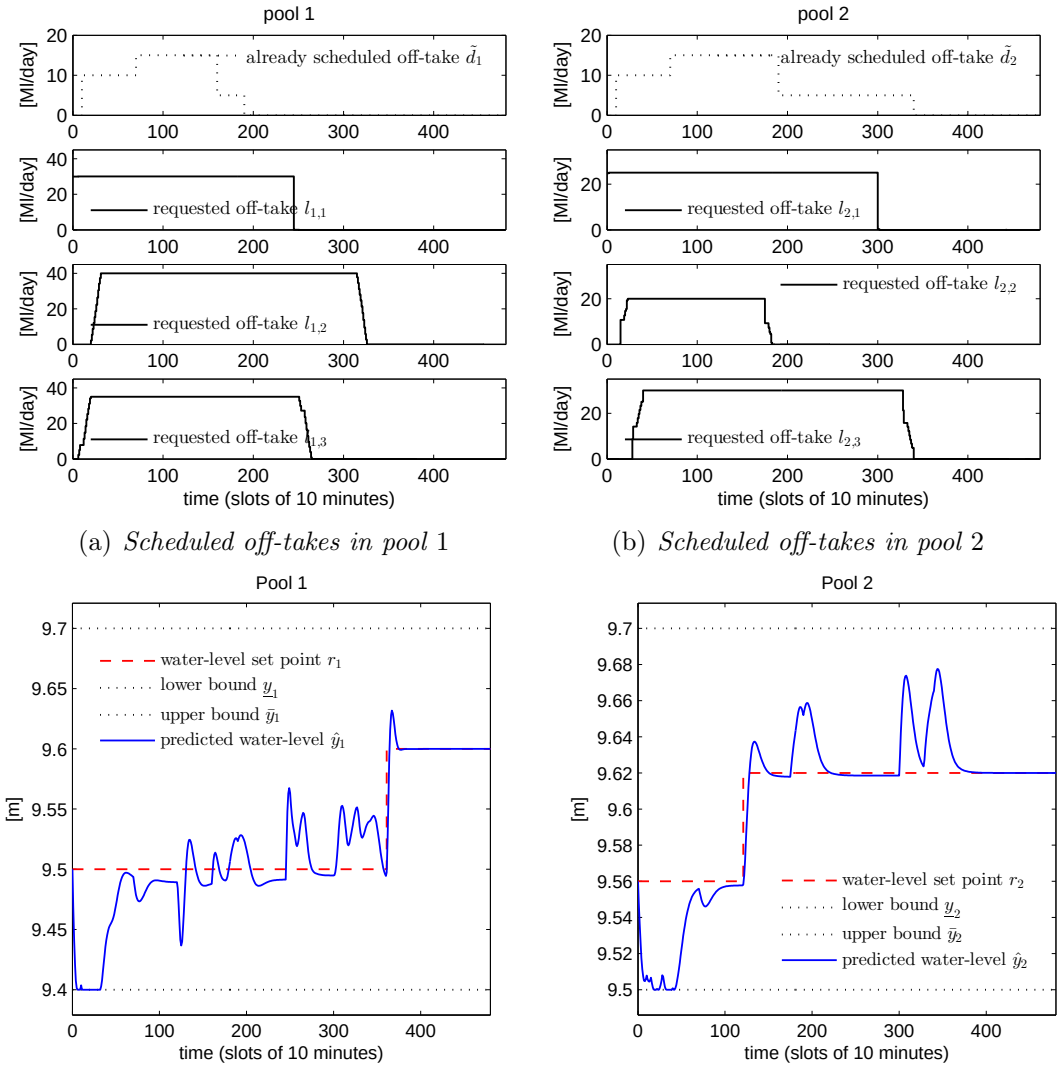
It is apparent that, if we set $\alpha_i = 0$, there is a number of equivalent optimal solutions, and therefore the problem is not well-posed, as far as the cost function is concerned. Therefore, this case is disregarded and we focus on the case where $\alpha_i \neq 0$. In particular, if we test QRP1 with $\alpha_i = 10$ and $\beta_i = 1$ for $i = 1, \ldots, N$, using the function *quadprog.m* of the *Optimization Toolbox* of MATLAB, we achieve the results shown in Figure 3.8.

In this case, the overall delay in water supply is 69 seconds. Moreover, in this case, we can calculate the total variance, as previously illustrated, and defined as the sum of the variance of each profile (3.35):

$$\sum_{i}^{N} \mathrm{var}(\xi_i) = 58.7 \tag{3.44}$$

The computational time is almost 85 seconds.

These results show that the solutions, obtained optimizing the variance (in addition to the delivery delays minimization), maintain a better profile shape. Nevertheless, there are some problems, related to such a solution, that will be discussed in Section 3.5.6.



(a) *Scheduled off-takes in pool* 1

(b) *Scheduled off-takes in pool* 2

(c) *Predicted water-level in pool 1*

(d) *Predicted water-level in pool 2*

Figure 3.8: Scheduled off-takes with quadratic program QRP1

3.5.5 Simulation Test on RP2 with the Linear Objective Function LRP2

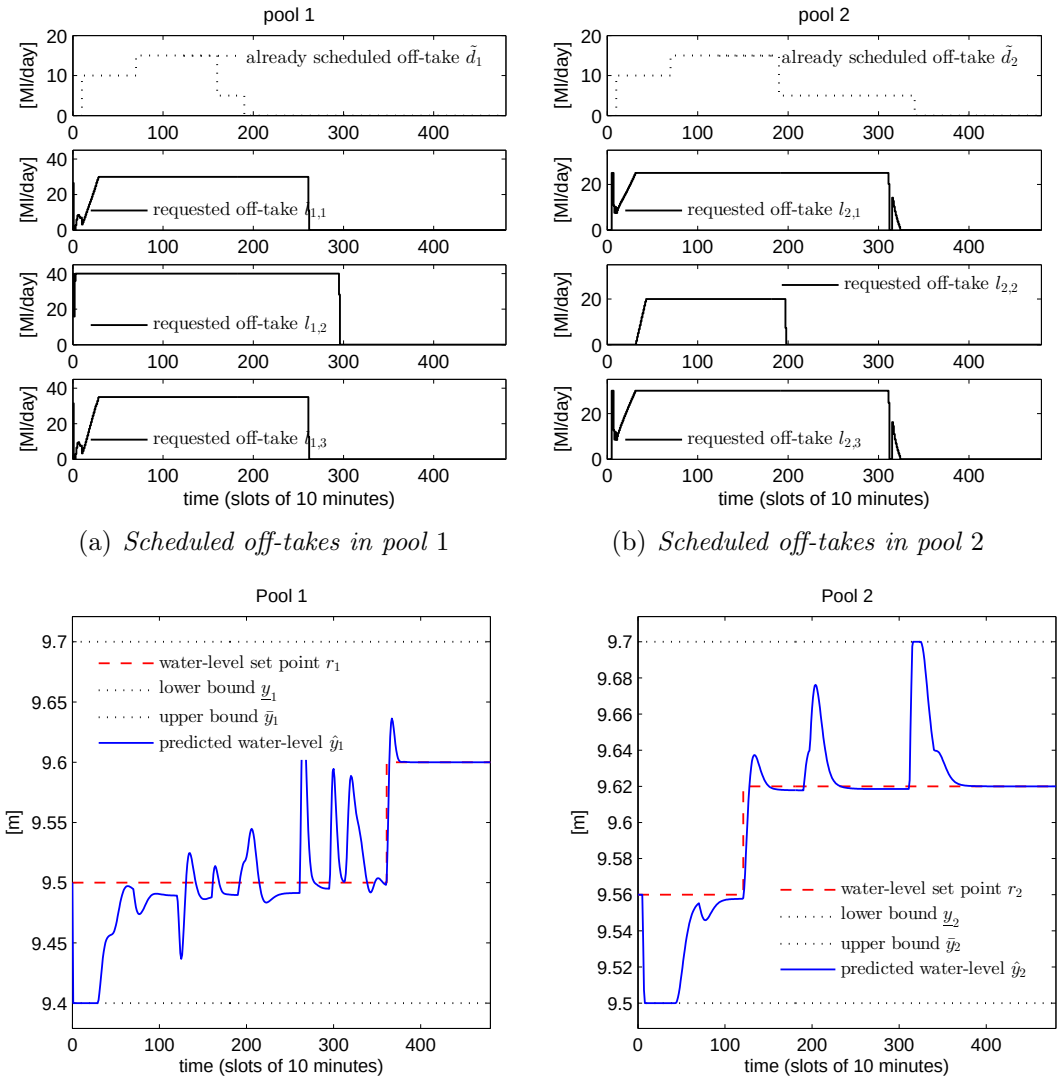
In this section, we test the solution RP2, focusing solely on the linear objective function:

$$\underset{\zeta_i}{minimize} \quad \sum_{i=1}^{N} C_i^T \zeta_i \tag{3.45}$$

As anticipated in Section 3.3.4, there are problems related to this formulation, in terms of scheduled profile shape. In Figure 3.9, the solution of LRP2, using the fuction *linprog.m*.

We can, immediately, note that the shape of the scheduled off-takes is much more distorted than using the relaxed solutions shown in the previous section.

The computational time is about 38 seconds.



(a) *Scheduled off-takes in pool* 1

(b) *Scheduled off-takes in pool* 2

(c) *Predicted water-level in pool 1*

(d) *Predicted water-level in pool 2*

Figure 3.9: Scheduled off-takes with linear relaxed program LRP2

3.5.6 Comparison between the Different Formulations of Centralized Optimization Problem

In this section, we want to compare the different solutions in order to discuss the advantages and the drawbacks of the mentioned ones and, if possible, elect the most suitable one for the future decomposition.

First of all, regarding the computational demand, it appears clear that the ILP solution is unacceptable. Indeed, while the amount of computational time, requested for solving LRP1, QRP1 and LRP2 is in the order of dozens of seconds, our laptop was not either able to solve the ILP.

Furthermore, we discard the LRP2 solution in view of the unacceptable distortion in the scheduled profile shape. We recall that LRP1 has got an intrinsic capability to cluster the profiles, while in LRP2 this is not true. To better understand, this point, it is necessary to think about how the two strategies has been built: in the first one, the free variable ξ_i decides the contribution that each profile gives to the scheduled off-take from a set of admissible delayed profiles for the i-th pool; instead, in the latter, the decisional variable ζ_i represents the amount of water to provide each time slot, separately.

Remarkably, the QRP1 formulation displays the most satisfactory solution, in terms of quality of the service, for our problem: the minimization of the delivery delays and the preservation of the original shape of the profiles. Nevertheless, in the following, we discard it for decomposition purposes for the undermentioned reasons:

- in QRP1 the computational time is almost doubled with respect to the linear case.
- a non-convex optimization problem is not desirable for a future decomposition.

Moreover, the scheduled profiles' shape (the variance of the overall solution) can be considered good also in the linear case: the total variance in LRP1 is equal to 116.4, while, with QRP1, it is about 59.

Given all these considerations, in the following, we will deal only with the relaxed linear program LRP1.

Chapter 4

Decomposition Methods for a Centralized Optimization Problem

In this chapter, our scope is to devise two distributed solutions, obtained from the decomposition of the linear centralized formulation LRP1 discussed in Section 3.3.3. First of all, we will recast the centralized optimization problem in such a way that suitable problem decompositions are allowed. As discussed in Section 2.1.2, this layout derives from the cascaded model structure and from the presence of the coupling variables v_i, included in the centralized model. Secondly, we will discuss two different algorithms allowing for a distributed solution to the centralized optimization problem: Round Robin (RR) and Primal Decomposition (PD), as introduced in Section 2.3.2. Finally, we will display some simulation tests where both the solutions are implemented and we will draw some conclusions about the two methods.

4.1 Decomposable Structure of the Centralized Optimization Problem

In this section, we are going to decompose the centralized model, in order to obtain a distributed model of the system. For this purpose, a suitable change of variables will be used, concerning the predictor equations. This structure will be useful to build the distributed algorithms that will be devised in the following sections.

Firstly, we recall the linear optimization program RP1 (3.21) solving the centralized problem. For reasons that will be clear soon, in this section some matrices and vectors related to the centralized model (3.21) will be denoted using the superscript (*), i.e.:

$$\underset{\xi_i}{minimize} \quad \sum_{i=1}^{N} C_i \xi_i \tag{4.1a}$$

$$\text{subject to} \quad \hat{y}^{(k+1:k+n_y)*} = \Gamma^* x^{(k)} + \Omega^* r^{(k:k+n_y-1)*} + \\ + \Psi^* \tilde{d}^{(k:k+n_y-1)*} + \Psi^* \sum_{i=1}^{N} \Pi_i M_i \xi_i \tag{4.1b}$$

$$\underline{y}^* \leq \hat{y}^{(k+1:k+n_y)*} \leq \bar{y}^* \tag{4.1c}$$

$$\sum_{j=1}^{n_i} \xi_{i,j} = 1 \tag{4.1d}$$

$$0 \leq \xi_{i,j} \leq 1 \quad \text{for} \quad j = \{1, \ldots, n_i\}, i = \{1, \ldots, N\} \tag{4.1e}$$

where the objective function (4.1a) is discussed in Section 3.4.3. We remark that the objective function is a sum-separable cost function. The solution to this centralized problem is discussed in Section 3.5.3.

We now focus on the equality constraint (4.1b), which is linear with respect to the decision variables ξ_i, and the inequality constraints (4.1c). The constraint (4.1b) can be arranged, isolating the term depending on the decisional variables, i.e.:

$$\Psi^* \sum_{i=1}^{N} \Pi_i M_i \xi_i = \hat{y}^{(k+1:k+n_y)*} - \left(\Gamma^* x^{(k)} + \Omega^* r^{(k:k+n_y-1)*} + \Psi^* \tilde{d}^{(k:k+n_y-1)*}\right) \tag{4.2}$$

Concerning the term on the right side of the equation (4.2), we define the vector $b^{(k+1:k+n_y)*}$ as:

$$b^{(k+1:k+n_y)*} = \hat{y}^{(k+1:k+n_y)*} - \left(\Gamma^* x^{(k)} + \Omega^* r^{(k:k+n_y-1)*} + \Psi^* \tilde{d}^{(k:k+n_y-1)*}\right) \tag{4.3}$$

The vector $b^{(k+1:k+n_y)*}$ is a linear combination of the initial state $x^{(k)}$, the predicted water-level $\hat{y}^{(k+1:k+n_y)*}$, the references vector $r^{(k:k+n_y-1)*}$ and the already scheduled load $\tilde{d}^{(k:k+n_y-1)*}$. Focusing on the last three vectors, we recall that the

notation $s^{(a:b)*}$ stands for $\begin{bmatrix} s(a)^T & s(a+1)^T & \dots & s(b)^T \end{bmatrix}^T$ for a discrete-time signal $s(k) \in \mathbb{R}^m$ and $s(a) = \begin{bmatrix} s_1(a) & s_2(a) & \dots & s_m(a) \end{bmatrix}^T$. We want now to sort the elements of the vectors $\hat{y}^{(k+1:k+n_y)*}$, $r^{(k:k+n_y-1)*}$ and $\tilde{d}^{(k:k+n_y-1)*}$ in such a way that the first elements of these vectors are variables related to the pool 1 and, then, the other elements correspond to the variables associated to the other pools in ascending order. Let $\hat{y}^{(k+1:k+n_y)}$, $r^{(k:k+n_y-1)}$ and $\tilde{d}^{(k:k+n_y-1)}$ be the sorted vectors, i.e.:

$$\hat{y}^{(k+1:k+n_y)} = \begin{bmatrix} \hat{y}_1^{(k+1:k+n_y)} \\ \hat{y}_2^{(k+1:k+n_y)} \\ \vdots \\ \hat{y}_N^{(k+1:k+n_y)} \end{bmatrix}, \qquad r^{(k+1:k+n_y)} = \begin{bmatrix} r_1^{(k+1:k+n_y)} \\ r_2^{(k+1:k+n_y)} \\ \vdots \\ r_N^{(k+1:k+n_y)} \end{bmatrix},$$
$$\tilde{d}^{(k+1:k+n_y)} = \begin{bmatrix} \tilde{d}_1^{(k+1:k+n_y)} \\ \tilde{d}_2^{(k+1:k+n_y)} \\ \vdots \\ \tilde{d}_N^{(k+1:k+n_y)} \end{bmatrix} \tag{4.4}$$

We define a suitable permutation matrix $\mathcal{P} \in \mathbb{R}^{Nn_y \times Nn_y}$ such that:

$$\begin{aligned} \hat{y}^{(k:k+n_y-1)*} &= \mathcal{P}\hat{y}^{(k:k+n_y-1)} \\ r^{(k:k+n_y-1)*} &= \mathcal{P}r^{(k:k+n_y-1)} \\ \tilde{d}^{(k:k+n_y-1)*} &= \mathcal{P}\tilde{d}^{(k:k+n_y-1)} \end{aligned} \tag{4.5}$$

Note that the elements in $x^{(k)}$ are already sorted in a desirable way, due to the construction of the predictor, i.e. $x^{(k)} = \begin{bmatrix} x_1^{(k)T} & x_2^{(k)T} & \cdots & x_N^{(k)T} \end{bmatrix}$

In the same way, we can sort the vector $b^{(k+1:k+n_y)*}$, i.e.:

$$b^{(k:k+n_y-1)*} = \mathcal{P}b^{(k:k+n_y-1)} \tag{4.6}$$

The equation (4.3), thus, becomes:

$$\mathcal{P}b^{(k+1:k+n_y)} = \mathcal{P}\hat{y}^{(k+1:k+n_y)} - \left(\Gamma^* x^{(k)} + \Omega^* \mathcal{P} r^{(k:k+n_y-1)} + \Psi^* \mathcal{P}\tilde{d}^{(k:k+n_y-1)}\right) \tag{4.7}$$

Since $\mathcal{P}$ is a squared and non-singular matrix, we can multiply all the elements

in (4.7) by $\mathcal{P}^{-1}$, i.e.:

$$\mathcal{P}^{-1}\mathcal{P}b^{(k+1:k+n_y)} = \mathcal{P}^{-1}\mathcal{P}\hat{y}^{(k+1:k+n_y)} - \big(\mathcal{P}^{-1}\Gamma^* x^{(k)} + \mathcal{P}^{-1}\Omega^*\mathcal{P}r^{(k:k+n_y-1)} + \\ + \mathcal{P}^{-1}\Psi^*\mathcal{P}\tilde{d}^{(k:k+n_y-1)}\big) \tag{4.8}$$

Defining the matrices Γ, Ω and Ψ as the "sorted" version of Γ^*, Ω^* and Ψ^*, i.e.:

$$\Gamma = \mathcal{P}^{-1}\Gamma^*\mathcal{P}, \quad \Omega = \mathcal{P}^{-1}\Omega^*\mathcal{P}, \quad \Psi = \mathcal{P}^{-1}\Psi^*\mathcal{P}, \tag{4.9}$$

and simplifying where possible, we obtain:

$$b^{(k+1:k+n_y)} = \hat{y}^{(k+1:k+n_y)} - \big(\Gamma x^{(k)} + \Omega r^{(k:k+n_y-1)} + \Psi\tilde{d}^{(k:k+n_y-1)}\big) \tag{4.10}$$

By construction, the vectors appearing in (4.10) can be split into sub-vectors, each one related to a single pool, according to (4.4): it follows that, in view of the cascade structure of the channel and, according to its dynamical model (see Figure 2.3), the matrices Γ, Ω and Ψ have an upper-triangular structure of the type:

$$\Gamma = \begin{bmatrix} \Gamma_{12} & \Gamma_{11} & \cdots & \Gamma_{1N} \\ 0 & \Gamma_{22} & & \vdots \\ \vdots & & \ddots & \\ 0 & \cdots & 0 & \Gamma_{NN} \end{bmatrix} \quad \Omega = \begin{bmatrix} \Omega_{12} & \Omega_{11} & \cdots & \Omega_{1N} \\ 0 & \Omega_{22} & & \vdots \\ \vdots & & \ddots & \\ 0 & \cdots & 0 & \Omega_{NN} \end{bmatrix} \\ \Psi = \begin{bmatrix} \Psi_{12} & \Psi_{11} & \cdots & \Psi_{1N} \\ 0 & \Psi_{22} & & \vdots \\ \vdots & & \ddots & \\ 0 & \cdots & 0 & \Psi_{NN} \end{bmatrix} \tag{4.11}$$

Renaming, for simplicity of notation, $b_i = b_i^{(k+1:k+n_y)}$, $\hat{y}_i = \hat{y}_i^{(k+1:k+n_y)}$, $r_i = r_i^{(k+1:k+n_y)}$, $\tilde{d}_i = \tilde{d}_i^{(k+1:k+n_y)}$ and $x_i = x_i^{(k)}$, we can express (4.10) in a decomposed way, i.e., through N equations of the type:

$$b_i = \hat{y}_i - \big(\sum_{h=i}^{N}\Gamma_{ih}x_h + \sum_{h=i}^{N}\Omega_{ih}r_h + \sum_{h=i}^{N}\Psi_{ih}\tilde{d}_h\big) \quad \text{for} \quad i = 1,\ldots,N \tag{4.12}$$

Focusing on the left-hand side of the equation (4.2), note that it can be

decomposed as the sum of N terms:

$$\begin{aligned}\Psi^* \sum_{i=1}^{N} \Pi_i M_i \xi_i = \Psi^* \big(\Pi_1 M_1 \xi_1 + \Pi_2 M_2 \xi_2 + \cdots + \Pi_N M_N \xi_N\big) = \\ = \Psi^* \Pi_1 M_1 \xi_1 + \Psi^* \Pi_2 M_2 \xi_2 + \cdots + \Psi^* \Pi_N M_N \xi_N\end{aligned} \tag{4.13}$$

Let $A_i^* = \Psi^* \Pi_i M_i$ for all i. We recall that Π_i is the matrix stacking the off-takes to be scheduled for the centralized model. Similarly to the previous equations, we comptute matrix $A_i \in \mathbb{R}^{N n_y \times n_i}$ as:

$$A_i = \mathcal{P}^{-1} \Psi^* \Pi_i M_i \quad \text{for} \quad i = 1, \ldots, N \tag{4.14}$$

It results that each matrix A_i, can be decomposed into N sub-matrices. Specifically, in view of the cascade structure of the channel model, we obtain that:

$$A_1 = \begin{bmatrix} A_{11} \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad A_2 = \begin{bmatrix} A_{12} \\ A_{22} \\ \vdots \\ 0 \end{bmatrix} \quad \cdots \quad A_N = \begin{bmatrix} A_{1N} \\ A_{2N} \\ \vdots \\ A_{NN} \end{bmatrix} \tag{4.15}$$

Recalling that $b = \begin{bmatrix} b_1^T & b_2^T & \ldots & b_N^T \end{bmatrix}^T$, the equation (4.2) can be, thus, divided into N sub-equations of the type:

$$A_{11}\xi_1 + A_{12}\xi_2 + \cdots + A_{1N}\xi_N = b_1 \tag{4.16a}$$

$$A_{22}\xi_2 + \cdots + A_{2N}\xi_N = b_2 \tag{4.16b}$$

$$\vdots$$

$$A_{NN}\xi_N = b_N \tag{4.16c}$$

Importantly, each equation is related to a single pool, i.e. (4.16a) is related to pool 1, (4.16b) to pool 2, etc.

Now, we will focus on the water-level upper and lower bounds expressed in inequality (4.1c). The latter inequality can be split into two inequalities in which

the terms are sorted according to the ordering structure previously used:

$$\begin{cases} \hat{y}^* \leq \bar{y}^* \\ \hat{y}^* \geq \underline{y}^* \end{cases} \quad \Rightarrow \quad \begin{cases} \hat{y} \leq \bar{y} \\ \hat{y} \geq \underline{y} \end{cases} \tag{4.17}$$

Therefore, concerning the upper bound, we have that, for each pool i:

$$\hat{y}_i \leq \bar{y}_i \tag{4.18}$$

Now, accordingly to the decomposition which has been carried out, and recalling that $b_i = \hat{y}_i - \left(\sum_{h=i}^{N} \Gamma_{ih} x_h + \sum_{h=i}^{N} \Omega_{ih} r_h + \sum_{h=i}^{N} \Psi_{ih} \tilde{d}_h\right)$, we can rewrite (4.18) as:

$$b_i \leq \bar{b}_i \tag{4.19}$$

where

$$\bar{b}_i = \bar{y}_i - \left(\sum_{h=i}^{N} \Gamma_{ih} x_h + \sum_{h=i}^{N} \Omega_{ih} r_h + \sum_{h=i}^{N} \Psi_{ih} \tilde{d}_h\right) \tag{4.20}$$

From (4.20) and (4.19), we rewrite (4.18), as:

$$\sum_{h=i}^{N} A_{ih} \xi_h \leq \bar{b}_i \quad \text{for} \quad i = 1, \ldots, N \tag{4.21}$$

We apply the same reasoning to the lower bound inequality, and we obtain:

$$\sum_{h=i}^{N} A_{ih} \xi_h \geq \underline{b}_i \quad \text{for} \quad i = 1, \ldots, N \tag{4.22}$$

where $\underline{b}_i = \underline{y}_i - \left(\sum_{h=i}^{N} \Gamma_{ih} x_h + \sum_{h=i}^{N} \Omega_{ih} r_h + \sum_{h=i}^{N} \Psi_{ih} \tilde{d}_h\right)$

For clarity, remark that the terms $\bar{b}_i$ and $\underline{b}_i$ are linearly dependent upon the already scheduled off-takes $\tilde{d}_h$ for all $i \leq h \leq N$, i.e.:

$$\begin{aligned} \bar{b}_i &= \bar{b}_i(\tilde{d}_h) \\ \underline{b}_i &= \underline{b}_i(\tilde{d}_h) \end{aligned} \tag{4.23}$$

and are given. Therefore, in (4.21) and (4.22), the decision variables are just ξ_i, for $i = 1, \ldots, N$.

Finally, we recall that the sum-separable objective function is already decom-

posed into a sum of "local" terms $C_i^T \xi_i$, each depending on the decision variable ξ_i, related to the i-th pool solely.

From the above reasoning, we obtain that the centralized problem can be recast as the following decomposable linear program:

$$
\begin{aligned}
\underset{\xi_1,\ldots,\xi_N}{minimize} \quad & \sum_{i=1}^{N} C_i^T \xi_i \\
\text{subject to} \quad & \sum_{h=i}^{N} A_{ih}\xi_h \leq \bar{y}_i - \Big(\sum_{h=i}^{N} \Gamma_{ih} x_h + \sum_{h=i}^{N} \Omega_{ih} r_h + \sum_{h=i}^{N} \Psi_{ih} \tilde{d}_h\Big) \\
& \sum_{h=i}^{N} A_{ih}\xi_h \geq \underline{y}_i - \Big(\sum_{h=i}^{N} \Gamma_{ih} x_h + \sum_{h=i}^{N} \Omega_{ih} r_h + \sum_{h=i}^{N} \Psi_{ih} \tilde{d}_h\Big) \qquad (4.24) \\
& \sum_{j=1}^{n_i} \xi_{i,j} = 1 \\
& 0 \leq \xi_{i,j} \leq 1 \quad \text{for} \quad j = \{1,\ldots,n_i\} \\
& \qquad\qquad\qquad\qquad \text{for all} \quad i = 1,\ldots,N
\end{aligned}
$$

Note that the sub-matrix A_{ih} represents the effect of the h-th off-take on the i-th pool. It is now clear what we introduced in Section 2.1.2, about the coupling variables v_i. Even though v_i are not explicitly shown, because they are included in the centralized model, they play the role of passing the information of the h-th off-take to the upstream pools i for $i = 1,\ldots,h$. Remark also that, as described in Section 2.1.2, the off-take of the i-th pool (for $i = 1,\ldots,N$) does not affect the downstream pools, i.e. pools h, for all $h > i$.

Such a problem structure is suitable for applying proper problem decomposition techniques, as discussed in details in the next sections. In view of this techniques, it will be possible to formulate the optimization problem as a number of small-scale local problems, each accounting for "local" constraints (4.21) and (4.22). Two different solutions will be discussed in the following, i.e.: the primal decomposition algorithm [5] and the Round Robin algorithm [9]. The effectiveness of these methods will be then tested in simulation.

4.2 Application of Round Robin Algorithm RR to LRP1

Round Robin is largely used for scheduling purposes in the process industry. We want to apply a similar idea to our case. In [9], a RR algorithm, adapted to integer linear program, is shown.

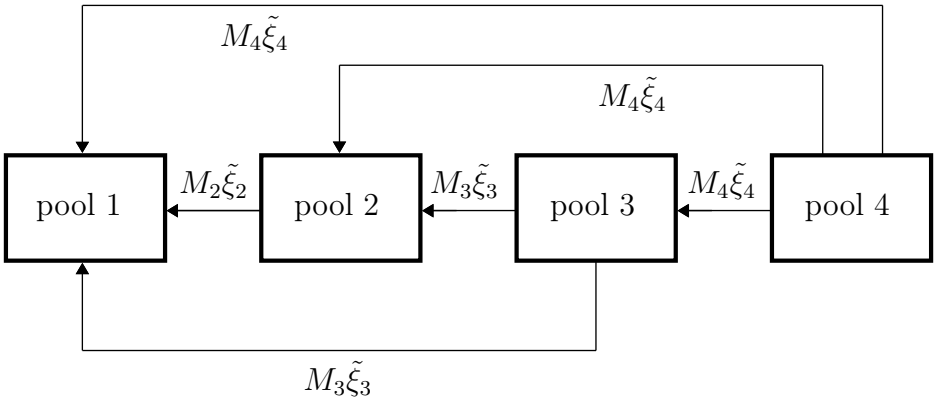


Figure 4.1: Informations passing among 4 pools in terms of scheduled off-takes

In Figure 4.1, $M_i\tilde{\xi}_i$ are the scheduled off-takes of the i-th pool. As discussed, the scheduled off-take in pool i-th does not affect all the downstream pools, but only the upstream ones (included the i-th pool). Due to the particular cascaded structure of our problem, it is natural to schedule the off-takes sequentially from pool N to pool 1.

In [9], a decomposed structure (different from the one we have discussed in the previous section) of the $\{0, 1\}$ integer linear program discussed in Section 3.3.2 is described and a solution is discussed. We recall that LRP1 and the integer linear program are equally expressed save for the difference in the constraints (3.19e) and (3.21e). The distributed structure we have obtained in Section 4.1 is, thus, suitable for the ILP as well. Nevertheless, we will discuss only the application of RR to the relaxed linear program LRP1.

4.2.1 Round Robin Algorithm

The algorithm considers the scheduling of off-takes one by one. Starting from the last pool N, the N-th off-take $M_N\xi_N$ is scheduled, paying attention that the constraints of all upstream pools are fulfilled and assuming that $\xi_i = 0$ for all

$i = 1, \ldots, N-1$. Once we obtain the scheduled off-take $\tilde{d_N} = M_N \tilde{\xi}_N$, it schedules the $N-1$-th off-take where $\tilde{d_N}$ is assumed given. We iterate this algorithm up to the 1-st pool, taking always into account of the downstream already scheduled off-takes.

Formally, the algorithm is the following:

Round Robin Algorithm

step 1 $k \leftarrow N$;

step 2 calculate $\bar{b}_i(\tilde{d}_h)$ and $\underline{b}_i(\tilde{d}_h)$ for all $i = 1, \ldots, k$, as shown in section 4.1 with the already scheduled off-takes:

$$\begin{aligned} \bar{b}_i &= \bar{y}_i - \big(\sum_{h=i}^{N} \Gamma_{ih} x_h + \sum_{h=i}^{N} \Omega_{ih} r_h + \sum_{h=i}^{N} \Psi_{ih} \tilde{d}_h\big) \\ \underline{b}_i &= \underline{y}_i - \big(\sum_{h=i}^{N} \Gamma_{ih} x_h + \sum_{h=i}^{N} \Omega_{ih} r_h + \sum_{h=i}^{N} \Psi_{ih} \tilde{d}_h\big) \end{aligned} \tag{4.25}$$

step 3 find optimal ξ_k satisfying

$$\begin{aligned} \underset{\xi_k}{minimize} \quad & C_k^T \xi_k \\ \text{subject to} \quad & \text{for all} \;\; i = 1, \ldots, k \\ & A_{ik} \xi_k \leq \bar{b}_i \\ & A_{ik} \xi_k \geq \underline{b}_i \\ & \sum_{j=1}^{n_i} \xi_{i,j} = 1 \\ & 0 \leq \xi_{i,j} \leq 1 \;\; \text{for} \;\; j = \{1, \ldots, n_i\} \end{aligned} \tag{4.26}$$

step 4 $\tilde{d}_k \leftarrow \tilde{d}_k + M_k \xi_k$; $k \leftarrow k-1$; if $k = 0$, then end; else go to step 2;

Applying RR on our decomposed two-pools problem, the algorithm can be divided in two steps. At the first step, the off-takes of the pool 2, i.e. ξ_2, are scheduled, taking into account of the constraints of both the sub-problems. At the second step, the algorithm schedules the off-takes related to the pool 1, including $\tilde{d}_2 = M_2 \xi_2$ in the constraints.

It is important to mention, at this point, that the previously-described method does not guarantee any system-wide optimality property. However, it is particularly suitable for a computationally-efficient application to cascaded systems.

4.2.2 Simulation Tests on LRP1, Decomposed using RR

In Figure 4.2, the results of a simulation test on LRP1, decomposed using the Round Robin algorithm, are shown.

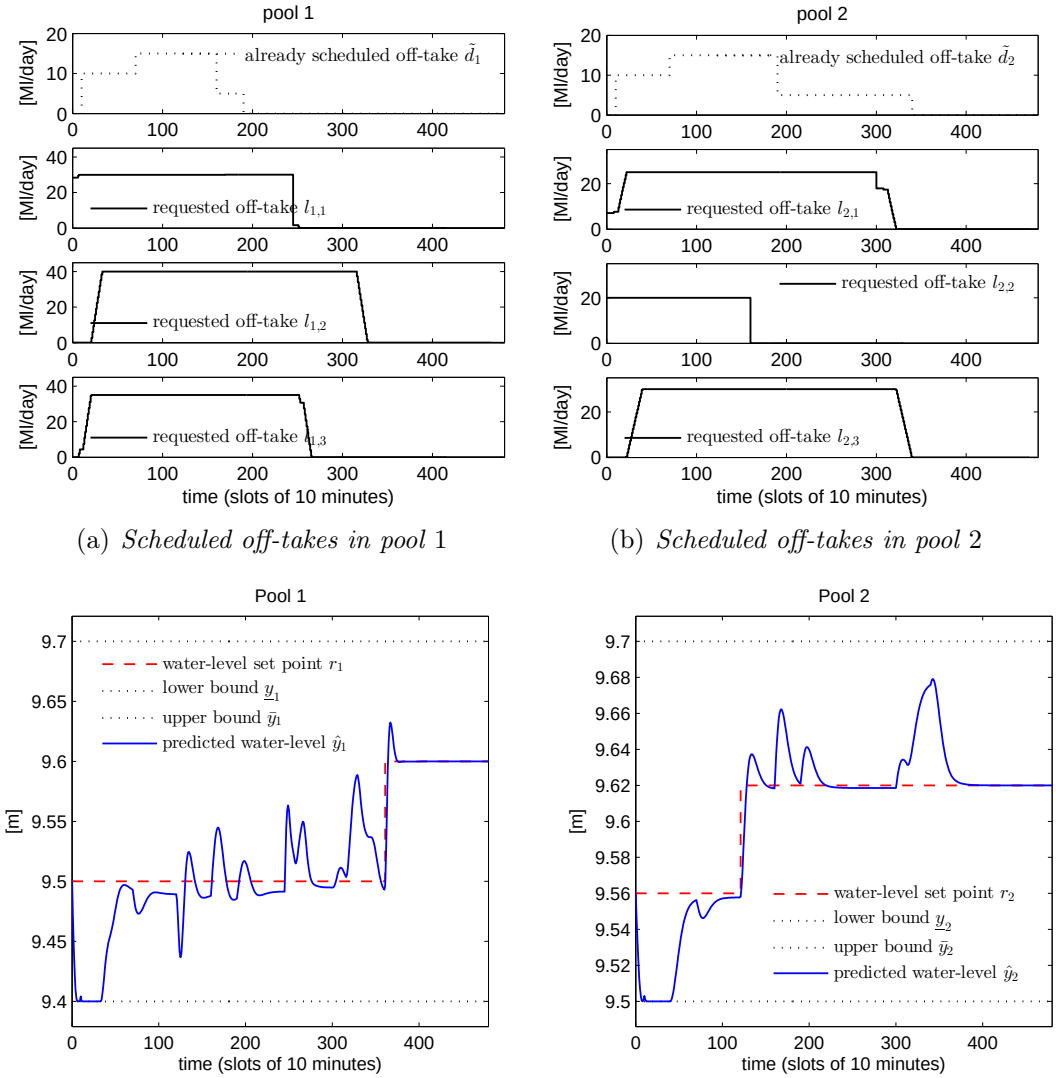
In terms of the computational demand, RR algorithm is very light. Indeed, our laptop takes 22 seconds to solve the problem: almost halved with respect to the centralized linear case. Note that the scheduling result in the case of two pools is not so different from the centralized one (Section 3.5.3): in terms of delays in water supply, indeed, the value of the overall delivery delay, with RR solution, is:

$$\sum_{i=1}^{N} \bar{t}_i = 50 \ \text{ seconds} \tag{4.27}$$

Nevertheless, we achieved such good results because, we applied RR algorithm to the 2-pool case. In fact, in channel models with several pools, the obtained results can be unsatisfactory. Note that RR algorithm gives priority to off-take requests in the downstream pools. This implies that, as the number of pools grows, the solution of the RR algorithm may differ from the centralized one.

We remark the fact that for a problem involving N pools, the algorithm solves the sub-problems in N steps. Therefore, in case of a centralized implementation, the computational demand scales linearly with respect to the number of pools N.

However, in case of a distributed implementation (i.e., the k-th problem (4.26) is solved by a local computing station located with the k-th pool), it is worth noticing that the computational demand depends just on the complexity of the k-th optimization problem, which is limited and does not scale with the number of pools. In the latter case, the exchange of useful information among the local control stations is required, i.e., consisting of the optimal local solutions $\tilde{\xi}_k$, which must be broadcast as the solution is attained.



(a) *Scheduled off-takes in pool* 1

(b) *Scheduled off-takes in pool* 2

(c) *Predicted water-level in pool 1*

(d) *Predicted water-level in pool 2*

Figure 4.2: Scheduled off-takes applying Round Robin on LRP1

4.3 Primal Decomposition PD

In this section, our purpose is to show a more common distributed algorithm for the solution of the decomposed problem. The primal decomposition is largely discussed in literature [12] and employed in many applications [13], [7]

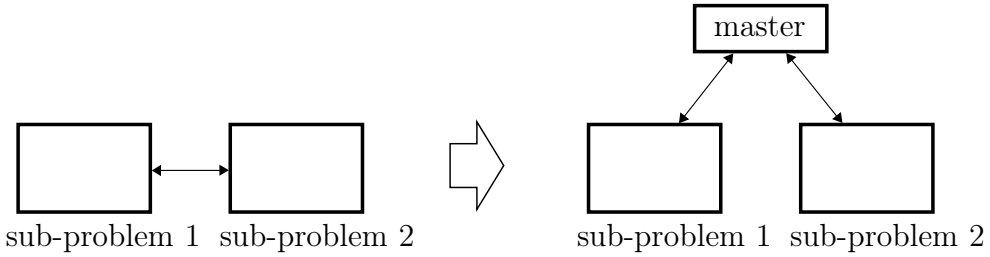


Figure 4.3: Primal Decomposition

4.3.1 Primal Decomposition Algorithm

Suppose our problem (for the sake of simplicity, we will consider only two pools, which allows the decomposition of the problem into two sub-problems) has the generic form:

$$\begin{aligned} \underset{x_i}{minimize} \quad & f_1(x_1) + f_2(x_2) \\ \text{subject to} \quad & x_1 \in \mathcal{X}_1, \quad x_2 \in \mathcal{X}_2 \\ & h_1(x_1) + h_2(x_2) \leq r \end{aligned} \tag{4.28}$$

where x_1 and x_2 are decision variables and $\mathcal{X}_1$ and $\mathcal{X}_2$ are the local set of constraints related to variables x_1 and x_2, respectively, and described by linear equalities and/or inequalities. The functions $h_i : \mathbb{R}^n \rightarrow \mathbb{R}^p$ are assumed linear with respect to their arguments (and therefore convex). Thereby, the sub-problems are said to be coupled via the p constraints, that involve both x_1 and x_2 at the same time. Such constraints are called *complicating constraints*, since without them the sub-problems could be easily solved separately. In [5], a suitable way to separate the complicating constraints is shown.

The problem can be seen as a resource allocation problem. The resource to be shared is limited and it is represented by the vector r. There is a conflict

between the sub-problems. Indeed, if we solve the sub-problems separately, i.e.:

$$\begin{array}{lll} \underset{x_1}{minimize} & f_1(x_1) & \underset{x_2}{minimize} \quad f_2(x_2) \\ \text{subject to} & x_1 \in \mathcal{X}_1 & \text{subject to} \quad x_2 \in \mathcal{X}_2 \\ & h_1(x_1) \leq r & \qquad\quad h_2(x_2) \leq r \end{array} \tag{4.29}$$

each sub-problem will use the resource without taking into account of the resource allocated by the other one. This situation will lead to a non-feasibility of the global problem even if both the local ones are feasible. Moreover, even if we halve the elements of the vector r among the sub-systems, the convergence to global optimal solution is not guaranteed.

In order to avoid this, we introduce a new variable $t \in \mathbb{R}^p$, such that $1/2r + t$ represents the amount of the resource allocated to the first sub-problem. Consequently, $1/2r - t$ will be the part allocated to the second sub-problem. The problem will be then split into two sub-problems, defined next:

$$\begin{array}{ll} \underset{x_1}{minimize} & f_1(x_1) \\ \text{subject to} & x_1 \in \mathcal{X}_1 \\ & h_1(x_1) \leq \dfrac{1}{2}r + t \end{array} \tag{4.30}$$

$$\begin{array}{ll} \underset{x_2}{minimize} & f_2(x_2) \\ \text{subject to} & x_2 \in \mathcal{X}_2 \\ & h_2(x_2) \leq \dfrac{1}{2}r - t \end{array} \tag{4.31}$$

These problems, (4.30) and (4.31), can be solved independently and simultaneously, when t is fixed. Let $\phi_1(t)$ and $\phi_1(t)$ denote the solutions, i.e. the optimal values, of the sub-problems (4.30) and (4.31), respectively. The original problem (4.28) is equivalent to the master problem of minimizing $\phi(t) = \phi_1(t) + \phi_2(t)$ with respect to the "allocation vector" t. As described is [5], we can find a sub-gradient for the optimal value of each sub-problem from an optimal dual variable associated with the coupling constraint. For this purpose, define with $p^\star(z)$ the

optimal value of the convex optimization problem:

$$\begin{array}{ll} \underset{x}{minimize} & f(x) \\ \text{subject to} & x \in \mathcal{X} \\ & h(x) \leq z \end{array} \tag{4.32}$$

and suppose $z \in \mathbf{dom}\ p$. Let $\lambda(z)^\star$ be an optimal dual variable associated with the constraint $h(z) \leq z$[1]. Then, it is possible to demonstrate (see [4] and [5]) that $-\lambda(z)^\star$ is a sub-gradient of p at z.

Thus, coming back to our problem, to find a sub-gradient of ϕ, first we solve the two sub-problems, where the vector t is fixed, to find the local optimal solutions $x_1^\star$ and $x_2^\star$, as well as the optimal dual variables $\lambda_1^\star$ and $\lambda_2^\star$, associated to the constraints $h_1(x_1) \leq 1/2r + t$ and $h_2(x_2) \leq 1/2r - t$, respectively. Hence, a sub-gradient of ϕ at t will be $\lambda_2^\star - \lambda_1^\star \in \partial\phi(t)$.

The master, thus, solves a simple problem in terms of updating the resource allocation t aiming to minimize $\phi(t)$ with respect to t, using a standard sub-gradient method.

The algorithm is, therefore, the following:

Primal Decomposition Algorithm

step 1 $k \leftarrow 0$; $t^{(k)} \leftarrow \mathbf{0}$;

step 2 solve the sub-problems (possibly in parallel):

$$\begin{array}{ll} \underset{x_1}{minimize} & f_1(x_1) \\ \text{subject to} & x_1 \in \mathcal{X}_1 \\ & h_1(x_1) \leq \dfrac{1}{2}r + t^{(k)} \end{array} \tag{4.34}$$

[1]Problem (4.32) can be recast as the following dual one:

$$\underset{\lambda \geq 0}{maximize} \quad g(\lambda) \tag{4.33}$$

where $g(\lambda) = \inf_{x \in \mathcal{X}} \big(f(x) + \lambda^T(h(x) - z)\big)$. The solution of (4.33) is the pair $(x^\star(z), \lambda^\star(z))$ [4]

and

$$\begin{aligned} \underset{x_2}{minimize} \quad & f_2(x_2) \\ \text{subject to} \quad & x_2 \in \mathcal{X}_2 \\ & h_2(x_2) \leq \frac{1}{2}r - t^{(k)} \end{aligned} \tag{4.35}$$

to find an optimal $x_1^{(k)} = x_1^\star(t^{(k)})$ and $x_2^{(k)} = x_2^\star(t^{(k)})$, $\lambda_1^{(k)} = \lambda_1^\star(t^{(k)})$ and $\lambda_2^{(k)} = \lambda_2^\star(t^{(k)})$, respectively;

step 3 calculate the optimal value at $t^{(k)}$:

$$\phi(t^{(k)}) = \phi_1(t^{(k)}) + \phi_2(t^{(k)}) = f_1(x_1^{(k)}; t^{(k)}) + f_2(x_2^{(k)}; t^{(k)}) = f(x^{(k)}; t^{(k)}) \tag{4.36}$$

step 4 if $f(x^{(k)}; t^{(k)})$ fulfills a certain stopping criterion, then end; else update the resource allocation:

$$t^{(k+1)} = t^{(k)} - \alpha^{(k)}(\lambda_2^{(k)} - \lambda_1^{(k)}) \tag{4.37}$$

$k \leftarrow k + 1$ and go to *step* 2

Here, $\alpha^{(k)}$ is an appropriate step-size. The stopping criterion and the step-size rules are discussed in the following.

4.3.2 Step-size Rules

In [16], a step-size rule, that can be used when the optimal value $f^\star$ is known, is suggested. The step size is:

$$\alpha^{(k)} = \frac{f(x^{(k)}) - f^\star}{\|\partial\phi(t^{(k)})\|_2} \tag{4.38}$$

This formulation would be optimal, but not suitable in our case. Indeed, we assume to be $f^\star$ not known. On this hand, in [3], several rules for suitably setting the step-size $\alpha^{(k)}$ are discussed. In the following we will list some solutions:

- *Constant step-size*: $\alpha^{(k)} = \alpha$ is a positive constant. Note that, in this case, the step-size is independent of the iteration k.

- *Constant step length*: $\alpha^{(k)} = \gamma / \left\|\partial\phi(t^{(k)})\right\|_2$, where $\gamma > 0$. This means that $\left\|x^{(k+1)} - x^{(k)}\right\|_2 = \gamma$.

- *Square-summable but not summable*: the step-size satisfy

$$\alpha^{(k)} \geq 0, \qquad \sum_{k=1}^{\infty} \alpha^{(k)2} < \infty, \qquad \sum_{k=1}^{\infty} \alpha^{(k)} = \infty \tag{4.39}$$

  for example, $\alpha^{(k)} = a/(b+k)$, where $a > 0$ and $b \geq 0$.

- *Non-summable vanishing*: the step-size satisfy

$$\alpha^{(k)} \geq 0, \qquad \lim_{k\to\infty} \alpha^{(k)} = 0, \qquad \sum_{k=1}^{\infty} \alpha^{(k)} = \infty \tag{4.40}$$

  In this case, the step-size that satisfies this conditions is called diminishing step-size. A typical example is $\alpha^{(k)} = a/\sqrt{k}$, where $a > 0$.

- *Non-summable diminishing step length*: the step sizes are chosen as $\alpha^{(k)} = \gamma^{(k)} / \left\|\partial\phi(t^{(k)})\right\|_2$, where

$$\gamma^{(k)} \geq 0, \qquad \lim_{k\to\infty} \gamma^{(k)} = 0, \qquad \sum_{k=1}^{\infty} \gamma^{(k)} = \infty. \tag{4.41}$$

4.3.3 Stopping Criterion

In [3] and [11], two stopping criterion are discussed: the former is based on finding a lower bound of the global optimum of the centralized problem and stop the algorithm as soon as the difference between the objective function and the lower bound is smaller than a given threshold; the latter is based on the derivative of the curve of the objective function. Nevertheless, in most practical cases, the stopping criterion is just based on a limit for the number of iterations or the number of steps without an improvement [2], [14].

4.3.4 Application of PD to LRP1

The distributed structure of our problem is particularly prone to be decomposed using the primal decomposition method. Let the vectors $\bar{b}_i$ and $\underline{b}_i$ represent the

resource that must be shared among the off-takes. We recall that b_i is a linear combination of the water-level upper and lower bounds. Thus, the resource, in this case, is the amount of water that is available for the different pools. Focusing on our case of study (the last two pools of the EGM channel), we can adapt the generic problem (4.24), to the 2-pool case:

$$\begin{aligned}
\underset{\xi_1,\xi_2}{minimize} \quad & C_1^T\xi_1 + C_2^T\xi_2 && \text{(4.42a)}\\
\text{subject to} \quad & A_{11}\xi_1 + A_{12}\xi_2 \leq \bar{b}_1 && \text{(4.42b)}\\
& A_{22}\xi_2 \leq \bar{b}_2 && \text{(4.42c)}\\
& A_{11}\xi_1 + A_{12}\xi_2 \geq \underline{b}_1 && \text{(4.42d)}\\
& A_{22}\xi_2 \geq \underline{b}_2 && \text{(4.42e)}\\
& x_1 \in \mathcal{X}_1, \quad x_2 \in \mathcal{X}_2 && \text{(4.42f)}
\end{aligned}$$

In (4.42), the inequalities (4.42b) and (4.42d) are the so-called complicating constraints. On the other hand, (4.42c), (4.42e) and (4.42f) are the local constraints. The resource to be shared is represented, only, by the vectors $\bar{b}_1$ and $\underline{b}_1$. Indeed, as largely discussed, the first off-take does not affect the pool 2.

As discussed in Section 4.3.1, we can apply the primal decomposition on the problem (4.42), by splitting it into two sub-problems. Defining the matrix $\tilde{A}_{ih} = \begin{bmatrix} A_{ih} \\ -A_{ih} \end{bmatrix}$ and the vector $\tilde{b}_i = \begin{bmatrix} \bar{b}_i \\ -\underline{b}_i \end{bmatrix}$, the complicating constraints (4.42b) and (4.42d), thus, become:

$$\tilde{A}_{11}\xi_1 + \tilde{A}_{12}\xi_2 \leq \tilde{b}_1 \tag{4.43}$$

The sub-problems 1 and 2, related to the 1-st and the 2-nd pools, are:

$$\begin{aligned}
\underset{\xi_1}{minimize} \quad & C_1^T\xi_1 \\
\text{subject to} \quad & x_1 \in \mathcal{X}_1 \\
& \tilde{A}_{11}\xi_1 \leq \frac{1}{2}\tilde{b}_1 + t
\end{aligned}
\qquad\qquad
\begin{aligned}
\underset{\xi_2}{minimize} \quad & C_2^T\xi_2 \\
\text{subject to} \quad & x_2 \in \mathcal{X}_2 \\
& \tilde{A}_{12}\xi_2 \leq \frac{1}{2}\tilde{b}_1 - t \\
& \tilde{A}_{22}\xi_2 \leq \tilde{b}_2
\end{aligned}
\tag{4.44}$$

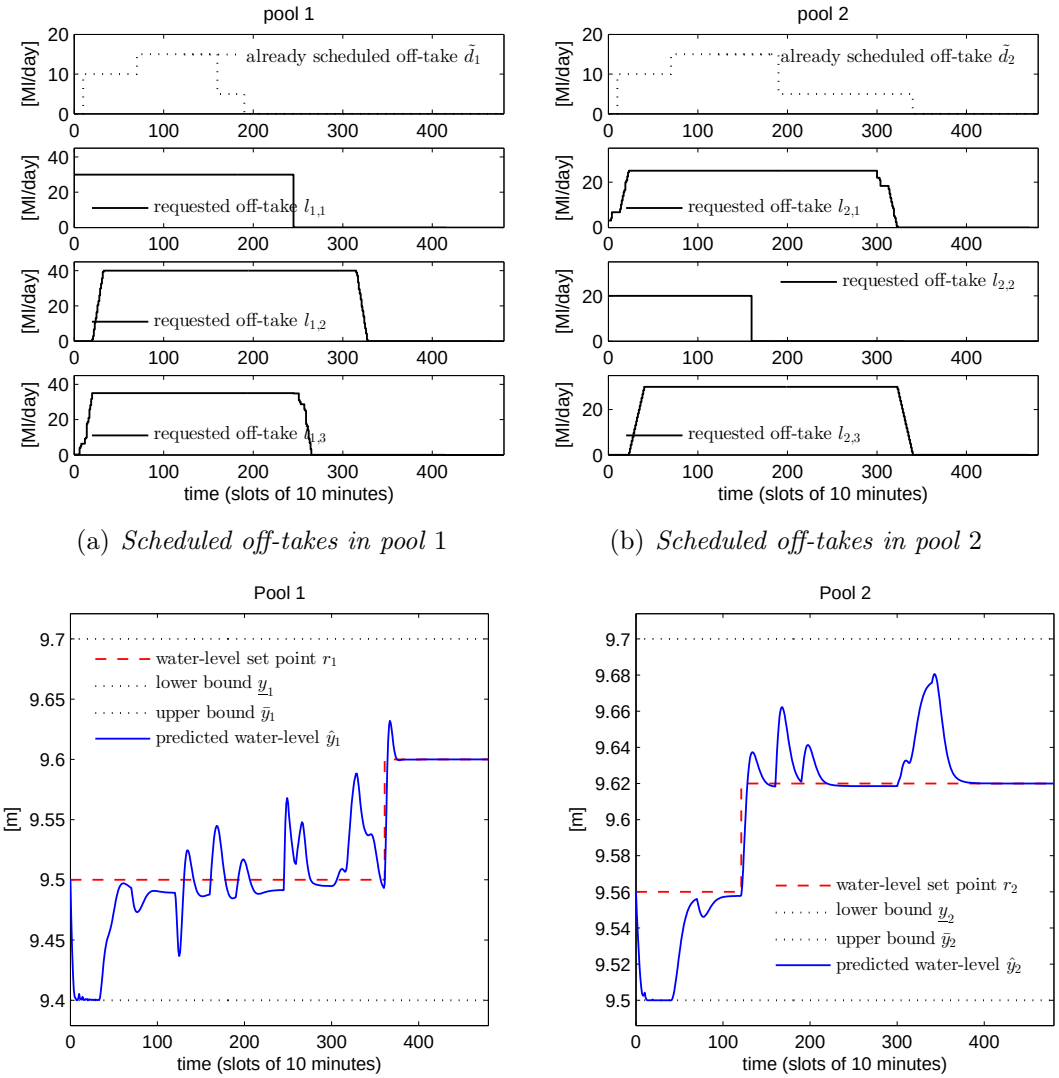
Initializing $t^{(k)} = \mathbf{0}^{2n_y}$, we can solve the two sub-problems in parallel in order

to obtain the optimal dual variable $\lambda_1^{(k)}$ and $\lambda_2^{(k)}$. Then the master updates t, according to the algorithm shown in Section 4.3.1.

Regarding the step-size, we choose an $\alpha^{(k)}$ diminishing with respect to the number of iteration. In particular, we use $\alpha^{(k)} = a/\sqrt{k}$. As discussed in Section 4.3.3, we define as the number of iteration $K = 300$.

4.3.5 Simulation Tests on LRP1, Decomposed using PD

Testing the PD algorithm on LRP1, we obtain the results shown in Figure 4.4.



(a) *Scheduled off-takes in pool* 1

(b) *Scheduled off-takes in pool* 2

(c) *Predicted water-level in pool 1*

(d) *Predicted water-level in pool 2*

Figure 4.4: Scheduled off-takes applying primal decomposition on LRP1

Comparing the results in Figure 4.4 with the simulation test on the centralized LRP1 (see Figure 3.7), we notice that the solutions are almost equivalent. Indeed, chosen an appropriate step-size, the primal decomposition methods achieves the global optimum, represented by the solution of the centralized LRP1. Other results, with different diminishing step-size choices, were obtain. Anyway, concerning the convergence speed, the best choice is the adopted one. In principle,

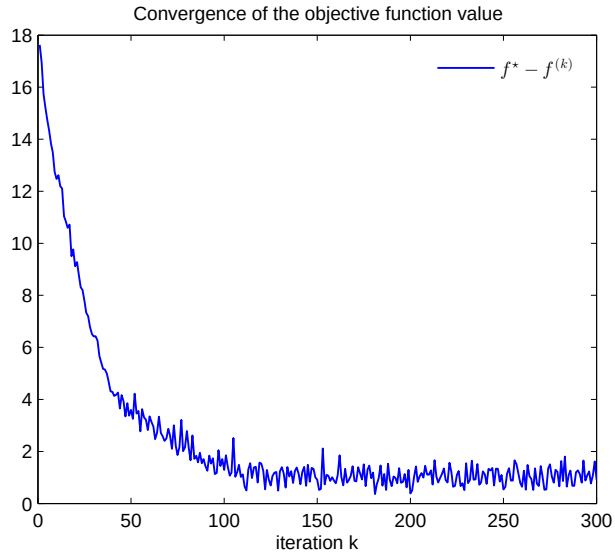


Figure 4.5: Convergence of the objective function to the global optimum

the primal decomposition method described in this section allows to decompose the problem in small-scale sub-problems, with non-scalable computational demand. Indeed, the overall computational time required for obtaining the solution can be estimated as more than two hours.

Secondly, as it can be noticed from Figure 4.5, although we can conclude that the objective function converges to global optimum, we also notice that the optimum is not properly reached. This problem is related to numerical problems in the solver, specifically, in the calculation of the dual variables.

4.4 Round Robin vs Primal Decomposition: Advantages and Drawbacks

As discussed in Section 4.2.2 and 4.3.5, both the algorithms have advantages and drawbacks. In this section, we are going to compare the results of PD and RR applied on LRP1.

Concerning the computational cost, the gap is very broad. In RR, each sub-problem is solved only once, starting from the N-th to the 1-st one. On the contrary, PD needs K iterations to find a suitable solution, where K depends on the adopted stopping criterion. In our case, for the sake of simplicity, K is fixed, but in general it is not a priori known. Anyway, the computational time demand in both the solutions proportionally depends on the number of sub-problems and, in general, on the number of off-takes to be scheduled. Nevertheless, we recall that, in the primal decomposition algorithm, the sub-problems can be solved in parallel, while the master solves a very simple updating problem. If we consider our case of study ($N = 2$), the computational time demand for solving the global problem should be halved. Also, it should be noted that since the primal algorithm is largely discussed in literature, more efficient and dedicated optimization tools can be used to further reduce the computational cost.

If we consider the optimality of the solutions, we can notice that Round Robin does not reach the global optimum (represented by the solution of the centralized RP1), because the downstream pools do not take into account of the decisions of the upstream ones. In primal decomposition, instead, each pool takes into account of the decisions of all the other pools through the resource allocation variable. This leads the overall system to achieve the global optimum. The gap, in terms of achievement of the global optimum, between the solution obtained with the Round Robin algorithm and the global optimum from the centralized problem increases as the number of pools augments.

For this reasons the most suitable solution will be the primal decomposition.

Chapter 5

Conclusions and Future Work

In this Thesis, we have implemented an high-level optimizer solving the problem of finding optimal solutions for scheduling issues on a gravity-fed large-scale irrigation network. Our aim has been the scheduling of the requested off-takes fulfilling given constraints on the water-level and minimizing the delivery delays in water supply. Concerning this point, we have developed different optimization problems in order to achieve our objective.

In particular, first of all, we have discussed a suitable model of the irrigation network. Since the entire system was too large and complex, we have focused our attention on a part of the network. Starting from the continuous-time model of a pool, we have developed a state-space discrete-time model of a string of pools in the centralized fashion. The low-level decentralized control system, which guaranteed satisfactory performances in terms of robustness and closed-loop stability, has been embedded.

Secondly, in order to guarantee the water-level to fulfill given upper and lower bounds among the entire prediction horizon, we have defined a generic optimization problem formulation whose dynamical constraints were explicitly accounted for. Since our aim was to minimize the delivery delays, we have chosen the initial time of the water supply as the decisional variable. This has led to a non-linear optimization problem. We have, then, decided to turn into a $\{0, 1\}$ integer linear formulation via change of variables. Since such a program was too computationally demanding, we have reformulated it by relaxing the integer constraints. We have, hence, obtained two relaxed problem formulations that did not guarantee the preservation of the original shape of the requested profile but, on the

other hand, which lead to more computationally affordable optimization problems. Consequently, we have proposed different possible objective functions for all the problems. In particular, for the relaxed formulation, a linear and two quadratic solutions have been discussed and tested.

Finally, we have applied two distributed methods to decompose the relaxed problem into smaller-scale ones. Regarding this, we have sorted the vectors of variables and parameter and the matrices appearing in the constraints, in order to highlight the decomposable structure of the problem. Then, we have implemented and tested the Round Robin and the Primal Decomposition method. Finally, we have compared the results of both the distributed solutions and we have remarked the drawbacks and the advantages of both. As it has been thoroughly discussed, the more advisable implementation is the one obtained using the Primal Decomposition in view of its optimality properties.

Future work includes the implementation and the testing of a more large and complex problem, including more pools and more channels. Due to the excessive computational demand of the discussed Primal Decomposition method, new algorithms, dedicated to the analysed program should be implemented.

Concerning this, the Primal Decomposition solution is also suitable for further decomposition: each sub-problem related to a pool, can be further divided into single units solving the local problem of the single user (see Figure 5.1).

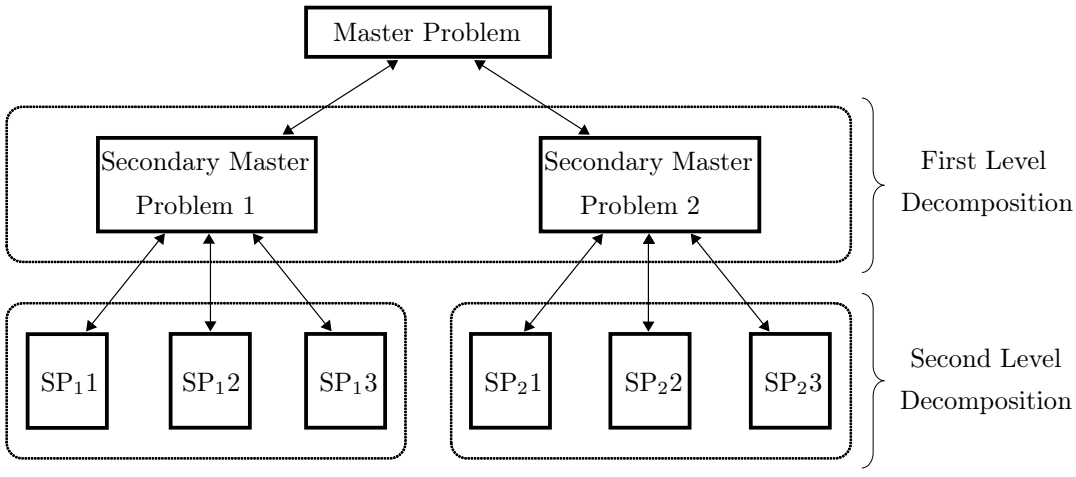


Figure 5.1: Example of hierarchical primal decomposition of the problem: the first level decomposition solves the problems related to the pools, the second one, the sub-problems of the single user

The information passing from the second to the first levels decomposition,

and vice versa, can be led by negotiation mechanisms, based on *Game Theory*. In the future, it will be possible to investigate also this structure.

Another problem, that should be further explored, is the decomposition of problems with tailored quadratic non definite objective functions that have been shown in this Thesis.

Bibliography

[1] J. Alende, Y. Li, and M. Cantoni. A {0,1} linear program for fixed-profile load scheduling and demand management in automated irrigation channels. *Proc. Joint 48th IEEE Conf. on Decision and Control and 8th Chinese Control Conf., Shanghai, P.R. China*, pages 597–602, December 2009.

[2] F. Barahona and R. Anbil. The volume algorithm: producing primal solutions with a subgradient method. *Mathematical Programming*, 87(3):385–399, 2000.

[3] S. Boyd and A. Mutapcic. Subgradient methods. *Noted for EE364B, Stanford University, Winter 2006-07*, January 2007.

[4] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[5] S. Boyd, L. Xiao, A. Mutapcic, and J. Mattingley. Notes on decomposition methods. *Notes for EE364B, Stanford University, Winter 2006-07*, April 2008.

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

[7] C. Farhat, M. Lesoinne, and K. Pierson. A scalable dual-primal domain decomposition method. *Numerical linear algebra with applications*, 7(7-8):687–714, 2000.

[8] S. Geerts and D. Raes. Deficit irrigation as an on-farm strategy to maximize crop water productivity in dry areas. *Agricultural Water Management*, 96(9):1275–1284, 2009.

[9] Y. Li, J. Alende, M. Cantoni, and B. De Schutter. Decomposition of a fixed-profile load scheduling method for large scale irrigation channel. *Proceedings of the 2010 IEEE International Conference on Control Applications, Yokohama, Japan*, pages 2166–2171, September 2010.

[10] I. Mareels, E. Weyer, M. Ooi, Ki Su ans Cantoni, Y. Li, and G. Nair. Systems engineering for irrigation systems: Successes and challenges. *Annual Reviews in Control*, 29(2):191–204, 2005.

[11] D. P. Palomar. Convex primal decomposition for multicarrier linear mimo transceivers. *Signal Processing, IEEE Transactions on*, 53(12):4661–4674, 2005.

[12] D. P. Palomar and M. Chiang. A tutorial on decomposition methods for network utility maximization. *IEEE Journal on Selected Areas in Communications*, 24(8):1439–1451, August 2006.

[13] D. P. Palomar and M. Chiang. Alternative distributed algorithms for network utility maximization: Framework and applications. *Automatic Control, IEEE Transactions on*, 52(12):2254–2269, 2007.

[14] H. Pennanen, A. Tolli, and M. Latva-aho. Multi-cell beamforming with decentralized coordination in cognitive and cellular networks. 2013, to appear.

[15] L. S. Pereira, T. Oweis, and A. Zairi. Irrigation management under water scarcity. *Agricultural water management*, 57(3):175–206, 2002.

[16] B. T. Poljak. *Introduction to optimization*. Optimization Software, 1987.

[17] L. Soltanian and M. Cantoni. Achieving string stability in irrigation channels under distributed distant-downstream control. *Proceedings of the 52nd IEEE Conference on Decision and Control*, 2013, to appear.

[18] E. Weyer. Decentralised pi control of an open water channel. In *Proceedings of the 15th IFAC world congress*, 2002.