Received: 10 September 2024 | Revised: 8 February 2025 | Accepted: 24 February 2025 | Published online: 9 May 2025

RESEARCH ARTICLE

Artificial Intelligence and Applications
2026, Vol. 4(1) 57–67
DOI: 10.47852/bonviewAIA52024314

BON VIEW PUBLISHING

Hierarchical Multi-objective Control of Nonlinear Systems with Dynamical Input Constraints

Ali Deeb[1,*], **Vladimir Khokhlovskiy**[1] **and Viacheslav Shkodyrev**[1]

[1]*Higher School of Cyberphysical Systems and Control, Peter the Great St. Petersburg Polytechnic University, Russia*

Abstract: This work presents a multi-level hierarchical control strategy to address the problem of complex multi-objective optimization-based control in real time. Our suggested strategy utilizes evolutionary algorithms to solve the high-level optimization problem, providing a control policy under which a lower-level control loop handles the dynamics of the control values while respecting both regional and dynamical input constraints. Moreover, a real-time under-policy prediction phase is developed to absorb the latency of the computationally expensive policy search phase. The overall strategy is designed to leverage nonlinear systems without the need for further linearization or operating point approximations. Experimental results on a drum boiler-turbine unit simulation demonstrate the capabilities of our suggested strategy to steer the system outputs toward desired values with faster convergence compared to traditional methods. The proposed approach shows significant improvements in control performance, handling complex nonlinear control problems in real time, and providing optimized and fast control signals to guide the system outputs towards different operating points.

Keywords: multi-objective optimization, advanced control, boiler-turbine system, evolutionary machine learning, nonlinear control

1. Introduction

Controlling nonlinear dynamic systems with input constraints poses significant challenges in modern control engineering. Such challenges are exemplified in systems like thermal power units, which are essential in converting chemical fuels such as coal, oil, or gas into electrical energy, and represent a critical component in modern energy conversion systems [1]. Controlling such units requires maintaining certain safety limitations and constraints while simultaneously meeting the fluctuating demands for power, electricity for example. The operation of thermal power units is fraught with challenges, particularly due to severe non-linearity and the intricate coupling of multiple variables. This complexity often results in performance and stability issues [2]. Controlling such thermal systems poses significant challenges due to its inherent non-linearity and the decoupling between its inputs and outputs [3, 4].

While classic controls are deemed sufficient for most control problems [5], recent insights suggest model predictive control (MPC) is versatile enough for almost all scenarios, including those previously challenging due to limited knowledge or feasibility [6]. It's shown in the literature that MPC is vastly utilized in solving control problems of nonlinear dynamic systems with constraints, and it's often accompanied by system linearization or fuzzification in the neighborhood of operating points to make the optimization problem computationally feasible for real-time implementation. On the other hand, machine learning (ML) offers "intelligent" controllers that outperform adaptive controls in handling system nonlinearities and uncertainties [7, 8]. However, implementing these ML controllers in real-time scenarios can be challenging due to their computational demands [9].

In this work, we introduce a novel hierarchical control methodology that leverages the harness of evolutionary ML (EML) to guide a real-time fast control strategy with dynamical (rate-of-change) constraints. Our proposed strategy is divided into three phases: Online Under-policy Prediction, Policy Search, and low-level control. This hierarchical structure helps leveraging the power of evolutionary algorithms (EAs) to build a control policy by solving a multi-objective optimization (MOO) problem, in the Policy Search phase. Under this selected policy, the Online Under-policy Prediction phase is designed to preempt the latency encountered in the policy search stage, thereby circumventing delays in the real-time control mechanism. The low-level control phase, however, keeps track on the latest updates on the policy to steer the control signals with respect to their regional and dynamical constraints. Experimental results on a boiler-turbine unit system demonstrate the capabilities of our suggested approach to steer the system outputs towards the desired values with respect to all input constraints and with minimal drifts from the balance points. Moreover, our framework can utilize the nonlinear dynamical equations of the system without the need for any local approximations or further linearization, with a sampling time of 0.1 s, allowing for a high degree of accuracy in capturing

*Corresponding author: Ali Deeb, Higher School of Cyberphysical Systems and Control, Peter the Great St. Petersburg Polytechnic University, Russia. Email: dib_a@spbstu.ru

the true nonlinear behavior of the system while ensuring stable and consistent performance under varying operational conditions. Finally, our results show the ability to steer the control inputs of a boiler-turbine system with a sampling period of 0.1 s, even if the EAs search latency is up to 0.7 s. This last notation indeed highlights the possibility of having our methodology applied in real-time control loops.

The remainder of this paper is organized as follows: Section 2 delves into a literature review, laying the groundwork for our approach. Section 3 articulates our unique problem formulation and methodology. Section 4 presents a test case on a boiler-turbine unit. Section 5 shows limitations and future work, while Section 6 is the conclusion.

2. Literature Review

Boiler-turbine system control has been extensively studied in the research literature, with a significant focus on system identification and linearization at nominal operating points, utilizing deterministic and fuzzy controllers. This task remains challenging due to the complex dynamics inherent to boiler-turbine systems, as noted in Wang et al. [10]. For instance, Wu et al. [11] introduce a fuzzy multi-model MPC approach using an extended fuzzy Lyapunov function, addressing control at nominal points through linear matrix inequalities (LMIs). Similarly, Wu et al. [12] propose a hierarchical control structure based on the Takagi-Sugeno fuzzy model, integrating an optimal reference governor with a fuzzy model predictive controller to improve the system's disturbance rejection and stability.

Further research has aimed to address nonlinear system control challenges, such as the work in Wang et al. [13], which presented a robust MPC strategy with bi-level optimization, incorporating multiple local models for nonlinear dynamics handling. Meanwhile, Zhao et al. [14] propose a nonlinear extended predictive self-adaptive control method, simplifying the cost function to integer order for easier implementation. Additionally, Köhler et al. [15] introduce a computationally efficient robust MPC framework for uncertain nonlinear systems, utilizing online constructed tubes based on incremental Lyapunov functions to ensure robust constraint satisfaction and practical asymptotic stability with minimal computational overhead.

Other approaches proposed by Zhao et al. [16] and Zhao et al. [17] explore advanced techniques such as recurrent fuzzy neural networks to model dynamic responses and Galerkin optimization algorithms for control, demonstrating the continuous evolution of methods to tackle nonlinear control issues. In parallel, Sanchez et al. [18] apply a discrete state-feedback controller enhanced by genetic algorithms (GAs) for LED drivers using a buck converter, highlighting the potential of GAs in optimizing closed-loop dynamics through LMIs and evolutionary techniques.

However, the reliance on linearization techniques, such as Taylor series approximations, presents limitations when trying to capture the full nonlinear behavior of these systems. As seen in Yang et al. [19], while linearization can be effective at steady-state points, identifying piecewise models is necessary for better system representation across different operational zones. For example, Ławryńczuk [20] suggest a MPC strategy based on online linearization of the state-space model at current operating points, facilitating future control policy determination. This method, combined with the use of local linearization and polytopic uncertain LPV models, enables robust MPC for output-tracking control. Yet, such approaches still face challenges when handling the non-linearity inherent in real-world systems, prompting further exploration of simulation-assisted methods.

Adaptive control methods, including those based on ML, have shown potential to overcome these limitations. Cornejo Maceda and Noack [21] discuss the benefits of reduced-order models and local linearization but also highlight their limitations in robustness and applicability. In this context, Song et al. [22] propose an adaptive MPC for the yaw system of variable-speed wind turbines, where the control horizon adapts based on predictive performance, demonstrating enhanced comprehensive performance over baseline MPC methods. Similarly, Cui et al. [23] develope a deep-neural-network-based economic MPC for ultrasupercritical power plants, utilizing deep belief networks to accurately model system dynamics and ensure closed-loop stability through embedded predictors.

ML controllers, by framing control as a regression task, offer more flexibility in adapting to complex dynamics. Wei et al. [24] introduce an optimal tracking control scheme for boiler-turbine systems using integral reinforcement learning, achieving faster convergence compared to traditional MPC methods. Despite the promise of these methods, challenges such as slow learning rates and high computational expenses persist, as noted in Slowik and Kwasnicka [25]. Nevertheless, EAs like those in Reference [18] provide gradual optimization, offering a more robust alternative to traditional control methods. This naturally leads to the need for novel approaches that merge traditional control theory with advanced ML techniques to better handle the nonlinearities and uncertainties present in boiler-turbine systems. By leveraging the strengths of both domains, such frameworks can achieve higher accuracy and stability, ensuring reliable and efficient control under varying operational conditions.

3. Problem Formulation

3.1. Motivation and preliminary research

Nonlinear MPC (NMPC) combined with EML has been demonstrated to efficiently control nonlinear systems, as shown in recent studies [26, 27]. In such framework, the ML controller solves an optimization problem, searching for a set of future control values to minimize the error over a future horizon.

However, this process is challenged by the curse of dimensionality, particularly when dealing with rate-of-change control constraints. While the optimization problem can identify the control parameters needed to steer the system toward the desired state, the search space must be sufficiently large to account for dependencies between parameters over the future horizon.

For instance, in a boiler system, the horizon may need to be 10 s to considerably affect the outcome function or gain to guide the search strategy. With a control time step of 0.1 s, this results in 100 different values for each, leading to a search space of $[-1, 1]^{300}$—a computationally intensive task to be solved in real time by EML.

3.2. Proposed solution

MPC has been extensively applied in literature for the control of dynamic systems. MPC necessitates a model of the system dynamics along with an optimization strategy. In this work, we proceed under the assumption that we possess a sufficiently accurate model of the system dynamics, bypassing the details of the identification process.

Consider a system defined by the following nonlinear equations:

$$\begin{aligned} \dot{X}(t) &= \mathcal{F}(X(t),\ U(t)), \\ Y(t) &= \ g(X(t),\ U(t)), \end{aligned} \tag{1}$$

Where:

1) $X(t) \in \mathbb{R}^n$ represents the state vector of the system at time t,
2) $U(t) \in \mathbb{R}^m$ denotes the control input vector at time t,
3) $Y(t) \in \mathbb{R}^p$ is the output vector of the system at time t,
4) $\mathcal{F} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ and $g : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^p$ are nonlinear functions describing the system dynamics and output, respectively.

To digitalize these dynamic equations, we apply Euler's method, which provides an approximate solution to the differential equations by discretizing time. The discretized version of the system's dynamics can be written as:

$$\begin{aligned} X_{k+1} &= X_k + \Delta t \cdot \mathcal{F}(X_k,\ U_k), \\ Y_k &= g(X_k,\ U_k), \end{aligned} \tag{2}$$

Where:

1) $X_k,\ Y_k,\ U_k$ are the discretized states, outputs, and control inputs at the k-th time step, respectively,
2) Δt is the time step size.

This discretization allows us to apply MPC by solving an optimization problem at each time step to determine the optimal control inputs U_k over a prediction horizon, subject to the system dynamics and any constraints.

To formulate the optimization problem, consider a series of control vectors $\mathbf{U} = \{U_0,\ U_1, \ldots, U_{H-1}\}$ over the prediction horizon H, where the goal is to minimize the deviation of the system's output from a desired trajectory $\mathbf{Y}^d = \left\{Y_0^d,\ Y_1^d,\ \ldots,\ Y_{H-1}^d\right\}$ The optimization problem can be expressed as:

$$\mathbf{U}^* := \min_{\mathbf{U}} J(\mathbf{U}) = \sum_{i=0}^{H-1} \gamma^i \left|\left| Y_i - Y_i^d \right|\right|^2 \tag{3}$$

Where:

1) $\mathbf{U} \in \mathbb{R}^{H \times m}$
2) $j(\mathbf{U})$ is the cost function to be minimized,
3) $\gamma < 1$ is a damping factor to progressively reduce the weight of future errors in the cost function,
4) $||Y_i - Y_i^d||^2$ represents the squared norm of the error between the actual output Y_i and the desired output Y_i^d at each step i within the horizon.

The optimization seeks to find the control sequence U that minimizes the weighted sum of squared output errors over the horizon, subject to the system dynamics as described by the discretized equations and any additional constraints on the states and control inputs.

In this work, we discuss two types of constraints, boundary constraints in which the value of the control input is limited, and first-order dynamic constraints where the first-order derivative (the speed) of every control input is limited.

$$\mathcal{H}(U) = \begin{cases} U \le U \le \bar{U}, \\ -\mu \le \dot{U} \le \mu, \end{cases} \tag{4}$$

where U, $\bar{U}$, and μ from $\mathbb{R}^m$ represent the lower and upper bounds on the control inputs, and the absolute value of control derivative, respectively.

These constraints ensure practical feasibility when applying the control method to real-world systems. The boundary constraints $(U \le U \le \bar{U})$ represent actuator saturation limits, while the rate-of-change constraints $\left(-\mu \le \dot{U} \le \mu\right)$ ensure smooth transitions, preventing abrupt control actions that could damage system components. For other systems, these limits should be defined based on physical constraints of the actuators, system safety requirements, and desired response speed.

Under a proper state-space representation of the control system, as inherently expressed in Equation (1), we can introduce an assumption that the system's evolution follows the Markov property. This implies that the future state of the system depends only on the current state and the current control action, not on the sequence of events that preceded it. Mathematically, this is expressed as:

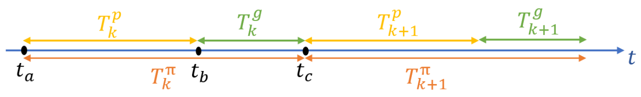
$$\begin{aligned} P(X_{k+1}|X_k, U_k) = P(X_{k+1}| \\ X_0, X_1, \ldots, X_k, \\ U_0, U_1, \ldots, U_k), \end{aligned} \tag{5}$$

and similarly, the output at the next time step depends only on the current state and control input:

$$\begin{aligned} P(Y_{k+1}|X_k, U_k) = P(Y_{k+1}| \\ X_0, X_1, \ldots, X_k, \\ Y_0,\ Y_1, \ldots, Y_k, \\ U_0, U_1, \ldots, U_k), \end{aligned} \tag{6}$$

These assumptions align the optimization problem with the principles of a Markov decision process, where the decision-making at each step k is based solely on the current state X_k and aims to optimize a cumulative cost function, as defined previously. The Markov property ensures that the system's dynamics and the optimization strategy can be effectively modeled and solved using the framework of MPC, leveraging the current state and control input to predict and influence future states and outputs.

Figure 1
Timeline of the proposed control strategy



3.3. Control strategy

In the context of developing our control strategy, we define the following terms and parameters:

1) f_s: Sampling Frequency. This represents the frequency at which the system's control loop is digitized, with a corresponding sampling period denoted by $T_s = \frac{1}{f_s}$.
2) T^p: Prediction Phase Time. Denotes the time interval from which the online prediction phase starts gathering data until making a prediction, i.e., producing an output.
3) Π: Control Policy. Defined as a set of high-level control recommendations that inform and guide the overarching control strategy.
4) T^π: Policy Application Time. This denotes the time interval during which a specific control policy, Π, remains applied to the system.
5) T^g: GA Search Latency. This term refers to the time required for the GA to identify a suitable policy, in policy search phase, as per the established search criteria.

Under these assumptions, the control strategy is divided into the following three main phases:

1) Online Under-Policy Prediction Phase: The initial phase, which is responsible for predicting the system state after a period of time based on online observations. It works in parallel without interfering with the control loop.
2) Policy Search Phase: The most computationally expensive phase, responsible for building a policy for the next T^π. This phase is also designed to be performed on parallel, i.e., through multi-threading or multi-processing.
3) Low-Level Control Phase: This phase applies the control policy on the actual values of control is and responsible for considering all control constraints including the dynamic ones.

Following the assumptions and categories denoted, Figure 1 shows a timeline of the proposed control strategy. It's clear that $T^\pi = T^p + T^g$. Moreover, Policy Search Phase is supposed to start at time t_a and end at time t_b based on a specific design that includes hyperparameters and hardware computational limitations. Therefore, it takes predictions about the system state at time t_b, from the online prediction phase, to design a policy for the next policy slot T^π_{k+1}. At the meanwhile, from ta to tb, and the old policy Π_k continues to be applied in the interval without disturbing or delaying the main control loop.

3.3.1. Online under-policy prediction phase

This phase is responsible for gathering online observations of the system dynamics in every policy interval T^π in order to make predictions about the system state at the end of this interval. Prediction strategies can vary from interpolations to simple ML strategies reaching online Neural Networks training as function approximators.

$$\tilde{X}_{(k+1)T^\pi} = \tilde{X}_{t_c} = \Phi\left(X_{t_a},\ X_{t_a+T_s}, \ldots, X_{t_b}\right) \tag{7}$$

This phase works on parallel without interfering with the control loop and is supposed to give a prediction at the end of T^p about the system state after T^g, as expressed in Equation (7). This prediction $\tilde{X}_{t_c}$ is then fed to the next phase to calculate the control policy.

3.3.2. Policy search phase

For designing the control policy, we utilize a GA to explore the space of possible control vectors U, aiming to find those that minimize our multi-objective function J. This function is directly tied to the system's performance as defined by the deviation from desired output trajectories and incorporates the system dynamics as characterized by the system's nonlinear equations. Therefore, the goal of GA search at time KT^π is to find the set of control parameters $U^{\pi,k}$ that minimize the following function:

$$\min j\left(U^{\pi,k}\right) = \sum_{j=0}^{H-1} \gamma^i \left|\left| Y_j^{\pi,k} - Y_j^{d,k} \right|\right|^2, \tag{8}$$

where $\gamma \in (0,1)$ is a damping factor, H is the horizon of the policy time with respect to the sampling time:

$$H = \left\lceil \frac{T^\pi}{T_s} \right\rceil, \tag{9}$$

here $\lceil . \rceil$ is the ceiling function. $Y_j^{\pi,k}$ and $Y_j^{d,k}$ are the output of the system model with respect to the policy's high-level dynamic constraints and the output reference at time

$$t = kT^\pi + jT_s. \tag{10}$$

Finally, we denote as $U^{\pi,k}$ the set of policy control values over the time interval from kT^π to $(k+1)T^\pi$:

$$U^{\pi,k} = \left(U_0^{\pi,k}, U_1^{\pi,k}, \cdots, U_{H-1}^{\pi,k}\right). \tag{11}$$

In this phase, initially we have

$$X_0^{\pi,k} = \widetilde{X}_{(k+1)T^\pi}, \tag{12}$$

is the result from the online prediction phase. is the result from the online prediction phase. Then, $\left(Y_j^{\pi,k}\right)_{0\leq j\leq H-1}$ and $\left(X_j^{\pi,k}\right)_{0\leq j\leq H-1}$ are calculated based on the system dynamics described in Equation (1), with control values $\left(U_j^{\pi,k}\right)_{0\leq j\leq H-1}$ subject to control boundaries limitations only:

$$U \leq U_j^{\pi,k} \leq \bar{U}, \quad 0 \leq j \leq H-1 \tag{13}$$

3.3.3. Pareto front-based path selection

Multi-goal control problems can be discussed in several ways, varying from scalarization to solving MOO problems. In our implementation, we choose to develop a MOO framework to build the Pareto front (PF), then select an optimization path from the resulted PF based on our current outputs deviation vector from the desired output vector.

MOO is an approach to problem-solving that involves the simultaneous optimization of multiple, often competing, objective functions. In mathematical terms, MOO seeks an optimal vector

$$\mathbf{X}^* = \{x_1^*, x_2^*, \cdots, x_n^*\}, \tag{14}$$

that maximizes or minimizes a set of objective functions

$$G(\mathbf{X}^*) = \{G_1(\mathbf{X}^*), G_2(\mathbf{X}^*), \cdots, G_k(\mathbf{X}^*)\}, \tag{15}$$

where the vector $\mathbf{X}^*$ consists of decision variables within the feasible decision space bounded by

$$\mathbf{x}_{\min} \leq \mathbf{X}^{*} \leq \mathbf{x}_{\max}. \tag{16}$$

The solution is subject to satisfying a series of constraints, which may include equalities

$$h_i(\mathbf{X}^{*}) = 0\ , \quad i = 1, 2, \ldots, p, \tag{17}$$

and inequalities

$$g_i(\mathbf{X}^{*}) \geq 0, \quad i = 1, 2, \ldots, m, \tag{18}$$

that define the permissible region of solutions, denoted as $\mathfrak{D}$. Each objective function G_i in the vector $\mathbf{G}(\mathbf{X}^{*})$ represents a distinct criterion to be optimized, and these criteria are typically non-commensurable, highlighting the complexity of the MOO process [28].

The aim is to identify the optimal set of decision variables $\mathbf{X}^{*}$ that yield the best possible outcomes across all objectives, considering the constraints that limit the feasible solutions. The decision-making is guided by trade-offs between the different objectives, as improving one function may lead to the detriment of another. This necessitates the use of advanced optimization techniques to navigate the complex solution landscape of MOO problems. Building on the previously outlined framework of MOO, the concept of Pareto optimality is crucial. This optimality is mathematically represented by a set of solutions $\mathbf{X}^{*}$ for which the corresponding vector of objective functions $\mathbf{G}(\mathbf{X}^{*})$ cannot be improved in any single objective without causing a trade-off in at least one other.

This state of Pareto optimality can be denoted by the following condition:

$$\forall \mathbf{X}^{*} \in \mathcal{X}, \nexists\ \mathbf{X}' \in \mathfrak{D}\ : \mathbf{G}(\mathbf{X}') \prec \mathbf{G}(\mathbf{X}^{*}) \tag{19}$$

where $\mathbf{X}^{*}$ is an optimal Pareto solution, $\mathbf{X}'$ is any other feasible solution, and the notation $\prec$ indicates that $\mathbf{X}'$ is strictly better than $\mathbf{X}^{*}$ across all objectives. The set $\mathfrak{D}$ encompasses all feasible solutions given the constraints, and $\mathcal{X}$ denotes the set of Pareto solutions. Therefore, the set $\mathcal{X}$ can be defined as the following:

$$\mathcal{X} = \{\mathbf{X}^{*} \in \mathfrak{D}\ : \nexists\ \mathbf{X}' \in \mathfrak{D}\ : \mathbf{G}(\mathbf{X}') \prec \mathbf{G}(\mathbf{X}^{*})\}. \tag{20}$$

The notion of Pareto optimality is essential as it establishes a framework within which decision-makers can evaluate and choose solutions that best meet their specific trade-offs and preferences among multiple objectives. This selection process is often guided by additional decision-making criteria or strategies, as the PF typically presents a multitude of equally optimal solutions in the absence of further preferences or constraints.

Based on the previous discussion, a strategy for selecting a specific path to guide the system towards PF is essential. In other words, we need to choose which of the PF points to drive the system towards according to the current state, or need of the system. This is equivalent to solving the following optimization problem:

$$\tau(\mathcal{X}) = \{X_\tau \in \mathcal{X}\ : \zeta(\mathbf{G}(X_\tau)) \leq \zeta(\mathbf{G}(X)), \forall X \in \mathcal{X}\}, \tag{21}$$

where ζ is a function representing the priority of minimizing the objective functions based on their current value.

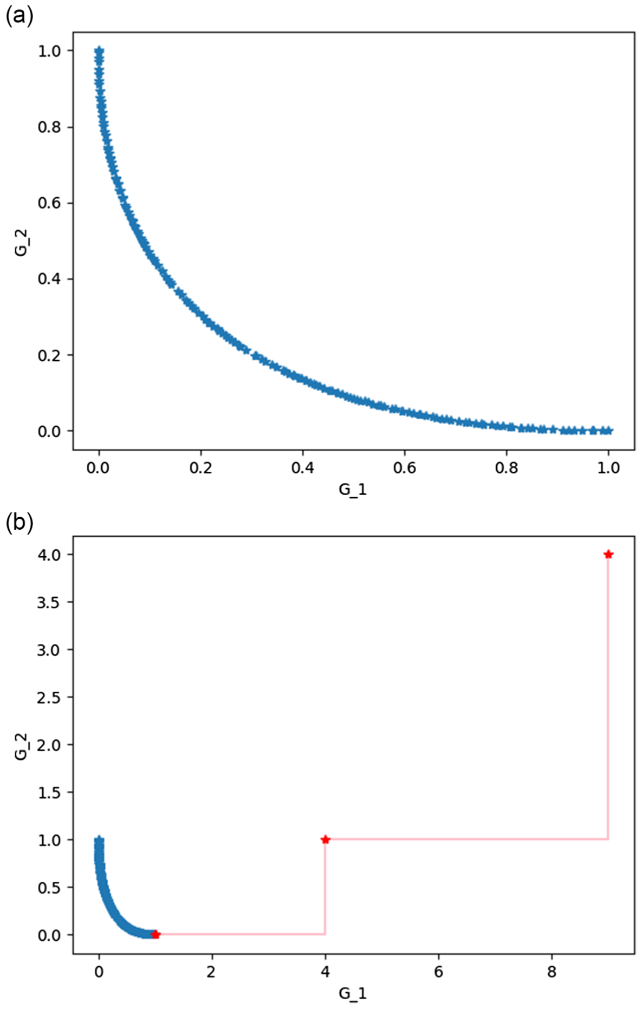
To give more insights about how PF and MOO can guide the control strategy we consider the following illustrative example:

$$\begin{aligned} x_{n+1} &= x_n + T_s u_j \\ g_n^1 &= (x_n - 1)^2 \\ g_n^2 &= (x_n)^2, \end{aligned} \tag{22}$$

the goal to minimize both g^1 and g^2, with $T_s = 1$ and $^{\pi} = 4$. We consider the initial state $x_0 = -3$, and we draw the PF after one policy time interval. Figure 2(a) shows the PF of updating the system in the illustrative example with four time steps. We can see that the PF is the minimum points of G_1 and G_1, and a set of points that compromises between minimizing these two conflicting objective functions.

The output of this phase, a set of optimized control policy vectors we refer to it as $U^{\pi,k}$, is then fed into the low-level control phase for further refinement and application.

Figure 2
Pareto front and guiding the system state of the illustrative example. (a) Pareto front of the illustrative example. (b) Guiding the system state to PF



3.3.4. Low-level control phase

This phase is the main control thread that utilizes the last update on the policy to design control vectors with respect to all constraints. The low-level control phase is set to operate in a high sampling rate without being interrupted by any calculations needed by the other two phases.

Assume receiving a high-level control vector $U^{\pi,k}$ from the policy search phase at time kT^{π}, then the goal is to design the control values denoted as:

$$U^{c,k} = \left(U_0^{c,k}, U_1^{c,k}, \cdots, U_{H-1}^{c,k}\right) \tag{23}$$

to operate within the time interval from kT^{π} to $(k+1)T^{\pi}$ following the policy control values with respect to the dynamic and regional input constraints. These values are said to be calculated iteratively as the following:

$$U_{j+1}^{c,k} = \Psi\left(U_j^{c,k}, U_{j+1}^{\pi,k}, \overline{U}_{j+1}^{c,k}, \underline{U}_{j+1}^{c,k}, \overline{U}, \underline{U}\right), \tag{24}$$

where Ψ is the low-level control function and $\underline{U}_{j+1}^{c,k}$ and $\bar{U}_{j+1}^{c,k}$ are the lower and upper boundaries of the control values at time $kT^{\pi} + (j+1)T_s$, respectively, and are calculated as the following:

$$\begin{aligned} \overline{U}_{j+1}^{c,k} &= U_j^{c,k} + \mu^T \cdot U_j^{c,k}, \\ \underline{U}_{j+1}^{c,k} &= U_j^{c,k} - \mu^T \cdot U_j^{c,k}, \end{aligned} \tag{25}$$

to respect the constraints defined in Equation (4).

3.4. Implementation and tuning guide

To apply the proposed control strategy to different systems, follow these key steps:

1) Define system dynamics

   Identify the system's *state-space equations* (e.g., nonlinear differential equations).
   Discretize the equations if working in a digital control environment.

2) Set control constraints

   Determine *boundary constraints* ($\underline{U} \le U \le \bar{U}$) based on actuator limits.
   Define *rate-of-change constraints* ($-\mu \le \dot{U} \le \mu$) to ensure smooth transitions and prevent abrupt control actions.

3) Configure the hierarchical framework

   Choose an appropriate *sampling time* (T_s) that balances control responsiveness and computational feasibility.
   Set the *policy interval* (T^{π}) based on system dynamics and GA latency.

4) Tune GA parameters

   Population size: Larger values (e.g., 80–100) improve solution quality but increase computation time.
   Number of generations: More generations (e.g., 30–50) enhance optimization but require more iterations.

5) Run parallel execution

   Ensure that *policy search (GA) runs asynchronously* while the low-level control executes at high speed.
   Validate that the *Under-Policy Prediction Phase compensates for GA search latency*.

6) Test and adjust

   Simulate the system and observe control performance.
   If GA search is too slow, reduce population size or shorten the horizon.
   If control actions are unstable, adjust rate-of-change constraints (μ).

By following these steps, users can adapt the proposed method to different systems while balancing accuracy and computational efficiency.

4. Test Case: Drum Boiler-Turbine Unit

Building upon the state of the art, this study explores the operational dynamics of a drum-type boiler-turbine system, central to thermal power units. Unlike traditional approaches that often resort to linearization around fixed points, our approach embraces the system's inherent nonlinearities. We focus on a comprehensive model integrating combustion, steam-water, control, electrical, and condensing systems, leveraging EML for dynamic adaptability.

We investigate the established model of a 160 MW boiler-turbine generator, previously analyzed in literature [29]. This model encapsulates the quintessential elements of the system, focusing on the dynamics of steam pressure, electrical power output, and the thermal-fluid processes.

The governing state equations of the system are outlined as follows:

$$\begin{aligned} \dot{x}_1 &= -0.0018u_2x_1^{9/8} + 0.9u_1 - 0.15u_3, \\ \dot{x}_2 &= (0.073u_2 - 0.016)x_1^{9/8} - 0.1x_2, \\ \dot{x}_3 &= 141u_3 - (1.1u_2 - 0.19)x_1. \end{aligned} \tag{26}$$

The outputs of the system, y_1, y_2, and y_3, are defined in relation to the state variables x_1 and x_2, along with a combined function of the states and control inputs, detailed below:

$$\begin{aligned} y_1 &= x_1, \\ y_2 &= x_2, \\ y_3 &= 0.05\left(0.13073x_3 + 100a_{cs} + \frac{q_e}{9} - 67.975\right). \end{aligned} \tag{27}$$

Here, x_1, x_2, and x_3 signify the steam pressure in the drum, electrical power output, and fluid density, respectively. The control inputs u_1, u_2, and u_3 correspond to the positioning of valves regulating fuel flow, steam, and feed-water flow. The level of water in the drum is indicated by y_3. The steam quality q_e and coefficient a_{cs}, which are pivotal for computing y_3, are given by:

$$\begin{aligned} q_e = (0.854u_2 - 0.147)x_1 + 45.59u_1 \\ -2.514u_3 - 2.096, \end{aligned} \tag{28}$$

$$a_{cs} = \frac{(1 - 0.001538x_3)(0.8x_1 - 25.6)}{x_3(1.0394 - 0.0012304x_1)}. \tag{29}$$

In the practical implementation of this model, the capabilities of the actuators are considered, leading to the establishment of constraints on the magnitude and rate of change of the control inputs. The magnitudes are confined within the range:

$$0 \le u_q \le 1, \qquad q = 1, 2, 3, \tag{30}$$

and the rate-of-change constraints are defined as:

$$\begin{array}{c} -0.007 \le \dot{u}_1 \le 0.007, \\ -2 \le \dot{u}_2 \le 2, \\ -0.05 \le \dot{u}_3 \le 0.05. \end{array} \tag{31}$$

For our control strategy, we choose to define the online under-policy prediction phase function Φ as a simple linear interpolation, therefore:

$$\tilde{X}(t_c) = \Phi\left(X_{t_a}, X_{t_b}\right) = \tfrac{T^{\pi}}{T^{p}}\left(X(t_b) - X(t_a)\right) + X(t_a) \tag{32}$$

For the online controller, we choose Ψ to guide the values of control towards increasing or decreasing within the limited speed based on the policy as the following:

$$\begin{aligned} U^{c,k}_{j+1} &= \Psi\left(U^{c,k}_{j}, U^{\pi,k}_{j+1}, \overline{U}^{c,k}_{j+1}, \underline{U}^{c,k}_{j+1}, \overline{U}, \underline{U}\right) \\ &= \mathfrak{C}\left(\mathfrak{C}\left(U^{\pi,k}_{j+1}, \underline{U}^{c,k}_{j+1}, \overline{U}^{c,k}_{j+1}\right), \underline{U}, \overline{U}\right) \end{aligned} \tag{33}$$

with $\mathfrak{C}$ being a trimming function, defined as the following:

$$\mathcal{C}(X, A, B) = \min(B, \max(A, X)), \tag{34}$$

for $X,\ A,\ B$ from $\mathbb{R}^n$. This will limit the rate of changes to the allowable interval for the provided control values.

We apply our control strategy on the boiler-turbine system on a simulation model for a simulation time of 2100 s. As in Wang et al. [29], the initial state of the boiler is:

$$\begin{aligned} X_{eq} &= [115, 85, 402.759]^T \\ U_{eq} &= [0.4147, 0.7787, 0.5436]^T \\ Y_{eq} &= [115, 85, 0]^T \end{aligned} \tag{35}$$

The goal in Wang et al. [29] was to balance the system around $Y_{ref} = Y_{eq}$. Our goal, however, is to assure balance then try balancing the system around other points. We specifically choose:

$$Y^d(t) = \begin{cases} [115, 85, 0], & 0 \le t \le 1000s, \\ [110, 85, 0], & 1000 \le t \le 2000s, \\ [110, 60, 0], & 2000 \le t \le 3000s. \end{cases} \tag{36}$$

By formulating three objective functions as the Euclidean distance between the desired value and the current value of each output, we use NSGA-II [30], that uses non-dominated sorting to get the PF. After which, we design our PF-based path selection function as the following:

$$\tau(\mathcal{X}) = \mathfrak{X}_{\xi}\left(\mathfrak{X}_{\xi}\left(\mathfrak{X}_{\xi}(\mathcal{X}, G_2), G_3\right), G_1\right), \tag{37}$$

where $\mathfrak{X}_{\xi}(A, f)$, with A a finite set and f is an evaluation function, is the set of values $B \subset A$ for which:

$$\forall x \in B,\ |f(x) - f(x^*)| \le \xi. \tag{38}$$

with x^* defined as:

$$x^* := (x^* \in A) \wedge \left(f(x^*) \le f(x),\ \forall x \in A\right). \tag{39}$$

The subset $\mathfrak{X}_{\xi}(A, f)$, with $Card(A) > 0$, can be proven not to be empty since it has at least one item x^*. Conversely, if the final set $\tau(\mathcal{X})$ has more than one item, we choose to select any.

In our strategy's hyperparameters, we set the population size to 80, the number of generations to 40, and the number of offsprings per generation also to 40. We define the GA search model's future horizon to be 10. The sampling time (T_s) is 0.1 s, and the policy interval (T^{π}) is set to 1 s. We also choose $\xi = 1$ in Equation (38).

These values were selected based on a balance between computational feasibility and control accuracy. In systems with limited computing resources, these parameters can be adjusted by reducing population size or demand at the cost of slightly slower convergence. Similarly, tuning the prediction horizon can help optimize performance for real-time constraints.

Building on the setup described, the experimental simulation took place on a Core-i5 12th generation processor equipped with 32GB of RAM and utilized Ubuntu 22.04 as its operating system. The development of the simulation model and algorithms was carried out using the Python programming language. On this hardware, the machine demonstrated the capability to perform the GAs policy search in approximately 0.3 seconds. Consequently, we executed two separate experiments; in the first, the T_g interval was maintained at 0.3 s, as illustrated in Figure 3, and in the second experiment, the T_g interval was extended to 0.7 s, as depicted in Figure 4. Moreover, Figure 5 shows how the control values (in blue) are guided by the policy values (in orange) with respect to their allowed rate of change as in Equation (31).

Results demonstrate the ability of the proposed control strategy to guide the boiler-turbine system towards the desired output in both cases. We can notice the small required change of water level u3 along the control interval. This highlights the possibility to use our strategy even with slower computational hardware to perform GA search.

Our hierarchical framework is specifically designed to ensure real-time feasibility. The under-policy prediction phase compensates for GA latency, ensuring continuous control without delays. Additionally, GA-based optimization is executed in parallel with low-level control, preventing bottlenecks in execution. While alternative methods like heuristic search or hybrid GA-local optimization could reduce computational demands, our results demonstrate that the current approach already achieves real-time performance within practical constraints.

4.1. Comparison with recent work

Our proposed methodology offers a more general approach to controlling the boiler-turbine unit without the need for linearization or operating point approximations, which is a key contribution of our research. To demonstrate the effectiveness of our approach, we compare it with existing methods for boiler control.

For this comparison, we selected the method reported in Reference [24], focusing on load-changing conditions tested by varying the load from 120 MW to 140 MW. The operating points under a load of 120 MW are given by:

$$\begin{aligned} x_{120} &= [137,\ 120, 299.09]^T, \\ u_{120} &= [0.563, 0.868, 0.743]^T. \end{aligned} \tag{40}$$

For a load of 140 MW, the operating points are:

$$\begin{aligned} x_{140} &= [148.94, 140.7, 356.3]^T \\ u_{140} &= [0.648, 0.9081, 0.8545]^T. \end{aligned} \tag{41}$$

Table 1 presents the convergence times (in seconds) for different control variables using the IRL method, MPC method, and our

Figure 3
Simulation results for $T_s = 0.1$, $T_p = 0.7$, $T_g = 0.3$, the output signals of (y_1, y_2, y_3) with respect to time in seconds

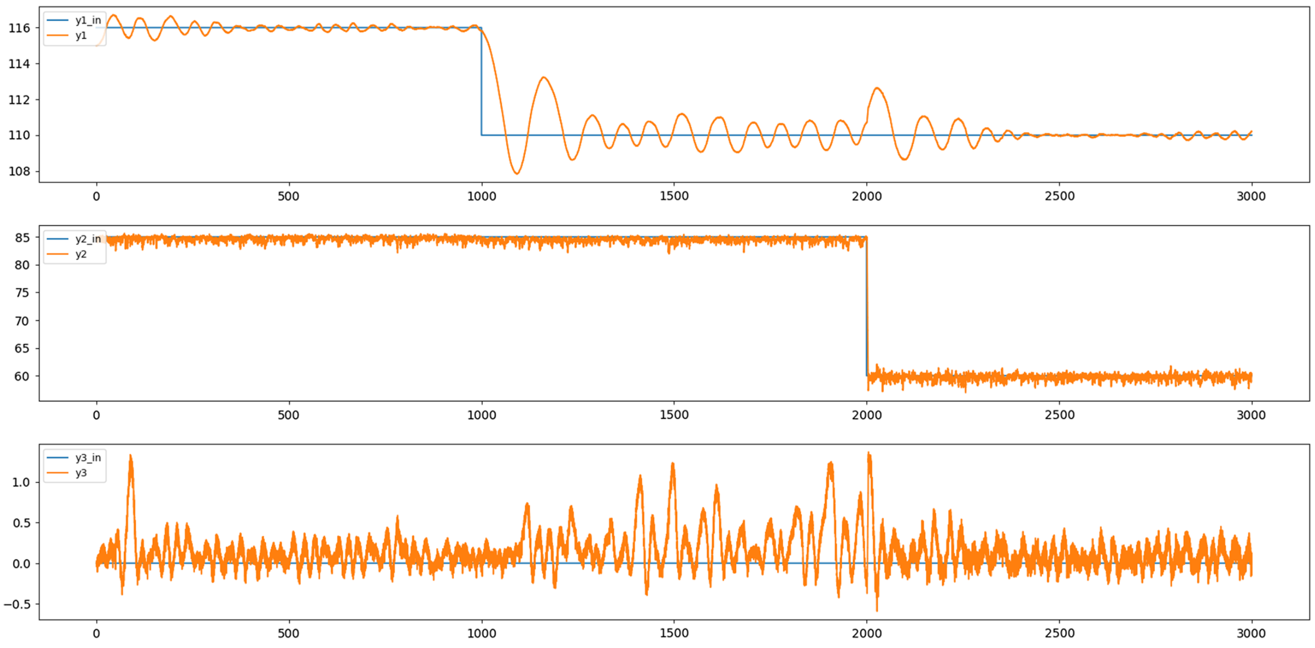
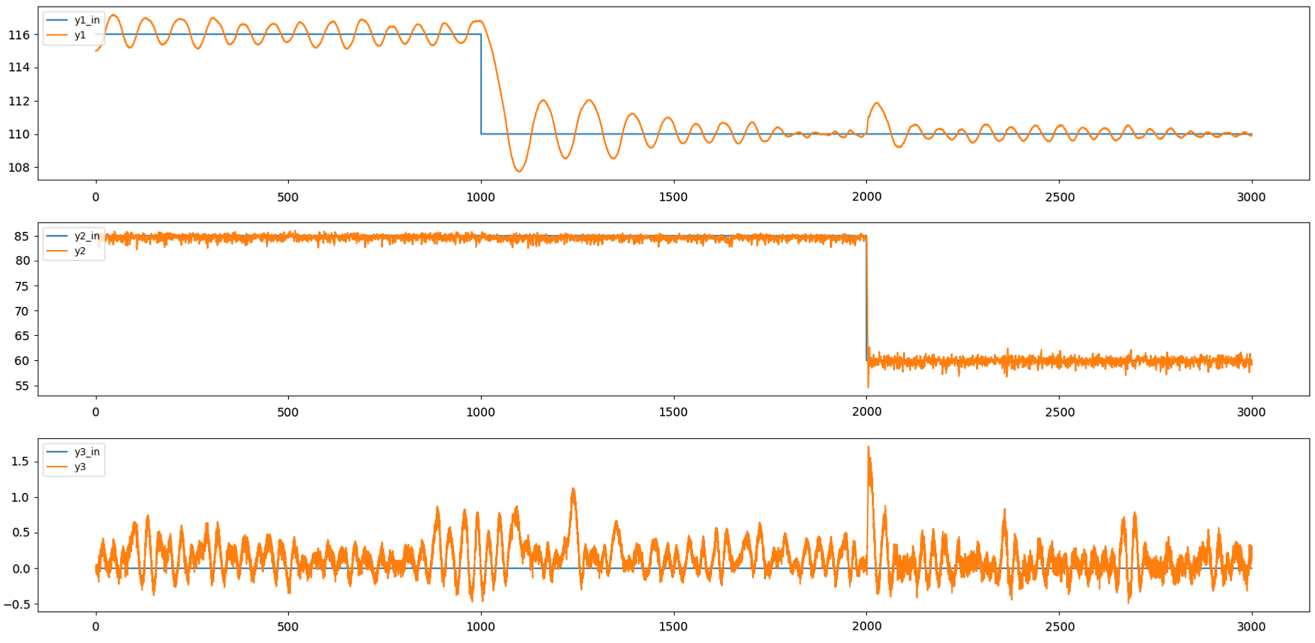


Figure 4
Simulation results for $T_s = 0.1$, $T_p = 0.3$, $T_g = 0.7$, the output signals of (y_1, y_2, y_3) with respect to time in seconds



proposed strategy. The results from Wei et al. [24] demonstrate that our approach achieves significantly faster convergence across all variables.

Figure 6 illustrates the performance comparison of the different control strategies under varying load conditions. The figure complements the tabulated convergence times, providing a visual representation of the superior efficiency and effectiveness of our proposed strategy in reducing convergence time and enhancing control performance of the boiler-turbine unit.

These results highlight the efficiency and effectiveness of our proposed strategy in reducing convergence time, thereby enhancing the control performance of the boiler-turbine unit under varying load conditions.

5. Limitations and Future Work

The current approach mitigates the computational demands of GA search through a hierarchical structure, parallel execution, and

Figure 5
Policy and control signals for inputs (u_1, u_2, u_3), with respect to time in seconds

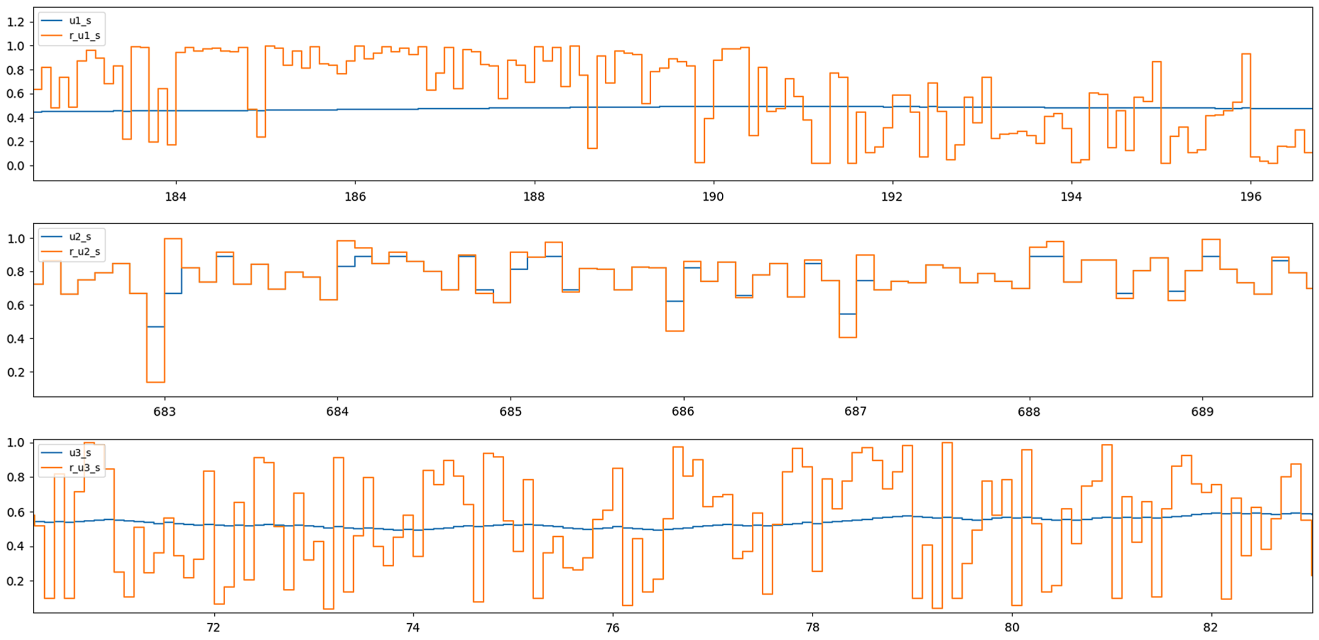
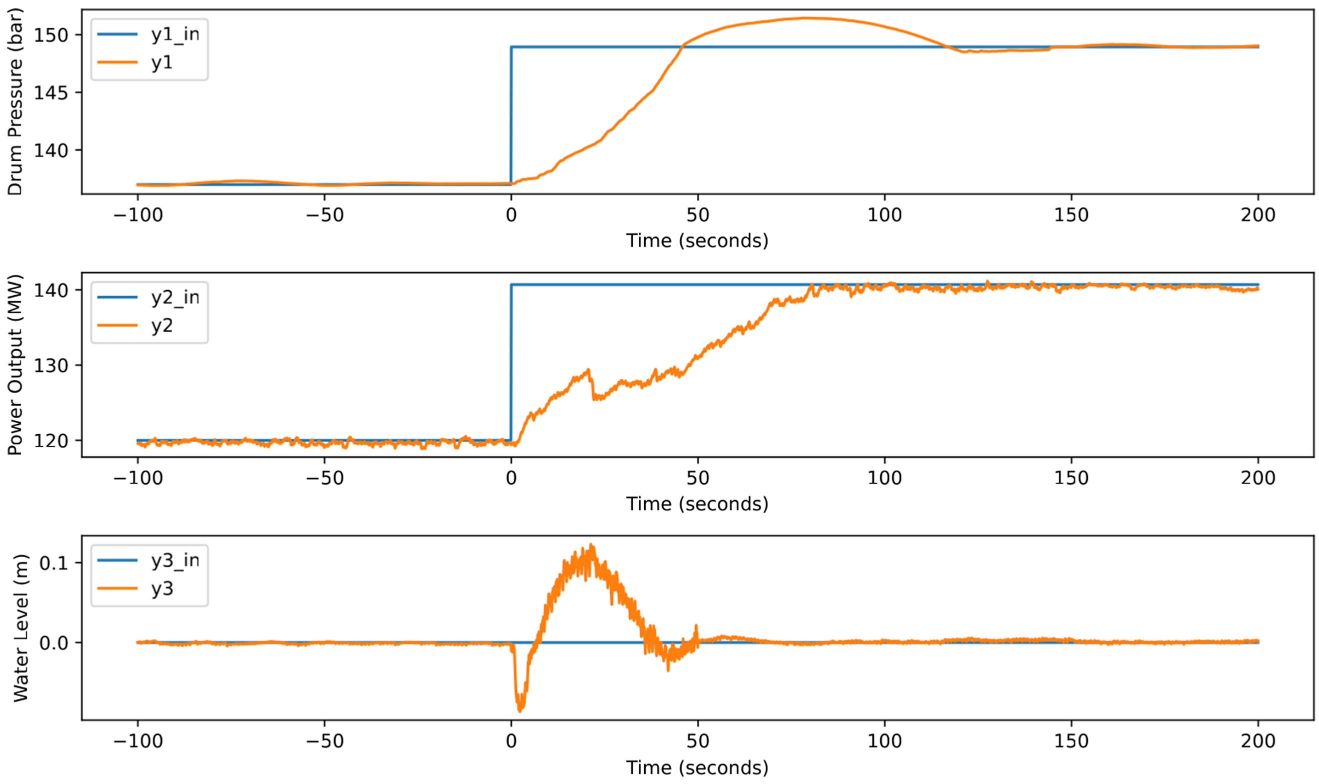


Table 1
Convergence time comparison (seconds)

Control variable	IRL method	MPC method	Proposed strategy
Drum Pressure	335	380	150
Water Level	287	195	55
Power Output	320	405	85

Figure 6
Results of the proposed control strategy on the load-varying test



predictive modeling. The Online Under-policy Prediction Phase reduces unnecessary search space exploration, while the parallelized GA execution ensures that policy search does not interfere with the real-time control loop. Additionally, low-level control smoothing prevents excessive fluctuations in control inputs, reducing the computational load.

However, real-time applications with stricter timing constraints may still pose challenges. Future enhancements could focus on:

1) Adaptive horizon adjustment: Dynamically adjusting the GA prediction horizon based on system conditions to optimize speed vs. accuracy trade-offs.
2) Efficient population management: Reducing unnecessary GA iterations by reusing prior solutions (e.g., warm-starting the GA with previous optimal policies).
3) Embedded system deployment: Evaluating the feasibility of this approach on resource-limited platforms such as microcontrollers or FPGAs, testing how well the framework scales under real-world conditions.

6. Conclusion

Our study introduces a multi-level hierarchical control strategy that effectively tackles complex, real-time MOO challenges in nonlinear systems through the use of EAs. This strategy integrates a high-level optimization problem solution with a dynamic lower-level control loop, avoiding linearization or operating point approximations. Our innovative approach, demonstrated through simulations on a drum boiler-turbine unit, highlights the strategy's capability to deliver optimized, real-time control signals for various operational goals. The promising results not only showcase the potential of EAs in refining control strategies for intricate systems but also open avenues for real-world applications and future enhancements. This work paves the way for advanced, adaptive control mechanisms in complex system management, promising significant advancements in control engineering.

Conflicts of Interest

The authors declare that they have no conflicts of interest to this work.

Data Availability Statement

Data sharing is not applicable to this article as no new data were created or analyzed in this study.

Author Contribution Statement

Ali Deeb: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Data curation, Writing – original draft, Writing – review & editing, Visualization. **Vladimir Khokhlovskiy:** Conceptualization, Methodology, Validation, Formal analysis, Investigation, Resources, Writing – original draft, Writing – review & editing, Visualization, Supervision. **Viacheslav Shkodyrev:** Conceptualization, Methodology, Validation, Formal analysis, Investigation, Resources, Writing – original draft, Writing – review & editing, Visualization, Supervision, Project administration.

References

[1] International Energy Agency. (2022). *World energy outlook 2022*. International Energy Agency. https://www.iea.org/reports/world-energy-outlook-2022

[2] Lee, Y., Yoo, E., Lee, T., & Moon, U.-C. (2018). Supplementary control of conventional coordinated control for 1000 MW ultra-supercritical thermal power plant using dynamic matrix control. *Journal of Electrical Engineering and Technology*, *13*(1), 97–104. https://doi.org/10.5370/JEET.2018.13.1.097

[3] Wu, X., Wang, M., Shen, J., Li, Y., Lawal, A., & Lee, K. Y. (2019). Reinforced coordinated control of coal-fired power plant retrofitted with solvent based CO_2 capture using model predictive controls. *Applied Energy*, *238*, 495–515. https://doi.org/10.1016/j.apenergy.2019.01.082

[4] Wu, Z., Gao, Z., Li, D., Chen, Y., & Liu, Y. (2021). On transitioning from PID to ADRC in thermal power plants. *Control Theory and Technology*, *19*(1), 3–18. https://doi.org/10.1007/s11768-021-00032-4

[5] Ogata, K. (2010). *Modern control engineering* (5th ed.). USA: Prentice Hall.

[6] Schwenzer, M., Ay, M., Bergs, T., & Abel, D. (2021). Review on model predictive control: An engineering perspective. *The International Journal of Advanced Manufacturing Technology*, *117*(5), 1327–1349. https://doi.org/10.1007/s00170-021-07682-3

[7] Duriez, T., Brunton, S. L., & Noack, B. R. (2017). *Machine learning control – Taming nonlinear dynamics and turbulence*. Switzerland: Springer. https://doi.org/10.1007/978-3-319-40624-4

[8] Lawrence, N. P., Damarla, S. K., Kim, J. W., Tulsyan, A., Amjad, F., Wang, K., . . . , & Gopaluni, R. B. (2024). Machine learning for industrial sensing and control: A survey and practical perspective. *Control Engineering Practice*, *145*, 105841. https://doi.org/10.1016/j.conengprac.2024.105841

[9] Ahmed, C. M., Ramani, G. R. M. I., & Mathur, A. P. (2020). Challenges in machine learning based approaches for real-time anomaly detection in industrial control systems. In *Proceedings of the 6th ACM on Cyber-Physical System Security Workshop*, 23–29. https://doi.org/10.1145/3384941.3409588

[10] Wang, C., Liu, M., Zhao, Y., Qiao, Y., Chong, D., & Yan, J. (2018). Dynamic modeling and operation optimization for the cold end system of thermal power plants during transient processes. *Energy*, *145*, 734–746. https://doi.org/10.1016/j.energy.2017.12.146

[11] Wu, X., Shen, J., Li, Y., & Lee, K. Y. (2011). Stable model predictive control based on TS fuzzy model with application to boiler-turbine coordinated system. In *2011 50th IEEE Conference on Decision and Control and European Control Conference*, 2981–2987. https://doi.org/10.1109/CDC.2011.6160553

[12] Wu, X., Shen, J., Li, Y., & Lee, K. Y. (2014). Hierarchical optimization of boiler–turbine unit using fuzzy stable model predictive control. *Control Engineering Practice*, *30*, 112–123. https://doi.org/10.1016/j.conengprac.2014.03.004

[13] Wang, L., Cai, Y., & Ding, B. (2021). Robust model predictive control with bi-level optimization for boiler-turbine system. *IEEE Access*, *9*, 48244–48253. https://doi.org/10.1109/ACCESS.2021.3066371

[14] Zhao, S., Wang, S., Cajo, R., Ren, W., & Li, B. (2022). Power tracking control of marine boiler-turbine system based on fractional order model predictive control algorithm. *Journal of Marine Science and Engineering*, *10*(9), 1307. https://doi.org/10.3390/jmse10091307

[15] Köhler, J., Soloperto, R., Müller, M. A., & Allgöwer, F. (2021). A computationally efficient robust model predictive control framework for uncertain nonlinear systems. *IEEE Transactions on Automatic Control*, *66*(2), 794–801. https://doi.org/10.1109/TAC.2020.2982585

[16] Zhao, M., Wan, J., & Peng, C. (2023). Generalized predictive control using improved recurrent fuzzy neural network for a boiler-turbine unit. *Engineering Applications of Artificial Intelligence*, *121*, 106053. https://doi.org/10.1016/j.engappai.2023.106053

[17] Zhao, G., Sun, Y., Su, Z.-G., & Hao, Y. (2023). Receding Galerkin optimal control with high-order sliding mode disturbance observer for a boiler-turbine unit. *Sustainability*, *15*(13), 10129. https://doi.org/10.3390/su151310129

[18] Sanchez, R. O., Rumbo Morales, J. Y., Ortiz Torres, G., Pérez Vidal, A. F., Valdez Resendiz, J. E., Sorcia Vázquez, F. J., & Nava, N. V. (2022). Discrete state-feedback control design with D-stability and genetic algorithm for LED driver using a buck converter. *International Transactions on Electrical Energy Systems*, *2022*(1), 8165149. https://doi.org/10.1155/2022/8165149

[19] Yang, C., Zhang, T., Zhang, Z., & Sun, L. (2022). MLD–MPC for ultra-supercritical circulating fluidized bed boiler unit using subspace identification. *Energies*, *15*(15), 5476. https://doi.org/10.3390/en15155476

[20] Ławryńczuk, M. (2017). Nonlinear predictive control of a boiler-turbine unit: A state-space approach with successive on-line model linearisation and quadratic optimisation. *ISA Transactions*, *67*, 476–495. https://doi.org/10.1016/j.isatra.2017.01.016

[21] Cornejo Maceda, G. Y., & Noack, B. R. (2024). Evolutionary machine learning in control. In W. Banzhaf, P. Machado, & M. Zhang (Eds.), *Handbook of evolutionary machine learning* (pp. 629–656). Springer. https://doi.org/10.1007/978-981-99-3814-8_22

[22] Song, D., Chang, Q., Zheng, S., Yang, S., Yang, J., & Joo, Y. H. (2021). Adaptive model predictive control for yaw system of variable-speed wind turbines. *Journal of Modern Power Systems and Clean Energy*, *9*(1), 219–224. https://doi.org/10.35833/MPCE.2019.000467

[23] Cui, J., Chai, T., & Liu, X. (2020). Deep-neural-network-based economic model predictive control for ultrasupercritical power plant. *IEEE Transactions on Industrial Informatics*, *16*(9), 5905–5913. https://doi.org/10.1109/TII.2020.2973721

[24] Wei, Q., Liu, Y., Lu, J., Ling, J., Luan, Z., & Chen, M. (2023). A new integral critic learning for optimal tracking control with applications to boiler-turbine systems. *Optimal Control Applications and Methods*, *44*(2), 830–845. https://doi.org/10.1002/oca.2792

[25] Slowik, A., & Kwasnicka, H. (2020). Evolutionary algorithms and their applications to engineering problems. *Neural Computing and Applications*, *32*(16), 12363–12379. https://doi.org/10.1007/s00521-020-04832-8

[26] Deeb, A., Khokhlovskiy, V. N., & Shkodyrev, V. P. (2024). Model predictive control and genetic algorithms for optimization of continuous stirred tank reactors. In I. L. Tarasova & B. A. Kulik (Eds.), *Smart electromechanical systems: Mathematical and software engineering* (pp. 185–191). Springer. https://doi.org/10.1007/978-3-031-64277-7_14

[27] Biegler, L. T. (2021). A perspective on nonlinear model predictive control. *Korean Journal of Chemical Engineering*, *38*(7), 1317–1332. https://doi.org/10.1007/s11814-021-0791-7

[28] Jones, D. F., & Florentino, H. O. (2022). Multi-objective optimization: Methods and applications. In S. Salhi & J. Boylan (Eds.), *The Palgrave handbook of operations research* (pp. 181–207). Springer. https://doi.org/10.1007/978-3-030-96935-6_6

[29] Wang, J., Ding, B., & Wang, P. (2022). Modeling and finite-horizon MPC for a boiler-turbine system using minimal realization state-space model. *Energies*, *15*(21), 7935. https://doi.org/10.3390/en15217935

[30] Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, *6*(2), 182–197. https://doi.org/10.1109/4235.996017

How to Cite: Deeb, A., Khokhlovskiy, V., & Shkodyrev, V. (2026). Hierarchical Multi-objective Control of Nonlinear Systems with Dynamical Input Constraints. *Artificial Intelligence and Applications*, *4*(1), 57–67. https://doi.org/10.47852/bonviewAIA52024314