Delft University of Technology

Delft Center for Systems and Control

Technical report 15-013

Delivery-oriented hierarchical predictive control of an irrigation canal: Event-driven versus time-driven approaches*

A. Sadowska, B. De Schutter, and P.-J. van Overloop

If you want to cite this report, please use the following reference instead:
A. Sadowska, B. De Schutter, and P.-J. van Overloop, "Delivery-oriented hierarchical predictive control of an irrigation canal: Event-driven versus time-driven approaches," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 5, pp. 1701–1716, 2015. doi:10.1109/TCST.2014.2381600

Delft Center for Systems and Control
Delft University of Technology
Mekelweg 2, 2628 CD Delft
The Netherlands
phone: +31-15-278.24.73 (secretary)
URL: `https://www.dcsc.tudelft.nl`

*This report can also be downloaded via `https://pub.deschutter.info/abs/15_013.html`

Delivery-oriented hierarchical predictive control of an irrigation canal: event-driven versus time-driven approaches

Anna Sadowska, Bart De Schutter and Peter-Jules van Overloop

Abstract

In this paper we present the concept of a hierarchical predictive controller used for irrigation canals. The motivation behind the work is the need in the field of irrigation to deliver water to farmers fast, but with minimal resources involved as the communication links in the field are not dependable in practice. In response to such a control problem we propose a hierarchical controller: the lower control layer is formed by decentralized PI controllers and the higher control layer is constituted by a centralized predictive controller the purpose of which is to control the inflow to the canal and, importantly, to coordinate the local controllers by modifying their setpoints. Having in mind the restrictions on the available communication infrastructure and the control equipment already present, the scheme is designed to be event-driven, i.e. activated when there are either delivery requests or non-delivery-related events of any sort, requiring special care on top of the control provided by the PI controllers. We also study a time-driven formulation with an additional post-processing step to avoid excessive negligible setpoint modifications. We compare the event-driven formulation and the time-driven formulation theoretically as well as by means of a simulation study for the West-M irrigation canal in Phoenix, Arizona, illustrating the findings of the paper. It is shown that the event-driven controller is able to provide a good balance between the control performance and the required update frequency of the control settings.

I. Introduction

Irrigation relies on providing water to agriculture to facilitate growth in crops in areas where natural precipitation does not suffice to produce adequate harvest yield. With that in mind, irrigation canals are often used to deliver water to farmers from a source like a river, a lake or a dam. In fact, irrigation systems originated as early as 6000 BC in ancient Egypt and Mesopotamia [1]. Nowadays, more than 90% of the total consumptive water use is generated by irrigation [2]. Therefore, it is of the highest importance to be able to operate the irrigation canals efficiently and dependably.

A. Inspirations and our solution

Irrigation canals consist of a number of cascade-connected pools, between which there are control structures like pumps or gates, controlling the flow between the neighboring pools. Over the years, multiple methods have been proposed to control irrigation canals, see [3–5] for an overview. Some of the proposed methods [6–11] rely on simple feedback mechanisms involving mainly PI controllers installed at each gate to maintain water levels in the individual pools at some specified setpoints (cf. [12], where application of a PI controller to a river stretch is discussed). Also, [13] introduced a simple to implement feedback controller, derived using the concept of Riemann invariants. Moreover, in [14] the design of linear feedback controllers enhanced with an anti-windup mechanism to account for the control variable constraints was considered. In contrast, a feedforward control method was proposed in [15], where the notion of differential flatness [16] was explored. In particular, a control law to manipulate the upstream water inflow to the pool was proposed in [15] ensuring that the discharge at the downstream end of the pool converges to the desired value. A different approach to feedforward control of an irrigation canal was pursued in [17]. The authors of that paper studied a method to adjust the gates in a canal so as to compensate for the volume change of water in a pool due to the known or predicted offtake.

A. Sadowska and B. De Schutter are with Delft Center for Systems and Control, Delft University of Technology, The Netherlands; P.-J. van Overloop is with the Water Resources Management, Delft University of Technology, The Netherlands (e-mail: {A.D.Sadowska, B.DeSchutter, P.J.A.T.M.vanOverloop}@tudelft.nl).

Other control methods that have been proposed include centralized and decentralized schemes using optimal control methods such as LQ control or Model Predictive Control [18–28]. Such optimal predictive control methods rely on finding optimal control actions minimizing a specified control objective, taking into account predictions of how the system will behave in the future given forecasts of external disturbances. Out of the aforementioned papers, [20, 26, 27] proposed a distributed scheme (cf. [29]) in which control settings of individual gates are found independently for each gate, taking into account the state of the pools immediately upstream and immediately downstream. While such a distributed control approach may facilitate more efficient computations, it may also be associated with inferior performance as the controllers are lacking global knowledge and use only local information available to them. Moreover, in distributed schemes there may be a need for a large volume of communication between neighboring gates during the process of negotiating local control actions. This, given the harsh environment that the irrigation canals are usually in, may prove unrealistic in practical applications. The use of local as opposed to global controllers was also the focus of attention of [30, 31], where distant downstream controllers were utilized [32], which results in the necessity for local communication links at each gate.

In [23] a predictive control scheme was introduced for deployment under different interchangeable operating modes of a canal. A supervisory strategy was studied to deal with the multiple operating modes of the system. In particular, the supervisory controller was designed to detect the operating mode in action and to activate a suitable predictive controller accordingly. Furthermore, in [33] a control scheme was discussed that aims at finding a suitable balance between the objectives of making water available to the users as much and as timely as possible and, on the other hand, rejecting unknown disturbances.

In [19] the performance of decentralized PI controllers is compared with that of a centralized LQ controller. It is found that the centralized LQ controller outperforms the decentralized PI controllers albeit at the cost of more issues that might unfold, e.g. a more difficult tuning process. In general, centralized controllers are able to achieve a better performance than decentralized or even distributed controllers as they are able to find the control actions that are adequate for the system as a whole, as opposed to multiple control actions each adequate for a small partition of the system but not necessary adding up together to be adequate for the whole system. However, decentralized PI controllers are still a popular and by far the most practically used controllers in the automated control of irrigation canals. The practitioners in the field mainly value them for the fact that they are simple, model-independent, and can provide robust functioning if tuned properly. In addition, decentralized PI controllers do not require any communication between each other.

In this paper, we consider the practical restriction widely present in the field of irrigation that the communication links are not always dependable, with frequently occurring equipment breakdowns due to the harsh outdoor environment that the communication links are located in. This unreliability is simply because of insufficient funds being spent on the maintenance and modernization of the communication equipment as this would need to be paid for by the farmers. Moreover, in view of the currently installed control equipment (i.e. decentralized PI controllers) and the appreciation they get by the practitioners, the approach we propose does not aim at replacing them by more advanced controllers but rather to incorporate them without making any changes to the local PI controllers (such as new control software, more advanced processors, extra memory, etc.). The scheme that we propose is designed not to rely on a frequent communication and in particular it works in an event-driven manner, where individual events are associated with delivery requests and also with other special circumstances occurring in the canal and requiring extra care e.g. heavy rainfall. Our scheme will thus step in only when there is a need for it and if so, it will modify the setpoints of the local PI controllers. In times of a normal operation, we propose to use local decentralized PI controllers along the canal for upstream control.

Altogether the scheme that we propose to control an irrigation canal is a hierarchical event-driven controller, see Figure 1. The lower control layer is formed by the PI controllers; hence, the scheme is based on the equipment and solution that is already working satisfactorily for the users. However, to speed up the delivery process, i.e. to make water available to the farmers faster than it is currently possible, we propose to use a higher-layer predictive centralized controller that coordinates the local controllers. In particular, this centralized controller - the Coordinator - modifies the setpoints of the PI controllers when it is needed, as well as controls the head gate. Note that it is unwelcome to allow frequent setpoint changes as this might generate an undesirable behavior, which reiterates the need for the higher-layer controller to be event-driven. Moreover, in this way, even when the communication links are temporarily down, the PI controllers can still autonomously control.

By the design of the hierarchical controller and in particular by the actions of the Coordinator, water can be

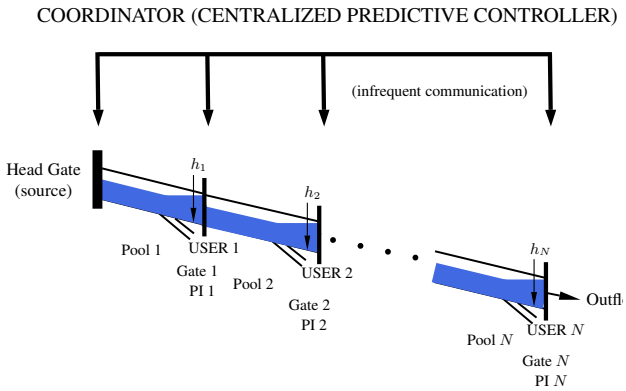

Figure 1. The structure of the hierarchical controller proposed in the paper.

made available to the farmers faster than when only local PI controllers operate. Indeed, if only PI controllers are utilized, the requested offtake is accomplished by first releasing the required amount of water from the head gate. Then, as that water travels downstream, the water level in Pool 1 starts to increase and hence the PI controller in Pool 1 reacts to the increasing deviation in the water level with respect to the given setpoint and water is released to Pool 2. This situation is reiterated in every pool until the water reaches the offtake point of the farmer requesting the delivery. Such a method is fit to deliver water to farmers, however, there may be a significant amount of time required before an offtake can be executed after it was announced.

Importantly, a restriction that we take into consideration when analyzing the water delivery process is that if a stretch of the canal is considered, with say N pools, the outflow from the N^{th} pool should be kept as close as possible to the given base flow due to a possible existence of further downstream users. Because of this assumption, a farmer is not allowed to simply start the offtake without waiting for the water to be delivered from the head gate as that would disturb the outflow from the canal. Thus, the time delay between the announcement moment and offtake starting moment is inevitable. However, as will be discussed in Sections III and IV, and further illustrated in Section V, the hierarchical controller proposed in this paper can significantly improve this situation.

In order to connect to the way the PI controllers operate, we propose that the setpoint changes are block-shaped (see Sections III and IV for more details), and introduce two methods to modify setpoints that are already changed in the previous runs of the higher control layer: the block-modifying approach and the block-adding approach. In short, in the block-modifying approach, once a block-shaped setpoint change is assigned, it can only be modified in the future by either extending a block or shortening it. In contrast, in the block-adding formulation, once a block-shaped change is ordered, it cannot change per se: future changes of the setpoints are done by adding new blocks on top of the previously changed setpoint profiles.

To aid an in-depth analysis of the event-driven controller, we also study a time-driven formulation of the Coordinator in which the Coordinator in activated at some regular intervals. Indeed, various sources in the control systems literature claim that the event-driven control can facilitate decreased resource utilization while it is argued that the time-driven control can serve to improve the system performance [34–36]. In this paper, we also verify this claim in the scope of the irrigation canal control by comparing the performance obtained through the time-driven and event-driven implementation of the Coordinator.

B. Previous developments

Hierarchical MPC has shown to offer a balance between local and global control perspectives [37, 38]. Amongst numerous applications for which hierarchical MPC has been proposed, [39] considered multiple-commodity transportation networks, [40] studied the air traffic control problem, and [41] introduced a flexible hierarchical structure for a multi-agent operation.

The idea of a supervisory control approach in which setpoints are changed by a higher-layer control layer was previously studied in e.g. [42]. The control structure in [42] consists of three layers, where the controllers in the two higher layers assign setpoints to the local PID controllers in the lower layer to ensure disturbance rejection in an integrated wastewater treatment system. Moreover, [43] treated supervisory control of a power network. The authors

of that paper introduced a scheme in which the higher control layer modifies the setpoints of the local controllers to avoid voltage collapse. This was done in a time-driven manner in every control step. For a water system, a supervisory control scheme was studied in [44], where various risk factors (e.g. political, operational, or financial) were considered. In view of these risk factors, the supervisory layer of the controller provides the lower layer with desired setpoints. Then, the lower layer attempts to find suitable control actions using a distributed scheme with intensive communication needed at each control step between the individual local sites to facilitate negotiations. Similarly, [45] proposed a hierarchical control approach for a drinking water network with a higher-layer controller assigning setpoints, and distributed predictive controllers in the lower layer. In general, distributed schemes suffer from a large amount of communication that is required at each control step between each pair of neighboring pools to reach consensus on the local control actions. This, given the unreliable communication in the system that we consider in this paper, may prove impracticable.

C. Our contributions

This papers present two new contributions with respect to the state of the art. First, given the communication restrictions widely present in the field of irrigation as well as the currently installed control equipment, to control an irrigation canal we introduce the concept of the Coordinator, i.e. a hierarchical controller, able to activate concurrently for multiple events, yet not requiring a continuous communication[1]. The controller design provides an improved performance with respect to the utilization of decentralized PI controllers only: in particular, with regard to water delivery requests, the required amount of water can be made available faster when the Coordinator is applied than when only the PI controllers operate. Second, we propose two approaches to deal with multiple activations: the block-adding one and the block-modifying one. In addition, we compare the proposed event-driven approach with a time-driven approach and show that the event-driven approach (specifically using the block-adding formulation) in general presents the best trade-off between reducing communication frequency and improving control performance.

The outline of this paper is as follows. In Section II we present the model of an irrigation canal and some background information on the control design tools used in the paper. Afterwards, in Section III we present the concept of the Coordinator in a simplified case, in which only one activation occurs for a single event. Then, in Section IV we propose our main result, i.e. we present the design of the hierarchical controller for multiple concurrent events, using the time-driven formulation (in Section IV-B) and the event-driven formulation (in Section IV-C). In Section V we illustrate the methods introduced in the paper by means of a simulation study using a numerical model of the West-M irrigation canal in Phoenix, Arizona. Final remarks are given in Section VI.

II. Preliminaries

In this section we present mathematical preliminaries employed in the paper. We start by describing the model of the canal in Section II-A. Then, in Section II-B we introduce the concept of the time instant optimization.

A. Model of an irrigation canal

This section describes the model of the canal as used in the paper, both as prediction model for MPC as well as plant model for the simulations. As discussed in [10, 46–49], a linear model of a canal is suitable to adequately capture its dynamics. We assume here that the canal consists of N pools. For pool i the model reads

$$\begin{aligned} h_i(k+1) &= h_i(k) + \tfrac{T_\mathrm{m}}{c_i}(u_{i-1}(k-k_{\mathrm{d}i}) - u_i(k) + d_i(k)), \\ u_i(k) &= u_i(k-1) + K_{\mathrm{P}i}(e_i(k) - e_i(k-1)) + K_{\mathrm{I}i}e_i(k), \qquad \text{(PI controller)} \\ u_0(k) &= Q_\mathrm{S}(k), \\ e_i(k) &= h_i(k) - h_i^{\mathrm{ref}}(k), \end{aligned} \tag{1}$$

where $k \in \mathbb{N}$ is the discrete time step counter, h_i the water level at the downstream end of Pool i, d_i an external net inflow, e_i the error between the water level in Pool i and the given setpoint, h_i^{ref}. Moreover, T_m denotes the sampling period (equal for all pools), c_i is the surface area, and $k_{\mathrm{d}i}$ is a time delay (in sampling steps) representing the time required for an inflow from the upstream gate $i-1$ to influence the water level h_i in Pool i. In addition, Q_S

[1]We note that the restrictions on the communications are not universal for water systems. For instance, for urban water networks (cf. [45]), reliable communication links are often present.

is the inflow to the canal (to Pool 1) from the head gate and u_i is the control input denoting the outflow from Pool i. Note that values of the control actions u_i are given by the PI controllers with the proportional and integral gains denoted with $K_{\mathrm{P}i}$ and $K_{\mathrm{I}i}$, respectively. Moreover, the dynamics (1) also depend on the control inputs determined by the Coordinator. These are, briefly speaking, the values of the setpoints h_i^{ref} and the inflow Q_{S} as discussed in detail in Sections III and IV.

B. Time Instant Optimization MPC

Time instant optimization is a special case of the classical MPC (see [50, 51] for more information on MPC in general), and it was first introduced for traffic control [52]. For a water system, it can be a useful tool to deal with discontinuous on/off hydraulic structures [53, 54]. When using classical MPC for discontinuous control structures, a decision needs to be made at each sampling step about the optimal control sequence from now on until the end of the prediction horizon N_{p}, i.e. a chain of N_{p} elements of *on* or *off* states needs to be found. This is a combinatorial problem resulting in a mixed-integer linear or nonlinear programming problem with N_{p} binary control variables for each control structure present in the system. Such a problem may prove to be intractable [55]: except for problems of a very limited dimension, the solution may be impossible to be found in real time. The way such a problem can be solved using time instant optimization MPC is to first decide how many switches of the control structure there should be during the prediction horizon N_{p} and consequently to write down the optimization problem so that the time instants of the on/off switches are now the direct control variables. Therefore, by recasting the problem into a linear or nonlinear programming problems with real variables only, the problem may be more efficiently solved in terms of the computational effort.

III. Hierarchical predictive controller design: single activation case

In this section we introduce the hierarchical predictive controller for the purpose of controlling an irrigation canal. To simplify the introduction of the concept of the Coordinator, the discussion in this section concerns the case when the Coordinator is only activated once and a single set of actions is ordered. In the sequel, see Section IV, the specifics regarding multiple activations of the Coordinator are studied.

The hierarchical controller proposed in this paper consists of two control layers (cf. [43–45], where hierarchical controllers in different settings are studied). In the lower control layer, local PI controllers take control of the canal and maintain the water levels at some predefined setpoints. The higher control layer is formed by a centralized predictive controller - the Coordinator - the purpose of which is to coordinate the local PI controllers. In doing so, the centralized higher-layer controller explicitly considers the coupling between neighboring pools to suitably coordinate the local controllers. The coupling mechanism exhibits itself through the relation between the outflow from an upstream pool and the water level in a downstream pool, see (1). We present the timing of the Coordinator for a single activation in Figure 2. Every time the Coordinator is activated, the time variable t, the step counter k, and the step counter associated with the Coordinator, k_{c}, are reset to 0. We assume $A_{\mathrm{c}} = T_{\mathrm{c}}/T_{\mathrm{m}} \in \mathbb{N}$, where T_{c} is the duration of the Coordinator's control cycle and T_{m} is the model sampling time.

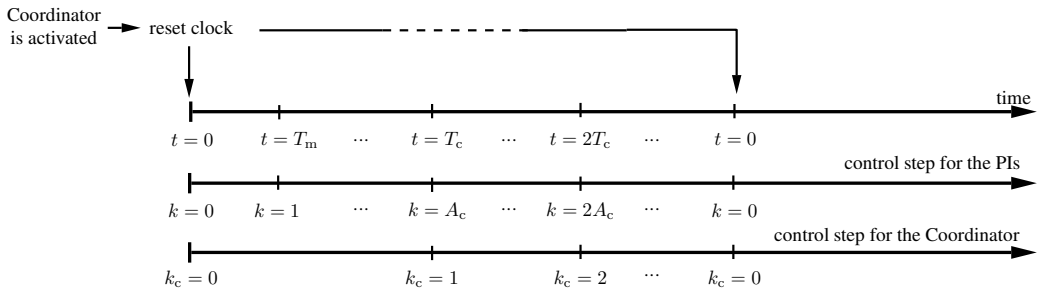

Figure 2. Timing of the Coordinator for the single activation case. A new activation can only occur after the system returns to the steady state after a preceding activation.

We propose that the Coordinator coordinates the PI controllers by modifying their setpoints when needed. The individual admissible setpoint changes are block-shaped, see Figure 3. However, due to possible overlapping of

setpoint changes owing to multiple activations of the Coordinator, more complex setpoint profiles may emerge (see Section IV).

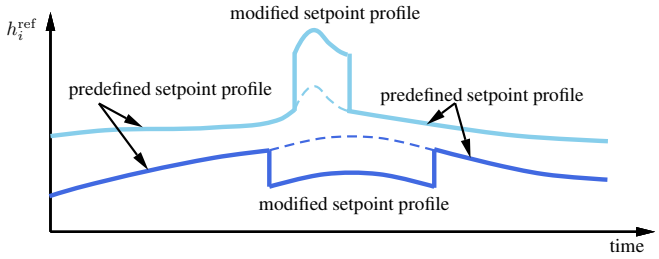

Figure 3. Admissible block-shaped setpoint profiles for a time-varying predefined setpoint level.

To define a single block-shaped setpoint change, three attributes are required. In that respect, the Coordinator needs to find a triple $(t_i^{\mathrm{on}}, t_i^{\mathrm{off}}, \Delta h_i^{\mathrm{ref,dynamic}})$ for each $i \in \{1, \ldots, N\}$ with $t_i^{\mathrm{on}}, t_i^{\mathrm{off}}, \Delta h_i^{\mathrm{ref,dynamic}} \in \mathbb{R}$. Here, t_i^{on} denotes the time instant of the setpoint change when the setpoint is altered from its predefined profile, t_i^{off} the time instant when the setpoint returns to its predefined profile, and $\Delta h_i^{\mathrm{ref,dynamic}}$ the difference that should be applied to the predefined profile. For the whole canal, we construct $T^{\mathrm{on}} = (t_1^{\mathrm{on}}, \ldots, t_N^{\mathrm{on}})^T$, $T^{\mathrm{off}} = (t_1^{\mathrm{off}}, \ldots, t_N^{\mathrm{off}})^T$, and $\Delta H^{\mathrm{ref,\,dynamic}} = (\Delta h_1^{\mathrm{ref,\,dynamic}}, \ldots, \Delta h_N^{\mathrm{ref,\,dynamic}})^T$. With the triples $(t_i^{\mathrm{on}}, t_i^{\mathrm{off}}, \Delta h_i^{\mathrm{ref,dynamic}})$, the setpoint profile for canal pool i can in general be evaluated according to

$$h_i^{\mathrm{ref}}(k) = \begin{cases} h_i^{\mathrm{ref,\,normal}}(k) & \text{if } k \leq k_i^{\mathrm{on}} \text{ or } k \geq k_i^{\mathrm{off}}, \\ h_i^{\mathrm{ref,\,normal}}(k) + \Delta h_i^{\mathrm{ref,\,dynamic}} & \text{otherwise,} \end{cases} \tag{2}$$

in which $h_i^{\mathrm{ref,\,normal}} \in \mathbb{R}$ is the normal, possibly time-varying setpoint level. Moreover, k_i^{on} and k_i^{off} are discrete-time equivalents of the continuous switching time instants t_i^{on} and t_i^{off} given a certain model sampling time T_{m}:

$$k_i^{\mathrm{on}} = \left[\frac{t_i^{\mathrm{on}}}{T_{\mathrm{m}}}\right] \qquad \text{and} \qquad k_i^{\mathrm{off}} = \left[\frac{t_i^{\mathrm{off}}}{T_{\mathrm{m}}}\right], \tag{3}$$

where we use $[x]$ to denote rounding a continuous variable $x \in \mathbb{R}$ to the nearest integer assuming a round-half-up rule. The parametrization of the setpoint profiles h_i^{ref} with respect to $(t_i^{\mathrm{on}}, t_i^{\mathrm{off}}, \Delta h_i^{\mathrm{ref,dynamic}})$ is selected to accommodate the communication and equipment limitations present in the system. In particular, this choice introduces block-shaped setpoint changes, which for memory-less PI controllers can be communicated with two transmissions in comparison to continuous setpoint changes that would require more frequent communication. Moreover, frequent setpoint changes could produce undesirable behavior and the parametrization limits the number of setpoint changes to two per activation. In addition, with the parametrization $(t_i^{\mathrm{on}}, t_i^{\mathrm{off}}, \Delta h_i^{\mathrm{ref,dynamic}})$ only three variables per pool suffice to define the setpoint profile whereas N_{p} elements would be required for continuous setpoint modifications, resulting in higher computational burden and possibly intractability. This could be particularly an issue if a more accurate nonlinear model of a canal were used, whereas the proposed parametrization could be easily extended to the case of a nonlinear prediction model.

In addition to the setpoint modifications, the Coordinator also determines the extra inflow from the head gate that is required on top of the base flow to accommodate for the offtake requested. Every time the Coordinator is activated, it provides the modification profile of the flow from the head gate for the whole prediction horizon N_{p}:

$$\widetilde{Q}_{\mathrm{S,\,demand}} = \left(Q_{\mathrm{S,\,demand}}(0), \ldots, Q_{\mathrm{S,\,demand}}(N_{\mathrm{p}} - 1)\right)^T. \tag{4}$$

The overall flow from the head gate Q_{S} for a given Coordinator's input $\widetilde{Q}_{\mathrm{S,\,demand}}$ and the base level of the flow $Q_{\mathrm{S,\,base}}$ is evaluated according to

$$Q_{\mathrm{S}}(jA_{\mathrm{c}} + i) = Q_{\mathrm{S,\,base}} + Q_{\mathrm{S,\,demand}}(j), \tag{5}$$

for $j = 0, \ldots, N_{\mathrm{p}} - 1$ and $i = 0, \ldots, A_{\mathrm{c}} - 1$.

The overall control input can be in general written as a tuple

$$\mathcal{U} = \left(\widetilde{Q}_{\mathrm{S,\,demand}}, \quad T^{\mathrm{on}}, \quad T^{\mathrm{off}}, \quad \Delta H^{\mathrm{ref,\,dynamic}} \right), \tag{6}$$

containing the information on the required setpoint modifications in all pools $(T^{\mathrm{on}}, T^{\mathrm{off}}, \Delta H^{\mathrm{ref,\,dynamic}})$ as well as the required modification to the head gate flow $\widetilde{Q}_{\mathrm{S,\,demand}}$. The Coordinator finds $\mathcal{U}$ to minimize the cost function

$$J = \alpha \sum_{j=1}^{A_{\mathrm{c}} N_{\mathrm{p}}} \left(u_N \left(j-1 \right) - Q_{\mathrm{S,\,base}} \right)^2 \tag{7a}$$

$$+ \sum_{i=1}^{N} \sum_{j=1}^{A_{\mathrm{c}} N_{\mathrm{p}}} \Bigg[\gamma_1 \left(\max(h_i \left(j \right) - h_i^{\mathrm{max,des}}, 0) \right)^2 \tag{7b}$$

$$+ \gamma_2 \left(\min(h_i \left(j \right) + h_i^{\mathrm{min,des}}, 0) \right)^2 \Bigg] \tag{7c}$$

$$+ \beta \sum_{i=1}^{N} \sum_{j=1}^{A_{\mathrm{c}} N_{\mathrm{p}}} \left(h_i \left(j \right) - h_i^{\mathrm{ref}} \left(j \right) \right)^2 \tag{7d}$$

$$+ \mu \sum_{i=1}^{N} \left(t_i^{\mathrm{off}} - t_i^{\mathrm{on}} \right)^2 + \zeta \sum_{i=1}^{N} \left(\sum_{j=1}^{A_{\mathrm{c}} N_{\mathrm{p}}} \left| h_i^{\mathrm{ref}}(j) - h_i^{\mathrm{ref,normal}}(j) \right| \right)^2, \tag{7e}$$

in which α, β, γ_1, γ_2, μ and ζ are positive weighting coefficients. Moreover, $u_N(k)$ denotes the flow through gate N at time step k (cf. (1)). The length of the prediction horizon N_{p} should correspond to the time span needed for the transient behavior to vanish in the sense of the norm of the transient signals going below a given threshold [50, 51]. This can be determined from real data or by using an accurate simulator. Note that if the prediction horizon N_{p} is chosen too short, unstable behavior may result.

The first term in the cost function J (7a) vanishes when the outflow from the last gate is exactly equal to the given base flow $Q_{\mathrm{S,\,base}}$ and grows as that outflow diverges from the base flow. The second (7b) and third (7c) terms penalize control actions resulting in the water levels departing from the desired operational range $[h_i^{\mathrm{min,des}}, h_i^{\mathrm{max,des}}]$, $i = 1, \ldots, N$. Furthermore, the fourth term (7d) adds a penalty on the error between the actual water levels and their respective setpoints. Finally, the last double term (7e) is used to induce the Coordinator to switch the setpoints back to their normal level as soon as possible. Overall, the main objective captured in the cost function J is that the events are accommodated for with possibly minimal disruptions to the rest of the canal.

In finding the control actions $\mathcal{U}$, the Coordinator must comply with the following hard constraints

$$h_i^{\mathrm{min}} \leq h_i(j) \leq h_i^{\mathrm{max}}, \qquad j = 1, \ldots, N_{\mathrm{p}} A_{\mathrm{c}}, \tag{8a}$$

$$h_i^{\mathrm{min}} \leq h_i^{\mathrm{ref}}(j) \leq h_i^{\mathrm{max}}, \qquad j = 0, \ldots, N_{\mathrm{p}} A_{\mathrm{c}} - 1, \tag{8b}$$

$$t_i^{\mathrm{off}} \geq t_i^{\mathrm{on}} + T_{\mathrm{m}}, \tag{8c}$$

$$t_i^{\mathrm{on}} \geq 0 \tag{8d}$$

$$0 \leq Q_{\mathrm{S}}(j) \leq Q_{\mathrm{capacity}}, \qquad j = 0, \ldots, N_{\mathrm{p}} A_{\mathrm{c}} - 1, \tag{8e}$$

for all $i \in \{1, \ldots, N\}$. Constraints (8a) and (8b) correspond to the physical constraints of the depth of the canal and are included to avoid flooding and ensure adequate functioning. Note that we require $h_i^{\mathrm{min}} \leq h_i^{\mathrm{min,des}} < h_i^{\mathrm{max,des}} \leq h_i^{\mathrm{max}}$. Constraints (8c) and (8d) limit the possible choice of the switching time instants in that the first switch t_i^{on} can only occur after the moment the Coordinator has been activated and the second moment t_i^{off} needs to occur at least one sampling step after the first one. In addition, the head gate flow needs to respect the maximum capacity of the head gate constraint (8e).

The resulting optimization problem is either a nonlinear nonsmooth real-valued problem if t_i^{on} and t_i^{off} are the optimization variables or a nonlinear mixed-integer problem if k_i^{on} and k_i^{off} are directly optimized. Various algorithms can be used to deal with such problems e.g. simulated annealing [56], genetic algorithms [57], pattern-search [58], or branch and bound [59].

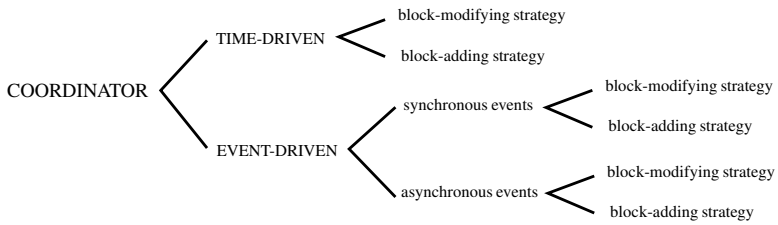

Figure 4. Various formulations of the Coordinator proposed.

Recall that the Coordinator design in this section deals with one event by means of a single activation. Therefore, the variables introduced in this section do not depend on the control step or the activation counter. However, in the next section we extend the notion of the Coordinator to enable its continuous operation over a long period of time in which multiple activations occur. This requires a special treatment as is shown in the following section.

IV. Hierarchical predictive controller design: multiple activations case

A. General setup

In order to allow multiple activations of the higher-layer controller, we propose two possible design structures of the Coordinator: the time-driven design and the event-driven design[2], see Figure 4. In the first one, see Section IV-B, the Coordinator is invoked at regular intervals, every T_{c} time units. This way, better performance can likely be achieved than in the event-driven design as setpoints can be changed at every control step to react to the actual situation in the canal. However, this requires a reliable communication infrastructure. In the event-driven structure, see Section IV-C, the Coordinator changes setpoints no more frequently than actual events occur. Possible events are delivery-related, i.e. delivery requests, and non-delivery-related, e.g. heavy precipitation.

We consider two possibilities how individual events are defined in the event-driven approach. These are called the synchronous case and the asynchronous case depending on whether the events can only occur at some prespecified or indeed arbitrary times.

For both the time-driven formulation and the event-driven formulation, we propose two sub-approaches that determine how to deal with multiple activations of the Coordinator, i.e. how to obtain the flow from the head gate as well as how to modify the setpoint profiles if they have already been modified before when the Coordinator was previously activated. These are called the block-modifying approach and the block-adding approach.

B. Time-driven formulation

In the time-driven formulation setup, the Coordinator is activated at regular intervals, every T_{c} time units, see Figure 5. Note that similarly to the settings in Section III in this scheme the time variable t and the sampling step k are reset to zero at every multiple of k_{c}. However, as opposed to the procedure proposed in Section III, the step counter associated with the Coordinator, k_{c}, is not reset to 0 after an activation of the Coordinator but it carries on incrementing from the initial moment to facilitate multiple activations of the Coordinator. In the figure, $k_{\mathrm{c},0}$ denotes some initial value of k_{c} for the time period captured in the graph.

Because of the regular activations, the Coordinator can e.g. counteract possible model-plant mismatches by incorporating a systematic feedback loop. However, if the Coordinator is activated regardless of whether or not there is a need for it, there may be a risk of excessive setpoint changes ordered by the Coordinator. In other words, the Coordinator may require that many small changes are done that in fact benefit the system very little. To prevent this, a parameter ϵ can be appropriately selected so that negligible setpoint changes less than ϵ with respect to the current settings, are not communicated to the local sites.

Note that in the time-driven approach, the cost function is evaluated at every control step k_{c}; hence we explicitly use the notation $J = J(k_{\mathrm{c}})$. The cost function $J(k_{\mathrm{c}})$ is minimized to find the optimal control action

[2]Just as in the case of a single activation, the resulting programming problem in both the time-driven and the event-driven approaches is a nonlinear nonsmooth problem or a nonlinear mixed-integer problem that can be solved using a number of algorithms e.g. simulated annealing [56], genetic algorithms [57], pattern-search [58], or branch and bound [59].

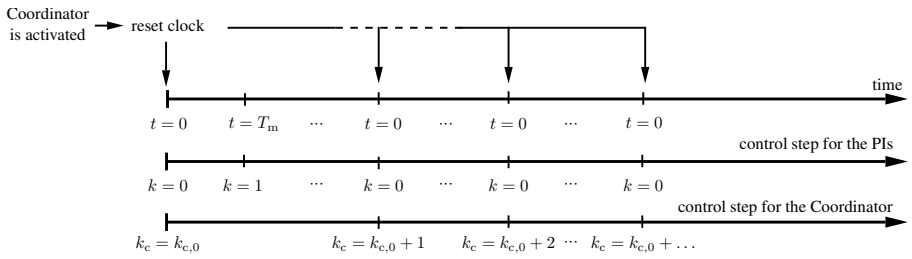

Figure 5. Timing of the Coordinator in the time-driven multiple activation case: the Coordinator is activated at every control step.

$\mathcal{U} = \mathcal{U}(k_{\mathrm{c}}) = (\widetilde{Q}_{\mathrm{S,\,demand}}(k_{\mathrm{c}}), \Delta H^{\mathrm{ref,\,dynamic}}(k_{\mathrm{c}}), T^{\mathrm{on}}(k_{\mathrm{c}}), T^{\mathrm{off}}(k_{\mathrm{c}}))^T$ as in (6). In particular, at every control step k_{c} new setpoint modifications $(t_i^{\mathrm{on}}(k_{\mathrm{c}}), t_i^{\mathrm{off}}(k_{\mathrm{c}}), \Delta h_i^{\mathrm{ref,dynamic}}(k_{\mathrm{c}}))$ are allowed. Given $\mathcal{U}(k_{\mathrm{c}})$, the flow from the head gate $Q_{\mathrm{S}}(k)$ at step k is calculated using the principles of the rolling horizon control as follows. Knowing the base flow in the canal $Q_{\mathrm{S,\,base}}$ and the optimal profile as ordered by the Coordinator at step k_{c} i.e. $\widetilde{Q}_{\mathrm{S,\,demand}}(k_{\mathrm{c}}) = (Q_{\mathrm{S,\,demand},k_{\mathrm{c}}}(0), \ldots, Q_{\mathrm{S,\,demand},k_{\mathrm{c}}}(N_{\mathrm{p}}-1))^T$ the flow from the head gate is

$$Q_{\mathrm{S}}(j) = Q_{\mathrm{S,\,base}} + Q_{\mathrm{S,\,demand},k_{\mathrm{c}}}(0), \qquad j = 0, \ldots, A_{\mathrm{c}} - 1, \tag{9}$$

where we used the fact that in the time-driven implementation of the Coordinator, the model step counter k is reset to zero whenever the control step k_{c} is incremented.

Before we describe the way to handle possibly overlapping setpoint modifications, let us recall from the classical MPC that from the optimal control sequence for the given predictions horizon only the first value is applied to the system, i.e. the one representing what should optimally be done in the current moment. This rolling horizon strategy is also used in optimizing the setpoint changing time instants. In particular, if according to the most recent control action $\mathcal{U}(k_{\mathrm{c}})$, there should be a setpoint modified from its predefined profile after $k_{\mathrm{c}}+1$, it is not implemented in control step k_{c} because such a control action can be recalculated at the next run of the Coordinator at step $k_{\mathrm{c}}+1$ utilizing more up-to-date data. However, if the modification is supposed to happen between $k_{\mathrm{c}}T_{\mathrm{c}}$ and $(k_{\mathrm{c}}+1)T_{\mathrm{c}}$, i.e. in the current control step k_{c}, this change is communicated as the next run of the Coordinator at step $k_{\mathrm{c}}+1$ will occur after the scheduled setpoint change.

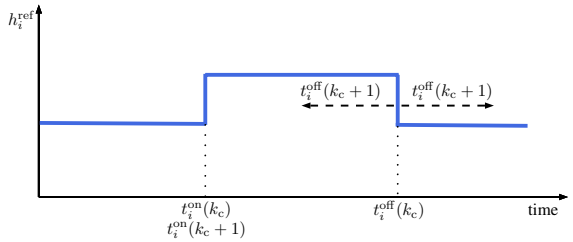

Figure 6. Possible setpoint profiles using block-modifying formulation.

We propose two possibilities to deal with re-activations of the Coordinator while setpoint changes from a preceding activation are ongoing: *the block-modifying approach* and *the block-adding approach*. In the block-modifying approach, see Figure 6, once a setpoint modification has started in one of the pools and is supposed to last until at least the next run of the Coordinator at step $k_{\mathrm{c}}+1$, this modification can only be altered in the future by changing when the setpoint should return to the normal operating level. Therefore, the setpoints profile will consist of multiple changes, each one starting after the previous change has finished. This means that if at some run of the Coordinator, a setpoint modification is ongoing i.e. the current time is passed the time instant of the first modification t_i^{on} for the current setpoint modification but before the second time instant t_i^{off} when the setpoint should return to its predefined profile, only the value of t_i^{off} can be still modified by the Coordinator.

In contrast, in the block-adding approach, new blocks are added to the setpoint profile on top of the existing, possibly already modified setpoint profile, see Figure 7. By doing so, a more complicated setpoint profile may emerge than in the block-modifying scheme, which can facilitate better performance of the system as setpoints may

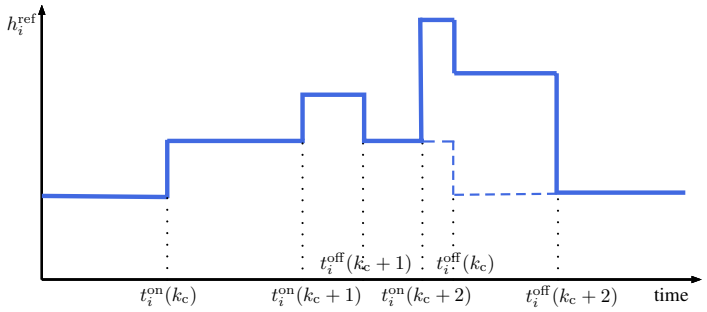

Figure 7. Possible setpoint profiles using block-adding formulation.

be more freely changed. Yet, more communication between the Coordinator and the local sites may be required to provide the local sites with the necessary information how the setpoints should change. Furthermore, performance of the PI controllers could be diminished if the setpoints are modified too frequently.

For the sake of clarity, we discuss the specific implementation issues related with the block-modifying and the block-adding approaches in Appendix A.

C. Event-driven formulation

In the event-driven approach, the Coordinator is only activated when there is an event. We define activations of the Coordinator in a twofold manner. Assume that the activations can occur no more often than the control interval T_{c} to avoid too frequent Coordinator's activations so it can at most be activated as often as in the time-driven case. Two ways of defining activations are shown in Figures 8 and 9. In the *synchronous case*, the Coordinator's activation can only occur at multiples of the control interval T_{c}. If there are events between two control steps, they are sent jointly to the Coordinator at the next control step. In Figure 8, dotted bars indicate individual events and vertical arrows indicate when the Coordinator is activated. The numeric values next to the arrows demonstrate for how many individual events the Coordinator is activated each time. It is seen that events do not activate the Coordinator immediately but at the nearest following multiple of the control step k_{c}.

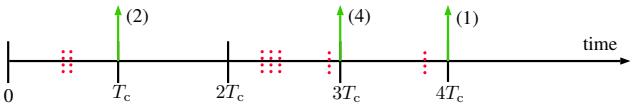

Figure 8. Definition of events causing activation of the Coordinator - synchronous case. Events are denoted with dotted bars. Arrows are used to indicate when the activation occurs with a label representing how many events are dealt with during each activation.

In contrast, in the *asynchronous case* the Coordinator can be activated at any moment, complying only with the model sampling time T_{m}, but not necessarily with the control interval T_{c}. This case is illustrated in Figure 9, where a time window δ is given and denoted with a horizontal bar, which is used to accumulate events before they are sent to the Coordinator. At the end of the time window, the Coordinator is activated for all events that occurred within the given time window. Then, another activation of the Coordinator can happen T_{c} time units after the previous one at the earliest, due to the prerequisite of the minimal reactivation time of the Coordinator. A special case is when $\delta = 0$ in which case no events are grouped together and every single event immediately activates the Coordinator. Any events occurring after the activating event need to wait for another activation after the minimal reactivation time passes by. Again, if there are multiple events before the reactivation time elapses, they are grouped together and jointly activate the Coordinator.

In both the synchronous and the asynchronous cases we define a new variable $s \in \mathbb{N}$, which is an activation counter. It is initialized with a value 0 and is increased every time the Coordinator is activated, see Figure 10. Note that for the purpose of readability of the graph, only the synchronous case is depicted in Figure 10. Given the dependence of the Coordinator's control actions on the given activation s, the cost function J in (7) is written

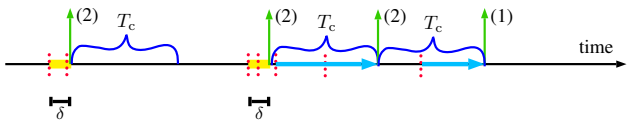

Figure 9. Definition of events causing activation of the Coordinator - asynchronous case. Events are denoted with dotted bars. Vertical arrows are used to indicate when an activation occurs with a label representing how many events are dealt with during each activation. Horizontal arrows show the delays with activation of the Coordinator for individual events because of the minimum reactivation time T_c. Horizontal bars indicate the length of the time window δ used to accumulate events occurring soon after each other.

as $J = J_s$ and is minimized to find an optimal control action of the Coordinator for activation s i.e. $\mathcal{U} = \mathcal{U}_s = \left(\widetilde{Q}_{\mathrm{S,\,demand},s},\quad T_s^{\mathrm{on}},\quad T_s^{\mathrm{off}},\quad \Delta H_s^{\mathrm{ref,\,dynamic}}\right)$, which are now written using the activation counter s in the subscript to explicitly show the dependence of the control actions $\mathcal{U}_s$ on the particular activation of the Coordinator.

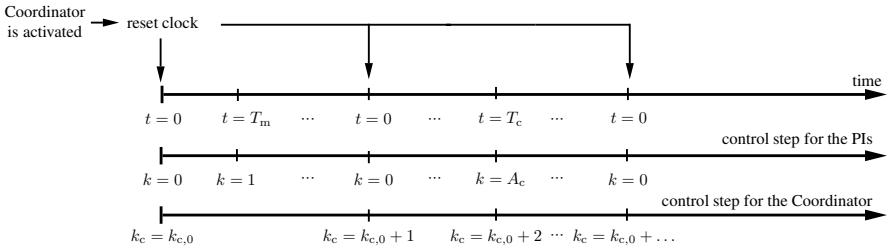

Figure 10. Timing of the Coordinator in the event-driven multiple activation case: there are three activations shown, fewer than there are control steps k_c.

While it is rather straightforward to determine how to calculate the flow from the head gate $Q_{\mathrm{S}}(k)$ given the profile $\widetilde{Q}_{\mathrm{S,\,demand}}(k_c)$ using the rolling-horizon strategy in the time-driven approach, in the event-driven formulation this task may prove to be more complicated as new profiles $\widetilde{Q}_{\mathrm{S,\,demand},s} = (Q_{\mathrm{S,\,demand},s}(0), \ldots, Q_{\mathrm{S,\,demand},s}(N_p - 1))^T$ are necessarily determined at regular intervals. Given the Coordinator's ruling $\widetilde{Q}_{\mathrm{S,\,demand},s}$, the flow from the head gate at step k after the activation is obtained for $i = 0, \ldots, A_c - 1$ and $j = 0, \ldots, N_p - 1$ as

$$Q_{\mathrm{S}}(jA_c + i) = Q_{\mathrm{S,\,base}} + Q_{\mathrm{S,\,demand},s}(j), \tag{10}$$

if $k = jA_c + i < N_p A_c - 1$ and otherwise, if there is no activation of the Coordinator after the activation s as

$$Q_{\mathrm{S}}(k) = Q_{\mathrm{S,\,base}} \tag{11}$$

for $k \geq N_p A_c$. This means that upon the s^{th} activation of the Coordinator, the profile $\widetilde{Q}_{\mathrm{S,\,demand},s_-}$, where s_- denotes all activations before the current activation s, is rendered void and no longer used and the newly found profile $Q_{\mathrm{S,\,demand},s}$ is executed instead.

Just as for to the time-driven formulation, we propose two approaches for the event-driven formulation: the block-modifying approach and the block-adding approach. In short, in the block-modifying approach, once a new block is formed for the activation s, if at the time of the $(s+1)^{\mathrm{th}}$ activation the time is after the time instant $t_{i,s}^{\mathrm{on}}$ but before $t_{i,s}^{\mathrm{off}}$, only the lengths of the existing blocks can change, i.e. the Coordinator can only modify $t_{i,s+1}^{\mathrm{off}}$ but the starting moment of the particular block $t_{i,s+1}^{\mathrm{on}}$ and its height $\Delta h_{i,s+1}^{\mathrm{ref,dynamic}}$ need to remain as ordered for activation s, see Figure 6. This is of course with the exception that the modifications from the previous activations have finished, i.e. the $(s+1)^{\mathrm{th}}$ activation time is after $t_{i,s}^{\mathrm{off}}$, in which case the Coordinator can again determine all parameters $t_{i,s+1}^{\mathrm{on}}$, $t_{i,s+1}^{\mathrm{off}}$ and $\Delta h_{i,s+1}^{\mathrm{ref,dynamic}}$ freely for the given pool i. Conversely, in the block-adding formulation, once a setpoint profile is computed for the s^{th} activation, during the activation $s+1$, the setpoint profiles obtained for the s^{th} activation are used in a way as normal operating levels and they can be modified by adding new blocks to these profiles as depicted in Figure 7. We present the details of the implementation of the block-modifying and the block-adding formulations in Appendix B.

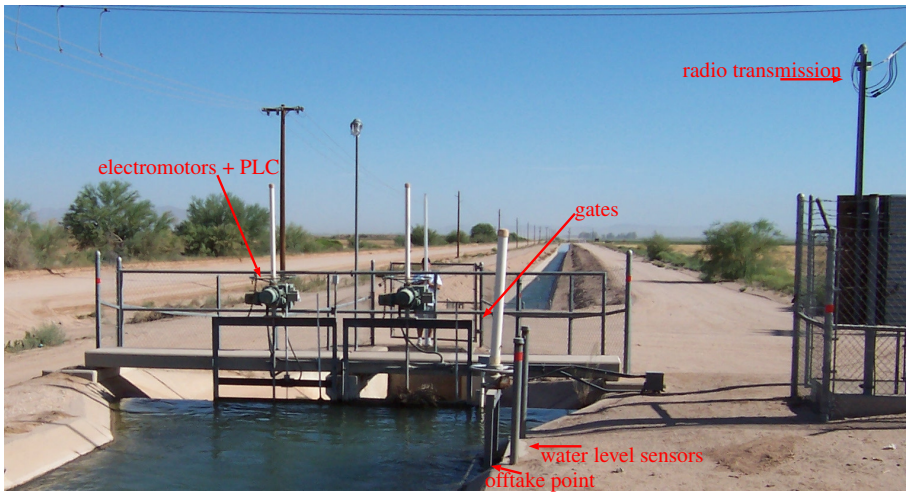

Figure 11. A photo of West-M irrigation canal in the South of Phoenix used in the case study.

D. *Comparative analysis*

In this section we compare the time-driven formulation of the Coordinator with the event-driven formulation. In the case of a single activation of the Coordinator or of multiple but very unfrequent activations, the event-driven formulation does not possess an intrinsic feedback strategy. In fact, it can be viewed as a feedforward scheme that for given current measurements uses the built-in model of the canal to determine the optimal control actions. If the next activation is far apart from the current one, there is no way the Coordinator can modify its actions once set. On the other hand, in extreme cases the Coordinator can of course be re-activated. As the time-driven controller by definition is invoked at regular intervals, it can react to the slightest problems. Therefore, the time-driven formulation provides means to naturally overcome issues like model-plant mismatches. However, this may imply that setpoint changes and thus communication between the control center and the local sites are needed too frequently.

For both the time-driven and the event-driven formulations, two additional sub-formulations are considered. These are the block-modifying approach and the block-adding approach. It should be noted that while better performance can be expected from the block-adding approach due to its more general nature, it is also burdened with a potentially higher computational effort needed to determine all setpoint modifications. Moreover, if too frequent changes of the setpoints of the PI controllers are executed, the performance of the PI controllers may worsen. In the block-modifying approach, there is by definition only one block at a time, which is indeed a benefit in terms of the computations and the frequency of setpoint changes required.

In the next section we further study the similarities and differences of the time-driven and the event-driven approaches by means of a simulation-based case study.

V. Case study

Test Canal 1, see Figure 11, is one of the benchmark canals proposed by the ASCE Task Committee on Canal Automation Algorithms [20, 60]. The benchmark model corresponds to the West-M irrigation canal in the South of Phoenix, Arizona. The canal consists of 7 pools and has a total length of nearly 10 km. The delays $k_{\mathrm{d}i}$ in the pools $i = 1, \ldots, 7$ are respectively: 1, 3, 1, 1, 9, 3, and 5 sampling steps. The surface areas c_i of the pools are: 397, 653, 503, 1530, 1614, 2000, and 1241 m^2. As in [20] we use a model sampling time T_{m} of 4 minutes. Control parameters are selected by first considering the system dynamics and next manual fine tuning. Accordingly, the control interval of the Coordinator is chosen to be $T_{\mathrm{c}} = 20$ minutes. To choose the prediction horizon N_{p} for the case study, we examined the transient behavior. Consequently, the prediction horizon for the time-driven approach is chosen to be $N_{\mathrm{p,t}} = 16$ control steps, which is equivalent to 80 model sampling steps. For the event-driven formulation, a longer prediction horizon of $N_{\mathrm{p,e}} = 24$ control steps, corresponding to 120 model sampling steps, is applied. The difference between the length of the prediction horizon in the time-driven formulation and the

event-driven formulation stems from the fact that while in the time-driven approach the Coordinator is invoked at every control step, in the event driven formulation it is only activated when there is an event. Therefore, in general it may happen that the Coordinator is only activated once in a long period of time, and thus it needs to be able to look into the future far enough to be able to fairly judge how its actions affect the whole system, given the dynamics of the system and the related delays.

The weighting parameters of the cost function J are selected as: $\alpha = 20$, $\beta = 4$, $\gamma_1 = \gamma_2 = 1$, $\mu = 5$, and $\zeta = 3$, while the control gains of the PI controllers for all pools are tuned to be $K_{\mathrm{I}i} = 0.2$ and $K_{\mathrm{P}i} = 1.8$. In addition, the head gate capacity is $Q_{\mathrm{capacity}} = 2.8\ \mathrm{m}^3/\mathrm{s}$, the base flow is $Q_{\mathrm{S,base}} = 1.5\ \mathrm{m}^3/\mathrm{s}$, and $h_i^{\min} = -1.2$ m, $h_i^{\max} = -0.1$ m, and $h_i^{\mathrm{ref,normal}} = -0.6$ m, $i = 1, \ldots, 7$. The parameter ϵ used for post-processing in the time-driven formulation is selected as $\epsilon = 0$. In addition, the notice period is 5 sample steps: this means that each event is announced 5 sample steps before it actually starts. Furthermore, all simulations are done using the MATLAB fmincon solver with the SQP algorithm [61] implemented using 20 random starting points. The time instants t_i^{on} and t_i^{off} to be optimized are continuous variables and they are afterwards rounded to the nearest integers (cf. (3)) resulting in a nonsmooth objective function. Therefore, we use fmincon with a smoothing factor[3]. In particular, the minimum step size for approximating derivatives through finite differences is set to $2T_{\mathrm{m}}$.

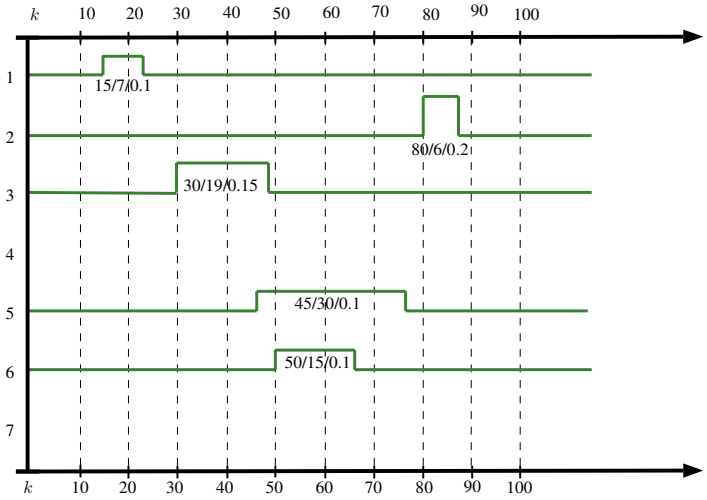

Figure 12. Delivery profile used in the case study. The numeric values $a/b/c$ denote: the step when the delivery should start (a), how long it should last (b), and what is its volume per second (c).

The events in the simulations are associated with delivery requests only. The delivery profile is depicted in Figure 12, where the individual profiles are shown with additional numeric values indicating when each delivery starts, how long it lasts, and what is its volume. It is seen that in pools 4 and 7 there are no deliveries happening, and in each of the remaining pools there is one event during the time of the simulation.

To be able to compare the effectiveness of the various formulations of the controller introduced in this paper, we perform a set of four simulations using the time-driven block-modifying approach, the time-driven block-adding approach, the event-driven block-modifying approach, and the event-driven block-adding approach. For the purpose of comparing the performance of the four approaches, we consider an a posteriori performance index[4]

$$J_{\mathrm{post,\,operation}} = \alpha \sum_{k=1}^{N_{\mathrm{f}}} \left(u_N(k) - Q_{\mathrm{S,base}}\right)^2 + \beta \sum_{i=1}^{N} \sum_{k=1}^{N_{\mathrm{f}}} e_i^2(k), \tag{12}$$

[3]We have tested a number of solvers (Matlab fmincon, Matlab pattern search, Matlab genetic algorithm, Tomlab fmincon) and the selected solver has given the best results in terms of the trade-off between the computation time, constraint satisfaction, and the optimality of the solution.

[4]In this paper we refrain from assessing the CPU times obtained by the different control approaches. This is motivated by the conceptual format of the paper, in which we mainly focus our attention on the introduction of the various formulations of the Coordinator and the evaluation of their performance based on the performance indicators $J_{\mathrm{post,\,operation}}$ and $J_{\mathrm{post,\,switch}}$ to verify which formulation of the Coordinator most closely meet the control performance.

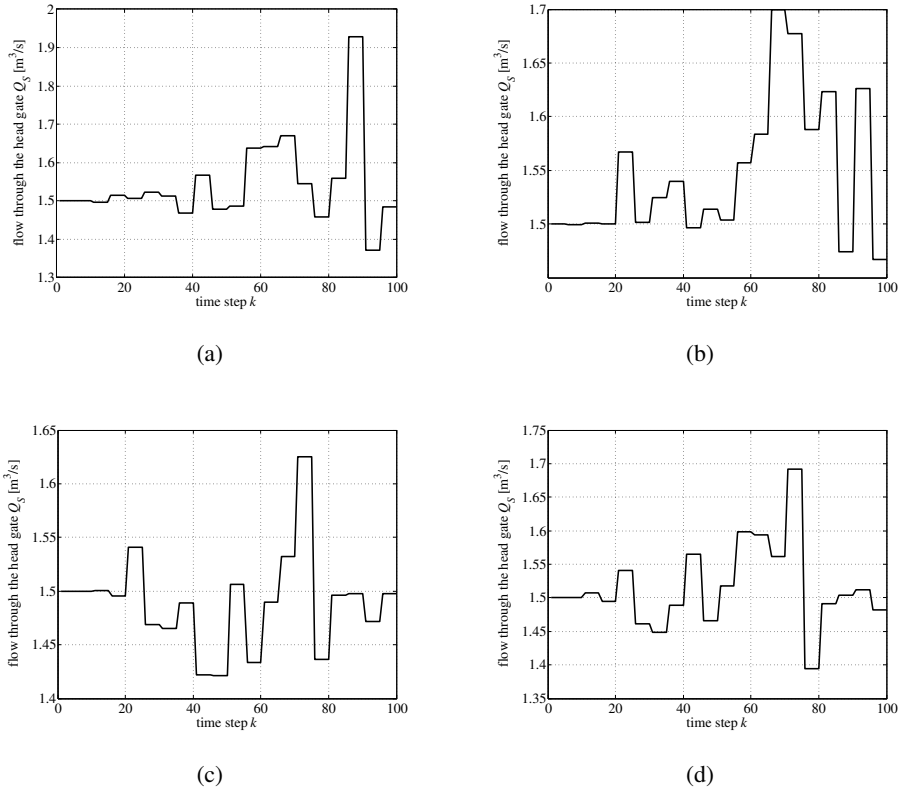

Figure 13. Inflows to the canals from the head gate for: (a) the time-driven block-modifying formulation, (b) the time-driven block-adding formulation, (c) the event-driven block-modifying formulation, (d) the event-driven block-adding formulation.

where $N_{\mathrm{f}} = 100$ steps, which is the total duration of the simulations. We also define an a posteriori performance index $J_{\mathrm{post,\,switch}}$ associated with the number of setpoint changes communicated to the local controllers during the simulation:

$$J_{\mathrm{post,\,switch}} = \sum_{i=1}^{N} \sum_{k=1}^{N_{\mathrm{f}}} \mathbb{1}_{h_i^{\mathrm{ref}}(k) \neq h_i^{\mathrm{ref}}(k-1)}, \tag{13}$$

where the indicator function $\mathbb{1}_A$ is defined as

$$\mathbb{1}_A = \begin{cases} 1 & \text{if } A \text{ is true,} \\ 0 & \text{if } A \text{ is false.} \end{cases} \tag{14}$$

The results obtained in the four simulations are given in Figures 13–15. In Figure 13 we present how the inflow to the canal from the head gate is changed by the Coordinator. Furthermore, the setpoint modifications as well as the actual water levels in Pool 1 (selected as a representative example) are plotted in Figure 14 for the time-driven block-modifying, the time-driven block-adding, the event-driven block-modifying and the event-driven block-adding formulations, respectively. It is not surprising that in the time-driven formulation we obtain much more varying setpoint profiles than in the event-driven formulation, although that could be further modulated by changing the weighting parameters ϵ. Also, as was expected, the block-modifying formulation yields fewer changes than the block-adding formulation in both the time-driven and the event-driven formulation.

Another indication of the performance of the different formulations of the hierarchical controller is the outflow from the canal (in our case this is the outflow from the seventh pool), which should ideally be kept as close as possible to the given base flow. It is observed in Figure 15 that the time-driven approach is able to achieve a tighter control in that regard. In fact, when calculating the indices $J_{\mathrm{post,\,operation}}$ for the four distinctive formulations, see Table I, we again arrive at that same conclusion: the performance yielded by the time-driven formulation is better than that of the event-driven formulations. For the block-adding approach we obtain the value of the a posteriori performance index of $J_{\mathrm{post,\,operation}} = 1.35$ in the time-driven approach, compared with $J_{\mathrm{post,\,operation}} = 3.50$ for the event-driven

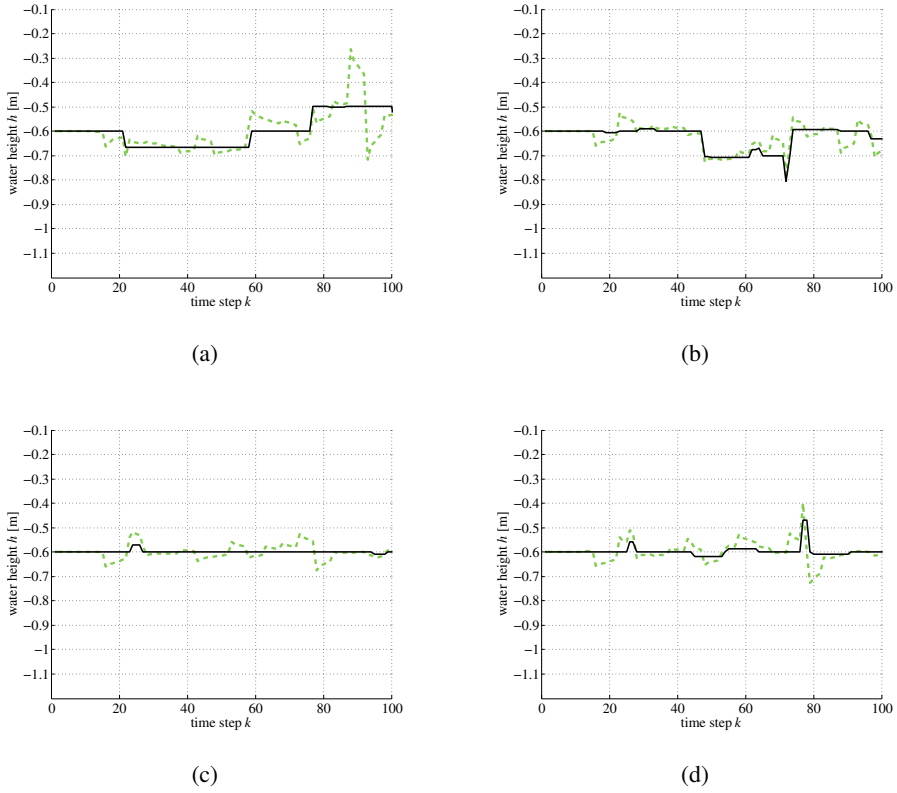

Figure 14. Water levels (dashed line) and setpoint (solid line) for Pool 1 obtained in the simulation of (a) the time-driven block-modifying scheme, (b) the time-driven block-adding scheme, (c) the event-driven block-modifying scheme, and (d) the event-driven block-adding scheme.

Table I
COMPARISON OF THE VALUES OF $J_{\mathrm{post,\,operation}}$ AND $J_{\mathrm{post,\,switch}}$ OBTAINED BY THE DIFFERENT CONTROL APPROACHES.

	$J_{\mathrm{post,\,operation}}$	$J_{\mathrm{post,\,switch}}$
time-driven block-modifying approach	3.22	134
time-driven block-adding approach	1.35	157
event-driven block-modifying approach	5.28	25
event-driven block-adding approach	3.50	60

block. For the block-modifying approach the values of the a posteriori performance indices are $J_{\mathrm{post,\,operation}} = 3.22$ for the time-driven controller and $J_{\mathrm{post,\,operation}} = 5.28$ for the event-driven controller. Comparing the values of $J_{\mathrm{post,\,operation}}$ for the block-modifying formulation and the block-adding formulation, it is observed that a lower value of $J_{\mathrm{post,\,operation}}$ is attained for the block-adding formulation than for the block-modifying formulation: for the time-driven approach and the event-driven approach, respectively, we have $J_{\mathrm{post,\,operation}} = 1.35$ and $J_{\mathrm{post,\,operation}} = 3.22$, versus $J_{\mathrm{post,\,operation}} = 3.50$ and $J_{\mathrm{post,\,operation}} = 5.28$. This is no surprise, as the block-adding formulation provides more freedom in the way setpoint profiles can be changed than the block-modifying formulation, which results in a better performance.

However, as it was previously seen in Figure 14, by examining the indices $J_{\mathrm{post,\,switch}}$ for the four approaches simulated we see that the setpoints are much more frequently changed in the time-driven formulations than in the event-driven formulation. The extreme cases are 157 changes for the time-driven block-adding formulation and only 25 changes for the event-driven block-modifying changes. We can therefore confirm the general claim of the event-driven systems versus the time-driven systems: better performance may be achieved with the time-driven controllers, i.e. the time-driven block-modifying controller outperforms the event-driven block-modifying controller and the time-driven block-adding controller outperforms the event-driven block-adding controller. At the same time,

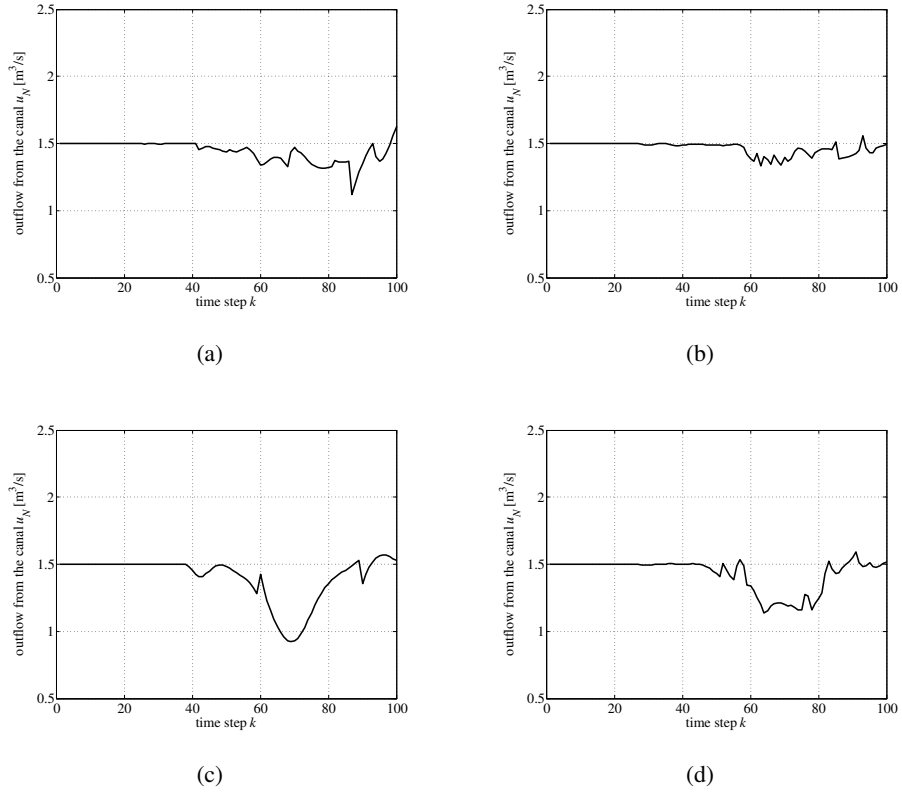

Figure 15. Outflows from the canal: (a) the time-driven block-modifying formulation, (b) the time-driven block-adding formulation, (c) the event-driven block-modifying formulation, (d) the event-driven block-adding formulation.

the block-adding controllers - outperforming the block-modifying controllers - require also more frequent setpoint updates than the block-modifying controllers. In particular, the number of updates for the time-driven controllers and the event-driven controllers, respectively, is: in the block-modifying approach 134 and 25, and in the block-adding approach 157 and 60.

In conclusion, the simulation results show that the event-driven approach is able to perform well: especially the event-driven block-adding controller performed adequately, resulting in indices $J_{\mathrm{post,\,operation}}$ and $J_{\mathrm{post,\,switch}}$ having reasonably low values. With that in mind, it appears plausible to state that given the current technological restrictions present in the field of irrigation, the event-driven controller activated only in face of pre-defined events (e.g. water delivery requests or extreme weather phenomena), is indeed fit for purpose and can be satisfactorily used.[5]

VI. Conclusions

We proposed a hierarchical controller to control an irrigation canal. The controller consists of two control layers. The lower layer is based on the equipment already present in the field and widely used in practice: local PI controllers, used for upstream control. The higher layer - the Coordinator - is a centralized predictive controller, the purpose of which is to control the inflow to the canal as well as to coordinate the PI controllers. The coordination is done by means of modifying the setpoints of the local controllers, which makes the scheme operational in the face of temporarily communication failures as the PI controllers are fit to control the canal autonomously.

We have considered a time-driven formulation and an event-driven formulation. The time-driven formulation is designed to activate the Coordinator at every control step, while the event-driven formulation the Coordinator is only activated in response to events of various kinds (e.g. a delivery request or a heavy rainfall). For both the

[5]It should be remarked that in our analysis we assume that $J_{\mathrm{post,\,operation}}$ and $J_{\mathrm{post,\,switch}}$ are equally important. However, if the weightings of the importance of $J_{\mathrm{post,\,operation}}$ and $J_{\mathrm{post,\,switch}}$ are not equal, different conclusions could potentially be drawn.

time-driven and event-driven formulations we have proposed two ways how setpoints can be changed if there are ongoing changes from previous activations of the Coordinator: the block-modifying and the block-adding strategies.

We have presented a simulation-based case study using a numerical model of a real canal to illustrate the approaches and in particular to compare the performance obtained by the four controllers. It was shown that there is a trade-off between the frequency of setpoint updates and the achieved performance. Nonetheless, given the current broadly present practical restrictions in connection with installed control equipment and unreliable communication in the field of irrigation, the event-driven controller appears to offer a good balance in that respect, and as such it is suitable for deployment in the field.

Further work will include testing the hierarchical controller using a simulation study on different setups and scenarios, and on a real irrigation canal to see how the controller performs in practice. Also the performance obtained with different solvers and for various settings of the tuning parameters of the solvers will be examined in more depth. Moreover, in the current implementation it is assumed that the access to the head gate is continuous. However, the head gate may in fact be away from the control center and it may not be reasonable to assume that a continuous changes to the inflow from the head gate can be made as done in the current formulation. Future work will therefore also include relaxations of the formulation in which the head gate does not need to be continuously accessed. Moreover, studying robust and stochastic controllers to account for uncertainties in the system dynamics seems an appealing problem too.

VII. Acknowledgements

Research supported by the European Union Seventh Framework Programme [FP7/2007-2013] under grant agreement no. 257462 HYCON2 Network of Excellence. The authors thank the anonymous Reviewers for their insightful comments that helped to improve the manuscript.

Appendix: Specifics on the implementation issues

We describe here the particularities regarding implementation of the hierarchical predictive controller introduced in Section IV. In Section A of the appendix we give details on the implementation assuming the time-driven formulation. Afterwards, in Section B we discuss the implementation specifics of the event-driven formulation.

A. Time-driven formulation

1) Block-modifying formulation: As explained in Section IV-B, a setpoint modification is only actually applied (and thus also communicated to the local sites) if it is scheduled to occur within the next T_c time units after the activation of the Coordinator. Assume that at step $k_\mathrm{c} = \hat{k}_{\mathrm{c},i}$ the Coordinator finds an optimal change $t_i^{\mathrm{on}}(k_\mathrm{c}) = t_i^{\mathrm{on}}(\hat{k}_{\mathrm{c},i}) \leq T_\mathrm{c}$ for pool i. This change is indeed executed, since it is scheduled to occur in the current control step. We thus consider a new variable $\hat{t}$ that corresponds to the time variable t but in comparison to t, $\hat{t}$ is not reset after an activation of the Coordinator. Indeed, one can show that the optimal change $t_i^{\mathrm{on}}(k_\mathrm{c})$, which is given with respect to the current activation time of the Coordinator, i.e. the time instant $\hat{k}_{\mathrm{c},i}T_\mathrm{c}$, satisfies $\hat{t}_i^{\mathrm{on}}(k_\mathrm{c}) = \hat{k}_{\mathrm{c},i}T_\mathrm{c} + t_i^{\mathrm{on}}(\hat{k}_{\mathrm{c},i})$, where $\hat{t}_i^{\mathrm{on}}(\hat{k}_{\mathrm{c},i})$ is used to denote the value of the current optimal change $t_i^{\mathrm{on}}(\hat{k}_{\mathrm{c},i})$ with respect to the starting time of the Coordinator $\hat{t} = 0$.

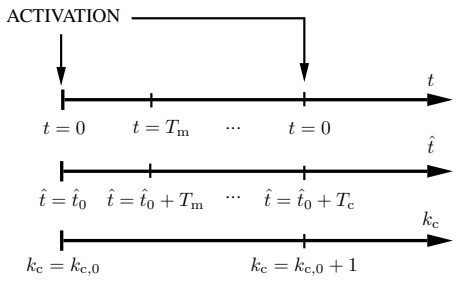

Figure 16. Definition of the variable $\hat{t}$. We use $\hat{t}_0 = k_{\mathrm{c},0}T_\mathrm{c}$ as the starting time instant captured in the graph.

The variable $\hat{t}_i^{\mathrm{on}}(k_{\mathrm{c}})$ is needed to re-write constraint (8c) and constraint (8d). In particular, what is required is that if a setpoint modification is not yet finished, i.e. $t_i^{\mathrm{off}}(k_{\mathrm{c}}-1) < T_{\mathrm{c}}$ does not hold, the current setpoint modification should only be changed at next step by extending or contracting the block. This yields[6]

$$\Delta h_i^{\mathrm{ref,dynamic}}(k_{\mathrm{c}}) = \Delta h_i^{\mathrm{ref,dynamic}}(k_{\mathrm{c}}-1), \tag{A.1a}$$

$$t_i^{\mathrm{off}}(k_{\mathrm{c}}) \geq \max(0, T_{\mathrm{m}} - \Delta t_i^{\mathrm{on}}(k_{\mathrm{c}})), \tag{A.1b}$$

$$t_i^{\mathrm{on}}(k_{\mathrm{c}}) = 0, \tag{A.1c}$$

where $\Delta t_i^{\mathrm{on}}(k_{\mathrm{c}}) = k_{\mathrm{c}}T_{\mathrm{c}} - \hat{t}_i^{\mathrm{on}}(k_{\mathrm{c}})$. In the above, the equality constraints (A.1a) and (A.1c) are used to impose the continuation of the current setpoint modification: its magnitude and the starting time should be as originally assigned. In contrast, the inequality constraint (A.1b) assures that the value of the time instant $t_i^{\mathrm{off}}(k_{\mathrm{c}})$ is only feasible if it is at least one sampling step T_{m} after the time instant starting the modification $t_i^{\mathrm{on}}(k_{\mathrm{c}})$ (cf. (8c)).

Now, two situations need to be considered. The first one is that[7] $t_i^{\mathrm{off}}(k_{\mathrm{c}}-1) < T_{\mathrm{c}}$ or $t_i^{\mathrm{on}}(k_{\mathrm{c}}-1) > T_{\mathrm{c}}$ which means that either the switching time instant t_i^{off} ordered at the previous step $k_{\mathrm{c}}-1$ was contained in the previous control interval and hence a new block can start or that the setpoint modification was not implemented in the previous control step $k_{\mathrm{c}}-1$. This means that for the next setpoint modification, constraints as in (8) apply. The situation is different if $t_i^{\mathrm{off}}(k_{\mathrm{c}}-1) \geq T_{\mathrm{c}}$ and $t_i^{\mathrm{on}}(k_{\mathrm{c}}-1) \leq T_{\mathrm{c}}$ which means that setpoint modification is ongoing. Therefore, the Coordinator can take into consideration the most up to date information available at step k_{c} and modify the value of $t_i^{\mathrm{off}}(k_{\mathrm{c}})$, i.e. the length of the block with the value of the first time instant $t_i^{\mathrm{on}}(k_{\mathrm{c}})$ remaining as chosen in the previous steps. Therefore, constraints (8a) and (8e) are valid but the remaining constraints are made void for pool i satisfying the specific conditions discussed. Instead, constraints (A.1) need to be added.

2) Block-adding formulation: We define a new variable $\alpha_{i,k_{\mathrm{c}}}^{\mathrm{setpoint}}(k)$ to store the changing setpoint profile, and initialize it for $k_{\mathrm{c}} = 0$ as $\alpha_{i,0}^{\mathrm{setpoint}}(k) = h_i^{\mathrm{ref,\,normal}}(k)$ for all $k \in \mathbb{N}$. Then, at every control step k_{c} the cost function $J(k_{\mathrm{c}})$ is minimized to find the optimal control action $\mathcal{U}(k_{\mathrm{c}})$ subject to constraints (8) and

$$h_i^{\mathrm{ref}}(k) = \begin{cases} \alpha_{i,k_{\mathrm{c}}-1}^{\mathrm{setpoint}}(k) & \text{if } k \leq k_i^{\mathrm{on}}(k_{\mathrm{c}}) \text{ or } k \geq k_i^{\mathrm{off}}(k_{\mathrm{c}}), \\ \alpha_{i,k_{\mathrm{c}}-1}^{\mathrm{setpoint}}(k) + \Delta h_i^{\mathrm{ref,\,dynamic}}(k_{\mathrm{c}}) & \text{otherwise,} \end{cases} \tag{A.2}$$

in place of the previously used (2). This is the method the setpoint profiles are predicted by the Coordinator in the optimization routine but it differs when the actual setpoint profiles are to change. As mentioned earlier, the changes are communicated to the local sites only if they are due to occur in the current control step. In the block-adding formulation this means that new blocks are only added to the setpoint profiles if they should start in the current control step, i.e. if $t_i^{\mathrm{on}}(k_{\mathrm{c}}) \leq T_{\mathrm{c}}$. Otherwise, no new blocks are added to the setpoint profile of pool i. Therefore, after the optimization problem we update the setpoint profiles according to

$$\alpha_{i,k_{\mathrm{c}}}^{\mathrm{setpoint}}(k) = \begin{cases} \alpha_{i,k_{\mathrm{c}}-1}^{\mathrm{setpoint}}(k) & \text{if } k \leq k_i^{\mathrm{on}}(k_{\mathrm{c}}) \text{ or } k \geq k_i^{\mathrm{off}}(k_{\mathrm{c}}) \text{ or } k_i^{\mathrm{on}}(k_{\mathrm{c}}) > A_{\mathrm{c}}, \\ \alpha_{i,k_{\mathrm{c}}-1}^{\mathrm{setpoint}}(k) + \Delta h_i^{\mathrm{ref,\,dynamic}}(k_{\mathrm{c}}) & \text{otherwise.} \end{cases} \tag{A.3}$$

Formula (A.3) is the actually executed setpoint profile for pool i, i.e. $h_i^{\mathrm{ref}}(k) = \alpha_{i,k_{\mathrm{c}}}^{\mathrm{setpoint}}(k)$. Note that (A.3) differs from the predicted profile (A.2) in that in (A.3) we assume that a new setpoint block is not implemented if it is scheduled to start after the present control step. However, in both the predicted and actual profiles the times when the blocks are to end are not restricted and once a setpoint profile block satisfies the condition $t_i^{\mathrm{on}}(k_{\mathrm{c}}) \leq T_{\mathrm{c}}$ i.e. it is actually executed, there are no limitations when $t_i^{\mathrm{off}}(k_{\mathrm{c}})$ should take place.

B. Event-driven formulation

1) Block-modifying formulation: We define $k_{\mathrm{activation},s}$ as the sampling step k when the Coordinator is activated for the s^{th} time with respect to the absolute starting moment of Coordinator. Because the step counter k is reset after each activation of the Coordinator, it is not possible to set $k_{\mathrm{activation},s}$ to be simply the value of the sampling

[6]If the introduction of the absolute time variable $\hat{t}$ is not desirable, one could instead at the time of setting up the setpoint change for pool i calculate $\tau_i(k_{\mathrm{c}}) = T_{\mathrm{m}} - k_{\mathrm{c}}T_{\mathrm{c}} + t_i^{\mathrm{on}}(k_{\mathrm{c}})$ and memorize it for future use. Then, at succeeding activations, the variable $\tau_i(k_{\mathrm{c}})$ would be decremented with T_{c} each time k_{c} increments, i.e. $\tau_i(k_{\mathrm{c}}) = \tau_i(k_{\mathrm{c}}-1) - T_{\mathrm{c}}$. Consequently, Constraint (A.1b) could be re-written as $t_i^{\mathrm{off}}(k_{\mathrm{c}}) \geq \max(0, \tau_i(k_{\mathrm{c}}))$.

[7]It is assumed that $t_i^{\mathrm{on}}(0) = t_i^{\mathrm{off}}(0) = 0$, $i = 1, \ldots, N$.

step k at the moment of the s^{th} activation. Instead, we use $k_{\text{activation},s} = k_{\text{c},s}A_{\text{c}} - \Delta_{k,s}$, where $k_{\text{c},s}$ is the value of the control step k_{c} at which the s^{th} activation took place and $\Delta_{k,s}$ is the number of the sampling steps k from the moment of the activation until $k_{\text{c}} = k_{\text{c},s}$. It can be easily seen that in the synchronous case $\Delta_{k,s} = 0$ as the activations can only occur exactly at multiples of the control step of the Coordinator k_{c}.

We consider two situations: one when a setpoint profile block from a previous activation has finished and the other one when a setpoint profile block has started and is ongoing. In the first situation, the basic constraints as listed in (8) need to be satisfied. To verify this occurrence, one might check if[8] $k^{\text{off}}_{i,s-1} + k_{\text{activation},s-1} \leq k_{\text{activation},s}$ for each pool separately. The condition $k^{\text{off}}_{i,s-1} + k_{\text{activation},s-1} \leq k_{\text{activation},s}$ is relevant because given the particular activation counter s for new events, the condition checks whether a previously assigned change has already finished.

If the condition $k^{\text{off}}_{i,s-1} + k_{\text{activation},s-1} \leq k_{\text{activation},s}$ does not hold for some of the pools $i \in \{1, \ldots, N\}$, for those pools that $k^{\text{off}}_{i,s-1} + k_{\text{activation},s-1} > k_{\text{activation},s}$ the Coordinator may no longer freely find new block-shaped setpoint changes, but needs to continue with ongoing changes and, if needed, modify the ongoing changes by either extending or contracting the lengths of the blocks. This means that the following constraints need to be satisfied:

$$\Delta h^{\text{ref,dynamic}}_{i,s} = \Delta h^{\text{ref,dynamic}}_{i,s-1}, \tag{A.4a}$$

$$t^{\text{off}}_{i,s} \geq \max(0, T_{\text{m}} - \Delta t^{\text{on}}_{i,s}), \tag{A.4b}$$

$$t^{\text{on}}_{i,s} = 0, \tag{A.4c}$$

where, similarly to the time-driven formulation, we use $\Delta t^{\text{on}}_{i,s} = (k_{\text{activation},s} - k_{\text{activation},s-1})T_{\text{m}} - t^{\text{on}}_{i,s-1}$ to denote the difference between the new activation time of the Coordinator and the time instant of the first change of the ongoing block $t^{\text{on}}_{i,s-1}$. This way, the constraint that the setpoint blocks need to have at least length of T_{m} is enforced, see (A.4b). In addition, (A.4a) is used to guarantee that for an ongoing setpoint block, the modified value of the setpoint, i.e. the height of the setpoint block, should remain as originally found for that particular block. Lastly, the constraint in (A.4c) is to make sure that at the time of the activation of the Coordinator for the s^{th} time, the setpoint starts from the modified value.

2) Block-adding formulation: In the block-adding formulation of the Coordinator as depicted in Figure 7, we define auxiliary variables $\alpha^{\text{setpoint}}_{i,s}(k)$ for $i = 1, \ldots, N$ to retain information about previous modifications of the setpoint profile. Correspondingly to the time-driven block-adding formulation, see Section A2, we initialize $\alpha^{\text{setpoint}}_{i,0}(k) = h^{\text{ref, normal}}_{i}(k)$ so that the variables $\alpha^{\text{setpoint}}_{i,0}(k)$, $i = 1, \ldots, N$, are identical to the normal levels of the setpoints. The variables $\alpha^{\text{setpoint}}_{i,s}(k)$ for $i = 1, \ldots, N$ are used store information about previous modifications of the setpoint to ultimately be able to add more blocks to the profiles when they are ordered by the Coordinator.

We start the procedure when activation s of the Coordinator occurs, which forces the Coordinator to provide the control action $\mathcal{U}_s$ subject to constraints (8) and

$$h^{\text{ref}}_{i}(k) = \begin{cases} \alpha^{\text{setpoint}}_{i,s-1}(k) & \text{if } k \leq k^{\text{on}}_{i,s} \text{ or } k \geq k^{\text{off}}_{i,s}, \\ \alpha^{\text{setpoint}}_{i,s-1}(k) + \Delta h^{\text{ref, dynamic}}_{i,s} & \text{otherwise.} \end{cases} \tag{A.5}$$

After finding $\mathcal{U}_s$, we set $\alpha^{\text{setpoint}}_{i,s}(k) = h^{\text{ref}}_{i}(k)$ to save the newly obtained setpoint profile $h^{\text{ref}}_{i}(k)$ for use in future activations of the Coordinator. In other words, by assigning $\alpha^{\text{setpoint}}_{i,s}$ during succeeding activations, new blocks can be added on top of the existing profiles, taking $\alpha^{\text{setpoint}}_{i,s-1}(k)$ as the normal time-varying setpoint level.

REFERENCES

[1] D. L. Bjorneberg, R. E. Sojka, and J. A. Entry, "Irrigation: Historical perspective," in *Encyclopedia of Soil Science, Second Edition*, 2002, pp. 945–949.

[2] S. Siebert and P. Döll, "Irrigation water use. a global perspective." in *Global Change: Enough Water for All?*, J. L. Lozán, H. Graß l, P. Hupfer, and L. Menzel, Eds., 2007, pp. 104–107.

[3] M. Cantoni, E. Weyer, Y. Li, S. K. Ooi, I. Mareels, and M. Ryan, "Control of large-scale irrigation networks," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 75–91, 2007.

[4] G. Bastin, A. Bayen, C. D'Apice, X. Litrico, and B. Piccoli, "Open problems and research perspectives for irrigation channels," *Networks and Heterogeneous Media*, vol. 4, no. 2, pp. i–v, 2009.

[5] P. Malaterre, D. Rogers, and J. Schuurmans, "Classification of canal control algorithms," *Journal of Irrigation and Drainage Engineering*, vol. 124, no. 1, pp. 3–10, 1998.

[8] We set $k^{\text{on}}_{i,0} = k^{\text{off}}_{i,0} = 0$, $i = 1, \ldots, N$.

[6] X. Litrico, V. Fromion, J.-P. Baume, and M. Rijo, "Modelling and PI controller design for an irrigation canal," in *Proceedings of the 2003 European Control Conference*, Cambridge, UK, 2003.

[7] X. Litrico, P.-O. Malaterre, J.-P. Baume, P. Vion, and J. Ribot-Bruno, "Automatic tuning of PI controllers for an irrigation canal pool," *Journal of Irrigation and Drainage Engineering-ASCE*, vol. 133, pp. 27–37, 2007.

[8] P.-J. van Overloop, J. Schuurmans, R. Brouwer, and C. Burt, "Multiple-model optimization of proportional integral controllers on canals," *Journal of Irrigation and Drainage Engineering-ASCE*, vol. 131, no. 2, pp. 190–196, 2005.

[9] J. Schuurmans, "Control of water levels in open-channels," Ph.D. dissertation, Delft University of Technology, The Netherlands, 1997.

[10] P.-O. Malaterre and J.-P. Baume, "Modeling and regulation of irrigation canals: existing applications and ongoing researches," in *Proceedings of the 1998 IEEE International Conference on Systems, Man, and Cybernetics*, vol. 4, San Diego, CA, 1998, pp. 3850–3855.

[11] S. K. Ooi and E. Weyer, "Control design for an irrigation channel from physical data," *Control Engineering Practice*, vol. 16, no. 9, pp. 1132–1150, 2008.

[12] M. Papageorgiou and A. Messmer, "Flow control of a long river stretch," *Automatica*, vol. 25, no. 2, pp. 177–183, 1989.

[13] J. de Halleux, C. Prieur, J.-M. Coron, B. d'Andréa Novel, and G. Bastin, "Boundary feedback control in networks of open channels," *Automatica*, vol. 39, no. 8, pp. 1365–1376, 2003.

[14] L. Zaccarian, Y. Li, E. Weyer, M. Cantoni, and A. Teel, "Anti-windup for marginally stable plants applied to open water channels," in *Proceedings of the 2004 Asian Control Conference*, vol. 3, July 2004, pp. 1692–1700.

[15] T. Rabbani, S. Munier, D. Dorchies, P.-O. Malaterre, A. Bayen, and X. Litrico, "Flatness-based control of open-channel flow in an irrigation canal using SCADA," *IEEE Control Systems Magazine*, vol. 29, no. 5, pp. 22–30, 2009.

[16] M. Fliess, J. Lévine, P. Martin, and P. Rouchon, "Flatness and defect of non-linear systems: introductory theory and examples," *International Journal of Control*, vol. 61, no. 6, pp. 1327–1361, 1995.

[17] E. Bautista and A. Clemmens, "Volume compensation method for routing irrigation canal demand changes," *Journal of Irrigation and Drainage Engineering*, vol. 131, no. 6, pp. 494–503, 2005.

[18] M. Xu, R. Negenborn, P. van Overloop, and N. van de Giesen, "De Saint-Venant equations-based model predictive control of open channel flow," *Advances in Water Resources*, vol. 37-45, pp. 37–45, 2012.

[19] E. Weyer, "Control of irrigation channels," *IEEE Transactions on Control Systems Technology*, vol. 16, no. 4, pp. 664–675, 2008.

[20] R. R. Negenborn, P. J. van Overloop, T. Keviczky, and B. De Schutter, "Distributed model predictive control for irrigation canals," *Networks and Heterogeneous Media*, vol. 4, no. 2, pp. 359–380, 2009.

[21] V. Rutz, C. Ruiz, and L. Ramirez, "Predictive control in irrigation canal operation," in *Proceedings of the 1998 IEEE International Conference on Systems, Man, and Cybernetics*, vol. 4, San Diego, CA, 1998, pp. 3897–3901.

[22] P. Silva, M. A. Botto, J. Figueiredo, and M. Rijo, "Model predictive control of an experimental water canal," in *Proceedings of the 2007 European Control Conference*, Kos, Greece, 2007, pp. 2977–2984.

[23] P. Charbonnaud, F. Carrillo, and E. Duviella, "A supervised robust predictive multi-controller for large operating conditions of an open-channel system," in *Proceedings of the 18th IFAC World Congress*, Milan, Italy, 2011, pp. 4620–4625.

[24] P.-J. van Overloop, A. Clemmens, R. Strand, and R. Wagemaker, "Real-time implementation of model predictive control on MSIDD's WM canal," *Journal of Irrigation and Drainage Engineering-ASCE*, vol. 136, no. 11, pp. 747–756, 2010.

[25] J. M. Lemos, F. Machado, N. Nogueira, L. Rato, and M. Rijo, "Adaptive and non-adaptive model predictive control of an irrigation channel," *Networks and Heterogenous Media*, vol. 4, no. 2, pp. 303–324, 2009.

[26] J. Igreja, J. Lemos, F. Cadete, L. Rato, and M. Rijo, "Control of a water delivery canal with cooperative distributed mpc," in *Proceedings of the 2012 American Control Conference*, June 2012, pp. 3346–3351.

[27] A. Álvarez, M. Ridao, D. Ramirez, and L. Sánchez, "Constrained predictive control of an irrigation canal," *Journal of Irrigation and Drainage Engineering*, 2013, accepted.

[28] H. Fawal, D. Georges, and G. Bornard, "Optimal control of complex irrigation systems via decomposition-coordination and the use of augmented lagrangian," in *Proceedings of the 1998 IEEE International Conference on Systems, Man, and Cybernetics*, vol. 4, Oct 1998, pp. 3874–3879.

[29] D. D. Šiljak, *Decentralized Control of Complex Systems*. Academic Press, Boston, Massachusetts, 1991.

[30] Y. Li and B. De Schutter, "Control of a string of identical pools using non-identical feedback controllers," *IEEE Transactions on Control Systems Technology*, vol. 20, no. 6, pp. 1638–1646, 2012.

[31] Y. Li and M. Cantoni, "Distributed controller design for open water channels," in *Proceedings of the 17th IFAC World Congress*, Seoul, Korea, 2008, pp. 10 033–10 038.

[32] P.-O. Malaterre and J.-P. Baume, "Optimum choice of control action variables and linked algorithms. comparison of different alternatives," in *Proceedings of the Workshop on Modernization of Irrigation Water Delivery Systems*, Phoenix, AZ, 1999, pp. 387–406.

[33] X. Litrico and V. Fromion, "H_∞ control of an irrigation canal pool with a mixed control politics," *IEEE Transactions on Control Systems Technology*, vol. 14, no. 1, pp. 99–111, Jan 2006.

[34] J. Scarlett and R. Brennan, "Re-evaluating event-triggered and time-triggered systems," in *Proceedings of the 2006 IEEE Conference on Emerging Technologies and Factory Automation*, Prague, Czech Republic, 2006, pp. 655–661.

[35] W. P. M. H. Heemels, J. H. Sandee, and P. P. J. van den Bosch, "Analysis of event-driven controllers for linear systems," *International Journal of Control*, vol. 81, no. 4, pp. 571–590, 2008.

[36] R. Obermaisser, *Event-Triggered and Time-Triggered Control Paradigms*. Santa Clara, CA, USA: Springer-Verlag, 2004.

[37] R. Scattolini, "Architectures for distributed and hierarchical model predictive control – a review," *Journal of Process Control*, vol. 19, no. 5, pp. 723–731, 2009.

[38] B. Picasso, D. D. Vito, R. Scattolini, and P. Colaneri, "An {MPC} approach to the design of two-layer hierarchical control systems," *Automatica*, vol. 46, no. 5, pp. 823–831, 2010.

[39] J. Nabais, R. Negenborn, R. Carmona-Benétez, L. Mendonça, and M. Botto, "Hierarchical MPC for multiple commodity transportation networks," in *Distributed Model Predictive Control Made Easy*, J. M. Maestre and R. R. Negenborn, Eds. Springer Netherlands, 2014, vol. 69, pp. 535–552.

[40] G. Chaloulos, P. Hokayem, and J. Lygeros, "Distributed hierarchical MPC for conflict resolution in air traffic control," in *Proceedings of the 2010 American Control Conference*, June 2010, pp. 3945–3950.

[41] A. Núñez, C. Ocampo-Martinez, B. De Schutter, F. Valencia, J. López, and J. Espinosa, "A multiobjective-based switching topology for hierarchical model predictive control applied to a hydro-power valley," in *Proceedings of the 3rd IFAC International Conference on Intelligent Control and Automation Science (ICONS 2013)*, Chengdu, China, Sep. 2013, pp. 529–534.

[42] M. Brdys, M. Grochowski, T. Gminski, K. Konarczak, and M. Drewa, "Hierarchical predictive control of integrated wastewater treatment systems," *Control Engineering Practice*, vol. 16, no. 6, pp. 751–767, 2008.

[43] R. R. Negenborn, S. Leirens, B. De Schutter, and J. Hellendoorn, "Supervisory nonlinear MPC for emergency voltage control using pattern search," *Control Engineering Practice*, vol. 17, no. 7, pp. 841–848, July 2009.

[44] A. Zafra-Cabeza, J. M. Maestre, M. A. Ridao, E. F. Camacho, and L. Sanchez, "Hierarchical distributed model predictive control for risk mitigation: An irrigation canal case study," in *Proceedings of the 2011 American Control Conference*, San Francisco, CA, 2011, pp. 3172–3177.

[45] C. Ocampo-Martinez, D. Barcelli, V. Puig, and A. Bemporad, "Hierarchical and decentralised model predictive control of drinking water networks: Application to barcelona case study," *IET Control Theory Applications*, vol. 6, no. 1, pp. 62–71, January 2012.

[46] V. T. Chow, *Open-Channel Hydraulics*, ser. McGraw-Hill Civil Engineering. London: McGraw-Hill, 1959.

[47] P.-J. van Overloop, "Model predictive control on open water systems," Ph.D. dissertation, Delft University of Technology, The Netherlands, 2006.

[48] J.-P. Baume, J. Sau, and P.-O. Malaterre, "Modelling of irrigation channel dynamics for controller design," in *Proceedings of the 1998 IEEE International Conference on Systems, Man, and Cybernetics*, vol. 4, 1998, pp. 3856–3861.

[49] J. Schuurmans, A. Clemmens, S. Dijkstra, A. Hof, and R. Brouwer, "Modeling of irrigation and drainage canals for controller design," *Journal of Irrigation and Drainage Engineering-ASCE*, vol. 125, no. 6, pp. 338–344, 1999.

[50] J. M. Maciejowski, *Predictive Control with Constraints*. Essex, England: Prentice Hall, 2002.

[51] E. Camacho and C. Bordons, *Model Predictive Control*. Berlin Heidelberg: Springer, 1999.

[52] B. De Schutter and B. De Moor, "Optimal traffic light control for a single intersection," *European Journal of Control*, vol. 4, no. 3, pp. 260–276, 1998.

[53] H. van Ekeren, R. R. Negenborn, P. J. van Overloop, and B. De Schutter, "Hybrid model predictive control using time-instant optimization for the Rhine-Meuse delta," in *Proceedings of the 2011 IEEE International Conference on Networking, Sensing and Control*, Barcelona, Spain, 2011, pp. 216–221.

[54] S. Cristea, C. de Prada, D. Sarabia, and G. Gutiérrez, "Aeration control of a wastewater treatment plant using hybrid NMPC," *Computers & Chemical Engineering*, vol. 35, no. 4, pp. 638–650, 2011.

[55] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, 1st ed. New York, NY, USA: W. H. Freeman & Co., 1979.

[56] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[57] M. Mitchell, *An Introduction to Genetic Algorithms*. Cambridge, MA, USA: MIT Press, 1996.

[58] V. Torczon, "On the convergence of pattern search algorithms," *SIAM Journal on Optimization*, vol. 7, no. 1, pp. 1–25, 1997.

[59] C. Floudas, *Nonlinear and Mixed-Integer Optimization*. Oxford, UK: Oxford University Press, 1995.

[60] A. Clemmens, T. Kacerek, B. Grawitz, and W. Schuurmans, "Test cases for canal control algorithms," *Journal of Irrigation and Drainage Engineering*, vol. 124, no. 1, pp. 23–30, 1998.

[61] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. New York: Springer, 2006.