Intelligent Automation And Soft Computing, 2020
Vol.26,no.4,663–677
DOI: 10.32604/iasc.2020.010101

Systematic Procedure for Optimal Controller Implementation Using Metaheuristic Algorithms

Viorel Minzu and Adrian Serbencu

Control and Electrical Engineering Department, "Dunarea de Jos" University of Galati, Romania

ABSTRACT
The idea for this work starts from the situation in which a metaheuristic-based algorithm has already been developed in order to solve an optimal control problem. This algorithm yields an offline "optimal" solution. On the other hand, the Receding Horizon Control (RHC) structure can be implemented if a process model is available. This work underlines some of the practical aspects of joining the RHC to an existing metaheuristic-based algorithm in order to obtain a closed-loop control structure that can be further used in real-time control. The result is a systematic procedure that integrates a given metaheuristic-based algorithm into a RHC structure.

KEY WORDS: Metaheuristic-based algorithm, optimal control, Receding Horizon Control, process model.

1 INTRODUCTION

A very large number of papers describe control systems that use artificial intelligence techniques for all the aspects that involve nonlinearities, or imprecise, incomplete, and uncertain knowledge. We recall here only a few topics; Model's Parameters Determination - Altinten (2007), System Identification - Makas and Yumusak, (2016), Optimization of Control Parameters - Kesarkar and Selvaganesan (2015); Aras et al. (2011); Jabri et al. (2011); Zhang et al. (2018); Optimal Sensors' Placing - Bruant et al. (2011); and Singh and Hahn (2006), Optimal Control Problems - Bououden et al. (2015); Naghizadeh, et al. (2016); Hu, X.B. et al. (2004); Faber et al. (2005); Qian et al. (2012); and Wong et al. (2016).

Solving an optimal control problem (OCP) means designing a control structure and implementing it as a closed-loop structure. This framework is the only one that takes into consideration real-time information. One of the control structures used in many applications is the Receding Horizon Control (RHC). This one uses a process model (MP) and organizes the moving of the prediction horizon for example; Hu et al. (2005a) and Attia et al. (2006). A particular case of the RHC is the well-known Model Predictive Control (MPC) that makes at each step a specific action: *The minimization of the prediction errors*. There are many works dealing with the MPC from different points of view such as; Theoretical works - Clarke (1994); Hiskens and Gong (2006); Zheng (2010), tutorial reviews - Christofides et al. (2013), or surveys of industrial applications - Qin and Badgwell (2003), Yang et al. (2014), Lopez Francol (2018).

Many papers have considered metaheuristics (e.g., Genetic Algorithm, Simulated Annealing, and Particle Swarm Optimization etc.) in conjunction with the RHC in order to implement closed-loop structures, which have been used afterwards in real-time control. The book by Jayaraman and Siarry (2014) has a section that shows a survey of this kind of work. The paper by Goggos and King (1996) introduces a new technique called the Evolutionary Predictive Control for the design of predictive controllers. The technique uses Evolutionary Algorithms to generate and evaluate a family of optimum predictive controllers having different design parameters at every sampling instant via an adaptive process from which the best performer is selected.

Many works deal with the adequation of GA to model-based predictive control. The emphasis is placed on the GA operators' definition. The paper by Sarimveis and Bafas (2003) proposes a specialized genetic algorithm optimization method for fuzzy predictive control based on the Takagi–Sugeno model. A new approach given in Causa et al. (2008) includes a prediction based on a hybrid fuzzy model of the process. The work done by Venkateswarlu and Reddy (2008) proposes stochastic optimization algorithms such as the GA and simulated annealing that are

CONTACT Viorel Minzu viorel_minzu@yahoo.fr, viorel.minzu@ugal.ro

combined with a polynomial-type process model to develop nonlinear model predictive control strategies.

An interesting application of the RHC is related to the flood control by Blanco et al. (2010). With the same topic, the paper by Chiang and Willems (2015) associates evolutionary optimization with the RHC, in order to obtain a real-time flood control system. This work is essentially similar to that presented in Section 4, which is the second case study of our paper.

We make the following observations that provide a better description of the authors' perspective with regards to this work's topic:

1. The statement of an OCP contains necessary elements; the process model (PM), functioning constraints, and objective function that must be minimized or maximized, initial conditions, and time horizon.
2. A stand-alone MbA, which is neither included in nor includes a closed-loop control structure can solve the OCP and produce an offline "optimal" solution, which is a sequence of values for the control variables covering the control horizon. This control sequence may eventually be used in real-time but only within an open-loop control structure, i.e., the control values a priori that are calculated are sent to the real process at precise time moments, without taking into consideration the real-time information. This is a possible but unusual situation, because unmodeled dynamics, perturbations, or noises can affect the real process and significantly change the performance index's value.
3. Obviously, a closed-loop (feedback) control structure is not equivalent to a real-time control structure. For example, the former may be used in simulations (like in this work) where there is no real-time information (only eventually a simulated one), and the simulated time is not synchronized with a real time clock.

This paper has a rather practical relevance, because it is addressed in the first place to a practicing professional engineer who wants to implement an optimal control structure using an MbA. The MbA solving the specific optimal control problem is already developed and tested off-line. The implementation of the control structure has to join knowledge from soft computing and control engineering that is not always a simple task. Our article is addressed to the researcher that has developed a new metaheuristic algorithm aiming to use it within an optimal control structure.

The objective of this work is to emphasize some practical aspects of how to join the RHC to an MbA in order to obtain a closed-loop control structure that can be further used in real-time control as a solution to the given OCP. *These practical aspects are taken into account in the systematic procedure proposed in this paper, which produces the desired control structure starting from the statement of the given OCP*.

When joining the RHC structure with an MbA, it is important to establish their relative "position". For example, in many applications the MbA is the genetic algorithm (GA). Some authors consider implicitly that the RHC structure is subordinated to the GA. On the other hand, others use the RHC structure that includes a GA. There are very good papers such as Hu and Chen (2005a); Hu and Chen (2005b) that have mentioned explicitly the relation between the two components and justified it in the context of the solved problem. Of course, when the designer of the control structure specifies the actions of his algorithm, the two variants are feasible, because it is always possible to rearrange the algorithm's actions. The solution proposed in this paper comes from the confirmed facts that an MbA has the vocation of optimizing an objective function, while the RHC explicitly provides the feedback (see observations 2 and 3). That is why, in the context of our procedure, the RHC integrates the MbA into its controller. Therefore, the general framework with feedback is created by the RHC structure whose controller repeatedly calls the MbA.

Section 2 of the paper is devoted to the presentation of the RHC structure and how the MbA can be integrated into its controller. This description leads to a systematic procedure that begins with the design and implementation of a stand-alone algorithm based on the chosen metaheuristic able to solve the given OCP. The stand-alone MbA will be slightly modified to generate an algorithm that can be integrated into the closed-loop structure, as described in Sections 2.1. and 2.2, which deals with some aspects concerning the implementation of the controller related to the influence of the performance index and time constraints over the prediction horizon.

In Sections 3 and 4, the proposed procedure is illustrated by two case studies of the OCPs that use two different metaheuristics. From the beginning, let us note that the emphasis is not on the importance or difficulty of the OCP, or the metaheuristic chosen to solve the problem or other tools that are concerns of the computational intelligence that can improve the metaheuristic's efficiency. The two case studies are just nontrivial exemplifications of how to implement the Receding Horizon Controller following the proposed procedure. Section 3 refers to an OCP with a nonlinear continuous model and Section 4 describes an OCP concerning a nonlinear discrete system.

2 OPTIMAL CONTROL STRUCTURE USING METAHEURISTIC-BASED ALGORITHMS

THE perspective of this paper starts from the difference between the solution of an OCP and a closed-loop control structure that implements this solution. We speak here about an optimization problem (OP) referring to a dynamic environment where the values of the decision variables (control inputs) have to optimize (maximize or minimize) an objective function and change periodically in order to adapt to environmental changes. A process model taking place in the dynamic environment can calculate for each moment the values of some dependent variables as a result of prior variation of the decision variables. The environmental changes are expressed by the values of the so-called state variables and controlled variables.

As the theoretical framework of the RHC is already well-known, only some elements sufficient to join the RHC with the metaheuristic-based algorithm will be reviewed in this section. Obviously, all the variables and functions may have a vectorial character. Complete mathematical details, such as vector dimensions, will be avoided to simplify the presentation.

2.1 Optimal Controller Implementation using Metaheuristic-based Algorithms

A given OCP considers the environment evolution on a control horizon $[t_0, t_H]$, with discrete moments $t_i = t_0 + i \cdot T,\ i = 0, \ldots H$, where T is the sampling period and t_0 is the initial moment. If the value $X(t_0)$ of the initial state and the sequence of control inputs $U(t_0)$, $U(t_1)$, …, $U(t_{H-1})$ are known, then the sequence of the state variables $X(t_1), \ldots, X(t_{H-1}), X(t_H)$ and the sequence of controlled (output) variables $Y(t_1), \ldots, Y(t_{H-1}), Y(t_H)$ can be calculated using a PM. In this work, we consider that the PM is a set of differential algebraic equations.

Let Π be the structure of the OCP defined as

$$\Pi = \langle f, \text{constraints}, t_0, H, X(t_0), J(t_0, H, X(t_0)) \rangle, \qquad (1)$$

where f is the function appearing in the state equation

$$\frac{dX}{dt} = f(X(t), U(t)), \qquad (2)$$

$J(t_0, H, X(t_0))$ is the objective function, and "constraints" is the set of all algebraic and differential constraints imposed by the dynamic environment. To solve Π means finding the control sequence that optimizes (maximizes or minimizes) the objective function $J(.)$ on the control horizon, starting from the initial state of $X(t_0)$.

For different reasons, especially when a deterministic algorithm is not known, we may decide to solve this problem using an approaching algorithm based on a metaheuristic, such as the Genetic Algorithm, Particle Swarm Optimization, Ant Colony Systems, Simulated Annealing, etc. The main reason is the ability of such algorithms to cope with the high complexity of Π.

For many control applications, a realistic way of solving Π is to accomplish the procedure in two phases.

(1) In the first phase, an MbA is designed for solving Π. Let us denote $\boldsymbol{A}(t_0, H, X(t_0))$ as an algorithm that finds a control sequence that optimizes the objective function $J(t_0, H, X(t_0))$ on the control horizon $[t_0, t_H]$ starting from the initial state $X(t_0)$; in another words, it solves Π. Obviously, this algorithm involves implicitly function f and the "constraints", which are specific to problem Π. $\boldsymbol{A}$ can also take into consideration elements that are not mentioned above, such as the parameters of the process, disturbances, uncertainties, estimated values for input variables other than control inputs, etc. In addition, the computational results of algorithm $\boldsymbol{A}$ must attest to its efficiency, good convergence, and acceptable computational complexity.

This phase is equivalent to an off-line solution of Π, which cannot be applied, because it will generate an open-loop control system that is extremely sensitive to the PM quality and disturbances. Hence, a closed-loop solution that can be further used in real-time control is compulsory. This will consider, as initial state values for the computation of predicted control sequence, only the actual values collected from the environment or estimated on the basis of real controlled variables.

(2) The second phase yields a controller integrated in a closed-loop structure, which implements the solution of Π. The solution proposed in this paper is to consider the RHC structure recalled in Figure 1.

We express the evolution of the dynamic environment by the state variables of $X(.)$. The state variables are either directly measured or estimated using variables $U(.)$ and $Y(.)$. As a general rule, the controller is tasked with minimizing the prediction error through the control input sequence along the prediction horizon.

In the proposed systematic approach, the RHC structure has a few particularities:

- The Receding Horizon Controller integrates a slightly modified version of algorithm $\boldsymbol{A}$, denoted as $\boldsymbol{A}_{RH}$, because the main adjustment is the replacement of the control horizon by the *prediction horizon* (also called the *receding horizon*) through the implementation of the Receding Horizon mechanism.
- The reference signals are related to the optimal state trajectory of the dynamic environment.
- The prediction error is a measure of the difference between the predicted and optimal state trajectories.

Since the second phase of the described procedure is compulsory, one may wonder if the first phase is really necessary.

Remark 1: It is necessary to design and implement the MbA algorithm $\boldsymbol{A}$ as an offline solution of Π since it is an effective way to verify whether it has the following features:

- it is appropriate for solving the given OCP (good solutions are obtained);
- it has good convergence speed when solving Π, a fact that must be carefully proven and tested;

- its computational complexity is acceptable and can justify the possibility of using algorithm ***A*** for a RHC structure.

In the sequel, let us consider t_0=0 and the discrete moments t_k=$k\cdot T$ will be specified simply by k. The control horizon is the interval [0, H]. We recall the following notations:

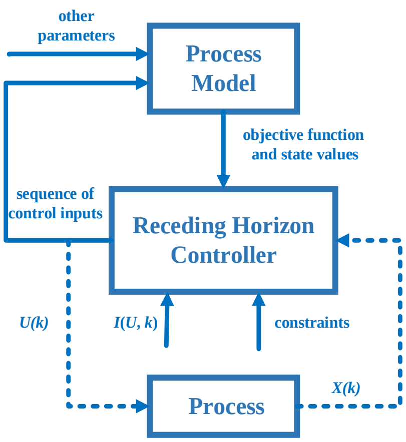

Figure 1. **Implementation of the Receding Horizon Control.**

- [k, k + h] is the prediction horizon, with h < H, k =0, 1,... H-h;
- $U(k+i|k)$, i=0... h-1 is the predicted value for $U(k+i)$ based on knowledge up to moment k;
- $X(k+i|k)$, i=1,..., h is the predicted value for $X(k+i)$ based on knowledge up to moment k;

Note that

$$U(k+i|k) \neq U(k+i),\ k>0,\ i=1,...,h-1, \qquad (3)$$

The same thing can be asserted for the state variables.

Figure 1 suggests how the RHC involves the predictive control technique. This means that at the present moment of k when the state variable is $X(k)$, the performance index J for the interval [k, k + h] is optimized subject to constraints through an *optimal control sequence*

$$ocs = \langle U^*(k|k), \ldots, U^*(k+h-1|k)\rangle \qquad (4)$$

The first element of $U(k)$=$U^*(k|k)$ of this sequence is applied to the system. Then the horizon is shifted by one sample and the optimization is restarted for interval [k+1, k+h+1]. The optimization is made within the Receding Horizon Controller.

The fact that the Receding Horizon Controller integrates $\boldsymbol{A}_{\mathrm{RH}}$ is the main point of joining the RHC with an MbA. But it also includes other parts with tasks related to real-time functioning. We propose a Receding Horizon Controller whose outline is given in *Algorithm* 1. Step 2 calls algorithm $\boldsymbol{A}_{\mathrm{RH}}$ devoted to the chosen metaheuristic. Using the PM based on knowledge up to time k, $\boldsymbol{A}_{\mathrm{RH}}$ tries to optimize the objective function $J(k, h, X(k))$ on the control horizon of [k, k + h] starting from the initial state of $X(k)$. Therefore, $\boldsymbol{A}_{\mathrm{RH}}$ is equivalent to $\boldsymbol{A}(k, h, X(k))$ and yields the best control sequence that it can find during the current sampling period. This is in fact *the predicted control sequence* (*pcs*):

$$pcs = \langle U(k|k), \ldots, U(k+h-1|k)\rangle. \qquad (5)$$

Algorithm 1 Outline of Receding Horizon Controller

1. Get the current value of the state vector, $X(k)$.
2. Call $\boldsymbol{A}_{\mathrm{RH}}$ in order to generate *pcs*.
3. Send the control input $U(k)$ that is the first element of *pcs* towards the dynamic system.

Return

The Receding Horizon Controller is called iteratively by the control program at each sampling of period k until the end of the control horizon.

2.2 Some Aspects Concerning the Implementation of the Receding Horizon Controller

The description of the systematic design procedure for the controller used within the RHC (see Section 2.1) can be completed with some practical aspects that will be very useful for the implementer.

2.2.1 Correspondence between the Performance Index and the Prediction Horizon

Usually, in our context, the objective function—and implicitly the performance index—can be expressed for the sake of simplicity by its continuous general form as

$$J = \int_{t_0}^{t_f} L(x(t), u(t), t)dt + M(t_f, x_f)\ . \qquad (6)$$

The first part is a Lagrange-type term that measures the quality along the trajectory of the dynamic system; and the second part is a Mayer-type term that measures the quality of the trajectory in its final extremity. If necessary, a discrete form can be derived. The structure of the performance index is decisive for the strategy of the RHC related to the prediction horizon. The Mayer-type term will be called the *terminal penalty* in the sequel.

As mentioned in other works Hu and Chen (2005a); and Hu and Chen (2005b) the prediction horizon can have different positions inside the control horizon [0, H]. Figure 2 shows the two possible situations generated using the RHC structure. In scheme (a), the prediction horizon includes the final moment of H of the control horizon. Accordingly, the prediction horizon's length is the variable having the value of h=H-k, where k is the current sampling time. Because k evolves from 0 to H-1, it holds h=H, …, 1.

In scheme (b), the prediction horizon has a constant length of h < H. Hence, from k=0 until k =H-h, the controller makes h-steps-ahead predictions. But over the final segment of the control horizon, from $k = H - h + 1$ until $k = H - 1$, the prediction horizon's length will decrease from h-1 to 1.

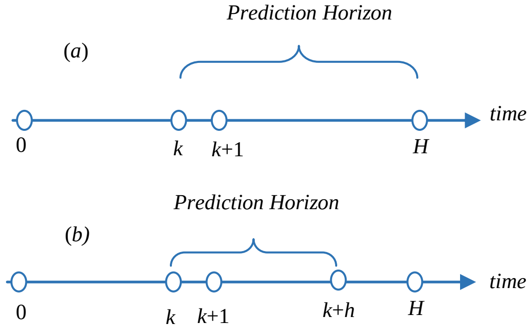

Figure 2. The Relative Position of the Prediction and the Control Horizons.

In our opinion, some remarks related to schemes (*a*) and (*b*) have to be underlined.

Remark 2: Schemes (*a*) and (*b*) are not options of the design procedure. Case (*a*) was presented by some authors as the conventional dynamic optimization and scheme (*b*) as the RHC. Generally, scheme (*a)* is compulsory when the performance index includes a terminal penalty, because this must be calculated, and consequently, the prediction horizon must include the final states. Scheme (*b*) is devoted to the OCPs whose performance index has only an integral component concerning the control and state variables.

However, scheme (*a*) meets the elements that define the RHC strategy:

- a decision is made by looking ahead for a number of steps in terms of a given cost/criterion but it is only implemented by one step;
- the prediction uses a dynamic PM;
- the implementation result is checked and a new decision is made by taking updated information into account and looking ahead for other number of steps.

The prediction horizon "recedes" at each sampling period but keeps the final extremity. Hence, its length decreases by one unit at each sampling period.

There is a particular situation when scheme (*b*) might be applied to an OCP whose performance index has a terminal penalty. This is an interesting case described in some papers such as Hu and Chen (2005b) when a terminal penalty must be considered to ensure the effectiveness of the solution. The basic idea is that the terminal penalty cannot be computed, because the prediction horizon does not include the final moment of H but it can be estimated.

The performance index at the moment of k can be expressed (see Figure 2(b)) by the following equation:

$$J(k, H, u)=J(k, k+h, u)+J(k+h+1, H, u) + FP(x(t_f)), \quad (7)$$

Where $J(k_1, k_2, u)$ is the integral component calculated on interval $[k_1, k_2]$ for the control sequence u, and $FP(x(t_f))$ is the final penalty for the final state $x(t_f)$ (that obviously depends on u). Scheme (*b*) implies that only the value of $J(k, k+h, u)$ is calculated by $\boldsymbol{A}_{\mathrm{RH}}$. If the sum of $J(k+h+1, H, u) + FP(x(t_f))$ can be estimated, the value of $J(k, H, u)$ can be calculated using this estimation. Generally speaking, there is no such estimation for each OCP. When such estimation exists, the receding controller may use scheme (*b*), which is more efficient than using scheme (*a*) from the point of view of a numerical complexity. In this work, because we consider as a matter of priority the dynamic processes modeled by differential algebraic equations estimating the terminal penalty is even more difficult.

2.2.2 Time Constraints and the Prediction Horizon

As in any closed-loop control system, the choice of T is related to the control engineering aspects, such as the discretization of the continuous signals and the time constants of the considered dynamic sub-systems.

The duration of the control horizon is an input data of the given OCP and is equal to $H \cdot T$. Hence, if T is chosen, then the H value can be deduced. The first constraint that can attest to the ability of $\boldsymbol{A}$ to solve the given OCP is that the execution time of algorithm $\boldsymbol{A}$ must be less than the control horizon.

A crucial parameter of algorithm $\boldsymbol{A}_{\mathrm{RH}}$ is the prediction horizon that is equal to $h \cdot T$. From the point of view of the control structure's optimal character, the ideal situation would be to determine the current control input as being the first element of the optimal control sequence for the entire control horizon, which is a solution of $\boldsymbol{A}$. As shown before, $\boldsymbol{A}_{\mathrm{RH}}$ takes into consideration only the *pcs* determined for the prediction horizon. Thus, a larger value for h is desirable since there is an increased chance of quasi-optimal behavior along the control horizon. However, this fact contradicts the execution time of $\boldsymbol{A}_{\mathrm{RH}}$.

The most restrictive time constraint for algorithm $\boldsymbol{A}_{\mathrm{RH}}$ is that the execution time is smaller than the sampling period. Therefore, h is chosen in such a way that $\boldsymbol{A}_{\mathrm{RH}}$ terminates inside the sampling period. This is a difficult constraint to meet and is the reason why the RHC can be used for processes with relatively large time constants.

2.2.3 Evaluation of the Optimal Behavior of the Closed-loop Control Structure

In comparison with the "ideal" situation of the open-loop structure that implements a quasi-optimal solution given by $\boldsymbol{A}$, the RHC is a more complex structure introducing feedback (creation of the closed-loop structure) accompanied by the prediction technique. Hence, the RHC generates degraded optimal behavior, meaning that the control input sequence and state trajectory involves a deterioration of the performance index. In order to analyze the effectiveness of our implementation approach, two simulation series may be carried out. The first simulation series consists of a number of executions of ***A*** (e.g., 30–40). The quasi-optimal evolution of the open loop system is simulated. The average performance index over the simulation series is calculated and a particular simulation of ***A***, whose performance index is the closest to the average, may be considered as the *typical execution*.

The second simulation series employs a simulation model where the real process is identical to the PM (see Figure 1). As a result, the feedback value of the real state $X(k)$ is equal to the state value computed by the PM. As in the first simulation series, the simulation was carried out a number of times (e.g., 30–40) to determine the average performance index and typical execution. By making a comparison with the "ideal" situation of the open loop, one can establish how much the RHC structure will alter the quasi-optimal character of the constructed solution.

2.2.4 Stability Analysis for the Closed-loop System

One of the first papers that deal with the stability of the RHC structure is by Mayne and Michalska (1990). The authors have proven that under certain assumptions the RHC yields a stable closed-loop system when applied to a time-invariant nonlinear system.

Generally speaking, it is impossible to ensure the applicability of this theorem in our context where the optimal solution is computed through an approaching algorithm. But the authors have proven another important theorem, which validates the use of *suboptimal controls* in constructing receding horizon laws. The suboptimal control value sent to the process is denoted by $h^*_{\text{sub}}(x)$ for any current state of x. With our notation from Section 2.1, it holds:

$$h^*_{\text{sub}}(x(k)) = U\,(k|k). \tag{8}$$

This control law yields a closed-loop system,

$$\dot{x} = f(x, h^*_{sub}(x)) \tag{9}$$

that uses the receding horizon strategy and can be asymptotically stable, if certain constraints are met.

The significance of the above result is very important, because it shows that exact optimization is not essential. The approximate optimal control laws work well in practice with discrete systems and gives stabilization towards a small neighborhood of the state space's origin. In our case, if $\boldsymbol{A}_{\text{RH}}$ has a good convergence speed, then *pcs* approaches well with the *ocs* and value $\|pcs(x) - ocs(x)\|_\infty$ may be sufficiently small. This fact involves the stabilization of the closed-loop system towards a small neighborhood of the origin. The larger the convergence speed, the smaller the value $\|pcs(x) - ocs(x)\|_\infty$ will be.

3 A RECEDING HORIZON CONTROLLER FOR THE PROTEIN PRODUCTION

3.1 Protein Production Problem

IN this section, an OCP described in Nikumbh et al. (2014) is considered. This problem involves the fed-batch production of an induced foreign protein by recombination bacteria. In order to maximize profitability from the fermenter, it is necessary to determine the optimal evolution of two control inputs, the inducer and nutrient feed that vary with time. The given batch time t_f is 10 h.

The nonlinear system is described by the following state equations:

$$\dot{x}_1 = u_1(t) + u_2(t) \tag{10}$$

$$\dot{x}_2 = g_1 \cdot x_2 \text{-} (u_1(t) + u_2(t)) \cdot \left(\frac{x_2}{x_1}\right), \tag{11}$$

$$\dot{x}_3 = \frac{100 \cdot u_1}{x_1} - (u_1(t) + u_2(t))\left(\frac{x_3}{x_1}\right) - g_1 \cdot \frac{x_2}{0.51} \tag{12}$$

$$\dot{x}_4 = g_2 \cdot x_2 - (u_1(t) + u_2(t)) \cdot \left(\frac{x_4}{x_1}\right) \tag{13}$$

$$\dot{x}_5 = \left(\frac{4 \cdot u_2(t)}{x_1}\right) - (u_1(t) + u_2(t)) \cdot \left(\frac{x_5}{x_1}\right) \tag{14}$$

$$\dot{x}_6 = \left(-\frac{0.09 \cdot x_5}{0.034 + x_5}\right) \cdot x_6 \tag{15}$$

$$\dot{x}_7 = \left(\frac{0.09 \cdot x_5}{0.034 + x_5}\right) \cdot (1 - x_7) \tag{16}$$

$$g_1 = \frac{x_3}{14.35 + x_3 \cdot (1 + x_3/111.5)} \cdot \left(x_6 + \frac{0.22 \cdot x_7}{0.22 + x_5}\right) \tag{17}$$

$$g_2 = \left(\frac{0.233 \cdot x_3}{14.35 + x_3 \cdot (1 + x_3/111.5)}\right) \cdot \left(\frac{0.005 + x_5}{0.022 + x_5}\right) \tag{18}$$

There are some algebraic constraints:

$$0 \le u_1(t), u_2(t) \le 1 \tag{19}$$

For the initial time t_i=0, the initial values of the seven state variables are

$$X_0 = <1, 0.1, 40, 0, 0, 1, 0> \tag{20}$$

The objective function is

$$J = x_1(t_f) \cdot x_4(t_f) - Q \cdot \int_{t_0}^{t_f} u_2(t) \cdot dt\,. \tag{21}$$

The performance index involves the maximization of the objective function:

$$\max_{u_1(t),u_2(t)} J \qquad (22)$$

Our objective is to implement a receding horizon controller for a real-time control structure using the MbA and RHC structure.

3.2 Implementation of the Receding Horizon Controller

As we have already seen, in the first phase of our procedure, we had to choose a metaheuristic to solve the considered OCP. In Nikumbh et al. (2014) the authors used the *Biogeography-Based Optimization* (BBO) and obtained good results by considering a sampling period equal to 1h. In Section 8.6 of the book by Jayaraman and Siarry (2014), the value Q=0 is considered within the objective function (21). Consequently, the performance index has only a terminal penalty term. Besides this value, the case Q=1.5 will be considered as well in our work, in order to have an objective function that represents a terminal penalty together with an integral type term.

We have used in our simulations an evolutionary algorithm that has similar efficiency in determining the offline solution. This is algorithm ***A*** upon which the implementation of the RHC structure is based on. Our intention is not to emphasize the use of another metaheuristic but to analyze the efficiency of the closed-loop structure obtained with the proposed implementation approach.

Every control input is represented in time by a sequence of 10 values corresponding to each sampling period, that is, H=10. Finally, a solution is coded by concatenating the two sequences. Algorithm ***A*** uses a direct encoding with real (non-binary) values and has some usual characteristics; the population of each generation has μ individuals; the offspring population has λ individuals; the selection strategy is based on the Stochastic Universal Sampling using the rank of individuals, which is scaled linearly using the selection pressure; one-point crossover operator that yields a single offspring; and mutation with global variance adaptation. $\boldsymbol{N}_{\mathrm{gen}}$ is the number of generations in which the population is evolving;

Algorithm ***A*** may be implemented as a function that returns the *ocs* for the Protein Production Problem:

$$ocs=\boldsymbol{A}(\lambda, \mu, N_{\mathrm{gen}}, t_0, H, X_0), \qquad (23)$$

where t_0=0 is the initial time and X_0 is the initial state vector. In comparison with Section 2.3.3, ***A*** has three additional arguments (λ, μ, N_{gen}) for the parameterization of the evolutionary algorithm. Owing to the time constraint, $\boldsymbol{A}_{\mathrm{RH}}$ will work with smaller values for these three parameters (λ_{RH}, μ_{RH}, N_{RH}) compared with algorithm ***A***. The values used in our simulations are listed in Table 1.

Table 1. Parameters of the Evolutionary Algorithm.

	λ	μ	N_{gen}
A	70	40	120
$\boldsymbol{A}_{\mathrm{RH}}$	60	30	60

Because the objective function given by equation (21) showing a terminal penalty and there is no estimation of the final state $X(t_f)$, algorithm $\boldsymbol{A}_{\mathrm{RH}}$ has to implement a prediction horizon using scheme (a) from Figure 2. As described in Algorithm 2, the length of the prediction horizon h decreases by 1 at each sampling period. The final time is always H. Hence, the *pcs* for the Protein Production Problem has $2\cdot h$ elements, with $h = H,\ldots, 2, 1$.

Algorithm 2: $\boldsymbol{A}_{\mathrm{RH}}$ using Scheme (a)

Input: Current sampling period: k;
Population size: λ_{RH}; Offspring size: $\mu_{\mathrm{RH};}$
The current state vector: $X(k)$;
Number of generations: N_{RH};
Other parameters necessary to run algorithm ***A***
Output: Predicted control sequence *pcs*

h= H-k
pcs=$\boldsymbol{A}(\lambda_{\mathrm{RH}}, \mu_{\mathrm{RH}}, N_{\mathrm{RH}}, k, h, X(k))$
Return pcs

3.3 Simulation Results

As mentioned before, in Section 8.6 of the book by Jayaraman and Siarry (2014) an optimal solution for Q=0 is given. Our first goal is to compare this solution to that one constructed by the RHC structure endowed with the $\boldsymbol{A}_{\mathrm{RH}}$ algorithm.

The algorithm based on the BBO found out a maximum value for the objective function, $J_{\max}$=6.15. The reported control inputs u_1(t) and u_2(t) that maximize the objective function are depicted in Figure 3. The evolution of the state variables obtained by simulation using these control inputs is presented in the same figure. The final values are $x_1(t_f)$=3.782 and $x_4(t_f)$=1.6263 that confirms the value of $J_{\max}$.

For the same problem with Q=0, the RHC structure was implemented and simulated using the MATLAB system. Its controller is based on the evolutionary algorithm represented in the previous section. In our simulation model, the real process is identical to the process model used for the state prediction. Because of its stochastic character, the evaluation of the RHC structure was made through a simulation series that consists of 30 executions of the simulation model. The results are presented in Table 2.

Table 2. Results of the Simulation Series for the RHC Structure.

Best	Worst	Mean	Var	$J_{\max}$
6.3397	6.0539	6.2821	0.0046	6.288

The first four columns of Table 2 show the maximum, minimum, mean value, and variance of the performance index, respectively. Column $J_{\max}$ shows the performance index value of the typical execution. The state final values are $x_1(t_f) = 4.0364$ and $x_4(t_f) = 1.5578$ that confirms the value 6.288. The quasi-optimal control inputs and the state evolution are depicted in Figure 4.

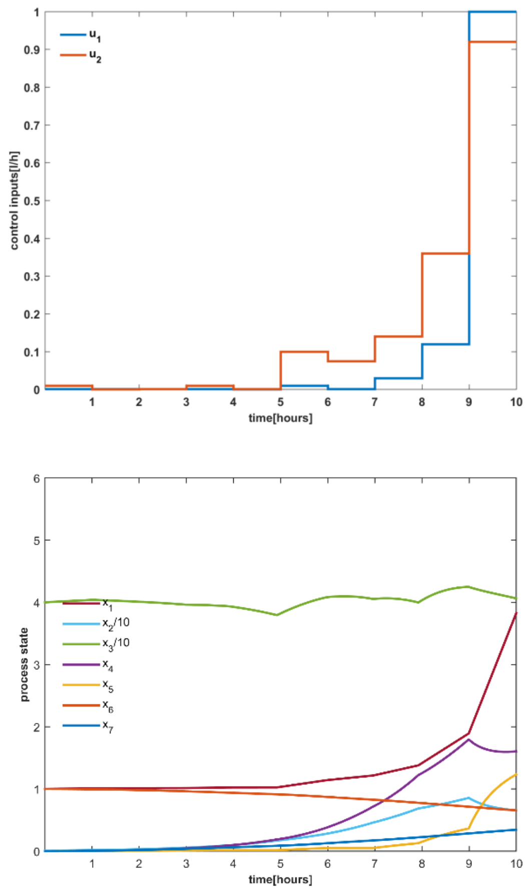

Figure 3. Control Inputs and State Variables Produced by the BBO Algorithm.

In conclusion, the second solution is better than the first one in spite of the RHC implementation.

The second objective of our simulations is to evaluate how much the performance index deteriorates just because the RHC mechanism is used. That is why the two simulation series described in Section 2.3.3 were carried out. Each simulation series consists of 30 executions on the simulation model. For each simulation series, a typical execution is selected from the 30 runs. The typical solution produced by algorithm $\boldsymbol{A}$ after its population evolved along 120 generations is depicted in Figure 5.

The curves, which show the evolution of variables x_2 and x_3 are the values divided by 10 for better graphical representation. The performance index has a typical value of $J_{\max}$=5.8885.

Figure 6 shows the evolution of the control inputs and state variables produced by a typical execution of the RHC with $\boldsymbol{A}_{\mathrm{RH}}$. The results are presented in Table 3, which allows the comparison between the running of algorithms $\boldsymbol{A}$ and $\boldsymbol{A}_{\mathrm{RH}}$ within the RHC.

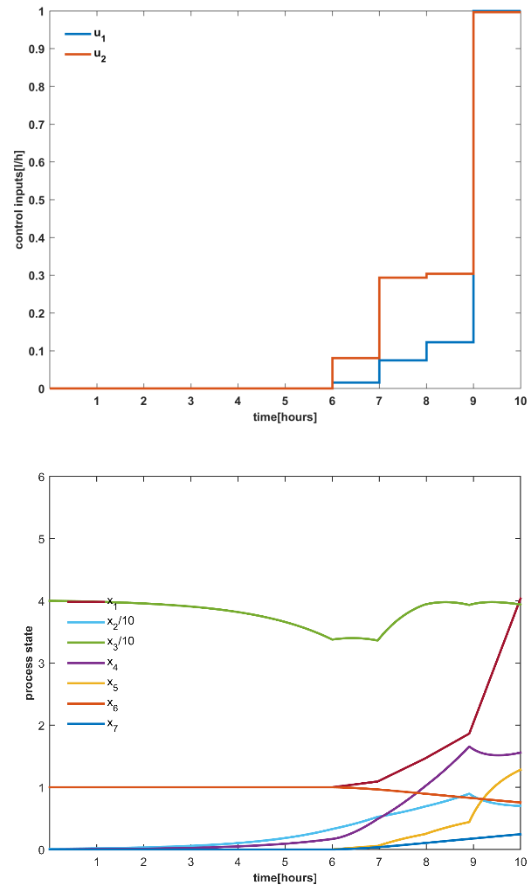

Figure 4. Evolution of the RHC Structure with the Evolutionary Algorithm.

Table3. Results of the Simulation Series.

Simulation Model	Best	Worst	Mean	Var	$J_{\max}$
$\boldsymbol{A}$	6.0312	5.7126	5.8884	0.08486	5.8885
RHC - $\boldsymbol{A}_{\mathrm{RH}}$	6.0226	5.3625	5.8600	0.15697	5.8601

The first four columns of Table 3 show the maximum, minimum, mean value, and variance of the performance index, respectively. Column $J_{\max}$ shows the performance index value of the typical execution. The degradation of the quasi-optimal character from $\boldsymbol{A}$ toward $\boldsymbol{A}_{\mathrm{RH}}$ is represented by the difference between the performance indices, which is 0.0284. The relative decrease of $J_{\max}$ is only 0.48%. This is the effect of introducing the loop closing with the prediction mechanism. Hence, the RHC structure in this case has good efficiency in keeping the quasi-optimal character of the OCP solution. Let us note also that the variance of the performance index doubled its value in comparison with ***A***.

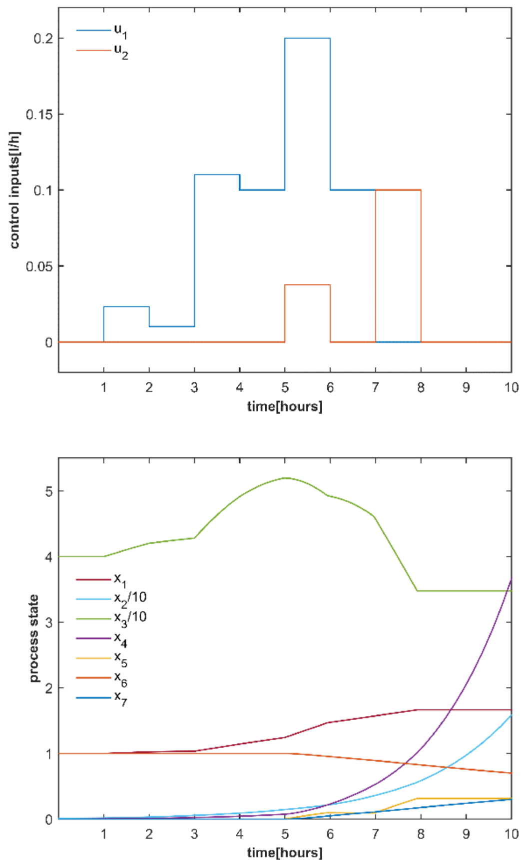

Figure 5. Control Inputs and State Variables Produced by *A* Algorithm.

4 A RECEDING HORIZON CONTROLLER FOR THE SEWER NETWORK DISCHARGE

4.1 Sewer Network Discharge Problem

IN this section, the sewer network discharge is presented as an OCP called the Sewer Network Discharge Optimization Problem (SNDOP). A nonlinear discrete model for a sewer network (SN) was proposed and developed in Minzu et al. (2014). Figure 7 illustrates an example of a SN with 10 retention tanks. The rectangular element represents the retention tank for wastewater and the circular element is the collector for all flow capacities that are inputs of the tanks.

The considered time is discrete, having the control horizon H, namely t=0, 1,…, H. The influent that affects the SN is represented by the flow capacities of $d_i(t)$, i=1,…, 10, from 10 catchments areas. The discharge of tank i caused by its evacuation pump is denoted by $u_i(t)$, i=1,…, 10. A characteristic of the SN is that each tank has only one downstream tank. When the tanks reach their maximum capacity M_i, i=1, …, 10, expressed by volume units, there will be overflows denoted by $q^{over}{}_i(t)$.

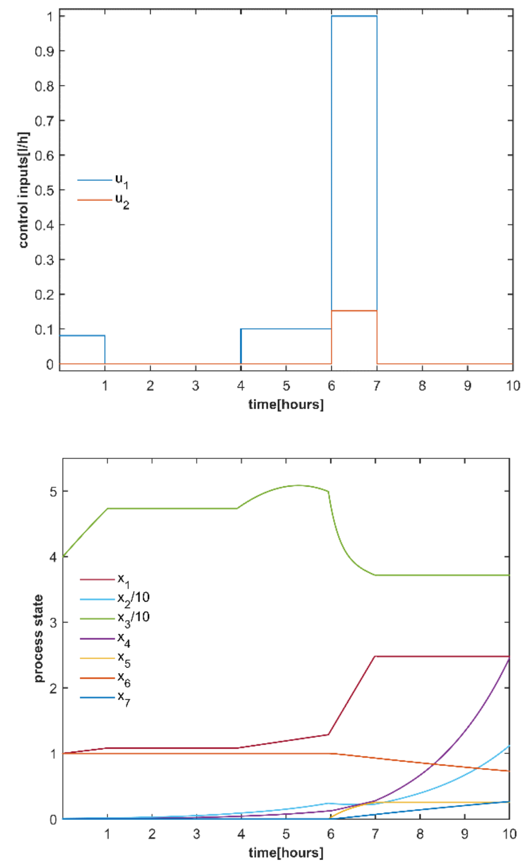

Figure 6. Control Inputs and State Variables Produced by the RHC with the A_{RH}.

The state variables represent the remaining wastewater volumes in the tanks. For example; in tank i at moment t, there is a wastewater volume of $(x_i(t)\cdot\Delta V)$ m^3:

$$x_i(t) \in \{0, 1, \cdots, M_i\},\ i = 1, \cdots, 10 \tag{24}$$

$$X(t) = [x_1(t) \quad x_2(t) \quad \cdots \quad x_{10}(t)]^T \tag{25}$$

Let $q_i(t)$ be the total flow capacity of the wastewater entering retention tank i. In order to simplify the model, variables $d_i(t)$, and $u_i(t)$ can be expressed as integer values. All the water columns have the same evacuation power, which may be expressed by the wastewater volume evacuated in sampling period T. Let ΔV be this volume. Hence, all the variables have values in multiples of ΔV. These values are denoted as; $D_i(t)$, $U_i(t)$, $Q_i(t)$, and $Q^{over}{}_i(t)$, i=1,…, 10 and are integer values. Under these conditions, a realistic hypothesis has been adopted:

$$U_i(t) \in \{0,1\} \tag{26}$$

The control input vector is

$$U(t) = [U_1(t)U_2(t)\ \cdots U_{10}(t)]^T \tag{27}$$

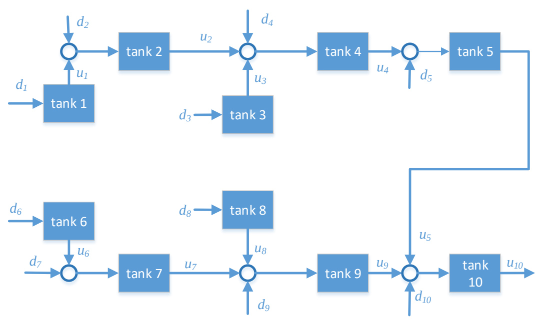

Figure 7. Sewer Network with 10 Retention Tanks.

The estimated influent is known as input data for all tanks over the entire time horizon:

$$D(t) = [D_1(t), \cdots, D_{10}(t)],\ t=0,\ldots,H\text{-}1 \tag{28}$$

The expressions of Q_i, i=1,…, 10, reflects the sewer network's structure. For our example, it holds;

$$\begin{aligned}
&Q_1(t) = D_1(t);\quad Q_2(t) = D_2(t) + U_1(t);\\
&Q_3(t) = D_3(t);\quad Q_4(t) = D_4(t) + U_2(t) + U_3(t);\\
&Q_5(t) = D_5(t) + U_4(t);\quad Q_6(t) = D_6(t);\\
&Q_7(t) = D_7(t) + U_6(t);\quad Q_8(t) = D_8(t);\\
&Q_9(t) = D_9(t) + U_7(t) + U_8(t);\\
&Q_{10}(t) = D_{10}(t) + U_5(t) + U_9(t);
\end{aligned} \tag{29}$$

The statement of SNDOP is composed of the following elements:

♦ the nonlinear discrete system is

$$X(t+1) = f(t, X(t), U(t)) = \\ = \begin{bmatrix} \min(M_1, x_1(t) + Q_1(t) - U_1(t)) \\ \cdots\cdots \\ \min(M_{10}, x_{10}(t) + Q_{10}(t) - U_{10}(t)) \end{bmatrix}, \\ t = 0, \ldots, H-1 \tag{30}$$

♦ the initial conditions are

$$X_0 = X(0) = [x_1(0), x_2(0), \cdots, x_{10}(0)]^T\ ; \tag{31}$$

♦ the bound constraints are

$$0 \le x_i(t) \le M_i,\ x_i(t) \in \mathbf{N}, i = 1,\ldots,10\,; \tag{32}$$

♦ the objective function is

$$I(t) = \sum_{s=t}^{H-1} \sum_{i=1}^{10} \max(0, x_i(s) + Q_i(s) - U_i(s) - M_i) \tag{33}$$

The inner sum is equal to the overflow of all tanks in sampling period s. Hence, $I(t)$ is equal to the total overflow for all tanks and for all sampling periods on the time horizon of $[t, H]$.

♦ the performance index is

$$I^* = \min_{U(t), t=0,\ldots,H-1} I(0)\ . \tag{34}$$

The SNDOP consists of finding the control input values that minimize the total overflow $I(0)$, whereas influent $D(.)$ is present in the SN and affects the state of the retention tanks, acting like a disturbance. Obviously, the estimated influent D and the initial state of vector X_0 are considered to be already known. The solution of the optimization problem is a sequence of control inputs, $U(t)$, with t=0, …, H-1. For a given SN structure, this optimization problem has the following initial data; T, H, M_1,…, M_{10}, the estimated influent D described by (28), and the initial state vector X_0.

4.2 Implementation of the Receding Horizon Controller

The BHPSO algorithm introduced in Beheshti et al. (2015) was specially developed for solving the SNDOP in Minzu et al. (2015). This is why it may assume the role of algorithm $\boldsymbol{A}$ in our approach. Because the aim of this section is to give another example showing how to implement and analyze the RHC structure, only the main characteristics of this algorithm are reviewed.

Let u be a solution of SNDOP. For algorithm $\boldsymbol{A}$, u is the position vector of a swarm's particle. Thus solution u is a sequence of control inputs for all the tanks and all the sampling periods belonging to the time horizon. With this assumption, the structure of u is given as

$$u = [\underbrace{U_1(t_0)\ldots U_{10}(t_0)}_{t_0=0} | \cdots | \underbrace{U_1(t_f)\ \ldots U_{10}(t_f)}_{t_f=H-1}] \tag{35}$$

Hence, the solution is a binary vector having $m = 10 \times H$ bits. With regard to the estimated influent, our simulations used a typical example corresponding to the situation when the SN is affected by significant rainfall. The evolution of the estimated influent $d_i(t)$ related to some retention tank's is shown in Figure 8. Because all the variables from the model described before have integer values, the estimated influent is subject to discretization. The result is a matrix whose element $D(t, i)$, t=0,…, H-1, i=1,…, 10 (expressed as multiples of ΔV) is the continuous influent $d_i(t)$ after discretization and quantization, such that the total volume of influent is preserved. This matrix is input data for both algorithms $\boldsymbol{A}$ and $\boldsymbol{A}_{\mathrm{RH}}$.

When the initial state and estimated influent D are set, solution u determines uniquely the state trajectory of the discrete system (30) and the total overflow Q^{over}=$I(0)$, (see Figure 9). In our implementation, the evaluation of the objective function was achieved by a function that calculated the evolution of SN in compliance with the model (26) - (33). This function returns the value of the total overflow Q^{over}, which has to be minimized. Implicitly, as indicated in Figure 9, the evaluation function determinates the sequence of states $X(0),\ldots,X(H)$, which begins with the initial state.

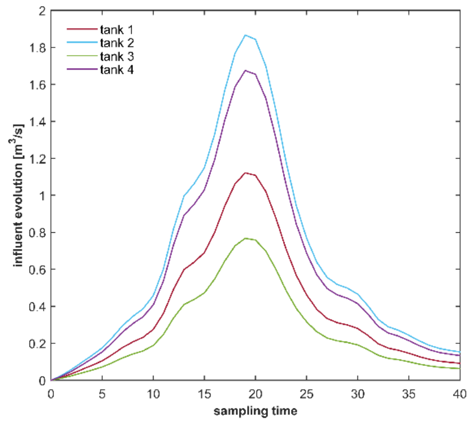

Figure 8. Example of the Estimated Influent.

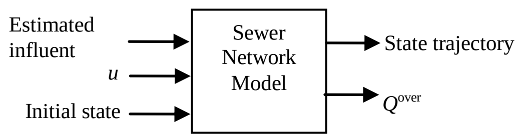

Figure 9. The Sewer Network as a Dynamic System.

Algorithm $\boldsymbol{A}$ may be implemented as a function that returns the *ocs* for SNDOP:

$$ocs=\boldsymbol{A}(n_s, D, t_0, H, X_0), \tag{36}$$

where t_0, X_0 and n_s are respectively the initial time showing the state vector and the number of particles in the swarm. There are other parameters necessary to the configuration of $\boldsymbol{A}$ as a PSO-based algorithm, which are not covered here. In our simulations, these parameters have identical values in both algorithms $\boldsymbol{A}$ and $\boldsymbol{A}_{RH}$. In principle, owing to time constraints, $\boldsymbol{A}_{RH}$ may use a swarm with a smaller number of particles n_s.

Algorithm 3: $\boldsymbol{A}_{RH}$ using the scheme (b)

Input: Current sampling period: k;
Length of prediction horizon: h ;
Current state vector: $X(k)$;
Other parameters necessary to run $\boldsymbol{A}_{RH}$
Output: Predicted control sequence *pcs*

if $k \le H\text{-}h$
$pcs=\boldsymbol{A}(n_s, D, k, h, X(k))$
else /* $H\text{-}h<k\le H\text{-}1$ */
$pcs=\boldsymbol{A}(n_s, D, k, H\text{-}k, X(k))$
Return pcs

In this case, the objective function given by equation (33) does not have a terminal penalty term. This is a favorable context for the computational complexity of algorithm $\boldsymbol{A}_{RH,}$ because it may implement a prediction horizon using scheme (*b*) from Figure 2. As described in Algorithm 3, the prediction horizon is the interval $[k, k+h]$. The length of the prediction horizon h is constant for the sampling periods $k=0,1,\ldots,H\text{-}h$. However, for the final segment of the control horizon $k=H\text{-}h+1,\ldots, H\text{-}1$ the prediction horizon decreases by 1 at each sampling period. Algorithm $\boldsymbol{A}_{\mathbf{RH}}$ uses the same estimated influent D as algorithm $\boldsymbol{A}$, but only the lines of matrix D correspond to the current prediction horizon. The real influent D_{real}, which is unknown, will act in the circumstances of the real-time control but will be considered in our simulations after some hypotheses are made concerning its value.

4.3 Simulation Results

The analysis of the Receding Horizon Controller for the SNDOP was made through simulations that comply with the procedure described in Section 2.2.3. In order to complete the definition of our problem, the remaining parameters have been set as:

M_1=10; M_2=14; M_3=6; M_4=20; M_5=50; M_6=10; M_7=14; M_8=6; M_9=20; M_{10}=40.

$X_0=[3\ 3\ 3\ 3\ 3\ 2\ 2\ 2\ 2\ 2]^T$; T=120 s; H=40.

Hence, the control horizon has 80 min. The size of the swarm has been set to n_s=20, for both algorithms $\boldsymbol{A}$ and $\boldsymbol{A}_{RH}$. For the initial state X_0 and for the estimated influent D represented in the previous section, the SNDOP has the optimal performance index of I^*=0, a fact ascertained by the execution of algorithm $\boldsymbol{A}$.

For the Receding Horizon Controller, the length of prediction horizon h is a crucial parameter that must meet the constraints already mentioned in Section 2.2.2. In order to select it, the control structure was simulated using different values of h. The results are represented in Table 4. The second column shows the number of bits encoding a solution (m=10·h), which is the main factor determining algorithm $\boldsymbol{A}_{RH}$'s complexity and efficiency. The subsequent columns provide the best, worst, mean values, and the variance of the total overflow found after 30 runs of the $\boldsymbol{A}_{RH}$, respectively.

Table 4. Simulation of the RHC with Different *h* Values.

h	*m*	Best	Worst	Mean	Var	Number of runs
5	50	3	12	7.53	4.67	30
10	100	3	11	6.86	3.56	30
16	160	0	5	2.36	1.098	30
25	250	0	3	1.26	0.547	30
35	350	0	1	0.10	0.305	30

By analyzing this table we establish a correlation; the poorer the solution, the greater the variance. An adequate value seems to be h=16, because it is a trade-off between the computational complexity and time constraint for $\boldsymbol{A}_{RH}$.

In the first simulation series, the evolution in the open loop of the system (27)–(33) was simulated. Thirty executions of algorithm $\boldsymbol{A}$ were carried out.

Algorithm ***A*** has greater computational complexity compared with $\boldsymbol{A}_{\mathrm{RH}}$, because of its control horizon H=40 but it has no time constraint. Some results are shown in the first row of Table 5. The column Q^{over} gives the total overflow found in a typical execution.

A typical state evolution produced by algorithm ***A***, whose particle swarm has evolved until convergence, is depicted in Figure 10. For the simulation series concerning the closed-loop control structure that uses

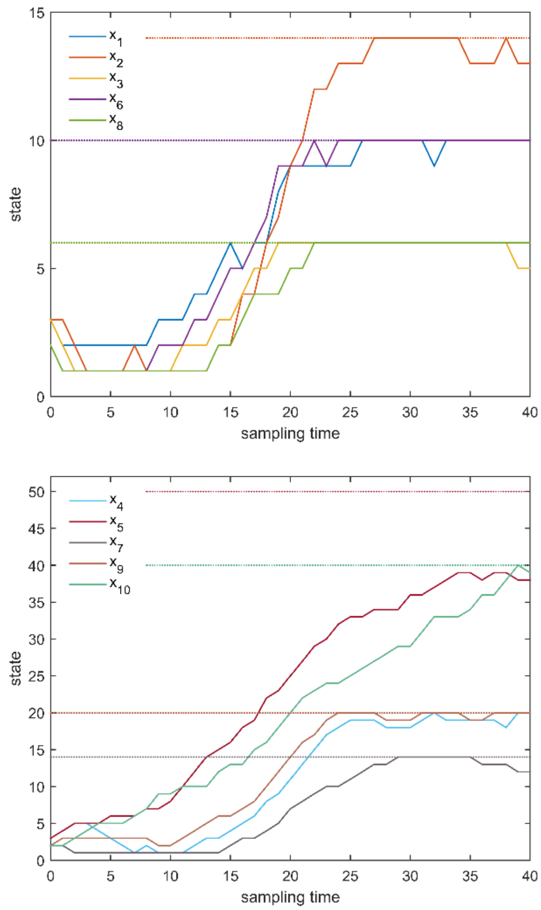

Figure 10. State Evolution in a Typical Execution of *A* Algorithm.

Table 5. Results of the Simulation Series for the SNDOP.

Simulation model	h	Best	Worst	Mean	Var	Q^{over}
A	-	0	4	1.43	0.727	1
RHC - $\boldsymbol{A}_{\mathrm{RH}}$	16	0	5	2.36	1.098	2
RHC - $\boldsymbol{A}_{\mathrm{RH}}$	35	0	1	0.10	0.305	0

$\boldsymbol{A}_{\mathrm{RH}}$, the results from Table 5 show an increase in total influent dispersion. The typical value of Q^{over} is now 2 volume units (1 vu = ΔV). It is the result of introducing the closed loop with the prediction mechanism that works for only 16-steps-ahead predictions, in contrast to algorithm ***A***, which makes predictions in 40 steps. Figure 11 depicts the state evolution obtained in a typical simulation using $\boldsymbol{A}_{\mathrm{RH}}$.

The implementer - who has to decide whether or not the RHC structure works well - can use the results of the simulation to answer to this issue.

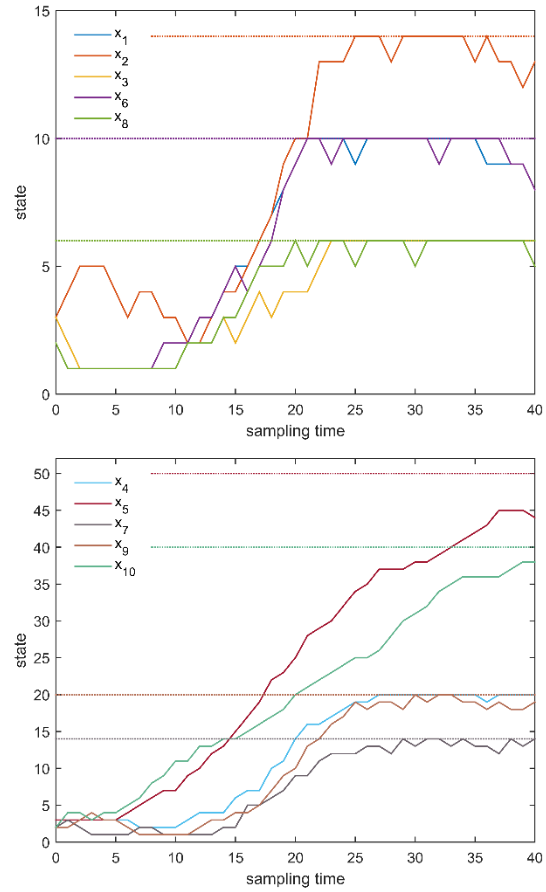

Figure 11. State Evolution in a Typical Execution of the RHC - $\boldsymbol{A}_{\mathrm{RH}}$.

5 CONCLUSION

THIS work proposes a systematic procedure to create a RHC structure whose controller is based on a metaheuristic algorithm. In the first phase, the MbA was implemented beginning with the statement of the control problem. The metaheuristic was chosen according to the previous experience of the designer, who also implemented the algorithm and verified its efficiency in solving the problem. In the second phase, the MbA was slightly modified, as mentioned in Section 2.1, in order to create algorithm $\boldsymbol{A}_{\mathrm{RH}}$ that is included in the controller. The simulation of the RHC structure may be considered as the last phase of the design procedure.

The main contribution of this work is the systematic procedure itself, which has a methodological contribution. The description of this design procedure gives the opportunity to underline some practical aspects.

An important aspect is the structure of the objective function to be optimized. The length of the prediction horizon will be variable (scheme (*a*)) or fixed (scheme (*b*)) depending on whether or not the the objective function has a terminal penalty term. Section 2.2.1 is a complete analysis of all the choices that the implementer can make from this point of view.

Other aspects may be considered drawbacks of the association between the RHC and MbA, but the proposed procedure analyzes these difficulties aiming to overcome them when it is possible.

The procedure has a restrictive time constraint mentioned in Section 2.2.2. That is why this method can be used for the control of slow nonlinear systems, such as chemical batch processes. For a prediction horizon equal to $h \cdot T$, h has to be chosen in such a way that the $\boldsymbol{A}_{\mathrm{RH}}$ would terminate inside the sampling period. It could result in showing a small value of h, a fact that could be in contradiction with an acceptable quasi-optimal behavior. We may try to select another MbA that is less complex or has a better convergence speed.

Another crucial aspect is the good convergence speed of algorithm $\boldsymbol{A}$ that must be carefully proven and tested. This is useful not only to meet the time constraint, but also to ensure the quality of the asymptotical stability.

The simulation is the only way to evaluate the MbA quality and closed loop behavior. Moreover, owing to the stochastic character of the algorithms $\boldsymbol{A}$, $\boldsymbol{A}_{\mathrm{RH}}$ and the entire RHC structure, the evaluation needs to reiterate simulations, in order to determine valid statistic parameters.

On the other hand, the association of the RHC plus the MbA is for many cases the unique way to solve an OCP that has two simultaneous characteristics; (a) the process is nonlinear and (b) there is not a feasible deterministic solution for the optimization problem.

Our proposed procedure has algorithmic parts that generate $\boldsymbol{A}_{\mathrm{RH}}$ and the controller. Obviously, these parts can facilitate significantly the implementation of the control structure.

The implementation of the closed loop has a price to be paid; a degradation of the quasi-optimal behavior. In Section 2.2.3, a simulation tool is proposed that establishes how much the RHC structure will alter the quasi-optimal character of the constructed solution. Afterwards, the control structure implementer has to decide whether the RHC structure works well.

The proposed procedure was illustrated by two examples of the OCP; a continuous system whose objective function includes a terminal penalty term and a discrete system with binary control inputs whose objective function has only an integral term. The simulations were organized such that one can evaluate the degradation of the optimal behavior by introducing the closed-loop. The optimal character of $\boldsymbol{A}$ was compared to that of the RHC, which includes the $\boldsymbol{A}_{\mathrm{RH}}$. The simulations showed that the proposed procedure resulted in closed-loop control structures exhibiting good behavior and can be used in real-time control.

6 REFERENCES

Altinten, A., (2007). Generalized predictive control applied to a ph neutralization process. *Computers and Chemical Engineering*, 31(10), 1199-1204.

Aras, O. et al. (2011). Optimization of scaled parameters and setting minimum rule base for a fuzzy controller in a lab-scale pH process. *Industrial and Engineering Chemistry Research*, 3335-3344.

Attia, S.A. et al., (2006). Voltage Collapse Avoidance in Power Systems: A Receding Horizon Approach. *Intelligent Automation & Soft Computing*, 12(1), 9–22.

Beheshti, Z. et al., (2015). Memetic binary particle swarm optimization for discrete optimization problems. *ELSEVIER, Information Sciences* 299, p. 58-84

Blanco, T.B. et al., (2010). Flood regulation using nonlinear model predictive control. *Control Engineering Practice* 18, 1147-1157

Bououden, S. et al., (2015). An ant colony optimization-based fuzzy predictive control approach for nonlinear processes, *Information Sciences, Elsevier*. 299, 143-158.

Bruant, I. et al., (2011). Optimization of Piezoelectric Sensors Location and Number Using a Genetic Algorithm. *Mechanics of Advanced Materials and Structures*, Taylor Francis, 18 (7), 469 - 475

Causa, J. et al., (2008). Hybrid fuzzy predictive control based on genetic algorithms for the temperature control of a batch reactor; *Computers and Chemical Engineering*, 32(12) 3254-3263.

Chiang, P-K., Willems, P., (2015). Combine Evolutionary Optimization with Model Predictive Control in Real-time Flood Control of a River System. *Water Resource Management*, 29: 2527–2542

Christofides, P.D. et al., (2013). Distributed model predictive control: A tutorial review and future research directions. *Computers and Chemical Engineering* 51, 21-41.

Clarke, D.W., 1994. *Advances in Model-based Predictive Control*. Oxford University Press.

Faber, R. et al. (2005). Dynamic optimization with simulated annealing. *Computers and Chemical Engineering* 29, 273–290

Goggos, V., King, R., (1996). Evolutionary predictive control. *Computer Chemical Engineering* 20 (Suppl 2) (6-7), S817-S822.

Kesarkar, A. and Selvaganesan, N., (2015) Tuning of optimal fractional-order *PID* controller using an artificial bee colony algorithm. *Systems Science & Control Engineering*, 3:1, 99-105.

Hiskens, I.A., Gong, B., (2006). Voltage Stability Enhancement Via Model Predictive Control of

Load. *Intelligent Automation & Soft Computing*, 12(1), 117–124.

Hu, X.B. et al. (2004). On-line free-flight path optimization based on improved genetic algorithms. *Engineering Application of Artificial Intelligence 17*, 897-907.

Hu, X.B., Chen, W.H., (2005a). Genetic algorithm based on receding horizon control for arrival sequencing and scheduling. *Engineering Application of Artificial Intelligence 18(2005)*, 633-642.

Hu, X.B., Chen, W.H., (2005b). Genetic algorithm based on receding horizon control for real-time implementations in dynamic environments; *16th Triennial World Congress, Prague*, Elsevier IFAC Publications.

Jabri, K. et al., (2011). Particle swarm optimization based tuning of a modified Smith predictor for mould level control in continuous casting. *Journal Process Control* 21(2), 263-270.

Jayaraman, V.K., Siarry, P. (Editors.), (2014); *Applications of Metaheuristics in Process Engineering*, ISBN 978-3-319-06507-6, Springer.

Mayne, D.Q., and H. Michalska, (1990). Receding horizon control of nonlinear systems, *IEEE Transactions on Automatic Control.* 35, 814-824

Makas, H. and Yumusak, N., (2016). System identification by using migrating bird's optimization algorithm: a comparative performance analysis. *Turkish Journal Of Electrical Engineering & Computer Science*, 1879-1900.

Lopez-Francol, C., (2018). Robot Pose Estimation Based on Visual Information and Particle Swarm Optimization, *Intelligent Automation & Soft Computing*, 24(2), 431-442.

Minzu, V. et al., (2014). Sewer network discharge control using a multi-agent approach. *Proceedings of the 18th International Conference on System Theory, Control and Computing*; Sinaia, Romania, ISBN 978-1-4799-4602-0 ©2014 IEEE, 779– 784;

Minzu, V. et al., (2015). A Binary Hybrid Topology Particle Swarm Optimization Algorithm for Sewer Network Discharge. *Proceedings of the 19th International Conference on System Theory, Control and Computing*, Cheile Gradistei, Romania, ISBN 978-1-4799-8480-0©2015 IEEE, 627-634.

Naghizadeh, R.A. et al., (2016) An Adaptive Approach for Simulation of Inrush Current in Three-phase Transformers Considering Hysteresis Effects, *Electric Power Components and Systems*, 44(6), 673-682

Nikumbh, S. et al., (2014). Biogeography-based optimization for dynamic optimization of chemical reactors, in: Jayaraman, V.K., Siarry, P. (Editors.., *Applications of Metaheuristics in Process Engineering*, ISBN 978-3-319-06507-6, Springer, 201-216.

Qin, S.J., Badgwell, T.A., (2003). A survey of industrial model predictive control technology. *Control Engineering Practice*, 11, 733-764

Qian, F. et al., (2012). Novel hybrid evolutionary algorithm for dynamic optimization problems and its application in an ethylene oxide hydration reactor. *Industrial and Engineering Chemistry Research* 51(49) 15974-15985.

Sarimveis, H., Bafas, G., (2003). Fuzzy model predictive control of non-linear processes using genetic algorithms. *Fuzzy Sets and Systems* 139(1), 59-80.

Singh, A.K., Hahn, J., (2006). Sensor location for stable nonlinear dynamic systems: multiple sensor case, *Industrial and Engineering Chemistry Research* 45(10), 3615-3623.

T. Wong, P.L,. et al., (2016) Empirical Comparison of Differential Evolution Variants for Industrial Controller Design. *International Journal of Computational Intelligence Systems*, 9(5), 957-970.

Venkateswarlu, C., Reddy, A.D., (2008). Nonlinear model predictive control of reactive distillation based on stochastic optimization. *Industrial and Engineering Chemistry Research*. 47(18), 6949-6960.

Yang, Y. et al., (2014). Predictive Control Strategy Based on Extreme Learning Machine for Path-Tracking of Autonomous Mobile Robot. *Intelligent Automation & Soft Computing*, 21(1), 1–19.

Zhang, W. et al., (2018). BDI Agent and QPSO-based Parameter Optimization for a Marine Generator Excitation Controller. *Intelligent Automation and Soft Computing*, 1-10.

Zheng, T., (2010). Model Predictive Control. Edited by Tao Zheng, ISBN 978-953-307-102-2, Publisher: Sciyo.

7 NOTES ON CONTRIBUTORS

V. Mînzu received ME degree in Computer Engineering, from the Polytechnic Institute of Bucharest, Romania in June 1981, Ph.D. degree in Automation Systems from University of Galati Romania in December 1993, and MS and Ph.D. degrees in Computer Science and Automation from the University of Franche-Comté, France in May 1995. He also received Doctor Honoris Causa degree, from the Le Havre University, France in July 2009. Currently, he is professor within the Control and Electrical Engineering Department, "Dunarea de Jos" University of Galati, Romania, teaching Automation Systems and Optimal Control Techniques, (Member of the IEEE, Control Systems Society, and Member of the IFAC Technical Committee 5.2).

A.E. Serbencu received Ph.D. degree in Automation Systems from the University of Galati Romania in June 2017. He works at the *Control and Electrical Engineering* Department, "Dunarea de Jos" University of Galati, Romania. He is associate professor at this University. His research interests are in artificial intelligence applied in control systems, robotics and optimal control.